SunFounder ESP32 Starter Kit

www.sunfounder.com

26 de marzo de 2024

Índice general

1. Descargar el Código **3**

2. Para Usuarios de Arduino **5**

2.1. 1.1 Instalar Arduino IDE (Importante) . . . 5
2.2. 1.2 Introducción a Arduino IDE . . . 11
2.3. 1.3 Instalar la Placa ESP32 (Importante) . . . 12
2.4. 1.4 Instalación de librerías (Importante) . . . 17
2.5. 2.1 ¡Hola, LED! . . . 20
2.6. 2.2 Desvanecimiento . . . 24
2.7. 2.3 Luz Colorida . . . 28
2.8. 2.4 Microchip - 74HC595 . . . 33
2.9. 2.5 Pantalla de 7 Segmentos . . . 36
2.10. 2.6 Mostrar Caracteres . . . 40
2.11. 2.7 Tira de LEDs RGB . . . 43
2.12. 3.1 Beep . . . 47
2.13. 3.2 Tono Personalizado . . . 50
2.14. 4.1 Motor . . . 54
2.15. 4.2 Bombeo . . . 57
2.16. 4.3 Balanceo del Servo . . . 60
2.17. 5.1 Lectura del Valor del Botón . . . 63
2.18. 5.2 ¡Inclínalo! . . . 68
2.19. 5.3 Detectar el Obstáculo . . . 71
2.20. 5.4 Detectar la Línea . . . 73
2.21. 5.5 Detección de Movimiento Humano . . . 76
2.22. 5.6 Dos Tipos de Transistores . . . 79
2.23. 5.7 Siente la Luz . . . 85
2.24. 5.8 Gira el Potenciómetro . . . 88
2.25. 5.9 Medir la Humedad del Suelo . . . 91
2.26. 5.10 Termómetro . . . 93
2.27. 5.11 Alternar el Joystick . . . 97
2.28. 5.12 Medición de Distancia . . . 99
2.29. 5.13 Temperatura - Humedad . . . 102
2.30. 5.14 Receptor IR . . . 106
2.31. 6.1 Piano de Frutas . . . 110
2.32. 6.2 Luz Fluyente . . . 114
2.33. 6.3 Ayuda para Reversa . . . 116

2.34. 6.4 Dados Digitales . . . 120
2.35. 6.5 Degradado de Color . . . 124
2.36. 6.6 Monitor de Plantas . . . 128
2.37. 6.7 Adivina el Número . . . 131
2.38. 7.1 Bluetooth . . . 135
2.39. 7.2 Control de LED RGB por Bluetooth . . . 144
2.40. 7.3 Reproductor de Audio Bluetooth . . . 150
2.41. 7.4 Escritura y Lectura de Tarjeta SD . . . 154
2.42. 7.5 Reproductor MP3 con Soporte de Tarjeta SD . . . 158
2.43. 7.6 Tomar foto y guardar en SD . . . 162
2.44. 8.1 Información del Tiempo en Tiempo Real de @OpenWeatherMap . . . 169
2.45. 8.2 Servidor Web de Cámara . . . 173
2.46. 8.3 Servidor Web de Transmisión de Video Personalizado . . . 177
2.47. 8.4 Comunicación IoT con MQTT . . . 182
2.48. 8.5 CheerLights . . . 189
2.49. 8.6 Monitoreo de Temperatura y Humedad con Adafruit IO . . . 193
2.50. 8.7 Cámara ESP con Bot de Telegram . . . 204
2.51. 8.8 Cámara con Home Assistant . . . 211
2.52. 8.9 Sistema de Notificación de Intrusión Basado en Blynk . . . 225
2.53. 8.10 Aplicación Android - Operación de LED RGB mediante Arduino y Bluetooth . . . 242

3. Curso de Video de Arduino **257**
3.1. SERIE ESP32 #0: Nueva Serie Esp32 2024 . . . 257
3.2. SERIE ESP32 #1: ¿Que Modelo de ESP32 Elegir? . . . 257
3.3. Esp32 #2: How to Setup Vscode for Esp32 Programming . . . 258
3.4. Serie Esp32: Como Crear un Página Web - Con Websockets . . . 258

4. Para Usuarios de MicroPython **259**
4.1. 1.1 Introducción a MicroPython . . . 259
4.2. 1.2 Instalar Thonny IDE . . . 260
4.3. 1.3 Instalar MicroPython en el ESP32(Importante) . . . 262
4.4. 1.4 Subir las Bibliotecas (Importante) . . . 266
4.5. 1.5 Guía Rápida sobre Thonny . . . 269
4.6. 1.6 (Opcional) Sintaxis Básica de MicroPython . . . 278
4.7. 2.1 ¡Hola, LED! . . . 305
4.8. 2.2 Atenuación de un LED . . . 310
4.9. 2.3 Luz Colorida . . . 313
4.10. 2.4 Microchip - 74HC595 . . . 318
4.11. 2.5 Visualización de Números . . . 323
4.12. 2.6 Mostrar Caracteres . . . 328
4.13. 2.7 Tira de LED RGB . . . 331
4.14. 3.1 Pitido . . . 334
4.15. 3.2 Tono Personalizado . . . 337
4.16. 4.1 Pequeño Ventilador . . . 342
4.17. 4.2 Bombeo . . . 347
4.18. 4.3 Servo Oscilante . . . 350
4.19. 5.1 Lectura del Valor del Botón . . . 354
4.20. 5.2 ¡Inclínalo! . . . 357
4.21. 5.3 Detección de Obstáculos . . . 361
4.22. 5.4 Detección de Líneas . . . 363
4.23. 5.5 Detección de Movimiento Humano . . . 366
4.24. 5.6 Dos Tipos de Transistores . . . 371
4.25. 5.7 Siente la Luz . . . 377
4.26. 5.8 Girar el Pomo . . . 380

4.27. 5.9 Medir la Humedad del Suelo . . . 384
4.28. 5.10 Detección de Temperatura . . . 386
4.29. 5.11 Alternar el Joystick . . . 392
4.30. 5.12 Medición de Distancia . . . 395
4.31. 5.13 Temperatura - Humedad . . . 398
4.32. 5.14 Control Remoto por Infrarrojos . . . 402
4.33. 6.1 Piano de Frutas . . . 407
4.34. 6.2 Luz Fluyente . . . 412
4.35. 6.3 Teremín de Luz . . . 415
4.36. 6.4 Asistente de Reversa . . . 419
4.37. 6.5 Degradado de Color . . . 424
4.38. 6.6 Dado Digital . . . 428
4.39. 6.7 Adivina el Número . . . 432
4.40. 6.8 Monitor de Plantas . . . 439

5. Jugar con Scratch **445**
5.1. 1.1 Instalar PictoBlox . . . 446
5.2. 1.2 Introducción a la Interfaz . . . 447
5.3. 1.3 Guía Rápida sobre PictoBlox . . . 448
5.4. 2.1 Lámpara de Mesa . . . 469
5.5. 2.2 LED Respirando . . . 476
5.6. 2.3 Bolas Coloridas . . . 484
5.7. 2.4 Ratón en Movimiento . . . 492
5.8. 2.5 Timbre . . . 499
5.9. 2.6 Alarma de Baja Temperatura . . . 506
5.10. 2.7 Reloj Despertador Luminoso . . . 514
5.11. 2.8 Leer Temperatura y Humedad . . . 521
5.12. 2.9 Ventilador Rotativo . . . 527
5.13. 2.10 Bola Sensible a la Luz . . . 535
5.14. 2.11 JUEGO - Disparos . . . 548
5.15. 2.12 JUEGO - Inflando el Globo . . . 561
5.16. 2.13 JUEGO - Estrellas Cruzadas . . . 571
5.17. 2.14 JUEGO - Comer Manzana . . . 581
5.18. 2.15 JUEGO - Loro Flappy . . . 592
5.19. 2.16 JUEGO - Clon de Breakout . . . 601
5.20. 2.17 JUEGO - Pesca . . . 612
5.21. 2.18 JUEGO - No Toques la Baldosa Blanca . . . 621
5.22. 2.19 JUEGO - Protege Tu Corazón . . . 641
5.23. 2.20 JUEGO - Matar al Dragón . . . 658

6. Aprende sobre los Componentes en tu Kit **681**
6.1. ESP32 WROOM 32E . . . 682
6.2. Extensión de Cámara ESP32 . . . 685
6.3. Protoboard . . . 691
6.4. Resistor . . . 695
6.5. Capacitor . . . 698
6.6. Cables Puente . . . 700
6.7. Transistor . . . 701
6.8. 74HC595 . . . 703
6.9. L293D . . . 708
6.10. LED . . . 709
6.11. LED RGB . . . 716
6.12. display de 7 Segmentos . . . 719
6.13. I2C LCD1602 . . . 725

6.14. Tira de 8 LEDs RGB WS2812 . . . 727
6.15. Zumbador . . . 729
6.16. Módulo de Audio y Altavoz . . . 730
6.17. Motor de Corriente Continua (DC) . . . 732
6.18. Servo . . . 734
6.19. Bomba Centrífuga . . . 736
6.20. Botón . . . 737
6.21. Interruptor de Inclinación . . . 739
6.22. Potenciómetro . . . 740
6.23. Módulo de Joystick . . . 741
6.24. Receptor de Infrarrojos . . . 743
6.25. Fotorresistor . . . 745
6.26. Termistor . . . 746
6.27. Sensor de Humedad y Temperatura DHT11 . . . 747
6.28. Módulo Sensor de Movimiento PIR . . . 749
6.29. Módulo de Seguimiento de Línea . . . 751
6.30. Módulo de Humedad del Suelo . . . 752
6.31. Módulo de Evitación de Obstáculos . . . 754
6.32. Módulo Ultrasonido . . . 755

7. FAQ **759**
7.1. Cómo usar Blynk en dispositivos móviles? . . . 759
7.2. ¿Cómo formatear la tarjeta SD? . . . 761
7.3. ¿Siempre aparece «COMxx desconocido»? . . . 764

8. Gracias **767**

9. Copyright Notice **769**

Gracias por elegir nuestro Kit de Inicio ESP32.

Nota: Este documento está disponible en los siguientes idiomas.

-
-
-
-

Por favor, haz clic en los enlaces respectivos para acceder al documento en tu idioma preferido.

¡Bienvenido al Kit de Aprendizaje ESP32! Este paquete integral está diseñado para ofrecer tanto a principiantes como a desarrolladores experimentados una inmersión profunda en el versátil mundo del microcontrolador ESP32. Con el ESP32 WROOM 32E en su núcleo, y una gama de componentes acompañantes como LEDs, sensores, motores y más, los usuarios pueden explorar una amplia variedad de proyectos.

Ya sea que estés interesado en electrónica básica, integraciones IoT, este kit lo tiene todo. Para los entusiastas de MicroPython, proporcionamos una introducción estructurada a MicroPython, completa con configuraciones de IDE y lecciones básicas de sintaxis. Los usuarios de Arduino no se quedan atrás, con una sección dedicada a comenzar con Arduino y una serie de proyectos básicos para iniciar el proceso de aprendizaje.

Para los creativos, hay una sección encantadora sobre la integración con Scratch, permitiendo una mezcla de programación y narración. Cada proyecto en el kit está meticulosamente delineado, asegurando que entiendas los objetivos, el montaje del circuito y los aspectos de programación.

Con una miríada de proyectos de juegos, aplicaciones prácticas y preguntas frecuentes de resolución de problemas, este kit promete una experiencia de aprendizaje enriquecedora para todos. ¡Sumérgete y que comience la aventura ESP32!

Si tienes alguna pregunta u otras ideas interesantes, no dudes en enviar un correo electrónico a service@sunfounder.com.

CAPÍTULO 1

Descargar el Código

Aquí tienes el paquete completo de código para el Kit de Inicio ESP32. Puedes hacer clic en el siguiente enlace para descargarlo:

- SunFounder ESP32 Starter Kit

Una vez que la descarga esté completa, descomprime el archivo y abre los archivos de ejemplo o proyectos relevantes en el software correspondiente. Esto te permitirá navegar y utilizar todo el código y los recursos proporcionados por el kit.

CAPÍTULO 2

Para Usuarios de Arduino

Aquí tienes el paquete completo de código para el Kit de Inicio ESP32. Puedes hacer clic en el siguiente enlace para descargarlo:

- `Kit de Inicio SunFounder ESP32`

Una vez completada la descarga, descomprime el archivo y abre los archivos de ejemplo o proyectos relevantes en el software correspondiente. Esto te permitirá explorar y utilizar todo el código y los recursos proporcionados por el kit.

1. Comenzar

2.1 1.1 Instalar Arduino IDE (Importante)

El Arduino IDE, conocido como Entorno de Desarrollo Integrado de Arduino, proporciona todo el soporte de software necesario para completar un proyecto de Arduino. Es un software de programación específicamente diseñado para Arduino, proporcionado por el equipo de Arduino, que nos permite escribir programas y subirlos a la placa de Arduino.

El Arduino IDE 2.0 es un proyecto de código abierto. Es un gran avance respecto a su sólido predecesor, Arduino IDE 1.x, y viene con una interfaz de usuario renovada, mejor gestión de placas y librerías, depurador, función de autocompletado y mucho más.

En este tutorial, mostraremos cómo descargar e instalar el Arduino IDE 2.0 en tu computadora Windows, Mac o Linux.

2.1.1 Requisitos

- Windows - Win 10 y más recientes, 64 bits
- Linux - 64 bits
- Mac OS X - Versión 10.14: «Mojave» o más reciente, 64 bits

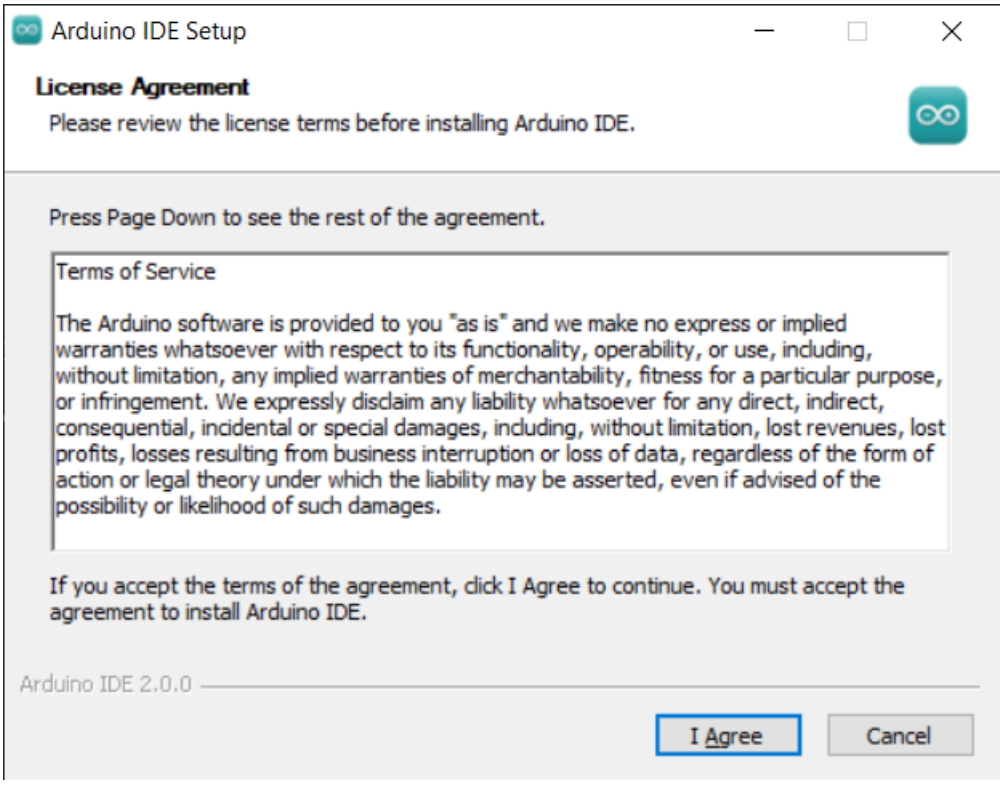
2.1.2 Descargar Arduino IDE 2.0

1. Visita .
2. Descarga el IDE para la versión de tu SO.

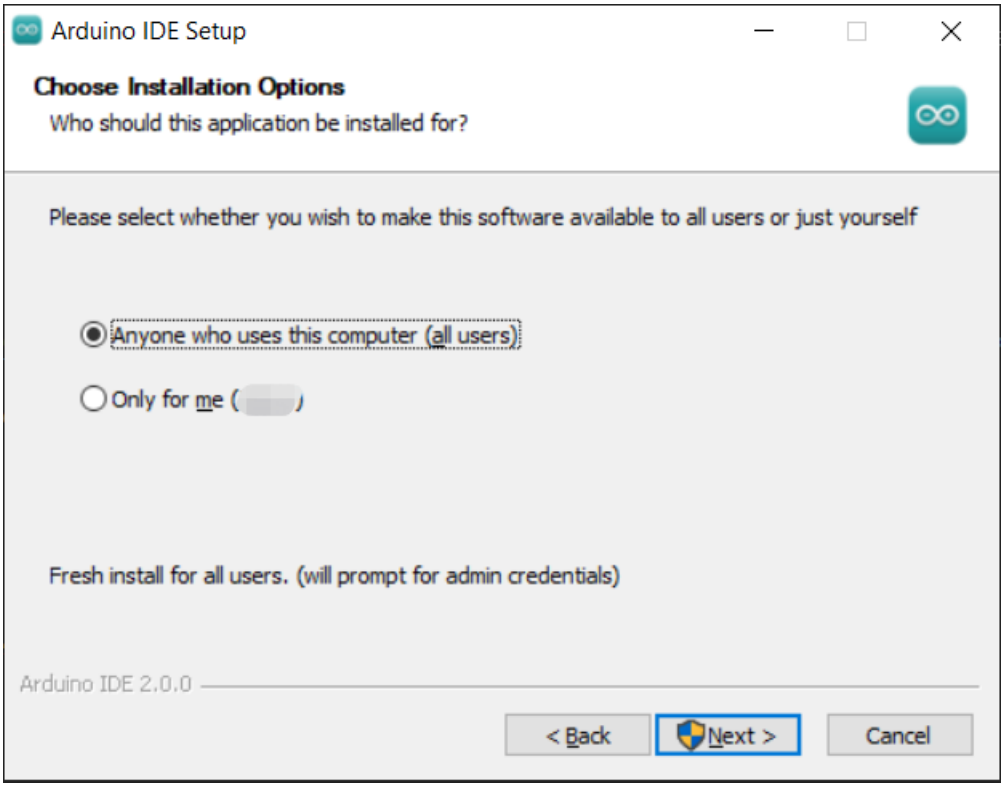
2.1.3 Instalación

Windows

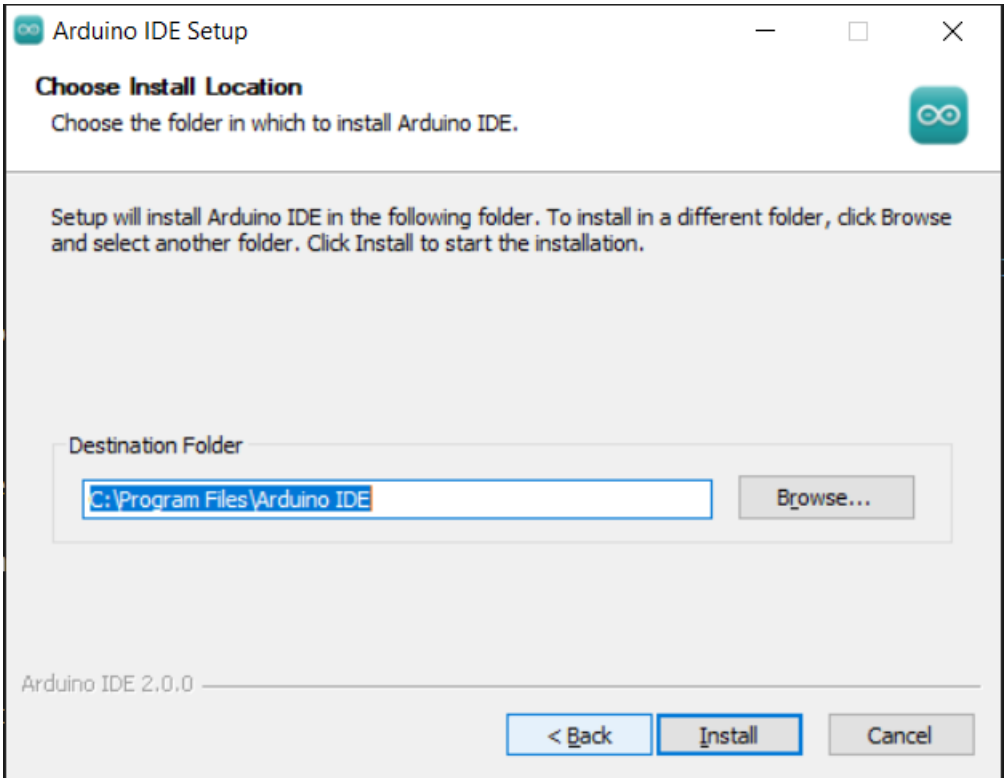
1. Haz doble clic en el archivo `arduino-ide_xxxx.exe` para ejecutar el archivo descargado.
2. Lee el Acuerdo de Licencia y acéptalo.



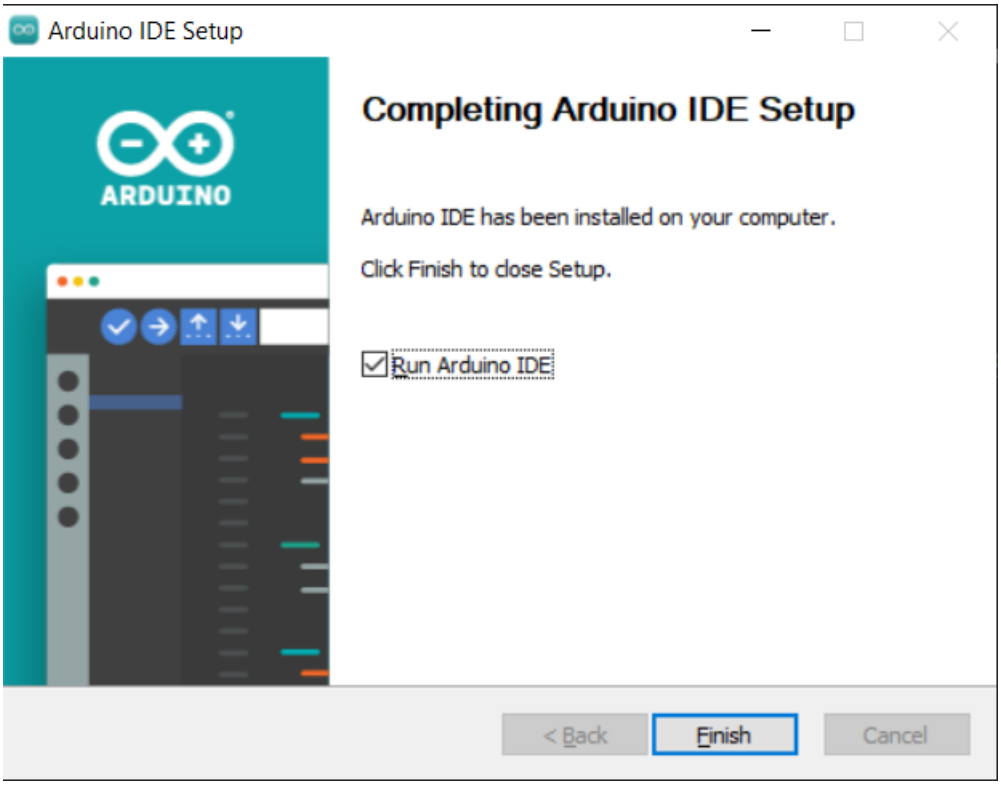
3. Elige las opciones de instalación.



4. Elige la ubicación de instalación. Se recomienda instalar el software en una unidad distinta a la unidad del sistema.

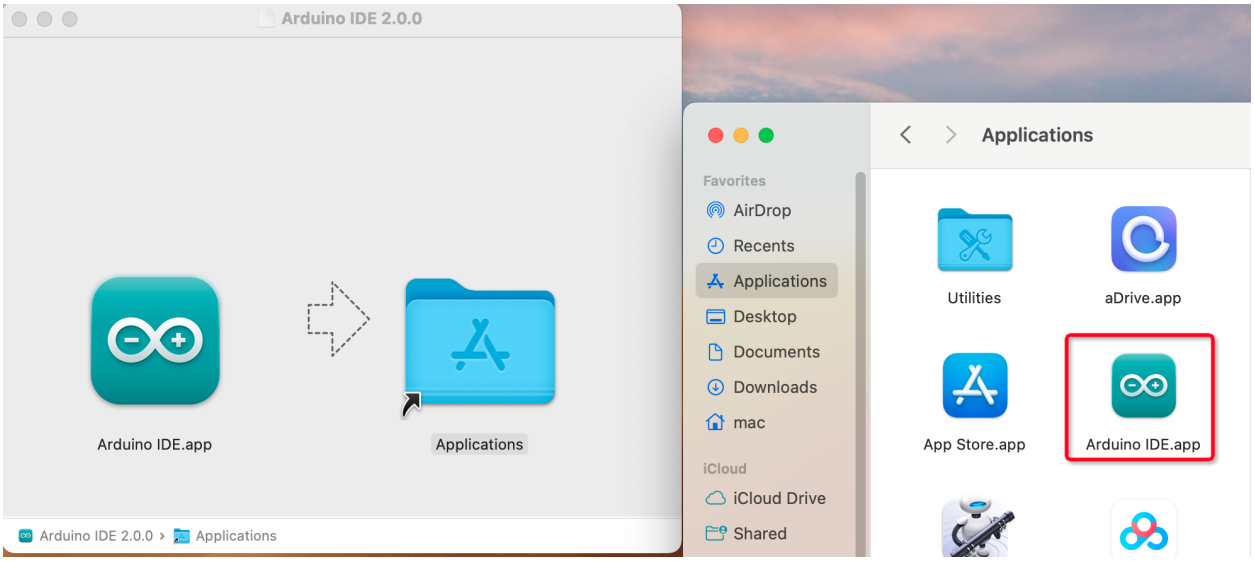


5. Luego Finaliza.



macOS

Haz doble clic en el archivo `arduino_ide_xxxx.dmg` descargado y sigue las instrucciones para copiar **Arduino IDE.app** a la carpeta **Aplicaciones**, verás el Arduino IDE instalado exitosamente después de unos segundos.

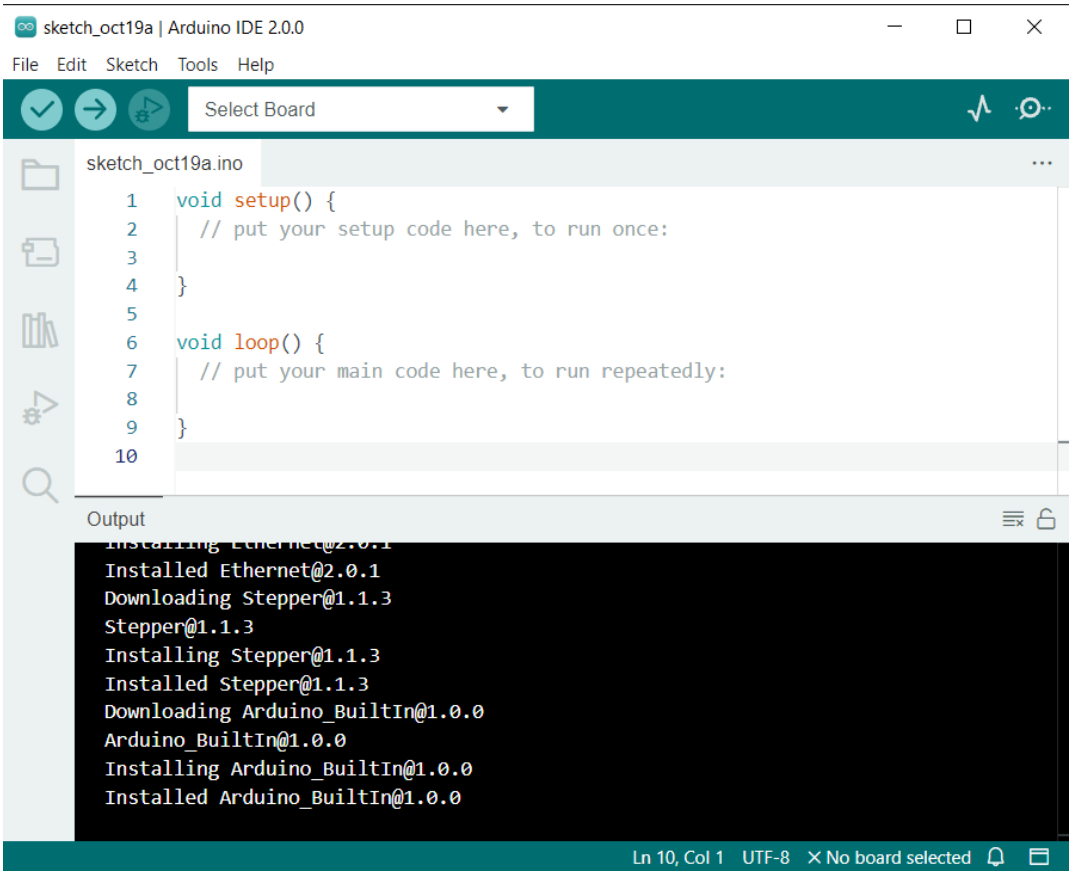


Linux

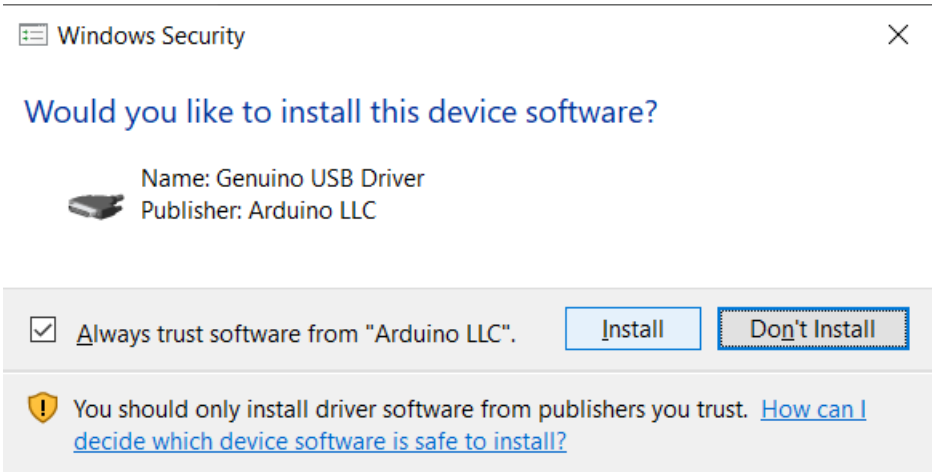
Para el tutorial sobre la instalación de Arduino IDE 2.0 en un sistema Linux, por favor consulta: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux

2.1.4 Abrir el IDE

1. Cuando abres por primera vez Arduino IDE 2.0, se instalan automáticamente las placas Arduino AVR, las librerías integradas y otros archivos requeridos.



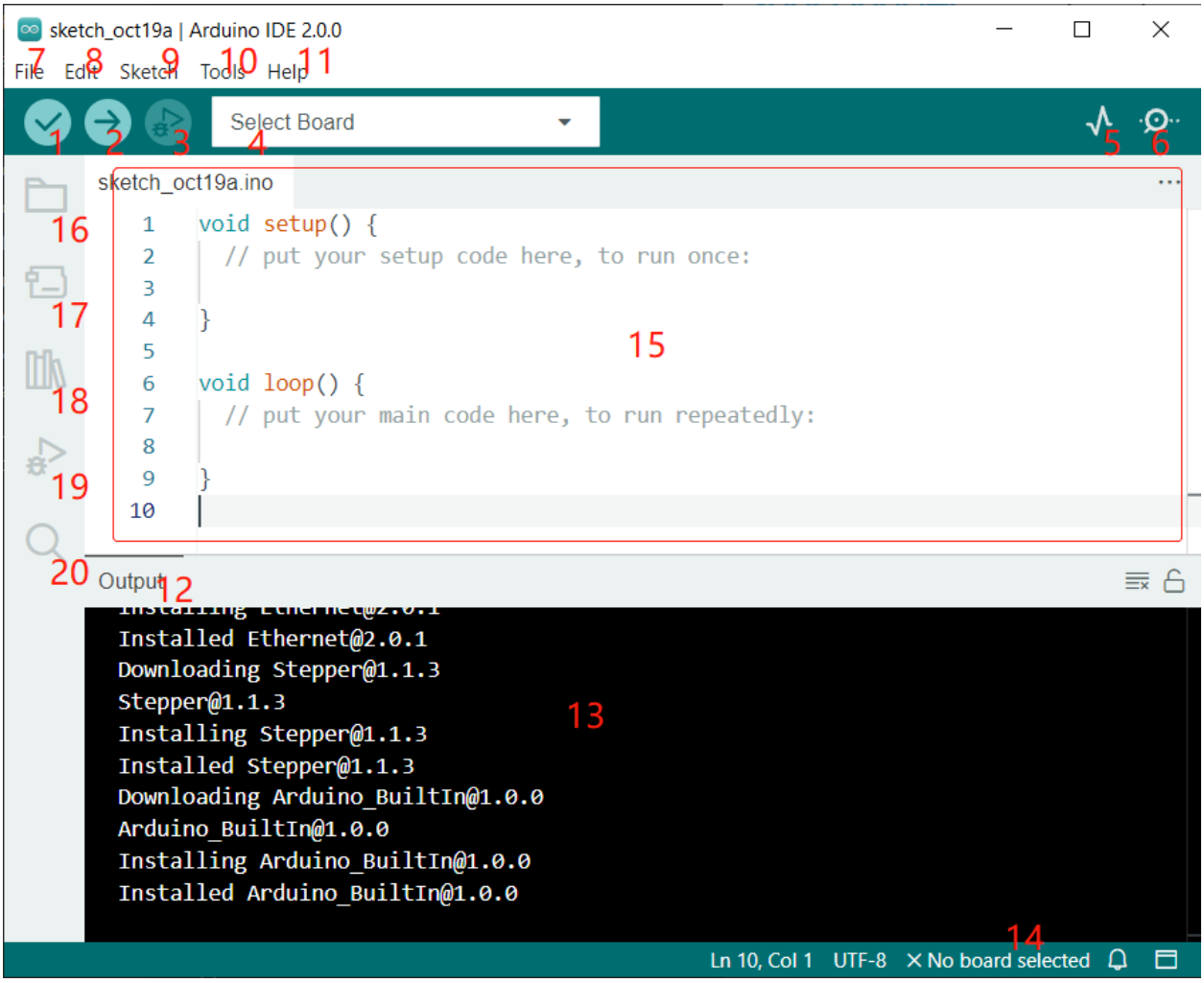
2. Además, tu firewall o centro de seguridad puede aparecer algunas veces preguntándote si deseas instalar algún controlador de dispositivo. Por favor, instala todos ellos.



3. ¡Ahora tu Arduino IDE está listo!

Nota: En el caso de que algunas instalaciones no funcionen debido a problemas de red u otras razones, puedes reabrir el Arduino IDE y este completará el resto de la instalación. La ventana de salida no se abrirá automáticamente después de que todas las instalaciones estén completas, a menos que hagas clic en Verificar o Subir.

2.2 1.2 Introducción a Arduino IDE



1. **Verificar**: Compila tu código. Cualquier problema de sintaxis se indicará con errores.
2. **Subir**: Sube el código a tu placa. Cuando hagas clic en el botón, los LEDs RX y TX en la placa parpadearán rápidamente y no se detendrán hasta que la subida haya terminado.
3. **Depurar**: Para la comprobación de errores línea por línea.
4. **Seleccionar Placa**: Configuración rápida de placa y puerto.
5. **Trazador Serial**: Verifica el cambio del valor leído.
6. **Monitor Serial**: Haz clic en el botón y aparecerá una ventana. Recibe los datos enviados desde tu placa de control. Es muy útil para depurar.
7. **Archivo**: Haz clic en el menú y aparecerá una lista desplegable, incluyendo la creación, apertura, guardado, cierre de archivos, configuración de algunos parámetros, etc.
8. **Editar**: Haz clic en el menú. En la lista desplegable, hay algunas operaciones de edición como **Cortar**, **Copiar**, **Pegar**, **Buscar**, y así sucesivamente, con sus correspondientes atajos.
9. **Boceto**: Incluye operaciones como **Verificar**, **Subir**, **Agregar** archivos, etc. La función más importante es **Incluir Librería** - donde puedes añadir librerías.

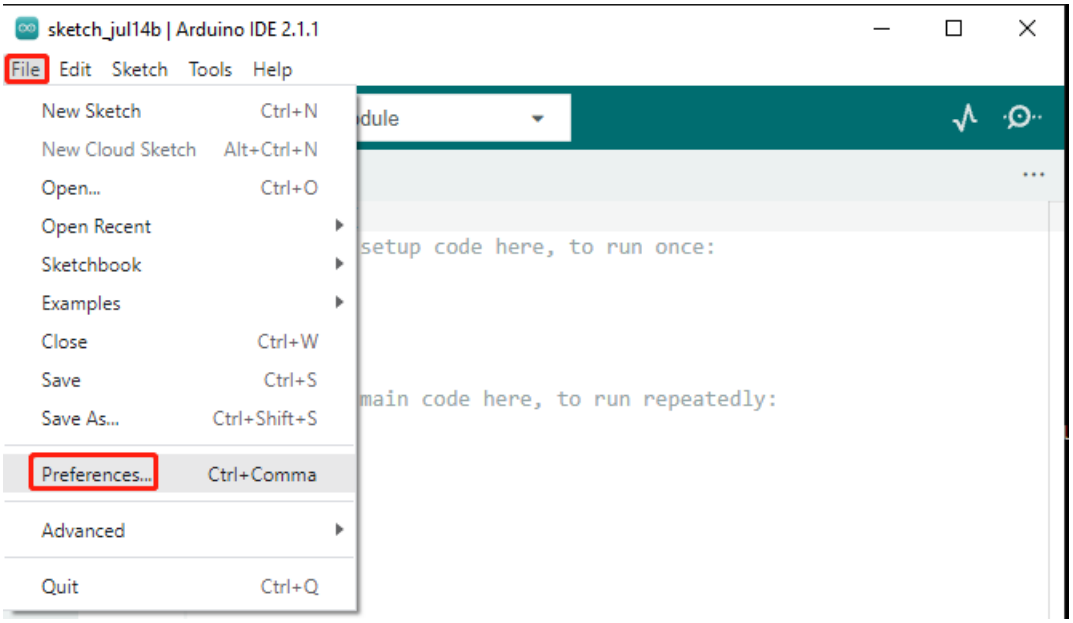
10. **Herramientas**: Incluye algunas herramientas - las más usadas son Placa (la placa que usas) y Puerto (el puerto en el que está tu placa). Cada vez que quieras subir el código, necesitas seleccionar o verificarlos.
11. **Ayuda**: Si eres principiante, puedes consultar las opciones bajo el menú y obtener la ayuda que necesitas, incluyendo operaciones en IDE, información de introducción, resolución de problemas, explicación de código, etc.
12. **Barra de Salida**: Cambia la pestaña de salida aquí.
13. **Ventana de Salida**: Imprime información.
14. **Placa y Puerto**: Aquí puedes previsualizar la placa y el puerto seleccionados para la subida del código. Puedes seleccionarlos de nuevo por **Herramientas** -> **Placa / Puerto** si alguno es incorrecto.
15. El área de edición del IDE. Puedes escribir código aquí.
16. **Libreta de Bocetos**: Para gestionar archivos de bocetos.
17. **Gestor de Placas**: Para gestionar el controlador de la placa.
18. **Gestor de Librerías**: Para gestionar tus archivos de librería.
19. **Depurar**: Ayuda a depurar el código.
20. **Buscar**: Busca los códigos de tus bocetos.

2.3 1.3 Instalar la Placa ESP32 (Importante)

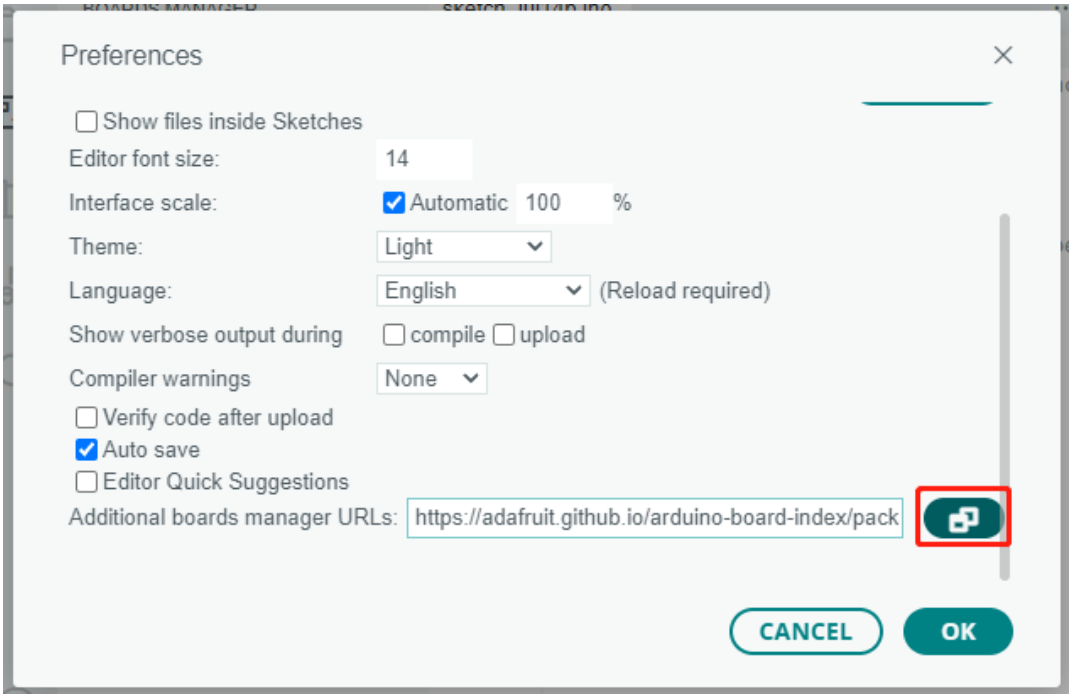
Para programar el microcontrolador ESP32, necesitamos instalar el paquete de la placa ESP32 en el Arduino IDE. Sigue la guía paso a paso a continuación:

Instalar la Placa ESP32

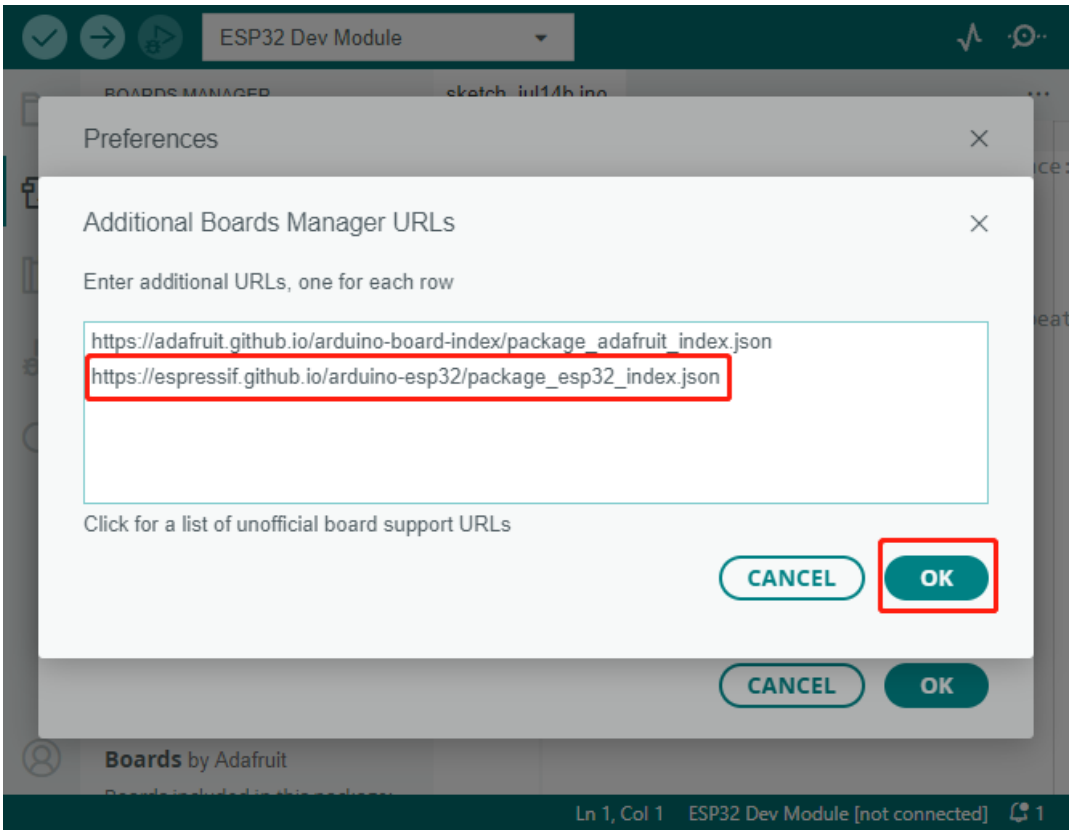
1. Abre el Arduino IDE. Ve a **Archivo** y selecciona **Preferencias** en el menú desplegable.



2. En la ventana de Preferencias, localiza el campo **URLs Adicionales de Gestores de Tarjetas**. Haz clic en él para activar el cuadro de texto.

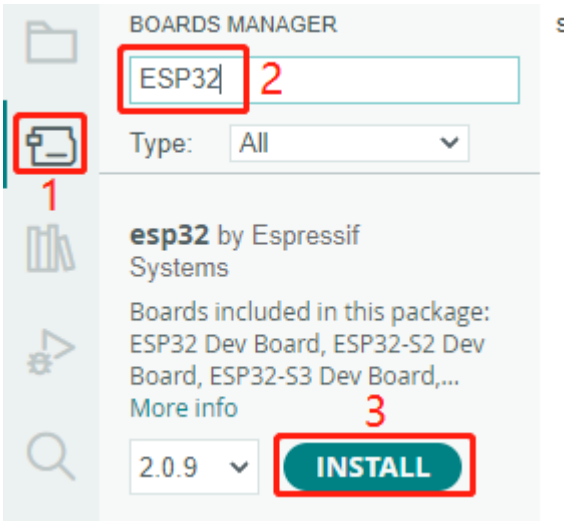


3. Añade la siguiente URL al campo **URLs Adicionales de Gestores de Tarjetas**: https://espressif.github.io/arduino-esp32/package_esp32_index.json. Esta URL apunta al archivo índice del paquete para las placas ESP32. Haz clic en el botón **OK** para guardar los cambios.



4. En la ventana del **Gestor de Tarjetas**, escribe **ESP32** en la barra de búsqueda. Haz clic en el botón **Instalar**

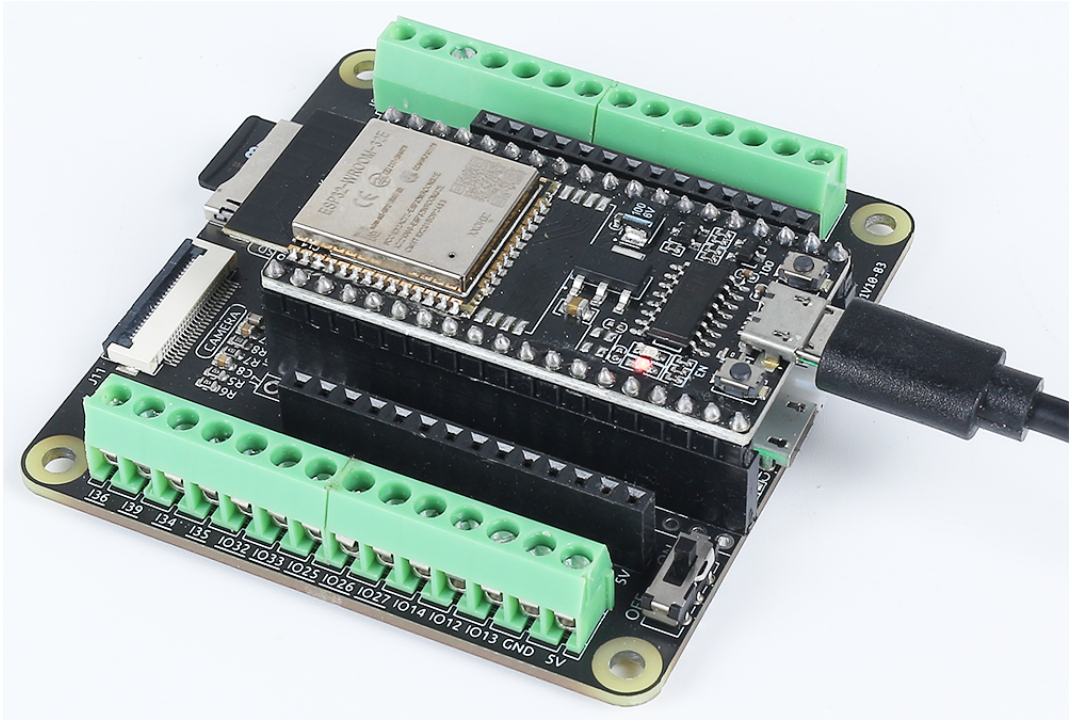
para comenzar el proceso de instalación. Esto descargará e instalará el paquete de la placa ESP32.



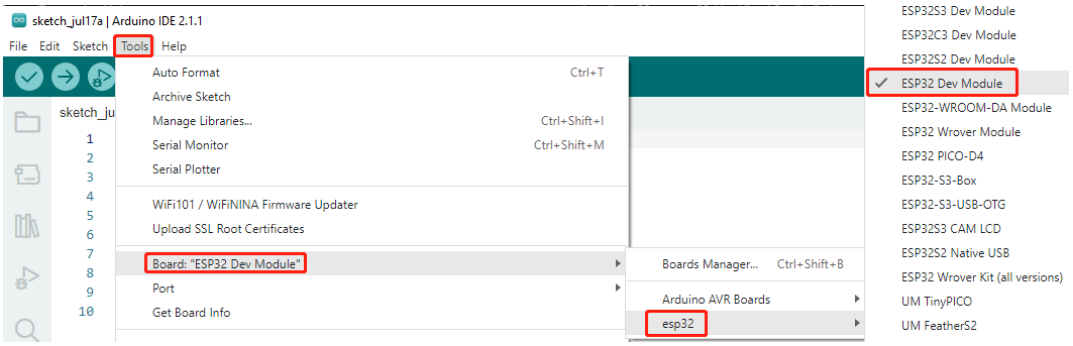
5. ¡Felicidades! Has instalado con éxito el paquete de la placa ESP32 en el Arduino IDE.

Subir el Código

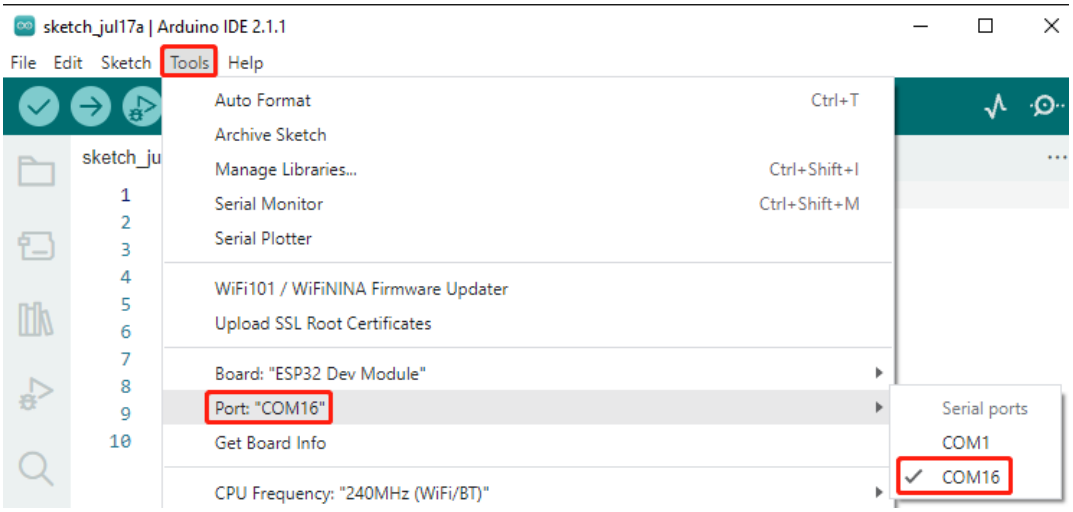
1. Ahora, conecta el ESP32 WROOM 32E a tu computadora usando un cable Micro USB.



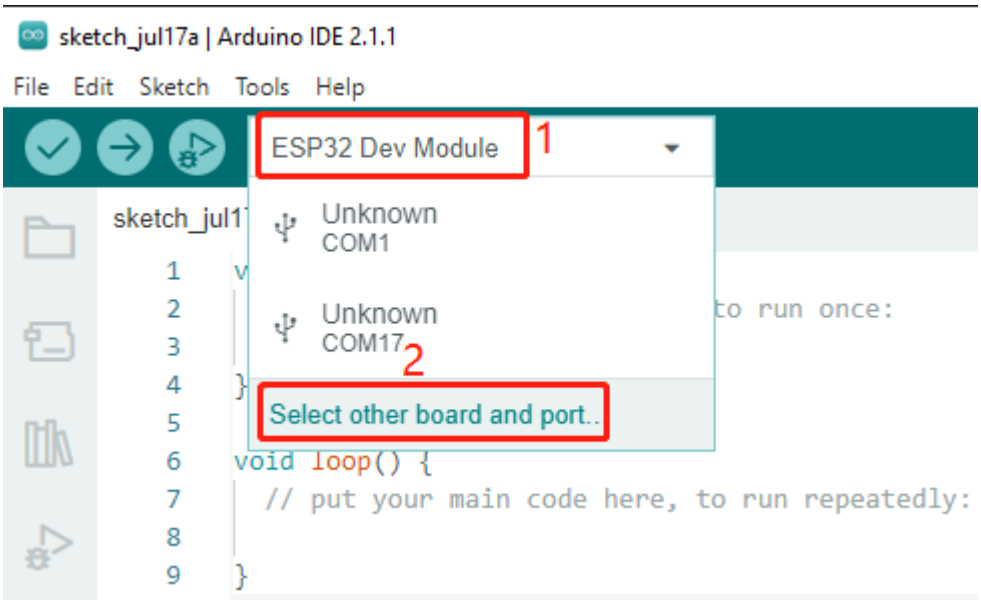
2. Luego selecciona la placa correcta, **ESP32 Dev Module**, haciendo clic en **Herramientas** -> **Placa** -> **esp32**.



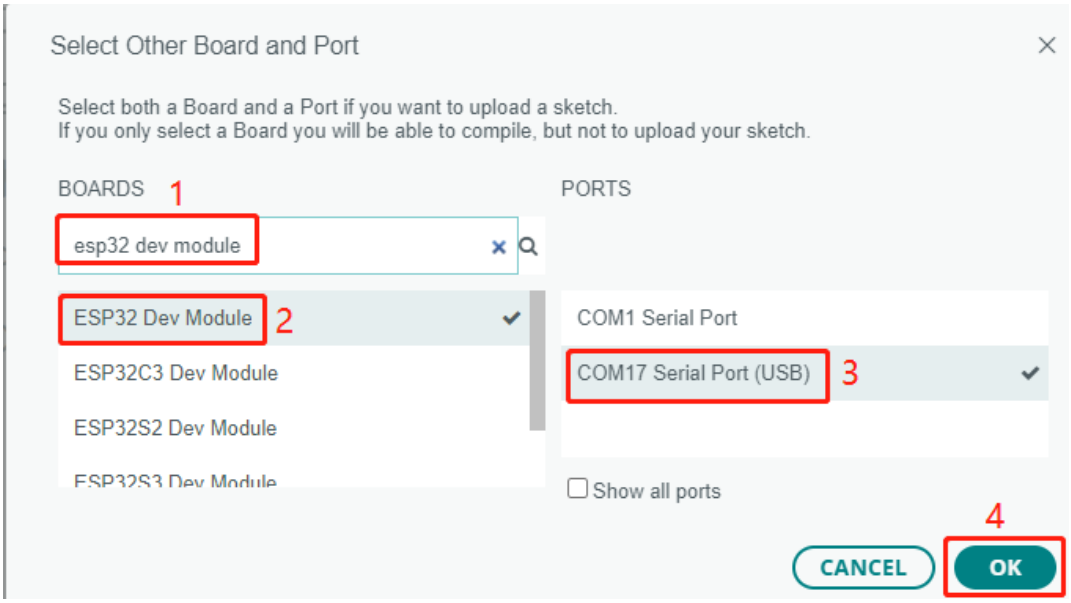
3. Si tu ESP32 está conectado a la computadora, puedes elegir el puerto correcto haciendo clic en **Herramientas -> Puerto**.



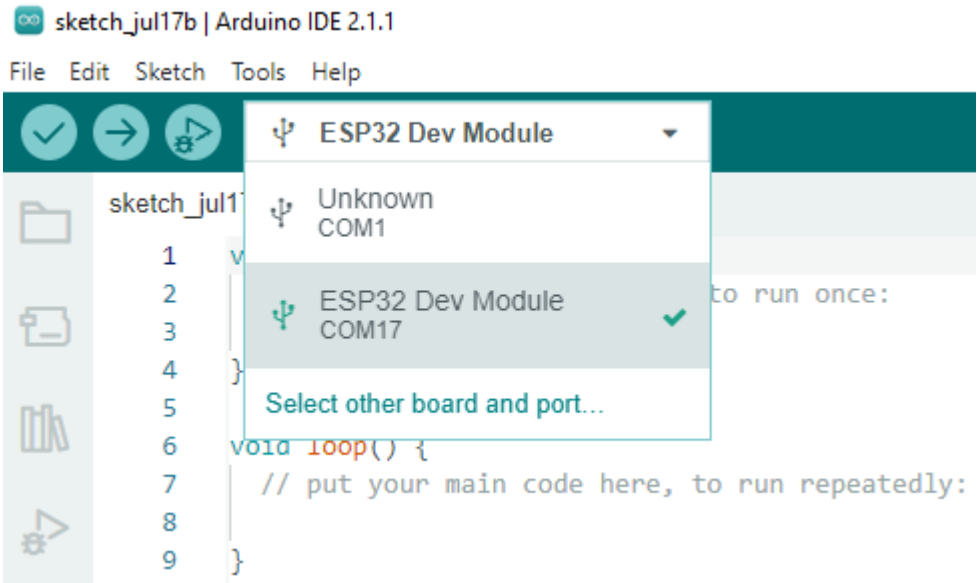
4. Además, Arduino 2.0 introdujo una nueva forma de seleccionar rápidamente la placa y el puerto. Para el ESP32, usualmente no se reconoce automáticamente, por lo que necesitas hacer clic en **Seleccionar otra placa y puerto**.



5. En el cuadro de búsqueda, escribe **ESP32 Dev Module** y selecciónalo cuando aparezca. Luego, elige el puerto correcto y haz clic en **OK**.



6. Después, puedes seleccionarlo a través de esta ventana de acceso rápido. Ten en cuenta que durante el uso subsiguiente, puede haber momentos en los que ESP32 no esté disponible en la ventana de acceso rápido, y necesitarás repetir los dos pasos anteriores.



7. Ambos métodos te permiten seleccionar la placa y el puerto correctos, así que elige el que mejor se adapte a ti. Ahora, todo está listo para subir el código al ESP32.

2.4 1.4 Instalación de librerías (Importante)

Una librería es una colección de código o funciones preescritas que amplían las capacidades del IDE de Arduino. Las librerías proporcionan código listo para usar en diversas funcionalidades, permitiéndote ahorrar tiempo y esfuerzo en la codificación de características complejas.

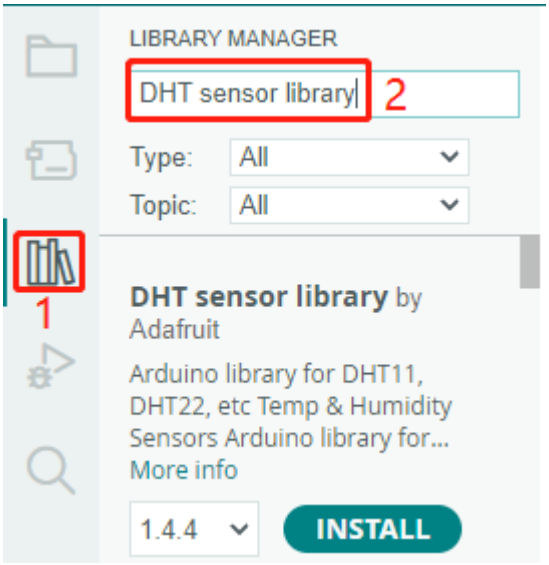
Existen dos maneras principales de instalar librerías:

2.4.1 Instalación desde el Gestor de Librerías

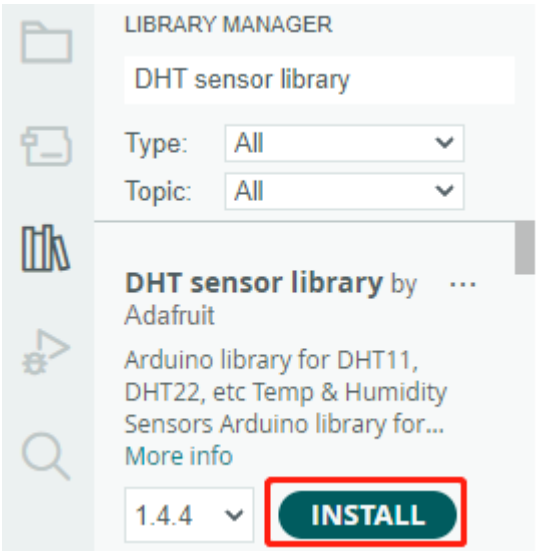
Muchas librerías están disponibles directamente a través del Gestor de Librerías de Arduino. Puedes acceder al Gestor de Librerías siguiendo estos pasos:

1. En el **Gestor de Librerías**, puedes buscar la librería deseada por nombre o navegar a través de diferentes categorías.

Nota: En proyectos donde se requiere la instalación de librerías, habrá indicaciones sobre qué librerías instalar. Sigue las instrucciones proporcionadas, como «Se utiliza aquí la librería del sensor DHT, puedes instalarla desde el Gestor de Librerías.» Simplemente instala las librerías recomendadas según se indica.



2. Una vez que encuentres la librería que deseas instalar, haz clic en ella y luego en el botón **Instalar**.

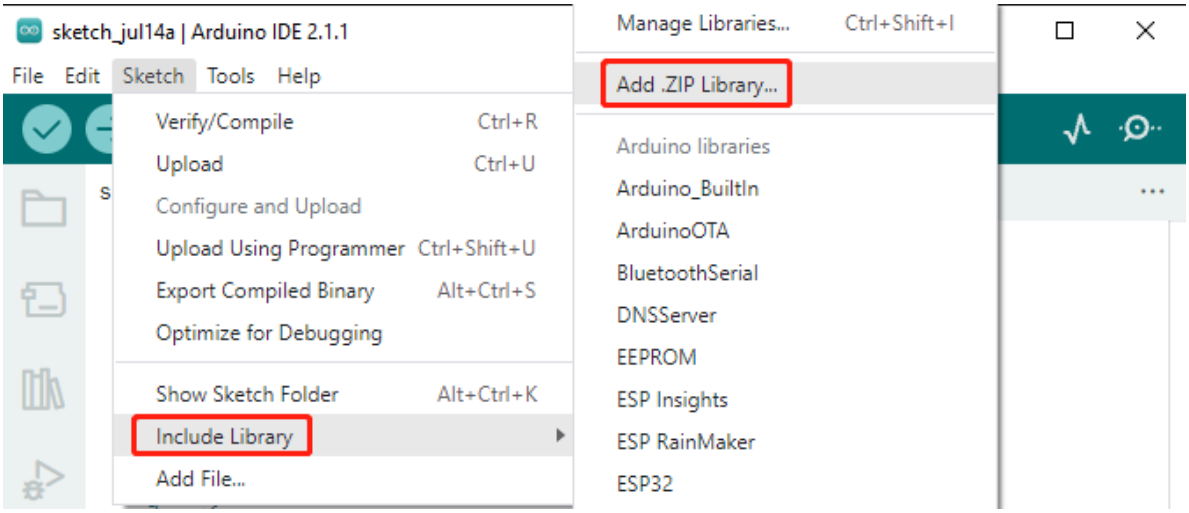


3. El IDE de Arduino descargará e instalará automáticamente la librería para ti.

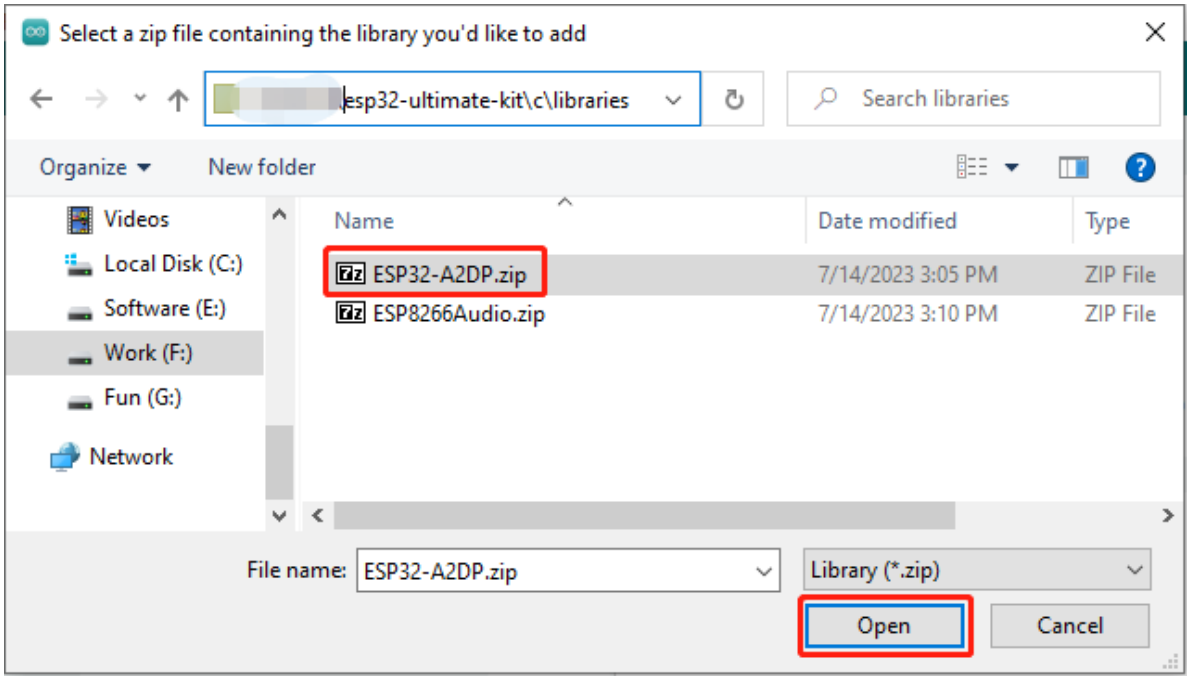
2.4.2 Instalación Manual

Algunas librerías no están disponibles a través del **Gestor de Librerías** y necesitan ser instaladas manualmente. Para instalar estas librerías, sigue estos pasos:

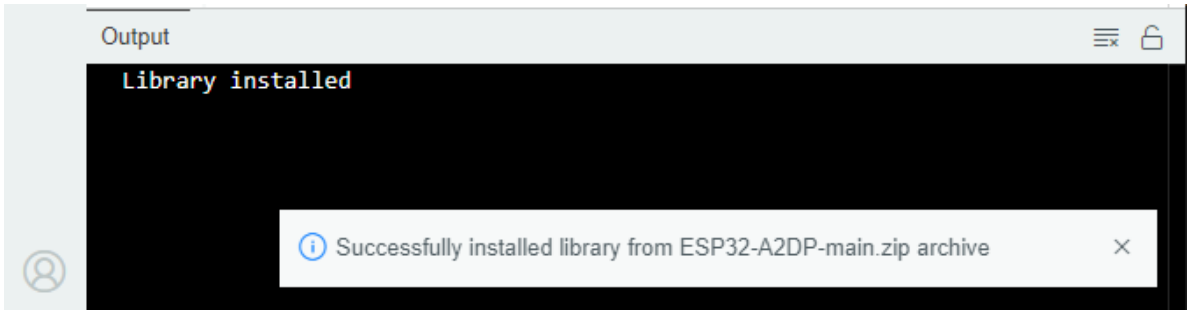
1. Abre el IDE de Arduino y ve a **Sketch** -> **Incluir Librería** -> **Añadir Librería .ZIP**.



2. Navega al directorio donde se encuentran los archivos de la librería, como la carpeta `esp32-starter-kit\c\libraries`, y selecciona el archivo de librería deseado, como `ESP32-A2DP.zip`. Luego, haz clic en **Abrir**.



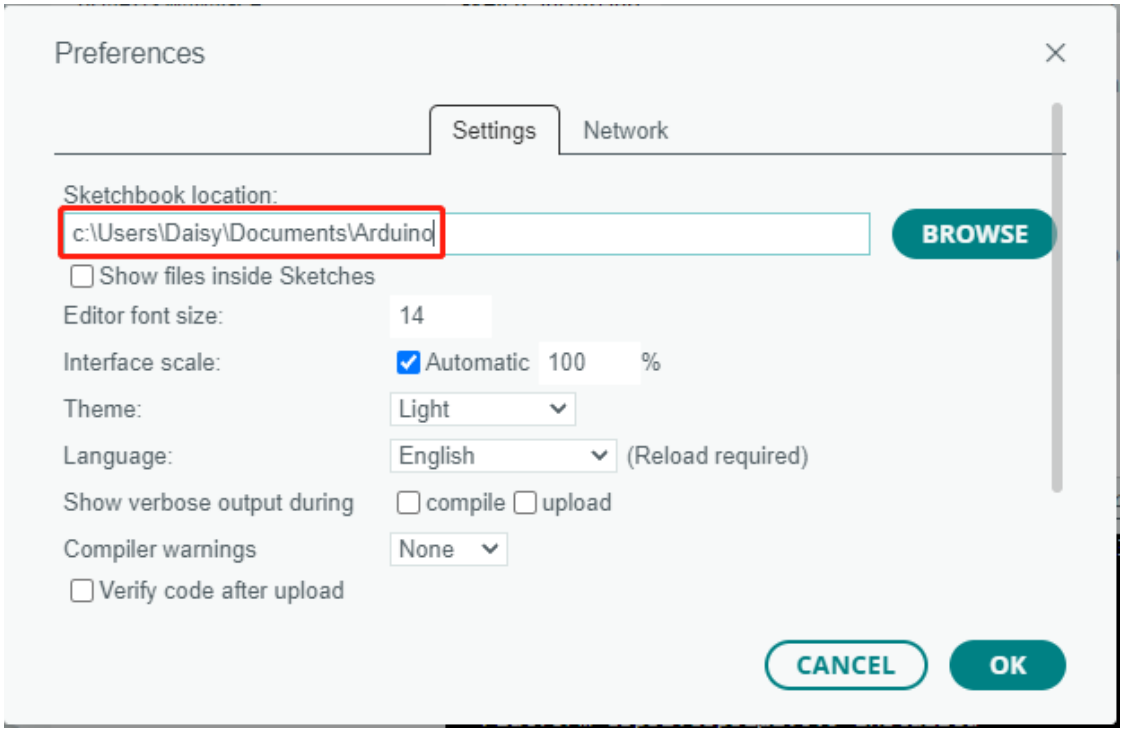
3. Después de un corto tiempo, recibirás una notificación indicando una instalación exitosa.



4. Repite el mismo proceso para añadir la librería `ESP8266Audio.zip`.

Nota: Las librerías instaladas usando cualquiera de los métodos anteriores se pueden encontrar en el directorio de librerías predeterminado del IDE de Arduino, que generalmente está ubicado en `C:\Usuarios\xxx\Documentos\Arduino\libraries`.

Si tu directorio de librerías es diferente, puedes verificarlo yendo a **Archivo** -> **Preferencias**.



2. Displays

2.5 2.1 ¡Hola, LED!

Así como imprimir «¡Hola, mundo!» es el primer paso para aprender a programar, usar un programa para encender un LED es la introducción tradicional para aprender programación física.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

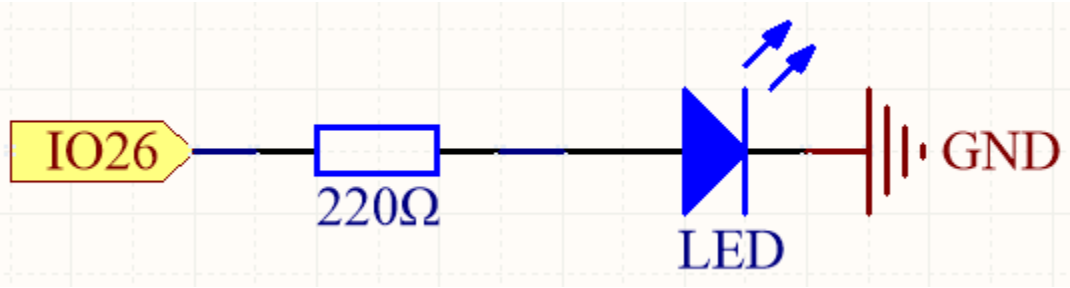
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

Pines Disponibles

Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

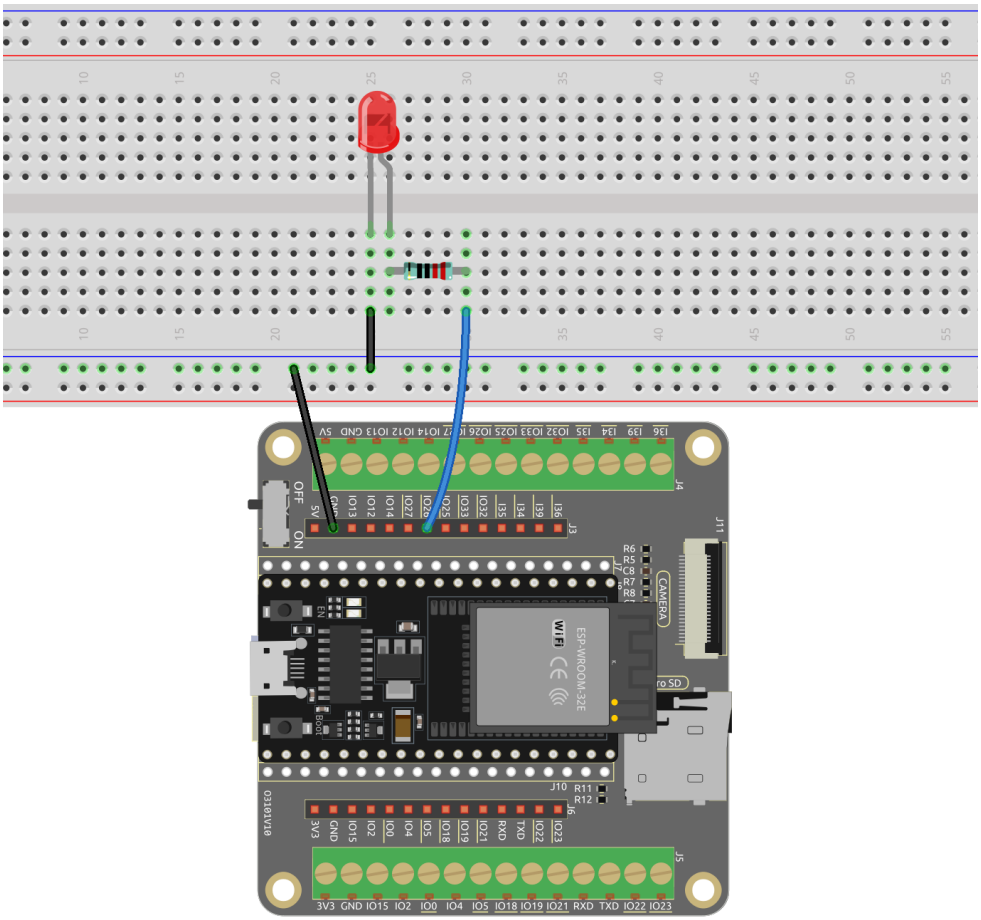
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Este circuito funciona sobre un principio simple, y la dirección de la corriente se muestra en la figura. El LED se encenderá después de la resistencia limitadora de corriente de 220ohm cuando el pin26 emita un nivel alto. El LED se apagará cuando el pin26 emita un nivel bajo.

Cableado

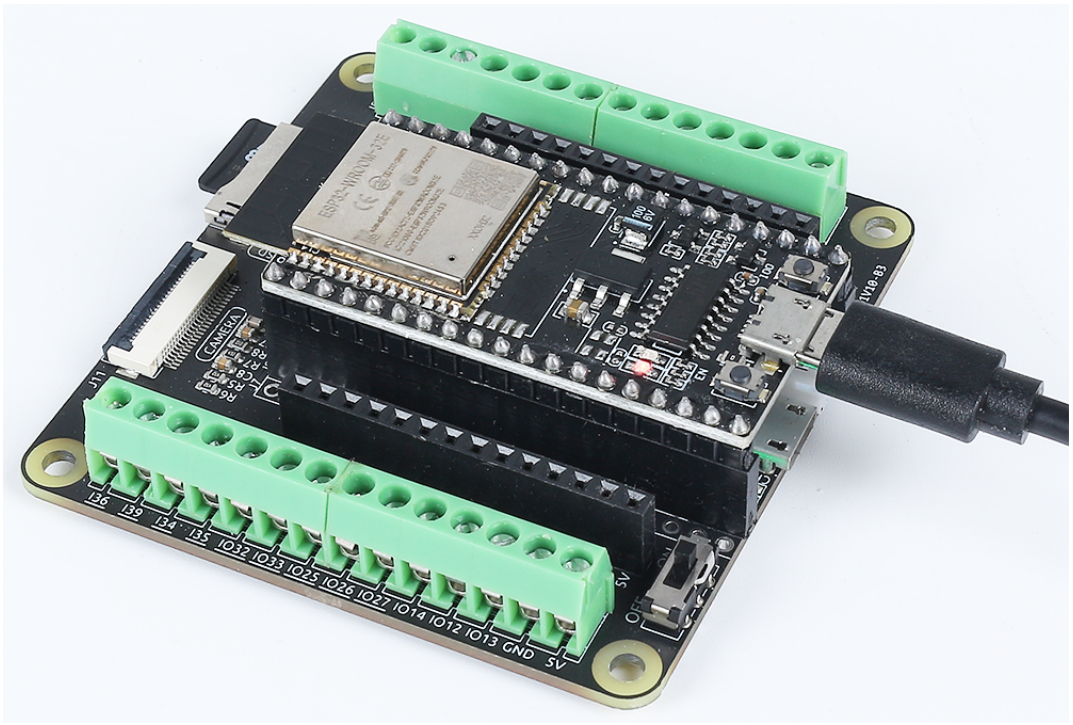


Subir Código

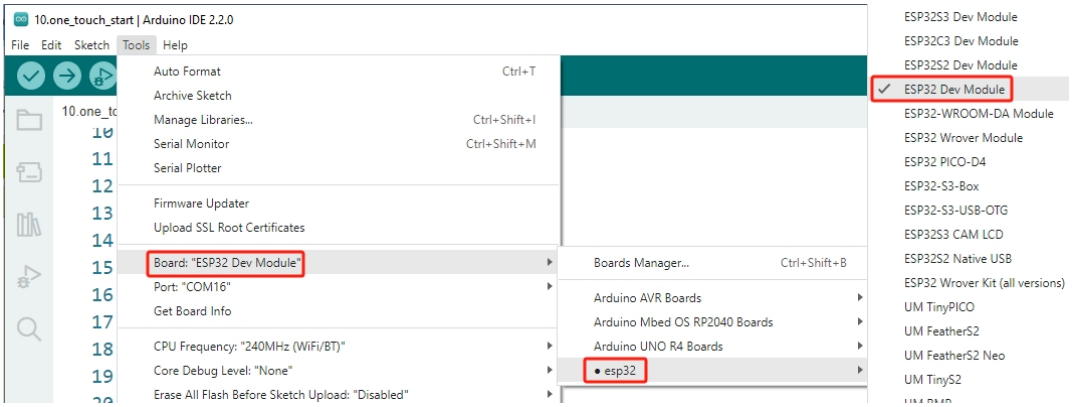
1. Puedes abrir el archivo `2.1_hello_led.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.1_hello_led`. O copia este código directamente en el IDE de Arduino.

2. Luego conecta el ESP32 WROOM 32E a tu computadora usando un cable Micro USB.

    - *¿Siempre aparece «COMxx desconocido»?*



3. Selecciona la placa (ESP32 Dev Module) y el puerto apropiado.



4. Ahora, haz clic en el botón **Subir** para cargar el código a la placa ESP32.

5. Después de que el código se haya subido con éxito, verás el LED parpadeando.

¿Cómo funciona?

1. Declara una constante entera llamada `ledPin` y asígnale el valor 26.

```
const int ledPin = 26;  // The GPIO pin for the LED
```

2. Ahora, inicializa el pin en la función `setup()`, donde necesitas inicializar el pin a modo `OUTPUT`.

```
void setup() {
    pinMode(ledPin, OUTPUT);
}
```

   - `void pinMode(uint8_t pin, uint8_t mode);`: Esta función se utiliza para definir el modo de operación GPIO para un pin específico.
     - `pin` define el número de pin GPIO.
     - `mode` establece el modo de operación.

     Los siguientes modos son compatibles para la entrada y salida básicas:
     - `INPUT` configura el GPIO como entrada sin pullup ni pulldown (alta impedancia).
     - `OUTPUT` configura el GPIO como modo de salida/lectura.
     - `INPUT_PULLDOWN` configura el GPIO como entrada con el pull-down interno.
     - `INPUT_PULLUP` configura el GPIO como entrada con el pull-up interno.

3. La función `loop()` contiene la lógica principal del programa y se ejecuta continuamente. Alterna entre establecer el pin en alto y bajo, con intervalos de un segundo entre los cambios.

```
void loop() {
    digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage␣
↪level)
    delay(1000);                       // wait for a second
    digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage␣
↪LOW
    delay(1000);                       // wait for a second
}
```

   - `void digitalWrite(uint8_t pin, uint8_t val);`: Esta función establece el estado del GPIO seleccionado en `HIGH` o `LOW`. Esta función solo se utiliza si el `pinMode` se configuró como `OUTPUT`.
     - `pin` define el número de pin GPIO.
     - `val` establece el estado digital de salida en `HIGH` o `LOW`.

2.6 2.2 Desvanecimiento

En el proyecto anterior, controlamos el LED encendiéndolo y apagándolo usando salida digital. En este proyecto, crearemos un efecto de respiración en el LED utilizando Modulación por Ancho de Pulso (PWM). PWM es una técnica que nos permite controlar el brillo de un LED o la velocidad de un motor variando el ciclo de trabajo de una señal de onda cuadrada.

Con PWM, en lugar de simplemente encender o apagar el LED, estaremos ajustando la cantidad de tiempo que el LED está encendido versus la cantidad de tiempo que está apagado dentro de cada ciclo. Al cambiar rápidamente el LED de encendido a apagado en intervalos variables, podemos crear la ilusión de que el LED se ilumina y se atenúa gradualmente, simulando un efecto de respiración.

Usando las capacidades de PWM del ESP32 WROOM 32E, podemos lograr un control suave y preciso sobre el brillo del LED. Este efecto de respiración añade un elemento dinámico y visualmente atractivo a tus proyectos, creando una exhibición llamativa o un ambiente.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

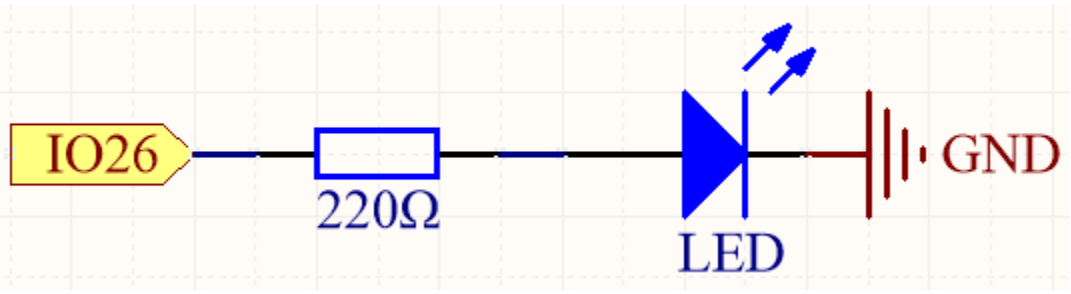
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

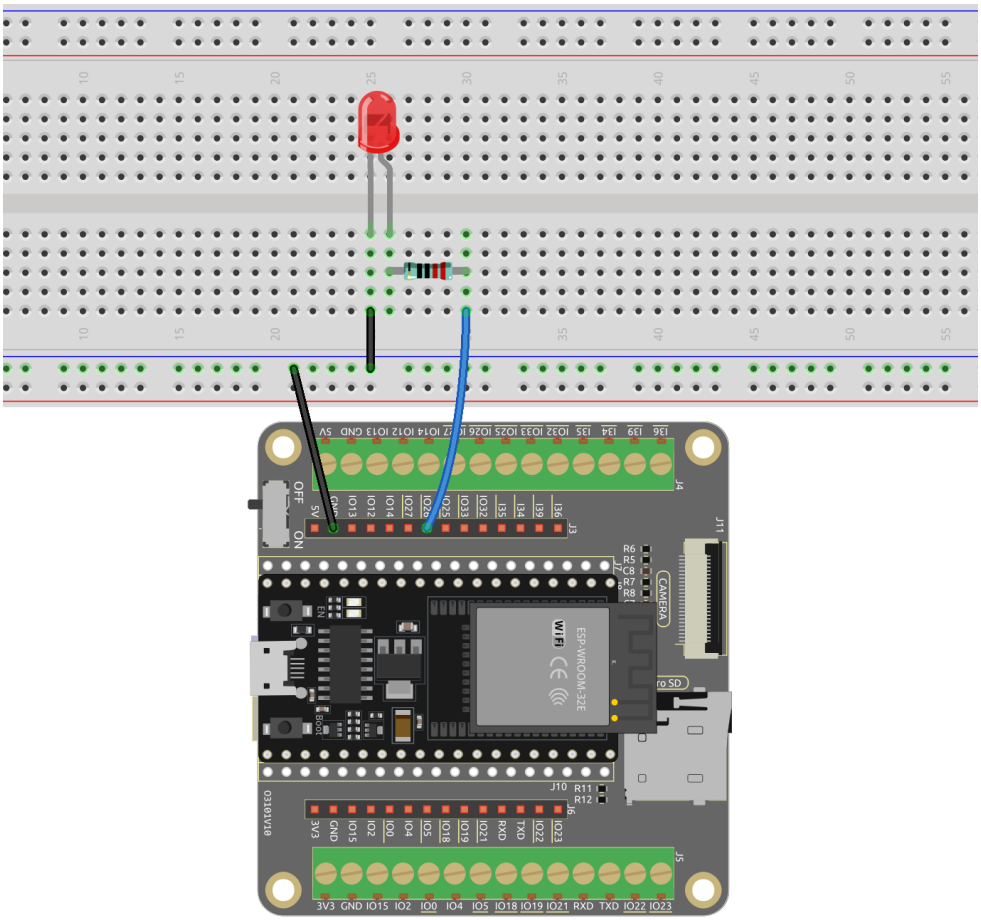
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Este proyecto es el mismo circuito que el primer proyecto *2.1 ¡Hola, LED!*, pero el tipo de señal es diferente. El primer proyecto es para emitir niveles altos y bajos digitales (0&1) directamente desde el pin26 para hacer que el LED se ilumine o se apague, este proyecto es para emitir señal PWM desde el pin26 para controlar el brillo del LED.

Cableado



Código

Nota:

- Puedes abrir el archivo `2.2_fading_led.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.2_fading_led`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de que el código se haya subido con éxito, puedes ver el LED respirando.

¿Cómo funciona?

1. Define constantes y variables

```
const int ledPin = 26; // The GPIO pin for the LED
int brightness = 0;
int fadeAmount = 5;
```

   - `ledPin`: El número de pin GPIO donde está conectado el LED (en este caso, GPIO 26).
   - `brightness`: El nivel actual de brillo del LED (inicialmente establecido en 0).
   - `fadeAmount`: La cantidad por la cual el brillo del LED cambiará en cada paso (establecido en 5).

2. Inicializa el canal PWM y configura el pin del LED.

```
void setup() {
    ledcSetup(0, 5000, 8); // Configure the PWM channel (0) with 5000Hz␣
↪frequency and 8-bit resolution
    ledcAttachPin(ledPin, 0); // Attach the LED pin to the PWM channel
}
```

Aquí usamos el periférico (control LED) que está diseñado principalmente para controlar la intensidad de los LEDs, aunque también se puede usar para generar señales PWM para otros fines.

- `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Esta función se utiliza para configurar la frecuencia y resolución del canal LEDC. Devolverá la `frecuencia` configurada para el canal LEDC. Si se devuelve 0, ocurre un error y el canal ledc no fue configurado.
  - `channel` selecciona el canal LEDC para configurar.
  - `freq` selecciona la frecuencia del pwm.
  - `resolution_bits` selecciona la resolución para el canal ledc. El rango es de 1-14 bits (1-20 bits para ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Esta función se utiliza para asociar el pin al canal LEDC.
  - `pin` selecciona el pin GPIO.
  - `chan` selecciona el canal LEDC.

3. La función `loop()` contiene la lógica principal del programa y se ejecuta continuamente. Actualiza el brillo del LED, invierte la cantidad de desvanecimiento cuando el brillo alcanza el valor mínimo o máximo, e introduce un retraso.

```
void loop() {
    ledcWrite(0, brightness); // Write the new brightness value to the PWM␣
↪channel
    brightness = brightness + fadeAmount;

    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }

    delay(50); // Wait for 20 milliseconds
    }
```

- `void ledcWrite(uint8_t chan, uint32_t duty);`: Esta función se utiliza para establecer el deber para el canal LEDC.
  - `chan` selecciona el canal LEDC para escribir el deber.
  - `duty` selecciona el deber a ser establecido para el canal seleccionado.

2.7 2.3 Luz Colorida

En este proyecto, nos adentraremos en el fascinante mundo de la mezcla de colores aditiva utilizando un LED RGB.

El LED RGB combina tres colores primarios, a saber, Rojo, Verde y Azul, en un solo paquete. Estos tres LEDs comparten un pin de cátodo común, mientras que cada pin de ánodo controla la intensidad del color correspondiente.

Variando la intensidad de la señal eléctrica aplicada a cada ánodo, podemos crear una amplia gama de colores. Por ejemplo, mezclar luz roja y verde de alta intensidad resultará en luz amarilla, mientras que combinar luz azul y verde producirá cian.

A través de este proyecto, exploraremos los principios de la mezcla de colores aditiva y desataremos nuestra creatividad manipulando el LED RGB para mostrar colores cautivadores y vibrantes.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

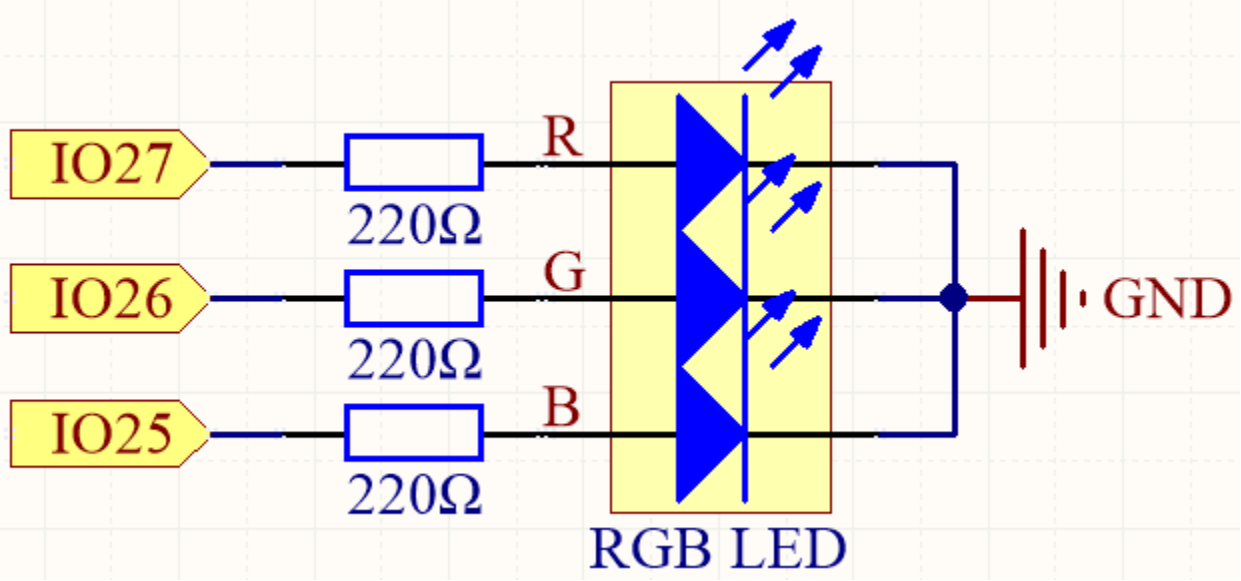
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED RGB	

Pines Disponibles

Aquí está la lista de pines disponibles en la placa ESP32 para este proyecto.

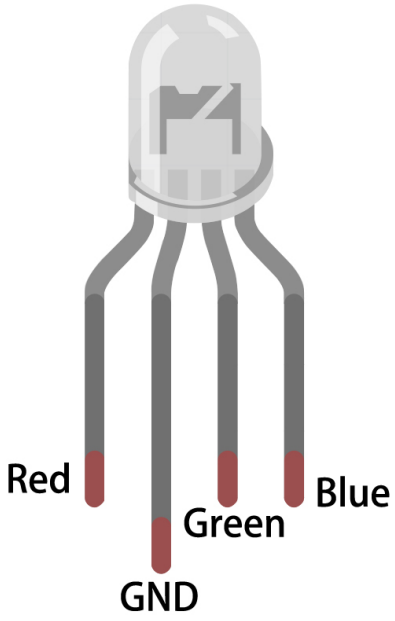
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquema

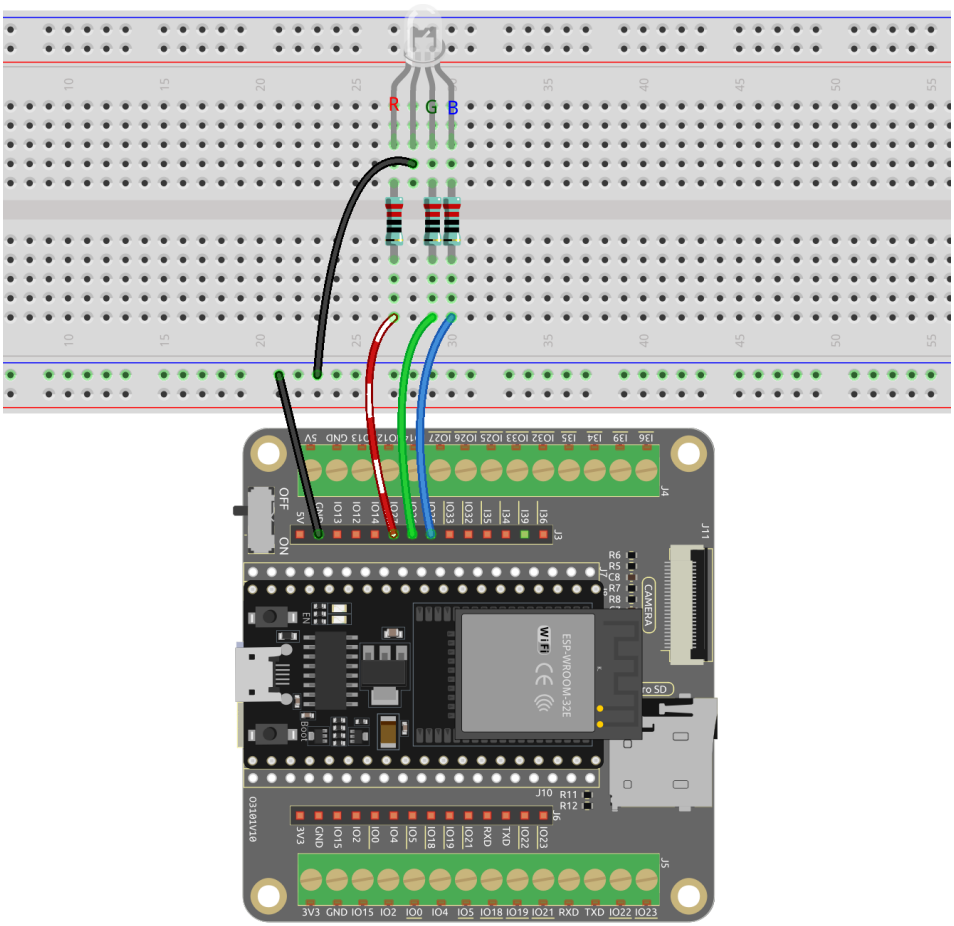


Los pines PWM pin27, pin26 y pin25 controlan los pines Rojo, Verde y Azul del LED RGB respectivamente, y conectan el pin de cátodo común a GND. Esto permite que el LED RGB muestre un color específico superponiendo luz en estos pines con diferentes valores PWM.

Cableado



El LED RGB tiene 4 pines: el pin largo es el pin de cátodo común, que generalmente se conecta a GND; el pin izquierdo junto al pin más largo es Rojo; y los dos pines a la derecha son Verde y Azul.

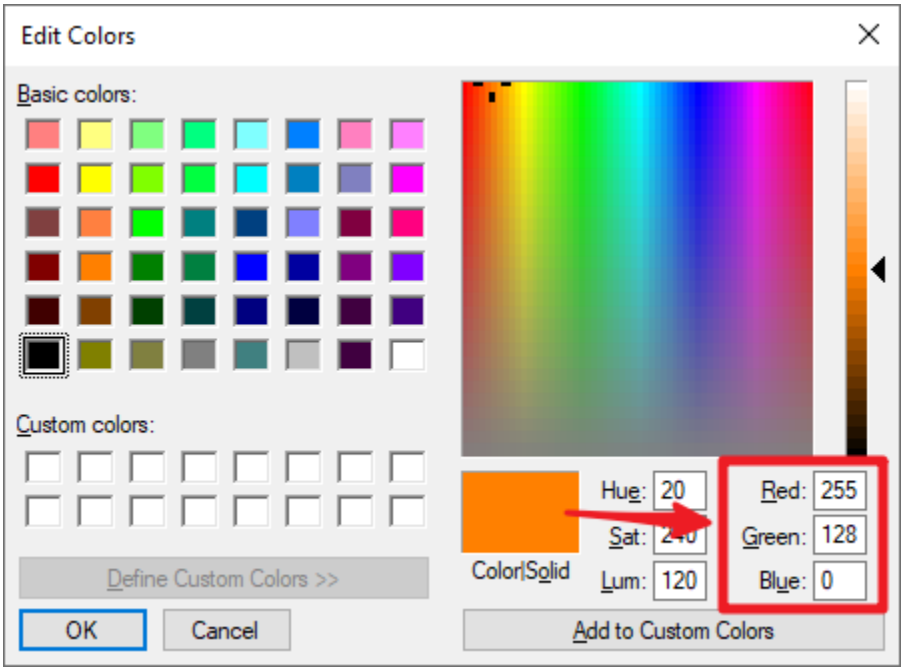


Código

Aquí, podemos elegir nuestro color favorito en software de dibujo (como paint) y mostrarlo con el LED RGB.

Nota:

- Puedes abrir el archivo `2.3_rgb_led.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.3_rgb_led`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*



Escribe el valor RGB en `color_set()`, podrás ver el LED RGB iluminar los colores que deseas.

¿Cómo funciona?

1. Define los pines GPIO, los canales PWM y la frecuencia (en Hz) y resolución (en bits).

```
// Define RGB LED pins
const int redPin = 27;
const int greenPin = 26;
const int bluePin = 25;

// Define PWM channels
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;

// Define PWM frequency and resolution
const int freq = 5000;
const int resolution = 8;
```

2. La función `setup()` inicializa los canales PWM con la frecuencia y resolución especificadas, y luego asocia los pines del LED a sus respectivos canales PWM.

```
void setup() {
    // Set up PWM channels
    ledcSetup(redChannel, freq, resolution);
    ledcSetup(greenChannel, freq, resolution);
    ledcSetup(blueChannel, freq, resolution);

    // Attach pins to corresponding PWM channels
    ledcAttachPin(redPin, redChannel);
    ledcAttachPin(greenPin, greenChannel);
    ledcAttachPin(bluePin, blueChannel);
```

(continué en la próxima página)

(proviene de la página anterior)

```
}
```

Aquí utilizamos el periférico (control de LED), diseñado primordialmente para controlar la intensidad de los LEDs, aunque también puede ser usado para generar señales PWM para otros propósitos. * `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Esta función se utiliza para configurar la frecuencia y resolución del canal LEDC. Devolverá la `frecuencia` configurada para el canal LEDC. Si se devuelve 0, se produce un error y el canal LEDC no fue configurado.

- `channel`: selecciona el canal LEDC a configurar.
- `freq`: selecciona la frecuencia del PWM.
- `resolution_bits`: selecciona la resolución para el canal LEDC. El rango es de 1-14 bits (1-20 bits para ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Esta función se utiliza para asociar el pin al canal LEDC.
  - `pin`: selecciona el pin GPIO.
  - `chan`: selecciona el canal LEDC.

3. La función `loop()` cicla a través de varios colores (rojo, verde, azul, amarillo, púrpura y cian) con intervalos de un segundo entre cada cambio de color.

```
void loop() {
    setColor(255, 0, 0); // Red
    delay(1000);
    setColor(0, 255, 0); // Green
    delay(1000);
    setColor(0, 0, 255); // Blue
    delay(1000);
    setColor(255, 255, 0); // Yellow
    delay(1000);
    setColor(80, 0, 80); // Purple
    delay(1000);
    setColor(0, 255, 255); // Cyan
    delay(1000);
}
```

4. La función `setColor()` establece el color deseado escribiendo los valores de ciclo de trabajo apropiados en cada canal PWM. La función toma tres argumentos enteros para los valores de color rojo, verde y azul.

```
void setColor(int red, int green, int blue) {
    // For common-anode RGB LEDs, use 255 minus the color value
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

- `void ledcWrite(uint8_t chan, uint32_t duty);`: Esta función se utiliza para establecer el ciclo de trabajo para el canal LEDC.
  - `chan`: selecciona el canal LEDC para escribir el ciclo de trabajo.
  - `duty`: selecciona el ciclo de trabajo a establecer para el canal seleccionado.

2.8 2.4 Microchip - 74HC595

¡Bienvenido a este emocionante proyecto! En este proyecto, utilizaremos el chip 74HC595 para controlar un flujo luminoso de 8 LEDs.

Imagina activar este proyecto y ser testigo de un flujo hipnotizante de luz, como si un arcoíris chispeante saltara entre los 8 LEDs. Cada LED se iluminará uno por uno y se apagará rápidamente, mientras el siguiente LED continúa brillando, creando un efecto hermoso y dinámico.

Utilizando de manera ingeniosa el chip 74HC595, podemos controlar los estados de encendido y apagado de múltiples LEDs para lograr el efecto de flujo. Este chip tiene múltiples pines de salida que pueden conectarse en serie para controlar la secuencia de iluminación de los LEDs. Además, gracias a la capacidad de expansión del chip, podemos agregar fácilmente más LEDs al display de flujo, creando efectos aún más espectaculares.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

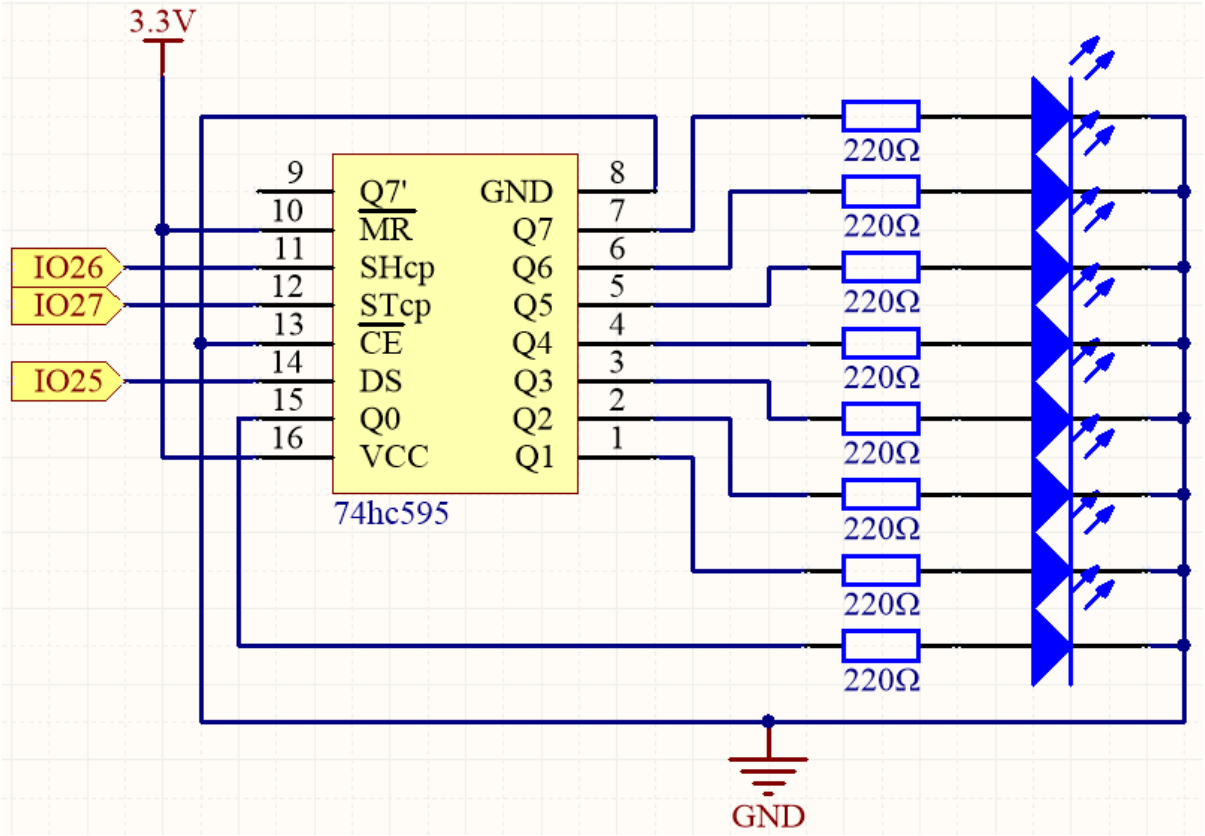
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
74HC595	

Pines Disponibles

Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

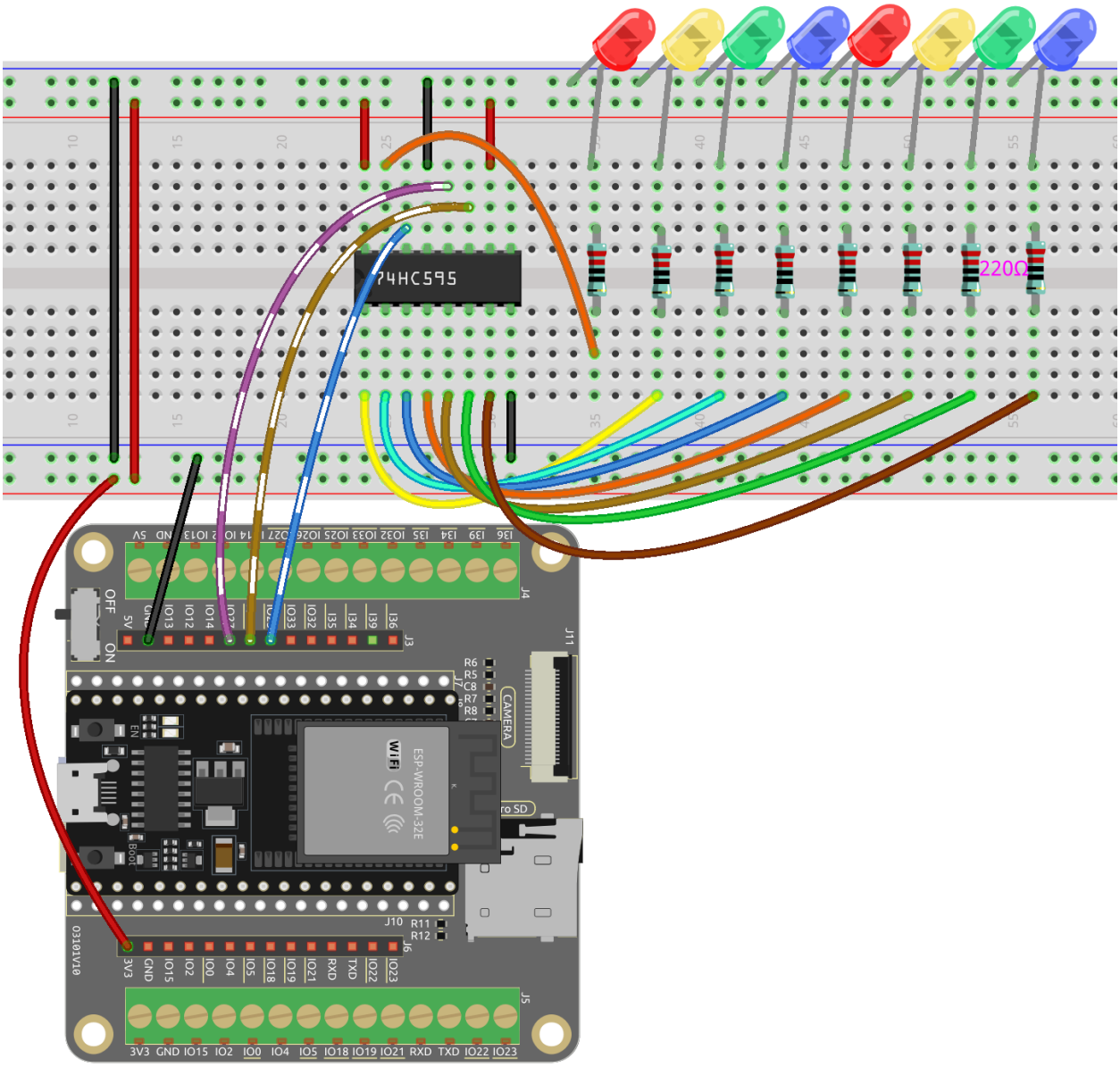
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



- Cuando MR (pin10) está a nivel alto y CE (pin13) está a nivel bajo, los datos se ingresan en el flanco ascendente de SHcp y pasan al registro de memoria a través del flanco ascendente de SHcp.
- Si los dos relojes están conectados juntos, el registro de desplazamiento siempre va un pulso antes que el registro de memoria.
- Hay un pin de entrada de desplazamiento serial (DS), un pin de salida serial (Q7") y un botón de reinicio asíncrono (nivel bajo) en el registro de memoria.
- El registro de memoria produce una salida de Bus con 8 bits paralelos y en tres estados.
- Cuando OE está habilitado (nivel bajo), los datos en el registro de memoria se envían al bus(Q0 ~ Q7).

Cableado



Código

Nota:

- Abre el archivo `2.4_74hc595.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.4_74hc595`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Cuando termines de subir los códigos a la placa ESP32, podrás ver los LEDs encendiéndose uno tras otro.

¿Cómo funciona?

1. Declara un arreglo, almacena varios números binarios de 8 bits que se utilizan para cambiar el estado de trabajo de los ocho LEDs controlados por 74HC595.

```
int datArray[] = {B00000000, B00000001, B00000011, B00000111, B00001111,␣
↪B00011111, B00111111, B01111111, B11111111};
```

2. Función `loop()`.

```
void loop()
    {
        for(int num = 0; num <10; num++)
        {
            digitalWrite(STcp,LOW); //Set ST_CP and hold low for as long as␣
↪you are transmitting
            shiftOut(DS,SHcp,MSBFIRST,datArray[num]);
            digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data
            delay(1000);
        }
    }
```

- Itera a través de `datArray[]`, enviando secuencialmente los valores binarios al registro de desplazamiento.
- Los comandos `digitalWrite(STcp, LOW)` y `digitalWrite(STcp, HIGH)` aseguran los datos en el registro de almacenamiento.
- La función `shiftOut()` envía los valores binarios desde `datArray[]` al registro de desplazamiento usando el pin de datos (DS) y el pin de reloj del registro de desplazamiento (SHcp). `MSBFIRST` significa moverse desde los bits más altos.
- Luego crea una pausa de 1 segundo entre cada actualización del patrón de LEDs.

2.9 2.5 Pantalla de 7 Segmentos

¡Bienvenido a este fascinante proyecto! En este proyecto, exploraremos el encantador mundo de mostrar números del 0 al 9 en una pantalla de siete segmentos.

Imagina activar este proyecto y ser testigo de cómo una pequeña y compacta pantalla brilla intensamente con cada número del 0 al 9. Es como tener una pantalla en miniatura que muestra los dígitos de una manera cautivadora. Controlando los pines de señal, puedes cambiar fácilmente el número mostrado y crear varios efectos atractivos.

A través de conexiones de circuito simples y programación, aprenderás cómo interactuar con la pantalla de siete segmentos y dar vida a los números deseados. Ya sea un contador, un reloj o cualquier otra aplicación intrigante, la pantalla de siete segmentos será tu compañero confiable, añadiendo un toque de brillantez a tus proyectos.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

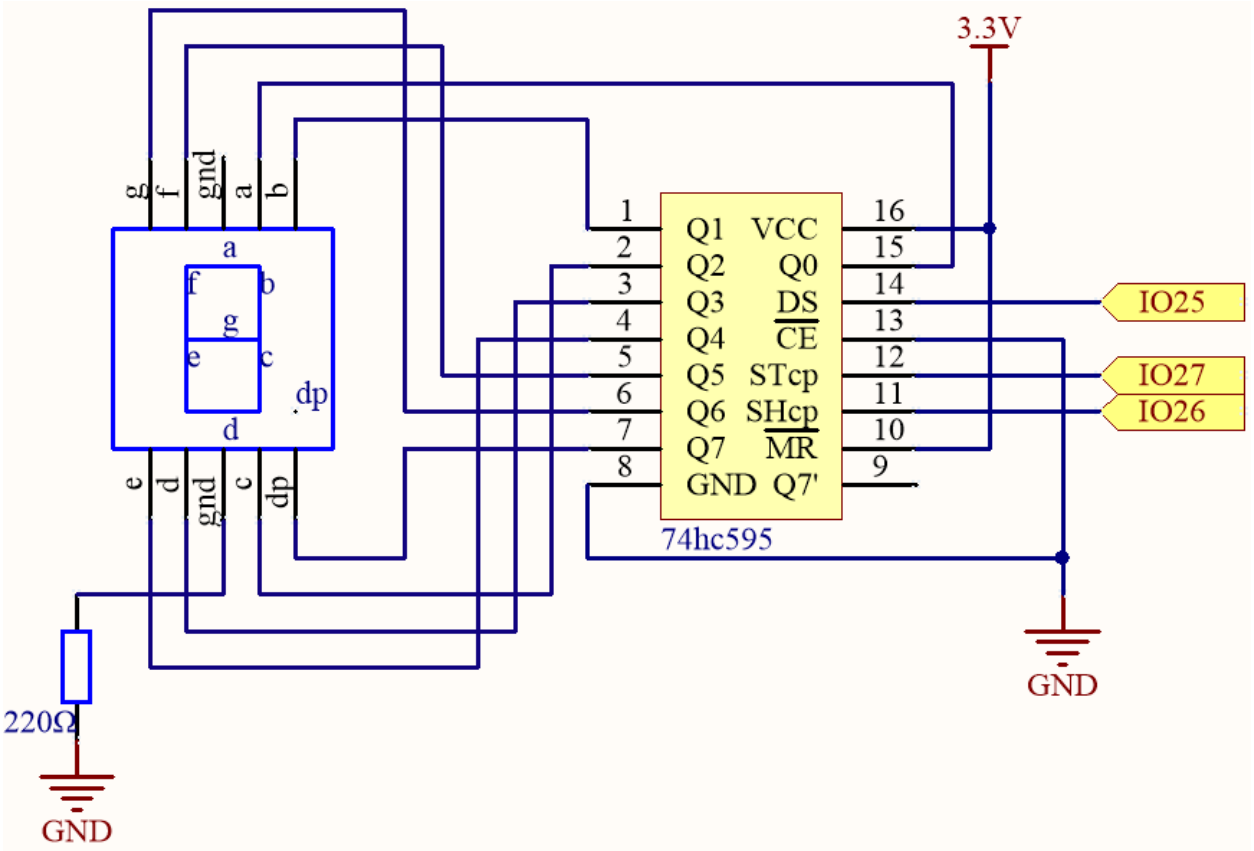
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
display de 7 Segmentos	
74HC595	

Pines Disponibles

Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático

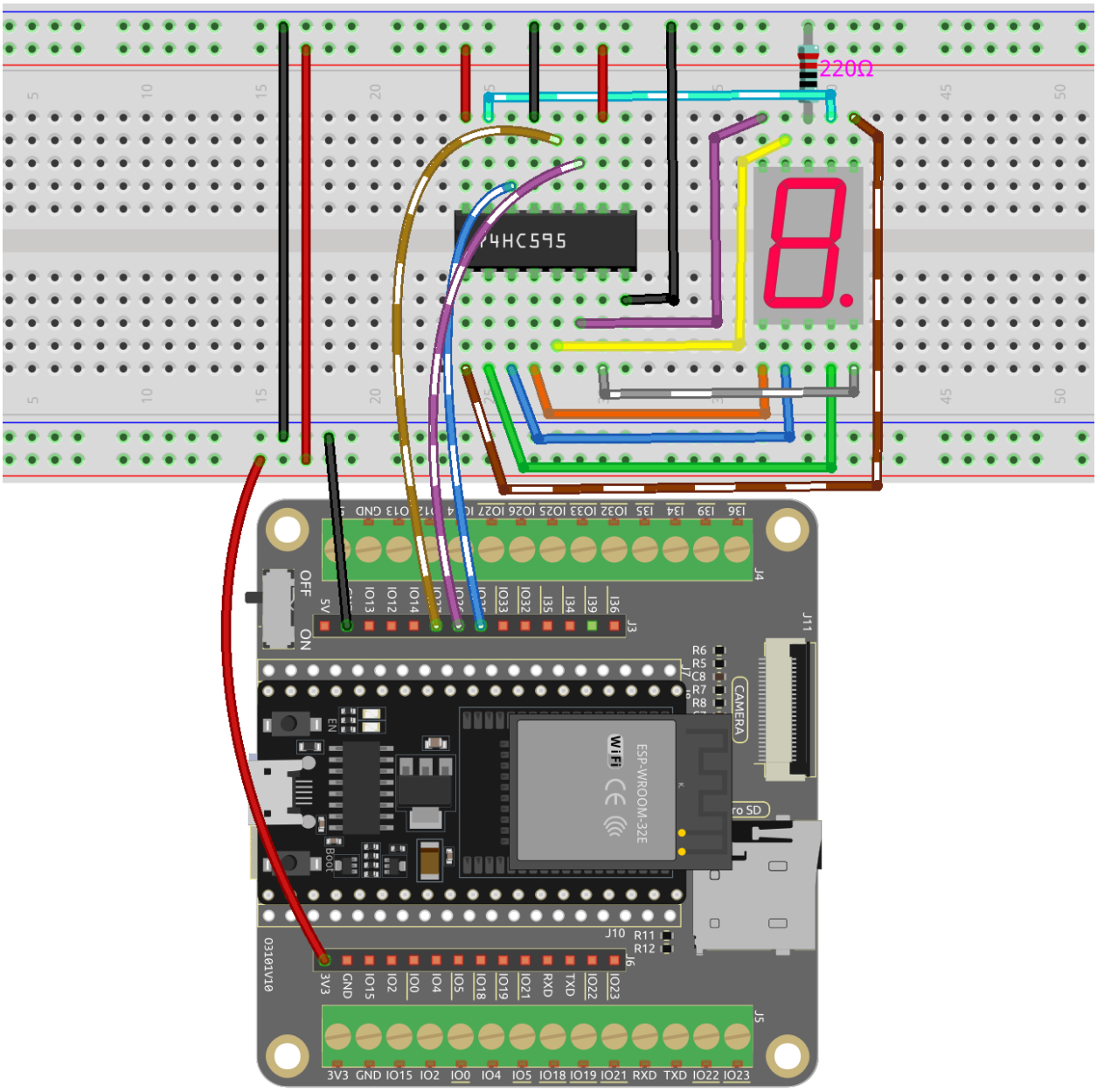


Aquí, el principio de cableado es básicamente el mismo que en *2.4 Microchip - 74HC595*, la única diferencia es que Q0-Q7 están conectados a los pines a ~ g de la Pantalla de 7 Segmentos.

Tabla 1: Cableado

74HC595	Pantalla LED de Segmento
Q0	a
Q1	b
Q2	c
Q3	d
Q4	e
Q5	f
Q6	g
Q7	dp

Cableado



Código

Nota:

- Abre el archivo `2.5_7segment.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.5_7segment`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir el código con éxito, podrás ver la Pantalla de Segmento LED mostrar 0~9 en secuencia.

¿Cómo funciona?

En este proyecto, estamos usando la función `shiftOut()` para escribir el número binario en el registro de desplazamiento.

Supongamos que la Pantalla de 7 Segmentos muestra el número «2». Este patrón de bits corresponde a los segmentos **f**, **c** y **dp** apagados (bajo), mientras que los segmentos **a**, **b**, **d**, **e** y **g** están encendidos (alto). Esto es «01011011» en binario y «0x5b» en notación hexadecimal.

Por lo tanto, necesitarías llamar a `shiftOut(DS,SHcp,MSBFIRST,0x5b)` para mostrar el número «2» en la pantalla de 7 segmentos.

- Hexadecimal
- Convertidor Binario a Hexadecimal

La siguiente tabla muestra los patrones hexadecimales que necesitan ser escritos en el registro de desplazamiento para mostrar los números del 0 al 9 en una pantalla de 7 segmentos.

Tabla 2: Código de Glifo

Números	Código Binario	Código Hex
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f

Escribe estos códigos en `shiftOut()` para hacer que la Pantalla de Segmento LED muestre los números correspondientes.

2.10 2.6 Mostrar Caracteres

Ahora, exploraremos el fascinante mundo de la visualización de caracteres usando el módulo I2C LCD1602.

A través de este proyecto, aprenderemos cómo inicializar el módulo LCD, establecer los parámetros de visualización deseados y enviar datos de caracteres para ser mostrados en la pantalla. Podemos exhibir mensajes personalizados, mostrar lecturas de sensores o crear menús interactivos. ¡Las posibilidades son infinitas!

Dominando el arte de la visualización de caracteres en el I2C LCD1602, desbloquearemos nuevas vías para la comunicación y la visualización de información en nuestros proyectos. Sumergámonos en este emocionante viaje y demos vida a nuestros caracteres en la pantalla LCD.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

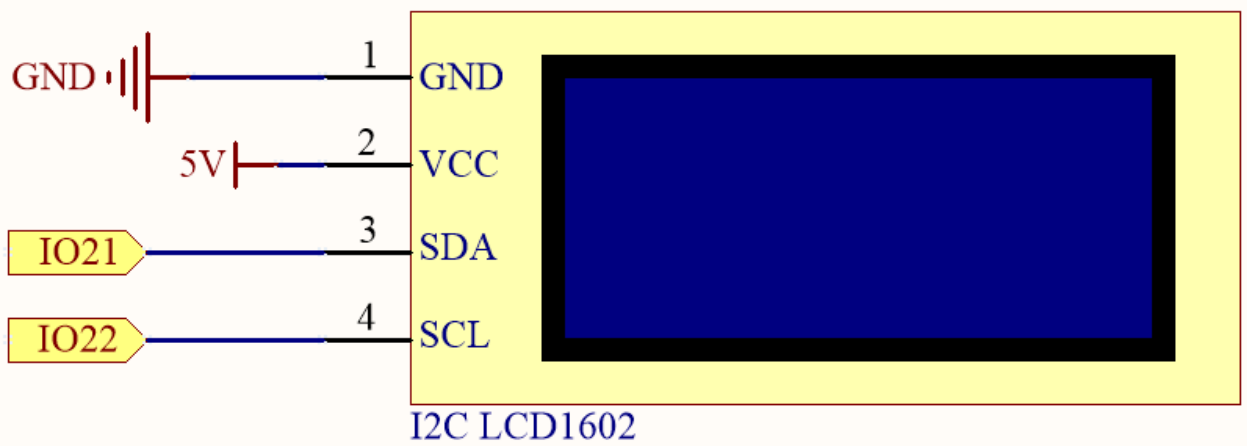
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
I2C LCD1602	

Pines Disponibles

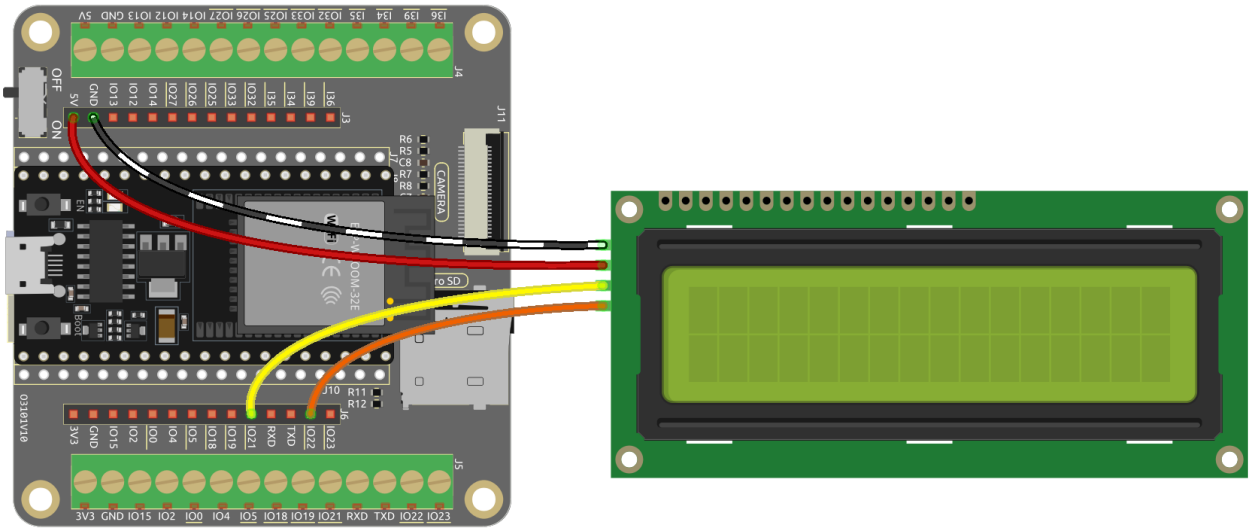
Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	Descripción de Uso
IO21	SDA
IO22	SCL

Esquemático



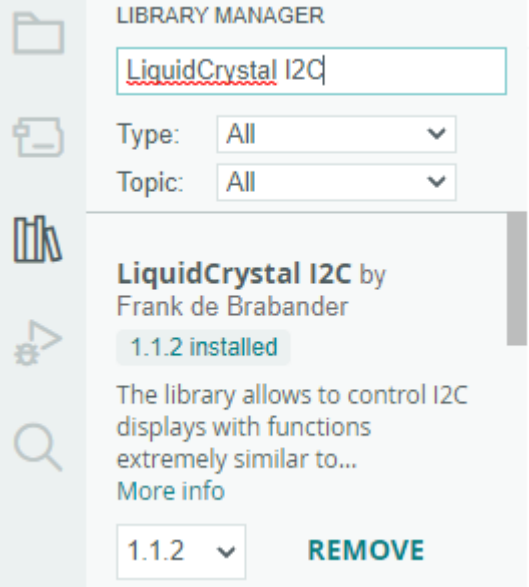
Cableado



Código

Nota:

- Abre el archivo `2.6_lcd1602.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.6_lcd1602`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- La biblioteca `LiquidCrystal I2C` se usa aquí, puedes instalarla desde el **Administrador de Bibliotecas**.



Cuando este programa se carga, el I2C LCD1602 mostrará el mensaje de bienvenida, «¡Hola, Sunfounder!», durante 3 segundos. Después de eso, la pantalla mostrará una etiqueta «CUENTA:» y el valor de cuenta, que se incrementa cada segundo.

Nota: Si el código y el cableado son correctos, pero el LCD todavía no muestra ningún contenido, puedes ajustar el potenciómetro en la parte trasera para aumentar el contraste.

¿Cómo funciona?

Al llamar a la biblioteca `LiquidCrystal_I2C.h`, puedes manejar fácilmente el LCD.

```
#include <LiquidCrystal_I2C.h>
```

Funciones de la Biblioteca:

- Crea una nueva instancia de la clase `LiquidCrystal_I2C` que representa un LCD particular adjunto a tu placa Arduino.

  ```
  LiquidCrystal_I2C(uint8_t lcd_Addr,uint8_t lcd_cols,uint8_t lcd_rows)
  ```

  - `lcd_AddR`: La dirección del LCD por defecto es 0x27.
  - `lcd_cols`: El LCD1602 tiene 16 columnas.
  - `lcd_rows`: El LCD1602 tiene 2 filas.

- Inicializa el lcd.

  ```
  void init()
  ```

- Enciende la luz de fondo (opcional).

  ```
  void backlight()
  ```

- Apaga la luz de fondo (opcional) rápidamente.

```
void nobacklight()
```

- Enciende la pantalla LCD.

```
void display()
```

- Apaga la pantalla LCD rápidamente.

```
void nodisplay()
```

- Limpia la pantalla, establece la posición del cursor en cero.

```
void clear()
```

- Establece la posición del cursor en col,row.

```
void setCursor(uint8_t col,uint8_t row)
```

- Imprime texto en el LCD.

```
void print(data,BASE)
```

  - `data`: Los datos a imprimir (char, byte, int, long, o string).
  - `BASE (opcional)`: La base en la que imprimir números.
    - `BIN` para binario (base 2)
    - `DEC` para decimal (base 10)
    - `OCT` para octal (base 8)
    - `HEX` para hexadecimal (base 16).

2.11 2.7 Tira de LEDs RGB

En este proyecto, exploraremos el fascinante mundo de controlar tiras de LEDs WS2812 y daremos vida a una vibrante exhibición de colores. Con la capacidad de controlar individualmente cada LED en la tira, podemos crear efectos de iluminación cautivadores que deslumbrarán los sentidos.

Además, hemos incluido una extensión emocionante a este proyecto, donde exploraremos el reino de la aleatoriedad. Introduciendo colores aleatorios y implementando un efecto de luz fluida, podemos crear una experiencia visual hipnotizante que cautiva y encanta.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

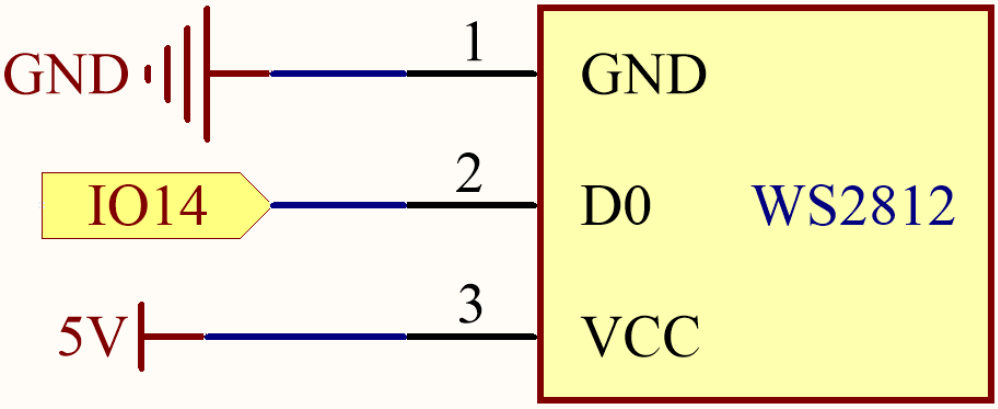
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Tira de 8 LEDs RGB WS2812	

Esquema



Pines Disponibles

Aquí está la lista de pines disponibles en la placa ESP32 para este proyecto.

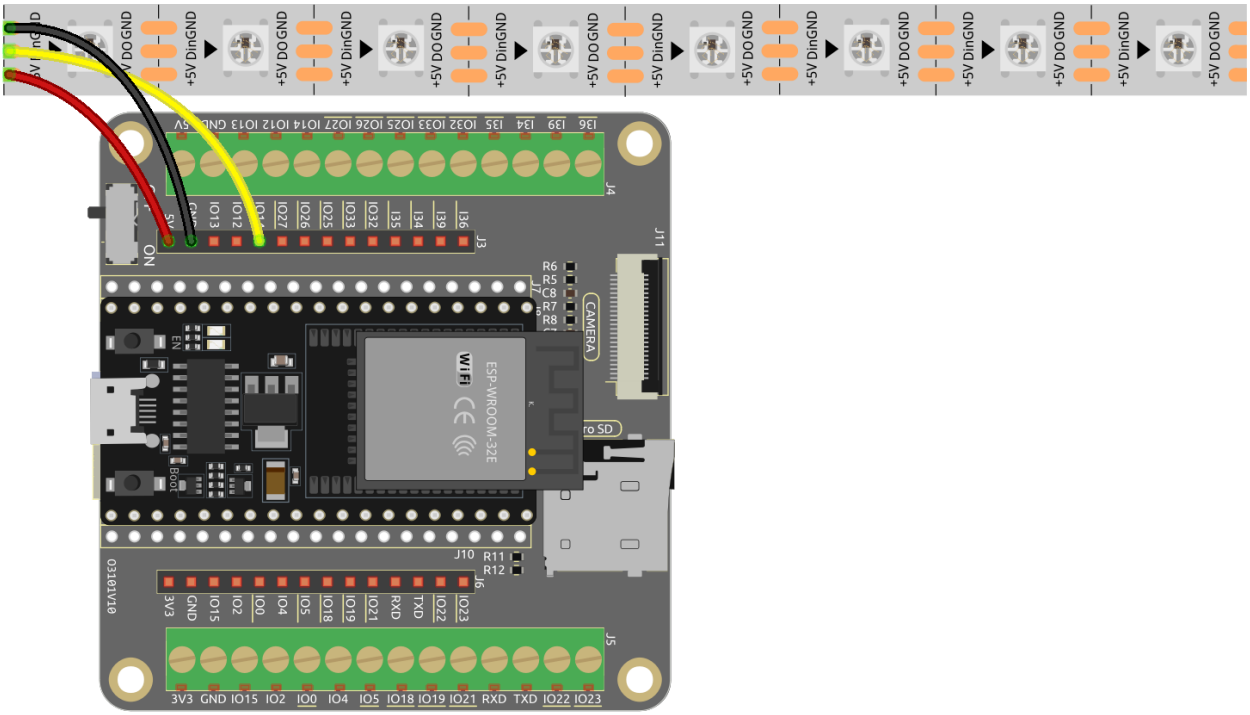
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Nota: IO33 no está disponible para este proyecto.

La tira LED WS2812 es un tipo de tira LED que requiere una señal de modulación de ancho de pulso (PWM) precisa. La señal PWM tiene requisitos precisos tanto en tiempo como en voltaje. Por ejemplo, un bit «0» para el WS2812 corresponde a un pulso de nivel alto de aproximadamente 0.4 microsegundos, mientras que un bit «1» corresponde a un pulso de nivel alto de aproximadamente 0.8 microsegundos. Esto significa que la tira necesita recibir cambios de voltaje de alta frecuencia.

Sin embargo, con una resistencia de pull-up de 4.7K y un condensador de pull-down de 100nf en IO33, se crea un filtro de paso bajo simple. Este tipo de circuito «suaviza» las señales de alta frecuencia, porque el condensador necesita algo de tiempo para cargarse y descargarse cuando recibe cambios de voltaje. Por lo tanto, si la señal cambia demasiado rápido (es decir, es de alta frecuencia), el condensador no podrá seguir el ritmo. Esto resulta en que la señal de salida se vuelva borrosa y no reconocible para la tira.

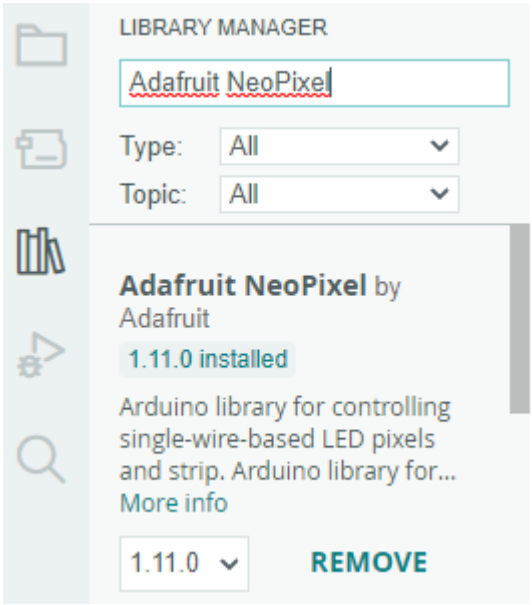
Cableado



Código

Nota:

- Puedes abrir el archivo `2.7_rgb_strip.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\2.7_rgb_strip`. O copiar este código en **Arduino IDE**.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- La biblioteca `Adafruit NeoPixel` se utiliza aquí, puedes instalarla desde el **Administrador de Bibliotecas**.



Cuando el código se haya subido con éxito, los LEDs en la tira se encenderán secuencialmente con un color amarillo y luego se apagarán, creando un simple efecto de persecución.

¿Cómo funciona?

1. Incluye la biblioteca Adafruit NeoPixel: Esta línea importa la biblioteca Adafruit NeoPixel para que el boceto pueda usar sus funciones y clases para controlar la tira de LED.

   ```
   #include <Adafruit_NeoPixel.h> // Include the Adafruit NeoPixel library
   ```

2. Define constantes para la tira LED.

   ```
   #define LED_PIN 13 // NeoPixel LED strip
   #define NUM_LEDS 8 // Number of LEDs
   ```

3. Crea una instancia de la clase Adafruit_NeoPixel.

   ```
   // Create an instance of the Adafruit_NeoPixel class
    Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, LED_PIN, NEO_GRB +␣
   ↪NEO_KHZ800);
   ```

   Esta línea crea una instancia de la clase `Adafruit_NeoPixel` llamada `strip` y la configura con el número de LEDs, el pin conectado a la tira de LED y los parámetros de señal (orden de color GRB y tasa de datos de 800 kHz).

   - `Adafruit_NeoPixel (uint16_t n, int16_t p = 6, neoPixelType t = NEO_GRB + NEO_KHZ800)`

   Constructor de NeoPixel cuando la longitud, el pin y el tipo de píxel se conocen en tiempo de compilación. Devuelve un objeto Adafruit_NeoPixel. Llamar a la función `begin()` antes de usar.

   - `n`: Número de NeoPixels en la tira.
   - `p`: Número de pin de Arduino que conducirá los datos de NeoPixel.
   - `t`: Tipo de píxel - suma las constantes `NEO_*` definidas en `Adafruit_NeoPixel.h`, por ejemplo, `NEO_GRB+NEO_KHZ800` para NeoPixels que esperan un flujo de datos de 800 KHz (vs 400 KHz) con bytes de color expresados en orden verde, rojo, azul por píxel.

4. Inicializa la tira RGB WS2812 y establece el color inicial de la tira en negro (apagado).

   ```
   void setup() {
       strip.begin(); // Initialize the NeoPixel strip
       strip.show(); // Set initial color to black
   }
   ```

   - `void begin (void)`: Configura el pin de NeoPixel para salida.
   - `void show (void)`: Transmite datos de píxeles en RAM a NeoPixels.

5. En la función `loop()`, los LEDs en la tira se encenderán secuencialmente con un color amarillo y luego se apagarán, creando un simple efecto de persecución.

   ```
   void loop() {
   // Turn on LEDs one by one
   for (int i = 0; i < NUM_LEDS; i++) {
       strip.setPixelColor(i, 100, 45, 0); // Set the color of the i-th LED to␣
   ↪red
       strip.show(); // Update the LED strip with the new colors
   ```

(continué en la próxima página)

(proviene de la página anterior)

```
    delay(100); // Wait for 100 milliseconds
}

// Turn off LEDs one by one
for (int i = 0; i < NUM_LEDS; i++) {
    strip.setPixelColor(i, 0, 0, 0); // Set the color of the i-th LED to␣
↪black (turn it off)
    strip.show(); // Update the LED strip with the new colors
    delay(100); // Wait for 100 milliseconds
}
}
```

- `void setPixelColor (uint16_t n, uint8_t r, uint8_t g, uint8_t b)`

Establece el color de un píxel usando componentes rojos, verdes y azules separados. Si se usan píxeles RGBW, el blanco se establecerá en 0.

`n`: Índice del píxel, comenzando desde 0. `r`: Brillo rojo, 0 = mínimo (apagado), 255 = máximo. `g`: Brillo verde, 0 = mínimo (apagado), 255 = máximo. `b`: Brillo azul, 0 = mínimo (apagado), 255 = máximo.

3. Sonidos

2.12 3.1 Beep

Este es un proyecto simple para hacer que un zumbador activo emita un pitido rápidamente cuatro veces cada segundo.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

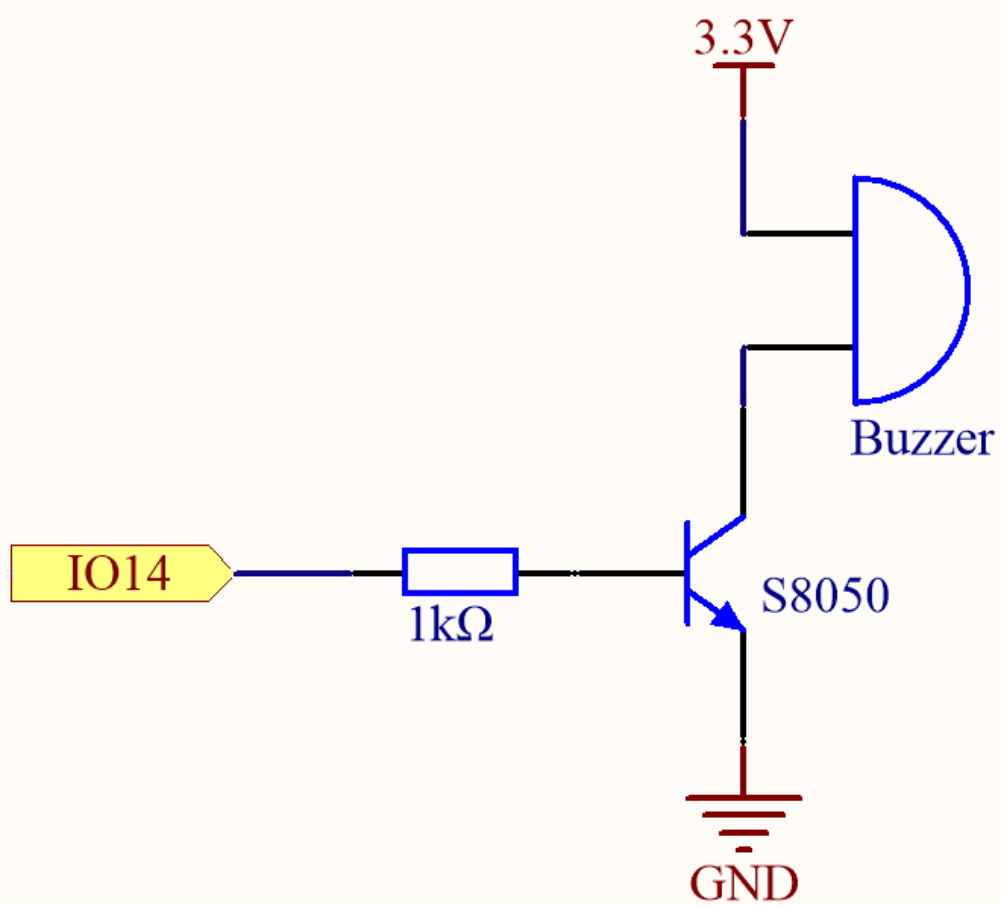
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Zumbador	-
Transistor	

Pines Disponibles

Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



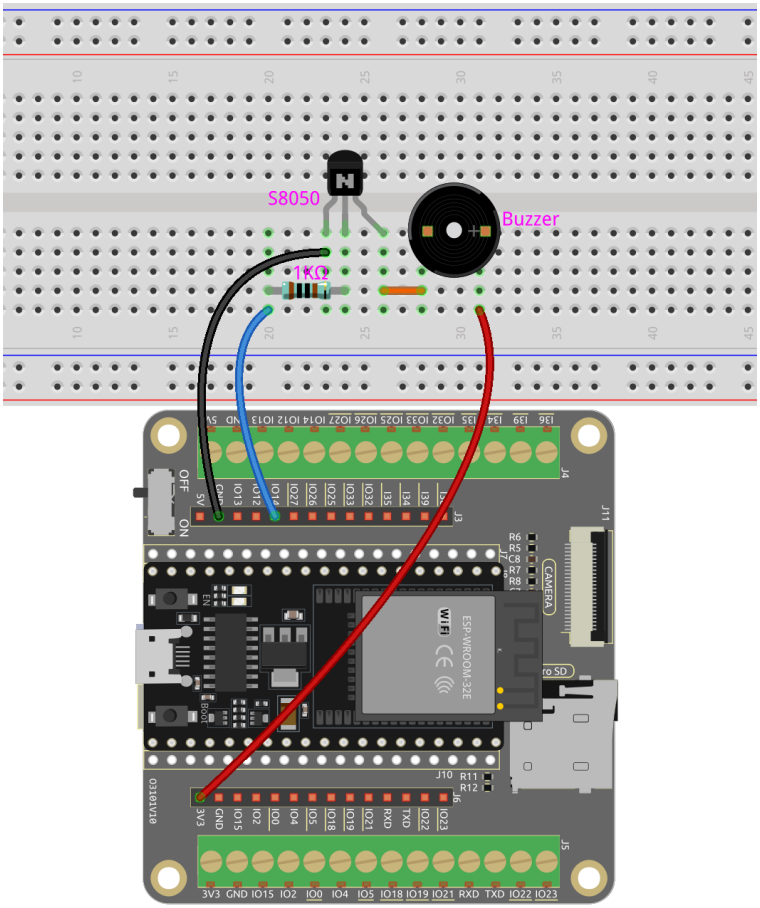
Cuando la salida IO14 es alta, después de la resistencia limitadora de corriente de 1K (para proteger el transistor), el S8050 (transistor NPN) conducirá, haciendo que el zumbador suene.

El rol de S8050 (transistor NPN) es amplificar la corriente y hacer que el sonido del zumbador sea más fuerte. De hecho, también puedes conectar el zumbador directamente a IO14, pero encontrarás que el sonido del zumbador es más bajo.

Cableado

Se incluyen dos tipos de zumbadores en el kit. Necesitamos usar el zumbador activo. Voltéalos, la parte trasera sellada (no el PCB expuesto) es la que queremos.

El zumbador necesita usar un transistor para funcionar, aquí usamos S8050 (Transistor NPN).



Código

Nota:

- Puedes abrir el archivo `3.1_beep.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\3.1_beep`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir el código con éxito, escucharás un pitido cada segundo.

2.13 3.2 Tono Personalizado

En el proyecto anterior, utilizamos un zumbador activo; esta vez, usaremos un zumbador pasivo.

Al igual que el zumbador activo, el zumbador pasivo también utiliza el fenómeno de inducción electromagnética para funcionar. La diferencia es que el zumbador pasivo no tiene una fuente oscilante, por lo que no emitirá un sonido si se usan señales de CC. Pero esto permite que el zumbador pasivo ajuste su propia frecuencia de oscilación y pueda emitir diferentes notas como «do, re, mi, fa, sol, la, si».

¡Hagamos que el zumbador pasivo emita una melodía!

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

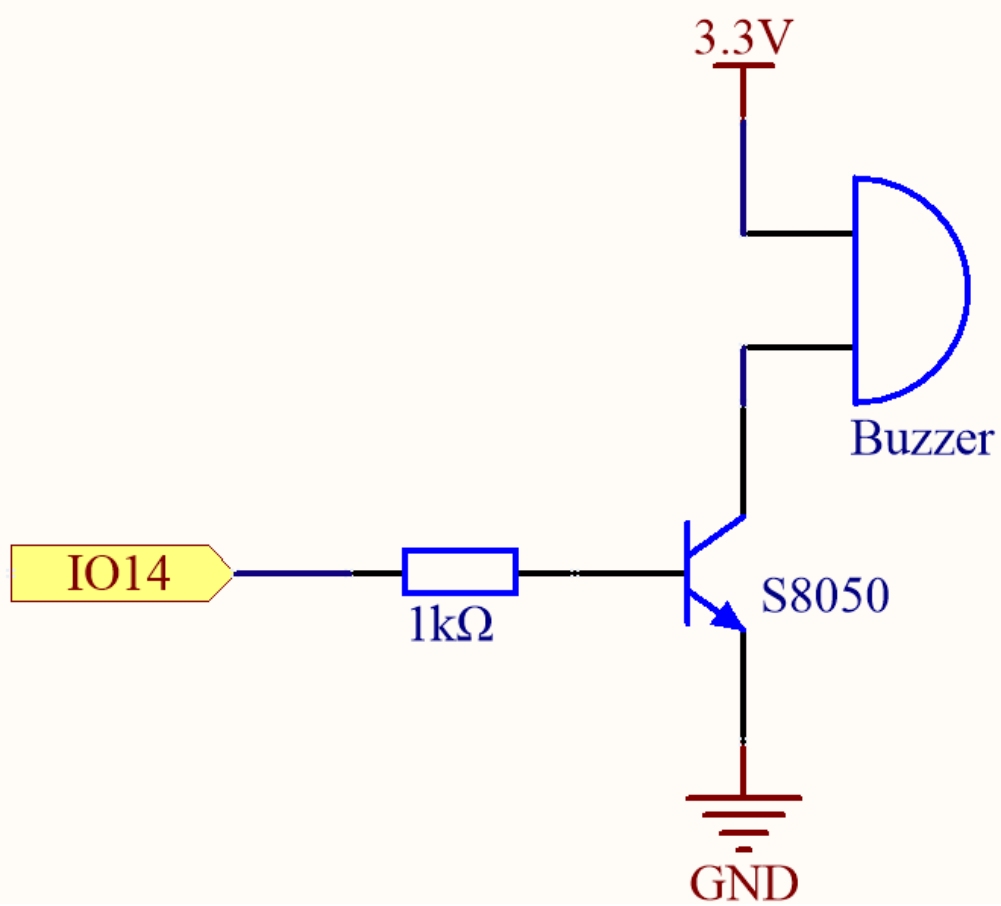
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Zumbador	-
Transistor	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



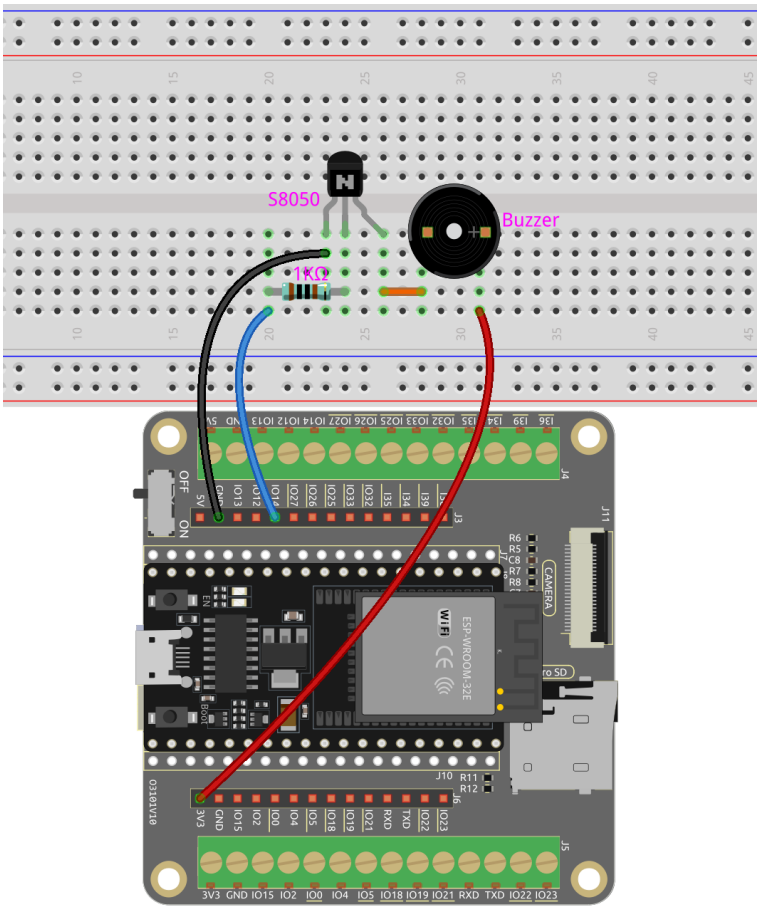
Cuando la salida IO14 es alta, después de la resistencia limitadora de corriente de 1K (para proteger el transistor), el S8050 (transistor NPN) conducirá, haciendo que el zumbador suene.

El rol del S8050 (transistor NPN) es amplificar la corriente y hacer que el sonido del zumbador sea más fuerte. De hecho, también puedes conectar el zumbador directamente a IO14, pero encontrarás que el sonido del zumbador es más bajo.

Cableado

Dos tipos de zumbadores están incluidos en el kit. Necesitamos usar el zumbador pasivo. Dale la vuelta, el PCB expuesto es el que queremos.

El zumbador necesita usar un transistor cuando trabaja, aquí usamos S8050 (Transistor NPN).



Código

Nota:

- Abre el archivo `3.2_custom_tone.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\3.2_custom_tone`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir exitosamente el código, escucharás el zumbador pasivo tocar una secuencia de 7 notas musicales.

Cómo funciona?

1. Define constantes para el pin del zumbador y la resolución PWM.

```
const int buzzerPin = 14; //buzzer pin
const int resolution = 8;
```

2. Define un arreglo que contiene las frecuencias de las 7 notas musicales en Hz.

```
int frequencies[] = {262, 294, 330, 349, 392, 440, 494};
```

3. Crea una función para tocar una frecuencia dada en el zumbador durante una duración especificada.

```
void playFrequency(int frequency, int duration) {
    ledcWriteTone(0, frequency); // Start the tone
    delay(duration); // Wait for the specified duration
    ledcWriteTone(0, 0); // Stop the buzzer
}
```

- `uint32_t ledcWriteTone(uint8_t chan, uint32_t freq);`: Esta función se usa para configurar el canal LEDC al tono PWM del 50 % en la frecuencia seleccionada.
  - `chan` selecciona el canal LEDC.
  - `freq` selecciona la frecuencia de la señal pwm.

Esta función retornará la `frecuencia` configurada para el canal. Si retorna `0`, ocurre un error y el canal ledc no fue configurado.

4. Configura el canal PWM y adjunta el pin del zumbador en la función `setup()`.

```
void setup() {
    ledcSetup(0, 2000, resolution); // Set up the PWM channel
    ledcAttachPin(buzzerPin, 0); // Attach the buzzer pin to the PWM channel
}
```

- `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Esta función se usa para configurar la frecuencia y resolución del canal LEDC. Retornará la `frecuencia` configurada para el canal LEDC. Si retorna `0`, ocurre un error y el canal ledc no fue configurado.
  - `channel` selecciona el canal LEDC a configurar.
  - `freq` selecciona la frecuencia del pwm.
  - `resolution_bits` selecciona la resolución para el canal ledc. El rango es de 1-14 bits (1-20 bits para ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Esta función se usa para adjuntar el pin al canal LEDC.
  - `pin` selecciona el pin GPIO.
  - `chan` selecciona el canal LEDC.

5. En la función `loop()`, toca la secuencia de 7 notas con una breve pausa entre cada nota y una pausa de 1 segundo antes de repetir la secuencia.

```
void loop() {
    for (int i = 0; i < 7; i++) {
        playFrequency(frequencies[i], 300); // Play each note for 300ms
        delay(50); // Add a brief pause between the notes
    }
    delay(1000); // Wait for 1 second before replaying the sequence
    }
```

4. Actuadores

2.14 4.1 Motor

En este proyecto atractivo, exploraremos cómo manejar un motor usando el L293D.

El L293D es un circuito integrado (CI) versátil comúnmente utilizado para el control de motores en proyectos de electrónica y robótica. Puede manejar dos motores en direcciones hacia adelante y hacia atrás, lo que lo hace una opción popular para aplicaciones que requieren un control preciso del motor.

Al finalizar este proyecto fascinante, habrás adquirido un conocimiento profundo de cómo las señales digitales y las señales PWM pueden ser utilizadas efectivamente para controlar motores. Este valioso conocimiento será una base sólida para tus futuros empeños en robótica y mecatrónica. ¡Prepárate para sumergirte en el emocionante mundo del control de motores con el L293D!

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

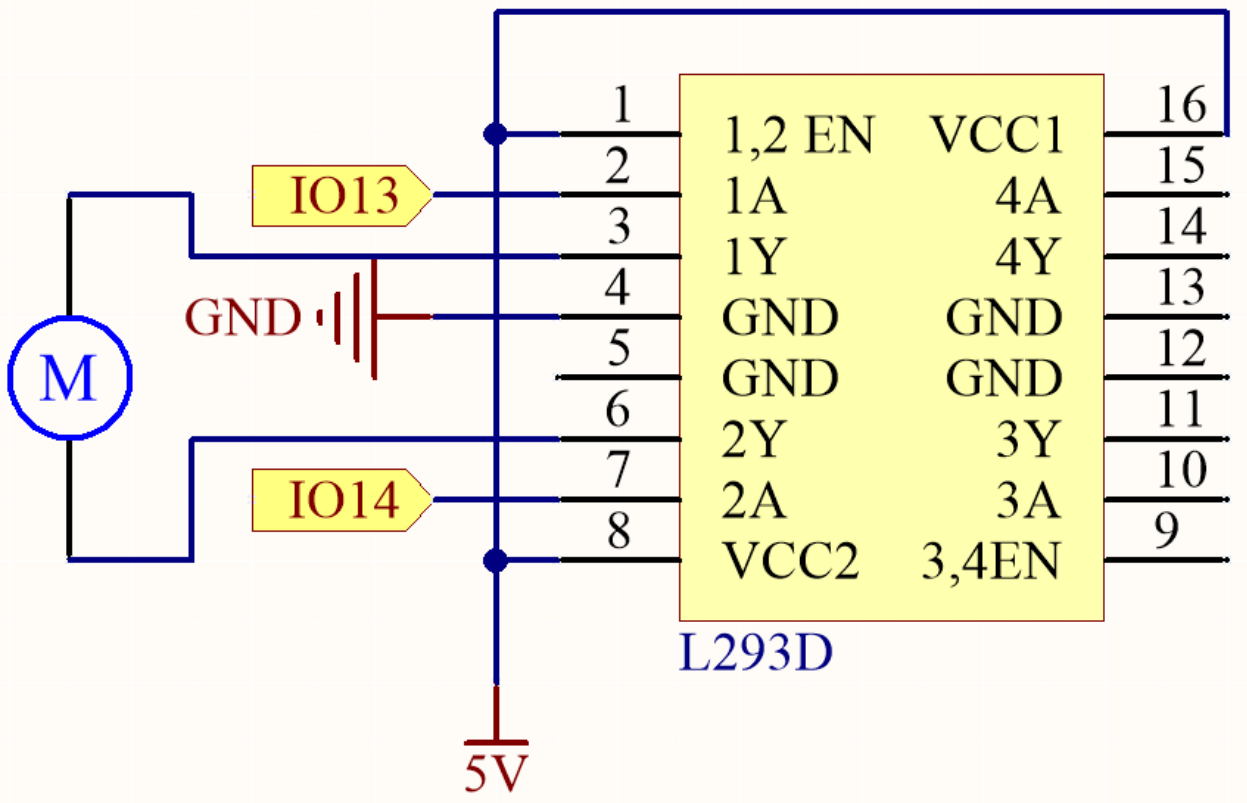
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Motor de Corriente Continua (DC)	
L293D	-

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

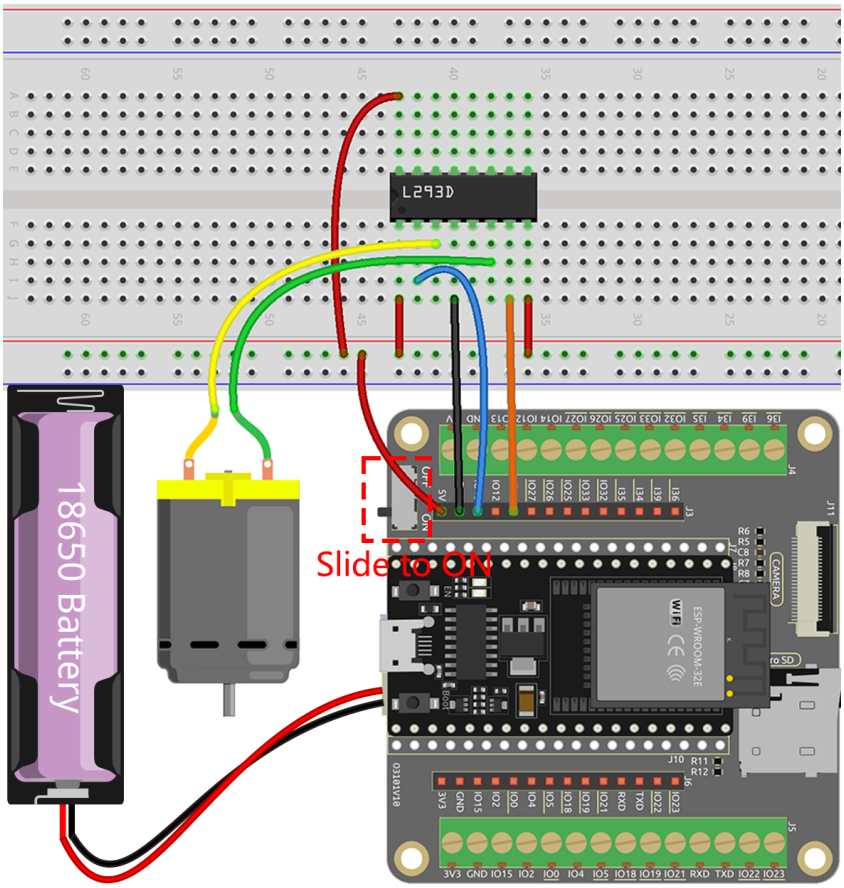
Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cableado

Nota: Dado que el motor requiere una corriente relativamente alta, es necesario primero insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de la batería.



Código

Nota:

- Abre el archivo `4.1_motor.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\4.1_motor`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Una vez que el código se haya cargado con éxito, observarás el motor girando en el sentido de las agujas del reloj durante un segundo, luego en sentido contrario durante un segundo, seguido por una pausa de dos segundos. Esta secuencia de acciones continuará en un bucle sin fin.

Aprende Más

Además de simplemente hacer girar el motor en el sentido de las agujas del reloj y en sentido contrario, también puedes controlar la velocidad de rotación del motor utilizando la modulación por ancho de pulso (PWM) en el pin de control, como se muestra a continuación.

Nota:

- Abre el archivo `4.1_motor_pwm.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\4.1_motor_pwm`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

El código anterior establece directamente los dos pines del motor a niveles de alto o bajo voltaje para controlar la rotación y la detención del motor.

Aquí usamos el (control de LED) periférico para generar señales PWM para controlar la velocidad del motor. A través de dos bucles `for`, el ciclo de trabajo del canal A se aumenta o disminuye de 0 a 255 mientras se mantiene el canal B en 0.

De esta manera, puedes observar cómo el motor aumenta gradualmente su velocidad a 255, luego disminuye a 0, repitiéndose infinitamente de esta manera.

Si deseas que el motor gire en la dirección opuesta, simplemente intercambia los valores de los canales A y B.

2.15 4.2 Bombeo

En este fascinante proyecto, exploraremos cómo controlar una bomba de agua utilizando el L293D.

En el ámbito del control de bombas de agua, las cosas son un poco más sencillas en comparación con el control de otros motores. La belleza de este proyecto radica en su simplicidad: no hay que preocuparse por la dirección de rotación. Nuestro objetivo principal es activar con éxito la bomba de agua y mantenerla funcionando.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

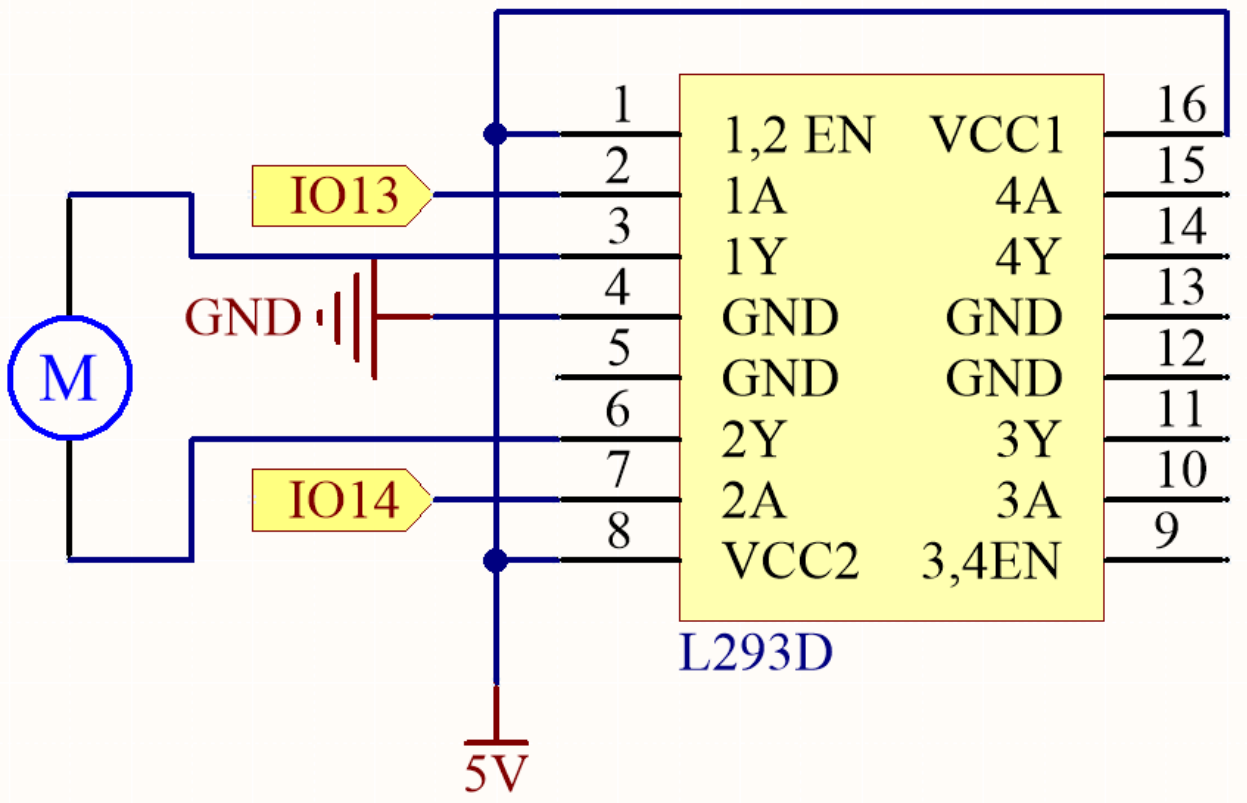
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Bomba Centrífuga	-
L293D	-

Pines Disponibles

Aquí está la lista de pines disponibles en la placa ESP32 para este proyecto.

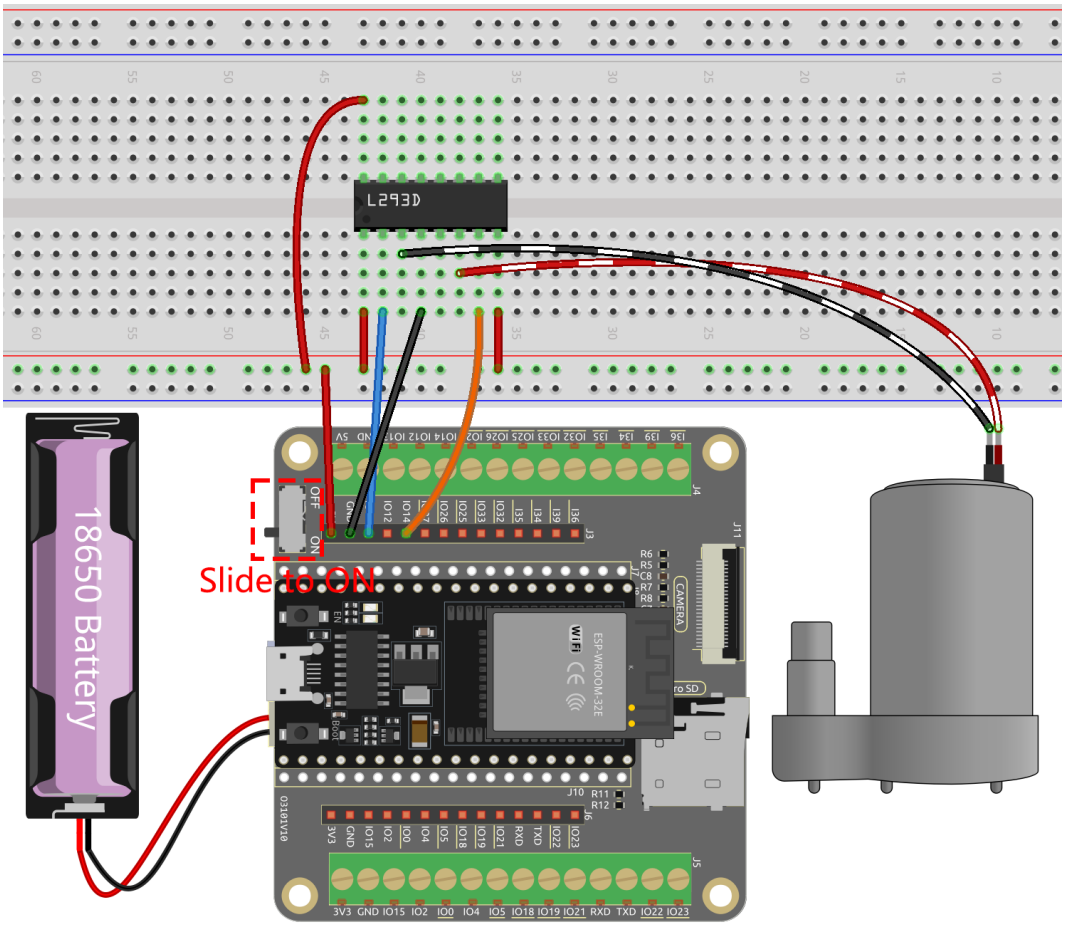
Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cableado

Nota: Se recomienda aquí insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de la batería.



Código

Nota:

- Puedes abrir el archivo `4.2_pump.ino` en la ruta `esp32-starter-kit-main\c\codes\4.2_pump`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Conecta el tubo a la bomba y colócala dentro del contenedor lleno de agua. Una vez que el código haya sido subido con éxito, observarás que el agua del contenedor se drena gradualmente. Durante este experimento, por favor asegúrate de que el circuito eléctrico se mantenga alejado del agua para evitar cortocircuitos!

2.16 4.3 Balanceo del Servo

Un Servo es un tipo de dispositivo basado en posición conocido por su habilidad para mantener ángulos específicos y proporcionar una rotación precisa. Esto lo hace altamente deseable para sistemas de control que demandan ajustes de ángulo consistentes. No es sorprendente que los Servos se hayan utilizado ampliamente en juguetes controlados remotamente de alta gama, desde modelos de aviones hasta réplicas de submarinos y robots controlados a distancia sofisticados.

En esta aventura fascinante, nos desafiaremos a manipular el Servo de una manera única: ¡haciéndolo oscilar! Este proyecto ofrece una oportunidad brillante para profundizar en la dinámica de los Servos, afilando tus habilidades en sistemas de control precisos y ofreciendo una comprensión más profunda de su operación.

¿Estás listo para hacer bailar al Servo al son de tus melodías? ¡Embarquémonos en este emocionante viaje!

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

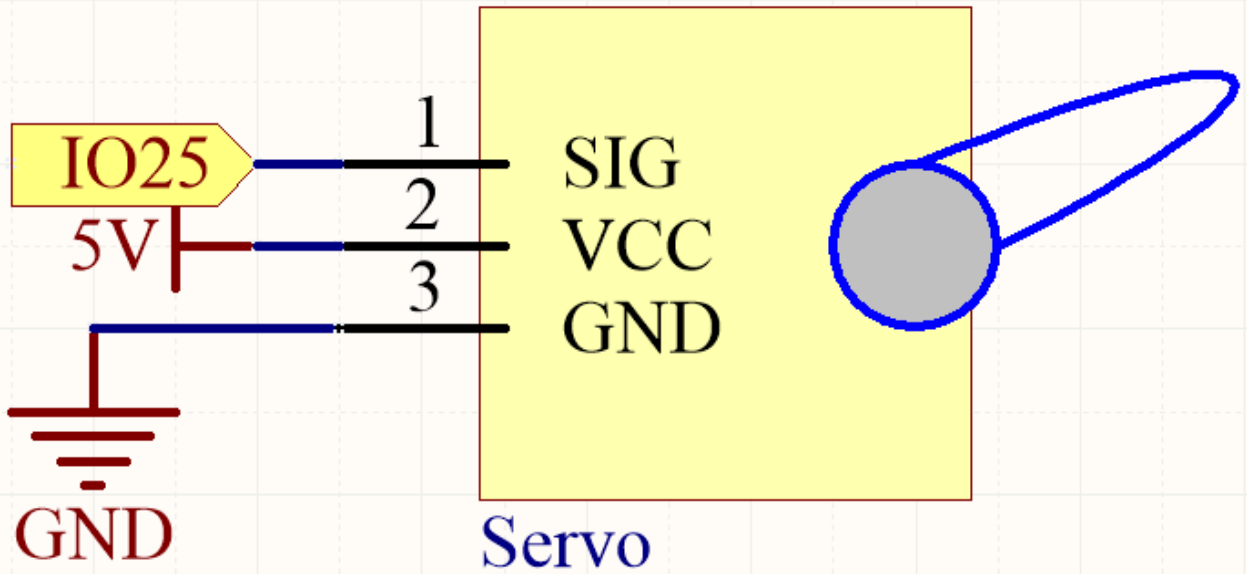
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Servo	

Pines Disponibles

Aquí tienes una lista de pines disponibles en la placa ESP32 para este proyecto.

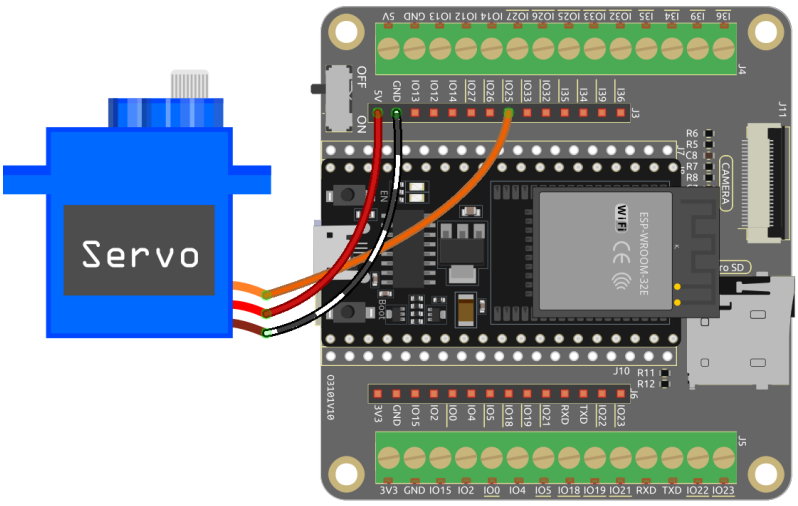
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Conexión

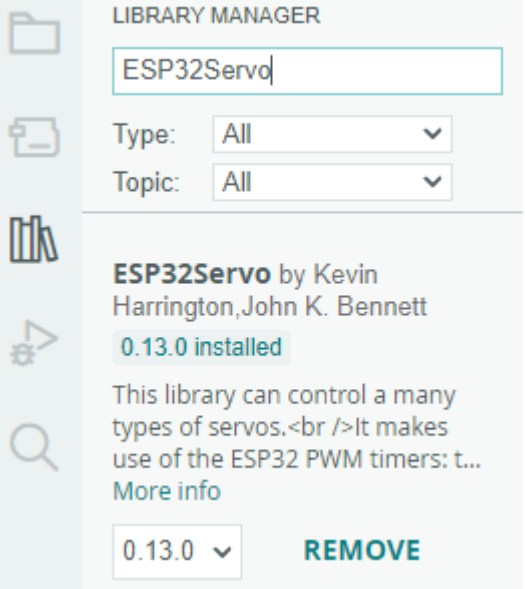
- El cable naranja es de señal y se conecta al IO25.
- El cable rojo es VCC y se conecta a 5V.
- El cable marrón es GND y se conecta a GND.



Código

Nota:

- Abre el archivo `4.3_servo.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\4.3_servo`. O copia este código en **Arduino IDE**.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- Se utiliza aquí la biblioteca `ESP32Servo`, puedes instalarla desde el **Gestor de Bibliotecas**.



Una vez que hayas subido el código, podrás ver el brazo del servo rotando en el rango de 0°~180°.

¿Cómo funciona?

1. Incluir la biblioteca : Esta línea importa la biblioteca ESP32Servo, necesaria para controlar el motor servo.

```
#include <ESP32Servo.h>
```

2. Definir el servo y el pin al que está conectado: Esta sección declara un objeto Servo (`myServo`) y un entero constante (`servoPin`) para representar el pin al que está conectado el motor servo (pin 25).

```
// Define the servo and the pin it is connected to
Servo myServo;
const int servoPin = 25;
```

3. Definir los anchos de pulso mínimo y máximo para el servo: Esta sección establece los anchos de pulso mínimo y máximo para el motor servo (0.5 ms y 2.5 ms, respectivamente).

```
// Define the minimum and maximum pulse widths for the servo
const int minPulseWidth = 500; // 0.5 ms
const int maxPulseWidth = 2500; // 2.5 ms
```

4. La función `setup` inicializa el motor servo adjuntándolo al pin especificado y configurando su rango de anchura de pulso. También establece la frecuencia PWM para el servo en el estándar de 50Hz.

```
void setup() {
    // Attach the servo to the specified pin and set its pulse width range
    myServo.attach(servoPin, minPulseWidth, maxPulseWidth);

    // Set the PWM frequency for the servo
    myServo.setPeriodHertz(50); // Standard 50Hz servo
}
```

   - `attach (int pin, int min, int max)`: Esta función conecta el motor servo al pin GPIO especificado y establece los anchos de pulso mínimo y máximo para el servo.

- `pin`: El número del pin GPIO al cual está conectado el servo.
- `min` y `max`: los anchos de pulso mínimo y máximo, respectivamente, en microsegundos. Estos valores definen el rango de movimiento del motor servo.

- `setPeriodHertz(int hertz)`: Esta función establece la frecuencia PWM para el motor servo en hertz.
  - `hertz`: La frecuencia PWM deseada en hertz. La frecuencia PWM predeterminada para servos es 50Hz, lo cual es adecuado para la mayoría de las aplicaciones.

5. La función `loop` es la parte principal del código que se ejecuta continuamente. Rota el motor servo de 0 a 180 grados y luego vuelve a 0 grados. Esto se logra mapeando el ángulo a la anchura de pulso correspondiente y actualizando el motor servo con el nuevo valor de anchura de pulso.

```
void loop() {
    // Rotate the servo from 0 to 180 degrees
    for (int angle = 0; angle <= 180; angle++) {
        int pulseWidth = map(angle, 0, 180, minPulseWidth, maxPulseWidth);
        myServo.writeMicroseconds(pulseWidth);
        delay(15);
    }

    // Rotate the servo from 180 to 0 degrees
    for (int angle = 180; angle >= 0; angle--) {
        int pulseWidth = map(angle, 0, 180, minPulseWidth, maxPulseWidth);
        myServo.writeMicroseconds(pulseWidth);
        delay(15);
    }
}
```

- `writeMicroseconds(int value)`: Esta función establece el ancho de pulso del motor servo en microsegundos.
  - `value`: El ancho de pulso deseado en microsegundos.

  La función `writeMicroseconds(int value)` toma un valor entero como argumento, representando el ancho de pulso deseado en microsegundos. Este valor típicamente debe estar dentro del rango especificado por los anchos de pulso mínimo y máximo (`minPulseWidth` y `maxPulseWidth`) definidos previamente en el código. Luego, la función establece el ancho de pulso para el motor servo, causando que se mueva a la posición correspondiente.

5. Sensores

2.17 5.1 Lectura del Valor del Botón

En este proyecto interactivo, nos aventuraremos en el ámbito de los controles de botones y la manipulación de LEDs.

El concepto es sencillo pero efectivo. Estaremos leyendo el estado de un botón. Cuando el botón está presionado, registra un nivel de voltaje alto, o "estado alto". Esta acción entonces hará que un LED se ilumine.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Botón	

Pines Disponibles

- **Pines Disponibles**

  Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Uso Condicional de Pines (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down incorporadas, por lo que no se requieren resistencias externas cuando **se usan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Pull-up con una resistencia de 47K por defecto el valor a alto.
IO27, IO26, IO33	Pull-up con una resistencia de 4.7K por defecto el valor a alto.
IO32	Pull-down con una resistencia de 1K por defecto el valor a bajo.

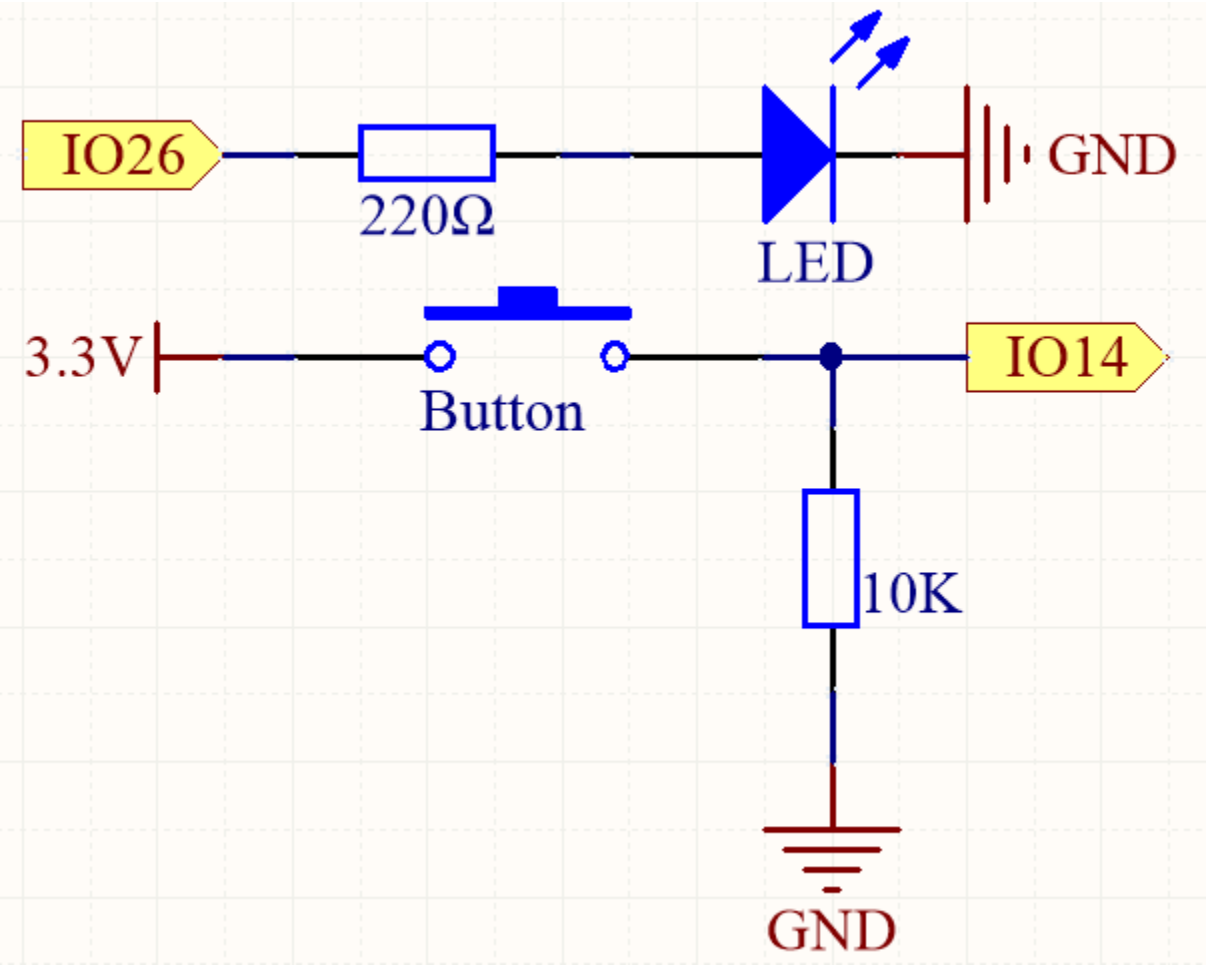
- **Pines de Strapping (Entrada)**

  Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reset por encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

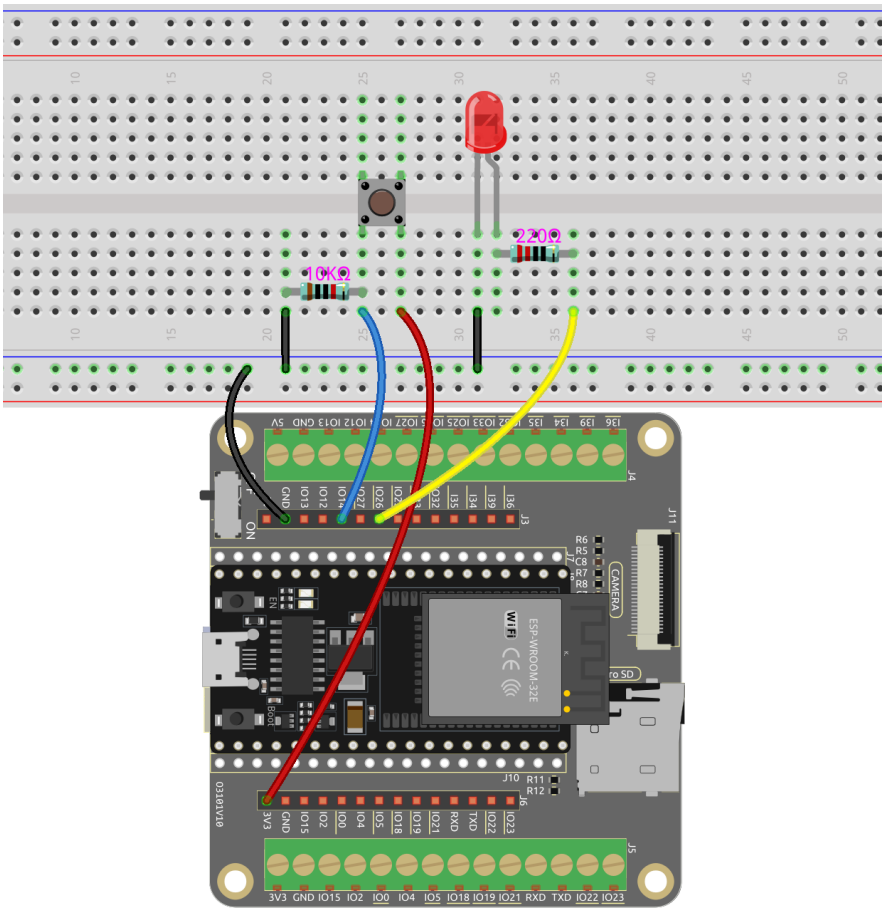
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, consulta la sección *Pines de Estrapeo*.

Esquemático



Para asegurar una funcionalidad adecuada, conecta un lado del pin del botón a 3.3V y el otro lado a IO14. Cuando el botón es presionado, IO14 se establece en alto, haciendo que el LED se ilumine. Cuando el botón es liberado, IO14 regresa a su estado suspendido, que puede ser alto o bajo. Para asegurar un nivel bajo estable cuando el botón no está presionado, IO14 debe conectarse a GND a través de una resistencia de pull-down de 10K.

Cableado



Nota: Un botón de cuatro pines está diseñado en forma de H. Cuando el botón no está presionado, los pines izquierdo y derecho están desconectados, y la corriente no puede fluir entre ellos. Sin embargo, cuando el botón es presionado, los pines izquierdo y derecho están conectados, creando un camino para que la corriente fluya.

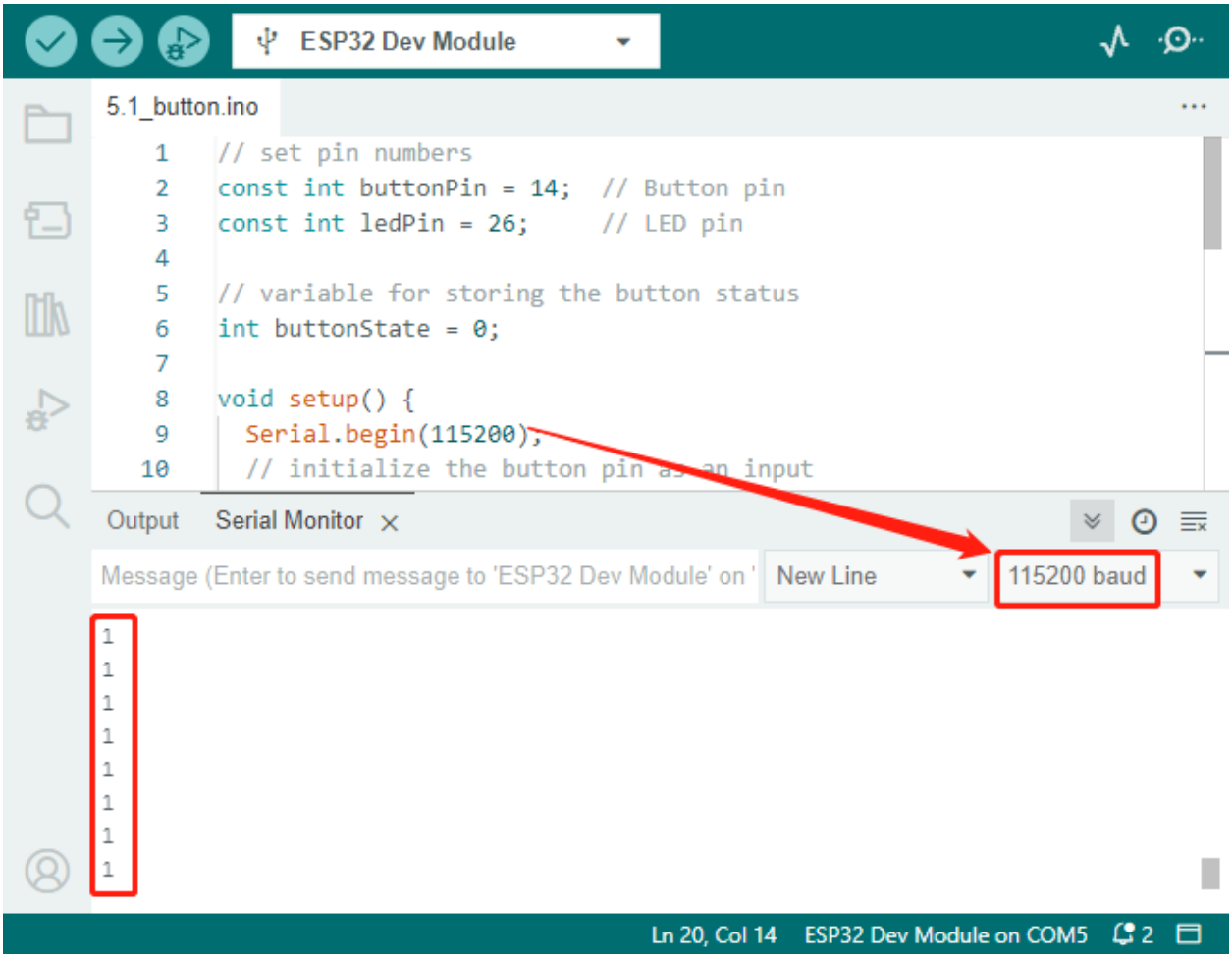
Código

Nota:

- Puedes abrir el archivo `5.1_button.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.1_button`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Una vez que el código se ha subido con éxito, el LED se ilumina cuando presionas el botón y se apaga cuando lo sueltas.

Al mismo tiempo, puedes abrir el Monitor Serial en la esquina superior derecha para observar el valor del botón, cuando el botón está presionado, se imprimirá «1», de lo contrario se imprimirá «0».



Cómo funciona

Los proyectos anteriores todos involucraron emitir señales, ya sea en forma de señales digitales o señales PWM.

Este proyecto involucra recibir señales de entrada de un componente externo hacia la placa ESP32. Puedes ver la señal de entrada a través del Monitor Serial en el IDE de Arduino.

1. En la función `setup()`, el pin del botón se inicializa como una `entrada` y el pin del LED se inicializa como una `salida`. La comunicación Serial también se inicia con una tasa de baudios de 115200.

```
void setup() {
    Serial.begin(115200);
    // initialize the button pin as an input
    pinMode(buttonPin, INPUT);
    // initialize the LED pin as an output
    pinMode(ledPin, OUTPUT);
}
```

   - `Serial.begin(velocidad)`: Establece la tasa de datos en bits por segundo (baudios) para la transmisión de datos serial.
     - `velocidad`: en bits por segundo (baudios). Tipos de datos permitidos: `long`.

2. En la función `loop()`, se lee el estado del botón y se almacena en la variable `buttonState`. El valor de `buttonState` se imprime en el Monitor Serial usando `Serial.println()`.

```
void loop() {
    // read the state of the button value
    buttonState = digitalRead(buttonPin);
    Serial.println(buttonState);
    delay(100);
    // if the button is pressed, the buttonState is HIGH
    if (buttonState == HIGH) {
        // turn LED on
        digitalWrite(ledPin, HIGH);

    } else {
        // turn LED off
        digitalWrite(ledPin, LOW);
    }
}
```

Si el botón está presionado y el `buttonState` es HIGH, el LED se enciende estableciendo el `ledPin` en `HIGH`. De lo contrario, apaga el LED.

- `int digitalRead(uint8_t pin);`: Para leer el estado de un pin dado configurado como INPUT, se usa la función digitalRead. Esta función devolverá el estado lógico del pin seleccionado como `HIGH` o `LOW`.
  - `pin` selecciona GPIO
- `Serial.println()`: Imprime datos al puerto serial como texto ASCII legible por humanos seguido de un carácter de retorno de carro (ASCII 13, o "r") y un carácter de nueva línea (ASCII 10, o "n").

2.18 5.2 ¡Inclínalo!

El interruptor de inclinación es un dispositivo de 2 pines simple pero efectivo que contiene una bola de metal en su centro. Cuando el interruptor está en posición vertical, los dos pines están eléctricamente conectados, permitiendo el paso de la corriente. Sin embargo, cuando el interruptor se inclina o se coloca en un cierto ángulo, la bola de metal se mueve y rompe la conexión eléctrica entre los pines.

En este proyecto, utilizaremos el interruptor de inclinación para controlar la iluminación de un LED. Posicionando el interruptor de manera que active la acción de inclinación, podemos encender y apagar el LED basándonos en la orientación del interruptor.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Interruptor de Inclinación	-

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Pines de Uso Condicional (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down incorporadas, por lo que no se requieren resistencias externas cuando **se usan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Con una resistencia de 47K haciendo pull-up por defecto el valor es alto.
IO27, IO26, IO33	Con una resistencia de 4.7K haciendo pull-up por defecto el valor es alto.
IO32	Con una resistencia de 1K haciendo pull-down por defecto el valor es bajo.

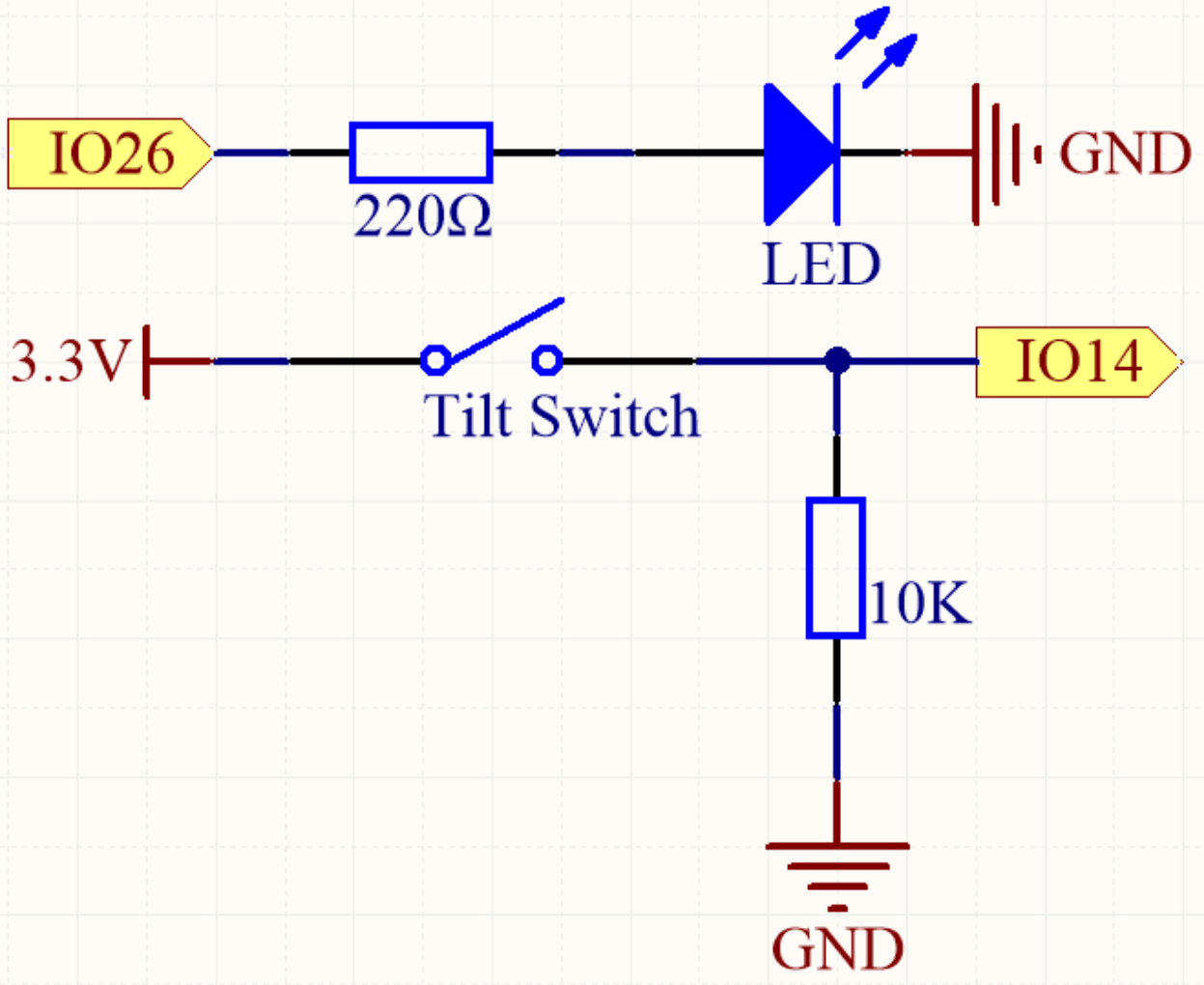
- **Pines de Arranque (Entrada)**

  Los pines de arranque son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reinicio por encendido).

Pines de Arranque	IO5, IO0, IO2, IO12, IO15

  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

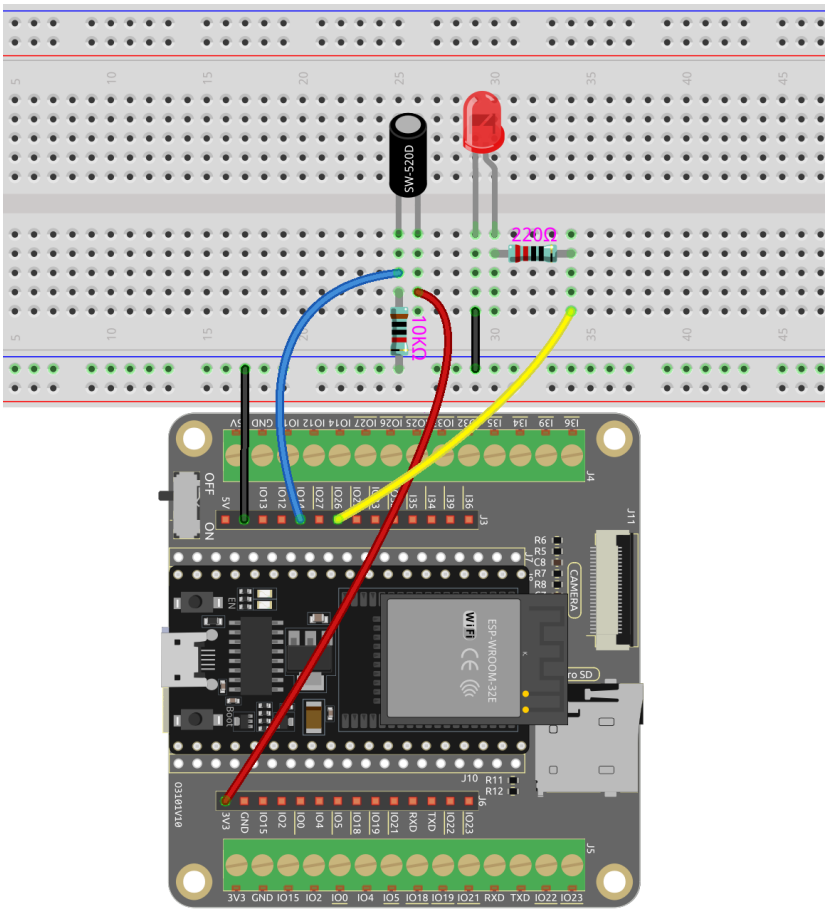
Esquemático



Cuando el interruptor de inclinación está en posición vertical, IO14 se establecerá en alto, resultando en el LED encendido. Por el contrario, cuando el interruptor de inclinación se inclina, IO14 se establecerá en bajo, causando que el LED se apague.

El propósito de la resistencia de 10K es mantener un estado bajo estable para IO14 cuando el interruptor de inclinación está en posición inclinada.

Cableado



Código

Nota:

- Puedes abrir el archivo `5.2_tilt_switch.ino` en la ruta de `esp32-starter-kit-main\c\codes\5.2_tilt_switch`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir el código con éxito, el LED se encenderá cuando el interruptor esté en posición vertical y se apagará cuando el interruptor esté inclinado.

2.19 5.3 Detectar el Obstáculo

Este módulo se instala comúnmente en coches y robots para juzgar la existencia de obstáculos adelante. También se utiliza ampliamente en dispositivos portátiles, grifos de agua y más.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Evitación de Obstáculos	

Pines Disponibles

- **Pines Disponibles**

  Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

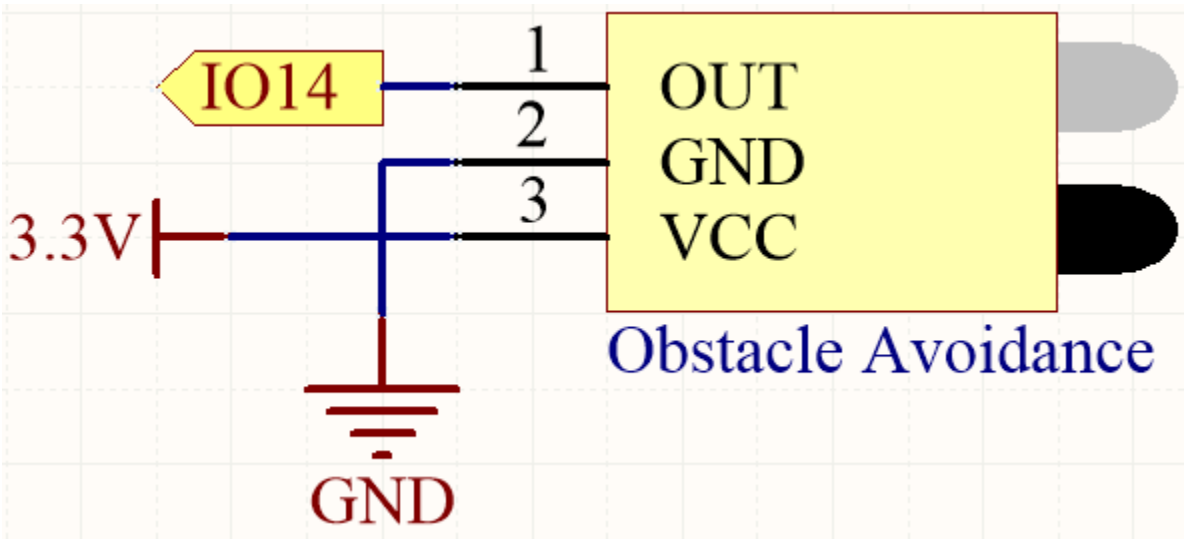
- **Pines de Strapping (Entrada)**

  Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reset por encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

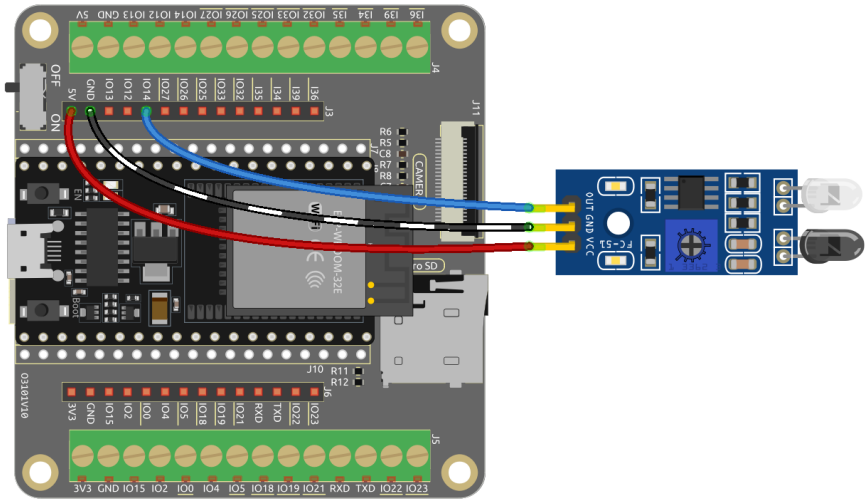
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor refiérete a la sección *Pines de Estrapeo*.

Esquemático



Cuando el módulo de evitación de obstáculos no detecta ningún obstáculo, IO14 devuelve un nivel alto. Sin embargo, cuando detecta un obstáculo, devuelve un nivel bajo. Puedes ajustar el potenciómetro azul para modificar la distancia de detección de este módulo.

Cableado



Código

Nota:

- Puedes abrir el archivo `5.3.detect_the_obstacle.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.3.detect_the_obstacle`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir el código con éxito, si el módulo de evitación de obstáculos IR detecta algo bloqueando frente a él, aparecerá «0» en el monitor serial, de lo contrario se mostrará «1».

2.20 5.4 Detectar la Línea

El módulo de seguimiento de línea se utiliza para detectar la presencia de áreas negras en el suelo, como líneas negras pegadas con cinta eléctrica.

Su emisor emite luz infrarroja adecuada hacia el suelo, la cual es absorbida relativamente y reflejada débilmente por superficies negras. Lo opuesto ocurre con las superficies blancas. Si se detecta luz reflejada, el suelo se indica actualmente como blanco. Si no se detecta, se indica como negro.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Seguimiento de Línea	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

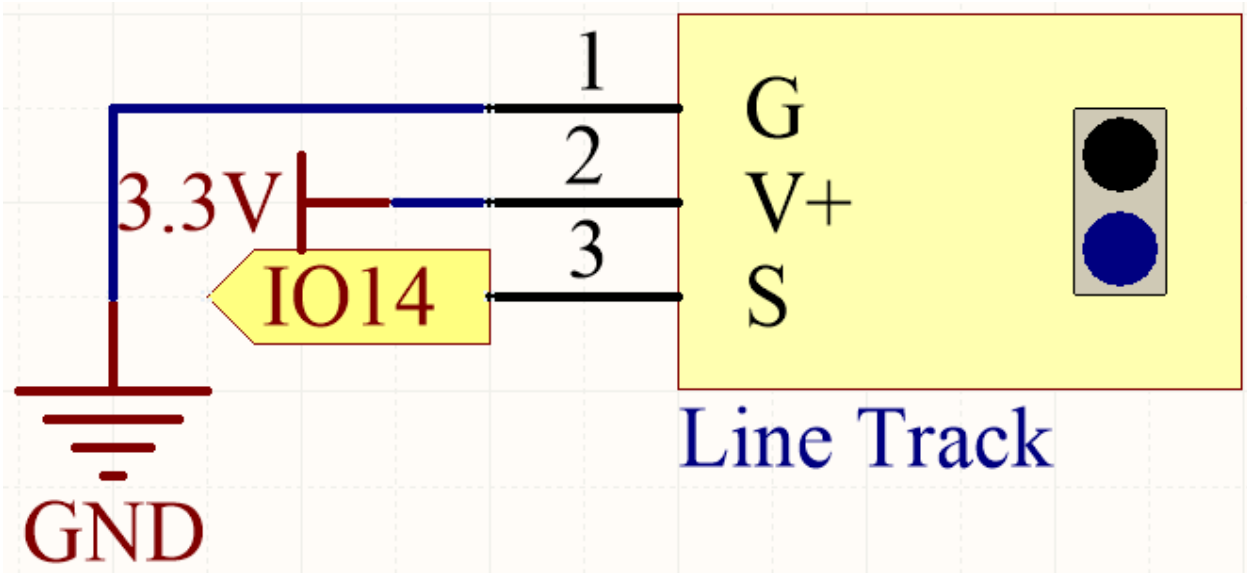
- **Pines de Strapping (Entrada)**

  Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reinicio de encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

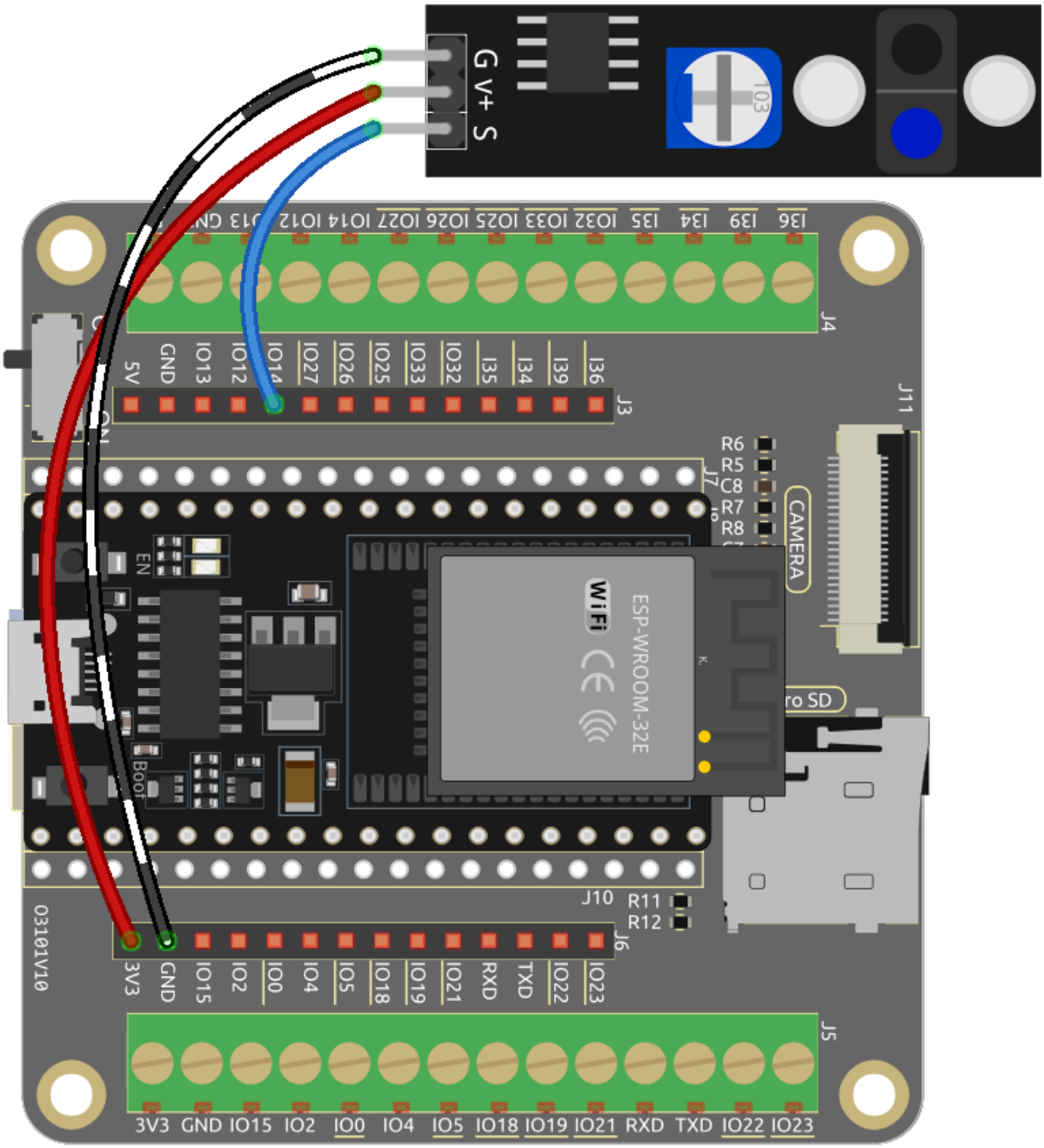
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

Esquemático



Cuando el módulo de seguimiento de línea detecta una línea negra, IO14 retorna un nivel alto. Por otro lado, cuando detecta una línea blanca, IO14 retorna un nivel bajo. Puedes ajustar el potenciómetro azul para modificar la sensibilidad de la detección de este módulo.

Cableado



Código

Nota:

- Puedes abrir el archivo `5.4_detect_the_line.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.4_detect_the_line`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Si el módulo de seguimiento de línea detecta una línea negra después de que el código se haya cargado con éxito, «Negro» se mostrará en el Monitor Serie. De lo contrario, se imprimirá «Blanco».

2.21 5.5 Detección de Movimiento Humano

El sensor infrarrojo pasivo (sensor PIR) es un sensor común que puede medir la luz infrarroja (IR) emitida por objetos en su campo de visión. En pocas palabras, recibirá la radiación infrarroja emitida por el cuerpo, detectando así el movimiento de personas y otros animales. Más específicamente, le indica a la placa de control principal que alguien ha entrado en su habitación.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Módulo Sensor de Movimiento PIR	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Nota: IO32 no puede ser utilizado **como pin de entrada** en este proyecto porque está internamente conectado a una resistencia de pull-down de 1K, lo que establece su valor predeterminado en 0.

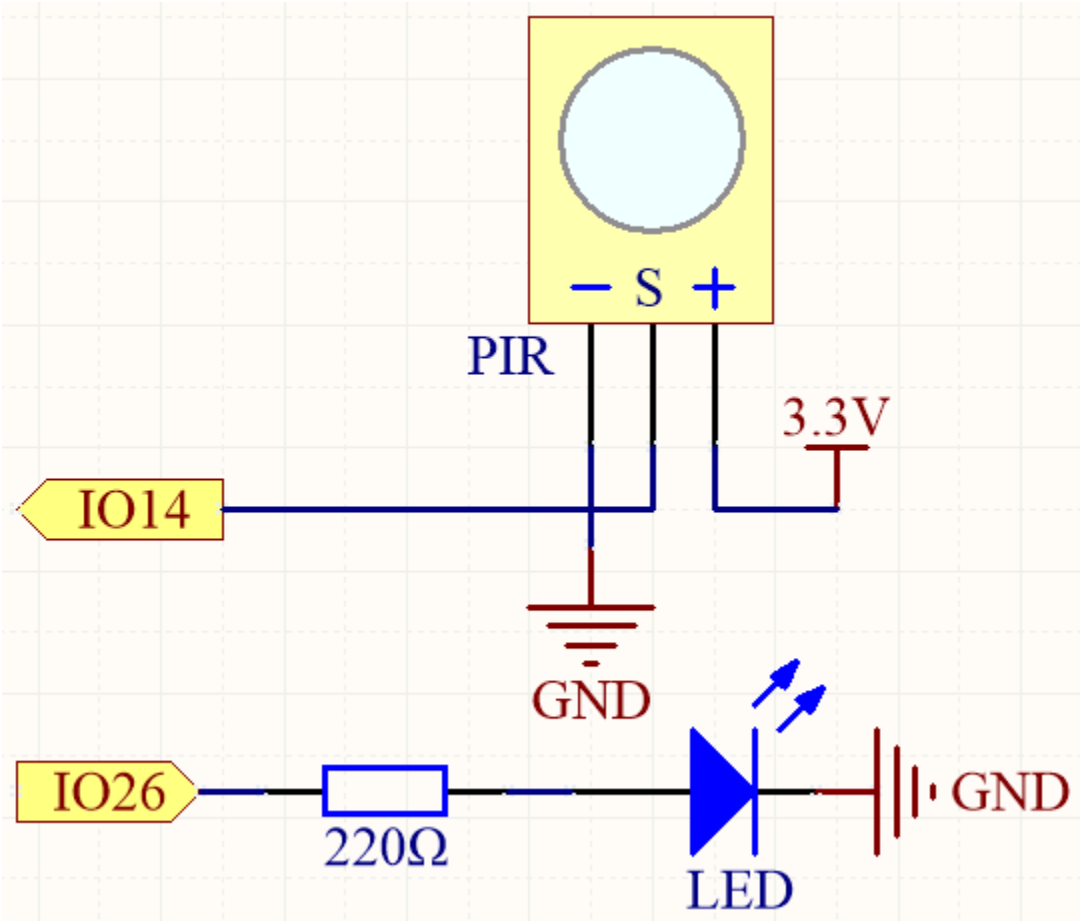
- **Pines de Arranque (Entrada)**

  Los pines de arranque son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, el reinicio por encendido).

Pines de Arranque	IO5, IO0, IO2, IO12, IO15

  Generalmente, **no se recomienda utilizarlos como pines de entrada**. Si deseas utilizar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

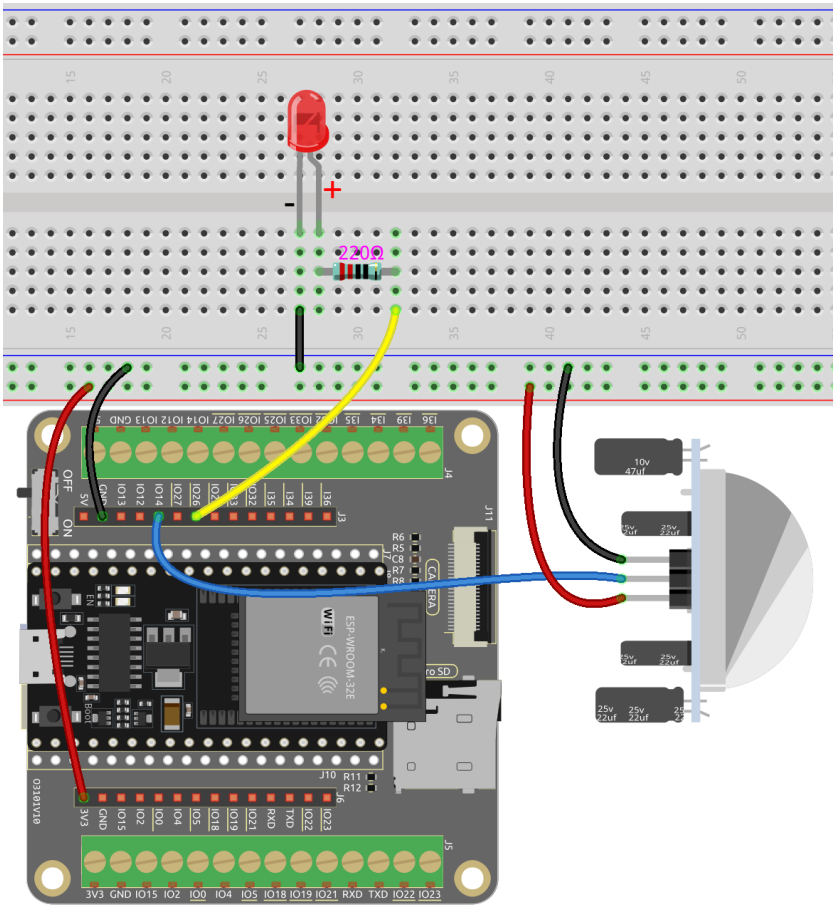
Esquemático



Cuando el módulo PIR detecta movimiento, IO14 se activará, y el LED se encenderá. De lo contrario, cuando no se detecta movimiento, IO14 estará en bajo, y el LED se apagará.

Nota: El módulo PIR tiene dos potenciómetros: uno ajusta la sensibilidad, el otro ajusta la distancia de detección. Para hacer que el módulo PIR funcione mejor, necesitas girar ambos en sentido antihorario hasta el final.

Cableado



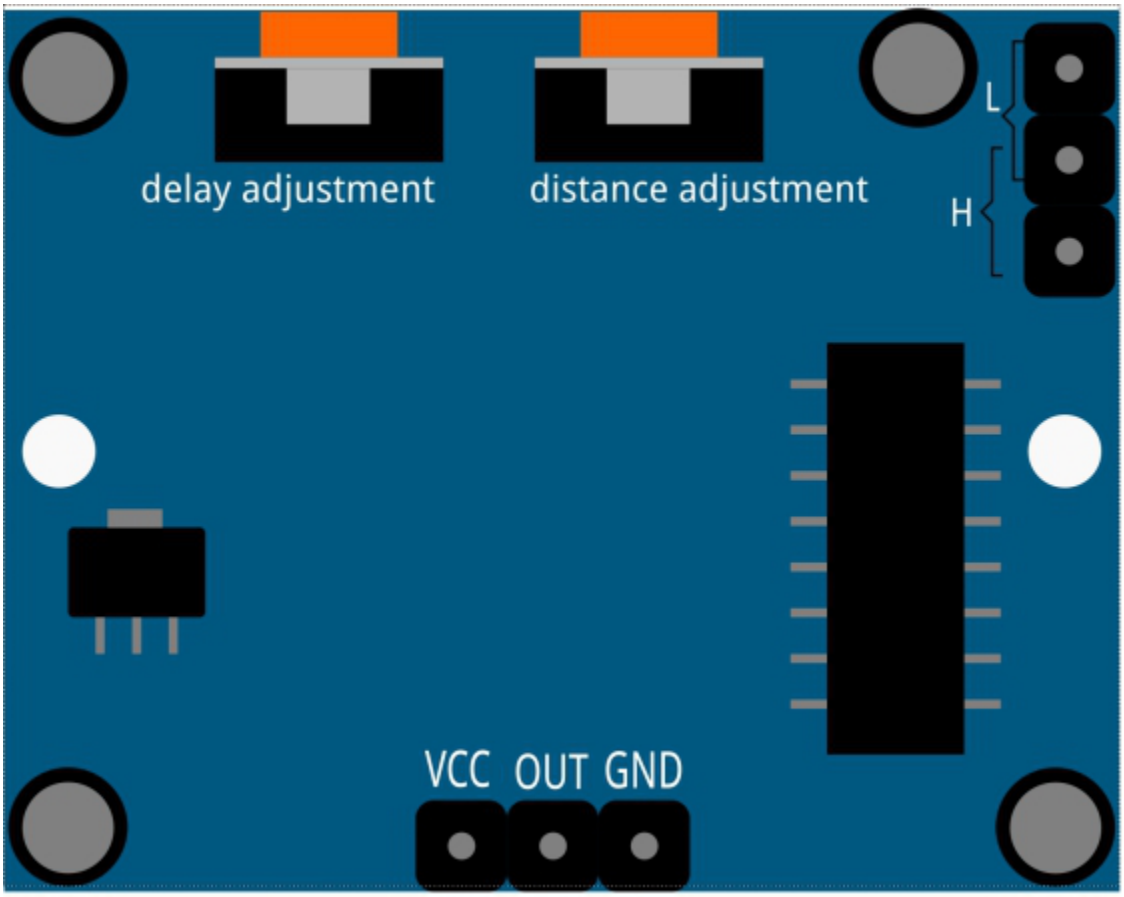
Código

Nota:

- Puedes abrir el archivo `5.5_pir.ino` en la ruta `esp32-starter-kit-main\c\codes\5.5_pir`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de haber subido el código con éxito, el LED se encenderá y luego se apagará cuando el módulo PIR detecte a alguien pasando.

Nota: El módulo PIR tiene dos potenciómetros: uno ajusta la sensibilidad, el otro ajusta la distancia de detección. Para hacer que el módulo PIR funcione mejor, necesitas girar ambos en sentido antihorario hasta el final.



2.22 5.6 Dos Tipos de Transistores

Este kit viene equipado con dos tipos de transistores, S8550 y S8050, siendo el primero PNP y el segundo NPN. Aunque se parecen mucho, necesitamos examinar cuidadosamente para ver sus etiquetas. Cuando una señal de nivel Alto pasa a través de un transistor NPN, este se activa. Pero uno PNP necesita una señal de nivel Bajo para activarse. Ambos tipos de transistores se usan frecuentemente para interruptores sin contacto, como en este experimento.

¡Vamos a usar un LED y un botón para entender cómo usar un transistor!

Componentes Necesarios

Para este proyecto necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Botón	
Transistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Pines de Uso Condicional (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down incorporadas, por lo que no se requieren resistencias externas cuando **se utilizan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Pull-up con una resistencia de 47K por defecto el valor es alto.
IO27, IO26, IO33	Pull-up con una resistencia de 4.7K por defecto el valor es alto.
IO32	Pull-down con una resistencia de 1K por defecto el valor es bajo.

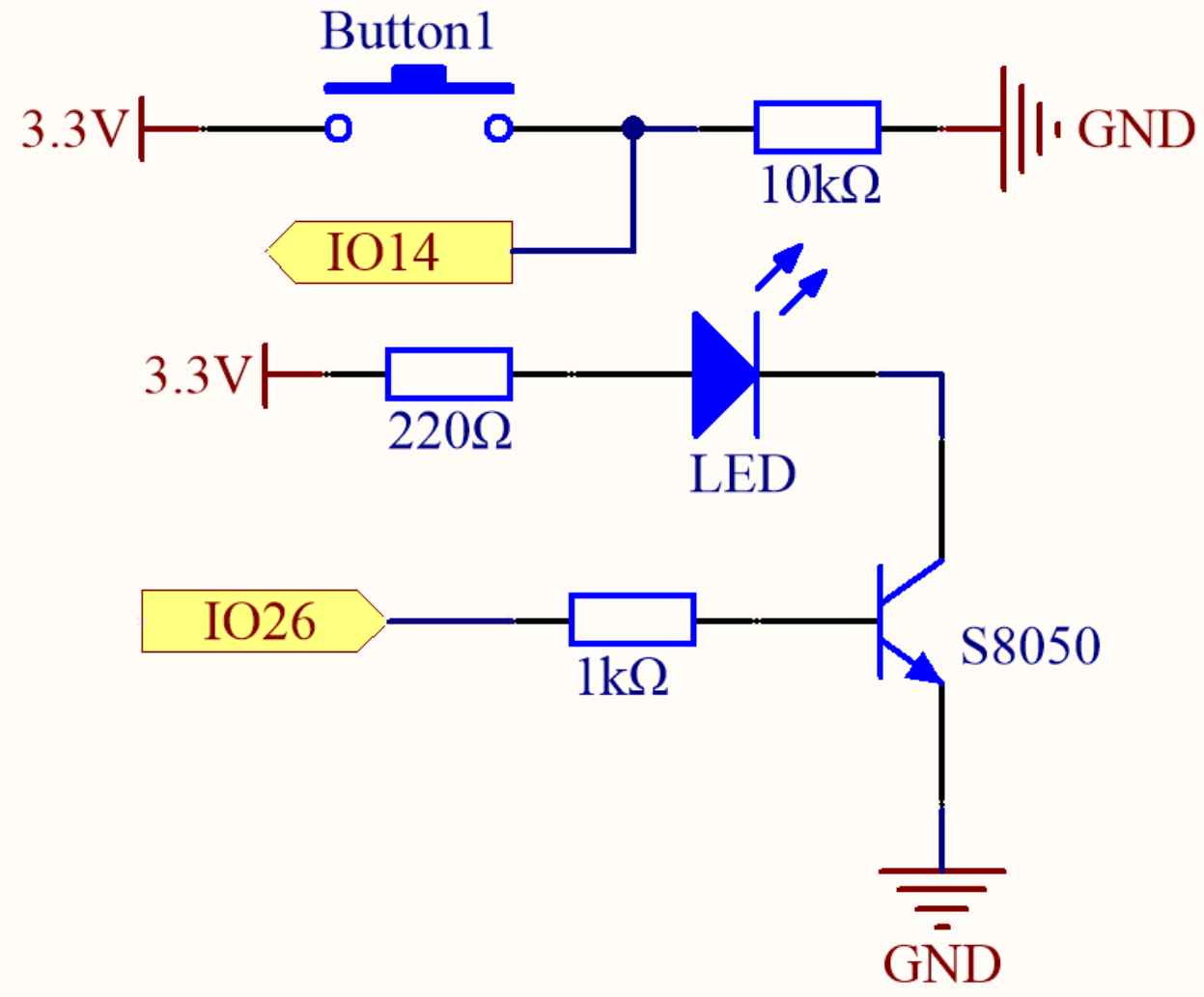
- **Pines de Configuración (Entrada)**

  Los pines de configuración son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reinicio por encendido).

Pines de Configuración	IO5, IO0, IO2, IO12, IO15

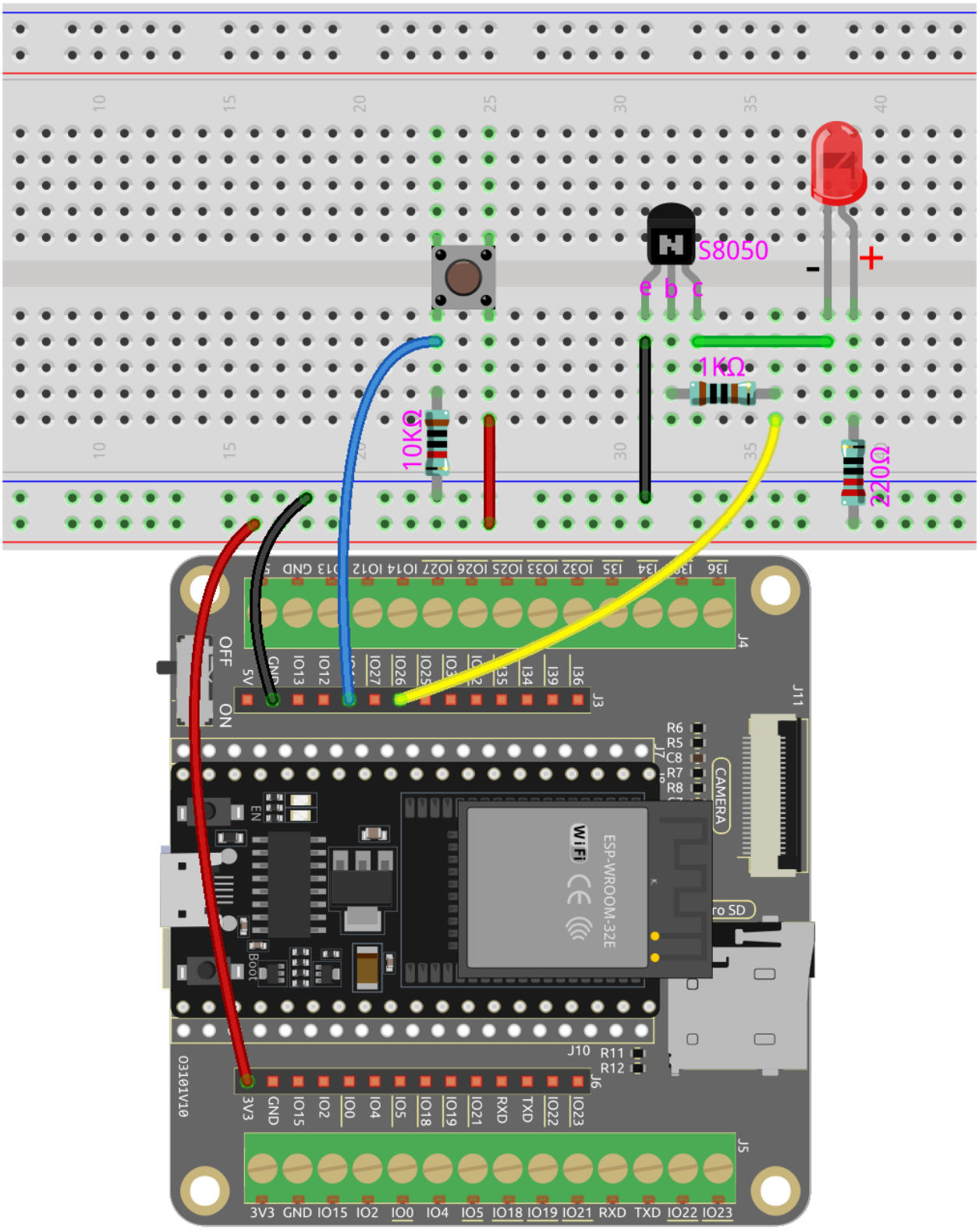
Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas utilizar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

Cómo Conectar el Transistor NPN (S8050)

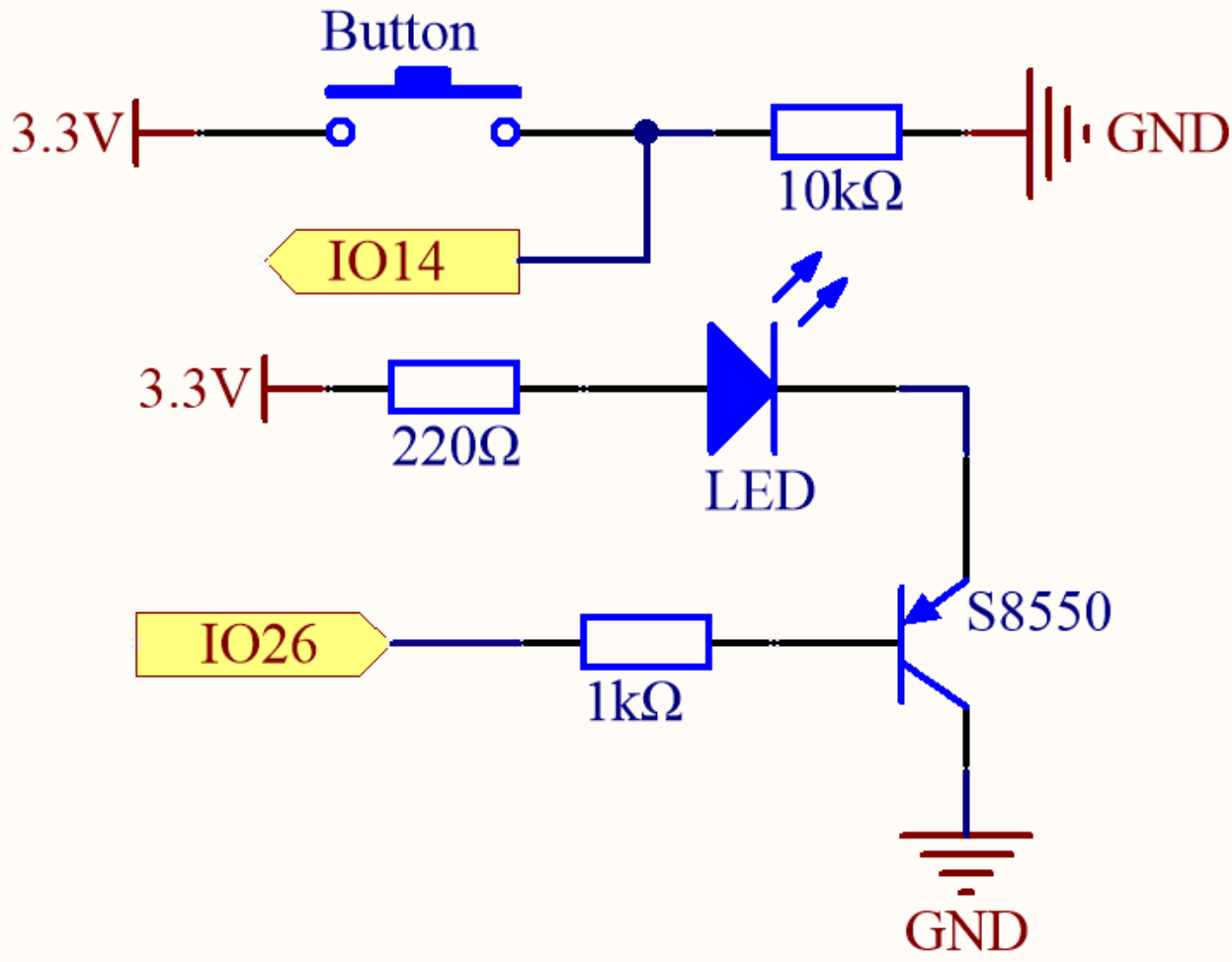


En este circuito, cuando se presiona el botón, IO14 está en alto.

Programando IO26 para que emita **alto**, tras una resistencia limitadora de corriente de 1k (para proteger el transistor), se permite que el S8050 (transistor NPN) conduzca, permitiendo así que el LED se ilumine.



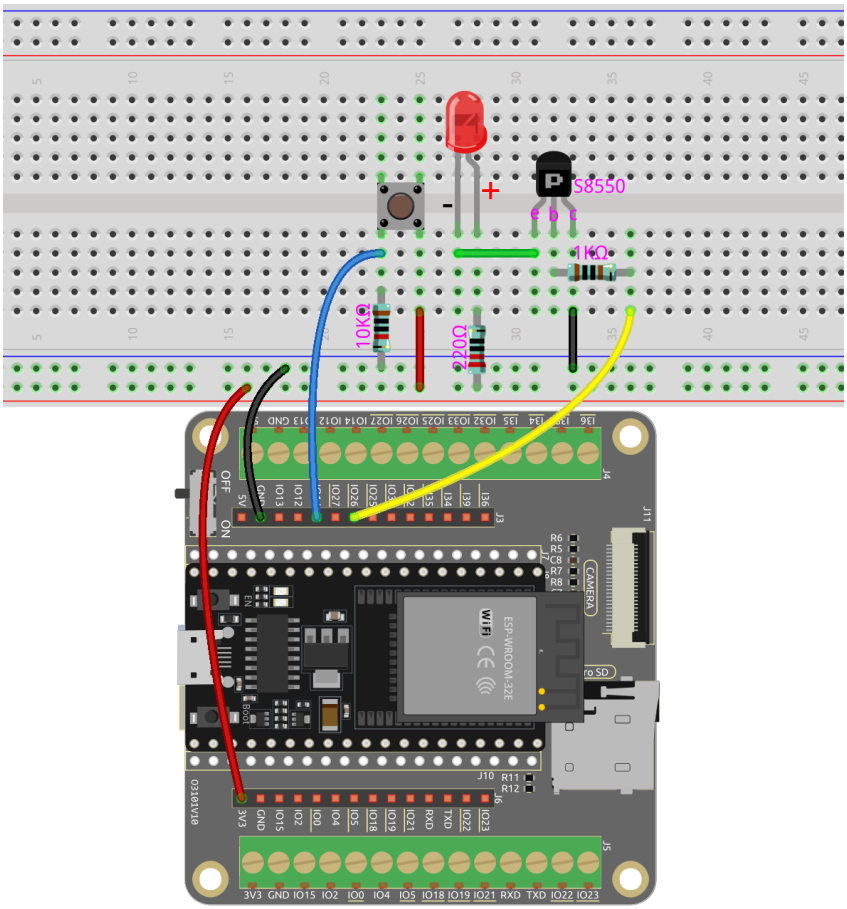
Cómo Conectar el Transistor PNP(S8550)



En este circuito, IO14 está en bajo por defecto y cambiará a alto cuando se presione el botón.

Programando IO26 para que emita **bajo**, tras una resistencia limitadora de corriente de 1k (para proteger el transistor), se permite que el S8550 (transistor PNP) conduzca, permitiendo así que el LED se ilumine.

La única diferencia que notarás entre este circuito y el anterior es que en el circuito anterior el cátodo del LED está conectado al **colector** del **S8050 (transistor NPN)**, mientras que en este está conectado al **emisor** del **S8550 (transistor PNP)**.



Código

Nota:

- Puedes abrir el archivo `5.6_transistor.ino` en la ruta `esp32-starter-kit-main\c\codes\5.6_transistor`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Se pueden controlar dos tipos de transistores utilizando el mismo código. Cuando presionamos el botón, el ESP32 enviará una señal de alto nivel al transistor; cuando lo soltemos, enviará una señal de bajo nivel.

- El circuito que utiliza el S8050 (transistor NPN) se iluminará al presionar el botón, indicando que se encuentra en un estado de conducción de alto nivel;
- El circuito que utiliza el S8550 (transistor PNP) se iluminará al soltar el botón, indicando que se encuentra en un estado de conducción de bajo nivel.

2.23 5.7 Siente la Luz

La fotorresistencia es un dispositivo comúnmente utilizado para entradas analógicas, similar a un potenciómetro. Su valor de resistencia cambia según la intensidad de la luz que recibe. Cuando se expone a una luz fuerte, la resistencia de la fotorresistencia disminuye, y a medida que la intensidad de la luz disminuye, la resistencia aumenta.

Al leer el valor de la fotorresistencia, podemos recopilar información sobre las condiciones de luz ambiental. Esta información puede ser utilizada para tareas como controlar el brillo de un LED, ajustar la sensibilidad de un sensor o implementar acciones dependientes de la luz en un proyecto.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Fotorresistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

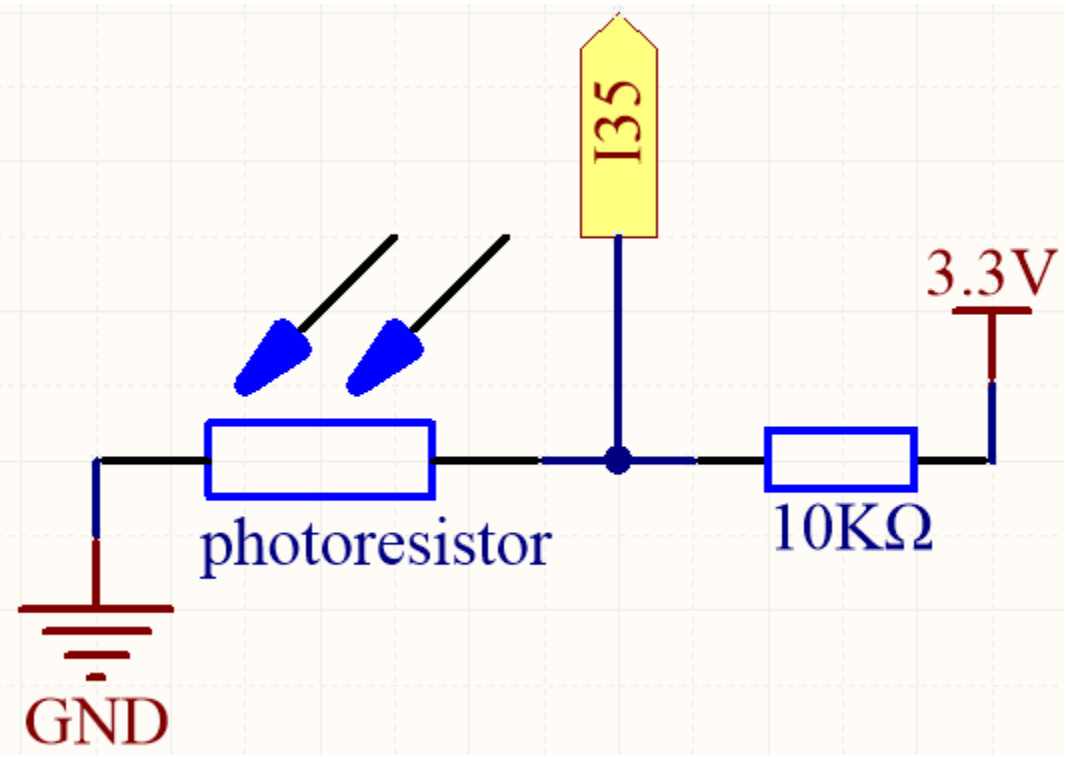
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Arranque**

  Los siguientes pines son pines de arranque, los cuales afectan el proceso de inicio del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 se ha iniciado con éxito, pueden ser utilizados como pines regulares.

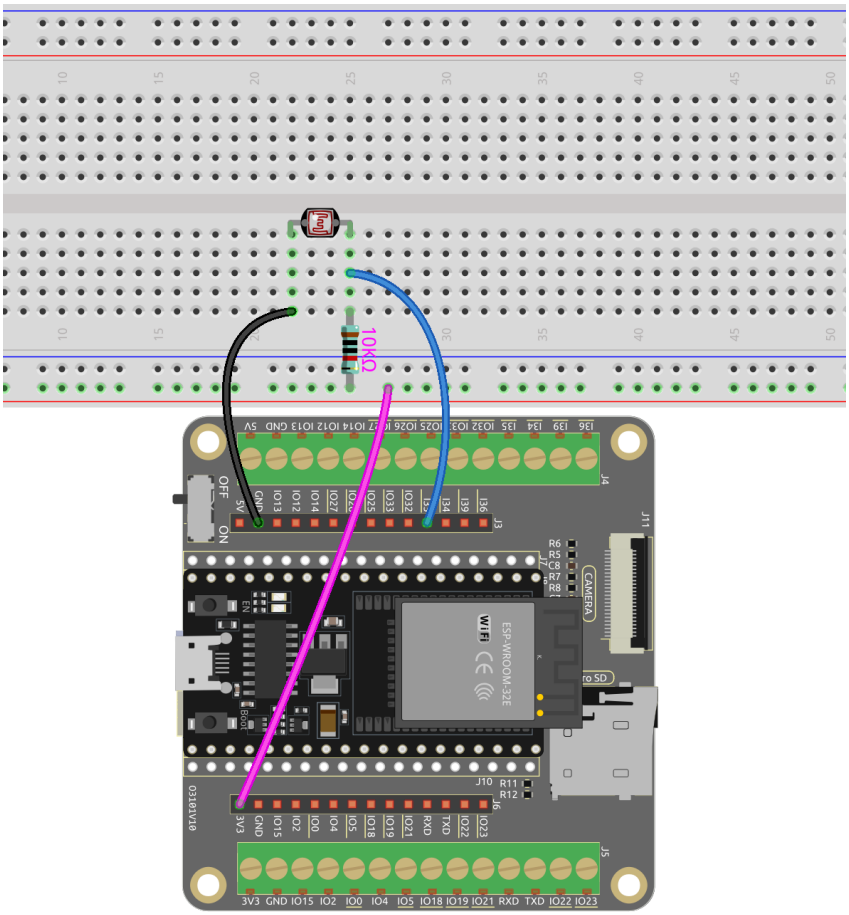
Pines de Arranque	IO0, IO12

Esquemático



A medida que aumenta la intensidad de la luz, la resistencia del resistor dependiente de la luz (LDR) disminuye, resultando en una disminución del valor leído en I35.

Cableado



Código

Nota:

- Abre el archivo `5.7_feel_the_light.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.7_feel_the_light`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir con éxito el código, el Monitor Serial imprime los valores de la fotorresistencia de 0 ~ 4095. Cuanto más fuerte es el brillo ambiental actual, mayor es el valor mostrado en el monitor serial.

Nota: Para el ESP32, la resolución está entre 9 y 12 y cambiará la resolución de hardware ADC. De lo contrario, el valor se desplazará.

El valor predeterminado es de 12 bits (rango de 0 a 4096) para todos los chips excepto ESP32S3 donde el predeterminado es de 13 bits (rango de 0 a 8192).

Puedes agregar `analogReadResolution(10);` a la función `setup()` para establecer una resolución diferente, como `20`.

2.24 5.8 Gira el Potenciómetro

Un potenciómetro es un dispositivo de tres terminales que se utiliza comúnmente para ajustar la resistencia en un circuito. Cuenta con un botón o una palanca deslizante que se puede utilizar para variar el valor de resistencia del potenciómetro. En este proyecto, lo utilizaremos para controlar el brillo de un LED, similar a una lámpara de escritorio en nuestra vida diaria. Al ajustar la posición del potenciómetro, podemos cambiar la resistencia en el circuito, regulando así la corriente que fluye a través del LED y ajustando su brillo en consecuencia. Esto nos permite crear una experiencia de iluminación personalizable y ajustable, similar a la de una lámpara de escritorio.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Potenciómetro	

Pines Disponibles

- **Pines Disponibles**

  Aquí está la lista de pines disponibles en la placa ESP32 para este proyecto.

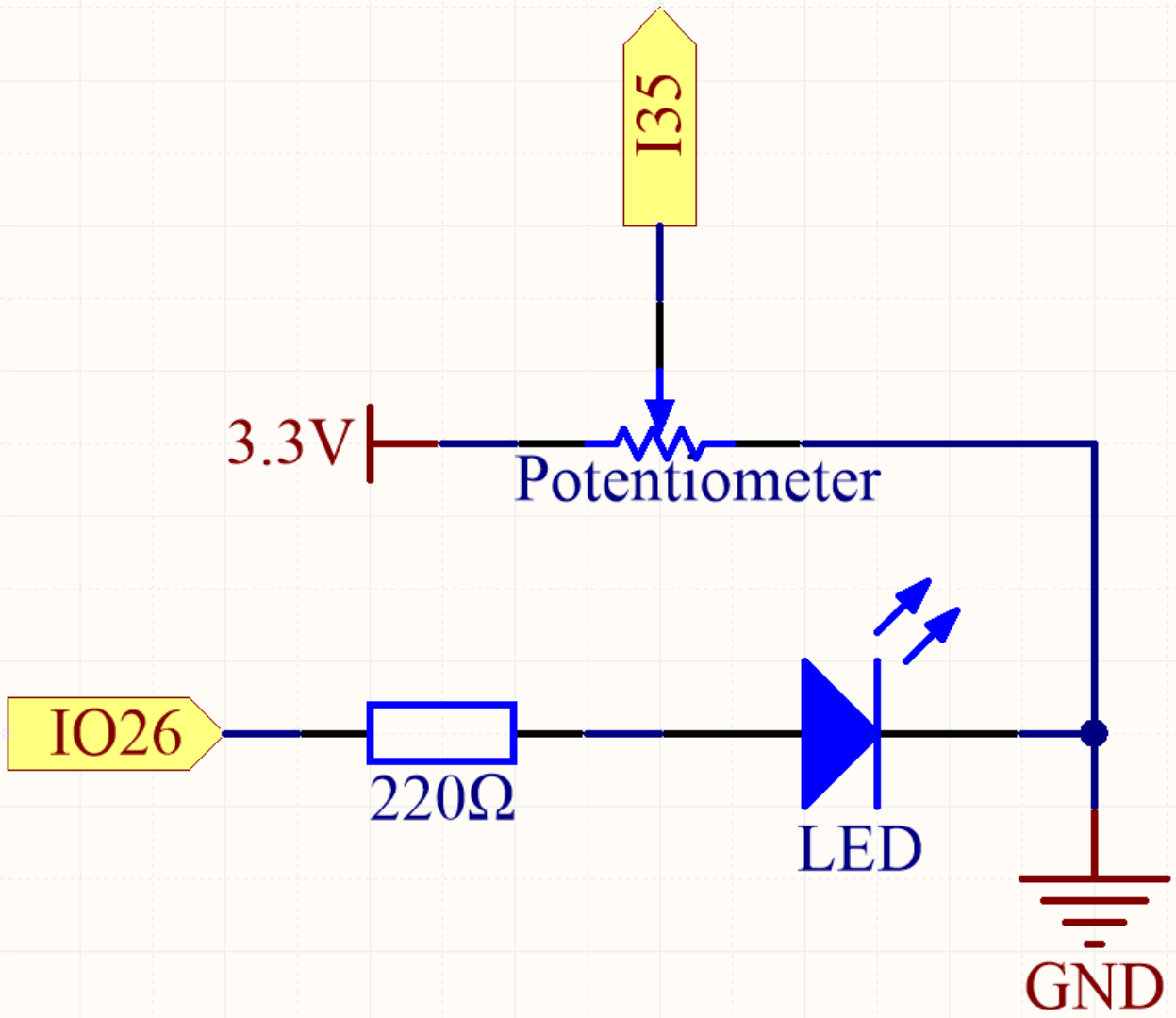
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Arranque**

  Los siguientes pines son pines de arranque, que afectan el proceso de inicio del ESP32 durante el encendido o el restablecimiento. Sin embargo, una vez que el ESP32 se ha iniciado correctamente, se pueden utilizar como pines regulares.

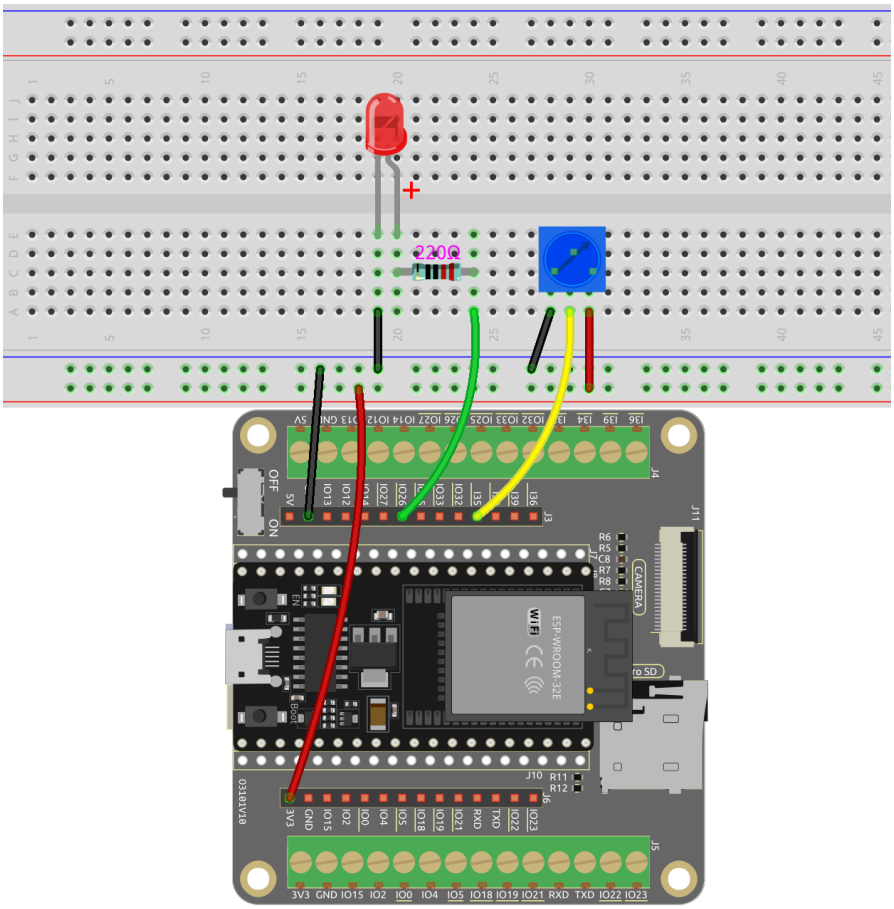
Pines de Arranque	IO0, IO12

Esquemático



Al girar el potenciómetro, el valor de I35 cambiará. Mediante programación, puedes usar el valor de I35 para controlar el brillo del LED. Por lo tanto, al girar el potenciómetro, el brillo del LED también cambiará en consecuencia.

Cableado



Código

Nota:

- Puedes abrir el archivo `5.8_pot.ino` en la ruta `esp32-starter-kit-main\c\codes\5.8_pot`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de subir el código con éxito, gira el potenciómetro y verás el brillo del LED cambiar en consecuencia. Al mismo tiempo, puedes ver los valores analógicos y de voltaje del potenciómetro en el monitor serie.

Cómo funciona?

1. Define constantes para las conexiones de pines y configuraciones PWM.

```
const int potPin = 14; // Potentiometer connected to GPIO14
const int ledPin = 26; // LED connected to GPIO26

// PWM settings
const int freq = 5000; // PWM frequency
const int resolution = 12; // PWM resolution (bits)
const int channel = 0; // PWM channel
```

Here the PWM resolution is set to 12 bits and the range is 0-4095.

2. Configura el sistema en la función `setup()`.

```
void setup() {
    Serial.begin(115200);

    // Configure PWM
    ledcSetup(channel, freq, resolution);
    ledcAttachPin(ledPin, channel);
}
```

   - En la función `setup()`, se inicia la comunicación Serial a una tasa de baudios de 115200.
   - La función `ledcSetup()` se llama para configurar el canal PWM con la frecuencia y resolución especificadas, y la función `ledcAttachPin()` se llama para asociar el pin LED especificado con el canal PWM.

3. Bucle principal (ejecutado repetidamente) en la función loop().

```
void loop() {

    int potValue = analogRead(potPin); // read the value of the␣
↪potentiometer
    uint32_t voltage_mV = analogReadMilliVolts(potPin); // Read the voltage␣
↪in millivolts

    ledcWrite(channel, potValue);

    Serial.print("Potentiometer Value: ");
    Serial.print(potValue);
    Serial.print(", Voltaje: ");
    Serial.print(voltage_mV / 1000.0); // Convierte milivoltios a voltios
    Serial.println(" V");

    delay(100);
}
```

   - `uint32_t analogReadMilliVolts(uint8_t pin);`: Esta función se utiliza para obtener el valor de ADC para un pin/canal de ADC dado en milivoltios.
     - `pin` Pin GPIO para leer el valor analógico.

El valor del potenciómetro se utiliza directamente como el ciclo de trabajo PWM para controlar el brillo del LED a través de la función `ledcWrite()`, ya que el rango de valores también es de 0 a 4095.

2.25 5.9 Medir la Humedad del Suelo

Este sensor de humedad del suelo capacitivo es diferente de la mayoría de los sensores resistivos en el mercado, utilizando el principio de inducción capacitiva para detectar la humedad del suelo.

Al leer visualmente los valores del sensor de humedad del suelo, podemos recopilar información sobre el nivel de humedad en el suelo. Esta información es útil para varias aplicaciones, como sistemas de riego automático, monitoreo de la salud de las plantas o proyectos de detección ambiental.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Humedad del Suelo	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

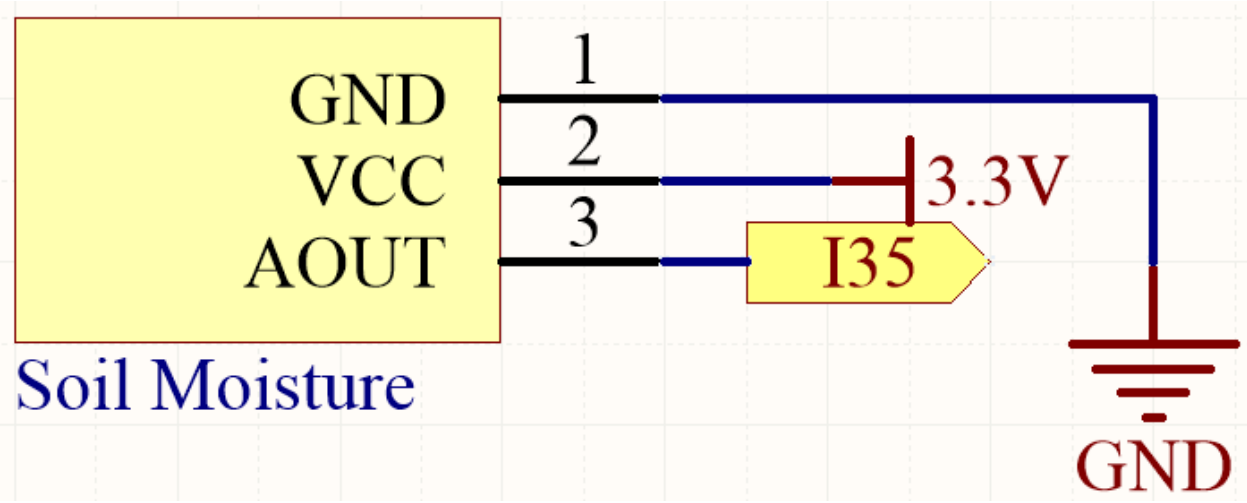
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Strapping**

  Los siguientes pines son pines de strapping, que afectan el proceso de arranque del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 se ha iniciado con éxito, se pueden usar como pines regulares.

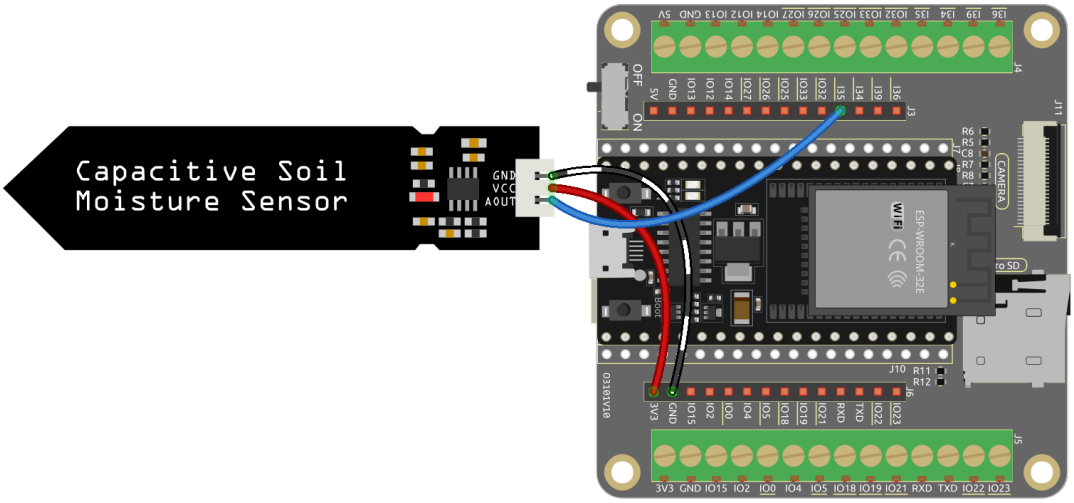
Pines de Strapping	IO0, IO12

Esquemático



Al insertar el módulo en el suelo y regarlo, el valor leído en I35 disminuirá.

Cableado



Código

Nota:

- Abre el archivo `5.9_moisture.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.9_moisture`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Una vez que el código se haya cargado con éxito, el monitor serie imprimirá el valor de la humedad del suelo.

Al insertar el módulo en el suelo y regarlo, el valor del sensor de humedad del suelo se volverá más pequeño.

2.26 5.10 Termómetro

Un termistor es un sensor de temperatura que muestra una fuerte dependencia con la temperatura y puede clasificarse en dos tipos: Coeficiente de Temperatura Negativo (NTC) y Coeficiente de Temperatura Positivo (PTC). La resistencia de un termistor NTC disminuye con el aumento de la temperatura, mientras que la resistencia de un termistor PTC aumenta con el incremento de la temperatura.

En este proyecto, utilizaremos un termistor NTC. Al conectar el termistor NTC a un pin de entrada analógica del microcontrolador ESP32, podemos medir su resistencia, que es directamente proporcional a la temperatura.

Incorporando el termistor NTC y realizando los cálculos necesarios, podemos medir la temperatura con precisión y mostrarla en el módulo LCD1602 I2C. Este proyecto permite el monitoreo de la temperatura en tiempo real y proporciona una interfaz visual para la visualización de la temperatura.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Termistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

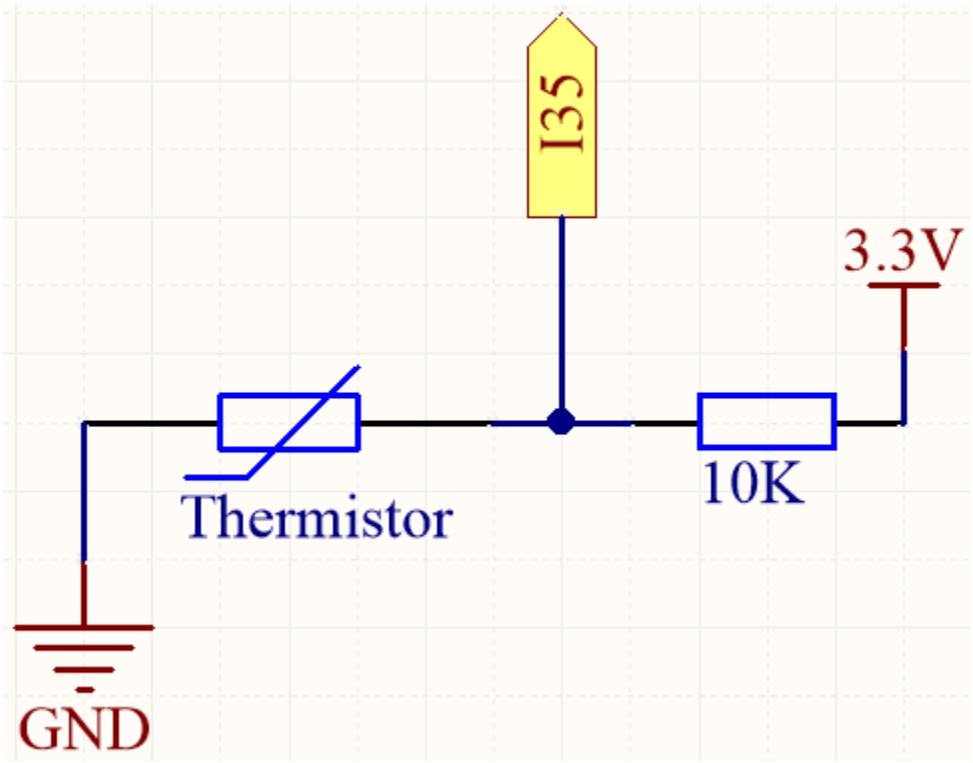
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Configuración**

  Los siguientes pines son pines de configuración, los cuales afectan el proceso de inicio del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 se ha iniciado correctamente, pueden ser utilizados como pines regulares.

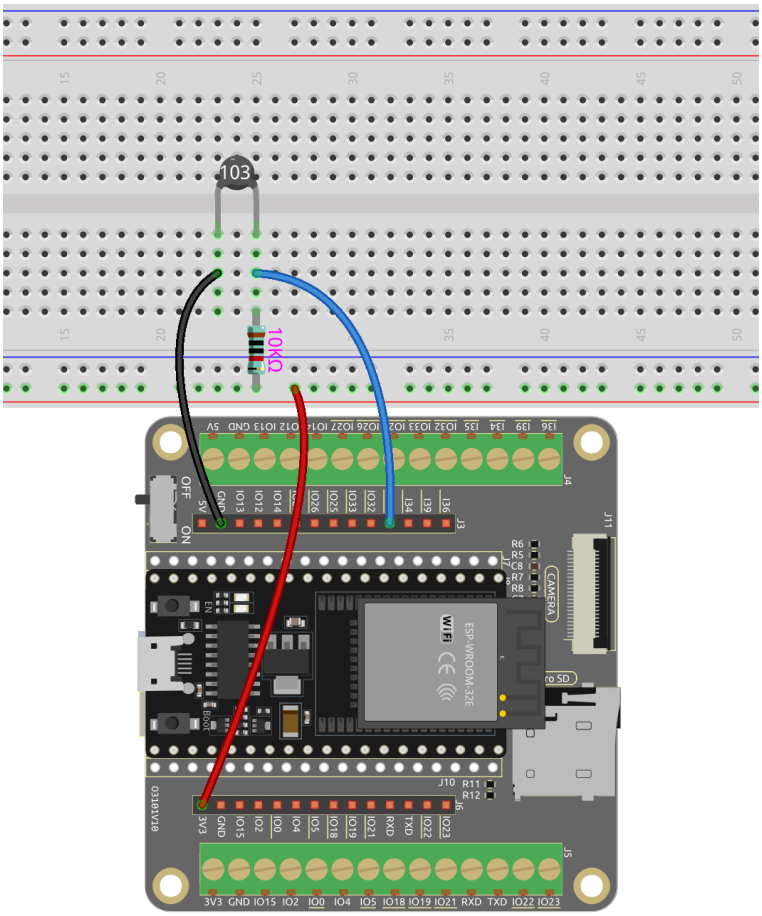
Pines de Configuración	IO0, IO12

Esquemático



Cuando la temperatura aumenta, la resistencia del termistor disminuye, causando que el valor leído en I35 disminuya. Además, utilizando una fórmula, puedes convertir el valor analógico en temperatura y luego imprimirlo.

Conexión



Nota:

- El termistor es negro y está marcado con 103.
- El anillo de color del resistor de 10K ohmios es rojo, negro, negro, rojo y marrón.

Código

Nota:

- Abre el archivo `5.10_thermistor.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.10_thermistor`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de que el código se haya subido con éxito, el Monitor Serial imprimirá las temperaturas en Celsius y Fahrenheit.

¿Cómo funciona?

Cada termistor tiene una resistencia normal. Aquí es de 10k ohmios, medida bajo 25 grados Celsius.

Cuando la temperatura aumenta, la resistencia del termistor disminuye. Luego, los datos de voltaje se convierten en cantidades digitales por el adaptador A/D.

La temperatura en Celsius o Fahrenheit se muestra mediante programación.

Aquí está la relación entre la resistencia y la temperatura:

RT = RN expB(1/TK - 1/TN)

- **RT** es la resistencia del termistor NTC cuando la temperatura es **TK**.
- **RN** es la resistencia del termistor NTC bajo la temperatura nominal TN. Aquí, el valor numérico de RN es 10k.
- **TK** es una temperatura en Kelvin y su unidad es K. Aquí, el valor numérico de **TK** es `373.15 + grados Celsius`.
- **TN** es una temperatura nominal en Kelvin; su unidad también es K. Aquí, el valor numérico de TN es `373.15+25`.
- Y **B(beta)**, la constante de material del termistor NTC, también se llama índice de sensibilidad al calor con un valor numérico `4950`.
- **exp** es la abreviatura de exponencial, y el número base `e` es un número natural que equivale aproximadamente a 2.7.

Convierte esta fórmula `TK=1/(ln(RT/RN)/B+1/TN)` para obtener la temperatura en Kelvin que menos 273.15 equivale a grados Celsius.

Esta relación es una fórmula empírica. Solo es precisa cuando la temperatura y la resistencia están dentro del rango efectivo.

Aprender Más

También puedes mostrar las temperaturas en Celsius y Fahrenheit calculadas en el LCD I2C LCD1602.

Nota:

- Puedes abrir el archivo `5.10_thermistor_lcd.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.10_thermistor_lcd`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.

2.27 5.11 Alternar el Joystick

Si juegas muchos videojuegos, entonces deberías estar muy familiarizado con el Joystick. Se suele utilizar para mover el personaje, rotar la pantalla, etc.

El principio detrás de la capacidad del Joystick para permitir que la computadora lea nuestras acciones es muy simple. Se puede pensar como compuesto por dos potenciómetros que están perpendiculares entre sí. Estos dos potenciómetros miden el valor analógico del joystick vertical y horizontalmente, resultando en un valor (x,y) en un sistema de coordenadas rectangulares planas.

El joystick de este kit también tiene una entrada digital, que se activa cuando se presiona el joystick.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Joystick	

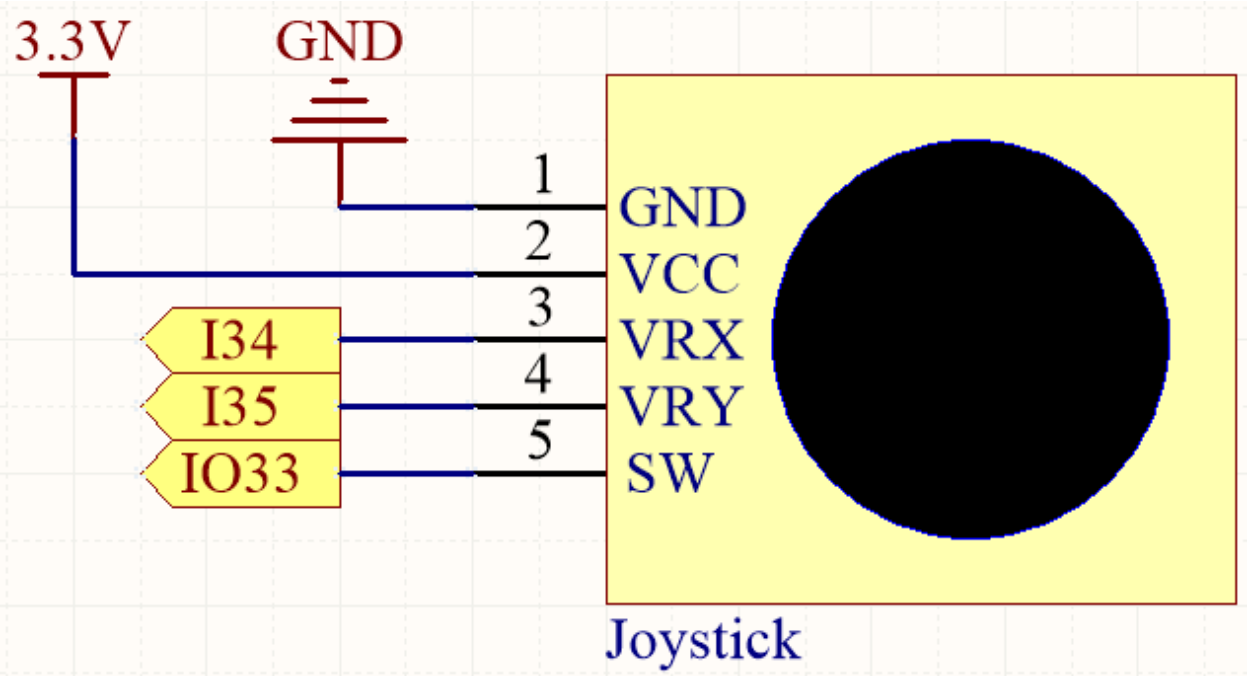
Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada Analógica	IO14, IO25, I35, I34, I39, I36
Para Entrada Digital	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

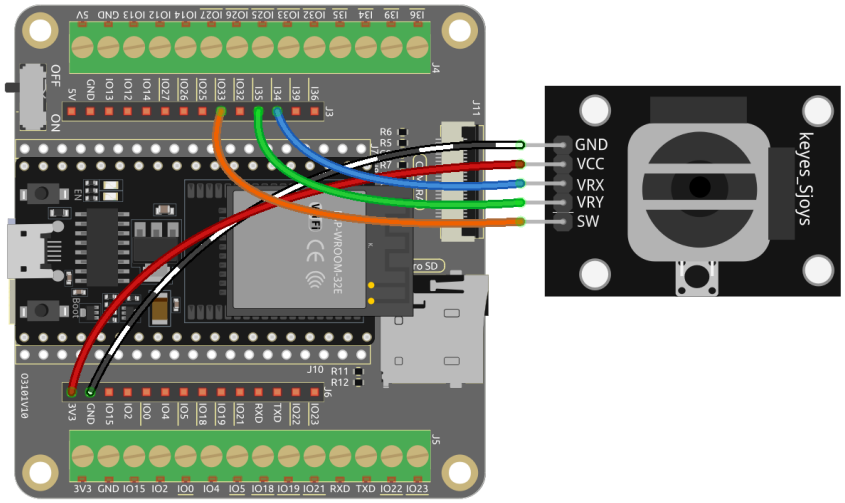
Esquemático



El pin SW (eje Z) está conectado a IO33, que tiene una resistencia de pull-up de 4.7K incorporada. Por lo tanto, cuando no se presiona el botón SW, emitirá un nivel alto. Cuando se presiona el botón, emitirá un nivel bajo.

I34 e I35 cambiarán sus valores a medida que manipules el joystick. El rango de valores es de 0 a 4095.

Cableado



Código

Nota:

- Abre el archivo `5.11_joystick.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.11_joystick`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Abre el monitor serie después de que el código se haya cargado con éxito para ver los valores x, y, z del joystick.

- Los valores de los ejes x e y son valores analógicos que varían de 0 a 4095.
- El eje Z es un valor digital con un estado de 1 o 0 (cuando se presiona, es 0).

2.28 5.12 Medición de Distancia

El módulo ultrasónico se utiliza para medir distancias o detectar objetos. En este proyecto, programaremos el módulo para obtener las distancias de los obstáculos. Enviando pulsos ultrasónicos y midiendo el tiempo que tardan en rebotar, podemos calcular distancias. Esto nos permite implementar acciones basadas en la distancia o comportamientos de evasión de obstáculos.

Componentes Necesarios

Para este proyecto necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo Ultrasonido	

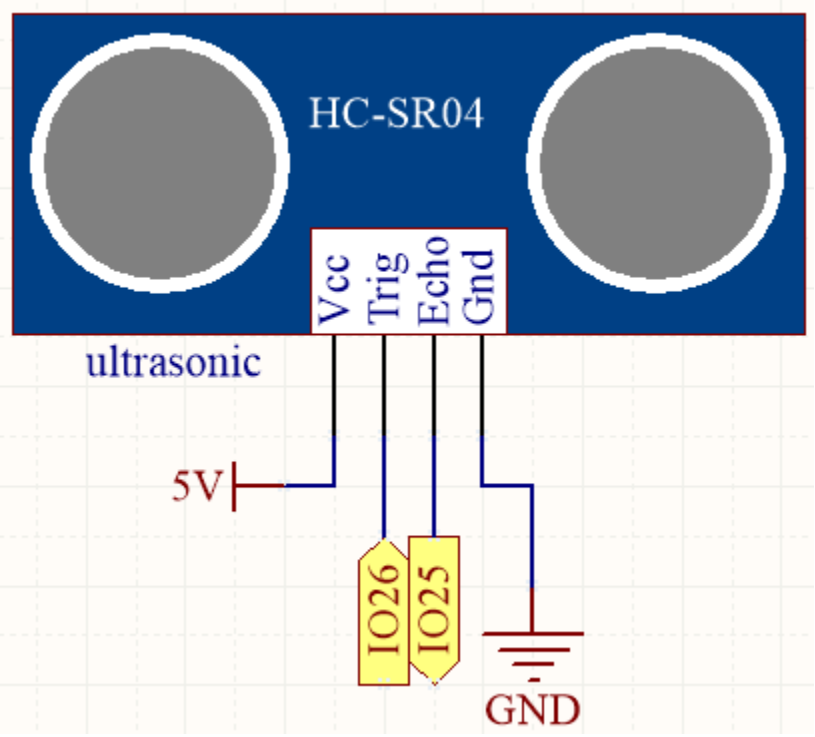
Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO13, IO14, IO27, IO26, IO25, IO33, IO32, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

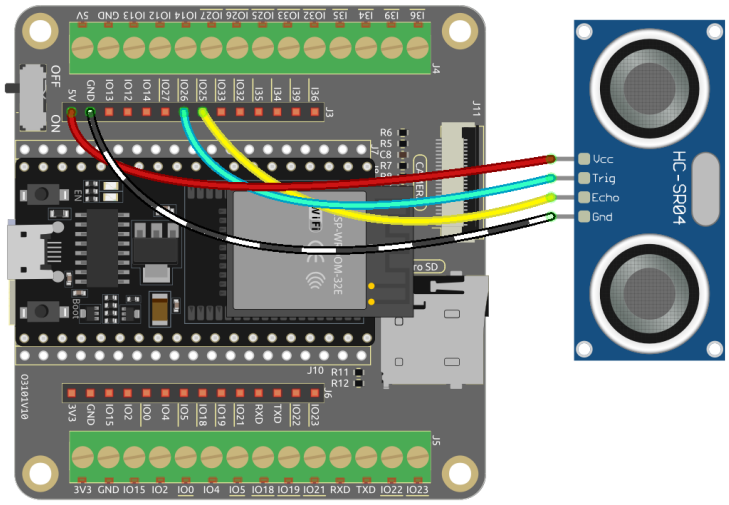
Esquemático



El ESP32 envía un conjunto de señales de onda cuadrada al pin Trig del sensor ultrasónico cada 10 segundos. Esto induce al sensor ultrasónico a emitir una señal de ultrasonido de 40kHz hacia el exterior. Si hay un obstáculo enfrente, las ondas de ultrasonido serán reflejadas de vuelta.

Registrando el tiempo que tarda desde el envío hasta la recepción de la señal, dividiéndolo por 2 y multiplicándolo por la velocidad de la luz, puedes determinar la distancia al obstáculo.

Conexión



Código

Nota:

- Abre el archivo `5.12_ultrasonic.ino` en la ruta `esp32-starter-kit-main\c\codes\5.12_ultrasonic`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Después de que el código se haya subido exitosamente, el monitor serie imprimirá la distancia entre el sensor ultrasónico y el obstáculo adelante.

¿Cómo funciona?

Acerca de la aplicación del sensor ultrasónico, podemos verificar directamente la subfunción.

```
float readSensorData(){// ...}
```

- El `trigPin` del módulo ultrasónico transmite una señal de onda cuadrada de 10us cada 2us.

```
// Trigger a low signal before sending a high signal
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Send a 10-microsecond high signal to the trigPin
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
// Return to low signal
digitalWrite(trigPin, LOW);
```

- El `echoPin` recibe una señal de nivel alto si hay un obstáculo dentro del rango y usa la función `pulseIn()` para registrar el tiempo desde el envío hasta la recepción.

```
unsigned long microsecond = pulseIn(echoPin, HIGH);
```

- La velocidad del sonido es 340 metros por segundo, lo que equivale a 29 microsegundos por centímetro. Midiendo el tiempo que tarda una onda cuadrada en viajar hacia un obstáculo y regresar, podemos calcular la distancia recorrida dividiendo el tiempo total por 2. Esto nos da la distancia del obstáculo desde la fuente de la onda sonora.

```
float distance = microsecond / 29.00 / 2;
```

Toma en cuenta que el sensor ultrasónico pausará el programa cuando esté trabajando, lo cual puede causar cierto retraso al escribir proyectos complejos.

2.29 5.13 Temperatura - Humedad

El DHT11 es un sensor de temperatura y humedad comúnmente utilizado para mediciones ambientales. Es un sensor digital que se comunica con un microcontrolador para proporcionar lecturas de temperatura y humedad.

En este proyecto, estaremos leyendo el sensor DHT11 e imprimiendo los valores de temperatura y humedad que detecta.

Al leer los datos proporcionados por el sensor, podemos obtener los valores actuales de temperatura y humedad en el ambiente. Estos valores pueden usarse para el monitoreo en tiempo real de condiciones ambientales, observaciones meteorológicas, control del clima interior, informes de humedad y más.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Sensor de Humedad y Temperatura DHT11	

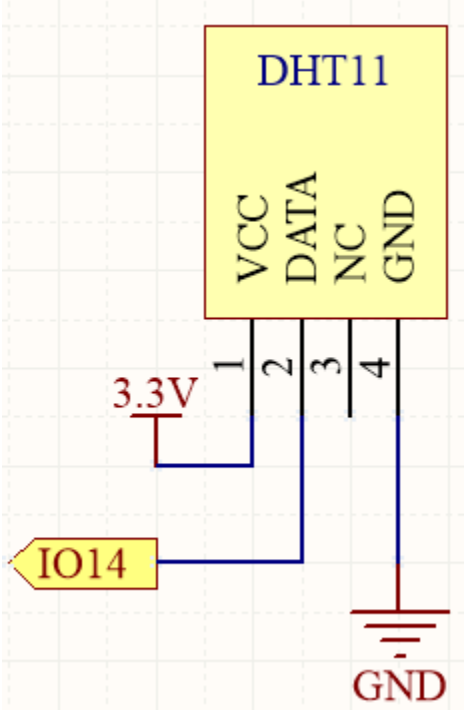
Pines Disponibles

- **Pines Disponibles**

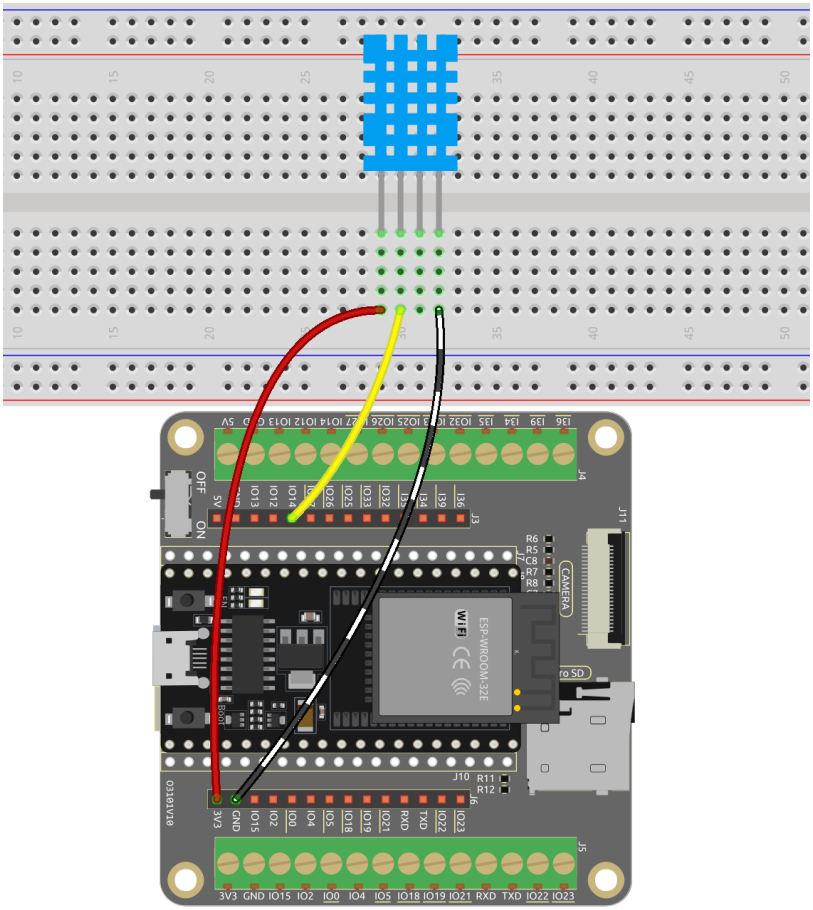
  Aquí está una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



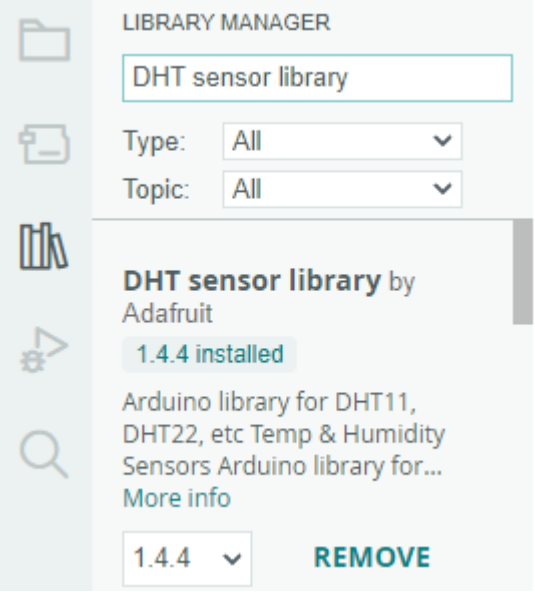
Cableado



Código

Nota:

- Abre el archivo `5.13_dht11.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.13_dht11`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- Se utiliza aquí la biblioteca `DHT sensor library`, puedes instalarla desde el **Administrador de Bibliotecas**.



Después de que el código se haya subido con éxito, verás el Monitor Serial imprimiendo continuamente la temperatura y humedad, y a medida que el programa se ejecute de manera estable, estos dos valores se volverán más y más precisos.

¿Cómo funciona?

1. Incluye la biblioteca `DHT.h`, que proporciona funciones para interactuar con los sensores DHT. Luego, establece el pin y tipo para el sensor DHT.

```
#include "DHT.h"

#define DHTPIN 14  // Set the pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);
```

2. Inicializa la comunicación serial a una tasa de baudios de 115200 e inicializa el sensor DHT.

```
void setup() {
    Serial.begin(115200);
    Serial.println("DHT11 test!");
    dht.begin();
}
```

3. En la función `loop()`, lee los valores de temperatura y humedad del sensor DHT11 e imprímelos en el monitor serial.

```
void loop() {
    // Wait a few seconds between measurements.
    delay(2000);

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (it's a very slow␣
↪sensor)
    float humidity = dht.readHumidity();
    // Read temperature as Celsius (the default)
    float temperture = dht.readTemperature();

    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temperture)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }
    // Print the humidity and temperature
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(temperture);
    Serial.println(" *C");
}
```

- La función `dht.readHumidity()` se llama para leer el valor de humedad del sensor DHT.
- La función `dht.readTemperature()` se llama para leer el valor de temperatura del sensor DHT.
- La función `isnan()` se usa para verificar si las lecturas son válidas. Si el valor de humedad o temperatura es NaN (no es un número), indica una lectura fallida del sensor, y se imprime un mensaje de error.

Aprende Más

También puedes mostrar la temperatura y humedad en el LCD I2C1602.

Nota:

- Puedes abrir el archivo `5.10_thermistor_lcd.ino` bajo la ruta de `euler-kit/arduino/5.10_thermistor_lcd`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- Se utilizan aquí las bibliotecas `LiquidCrystal_I2C` y `DHT sensor library`, puedes instalarlas desde el **Administrador de Bibliotecas**.

2.30 5.14 Receptor IR

Un receptor infrarrojo es un componente que recibe señales infrarrojas y puede detectar y emitir señales compatibles con el nivel TTL de forma independiente. Es similar en tamaño a un transistor empaquetado en plástico regular y se utiliza comúnmente en diversas aplicaciones como control remoto infrarrojo y transmisión infrarroja.

En este proyecto, usaremos un receptor infrarrojo para detectar señales de un control remoto. Cuando se presiona un botón en el control remoto y el receptor infrarrojo recibe la señal correspondiente, puede decodificar la señal para determinar qué botón se presionó. Al decodificar la señal recibida, podemos identificar la tecla o comando específico asociado con ella.

El receptor infrarrojo nos permite incorporar funcionalidad de control remoto en nuestro proyecto, permitiéndonos interactuar con y controlar dispositivos usando señales infrarrojas.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

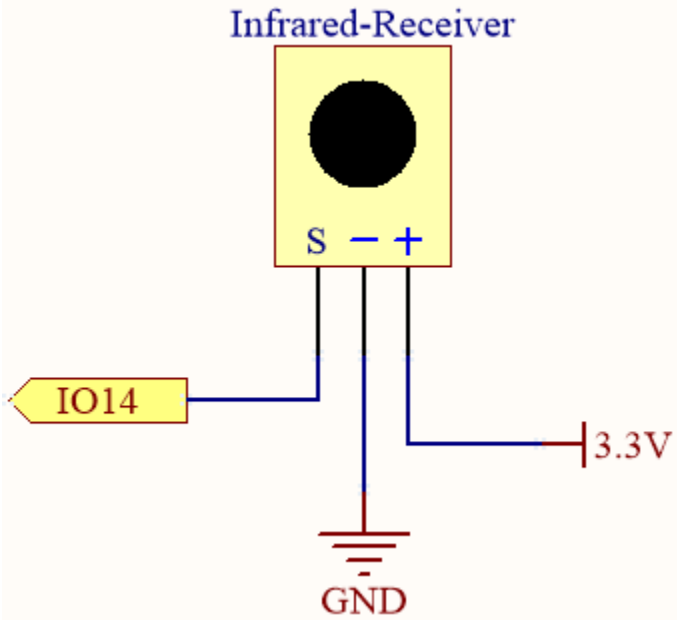
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Receptor de Infrarrojos	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

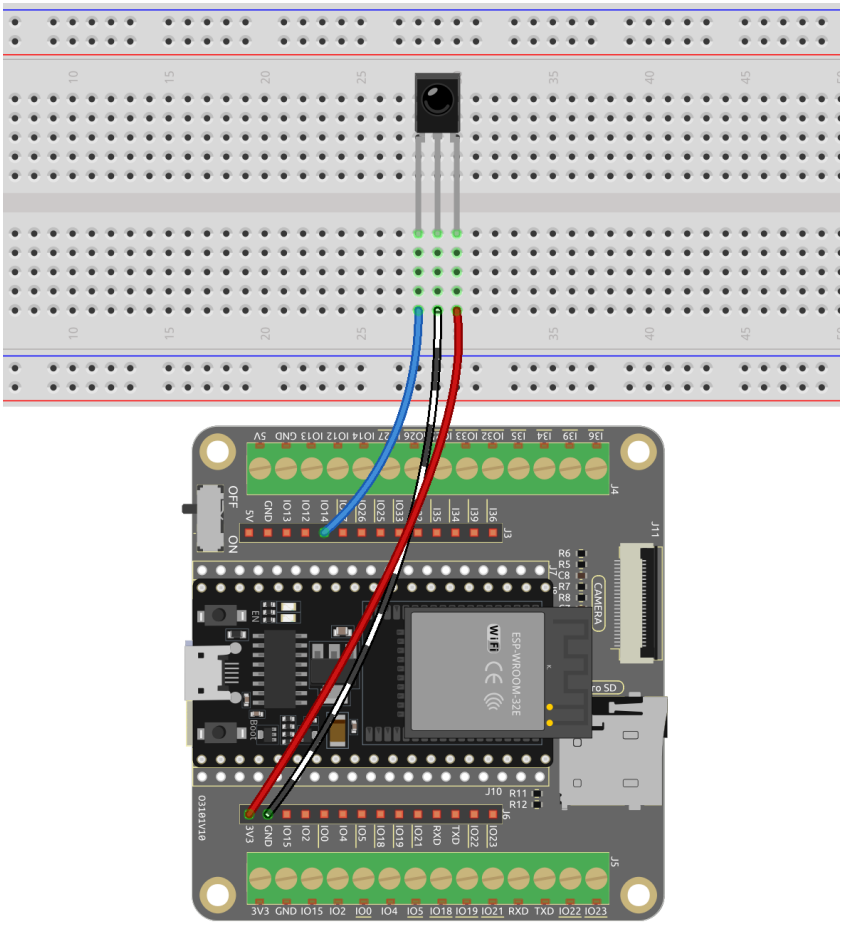
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO15, IO0, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cuando presionas un botón en el control remoto, el receptor infrarrojo detecta la señal, y puedes usar una biblioteca infrarroja para decodificarla. Este proceso de decodificación te permite obtener el valor de la tecla asociada con la presión del botón.

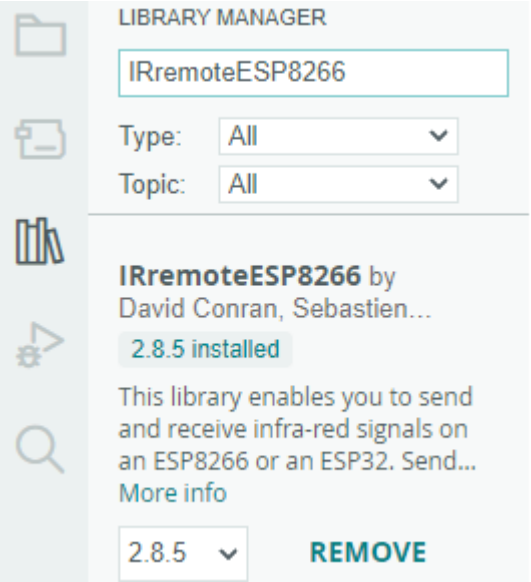
Cableado



Código

Nota:

- Abre el archivo `5.14_ir_receiver.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\5.14_ir_receiver`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- La biblioteca `IRremoteESP8266` se usa aquí, puedes instalarla desde el **Administrador de Bibliotecas**.



Después de que el código se haya cargado con éxito, presiona los diferentes botones en el control remoto y verás aparecer los nombres de estas teclas en el monitor serie.

Nota:

- La biblioteca `IRremoteESP8266` incluye implementaciones para muchos protocolos y dispositivos infrarrojos diferentes, por lo que el tamaño de la biblioteca es relativamente grande. Cuando el compilador tiene que procesar más código, el tiempo de compilación también aumentará en consecuencia. Por favor, sé paciente y espera a que finalice la compilación.
- El control remoto nuevo cuenta con una lengüeta de plástico en el extremo para aislar la batería en su interior. Para activar el control al usarlo, simplemente retira esta pieza de plástico.

¿Cómo funciona?

1. Este código utiliza la biblioteca `IRremoteESP8266` para recibir señales infrarrojas (IR) usando un módulo receptor IR.

```
#include <IRremoteESP8266.h>
#include <IRrecv.h>

// Define the IR receiver pin
const uint16_t IR_RECEIVE_PIN = 14;
```

(continué en la próxima página)

(proviene de la página anterior)

```
// Create an IRrecv object
IRrecv irrecv(IR_RECEIVE_PIN);

// Create a decode_results object
decode_results results;
```

2. En la función `setup()`, la comunicación serie se inicia a una tasa de baudios de 115200, y el receptor IR se habilita usando `irrecv.enableIRIn()`.

```
void setup() {
    // Start serial communication
    Serial.begin(115200);

    // Start the IR receiver
    irrecv.enableIRIn();
}
```

3. Cuando presionas una tecla en el control remoto, el monitor serie imprimirá el nombre de la tecla si es recibido por el receptor IR.

```
void loop() {
    // If an IR signal is received
    if (irrecv.decode(&results)) {
        String key = decodeKeyValue(results.value);
        if (key != "ERROR") {
            // Print the value of the signal to the serial monitor
            Serial.println(key);
        }
        irrecv.resume(); // Continue to receive the next signal
    }
}
```

   - Primero, verifica si se recibió una señal IR usando la función `irrecv.decode()`.
   - Si se recibe una señal, entonces llama a la función `decodeKeyValue()` para decodificar el valor de la señal.
   - Si la señal se decodifica con éxito, el valor decodificado se imprime en el monitor serie usando `Serial.println()`.
   - Finalmente, `irrecv.resume()` se llama para continuar recibiendo la siguiente señal.

4. La función `decodeKeyValue()` toma el valor decodificado de la señal IR como argumento y devuelve una cadena que representa la tecla presionada en el control remoto.

```
String decodeKeyValue(long result)
{
    switch(result){
        case 0xFF6897:
        return "0";
        case 0xFF30CF:
        return "1";
        case 0xFF18E7:
        return "2";
```

(continué en la próxima página)

(proviene de la página anterior)

```
        case 0xFF7A85:
        ...
```

- La función utiliza una instrucción switch para hacer coincidir el valor decodificado con la tecla correspondiente y devuelve la representación en cadena de la tecla.
- Si el valor decodificado no coincide con ninguna tecla conocida, la función devuelve la cadena «ERROR».

6. Proyectos Divertidos

2.31 6.1 Piano de Frutas

¿Alguna vez has querido tocar el piano pero no podías permitírtelo? ¿O tal vez solo quieres divertirte haciendo un piano de frutas tú mismo? Bueno, ¡este proyecto es para ti!

Con solo unos pocos sensores táctiles en la placa ESP32, ahora puedes tocar tus melodías favoritas y disfrutar de la experiencia de tocar el piano sin gastar mucho.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Resistor	
Zumbador	
Transistor	

Acerca de los Pines Táctiles

El microcontrolador ESP32 tiene funcionalidad de sensor táctil incorporada, lo que te permite usar ciertos pines en la placa como entradas sensibles al tacto. El sensor táctil funciona midiendo cambios en la capacitancia en los pines táctiles, que son causados por las propiedades eléctricas del cuerpo humano.

Aquí hay algunas características clave del sensor táctil en el ESP32:

- **Número de pines táctiles**

El ESP32 tiene hasta 10 pines táctiles, dependiendo de la placa específica. Los pines táctiles suelen estar etiquetados con una «T» seguida de un número.

- GPIO4: TOUCH0
- GPIO0: TOUCH1
- GPIO2: TOUCH2
- GPIO15: TOUCH3
- GPIO13: TOUCH4
- GPIO12: TOUCH5
- GPIO14: TOUCH6
- GPIO27: TOUCH7
- GPIO33: TOUCH8
- GPIO32: TOUCH9

Nota: Los pines GPIO0 y GPIO2 se utilizan para el arranque y la carga del firmware en el ESP32, respectivamente. Estos pines también están conectados al LED y botón integrados. Por lo tanto, generalmente no se recomienda usar estos pines para otros propósitos, ya que podría interferir con el funcionamiento normal de la placa.

- **Sensibilidad**

  El sensor táctil en el ESP32 es muy sensible y puede detectar incluso pequeños cambios en la capacitancia. La sensibilidad se puede ajustar usando configuraciones de software.

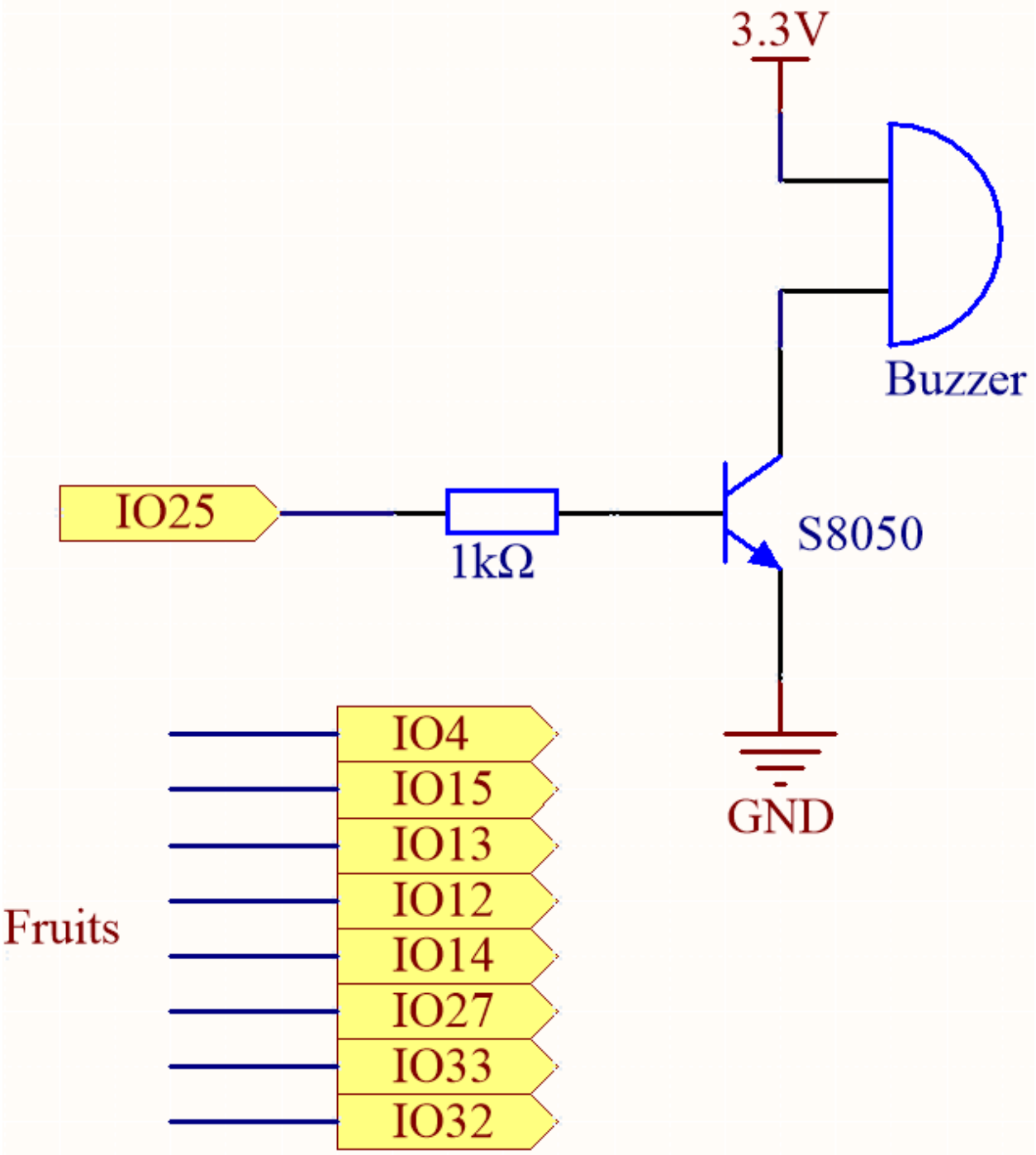
- **Protección ESD**

  Los pines táctiles en el ESP32 tienen protección ESD (Descarga Electroestática) incorporada, lo que ayuda a prevenir daños en la placa por electricidad estática.

- **Multitáctil**

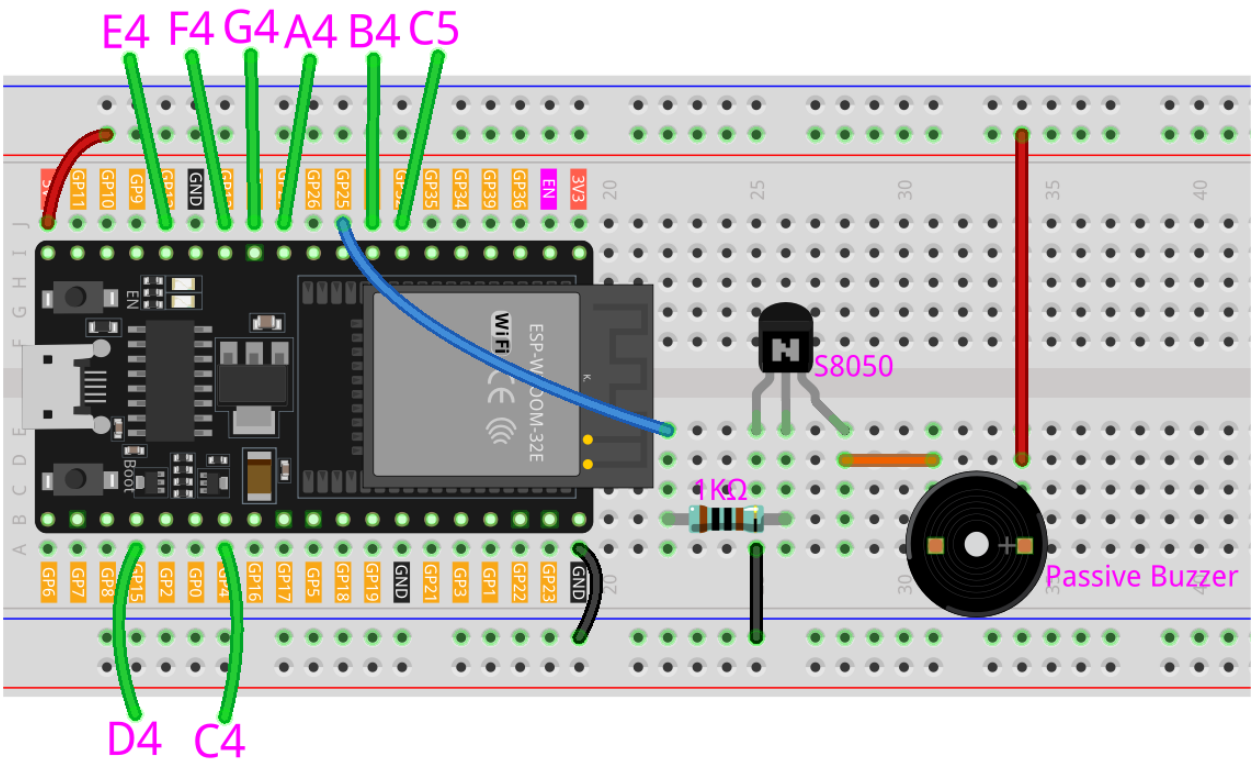
  El sensor táctil en el ESP32 admite multitáctil, lo que significa que puedes detectar varios eventos táctiles simultáneamente.

Esquemático



La idea detrás de este proyecto es usar sensores táctiles para detectar cuando un usuario toca un pin específico. Cada pin táctil está asociado con una nota específica, y cuando el usuario toca un pin, la nota correspondiente se reproduce en el zumbador pasivo. El resultado es una forma simple y asequible de disfrutar de la experiencia de tocar el piano.

Cableado



En este proyecto, necesitas quitar el ESP32 WROOM 32E de la placa de expansión y luego insertarlo en el protoboard. Esto se debe a que algunos pines en la placa de expansión están conectados a resistencias, lo que afectará la capacitancia de los pines.

Código

Nota:

- Puedes abrir el archivo `6.1_fruit_piano.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\6.1_fruit_piano` directamente.
- O copia este código en el IDE de Arduino.

Puedes conectar frutas a estos pines del ESP32: 4, 15, 13, 12, 14, 27, 33, 32.

Cuando el script se ejecuta, tocar estas frutas reproducirá las notas C, D, E, F, G, A, B y C5.

¿Cómo funciona?

- `touchRead(uint8_t pin);`

  Esta función obtiene los datos del sensor táctil. Cada sensor táctil tiene un contador para contar el número de ciclos de carga/descarga. Cuando el pad es **tocado**, el valor en el contador cambiará debido a la mayor capacitancia equivalente. El cambio de los datos determina si el pad ha sido tocado o no.

  - `pin` pin GPIO para leer el valor TOUCH

  Esta función devuelve un valor entre 0 y 4095, con un valor más bajo indicando una entrada táctil más fuerte.

Nota: `threshold` necesita ser ajustado basado en la conductividad de diferentes frutas.

Puedes ejecutar el script primero para ver los valores impresos por el shell.

```
0: 60
1: 62
2: 71
3: 74
4: 73
5: 78
6: 80
7: 82
```

Después de tocar las frutas en los pines 12, 14 y 27, los valores impresos son los siguientes. Por lo tanto, establecí el `threshold` en 30, lo que significa que cuando se detecta un valor menor a 30, se considera tocado, y el zumbador emitirá diferentes notas.

```
0: 60
1: 62
2: 71
3: 9
4: 12
5: 14
6: 75
7: 78
```

2.32 6.2 Luz Fluyente

¿Alguna vez has querido añadir un elemento divertido e interactivo a tu espacio vital? Este proyecto implica crear una luz corriente usando una tira de LED WS2812 y un módulo de evitación de obstáculos. La luz corriente cambia de dirección cuando se detecta un obstáculo, lo que la convierte en una adición emocionante a la decoración de tu hogar u oficina.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

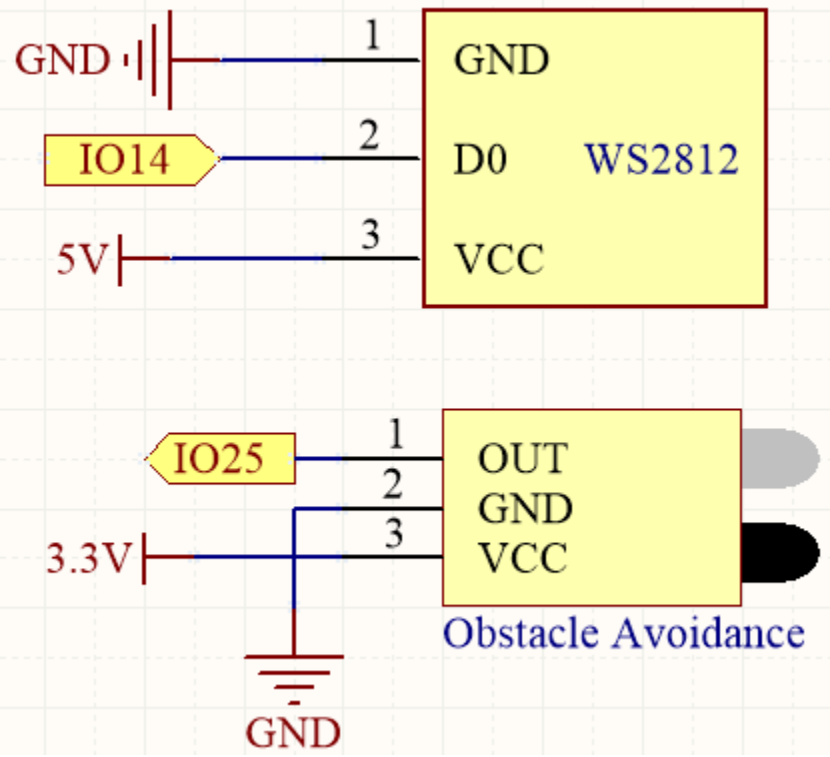
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

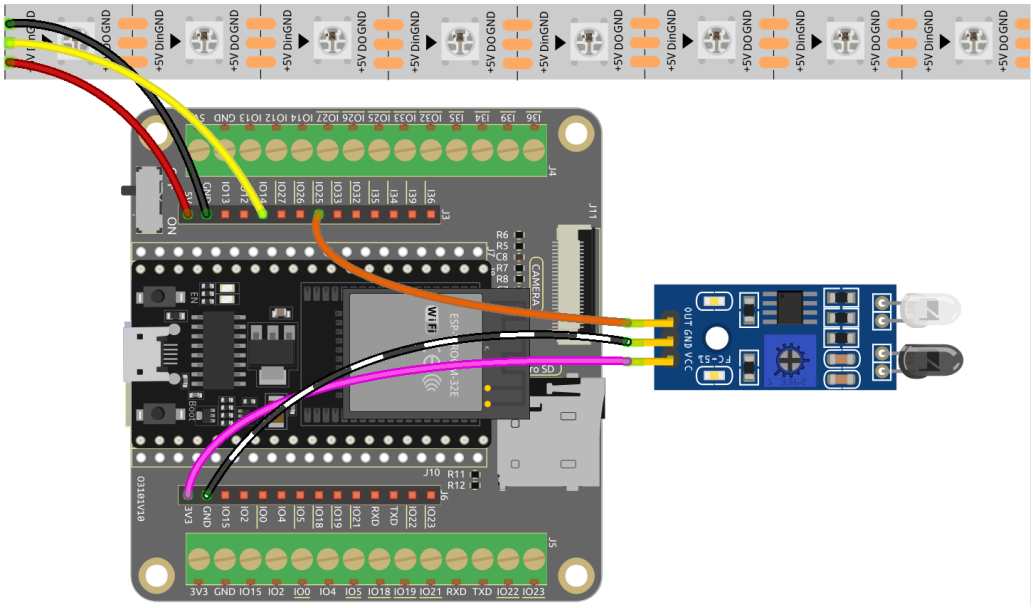
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Evitación de Obstáculos	
Tira de 8 LEDs RGB WS2812	

Diagrama Esquemático



La tira de LED WS2812 está compuesta por una serie de LEDs individuales que pueden ser programados para mostrar diferentes colores y patrones. En este proyecto, la tira está configurada para mostrar una luz corriente que se mueve en una dirección particular y cambia de dirección cuando un obstáculo es detectado por el módulo de evitación de obstáculos.

Cableado



Código

Nota:

- Puedes abrir el archivo `6.2_flowing_led.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\6.2_flowing_led` directamente.
- O copia este código en el IDE de Arduino.

Este proyecto extiende la funcionalidad del proyecto *2.7 Tira de LEDs RGB* añadiendo la capacidad de mostrar colores aleatorios en la tira de LED. Adicionalmente, se ha incluido un módulo de evitación de obstáculos para cambiar dinámicamente la dirección de la luz corriente.

2.33 6.3 Ayuda para Reversa

Imagina esto: estás en tu coche, a punto de estacionarte en reversa en un lugar estrecho. Con nuestro proyecto, tendrás un módulo ultrasónico montado en la parte trasera de tu vehículo, actuando como un ojo digital. Al poner la marcha en reversa, el módulo cobra vida, emitiendo pulsos ultrasónicos que rebotan en los obstáculos detrás de ti.

La magia ocurre cuando estos pulsos regresan al módulo. Calcula rápidamente la distancia entre tu coche y los objetos, transformando estos datos en retroalimentación visual en tiempo real mostrada en una vibrante pantalla LCD. Presenciarás indicadores dinámicos y codificados por colores que describen la proximidad de los obstáculos, asegurando que tengas una comprensión cristalina del entorno circundante.

Pero no nos detuvimos ahí. Para sumergirte aún más en esta experiencia de conducción, incorporamos un zumbador vivaz. A medida que tu coche se acerca a un obstáculo, el ritmo del zumbador se intensifica, creando una sinfonía auditiva de advertencias. Es como tener una orquesta personal guiándote a través de las complejidades del estacionamiento en reversa.

Este innovador proyecto combina tecnología de vanguardia con una interfaz de usuario interactiva, haciendo que tu experiencia al reversar sea segura y libre de estrés. Con el módulo ultrasónico, la pantalla LCD y el zumbador vivaz trabajando armoniosamente, te sentirás empoderado y seguro mientras maniobras en espacios estrechos, dejándote libre para enfocarte en el placer de conducir.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

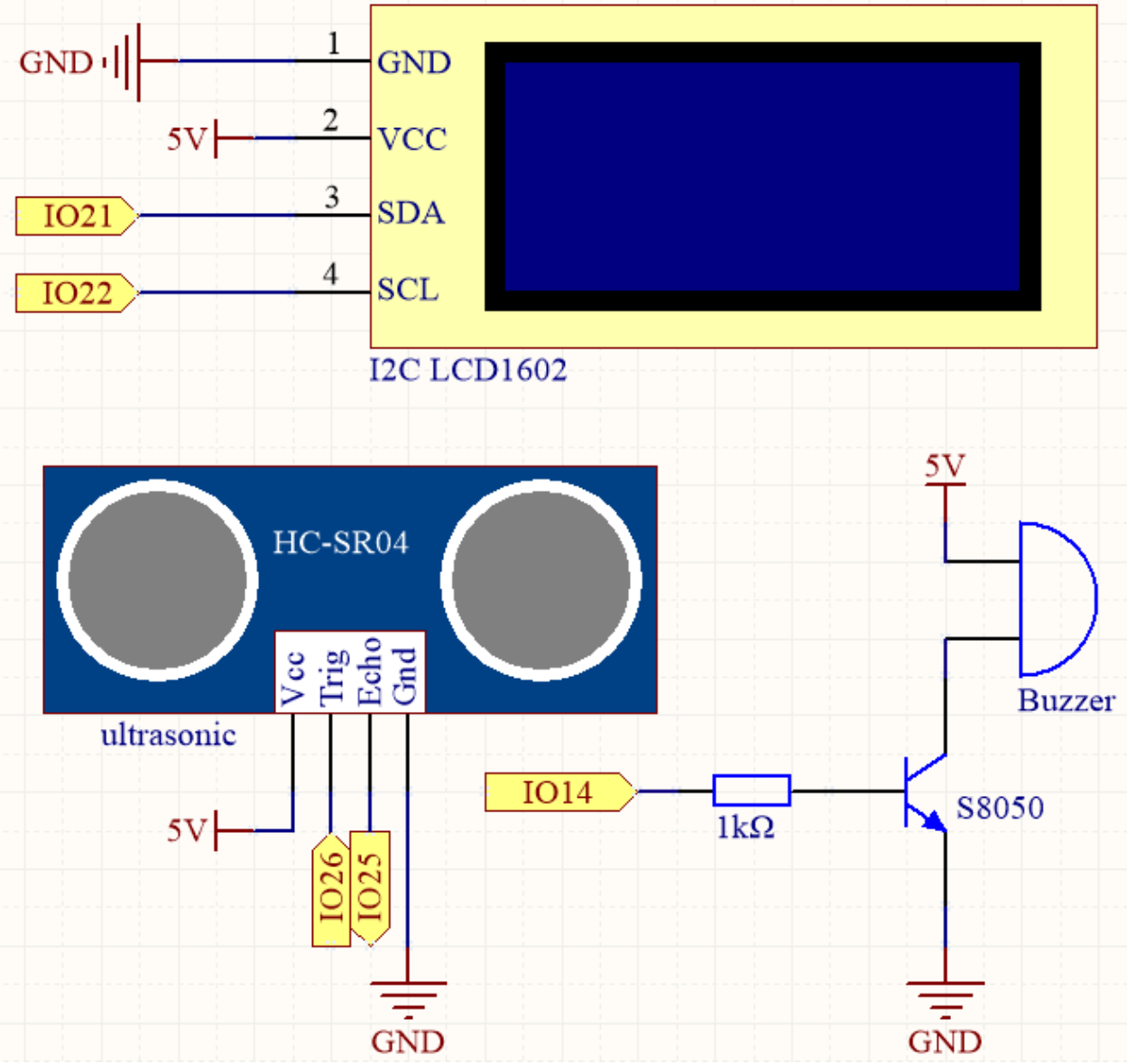
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

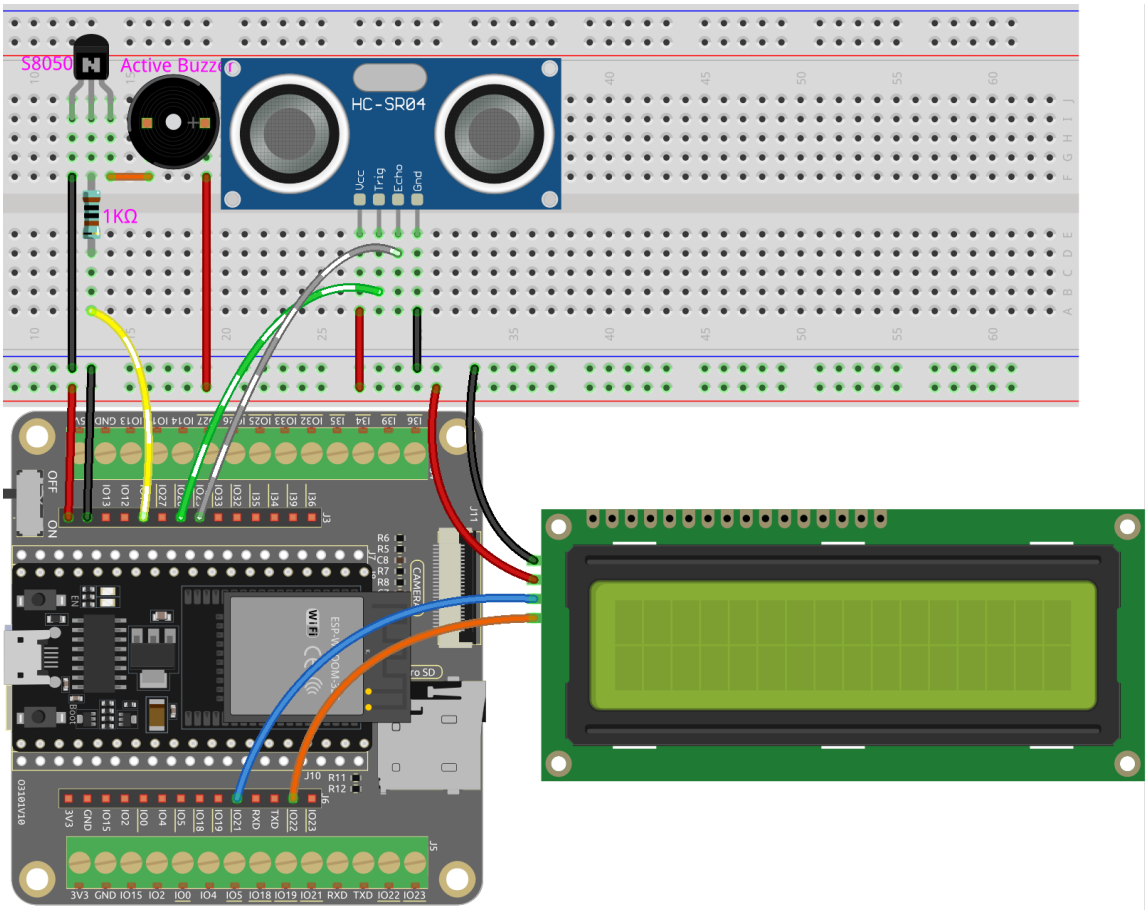
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Módulo Ultrasonido	
Zumbador	-
Transistor	
I2C LCD1602	

Esquema



El sensor ultrasónico en el proyecto emite ondas sonoras de alta frecuencia y mide el tiempo que tardan en rebotar después de golpear un objeto. Al analizar estos datos, se puede calcular la distancia entre el sensor y el objeto. Para proporcionar una advertencia cuando el objeto esté demasiado cerca, se utiliza un zumbador para producir una señal audible. Además, la distancia medida se muestra en una pantalla LCD para una fácil visualización.

Cableado



Código

Nota:

- Puedes abrir el archivo `6.3_reversing_aid.ino` directamente bajo la ruta de `esp32-starter-kit-main\c\codes\6.3_reversing_aid`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- La biblioteca `LiquidCrystal I2C` se utiliza aquí, puedes instalarla desde el **Administrador de Bibliotecas**.

Después de que el código se haya subido con éxito, la distancia detectada actual se mostrará en la LCD. Luego, el zumbador cambiará la frecuencia de sonido según las diferentes distancias.

Nota: Si el código y el cableado son correctos, pero la LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte posterior para aumentar el contraste.

¿Cómo funciona?

Este código nos ayuda a crear un dispositivo simple de medición de distancia que puede medir la distancia entre objetos y proporcionar retroalimentación a través de una pantalla LCD y un zumbador.

La función `loop()` contiene la lógica principal del programa y se ejecuta continuamente. Echemos un vistazo más de cerca a la función `loop()`.

1. Bucle para leer la distancia y actualizar parámetros

   En el `loop`, el código primero lee la distancia medida por el módulo ultrasónico y actualiza el parámetro del intervalo basado en la distancia.

   ```
   // Update the distance
   distance = readDistance();

   // Update intervals based on distance
   if (distance <= 10) {
       intervals = 300;
   } else if (distance <= 20) {
       intervals = 500;
   } else if (distance <= 50) {
       intervals = 1000;
   } else {
       intervals = 2000;
   }
   ```

2. Verificar si es momento de pitar

   El código calcula la diferencia entre el tiempo actual y el tiempo anterior del pitido, y si la diferencia es mayor o igual al tiempo del intervalo, activa el zumbador y actualiza el tiempo anterior del pitido.

   ```
   unsigned long currentMillis = millis();
   if (currentMillis - previousMillis >= intervals) {
       Serial.println("Beeping!");
       beep();
       previousMillis = currentMillis;
   }
   ```

3. Actualizar la pantalla LCD

   El código limpia la pantalla LCD y luego muestra «Dis:» y la distancia actual en centímetros en la primera línea.

   ```
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Dis: ");
   lcd.print(distance);
   lcd.print(" cm");

   delay(100);
   ```

2.34 6.4 Dados Digitales

Este proyecto se basa en el proyecto *2.5 Visualización de Números* agregando un botón para controlar el dígito mostrado en el display de siete segmentos.

En este proyecto, se genera un número aleatorio y se muestra en el display de siete segmentos para simular un lanzamiento de dados. Cuando se presiona el botón, se muestra un número estable (seleccionado al azar de 1 a 6) en el display de siete segmentos. Presionar el botón nuevamente iniciará la simulación de un lanzamiento de dados, generando números aleatorios como antes. Este ciclo continúa cada vez que se presiona el botón.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

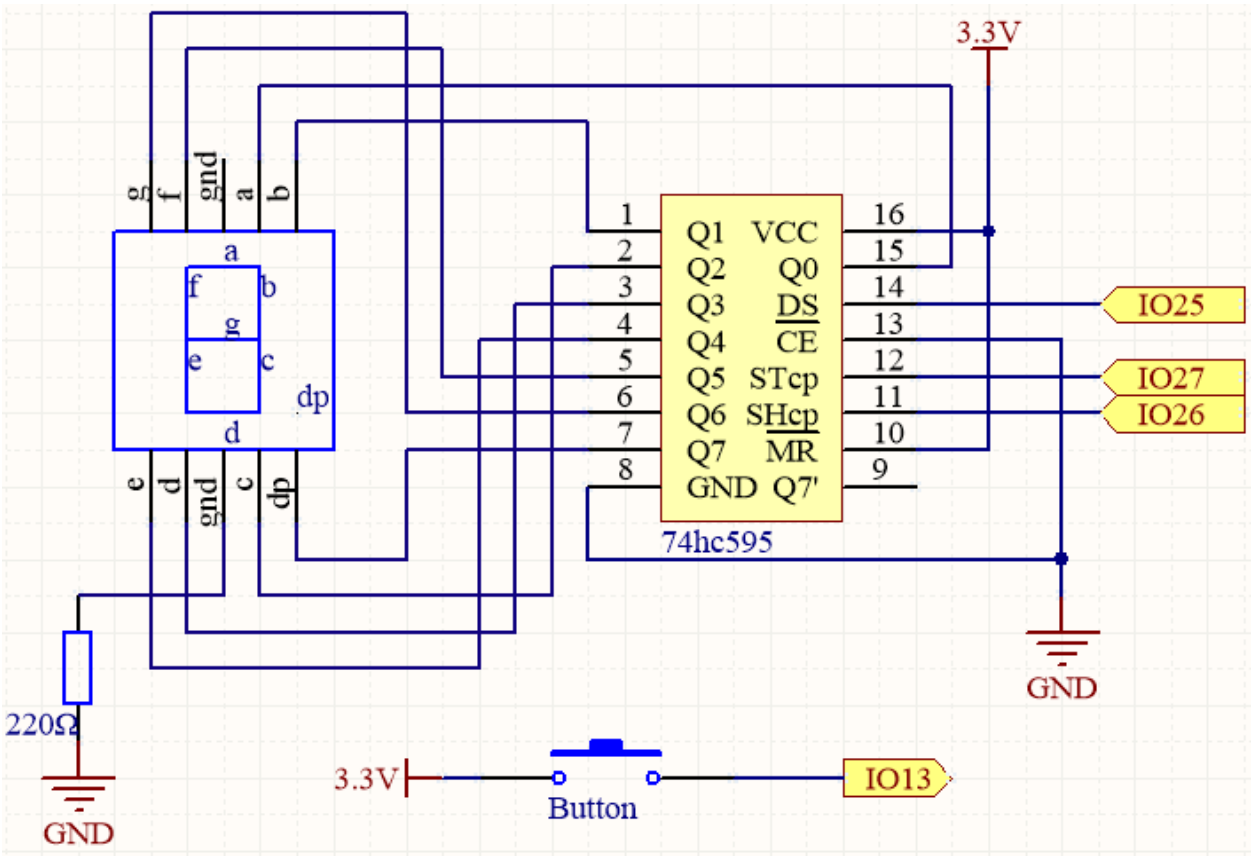
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
74HC595	
display de 7 Segmentos	
Botón	

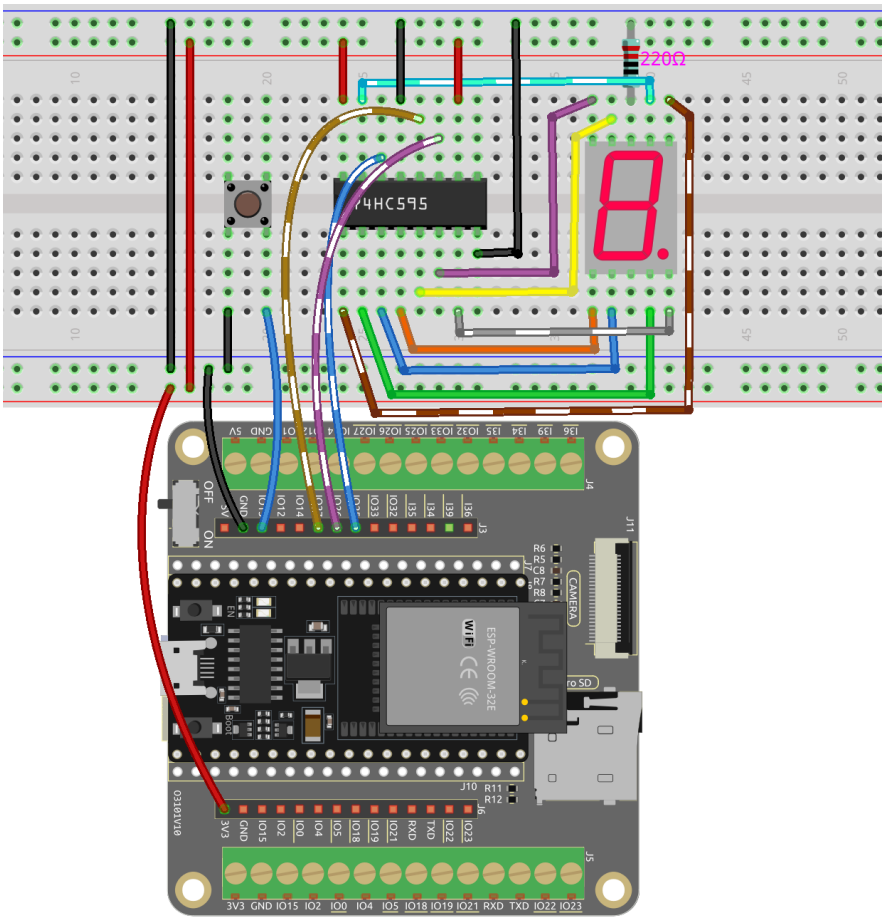
Esquemático



Este proyecto se basa en el proyecto *2.5 Pantalla de 7 Segmentos* agregando un botón para controlar el dígito mostrado en el display de siete segmentos.

El botón está conectado directamente a IO13 sin una resistencia de pull-up o pull-down externa porque IO13 tiene una resistencia de pull-up interna de 47K, eliminando la necesidad de una resistencia externa adicional.

Cableado



Código

Nota:

- Abre el archivo `6.4_digital_dice.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\6.4_digital_dice`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Este proyecto se basa en *2.5 Pantalla de 7 Segmentos* con un botón para iniciar/pausar el desplazamiento de la pantalla en el Display de 7 segmentos.

Cuando se presiona el botón, el display de 7 segmentos desplaza los números del 1-6, y cuando se suelta el botón, muestra un número aleatorio.

2.35 6.5 Degradado de Color

¿Estás listo para experimentar un mundo de color? Este proyecto te llevará en un viaje mágico donde podrás controlar un LED RGB y lograr transiciones suaves de color. Ya sea que busques añadir algo de color a tu decoración del hogar o buscando un proyecto de programación divertido, este proyecto lo tiene todo. ¡Sumérgete juntos en este mundo colorido!

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

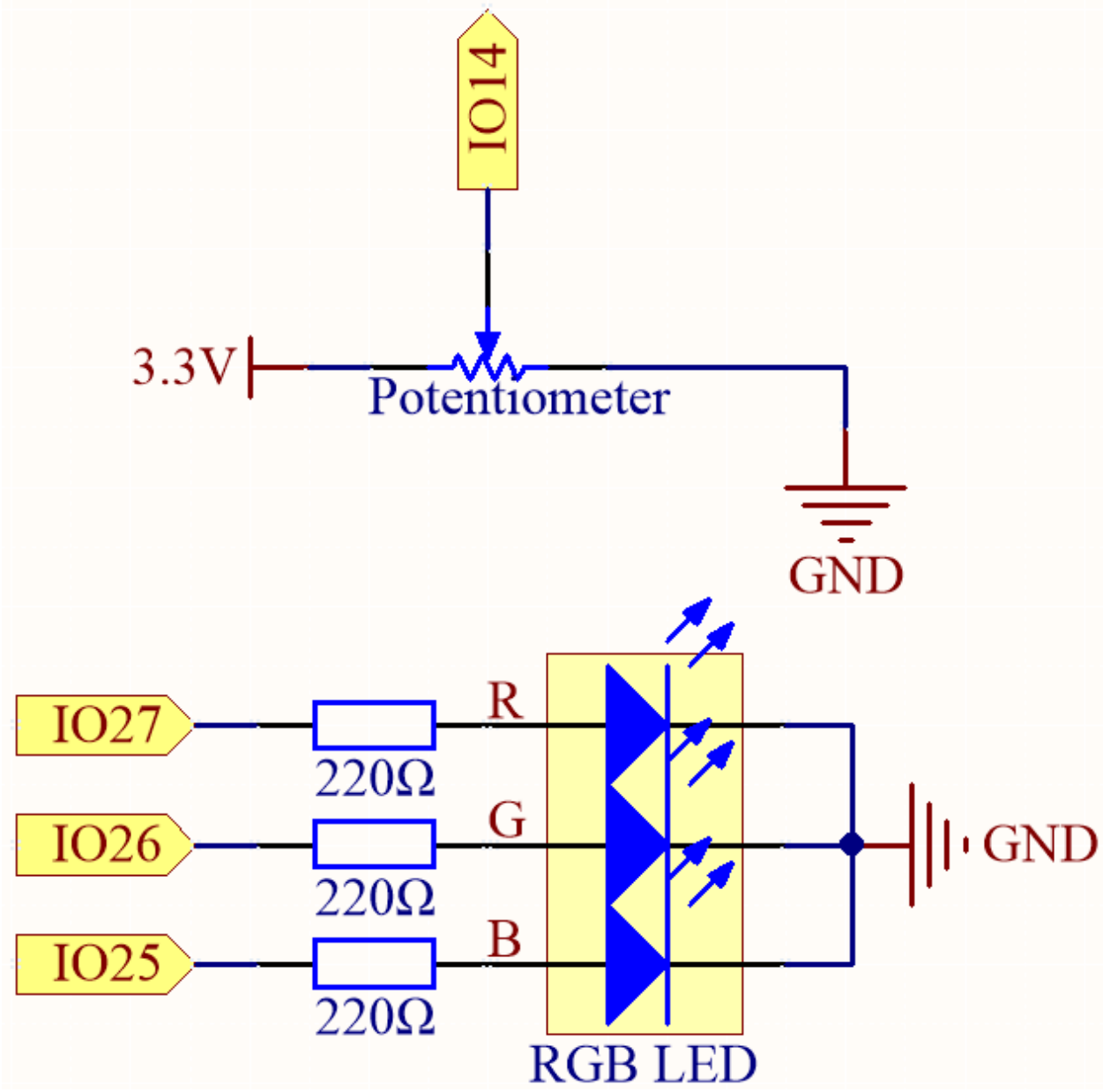
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

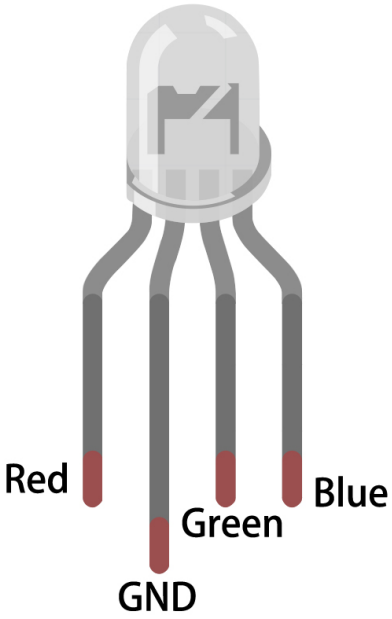
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Potenciómetro	
LED RGB	

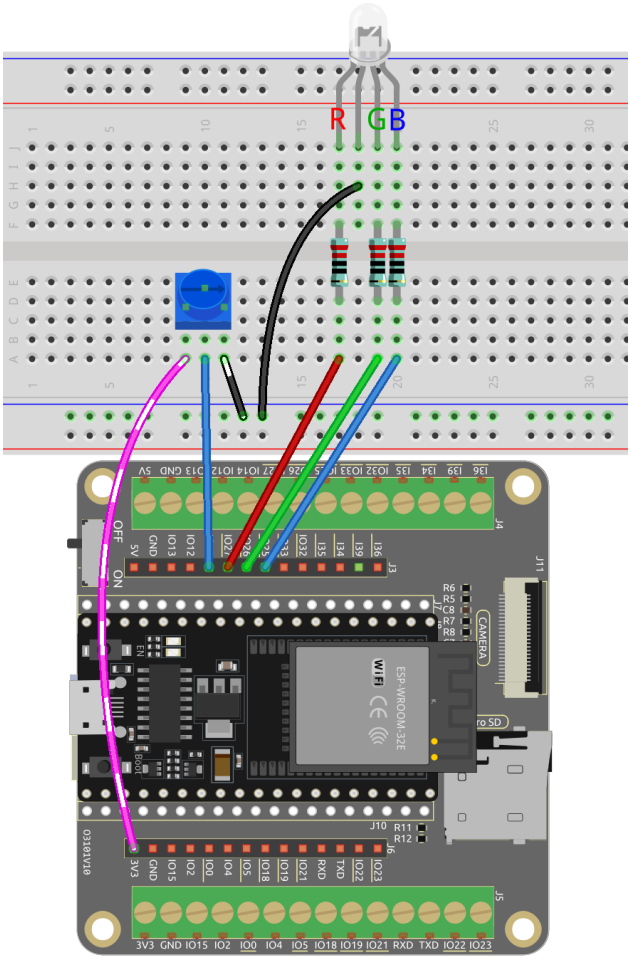
Esquemático


IO14
3.3V
Potentiometer
GND
IO27
220Ω
R
IO26
220Ω
G
IO25
220Ω
B
GND
RGB LED

Cableado



El LED RGB tiene 4 pines: el pin largo es el pin cátodo común, que generalmente se conecta a GND; el pin izquierdo al lado del pin más largo es Rojo; y los dos pines a la derecha son Verde y Azul.



Código

Nota:

- Puedes abrir el archivo `6.5_color_gradient.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\6.5_color_gradient`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

Este proyecto utiliza un LED RGB y un potenciómetro para crear un efecto de mezcla de colores. El potenciómetro se utiliza para ajustar el valor del tono del LED, que luego se convierte en valores RGB utilizando una función de conversión de color. Los valores RGB se utilizan luego para actualizar el color del LED.

¿Cómo funciona?

Este proyecto se basa en el proyecto *2.3 Luz Colorida* añadiendo un potenciómetro para ajustar el valor del tono del LED. El valor del tono se convierte a valores RGB utilizando la función `HUEtoRGB()`.

1. En la función loop, lee el valor del potenciómetro y conviértelo a un valor de tono (0-360).

```
int knobValue = analogRead(KNOB_PIN);
float hueValue = (float) knobValue / 4095.0;
int hue = (int) (hueValue * 360);
```

2. Convierte el valor del tono a valores RGB utilizando la función `HUEtoRGB()`, y actualiza el LED con los nuevos valores de color.

```
int red, green, blue;
HUEtoRGB(hue, &red, &green, &blue);
setColor(red, green, blue);
```

3. La función `setColor()` establece el valor de los canales rojo, verde y azul utilizando la biblioteca `LEDC`.

```
void setColor(int red, int green, int blue) {
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

4. La función `HUEtoRGB` convierte un valor de tono a valores RGB utilizando el modelo de color HSL.

```
void HUEtoRGB(int hue, int* red, int* green, int* blue) {
    float h = (float) hue / 60.0;
    float c = 1.0;
    float x = c * (1.0 - fabs(fmod(h, 2.0) - 1.0));
    float r, g, b;
    if (h < 1.0) {
        r = c;
        g = x;
        b = 0;
...
```

2.36 6.6 Monitor de Plantas

¡Bienvenidos al proyecto Monitor de Plantas!

En este proyecto, utilizaremos una placa ESP32 para crear un sistema que nos ayude a cuidar de nuestras plantas. Con este sistema, podemos monitorear la temperatura, humedad, humedad del suelo y niveles de luz de nuestras plantas, asegurando que reciban el cuidado y la atención necesarios para prosperar.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

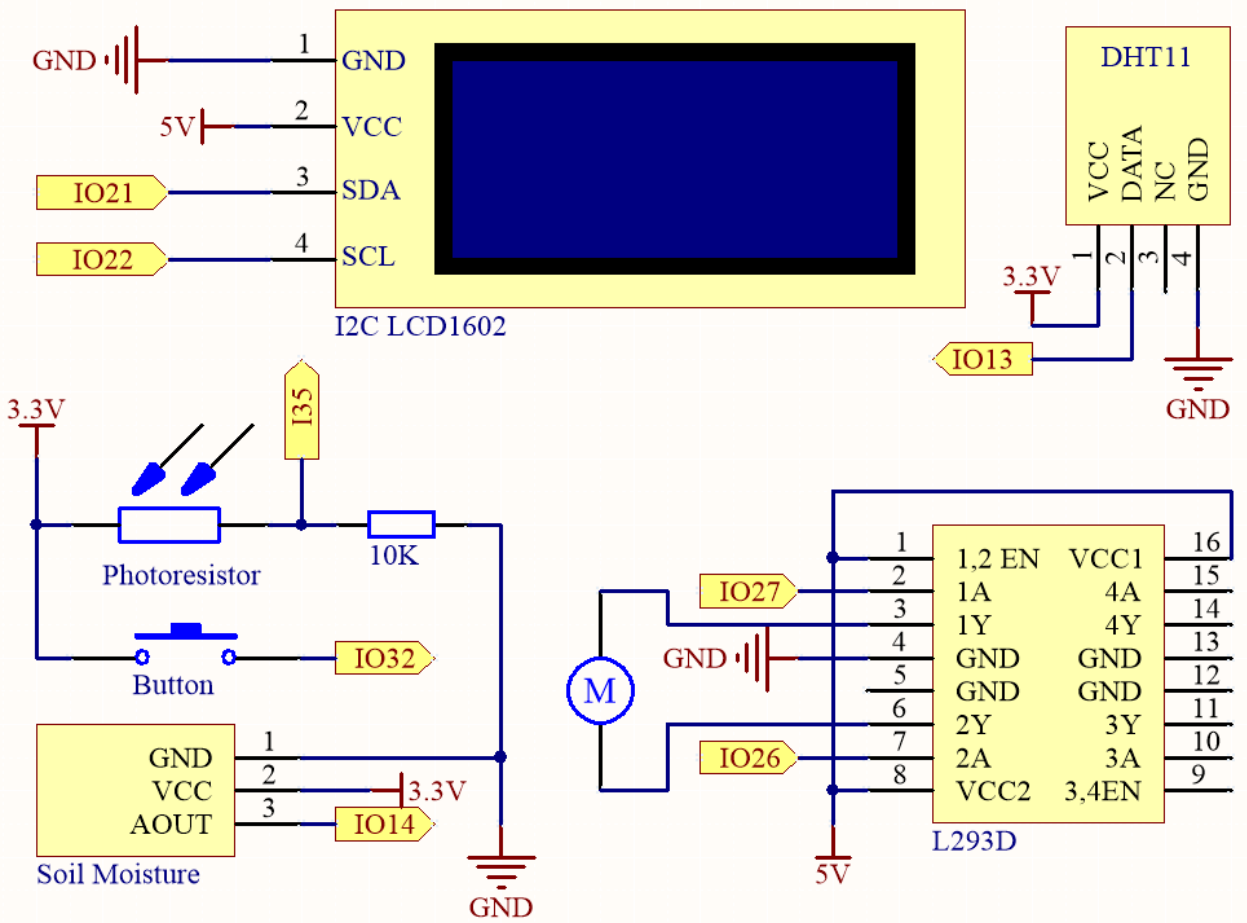
Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Sensor de Humedad y Temperatura DHT11	
I2C LCD1602	
Bomba Centrífuga	-
L293D	-
Botón	
Fotorresistor	
Resistor	
Módulo de Humedad del Suelo	

Esquemático

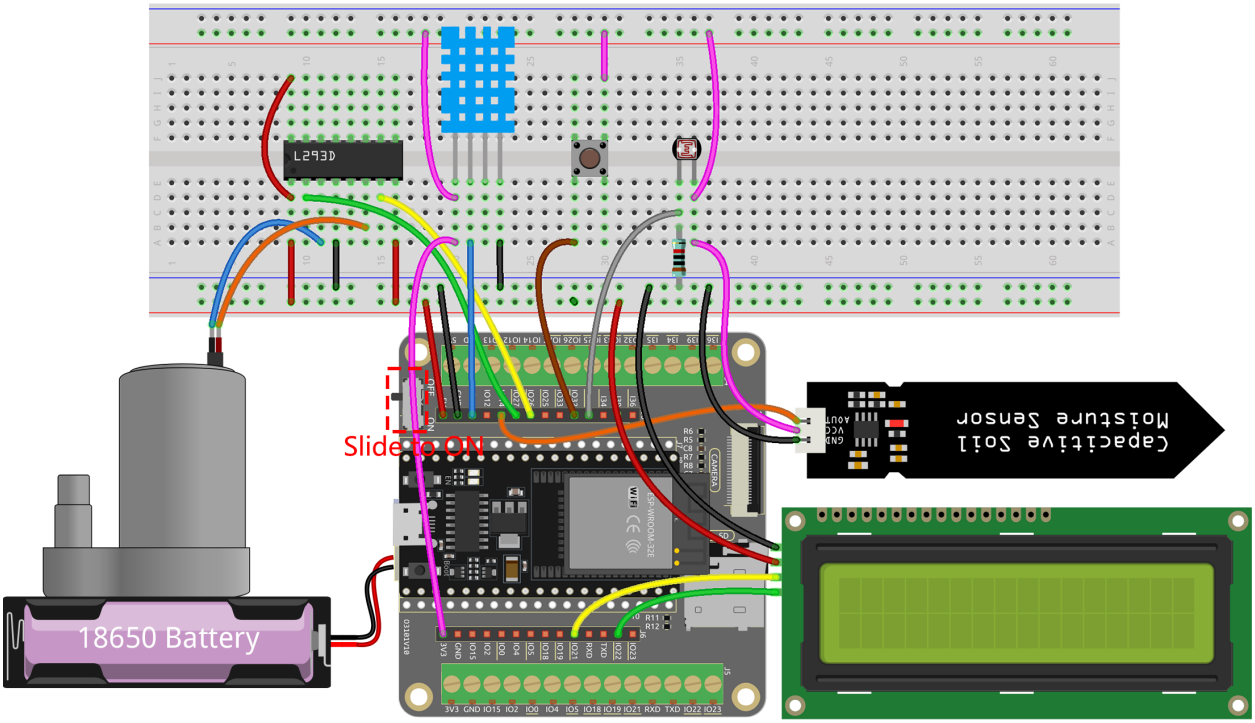


El sistema utiliza un sensor DHT11 para medir los niveles de temperatura y humedad del ambiente circundante. Mientras tanto, un módulo de humedad del suelo se utiliza para medir el nivel de humedad del suelo y un fotoresistor se utiliza para medir el nivel de luz. Las lecturas de estos sensores se muestran en una pantalla LCD, y una bomba de agua puede ser controlada usando un botón para regar la planta cuando sea necesario.

IO32 tiene una resistencia de pull-down interna de 1K, y por defecto, está a un nivel lógico bajo. Cuando se presiona el botón, se establece una conexión a VCC (alto voltaje), resultando en un nivel lógico alto en IO32.

Cableado

Nota: Aquí se recomienda insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de la batería.



Código

Nota:

- Puedes abrir el archivo `6.6_plant_monitor.ino` en la ruta `esp32-starter-kit-main\c\codes\6.6_plant_monitor`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*
- Las bibliotecas `LiquidCrystal_I2C` y `DHT sensor library` se utilizan aquí, puedes instalarlas desde el **Administrador de Bibliotecas**.

- Después de subir el código, el I2C LCD1602 muestra alternativamente la temperatura y la humedad, así como los valores analógicos de humedad del suelo e intensidad de la luz, con un intervalo de 2 segundos.
- La bomba de agua se controla mediante la presión de un botón. Para regar las plantas, mantén presionado el botón y suéltalo para detener el riego.

Nota: Si el código y el cableado son correctos, pero el LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte posterior para aumentar el contraste.

2.37 6.7 Adivina el Número

¿Te sientes con suerte? ¿Quieres probar tu intuición y ver si puedes adivinar el número correcto? ¡Entonces no busques más allá del juego Adivina el Número!

Con este proyecto, puedes jugar un juego divertido y emocionante de azar.

Usando un control remoto IR, los jugadores ingresan números entre 0 y 99 para intentar adivinar el número de punto de suerte generado aleatoriamente. El sistema muestra el número de entrada del jugador en una pantalla LCD, junto con consejos de límite superior e inferior para ayudar a guiar al jugador hacia la respuesta correcta. Con cada adivinanza, los jugadores se acercan al número de punto de suerte, hasta que finalmente, alguien acierta y gana el juego.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

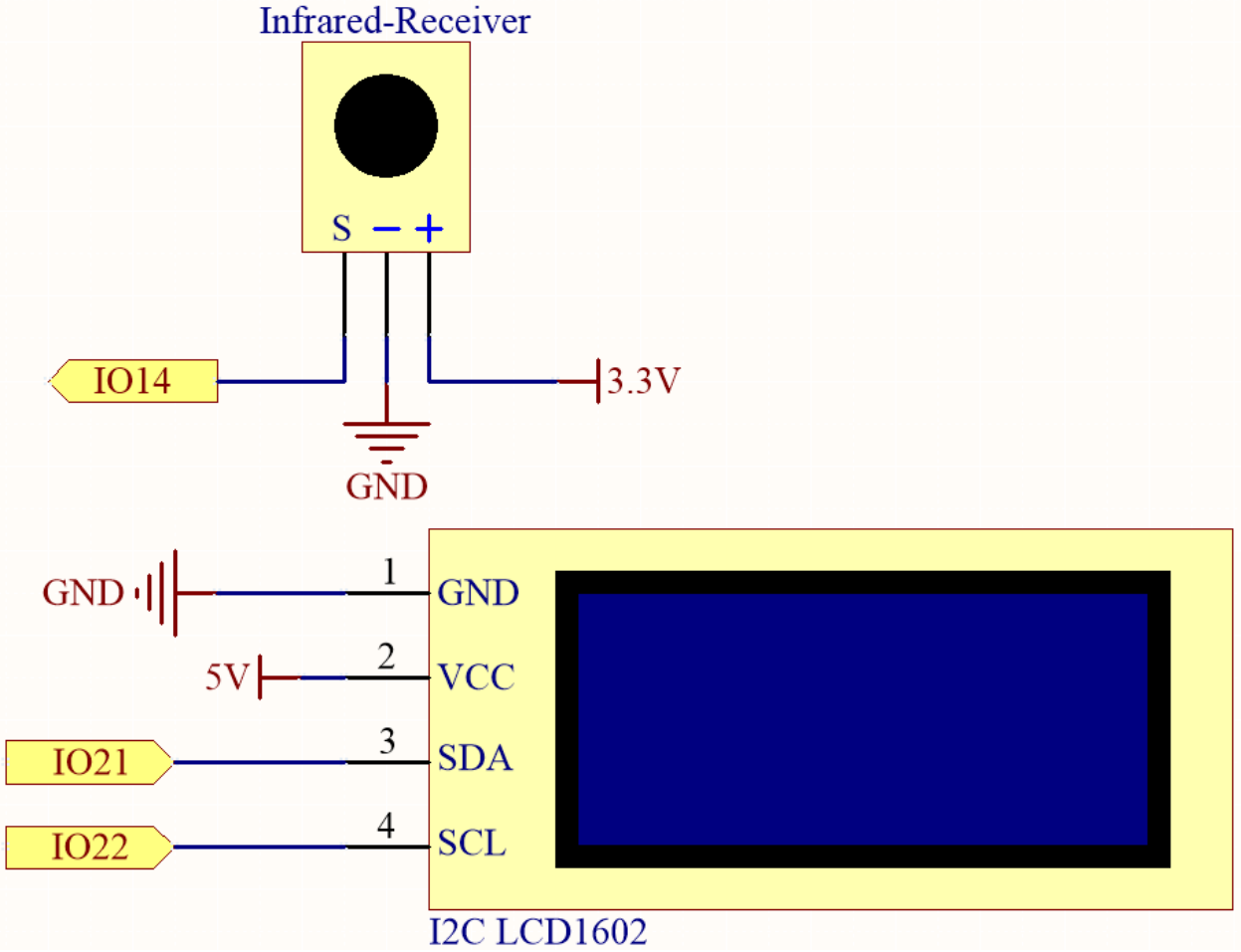
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

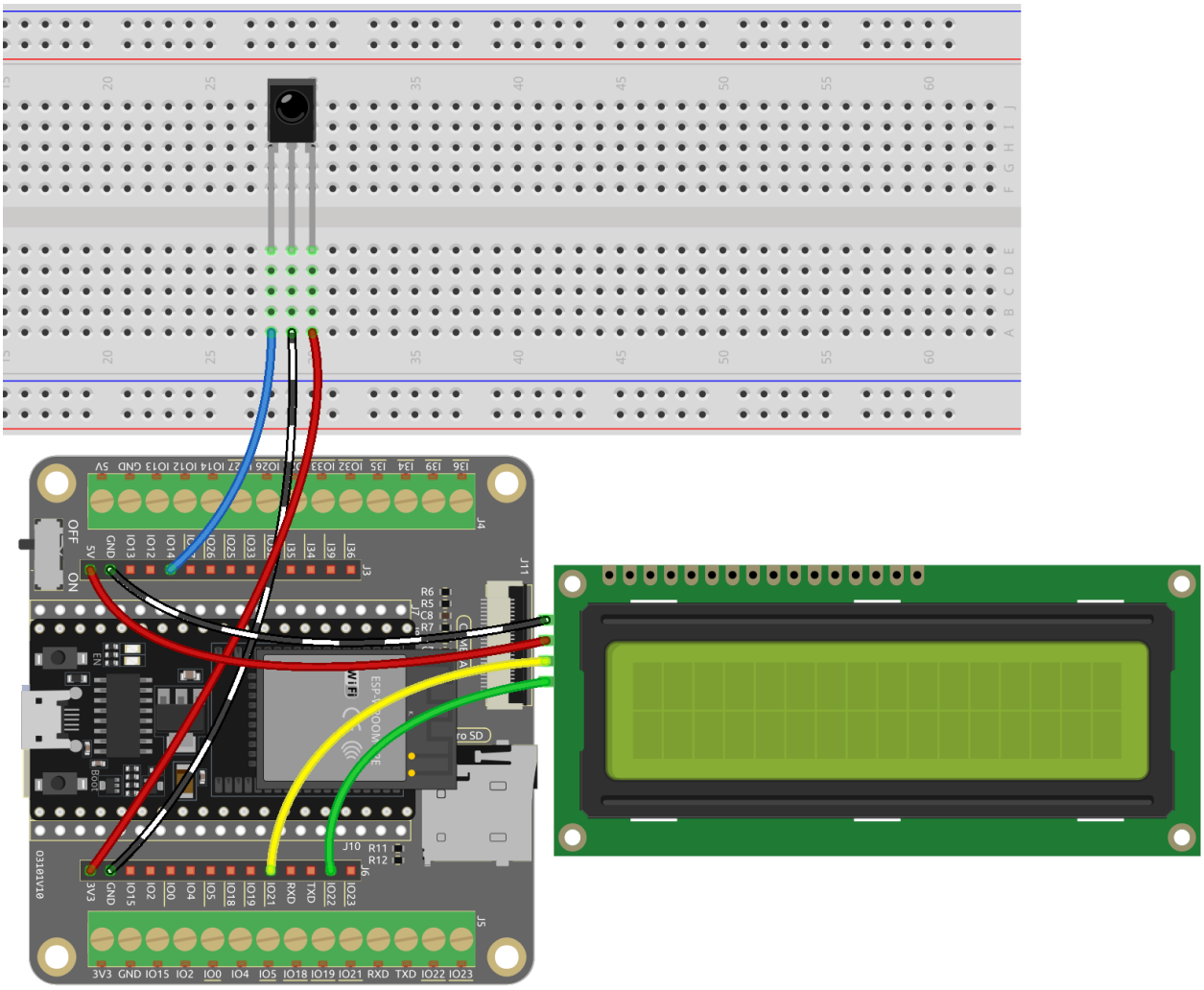
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Receptor de Infrarrojos	
I2C LCD1602	

Esquemático


Infrared-Receiver
S
IO14
3.3V
GND
GND
1
GND
5V
2
VCC
IO21
3
SDA
IO22
4
SCL
I2C LCD1602

Cableado



Código

Nota:

- Puedes abrir el archivo `6.7_guess_number.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\6.7_guess_number` directamente.
- Las bibliotecas `LiquidCrystal_I2C` y `IRremoteESP8266` se utilizan aquí, consulta *Instalación Manual* para obtener un tutorial de instalación.

- Después de que el código se haya cargado con éxito, presiona cualquier botón numérico en el control remoto para iniciar el juego.
- Ingresa un número usando los botones numéricos en el control remoto. Para ingresar un solo dígito, necesitas presionar la tecla **ciclo** para confirmar.
- El sistema mostrará el número de entrada y los consejos de límite superior e inferior en la pantalla LCD.
- Sigue adivinando hasta que adivines correctamente el número de punto de suerte.
- Después de una adivinanza exitosa, el sistema mostrará un mensaje de éxito y generará un nuevo número de punto de suerte.

Nota: Si el código y el cableado son correctos, pero la LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte posterior para aumentar el contraste.

¿Cómo funciona?

1. En la función `setup()`, la pantalla LCD I2C y el receptor IR se inicializan. Luego llama a la función `initNewValue()` para generar un nuevo número de suerte aleatorio, y se muestra un mensaje de bienvenida en la pantalla LCD.

```
void setup() {
    // Initialize the LCD screen
    lcd.init();
    lcd.backlight();

    // Start the serial communication
    Serial.begin(9600);

    // Enable the IR receiver
    irrecv.enableIRIn();

    // Initialize a new lucky point value
    initNewValue();
}
```

2. En la función `loop`, el código espera una señal del receptor IR. Cuando se recibe una señal, la función `decodeKeyValue` se llama para decodificar la señal y obtener el valor del botón correspondiente.

```
void loop() {
// If a signal is received from the IR receiver
if (irrecv.decode(&results)) {
    bool result = 0;
    String num = decodeKeyValue(results.value);

    // If the POWER button is pressed
    if (num == "POWER") {
        initNewValue(); // Initialize a new lucky point value
    }

    // If the CYCLE button is pressed
    else if (num == "CYCLE") {
        result = detectPoint(); // Detect the input number
        lcdShowInput(result); // Show the result on the LCD screen
    }

    // If a number button (0-9) is pressed,
    //add the digit to the input number
    //and detect the number if it is greater than or equal to 10
    else if (num >= "0" && num <= "9") {
        count = count * 10;
        count += num.toInt();
        if (count >= 10) {
            result = detectPoint();
        }
```

(continué en la próxima página)

(proviene de la página anterior)

```
        lcdShowInput(result);
    }
    irrecv.resume();
}
}
```

- Dependiendo del valor del botón, se llama a la función apropiada. Si se presiona un botón numérico, la variable `count` se actualiza, y la función `detectPoint` se llama para detectar si el número de entrada es correcto. La función `lcdShowInput` se llama para mostrar el número de entrada y los consejos de límite superior e inferior en la pantalla LCD.
- Si se presiona el botón `POWER`, la función `initNewValue` se llama para generar un nuevo número de punto de suerte y mostrar el mensaje de bienvenida en la pantalla LCD.
- Si se presiona el botón `CYCLE`, la función `detectPoint` se llama para detectar si el número de entrada es correcto. La función `lcdShowInput` se llama para mostrar el número de entrada y los consejos de límite superior e inferior en la pantalla LCD.

7. Bluetooth&Tarjeta SD&Cámara&Altavoz

2.38 7.1 Bluetooth

Este proyecto proporciona una guía para desarrollar una aplicación simple de comunicación serial Bluetooth Low Energy (BLE) utilizando el microcontrolador ESP32. El ESP32 es un microcontrolador potente que integra conectividad Wi-Fi y Bluetooth, lo que lo hace un candidato ideal para desarrollar aplicaciones inalámbricas. BLE es un protocolo de comunicación inalámbrica de baja potencia diseñado para la comunicación de corto alcance. Este documento cubrirá los pasos para configurar el ESP32 para actuar como un servidor BLE y comunicarse con un cliente BLE a través de una conexión serial.

Acerca de la Función Bluetooth

El ESP32 WROOM 32E es un módulo que integra conectividad Wi-Fi y Bluetooth en un solo chip. Soporta los protocolos Bluetooth Low Energy (BLE) y Bluetooth Clásico.

El módulo puede usarse como cliente o servidor Bluetooth. Como cliente Bluetooth, el módulo puede conectarse a otros dispositivos Bluetooth e intercambiar datos con ellos. Como servidor Bluetooth, el módulo puede proporcionar servicios a otros dispositivos Bluetooth.

El ESP32 WROOM 32E soporta varios perfiles Bluetooth, incluyendo el Perfil de Acceso Genérico (GAP), Perfil de Atributo Genérico (GATT) y Perfil de Puerto Serie (SPP). El perfil SPP permite que el módulo emule un puerto serie a través de Bluetooth, habilitando la comunicación serial con otros dispositivos Bluetooth.

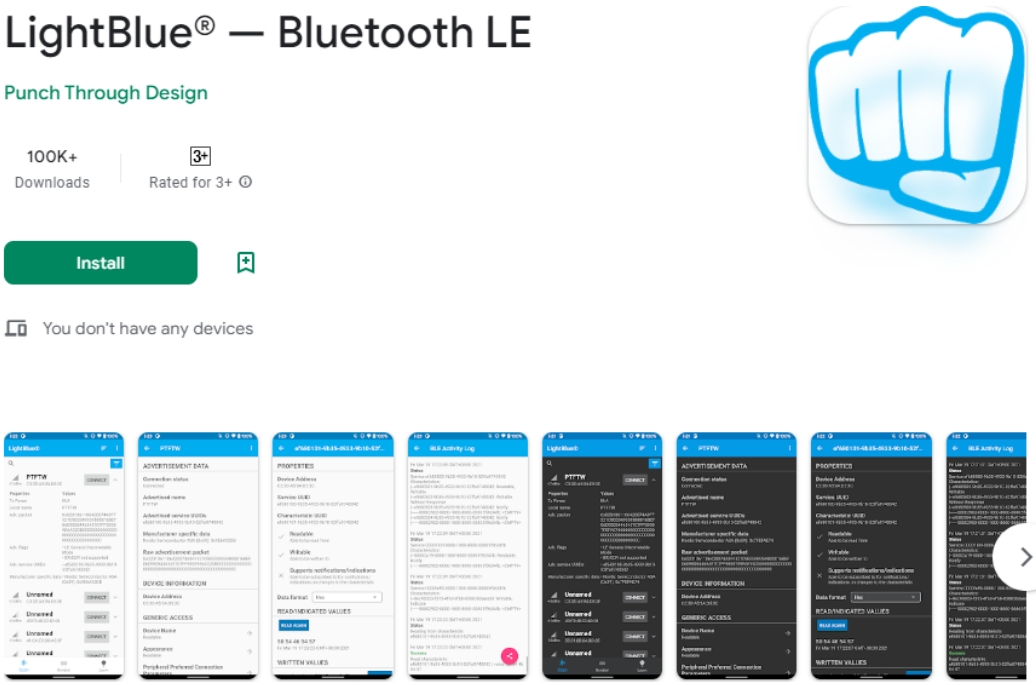
Para usar la función Bluetooth del ESP32 WROOM 32E, necesitas programarlo usando un Kit de Desarrollo de Software (SDK) apropiado o usando el IDE de Arduino con la biblioteca BLE de ESP32. La biblioteca BLE de ESP32 proporciona una interfaz de alto nivel para trabajar con BLE. Incluye ejemplos que demuestran cómo usar el módulo como cliente y servidor BLE.

En general, la función Bluetooth del ESP32 WROOM 32E proporciona una forma conveniente y de baja potencia para habilitar la comunicación inalámbrica en tus proyectos.

Pasos Operativos

Aquí están las instrucciones paso a paso para configurar la comunicación Bluetooth entre tu ESP32 y un dispositivo móvil usando la aplicación LightBlue:

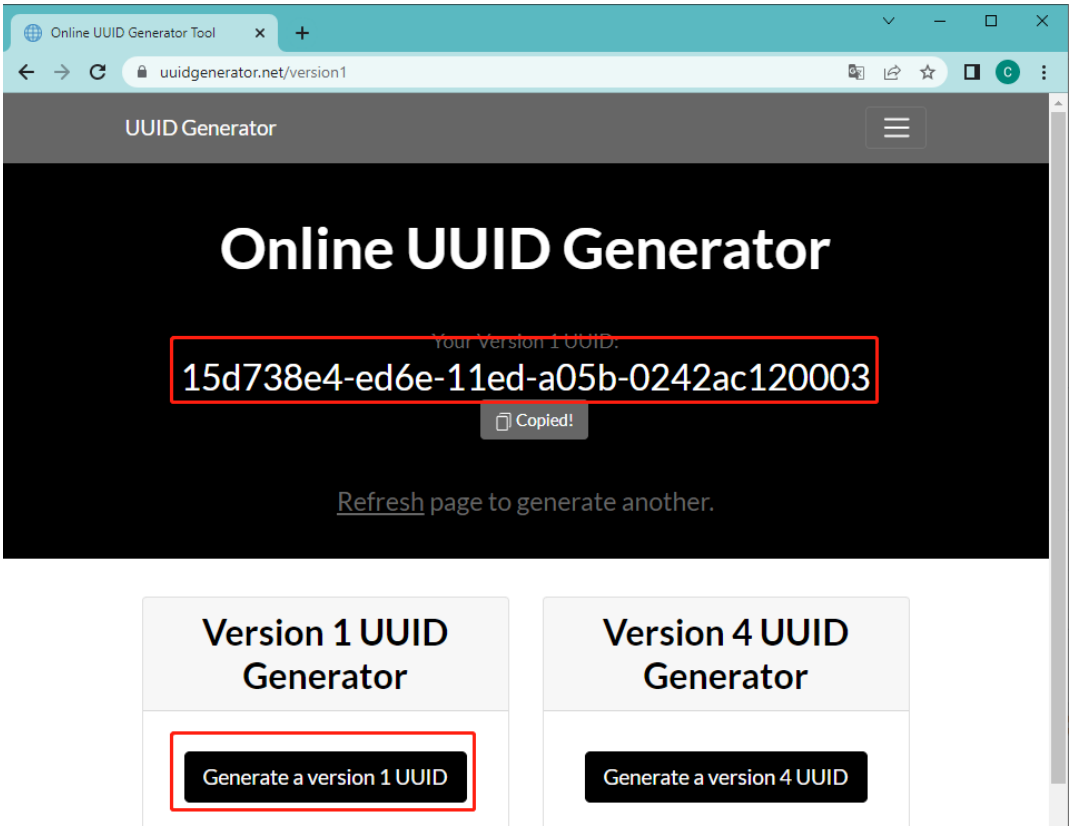
1. Descarga la aplicación LightBlue desde el **App Store** (para iOS) o **Google Play** (para Android).



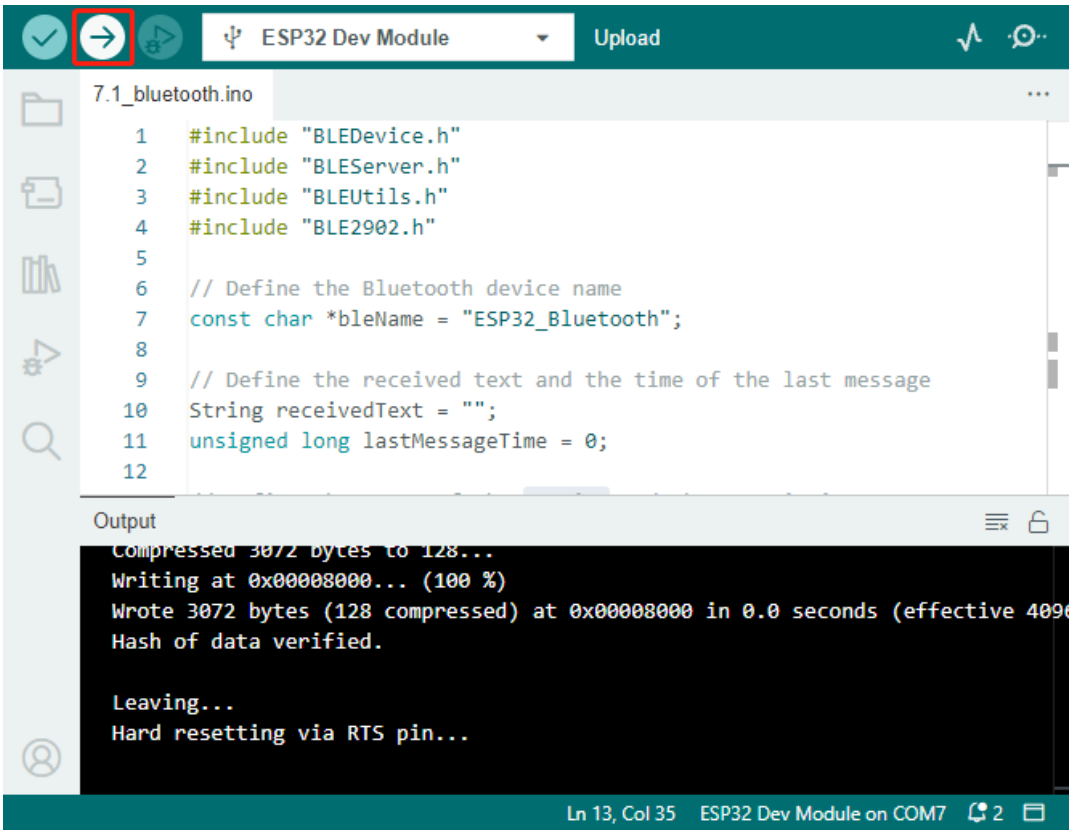
2. Abre el archivo `7.1_bluetooth.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\7.1_bluetooth`, o copia el código en el IDE de Arduino.

3. Para evitar conflictos de UUID, se recomienda generar aleatoriamente tres nuevos UUIDs usando el , y rellenarlos en las siguientes líneas de código.

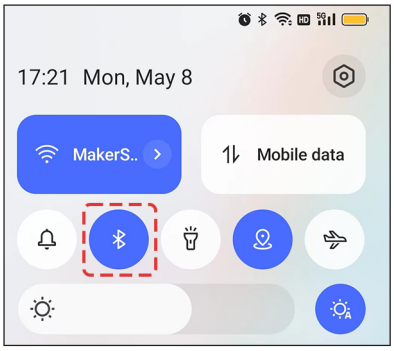
```
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"
```



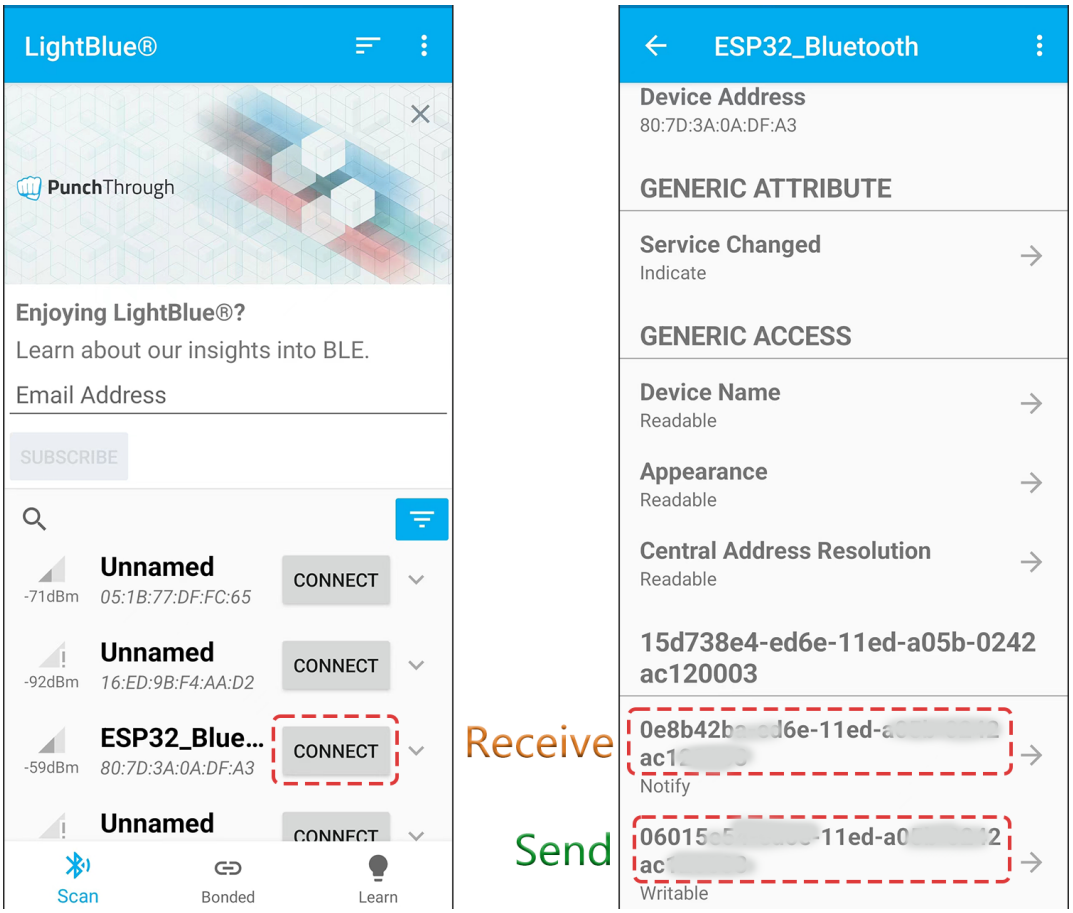
4. Selecciona la placa y el puerto correctos, luego haz clic en el botón **Subir**.



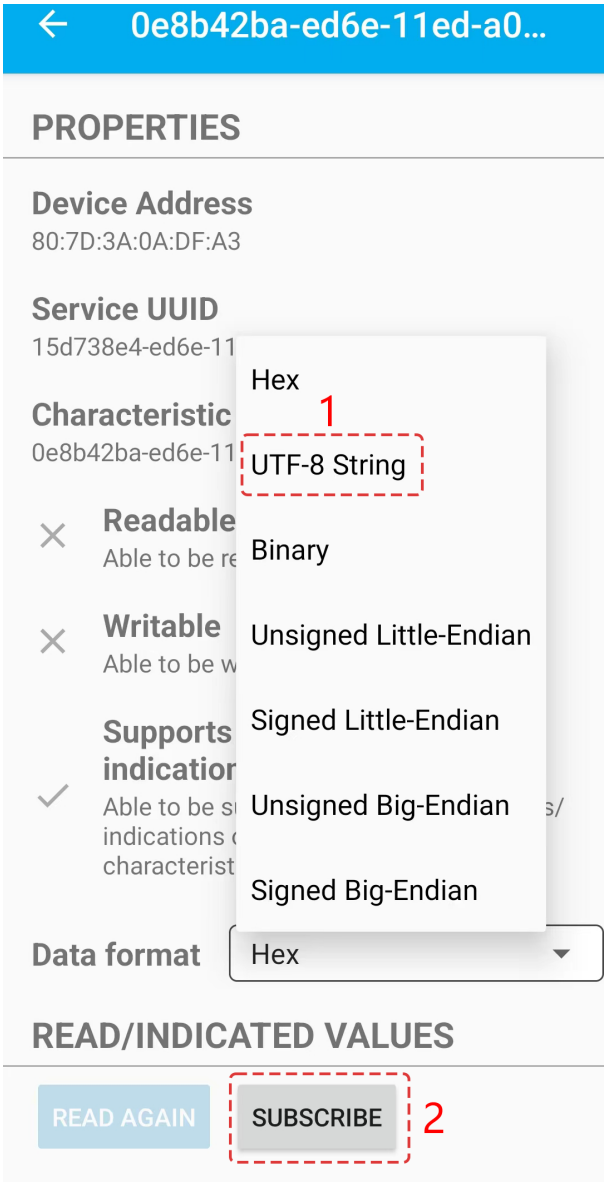
5. Después de que el código se haya cargado con éxito, activa el **Bluetooth** en tu dispositivo móvil y abre la aplicación **LightBlue**.



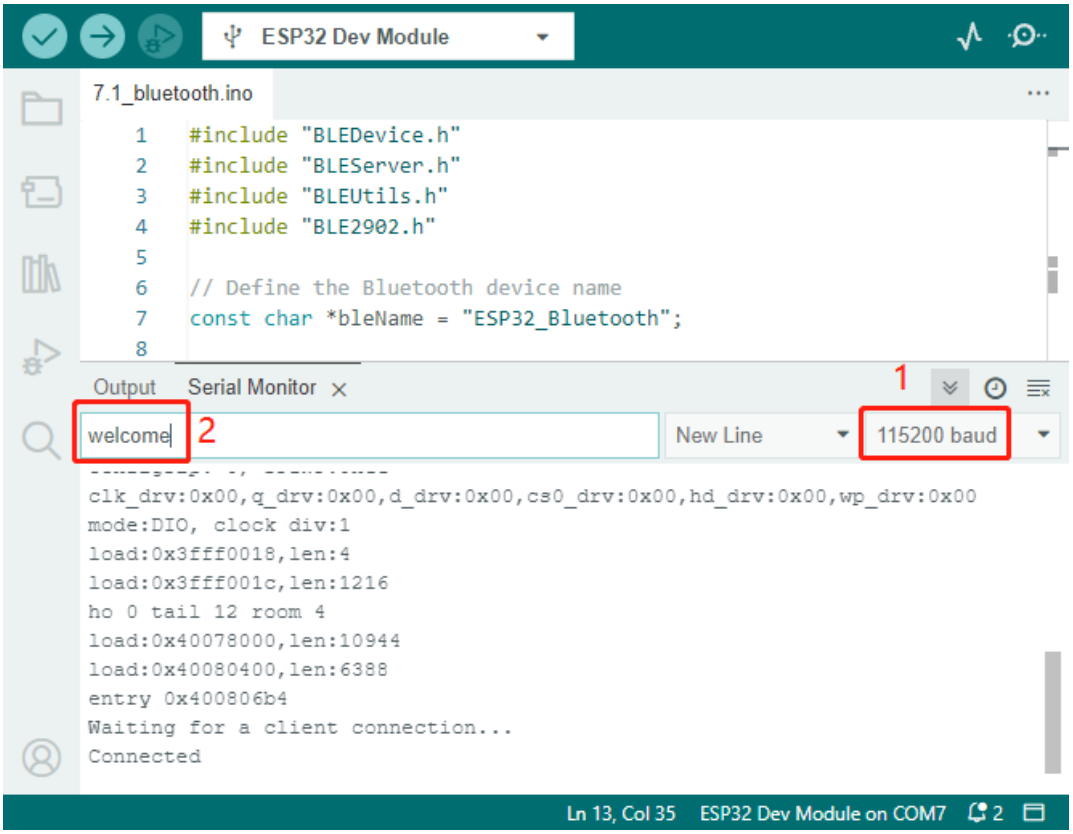
6. En la página de **Scan**, encuentra **ESP32-Bluetooth** y haz clic en **CONECTAR**. Si no lo ves, intenta actualizar la página unas cuantas veces. Cuando aparezca **«Conectado al dispositivo!»**, la conexión Bluetooth es exitosa. Desplázate hacia abajo para ver los tres UUIDs configurados en el código.



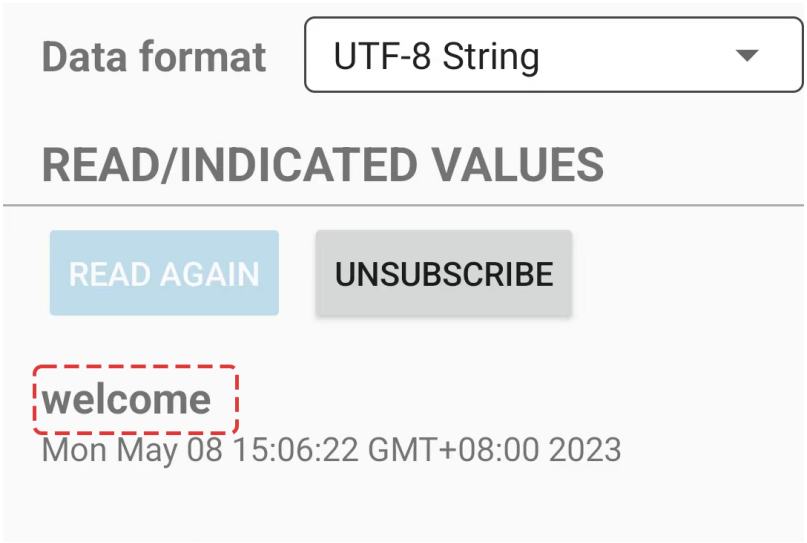
7. Haz clic en el UUID de **Recibir**. Selecciona el formato de datos apropiado en el cuadro a la derecha de **Formato de Datos**, como «HEX» para hexadecimal, «Cadena UTF-8» para carácter o «Binario» para binario, etc. Luego haz clic en **SUSCRIBIRSE**.



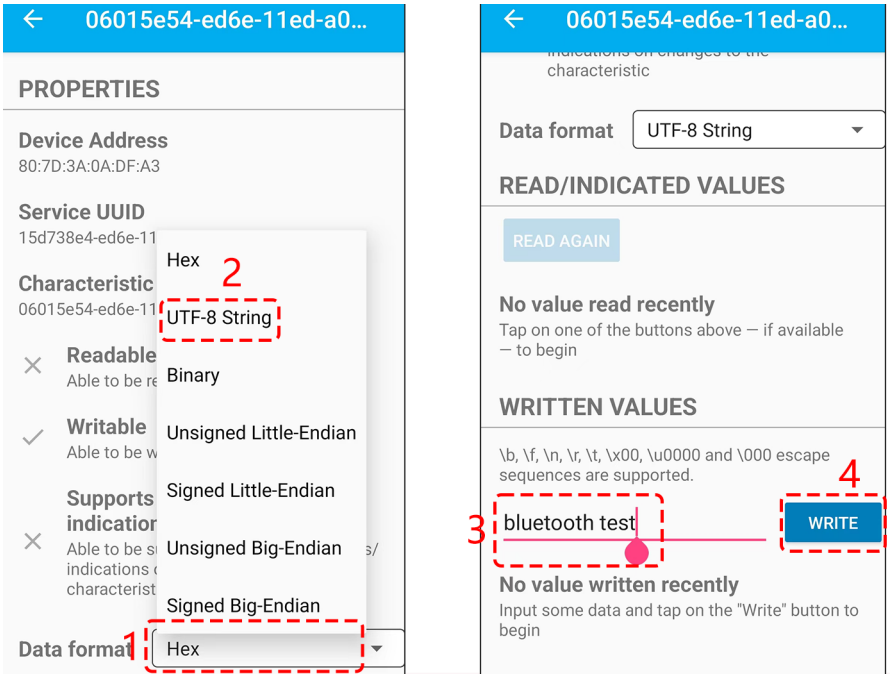
8. Regresa al IDE de Arduino, abre el Monitor Serial, establece la tasa de baudios a 115200, luego escribe «welcome» y presiona Enter.



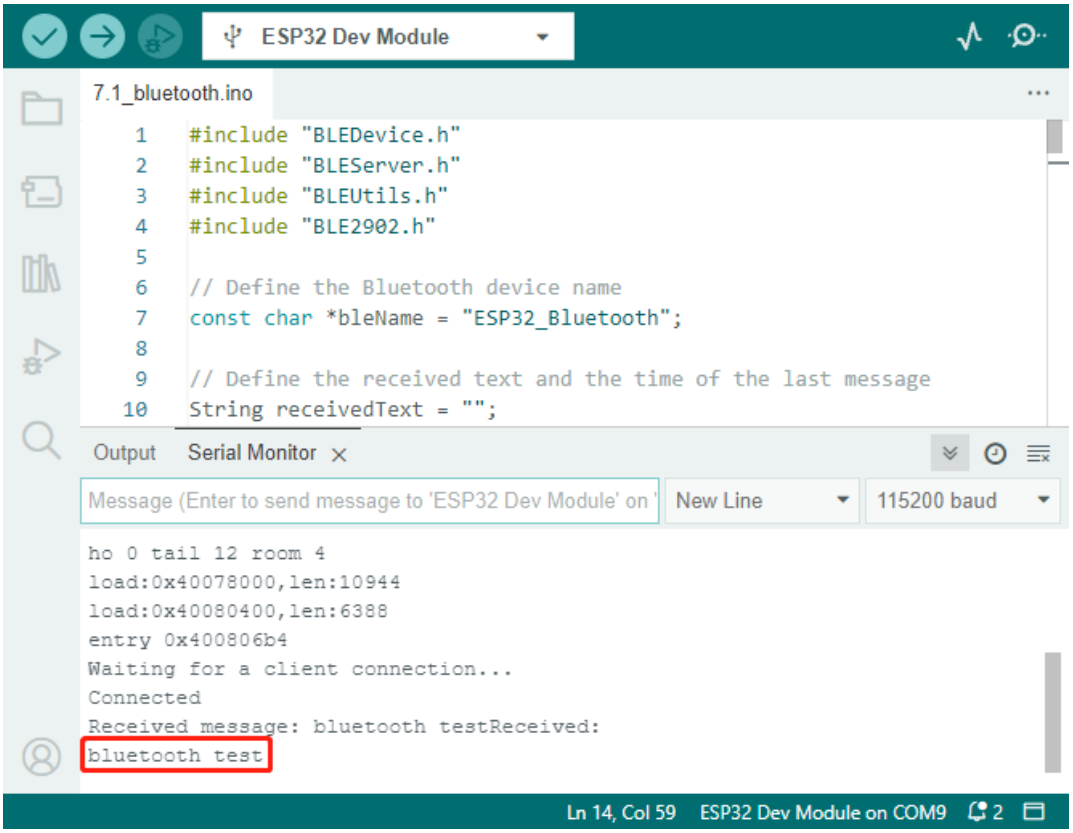
9. Ahora deberías ver el mensaje «welcome» en la aplicación LightBlue.



10. Para enviar información desde el dispositivo móvil al Monitor Serial, haz clic en el UUID de **Enviar**, establece el formato de datos a «Cadena UTF-8» y escribe un mensaje.



11. Deberías ver el mensaje en el Monitor Serial.



¿Cómo funciona?

Este código de Arduino está escrito para el microcontrolador ESP32 y lo configura para comunicarse con un dispositivo Bluetooth Low Energy (BLE).

El siguiente es un resumen breve del código:

- **Incluir las bibliotecas necesarias**: El código comienza incluyendo las bibliotecas necesarias para trabajar con Bluetooth Low Energy (BLE) en el ESP32.

```
#include "BLEDevice.h"
#include "BLEServer.h"
#include "BLEUtils.h"
#include "BLE2902.h"
```

- **Variables Globales**: El código define un conjunto de variables globales incluyendo el nombre del dispositivo Bluetooth (`bleName`), variables para llevar el seguimiento del texto recibido y el tiempo del último mensaje, UUIDs para el servicio y las características, y un objeto `BLECharacteristic` (`pCharacteristic`).

```
// Define the Bluetooth device name
const char *bleName = "ESP32_Bluetooth";

// Define the received text and the time of the last message
String receivedText = "";
unsigned long lastMessageTime = 0;

// Define the UUIDs of the service and characteristics
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"

// Define the Bluetooth characteristic
BLECharacteristic *pCharacteristic;
```

- **Configuración**: En la función `setup()`, se inicializa el puerto serial con una tasa de baudios de 115200 y se llama a la función `setupBLE()` para configurar el Bluetooth BLE.

```
void setup() {
    Serial.begin(115200);  // Initialize the serial port
    setupBLE();            // Initialize the Bluetooth BLE
}
```

- **Bucle Principal**: En la función `loop()`, si se recibió una cadena a través de BLE (es decir, `receivedText` no está vacío) y ha pasado al menos 1 segundo desde el último mensaje, el código imprime la cadena recibida en el monitor serial, establece el valor de la característica a la cadena recibida, envía una notificación y luego borra la cadena recibida. Si hay datos disponibles en el puerto serial, lee la cadena hasta encontrar un carácter de nueva línea, establece el valor de la característica a esta cadena y envía una notificación.

```
void loop() {
    // When the received text is not empty and the time since the last␣
↪message is over 1 second
    // Send a notification and print the received text
    if (receivedText.length() > 0 && millis() - lastMessageTime > 1000) {
        Serial.print("Received message: ");
        Serial.println(receivedText);
        pCharacteristic->setValue(receivedText.c_str());
        pCharacteristic->notify();
        receivedText = "";
    }
```

(continué en la próxima página)

(proviene de la página anterior)

```
    // Read data from the serial port and send it to BLE characteristic
    if (Serial.available() > 0) {
        String str = Serial.readStringUntil('\n');
        const char *newValue = str.c_str();
        pCharacteristic->setValue(newValue);
        pCharacteristic->notify();
    }
}
```

- **Callbacks**: Se definen dos clases de callbacks (`MyServerCallbacks` y `MyCharacteristicCallbacks`) para manejar eventos relacionados con la comunicación Bluetooth. `MyServerCallbacks` se usa para manejar eventos relacionados con el estado de conexión (conectado o desconectado) del servidor BLE. `MyCharacteristicCallbacks` se usa para manejar eventos de escritura en la característica BLE, es decir, cuando un dispositivo conectado envía una cadena al ESP32 a través de BLE, se captura y almacena en `receivedText`, y se registra el tiempo actual en `lastMessageTime`.

```
// Define the BLE server callbacks
class MyServerCallbacks : public BLEServerCallbacks {
    // Print the connection message when a client is connected
    void onConnect(BLEServer *pServer) {
    Serial.println("Connected");
    }
    // Print the disconnection message when a client is disconnected
    void onDisconnect(BLEServer *pServer) {
    Serial.println("Disconnected");
    }
};

// Define the BLE characteristic callbacks
class MyCharacteristicCallbacks : public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        // When data is received, get the data and save it to receivedText,␣
↪and record the time
        std::string value = pCharacteristic->getValue();
        receivedText = String(value.c_str());
        lastMessageTime = millis();
        Serial.print("Received: ");
        Serial.println(receivedText);
    }
};
```

- **Configurar BLE**: En la función `setupBLE()`, el dispositivo BLE y el servidor se inicializan, se establecen los callbacks del servidor, se crea el servicio BLE usando el UUID definido, se crean y añaden al servicio las características para enviar notificaciones y recibir datos, y se establecen los callbacks de la característica. Finalmente, el servicio se inicia y el servidor comienza a anunciarse.

```
// Initialize the Bluetooth BLE
void setupBLE() {
    BLEDevice::init(bleName);                          // Initialize the BLE␣
↪device
    BLEServer *pServer = BLEDevice::createServer();  // Create the BLE␣
↪server
```

(continué en la próxima página)

(proviene de la página anterior)

```
    // Print the error message if the BLE server creation fails
    if (pServer == nullptr) {
        Serial.println("Error creating BLE server");
        return;
    }
    pServer->setCallbacks(new MyServerCallbacks());  // Set the BLE server␣
↪callbacks

    // Create the BLE service
    BLEService *pService = pServer->createService(SERVICE_UUID);
    // Print the error message if the BLE service creation fails
    if (pService == nullptr) {
        Serial.println("Error creating BLE service");
        return;
    }
    // Create the BLE characteristic for sending notifications
    pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
↪ BLECharacteristic::PROPERTY_NOTIFY);
    pCharacteristic->addDecodeor(new BLE2902());  // Add the decodeor
    // Create the BLE characteristic for receiving data
    BLECharacteristic *pCharacteristicRX = pService->
↪createCharacteristic(CHARACTERISTIC_UUID_RX, BLECharacteristic::PROPERTY_
↪WRITE);
    pCharacteristicRX->setCallbacks(new MyCharacteristicCallbacks());  //␣
↪Set the BLE characteristic callbacks
    pService->start();                                                 //␣
↪Start the BLE service
    pServer->getAdvertising()->start();                                //␣
↪Start advertising
    Serial.println("Waiting for a client connection...");              //␣
↪Wait for a client connection
}
```

Ten en cuenta que este código permite la comunicación bidireccional: puede enviar y recibir datos a través de BLE. Sin embargo, para interactuar con hardware específico como encender/apagar un LED, se debe agregar código adicional para procesar las cadenas recibidas y actuar en consecuencia.

2.39 7.2 Control de LED RGB por Bluetooth

Este proyecto es una extensión de un proyecto anterior(*7.1 Bluetooth*), añadiendo configuraciones de LED RGB y comandos personalizados como «led_off», «red», «green», etc. Estos comandos permiten controlar el LED RGB enviando comandos desde un dispositivo móvil usando LightBlue.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

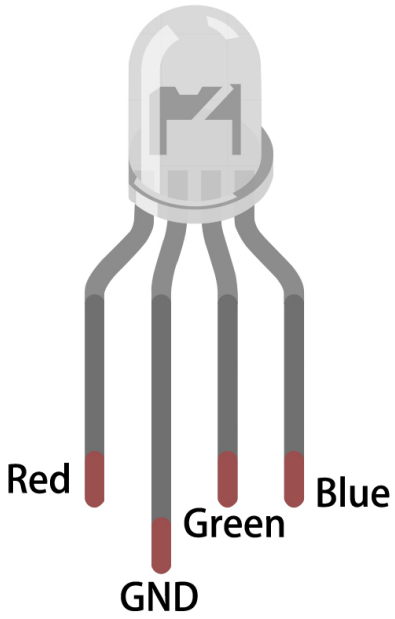
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

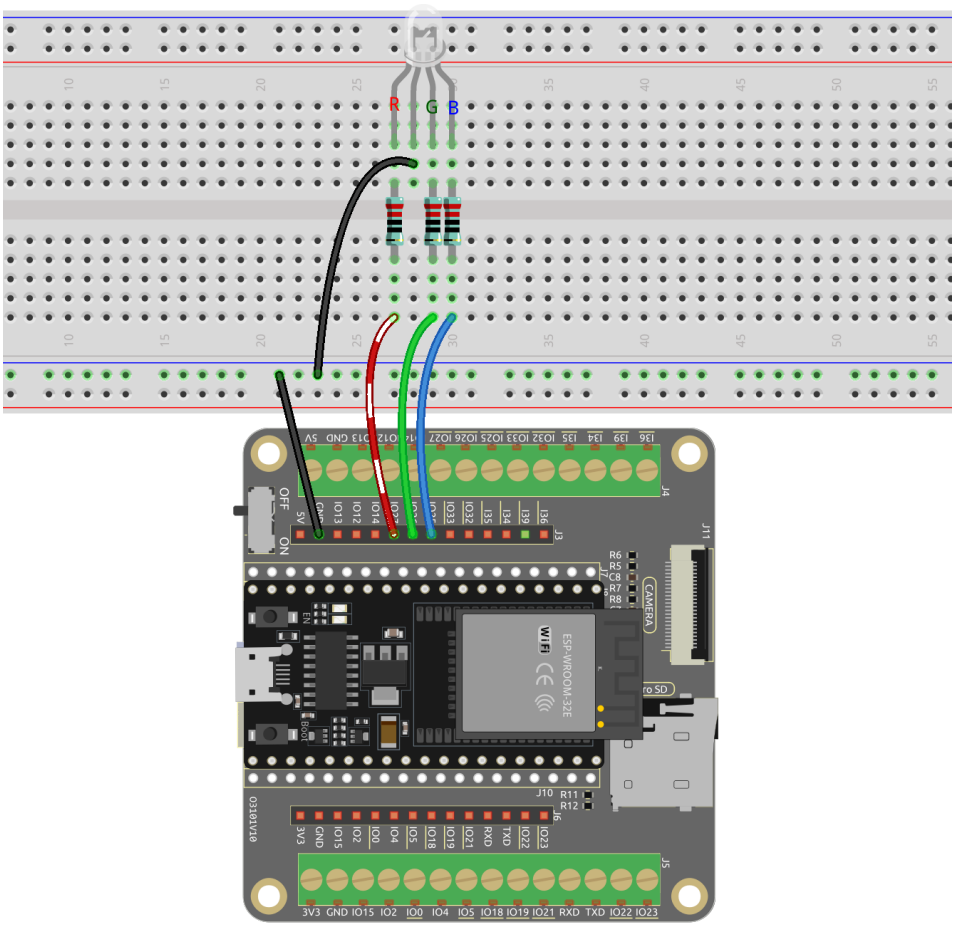
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED RGB	

Pasos Operativos

1. Construye el circuito.



El LED RGB tiene 4 pines: el pin largo es el pin común cátodo, que generalmente se conecta a GND; el pin izquierdo junto al pin más largo es Rojo; y los dos pines a la derecha son Verde y Azul.

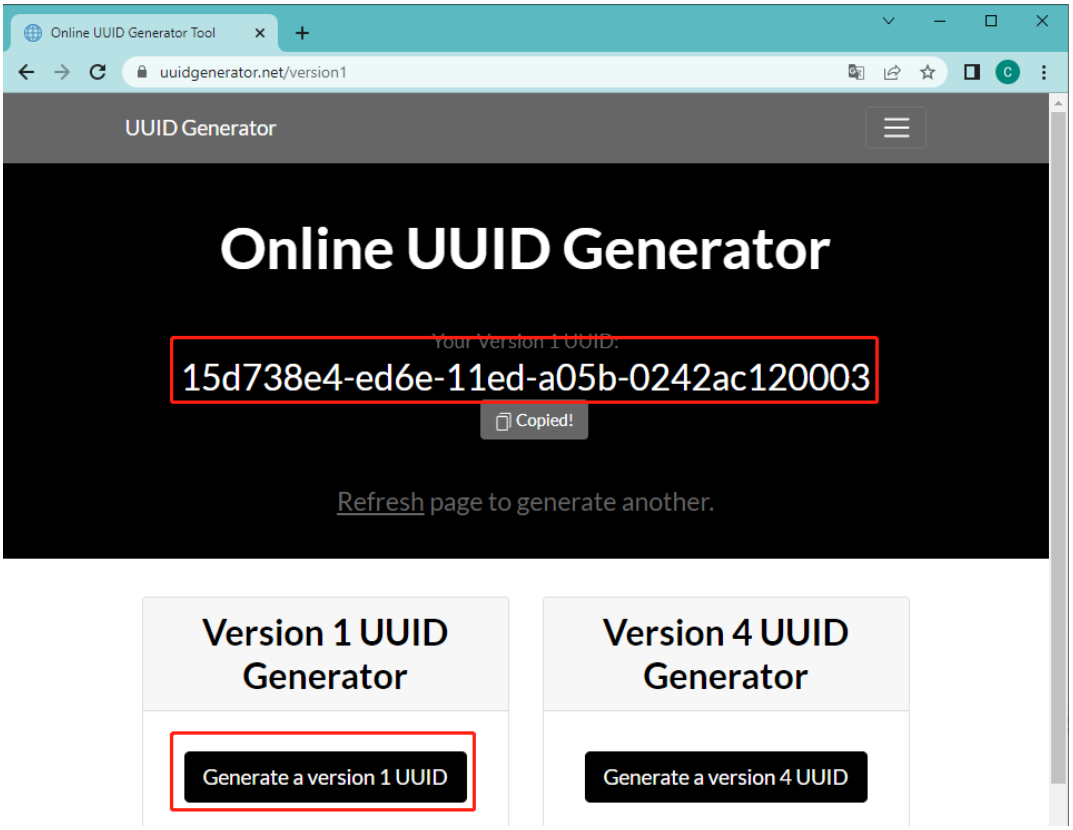


2. Abre el archivo `7.2_bluetooth_rgb_led.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\7.2_bluetooth_rgb_led`, o copia el código en el IDE de Arduino.

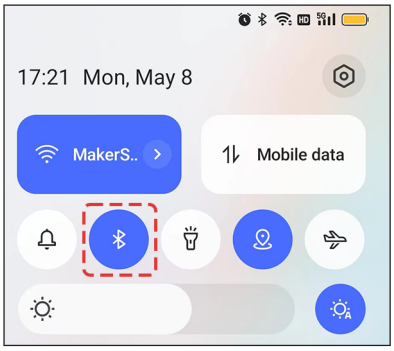
3. Para evitar conflictos de UUID, se recomienda generar aleatoriamente tres nuevos UUIDs usando el proporcionado por el Bluetooth SIG, y rellenarlos en las siguientes líneas de código.

Nota: Si ya has generado tres nuevos UUIDs en el proyecto *7.1 Bluetooth*, entonces puedes continuar usándolos.

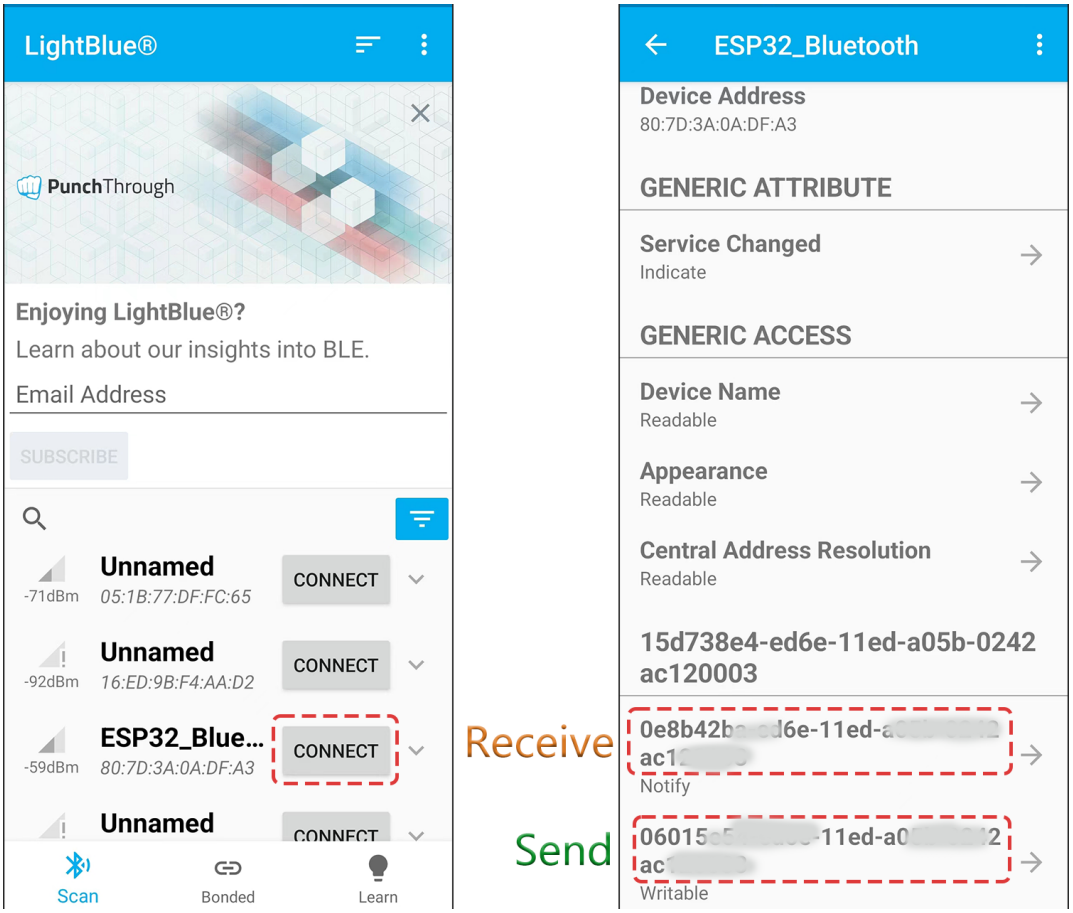
```
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"
```



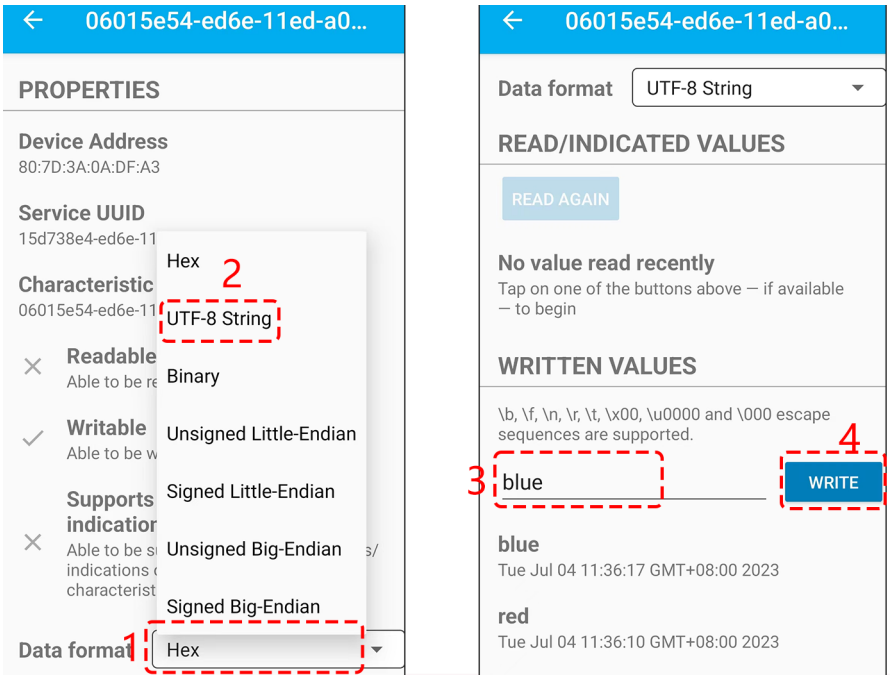
4. Selecciona la placa y el puerto correctos, luego haz clic en el botón **Subir**.
5. Después de que el código se haya cargado con éxito, activa el **Bluetooth** en tu dispositivo móvil y abre la aplicación **LightBlue**.



6. En la página de **Scan**, busca **ESP32-Bluetooth** y haz clic en **CONNECTAR**. Si no lo ves, intenta actualizar la página unas cuantas veces. Cuando aparezca **«Conectado al dispositivo!»**, la conexión Bluetooth es exitosa. Desplázate hacia abajo para ver los tres UUIDs configurados en el código.



7. Toca el UUID de envío, luego configura el formato de datos a «Cadena UTF-8». Ahora puedes escribir estos comandos: «led_off», «red», «green», «blue», «yellow» y «purple» para ver si el LED RGB responde a estas instrucciones.



¿Cómo funciona?

Este código es una extensión de un proyecto anterior(*7.1 Bluetooth*), añadiendo configuraciones de LED RGB y comandos personalizados como «led_off», «red», «green», etc. Estos comandos permiten controlar el LED RGB enviando comandos desde un dispositivo móvil usando LightBlue.

Desglosemos el código paso a paso:

- Añadir nuevas variables globales para los pines del LED RGB, canales PWM, frecuencia y resolución.

```
...

// Define RGB LED pins
const int redPin = 27;
const int greenPin = 26;
const int bluePin = 25;

// Define PWM channels
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;

...
```

- Dentro de la función `setup()`, los canales PWM se inicializan con la frecuencia y resolución predefinidas. Los pines del LED RGB se conectan luego a sus respectivos canales PWM.

```
void setup() {
    ...

    // Set up PWM channels
    ledcSetup(redChannel, freq, resolution);
    ledcSetup(greenChannel, freq, resolution);
    ledcSetup(blueChannel, freq, resolution);

    // Attach pins to corresponding PWM channels
    ledcAttachPin(redPin, redChannel);
    ledcAttachPin(greenPin, greenChannel);
    ledcAttachPin(bluePin, blueChannel);

}
```

- Modificar el método `onWrite` en la clase `MyCharacteristicCallbacks`. Esta función escucha los datos que llegan de la conexión Bluetooth. Basándose en la cadena recibida (como `"led_off"`, `"red"`, `"green"`, etc.), controla el LED RGB.

```
// Define the BLE characteristic callbacks
class MyCharacteristicCallbacks : public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        std::string value = pCharacteristic->getValue();
        if (value == "led_off") {
            setColor(0, 0, 0); // turn the RGB LED off
            Serial.println("RGB LED turned off");
        } else if (value == "red") {
            setColor(255, 0, 0); // Red
```

(continué en la próxima página)

(proviene de la página anterior)

```
            Serial.println("red");
        }
        else if (value == "green") {
            setColor(0, 255, 0); // green
            Serial.println("green");
        }
        else if (value == "blue") {
            setColor(0, 0, 255); // blue
            Serial.println("blue");
        }
        else if (value == "yellow") {
            setColor(255, 150, 0); // yellow
            Serial.println("yellow");
        }
        else if (value == "purple") {
            setColor(80, 0, 80); // purple
            Serial.println("purple");
        }
    }
};
```

- Finalmente, se añade una función para establecer el color del LED RGB.

```
void setColor(int red, int green, int blue) {
    // For common-anode RGB LEDs, use 255 minus the color value
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

En resumen, este script habilita un modelo de interacción de control remoto, donde el ESP32 opera como un servidor de Energía Baja de Bluetooth (BLE).

El cliente BLE conectado (como un smartphone) puede enviar comandos de cadena para cambiar el color de un LED RGB. El ESP32 también da retroalimentación al cliente enviando de vuelta la cadena recibida, permitiendo al cliente saber qué operación se realizó.

2.40 7.3 Reproductor de Audio Bluetooth

El objetivo del proyecto es proporcionar una solución simple para reproducir audio desde un dispositivo habilitado para Bluetooth usando el DAC interno del ESP32.

El proyecto involucra el uso de la biblioteca `ESP32-A2DP` para recibir datos de audio desde un dispositivo habilitado para Bluetooth. Los datos de audio recibidos se transmiten al DAC interno del ESP32 usando la interfaz I2S. La interfaz I2S está configurada para operar en modo maestro, modo de transmisión y modo DAC integrado. Los datos de audio se reproducen luego a través del altavoz conectado al DAC.

Al usar el DAC interno del ESP32, es importante tener en cuenta que el nivel de voltaje de salida está limitado a 1.1V. Por lo tanto, se recomienda usar un amplificador externo para aumentar el nivel de voltaje de salida al nivel deseado. También es importante asegurar que los datos de audio estén en el formato y la tasa de muestreo correctos para evitar distorsiones o ruidos durante la reproducción.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Módulo de Audio y Altavoz	-

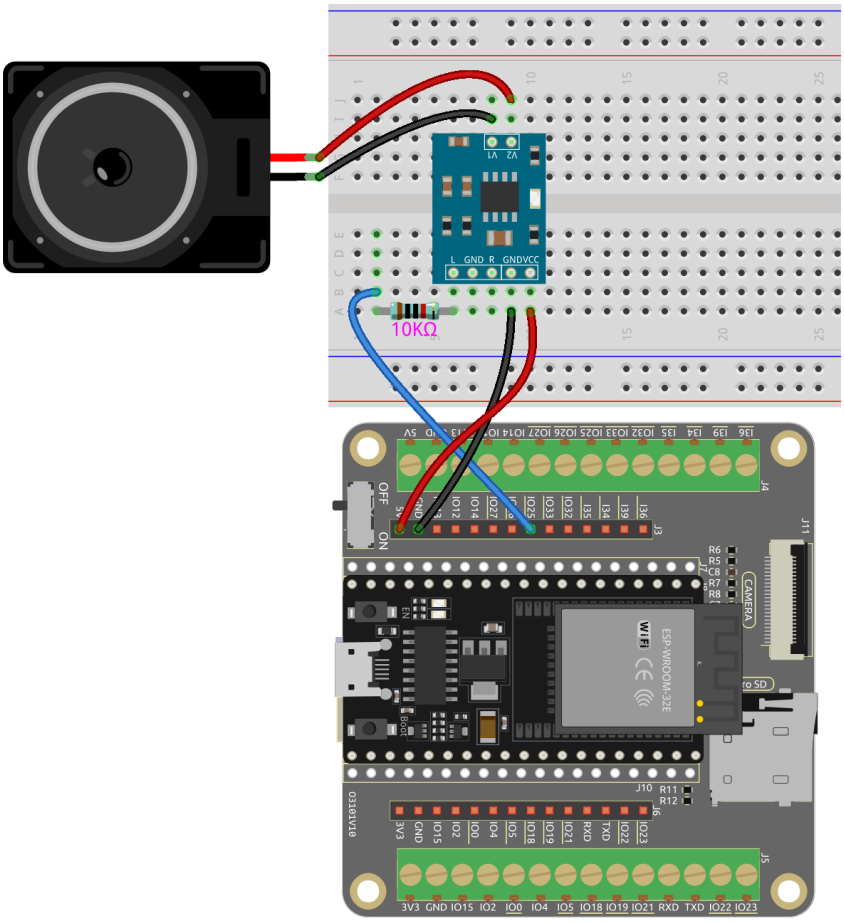
Pasos Operativos

1. Construye el circuito.

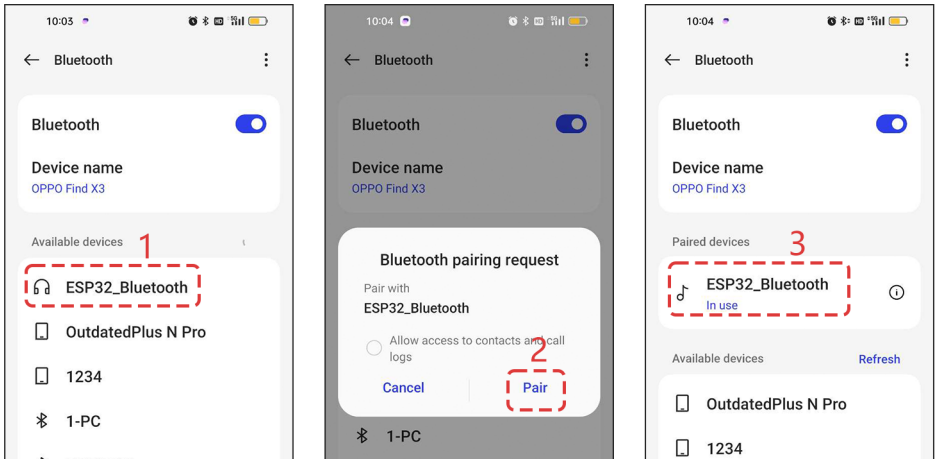
   Como este es un amplificador mono, puedes conectar IO25 al pin L o R del módulo amplificador de audio.

   La resistencia de 10K se utiliza para reducir el ruido de alta frecuencia y bajar el volumen del audio. Forma un filtro pasabajos RC con la capacitancia parásita del DAC y el amplificador de audio. Este filtro disminuye la amplitud de las señales de alta frecuencia, reduciendo efectivamente el ruido de alta frecuencia. Por lo tanto, agregar la resistencia de 10K hace que la música suene más suave y elimina el ruido de alta frecuencia no deseado.

   Si la música de tu tarjeta SD ya es suave, puedes quitar o reemplazar la resistencia con un valor más pequeño.



2. Abre el código.
    - Abre el archivo `7.3_bluetooth_audio_player.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\7.3_bluetooth_audio_player`.
    - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
    - *¿Siempre aparece «COMxx desconocido»?*
    - Aquí se utiliza la biblioteca `ESP32-A2DP`, consulta *Instalación Manual* para obtener un tutorial para instalar.
3. Después de seleccionar la placa y el puerto correctos, haz clic en el botón Subir.
4. Una vez que el código se haya subido con éxito, enciende el dispositivo habilitado para Bluetooth y busca dispositivos disponibles, luego conéctate al `ESP32_Bluetooth`.



5. Reproduce audio en el dispositivo y el audio debería reproducirse a través del altavoz conectado al ESP32.

Explicación del Código

1. El código comienza incluyendo la biblioteca `BluetoothA2DPSink.h`, que se utiliza para recibir datos de audio desde el dispositivo habilitado para Bluetooth. Luego se crea y configura el objeto `BluetoothA2DPSink` con los ajustes de la interfaz I2S.

```
#include "BluetoothA2DPSink.h"

BluetoothA2DPSink a2dp_sink;
```

2. En la función setup, el código inicializa una estructura `i2s_config_t` con la configuración deseada para la interfaz I2S (Inter-IC Sound).

```
void setup() {
const i2s_config_t i2s_config = {
    .mode = (i2s_mode_t) (I2S_MODE_MASTER | I2S_MODE_TX | I2S_MODE_DAC_
↪BUILT_IN),
    .sample_rate = 44100, // corrected by info from bluetooth
    .bits_per_sample = (i2s_bits_per_sample_t) 16, // the DAC module will␣
↪only take the 8bits from MSB
    .channel_format =  I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = (i2s_comm_format_t)I2S_COMM_FORMAT_STAND_MSB,
    .intr_alloc_flags = 0, // default interrupt priority
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false
};

a2dp_sink.set_i2s_config(i2s_config);
a2dp_sink.start("ESP32_Bluetooth");

}
```

- La interfaz I2S se utiliza para transferir datos de audio digital entre dispositivos.
- La configuración incluye el `modo I2S`, `tasa de muestreo`, `bits por muestra`, `formato de canal`, `formato de comunicación`, `flags de asignación de interrupción`, `conteo de buffer DMA`, `longitud del buffer DMA` y si usar o no el APLL (Audio PLL).

- La estructura `i2s_config_t` se pasa luego como argumento a la función `set_i2s_config` del objeto `BluetoothA2DPSink` para configurar la interfaz I2S para la reproducción de audio.
- La función `start` del objeto `BluetoothA2DPSink` se llama para iniciar el receptor de audio Bluetooth y comenzar a reproducir audio a través del DAC integrado.

2.41 7.4 Escritura y Lectura de Tarjeta SD

Este proyecto demuestra las capacidades esenciales de usar una tarjeta SD con el microcontrolador ESP32. Muestra operaciones fundamentales como montar la tarjeta SD, crear un archivo, escribir datos en el archivo, y listar todos los archivos dentro del directorio raíz. Estas operaciones forman la base de muchas aplicaciones de registro y almacenamiento de datos, haciendo de este proyecto un pilar crucial en la comprensión y utilización del periférico SDMMC host integrado del ESP32.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

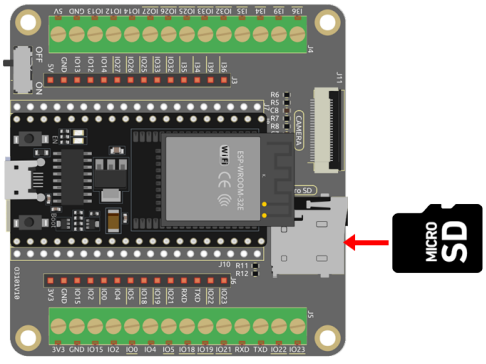
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

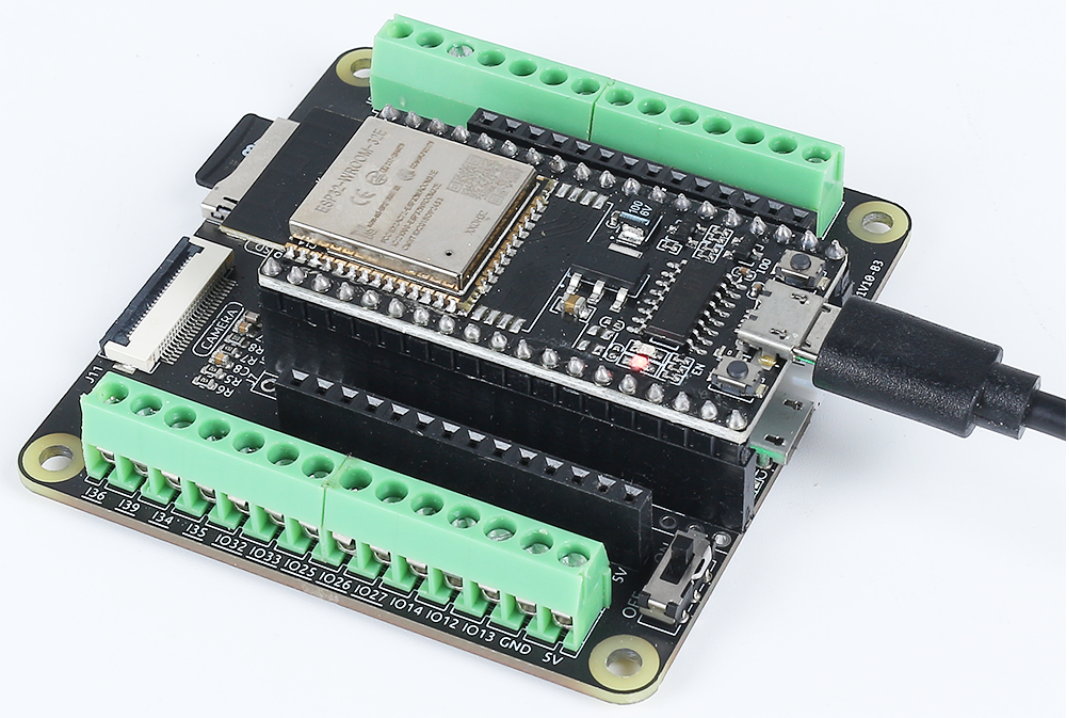
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-

Pasos Operativos

1. Antes de conectar el cable USB, inserta la tarjeta SD en la ranura de la tarjeta SD de la placa de extensión.



2. Conecta ESP32-WROOM-32E al ordenador usando el cable USB.



3. Selecciona el puerto y la placa adecuados en el IDE de Arduino y sube el código a tu ESP32.

Nota:

- Abre el archivo `7.4_sd_read_write.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\7.4_sd_read_write`.
- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

---

4. Una vez subido el código con éxito, verás un mensaje indicando la escritura de archivo exitosa, junto con una lista de todos los nombres de archivos y tamaños en la tarjeta SD. Si no ves ningún mensaje después de abrir el monitor serie, necesitas presionar el botón EN (RST) para volver a ejecutar el código.

Nota: Si ves la siguiente información.

```
E (528) vfs_fat_sdmmc: mount_to_vfs failed (0xffffffff).
Failed to mount SD card
```

Primero, verifica si tu tarjeta SD está correctamente insertada en la placa de extensión.

Si está insertada correctamente, podría haber un problema con tu tarjeta SD. Puedes intentar usar una goma de borrar para limpiar los contactos metálicos.

Si el problema persiste, se recomienda formatear la tarjeta SD, consulta *¿Cómo formatear la tarjeta SD?*.

Cómo funciona?

El propósito de este proyecto es demostrar el uso de la tarjeta SD con la placa ESP32. Se utiliza el periférico SDMMC host integrado del ESP32 para conectarse con la tarjeta SD.

El proyecto comienza inicializando la comunicación serie y luego intenta montar la tarjeta SD. Si la tarjeta SD no se monta con éxito, el programa imprimirá un mensaje de error y saldrá de la función de configuración.

Una vez que la tarjeta SD se monta con éxito, el programa procede a crear un archivo llamado «test.txt» en el directorio raíz de la tarjeta SD. Si el archivo se abre con éxito en modo de escritura, el programa escribe una línea de texto - «Hello, world!» en el archivo. El programa imprimirá un mensaje de éxito si la operación de escritura es exitosa, de lo contrario, se imprimirá un mensaje de error.

Después de la operación de escritura, el programa cierra el archivo y luego abre el directorio raíz de la tarjeta SD. Comienza a recorrer todos los archivos en el directorio raíz, imprimiendo el nombre y el tamaño del archivo de cada archivo encontrado.

En la función de bucle principal, no hay operaciones. Este proyecto se centra en operaciones de tarjeta SD como montar la tarjeta, crear un archivo, escribir en un archivo y leer el directorio del archivo, todas las cuales se ejecutan en la función de configuración.

Este proyecto sirve como una útil introducción al manejo de tarjetas SD con el ESP32, lo cual puede ser crucial en aplicaciones que requieren registro o almacenamiento de datos.

Aquí hay un análisis del código:

1. Incluye la biblioteca `SD_MMC`, que es necesaria para trabajar con tarjetas SD usando el periférico SDMMC host integrado del ESP32.

```
#include "SD_MMC.h"
```

2. Dentro de la función `setup()`, se realizan las siguientes tareas.

   - **Inicializar la tarjeta SD**

   Inicializa y monta la tarjeta SD. Si la tarjeta SD no se monta, imprimirá «Failed to mount SD card» en el monitor serie y detendrá la ejecución.

```
if(!SD_MMC.begin()) { // Intenta montar la tarjeta SD
    Serial.println("Failed to mount card"); // Si el montaje falla, imprime␣
↪en serie y sale de la configuración
    return;
}
```

   - **Abrir el archivo**

   Abre un archivo llamado `"test.txt"` ubicado en el directorio raíz de la tarjeta SD en modo de escritura. Si el archivo no se abre, imprime «Failed to open file for writing» y regresa.

```
File file = SD_MMC.open("/test.txt", FILE_WRITE);
if (!file) {
    Serial.println("Failed to open file for writing"); // Print error␣
↪message if file failed to open
    return;
}
```

   - **Escribir datos en el archivo**

   Escribe el texto «Test file write» en el archivo. Si la operación de escritura es exitosa, imprime «File write successful»; de lo contrario, imprime «File write failed».

```
if(file.print("Test file write")) { // Write the message to the file
    Serial.println("File write success"); // If write succeeds, print to␣
↪serial
} else {
    Serial.println("File write failed"); // If write fails, print to serial
}
```

   - **Cerrar el archivo**

   Cierra el archivo abierto. Esto asegura que cualquier dato almacenado en el búfer se escriba en el archivo y que el archivo se cierre correctamente.

```
file.close(); // Close the file
```

   - **Abrir el directorio raíz**

   Abre el directorio raíz de la tarjeta SD. Si el directorio no se abre, imprime «Failed to open directory» y regresa.

```
File root = SD_MMC.open("/"); // Open the root directory of SD card
if (!root) {
    Serial.println("Failed to open directory"); // Print error message if␣
↪directory failed to open
    return;
}
```

- **Imprimir el nombre y tamaño de cada archivo**

El bucle que comienza con while (`File file = root.openNextFile()`) itera sobre todos los archivos en el directorio raíz, imprimiendo el nombre y el tamaño de cada archivo en el monitor serie.

```
Serial.println("Files found in root directory:"); // Print the list of␣
↪files found in the root directory
while (File file = root.openNextFile()) { // Loop through all the files in␣
↪the root directory
      Serial.print("  ");
      Serial.print(file.name()); // Print the filename
      Serial.print("\t");
      Serial.println(file.size()); // Print the filesize
      file.close(); // Close the file
}
```

3. Esta función `loop()` es un bucle vacío y no hace nada en el programa actual. Sin embargo, en un programa típico de Arduino, esta función repetiría continuamente y ejecutaría el código dentro de ella. En este caso, ya que todas las tareas requeridas se han realizado en la función de configuración, la función de bucle no es necesaria.

```
void loop() {} // Empty loop function, does nothing
```

2.42 7.5 Reproductor MP3 con Soporte de Tarjeta SD

¡Bienvenido al emocionante mundo de la música con tu ESP32! Este proyecto lleva el poder del procesamiento de audio a tus manos, convirtiendo tu ESP32 no solo en un microcontrolador increíble para la computación, sino también en tu reproductor de música personalizado. Imagina entrar a tu habitación y tener tu pista favorita sonando directamente desde este pequeño dispositivo. Ese es el poder que estamos trayendo a tus manos hoy.

Componentes Necesarios

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar todo el kit, aquí está el enlace:

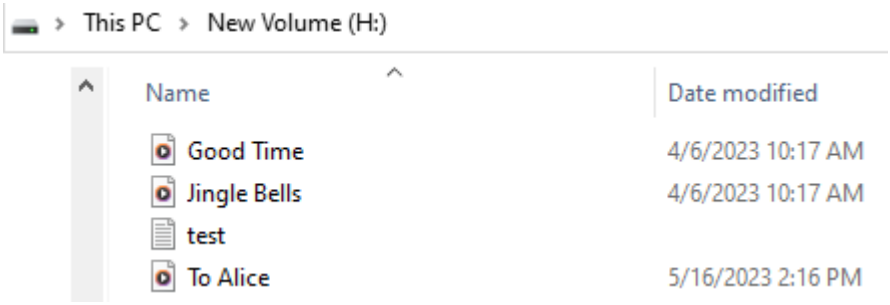
Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

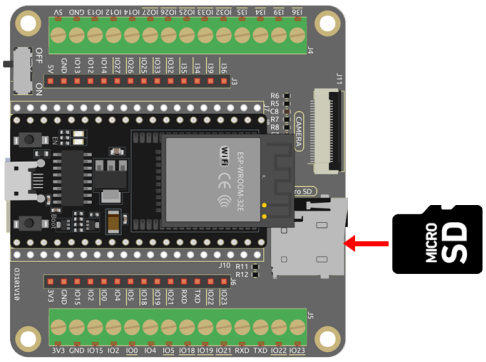
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Módulo de Audio y Altavoz	-

Pasos Operativos

1. Inserta tu tarjeta SD en la computadora usando un lector de tarjetas, y luego formátela. Puedes referirte al tutorial en *¿Cómo formatear la tarjeta SD?*.
2. Copia tu archivo MP3 favorito a tu tarjeta SD.



3. Inserta la tarjeta SD en la ranura para tarjetas SD de la placa de expansión.

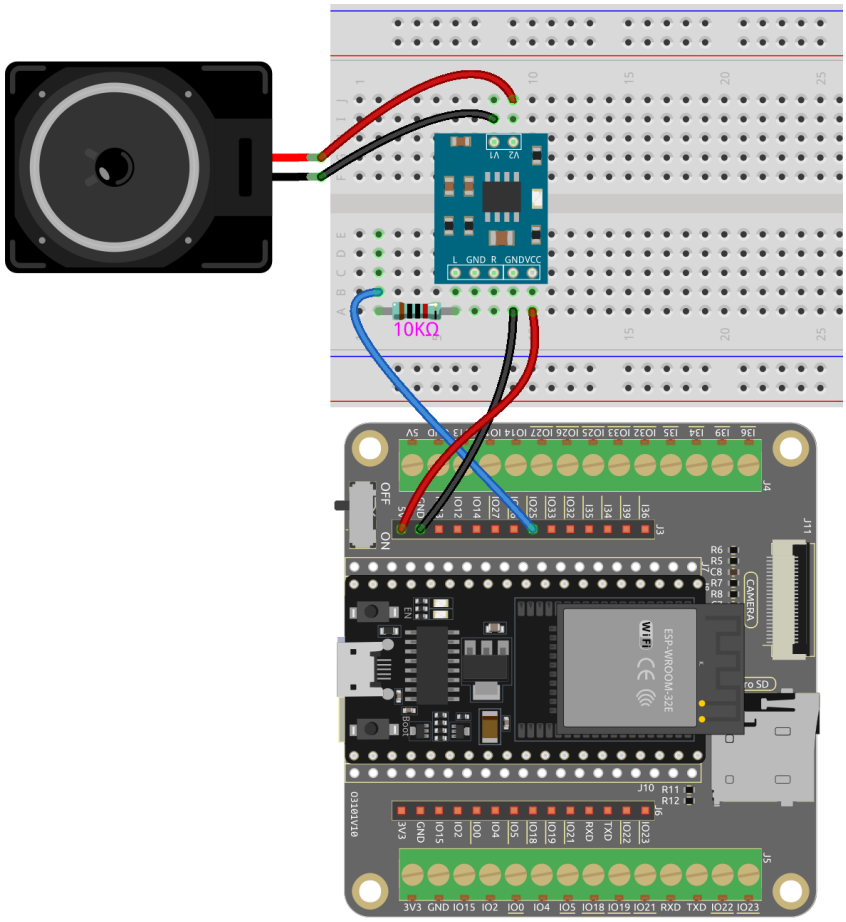


4. Construye el circuito.

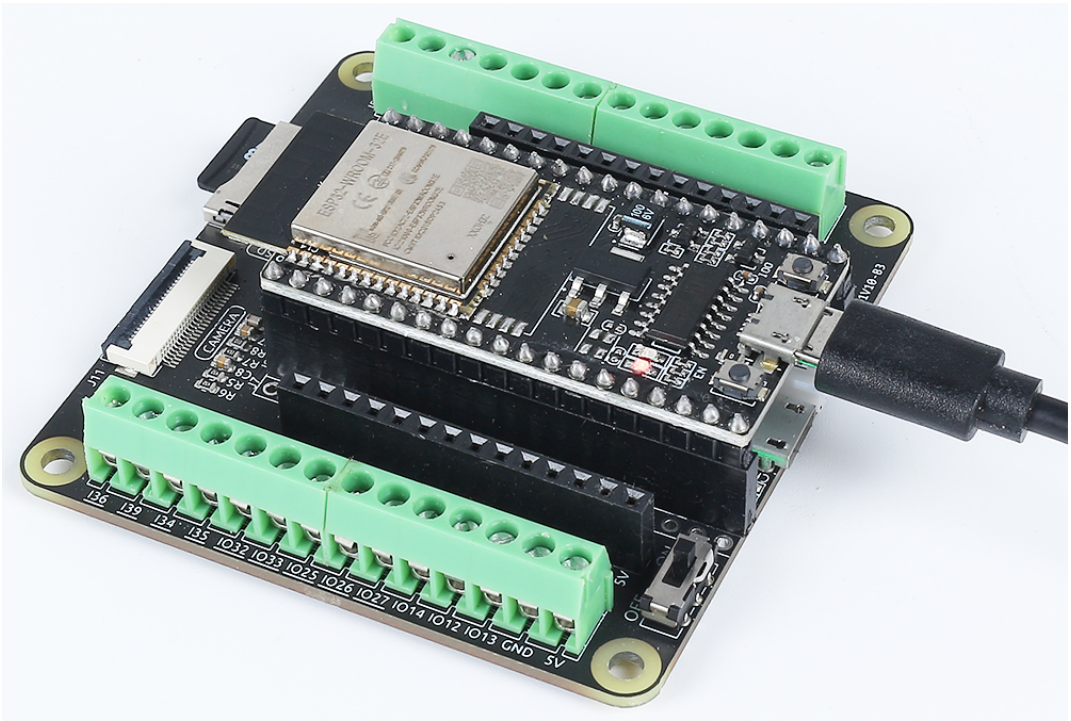
   Como esto es un amplificador mono, puedes conectar IO25 al pin L o R del módulo de amplificador de audio.

   La resistencia de 10K se utiliza para reducir el ruido de alta frecuencia y bajar el volumen del audio. Forma un filtro pasa bajos RC con la capacitancia parasitaria del DAC y el amplificador de audio. Este filtro disminuye la amplitud de las señales de alta frecuencia, reduciendo efectivamente el ruido de alta frecuencia. Por lo tanto, agregar la resistencia de 10K hace que la música suene más suave y elimina el ruido de alta frecuencia no deseado.

   Si la música de tu tarjeta SD ya es suave, puedes quitar o reemplazar la resistencia por un valor menor.



5. Conecta el ESP32-WROOM-32E a la computadora usando el cable USB.



6. Modifica el código.

   Modifica la línea de código `file = new AudioFileSourceSD_MMC("/To Alice.mp3");` para reflejar el nombre y ruta de tu archivo.

   Nota:

   - Abre el archivo `7.5_mp3_player_sd.ino` bajo la ruta de `esp32-starter-kit-main\c\codes\7.5_mp3_player_sd`. O copia este código en **Arduino IDE**.
   - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
   - *¿Siempre aparece «COMxx desconocido»?*
   - La biblioteca `ESP8266Audio` se utiliza aquí, consulta *Instalación Manual* para un tutorial de instalación.

   ---

7. Selecciona el puerto y la placa apropiados en el IDE de Arduino y sube el código a tu ESP32.
8. Después de subir exitosamente el código, escucharás tu música favorita sonando.

¿Cómo funciona?

- El código utiliza varias clases de la biblioteca `ESP8266Audio` para reproducir un archivo MP3 desde una tarjeta SD a través de I2S:

  ```
  #include "AudioFileSourceSD_MMC.h"
  #include "AudioOutputI2S.h"
  #include "AudioGeneratorMP3.h"
  #include "SD_MMC.h"
  #include "FS.h"
  ```

  - `AudioGeneratorMP3` es una clase que decodifica audio MP3.
  - `AudioFileSourceSD_MMC` es una clase que lee datos de audio desde una tarjeta SD.
  - `AudioOutputI2S` es una clase que envía datos de audio a la interfaz I2S.

- En la función `setup()`, inicializamos la tarjeta SD, abrimos el archivo MP3 desde la tarjeta SD, configuramos la salida I2S en el DAC interno del ESP32, configuramos la salida a mono y comenzamos el generador MP3.

  ```
  void setup() {
      // Start the serial communication.
      Serial.begin(115200);
      delay(1000);

      // Initialize the SD card. If it fails, print an error message.
      if (!SD_MMC.begin()) {
          Serial.println("SD card mount failed!");
      }

      // Open the MP3 file from the SD card. Replace "/To Alice.mp3" with␣
  ↪your own MP3 file name.
      file = new AudioFileSourceSD_MMC("/To Alice.mp3");

      // Set up the I2S output on ESP32's internal DAC.
      out = new AudioOutputI2S(0, 1);
  ```

(continué en la próxima página)

(proviene de la página anterior)

```
    // Set the output to mono.
    out->SetOutputModeMono(true);

    // Initialize the MP3 generator with the file and output.
    mp3 = new AudioGeneratorMP3();
    mp3->begin(file, out);
}
```

- En la función `loop()`, verificamos si el generador MP3 está funcionando. Si es así, continuamos en bucle; de lo contrario, lo detenemos e imprimimos «MP3 terminado» en el monitor serial.

```
void loop() {
    // If the MP3 is running, loop it. Otherwise, stop it.
    if (mp3->isRunning()) {
        if (!mp3->loop()) mp3->stop();
    }
    // If the MP3 is not running, print a message and wait for 1 second.
    else {
        Serial.println("MP3 done");
        delay(1000);
    }
}
```

2.43 7.6 Tomar foto y guardar en SD

Este documento describe un proyecto que implica tomar una foto usando la ESP32-CAM y guardarla en una tarjeta SD. El objetivo del proyecto es proporcionar una solución simple para capturar imágenes con la ESP32-CAM y almacenarlas en una tarjeta SD.

Componentes necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-

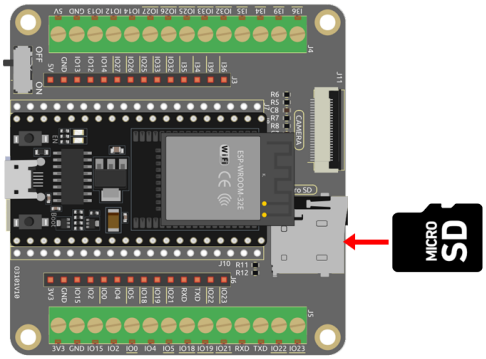
Precauciones relacionadas

Al usar la ESP32-CAM, es importante notar que el pin GPIO 0 debe estar conectado a GND para subir un sketch. Además, después de conectar GPIO 0 a GND, se debe presionar el botón de RESET en la ESP32-CAM para poner la

placa en modo de carga. También es importante asegurarse de que la tarjeta SD esté correctamente montada antes de guardar imágenes en ella.

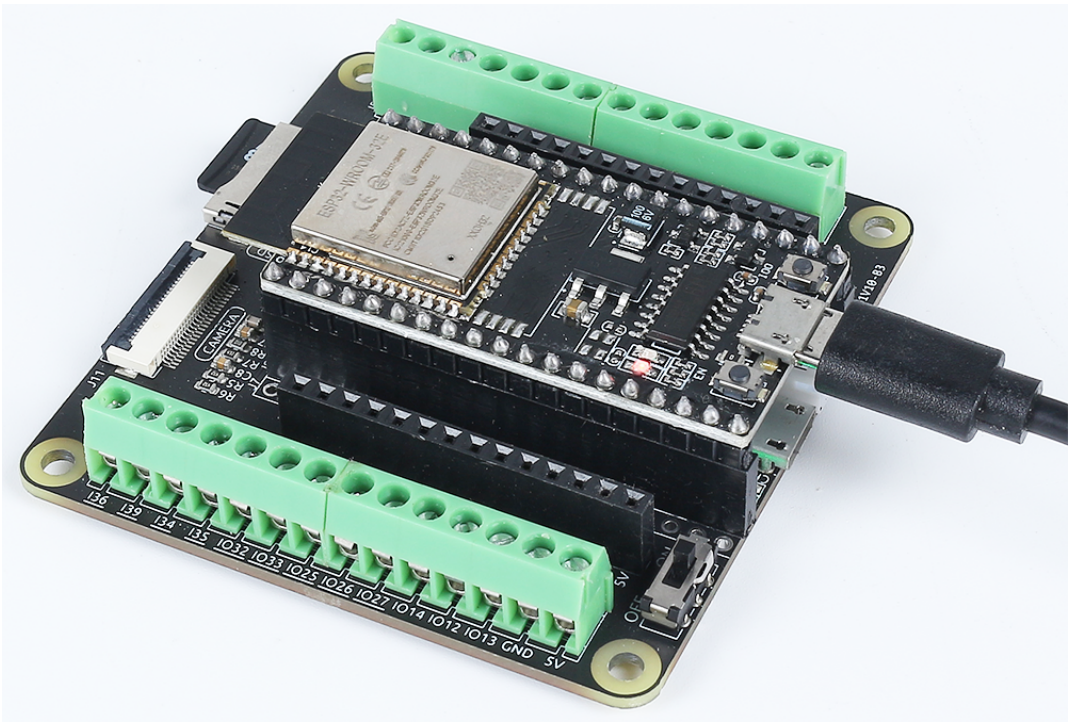
Pasos operativos

1. Inserta tu tarjeta SD en la computadora usando un lector de tarjetas y luego formátela. Puedes referirte al tutorial en *¿Cómo formatear la tarjeta SD?*.
2. Luego, retira el lector de tarjetas e inserta la tarjeta SD en la placa de expansión.



3. Ahora, conecta la cámara.

4. Conecta el ESP32-WROOM-32E a la computadora usando el cable USB.



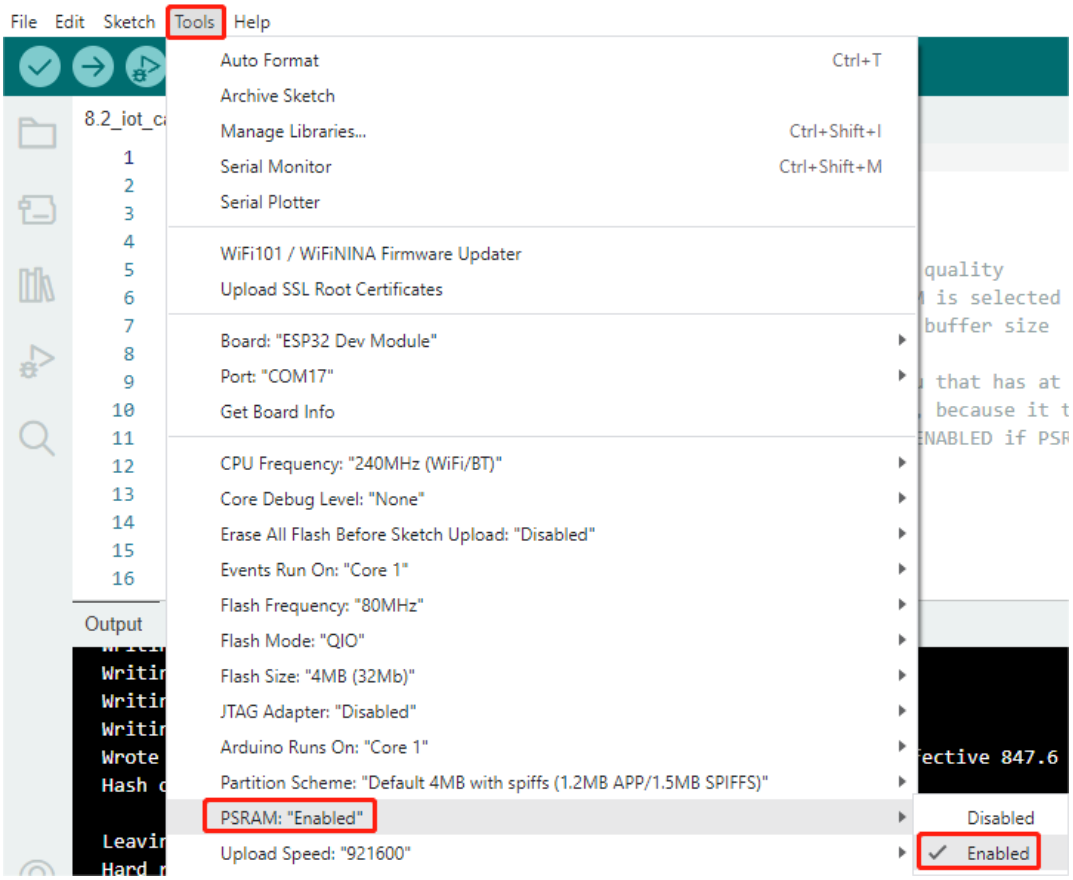
5. Abre el código.

Nota:

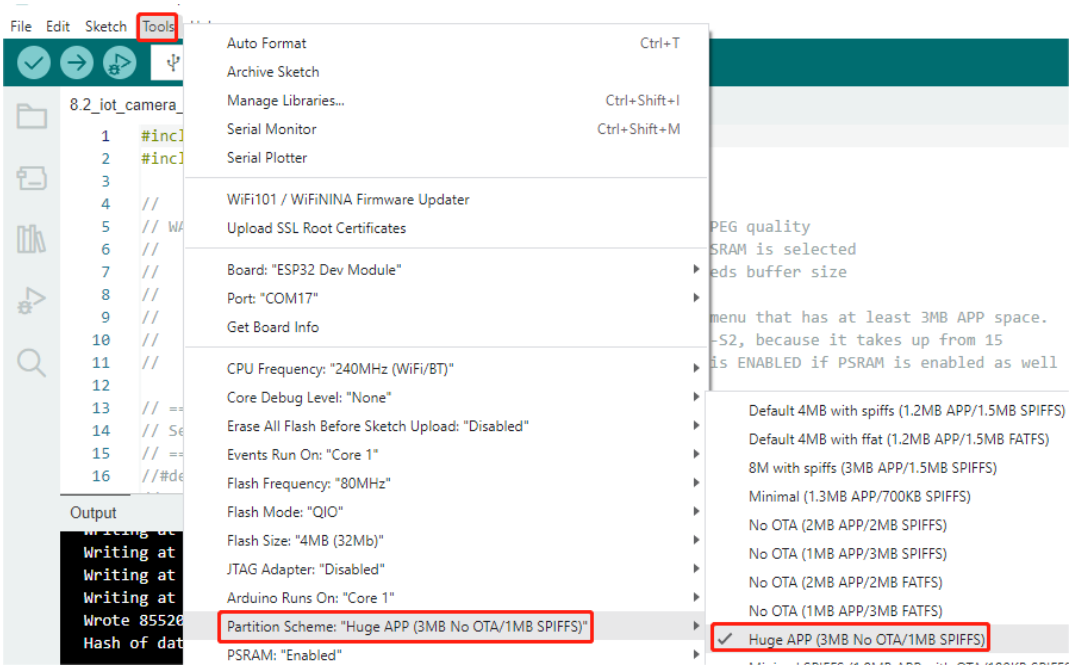
- Abre el archivo `7.6_take_photo_sd.ino` en la ruta `esp32-starter-kit-main\c\codes\7.6_take_photo_sd`.

- Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
- *¿Siempre aparece «COMxx desconocido»?*

6. Ahora, habilita **PSRAM**.

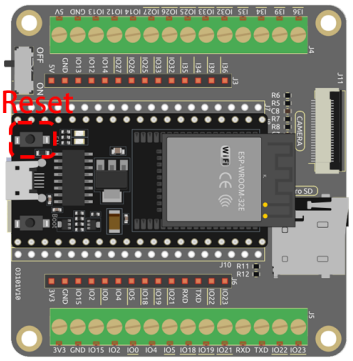


7. Establece el esquema de partición a **Gran APP (3MB Sin OTA/1MB SPIFFS)**.

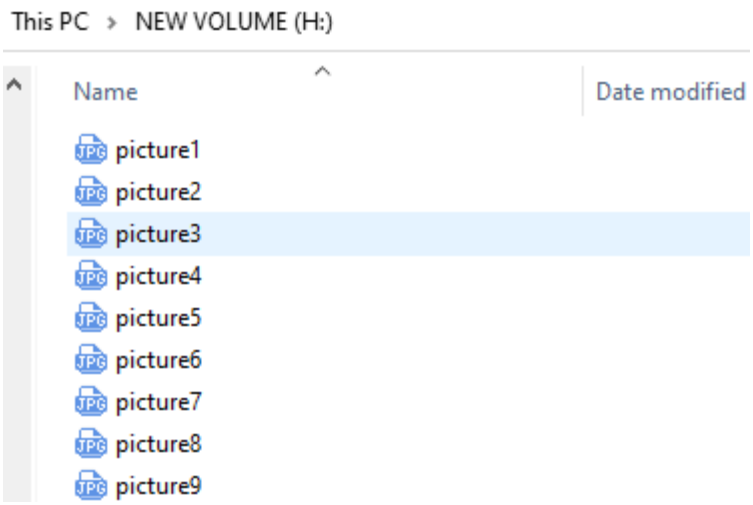


8. Selecciona el puerto y la placa apropiados en el IDE de Arduino y sube el código a tu ESP32.
9. Después de la carga exitosa del código, presiona el botón **Reset** para tomar una foto. Además, puedes verificar el Monitor Serial para ver la siguiente información indicando la captura exitosa.

```
Picture file name: /picture9.jpg
Saved file to path: /picture9.jpg
Going to sleep now
```



10. Ahora, retira la tarjeta SD de la placa de expansión e insértala en tu computadora. Podrás ver las fotos que acabas de tomar.



¿Cómo funciona?

Este código controla una cámara AI Thinker ESP32-CAM para tomar una foto, guardarla en una tarjeta SD y luego poner la ESP32-CAM en modo de sueño profundo. Aquí se detallan las partes clave:

- **Bibliotecas**: El código inicia con la inclusión de las bibliotecas necesarias para la ESP32-CAM, sistema de archivos (FS), tarjeta SD y EEPROM (usada para almacenar datos entre ciclos de energía).

```
#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>  // read and write from flash memory
```

- **Definiciones de Pines**: Esta sección configura constantes que representan las conexiones de pines de la ESP32-CAM al módulo de la cámara.

```
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27

#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22
```

- **Variables Globales**: Se declara una variable global `pictureNumber` para llevar el registro del número de fotos

tomadas y guardadas en la tarjeta SD.

```
int pictureNumber = 0;
```

- **Función de Configuración**: En la función `setup()`, se realizan varias tareas:
  - Primero, se desactiva el detector de brown-out para prevenir que la ESP32-CAM se reinicie durante consumos altos de corriente (como cuando la cámara está operando).

    ```
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);  //disable brownout detector
    ```

  - Se inicializa la comunicación Serial para depuración.

    ```
    Serial.begin(115200);
    ```

  - La configuración de la cámara se establece con `camera_config_t`, incluyendo los pines GPIO, frecuencia XCLK, formato de píxel, tamaño de cuadro, calidad de jpeg y conteo de búfer de cuadros.

    ```

    ```

    camera_config_t config; config.ledc_channel = LEDC_CHANNEL_0; config.ledc_timer = LEDC_TIMER_0; config.pin_d0 = Y2_GPIO_NUM; config.pin_d1 = Y3_GPIO_NUM; config.pin_d2 = Y4_GPIO_NUM; config.pin_d3 = Y5_GPIO_NUM; config.pin_d4 = Y6_GPIO_NUM; config.pin_d5 = Y7_GPIO_NUM; config.pin_d6 = Y8_GPIO_NUM; config.pin_d7 = Y9_GPIO_NUM; config.pin_xclk = XCLK_GPIO_NUM; config.pin_pclk = PCLK_GPIO_NUM; config.pin_vsync = VSYNC_GPIO_NUM; config.pin_href = HREF_GPIO_NUM; config.pin_sscb_sda = SIOD_GPIO_NUM; config.pin_sscb_scl = SIOC_GPIO_NUM; config.pin_pwdn = PWDN_GPIO_NUM; config.pin_reset = RESET_GPIO_NUM; config.xclk_freq_hz = 20000000; config.pixel_format = PIXFORMAT_JPEG;

  - Luego, la cámara se inicializa con la configuración, y si falla, se imprime un mensaje de error.

    ```
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        Serial.printf("Camera init failed with error 0x%x", err);
        return;
    }
    ```

  - La tarjeta SD se inicializa y, si falla, se imprime un mensaje de error.

    ```
    if (!SD_MMC.begin()) {
       Serial.println("SD Card Mount Failed");
       return;
    }

    uint8_t cardType = SD_MMC.cardType();
    if (cardType == CARD_NONE) {
        Serial.println("No SD Card attached");
        return;
    }
    ```

  - Se captura una foto con la cámara y se almacena en el búfer de cuadros.

    ```
    fb = esp_camera_fb_get();
    if (!fb) {
    ```

(continué en la próxima página)

(proviene de la página anterior)

```
    Serial.println("Camera capture failed");
    return;
}
```

- Se lee el EEPROM para recuperar el número de la última foto, luego se incrementa el número de la foto para la nueva foto.

```
EEPROM.begin(EEPROM_SIZE);
pictureNumber = EEPROM.read(0) + 1;
```

- Se crea una ruta para la nueva foto en la tarjeta SD, con un nombre de archivo correspondiente al número de la foto.

```
String path = "/picture" + String(pictureNumber) + ".jpg";

fs::FS &fs = SD_MMC;
Serial.printf("Picture file name: %s\n", path.c_str());
```

- Después de guardar la foto, el número de la foto se almacena de nuevo en el EEPROM para su recuperación en el próximo ciclo de energía.

```
File file = fs.open(path.c_str(), FILE_WRITE);
if (!file) {
    Serial.println("Failed to open file in writing mode");
} else {
    file.write(fb->buf, fb->len);  // payload (image), payload length
    Serial.printf("Saved file to path: %s\n", path.c_str());
    EEPROM.write(0, pictureNumber);
    EEPROM.commit();
}
file.close();
esp_camera_fb_return(fb);
```

- Finalmente, se apaga el LED a bordo (flash) y la ESP32-CAM entra en sueño profundo.

```
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
rtc_gpio_hold_en(GPIO_NUM_4);
```

- Modo de sueño profundo: La ESP32-CAM se pone en modo de sueño profundo para conservar energía. La duración del sueño se puede ajustar según sea necesario. En este ejemplo, no se especifica un tiempo de sueño, por lo que se espera un despertar externo.

```
delay(2000);
Serial.println("Going to sleep now");
delay(2000);
esp_deep_sleep_start();
Serial.println("This will never be printed");
```

- Función Loop: La función `loop()` está vacía porque después del proceso de configuración, el ESP32-CAM entra inmediatamente en modo de sueño profundo.

Ten en cuenta que para que este código funcione, necesitas asegurarte de que el GPIO 0 esté conectado a GND al cargar el sketch, y puede que tengas que presionar el botón de RESET en la placa para poner tu placa en modo de carga.

Además, recuerda reemplazar «/picture» con tu propio nombre de archivo. El tamaño de la EEPROM se establece en 1, lo que significa que puede almacenar valores de 0 a 255. Si planeas tomar más de 255 fotos, necesitarás aumentar el tamaño de la EEPROM y ajustar cómo almacenas y lees el número de la foto.

8. Proyectos de IoT

2.44 8.1 Información del Tiempo en Tiempo Real de @OpenWeatherMap

El proyecto de Pantalla del Tiempo IoT utiliza la placa ESP32 y un módulo LCD1602 I2C para crear una pantalla de información meteorológica que recupera datos de la API de OpenWeatherMap.

Este proyecto sirve como una excelente introducción al trabajo con APIs, conectividad Wi-Fi y visualización de datos en un módulo LCD utilizando la placa ESP32. Con la Pantalla del Tiempo IoT, puedes acceder cómodamente a actualizaciones del tiempo en tiempo real de un vistazo, lo que la convierte en una solución ideal para ambientes domésticos o de oficina.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

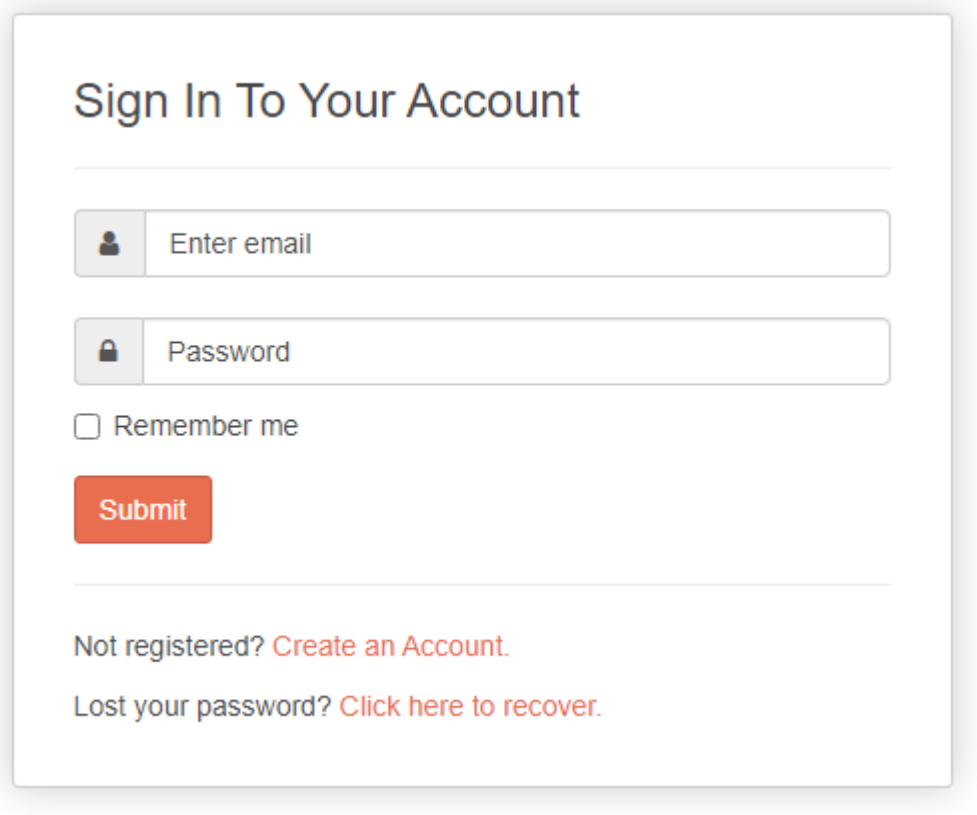
También puedes comprarlos por separado desde los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
I2C LCD1602	

Obtener las Claves API de OpenWeather

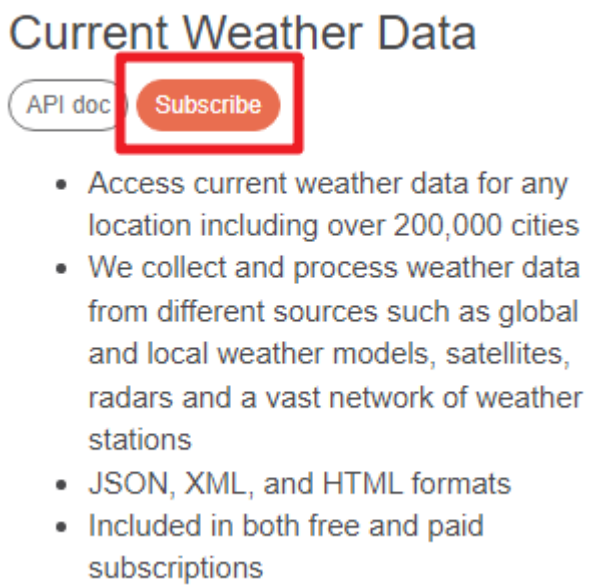
es un servicio en línea, propiedad de OpenWeather Ltd, que proporciona datos meteorológicos globales a través de API, incluyendo datos meteorológicos actuales, pronósticos, nowcasts y datos meteorológicos históricos para cualquier ubicación geográfica.

1. Visita para iniciar sesión/crear una cuenta.



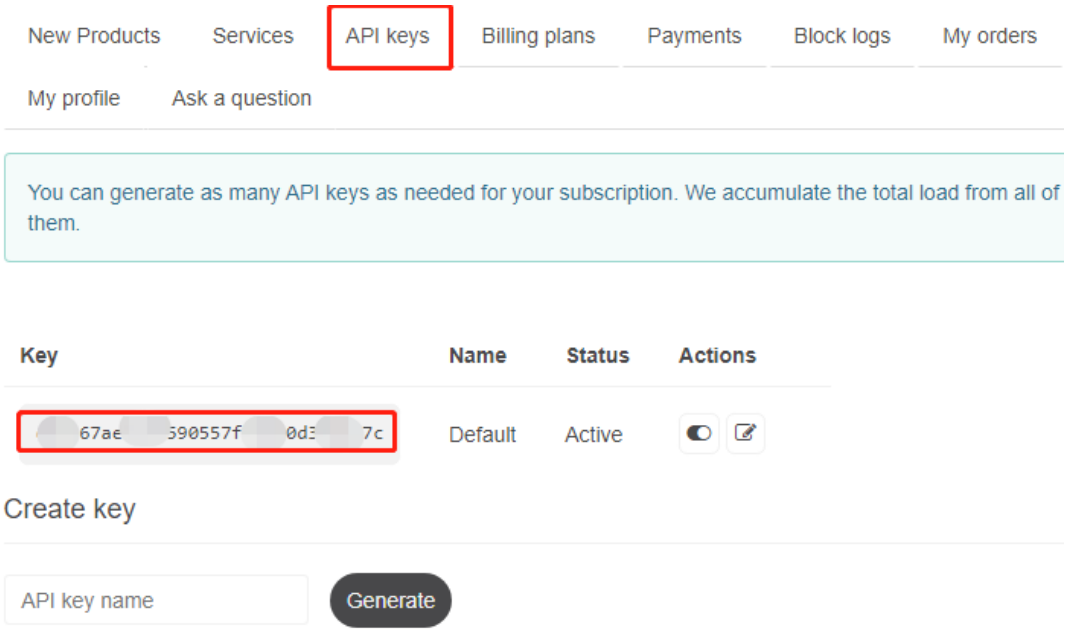
2. Haz clic en la página de API desde la barra de navegación.

3. Encuentra **Datos Meteorológicos Actuales** y haz clic en Suscribirse.



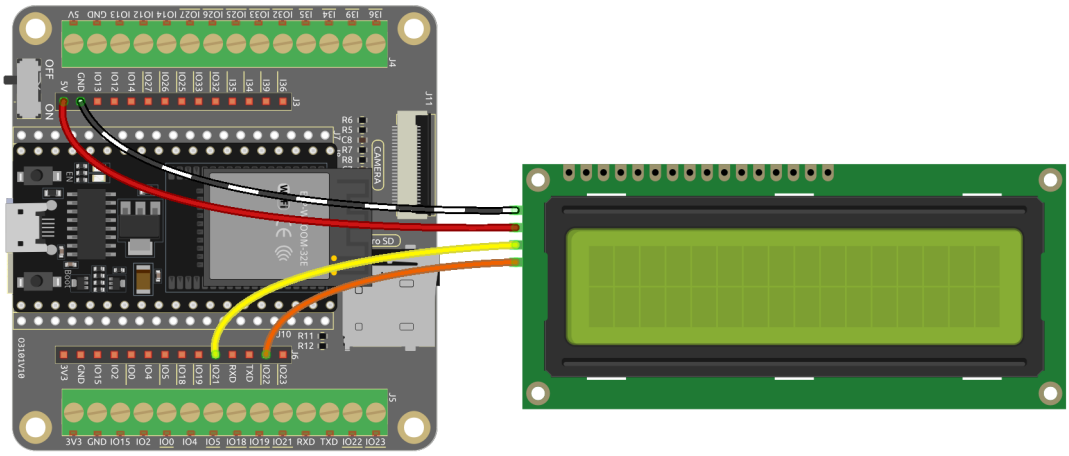
4. Bajo **Colección de datos meteorológicos actuales y pronósticos**, suscríbete al servicio apropiado. En nuestro proyecto, Gratis es suficiente.

5. Copia la clave desde la página de **Claves API**.



Completa Tu Dispositivo

1. Construye el circuito.



2. Abre el código.

   - Abre el archivo `iot_1_open_weather.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_1_open_weather`, o copia el código en el IDE de Arduino.
   - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
   - *¿Siempre aparece «COMxx desconocido»?*
   - Las bibliotecas `LiquidCrystal I2C` y `Arduino_JSON` se utilizan aquí, puedes instalarlas desde el **Administrador de Bibliotecas**.

3. Localiza las siguientes líneas y modifícalas con tu `<SSID>` y `<PASSWORD>`.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

4. Rellena las claves API que copiaste anteriormente en `openWeatherMapApiKey`.

```
// Your Domain name with URL path or IP address with path
String openWeatherMapApiKey = "<openWeatherMapApiKey>";
```

5. Reemplaza con tu código de país y ciudad.

```
// Replace with your country code and city
// Fine the country code by https://openweathermap.org/find
String city = "<CITY>";
String countryCode = "<COUNTRY CODE>";
```

6. Después de que el código se ejecute, verás la información del tiempo y la hora de tu ubicación en el I2C LCD1602.

Nota: Cuando el código esté corriendo, si la pantalla está en blanco, puedes girar el potenciómetro en la parte trasera del módulo para aumentar el contraste.

2.45 8.2 Servidor Web de Cámara

Este proyecto combina la placa ESP32 con un módulo de cámara para transmitir video de alta calidad a través de una red local. Configura tu propio sistema de cámaras sin esfuerzo y monitorea cualquier lugar en tiempo real.

Con la interfaz web del proyecto, puedes acceder y controlar el flujo de la cámara desde cualquier dispositivo conectado a la red. Personaliza la configuración de la cámara para optimizar la experiencia de transmisión y ajusta fácilmente los ajustes con la interfaz amigable.

Mejora tus capacidades de vigilancia o transmisión en vivo con el versátil proyecto de Transmisión de Cámara ESP32. Monitorea tu hogar, oficina o cualquier lugar deseado con facilidad y fiabilidad.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

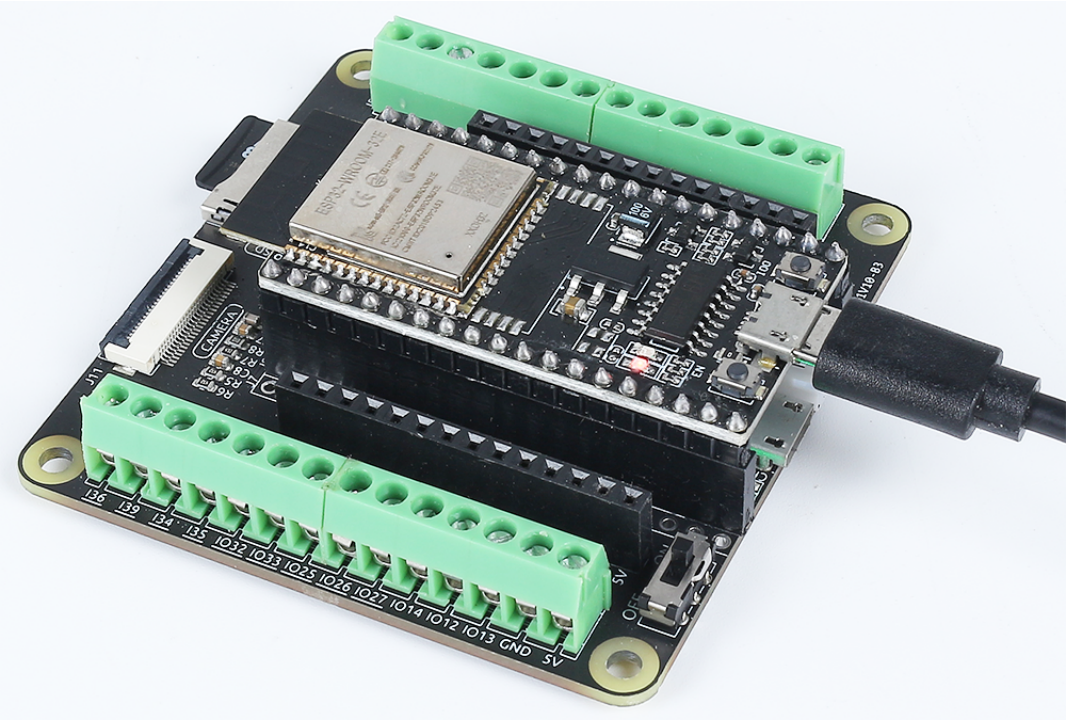
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-

¿Cómo hacerlo?

1. Primero conecta la cámara.

2. Luego, conecta el ESP32-WROOM-32E al ordenador usando el cable USB.



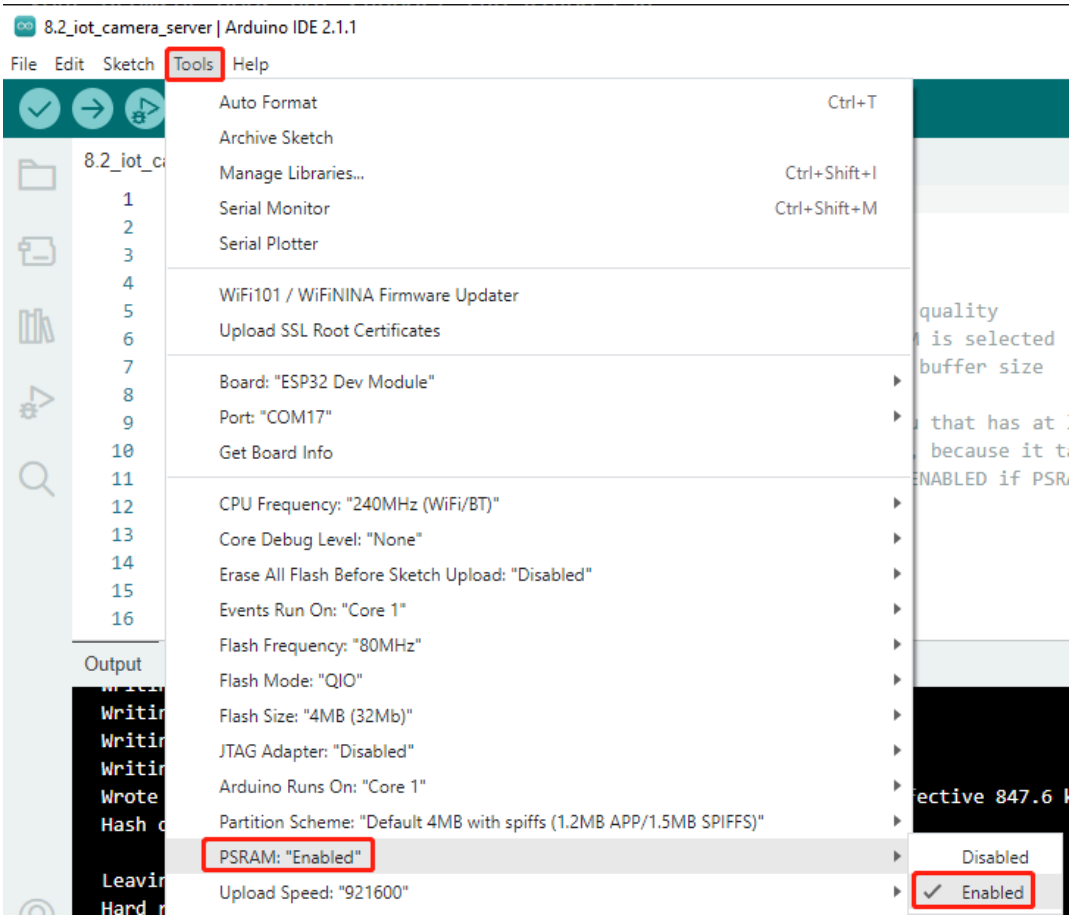
3. Abre el código.

   - Abre el archivo `iot_2_camera_server.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_2_camera_server`, o copia el código en el IDE de Arduino.
   - Después de seleccionar la placa (ESP32 Dev Module) y el puerto adecuado, haz clic en el botón **Subir**.
   - *¿Siempre aparece «COMxx desconocido»?*

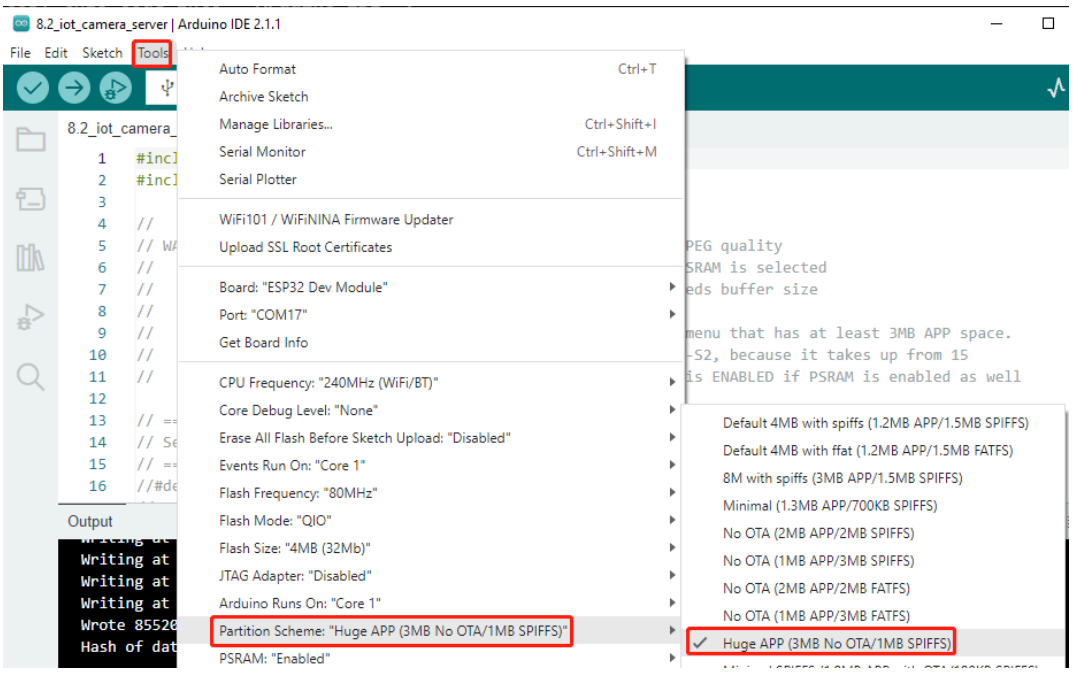
4. Localiza las siguientes líneas y modifícalas con tu <SSID> y <CONTRASEÑA>.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Ahora, habilita **PSRAM**.



6. Establece el esquema de partición a **Huge APP (3MB No OTA/1MB SPIFFS)**.

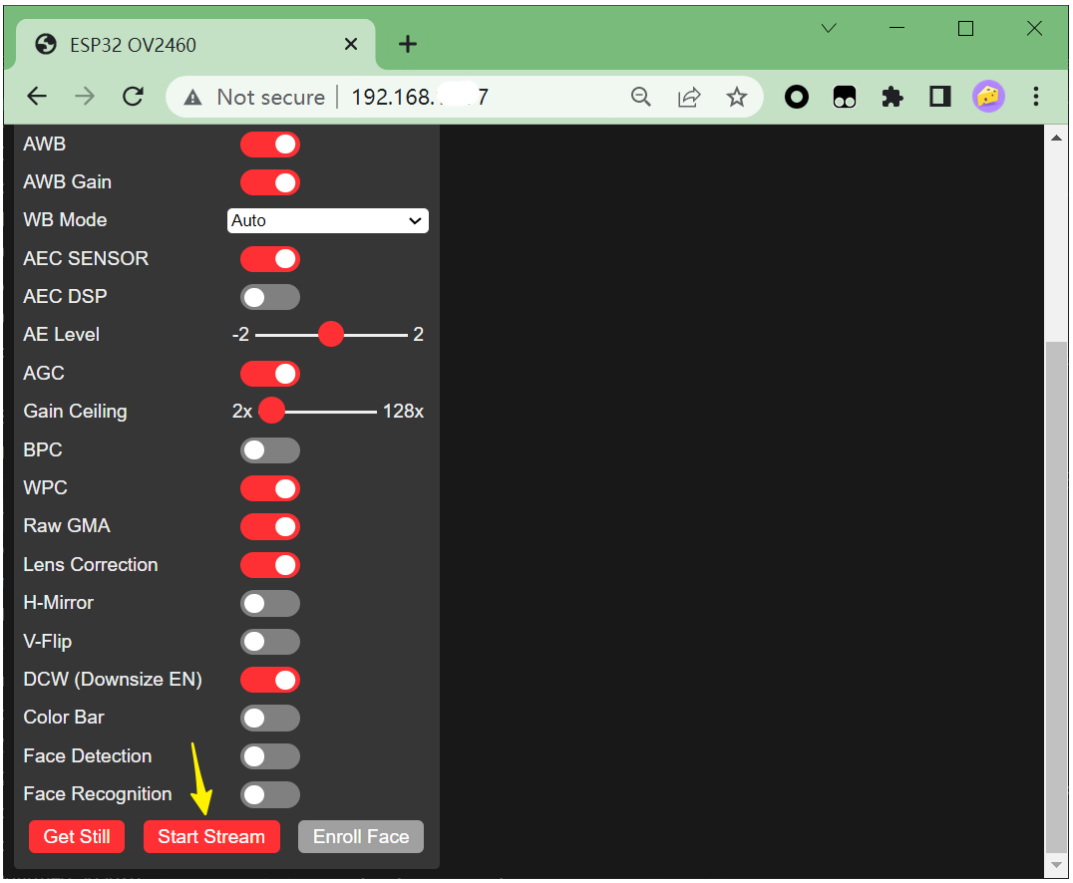


7. Después de seleccionar la placa correcta (ESP32 Dev Module) y puerto, haz clic en el botón «Subir».

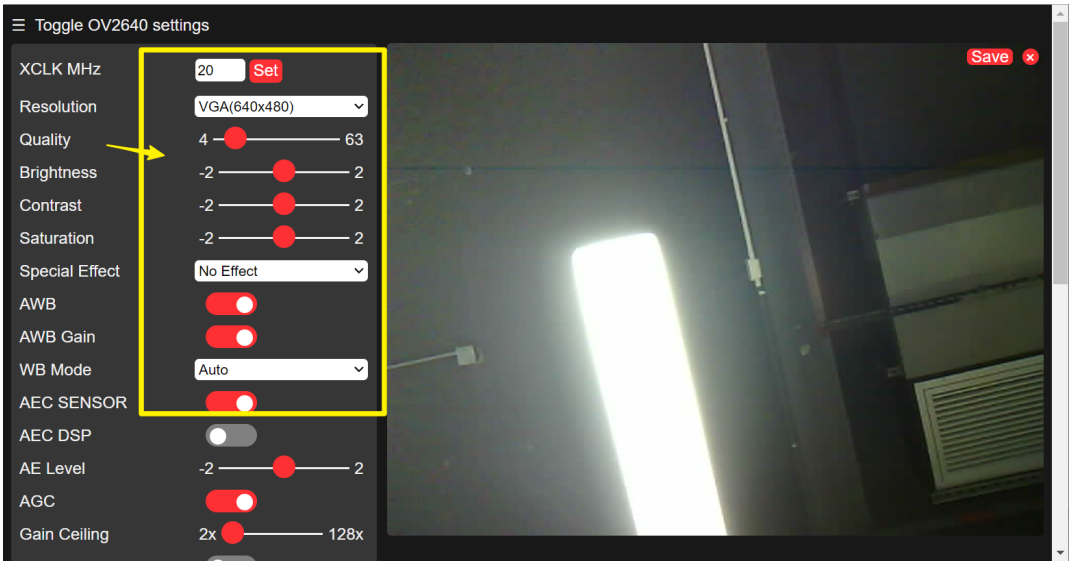
8. Verás un mensaje de conexión WiFi exitosa y la dirección IP asignada en el Monitor Serial.

```
.....
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.18.77' to connect
```

9. Ingresa la dirección IP en tu navegador web. Verás una interfaz web donde puedes hacer clic en **Iniciar Transmisión** para ver el flujo de la cámara.



10. Desplázate de vuelta al inicio de la página, donde verás el flujo de la cámara en vivo. Puedes ajustar la configuración en el lado izquierdo de la interfaz.



Nota:

- Este módulo ESP32 soporta Detección de Rostros. Para habilitarlo, ajusta la resolución a 240x240 y activa la opción de Detección de Rostros en la parte inferior de la interfaz.
- Este módulo ESP32 no soporta Reconocimiento de Rostros.

2.46 8.3 Servidor Web de Transmisión de Video Personalizado

El proyecto de Servidor Web de Transmisión de Video Personalizado ofrece una oportunidad para aprender cómo crear una página web desde cero y personalizarla para reproducir transmisiones de video. Además, podrás incorporar botones interactivos, como ENCENDER y APAGAR, para controlar la intensidad del LED.

Al construir este proyecto, ganarás experiencia práctica en desarrollo web, HTML, CSS y JavaScript. Aprenderás cómo crear una página web responsive que pueda mostrar transmisiones de video en tiempo real. Además, descubrirás cómo integrar botones interactivos para controlar el estado del LED, proporcionando una experiencia de usuario dinámica.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

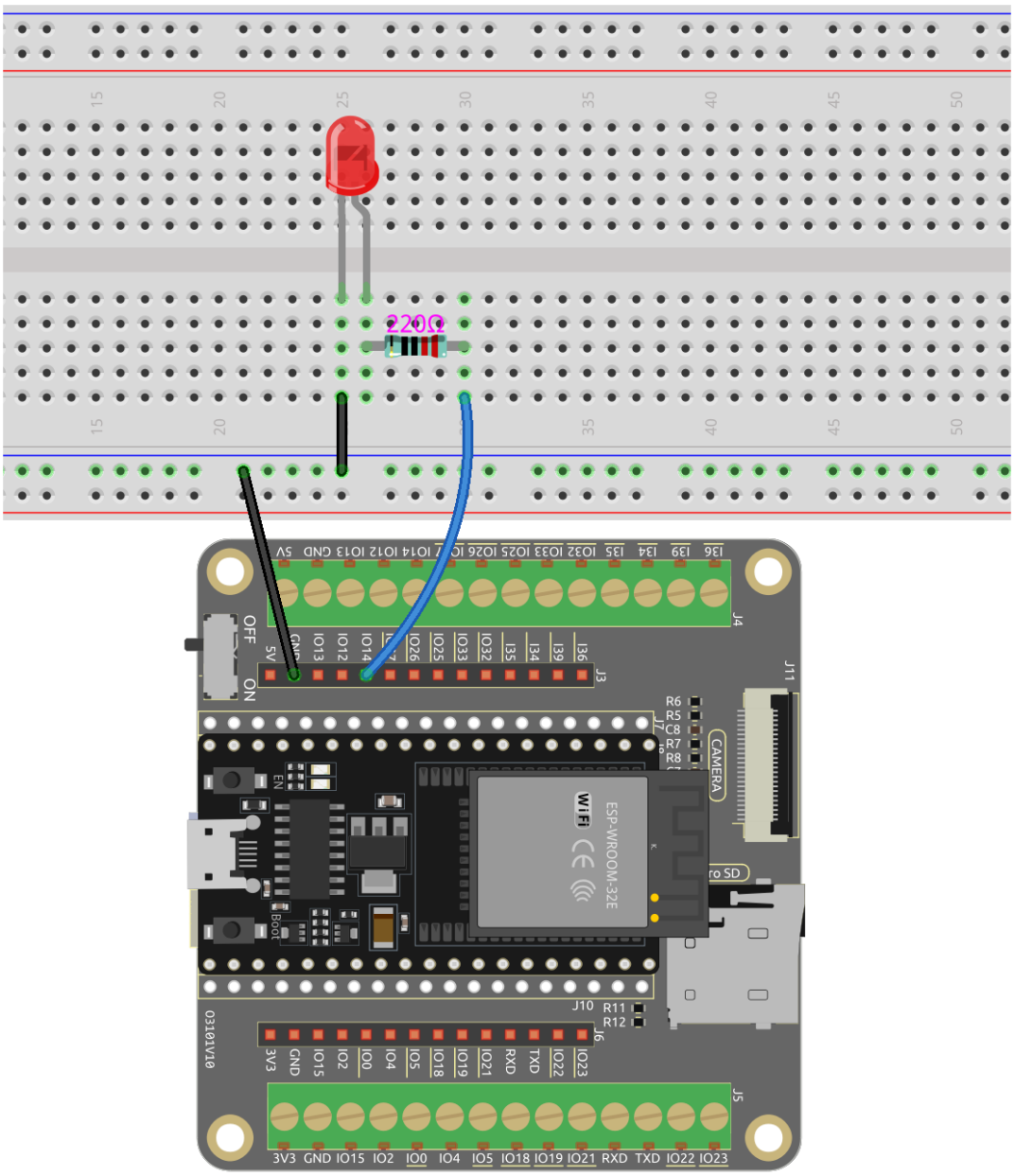
También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

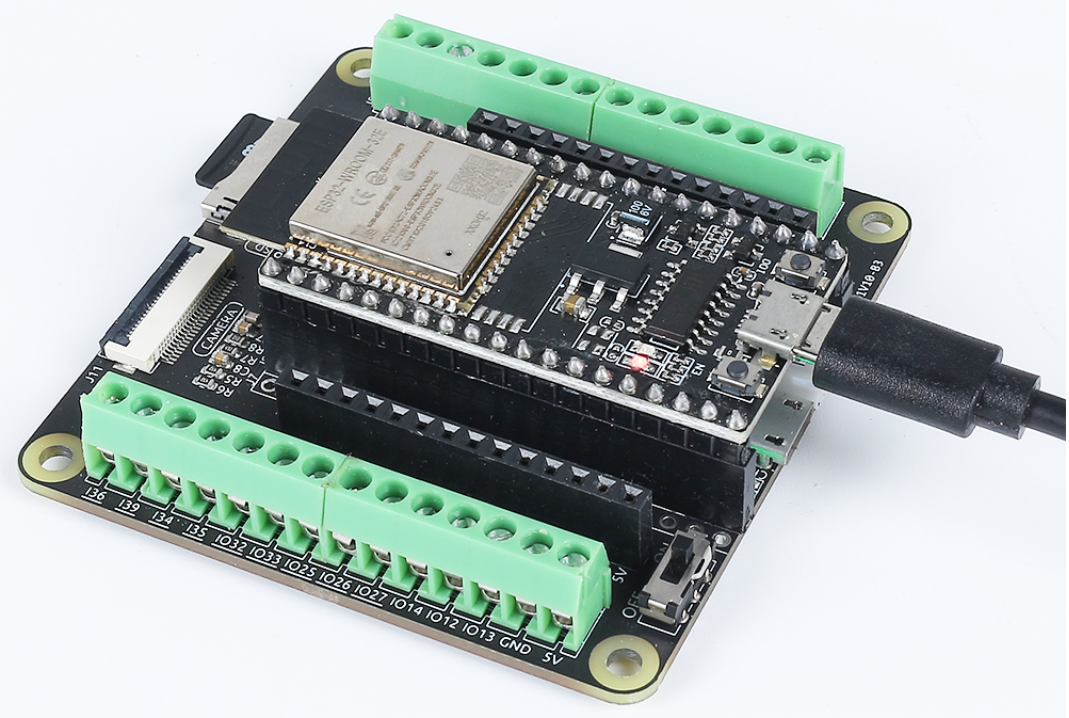
¿Cómo hacerlo?

1. Primero conecta la cámara.

2. Construye el circuito.



3. Luego, conecta el ESP32-WROOM-32E al ordenador mediante el cable USB.



4. Abre el código.

    - Abre el archivo `iot_3_html_cam_led.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_3_html_cam_led`, o copia el código en el IDE de Arduino.
    - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
    - *¿Siempre aparece «COMxx desconocido»?*

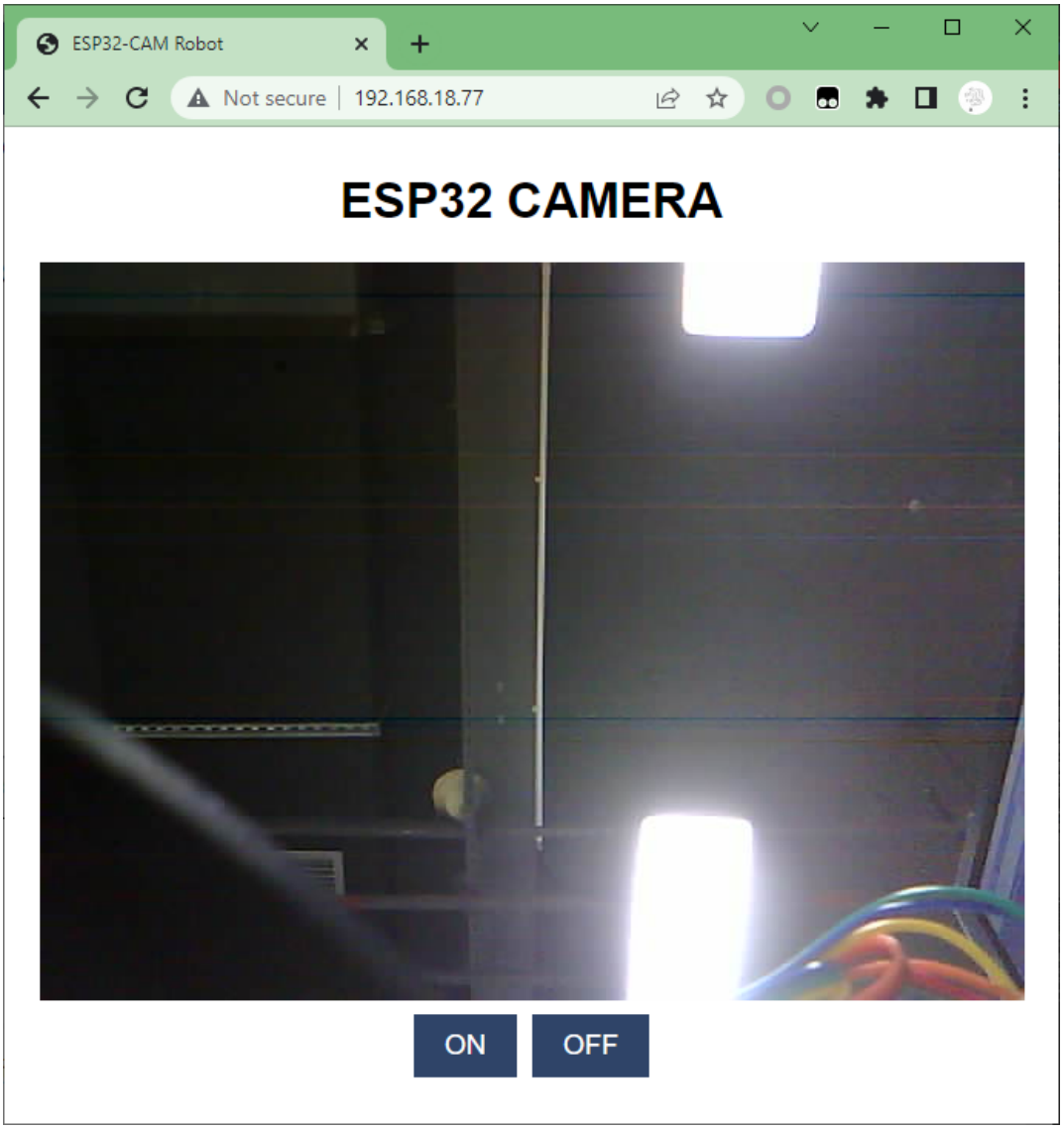
5. Localiza las siguientes líneas y modifícalas con tu <SSID> y <PASSWORD>.

```
// Reemplaza las siguientes variables con tu combinación de SSID/Contraseña
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

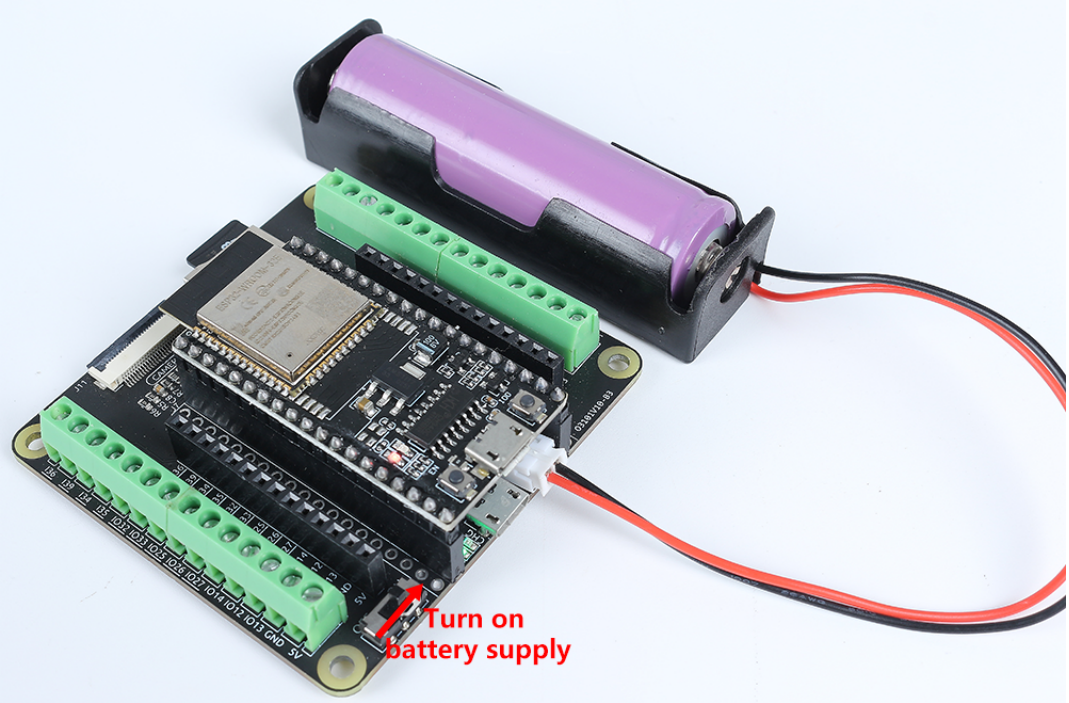
6. Después de seleccionar la placa correcta (ESP32 Dev Module) y el puerto, haz clic en el botón **Subir**.
7. Verás un mensaje de conexión exitosa a WiFi y la dirección IP asignada en el Monitor Serie.

```
Conexión WiFi exitosa
¡Transmisión de Cámara Lista! Ve a: http://192.168.18.77
```

8. Ingresa la dirección IP en tu navegador web. Serás dirigido a la página web mostrada a continuación, donde podrás usar los botones personalizados de ENCENDER y APAGAR para controlar el LED.



9. Inserta una batería en la placa de expansión y retira el cable USB. Ahora puedes colocar el dispositivo en cualquier lugar que desees dentro del alcance del Wi-Fi.



2.47 8.4 Comunicación IoT con MQTT

Este proyecto se centra en el uso de MQTT, un protocolo de comunicación popular en el dominio de Internet de las Cosas (IoT). MQTT permite a los dispositivos IoT intercambiar datos mediante un modelo de publicación/suscripción, donde los dispositivos comunican a través de temas.

En este proyecto, exploramos la implementación de MQTT construyendo un circuito que incluye un LED, un botón y un termistor. Se utiliza el microcontrolador ESP32-WROOM-32E para establecer una conexión WiFi y comunicarse con un broker MQTT. El código permite al microcontrolador suscribirse a temas específicos, recibir mensajes y controlar el LED basado en la información recibida. Adicionalmente, el proyecto demuestra la publicación de datos de temperatura del termistor a un tema designado cuando se presiona el botón.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

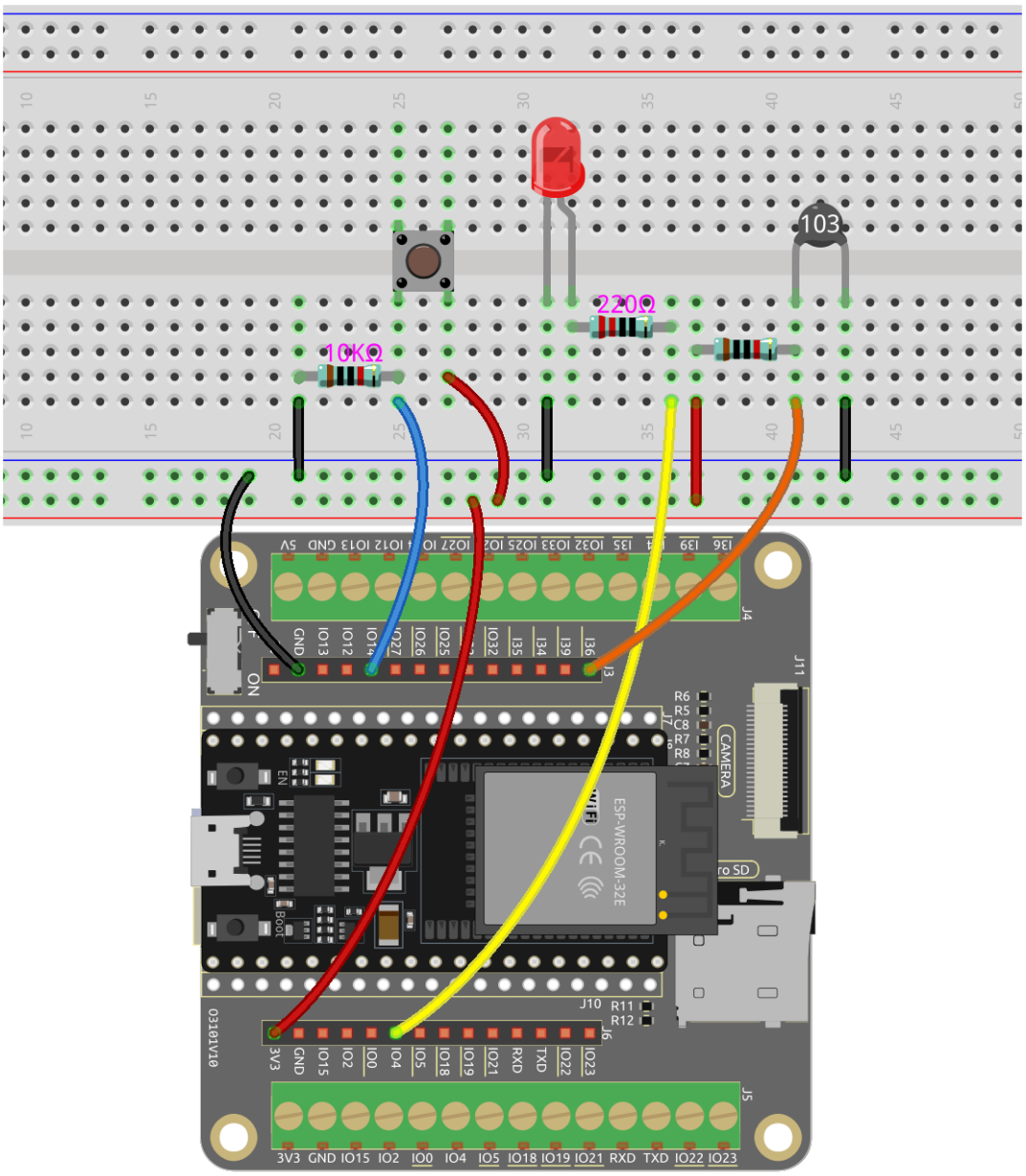
También puedes comprarlos por separado desde los siguientes enlaces.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Botón	
Termistor	

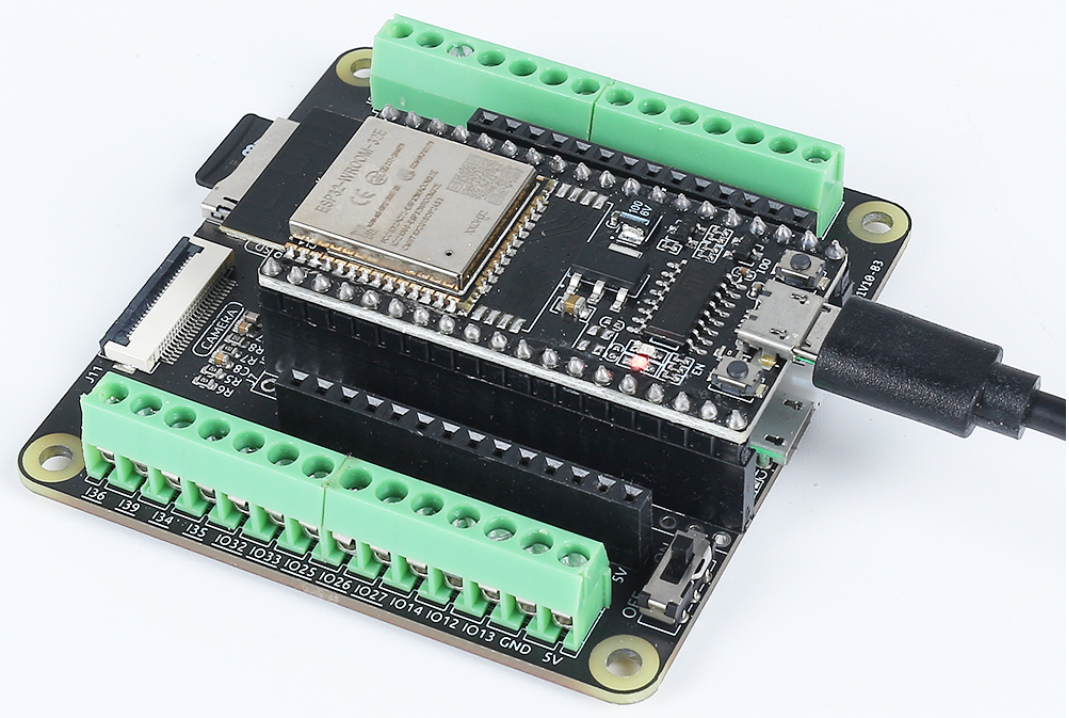
Carga del Código

1. Construye el circuito.

> **Nota:** Al establecer una conexión WiFi, solo se pueden emplear los pines 36, 39, 34, 35, 32, 33 para la lectura analógica. Por favor, asegúrate de que el termistor esté conectado a estos pines designados.

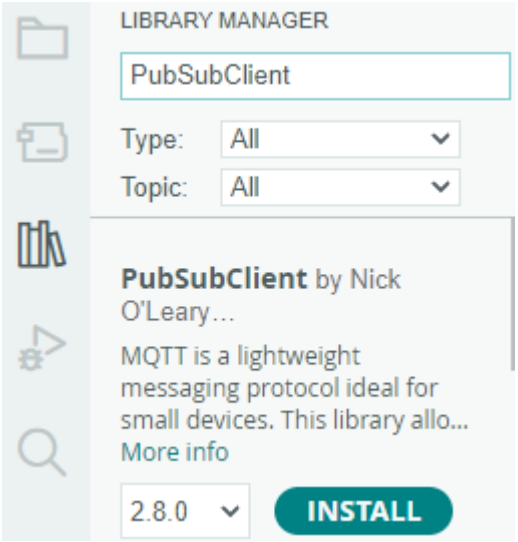


2. Luego, conecta el ESP32-WROOM-32E al computador usando el cable USB.



3. Abre el código.

   - Abre el archivo `iot_4_mqtt.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_4_mqtt`, o copia el código en el IDE de Arduino.
   - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Upload**.
   - *¿Siempre aparece «COMxx desconocido»?*
   - Aquí se utiliza la biblioteca `PubSubClient`, la puedes instalar desde el **Administrador de Bibliotecas**.



4. Localiza las siguientes líneas y modifícalas con tu <SSID> y <PASSWORD>.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Encuentra la siguiente línea y modifica tu `unique_identifier`. Asegúrate de que tu `identificador_único` sea realmente único ya que cualquier ID que sea idéntico intentando iniciar sesión en el mismo broker MQTT puede resultar en un fallo de inicio de sesión.

```
// Add your MQTT Broker address, example:
const char* mqtt_server = "broker.hivemq.com";
const char* unique_identifier = "sunfounder-client-sdgvsda";
```

Suscripción a Temas

1. Para evitar interferencias de mensajes enviados por otros participantes, puedes configurarlo como una cadena oscura o poco común. Simplemente reemplaza el tema actual `SF/LED` con el nombre de tema que desees.

Nota: Tienes la libertad de establecer el Tema con cualquier carácter que desees. Cualquier dispositivo MQTT que se haya suscrito al mismo Tema podrá recibir el mismo mensaje. También puedes suscribirte simultáneamente a múltiples Temas.

```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(unique_identifier)) {
            Serial.println("connected");
            // Subscribe
            client.subscribe("SF/LED");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
```

2. Modifica la funcionalidad para responder al tema suscrito. En el código proporcionado, si se recibe un mensaje en el tema `SF/LED`, verifica si el mensaje es `on` o `off`. Dependiendo del mensaje recibido, cambia el estado de salida para controlar el estado de encendido o apagado del LED.

Nota: Puedes modificarlo para cualquier tema al que estés suscrito, y puedes escribir múltiples declaraciones if para responder a múltiples temas.

```
void callback(char* topic, byte* message, unsigned int length) {
    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
```

(continué en la próxima página)

(proviene de la página anterior)

```
    String messageTemp;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    // If a message is received on the topic "SF/LED", you check if the␣
↪message is either "on" or "off".
    // Changes the output state according to the message
    if (String(topic) == "SF/LED") {
        Serial.print("Changing state to ");
        if (messageTemp == "on") {
            Serial.println("on");
            digitalWrite(ledPin, HIGH);
        } else if (messageTemp == "off") {
            Serial.println("off");
            digitalWrite(ledPin, LOW);
        }
    }
}
```

3. Tras seleccionar la placa correcta (ESP32 Dev Module) y puerto, haz clic en el botón **Subir**.
4. Abre el monitor serial y si se imprime la siguiente información, indica una conexión exitosa al servidor MQTT.

```
WiFi connected
IP address:
192.168.18.77
Attempting MQTT connection...connected
```

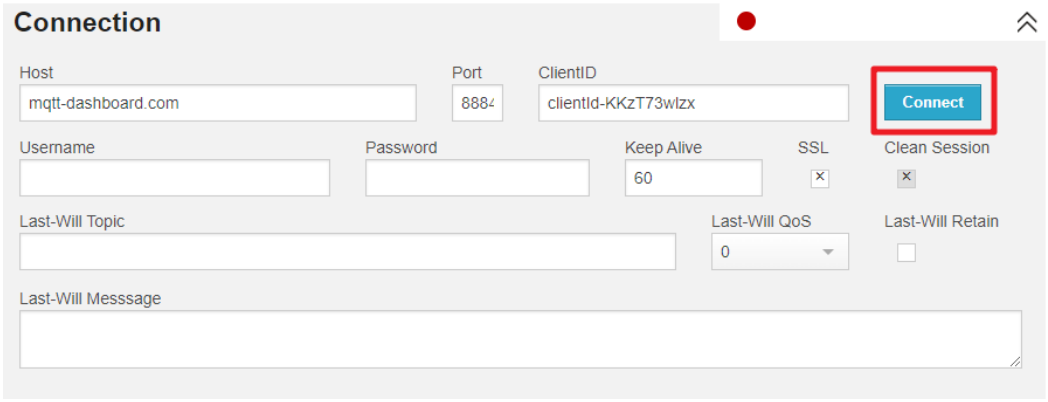
Publicación de Mensajes via HiveMQ

HiveMQ es una plataforma de mensajería que funciona como un broker MQTT, facilitando la transferencia de datos rápida, eficiente y fiable a dispositivos IoT.

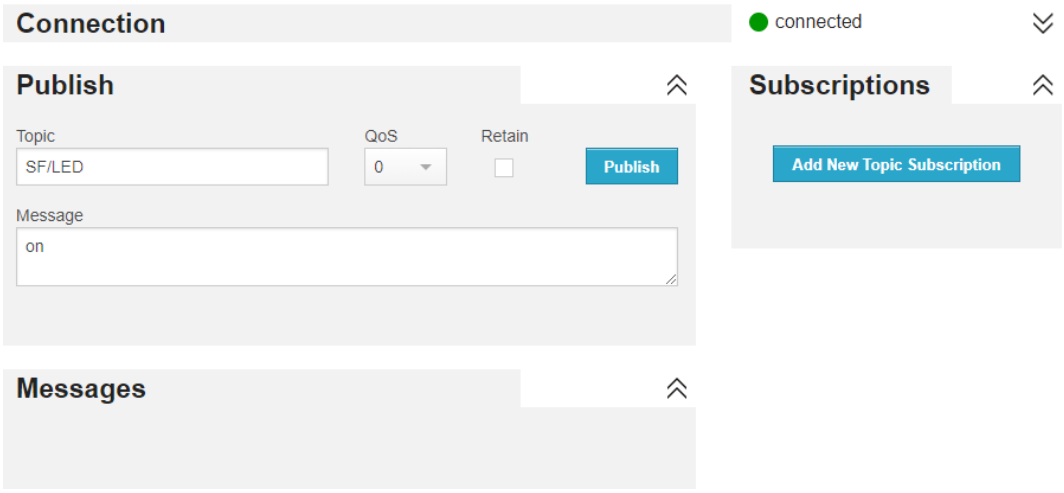
Nuestro código utiliza específicamente el broker MQTT proporcionado por HiveMQ. Hemos incluido la dirección del broker MQTT de HiveMQ en el código de la siguiente manera:

```
// Add your MQTT Broker address, example:
const char* mqtt_server = "broker.hivemq.com";
```

1. Actualmente, abre el en tu navegador web.
2. Conecta el cliente al proxy público predeterminado.



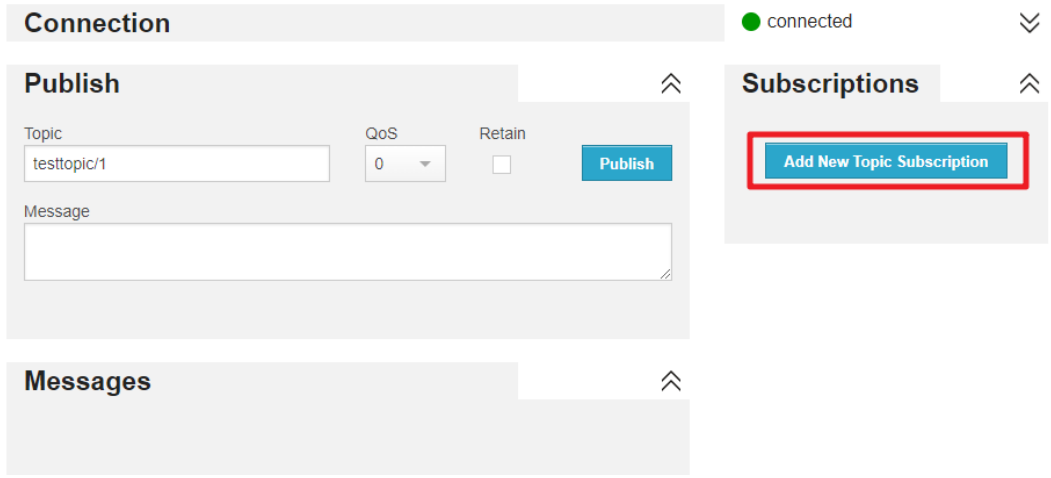
3. Publica un mensaje en el Tema al que te has suscrito. En este proyecto, puedes publicar `on` o `off` para controlar tu LED.



Publicación de Mensajes a MQTT

También podemos utilizar el código para publicar información en el Tema. En esta demostración, hemos codificado una característica que envía la temperatura medida por el termistor al Tema cuando presionas el botón.

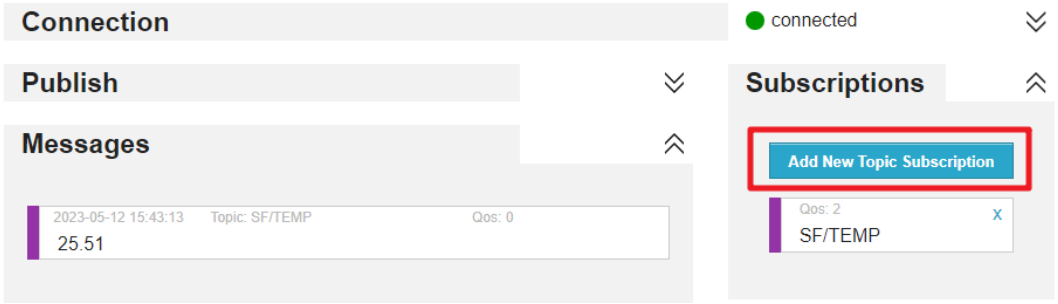
1. Haz clic en **Añadir Nueva Suscripción al Tema**.



2. Rellena los temas que deseas seguir y haz clic en **Suscribirse**. En el código, enviamos información de la temperatura al tema `SF/TEMP`.

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    // if the button pressed, publish the temperature to topic "SF/TEMP"
    if (digitalRead(buttonPin)) {
            long now = millis();
            if (now - lastMsg > 5000) {
            lastMsg = now;
            char tempString[8];
            dtostrf(thermistor(), 1, 2, tempString);
            client.publish("SF/TEMP", tempString);
        }
    }
}
```

3. Por lo tanto, podemos monitorear este Tema en HiveMQ, permitiéndonos ver la información que has publicado.



2.48 8.5 CheerLights

CheerLights es una red global de luces sincronizadas que pueden ser controladas por cualquiera.

Únete a la comunidad de cambio de color de LEDs , que permite a los LEDs alrededor del mundo cambiar de color simultáneamente.

Puedes colocar tus LEDs en un rincón de tu oficina para recordarte que no estás solo.

En este caso, también utilizamos MQTT, pero en lugar de publicar nuestros propios mensajes, nos suscribimos al tema «cheerlights». Esto nos permite recibir mensajes enviados por otros al tema «cheerlights» y usar esa información para cambiar el color de nuestra tira de LED en consecuencia.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

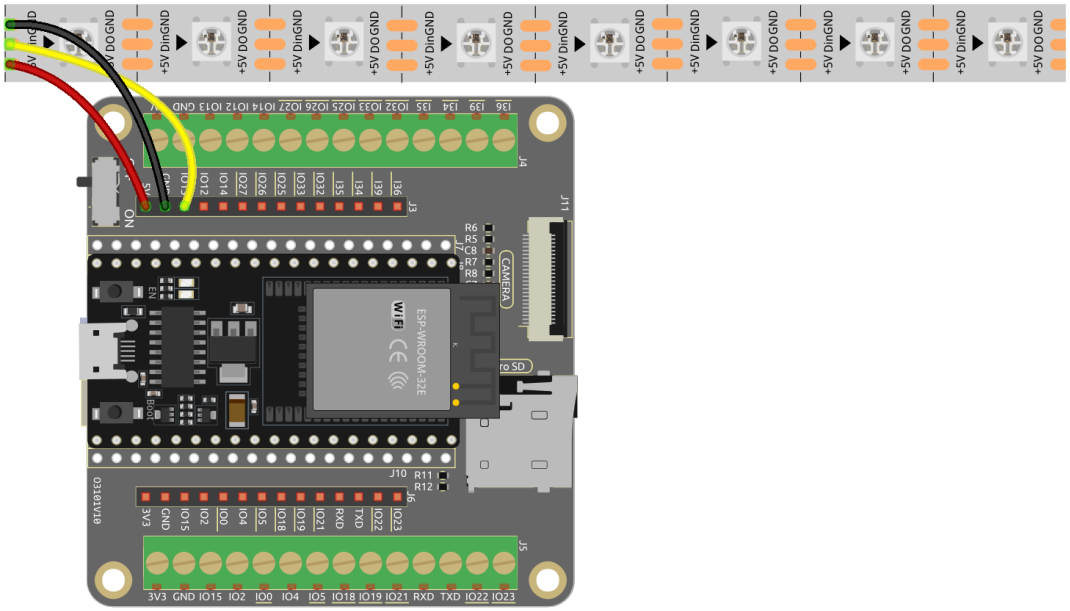
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

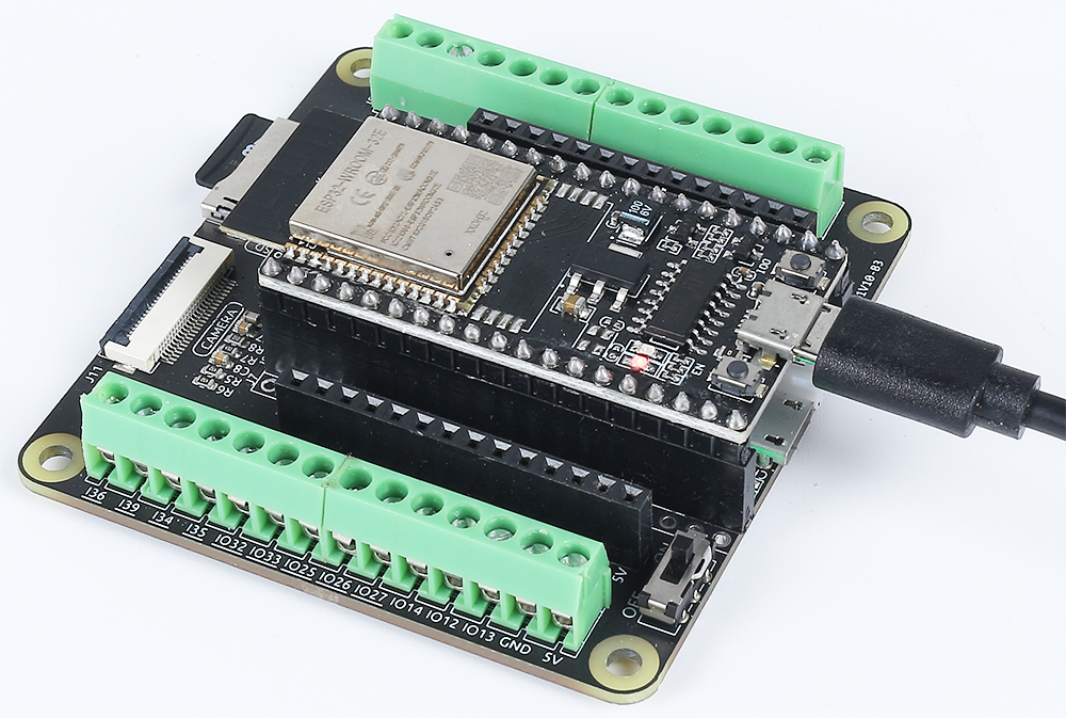
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Tira de 8 LEDs RGB WS2812	

¿Cómo hacerlo?

1. Construye el circuito.

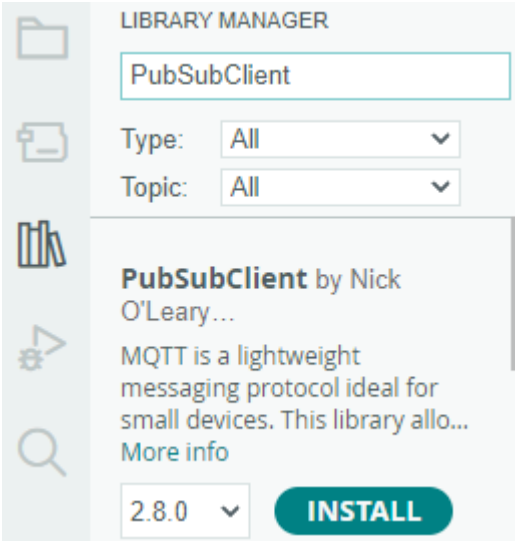


2. Luego, conecta el ESP32-WROOM-32E al computador usando el cable USB.



3. Abre el código.

   - Abre el archivo `iot_5_cheerlights.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_5_cheerlights`, o copia el código en el IDE de Arduino.
   - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
   - *¿Siempre aparece «COMxx desconocido»?*
   - Aquí se utilizan las bibliotecas `PubSubClient` y `Adafruit_NeoPixel`, puedes instalarlas desde el **Gestor de Bibliotecas**.



4. Localiza las siguientes líneas y modifícalas con tu `<SSID>` y `<PASSWORD>`.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Encuentra la siguiente línea y modifica tu `identificador_único`. Asegúrate de que tu `identificador_único` sea verdaderamente único ya que cualquier ID idéntico que intente iniciar sesión en el mismo **Broker MQTT** puede resultar en un fallo de inicio de sesión.

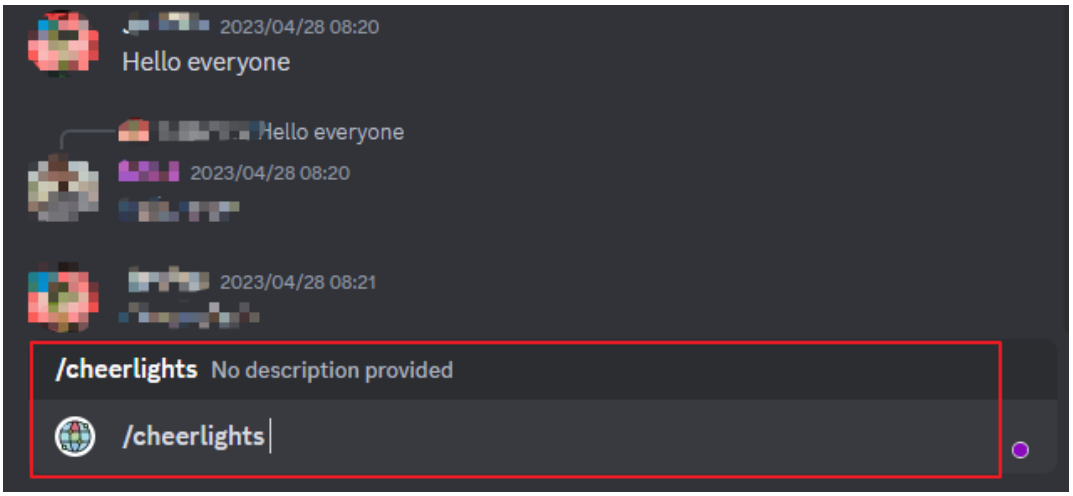
```
// Add your MQTT Broker address:
const char* mqtt_server = "mqtt.cheerlights.com";
const char* unique_identifier = "sunfounder-client-sdgvsasdda";
```

6. Después de seleccionar la placa correcta (ESP32 Dev Module) y el puerto, haz clic en el botón **Subir**.
7. En este punto, puedes ver que tu tira RGB muestra un cierto color. Colócala en tu escritorio y notarás que cambia de color periódicamente. ¡Esto se debe a que otros seguidores de @CheerLights están cambiando el color de tus luces!
8. Abre el Monitor Serie. Verás mensajes similares a los siguientes:

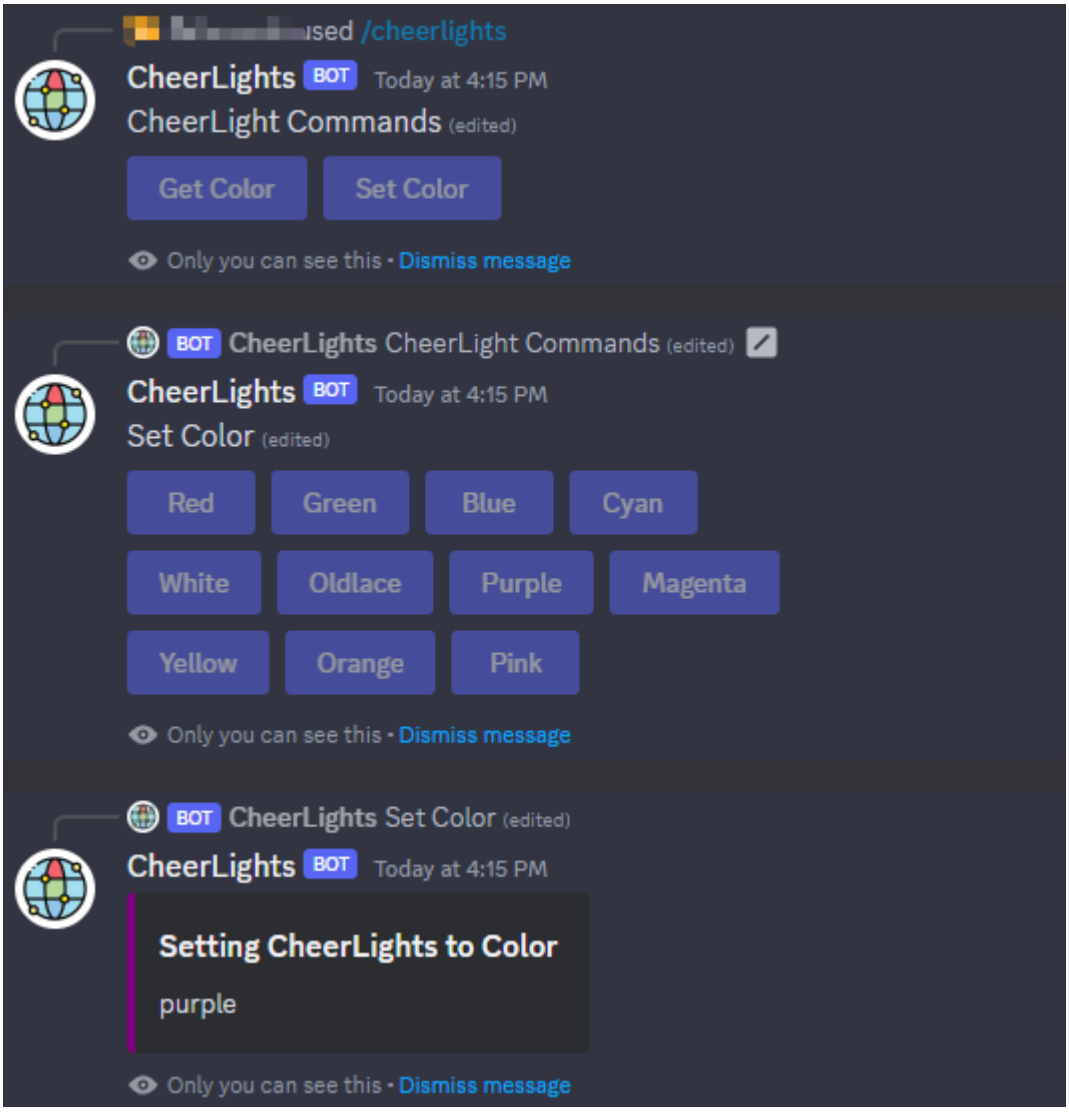
```
WiFi connected
IP address:
192.168.18.77
Attempting MQTT connection...connected
Message arrived on topic: cheerlights.
Message: oldlace
Changing color to oldlace
```

Controla los dispositivos globales @CheerLights

1. Únete al y utiliza el bot de CheerLights para establecer el color. Simplemente escribe `/cheerlights` en cualquiera de los canales del **Servidor de Discord de CheerLights** para activar el bot.



2. Sigue las instrucciones proporcionadas por el bot para establecer el color. Esto te permitirá controlar dispositivos CheerLights globalmente.



2.49 8.6 Monitoreo de Temperatura y Humedad con Adafruit IO

En este proyecto, te guiaremos sobre cómo usar una plataforma de IoT popular. Hay muchas plataformas gratuitas (o de bajo costo) disponibles en línea para los entusiastas de la programación. Algunos ejemplos son Adafruit IO, Blynk, Arduino Cloud, ThingSpeak, etc. El uso de estas plataformas es bastante similar. Aquí, nos centraremos en Adafruit IO.

Escribiremos un programa de Arduino que utiliza el sensor DHT11 para enviar lecturas de temperatura y humedad al tablero de Adafruit IO. También puedes controlar un LED en el circuito a través de un interruptor en el tablero.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

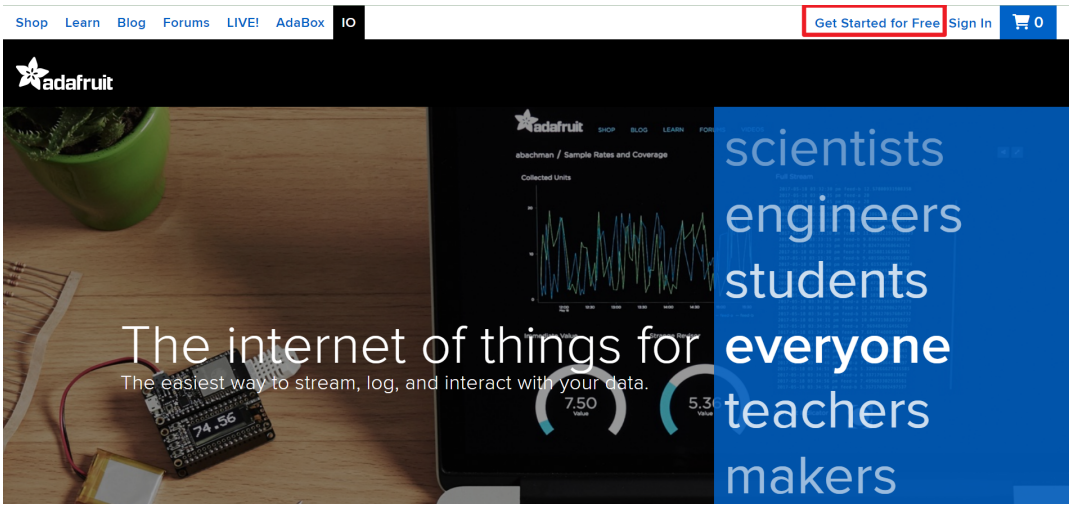
Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado desde los enlaces a continuación.

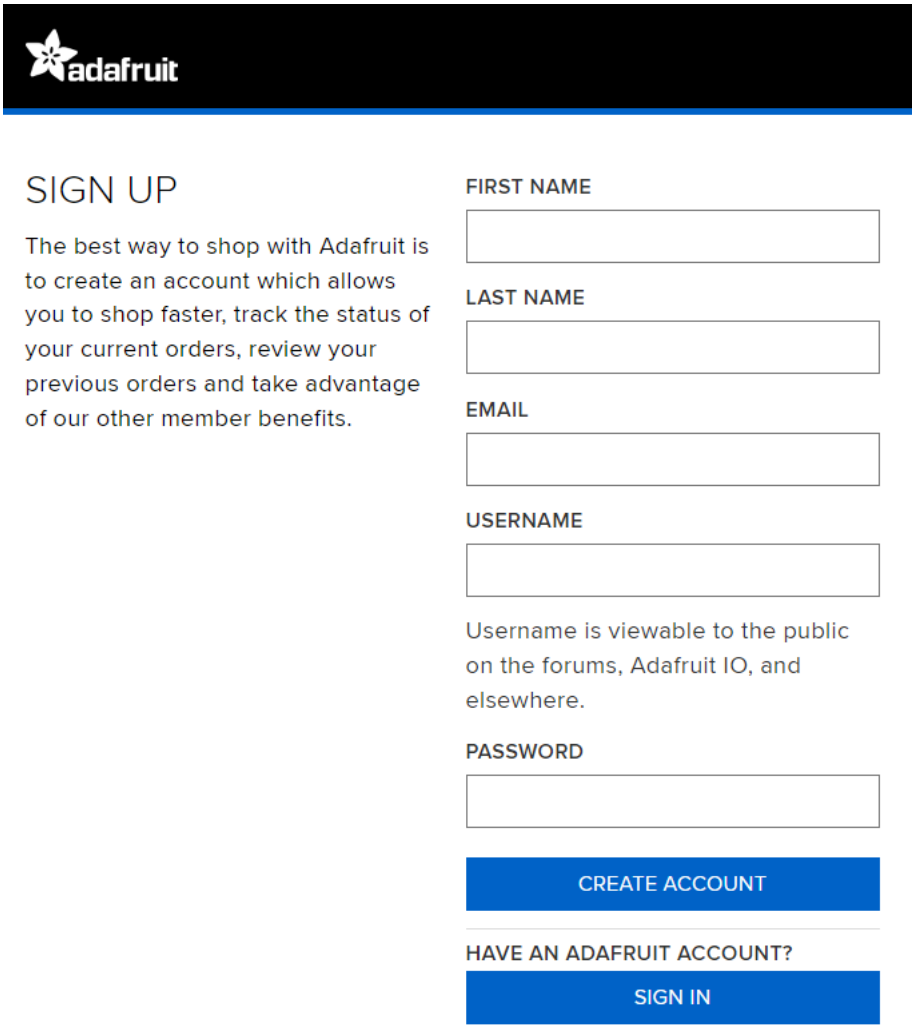
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Sensor de Humedad y Temperatura DHT11	

Configuración del Tablero

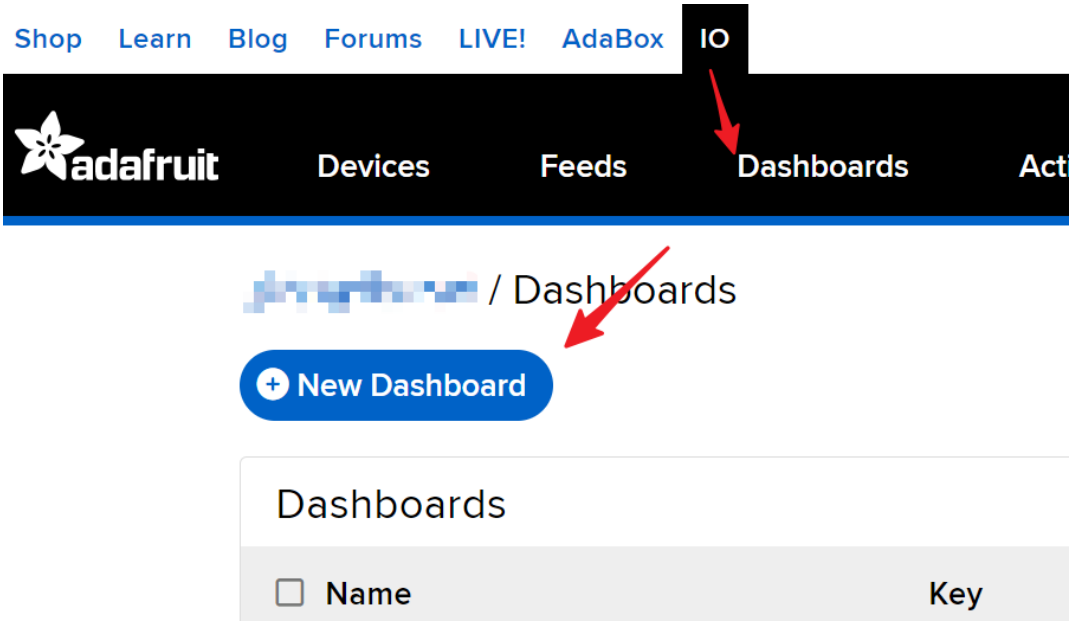
1. Visita , luego haz clic en **Comenzar gratis** para crear una cuenta gratuita.



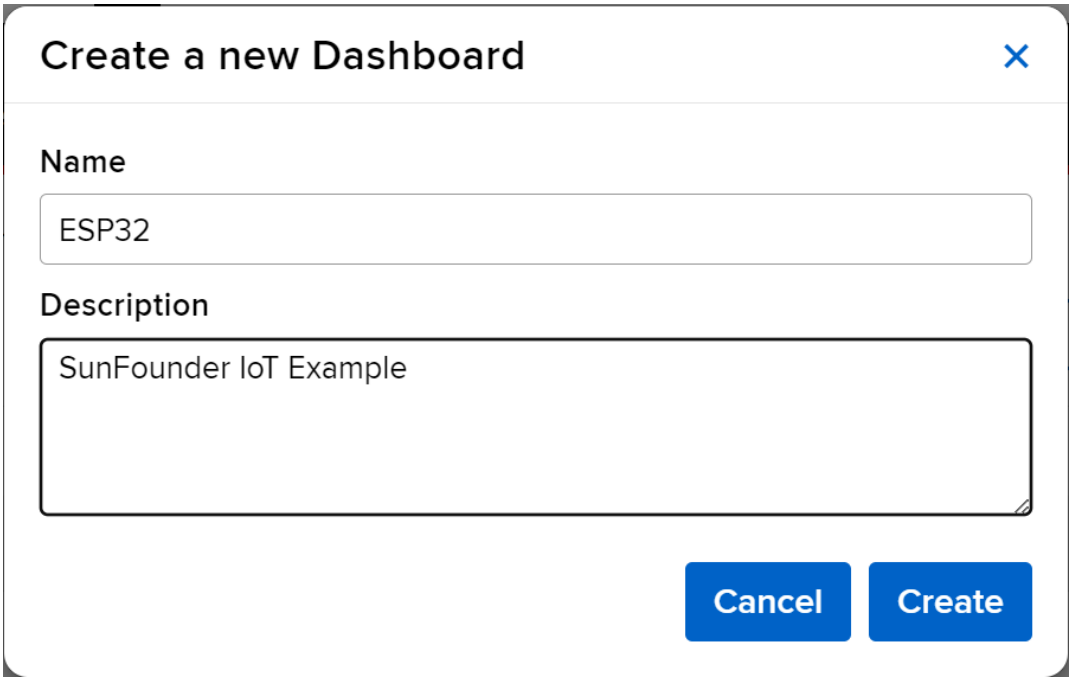
2. Completa el formulario para crear una cuenta.



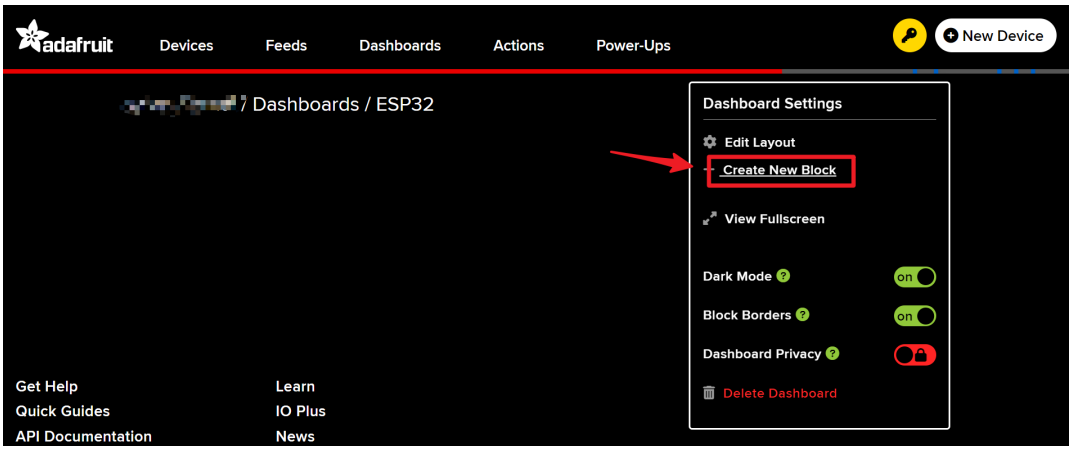
3. Después de crear una cuenta en Adafruit, necesitarás reabrir Adafruit io. Haz clic en **Tableros**, luego en **Nuevo Tablero**.



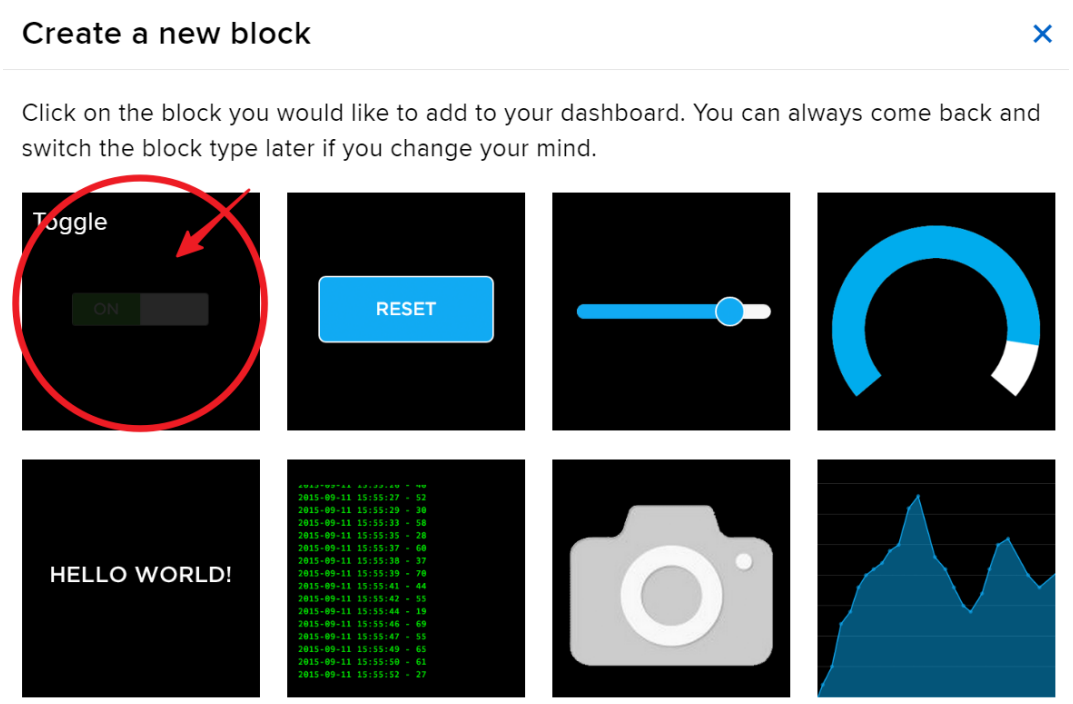
4. Crea un **Nuevo Tablero**.



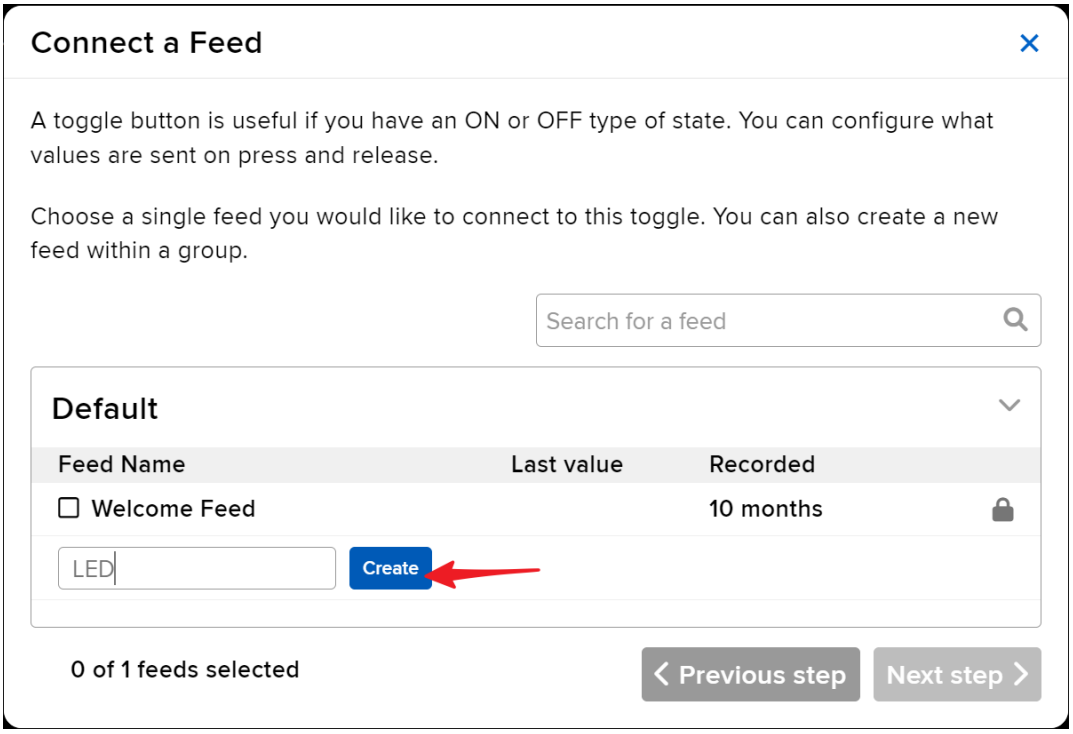
5. Ingresa al **Tablero** recién creado y crea un nuevo bloque.



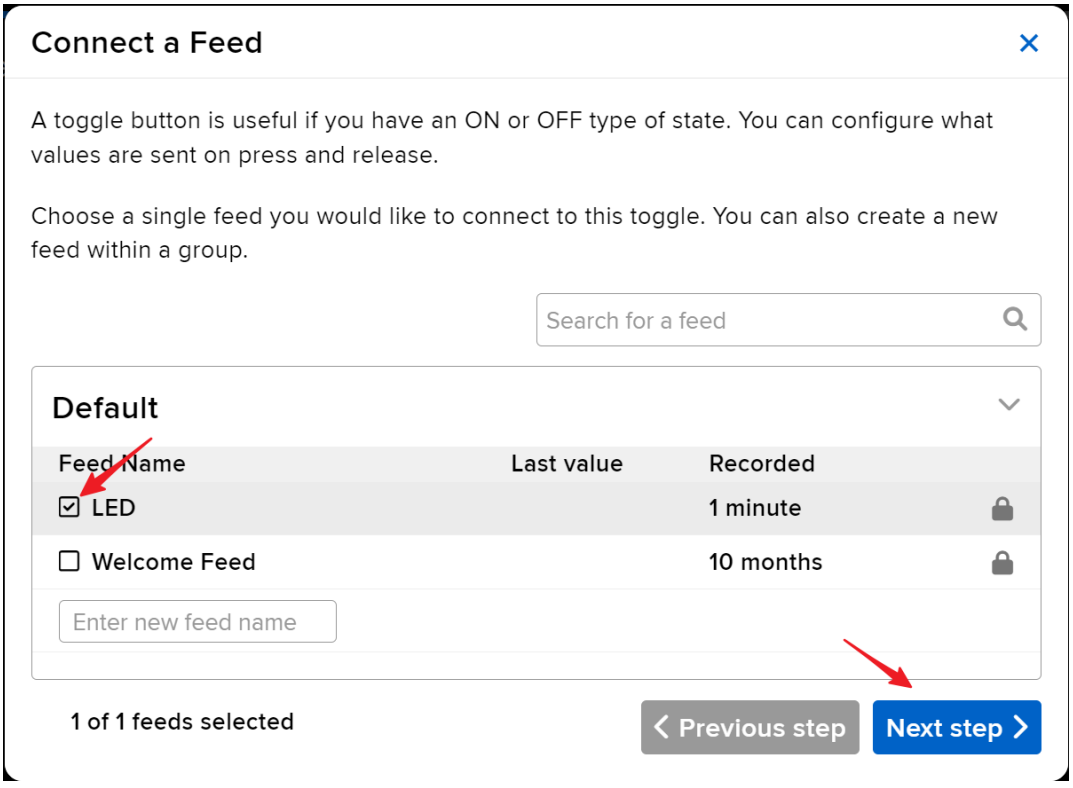
6. Crea 1 bloque **Interruptor**.



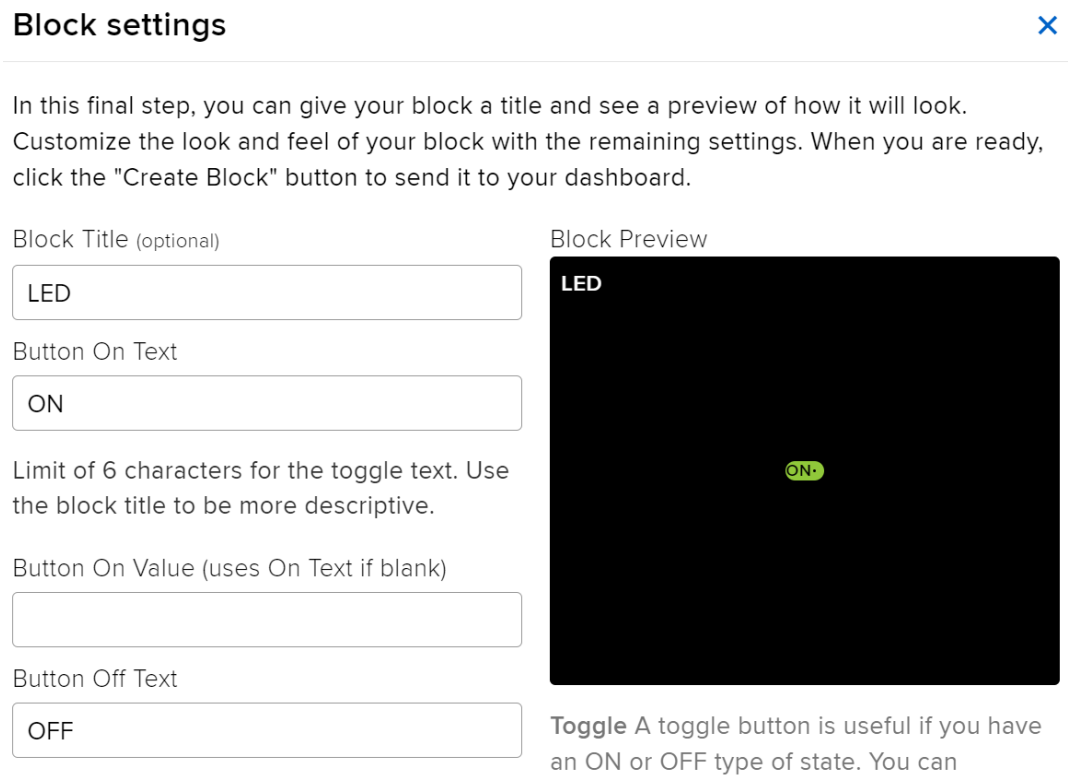
7. A continuación, necesitarás crear un nuevo canal aquí. Este interruptor se utilizará para controlar el LED, y nombraremos este canal «LED».



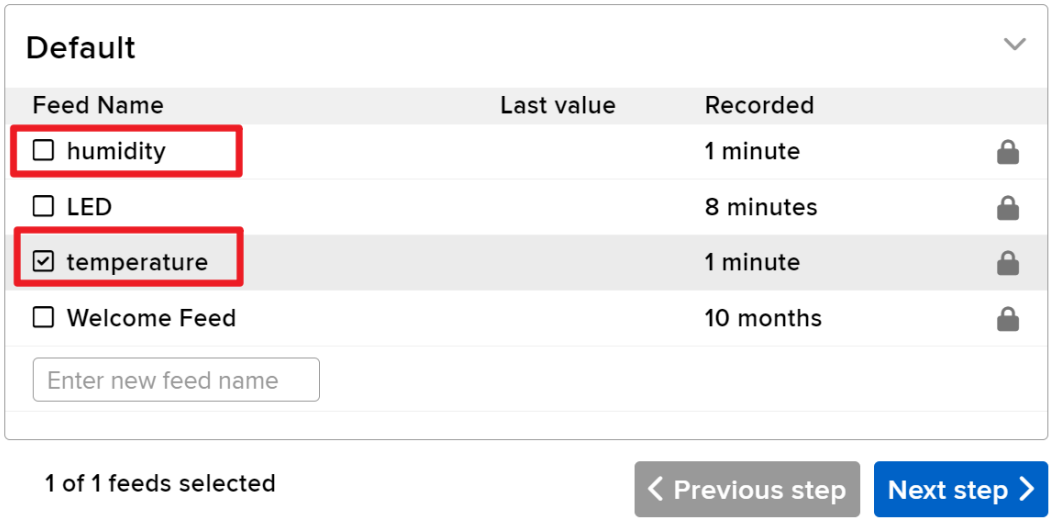
8. Verifica el canal **LED**, luego avanza al siguiente paso.



9. Completa la configuración del bloque (principalmente Título del Bloque, Texto de Encendido y Texto de Apagado), luego haz clic en el botón **Crear bloque** en la parte inferior derecha para finalizar.

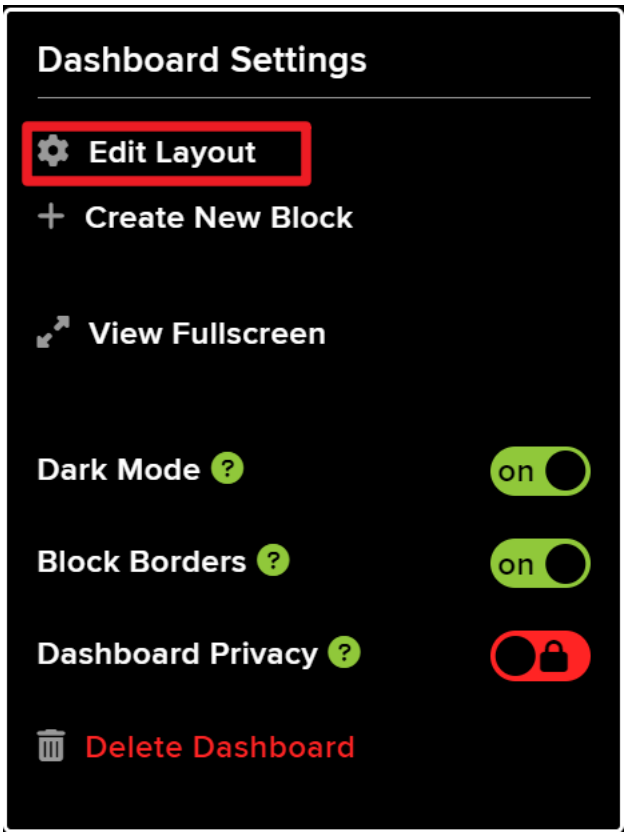


10. También necesitaremos crear dos **Bloques de Texto** a continuación. Se utilizarán para mostrar la temperatura y la humedad. Por lo tanto, crea dos canales denominados **temperatura** y **humedad**.

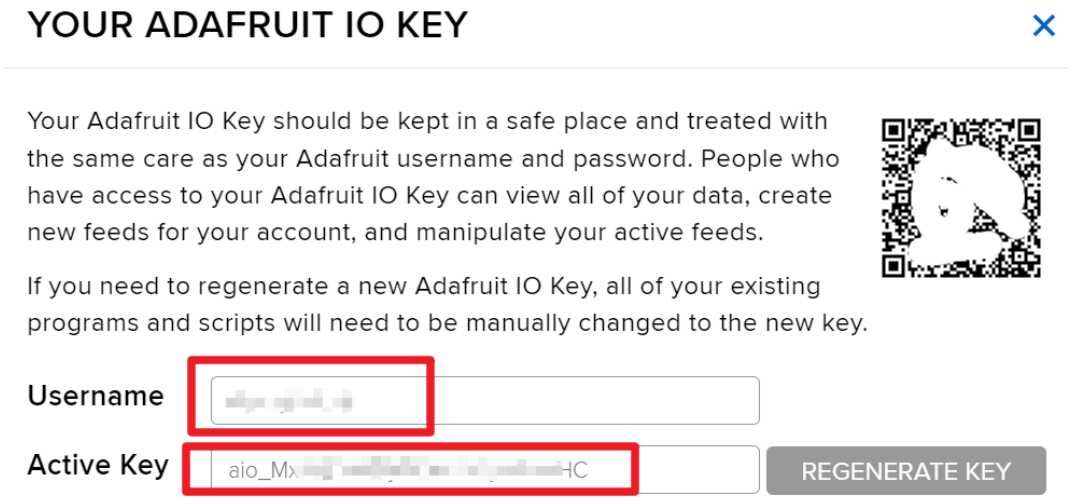


11. Después de la creación, tu Tablero debería verse algo así:

12. Puedes ajustar el diseño utilizando la opción **Editar Diseño** en el Tablero.

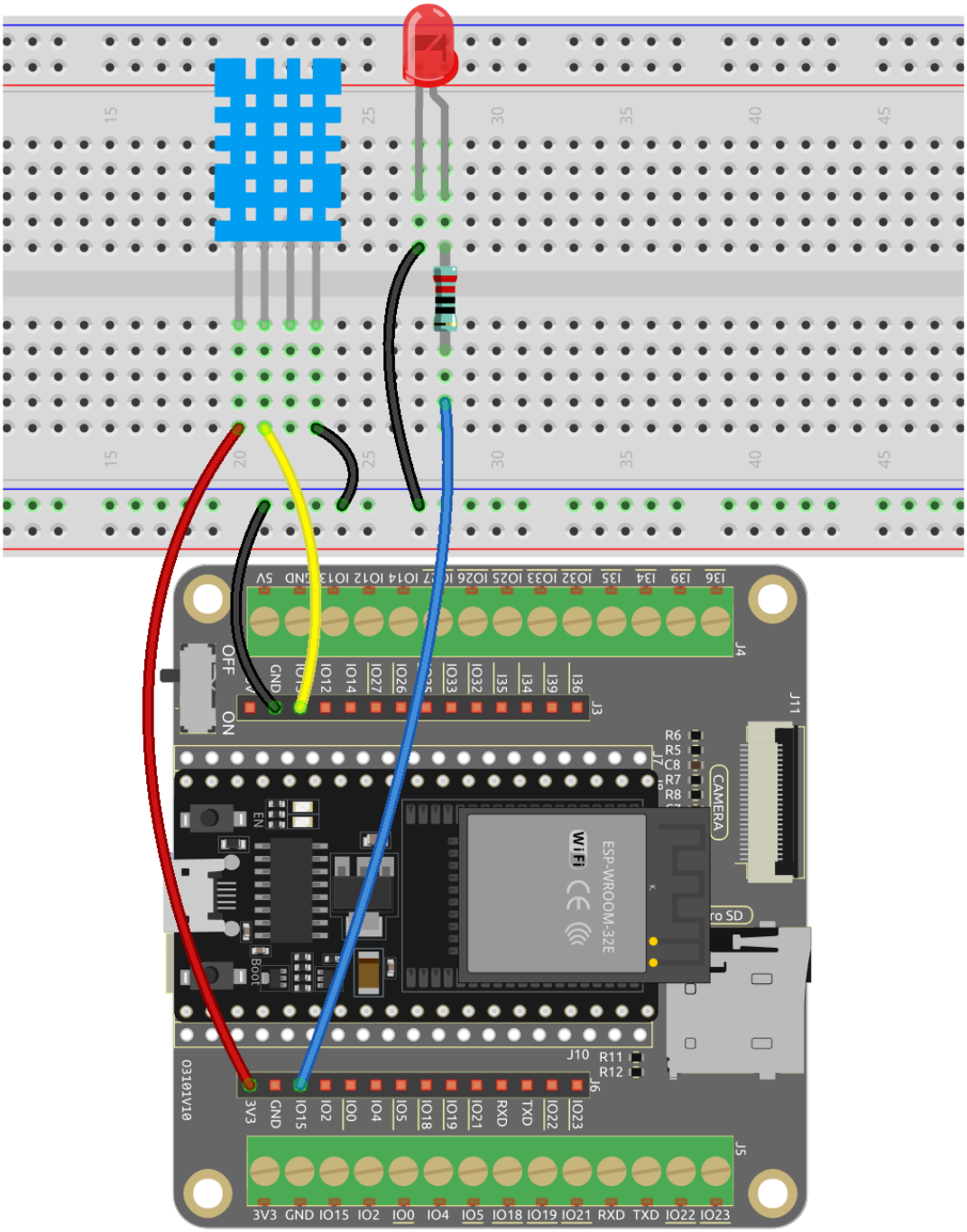


13. Haz clic en **CLAVE API**, y verás tu nombre de usuario y **CLAVE API** mostrados. Anótalos, ya que los necesitarás para tu código.

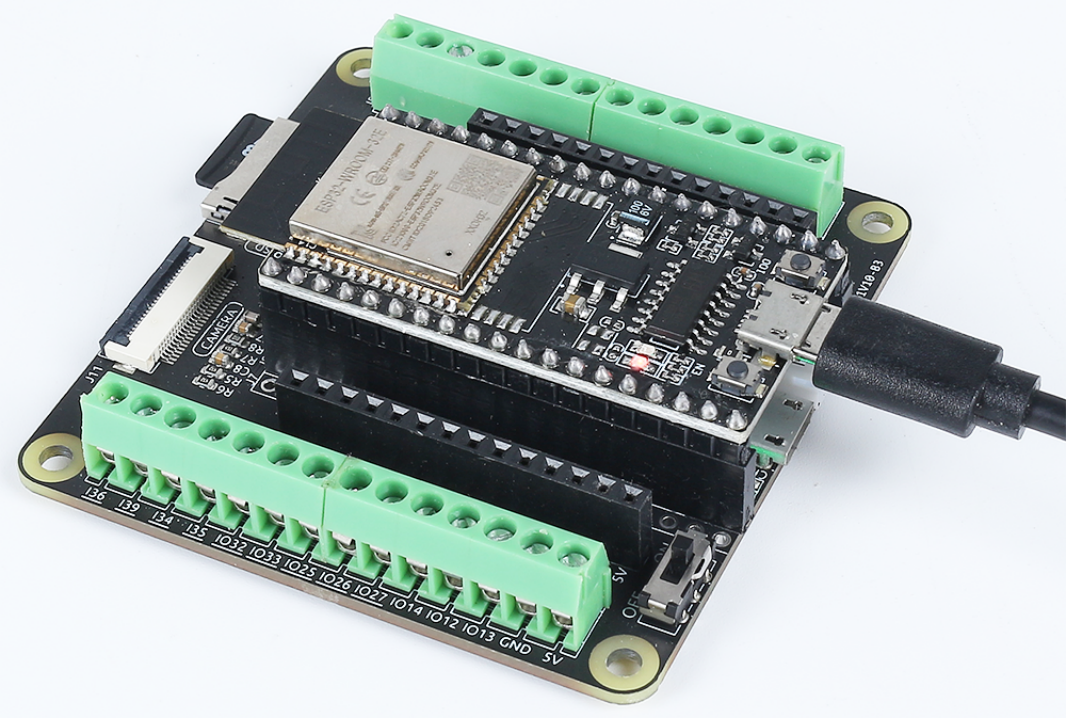


Ejecutando el Código

1. Construye el circuito.



2. Luego, conecta el ESP32-WROOM-32E al ordenador mediante el cable USB.



3. Abre el código.

    - Abre el archivo `iot_6_adafruit_io.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_6_adafruit_io`, o copia el código en el IDE de Arduino.
    - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
    - *¿Siempre aparece «COMxx desconocido»?*
    - Aquí se utilizan la `Biblioteca Adafruit_MQTT` y la `Biblioteca del sensor DHT`, puedes instalarlas desde el **Gestor de Bibliotecas**.

4. Encuentra las siguientes líneas y reemplaza `<SSID>` y `<PASSWORD>` con los detalles específicos de tu red WiFi.

```
/************************* WiFi Access Point␣
↪*********************************/

#define WLAN_SSID "<SSID>"
#define WLAN_PASS "<PASSWORD>"
```

5. Luego reemplaza `<TU_NOMBRE_DE_USUARIO_ADAFRUIT_IO>` con tu nombre de usuario de Adafruit IO y `<TU_CLAVE_ADAFRUIT_IO>` con la **CLAVE API** que acabas de copiar.

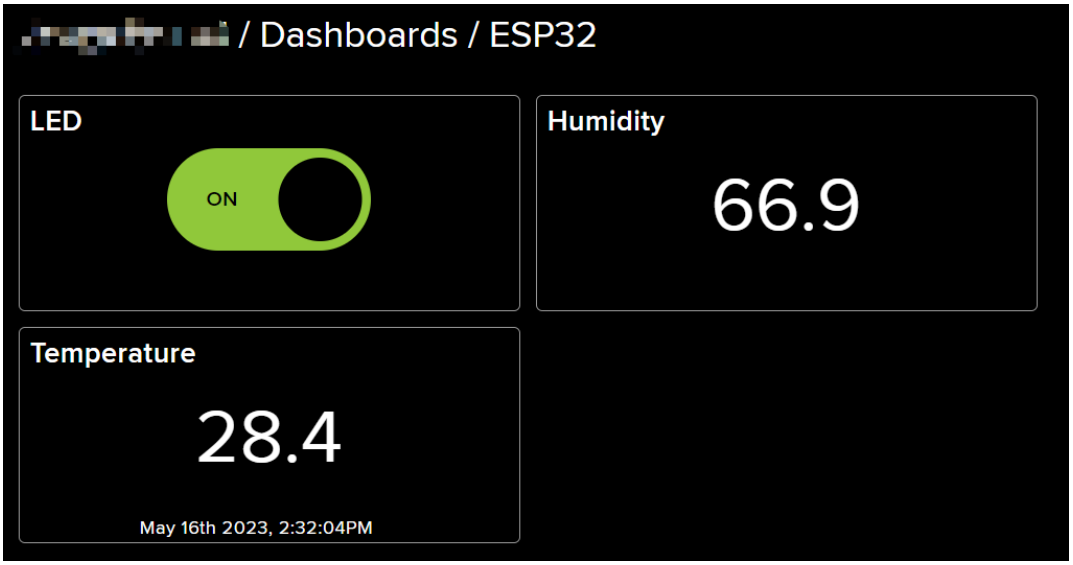
```
// Adafruit IO Account Configuration
// (to obtain these values, visit https://io.adafruit.com and click on␣
↪Active Key)
#define AIO_USERNAME "<YOUR_ADAFRUIT_IO_USERNAME>"
#define AIO_KEY      "<YOUR_ADAFRUIT_IO_KEY>"
```

6. Después de seleccionar la placa correcta (ESP32 Dev Module) y el puerto correspondiente, haz clic en el botón **Subir** para cargar el programa a tu ESP32.

7. Una vez que el código se haya subido con éxito, observarás el siguiente mensaje en el monitor serial, indicando una comunicación exitosa con Adafruit IO.

```
Adafruit IO MQTTS (SSL/TLS) Example


Connecting to xxxxx
WiFi connected
IP address:
192.168.18.76
Connecting to MQTT... MQTT Connected!
Temperature: 27.10
Humidity: 61.00
```

8. Regresa a Adafruit IO. Ahora puedes observar las lecturas de temperatura y humedad en el tablero de control, o utilizar el interruptor de palanca LED para controlar el estado de encendido/apagado del LED externo conectado al circuito.



2.50 8.7 Cámara ESP con Bot de Telegram

En este proyecto, demostraremos cómo integrar el ESP32 con tu aplicación de mensajería favorita. Para esta demostración, usaremos Telegram.

Crea un Bot de Telegram, permitiéndote controlar tu circuito desde cualquier lugar, capturar fotos y gestionar el flash. Además, cada vez que alguien pase por tu dispositivo, tomará una nueva foto y enviará una notificación a tu cuenta de Telegram.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

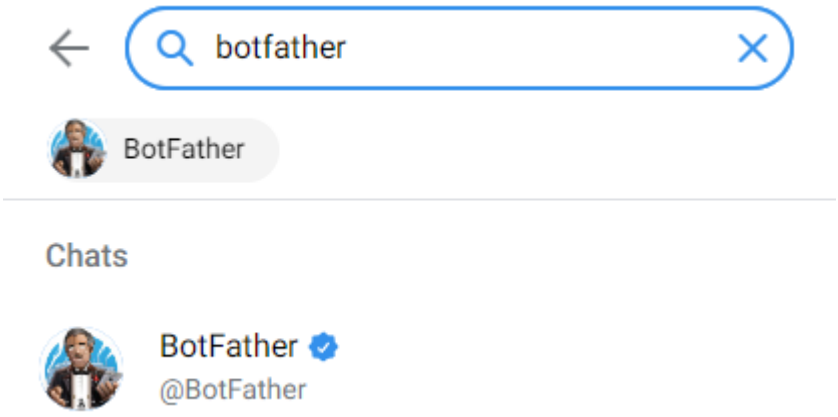
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

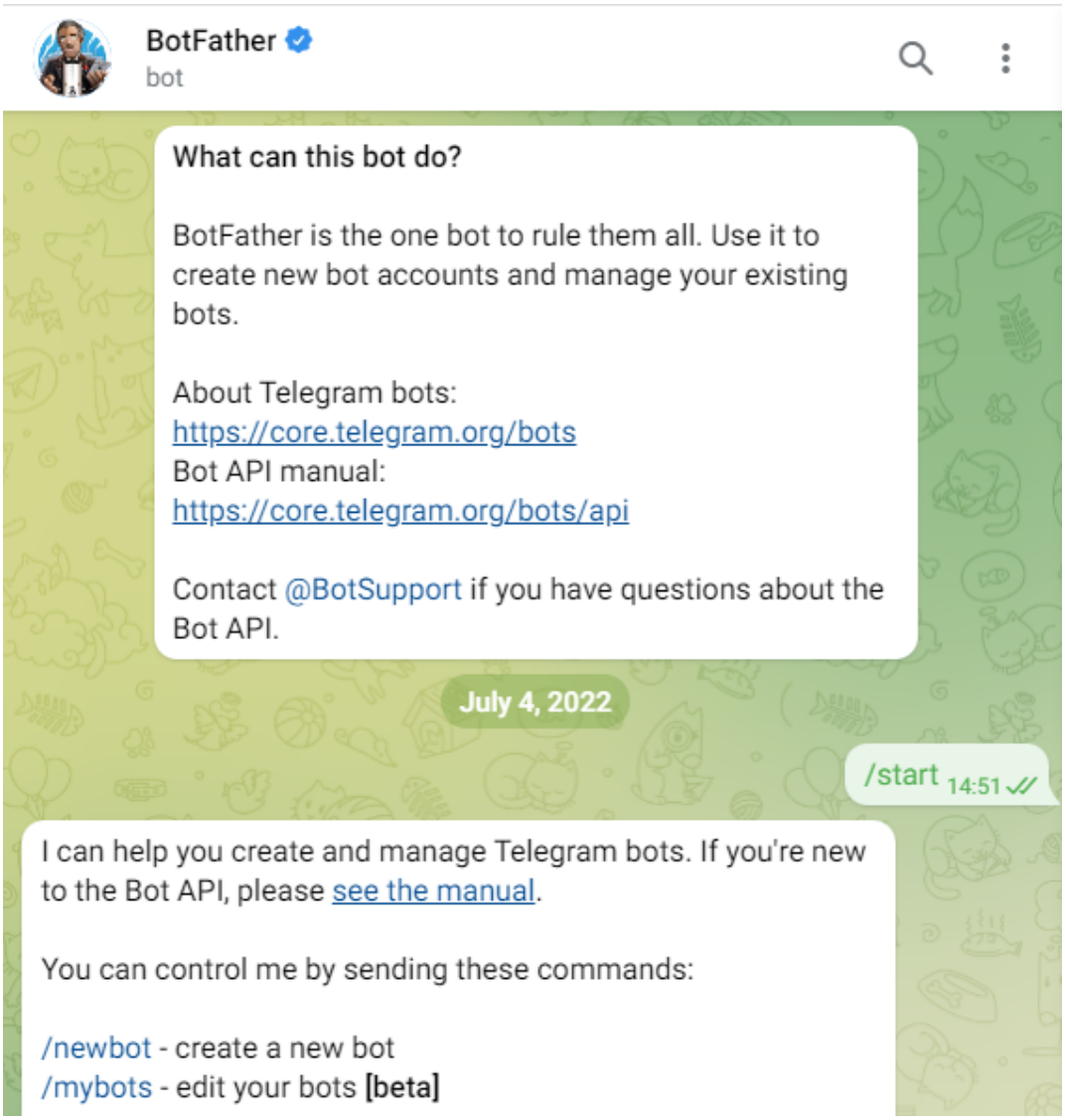
INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Módulo Sensor de Movimiento PIR	

Creando un Bot de Telegram

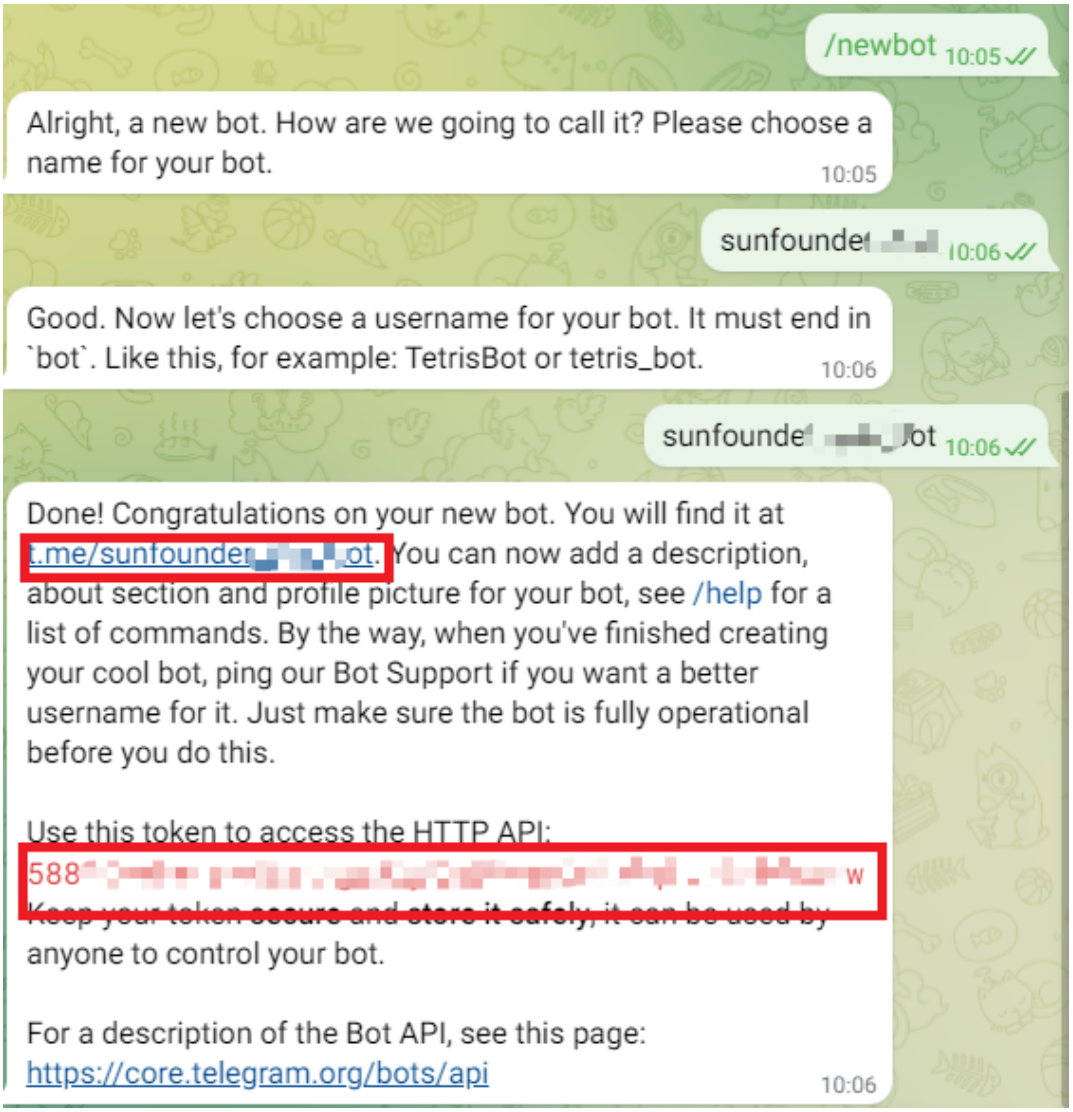
1. Dirígete a **Google Play** o a la **App Store** y descarga e instala **Telegram**.
2. Busca `botfather` en la aplicación de **Telegram** y una vez que aparezca, haz clic en él para abrirlo. O puedes acceder directamente a este enlace: t.me/botfather.



3. Al abrirlo, se presentará una ventana de chat. Envía el comando `/start`.



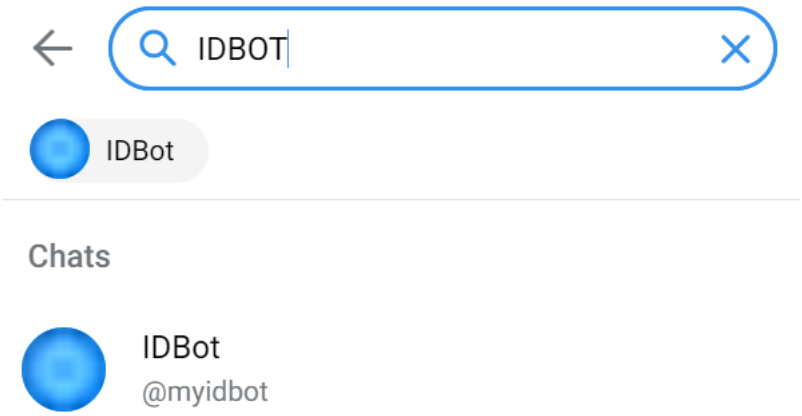
4. Introduce `/newbot` y sigue las instrucciones proporcionadas para crear tu bot. Una vez exitoso, el BotFather te proporcionará el enlace de acceso y la API para tu nuevo bot.



Autorizando Usuarios de Telegram

Como cualquiera puede interactuar con el bot que has creado, existe un riesgo de fuga de información. Para solucionar esto, queremos que el bot solo responda a usuarios autorizados.

1. En tu cuenta de **Telegram**, busca `IDBot` o abre el enlace: t.me/myidbot.

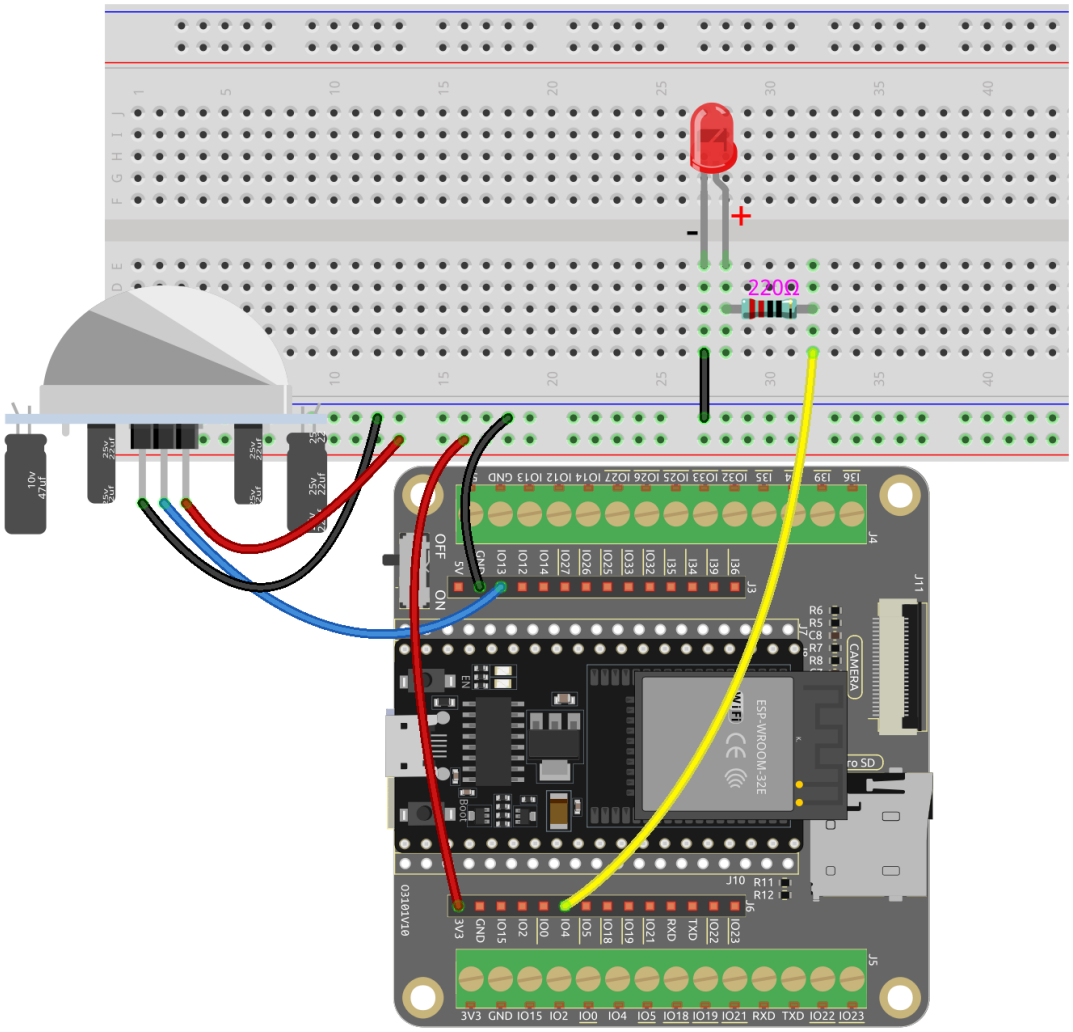


2. Envía el comando `/getid`. Guarda el ID proporcionado para su uso posterior en nuestro programa.

Subir el Código

1. Primero conecta la cámara.

2. Construye el circuito.



3. Abre el código.

    - Abre el archivo `iot_7_cam_telegram.ino` ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_7_cam_telegram` o copia el código en el IDE de Arduino.
    - Después de seleccionar la placa (ESP32 Dev Module) y el puerto apropiado, haz clic en el botón **Subir**.
    - *¿Siempre aparece «COMxx desconocido»?*
    - Se utilizan las bibliotecas `UniversalTelegramBot` y `ArduinoJson`, puedes instalarlas desde el **Administrador de Bibliotecas**.

4. Localiza y modifica las siguientes líneas con los detalles de tu WiFi, reemplazando `<SSID>` y `<PASSWORD>`:

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Actualiza la siguiente línea, reemplazando `<CHATID>` con tu ID de Telegram, que obtuviste de @IDBot.

```
// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
String chatId = "<CHATID>";
```

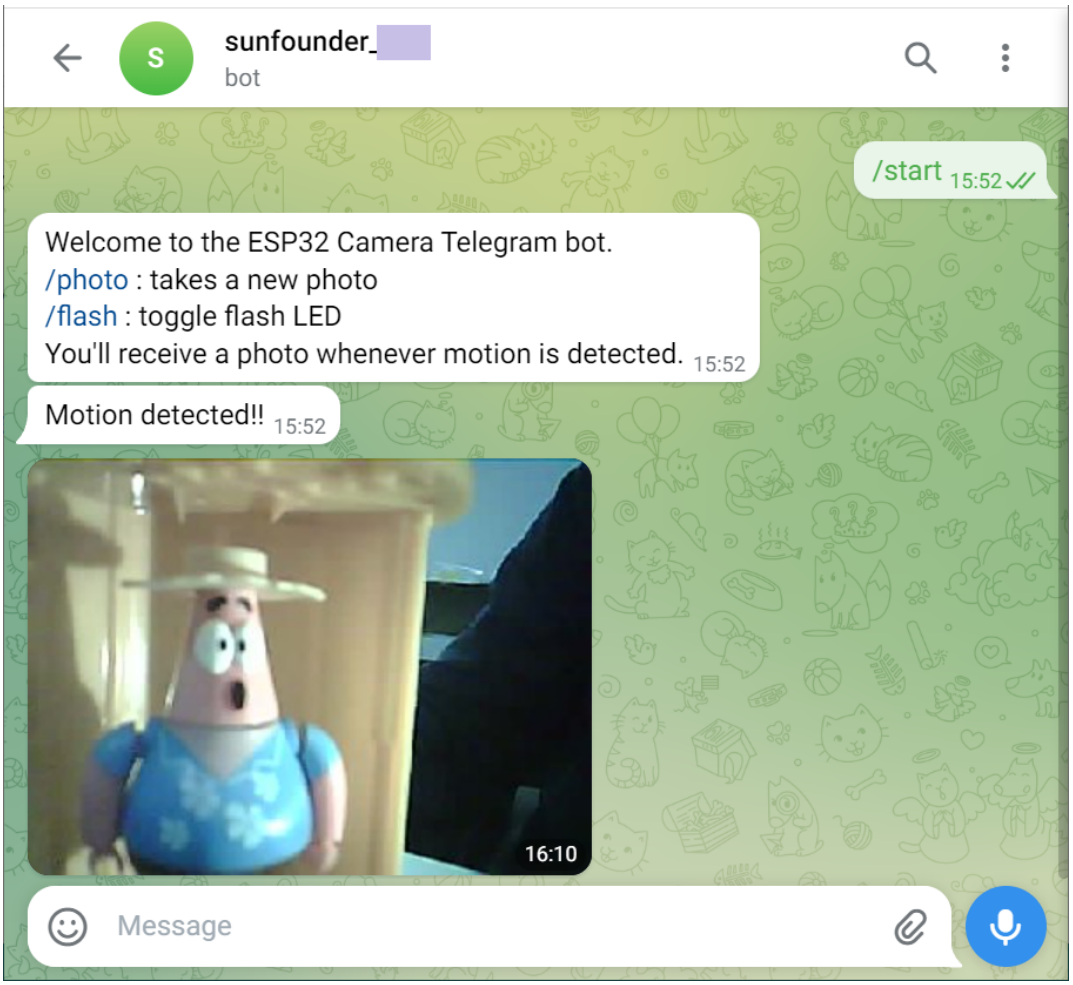
6. Actualiza la siguiente línea, sustituyendo <BOTTOKEN> con el token de tu BOT de Telegram, que fue proporcionado por @BotFather.

```
// Initialize Telegram BOT
String BOTtoken = "<BOTTOKEN>";
```

7. Después de seleccionar la placa correcta (ESP32 Dev Module) y el puerto, haz clic en el botón **Subir**.
8. Abre el Monitor Serial. Si se imprime una dirección IP, esto indica una ejecución exitosa.

```
Connecting to xxxx
ESP32-CAM IP Address: 192.168.18.76
Init Done!
```

9. Ahora, puedes interactuar con tu ESP32 a través de Telegram.



2.51 8.8 Cámara con Home Assistant

Este proyecto te guiará en la configuración de un servidor de transmisión de video para la cámara ESP32 e integrarlo con la popular plataforma de automatización del hogar, Home Assistant. Esta integración te permitirá acceder al servidor desde cualquier dispositivo en tu red.

Nota: Antes de sumergirte en este proyecto, necesitas tener un sistema operativo con Home Assistant instalado.

Recomendamos instalar Home Assistant OS en un Raspberry Pi.

Si no tienes un Raspberry Pi, también puedes instalarlo en una máquina virtual que funcione en Windows, macOS o Linux.

Para instrucciones de instalación, consulta el enlace oficial: https://www.home-assistant.io/installation/

Por favor, procede con este proyecto solo después de una instalación exitosa.

Componentes Requeridos

En este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ÍTEMS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

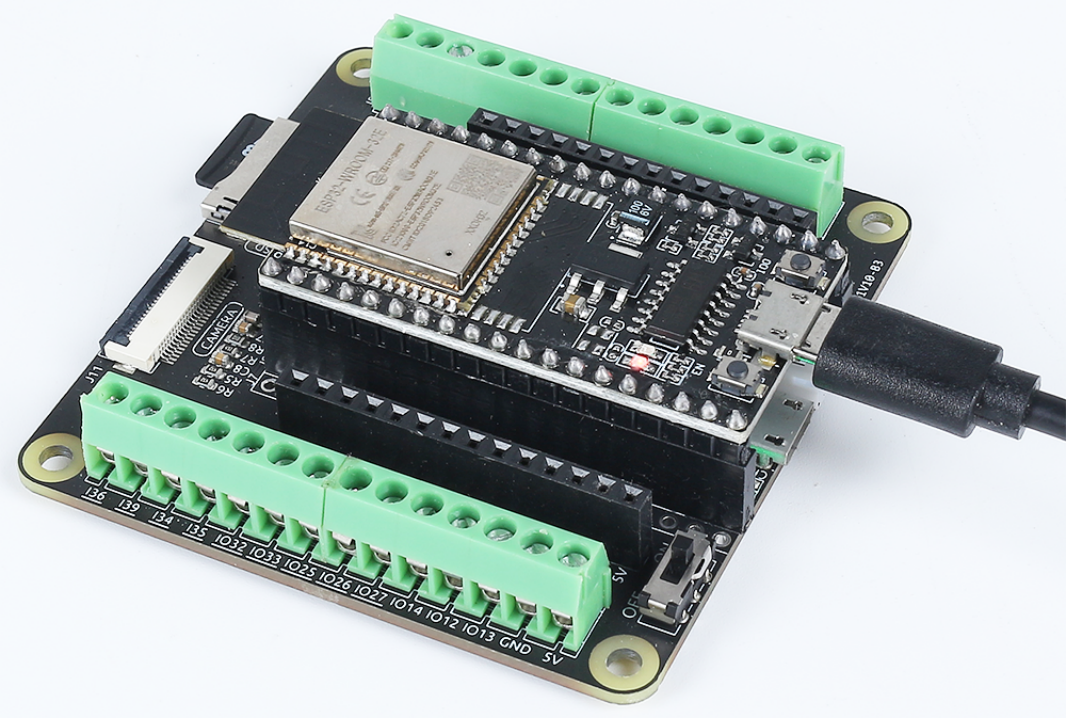
También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-

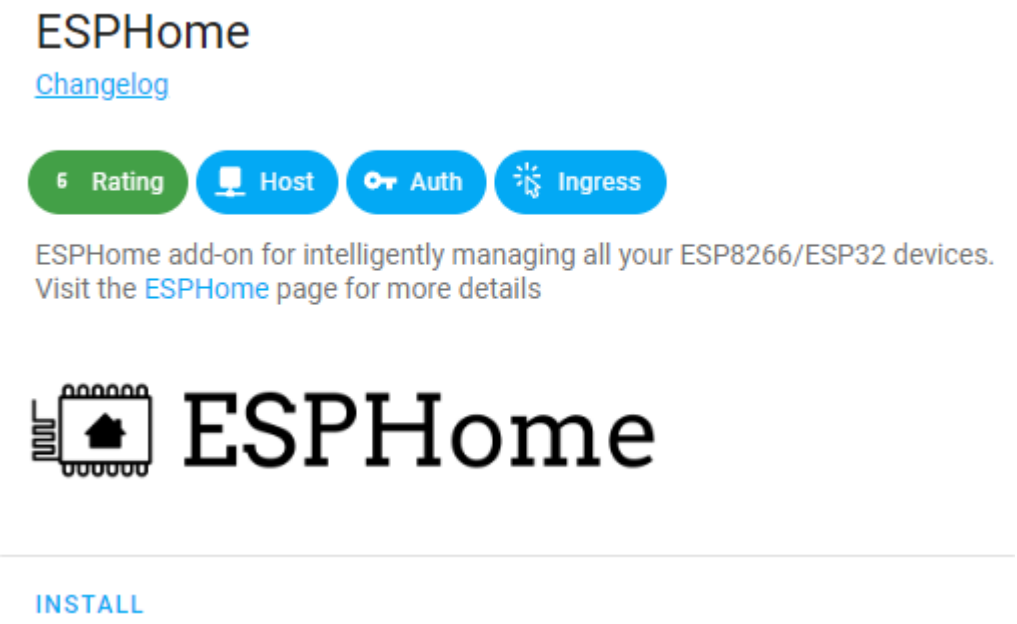
1. Configuración en ESP Home

1. Primero conecta la cámara.

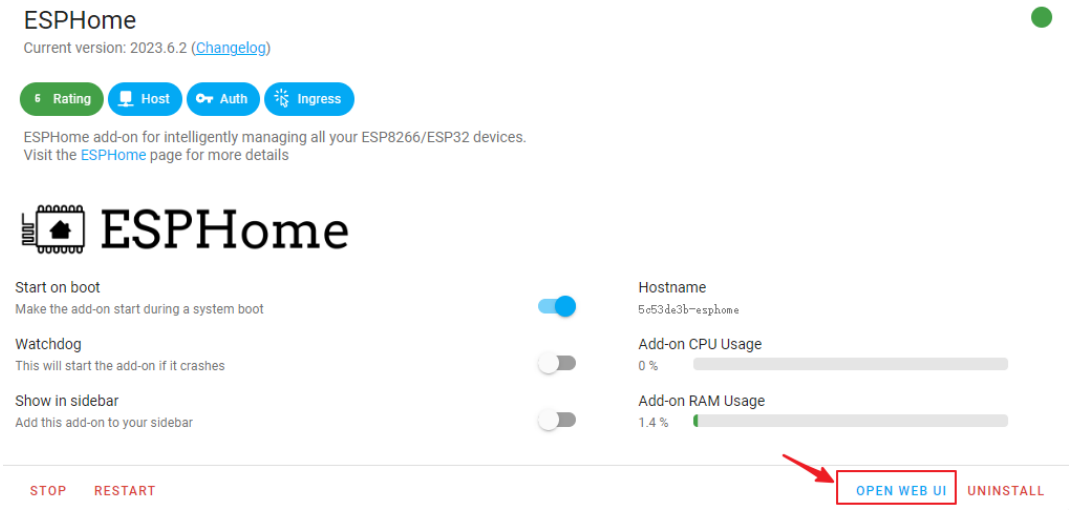
2. Conecta tu ESP32 al host donde has instalado el sistema Home Assistant (por ejemplo, si está instalado en un Raspberry Pi, conéctalo al Pi).



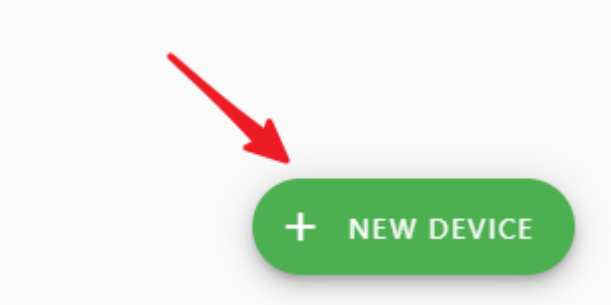
3. Instala el Addon ESPHome.



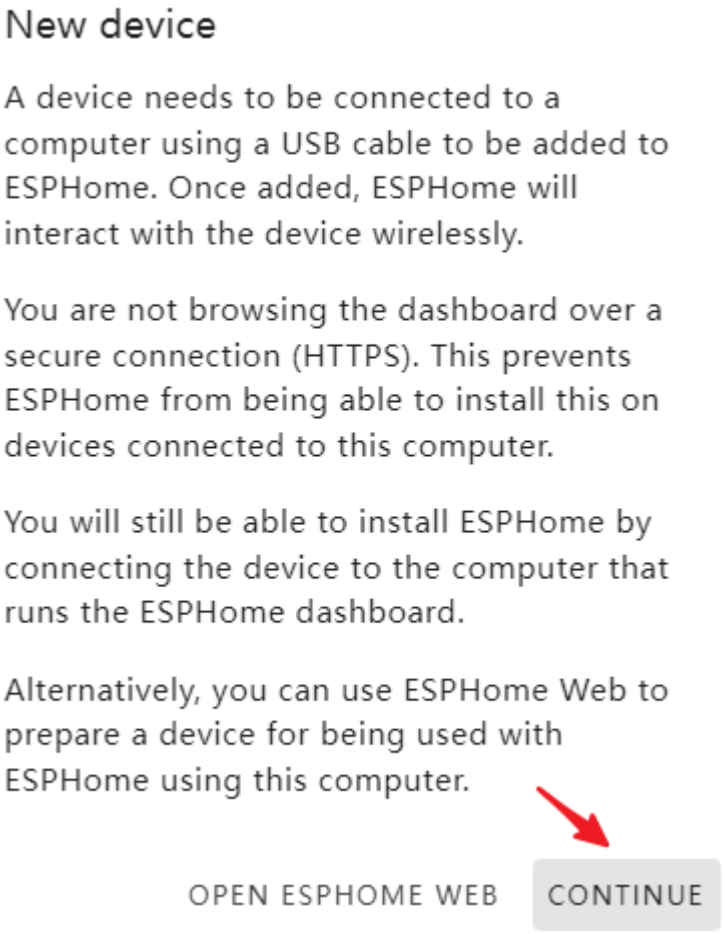
4. Haz clic en **START**, luego en **OPEN WEB UI**.



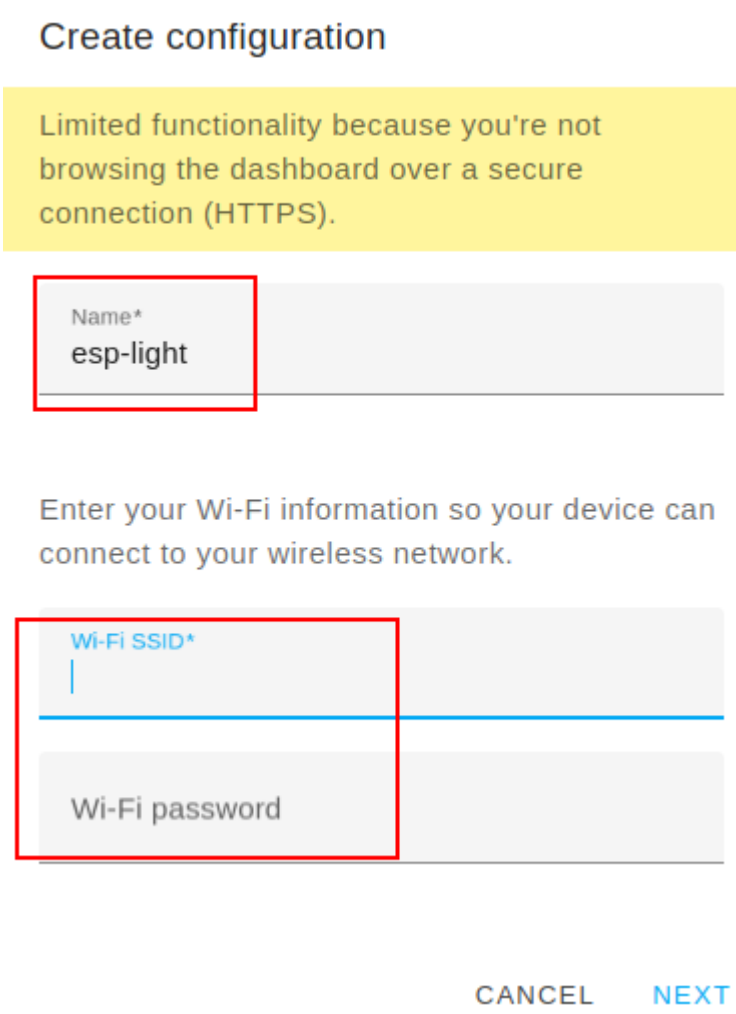
5. Agrega nuevos dispositivos.



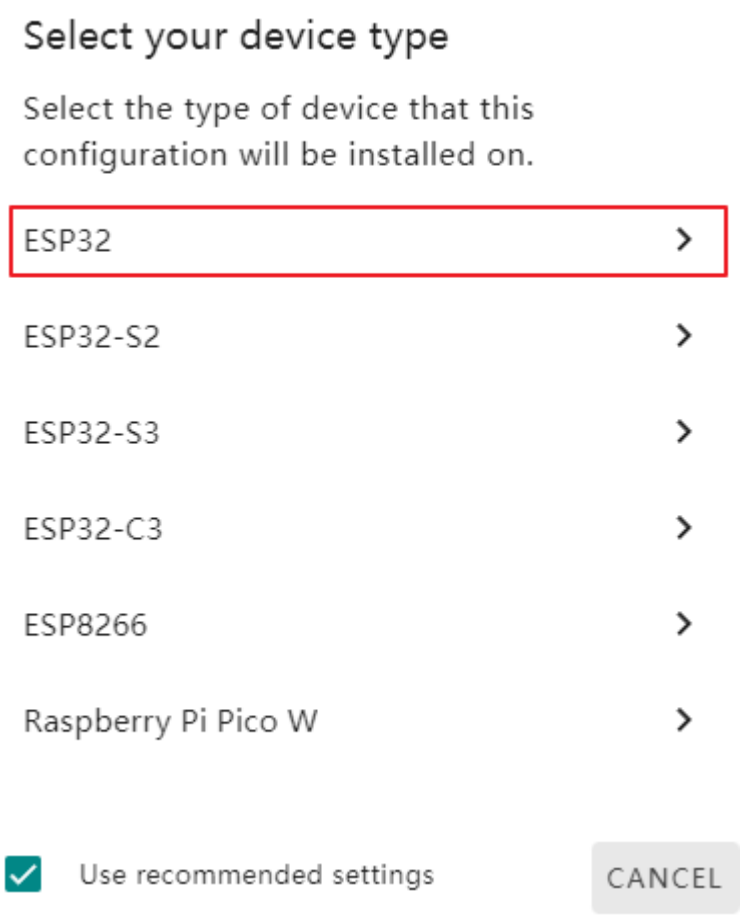
6. Podría aparecer un aviso. Haz clic en **CONTINUE**.



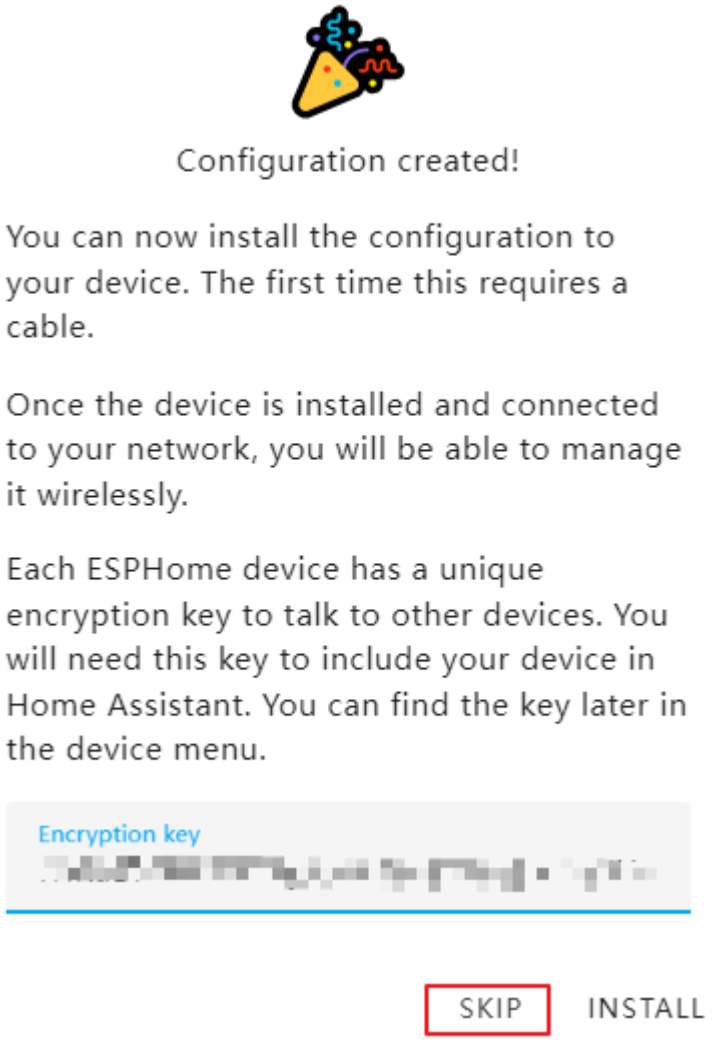
7. Crea una configuración. Aquí, puedes ingresar cualquier nombre deseado para **Name**. Para WiFi, ingresa los detalles de la red en la que tu sistema Home Assistant está presente.



8. Selecciona **ESP32** como el tipo de dispositivo.

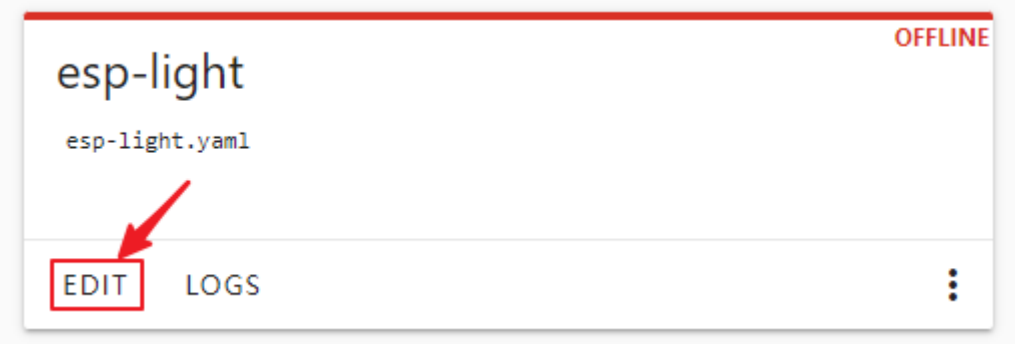


9. Cuando veas un icono de celebración con fuegos artificiales, significa que has creado exitosamente el dispositivo. Haz clic en omitir (NO hagas clic en **INSTALL**).



En este punto, solo has agregado el dispositivo en ESPHome. Para integrar el módulo ESP32 en Home Assistant, se necesitan configuraciones adicionales:

10. Haz clic en **EDIT**.



11. Después de entrar a la interfaz `.yaml`, modifica el `ssid` y `password` con los detalles de tu WiFi.

esp-light.yaml

```yaml
esphome:
  name: esp-light
  friendly_name: esp-light

esp32:
  board: esp32dev
  framework:
    type: arduino

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "[redacted]"

ota:
  password: "[redacted]"

wifi:
  ssid: "[redacted]"
  password: "[redacted]"

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Esp-Light Fallback Hotspot"
    password: "F61FWfE2yH3f"

captive_portal:
```

12. Bajo la sección `captive_portal`, pega el siguiente código:

```
# Example configuration entry
esp32_camera:
    external_clock:
        pin: GPIO0
        frequency: 20MHz
    i2c_pins:
        sda: GPIO26
        scl: GPIO27
    data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39, GPIO34,␣
↪GPIO35]
    vsync_pin: GPIO25
    href_pin: GPIO23
    pixel_clock_pin: GPIO22
    power_down_pin: GPIO32

    # Image settings
    name: My Camera
```

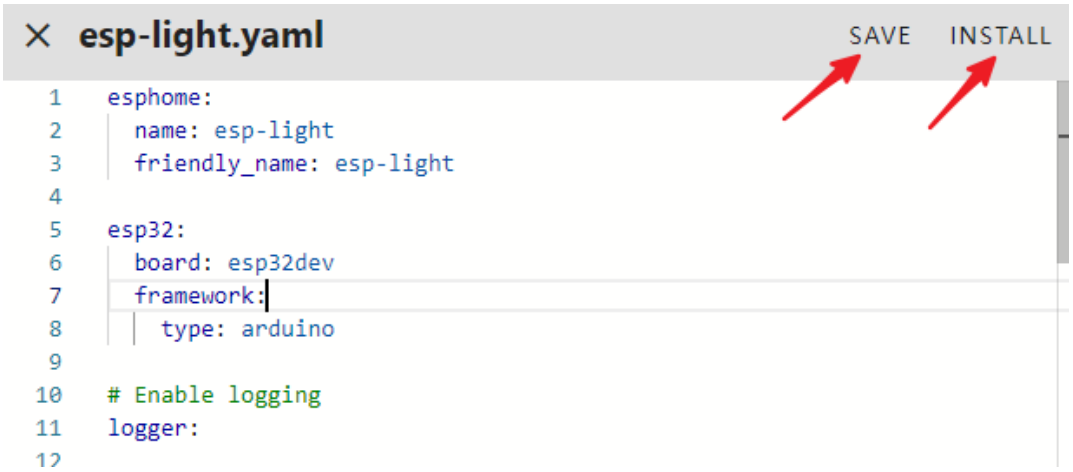
(continué en la próxima página)

(proviene de la página anterior)

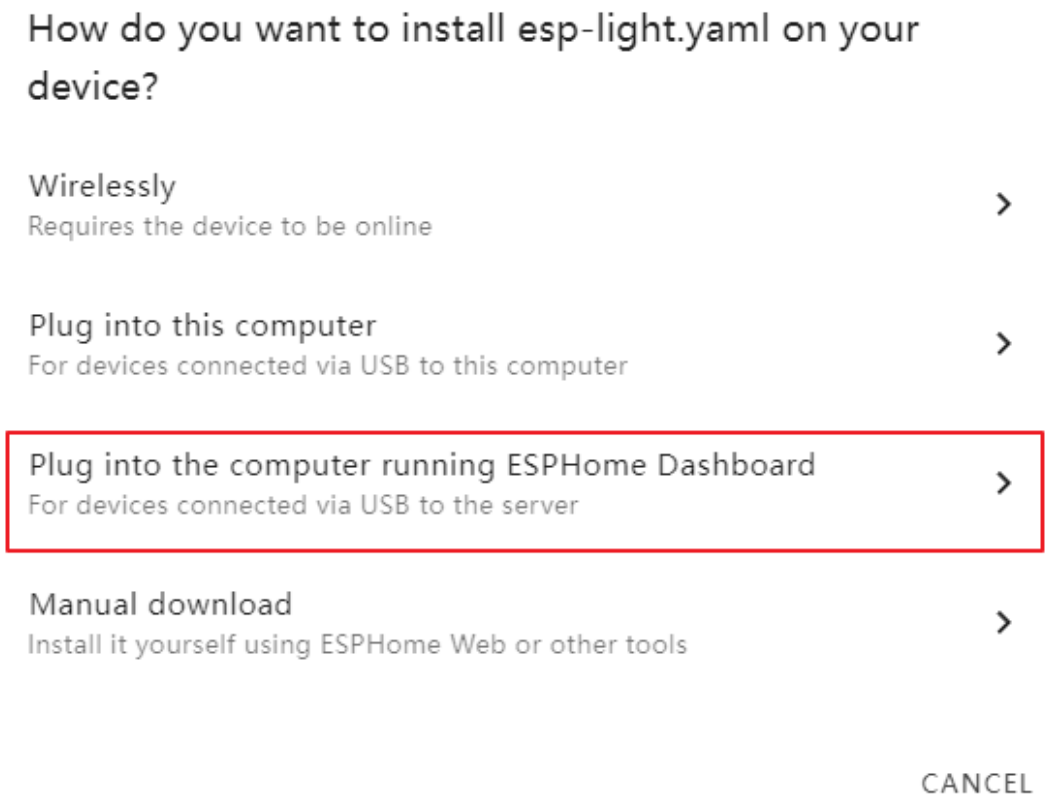
```
    # ...
```

Nota: Para más detalles sobre la configuración `.yaml` para ESP32, puedes referirte a ESP32 Camera - ESPHome.

13. **Guarda**, luego haz clic en **INSTALL**.

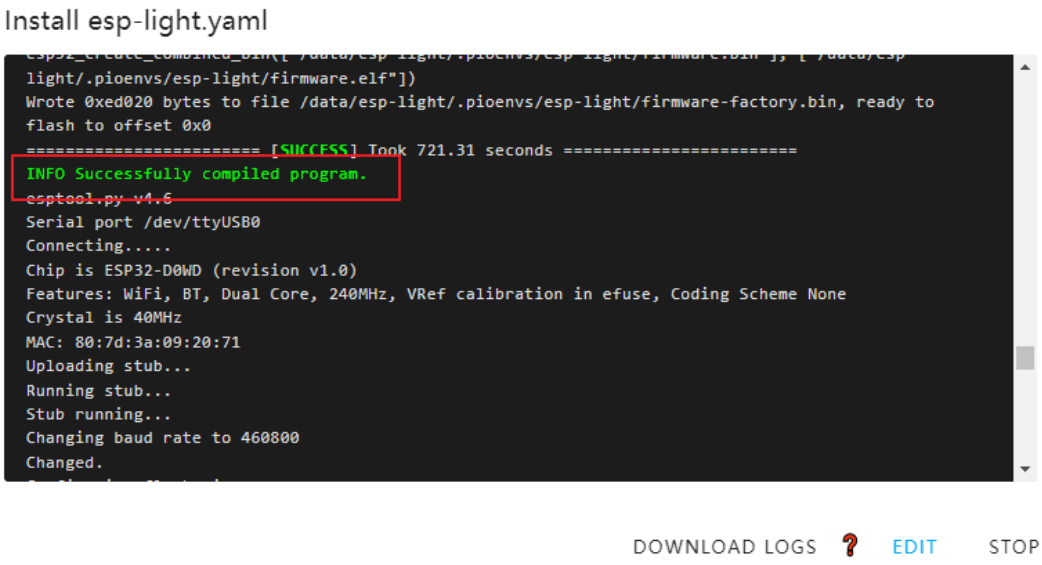


14. Elige el método de puerto USB para la instalación.

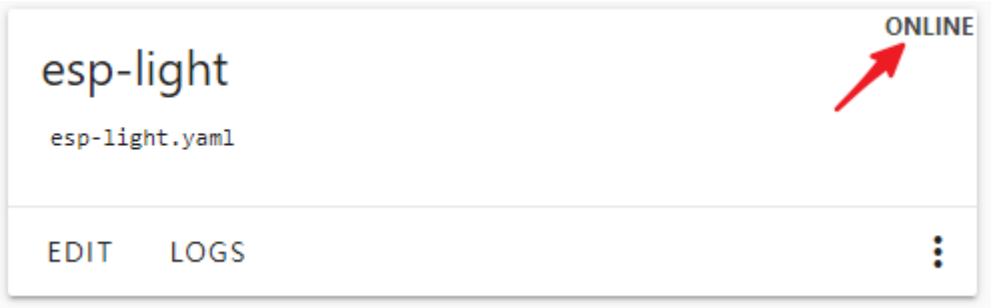


Nota: La compilación inicial descargará paquetes de dependencia, lo cual podría tomar alrededor de 10 minutos. Por favor, ten paciencia. Si el proceso se estanca por mucho tiempo, verifica si hay suficiente espacio en disco en tu sistema.

15. Espera el mensaje `INFO Successfully compiled program.`, indicando que la compilación del firmware está completa.



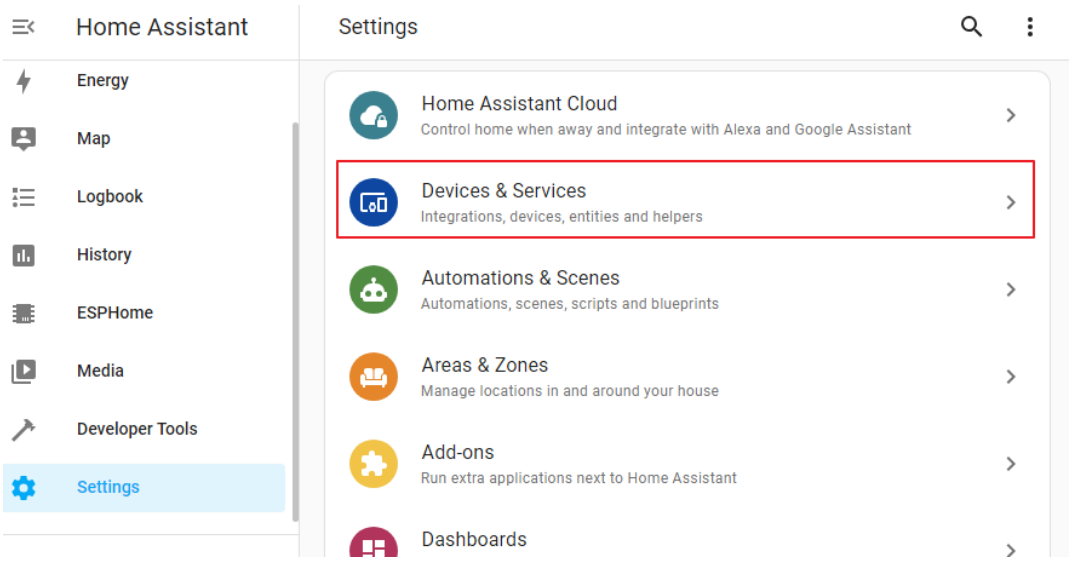
Nota: En este punto, deberías ver el nodo como **ONLINE**. Si no, asegúrate de que tu ESP32 esté en el mismo segmento de red o intenta reiniciar el dispositivo.



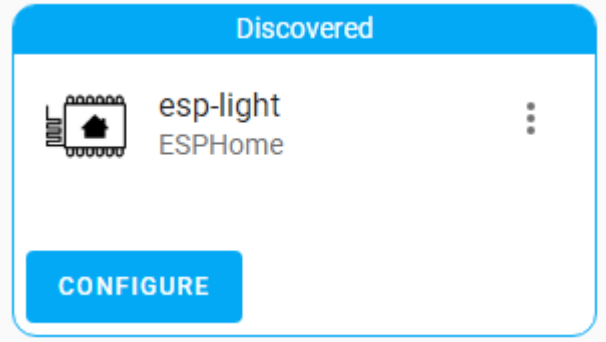
2. Configuración en Home Assistant

Después de integrarlo con Esphome, aún necesitas configurar la cámara en homeassistant.

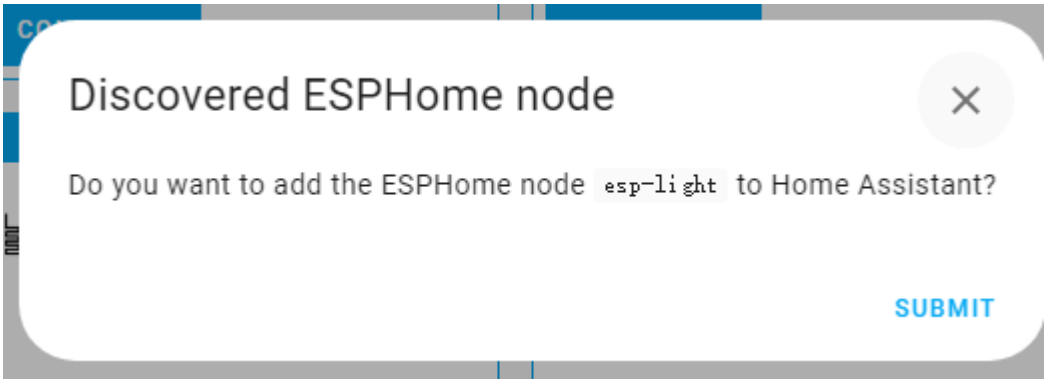
1. Ve a **Settings** > **Devices & Services**.



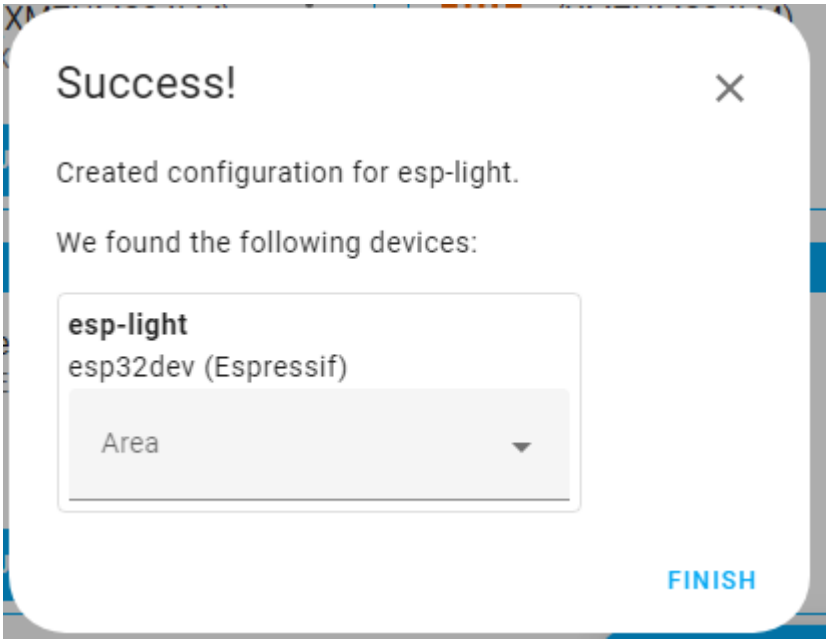
2. Ahora deberías ver la pestaña de esphome. Haz clic en **CONFIGURE**.



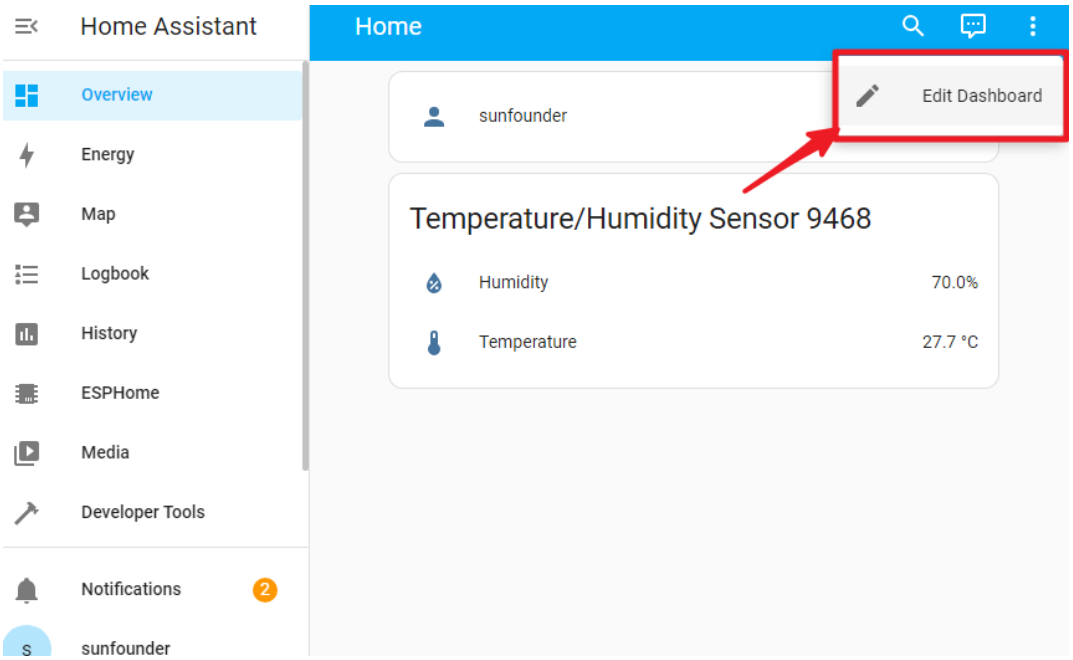
3. Haz clic en **SUBMIT**.



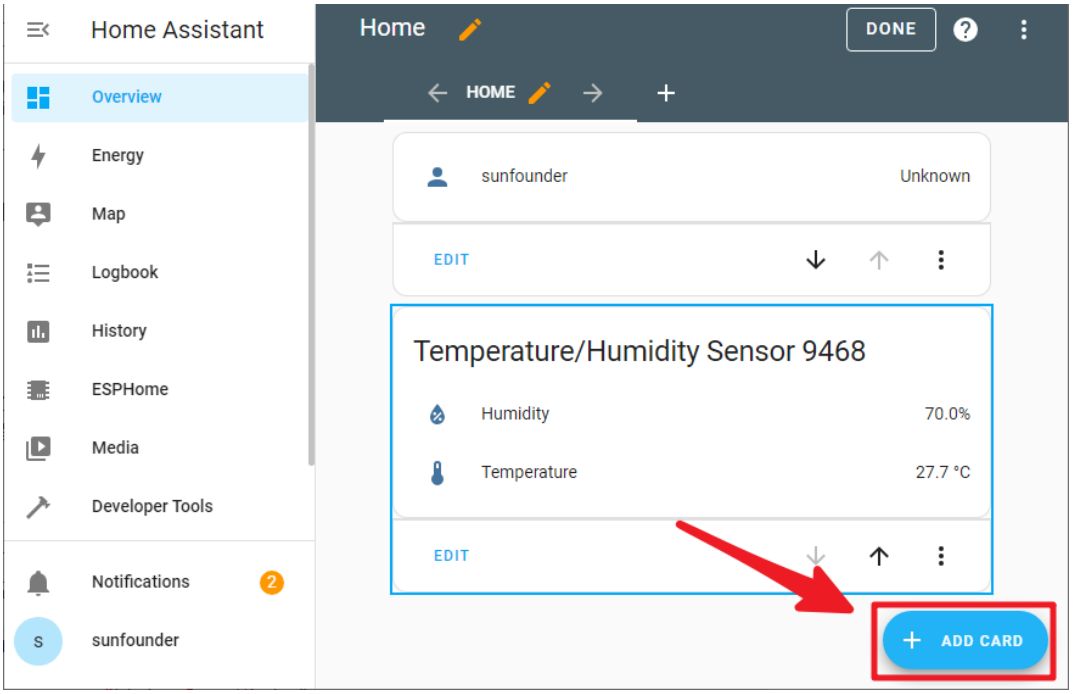
4. Espera el mensaje de **Success**.



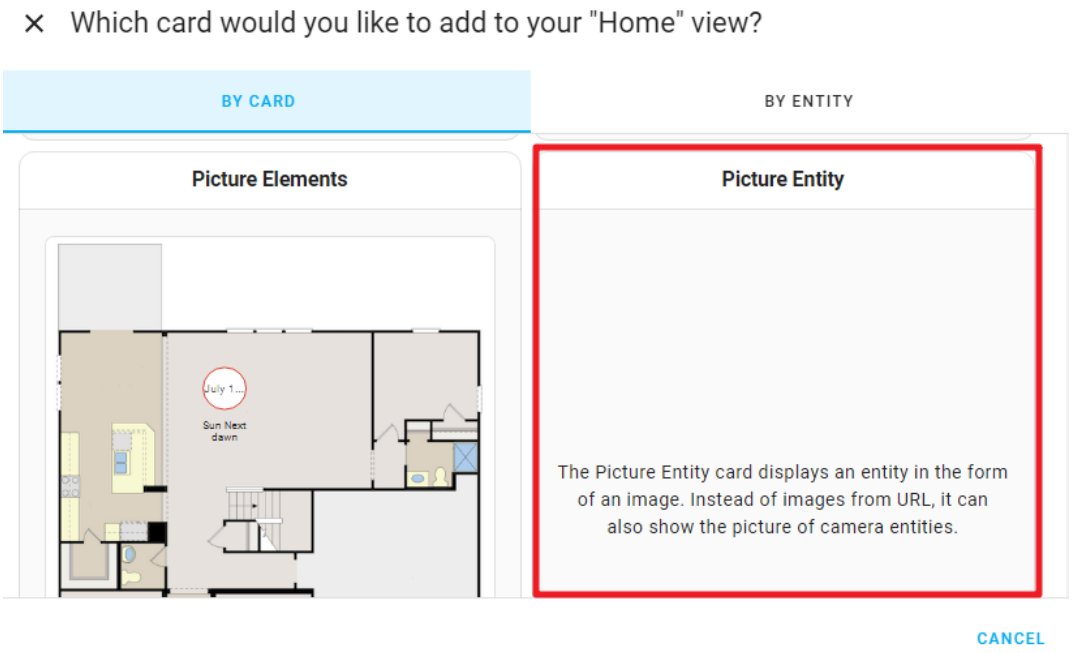
5. En **Overview**, haz clic en el menú superior derecho y selecciona **Edit Dashboard**.



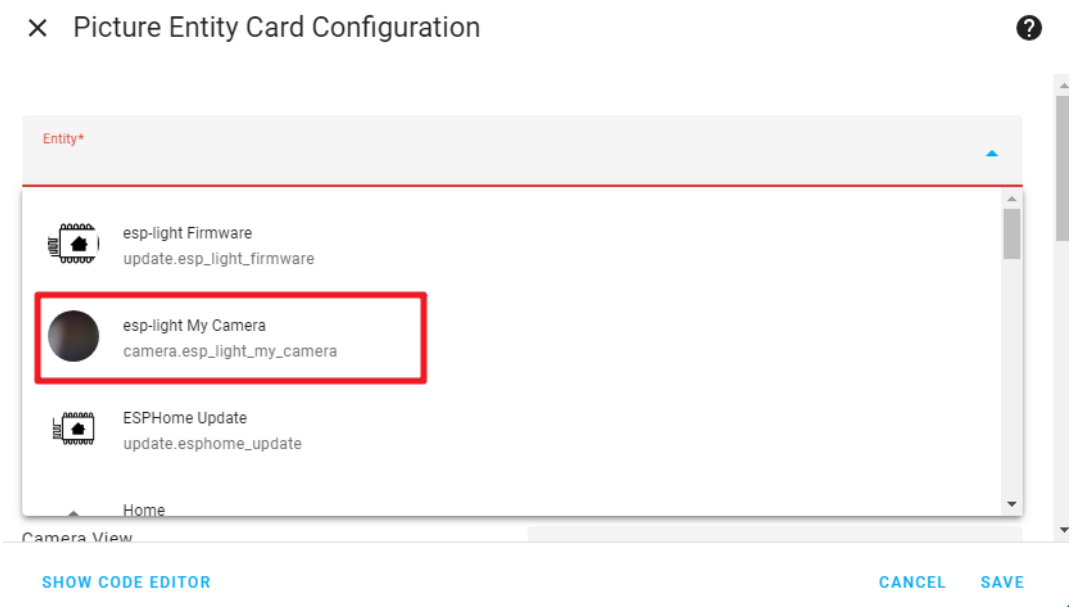
6. Haz clic en **ADD CARD**.



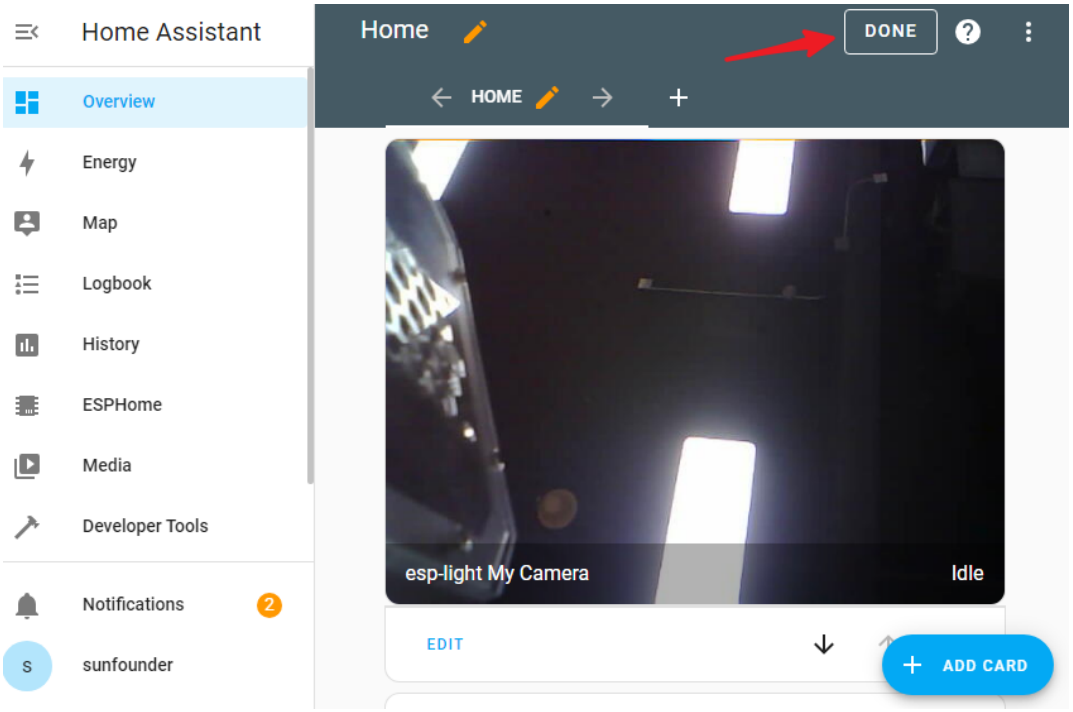
7. Elige **Picture entity**.



8. En **Entity**, selecciona la cámara que acabas de agregar. Luego haz clic en **SAVE**.



9. Ahora deberías ser capaz de ver la transmisión en vivo de tu cámara en Home Assistant.



Ahora, puedes ver el contenido de tu cámara en el Asistente en Casa.

2.52 8.9 Sistema de Notificación de Intrusión Basado en Blynk

Este proyecto demuestra un sistema simple de detección de intrusiones en el hogar usando un sensor de movimiento PIR (HC-SR501). Cuando el sistema está configurado en modo «Ausente» a través de la aplicación Blynk, el sensor PIR monitorea el movimiento. Cualquier movimiento detectado activa una notificación en la aplicación Blynk, alertando al usuario de una posible intrusión.

Componentes Necesarios

Para este proyecto, necesitamos los siguientes componentes.

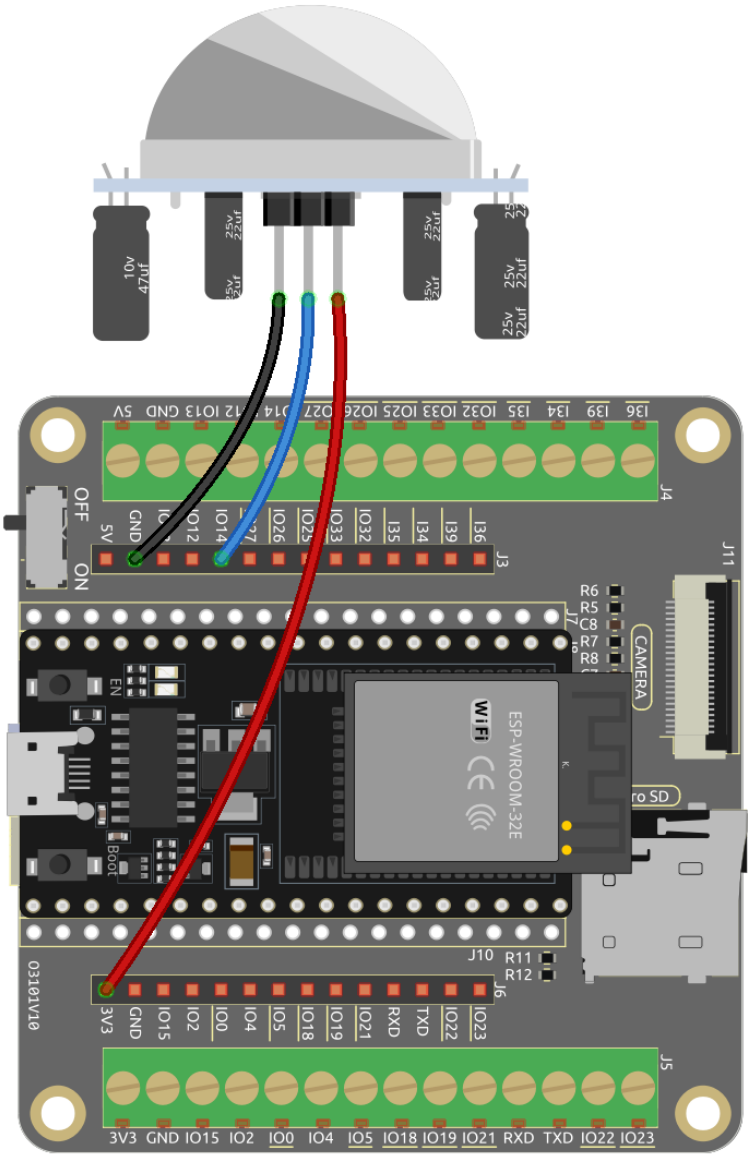
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo Sensor de Movimiento PIR	

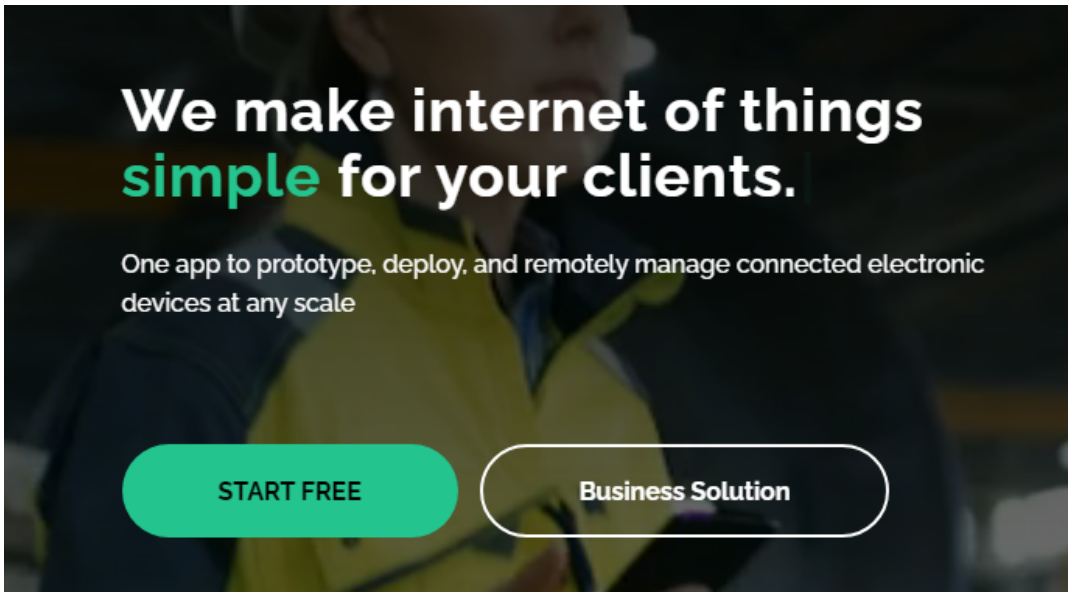
2.52.1 1. Montaje del Circuito



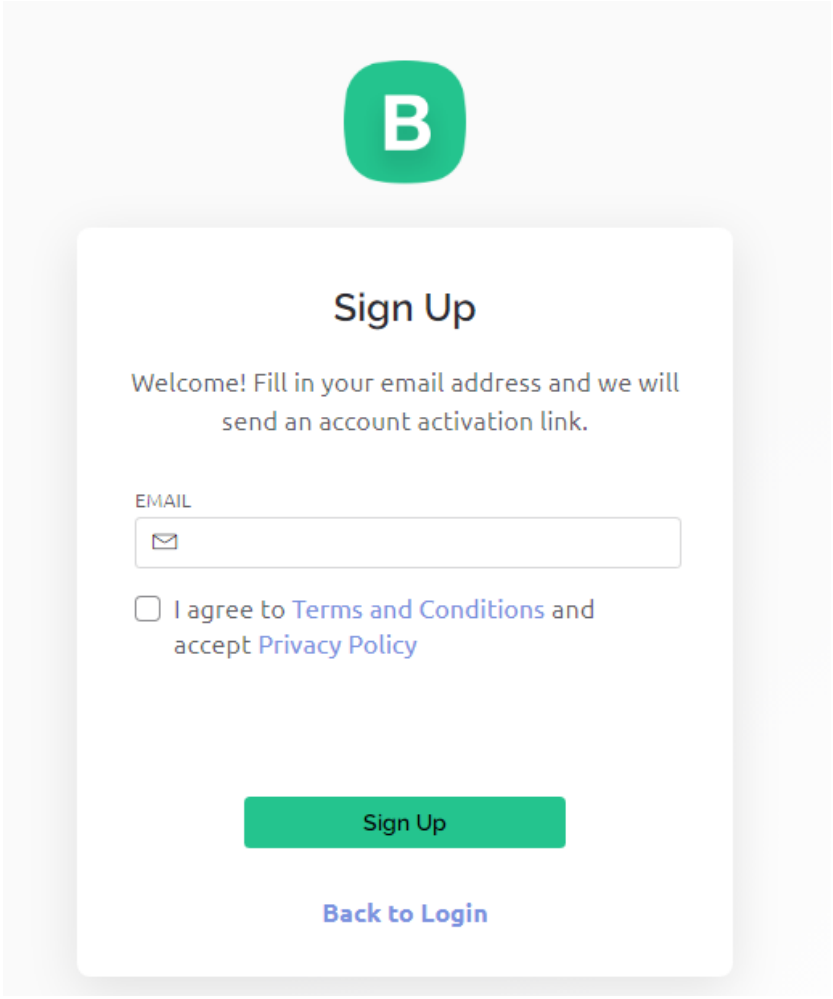
2.52.2 2. Configuración de Blynk

2.1 Inicialización de Blynk

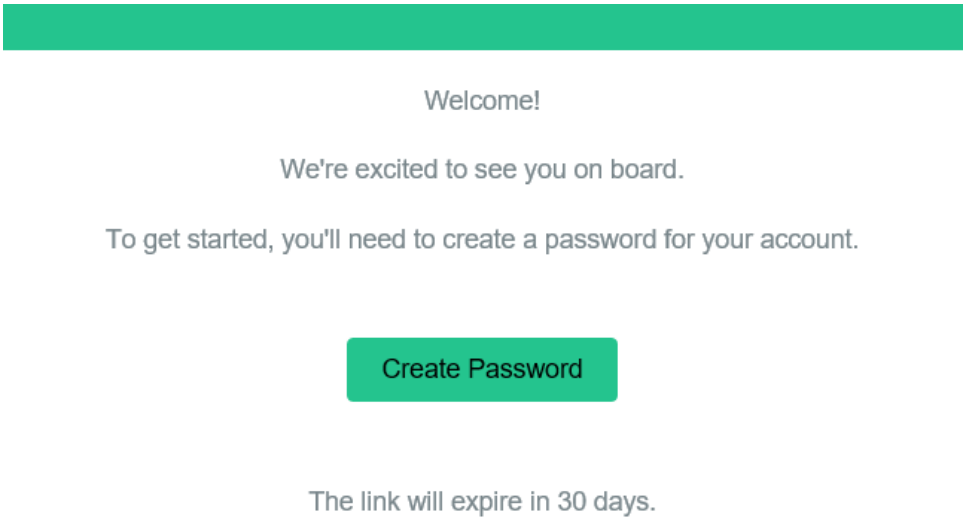
1. Navega a y selecciona **EMPEZAR GRATIS**.



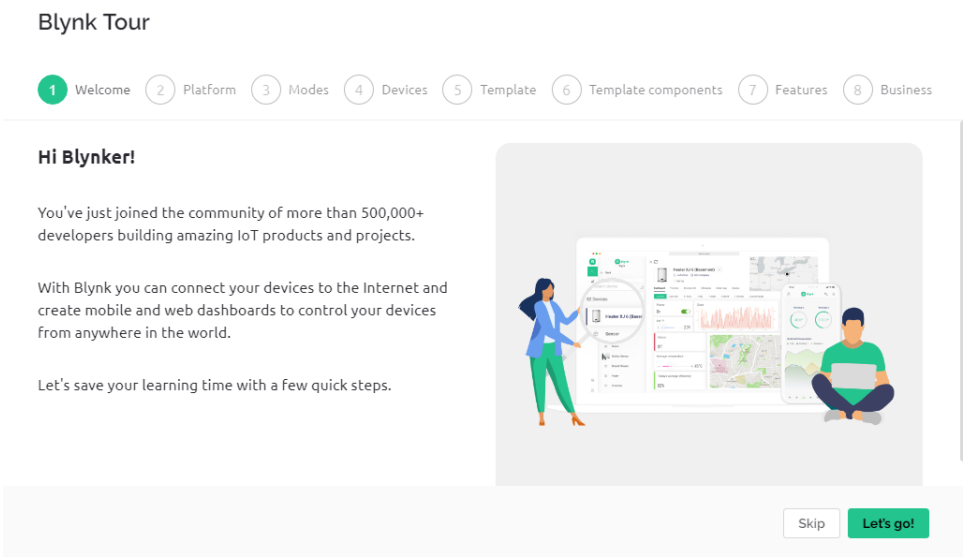
2. Ingresa tu correo electrónico para iniciar el proceso de registro.



3. Confirma tu registro a través de tu correo electrónico.

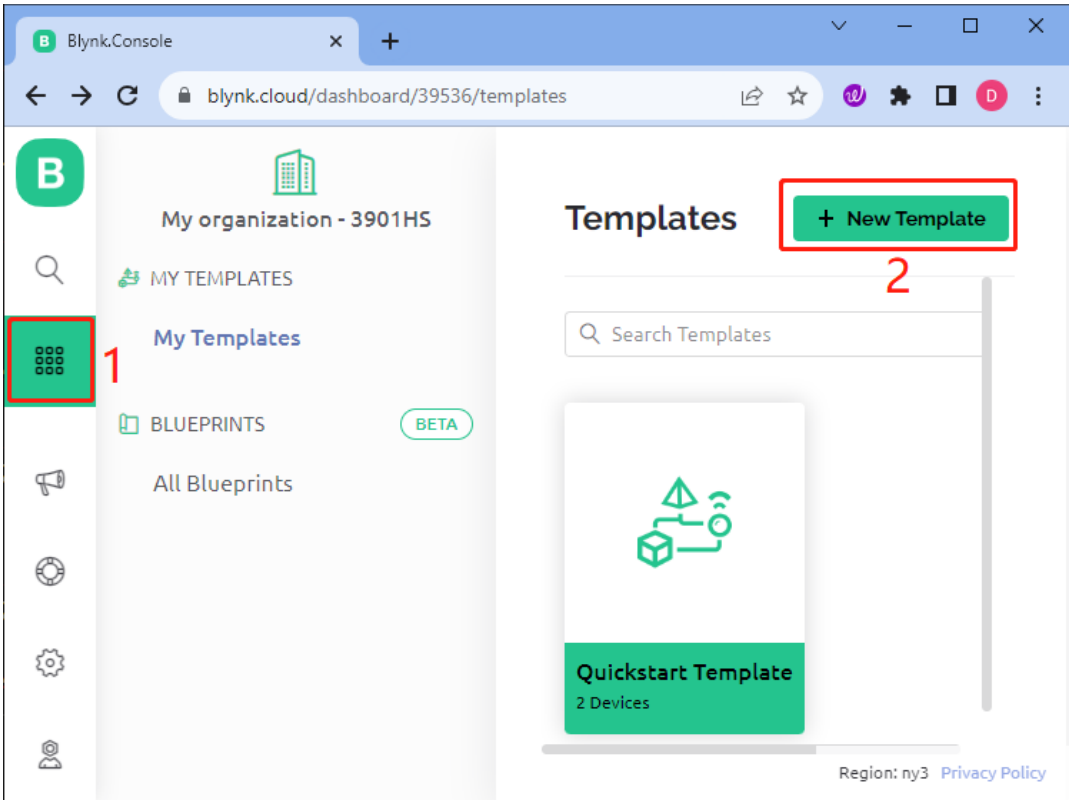


4. Después de la confirmación, aparecerá **Tour de Blynk**. Se recomienda seleccionar «Omitir». Si también aparece **Inicio Rápido**, considera omitirlo igualmente.

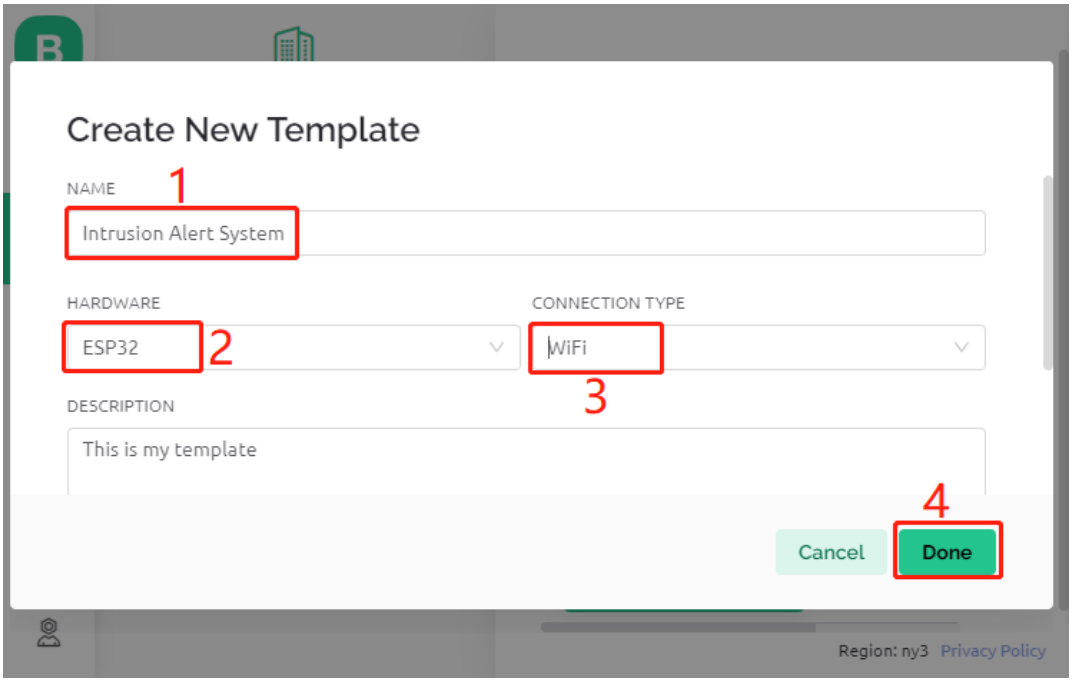


2.2 Creación de Plantilla

1. Primero, crea una plantilla en Blynk. Sigue las instrucciones subsiguientes para crear la plantilla **Sistema de Alerta de Intrusión**.



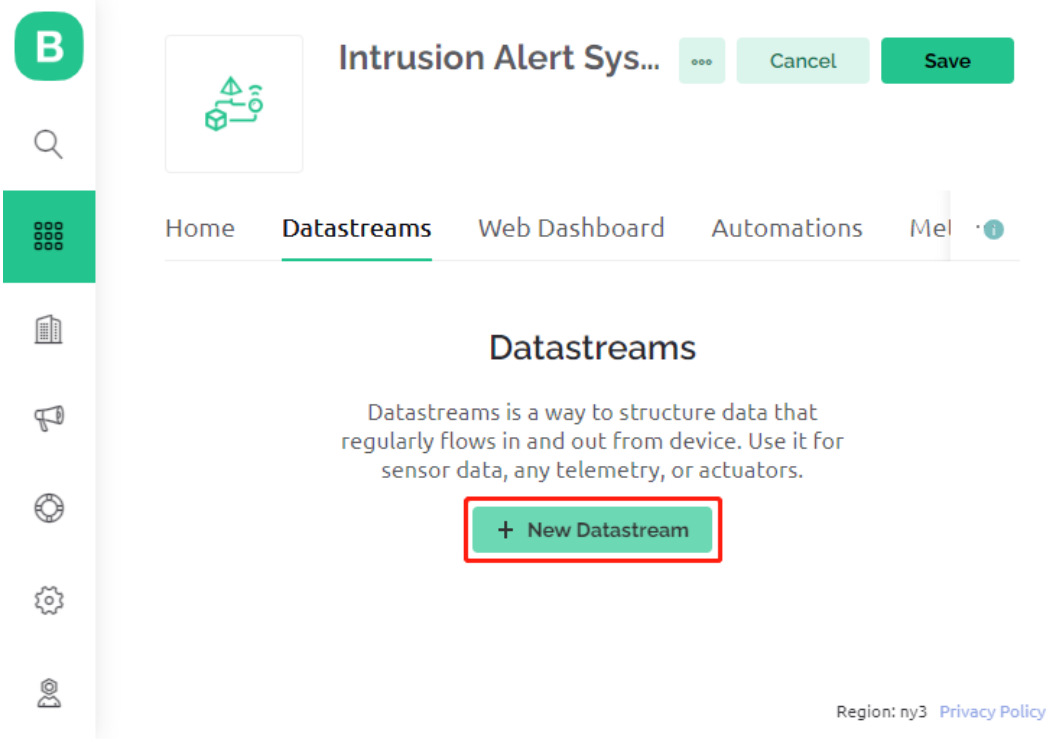
2. Asigna un nombre a la plantilla, selecciona **ESP32** como Hardware y **Tipo de Conexión** como **WiFi**, luego selecciona **Hecho**.



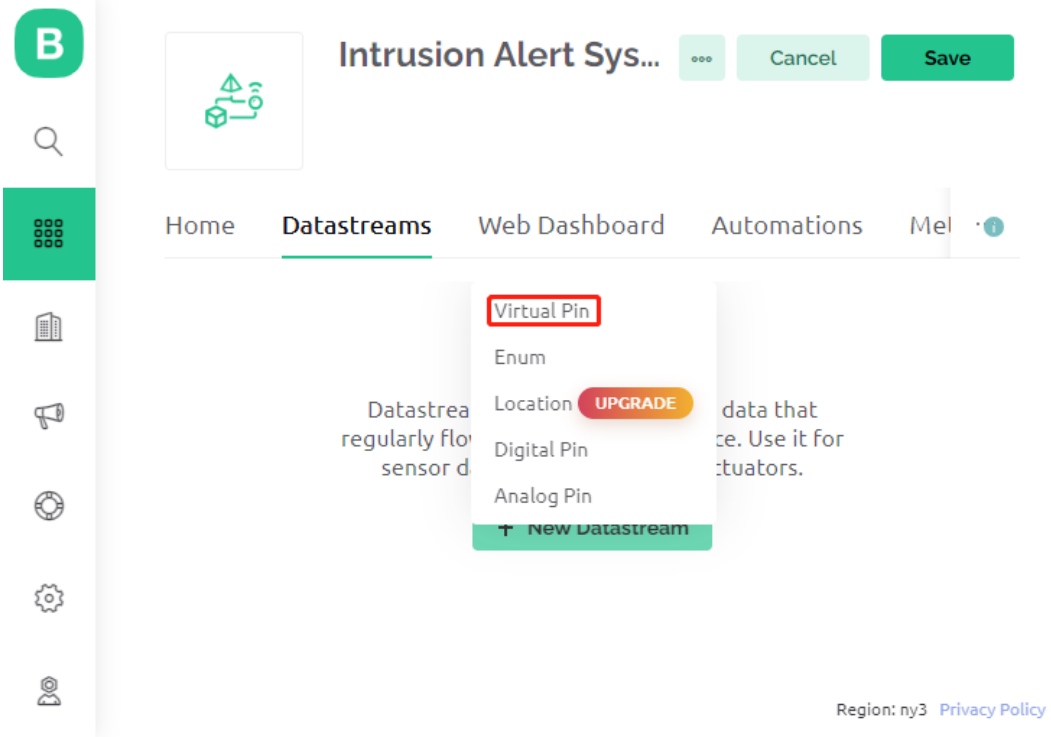
2.3 Generación de Flujo de Datos

Abre la plantilla que acabas de configurar, vamos a crear dos flujos de datos.

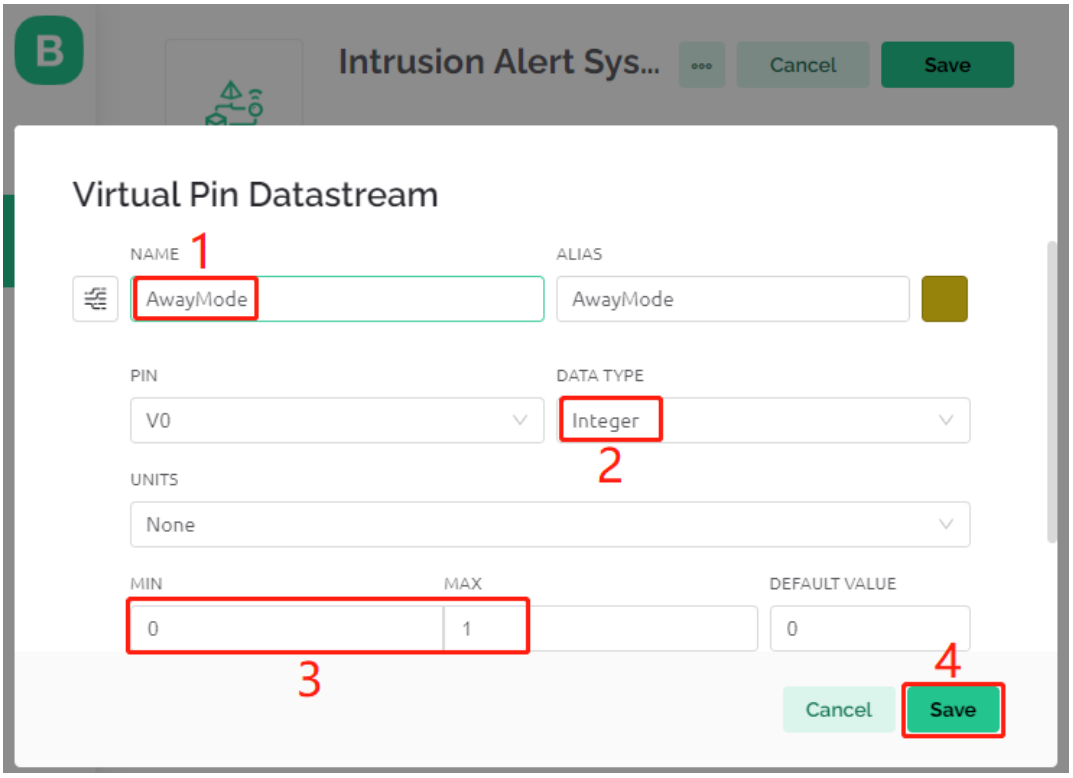
1. Haz clic en **Nuevo Flujo de Datos**.



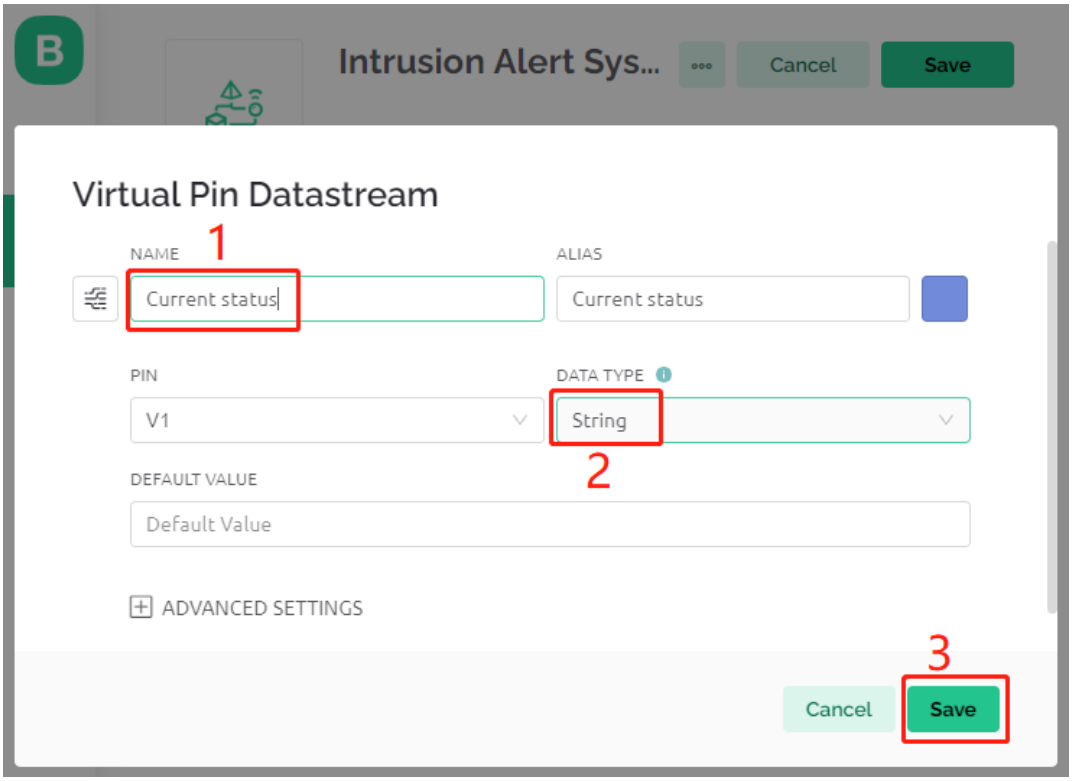
2. En el popup, elige **Pin Virtual**.



3. Nombra el **Pin Virtual V0** como **ModoAusente**. Establece el **TIPO DE DATO** como **Entero** con valores **MIN** y **MAX** como **0** y **1**.



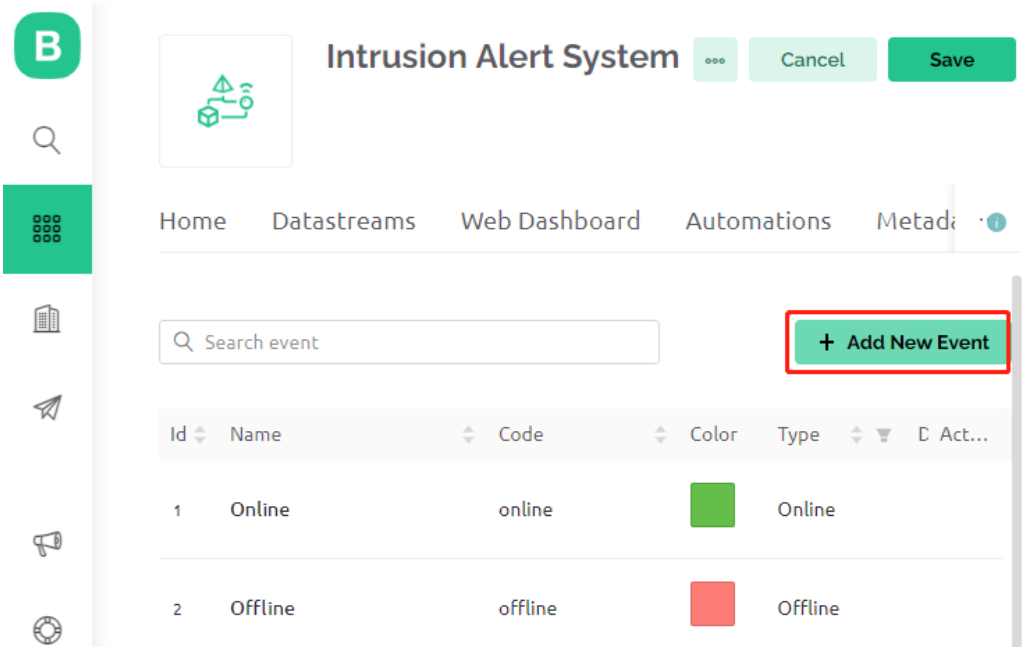
4. De manera similar, crea otro flujo de datos de **Pin Virtual**. Nómbralo **Estado Actual** y establece el **TIPO DE DATO** a **Cadena**.



2.4 Configuración de un Evento

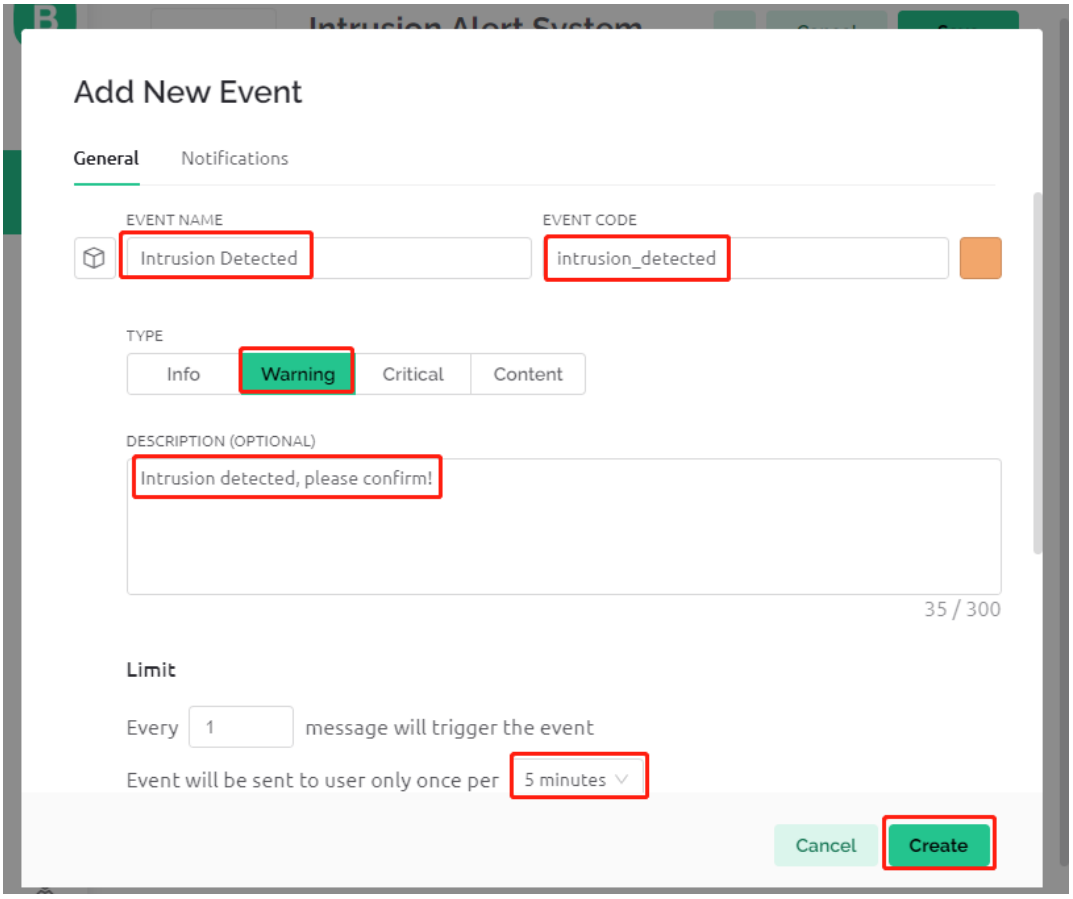
A continuación, configuraremos un evento que envía una notificación por correo electrónico si se detecta una intrusión.

1. Haz clic en **Añadir Nuevo Evento**.

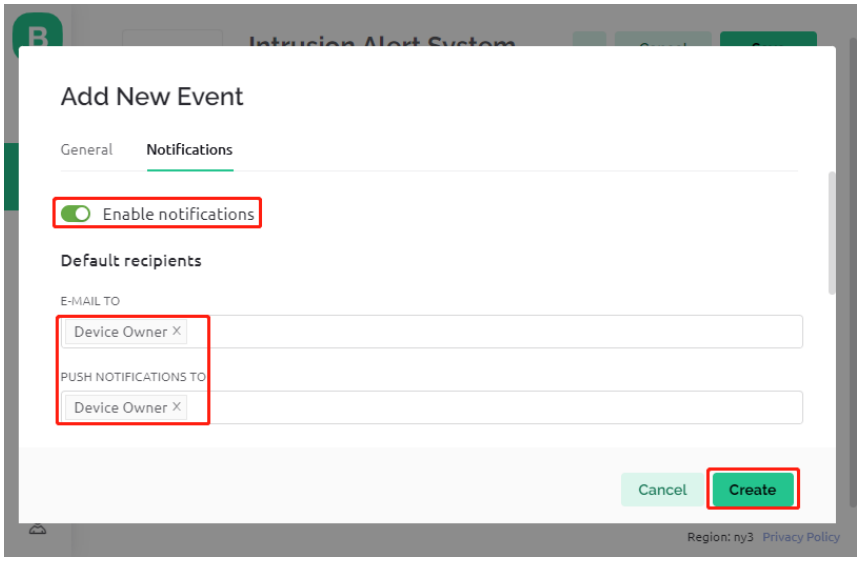


2. Define el nombre del evento y su código específico. Para **TIPO**, elige **Advertencia** y escribe una breve descripción para el correo que se enviará cuando ocurra el evento. También puedes ajustar con qué frecuencia recibes notificaciones.

Nota: Asegúrate de que el **CÓDIGO DEL EVENTO** esté establecido como `intrusion_detected`. Esto está predefinido en el código, por lo que cualquier cambio significaría que necesitas ajustar el código también.



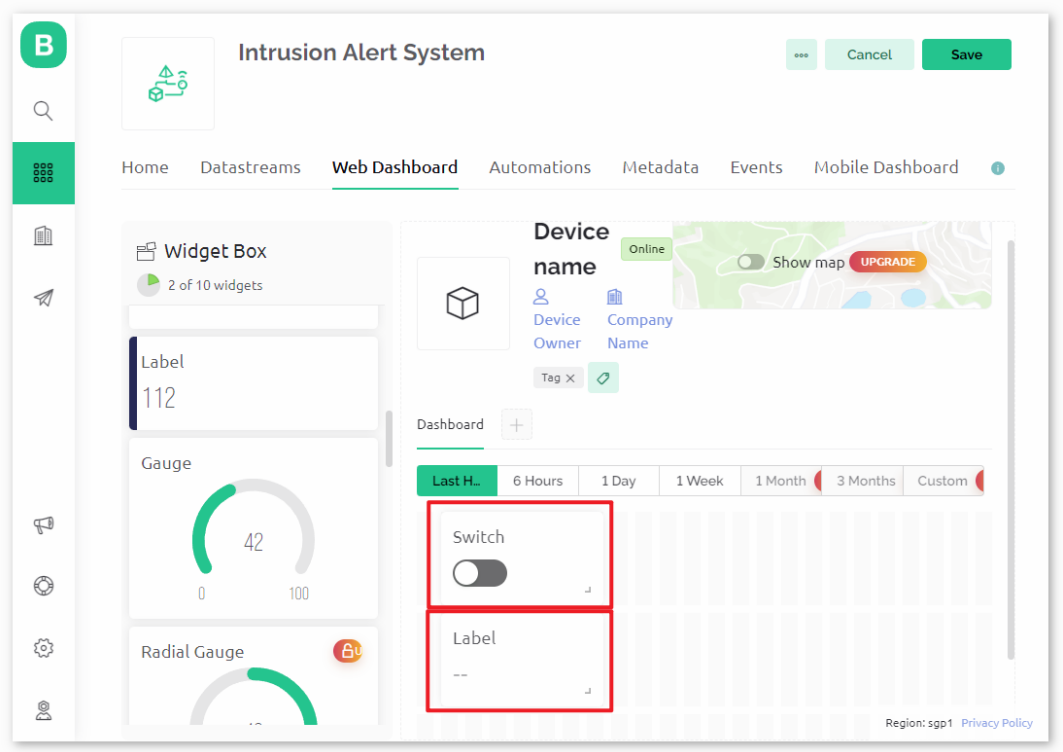
3. Ve a la sección de **Notificaciones** para activar las notificaciones y configurar los detalles del correo electrónico.



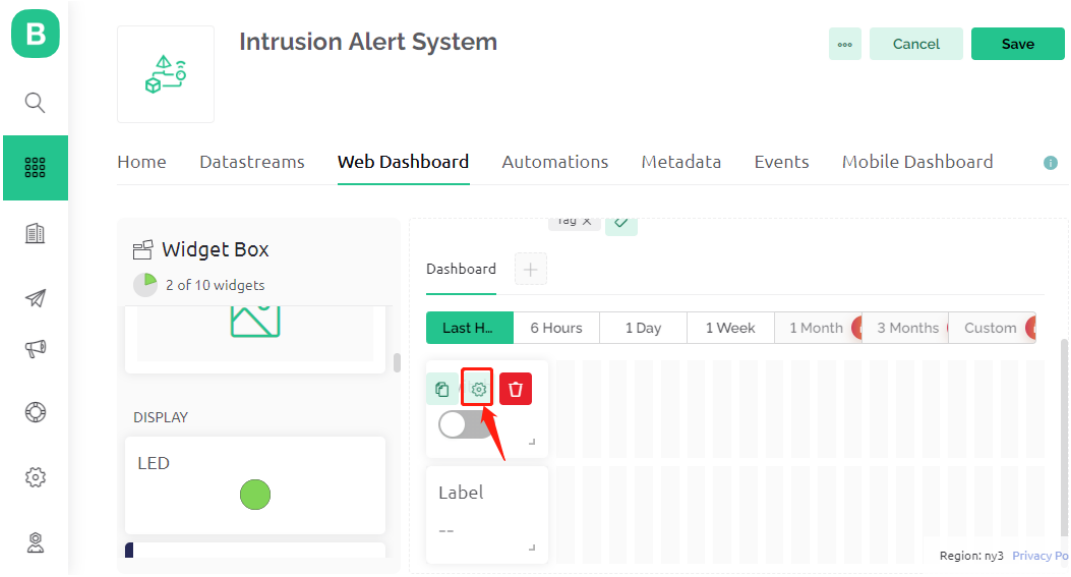
2.5 Ajuste Fino del Tablero Web

Asegurarse de que el **Tablero Web** interactúe perfectamente con el Sistema de Alerta de Intrusión es vital.

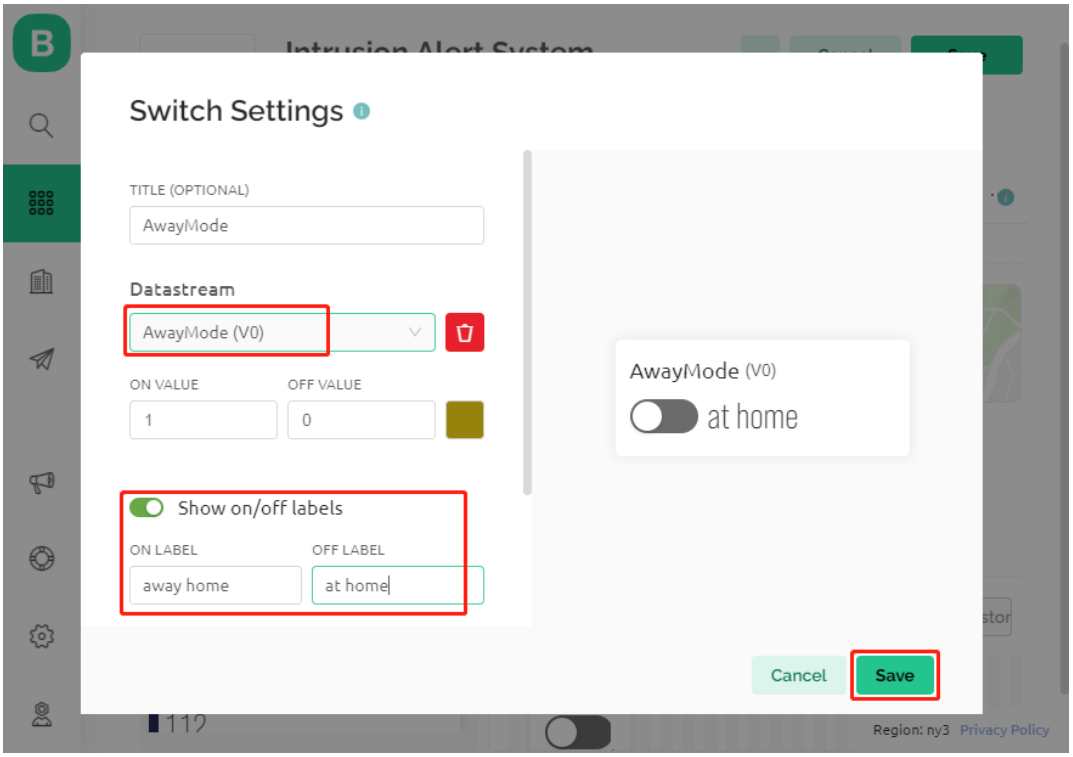
1. Simplemente arrastra y coloca tanto el **Widget de Interruptor** como el **Widget de Etiqueta** en el **Tablero Web**.



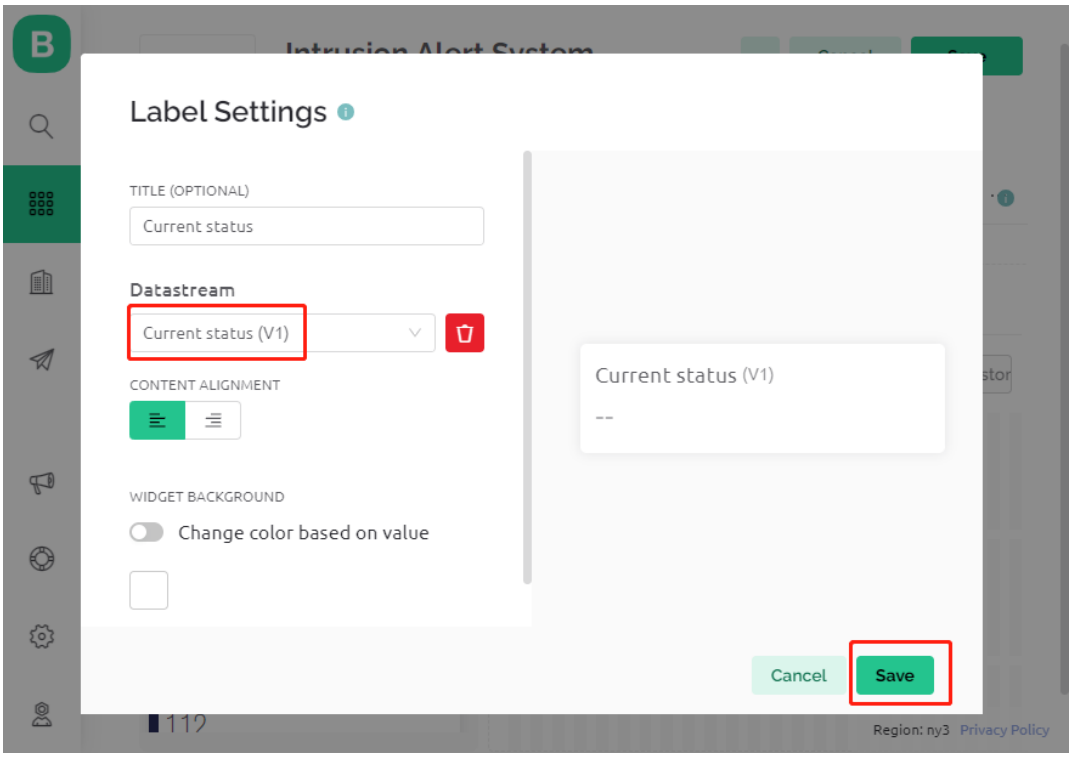
2. Cuando pases el cursor sobre un widget, aparecerán tres iconos. Usa el icono de configuración para ajustar las propiedades del widget.



3. En la configuración del **Widget de Interruptor**, selecciona **Flujo de Datos** como **ModoAusente(V0)**. Establece **EtiquetaON** y **EtiquetaOFF** para mostrar **«ausente»** y **«casa»**, respectivamente.

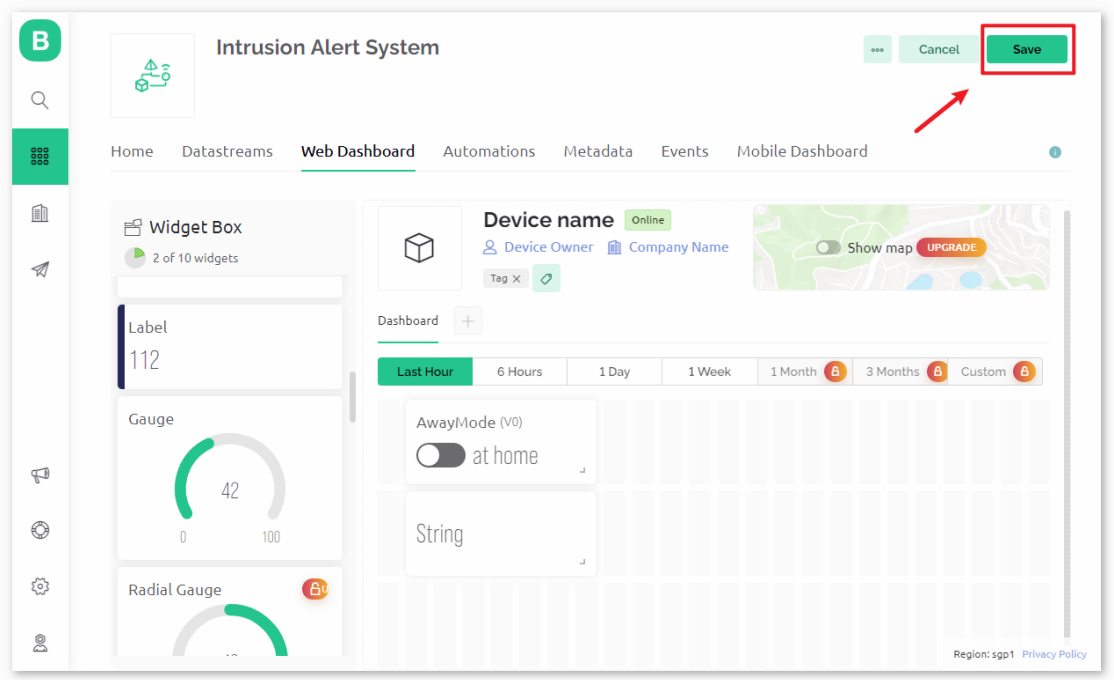


4. En la configuración del **Widget de Etiqueta**, selecciona **Flujo de Datos** como **Estado Actual(V1)**.



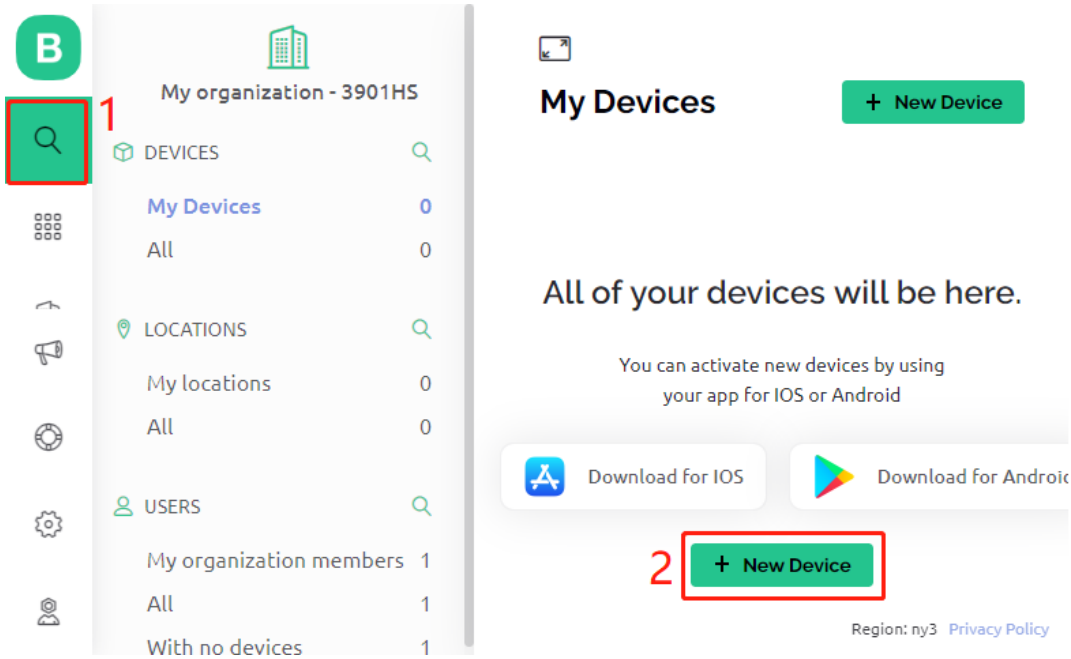
2.6 Guardando la Plantilla

Por último, no olvides guardar tu plantilla.

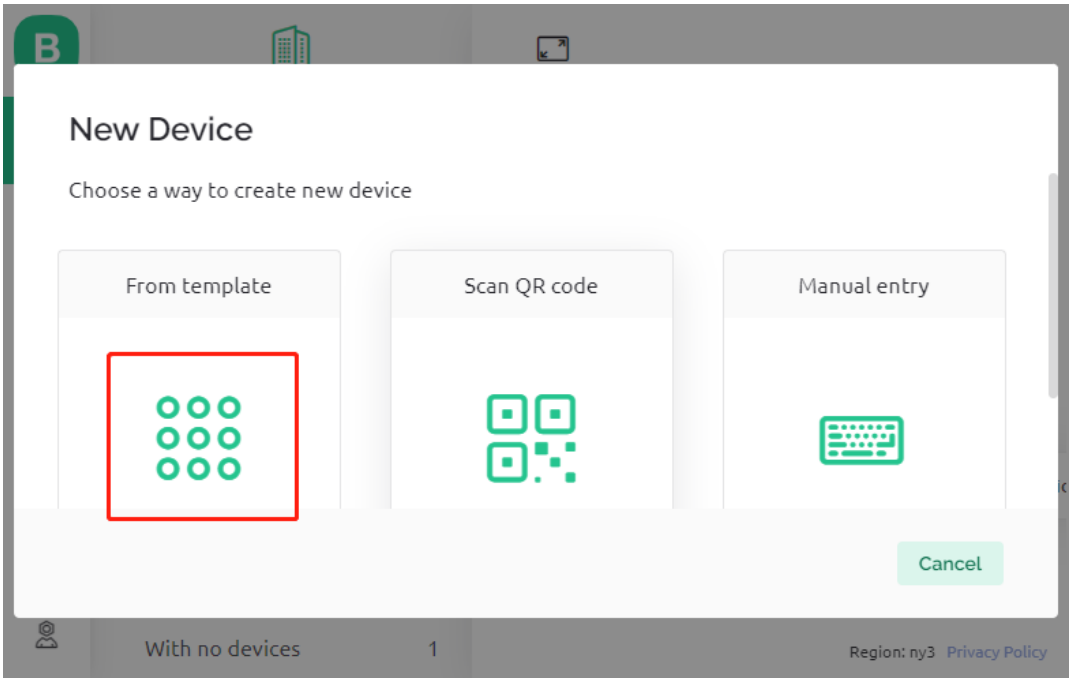


2.7 Creando un Dispositivo

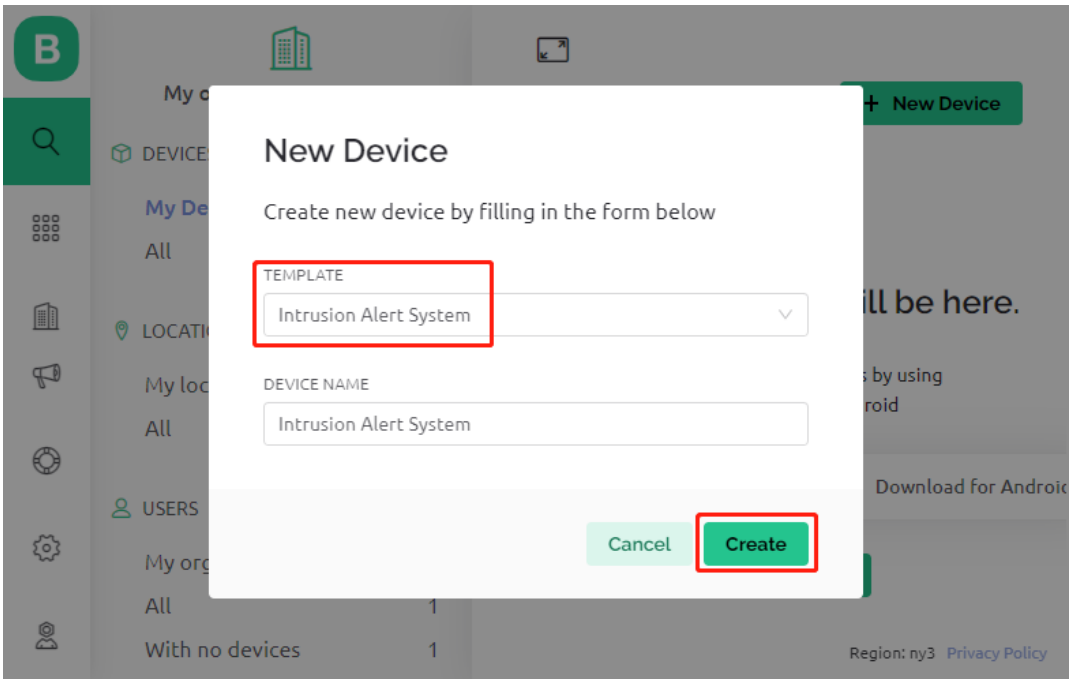
1. Es hora de crear un nuevo dispositivo.



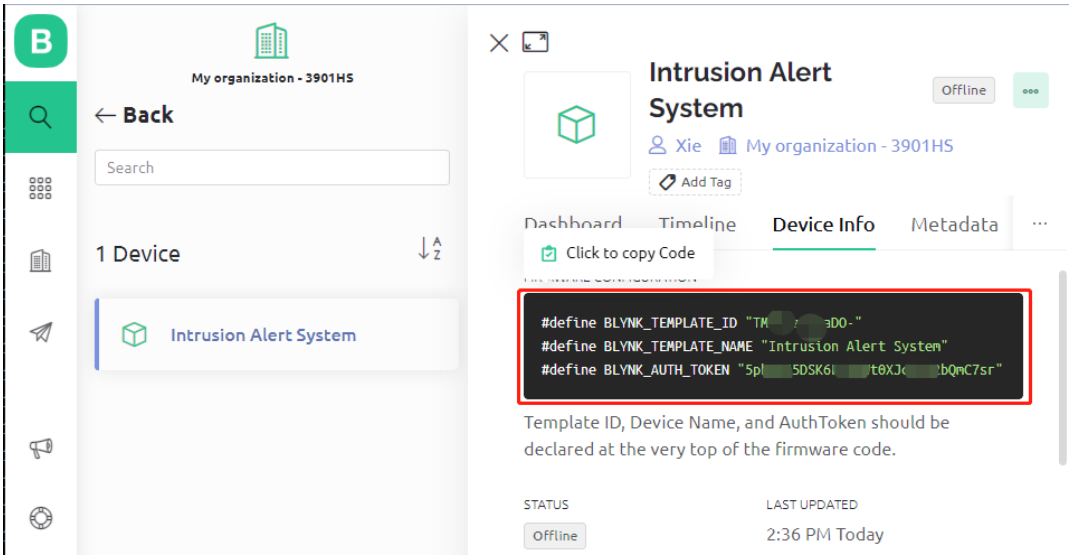
2. Haz clic en **Desde plantilla** para comenzar con una nueva configuración.



3. Luego, elige la plantilla **Sistema de Alerta de Intrusión** y haz clic en **Crear**.

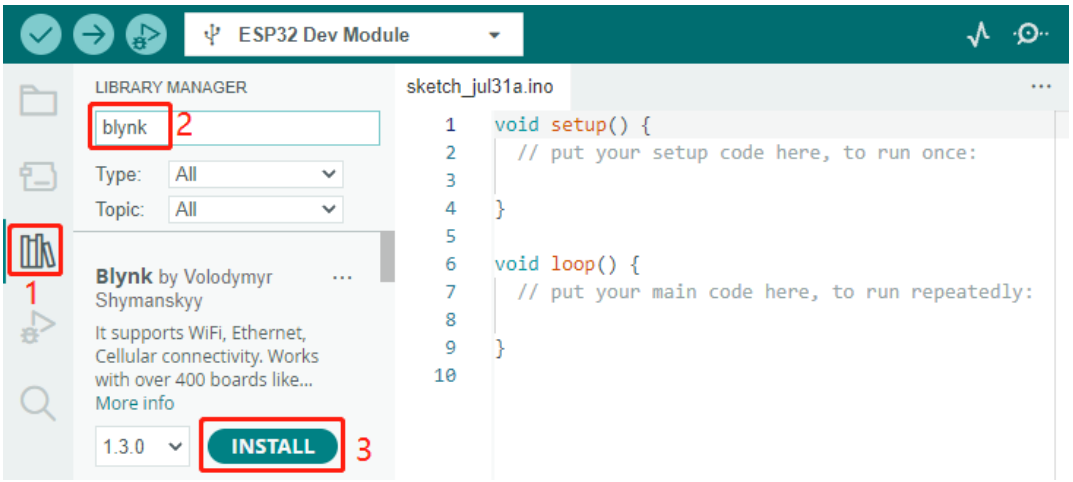


4. Aquí, verás el `ID de la Plantilla`, `Nombre del Dispositivo` y `AuthToken`. Necesitas copiar estos en tu código para que el ESP32 pueda trabajar con Blynk.



2.52.3 3. Ejecución del Código

1. Antes de ejecutar el código, asegúrate de instalar la biblioteca `Blynk` desde el **Administrador de Bibliotecas** en el Arduino IDE.



2. Abre el archivo `iot_9_intrusion_alert_system.ino`, ubicado en el directorio `esp32-starter-kit-main\c\codes\iot_9_intrusion_alert_system`. También puedes copiar su contenido en el Arduino IDE.

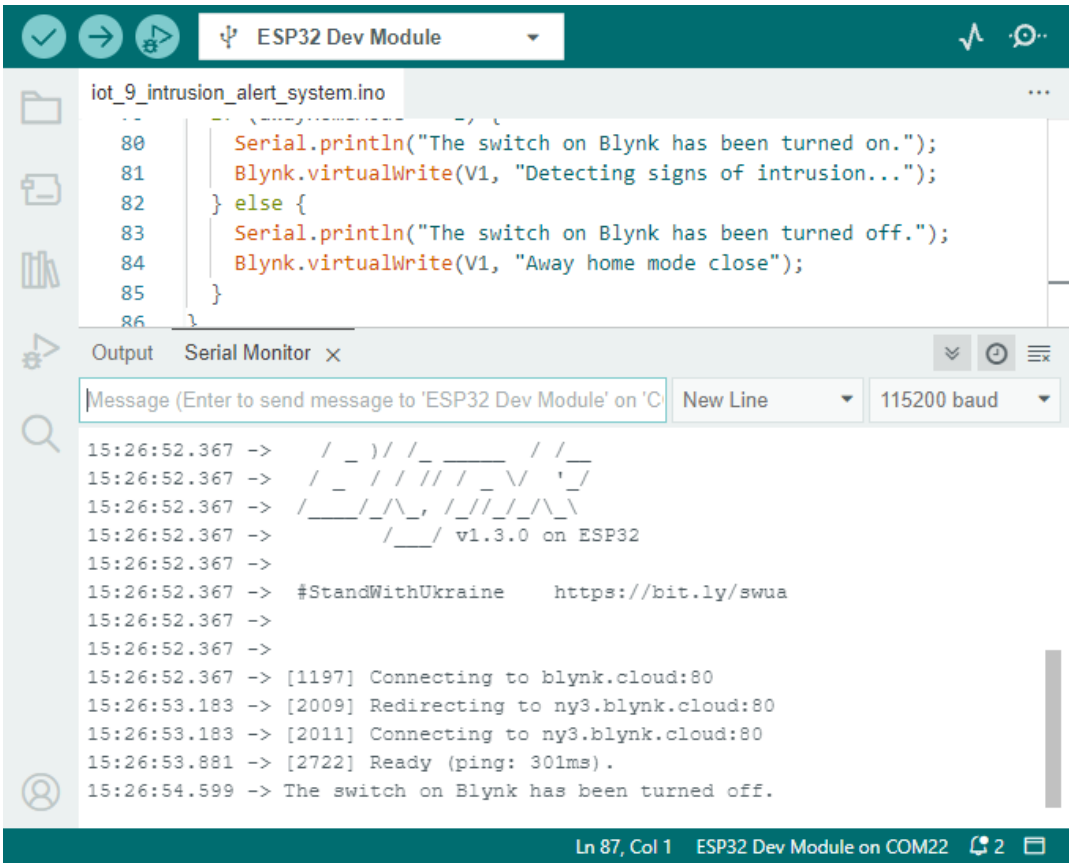
3. Sustituye los marcadores de posición de `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` y `BLYNK_AUTH_TOKEN` por tus propios IDs únicos.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxx"
```

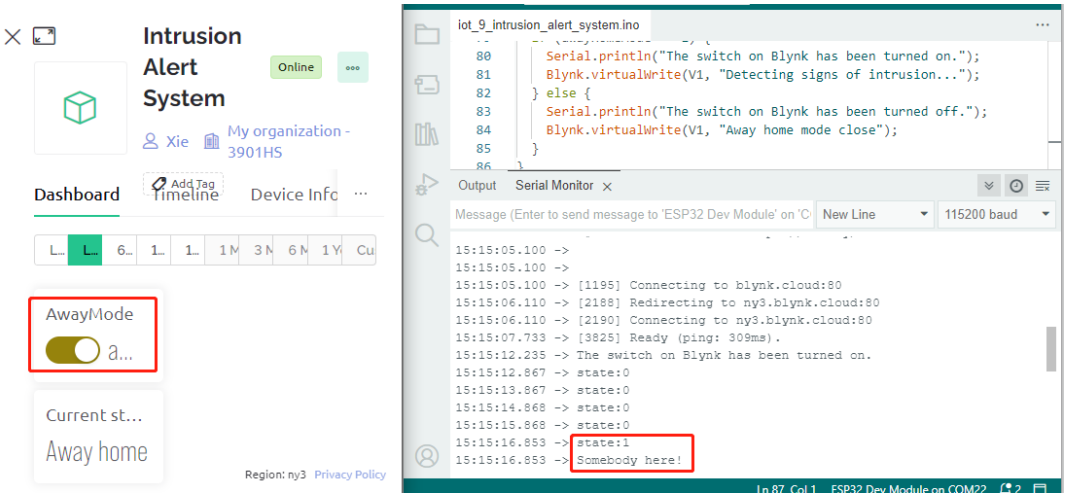
4. También necesitas ingresar el `ssid` y `password` de tu red WiFi.

```
char ssid[] = "your_ssid";
char pass[] = "your_password";
```

5. Elige la placa correcta (**ESP32 Dev Module**) y puerto, luego haz clic en el botón **Subir**.
6. Abre el Monitor Serial (configura la tasa de baudios a 115200) y espera un mensaje de conexión exitosa.



7. Tras una conexión exitosa, activar el interruptor en Blynk iniciará la vigilancia del módulo PIR. Cuando se detecte movimiento (estado de 1), dirá, «¡Alguien aquí!» y enviará una alerta a tu correo electrónico.



2.52.4 4. Explicación del Código

1. **Configuración & Bibliotecas**

   Aquí, configuras las constantes y credenciales de Blynk. También incluyes las bibliotecas necesarias para el ESP32 y Blynk.

   ```
   /* Comment this out to disable prints and save space */
   #define BLYNK_PRINT Serial

   #define BLYNK_TEMPLATE_ID "xxxxxxxxxx"
   #define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
   #define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxxxxxxxxxxxxxxx"

   #include <WiFi.h>
   #include <WiFiClient.h>
   #include <BlynkSimpleEsp32.h>
   ```

2. **Configuración WiFi**

   Ingresa tus credenciales WiFi.

   ```
   char ssid[] = "your_ssid";
   char pass[] = "your_password";
   ```

3. **Configuración del Sensor PIR**

   Establece el pin donde está conectado el sensor PIR e inicializa las variables de estado.

   ```
   const int sensorPin = 14;
   int state = 0;
   int awayHomeMode = 0;
   BlynkTimer timer;
   ```

4. **Función setup()**

   Esta función inicializa el sensor PIR como entrada, configura la comunicación serial, se conecta a WiFi y configura Blynk.

   - Usamos `timer.setInterval(1000L, myTimerEvent)` para establecer el intervalo del temporizador en `setup()`, aquí lo configuramos para ejecutar la función `myTimerEvent()` cada **1000ms**. Puedes modificar el primer parámetro de `timer.setInterval(1000L, myTimerEvent)` para cambiar el intervalo entre ejecuciones de `myTimerEvent`.

   ```
   void setup() {

       pinMode(sensorPin, INPUT);  // Set PIR sensor pin as input
       Serial.begin(115200);           // Start serial communication at 115200 baud␣
   ↪rate for debugging

       // Configure Blynk and connect to WiFi
       Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

       timer.setInterval(1000L, myTimerEvent);  // Setup a function to be called every␣
   ↪second
   }
   ```

5. **Función loop()**

   La función loop ejecuta continuamente Blynk y las funciones del temporizador de Blynk.

```
void loop() {
   Blynk.run();
   timer.run();
}
```

6. **Interacción con la Aplicación Blynk**

   Estas funciones se llaman cuando el dispositivo se conecta a Blynk y cuando hay un cambio en el estado del pin virtual V0 en la aplicación Blynk.

   - Cada vez que el dispositivo se conecta al servidor Blynk, o se reconecta debido a condiciones de red deficientes, se llama a la función `BLYNK_CONNECTED()`. El comando `Blynk.syncVirtual()` solicita el valor de un Pin Virtual único. El Pin Virtual especificado realizará la llamada `BLYNK_WRITE()`.
   - Siempre que el valor de un pin virtual en el servidor BLYNK cambia, se activará `BLYNK_WRITE()`.

```
// This function is called every time the device is connected to the Blynk.Cloud
BLYNK_CONNECTED() {
    Blynk.syncVirtual(V0);
}

// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0) {
    awayHomeMode = param.asInt();
    // additional logic
}
```

7. **Manejo de Datos**

   Cada segundo, la función `myTimerEvent()` llama a `sendData()`. Si el modo ausente está habilitado en Blynk, verifica el sensor PIR y envía una notificación a Blynk si se detecta movimiento.

   - Usamos `Blynk.virtualWrite(V1, "¡Alguien en tu casa! ¡Por favor, revisa!");` para cambiar el texto de una etiqueta.
   - Usa `Blynk.logEvent("intrusion_detected");` para registrar el evento en Blynk.

```
void myTimerEvent() {
   sendData();
}

void sendData() {
   if (awayHomeMode == 1) {
      state = digitalRead(sensorPin);  // Read the state of the PIR sensor

      Serial.print("state:");
      Serial.println(state);

      // If the sensor detects movement, send an alert to the Blynk app
      if (state == HIGH) {
        Serial.println("Somebody here!");
        Blynk.virtualWrite(V1, "Somebody in your house! Please check!");
         Blynk.logEvent("intrusion_detected");
```

(continué en la próxima página)

(proviene de la página anterior)

```
        }
    }
}
```

Reference

-
-
-
-
-
-
-

2.53 8.10 Aplicación Android - Operación de LED RGB mediante Arduino y Bluetooth

El objetivo de este proyecto es desarrollar una aplicación Android capaz de manipular el tono de un LED RGB a través de un smartphone utilizando tecnología Bluetooth.

Esta aplicación Android se construirá utilizando una plataforma web complementaria conocida como MIT App Inventor 2. El proyecto presenta una excelente oportunidad para familiarizarse con la interfaz de un Arduino con un smartphone.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

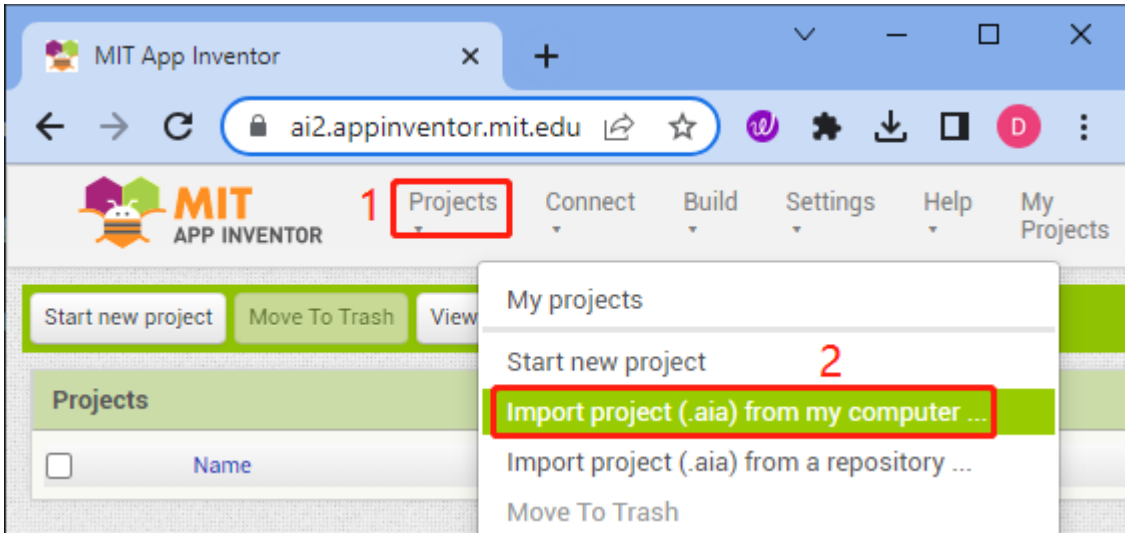
INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED RGB	

1. Creación de la Aplicación Android

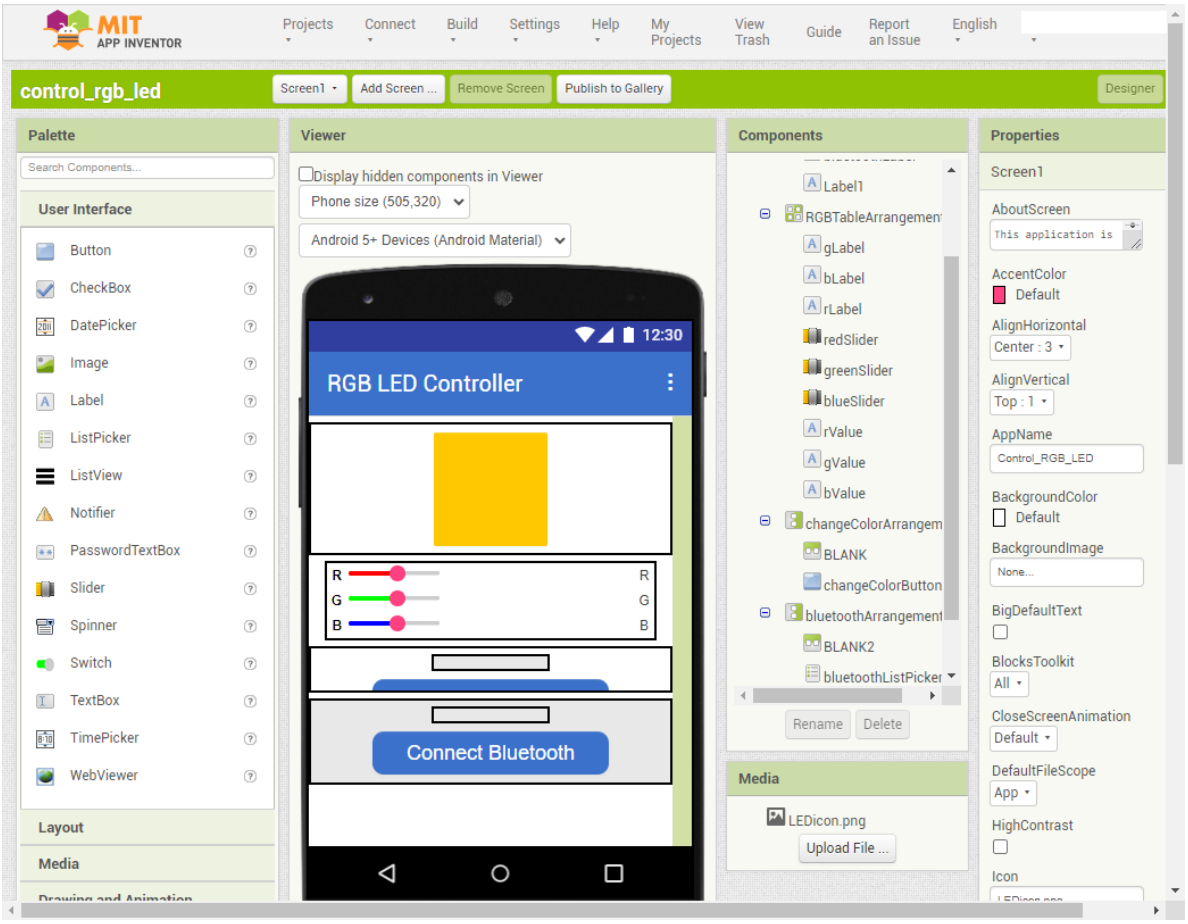
La aplicación Android se diseñará utilizando una aplicación web gratuita conocida como . MIT App Inventor sirve como un excelente punto de partida para el desarrollo de Android, gracias a sus intuitivas características de arrastrar y soltar que permiten la creación de aplicaciones sencillas.

Ahora, comencemos.

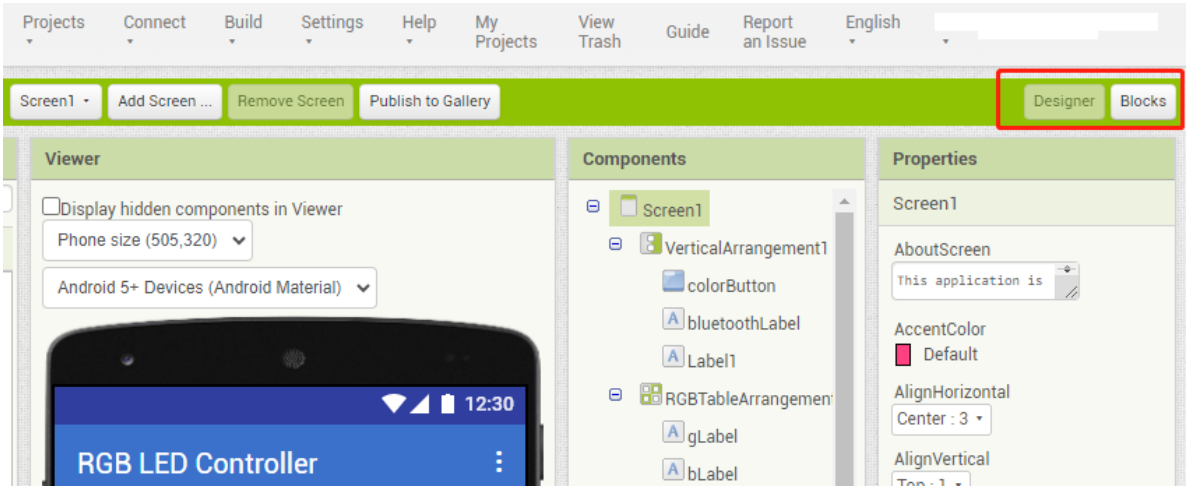
1. Aquí está la página de inicio de sesión: http://ai2.appinventor.mit.edu. Necesitarás una cuenta de Google para registrarte en MIT App Inventor.
2. Después de iniciar sesión, navega a **Proyectos** -> **Importar proyecto (.aia) desde mi computadora**. Posteriormente, sube el archivo `control_rgb_led.aia` ubicado en la ruta `esp32-starter-kit-main\c\codes\iot_10_bluetooth_app_inventor`.



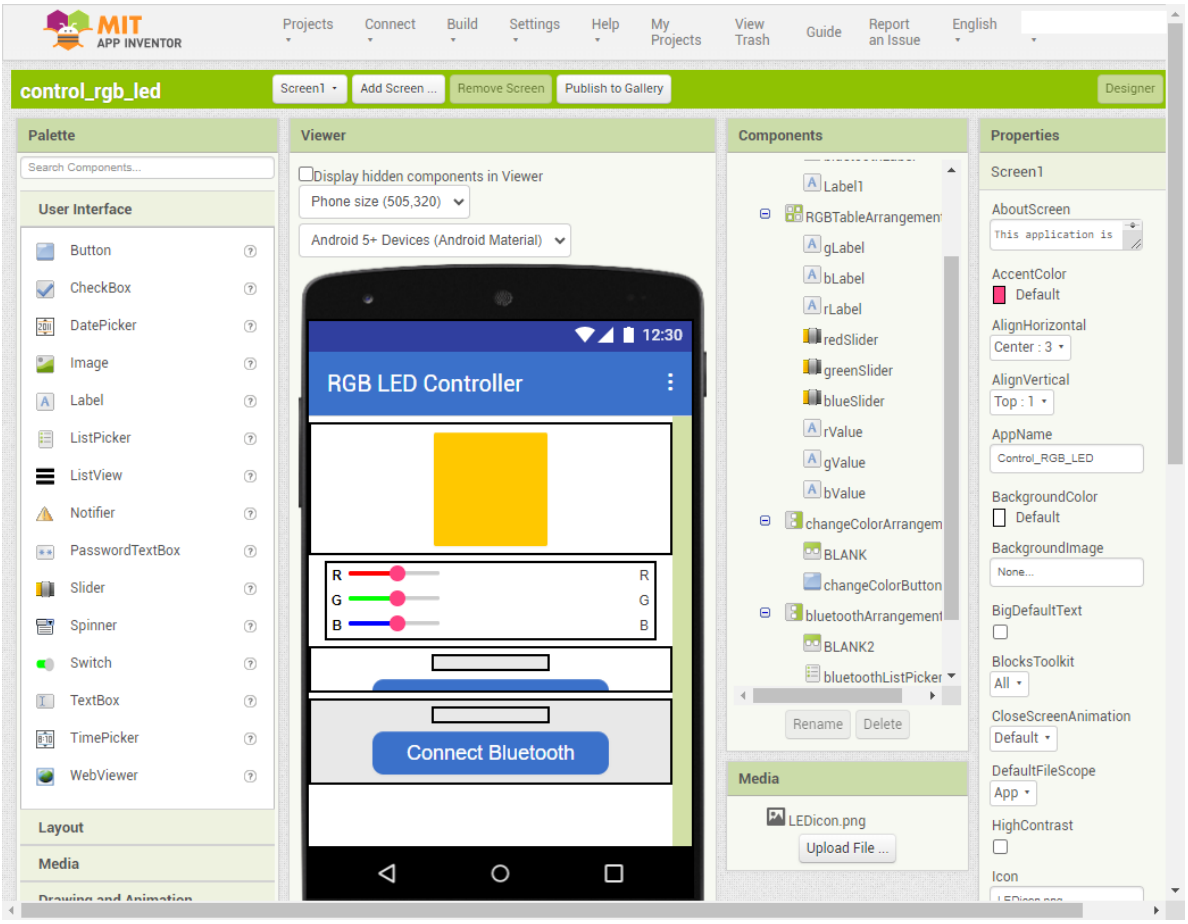
3. Al subir el archivo `.aia`, verás la aplicación en el software **MIT App Inventor**. Esta es una plantilla preconfigurada. Puedes modificar esta plantilla después de haberte familiarizado con **MIT App Inventor** a través de los siguientes pasos.



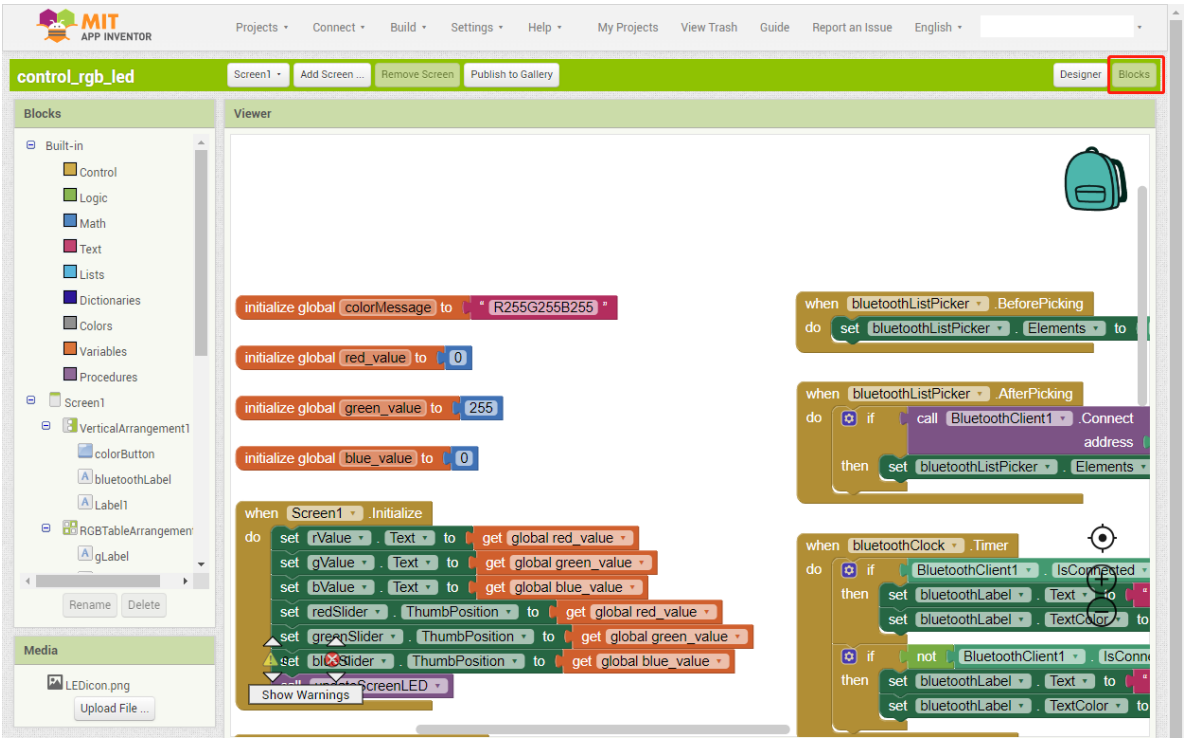
4. En **MIT App Inventor**, tienes 2 secciones principales: el **Diseñador** y los **Bloques**.



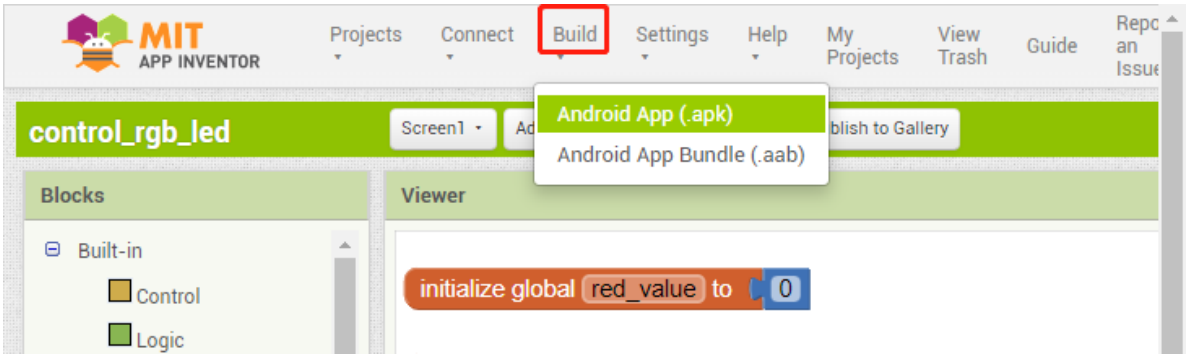
5. El **Diseñador** te permite agregar botones, texto, pantallas y modificar la estética general de tu aplicación.



6. Posteriormente, tienes la sección de **Bloques**. La sección de **Bloques** facilita la creación de funciones a medida para tu aplicación.



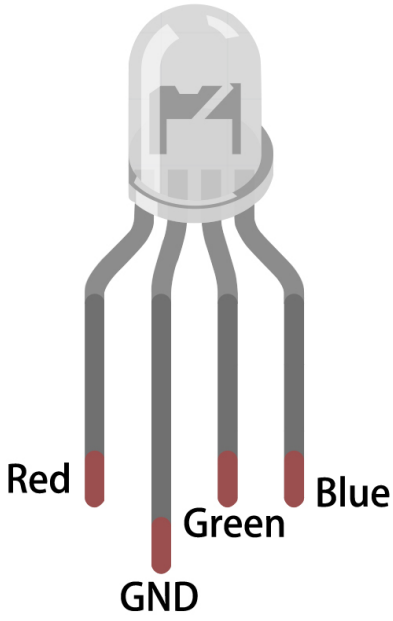
7. Para instalar la aplicación en un smartphone, navega a la pestaña **Construir**.



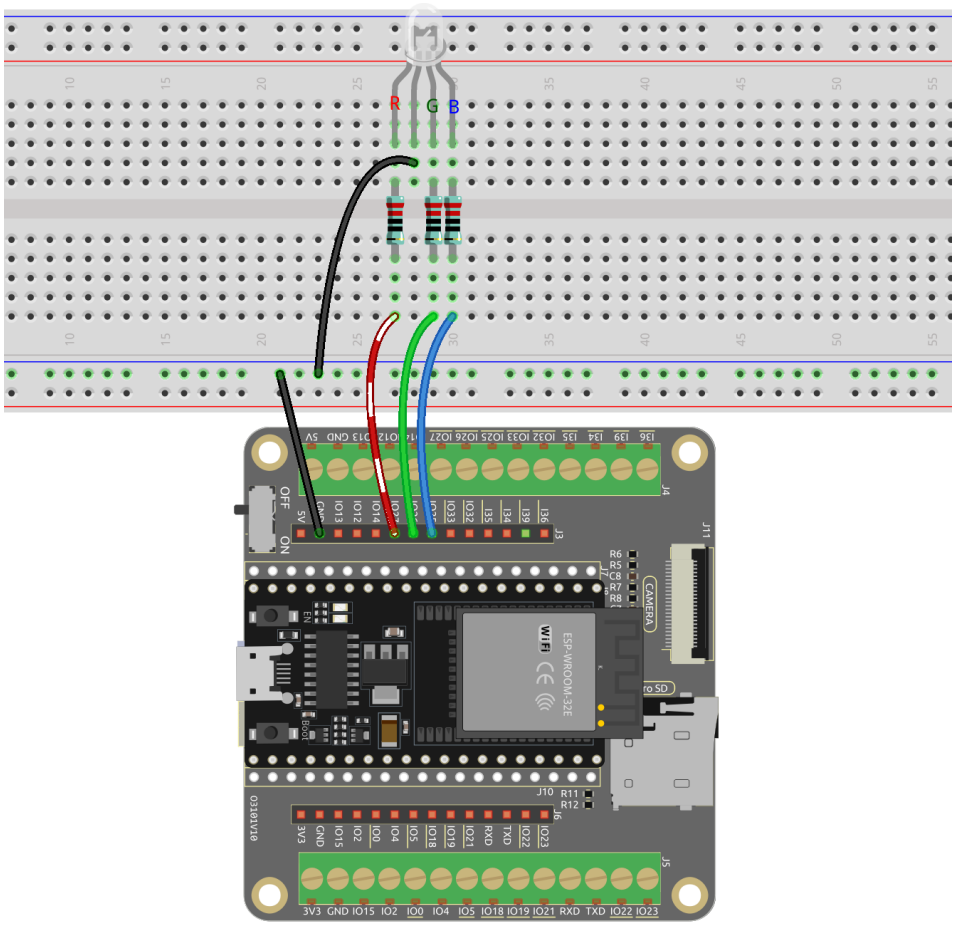
- Puedes generar un archivo `.apk`. Después de seleccionar esta opción, aparecerá una página que te permitirá elegir entre descargar un archivo `.apk` o escanear un código QR para la instalación. Sigue la guía de instalación para completar la instalación de la aplicación.
- Si deseas subir esta app a **Google Play** u otro mercado de aplicaciones, puedes generar un archivo `.aab`.

2. Subida del código

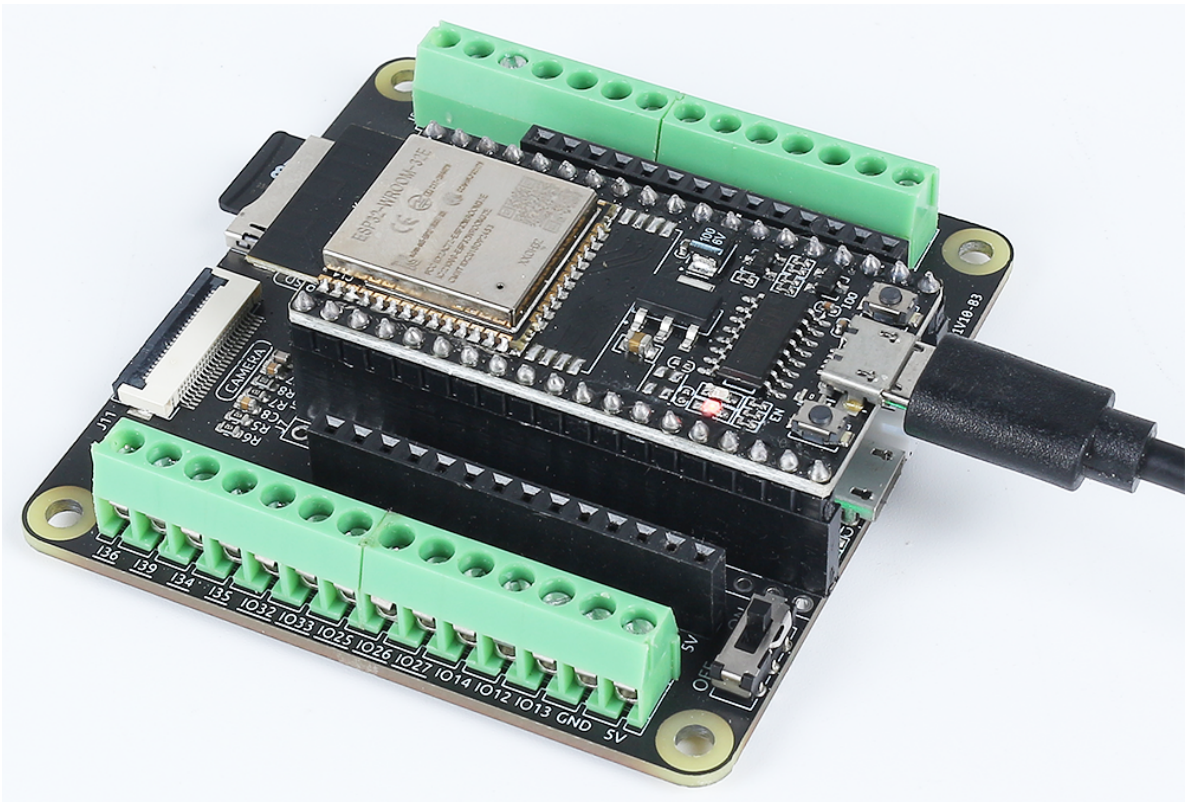
1. Construye el circuito.



El LED RGB consta de 4 pines: el pin alargado es el pin cátodo común, típicamente conectado a GND; el pin a la izquierda del pin más largo representa el Rojo; y los dos pines a la derecha simbolizan el Verde y el Azul.



2. Posteriormente, conecta el ESP32-WROOM-32E a tu computadora utilizando un cable USB.

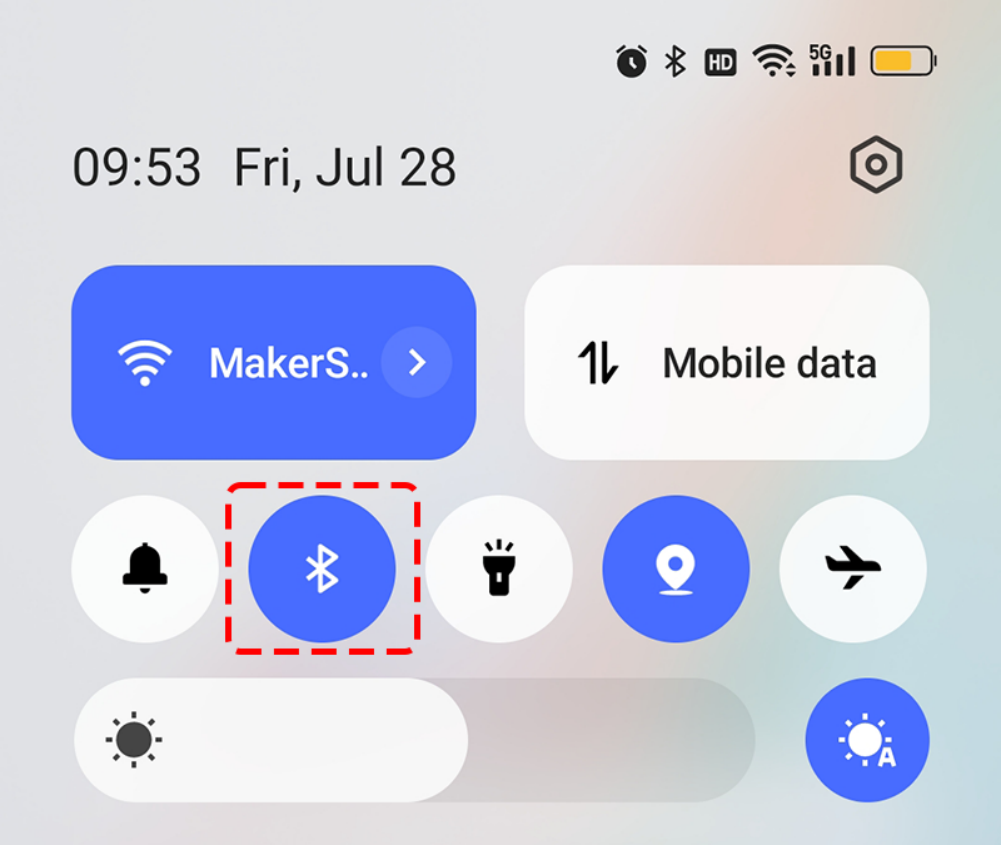


3. Abre el archivo `iot_10_bluetooth_app_inventor.ino` situado en la carpeta `esp32-starter-kit-main\c\codes\iot_10_bluetooth_app_inventor`, o copia el código en el IDE de Arduino.
4. Tras seleccionar la placa adecuada (**ESP32 Dev Module**) y el puerto, haz clic en el botón **Subir**.

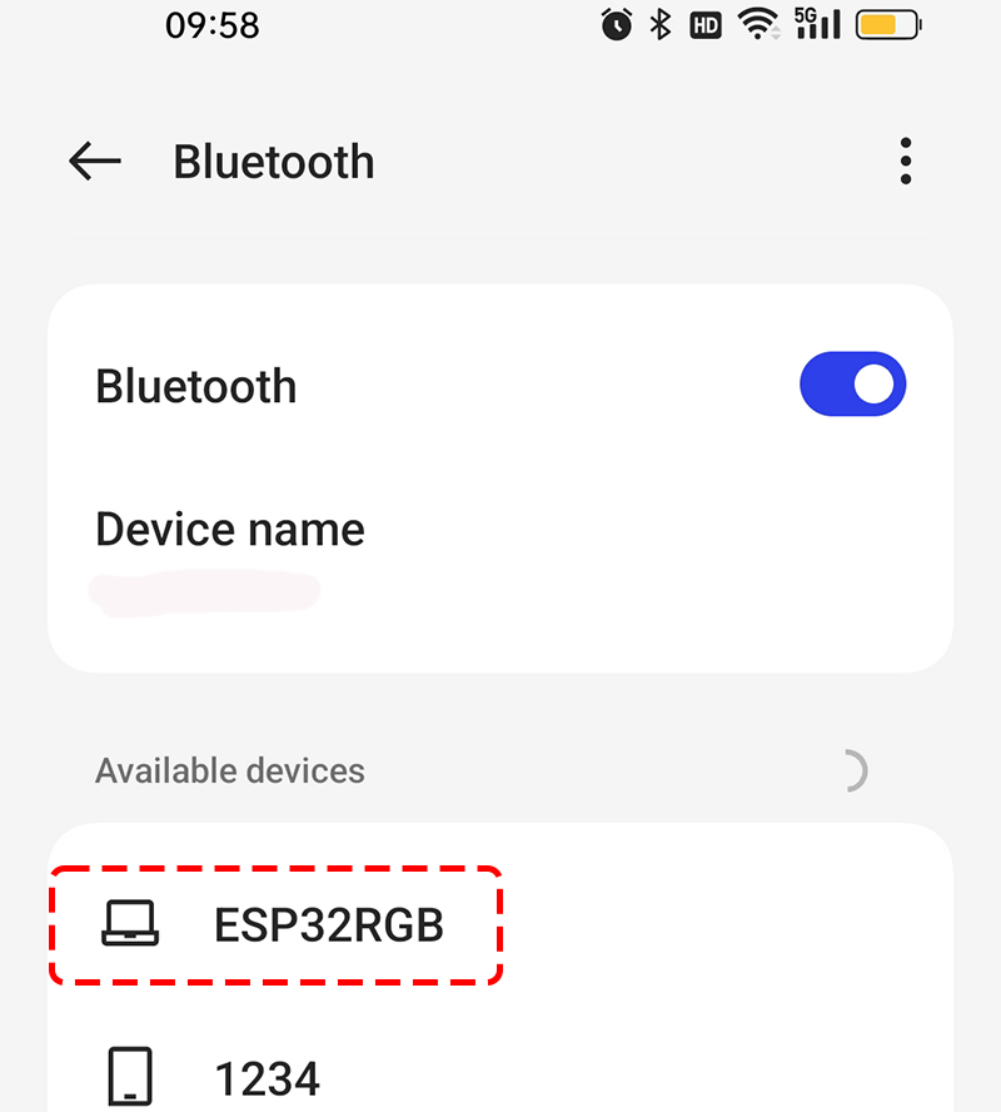
3. Conexión de la App y ESP32

Asegúrate de que la aplicación creada anteriormente esté instalada en tu smartphone.

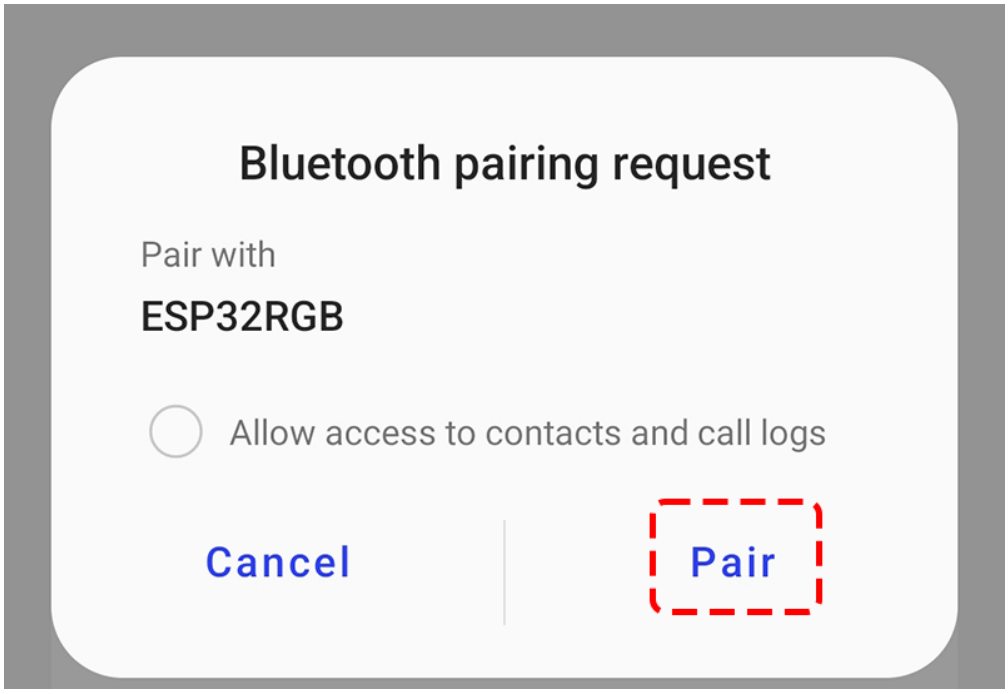
1. Inicialmente, activa el **Bluetooth** en tu smartphone.



2. Navega a los **Ajustes de Bluetooth** en tu smartphone y busca **ESP32RGB**.

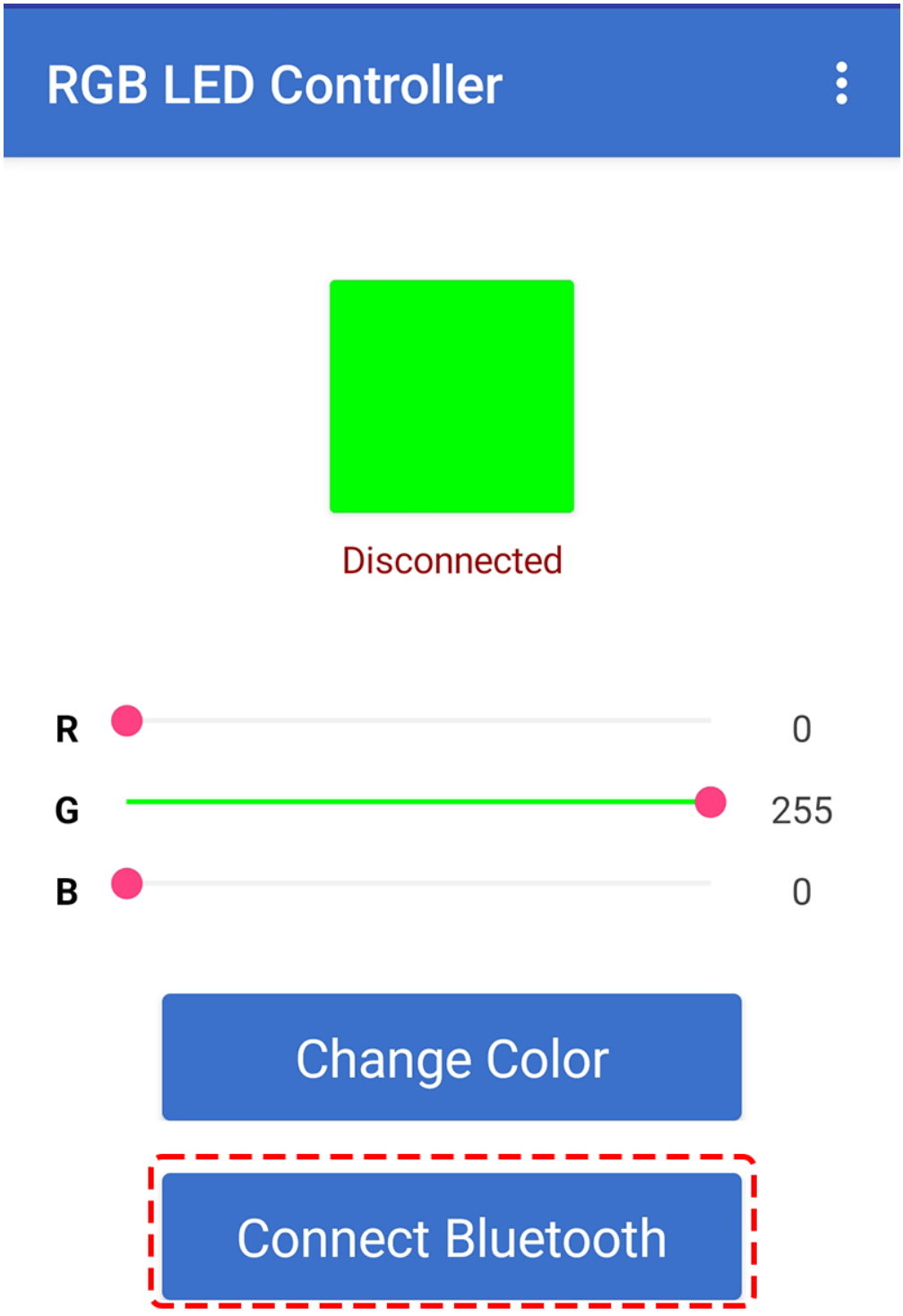


3. Tras hacer clic, acepta la solicitud de **Emparejamiento** en la ventana emergente.

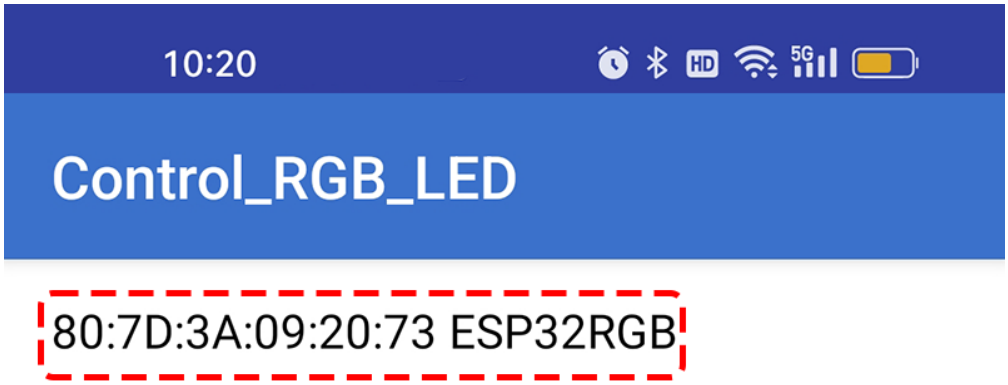


4. Ahora abre la aplicación **Control_RGB_LED** que acabas de instalar.

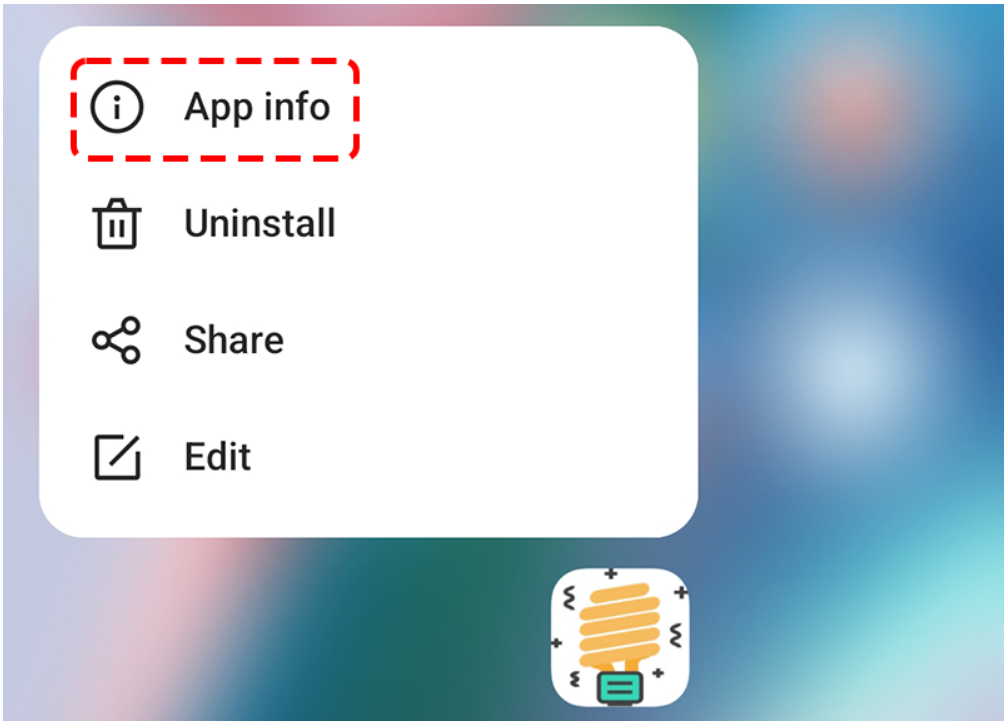
5. En la aplicación, haz clic en **Conectar Bluetooth** para establecer una conexión entre la aplicación y el ESP32.



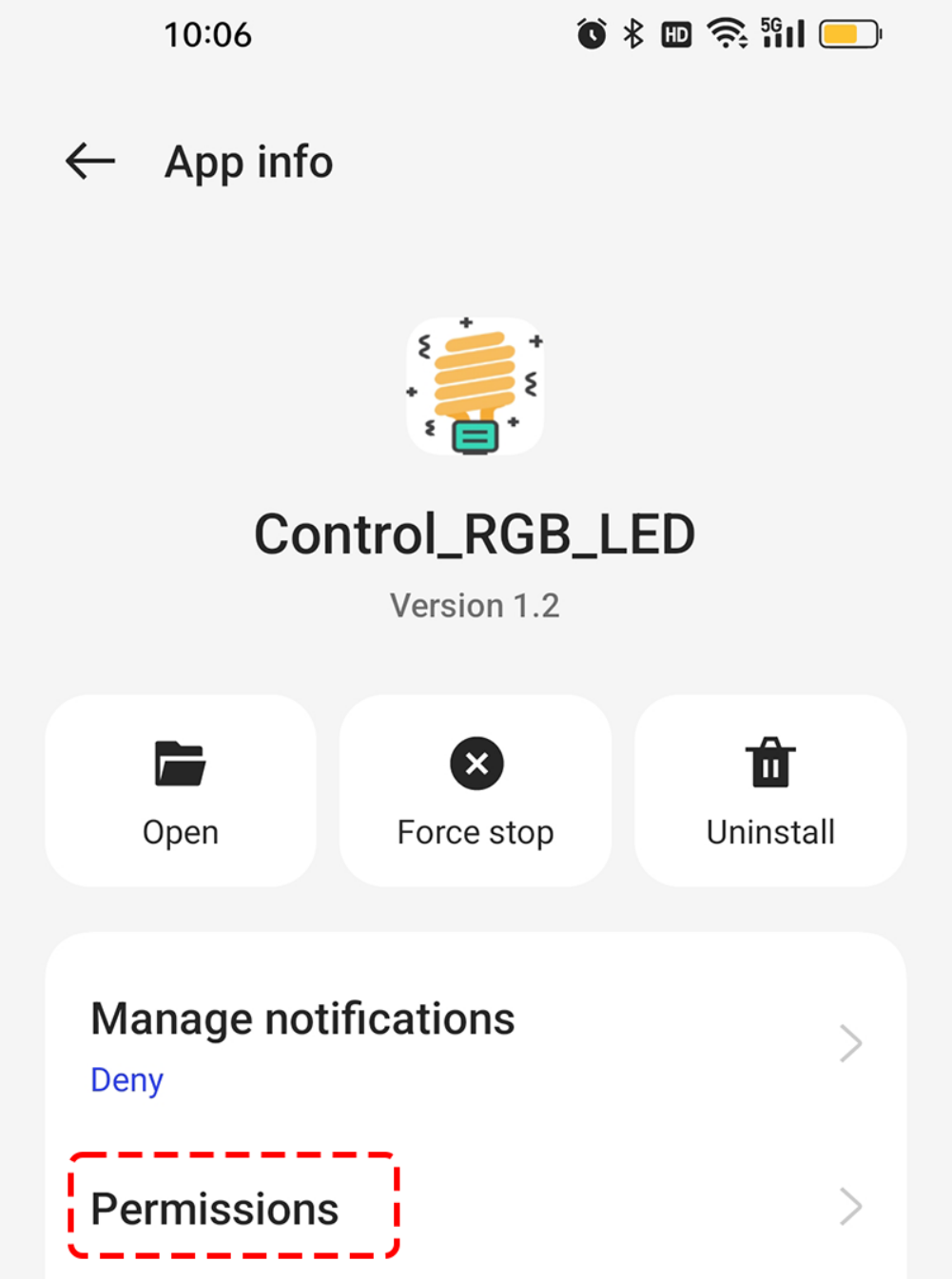
6. Selecciona el `xx.xx.xx.xx.xx.xx ESP32RGB` que aparezca. Si cambiaste `SerialBT.begin("ESP32RGB");` en el código, entonces selecciona el nombre que configuraste.



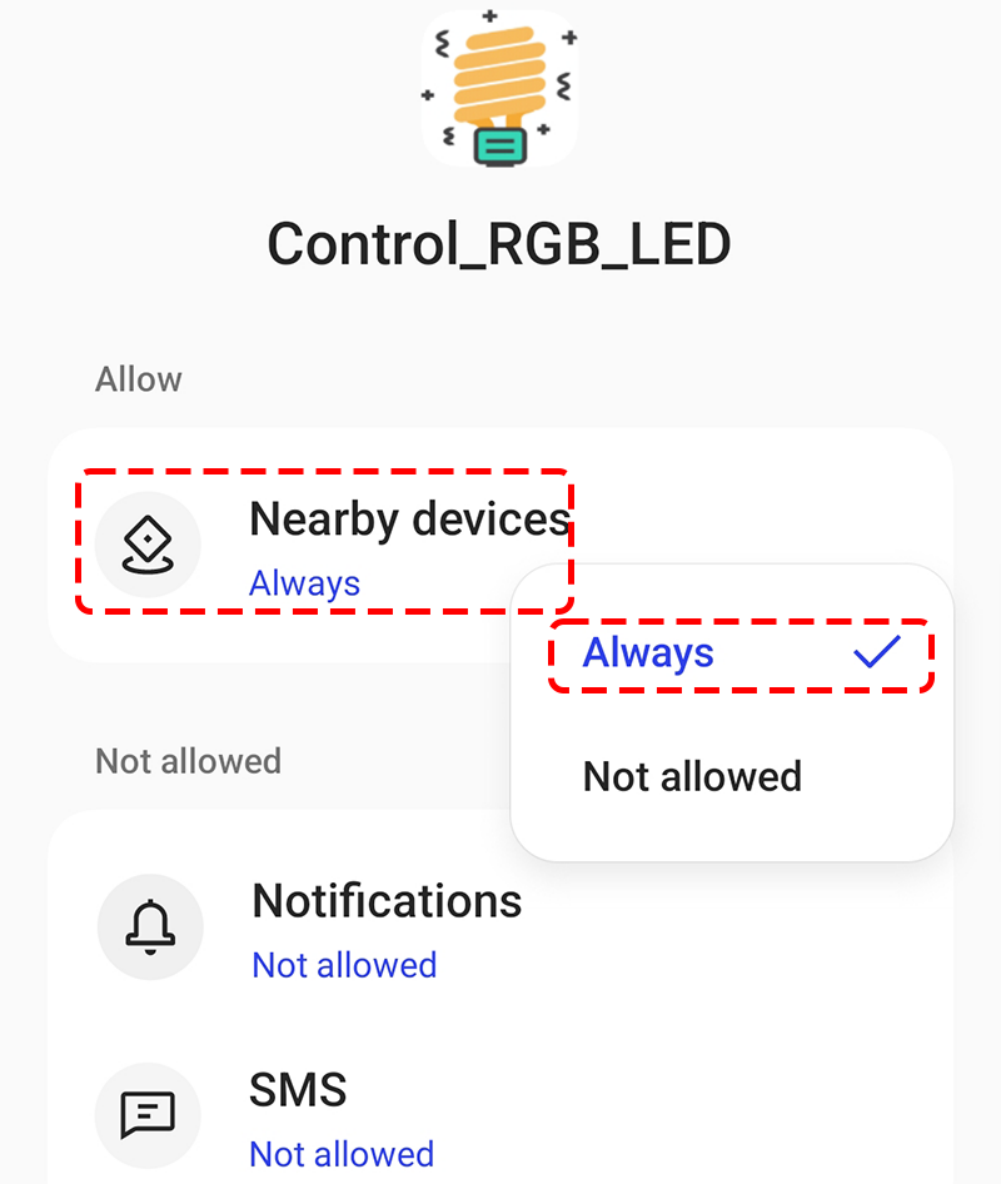
7. Si has esperado un rato y aún no puedes ver ningún nombre de dispositivo, puede ser que esta aplicación no tenga permiso para escanear dispositivos cercanos. En este caso, necesitarás ajustar los ajustes manualmente.
    - Mantén presionado el icono de la aplicación y haz clic en **Información de la APP** que aparece. Si tienes otro método para acceder a esta página, síguelo.



    - Navega a la página de **Permisos**.

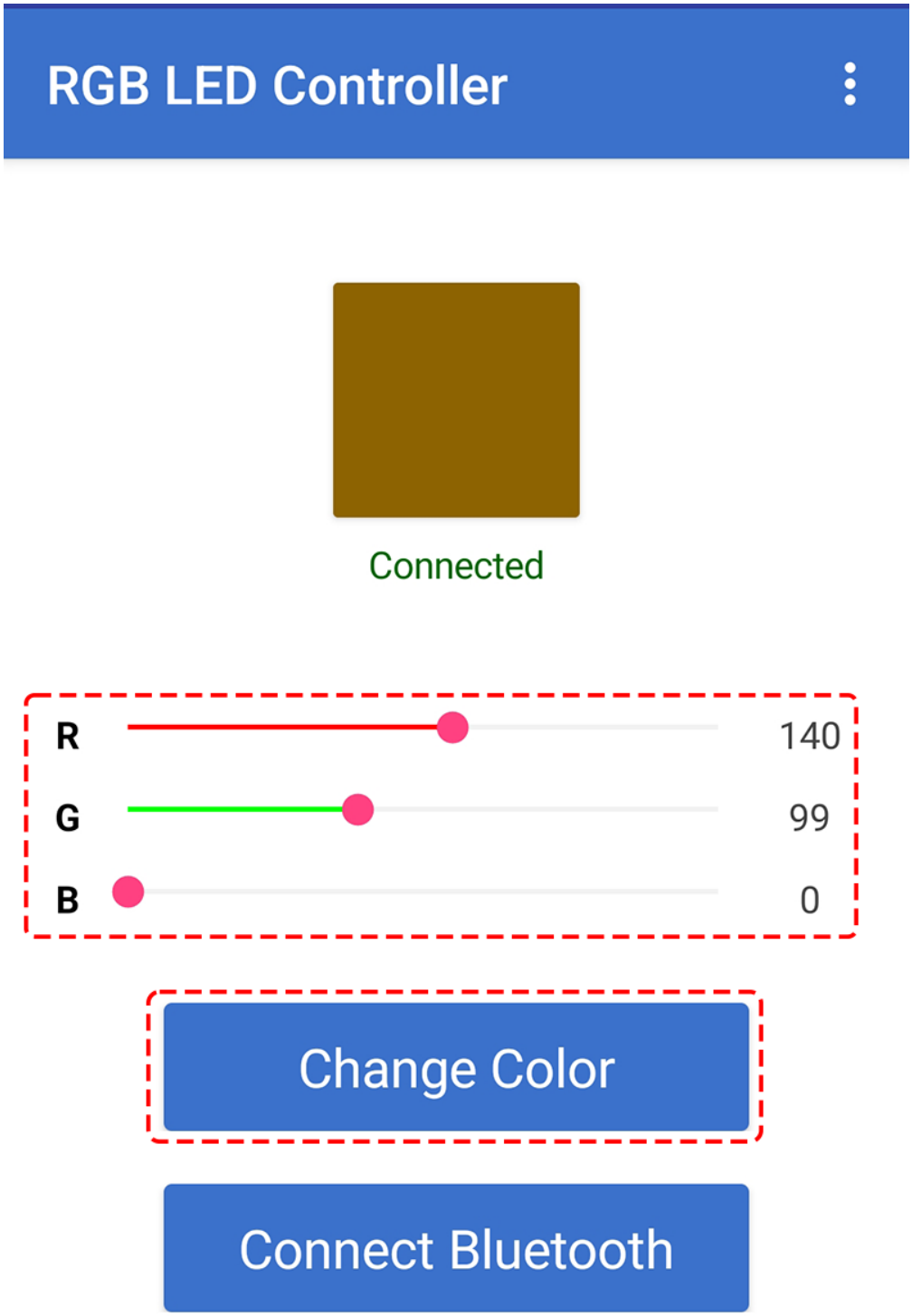


- Localiza **Dispositivos cercanos**, y selecciona **Siempre** para permitir que esta aplicación escanee dispositivos cercanos.



- Ahora, reinicia la aplicación y repite los pasos 5 y 6 para conectarte exitosamente al Bluetooth.

8. Una vez establecida la conexión exitosamente, volverás automáticamente a la página principal, donde se mostrará como conectado. Ahora puedes ajustar los valores RGB y cambiar el color de la pantalla RGB presionando el botón **Cambiar Color**.


RGB LED Controller
Connected
R
140
G
99
B
0
Change Color
Connect Bluetooth

CAPÍTULO 3

Curso de Video de Arduino

Proyectos

3.1 SERIE ESP32 #0: Nueva Serie Esp32 2024

Descubre las novedades de la actualizada serie ESP32 para este 2024, donde te llevaremos a través de los últimos avances en software y hardware. Desde la transición al Arduino IDE versión 2 hasta la integración de nuevas plataformas y tecnologías, este video marca el inicio de una temporada llena de aprendizaje y exploración. Además, no te pierdas el detallado unboxing del ESP32 Starter Kit de SunFounder, un kit esencial para seguir paso a paso nuestra serie. Si eres nuevo en la programación del ESP32 o buscas profundizar tus conocimientos, este video es para ti.

- **Actualización de Contenidos**: Adaptación de la serie a las últimas versiones de software y plataformas.
- **Participación de la Comunidad**: Tu oportunidad de sugerir temas y contenidos para futuros episodios.
- **ESP32 Starter Kit de SunFounder**: Todo lo que necesitas saber sobre este kit completo, desde su contenido hasta dónde adquirirlo.

Prepárate para una temporada de aprendizaje avanzado con la serie ESP32, diseñada tanto para principiantes como para desarrolladores intermedios y avanzados. ¡Únete a nosotros en esta aventura tecnológica en 2024!

Video

3.2 SERIE ESP32 #1: ¿Que Modelo de ESP32 Elegir?

En este video, te guiaremos a través del proceso de selección del modelo de ESP32 perfecto para tus proyectos de Internet de las Cosas, robótica, electrónica y prototipado. Con una amplia gama de modelos disponibles, entender las diferencias y características específicas es crucial para tomar la decisión correcta. Desde la comparación de modelos hasta consejos sobre qué buscar según las necesidades de tu proyecto, cubrimos todo lo que necesitas saber para elegir sabiamente.

- **Selección del Modelo Adecuado**: Aprende a elegir el ESP32 que mejor se adapte a las necesidades específicas de tu proyecto.

- **Recursos de Espressif**: Descubre cómo utilizar la página oficial de Espressif para comparar y seleccionar productos.
- **Conceptos Clave Explicados**: Entiende las diferencias entre SOC, módulos ESP32 y kits de desarrollo.
- **Series de Módulos ESP32**: Conoce las distintas series de módulos ESP32 y sus aplicaciones.

Ya sea que estés trabajando en un proyecto de monitoreo ambiental compacto o necesites una solución con alto rendimiento y eficiencia energética, este video te proporcionará las herramientas necesarias para seleccionar el ESP32 ideal. ¡Acompáñanos y asegura el éxito de tu próximo proyecto!

Video

3.3 Esp32 #2: How to Setup Vscode for Esp32 Programming

Descubre cómo configurar Visual Studio Code y Arduino IDE 2.0 para programar el ESP32. Soluciona errores de compilación, maneja archivos compilados y selecciona correctamente tu módulo ESP32.

- **Instalación de Arduino IDE 2.0**: Paso a paso para sistemas operativos principales.
- **Configuración del ESP32 en Arduino IDE**: Cómo agregar URLs para instalar paquetes de ESP32.
- **Visual Studio Code para ESP32**: Instalación de la extensión de Arduino y configuración para un desarrollo eficiente.
- **Creación y compilación de sketches**: Ejemplo práctico y solución de problemas comunes.

Video

Related On-line Tutorials

- *1.1 Instalar Arduino IDE (Importante)*

3.4 Serie Esp32: Como Crear un Página Web - Con Websockets

Descubre cómo crear una página web para visualizar en tiempo real la temperatura y humedad con un ESP32 y un sensor DHT. Este tutorial paso a paso te enseñará a configurar el hardware necesario, programar el ESP32 para enviar datos a través de Wi-Fi, y utilizar la librería JustGage para añadir indicadores gráficos a tu página web. Ideal para proyectos de monitoreo ambiental, este video es una guía completa para integrar el ESP32 con tecnologías web.

- **Componentes y Herramientas**: ESP32, sensor DHT, ESP32 Starter Kit de SounFounder.
- **Configuración y Programación**: Detalles sobre la conexión del sensor DHT al ESP32 y la programación necesaria para la lectura y envío de datos.
- **Diseño de la Página Web**: Instrucciones para crear una interfaz de usuario interactiva y visualmente atractiva.
- **Visualización en Tiempo Real**: Accede a la información de temperatura y humedad desde cualquier dispositivo conectado a la red.

Video

CAPÍTULO 4

Para Usuarios de MicroPython

Este capítulo es una guía completa diseñada específicamente para usuarios que prefieren trabajar con MicroPython. Cubre varios temas, incluyendo cómo empezar con MicroPython, trabajar con pantallas, generar sonidos, controlar actuadores, utilizar sensores y explorar proyectos divertidos. Este capítulo proporciona a los usuarios de MicroPython los conocimientos y recursos necesarios para usar efectivamente este kit y desatar su creatividad en la construcción de proyectos emocionantes.

Aquí está el paquete de código completo para el Kit de Inicio ESP32. Puedes hacer clic en el siguiente enlace para descargarlo:

- `Kit de Inicio ESP32 de SunFounder`

Una vez completada la descarga, descomprime el archivo y abre el código de ejemplo relevante o los archivos de proyecto en el software correspondiente. Esto te permitirá navegar y utilizar todo el código y recursos proporcionados por el kit.

1. Comenzar

4.1 1.1 Introducción a MicroPython

MicroPython es una implementación de software de un lenguaje de programación ampliamente compatible con Python 3, escrito en C, que está optimizado para ejecutarse en un microcontrolador.[3][4]

MicroPython consta de un compilador de Python a bytecode y un intérprete en tiempo de ejecución de ese bytecode. El usuario se presenta con un prompt interactivo (el REPL) para ejecutar comandos compatibles inmediatamente. Incluye una selección de bibliotecas centrales de Python; MicroPython incluye módulos que brindan al programador acceso a hardware de bajo nivel.

- Referencia: MicroPython - Wikipedia

4.1.1 La Historia Comienza Aquí

Las cosas cambiaron en 2013 cuando Damien George lanzó una campaña de crowdfunding (Kickstarter).

Damien era un estudiante de pregrado en la Universidad de Cambridge y un apasionado programador de robótica. Quería reducir el mundo de Python de una máquina de gigabytes a una de kilobytes. Su campaña de Kickstarter fue para apoyar su desarrollo mientras convertía su prueba de concepto en una implementación finalizada.

MicroPython está respaldado por una diversa comunidad de Pythonistas que tiene un gran interés en ver el proyecto tener éxito.

Aparte de probar y apoyar la base de código, los desarrolladores proporcionaron tutoriales, bibliotecas de código y portabilidad de hardware, por lo que Damien pudo enfocarse en otros aspectos del proyecto.

- Referencia: realpython

4.1.2 ¿Por qué MicroPython

Aunque la campaña de Kickstarter original lanzó MicroPython como una placa de desarrollo «pyboard» con STM32F4, MicroPython admite muchas arquitecturas de productos basadas en ARM. Los puertos principales soportados son ARM Cortex-M (muchas placas STM32, TI CC3200/WiPy, placas Teensy, series Nordic nRF, SAMD21 y SAMD51), ESP8266, ESP32, PIC de 16 bits, Unix, Windows, Zephyr y JavaScript. En segundo lugar, MicroPython permite una retroalimentación rápida. Esto se debe a que puedes usar REPL para ingresar comandos de manera interactiva y obtener respuestas. Incluso puedes ajustar el código y ejecutarlo inmediatamente en lugar de atravesar el ciclo de codificar-compilar-subir-ejecutar.

Mientras que Python tiene las mismas ventajas, para algunas placas de microcontroladores como el ESP32, son pequeñas, simples y tienen poca memoria para ejecutar el lenguaje Python en absoluto. Es por eso que MicroPython ha evolucionado, manteniendo las principales características de Python y agregando un montón de nuevas para trabajar con estas placas de microcontroladores.

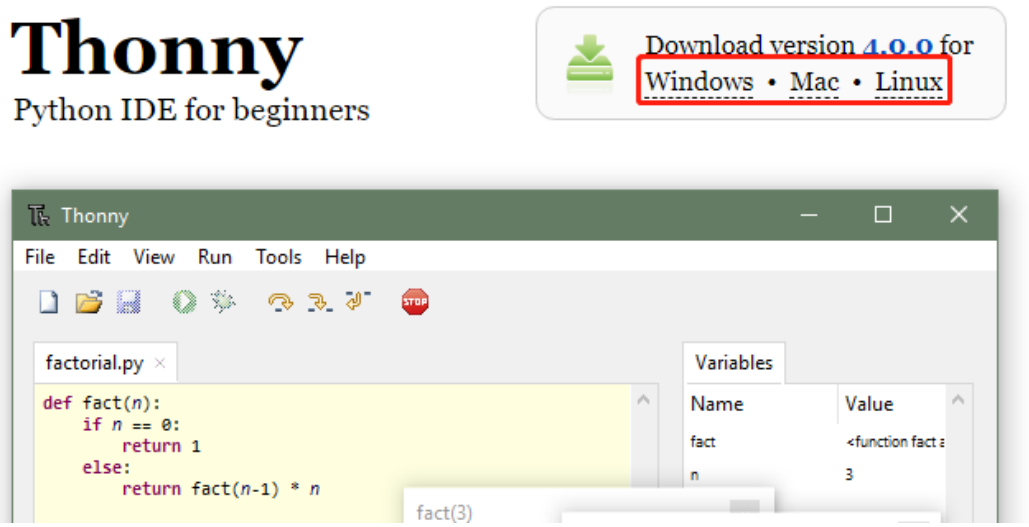
A continuación, aprenderás a instalar MicroPython en el ESP32.

- Referencia: MicroPython - Wikipedia
- Referencia: realpython

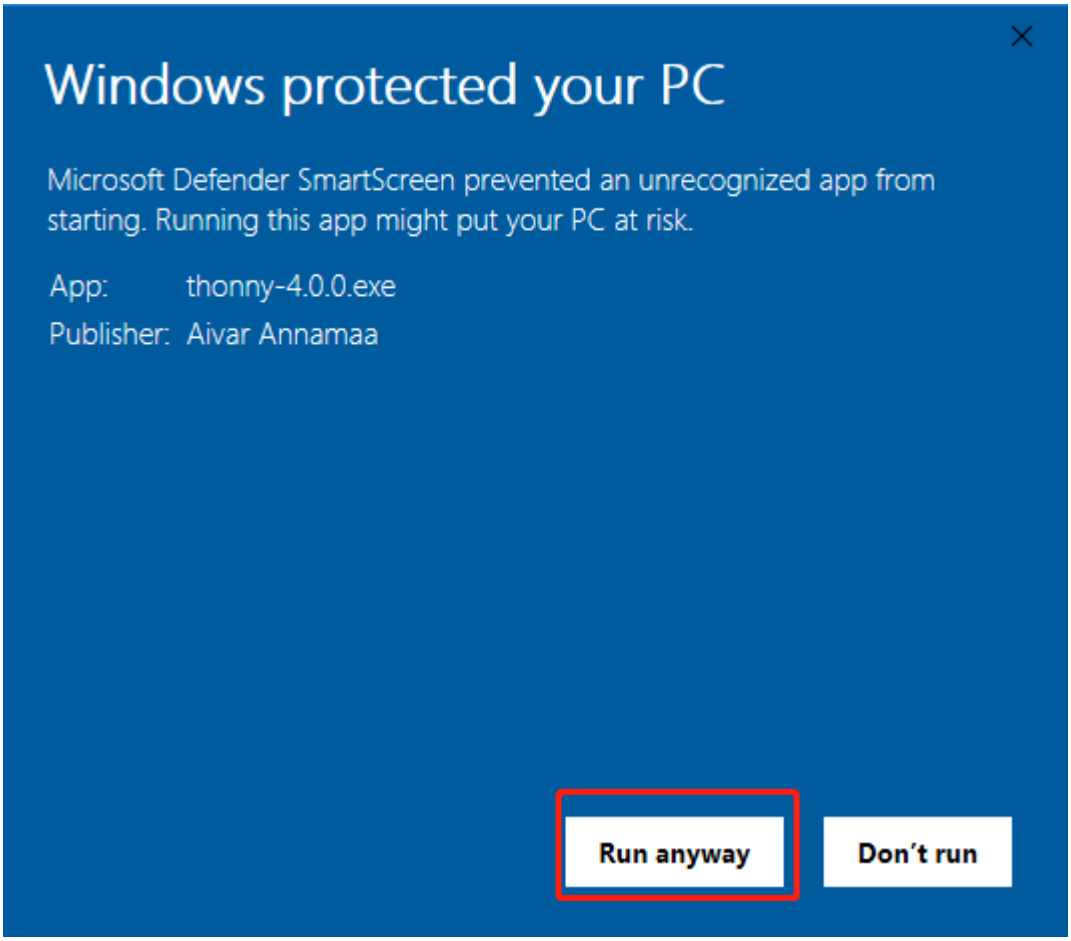
4.2 1.2 Instalar Thonny IDE

Antes de poder comenzar a programar el ESP32 con MicroPython, necesitas un entorno de desarrollo integrado (IDE), aquí recomendamos Thonny. Thonny viene con Python 3.7 incorporado, solo se necesita un instalador simple y ya estás listo para aprender programación.

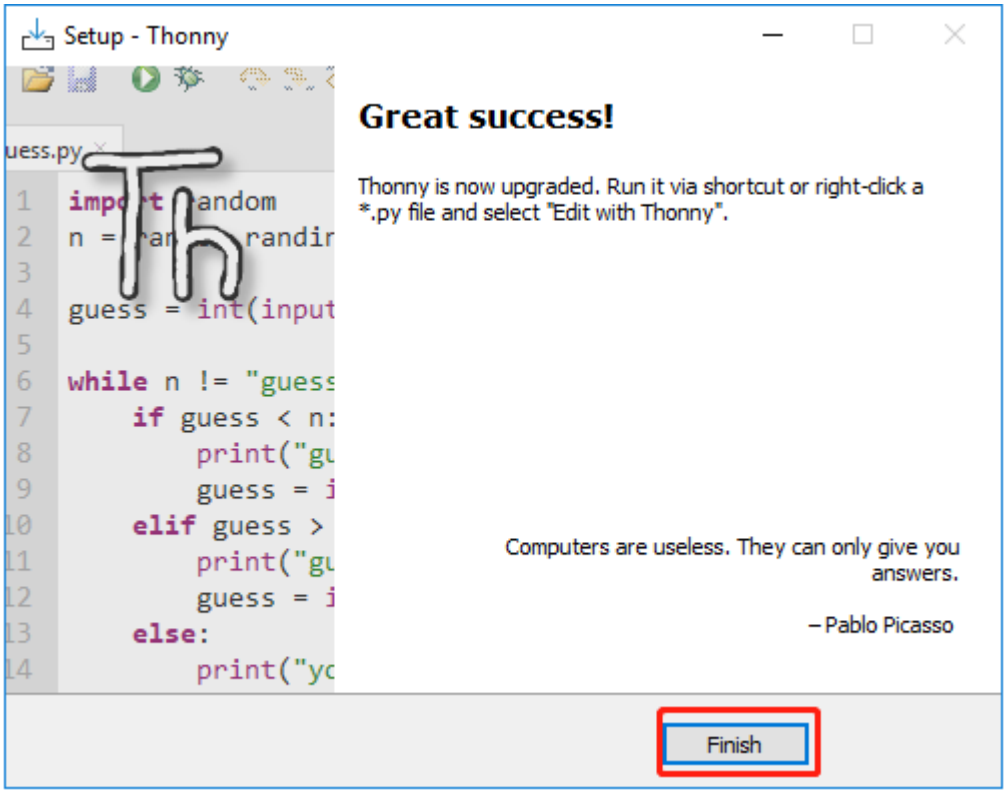
1. Puedes descargarlo visitando el sitio web de . Una vez que abras la página, verás un cuadro de color gris claro en la esquina superior derecha, haz clic en el enlace que corresponda a tu sistema operativo.



2. Los instaladores han sido firmados con un certificado nuevo que aún no ha construido su reputación. Puede que necesites pasar por alto la advertencia de tu navegador (por ejemplo, elegir «Mantener» en lugar de «Descartar» en Chrome) y la advertencia del Windows Defender (**Más información Ejecutar de todas formas**).

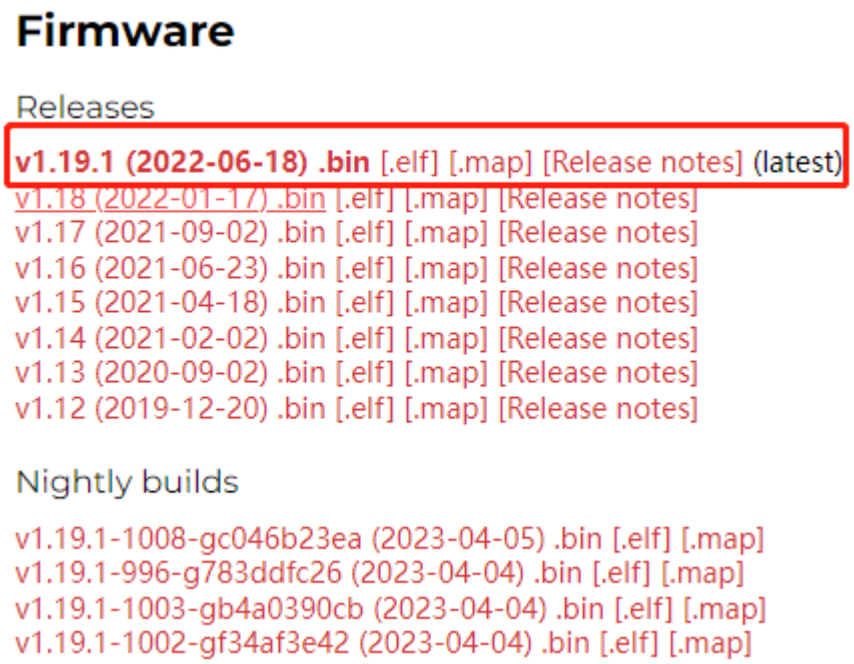


3. A continuación, haz clic en **Siguiente** e **Instalar** para finalizar la instalación de Thonny.

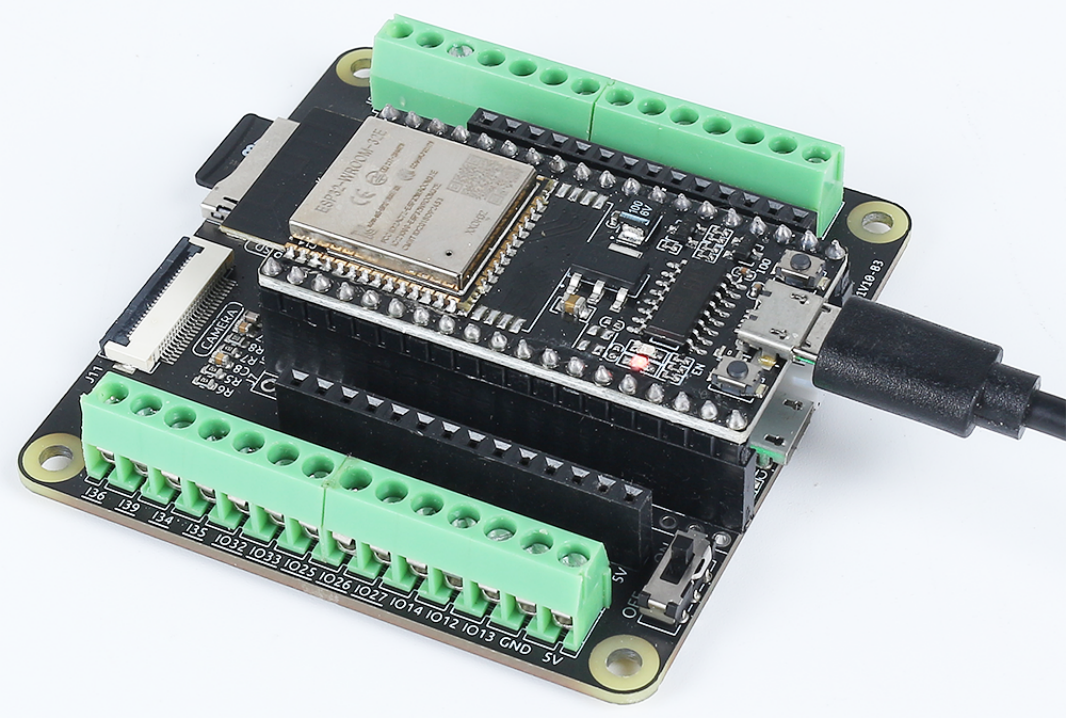


4.3 1.3 Instalar MicroPython en el ESP32(Importante)

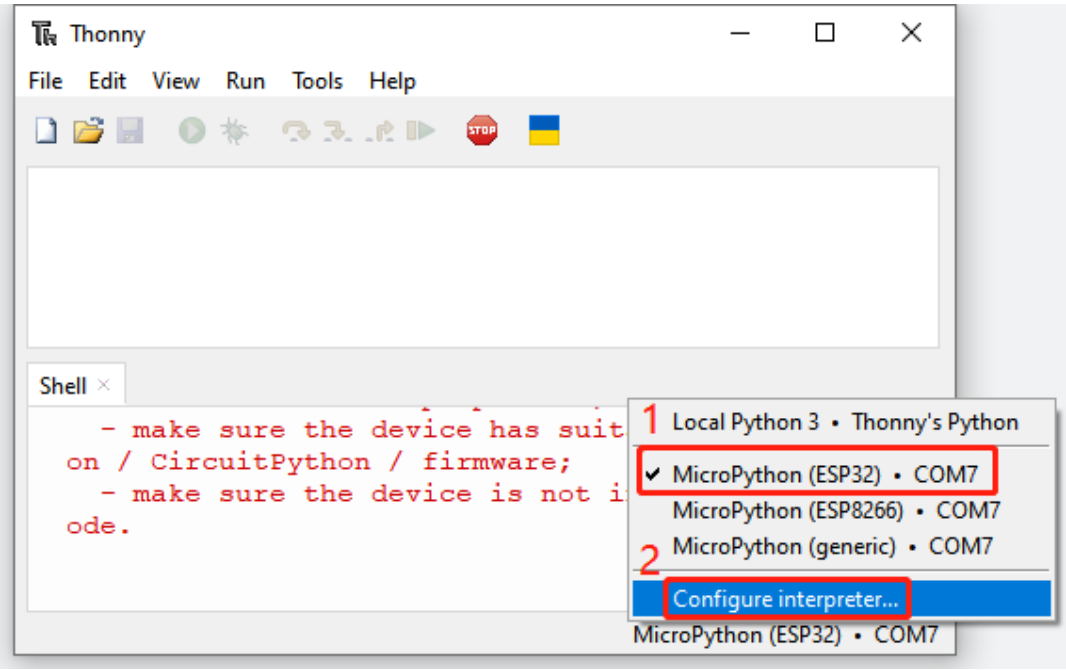
1. Descarga el desde la página oficial de MicroPython y luego descarga la última versión del firmware.



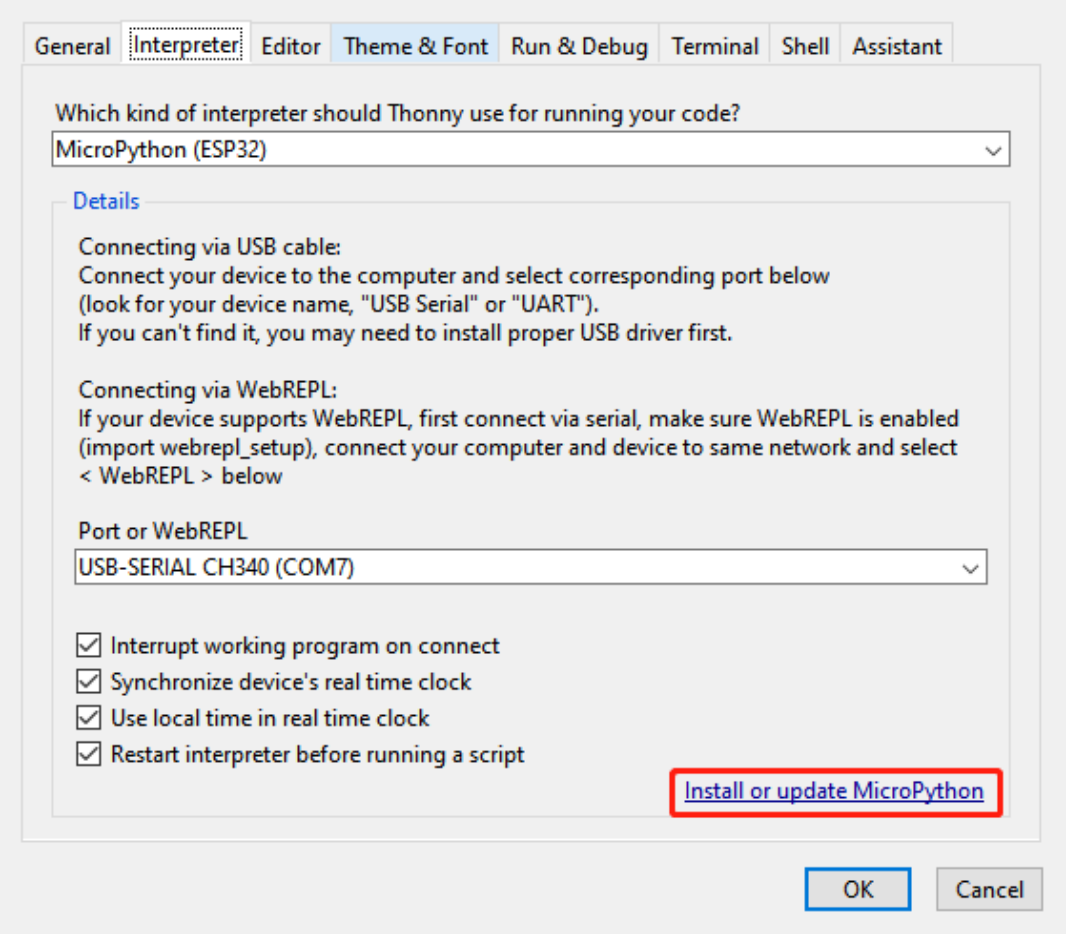
2. Conecta el ESP32 WROOM 32E a tu computadora utilizando un cable Micro USB.



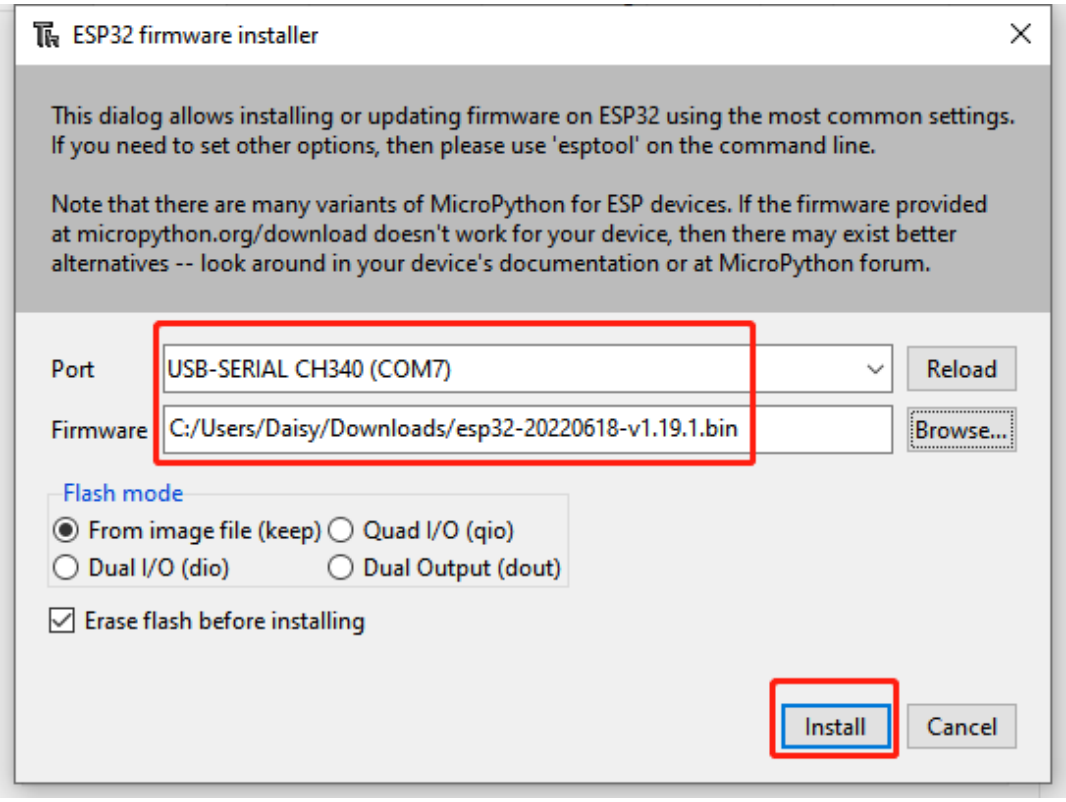
3. Haz clic en la esquina inferior derecha del IDE de Thonny, selecciona **«MicroPython(ESP32).COMXX»** del menú emergente y luego selecciona **«Configurar intérprete»**.



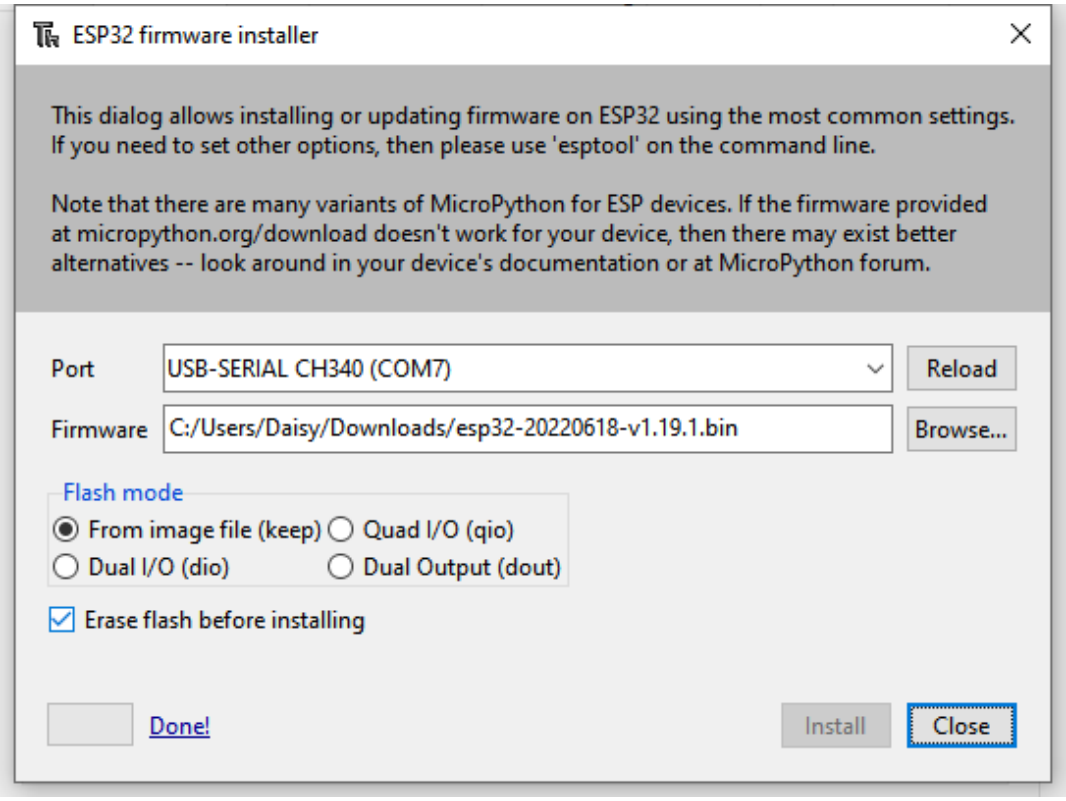
4. Haz clic en **«Instalar o Actualizar MicroPython»** en la nueva ventana emergente.



5. Selecciona el puerto correcto y el firmware que descargaste anteriormente, y haz clic en **«Instalar»**.

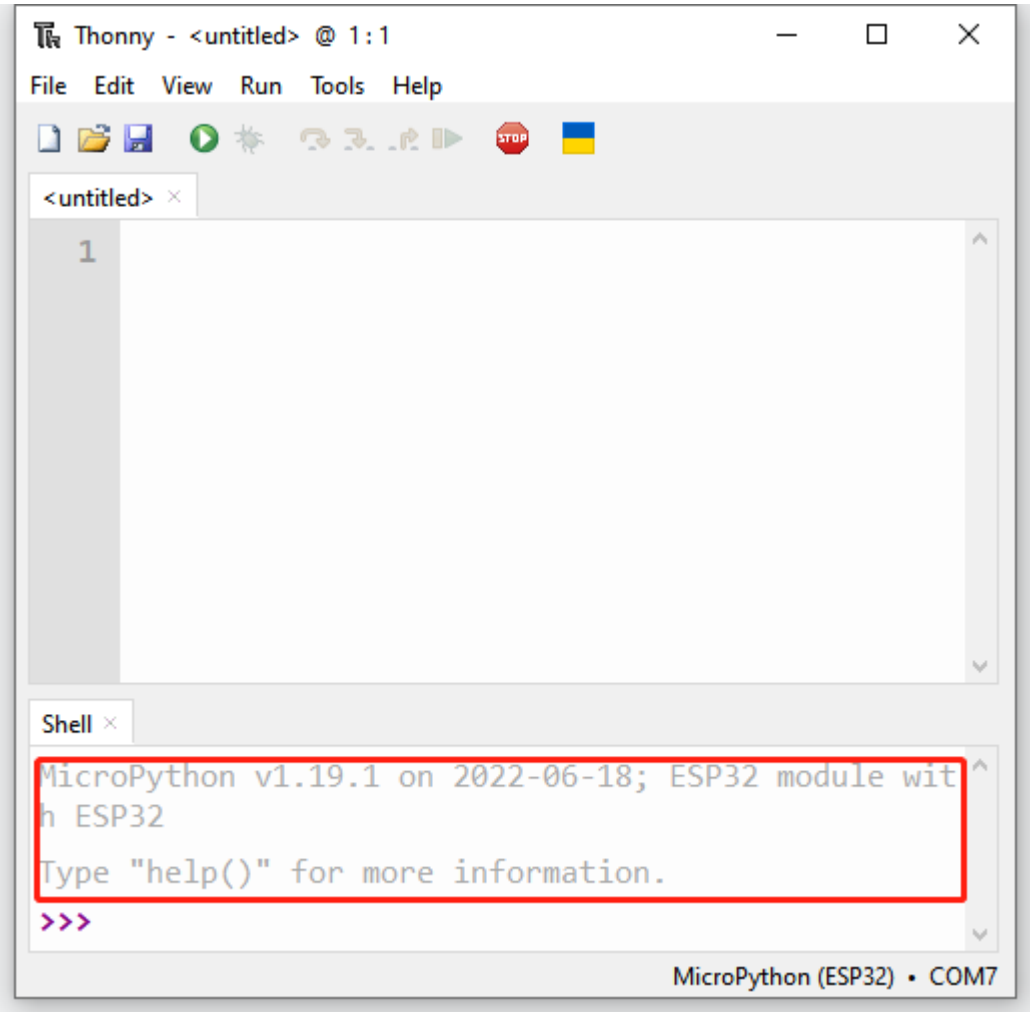


6. Después de una instalación exitosa, puedes cerrar esta página.



7. Cuando regreses a la página principal de Thonny, verás la versión de MicroPython y los mensajes relacionados

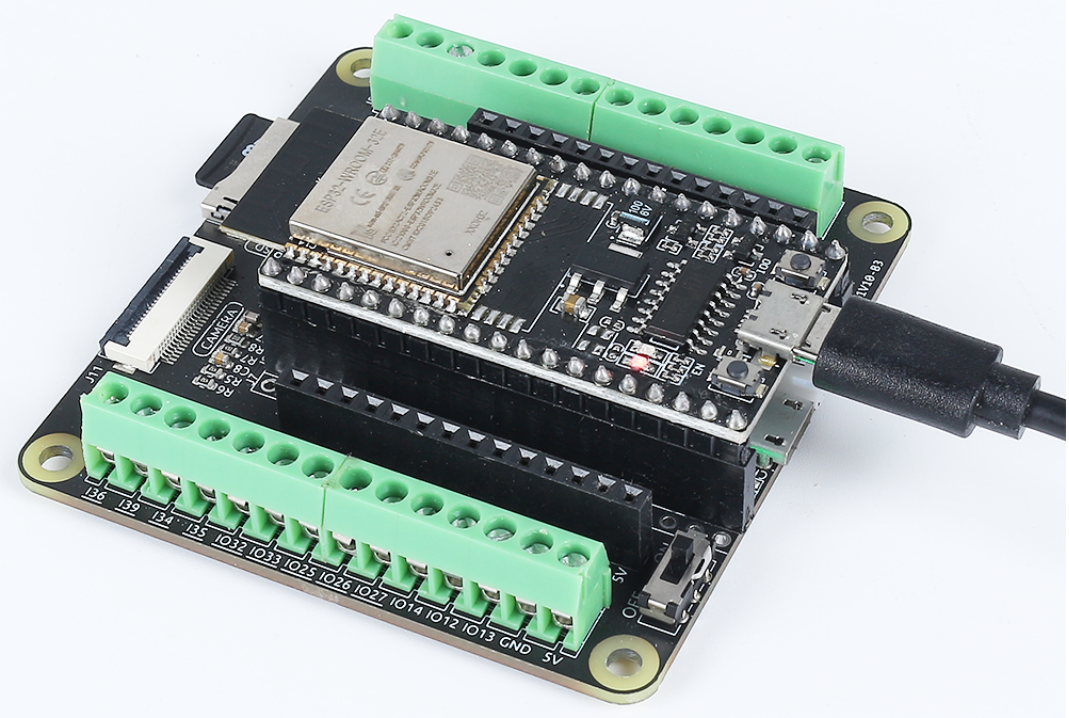
con ESP32, en lugar de mensajes de error en rojo.



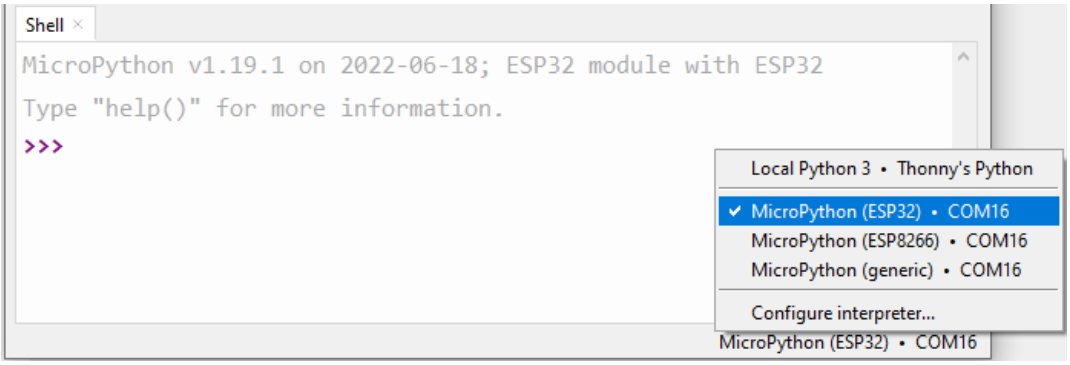
4.4 1.4 Subir las Bibliotecas (Importante)

En algunos proyectos, necesitarás bibliotecas adicionales. Así que aquí subimos estas bibliotecas al ESP32 primero, y luego podemos ejecutar el código directamente después.

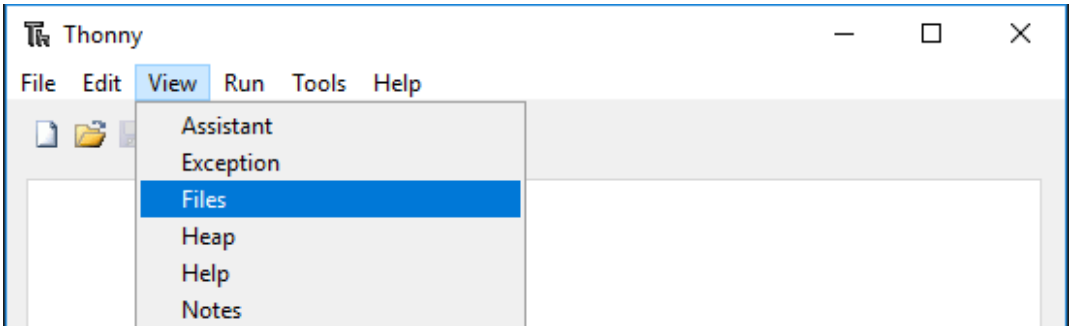
1. Descarga el código relevante desde el enlace a continuación.
    - `Kit de inicio ESP32 de SunFounder`
2. Conecta el ESP32 WROOM 32E a tu computadora usando un cable Micro USB.



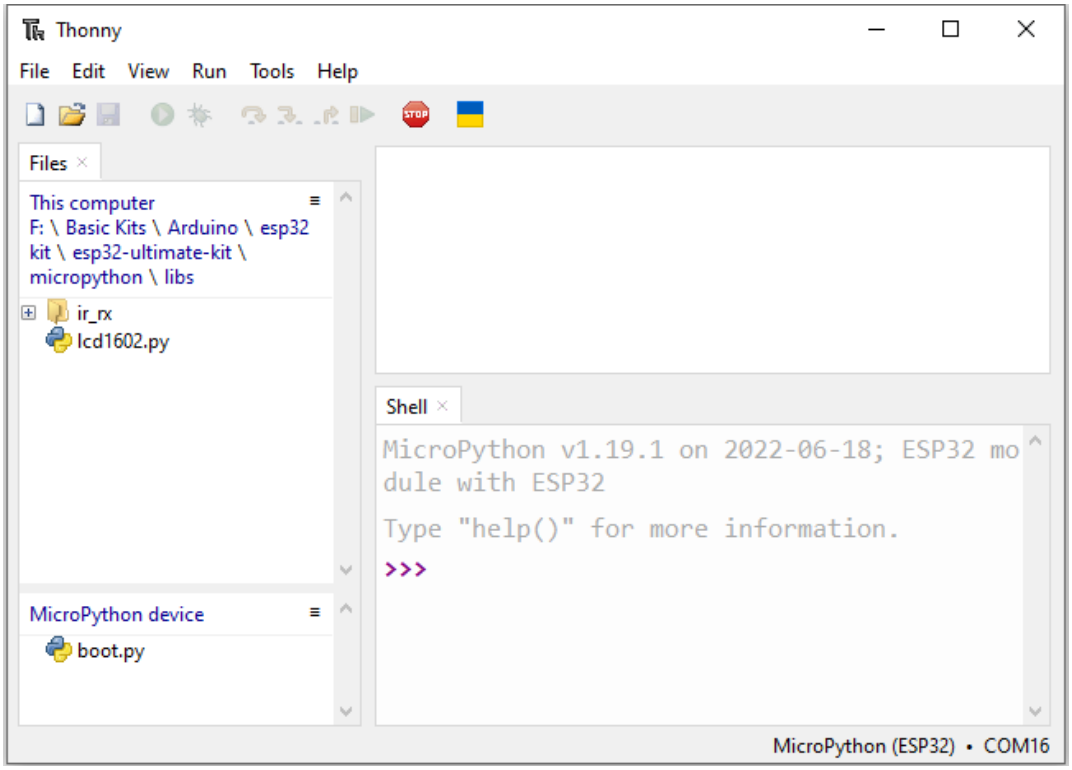
3. Abre Thonny IDE y haz clic en el intérprete «MicroPython (ESP32).COMXX» en la esquina inferior derecha.



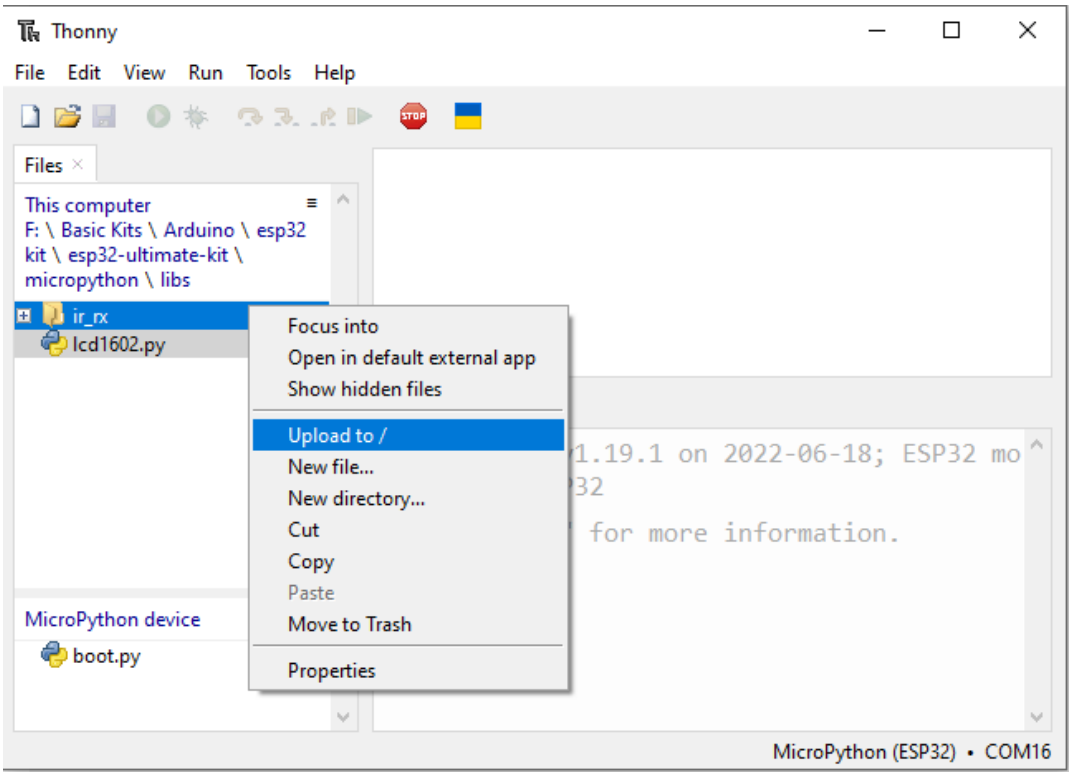
4. En la barra de navegación superior, haz clic en **Ver** -> **Archivos**.



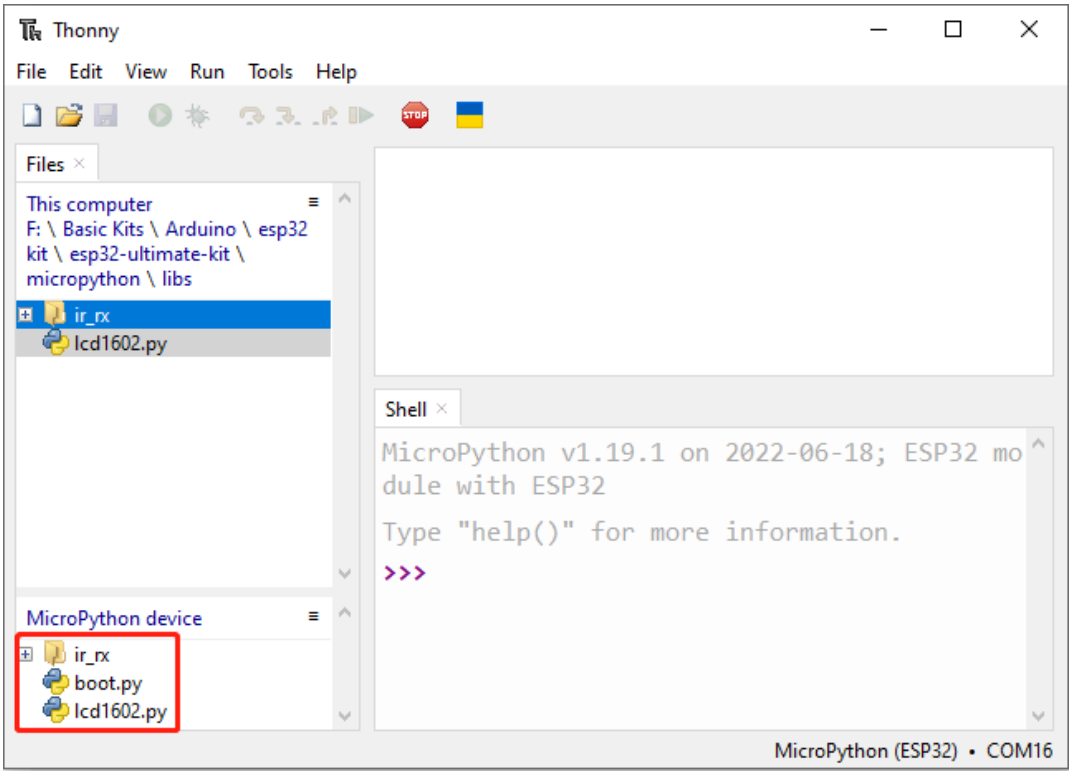
5. Cambia la ruta a la carpeta donde descargaste el paquete de código antes, y luego ve a la carpeta `esp32-starter-kit-main\micropython\libs`.



6. Selecciona todos los archivos o carpetas en la carpeta `libs/`, haz clic derecho y haz clic en **Subir a**, tomará un tiempo subir.



7. Ahora verás los archivos que acabas de subir dentro de tu unidad `Dispositivo MicroPython`.



4.5 1.5 Guía Rápida sobre Thonny

4.5.1 Abrir y Ejecutar Código Directamente

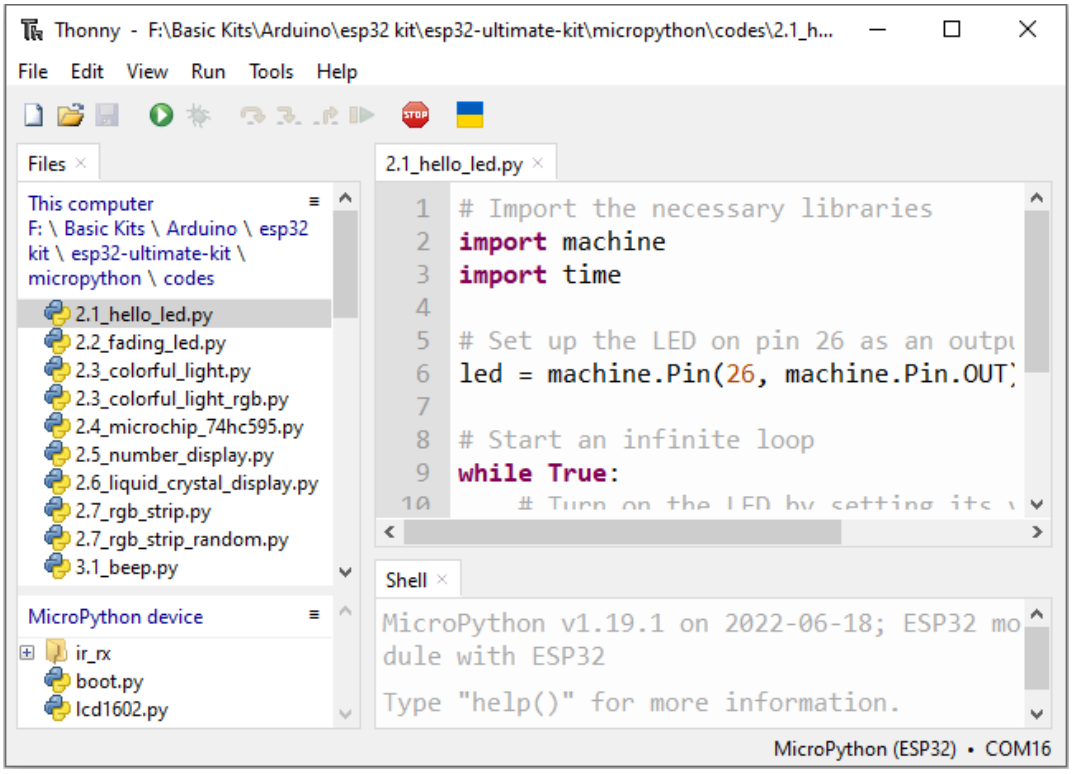
La sección de código en los proyectos te indica exactamente qué código se utiliza, así que haz doble clic en el archivo `.py` con el número de serie en el camino `esp32-starter-kit-main\micropython\codes\` para abrirlo.

Sin embargo, primero debes descargar el paquete y subir las bibliotecas, como se describe en *1.4 Subir las Bibliotecas (Importante)*.

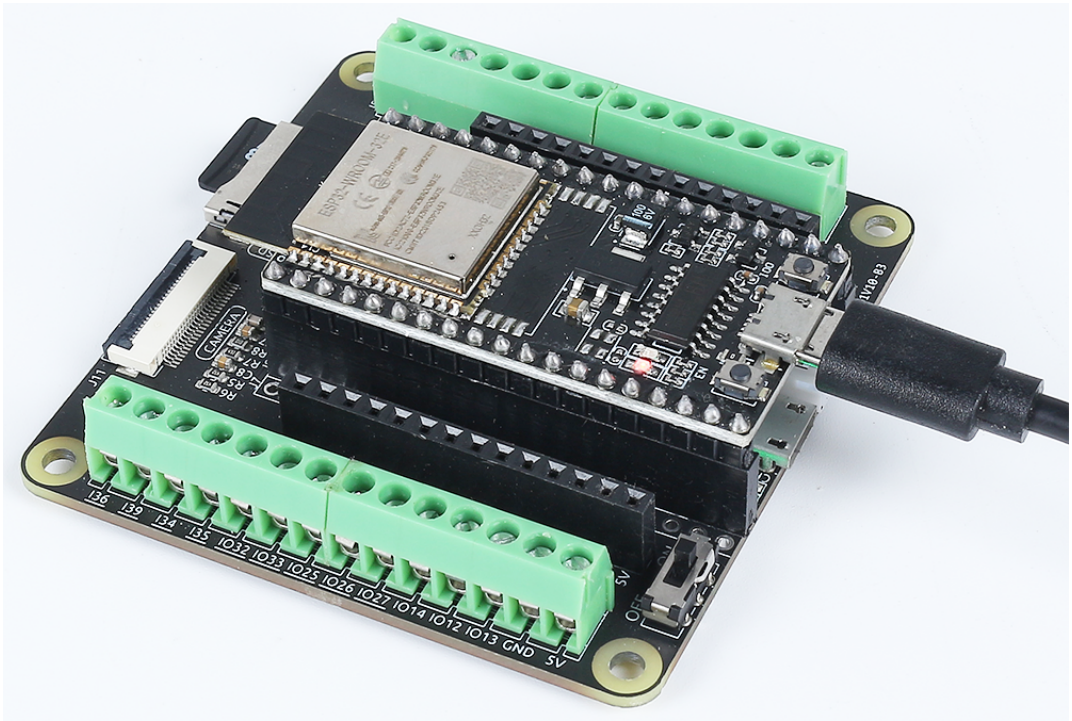
1. Abrir código.

   Por ejemplo, `1.1_hello_led.py`.

   Si haces doble clic en él, se abrirá una nueva ventana a la derecha. Puedes abrir más de un código al mismo tiempo.

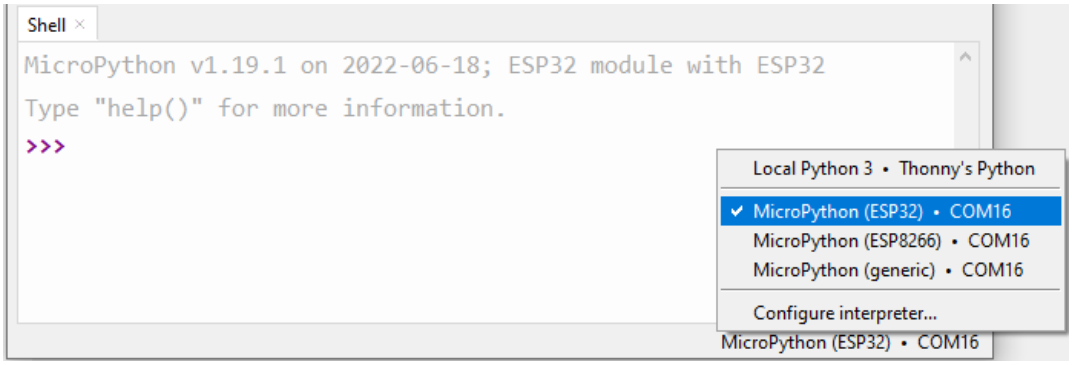


2. Conecta el esp32 a tu computadora con un cable micro USB.



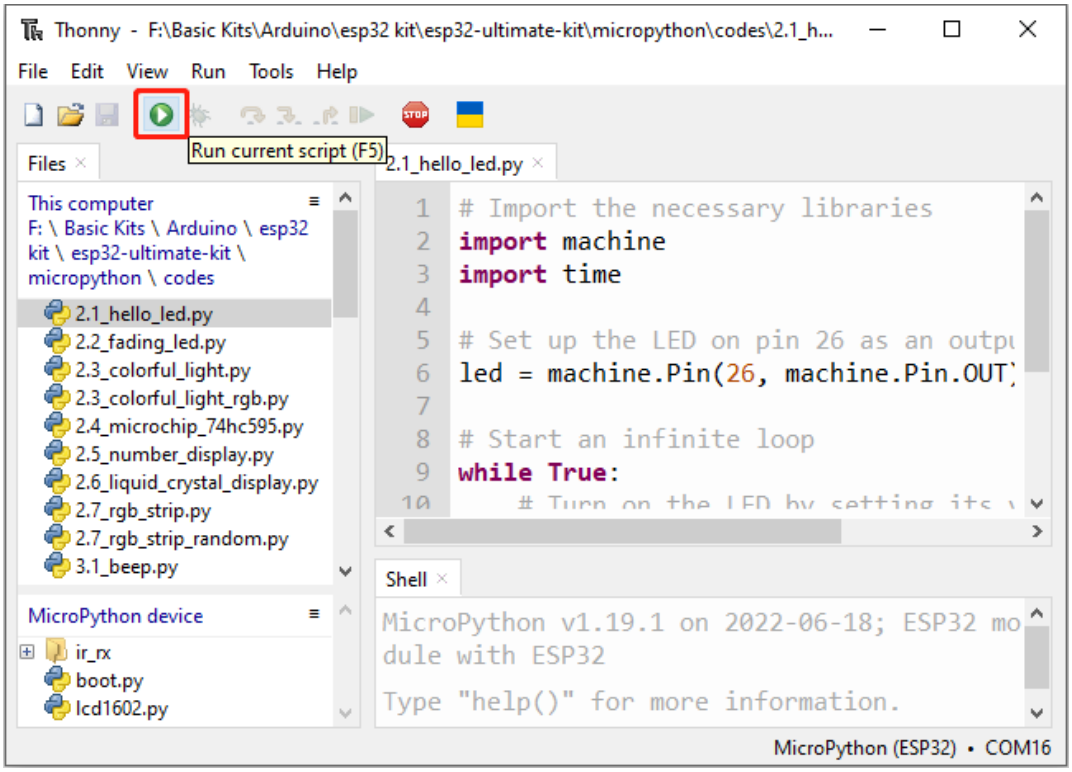
3. Seleccionar el intérprete correcto

   Selecciona el intérprete «MicroPython (ESP32).COMxx».



4. Ejecutar el código

   Para ejecutar el script, haz clic en el botón **Ejecutar script actual** o presiona F5.



   Si el código contiene alguna información que necesite ser impresa, aparecerá en la Shell; de lo contrario, solo aparecerá la siguiente información.

   Haz clic en **Ver** -> **Editar** para abrir la ventana de Shell si no aparece en tu Thonny.

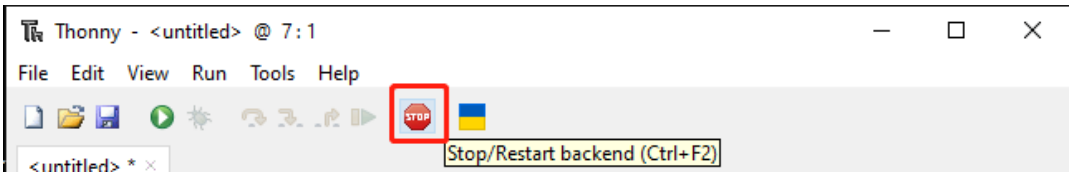
```
MicroPython v1.19.1 el 2022-06-18; módulo ESP32 con ESP32

Escribe "help()" para más información.

>>> %Run -c $EDITOR_CONTENT
```

   - La primera línea muestra la versión de MicroPython, la fecha y la información de tu dispositivo.
   - La segunda línea te invita a ingresar «help()» para obtener ayuda.
   - La tercera línea es un comando de Thonny indicando al intérprete de MicroPython en tu Pico W

ejecutar el contenido del área del script - «EDITOR_CONTENT».

- Si hay algún mensaje después de la tercera línea, usualmente es un mensaje que le indicas a MicroPython imprimir, o un mensaje de error para el código.

5. Detener la ejecución

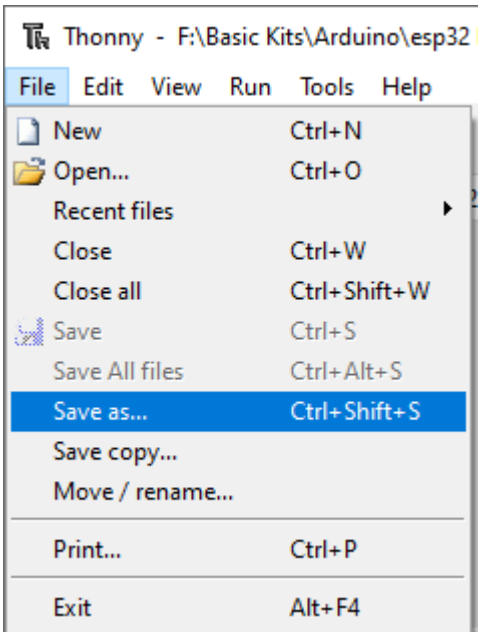


Para detener el código en ejecución, haz clic en el botón **Detener/Reiniciar backend**. El comando **%RUN -c $EDITOR_CONTENT** desaparecerá después de detener.

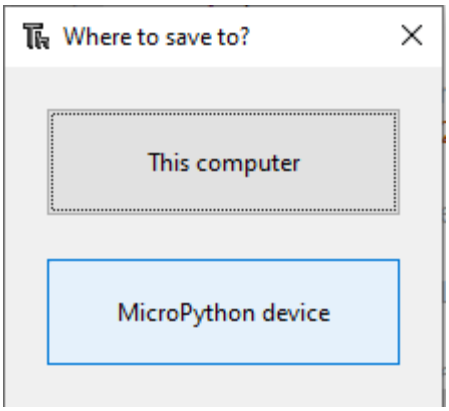
6. Guardar o guardar como

Puedes guardar los cambios realizados en el ejemplo abierto presionando **Ctrl+S** o haciendo clic en el botón **Guardar** en Thonny.

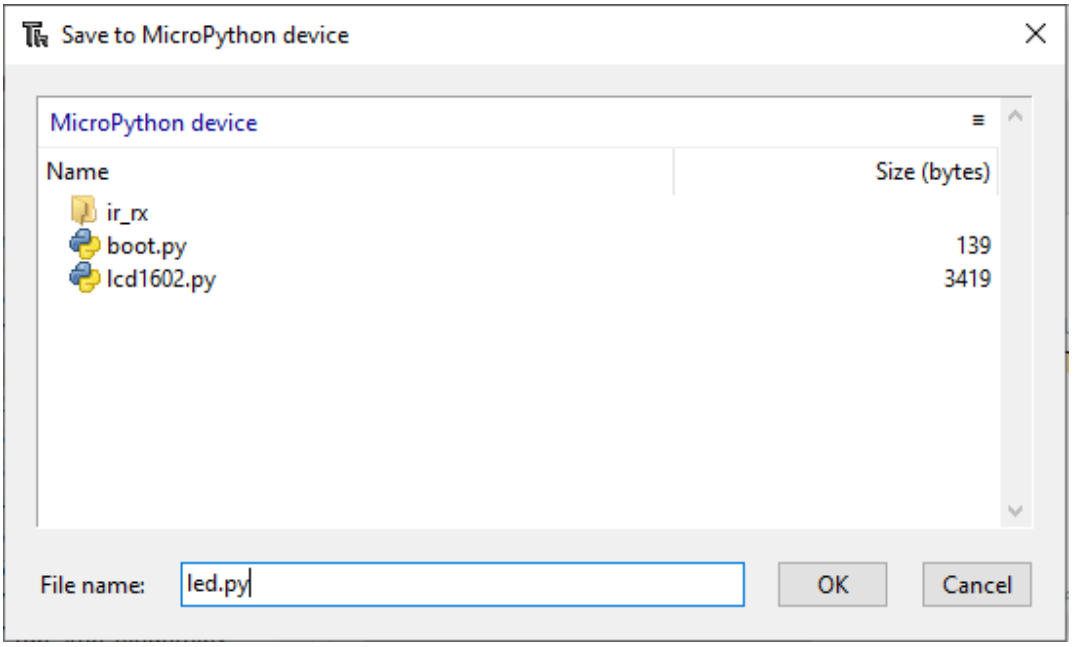
El código se puede guardar como un archivo separado dentro del **disco MicroPython(ESP32)** haciendo clic en **Archivo** -> **Guardar como**.



Selecciona **disco MicroPython**.



Luego haz clic en **Aceptar** después de ingresar el nombre del archivo y la extensión **.py**. En el disco MicroPython, verás tu archivo guardado.



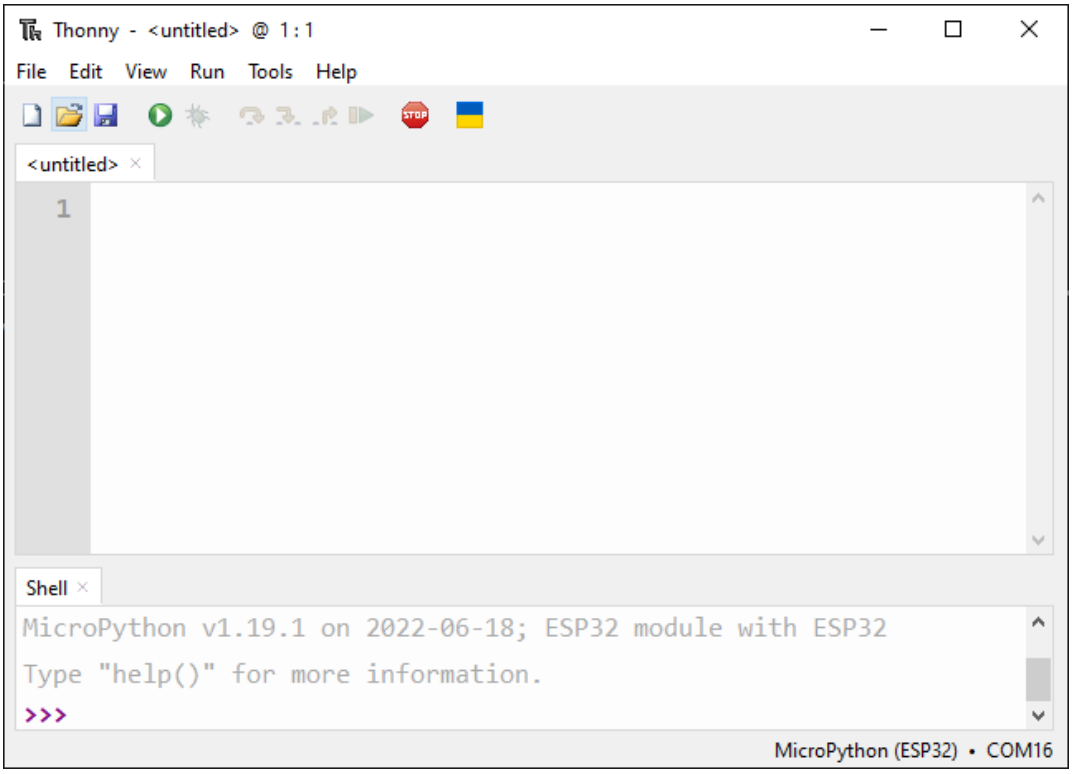
Nota: Independientemente del nombre que le des a tu código, es mejor describir qué tipo de código es, y no darle un nombre sin sentido como `abc.py`. Cuando guardas el código como `main.py`, se ejecutará automáticamente cuando se encienda la alimentación.

4.5.2 Crear Archivo y Ejecutarlo

El código se muestra directamente en la sección de código. Puedes copiarlo en Thonny y ejecutarlo de la siguiente manera.

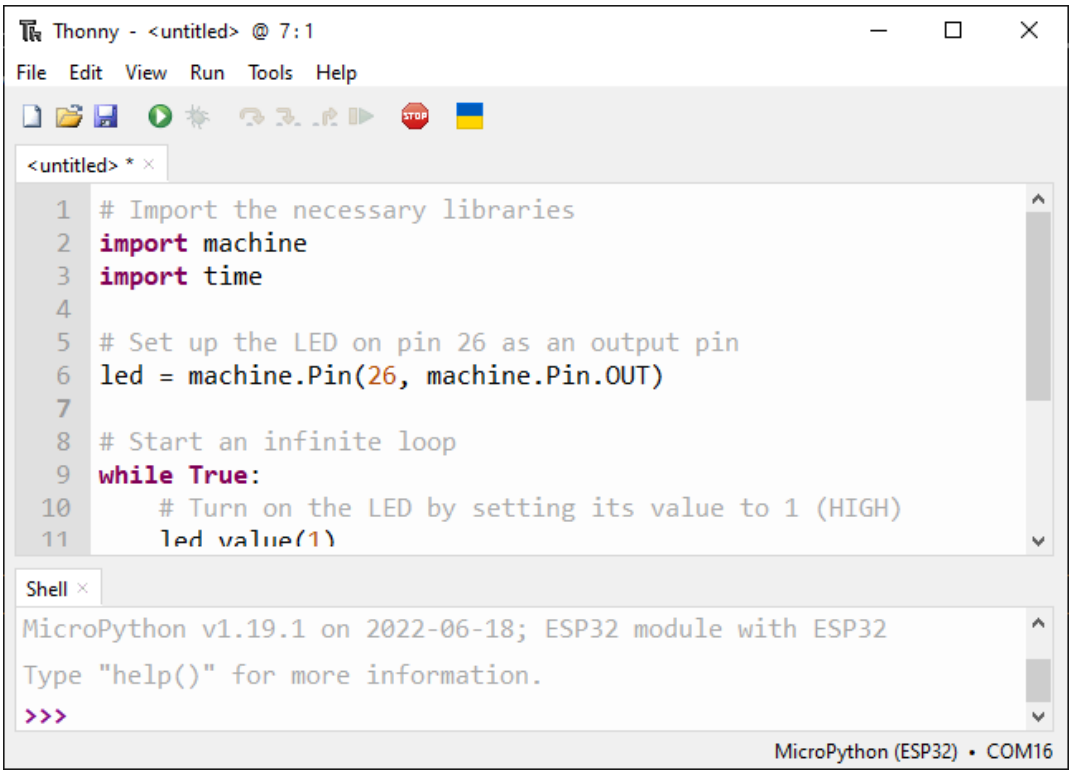
1. Crear un nuevo archivo

   Abre Thonny IDE, haz clic en el botón **Nuevo** para crear un nuevo archivo en blanco.

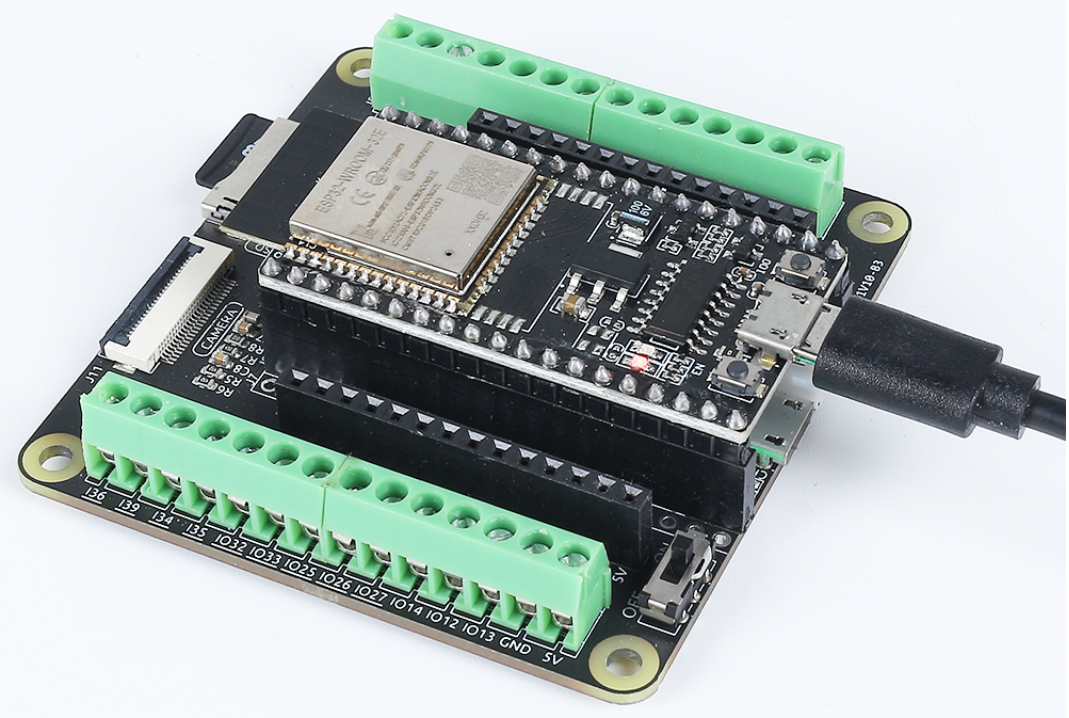


2. Copiar código

   Copia el código del proyecto al IDE de Thonny.

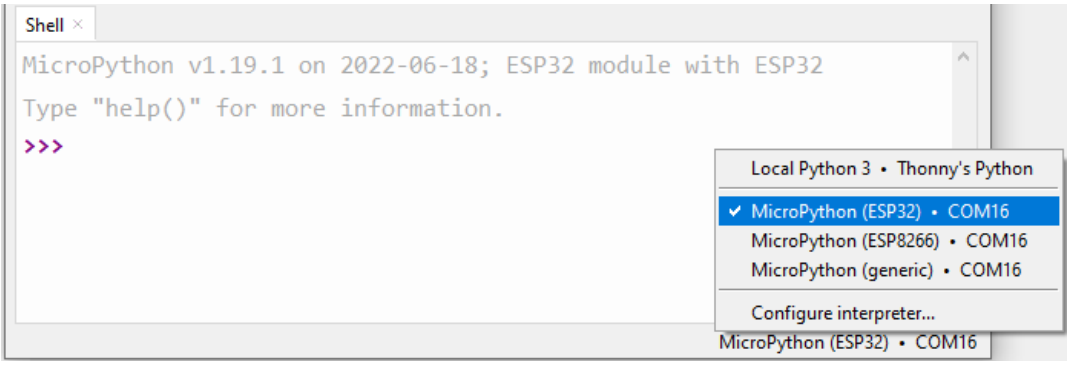


3. Conecta el esp32 a tu computadora con un cable micro USB.



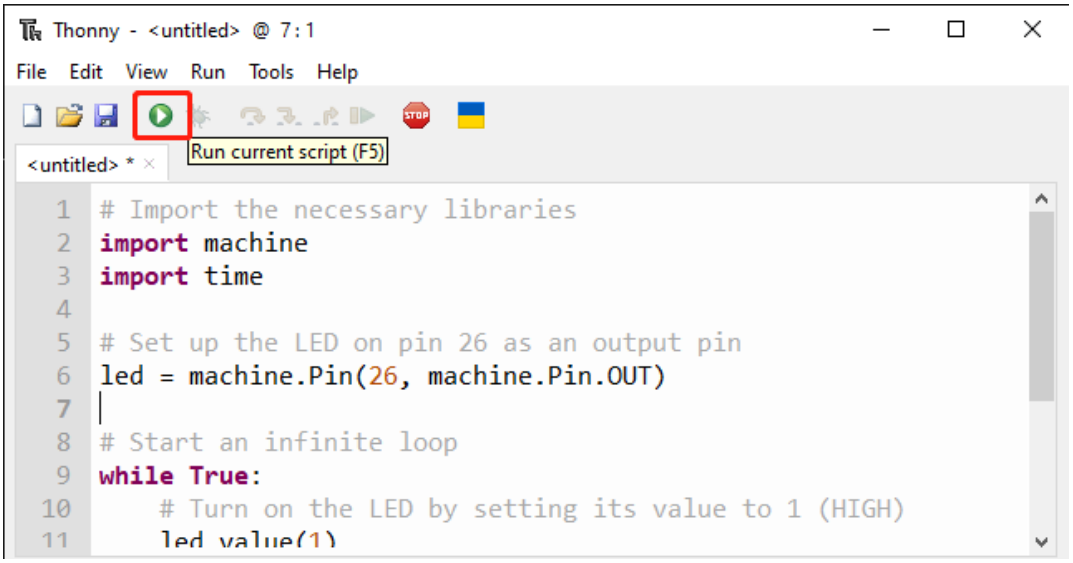
4. Seleccionar el intérprete correcto

   Selecciona el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.



5. Ejecutar el código

   Necesitas hacer clic en **Ejecutar Script Actual** o simplemente presionar `F5` para ejecutarlo.



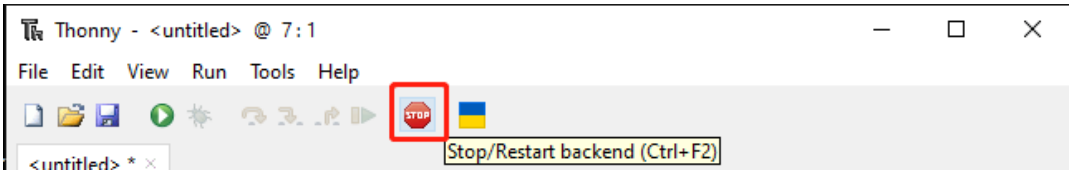
Si el código contiene alguna información que necesite ser impresa, aparecerá en la Shell; de lo contrario, solo aparecerá la siguiente información.

Haz clic en **Ver** -> **Editar** para abrir la ventana de Shell si no aparece en tu Thonny.

```
MicroPython v1.19.1 el 2022-06-18; módulo ESP32 con ESP32

Escribe "help()" para más información.

>>> %Run -c $EDITOR_CONTENT
```

- La primera línea muestra la versión de MicroPython, la fecha y la información de tu dispositivo.
- La segunda línea te invita a ingresar «help()» para obtener ayuda.
- La tercera línea es un comando de Thonny indicando al intérprete de MicroPython en tu Pico W ejecutar el contenido del área del script - «EDITOR_CONTENT».
- Si hay algún mensaje después de la tercera línea, usualmente es un mensaje que le indicas a MicroPython imprimir, o un mensaje de error para el código.

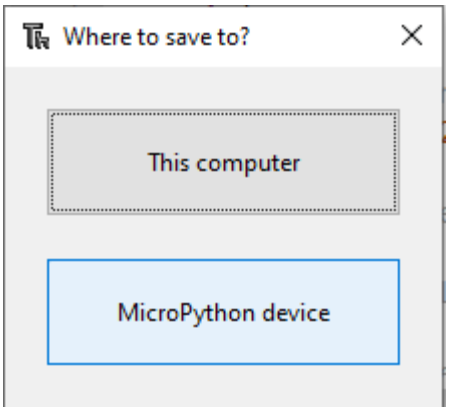
6. Detener la ejecución



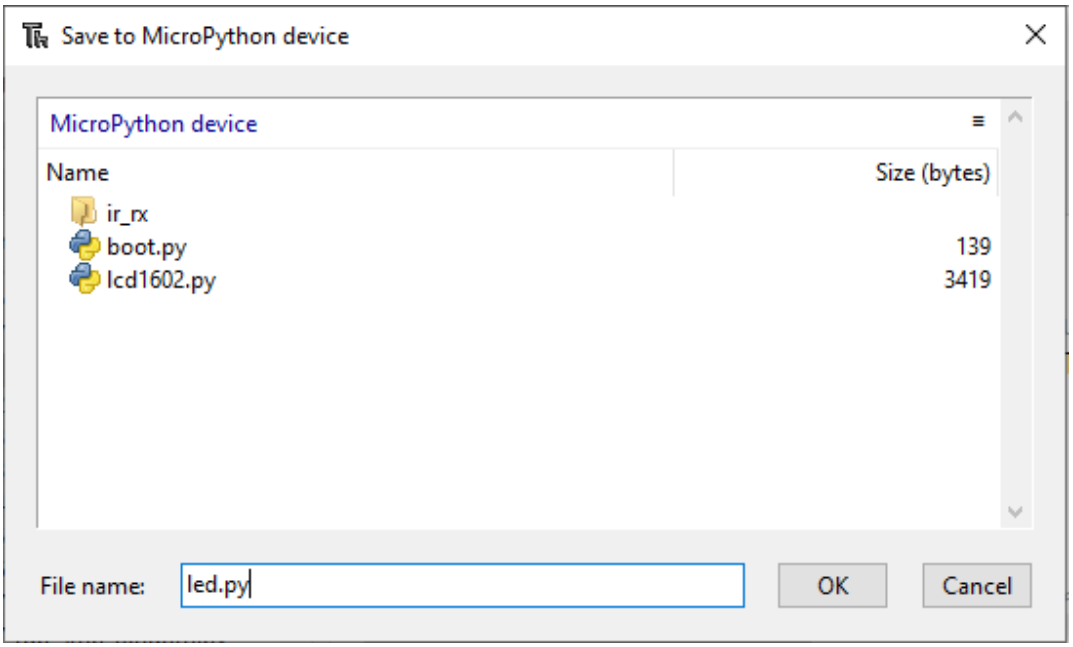
Para detener el código en ejecución, haz clic en el botón **Detener/Reiniciar backend**. El comando **%RUN -c $EDITOR_CONTENT** desaparecerá después de detener.

7. Guardar o guardar como

Puedes guardar el código presionando **Ctrl+S** o haciendo clic en el botón **Guardar** en Thonny. En la ventana emergente, selecciona el lugar donde quieres guardar el archivo.



Luego haz clic en **Aceptar** o **Guardar** después de ingresar el nombre del archivo y la extensión **.py**.



Nota: Independientemente del nombre que le des a tu código, es mejor describir qué tipo de código es, y no darle un nombre sin sentido como `abc.py`. Cuando guardas el código como `main.py`, se ejecutará automáticamente cuando se encienda la alimentación.

---

8. Abrir archivo

   Aquí hay dos maneras de abrir un archivo de código guardado.

   - El primer método es hacer clic en el icono de abrir en la barra de herramientas de Thonny, igual que cuando guardas un programa, se te preguntará si quieres abrirlo desde **este computador** o **dispositivo MicroPython**, por ejemplo, haz clic en **dispositivo MicroPython** y verás una lista de todos los programas que has guardado en el ESP32.
   - El segundo es abrir la vista previa del archivo directamente haciendo clic en **Ver** -> **Archivos** -> y luego hacer doble clic en el correspondiente archivo `.py` para abrirlo.

4.6 1.6 (Opcional) Sintaxis Básica de MicroPython

4.6.1 Indentación

La indentación se refiere a los espacios al principio de una línea de código. Al igual que los programas estándar de Python, los programas de MicroPython generalmente se ejecutan de arriba hacia abajo: Recorre cada línea por turno, la ejecuta en el intérprete y luego continúa con la siguiente línea, Justo como si los escribieras línea por línea en el Shell. Sin embargo, un programa que simplemente recorre la lista de instrucciones línea por línea no es muy inteligente, por lo que MicroPython, al igual que Python, tiene su propio método para controlar la secuencia de ejecución de su programa: la indentación.

Debes poner al menos un espacio antes de print(), de lo contrario aparecerá un mensaje de error «Sintaxis inválida». Generalmente se recomienda estandarizar los espacios presionando uniformemente la tecla Tab.

```
if 8 > 5:
print("¡Ocho es mayor que Cinco!")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

Debes usar el mismo número de espacios en el mismo bloque de código, o Python te mostrará un error.

```
if 8 > 5:
print("¡Ocho es mayor que Cinco!")
        print("Ocho es mayor que Cinco")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

4.6.2 Comentarios

Los comentarios en el código nos ayudan a entender el código, hacen que todo el código sea más legible y comentan parte del código durante las pruebas, de modo que esta parte del código no se ejecute.

Comentario de Una Línea

Los comentarios de una línea en MicroPython comienzan con #, y el texto siguiente se considera un comentario hasta el final de la línea. Los comentarios se pueden colocar antes o después del código.

```
print("hello world") #This is a annotationhello world
```

```
>>> %Run -c $EDITOR_CONTENT
hello world
```

Los comentarios no son necesariamente texto usado para explicar el código. También puedes comentar parte del código para evitar que micropython ejecute el código.

```
#print("Can't run it")
print("hello world") #This is a annotationhello world
```

```
>>> %Run -c $EDITOR_CONTENT
hello world
```

Comentario Multilínea

Si deseas comentar en múltiples líneas, puedes usar múltiples signos #.

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello, World!
```

O, puedes usar cadenas de múltiples líneas en lugar de lo esperado.

Dado que MicroPython ignora las literales de cadena que no se asignan a variables, puedes agregar múltiples líneas de cadenas (comillas triples) al código y poner comentarios en ellas:

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello, World!
```

Mientras la cadena no se asigne a una variable, MicroPython la ignorará después de leer el código y la tratará como si hubieras hecho un comentario multilínea.

4.6.3 Print()

La función `print()` imprime el mensaje especificado en la pantalla o en otro dispositivo de salida estándar. El mensaje puede ser una cadena de texto o cualquier otro objeto; el objeto se convertirá en una cadena antes de ser escrito en la pantalla.

Imprimir múltiples objetos:

```
print("Welcome!", "Enjoy yourself!")
```

```
>>> %Run -c $EDITOR_CONTENT
Welcome! Enjoy yourself!
```

Imprimir tuplas:

```
x = ("pear", "apple", "grape")
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT
('pear', 'apple', 'grape')
```

Imprimir dos mensajes y especificar el separador:

```
print("Hello", "how are you?", sep="---")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello---how are you?
```

4.6.4 Variables

Las variables son contenedores utilizados para almacenar valores de datos.

Crear una variable es muy sencillo. Solo necesitas nombrarla y asignarle un valor. No es necesario especificar el tipo de dato de la variable al asignarla, porque la variable es una referencia y accede a objetos de diferentes tipos de datos mediante la asignación.

El nombramiento de variables debe seguir las siguientes reglas:

- Los nombres de variables solo pueden contener números, letras y guiones bajos
- El primer carácter del nombre de la variable debe ser una letra o un guion bajo
- Los nombres de las variables distinguen entre mayúsculas y minúsculas

Crear Variable

No hay un comando para declarar variables en MicroPython. Las variables se crean cuando se les asigna un valor por primera vez. No es necesario utilizar ningún tipo de declaración específica, e incluso puedes cambiar el tipo después de establecer la variable.

```
x = 8       # x is of type int
x = "lily" # x is now of type str
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT
lily
```

Casting

Si deseas especificar el tipo de dato para la variable, puedes hacerlo mediante el casting.

```
x = int(5)    # y will be 5
y = str(5)    # x will be '5'
z = float(5)  # z will be 5.0
print(x,y,z)
```

```
>>> %Run -c $EDITOR_CONTENT
5 5 5.0
```

Obtener el Tipo

Puedes obtener el tipo de dato de una variable con la función `type()`.

```
x = 5
y = "hello"
z = 5.0
print(type(x),type(y),type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'int'> <class 'str'> <class 'float'>
```

¿Comillas simples o dobles?

En MicroPython, se pueden usar comillas simples o dobles para definir variables de cadena.

```
x = "hello"
# is the same as
x = 'hello'
```

Sensibilidad a Mayúsculas y Minúsculas

Los nombres de las variables son sensibles a mayúsculas y minúsculas.

```
a = 5
A = "lily"
#A will not overwrite a
print(a, A)
```

```
>>> %Run -c $EDITOR_CONTENT
5 lily
```

4.6.5 If Else

La toma de decisiones es necesaria cuando queremos ejecutar un código solo si se cumple una determinada condición.

if

```
if test expression:
    statement(s)
```

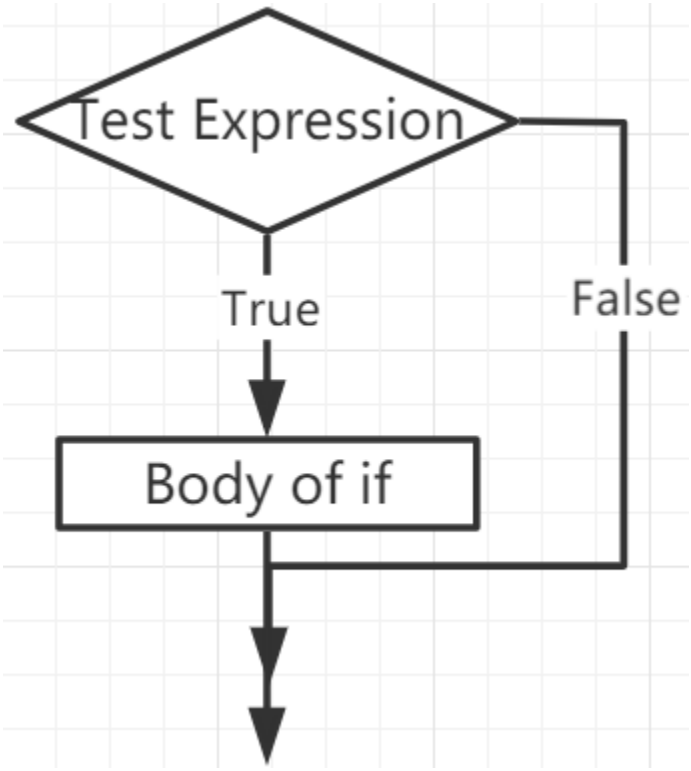
Aquí, el programa evalúa la `test expression` y ejecuta la `statement` solo cuando la `test expression` es Verdadera.

Si la `test expression` es Falsa, entonces la(s) `statement(s)` no será(n) ejecutada(s).

En MicroPython, la indentación indica el cuerpo de la declaración `if`. El cuerpo comienza con una indentación y termina con la primera línea sin indentar.

Python interpreta los valores no cero como `True`. None y 0 se interpretan como `False`.

Flujograma de la Declaración if



Ejemplo

```
num = 8
if num > 0:
    print(num, "is a positive number.")
print("End with this line")
```

```
>>> %Run -c $EDITOR_CONTENT
8 is a positive number.
End with this line
```

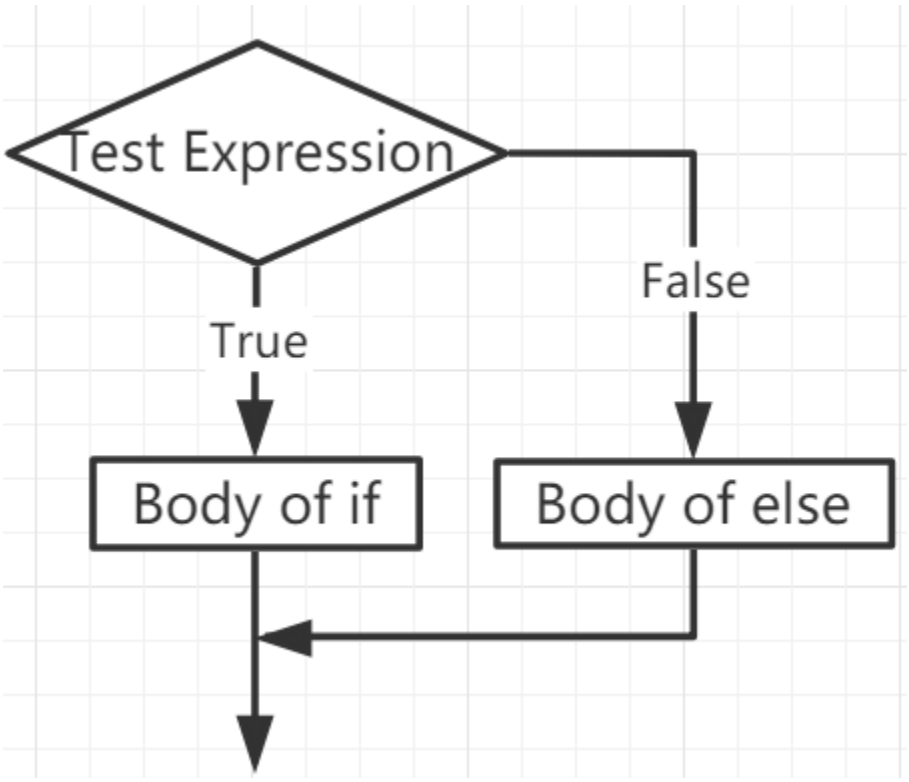
if...else

```
if test expression:
    Body of if
else:
    Body of else
```

La declaración `if..else` evalúa la `test expression` y ejecutará el cuerpo de `if` solo cuando la condición de prueba sea `True`.

Si la condición es `False`, se ejecuta el cuerpo de `else`. Se utiliza la indentación para separar los bloques.

Flujograma de la Declaración if...else



Ejemplo

```
num = -8
if num > 0:
    print(num, "is a positive number.")
else:
    print(num, "is a negative number.")
```

```
>>> %Run -c $EDITOR_CONTENT
-8 is a negative number.
```

if...elif...else

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

`Elif` es la abreviatura de `else if`. Nos permite verificar múltiples expresiones.

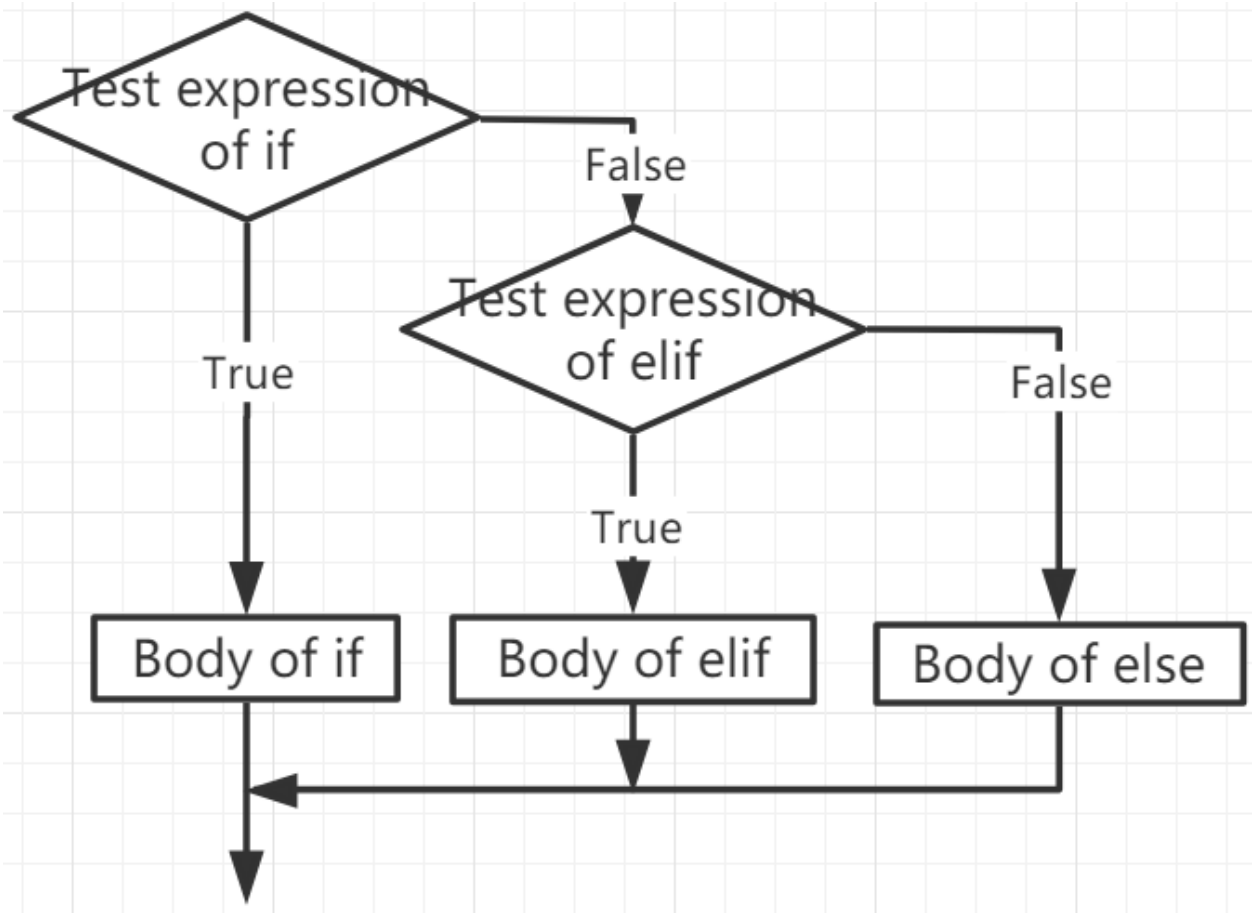
Si la condición del `if` es Falsa, se verifica la condición del siguiente bloque elif, y así sucesivamente.

Si todas las condiciones son `False`, se ejecuta el cuerpo de `else`.

Solo uno de varios bloques `if...elif...else` se ejecuta según las condiciones.

El bloque `if` solo puede tener un bloque `else`. Pero puede tener múltiples bloques `elif`.

Flujograma de la Declaración if…elif…else



Ejemplo

```
x = 10
y = 9

if x > y:
```

(continué en la próxima página)

(proviene de la página anterior)

```
    print("x is greater than y")
elif x == y:
    print("x and y are equal")
else:
    print("x is greater than y")
```

```
>>> %Run -c $EDITOR_CONTENT
x is greater than y
```

Nested if

Podemos incrustar una declaración if dentro de otra declaración if, a lo que se llama una declaración if anidada.

Ejemplo

```
x = 67

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

```
>>> %Run -c $EDITOR_CONTENT
Above ten,
and also above 20!
```

4.6.6 Bucles While

La instrucción `while` se utiliza para ejecutar un programa en un bucle, es decir, para ejecutar un programa en bucle bajo ciertas condiciones para manejar la misma tarea que necesita ser procesada repetidamente.

Su forma básica es:

```
while test expression:
    Body of while
```

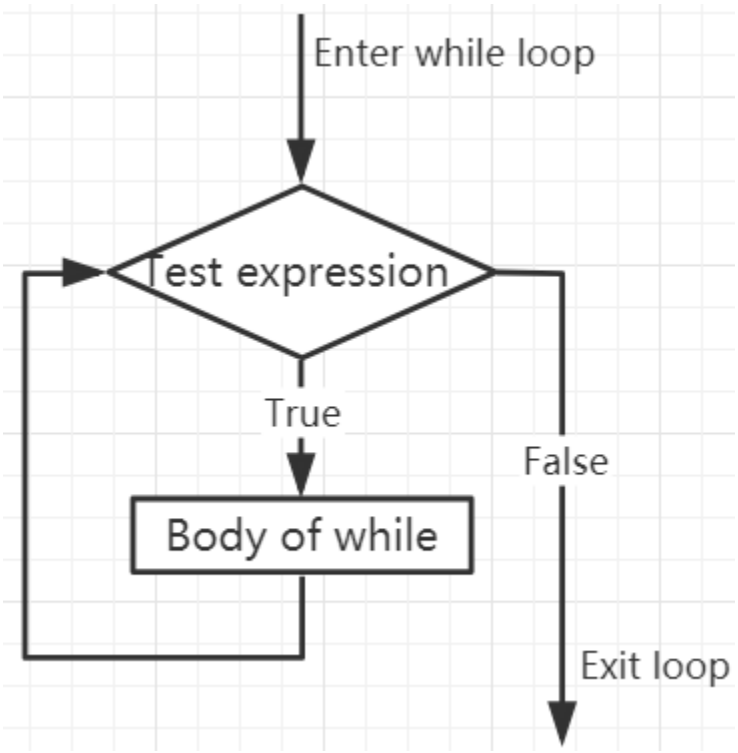
En el bucle `while`, primero se verifica la `test expression`. Solo cuando la `test expression` se evalúa como `True`, se entra en el cuerpo del while. Después de una iteración, se verifica nuevamente la `test expression`. Este proceso continúa hasta que la `test expression` se evalúa como `False`.

En MicroPython, el cuerpo del bucle `while` se determina por la indentación.

El cuerpo comienza con una indentación y termina con la primera línea sin indentar.

Python interpreta cualquier valor no cero como `True`. None y 0 se interpretan como `False`.

Diagrama de flujo del bucle while




```
x = 10

while x > 0:
    print(x)
    x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
5
4
3
2
1
```

Instrucción Break

Con la instrucción break podemos detener el bucle incluso si la condición del while es verdadera:

```
x = 10

while x > 0:
    print(x)
    if x == 6:
```

(continué en la próxima página)

(proviene de la página anterior)

```
        break
    x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
```

Bucle While con Else

Al igual que el bucle `if`, el bucle `while` también puede tener un bloque `else` opcional.

Si la condición en el bucle `while` se evalúa como `False`, se ejecuta la parte `else`.

```
x = 10

while x > 0:
    print(x)
    x -= 1
else:
    print("Game Over")
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
5
4
3
2
1
Game Over
```

4.6.7 Bucles For

El bucle `for` puede recorrer cualquier secuencia de elementos, como una lista o una cadena.

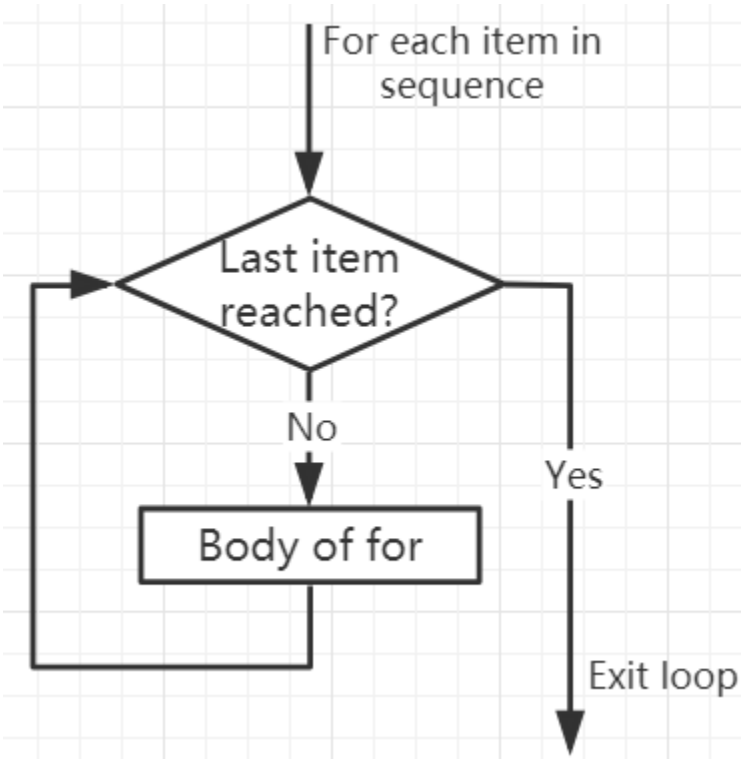
El formato de sintaxis del bucle for es el siguiente:

```
for val in secuencia:
    Cuerpo del for
```

Aquí, `val` es una variable que obtiene el valor del elemento en la secuencia en cada iteración.

El bucle continúa hasta que alcanzamos el último elemento en la secuencia. Usa la indentación para separar el cuerpo del bucle `for` del resto del código.

Diagrama de flujo del bucle For




```
numbers = [1, 2, 3, 4]
sum = 0

for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

```
>>> %Run -c $EDITOR_CONTENT
The sum is 10
```

La sentencia break

Con la sentencia break podemos detener el bucle antes de que haya recorrido todos los elementos:

```
numbers = [1, 2, 3, 4]
sum = 0

for val in numbers:
    sum = sum+val
    if sum == 6:
        break
print("The sum is", sum)
```

```
>>> %Run -c $EDITOR_CONTENT
The sum is 6
```

La sentencia continue

Con la sentencia `continue` podemos detener la iteración actual del bucle y continuar con la siguiente:

```
numbers = [1, 2, 3, 4]

for val in numbers:
    if val == 3:
        continue
    print(val)
```

```
>>> %Run -c $EDITOR_CONTENT
1
2
4
```

La función `range()`

Podemos usar la función `range()` para generar una secuencia de números. `range(6)` producirá números entre 0 y 5 (6 números).

También podemos definir inicio, parada y tamaño de paso como `range(start, stop, step_size)`. Si no se proporciona, el tamaño_de_paso por defecto es 1.

En un sentido de range, el objeto es «perezoso» porque cuando creamos el objeto, no genera cada número que «contiene». Sin embargo, esto no es un iterador porque soporta operaciones in, len y `__getitem__`.

Esta función no almacenará todos los valores en la memoria; sería ineficiente. Por lo tanto, recordará el inicio, la parada, el tamaño de paso y generará el siguiente número durante el recorrido.

Para forzar a esta función a mostrar todos los elementos, podemos usar la función `list()`.

```
print(range(6))

print(list(range(6)))

print(list(range(2, 6)))

print(list(range(2, 10, 2)))
```

```
>>> %Run -c $EDITOR_CONTENT
range(0, 6)
[0, 1, 2, 3, 4, 5]
[2, 3, 4, 5]
[2, 4, 6, 8]
```

Podemos usar `range()` en un bucle `for` para iterar sobre una secuencia de números. Se puede combinar con la función `len()` para usar el índice para recorrer la secuencia.

```
fruits = ['pear', 'apple', 'grape']

for i in range(len(fruits)):
    print("I like", fruits[i])
```

```
>>> %Run -c $EDITOR_CONTENT
I like pear
I like apple
I like grape
```

Else in For Loop

El bucle `for` también puede tener un bloque `else` opcional. Si los elementos en la secuencia utilizada para el bucle se agotan, se ejecuta la parte `else`.

La palabra clave `break` puede usarse para detener el bucle `for`. En este caso, se ignorará la parte `else`.

Por lo tanto, si no ocurre ninguna interrupción, la parte `else` del bucle `for` se ejecutará.

```
for val in range(5):
    print(val)
else:
    print("Finished")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
2
3
4
Finished
```

El bloque else NO se ejecutará si el bucle se detiene mediante una sentencia break.

```
for val in range(5):
    if val == 2: break
    print(val)
else:
    print("Finished")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
```

4.6.8 Funciones

En MicroPython, una función es un grupo de declaraciones relacionadas que realizan una tarea específica.

Las funciones ayudan a descomponer nuestro programa en bloques modulares más pequeños. A medida que nuestro proyecto se hace más grande, las funciones lo hacen más organizado y manejable.

Además, evitan la duplicación y hacen que el código sea reutilizable.

Crear una Función

```
def function_nameparameters)
    """docstring"""
    statement(s)
```

- Una función se define usando la palabra clave `def`.
- Un nombre de función para identificarla de manera única. La nomenclatura de funciones es la misma que la de las variables, y ambas siguen las siguientes reglas.
  - Solo pueden contener números, letras y guiones bajos.
  - El primer carácter debe ser una letra o un guión bajo.
  - Sensible a mayúsculas y minúsculas.
- Parámetros (argumentos) a través de los cuales pasamos valores a la función. Son opcionales.
- El dos puntos (:) marca el final del encabezado de la función.
- Docstring opcional, utilizado para describir la función de la función. Usualmente utilizamos comillas triples para que el docstring pueda extenderse a múltiples líneas.
- Una o más declaraciones válidas de Micropython que conforman el cuerpo de la función. Las declaraciones deben tener el mismo nivel de indentación (generalmente 4 espacios).
- Cada función necesita al menos una declaración, pero si por alguna razón hay una función que no contiene ninguna declaración, por favor, inserte la declaración pass para evitar errores.
- Una declaración `return` opcional para devolver un valor de la función.

Llamar a una Función

Para llamar a una función, añade paréntesis después del nombre de la función.

```
def my_function():
    print("Your first function")

my_function()
```

```
>>> %Run -c $EDITOR_CONTENT
Your first function
```

La declaración return

La declaración return se utiliza para salir de una función y volver al lugar donde fue llamada.

Sintaxis de return

```
return [expression_list]
```

La declaración puede contener una expresión que se evalúa y devuelve un valor. Si no hay una expresión en la declaración, o la declaración `return` en sí misma no existe en la función, la función devolverá un objeto `None`.

```
def my_function():
        print("Your first function")

print(my_function())
```

```
>>> %Run -c $EDITOR_CONTENT
Your first function
None
```

Aquí, None es el valor de retorno, porque la declaración return no se utilizó.

Argumentos

La información puede pasarse a la función como argumentos.

Especifica argumentos entre paréntesis después del nombre de la función. Puedes añadir tantos argumentos como necesites, solo sepáralos con comas.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

Número de Argumentos

Por defecto, una función debe ser llamada con el número correcto de argumentos. Esto significa que si tu función espera 2 parámetros, tienes que llamar a la función con 2 argumentos, ni más ni menos.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

Aquí, la función welcome() tiene 2 parámetros.

Dado que llamamos a esta función con dos argumentos, la función funciona sin problemas sin errores.

Si se llama con un número diferente de argumentos, el intérprete mostrará un mensaje de error.

A continuación, se muestra la llamada a esta función, que contiene uno y ningún argumento y sus respectivos mensajes de error.

```
welcome("Lily")Only one argument
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 1 were given
```

```
welcome()No arguments
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 0 were given
```

Argumentos por Defecto

En MicroPython, podemos usar el operador de asignación (=) para proporcionar un valor por defecto para el parámetro.

Si llamamos a la función sin argumento, utiliza el valor por defecto.

```
def welcome(name, msg = "Welcome to China!"):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)
welcome("Lily")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

En esta función, el parámetro `name` no tiene un valor por defecto y es requerido (obligatorio) durante la llamada.

Por otro lado, el valor por defecto del parámetro `msg` es «¡Bienvenido a China!». Por lo tanto, es opcional durante la llamada. Si se proporciona un valor, sobrescribirá el valor por defecto.

Cualquier número de argumentos en la función puede tener un valor por defecto. Sin embargo, una vez que hay un argumento por defecto, todos los argumentos a su derecha también deben tener valores por defecto.

Esto significa que los argumentos no por defecto no pueden seguir a los argumentos por defecto.

Por ejemplo, si definimos el encabezado de la función anterior como:

```
def welcome(name = "Lily", msg):
```

We will receive the following error message:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
SyntaxError: non-default argument follows default argument
```

Argumentos de Palabra Clave

Cuando llamamos a una función con ciertos valores, estos valores se asignarán a los argumentos basándose en su posición.

Por ejemplo, en la función `welcome()` anterior, cuando la llamamos como bienvenida(«Lily», «¡Bienvenido a China!»), el valor «Lily» se asigna al `nombre` y de manera similar «¡Bienvenido a China!» al parámetro `msg`.

MicroPython permite llamar a funciones con argumentos de palabra clave. Cuando llamamos a la función de esta manera, el orden (posición) de los argumentos puede cambiarse.

```
# keyword arguments
welcome(name = "Lily",msg = "Welcome to China!")

# keyword arguments (out of order)
welcome(msg = "Welcome to China",name = "Lily")

# 1 argumento posicional, 1 argumento de palabra clave
bienvenida("Lily", msg = "¡Bienvenido a China!")
```

Como podemos ver, podemos mezclar argumentos posicionales y argumentos de palabra clave durante las llamadas a funciones. Pero debemos recordar que los argumentos de palabra clave deben venir después de los argumentos posicionales.

Tener un argumento posicional después de un argumento de palabra clave resultará en un error.

Por ejemplo, si la llamada a la función es como sigue:

```
bienvenida(nombre="Lily", "¡Bienvenido a China!")
```

Resultará en un error:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
SyntaxError: arg no-keyword después de arg keyword
```

Argumentos Arbitrarios

A veces, si no sabes el número de argumentos que se pasarán a la función de antemano.

En la definición de la función, podemos añadir un asterisco (*) antes del nombre del parámetro.

```
def welcome(*names):
    """This function welcomes all the person
    in the name tuple"""
    #names is a tuple with arguments
    for name in names:
        print("Welcome to China!", name)

welcome("Lily","John","Wendy")
```

```
>>> %Run -c $EDITOR_CONTENT
Welcome to China! Lily
```

(continué en la próxima página)

(proviene de la página anterior)

```
Welcome to China! John
Welcome to China! Wendy
```

Aquí, hemos llamado a la función con múltiples argumentos. Estos argumentos se empaquetan en una tupla antes de ser pasados a la función.

Dentro de la función, usamos un bucle for para recuperar todos los argumentos.

Recursión

En Python, sabemos que una función puede llamar a otras funciones. Incluso es posible que la función se llame a sí misma. A este tipo de construcciones se les denomina funciones recursivas.

Esto tiene la ventaja de significar que puedes iterar a través de los datos para alcanzar un resultado.

El desarrollador debe ser muy cuidadoso con la recursión ya que puede ser bastante fácil caer en la escritura de una función que nunca termina, o una que usa cantidades excesivas de memoria o potencia del procesador. Sin embargo, cuando se escribe correctamente, la recursión puede ser un enfoque muy eficiente y matemáticamente elegante para la programación.

```
def rec_func(i):
    if(i > 0):
        result = i + rec_func(i - 1)
        print(result)
    else:
        result = 0
    return result

rec_func(6)
```

```
>>> %Run -c $EDITOR_CONTENT
1
3
6
10
15
21
```

En este ejemplo, `rec_func()` es una función que hemos definido para llamarse a sí misma («recursión»). Utilizamos la variable `i` como los datos, y se decrementará (-1) cada vez que recursamos. Cuando la condición no es mayor que 0 (es decir, 0), la recursión termina.

Para los desarrolladores nuevos, puede tomar algún tiempo determinar cómo funciona, y la mejor manera de probarlo es experimentar y modificarlo.

Ventajas de la Recursión

- Las funciones recursivas hacen que el código se vea limpio y elegante.
- Una tarea compleja puede desglosarse en subproblemas más simples usando recursión.
- La generación de secuencias es más fácil con la recursión que usando alguna iteración anidada.

Desventajas de la Recursión

- A veces, la lógica detrás de la recursión es difícil de seguir.

- Las llamadas recursivas son costosas (ineficientes) ya que ocupan mucha memoria y tiempo.
- Las funciones recursivas son difíciles de depurar.

4.6.9 Tipos de Datos

Tipos de Datos Incorporados

MicroPython tiene los siguientes tipos de datos:

- Tipo de Texto: str
- Tipos Numéricos: int, float, complex
- Tipos de Secuencia: list, tuple, range
- Tipo de Mapeo: dict
- Tipos de Conjuntos: set, frozenset
- Tipo Booleano: bool
- Tipos Binarios: bytes, bytearray, memoryview

Obtener el Tipo de Datos

Puedes obtener el tipo de datos de cualquier objeto utilizando la función `type()`:

```
a = 6.8
print(type(a))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'float'>
```

Estableciendo el Tipo de Datos

MicroPython no necesita establecer el tipo de datos específicamente, se determina cuando asignas un valor a la variable.

```
x = "welcome"
y = 45
z = ["apple", "banana", "cherry"]

print(type(x))
print(type(y))
print(type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'str'>
<class 'int'>
<class 'list'>
>>>
```

Estableciendo el Tipo de Datos Específico

Si deseas especificar el tipo de datos, puedes usar las siguientes funciones constructoras:

Ejemplo	Tipo de Dato
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = str(«Hello World»)	str
x = list((«apple», «banana», «cherry»))	list
x = tuple((«apple», «banana», «cherry»))	tuple
x = range(6)	range
x = dict(name=»John», age=36)	dict
x = set((«apple», «banana», «cherry»))	set
x = frozenset((«apple», «banana», «cherry»))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Puedes imprimir algunos de ellos para ver el resultado.

```
a = float(20.5)
b = list(("apple", "banana", "cherry"))
c = bool(5)

print(a)
print(b)
print(c)
```

```
>>> %Run -c $EDITOR_CONTENT
20.5
['apple', 'banana', 'cherry']
True
>>>
```

Conversión de Tipos

Puedes convertir de un tipo a otro con los métodos int(), float() y complex(): La conversión en python se realiza, por lo tanto, usando funciones constructoras:

- int() - construye un número entero a partir de un literal entero, un literal flotante (eliminando todos los decimales) o un literal de cadena (siempre que la cadena represente un número entero)
- float() - construye un número flotante a partir de un literal entero, un literal flotante o un literal de cadena (siempre que la cadena represente un flotante o un entero)
- str() - construye una cadena a partir de una amplia variedad de tipos de datos, incluyendo cadenas, literales enteros y literales flotantes

```
a = float("5")
b = int(3.7)
c = str(6.0)
```

(continué en la próxima página)

(proviene de la página anterior)

```
print(a)
print(b)
print(c)
```

Nota: No puedes convertir números complejos en otro tipo de número.

4.6.10 Operadores

Los operadores se utilizan para realizar operaciones con variables y valores.

- *Operadores Aritméticos*
- *Operadores de Asignación*
- *Operadores de Comparación*
- *Operadores Lógicos*
- *Operadores de Identidad*
- *Operadores de Pertenencia*
- *Operadores a Nivel de Bits*

Operadores Aritméticos

Puedes utilizar los operadores aritméticos para realizar algunas operaciones matemáticas comunes.

Operador	Nombre
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
**	Exponenciación
//	División entera

```
x = 5
y = 3

a = x + y
b = x - y
c = x * y
d = x / y
e = x % y
f = x ** y
g = x // y

print(a)
print(b)
print(c)
print(d)
```

(continué en la próxima página)

(proviene de la página anterior)

```
print(e)
print(f)
print(g)
```

```
>>> %Run -c $EDITOR_CONTENT
8
2
15
1.66667
2
125
1
8
2
15
>>>
```

Operadores de Asignación

Los operadores de asignación se utilizan para asignar valores a variables.

Operador	Ejemplo	Equivalente a
=	a = 6	a =6
+=	a += 6	a = a + 6
-=	a -= 6	a = a - 6
*=	a *= 6	a = a * 6
/=	a /= 6	a = a / 6
%=	a %= 6	a = a % 6
**=	a **= 6	a = a ** 6
//=	a //= 6	a = a // 6
&=	a &= 6	a = a & 6
|=	a |= 6	a = a | 6
^=	a ^= 6	a = a ^ 6
>>=	a >>= 6	a = a >> 6
<<=	a <<= 6	a = a << 6

```
a = 6

a *= 6
print(a)
```

```
>>> %Run test.py
36
>>>
```

Operadores de Comparación

Los operadores de comparación se utilizan para comparar dos valores.

Operador	Nombre
==	Igual
!=	Diferente
<	Menor que
>	Mayor que
>=	Mayor o igual que
<=	Menor o igual que

```
a = 6
b = 8

print(a>b)
```

```
>>> %Run test.py
False
>>>
```

Devuelve **Falso**, porque **a** es menor que **b**.

Operadores Lógicos

Los operadores lógicos se utilizan para combinar declaraciones condicionales.

Operador	Descripción
and	Devuelve Verdadero si ambas declaraciones son verdaderas
or	Devuelve Verdadero si alguna de las declaraciones es verdadera
not	Invierte el resultado, devuelve Falso si el resultado es verdadero

```
a = 6
print(a > 2 and a < 8)
```

```
>>> %Run -c $EDITOR_CONTENT
Verdadero
>>>
```

Operadores de Identidad

Los operadores de identidad se utilizan para comparar los objetos, no si son iguales, sino si son realmente el mismo objeto, con la misma ubicación en la memoria.

Operador	Descripción
is	Devuelve Verdadero si ambas variables son el mismo objeto
is not	Devuelve Verdadero si ambas variables no son el mismo objeto

```
a = ["hola", "bienvenido"]
b = ["hola", "bienvenido"]
c = a

print(a is c)
# devuelve Verdadero porque z es el mismo objeto que x

print(a is b)
# devuelve Falso porque x no es el mismo objeto que y, incluso si tienen el mismo␣
↪contenido

print(a == b)
# devuelve Verdadero porque x es igual a y
```

```
>>> %Run -c $EDITOR_CONTENT
Verdadero
Falso
Verdadero
>>>
```

Operadores de Pertenencia

Los operadores de pertenencia se utilizan para probar si una secuencia está presente en un objeto.

Operador	Descripción
in	Devuelve Verdadero si una secuencia con el valor especificado está presente en el objeto
not in	Devuelve Verdadero si una secuencia con el valor especificado no está presente en el objeto

```
a = ["hola", "bienvenido", "buenos días"]

print("bienvenido" in a)
```

```
>>> %Run -c $EDITOR_CONTENT
Verdadero
>>>
```

Operadores a Nivel de Bits

Los operadores a nivel de bits se utilizan para comparar (binariamente) números.

Operador	Nombre	Descripción
&	AND	Establece cada bit en 1 si ambos bits son 1
|	OR	Establece cada bit en 1 si uno de los dos bits es 1
^	XOR	Establece cada bit en 1 solo si uno de los dos bits es 1
~	NOT	Invierte todos los bits
<<	Desplazamiento a la izquierda con relleno de ceros	Desplaza a la izquierda introduciendo ceros desde la derecha y dejando caer los bits más a la izquierda
>>	Desplazamiento a la derecha con signo	Desplaza a la derecha introduciendo copias del bit más a la izquierda desde la izquierda, y dejando caer los bits más a la derecha

```
num = 2

print(num & 1)
print(num | 1)
print(num << 1)
```

```
>>> %Run -c $EDITOR_CONTENT
0
3
4
>>>
```

4.6.11 Listas

Las listas se utilizan para almacenar múltiples elementos en una única variable y se crean utilizando corchetes:

```
B_list = ["Blossom", "Bubbles","Buttercup"]
print(B_list)
```

Los elementos de la lista son modificables, ordenados y permiten valores duplicados. Los elementos de la lista están indexados, teniendo el primer elemento el índice [0], el segundo elemento el índice [1], y así sucesivamente.

```
C_list = ["Red", "Blue", "Green", "Blue"]
print(C_list)            # duplicate
print(C_list[0])
print(C_list[1])         # ordered
C_list[2] = "Purple"     # changeable
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Blue', 'Green', 'Blue']
Red
Blue
['Red', 'Blue', 'Purple', 'Blue']
```

Una lista puede contener diferentes tipos de datos:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', 255, False, 3.14]
```

Longitud de la Lista

Para determinar cuántos elementos hay en la lista, usa la función len().

```
A_list = ["Banana", 255, False, 3.14]
print(len(A_list))
```

```
>>> %Run -c $EDITOR_CONTENT
4
```

Verificar Elementos de la Lista

Imprimir el segundo elemento de la lista:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[1])
```

```
>>> %Run -c $EDITOR_CONTENT
[255]
```

Imprimir el último elemento de la lista:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[-1])
```

```
>>> %Run -c $EDITOR_CONTENT
[3.14]
```

Imprimir el segundo y tercer elemento:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[1:3])
```

```
>>> %Run -c $EDITOR_CONTENT
[255, False]
```

Cambiar Elementos de la Lista

Cambiar el segundo y tercer elemento:

```
A_list = ["Banana", 255, False, 3.14]
A_list[1:3] = [True,"Orange"]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', True, 'Orange', 3.14]
```

Cambiar el segundo valor reemplazándolo por dos valores:

```
A_list = ["Banana", 255, False, 3.14]
A_list[1:2] = [True,"Orange"]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', True, 'Orange', False, 3.14]
```

Agregar Elementos a la Lista

Utilizando el método append() para añadir un elemento:

```
C_list = ["Red", "Blue", "Green"]
C_list.append("Orange")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Blue', 'Green', 'Orange']
```

Insertar un elemento en la segunda posición:

```
C_list = ["Red", "Blue", "Green"]
C_list.insert(1, "Orange")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Orange', 'Blue', 'Green']
```

Eliminar Elementos de la Lista

El método remove() elimina el elemento especificado.

```
C_list = ["Red", "Blue", "Green"]
C_list.remove("Blue")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Green']
```

El método pop() elimina el índice especificado. Si no especificas el índice, el método pop() elimina el último elemento.

```
A_list = ["Banana", 255, False, 3.14, True,"Orange"]
A_list.pop(1)
print(A_list)
A_list.pop()
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
255
['Banana', False, 3.14, True, 'Orange']
'Orange'
['Banana', False, 3.14, True]
```

La palabra clave `del` también elimina el índice especificado:

```
C_list = ["Red", "Blue", "Green"]
del C_list[1]
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Green']
```

El método clear() vacía la lista. La lista sigue existiendo, pero no tiene contenido.

```
C_list = ["Red", "Blue", "Green"]
C_list.clear()
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
[]
```

2. Pantallas

4.7 2.1 ¡Hola, LED!

Así como imprimir «¡Hola, mundo!» es el primer paso para aprender a programar, usar un programa para controlar un LED es la introducción tradicional para aprender la programación física.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

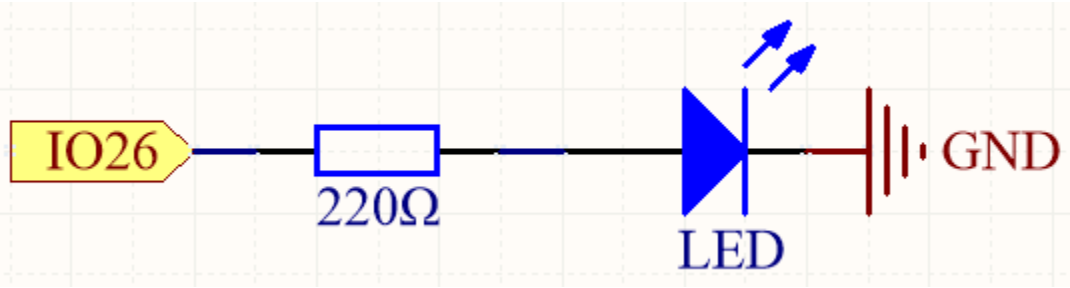
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

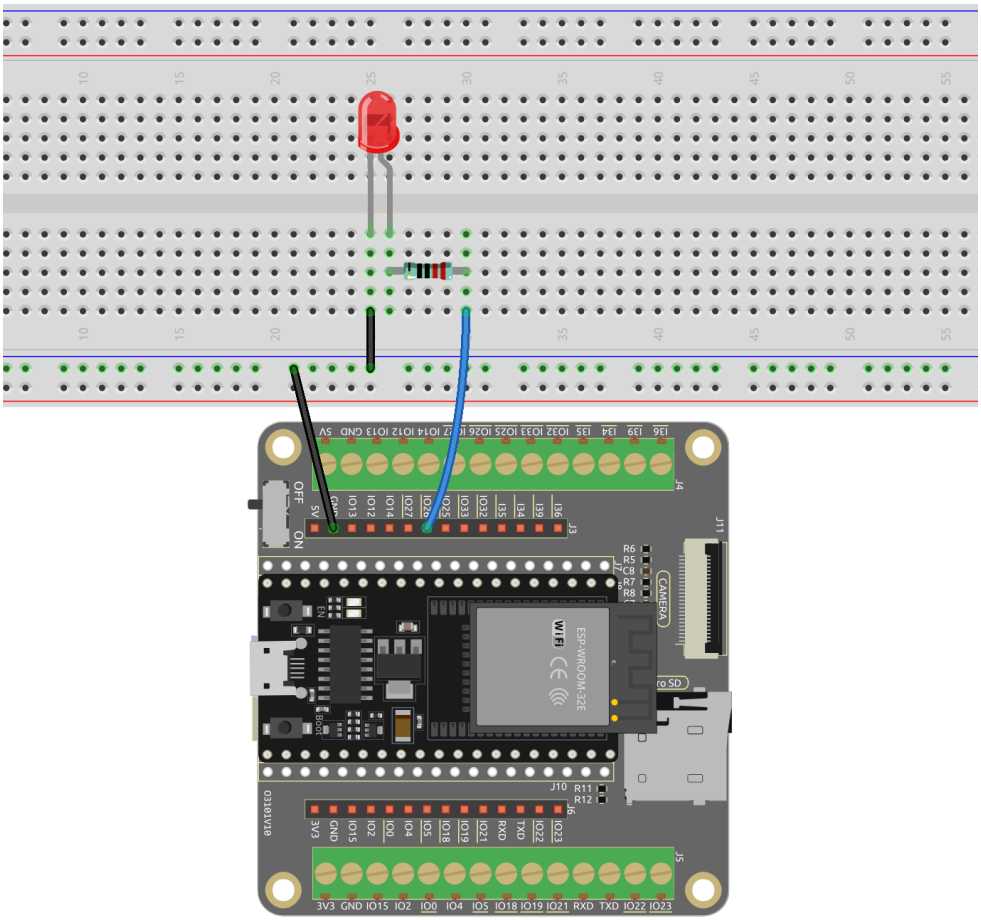
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Este circuito funciona bajo un principio simple, y la dirección de la corriente se muestra en la figura. El LED se iluminará después de la resistencia limitadora de corriente de 220ohm cuando el pin26 emita un nivel alto. El LED se apagará cuando el pin26 emita un nivel bajo.

Conexión

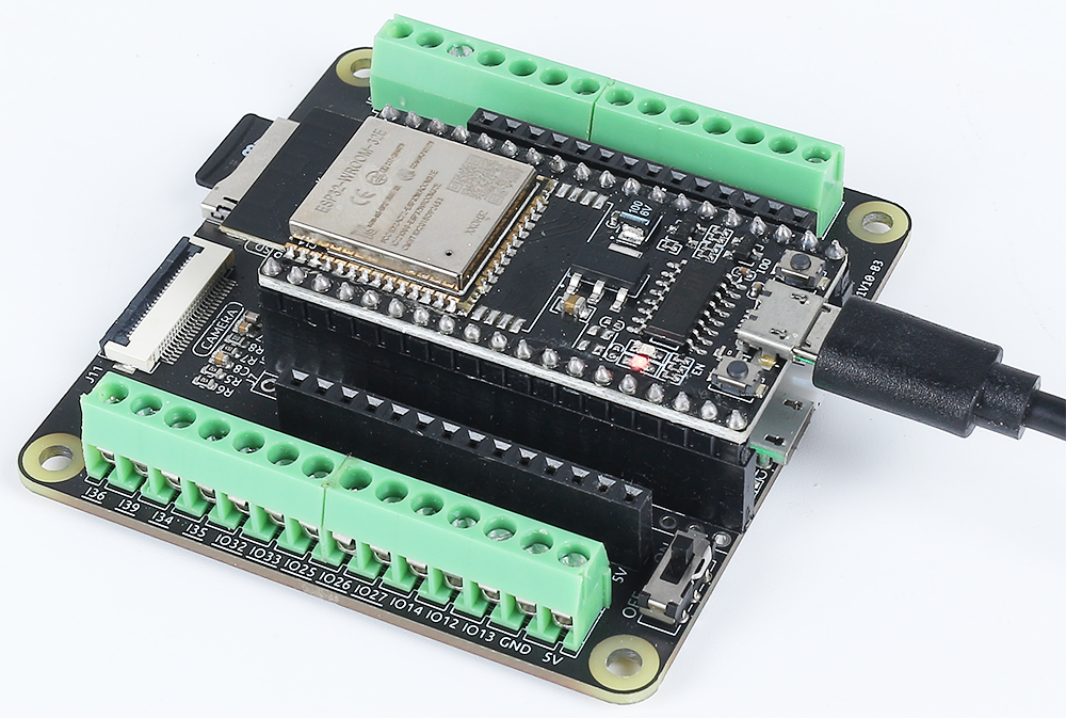


Ejecutar el Código

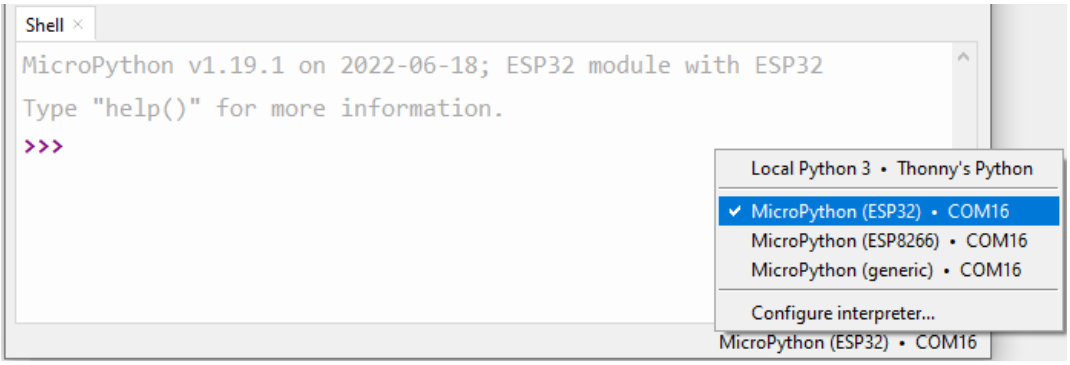
1. Abre el archivo `2.1_hello_led.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny.

```
# Import the necessary libraries
import machine
import time

# Set up the LED on pin 26 as an output pin
led = machine.Pin(26, machine.Pin.OUT)

# Start an infinite loop
while True:
    # Turn on the LED by setting its value to 1 (HIGH)
    led.value(1)
    # Wait for 1 second (1000 milliseconds) while the LED is on
    time.sleep(1)

    # Turn off the LED by setting its value to 0 (LOW)
    led.value(0)
    # Wait for 0.5 seconds (500 milliseconds) while the LED is off
    time.sleep(0.5)
```

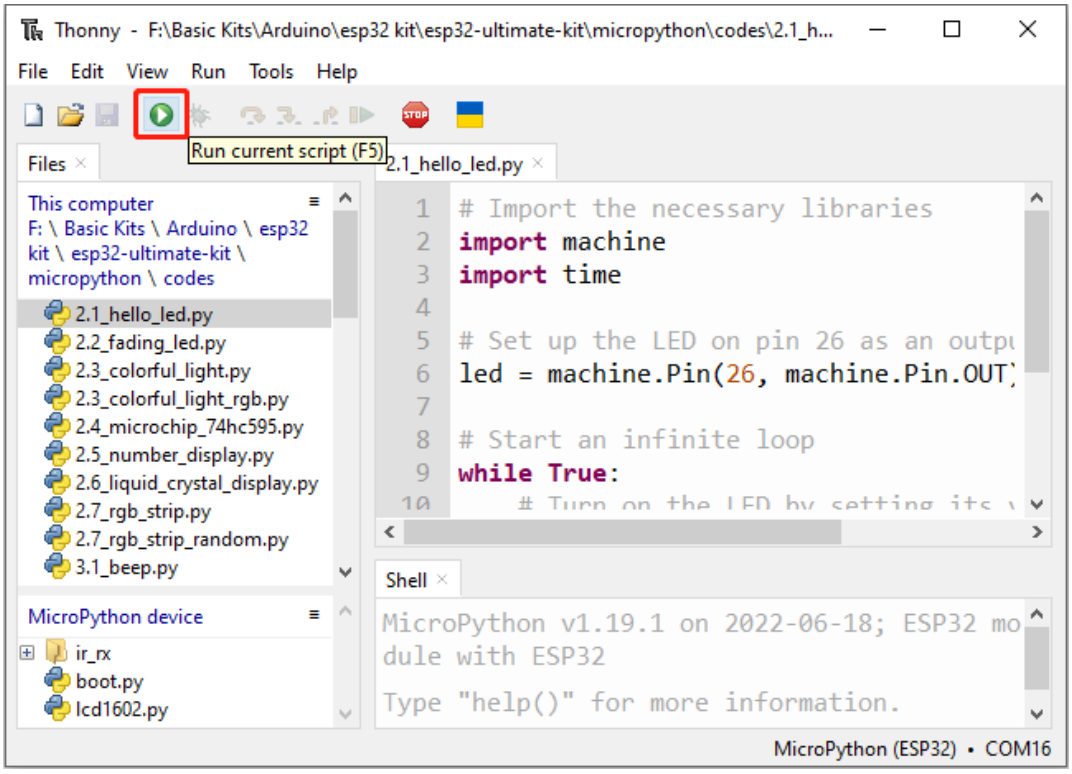
2. Conecta el ESP32 WROOM 32E a tu computadora usando un cable Micro USB.



3. Luego haz clic en el intérprete «MicroPython (ESP32).COMXX» en la esquina inferior derecha.



4. Finalmente, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.



5. Después de ejecutar el código, verás el LED parpadeando.

¿Cómo funciona?

1. Importa dos módulos, `machine` y `time`. El módulo `machine` proporciona acceso a bajo nivel al hardware del microcontrolador, mientras que el módulo `time` ofrece funciones para operaciones relacionadas con el tiempo.

```
import machine
import time
```

2. Luego configura el pin26 como un pin de salida usando la función `machine.Pin()` con el argumento `machine.Pin.OUT`.

```
led = machine.Pin(26, machine.Pin.OUT)
```

3. En el bucle `While True`, el LED se enciende durante un segundo estableciendo el valor del pin26 en 1 usando `led.value(1)` y luego se establece en 0 (`led.value(0)`) para apagarlo durante un segundo, y así sucesivamente en un bucle infinito.

```
while True:
    # Turn on the LED by setting its value to 1 (HIGH)
    led.value(1)
    # Wait for 1 second (1000 milliseconds) while the LED is on
    time.sleep(1)

    # Turn off the LED by setting its value to 0 (LOW)
    led.value(0)
    # Wait for 0.5 seconds (500 milliseconds) while the LED is off
    time.sleep(0.5)
```

Aprende Más

En este proyecto, utilizamos los módulos `machine` y `time` de MicroPython, podemos encontrar más formas de usarlos aquí.

- machine.Pin
- time

4.8 2.2 Atenuación de un LED

En el proyecto anterior, controlamos el LED encendiéndolo y apagándolo mediante salida digital. En este proyecto, crearemos un efecto de respiración en el LED utilizando la Modulación por Ancho de Pulso (PWM). PWM es una técnica que nos permite controlar el brillo de un LED o la velocidad de un motor variando el ciclo de trabajo de una señal de onda cuadrada.

Con PWM, en lugar de simplemente encender o apagar el LED, ajustaremos la cantidad de tiempo que el LED está encendido versus la cantidad de tiempo que está apagado en cada ciclo. Al cambiar rápidamente el LED de encendido a apagado en intervalos variables, podemos crear la ilusión de que el LED se ilumina y se atenúa gradualmente, simulando un efecto de respiración.

Utilizando las capacidades de PWM del ESP32 WROOM 32E, podemos lograr un control suave y preciso sobre el brillo del LED. Este efecto de respiración agrega un elemento dinámico y visualmente atractivo a tus proyectos, creando una exhibición llamativa o ambiente.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

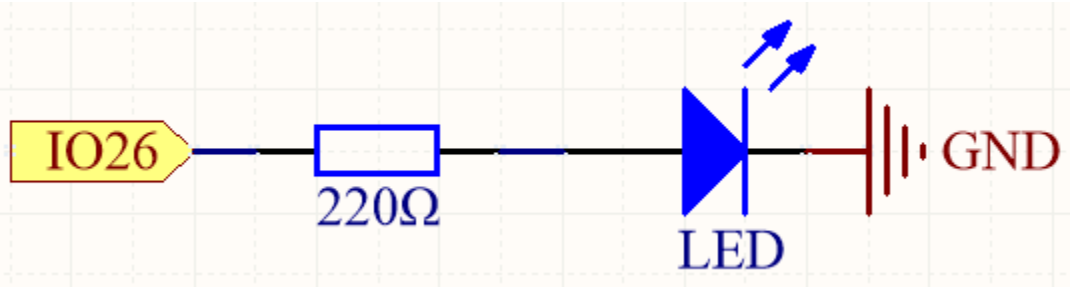
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

Pines Disponibles

Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

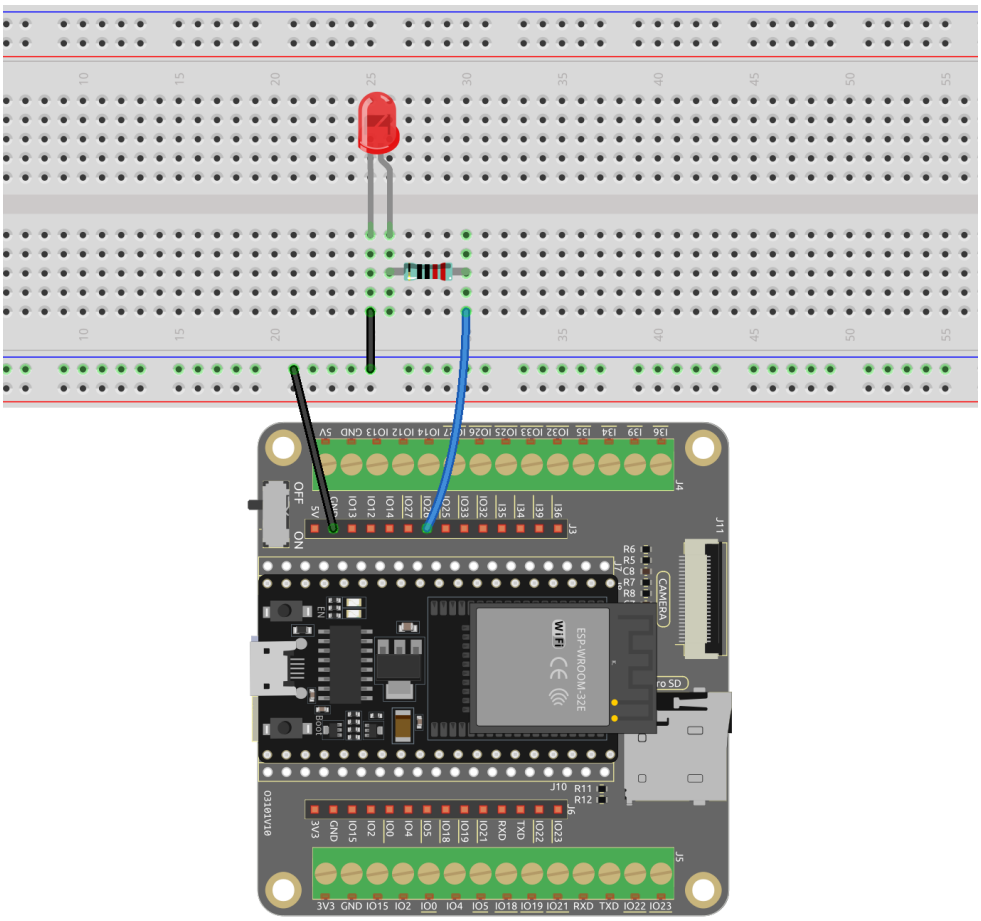
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Este proyecto utiliza el mismo circuito que el primer proyecto *2.1 ¡Hola, LED!*, pero el tipo de señal es diferente. El primer proyecto es para emitir niveles altos y bajos digitales (0&1) directamente desde el pin26 para hacer que el LED se ilumine o se apague, este proyecto es para emitir una señal PWM desde el pin26 para controlar el brillo del LED.

Cableado



Código

Nota:

- Abre el archivo `2.2_fading_led.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
# Import the necessary libraries
from machine import Pin, PWM
import time

# Create a PWM object
led = PWM(Pin(26), freq=1000)

while True:
    # Gradually increase brightness
    for duty_cycle in range(0, 1024, 1):
        led.duty(duty_cycle)
        time.sleep(0.01)

    # Gradually decrease brightness
    for duty_cycle in range(1023, -1, -1):
        led.duty(duty_cycle)
        time.sleep(0.01)
```

The LED will gradually become brighter as the code runs.

¿Cómo funciona?

En general, este código demuestra cómo usar señales PWM para controlar el brillo de un LED.

1. Importa dos módulos, `machine` y `time`. El módulo `machine` proporciona acceso de bajo nivel al hardware del microcontrolador, mientras que el módulo `time` proporciona funciones para operaciones relacionadas con el tiempo.

   ```
   import machine
   import time
   ```

2. Luego inicializa un objeto `PWM` para controlar el LED conectado al pin 26 y establece la frecuencia de la señal PWM a 1000 Hz.

   ```
   led = PWM(Pin(26), freq=1000)
   ```

3. Desvanece el LED de forma gradual usando un bucle: El bucle externo `while True` se ejecuta indefinidamente. Dos bucles `for` anidados se utilizan para aumentar y disminuir gradualmente el brillo del LED. El ciclo de trabajo varía de 0 a 1023, representando un ciclo de trabajo del 0 % al 100 %.

   ```
   # Import the necessary libraries
   from machine import Pin, PWM
   import time

   # Create a PWM object
   led = PWM(Pin(26), freq=1000)

   while True:
       # Gradually increase brightness
       for duty_cycle in range(0, 1024, 2):
           led.duty(duty_cycle)
           time.sleep(0.01)

       # Gradually decrease brightness
       for duty_cycle in range(1023, -1, -2):
   ```

(continué en la próxima página)

(proviene de la página anterior)

```
        led.duty(duty_cycle)
        time.sleep(0.01)
```

- `range()`: Crea una secuencia de enteros de 0 a 1023.
- El ciclo de trabajo de la señal PWM se establece en cada valor de la secuencia usando el método `duty()` del objeto `PWM`.
- `time.sleep()`: Pausa la ejecución del programa durante 10 milisegundos entre cada iteración del bucle, creando un aumento gradual en el brillo con el tiempo.

4.9 2.3 Luz Colorida

En este proyecto, exploraremos el fascinante mundo de la mezcla de colores aditivos usando un LED RGB.

El LED RGB combina tres colores primarios, que son Rojo, Verde y Azul, en un solo paquete. Estos tres LEDs comparten un pin de cátodo común, mientras que cada pin de ánodo controla la intensidad del color correspondiente.

Variando la intensidad de la señal eléctrica aplicada a cada ánodo, podemos crear una amplia gama de colores. Por ejemplo, mezclar luz roja y verde de alta intensidad resultará en luz amarilla, mientras que combinar luz azul y verde producirá cian.

A través de este proyecto, exploraremos los principios de la mezcla de colores aditivos y desataremos nuestra creatividad manipulando el LED RGB para mostrar colores vibrantes y cautivadores.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

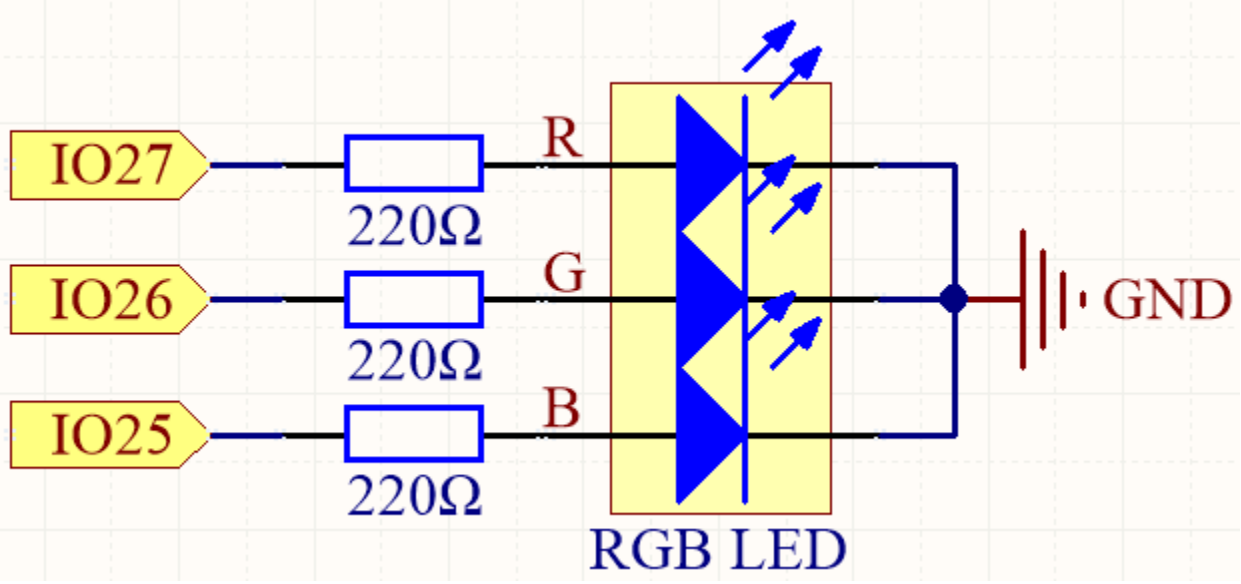
INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED RGB	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

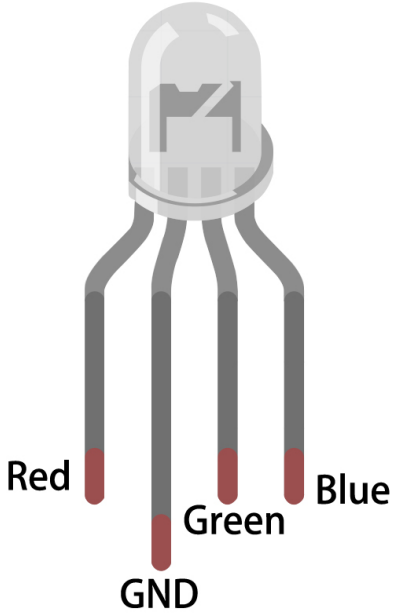
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático

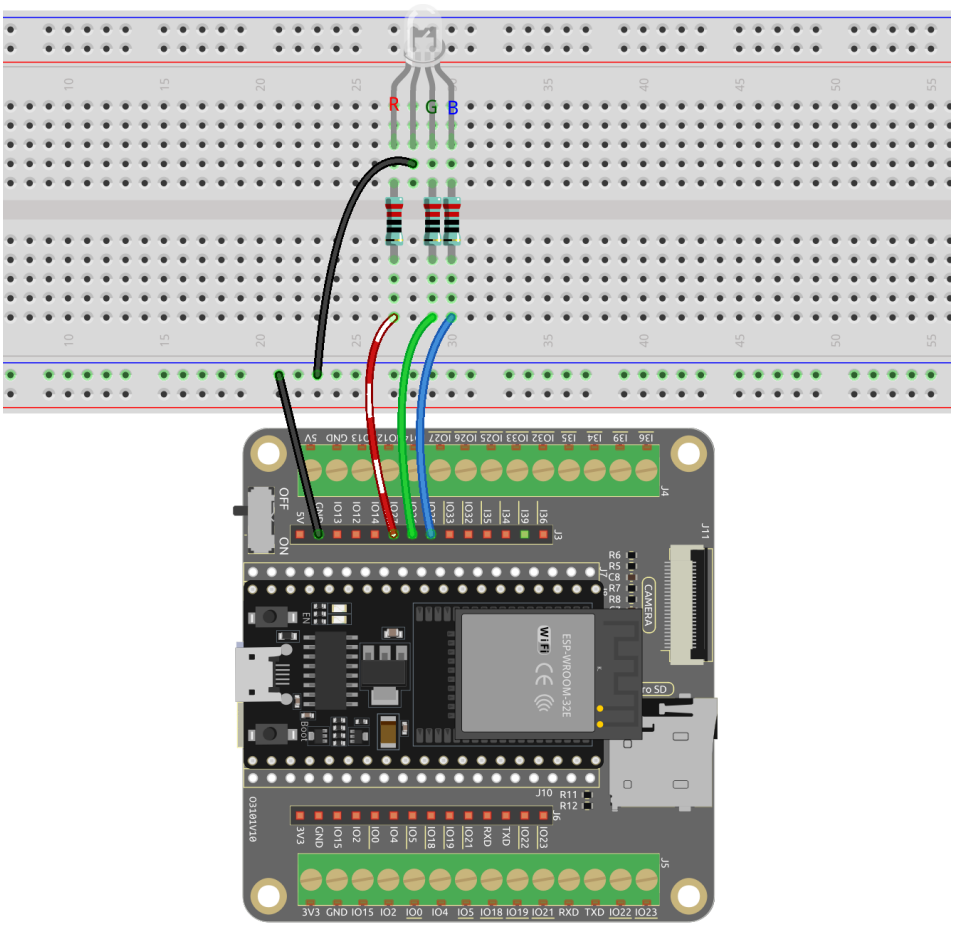


Los pines PWM pin27, pin26 y pin25 controlan los pines Rojo, Verde y Azul del LED RGB respectivamente, y conectan el pin de cátodo común a GND. Esto permite que el LED RGB muestre un color específico superponiendo luz en estos pines con diferentes valores PWM.

Conexión



El LED RGB tiene 4 pines: el pin largo es el pin de cátodo común, que usualmente se conecta a GND; el pin izquierdo junto al pin más largo es Rojo; y los dos pines a la derecha son Verde y Azul.



Código

Nota:

- Abre el archivo `2.3_colorful_light.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, PWM
import time

# Define the GPIO pins for the RGB LED
RED_PIN = 27
GREEN_PIN = 26
BLUE_PIN = 25


# Set up the PWM channels
red = PWM(Pin(RED_PIN))
green = PWM(Pin(GREEN_PIN))
blue = PWM(Pin(BLUE_PIN))
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Set the PWM frequency
red.freq(1000)
green.freq(1000)
blue.freq(1000)

def set_color(r, g, b):
    red.duty(r)
    green.duty(g)
    blue.duty(b)

while True:
    # Set different colors and wait for a while
    set_color(1023, 0, 0) # Red
    time.sleep(1)
    set_color(0, 1023, 0) # Green
    time.sleep(1)
    set_color(0, 0, 1023) # Blue
    time.sleep(1)
    set_color(1023, 0, 1023) # purple
    time.sleep(1)
```

Cuando se ejecute el script, verás que los LEDs RGB muestran rojo, verde, azul y morado, y así sucesivamente.

Aprende Más

También puedes establecer el color que desees con el siguiente código con los valores de color familiares de 0~255.

Nota:

- Abre el archivo `2.3_colorful_light_rgb.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, PWM
import time

# Define the GPIO pins for the RGB LED
RED_PIN = 27
GREEN_PIN = 26
BLUE_PIN = 25

# Set up the PWM channels
red = PWM(Pin(RED_PIN))
green = PWM(Pin(GREEN_PIN))
blue = PWM(Pin(BLUE_PIN))

# Set the PWM frequency
red.freq(1000)
green.freq(1000)
blue.freq(1000)
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Map input values from one range to another
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Convert color values (0-255) to duty cycle values (0-1023)
def color_to_duty(rgb_value):
    rgb_value = int(interval_mapping(rgb_value,0,255,0,1023))
    return rgb_value

def set_color(red_value,green_value,blue_value):
    red.duty(color_to_duty(red_value))
    green.duty(color_to_duty(green_value))
    blue.duty(color_to_duty(blue_value))

while True:
    # Set different colors and wait for a while
    set_color(255, 0, 0) # Red
    time.sleep(1)
    set_color(0, 255, 0) # Green
    time.sleep(1)
    set_color(0, 0, 255) # Blue
    time.sleep(1)
    set_color(255, 0, 255) # purple
    time.sleep(1)
```

Este código se basa en el ejemplo anterior, pero mapea los valores de color de 0 a 255 a un rango de ciclo de trabajo de 0 a 1023.

- La función `interval_mapping` es una función de utilidad que mapea un valor de un rango a otro. Toma cinco argumentos: el valor de entrada, los valores mínimos y máximos del rango de entrada, y los valores mínimos y máximos del rango de salida. Devuelve el valor de entrada mapeado al rango de salida.

  ```
  def color_to_duty(rgb_value):
      rgb_value = int(interval_mapping(rgb_value,0,255,0,1023))
      return rgb_value
  ```

- La función `color_to_duty` toma un valor RGB entero (por ejemplo, 255,0,255) y lo mapea a un valor de ciclo de trabajo adecuado para los pines PWM. El valor RGB de entrada se mapea primero del rango 0-255 al rango 0-1023 usando la función `interval_mapping`. El resultado de `interval_mapping` se devuelve entonces como el valor del ciclo de trabajo.

  ```
  def color_to_duty(rgb_value):
      rgb_value = int(interval_mapping(rgb_value,0,255,0,1023))
      return rgb_value
  ```

- La función `set_color` toma tres argumentos enteros: los valores rojo, verde y azul para el LED. Estos valores se pasan a `color_to_duty` para obtener los valores del ciclo de trabajo para los pines PWM. Los valores del ciclo de trabajo se establecen entonces para los pines correspondientes usando el método `duty`.

  ```
  def set_color(red_value,green_value,blue_value):
      red.duty(color_to_duty(red_value))
      green.duty(color_to_duty(green_value))
  ```

(continué en la próxima página)

(proviene de la página anterior)

```
    blue.duty(color_to_duty(blue_value))
```

4.10 2.4 Microchip - 74HC595

¡Bienvenidos a este emocionante proyecto! En este proyecto, utilizaremos el chip 74HC595 para controlar una secuencia fluida de 8 LEDs.

Imagina activar este proyecto y ser testigo de un flujo hipnotizante de luz, como si un arcoíris chispeante saltara entre los 8 LEDs. Cada LED se iluminará uno tras otro y se apagará rápidamente, mientras el siguiente LED sigue brillando, creando un efecto dinámico y espléndido.

Mediante el uso ingenioso del chip 74HC595, podemos controlar los estados de encendido y apagado de múltiples LEDs para lograr el efecto fluido. Este chip tiene múltiples pines de salida que pueden conectarse en serie para controlar la secuencia de iluminación de los LEDs. Además, gracias a la capacidad de expansión del chip, podemos añadir fácilmente más LEDs a la secuencia, creando efectos aún más espectaculares.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los siguientes enlaces.

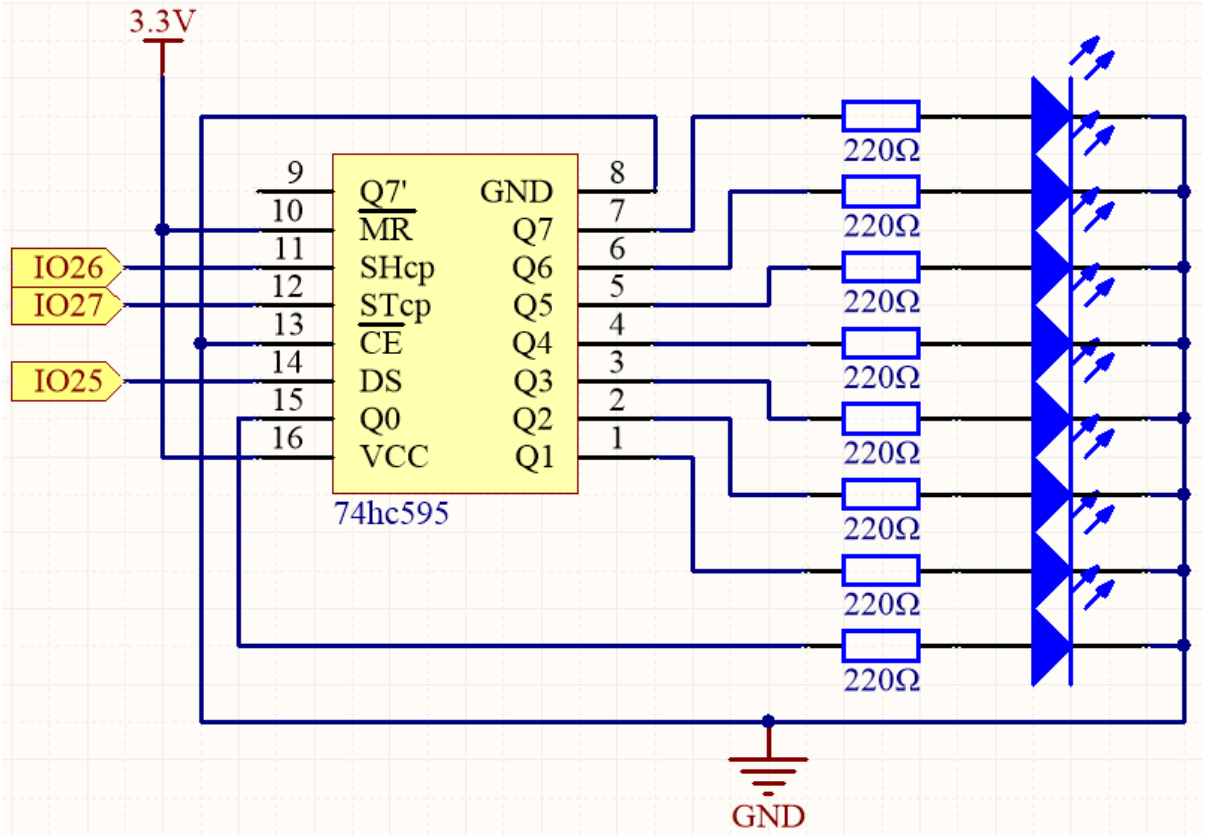
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
74HC595	

Pines Disponibles

Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

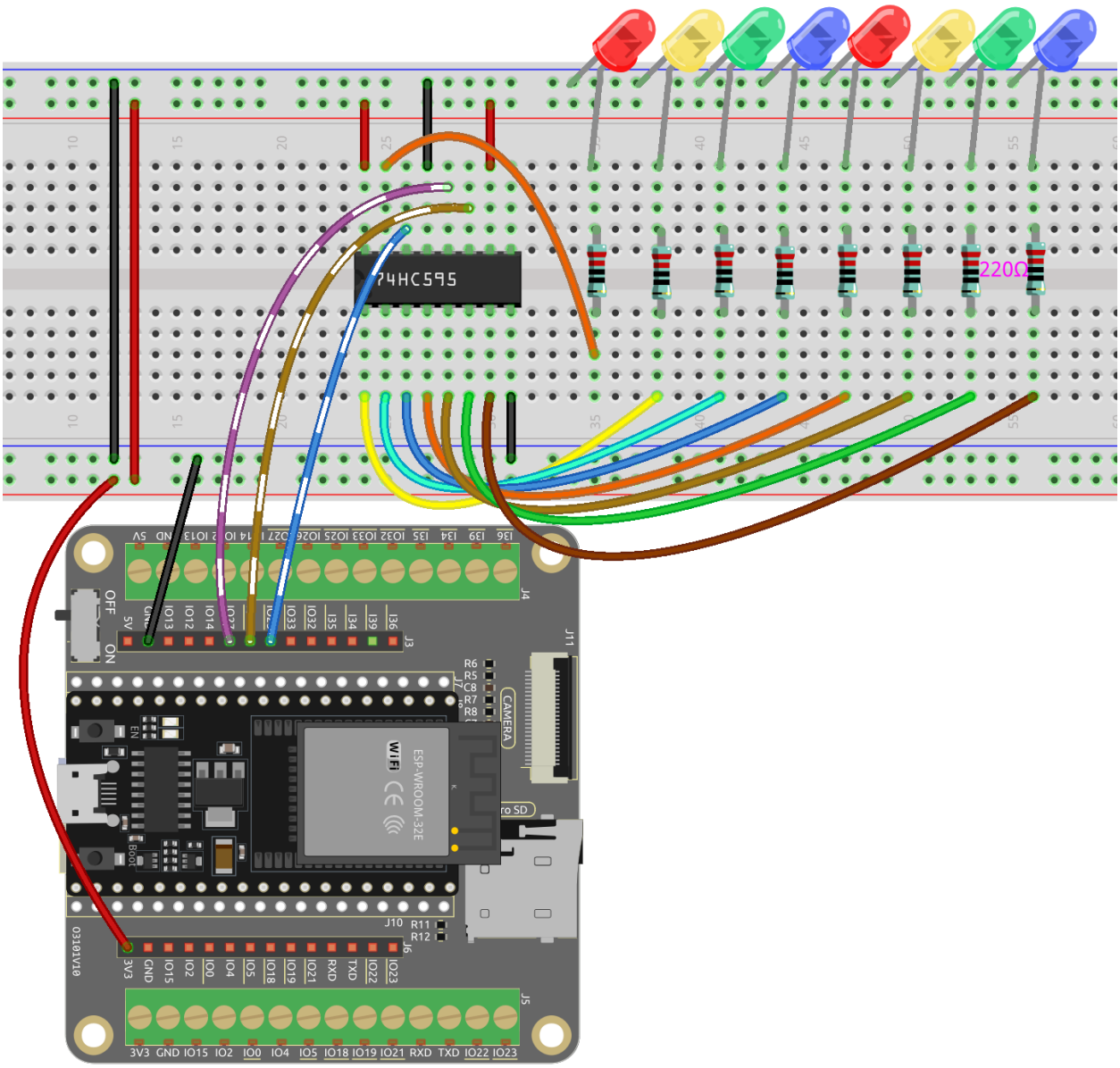
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



- Cuando MR (pin10) está en alto y CE (pin13) en bajo, los datos se ingresan en el flanco ascendente de SHcp y pasan al registro de memoria a través del flanco ascendente de SHcp.
- Si los dos relojes están conectados juntos, el registro de desplazamiento siempre está un pulso antes que el registro de memoria.
- Hay un pin de entrada de desplazamiento serial (DS), un pin de salida serial (Q7") y un botón de reinicio asincrónico (nivel bajo) en el registro de memoria.
- El registro de memoria produce un Bus con 8 bits paralelos y en tres estados.
- Cuando OE está activado (nivel bajo), los datos en el registro de memoria se envían al bus (Q0 ~ Q7).

Cableado



Código

Nota:

- Abre el archivo `2.4_microchip_74hc595.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Initialize the pins for the 74HC595 shift register
```

(continué en la próxima página)

(proviene de la página anterior)

```
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp

# Define the hc595_shift function to shift data into the 74HC595 shift register
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

num = 0

# Shift data into the 74HC595 to create a moving LED pattern
for i in range(16):
    if i < 8:
        num = (num << 1) + 1  # Shift left and set the least significant bit to 1
    elif i >= 8:
        num = (num & 0b01111111) << 1  # Mask the most significant bit and shift left
    hc595_shift(num)  # Shift the current value into the 74HC595
    print("{:0>8b}".format(num))  # Print the current value in binary format
    time.sleep_ms(200)  # Wait 200 milliseconds before shifting the next value
```

Durante la ejecución del script, verás cómo se iluminan los LEDs uno por uno y luego se apagan en el orden original.

¿Cómo funciona?

Este código se utiliza para controlar un registro de desplazamiento de 8 bits (74595) y emitir diferentes valores binarios al registro de desplazamiento, con cada valor mostrado en un LED por un cierto período de tiempo.

1. El código importa los módulos `machine` y `time`, donde el módulo `machine` se utiliza para controlar E/S de hardware, y el módulo `time` se utiliza para implementar retrasos de tiempo y otras funciones.

    ```
    import machine
    import time
    ```

2. Tres puertos de salida se inicializan usando la función `machine.Pin()`, correspondiendo al puerto de datos (SDI), puerto de reloj de almacenamiento (RCLK) y puerto de reloj del registro de desplazamiento (SRCLK) del registro de desplazamiento.

```
# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp
```

3. Se define una función llamada `hc595_shift()` para escribir un dato de 8 bits en el registro de desplazamiento.

```
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the␣
↪SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to␣
↪high
    rclk.on()
```

4. Acerca del bucle `for`.

```
for i in range(16):
        if i < 8:
            num = (num << 1) + 1  # Shift left and set the least␣
↪significant bit to 1
        elif i >= 8:
            num = (num & 0b01111111) << 1  # Mask the most significant bit␣
↪and shift left
        hc595_shift(num)  # Shift the current value into the 74HC595
        print("{:0>8b}".format(num))  # Print the current value in binary␣
↪format
        time.sleep_ms(200)  # Wait 200 milliseconds before shifting the␣
↪next value
```

   - La variable `i` se utiliza para controlar el valor binario de salida. En las primeras 8 iteraciones, el valor de num será sucesivamente 00000001, 00000011, 00000111, ..., 11111111, que se desplaza a la izquierda por un bit y luego se suma 1.
   - En las iteraciones del 9 al 16, el bit más alto de 1 se cambia primero a 0, y luego se desplaza a la izquierda por un bit, resultando en los valores de salida de 00000010, 00000100, 00001000, ..., 10000000.
   - En cada iteración, el valor de `num` se pasa a la función `hc595_shift()` para controlar el registro de desplazamiento para emitir el valor binario correspondiente.

- Al mismo tiempo que se emite el valor binario, la función `print()` muestra el valor binario como una cadena en el terminal.
- Después de emitir el valor binario, el programa hace una pausa durante 200 milisegundos usando la función `time.sleep_ms()`, para que el valor en el LED permanezca mostrado por un cierto período de tiempo.

4.11 2.5 Visualización de Números

¡Bienvenidos a este fascinante proyecto! En este proyecto, exploraremos el encantador mundo de mostrar números del 0 al 9 en una pantalla de siete segmentos.

Imagina activar este proyecto y ser testigo de cómo una pequeña y compacta pantalla brilla intensamente mostrando cada número del 0 al 9. Es como tener una mini pantalla que exhibe los dígitos de una manera cautivadora. Controlando los pines de señal, puedes cambiar sin esfuerzo el número mostrado y crear varios efectos atractivos.

A través de conexiones de circuito simples y programación, aprenderás cómo interactuar con la pantalla de siete segmentos y dar vida a los números que desees. Ya sea un contador, un reloj o cualquier otra aplicación intrigante, la pantalla de siete segmentos será tu fiel compañera, añadiendo un toque de brillantez a tus proyectos.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

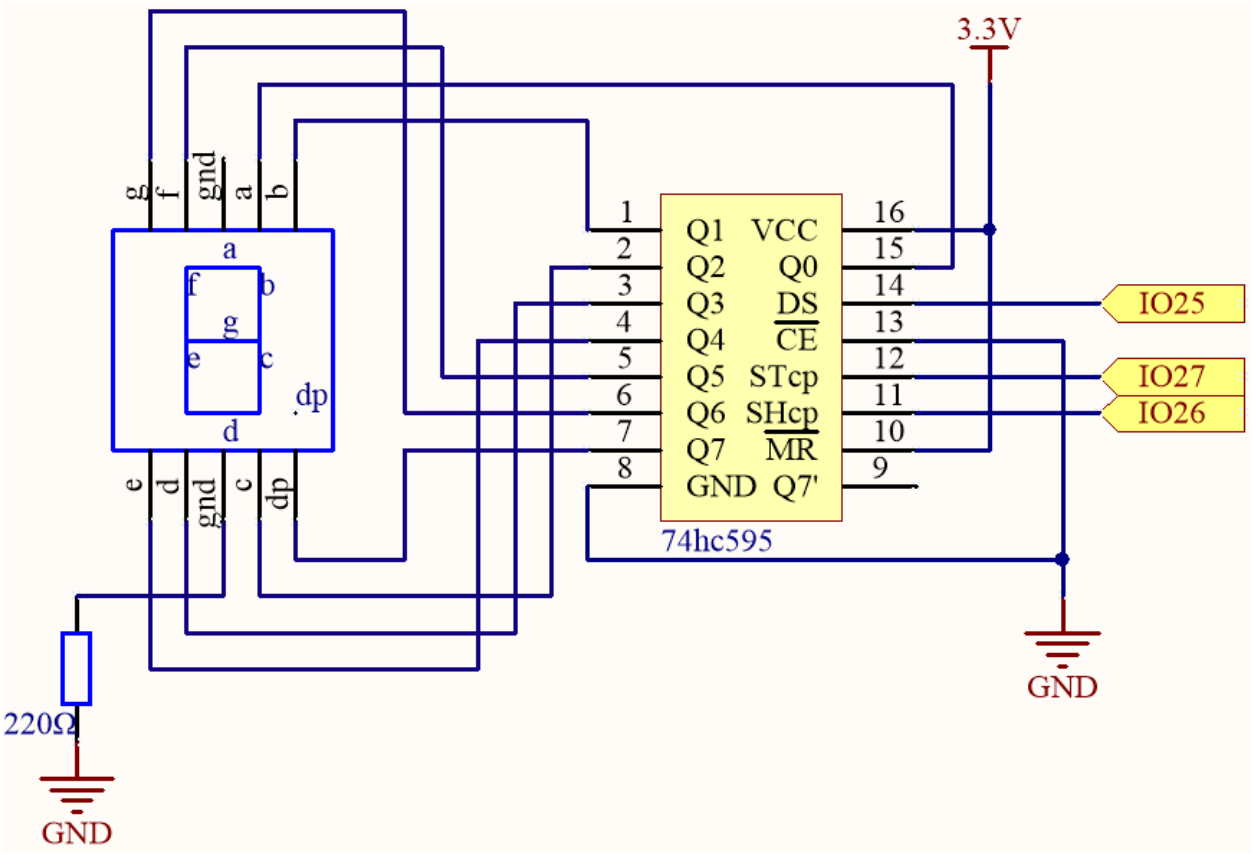
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
display de 7 Segmentos	
74HC595	

Pines Disponibles

Aquí tienes una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático

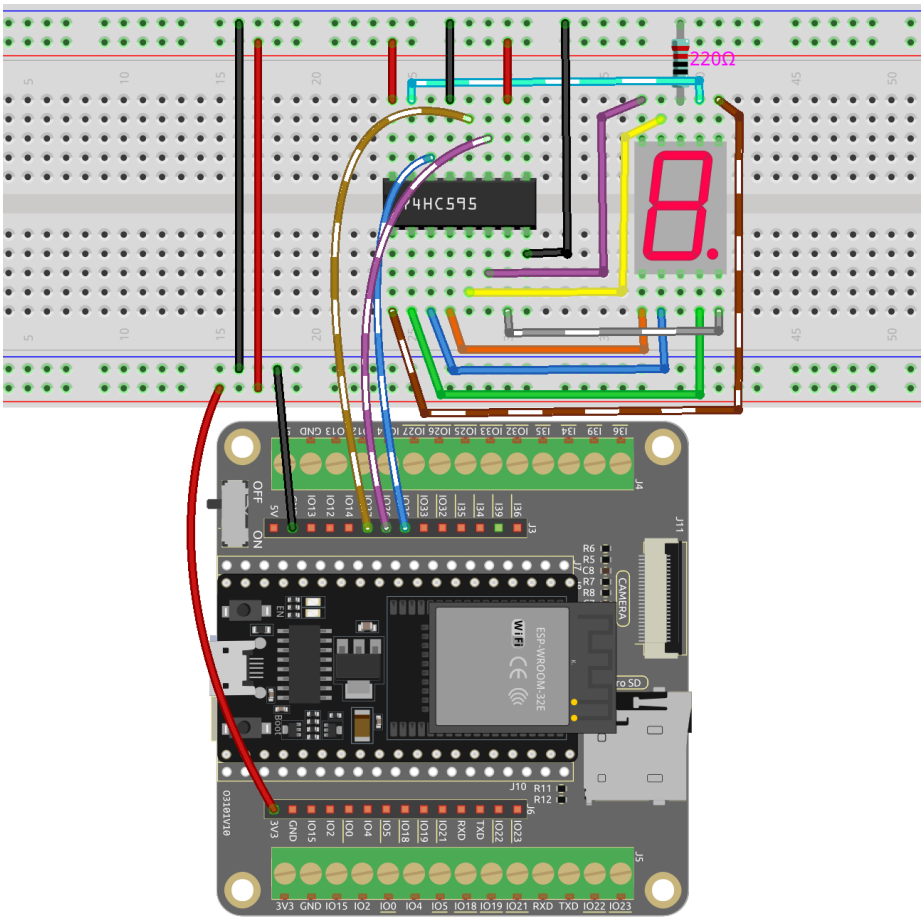


Aquí el principio de cableado es básicamente el mismo que en *2.4 Microchip - 74HC595*, la única diferencia es que Q0-Q7 están conectados a los pines a ~ g de la Pantalla de 7 Segmentos.

Tabla 1: Cableado

74HC595	Pantalla de Segmento LED
Q0	a
Q1	b
Q2	c
Q3	d
Q4	e
Q5	f
Q6	g
Q7	dp

Cableado



Código

Nota:

- Abre el archivo `2.5_number_display.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Run Current Script» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Define the segment code for a common anode 7-segment display
SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp

# Define the hc595_shift function to shift data into the 74HC595 shift register
def hc595_shift(dat):
    # Set the RCLK pin to low
```

(continué en la próxima página)

(proviene de la página anterior)

```
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

# Continuously loop through the numbers 0 to 9 and display them on the 7-segment display
while True:
    for num in range(10):
        hc595_shift(SEGCODE[num])  # Shift the segment code for the current number into␣
↪the 74HC595
        time.sleep_ms(500)  # Wait 500 milliseconds before displaying the next number
```

Cuando el script esté en ejecución, podrás ver cómo el Display de Segmentos LED muestra los números del 0 al 9 en secuencia.

¿Cómo funciona?

En este proyecto, utilizamos la función `hc595_shift()` para escribir el número binario en el registro de desplazamiento.

Supongamos que el Display de 7 segmentos muestra el número «2». Este patrón de bits corresponde a los segmentos **f**, **c** y **dp** apagados (bajo), mientras que los segmentos **a**, **b**, **d**, **e** y **g** están encendidos (alto). Esto es «01011011» en binario y «0x5b» en notación hexadecimal.

Por lo tanto, necesitarías llamar a **hc595_shift(0x5b)** para mostrar el número «2» en el display de 7 segmentos.

- Hexadecimal
- Conversor BinarioHex

La siguiente tabla muestra los patrones hexadecimales que deben escribirse en el registro de desplazamiento para mostrar los números del 0 al 9 en un display de 7 segmentos.

Tabla 2: Código de Glifo

Números	Código Binario	Código Hex
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f

Escribe estos códigos en `hc595_shift()` para que el Display de Segmentos LED muestre los números correspondientes.

4.12 2.6 Mostrar Caracteres

Ahora, exploraremos el fascinante mundo de la visualización de caracteres utilizando el módulo LCD1602 I2C.

A través de este proyecto, aprenderemos cómo inicializar el módulo LCD, configurar los parámetros de visualización deseados y enviar datos de caracteres para ser mostrados en la pantalla. Podemos mostrar mensajes personalizados, exhibir lecturas de sensores o crear menús interactivos. ¡Las posibilidades son infinitas!

Dominando el arte de la visualización de caracteres en el LCD1602 I2C, desbloquearemos nuevas vías para la comunicación y la muestra de información en nuestros proyectos. Sumérgete en este emocionante viaje y da vida a tus caracteres en la pantalla del LCD.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

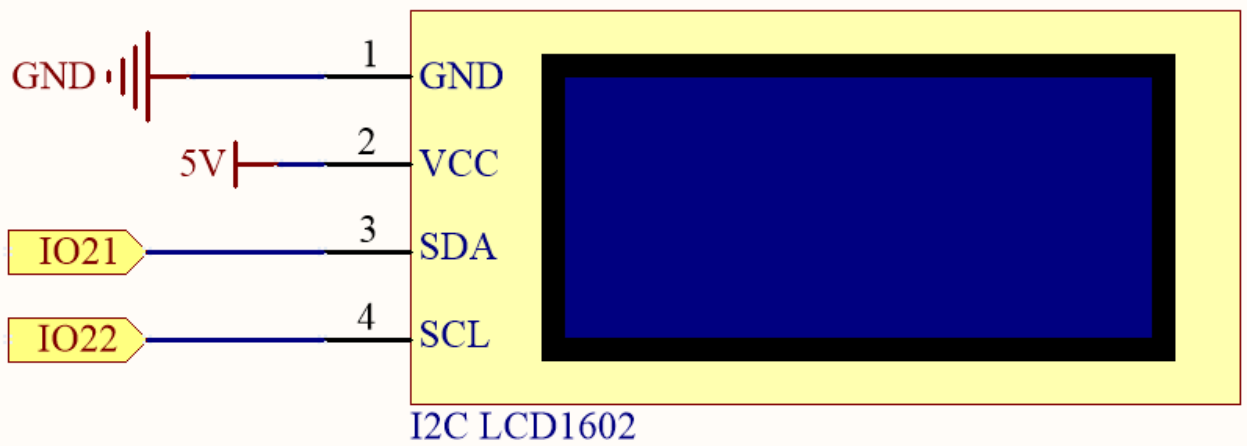
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
I2C LCD1602	

Pines Disponibles

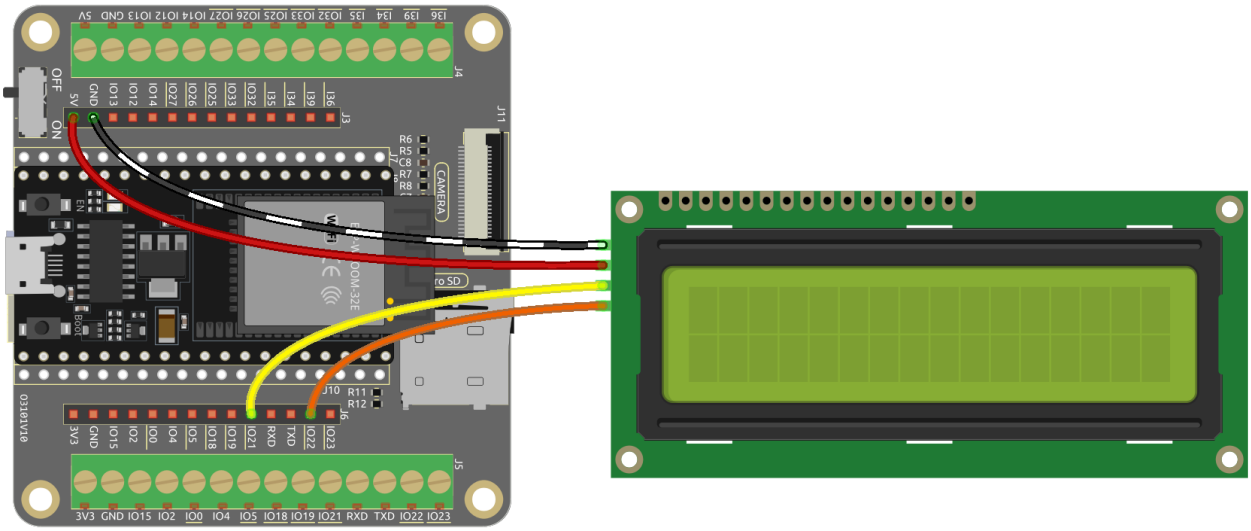
Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	Descripción de Uso
IO21	SDA
IO22	SCL

Esquemático



Conexión



Código

Nota:

- Abre el archivo `2.6_liquid_crystal_display.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.
- Se utiliza aquí la biblioteca `lcd1602.py` y verifica si se ha cargado en el ESP32. Consulta *1.4 Subir las Bibliotecas (Importante)* para obtener un tutorial.

```
# Import the LCD class from the lcd1602 module
from lcd1602 import LCD

import time

# Create an instance of the LCD class and assign it to the lcd variable
lcd = LCD()
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Set the string " Hello!\n"
string = " Hello!\n"
# Display the string on the LCD screen
lcd.message(string)

time.sleep(2)
# Set the string "    Sunfounder!"
string = "    Sunfounder!"
# Display the string on the LCD screen
lcd.message(string)

time.sleep(2)
# Clear the LCD screen
lcd.clear()
```

Después de ejecutar el script, podrás ver dos líneas de texto que aparecen en la pantalla LCD por turnos y luego desaparecen.

Nota: Si el código y la conexión son correctos, pero el LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte posterior para aumentar el contraste.

¿Cómo funciona?

En la biblioteca `lcd1602`, integramos las funciones relevantes del lcd1602 en la clase LCD.

1. Importa el módulo `lcd1602`.

   ```
   from lcd1602 import LCD
   ```

2. Declara un objeto de la clase `LCD` y nómbralo `lcd`.

   ```
   lcd = LCD()
   ```

3. Esta instrucción mostrará el texto en el LCD. Es importante señalar que el argumento debe ser de tipo cadena. Si queremos pasar un entero o un flotante, debemos utilizar la instrucción de conversión forzada `str()`.

   ```
   lcd.message(string)
   ```

4. Si llamas a esta instrucción varias veces, el LCD superpondrá los textos. Esto requiere el uso de la siguiente instrucción para limpiar la pantalla.

   ```
   lcd.clear()
   ```

4.13 2.7 Tira de LED RGB

En este proyecto, nos adentraremos en el fascinante mundo de controlar tiras de LED WS2812, dando vida a un vibrante espectáculo de colores. Con la capacidad de controlar individualmente cada LED de la tira, podemos crear efectos de iluminación cautivadores que deslumbrarán los sentidos.

Además, hemos incluido una extensión emocionante a este proyecto, donde exploraremos el reino de la aleatoriedad. Introduciendo colores aleatorios y implementando un efecto de luz fluida, podemos crear una experiencia visual hipnotizante que cautiva y encanta.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

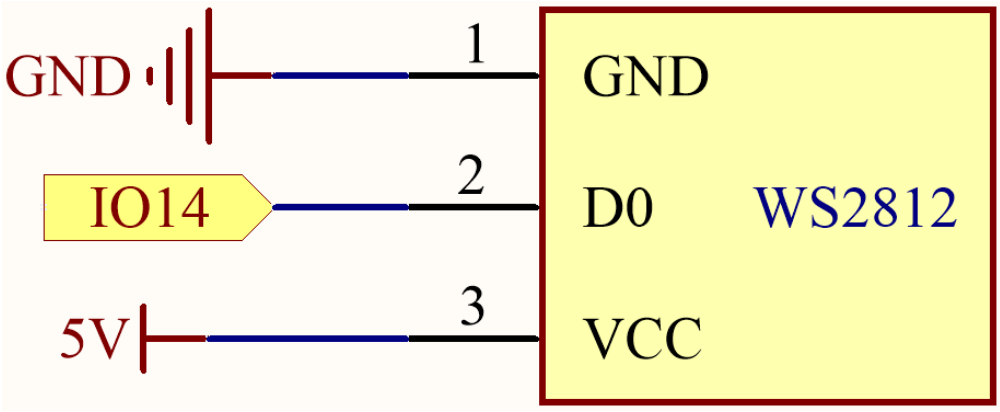
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Tira de 8 LEDs RGB WS2812	

Esquemático



Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

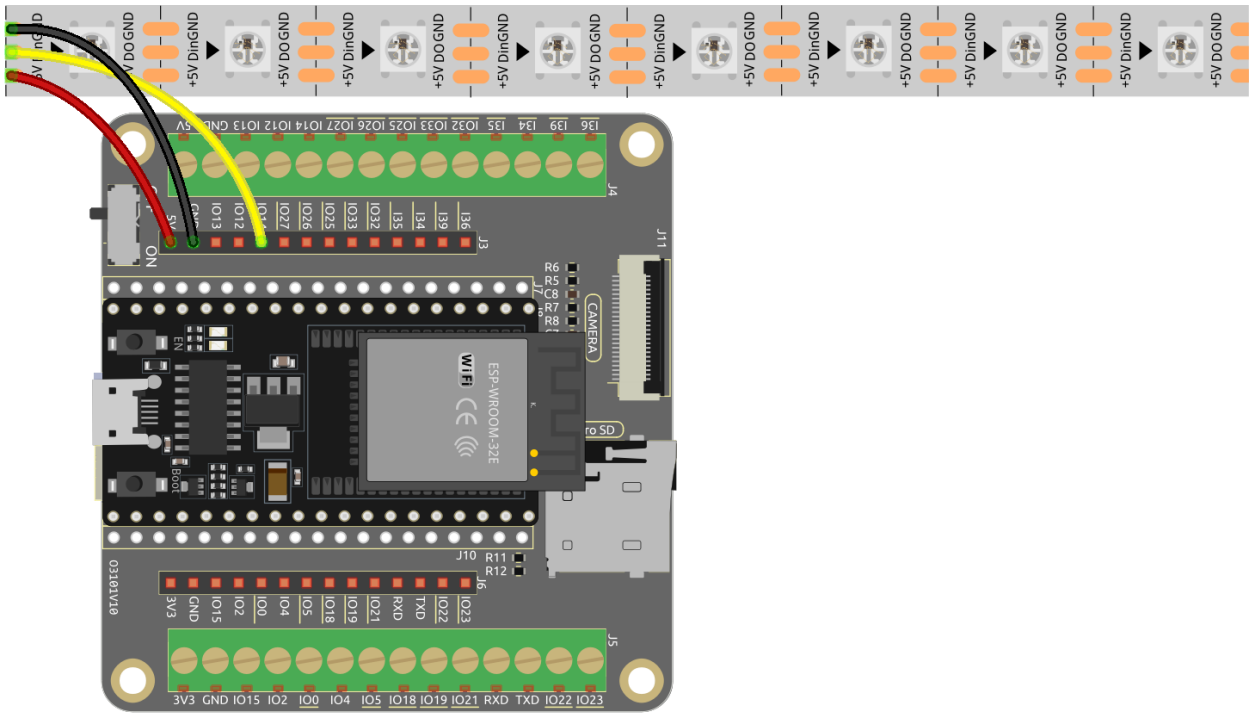
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Nota: IO33 no está disponible para este proyecto.

La tira de LED WS2812 es un tipo de tira de LED que requiere una señal precisa de modulación por ancho de pulso (PWM). La señal PWM tiene requisitos precisos tanto en tiempo como en voltaje. Por ejemplo, un bit «0» para el WS2812 corresponde a un pulso de alto nivel de aproximadamente 0.4 microsegundos, mientras que un bit «1» corresponde a un pulso de alto nivel de aproximadamente 0.8 microsegundos. Esto significa que la tira necesita recibir cambios de voltaje de alta frecuencia.

Sin embargo, con una resistencia de pull-up de 4.7K y un condensador de pull-down de 100nf en IO33, se crea un filtro pasa bajo simple. Este tipo de circuito «suaviza» las señales de alta frecuencia, porque el condensador necesita algún tiempo para cargar y descargar cuando recibe cambios de voltaje. Por lo tanto, si la señal cambia demasiado rápido (es decir, es de alta frecuencia), el condensador no podrá seguir el ritmo. Esto resulta en que la señal de salida se vuelva borrosa e irreconocible para la tira.

Conexión



Código

Nota:

- Abre el archivo `2.7_rgb_strip.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin
from neopixel import NeoPixel

pin = Pin(14, Pin.OUT)    # set a pin to output to drive NeoPixels
pixels = NeoPixel(pin, 8)    # create NeoPixel driver on pin for 8 pixels
```

(continué en la próxima página)

(proviene de la página anterior)

```
pixels[0] = [64,154,227]    # set the pixel
pixels[1] = [128,0,128]
pixels[2] = [50,150,50]
pixels[3] = [255,30,30]
pixels[4] = [0,128,255]
pixels[5] = [99,199,0]
pixels[6] = [128,128,128]
pixels[7] = [255,100,0]

pixels.write()              # write data to all pixels
```

¡Seleccionemos algunos colores favoritos y mostrémoslos en la Tira de LED RGB!

¿Cómo funciona?

1. En el módulo `neopixel`, hemos integrado funciones relacionadas en la clase `NeoPixel`.

```
from neopixel import NeoPixel
```

1. Utiliza la clase `NeoPixel` del módulo `neopixel` para inicializar el objeto `pixels`, especificando el pin de datos y el número de LEDs.

```
pixels = NeoPixel(pin, 8)   # create NeoPixel driver on pin for 8 pixels
```

1. Configura el color de cada LED y usa el método `write()` para enviar los datos al LED WS2812 y actualizar su visualización.

```
pixels[0] = [64,154,227]    # set the pixel
pixels[1] = [128,0,128]
pixels[2] = [50,150,50]
pixels[3] = [255,30,30]
pixels[4] = [0,128,255]
pixels[5] = [99,199,0]
pixels[6] = [128,128,128]
pixels[7] = [255,100,0]

pixels.write()              # write data to all pixels
```

Aprende Más

Podemos generar colores aleatoriamente y hacer una luz fluida colorida.

Nota:

- Abre el archivo `2.7_rgb_strip_random.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo. * Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin
import neopixel
import time
```

(continué en la próxima página)

(proviene de la página anterior)

```
import random

# Set the number of pixels for the running light
num_pixels = 8

# Set the data pin for the RGB LED strip
data_pin = Pin(14, Pin.OUT)

# Initialize the RGB LED strip object
pixels = neopixel.NeoPixel(data_pin, num_pixels)

# Continuously loop the running light
while True:
    for i in range(num_pixels):
        # Generate a random color for the current pixel
        color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

        # Turn on the current pixel with the random color
        pixels[i] = color

        # Update the RGB LED strip display
        pixels.write()

        # Turn off the current pixel
        pixels[i] = (0, 0, 0)

        # Wait for a period of time to control the speed of the running light
        time.sleep_ms(100)
```

- En el bucle `while`, usamos un bucle `for` para encender cada píxel de la tira de LED RGB uno por uno.
- Primero usa la función `random.randint()` para generar un color aleatorio para el píxel actual.
- Luego enciende el píxel actual con el color aleatorio, usa el método `write()` del objeto `NeoPixel` para enviar los datos de color a la tira de LED RGB y actualizar su visualización.
- Finalmente, apaga el píxel actual configurando su color a (0, 0, 0), y espera un período de tiempo para controlar la velocidad de la luz corriente.

3. Sonidos

4.14 3.1 Pitido

Este es un proyecto simple para hacer que un zumbador activo emita un pitido rápidamente cuatro veces cada segundo.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

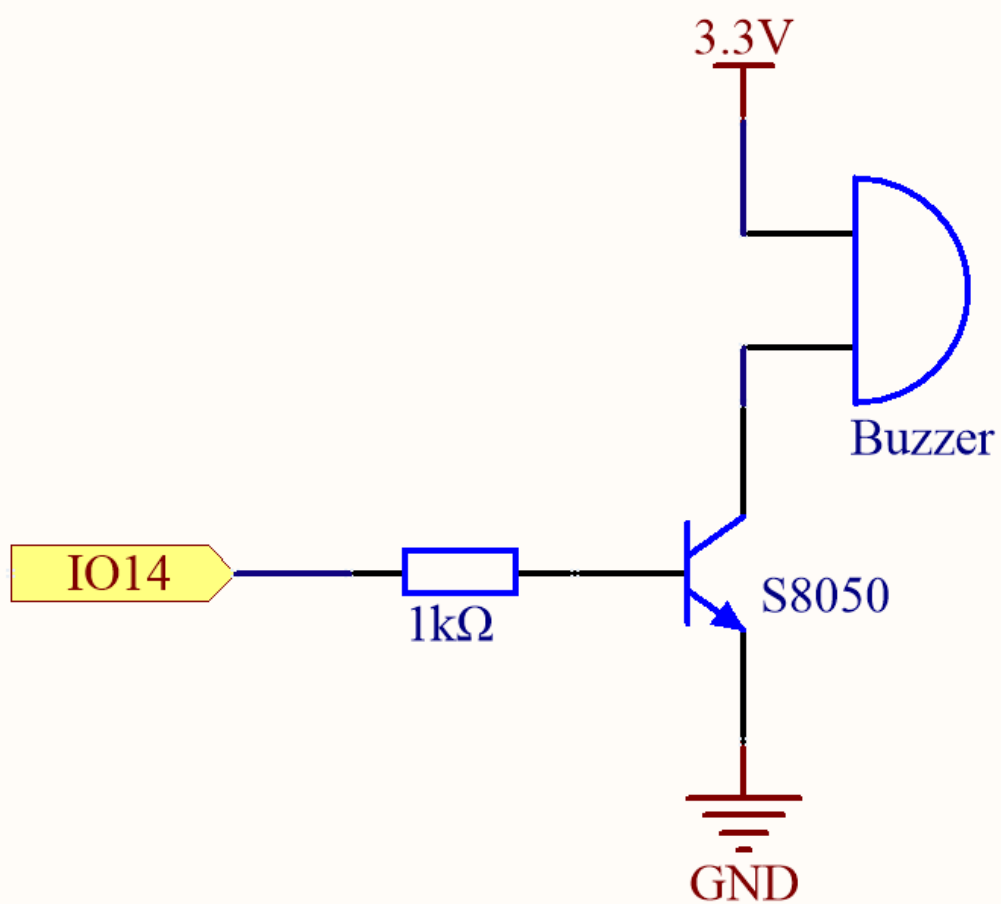
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Zumbador	-
Transistor	

Pines Disponibles

Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



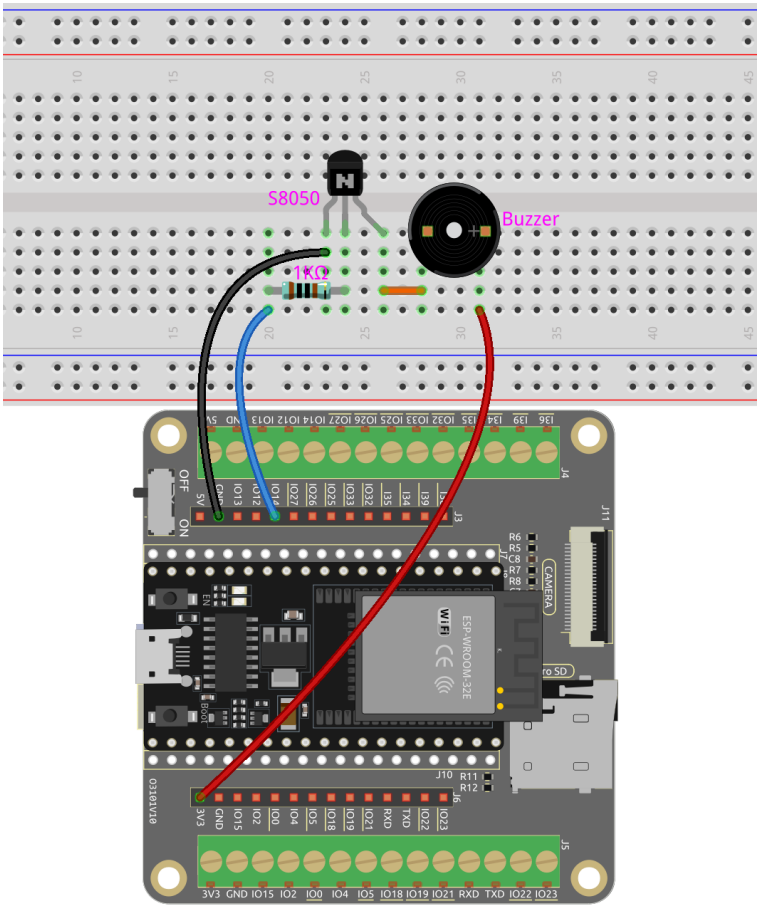
Cuando la salida IO14 es alta, después de la resistencia limitadora de corriente de 1K (para proteger el transistor), el S8050 (transistor NPN) conducirá, haciendo que el zumbador suene.

El rol del S8050 (transistor NPN) es amplificar la corriente y hacer que el zumbador suene más fuerte. De hecho, también puedes conectar el zumbador directamente a IO14, pero encontrarás que el sonido del zumbador es menor.

Cableado

El kit incluye dos tipos de zumbadores. Necesitamos usar el zumbador activo. Gíralos, el que tiene la parte trasera sellada (no el PCB expuesto) es el que queremos.

El zumbador necesita usar un transistor para funcionar, aquí usamos S8050 (Transistor NPN).



Código

Nota:

- Abre el archivo `3.1_beep.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Create a Pin object representing pin 14 and set it to output mode
buzzer = machine.Pin(14, machine.Pin.OUT)

# Enter an infinite loop
while True:
    # Iterate over the values 0 to 3 using a for loop
    for i in range(4):
        # Turn on the buzzer by setting its value to 1
        buzzer.value(1)
        # Pause for 0.2 seconds
        time.sleep(0.2)
        # Turn off the buzzer
        buzzer.value(0)
        # Pause for 0.2 seconds
        time.sleep(0.2)
    # Pause for 1 second before restarting the for loop
    time.sleep(1)
```

Cuando el script está en ejecución, el zumbador emitirá un pitido rápidamente cuatro veces cada segundo.

4.15 3.2 Tono Personalizado

Hemos utilizado un zumbador activo en el proyecto anterior, esta vez usaremos un zumbador pasivo.

Al igual que el zumbador activo, el zumbador pasivo también utiliza el fenómeno de inducción electromagnética para funcionar. La diferencia es que un zumbador pasivo no tiene fuente oscilante, por lo que no emitirá un pitido si se utilizan señales de CC. Pero esto permite que el zumbador pasivo ajuste su propia frecuencia de oscilación y pueda emitir diferentes notas como «do, re, mi, fa, sol, la, si».

¡Hagamos que el zumbador pasivo emita una melodía!

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

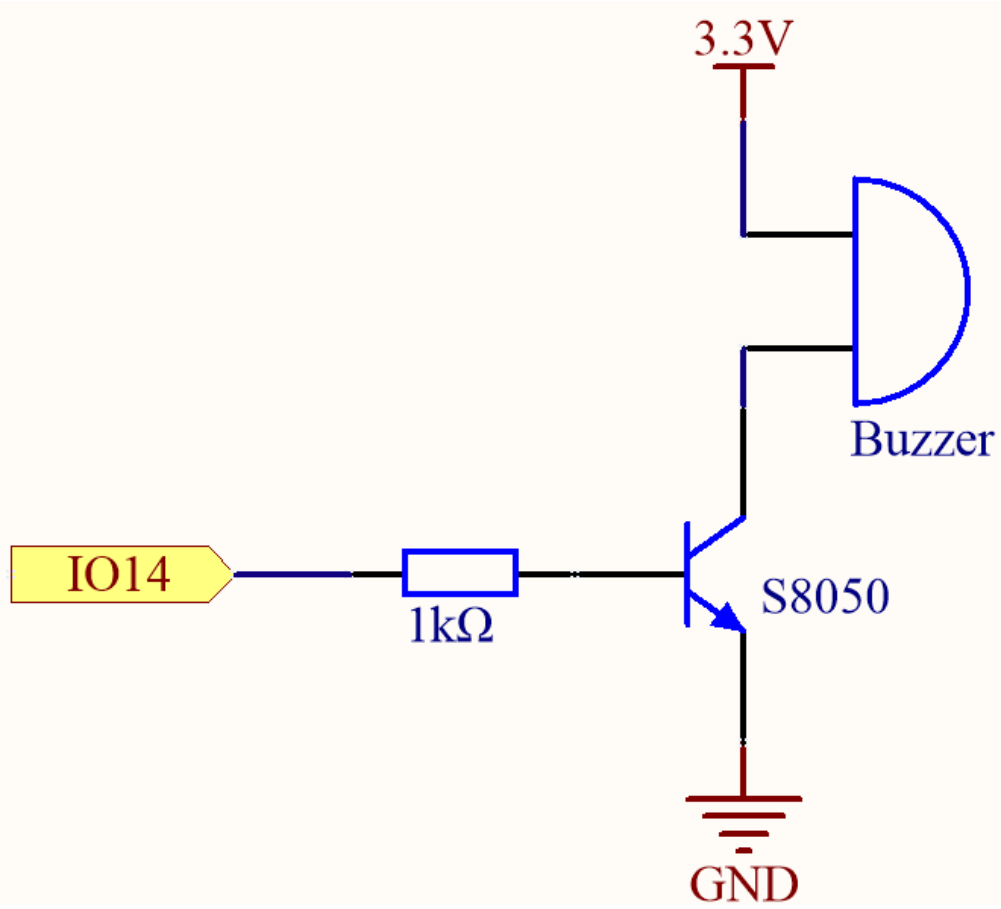
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Zumbador	-
Transistor	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cuando la salida IO14 es alta, después de la resistencia limitadora de corriente de 1K (para proteger el transistor), el

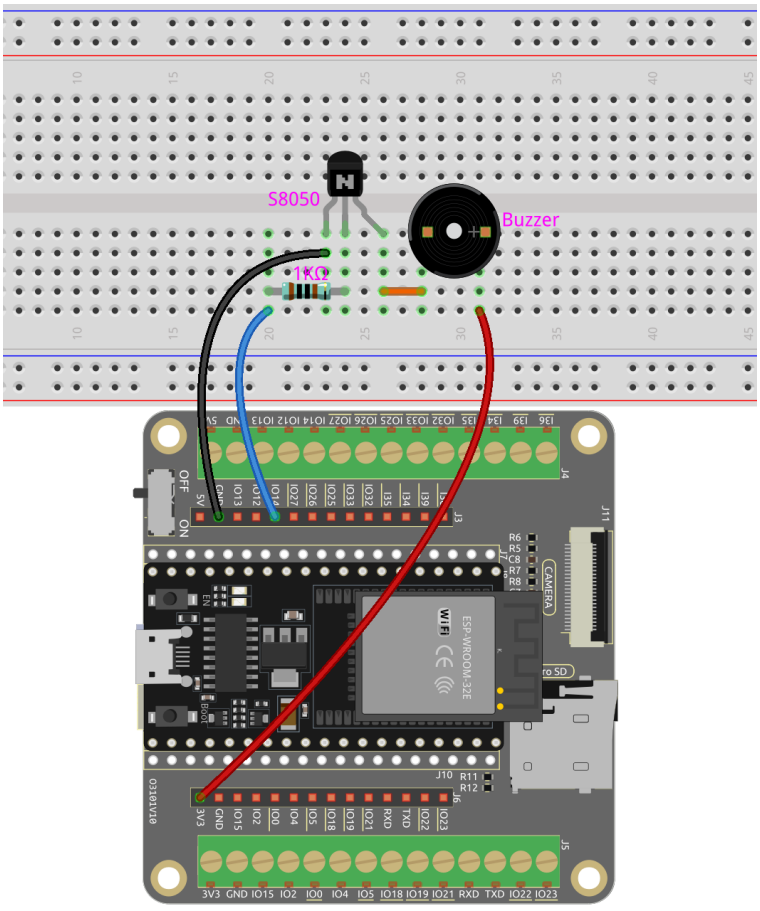
S8050 (transistor NPN) conducirá, haciendo que el zumbador suene.

El papel de S8050 (transistor NPN) es amplificar la corriente y hacer que el zumbador suene más fuerte. De hecho, también puedes conectar el zumbador directamente a IO14, pero encontrarás que el sonido del zumbador es más bajo.

Conexión

El kit incluye dos tipos de zumbadores. Necesitamos usar el zumbador activo. Voltéalos, el lado sellado (no el PCB expuesto) es el que queremos.

El zumbador necesita usar un transistor cuando funciona, aquí usamos S8050 (Transistor NPN).



Código

Nota:

- Abre el archivo `3.2_custom_tone.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Define the frequencies of several musical notes in Hz
C4 = 262
D4 = 294
E4 = 330
F4 = 349
G4 = 392
A4 = 440
B4 = 494

# Create a PWM object representing pin 14 and assign it to the buzzer variable
buzzer = machine.PWM(machine.Pin(14))

# Define a tone function that takes as input a Pin object representing the buzzer, a␣
↪frequency in Hz, and a duration in milliseconds
def tone(pin, frequency, duration):
    pin.freq(frequency) # Set the frequency
    pin.duty(512) # Set the duty cycle
    time.sleep_ms(duration) # Pause for the duration in milliseconds
    pin.duty(0) # Set the duty cycle to 0 to stop the tone

# Play a sequence of notes with different frequency inputs and durations
tone(buzzer, C4, 250)
time.sleep_ms(500)
tone(buzzer, D4, 250)
time.sleep_ms(500)
tone(buzzer, E4, 250)
time.sleep_ms(500)
tone(buzzer, F4, 250)
time.sleep_ms(500)
tone(buzzer, G4, 250)
time.sleep_ms(500)
tone(buzzer, A4, 250)
time.sleep_ms(500)
tone(buzzer, B4, 250)
```

¿Cómo funciona?

Si al zumbador pasivo se le da una señal digital, solo puede continuar empujando el diafragma sin producir sonido.

Por lo tanto, utilizamos la función `tone()` para generar la señal PWM y hacer que el zumbador pasivo emita sonido.

Esta función tiene tres parámetros:

- `pin`: El pin que controla el zumbador.
- `frecuencia`: El tono del zumbador está determinado por la frecuencia; cuanto mayor sea la frecuencia, más alto será el tono.
- `Duración`: La duración del tono.

Usamos la función `duty()` para establecer el ciclo de trabajo en 512 (alrededor del 50 %). Puede ser otros números, y solo necesita generar una señal eléctrica discontinua para oscilar.

Aprender Más

Podemos simular tonos específicos y así reproducir una pieza musical completa.

Nota:

- Abre el archivo `3.2_custom_tone_music.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Define the GPIO pin that is connected to the buzzer
buzzer = machine.PWM(machine.Pin(14))

# Define the frequencies of the notes in Hz
C5 = 523
D5 = 587
E5 = 659
F5 = 698
G5 = 784
A5 = 880
B5 = 988

# Define the durations of the notes in milliseconds
quarter_note = 250
half_note = 300
whole_note = 1000

# Define the melody as a list of tuples (note, duration)
melody = [
    (E5, quarter_note),
    (E5, quarter_note),
    (F5, quarter_note),
    (G5, half_note),
    (G5, quarter_note),
    (F5, quarter_note),
    (E5, quarter_note),
    (D5, half_note),
    (C5, quarter_note),
    (C5, quarter_note),
    (D5, quarter_note),
    (E5, half_note),
    (E5, quarter_note),
    (D5, quarter_note),
    (D5, half_note),
    (E5, quarter_note),
    (E5, quarter_note),
```

(continué en la próxima página)

(proviene de la página anterior)

```
    (F5, quarter_note),
    (G5, half_note),
    (G5, quarter_note),
    (F5, quarter_note),
    (E5, quarter_note),
    (D5, half_note),
    (C5, quarter_note),
    (C5, quarter_note),
    (D5, quarter_note),
    (E5, half_note),
    (D5, quarter_note),
    (C5, quarter_note),
    (C5, half_note),

]

# Define a function to play a note with the given frequency and duration
def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty(512)
    time.sleep_ms(duration)
    pin.duty(0)

# Play the melody
for note in melody:
    tone(buzzer, note[0], note[1])
    time.sleep_ms(50)
```

- La función `tone` establece la frecuencia del pin al valor de `frecuencia` usando el método `freq` del objeto `pin`.
- Luego establece el ciclo de trabajo del pin en 512 usando el método `duty` del objeto `pin`.
- Esto hará que el pin produzca un tono con la frecuencia y volumen especificados durante la duración de `duración` en milisegundos usando el método `sleep_ms` del módulo de tiempo.
- El código luego reproduce una melodía iterando a través de una secuencia llamada `melody` y llamando a la función `tone` para cada nota en la melodía con la frecuencia y duración de la nota.
- También inserta una breve pausa de 50 milisegundos entre cada nota usando el método `sleep_ms` del módulo de tiempo.

4. Actuadores

4.16 4.1 Pequeño Ventilador

En este proyecto apasionante, exploraremos cómo manejar un motor utilizando el L293D.

El L293D es un circuito integrado (CI) versátil comúnmente utilizado para el control de motores en proyectos de electrónica y robótica. Puede manejar dos motores en direcciones hacia adelante y hacia atrás, lo que lo convierte en una opción popular para aplicaciones que requieren un control preciso del motor.

Al final de este proyecto fascinante, habrás ganado una comprensión profunda de cómo se pueden utilizar eficazmente las señales digitales y las señales PWM para controlar motores. Este conocimiento invaluable será una base sólida para tus futuros empeños en robótica y mecatrónica. ¡Prepárate y sumérgete en el emocionante mundo del control de motores con el L293D!

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

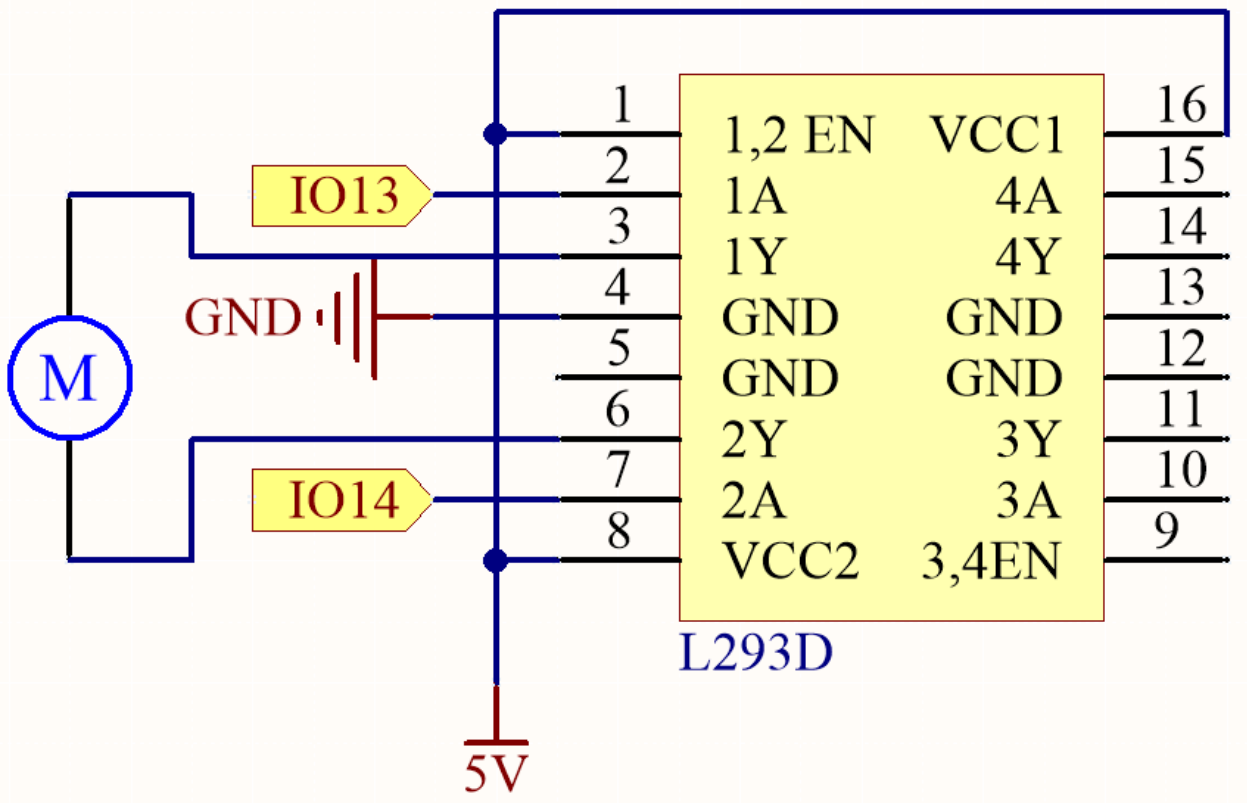
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Motor de Corriente Continua (DC)	
L293D	-

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

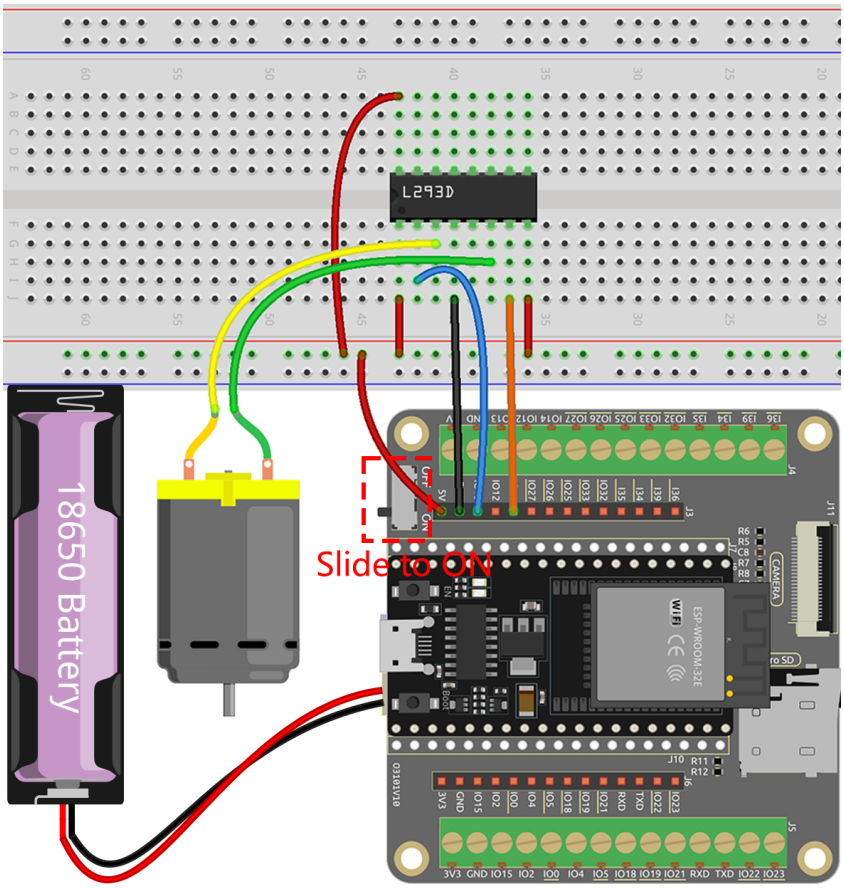
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Conexión

Nota: Dado que el motor requiere una corriente relativamente alta, es necesario primero insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de batería.



Código

Nota:

- Abre el archivo `4.1_motor_turn.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Create Pin objects representing the motor control pins and set them to output mode
motor1A = machine.Pin(13, machine.Pin.OUT)
motor2A = machine.Pin(14, machine.Pin.OUT)

# Define a function to rotate the motor clockwise
def clockwise():
    motor1A.value(1)
    motor2A.value(0)

# Define a function to rotate the motor anticlockwise
def anticlockwise():
    motor1A.value(0)
```

(continué en la próxima página)

(proviene de la página anterior)

```
    motor2A.value(1)

# Define a function to stop the motor
def stop():
    motor1A.value(0)
    motor2A.value(0)

# Enter an infinite loop

try:
    while True:
        clockwise() # Rotate the motor clockwise
        time.sleep(1) # Pause for 1 second
        anticlockwise() # Rotate the motor anticlockwise
        time.sleep(1)
        stop() # Stop the motor
        time.sleep(2)

except KeyboardInterrupt:
    stop()  # Stop the motor when KeyboardInterrupt is caught
```

Durante la ejecución del script, verás el motor girando alternativamente en el sentido de las agujas del reloj y en sentido contrario cada segundo.

Aprender Más

Además de simplemente hacer girar el motor en sentido horario y antihorario, también puedes controlar la velocidad de rotación del motor utilizando modulación por ancho de pulso (PWM) en el pin de control, como se muestra a continuación.

Nota:

- Abre el archivo `4.1_motor_turn_pwm.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, PWM
import time

# Create PWM objects representing the motor control pins and set their frequency to 1000␣
↪Hz
motor1A = PWM(Pin(13, Pin.OUT))
motor2A = PWM(Pin(14, Pin.OUT))
motor1A.freq(500)
motor2A.freq(500)

# Enter an infinite loop
while True:
    # Rotate the motor forward by gradually increasing the power on the motor1A pin
    for power in range(0, 1023, 20):
        motor1A.duty(power)
```

(continué en la próxima página)

(proviene de la página anterior)

```
        motor2A.duty(0)
        time.sleep(0.1)
    # Decreasing the power on the motor1A pin
    for power in range(1023, 0, -20):
        motor1A.duty(power)
        motor2A.duty(0)
        time.sleep(0.1)
    # Rotate the motor in the opposite direction by gradually increasing the power on␣
↪the motor2A pin
    for power in range(0, 1023, 20):
        motor1A.duty(0)
        motor2A.duty(power)
        time.sleep(0.1)
    # Decreasing the power on the motor2A pin
    for power in range(1023, 0, -20):
        motor1A.duty(0)
        motor2A.duty(power)
        time.sleep(0.1)
```

A diferencia del script anterior, aquí el motor es controlado por señales PWM con una frecuencia de 1000 Hz, lo cual determina la velocidad del motor.

- El código utiliza un bucle `while True` para funcionar continuamente. Dentro del bucle, hay cuatro bucles `for` que controlan los motores en secuencia.
- Los primeros dos bucles `for` aumentan y disminuyen la velocidad de IN1 manteniendo IN2 a 0 de velocidad.
- Los siguientes dos bucles `for` aumentan y disminuyen la velocidad de IN2 manteniendo IN1 a 0 de velocidad.
- La función `range` en cada bucle `for` produce una serie de números que sirve como el ciclo de trabajo de la señal PWM. Esto se envía luego a IN1 o IN2 a través del método `duty`. El ciclo de trabajo determina el porcentaje de tiempo que la señal PWM está alta, lo que a su vez determina el voltaje promedio aplicado al motor, y por lo tanto la velocidad del motor.
- La función `time.sleep` se utiliza para introducir un retraso de 0.1 segundos entre cada paso en la secuencia, lo que permite que el motor cambie de velocidad gradualmente, en lugar de saltar de una velocidad a otra instantáneamente.

4.17 4.2 Bombeo

En este intrigante proyecto, exploraremos cómo controlar una bomba de agua usando el L293D.

En el ámbito del control de bombas de agua, las cosas son un poco más sencillas en comparación con el control de otros motores. La belleza de este proyecto radica en su simplicidad: no hay necesidad de preocuparse por la dirección de rotación. Nuestro objetivo principal es activar con éxito la bomba de agua y mantenerla funcionando.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

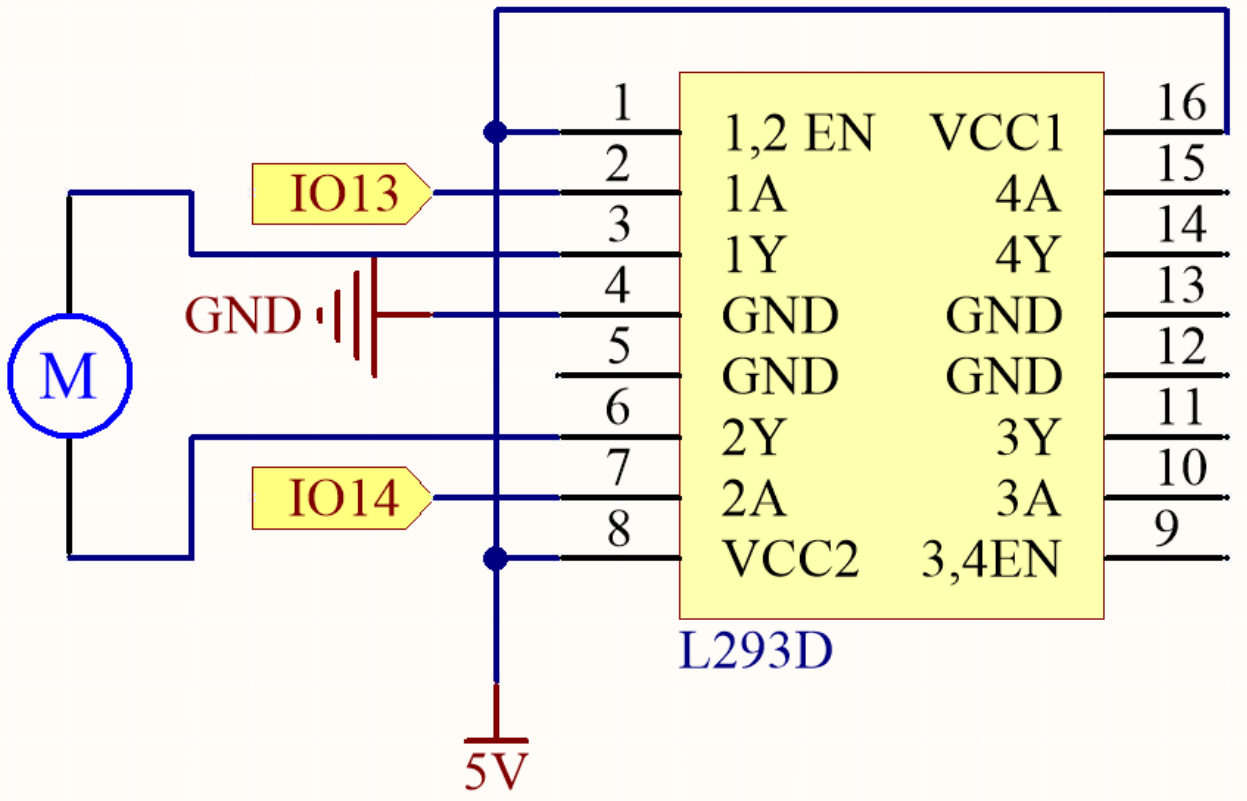
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Bomba Centrífuga	-
L293D	-

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

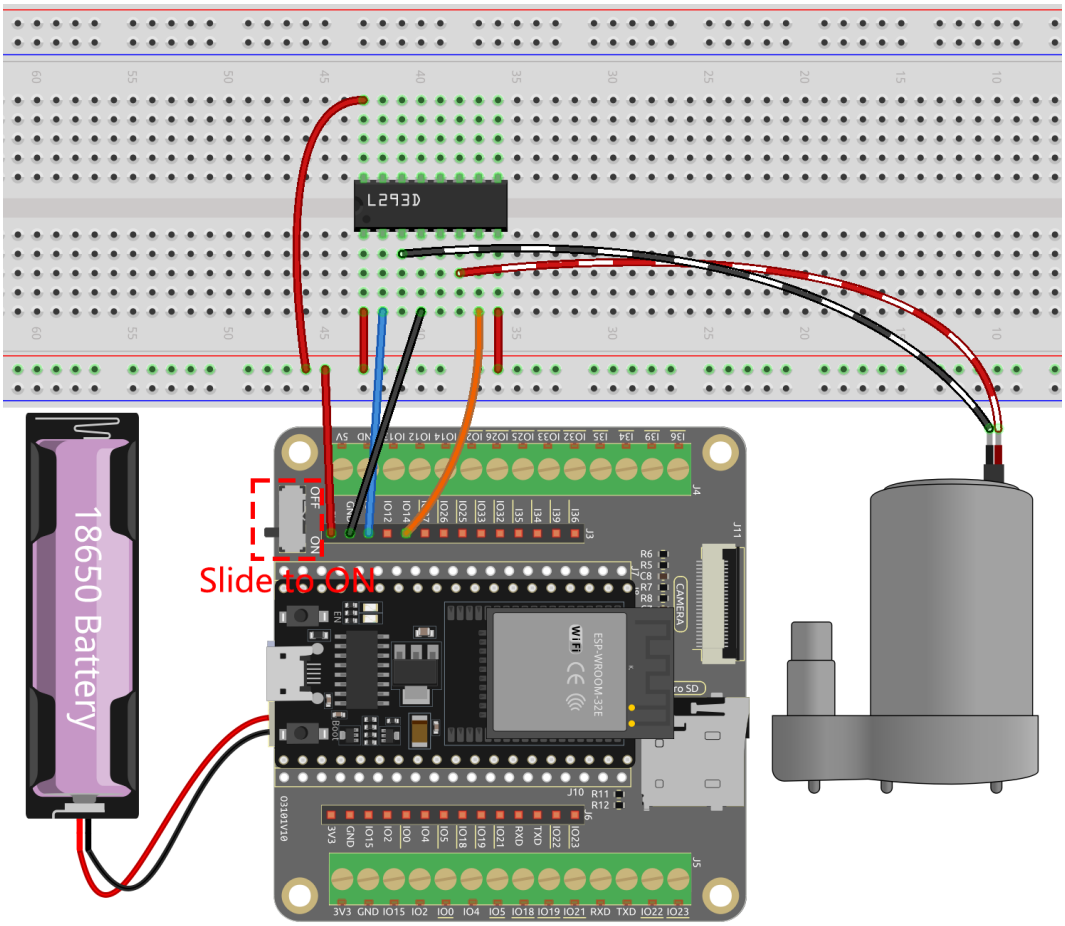
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Conexión

Nota: Se recomienda aquí insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de la batería.



Código

Nota:

- Abre el archivo `4.2_pumping.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Crear objetos Pin que representan los pines de control del motor y configurarlos en␣
↪modo de salida
motor1A = machine.Pin(13, machine.Pin.OUT)
motor2A = machine.Pin(14, machine.Pin.OUT)

# Definir una función para rotar la bomba
def rotate():
    motor1A.value(1)
    motor2A.value(0)

# Definir una función para detener la bomba
```

(continué en la próxima página)

(proviene de la página anterior)

```
def stop():
    motor1A.value(0)
    motor2A.value(0)

try:
    while True:
        rotate()  # Rotar el motor en sentido horario
        time.sleep(5)  # Pausar durante 5 segundos
        stop()  # Detener el motor
        time.sleep(2)

except KeyboardInterrupt:
    stop()  # Detener el motor cuando se captura KeyboardInterrupt
```

Durante la ejecución del script, verás la bomba funcionando y el agua saliendo del tubo, luego se detendrá durante 2 segundos antes de comenzar a funcionar de nuevo.

4.18 4.3 Servo Oscilante

Un Servo es un tipo de dispositivo basado en la posición conocido por su capacidad para mantener ángulos específicos y proporcionar una rotación precisa. Esto lo hace altamente deseable para sistemas de control que demandan ajustes de ángulo consistentes. No es sorprendente que los Servos se hayan utilizado ampliamente en juguetes controlados remotamente de alta gama, desde modelos de aviones hasta réplicas de submarinos y robots controlados remotamente sofisticados.

En esta aventura intrigante, nos desafiaremos a manipular el Servo de una manera única: ¡haciéndolo balancearse! Este proyecto ofrece una brillante oportunidad para profundizar en la dinámica de los Servos, afilando tus habilidades en sistemas de control precisos y ofreciendo una comprensión más profunda de su operación.

¿Estás listo para hacer bailar al Servo a tu ritmo? ¡Embarquémonos en este emocionante viaje!

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

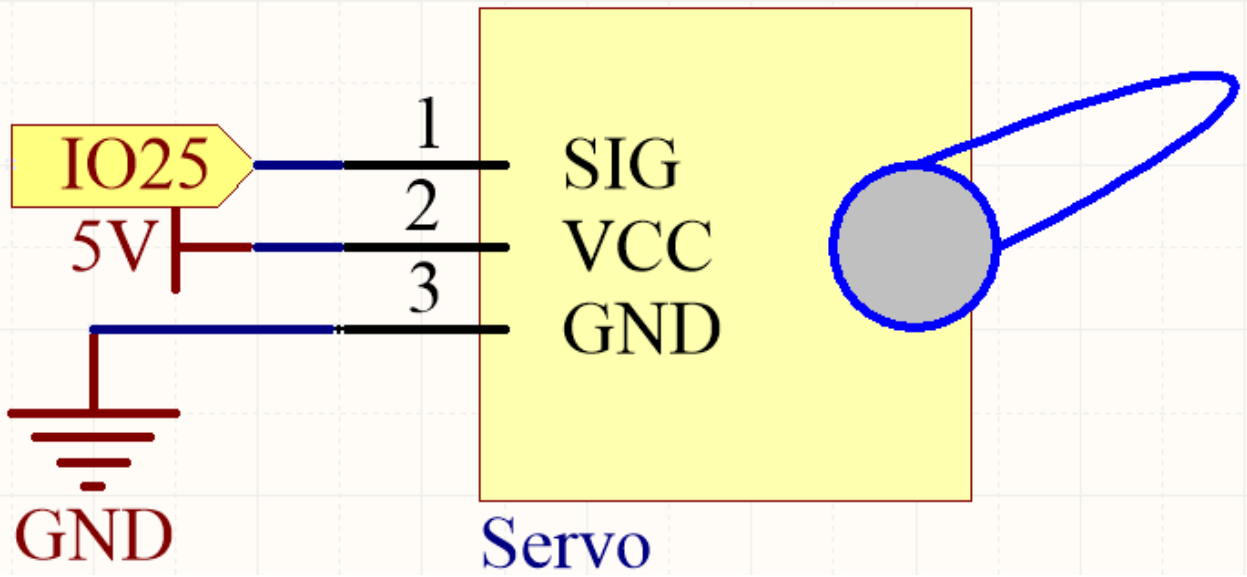
INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Servo	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

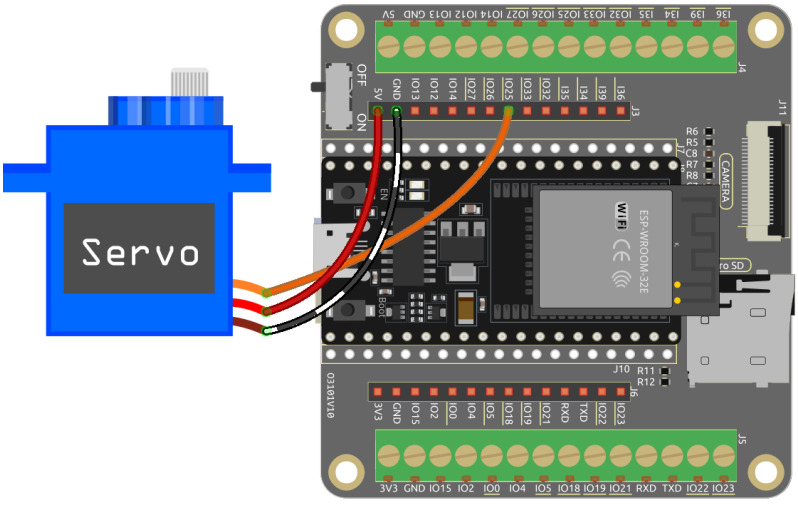
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Conexión

- El cable naranja es de señal y se conecta a IO25.
- El cable rojo es VCC y se conecta a 5V.
- El cable marrón es GND y se conecta a GND.



Código

Nota:

- Abre el archivo `4.3_swinging_servo.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para

ejecutarlo.

- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Create a PWM (Pulse Width Modulation) object on Pin 25
servo = machine.PWM(machine.Pin(25))

# Set the frequency of the PWM signal to 50 Hz, common for servos
servo.freq(50)

# Define a function for interval mapping
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Define a function to write an angle to the servo
def servo_write(pin, angle):

    pulse_width = interval_mapping(angle, 0, 180, 0.5, 2.5) # Calculate the pulse width
    duty = int(interval_mapping(pulse_width, 0, 20, 0, 1023))     # Calculate the duty␣
↪cycle
    pin.duty(duty) # Set the duty cycle of the PWM signal

# Create an infinite loop
while True:
    # Loop through angles from 0 to 180 degrees
    for angle in range(180):
        servo_write(servo, angle)
        time.sleep_ms(20)

    # Loop through angles from 180 to 0 degrees in reverse
    for angle in range(180, -1, -1):
        servo_write(servo, angle)
        time.sleep_ms(20)
```

Al ejecutar este código, el servo oscilará continuamente de ida y vuelta entre 0 y 180 grados.

¿Cómo funciona?

1. Importar las bibliotecas necesarias: `machine` para controlar el hardware del microcontrolador, y `time` para agregar retrasos.

   ```
   import machine
   import time
   ```

2. Crear un objeto PWM (Modulación por Ancho de Pulso) en el Pin 25 y establecer su frecuencia a 50 Hz, que es común para servo.

   ```
   # Create a PWM (Pulse Width Modulation) object on Pin 25
   servo = machine.PWM(machine.Pin(25))

   # Set the frequency of the PWM signal to 50 Hz, common for servos
   servo.freq(50)
   ```

3. Definir una función `interval_mapping` para mapear valores de un rango a otro. Esto se usará para convertir el ángulo al ancho de pulso y ciclo de trabajo apropiados.

```
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
```

4. Definir una función `servo_write` que toma un objeto PWM y un ángulo como entradas. Calcula el ancho de pulso y ciclo de trabajo basado en el ángulo dado, y luego establece la salida PWM en consecuencia.

```
def servo_write(pin, angle):

    pulse_width = interval_mapping(angle, 0, 180, 0.5, 2.5) # Calculate the␣
↪pulse width
    duty = int(interval_mapping(pulse_width, 0, 20, 0, 1023))     #␣
↪Calculate the duty cycle
    pin.duty(duty) # Set the duty cycle of the PWM signal
```

   - En esta función, se llama a `interval_mapping()` para mapear el rango de ángulo 0 ~ 180 al rango de ancho de pulso 0.5 ~ 2.5ms.
   - ¿Por qué es 0.5~2.5? Esto está determinado por el modo de trabajo del *Servo*.
   - Luego, convertir el ancho de pulso de período a ciclo de trabajo.
   - Dado que `duty()` no puede tener decimales cuando se usa (el valor no puede ser de tipo flotante), usamos `int()` para forzar que el ciclo de trabajo se convierta a tipo entero.

5. Crear un bucle infinito con dos bucles anidados.

```
while True:
    # Loop through angles from 0 to 180 degrees
    for angle in range(180):
        servo_write(servo, angle)
        time.sleep_ms(20)

    # Loop through angles from 180 to 0 degrees in reverse
    for angle in range(180, -1, -1):
        servo_write(servo, angle)
        time.sleep_ms(20)
```

   - El primer bucle anidado itera a través de ángulos de 0 a 180 grados, y el segundo bucle anidado itera a través de ángulos de 180 a 0 grados en reversa.
   - En cada iteración, se llama a la función `servo_write` con el ángulo actual, y se añade un retraso de 20 milisegundos.

5. Sensores

4.19 5.1 Lectura del Valor del Botón

En este proyecto interactivo, nos adentraremos en el ámbito de los controles por botón y la manipulación de LEDs.

El concepto es sencillo pero efectivo. Leeremos el estado de un botón. Cuando el botón se presiona, registra un nivel de voltaje alto, o "estado alto". Esta acción entonces hará que un LED se ilumine.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Botón	

Pines Disponibles

- **Pines Disponibles**

  Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Uso Condicional de Pines (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down integradas, por lo que no se requieren resistencias externas cuando **se usan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Pull-up con una resistencia de 47K por defecto da un valor alto.
IO27, IO26, IO33	Pull-up con una resistencia de 4.7K por defecto da un valor alto.
IO32	Pull-down con una resistencia de 1K por defecto da un valor bajo.

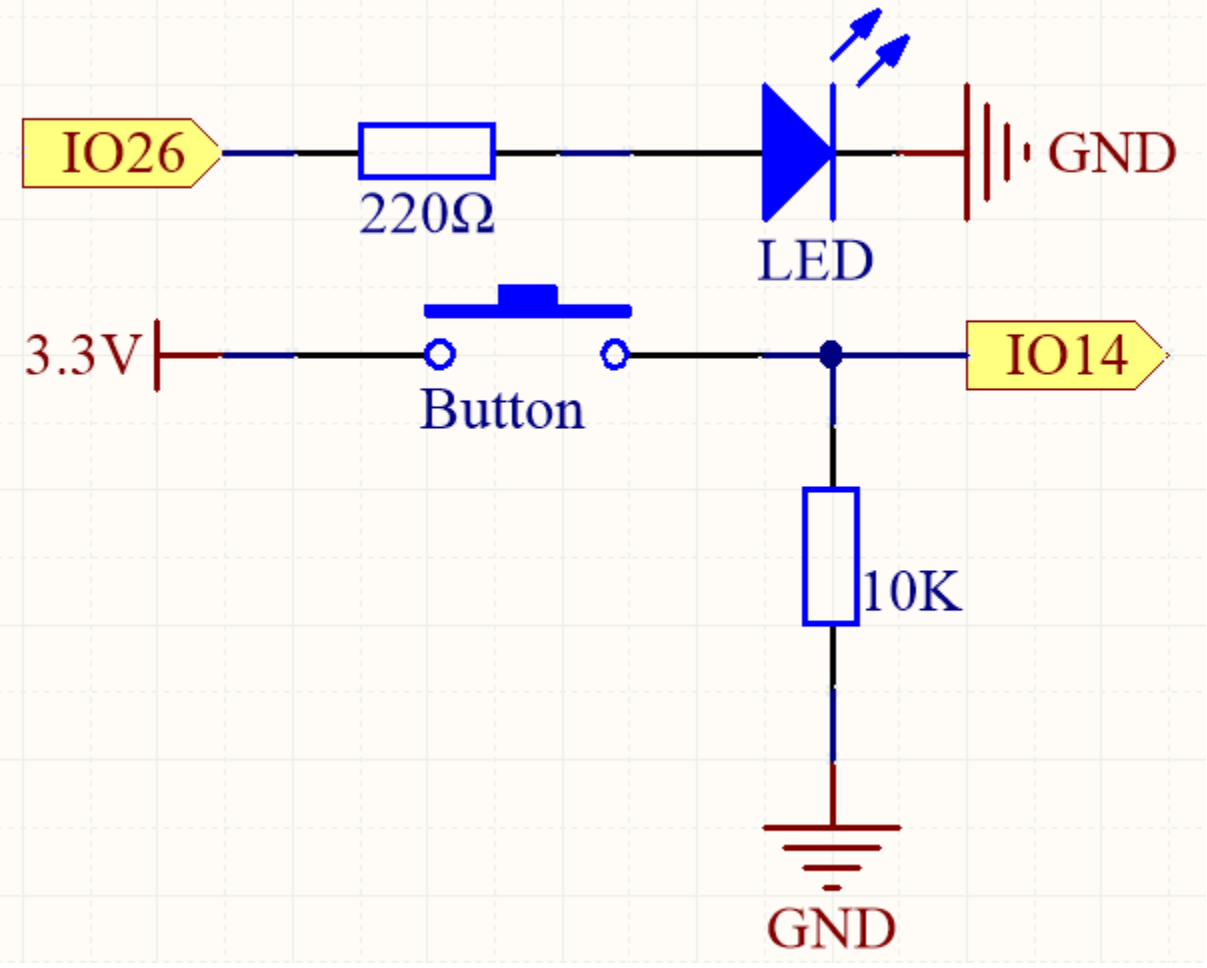
- **Pines de Arranque (Entrada)**

  Los pines de arranque son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reseteo por encendido).

Pines de Arranque	IO5, IO0, IO2, IO12, IO15

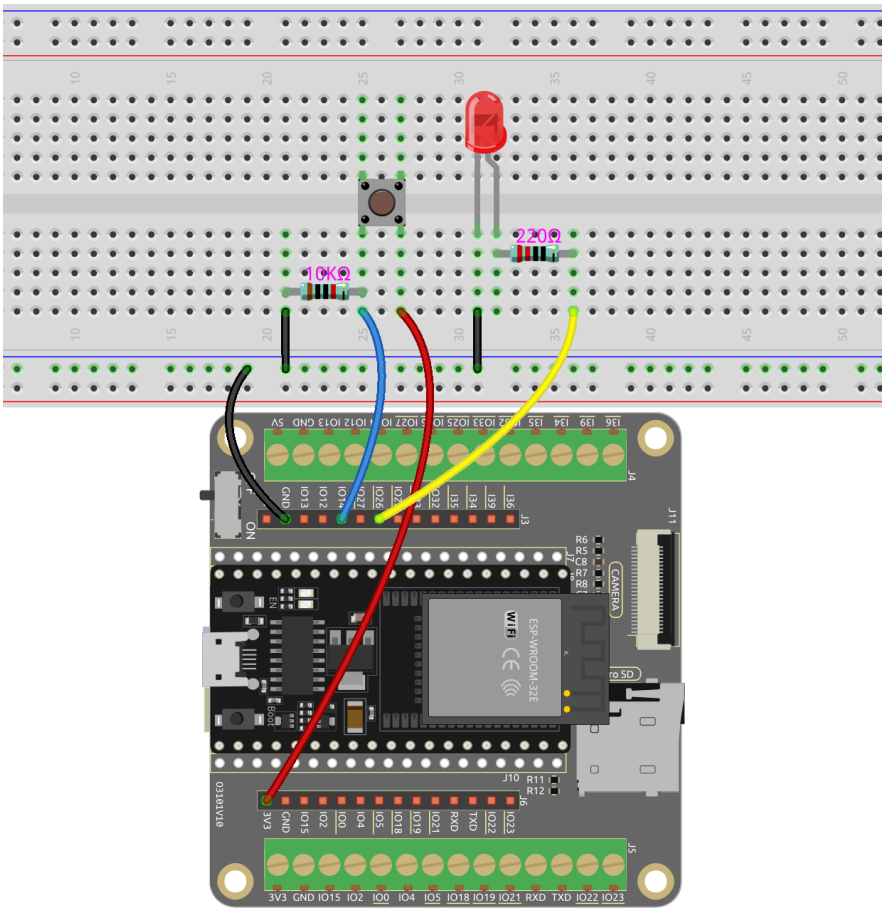
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas utilizar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor refiérete a la sección *Pines de Estrapeo*.

Esquemático



Para asegurar una funcionalidad adecuada, conecta un lado del pin del botón a 3.3V y el otro lado a IO14. Cuando el botón se presiona, IO14 se establece en alto, causando que el LED se ilumine. Cuando el botón se suelta, IO14 volverá a su estado suspendido, que puede ser alto o bajo. Para asegurar un nivel bajo estable cuando el botón no está presionado, IO14 debe conectarse a GND a través de una resistencia de pull-down de 10K.

Cableado



Nota: Un botón de cuatro pines está diseñado en forma de H. Cuando el botón no está presionado, los pines izquierdo y derecho están desconectados, y la corriente no puede fluir entre ellos. Sin embargo, cuando el botón se presiona, los pines izquierdo y derecho se conectan, creando un camino para que fluya la corriente.

Código

Nota:

- Abre el archivo `5.1_read_button_value.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

button = machine.Pin(14, machine.Pin.IN) # Button pin
led = machine.Pin(26, machine.Pin.OUT) # LED pin


while True:
```

(continué en la próxima página)

(proviene de la página anterior)

```
    # If the button is pressed by reading its value
    if button.value() == 1:
        # Turn on the LED by setting its value to 1
        led.value(1)
#         time.sleep(0.5)
    else:
        # Turn off the LED
        led.value(0)
```

Durante la ejecución del script, el LED se ilumina cuando presionas el botón y se apaga cuando lo sueltas.

4.20 5.2 ¡Inclínalo!

El interruptor de inclinación es un dispositivo simple pero efectivo de 2 pines que contiene una bola metálica en su centro. Cuando el interruptor está en posición vertical, los dos pines están eléctricamente conectados, permitiendo que la corriente fluya. Sin embargo, cuando el interruptor se inclina o se inclina a cierto ángulo, la bola metálica se mueve y rompe la conexión eléctrica entre los pines.

En este proyecto, utilizaremos el interruptor de inclinación para controlar la iluminación de un LED. Posicionando el interruptor de manera que se active la acción de inclinación, podemos alternar el LED encendido y apagado basado en la orientación del interruptor.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Interruptor de Inclinación	-

Pines Disponibles

- **Pines Disponibles**

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Pines de Uso Condicional (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down integradas, por lo que no se requieren resistencias externas cuando **se usan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Pull-up con una resistencia de 47K por defecto el valor es alto.
IO27, IO26, IO33	Pull-up con una resistencia de 4.7K por defecto el valor es alto.
IO32	Pull-down con una resistencia de 1K por defecto el valor es bajo.

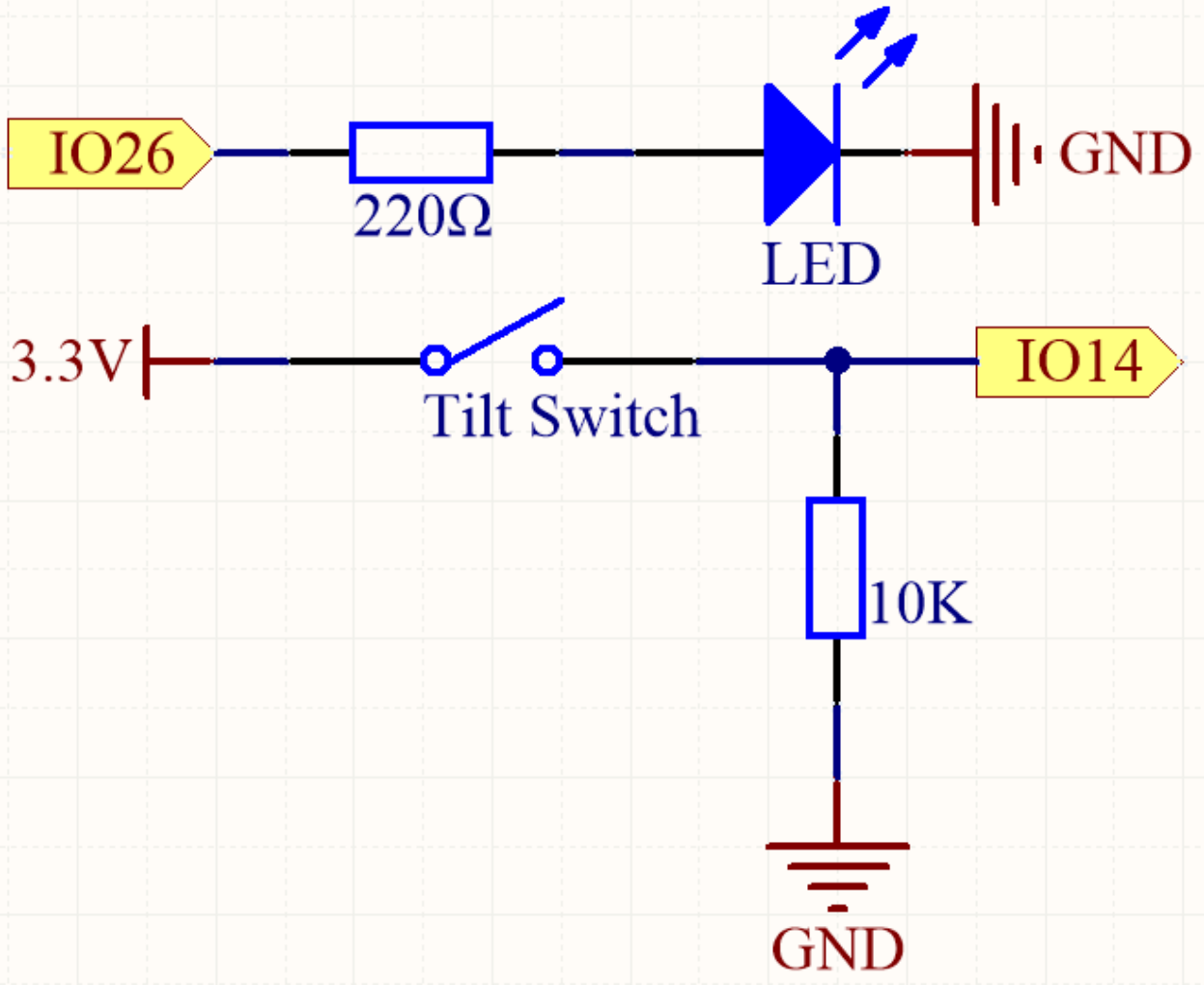
- **Pines de Configuración (Entrada)**

  Los pines de configuración son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reseteo al encender).

Pines de Configuración	IO5, IO0, IO2, IO12, IO15

  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

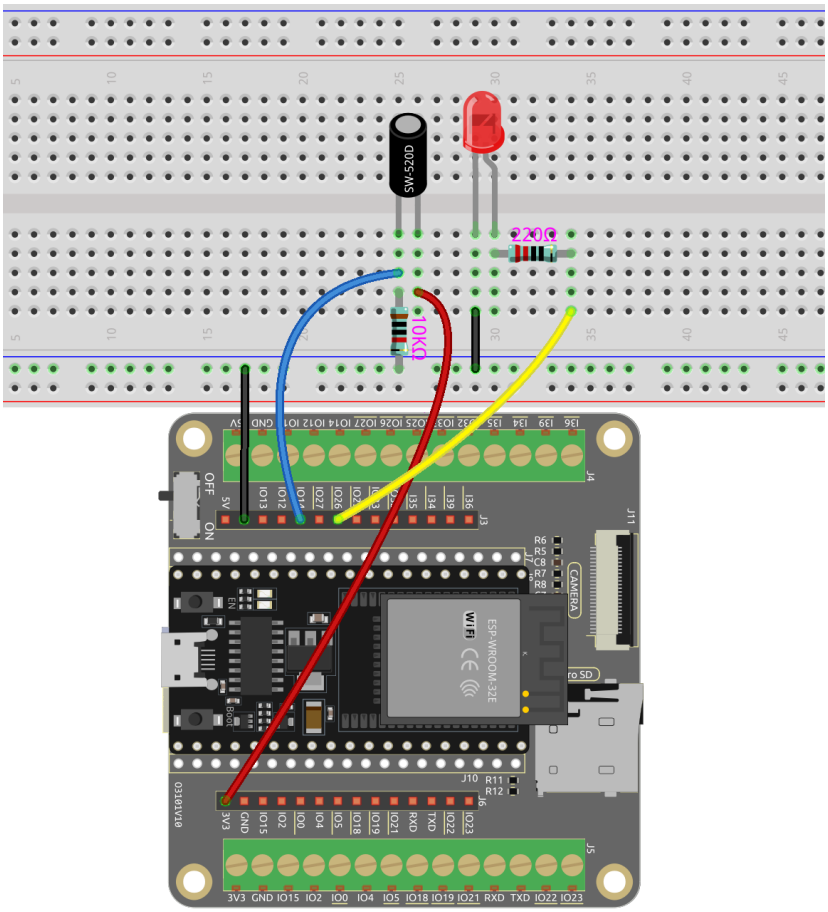
Esquemático



Cuando el interruptor de inclinación está en posición vertical, IO14 se establecerá en alto, resultando en que el LED se ilumine. Por el contrario, cuando el interruptor de inclinación se inclina, IO14 se establecerá en bajo, causando que el LED se apague.

El propósito de la resistencia de 10K es mantener un estado bajo estable para IO14 cuando el interruptor de inclinación está en posición inclinada.

Conexión



Código

Nota:

- Abre el archivo `5.2_tilt_switch.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

switch = machine.Pin(14, machine.Pin.IN) # Tilt switch pin
led = machine.Pin(26, machine.Pin.OUT) # LED pin

while True:
    # Check if the switch is tilted by reading its value
    if switch.value() == 1:
        # Turn on the LED by setting its value to 1
        led.value(1)
    else:
        # Turn off the LED
        led.value(0)
```

Cuando el script está en ejecución, el LED se encenderá cuando el interruptor esté vertical y se apagará cuando el interruptor se incline.

4.21 5.3 Detección de Obstáculos

Este módulo se instala comúnmente en coches y robots para determinar la existencia de obstáculos adelante. También se utiliza ampliamente en dispositivos portátiles, grifos de agua y otros.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Evitación de Obstáculos	

Pines Disponibles

- **Pines Disponibles**

  Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

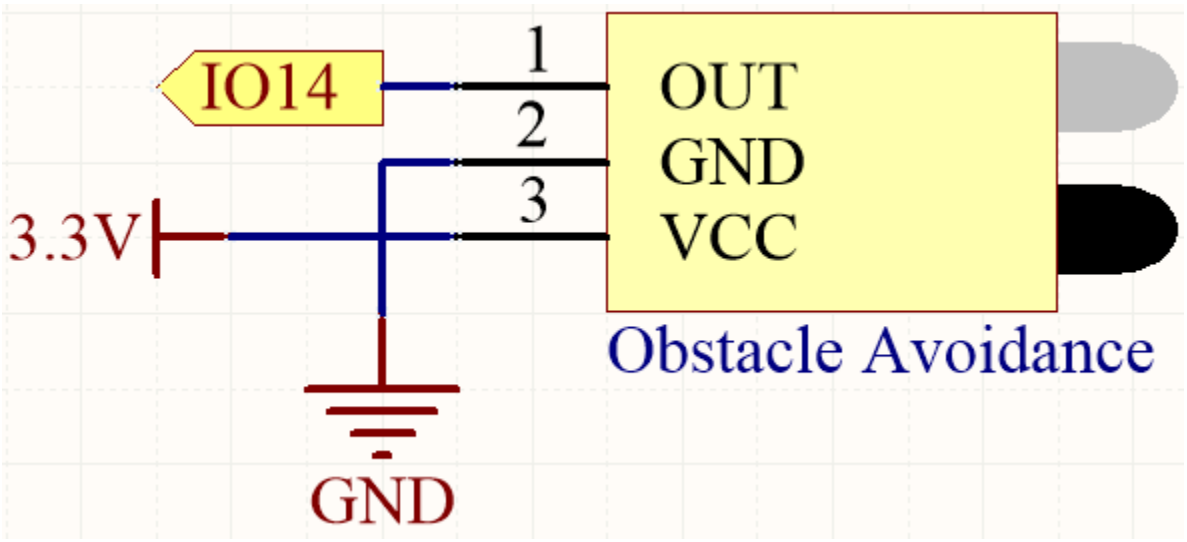
- **Pines de Arranque (Entrada)**

  Los pines de arranque son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reseteo por encendido).

Pines de Arranque	IO5, IO0, IO2, IO12, IO15

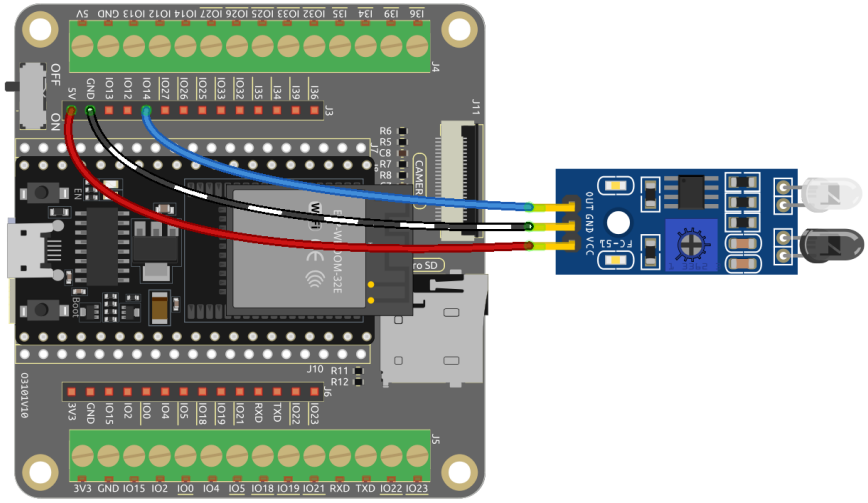
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas utilizar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor refiérete a la sección *Pines de Estrapeo*.

Esquemático



Cuando el módulo de evitación de obstáculos no detecta ningún obstáculo, IO14 devuelve un nivel alto. Sin embargo, cuando detecta un obstáculo, devuelve un nivel bajo. Puedes ajustar el potenciómetro azul para modificar la distancia de detección de este módulo.

Cableado



Código

Nota:

- Abre el archivo `5.3_avoid.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

ir_avoid = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP) # pin del módulo de␣
↪evitación
```

(continué en la próxima página)

(proviene de la página anterior)

```
while True:

    # Imprime los valores del módulo de evitación de obstáculos
    print(ir_avoid.value())
    time.sleep(0.1)
```

Mientras el programa esté en ejecución, si el módulo IR de evitación de obstáculos detecta un obstáculo frente a él, el valor «0» se mostrará en el Monitor Serial, de lo contrario, se mostrará el valor «1».

4.22 5.4 Detección de Líneas

El módulo de seguimiento de líneas se utiliza para detectar la presencia de áreas negras en el suelo, como líneas negras pegadas con cinta aislante.

Su emisor emite luz infrarroja adecuada hacia el suelo, la cual es absorbida en mayor medida y reflejada débilmente por superficies negras. Lo contrario ocurre con las superficies blancas. Si se detecta luz reflejada, se indica que el suelo es blanco actualmente. Si no se detecta, se indica como negro.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Seguimiento de Línea	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

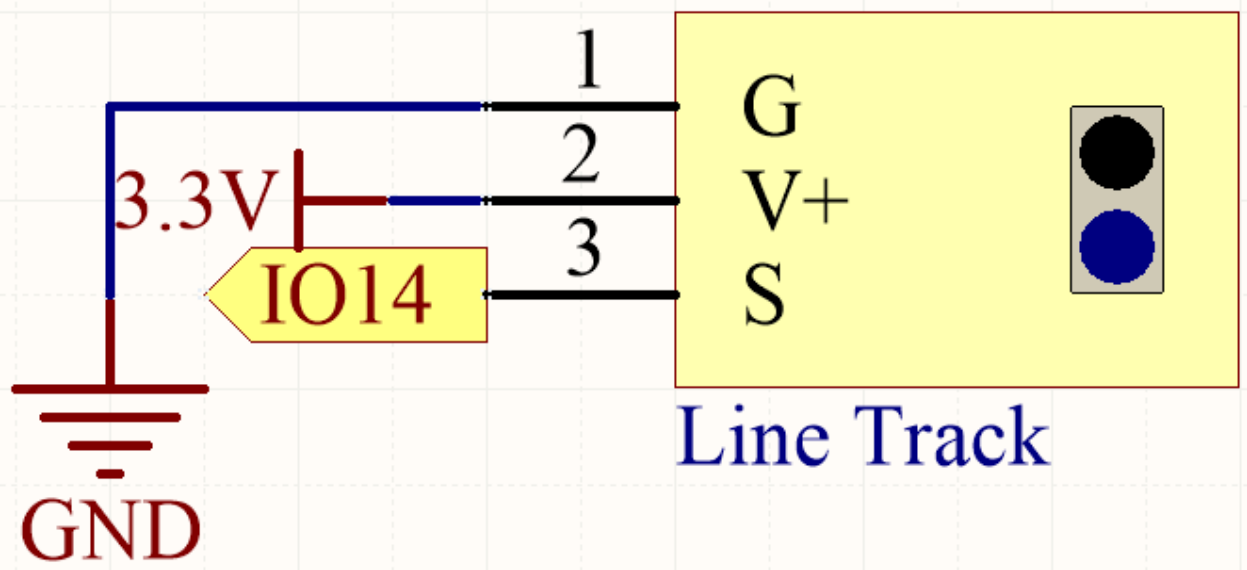
- **Pines de Strapping (Entrada)**

Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reinicio por encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

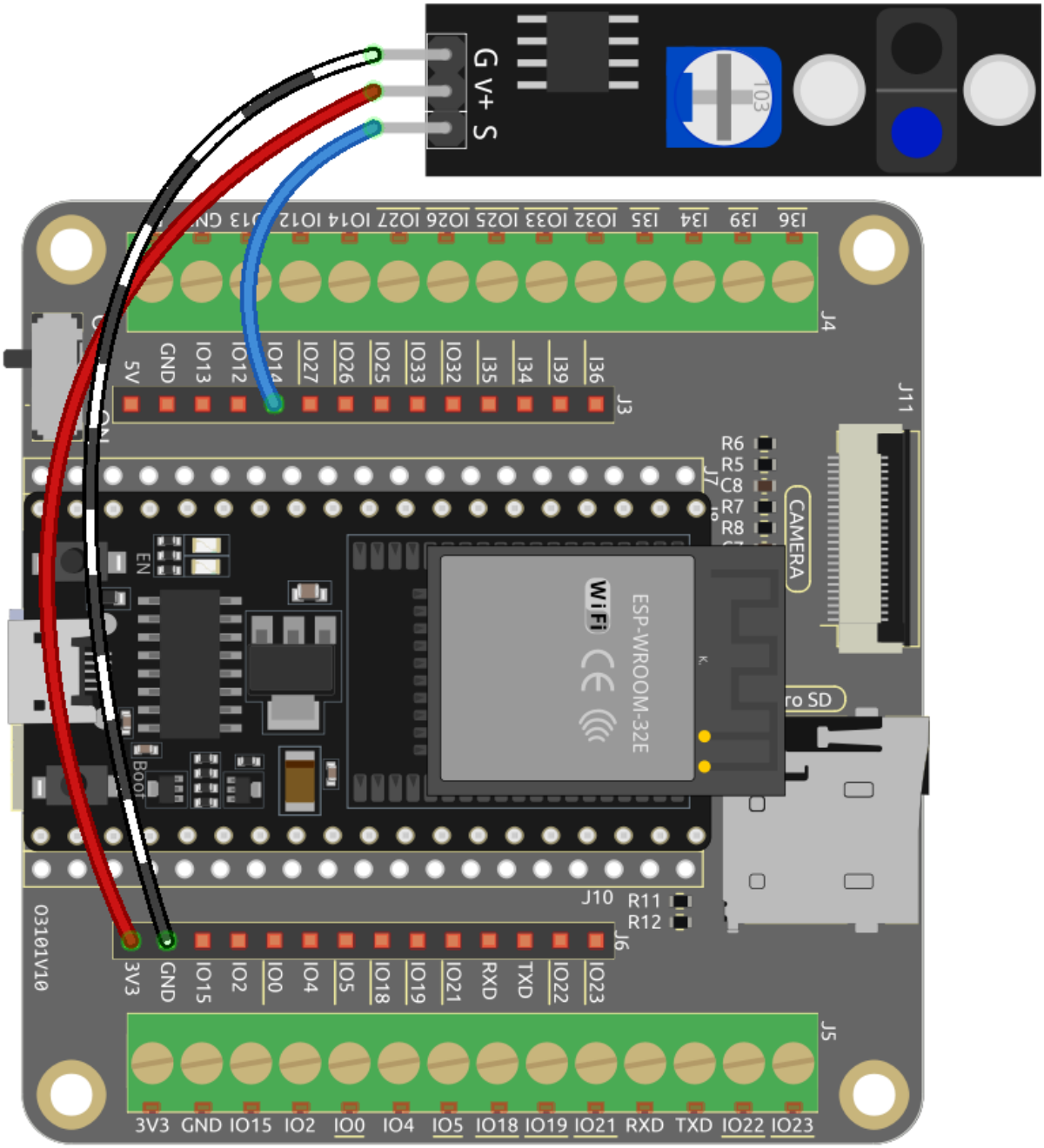
Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

Esquemático



Cuando el módulo de seguimiento de línea detecta una línea negra, IO14 devuelve un nivel alto. Por otro lado, cuando detecta una línea blanca, IO14 devuelve un nivel bajo. Puedes ajustar el potenciómetro azul para modificar la sensibilidad de detección de este módulo.

Conexión



Código

Nota:

- Abre el archivo `5.4_detect_the_line.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
```

(continué en la próxima página)

(proviene de la página anterior)

```
import time

# Create a pin object named greyscale, set pin number 14 as input
line = machine.Pin(14, machine.Pin.IN)


while True:
    # Check if the value is 1 (black)
    if line.value() == 1:
        # Print "black"
        print("black")
        time.sleep(0.5)
    # If the value is not 1 (it's 0, which means white)
    else :
        # Print "white"
        print("white")
        time.sleep(0.5)
```

Cuando el módulo de seguimiento de línea detecta que hay una línea negra, aparece «negro» en la consola; de lo contrario, se muestra «blanco».

4.23 5.5 Detección de Movimiento Humano

El sensor infrarrojo pasivo (sensor PIR) es un sensor común que puede medir la luz infrarroja (IR) emitida por objetos en su campo de visión. En términos simples, recibe la radiación infrarroja emitida por el cuerpo, detectando así el movimiento de personas y otros animales. Más específicamente, le indica a la placa de control principal que alguien ha entrado a su habitación.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puede comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Módulo Sensor de Movimiento PIR	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Nota: El pin IO32 no se puede usar **como pin de entrada** en este proyecto porque está internamente conectado a una resistencia de pull-down de 1K, lo que establece su valor predeterminado en 0.

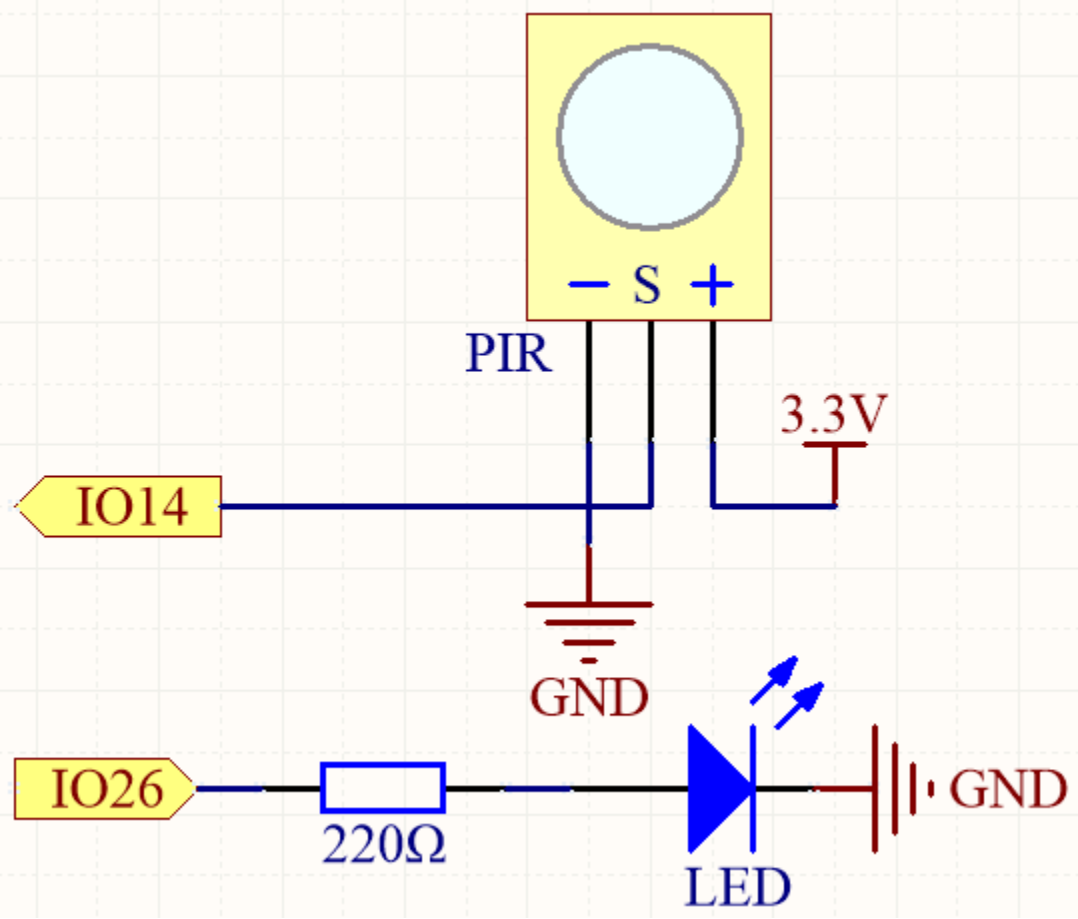
- **Pines de Strapping (Entrada)**

  Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reseteo por encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

  Generalmente, **no se recomienda usarlos como pines de entrada**. Si desea usar estos pines, considere el impacto potencial en el proceso de arranque. Para más detalles, por favor refiérase a la sección *Pines de Estrapeo*.

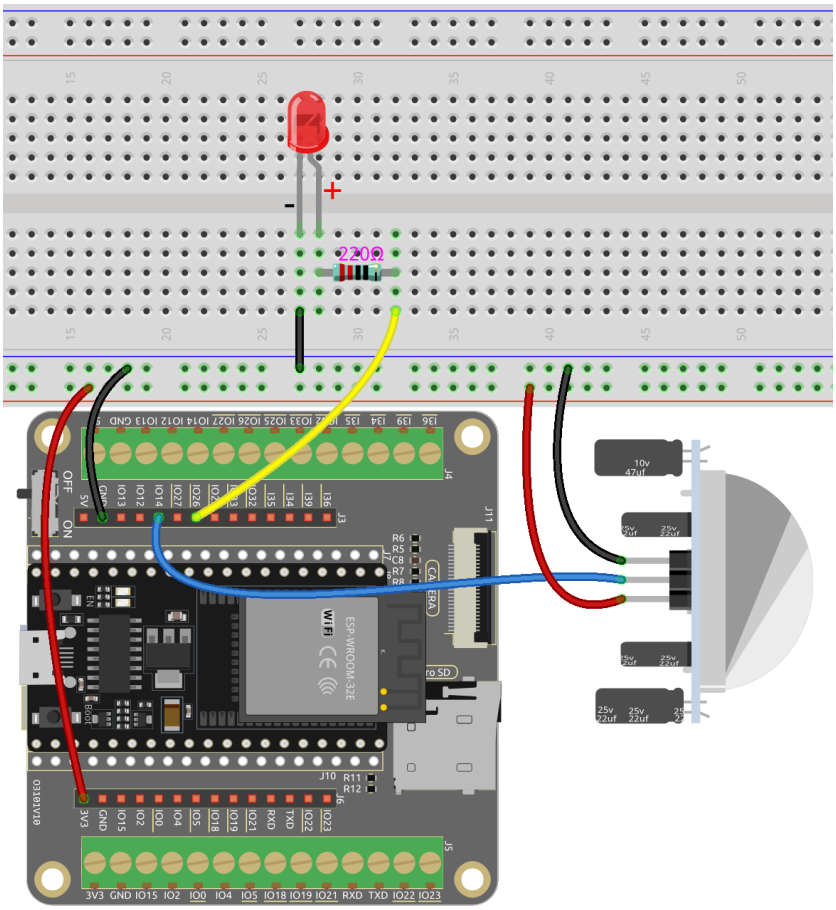
Esquemático



Cuando el módulo PIR detecta movimiento, IO14 se activará, y el LED se encenderá. De lo contrario, cuando no se detecta movimiento, IO14 estará bajo, y el LED se apagará.

Nota: El módulo PIR tiene dos potenciómetros: uno ajusta la sensibilidad, el otro ajusta la distancia de detección. Para que el módulo PIR funcione mejor, necesita girar ambos en sentido antihorario hasta el final.

Conexión



Código

Nota:

- Abra el archivo `5.5_detect_human_movement.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copie y pegue el código en Thonny. Luego, haga clic en «Ejecutar script actual» o presione F5 para ejecutarlo.
- Asegúrese de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Define pins
PIR_PIN = 14    # PIR sensor
LED_PIN = 26    # LED

# Initialize the PIR sensor pin as an input pin
pir_sensor = machine.Pin(PIR_PIN, machine.Pin.IN, machine.Pin.PULL_DOWN)
# Initialize the LED pin as an output pin
led = machine.Pin(LED_PIN, machine.Pin.OUT)

# Global flag to indicate motion detected
motion_detected_flag = False
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Function to handle the interrupt
def motion_detected(pin):
    global motion_detected_flag
    print("Motion detected!")
    motion_detected_flag = True

# Attach the interrupt to the PIR sensor pin
pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)

# Main loop
while True:
    if motion_detected_flag:
        led.value(1)  # Turn on the LED
        time.sleep(5)  # Keep the LED on for 5 seconds
        led.value(0)  # Turn off the LED
        motion_detected_flag = False
```

Cuando el script esté ejecutándose, el LED se encenderá durante 5 segundos y luego se apagará cuando el módulo PIR detecte a alguien pasando.

Nota: El módulo PIR tiene dos potenciómetros: uno ajusta la sensibilidad, el otro ajusta la distancia de detección. Para que el módulo PIR funcione mejor, necesita girar ambos en sentido antihorario hasta el final.

¿Cómo funciona?

Este código establece un sistema simple de detección de movimiento usando un sensor PIR y un LED. Cuando se detecta movimiento, el LED se encenderá durante 5 segundos.

Aquí hay un desglose del código:

1. Definir la función del manejador de interrupción que se ejecutará cuando se detecte movimiento:

```
def motion_detected(pin):
    global motion_detected_flag
    print("Motion detected!")
    motion_detected_flag = True
```

2. Adjuntar la interrupción al pin del sensor PIR, con el disparador establecido en «ascendente» (es decir, cuando el pin pasa de un voltaje bajo a alto):

```
pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)
```

   Esto configura una interrupción en el pin `pir_sensor`, que está conectado al sensor de movimiento PIR.

   Aquí hay una explicación detallada de los parámetros:

   - `trigger=machine.Pin.IRQ_RISING`: Este parámetro establece la condición de disparo para la interrupción. En este caso, la interrupción se disparará en un borde ascendente. Un borde ascendente es cuando el voltaje en el pin cambia de un estado bajo (0V) a un estado alto (típicamente 3.3V o 5V, dependiendo de su hardware). Para un sensor de movimiento PIR, cuando se detecta movimiento, el pin de salida generalmente pasa de bajo a alto, haciendo que el borde ascendente sea una condición de disparo adecuada.
   - `handler=motion_detected`: Este parámetro especifica la función que se ejecutará cuando se dispare la interrupción. En este caso, la función `motion_detected` se proporciona como el manejador de interrupción. Esta función será llamada automáticamente cuando se detecte la condición de interrupción (borde ascendente) en el pin `pir_sensor`.

   Entonces, esta línea de código configura el sensor PIR para llamar a la función `motion_detected` siempre que el sensor detecte movimiento, debido a que el pin de salida pasa de un estado bajo a un estado alto.

3. En el bucle principal, si la `motion_detected_flag` se establece en `True`, el LED se encenderá durante 5 segundos y luego se apagará. La bandera luego se restablece a `False` para esperar el próximo evento de movimiento.

```
while True:
    if motion_detected_flag:
        led.value(1)  # Turn on the LED
        time.sleep(5)  # Keep the LED on for 5 seconds
        led.value(0)  # Turn off the LED
        motion_detected_flag = False
```

4.24 5.6 Dos Tipos de Transistores

Este kit está equipado con dos tipos de transistores, S8550 y S8050, siendo el primero PNP y el segundo NPN. Se parecen mucho, y necesitamos verificar cuidadosamente para ver sus etiquetas. Cuando una señal de nivel alto pasa a través de un transistor NPN, se activa. Pero uno PNP necesita una señal de nivel bajo para gestionarlo. Ambos tipos de transistor se usan frecuentemente para interruptores sin contacto, justo como en este experimento.

¡Usemos un LED y un botón para entender cómo usar un transistor!

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Botón	
Transistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Pines de Uso Condicional (Entrada)**

  Los siguientes pines tienen resistencias de pull-up o pull-down integradas, por lo que no se requieren resistencias externas cuando **se usan como pines de entrada**:

Pines de Uso Condicional	Descripción
IO13, IO15, IO2, IO4	Pull-up con una resistencia de 47K por defecto el valor es alto.
IO27, IO26, IO33	Pull-up con una resistencia de 4.7K por defecto el valor es alto.
IO32	Pull-down con una resistencia de 1K por defecto el valor es bajo.

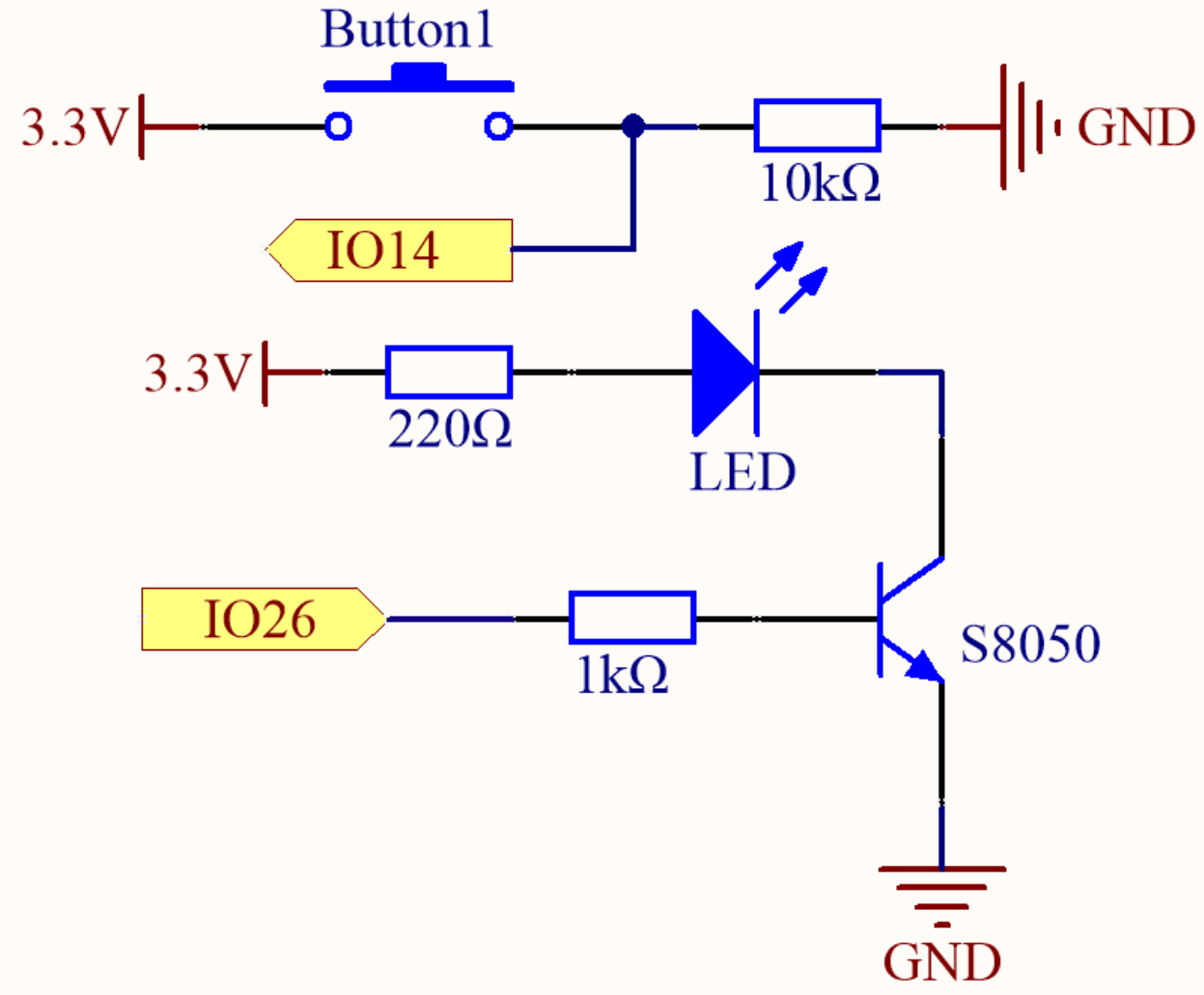
- **Pines de Configuración (Entrada)**

  Los pines de configuración son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reseteo al encender).

Pines de Configuración	IO5, IO0, IO2, IO12, IO15

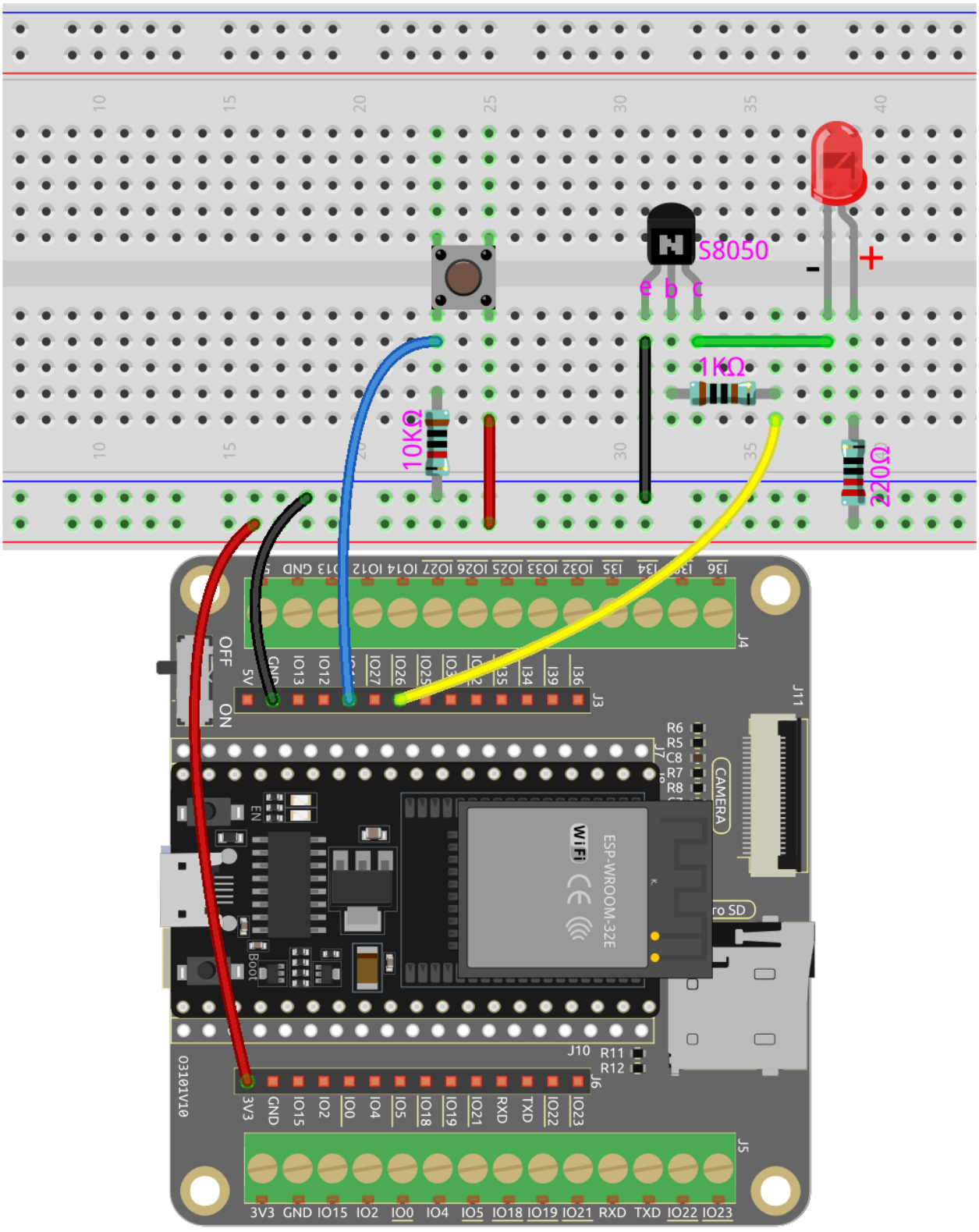
  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor consulta la sección *Pines de Estrapeo*.

Forma de conectar el transistor NPN (S8050)

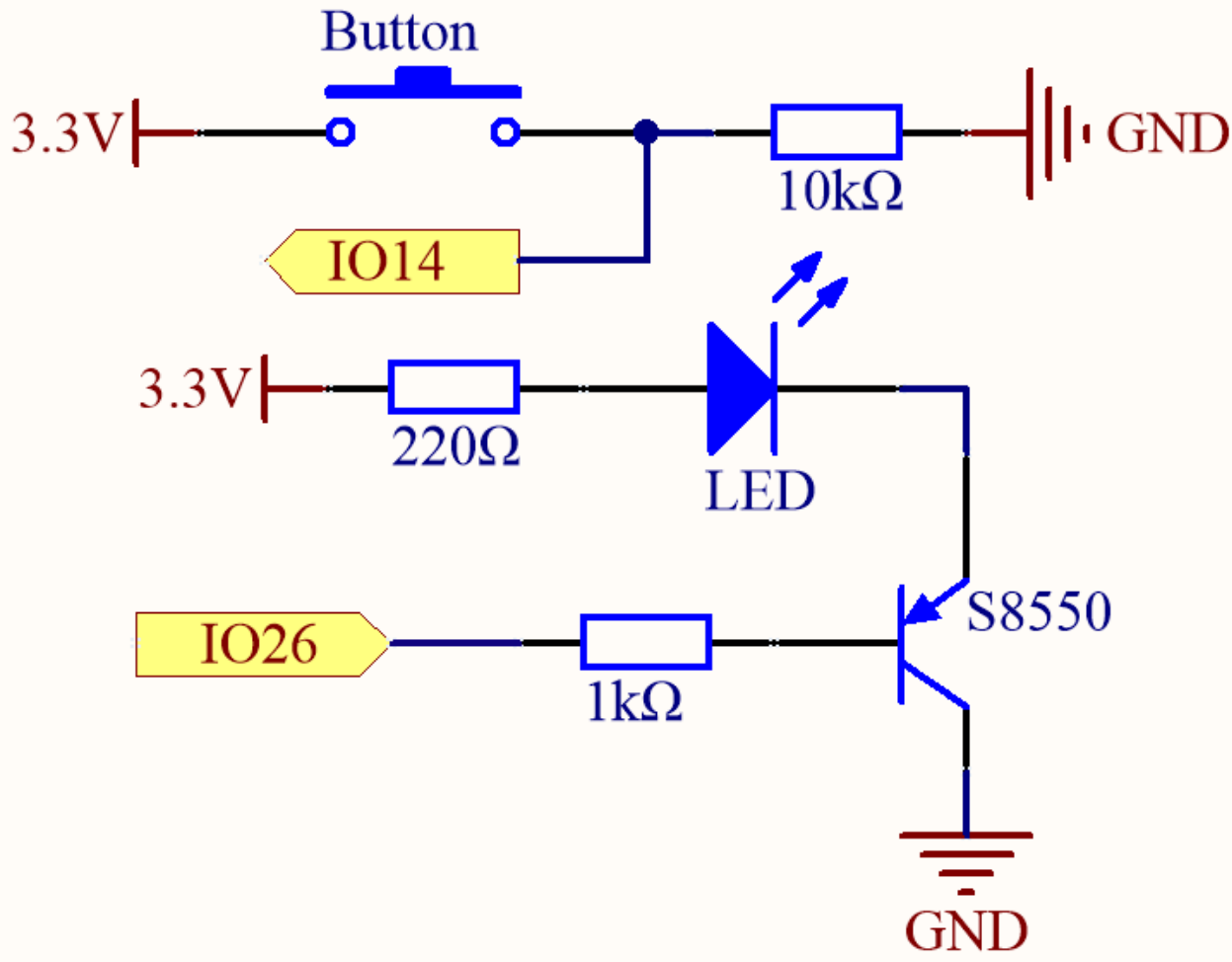


En este circuito, cuando se presiona el botón, IO14 está alto.

Programando IO26 para que emita **alto**, después de una resistencia limitadora de corriente de 1k (para proteger el transistor), se permite que el S8050 (transistor NPN) conduzca, permitiendo así que el LED se ilumine.



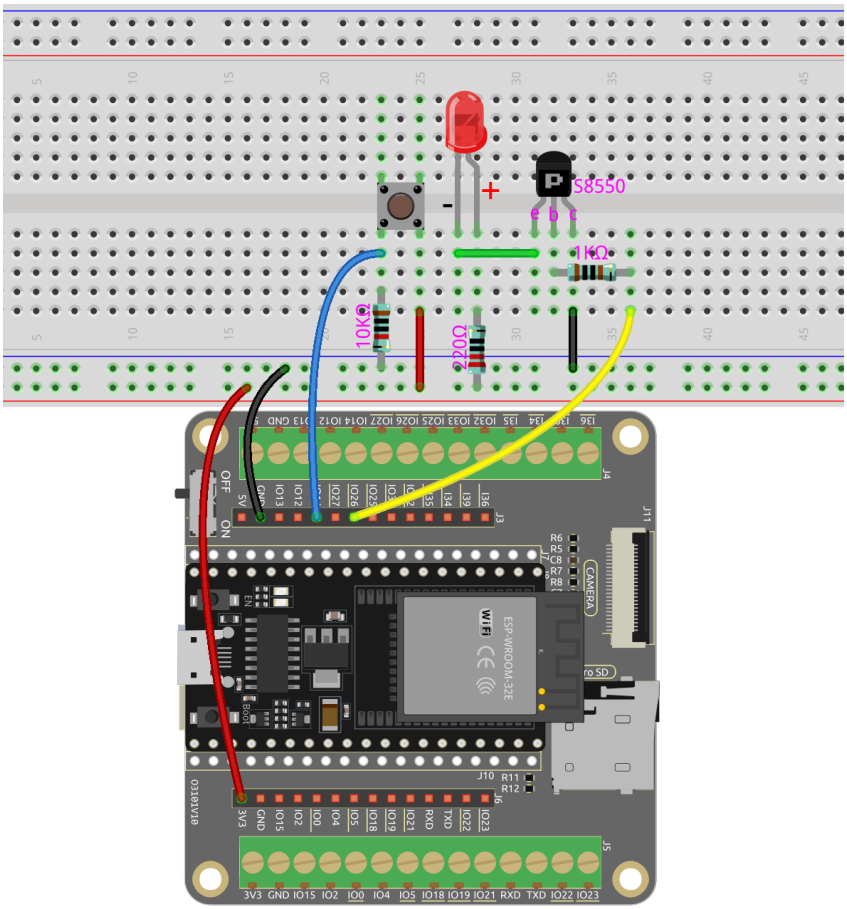
Forma de conectar el transistor PNP(S8550)



En este circuito, IO14 está bajo por defecto y cambiará a alto cuando se presione el botón.

Programando IO26 para que emita **bajo**, después de una resistencia limitadora de corriente de 1k (para proteger el transistor), se permite que el S8550 (transistor PNP) conduzca, permitiendo así que el LED se ilumine.

La única diferencia que notarás entre este circuito y el anterior es que en el circuito anterior el cátodo del LED está conectado al **colector** del **S8050 (transistor NPN)**, mientras que en este está conectado al **emisor** del **S8550 (transistor PNP)**.



Código

Nota:

- Abre el archivo `5.6_transistor.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine

button = machine.Pin(14, machine.Pin.IN)    # Button
led = machine.Pin(26, machine.Pin.OUT)   # LED

# Start an infinite loop
while True:
    # Read the current value of the 'button' object (0 or 1) and store it in the 'button_
↪status' variable
    button_status = button.value()
    # If the button is pressed (value is 1)
    if button_status == 1:
        led.value(1) # Turn the LED on
    # If the button is not pressed (value is 0)
```

(continué en la próxima página)

(proviene de la página anterior)

```
    else:
        led.value(0)            # turn the LED off
```

Two types of transistors can be controlled using the same code. When we press the button, the ESP32 will send a high-level signal to the transistor; when we release it, it will send a low-level signal.

- El circuito usando el S8050 (transistor NPN) se iluminará cuando se presione el botón, indicando que está en un estado de conducción de nivel alto;
- El circuito usando el S8550 (transistor PNP) se iluminará cuando se suelte el botón, indicando que está en un estado de conducción de nivel bajo.

4.25 5.7 Siente la Luz

El fotoresistor es un dispositivo comúnmente utilizado para entradas analógicas, similar a un potenciómetro. Su valor de resistencia cambia basado en la intensidad de la luz que recibe. Cuando está expuesto a luz fuerte, la resistencia del fotoresistor disminuye, y a medida que la intensidad de la luz disminuye, la resistencia aumenta.

Al leer el valor del fotoresistor, podemos recopilar información sobre las condiciones de luz ambiental. Esta información puede ser utilizada para tareas tales como controlar el brillo de un LED, ajustar la sensibilidad de un sensor, o implementar acciones dependientes de la luz en un proyecto.

Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Fotorresistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

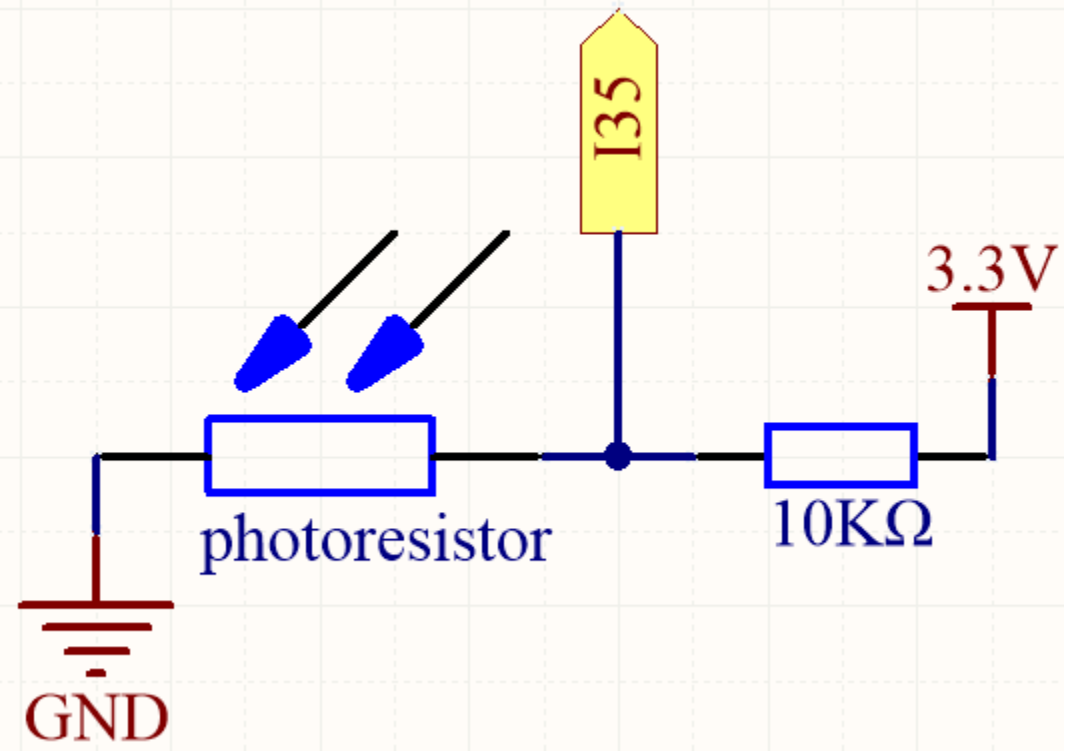
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Amarrado**

  Los siguientes pines son pines de amarrado, que afectan el proceso de arranque del ESP32 durante el encendido o reinicio. Sin embargo, una vez que el ESP32 se ha iniciado correctamente, pueden ser utilizados como pines regulares.

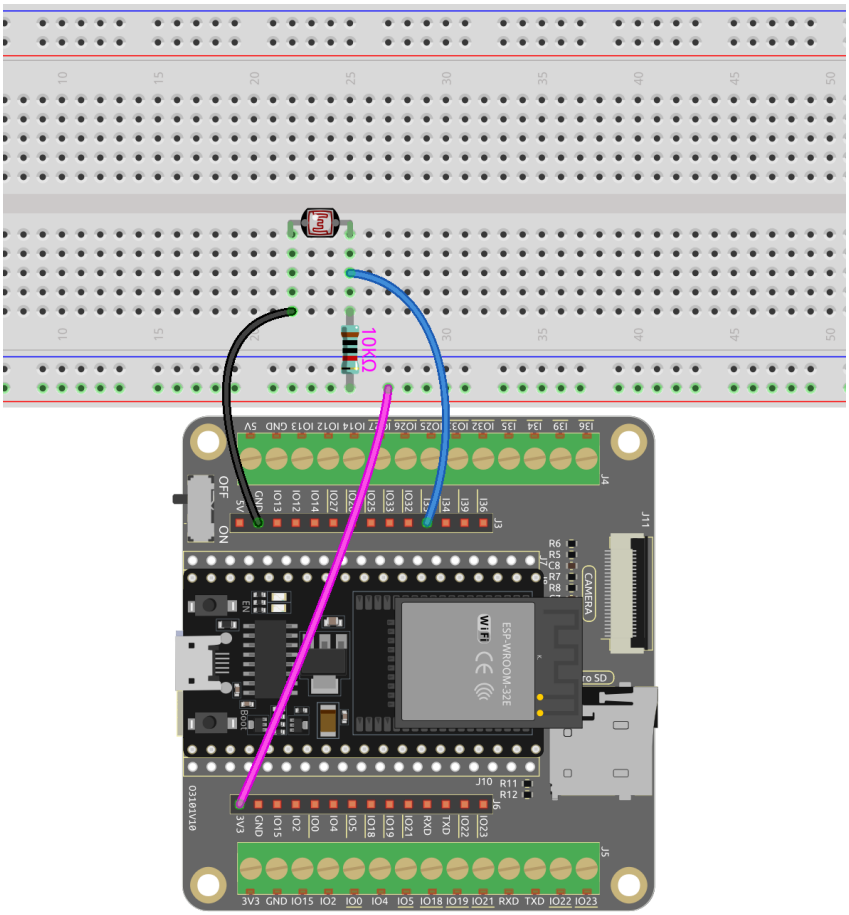
Pines de Amarrado	IO0, IO12

Esquemático



A medida que aumenta la intensidad de la luz, la resistencia del resistor dependiente de la luz (LDR) disminuye, resultando en una disminución en el valor leído en I35.

Conexión



Código

Nota:

- Abre el archivo `5.7_feel_the_light.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import ADC,Pin
import time

# create an ADC object acting on a pin
photoresistor = ADC(Pin(35, Pin.IN))

# Configure the ADC attenuation to 11dB for full range
photoresistor.atten(photoresistor.ATTN_11DB)

while True:

    # read a raw analog value in the range 0-4095
    value = photoresistor.read()
    print(value)
```

(continué en la próxima página)

(proviene de la página anterior)

```
    time.sleep(0.05)
```

Después de ejecutar el programa, la consola muestra los valores del fotoresistor. Puedes iluminarlo con una linterna o cubrirlo con tu mano para ver cómo cambia el valor.

- `atten(photoresistor.ATTN_11DB)`: Configura la atenuación del ADC a 11dB para el rango completo.

  Para leer voltajes por encima del voltaje de referencia, aplica la atenuación de entrada con el argumento de palabra clave atten.

  Valores válidos (y rangos de medición lineal aproximados) son:

  - ADC.ATTN_0DB: Sin atenuación (100mV - 950mV)
  - ADC.ATTN_2_5DB: Atenuación de 2.5dB (100mV - 1250mV)
  - ADC.ATTN_6DB: Atenuación de 6dB (150mV - 1750mV)
  - ADC.ATTN_11DB: Atenuación de 11dB (150mV - 2450mV)
- machine.ADC - MicroPython Docs

4.26 5.8 Girar el Pomo

Un potenciómetro es un dispositivo de tres terminales que se utiliza comúnmente para ajustar la resistencia en un circuito. Cuenta con un pomo o una palanca deslizante que se puede usar para variar el valor de la resistencia del potenciómetro. En este proyecto, lo utilizaremos para controlar el brillo de un LED, similar a una lámpara de escritorio en nuestra vida cotidiana. Al ajustar la posición del potenciómetro, podemos cambiar la resistencia en el circuito, regulando así la corriente que fluye a través del LED y ajustando su brillo en consecuencia. Esto nos permite crear una experiencia de iluminación personalizable y ajustable, similar a la de una lámpara de escritorio.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Potenciómetro	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

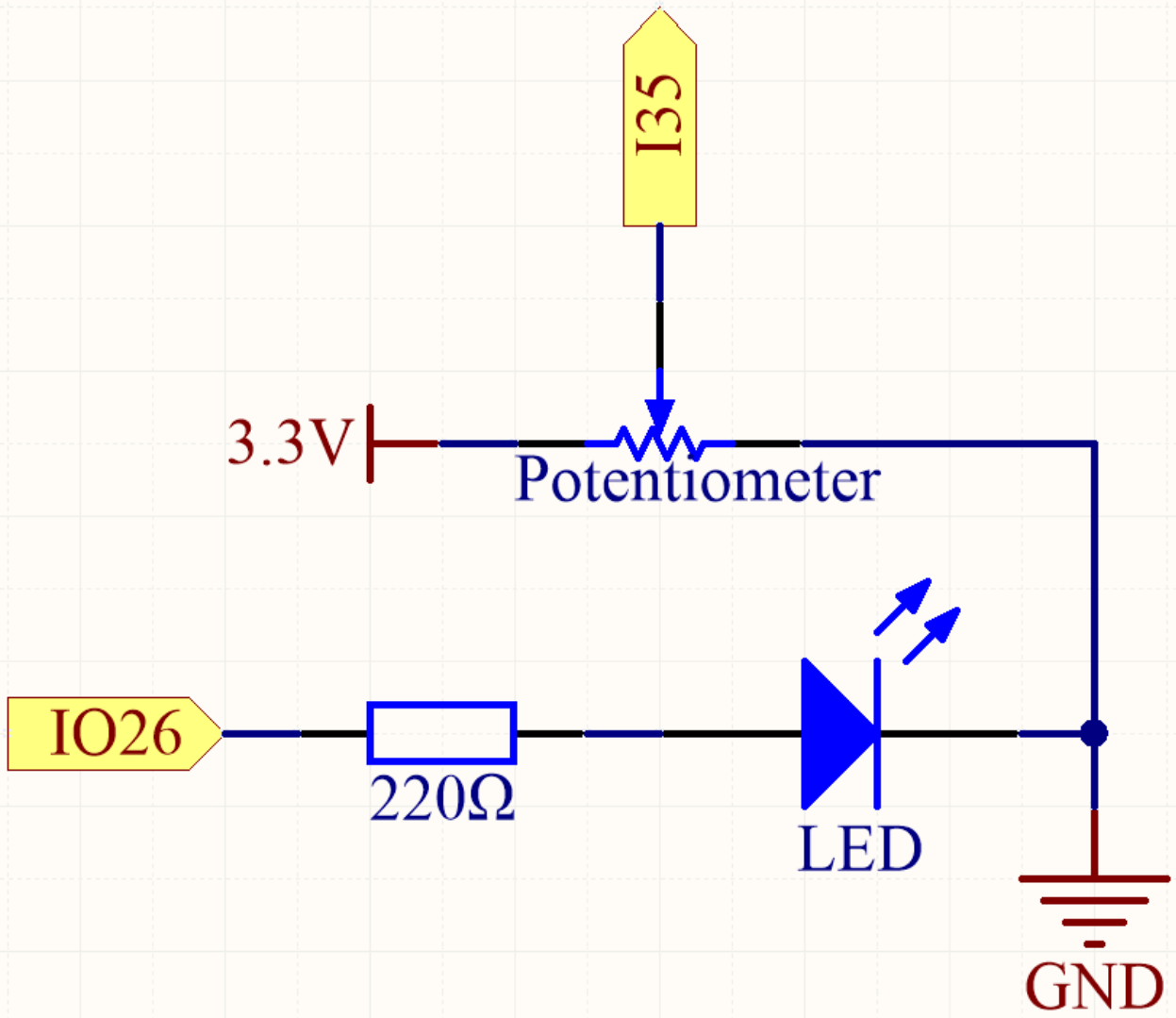
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Strapping**

  Los siguientes pines son pines de strapping, que afectan el proceso de arranque del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 ha arrancado con éxito, pueden ser utilizados como pines regulares.

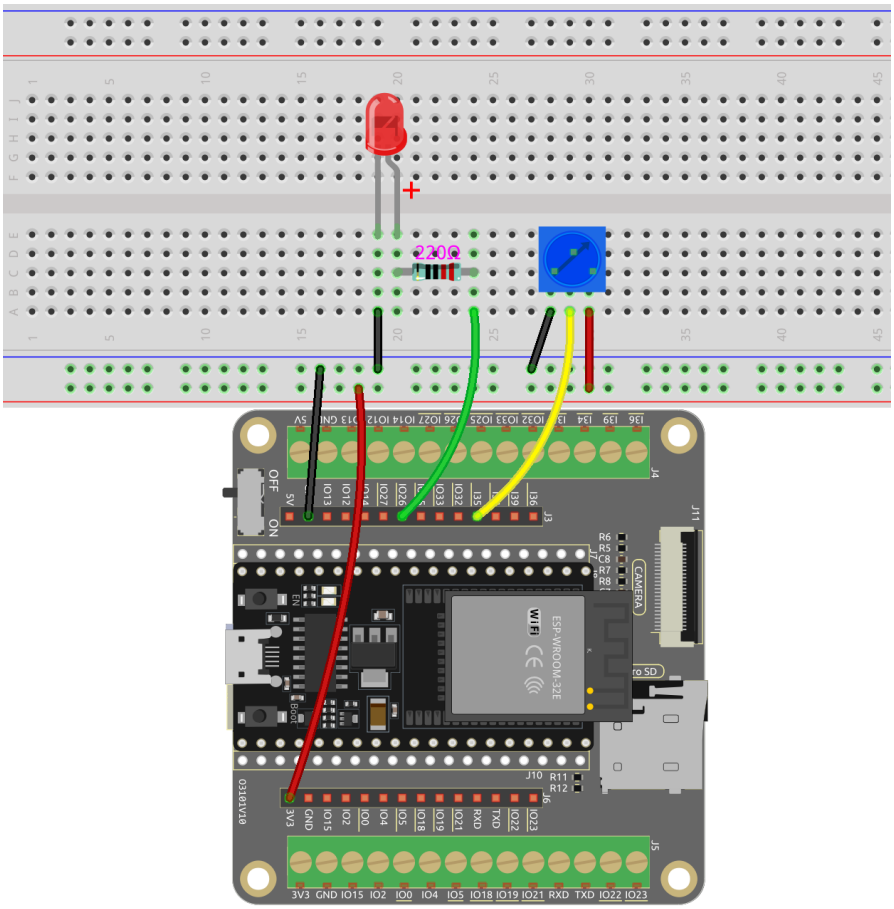
Pines de Strapping	IO0, IO12

Esquemático



Al girar el potenciómetro, el valor de I35 cambiará. Mediante programación, puedes usar el valor de I35 para controlar el brillo del LED. Por lo tanto, a medida que gires el potenciómetro, el brillo del LED también cambiará en consecuencia.

Conexión



Código

Nota:

- Abre el archivo `5.8_turn_the_knob.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import ADC, Pin, PWM
import time

pot = ADC(Pin(35, Pin.IN)) # create an ADC object acting on a pin

# Configure the ADC attenuation to 11dB for full range
pot.atten(pot.ATTN_11DB)

# Create a PWM object
led = PWM(Pin(26), freq=1000)

while True:
    # Read a raw analog value in the range of 0-4095
    value = pot.read()
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Scale the value to the range of 0-1023 for ESP32 PWM duty cycle
pwm_value = int(value * 1023 / 4095)

# Update the LED brightness based on the potentiometer value
led.duty(pwm_value)

# Read the voltage in microvolts and convert it to volts
voltage = pot.read_uv() / 1000000

# Print the raw value and the voltage
print(f"value: {value}, Voltage: {voltage}V")

# Wait for 0.5 seconds before taking the next reading
time.sleep(0.5)
```

Con este script ejecutado, el brillo del LED cambia a medida que se gira el potenciómetro, mientras que el valor analógico y el voltaje en este punto se muestran en el Shell.

- machine.ADC - Documentación de MicroPython

4.27 5.9 Medir la Humedad del Suelo

Este sensor de humedad del suelo capacitivo es diferente a la mayoría de los sensores resistivos en el mercado, ya que utiliza el principio de inducción capacitiva para detectar la humedad del suelo.

Al leer visualmente los valores del sensor de humedad del suelo, podemos recopilar información sobre el nivel de humedad en el suelo. Esta información es útil para varias aplicaciones, como sistemas de riego automático, monitoreo de la salud de las plantas o proyectos de detección ambiental.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Humedad del Suelo	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

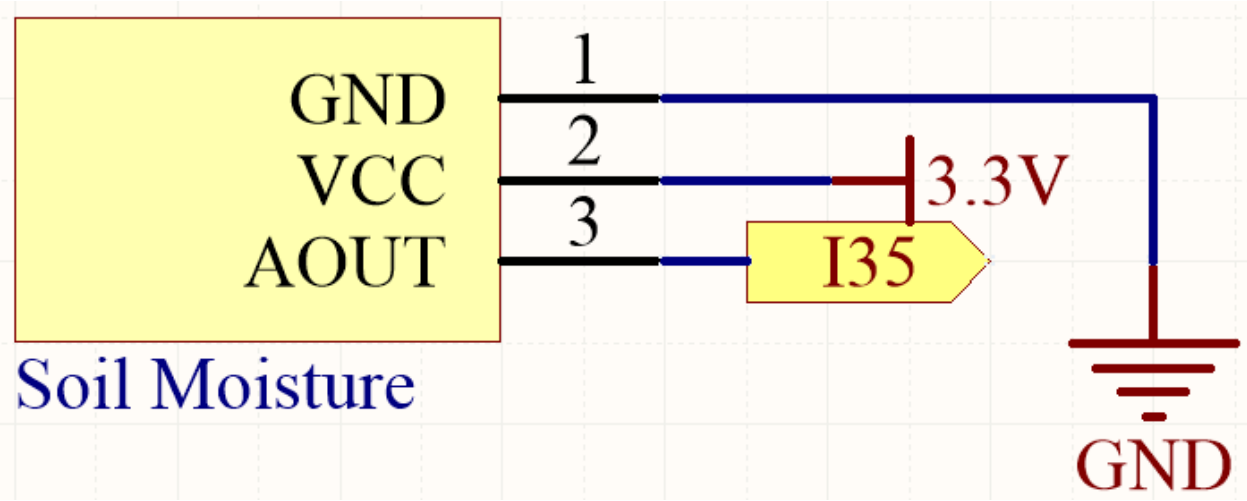
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Strapping**

  Los siguientes pines son pines de strapping, los cuales afectan el proceso de arranque del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 se ha iniciado correctamente, pueden ser utilizados como pines regulares.

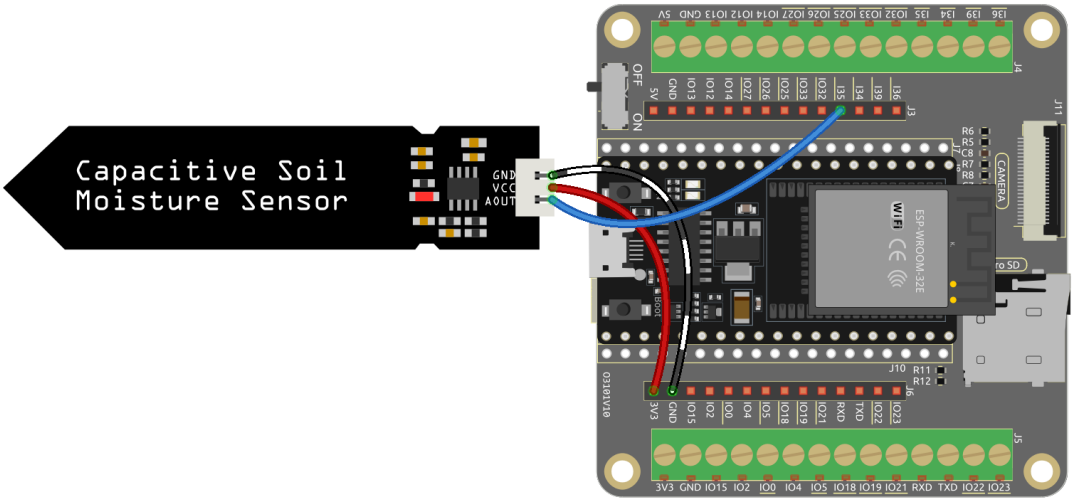
Pines de Strapping	IO0, IO12

Esquemático



Al insertar el módulo en el suelo y regarlo, el valor leído en I35 disminuirá.

Conexión



Código

Nota:

- Abre el archivo `5.9_moisture.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import ADC,Pin
import time

# crear un objeto ADC actuando sobre un pin
moisture = ADC(Pin(35, Pin.IN))

# Configurar la atenuación del ADC a 11dB para el rango completo
moisture.atten(moisture.ATTN_11DB)

while True:

    # leer un valor analógico crudo en el rango de 0-4095
    value = moisture.read()
    print(value)
    time.sleep(0.05)
```

Cuando el script se ejecute, verás el valor de la humedad del suelo en la Shell.

Al insertar el módulo en el suelo y regarlo, el valor del sensor de humedad del suelo se volverá más pequeño.

4.28 5.10 Detección de Temperatura

Un termistor es un sensor de temperatura que muestra una fuerte dependencia de la temperatura, y puede clasificarse en dos tipos: Coeficiente de Temperatura Negativo (NTC) y Coeficiente de Temperatura Positivo (PTC). La resistencia de un termistor NTC disminuye con el aumento de la temperatura, mientras que la resistencia de un termistor PTC aumenta con el aumento de la temperatura.

En este proyecto, usaremos un termistor NTC. Al conectar el termistor NTC a un pin de entrada analógica del microcontrolador ESP32, podemos medir su resistencia, que es directamente proporcional a la temperatura.

Incorporando el termistor NTC y realizando los cálculos necesarios, podemos medir con precisión la temperatura y mostrarla en el módulo I2C LCD1602. Este proyecto permite la monitorización de la temperatura en tiempo real y proporciona una interfaz visual para la visualización de la temperatura.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Termistor	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

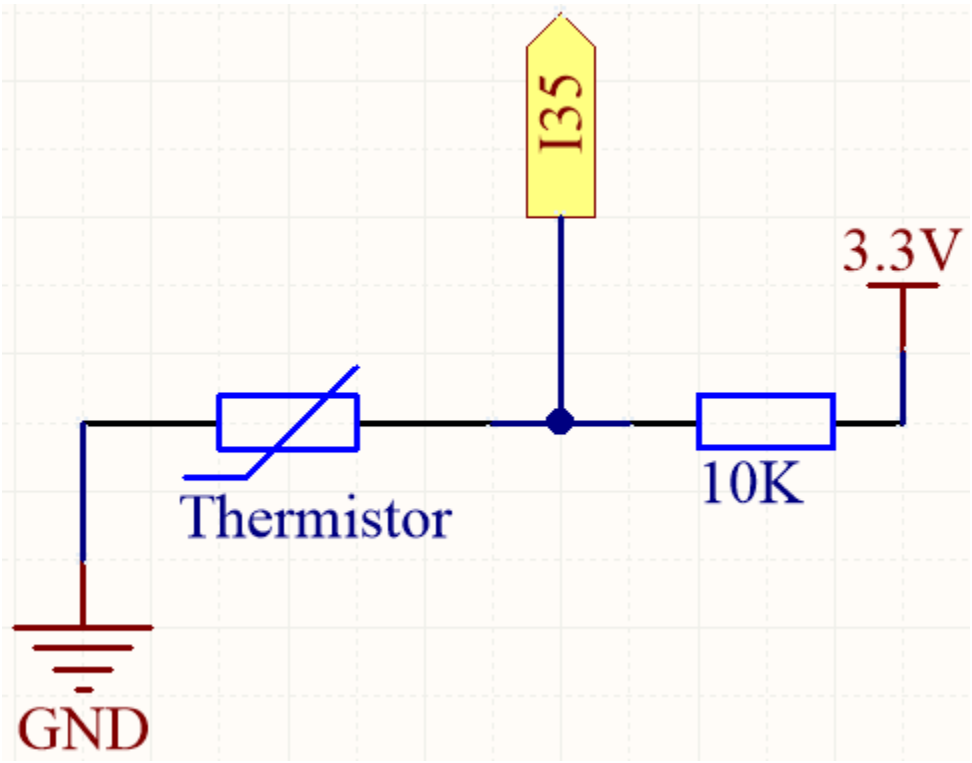
Pines Disponibles	IO14, IO25, I35, I34, I39, I36

- **Pines de Configuración**

  Los siguientes pines son pines de configuración, que afectan el proceso de arranque del ESP32 durante el encendido o el reinicio. Sin embargo, una vez que el ESP32 ha arrancado con éxito, se pueden usar como pines regulares.

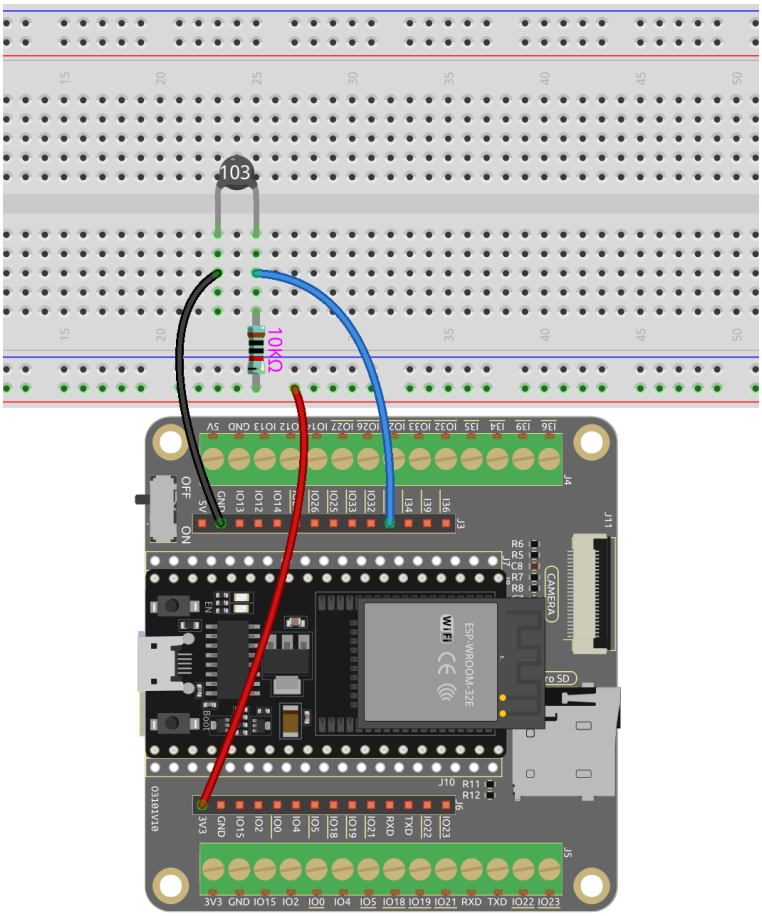
Pines de Configuración	IO0, IO12

Esquemático



Cuando la temperatura aumenta, la resistencia del termistor disminuye, causando que el valor leído en I35 disminuya. Además, utilizando una fórmula, puedes convertir el valor analógico en temperatura y luego imprimirlo.

Conexión



Nota:

- El termistor es negro y está marcado con 103.
- El anillo de color de la resistencia de 10K ohm es rojo, negro, negro, rojo y marrón.

Código

Nota:

- Abre el archivo `5.10_thermistor.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
# Import the necessary libraries
from machine import ADC, Pin
import time
import math

# Define the beta value of the thermistor, typically provided in the datasheet
beta = 3950
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Create an ADC object (thermistor)
thermistor = ADC(Pin(35, Pin.IN))

# Set the attenuation
thermistor.atten(thermistor.ATTN_11DB)

# Start an infinite loop to continuously monitor the temperature
while True:
    # Read the voltage in microvolts and convert it to volts
    Vr = thermistor.read_uv() / 1000000

    # Calculate the resistance of the thermistor based on the measured voltage
    Rt = 10000 * Vr / (3.3 - Vr)

    # Use the beta parameter and resistance value to calculate the temperature in Kelvin
    temp = 1 / (((math.log(Rt / 10000)) / beta) + (1 / (273.15 + 25)))

    # Convert to Celsius
    Cel = temp - 273.15

    # Convert to Fahrenheit
    Fah = Cel * 1.8 + 32

    # Print the temperature values in both Celsius and Fahrenheit
    print('Celsius: %.2f C  Fahrenheit: %.2f F' % (Cel, Fah))
    time.sleep(0.5)
```

Cuando se ejecuta el código, la Shell imprimirá las temperaturas en Celsius y Fahrenheit.

¿Cómo funciona?

Cada termistor tiene una resistencia normal. Aquí es de 10k ohmios, medida a 25 grados Celsius.

Cuando la temperatura aumenta, la resistencia del termistor disminuye. Luego, los datos de voltaje se convierten a cantidades digitales por el adaptador A/D.

La temperatura en Celsius o Fahrenheit se muestra mediante programación.

Aquí está la relación entre la resistencia y la temperatura:

RT = RN expB(1/TK - 1/TN)

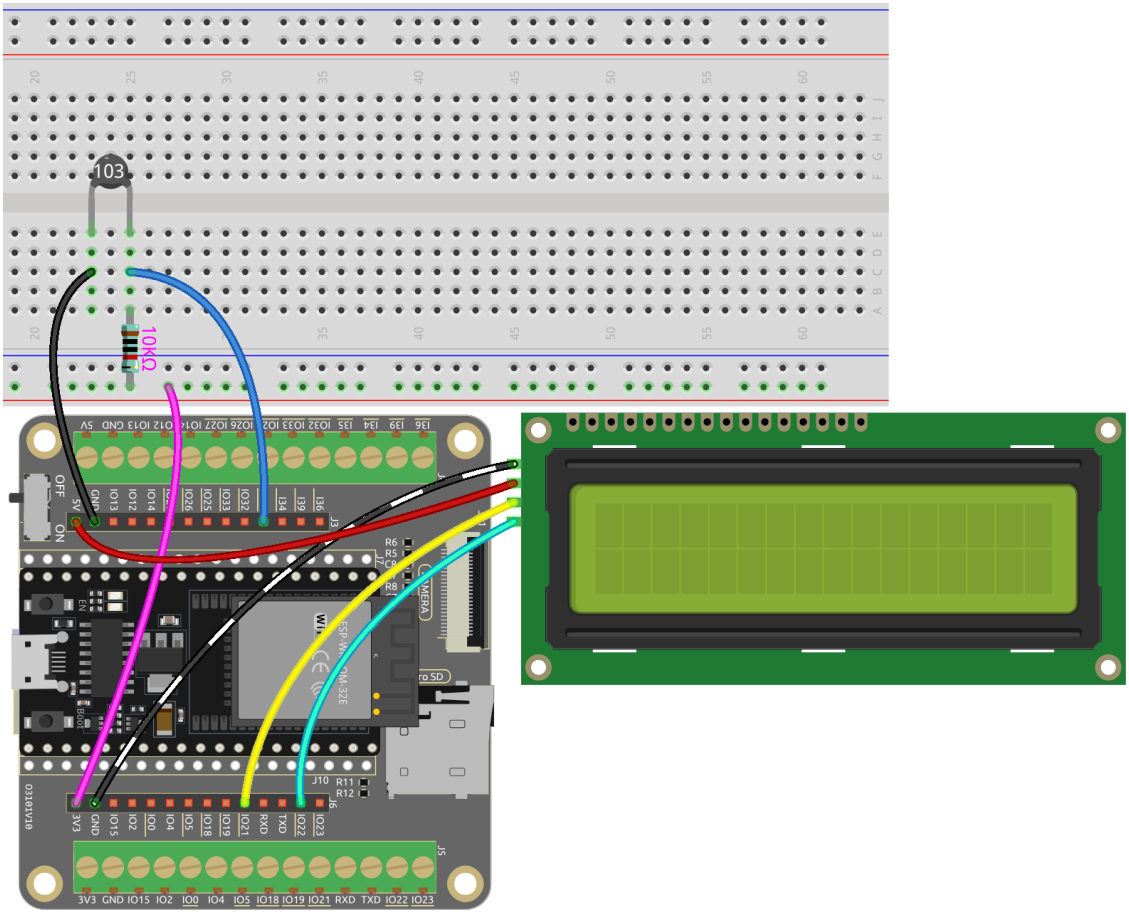
- **RT** es la resistencia del termistor NTC cuando la temperatura es **TK**.
- **RN** es la resistencia del termistor NTC bajo la temperatura nominal TN. Aquí, el valor numérico de RN es 10k.
- **TK** es una temperatura Kelvin y la unidad es K. Aquí, el valor numérico de **TK** es `373.15 + grado Celsius`.
- **TN** es una temperatura Kelvin nominal; la unidad también es K. Aquí, el valor numérico de TN es `373.15+25`.
- Y **B(beta)**, la constante material del termistor NTC, también se llama índice de sensibilidad al calor con un valor numérico `4950`.
- **exp** es la abreviatura de exponencial, y el número base `e` es un número natural y equivale aproximadamente a 2.7.

Convierte esta fórmula `TK = 1/(ln(RT/RN)/B + 1/TN)` para obtener la temperatura Kelvin que menos 273.15 equivale a grados Celsius.

Esta relación es una fórmula empírica. Es precisa solo cuando la temperatura y la resistencia están dentro del rango efectivo.

Aprender Más

También puedes mostrar las temperaturas Celsius y Fahrenheit calculadas en el I2C LCD1602.



Nota:

- Abre el archivo `5.10_thermistor_lcd.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.
- Aquí necesitas usar la biblioteca llamada `lcd1602.py`, por favor verifica si ha sido cargada en ESP32, para un tutorial detallado consulta *1.4 Subir las Bibliotecas (Importante)*.

```
# Import the necessary libraries
from machine import ADC, Pin
from lcd1602 import LCD
import time
```

(continué en la próxima página)

(proviene de la página anterior)

```
import math

# Define the beta value of the thermistor, typically provided in the datasheet
beta = 3950

# Create an ADC object (thermistor)
thermistor = ADC(Pin(35, Pin.IN))

# Set the attenuation
thermistor.atten(thermistor.ATTN_11DB)

lcd = LCD()

# Start an infinite loop to continuously monitor the temperature
while True:
    # Read the voltage in microvolts and convert it to volts
    Vr = thermistor.read_uv() / 1000000

    # Calculate the resistance of the thermistor based on the measured voltage
    Rt = 10000 * Vr / (3.3 - Vr)

    # Use the beta parameter and resistance value to calculate the temperature in Kelvin
    temp = 1 / (((math.log(Rt / 10000)) / beta) + (1 / (273.15 + 25)))

    # Convert to Celsius
    Cel = temp - 273.15

    # Convert to Fahrenheit
    Fah = Cel * 1.8 + 32

    # Print the temperature values in both Celsius and Fahrenheit
    print('Celsius: %.2f C  Fahrenheit: %.2f F' % (Cel, Fah))

    # Clear the LCD screen
    lcd.clear()

    # Display the temperature values in both Celsius and Fahrenheit
    lcd.message('Cel: %.2f \xDFC \n' % Cel)
    lcd.message('Fah: %.2f \xDFF' % Fah)
    time.sleep(1)
```

4.29 5.11 Alternar el Joystick

Si juegas muchos videojuegos, entonces deberías estar muy familiarizado con el Joystick. Se utiliza habitualmente para mover el personaje, rotar la pantalla, etc.

El principio detrás de la capacidad del Joystick para permitir que la computadora lea nuestras acciones es muy simple. Se puede pensar como compuesto por dos potenciómetros que están perpendiculares entre sí. Estos dos potenciómetros miden el valor analógico del joystick vertical y horizontalmente, resultando en un valor (x,y) en un sistema de coordenadas planas en ángulo recto.

El joystick de este kit también tiene una entrada digital, que se activa cuando se presiona el joystick.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Joystick	

Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada Analógica	IO14, IO25, I35, I34, I39, I36
Para Entrada Digital	IO13, IO14, IO27, IO26, IO25, IO33, IO4, IO18, IO19, IO21, IO22, IO23

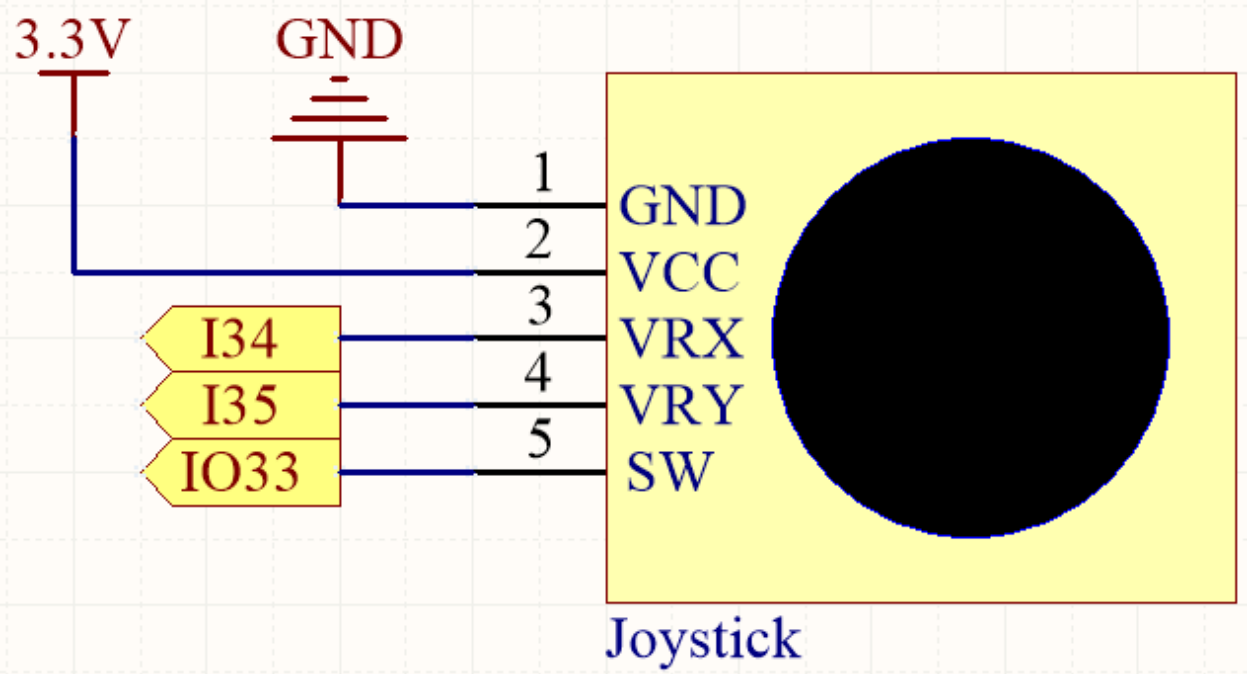
- **Pines de Strapping (Entrada)**

  Los pines de strapping son un conjunto especial de pines que se utilizan para determinar modos de arranque específicos durante el inicio del dispositivo (es decir, reinicio por encendido).

Pines de Strapping	IO5, IO0, IO2, IO12, IO15

  Generalmente, **no se recomienda usarlos como pines de entrada**. Si deseas usar estos pines, considera el impacto potencial en el proceso de arranque. Para más detalles, por favor refiérete a la sección *Pines de Estrapeo*.

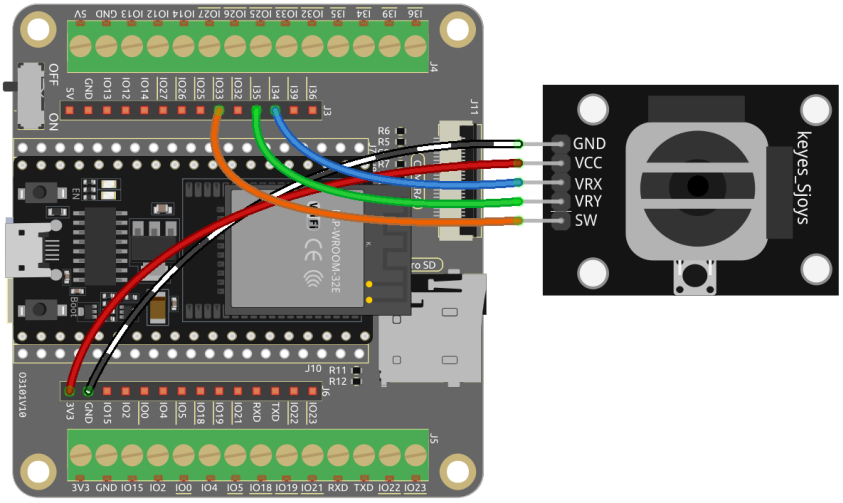
Esquemático



El pin SW (eje Z) está conectado a IO33, que tiene una resistencia pull-up de 4.7K incorporada. Por lo tanto, cuando el botón SW no está presionado, emitirá un nivel alto. Cuando el botón está presionado, emitirá un nivel bajo.

I34 e I35 cambiarán sus valores a medida que manipules el joystick. El rango de valores es de 0 a 4095.

Conexión



Código

Nota:

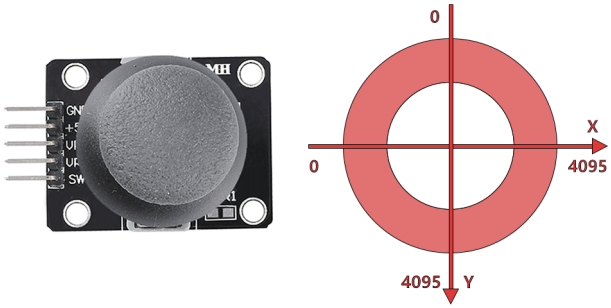
- Abre el archivo `5.11_joystick.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import ADC,Pin
import time

xAxis = ADC(Pin(34, Pin.IN)) # create an ADC object acting on a pin
xAxis.atten(xAxis.ATTN_11DB)
yAxis = ADC(Pin(35, Pin.IN)) # create an ADC object acting on a pin
yAxis.atten(yAxis.ATTN_11DB)
button = Pin(33, Pin.IN, Pin.PULL_UP)

while True:
    xValue = xAxis.read()  # read a raw analog value in the range 0-4095
    yValue = yAxis.read()  # read a raw analog value in the range 0-4095
    btnValue = button.value()
    print(f"X:{xValue}, Y:{yValue}, Button:{btnValue}")
    time.sleep(0.1)
```

Cuando el programa se ejecuta, la Shell imprime los valores x, y y del botón del joystick.

```
X:1921, Y:1775, Button:0
X:1921, Y:1775, Button:0
X:1923, Y:1775, Button:0
X:1924, Y:1776, Button:0
X:1926, Y:1777, Button:0
X:1925, Y:1776, Button:0
X:1924, Y:1776, Button:0
```

- Los valores de los ejes x e y son valores analógicos que varían de 0 a 4095.
- El botón es un valor digital con un estado de 1(suelto) o 0(presionado).



4.30 5.12 Medición de Distancia

El módulo ultrasónico se utiliza para la medición de distancia o la detección de objetos. En este proyecto, programaremos el módulo para obtener las distancias de los obstáculos. Enviando pulsos ultrasónicos y midiendo el tiempo que tardan en rebotar, podemos calcular distancias. Esto nos permite implementar acciones basadas en la distancia o comportamientos de evasión de obstáculos.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo Ultrasonido	

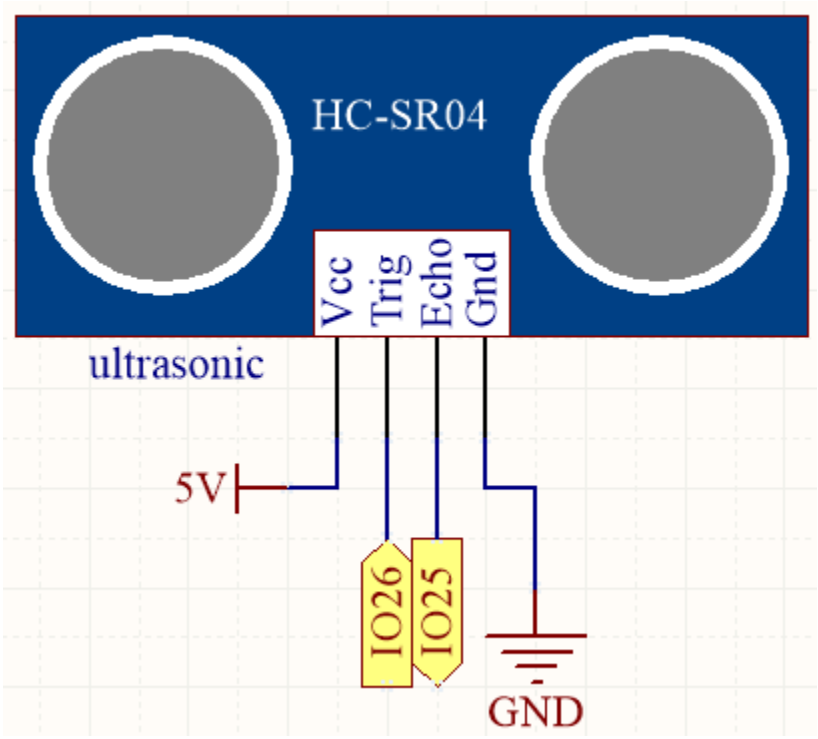
Pines Disponibles

- **Pines Disponibles**

  Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

Para Entrada	IO13, IO14, IO27, IO26, IO25, IO33, IO32, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Para Salida	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático

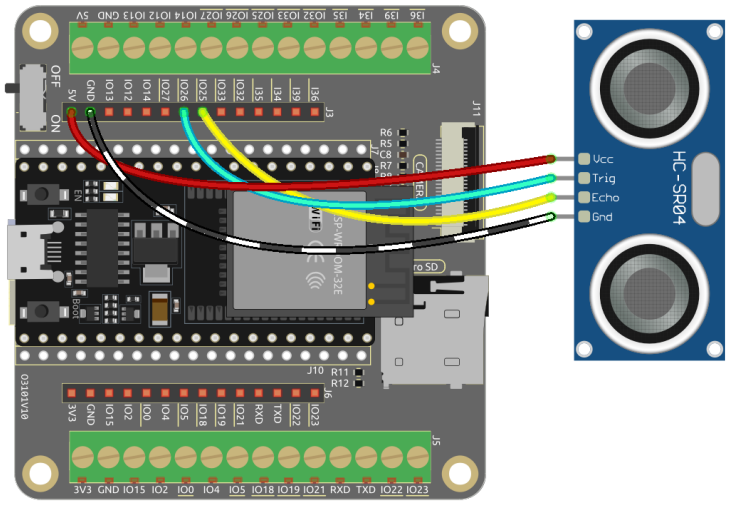


El ESP32 envía un conjunto de señales de onda cuadrada al pin Trig del sensor ultrasónico cada 10 segundos. Esto

incita al sensor ultrasónico a emitir una señal de ultrasonido de 40kHz hacia afuera. Si hay un obstáculo al frente, las ondas de ultrasonido se reflejarán hacia atrás.

Registrando el tiempo que toma desde el envío hasta la recepción de la señal, dividiéndolo por 2 y multiplicándolo por la velocidad de la luz, se puede determinar la distancia al obstáculo.

Conexión



Código

Nota:

- Abre el archivo `5.12_ultrasonic.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time

# Define the trigger and echo pins for the distance sensor
TRIG = machine.Pin(26, machine.Pin.OUT)
ECHO = machine.Pin(25, machine.Pin.IN)

# Calculate the distance using the ultrasonic sensor
def distance():
    # Ensure trigger is off initially
    TRIG.off()
    time.sleep_us(2)  # Wait for 2 microseconds

    # Send a 10-microsecond pulse to the trigger pin
    TRIG.on()
    time.sleep_us(10)
    TRIG.off()

    # Wait for the echo pin to go high
    while not ECHO.value():
```

(continué en la próxima página)

(proviene de la página anterior)

```
        pass

    # Record the time when the echo pin goes high
    time1 = time.ticks_us()

    # Wait for the echo pin to go low
    while ECHO.value():
        pass

    # Record the time when the echo pin goes low
    time2 = time.ticks_us()

    # Calculate the time difference between the two recorded times
    during = time.ticks_diff(time2, time1)

    # Calculate and return the distance (in cm) using the speed of sound (340 m/s)
    return during * 340 / 2 / 10000

# Continuously measure and print the distance
while True:
    dis = distance()
    print('Distance: %.2f' % dis)
    time.sleep_ms(300)  # Wait for 300 milliseconds before repeating
```

Una vez que el programa esté en ejecución, la Shell imprimirá la distancia del sensor ultrasónico al obstáculo adelante.

4.31 5.13 Temperatura - Humedad

El DHT11 es un sensor de temperatura y humedad comúnmente utilizado para mediciones ambientales. Es un sensor digital que se comunica con un microcontrolador para proporcionar lecturas de temperatura y humedad.

En este proyecto, leeremos el sensor DHT11 e imprimiremos los valores de temperatura y humedad que detecta.

Al leer los datos proporcionados por el sensor, podemos obtener los valores actuales de temperatura y humedad en el ambiente. Estos valores pueden ser utilizados para el monitoreo en tiempo real de las condiciones ambientales, observaciones meteorológicas, control climático interior, informes de humedad y más.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Sensor de Humedad y Temperatura DHT11	

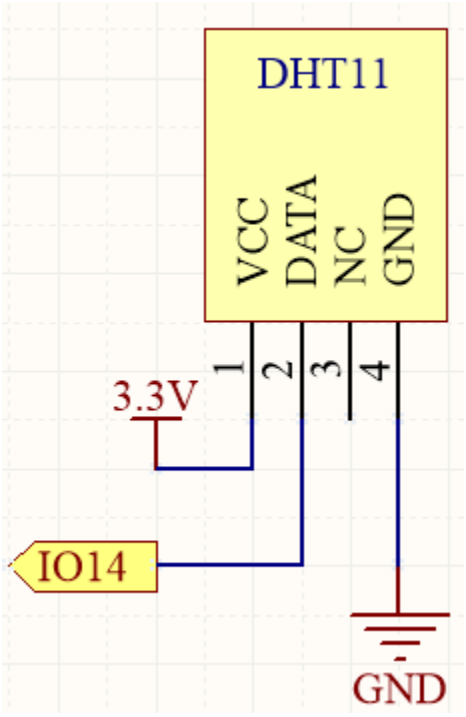
Pines Disponibles

- **Pines Disponibles**

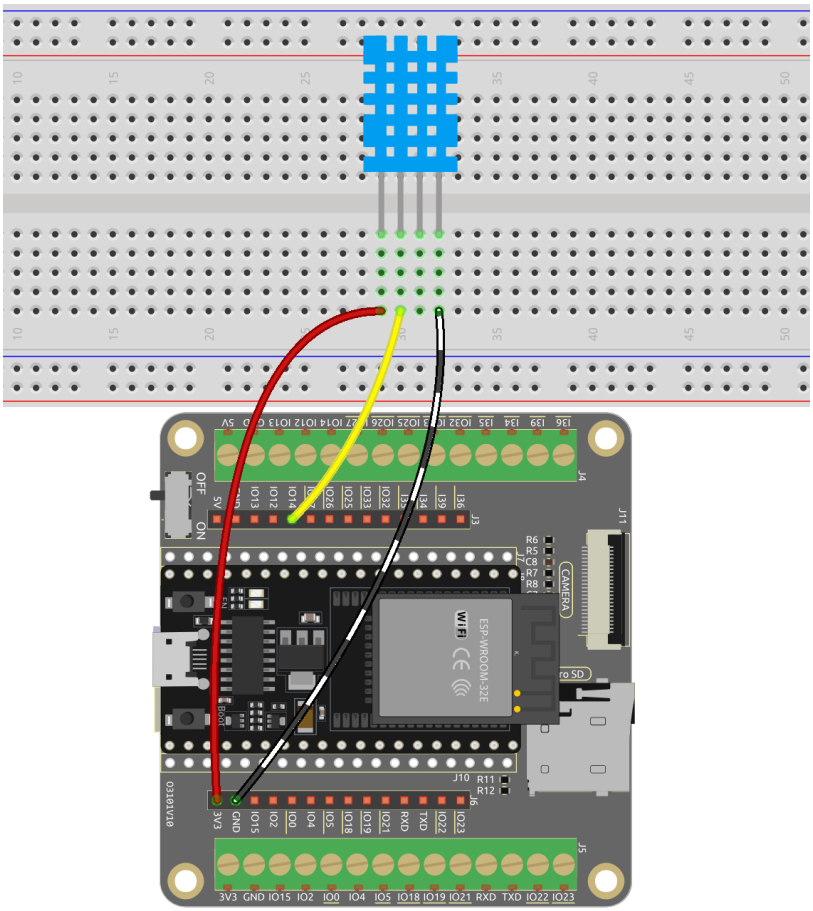
  Aquí tienes una lista de los pines disponibles en la placa ESP32 para este proyecto.

Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cableado



Código

Nota:

- Abre el archivo `5.13_dht11.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import dht
import machine
import time

# Initialize the DHT11 sensor and connect it to pin 14
sensor = dht.DHT11(machine.Pin(14))

# Loop indefinitely to continuously measure temperature and humidity
while True:
    try:
        # Measure temperature and humidity
        sensor.measure()

        # Get temperature and humidity values
        temp = sensor.temperature()
```

(continué en la próxima página)

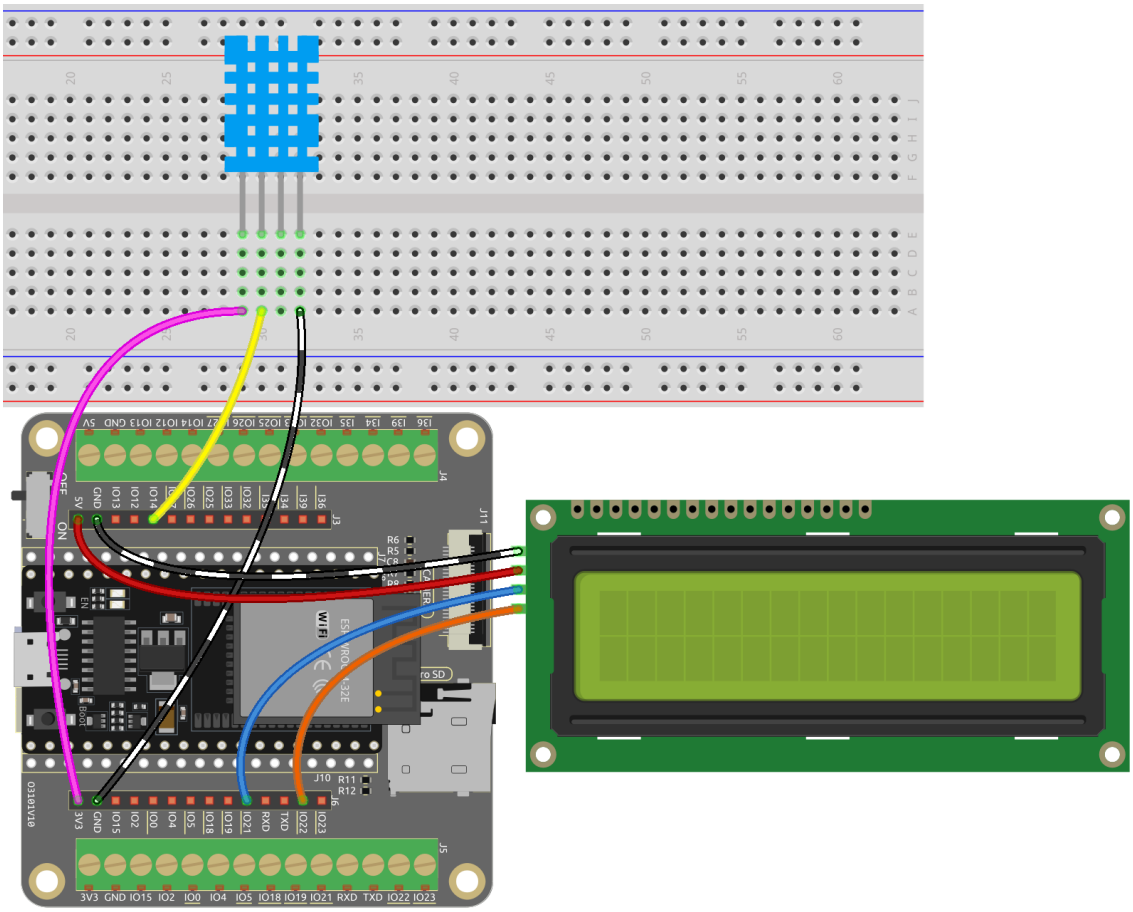
(proviene de la página anterior)

```
        humi = sensor.humidity()

        # Print temperature and humidity
        print("Temperature: {}, Humidity: {}".format(temp, humi))

        # Wait for 1 second between measurements
        time.sleep(1)
    except Exception as e:
        print("Error: ", e)
        time.sleep(1)
```

Cuando el código está en ejecución, verás que la Shell imprime continuamente la temperatura y la humedad, y a medida que el programa se ejecuta de manera estable, estos dos valores se volverán cada vez más precisos.

Aprender Más

También puedes mostrar la temperatura y la humedad en el LCD I2C 1602.



Nota:

- Abre el archivo `5.13_dht11_lcd.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

- Aquí necesitas usar la biblioteca llamada `lcd1602.py`, por favor verifica si ha sido cargada en el ESP32, para un tutorial detallado refiérete a *1.4 Subir las Bibliotecas (Importante)*.

```
import dht
import machine
import time
from lcd1602 import LCD

# Initialize the DHT11 sensor and connect it to pin 14
sensor = dht.DHT11(machine.Pin(14))

# Initialize the LCD1602 display
lcd = LCD()

# Loop to measure temperature and humidity
while True:
    try:
        # Measure temperature and humidity
        sensor.measure()

        # Get temperature and humidity values
        temp = sensor.temperature()
        humi = sensor.humidity()

        # Print temperature and humidity
        print("Temperature: {}, Humidity: {}".format(temp, humi))

        # Clear the LCD display
        lcd.clear()

        # Display temperature and humidity on the LCD1602 screen
        lcd.write(0, 0, "Temp: {}\xDFC".format(temp))
        lcd.write(0, 1, "Humi: {}%".format(humi))

        # Wait for 2 seconds before measuring again
        time.sleep(2)

    except Exception as e:
        print("Error: ", e)
        time.sleep(2)
```

4.32 5.14 Control Remoto por Infrarrojos

Un receptor de infrarrojos es un componente que recibe señales infrarrojas y puede detectar y emitir señales compatibles con el nivel TTL de forma independiente. Su tamaño es similar al de un transistor empaquetado en plástico convencional y se utiliza comúnmente en diversas aplicaciones como el control remoto por infrarrojos y la transmisión infrarroja.

En este proyecto, utilizaremos un receptor de infrarrojos para detectar señales de un control remoto. Cuando se presiona un botón en el control remoto y el receptor de infrarrojos recibe la señal correspondiente, puede decodificar la señal para determinar qué botón fue presionado. Decodificando la señal recibida, podemos identificar la tecla o comando específico asociado con ella.

El receptor de infrarrojos nos permite incorporar la funcionalidad de control remoto en nuestro proyecto, permitiéndonos interactuar con y controlar dispositivos usando señales infrarrojas.

Componentes Requeridos

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

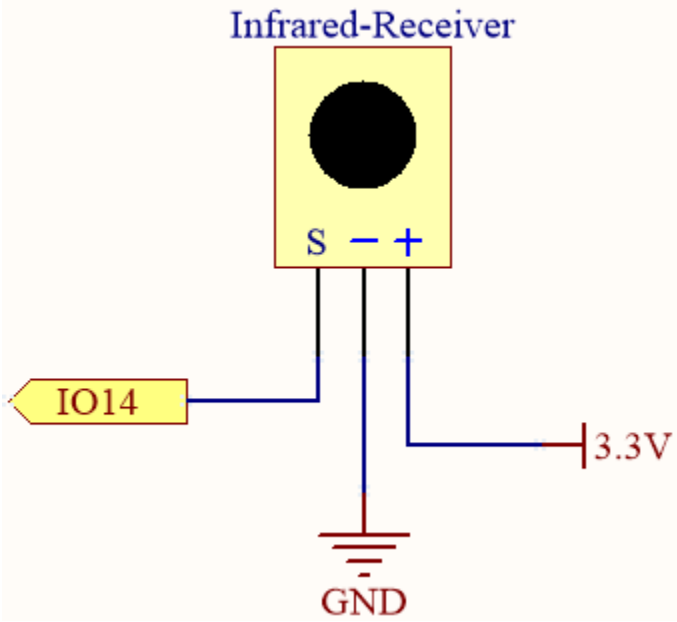
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Receptor de Infrarrojos	

Pines Disponibles

Aquí hay una lista de pines disponibles en la placa ESP32 para este proyecto.

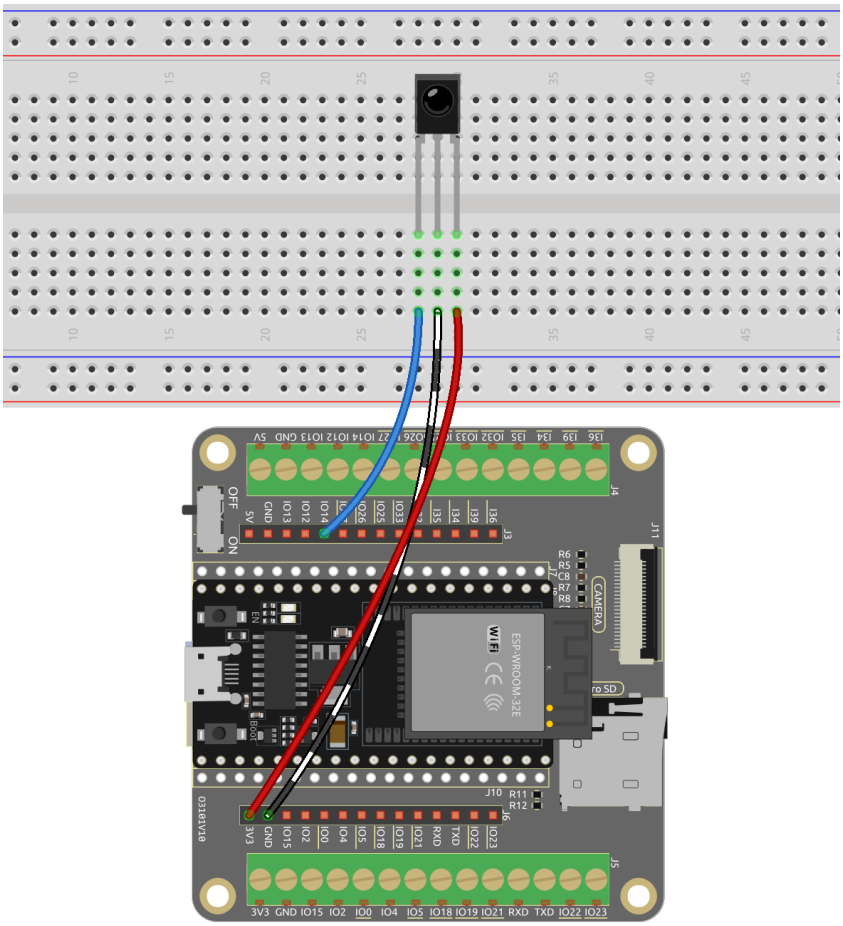
Pines Disponibles	IO13, IO12, IO14, IO27, IO26, IO25, IO15, IO0, IO5, IO18, IO19, IO21, IO22, IO23

Esquemático



Cuando presionas un botón en el control remoto, el receptor de infrarrojos detecta la señal y puedes usar una biblioteca de infrarrojos para decodificarla. Este proceso de decodificación te permite obtener el valor de la tecla asociada con la presión del botón.

Conexión



Código

Nota:

- Abre el archivo `5.14_ir_receiver.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.
- Aquí, necesitas utilizar las bibliotecas encontradas en la carpeta `ir_rx`. Por favor, asegúrate de que hayan sido cargadas al ESP32. Para un tutorial completo, consulta *1.4 Subir las Bibliotecas (Importante)*.

```
import time
from machine import Pin, freq
from ir_rx.print_error import print_error
from ir_rx.nec import NEC_8

pin_ir = Pin(14, Pin.IN) # IR receiver
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Decode the received data and return the corresponding key name
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
        return "3"
    if data == 0x08:
        return "4"
    if data == 0x1C:
        return "5"
    if data == 0x5A:
        return "6"
    if data == 0x42:
        return "7"
    if data == 0x52:
        return "8"
    if data == 0x4A:
        return "9"
    if data == 0x09:
        return "+"
    if data == 0x15:
        return "-"
    if data == 0x7:
        return "EQ"
    if data == 0x0D:
        return "U/SD"
    if data == 0x19:
        return "CYCLE"
    if data == 0x44:
        return "PLAY/PAUSE"
    if data == 0x43:
        return "FORWARD"
    if data == 0x40:
        return "BACKWARD"
    if data == 0x45:
        return "POWER"
    if data == 0x47:
        return "MUTE"
    if data == 0x46:
        return "MODE"
    return "ERROR"

# User callback
def callback(data, addr, ctrl):
    if data < 0:  # NEC protocol sends repeat codes.
        pass
    else:
        print(decodeKeyValue(data))
```

(continué en la próxima página)

(proviene de la página anterior)

```
ir = NEC_8(pin_ir, callback) # Instantiate the NEC_8 receiver

# Show debug information
ir.error_function(print_error)

# keep the script running until interrupted by a keyboard interrupt (Ctrl+C)
try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()  # Close the receiver
```

Cuando el programa esté en ejecución, presiona la tecla en el control remoto, el valor y el nombre de la tecla aparecerán en la Shell.

Nota: El nuevo control remoto cuenta con una lengüeta de plástico en el extremo para aislar la batería en su interior. Para activar el control remoto al usarlo, simplemente retira esta pieza de plástico.

¿Cómo funciona?

1. Aunque este programa pueda parecer algo complejo a primera vista, en realidad realiza las funciones fundamentales del receptor IR utilizando solo unas pocas líneas de código.

```
import time
from machine import Pin, freq
from ir_rx.nec import NEC_8

pin_ir = Pin(14, Pin.IN) # IR receiver

# User callback
def callback(data, addr, ctrl):
    if data < 0:  # NEC protocol sends repeat codes.
        pass
    else:
        print(decodeKeyValue(data))

ir = NEC_8(pin_ir, callback)  # Instantiate receiver
```

   - En este código, se instancia un objeto `ir`, permitiéndole leer las señales capturadas por el receptor IR en cualquier momento.
   - La información resultante se almacena entonces en la variable `data` dentro de la función de callback.
     - Función de Callback - Wikipedia
   - Si el receptor IR recibe valores duplicados (por ejemplo, cuando se presiona un botón y se mantiene presionado), el `data` será menor que 0, y este `data` necesita ser filtrado.
   - De lo contrario, el `data` sería un valor utilizable, aunque en un código ilegible. La función `decodeKeyValue(data)` se utiliza entonces para decodificarlo en un formato más comprensible.

```
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
    ...
```

2. A continuación, incorporamos varias funciones de depuración en el programa. Aunque estas funciones son esenciales, no están directamente relacionadas con el resultado deseado que buscamos lograr.

```
from ir_rx.print_error import print_error

ir.error_function(print_error) # Show debug information
```

3. Por último, utilizamos un bucle vacío para el programa principal e implementamos una estructura try-except para asegurar que el programa salga con el objeto `ir` correctamente terminado.

```
try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()
```

- Try Statement - Python Docs

6. Proyectos Divertidos

4.33 6.1 Piano de Frutas

¿Alguna vez has querido tocar el piano pero no podías permitírtelo? ¿O tal vez solo quieres divertirte haciendo un piano de frutas por ti mismo? ¡Pues este proyecto es para ti!

Con solo unos pocos sensores táctiles en la placa ESP32, ahora puedes tocar tus melodías favoritas y disfrutar de la experiencia de tocar el piano sin gastar una fortuna.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Zumbador	
Transistor	

Acerca de los Pines Táctiles

El microcontrolador ESP32 tiene funcionalidad de sensor táctil incorporada, lo que te permite usar ciertos pines en la placa como entradas sensibles al tacto. El sensor táctil funciona midiendo cambios en la capacitancia en los pines táctiles, que son causados por las propiedades eléctricas del cuerpo humano.

Aquí hay algunas características clave del sensor táctil en el ESP32:

- **Número de pines táctiles**

  El ESP32 tiene hasta 10 pines táctiles, dependiendo de la placa específica. Los pines táctiles generalmente están etiquetados con una «T» seguida de un número.

  - GPIO4: TOUCH0
  - GPIO0TOUCH1
  - GPIO2: TOUCH2
  - GPIO15: TOUCH3
  - GPIO13: TOUCH4
  - GPIO12: TOUCH5
  - GPIO14: TOUCH6
  - GPIO27: TOUCH7
  - GPIO33: TOUCH8
  - GPIO32: TOUCH9

  Nota: Los pines GPIO0 y GPIO2 se usan para el arranque y la carga de firmware en el ESP32, respectivamente. Estos pines también están conectados al LED y botón integrados en la placa. Por lo tanto, generalmente no se recomienda usar estos pines para otros fines, ya que podría interferir con el funcionamiento normal de la placa.

- **Sensibilidad**

  El sensor táctil en el ESP32 es muy sensible y puede detectar incluso pequeños cambios en la capacitancia. La sensibilidad se puede ajustar usando configuraciones de software.

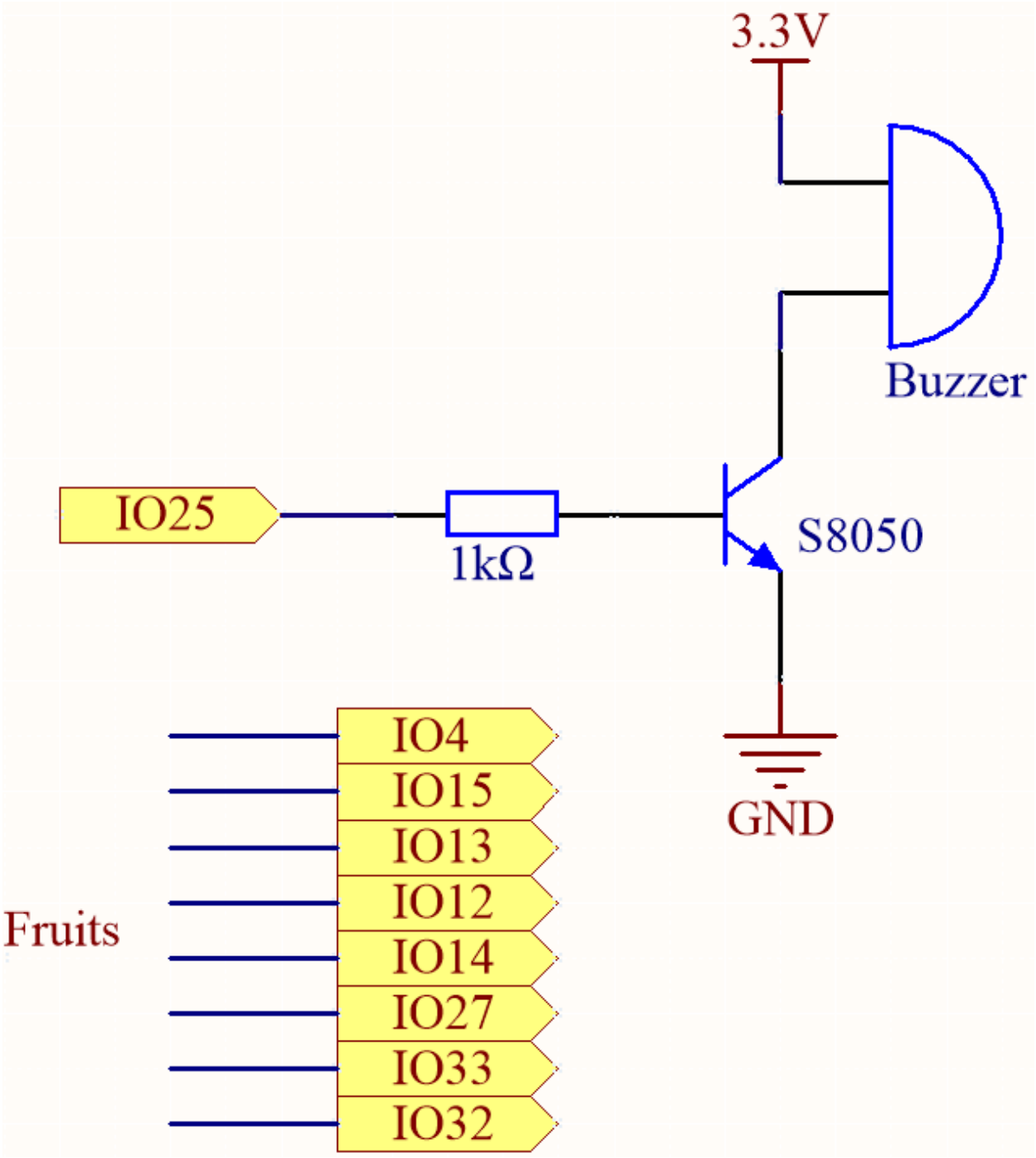
- **Protección ESD**

Los pines táctiles en el ESP32 tienen protección incorporada contra descargas electrostáticas (ESD), lo que ayuda a prevenir daños a la placa por electricidad estática.

- **Multitáctil**

  El sensor táctil en el ESP32 admite multitáctil, lo que significa que puedes detectar varios eventos táctiles simultáneamente.

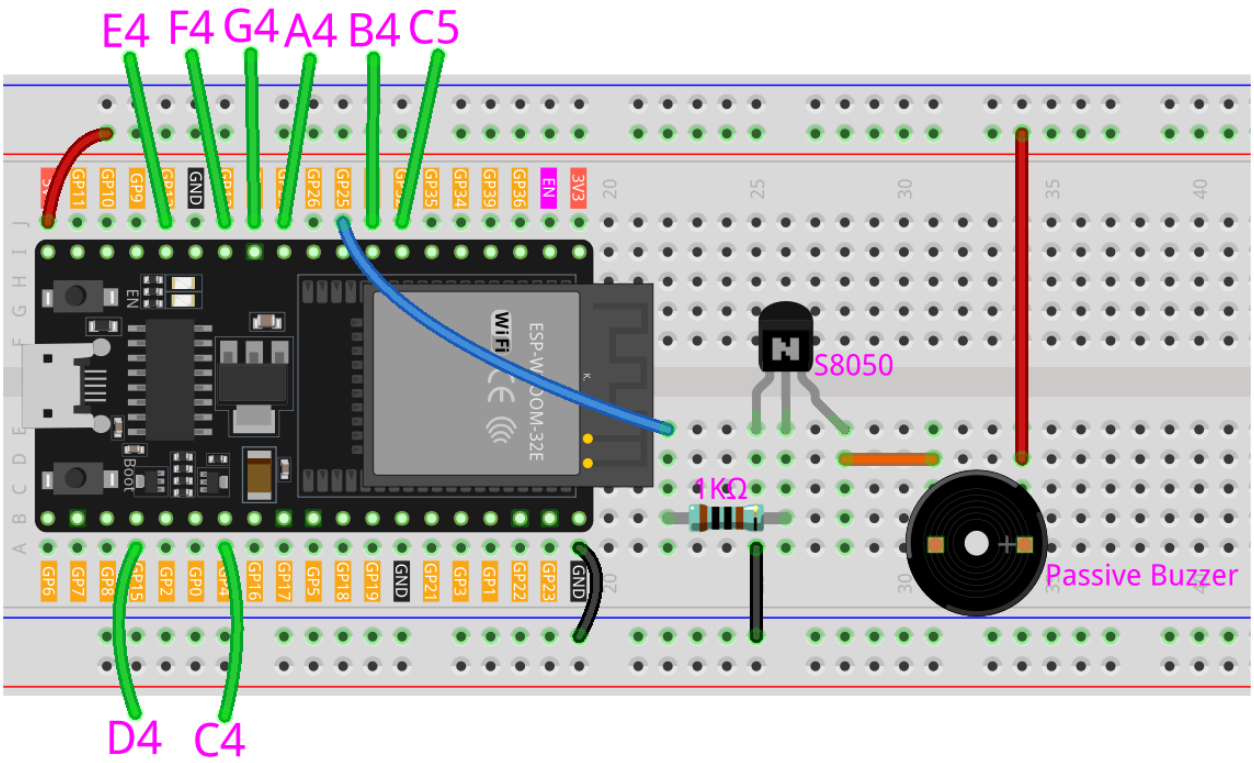
Esquemático



La idea detrás de este proyecto es utilizar sensores táctiles para detectar cuándo un usuario toca un pin específico. Cada

pin táctil está asociado con una nota específica, y cuando el usuario toca un pin, la nota correspondiente se reproduce en el zumbador pasivo. El resultado es una forma simple y económica de disfrutar de la experiencia de tocar el piano.

Cableado



En este proyecto, necesitas quitar el ESP32 WROOM 32E de la placa de expansión y luego insertarlo en el protoboard. Esto se debe a que algunos pines en la placa de expansión están conectados a resistencias, lo que afectará la capacitancia de los pines.

Código

Nota:

- Abre el archivo `6.1_fruit_piano.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, PWM, TouchPad
import time

# Define the touch pins and their corresponding notes
touch_pins = [4, 15, 13, 12, 14, 27, 33, 32]  # Use valid touch-capable pins
notes = [262, 294, 330, 349, 392, 440, 494, 523]

# Initialize the touch sensors
touch_sensors = [TouchPad(Pin(pin)) for pin in touch_pins]

# Initialize the buzzer
buzzer = PWM(Pin(25), duty=0)
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Function to play a tone
def play_tone(frequency, duration):
    buzzer.freq(frequency)
    buzzer.duty(512)
    time.sleep_ms(duration)
    buzzer.duty(0)

touch_threshold = 200

# Main loop to check for touch inputs and play the corresponding note
while True:
    for i, touch_sensor in enumerate(touch_sensors):
        value = touch_sensor.read()
        print(i,value)
        if value < touch_threshold:
            play_tone(notes[i], 100)
            time.sleep_ms(50)
        time.sleep(0.01)
```

Puedes conectar frutas a estos pines del ESP32: 4, 15, 13, 12, 14, 27, 33, 32.

Cuando el script se ejecuta, tocar estas frutas reproducirá las notas C, D, E, F, G, A, B y C5.

Nota: `Touch_threshold` necesita ser ajustado basado en la conductividad de diferentes frutas.

Puedes ejecutar el script primero para ver los valores impresos por la shell.

```
0 884
1 801
2 856
3 964
4 991
5 989
6 1072
7 1058
```

Después de tocar las frutas en los pines 12, 14 y 27, los valores impresos son los siguientes. Por lo tanto, establecí el `touch_threshold` en 200, lo que significa que cuando se detecta un valor menor a 200, se considera que ha sido tocado, y el zumbador emitirá diferentes notas.

```
0 882
1 810
2 799
3 109
4 122
5 156
6 1068
7 1055
```

4.34 6.2 Luz Fluyente

¿Alguna vez has querido añadir un elemento divertido e interactivo a tu espacio vital? Este proyecto implica crear una luz corriente usando una tira de LED WS2812 y un módulo de evitación de obstáculos. La luz corriente cambia de dirección cuando se detecta un obstáculo, convirtiéndola en una adición emocionante a tu decoración del hogar u oficina.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

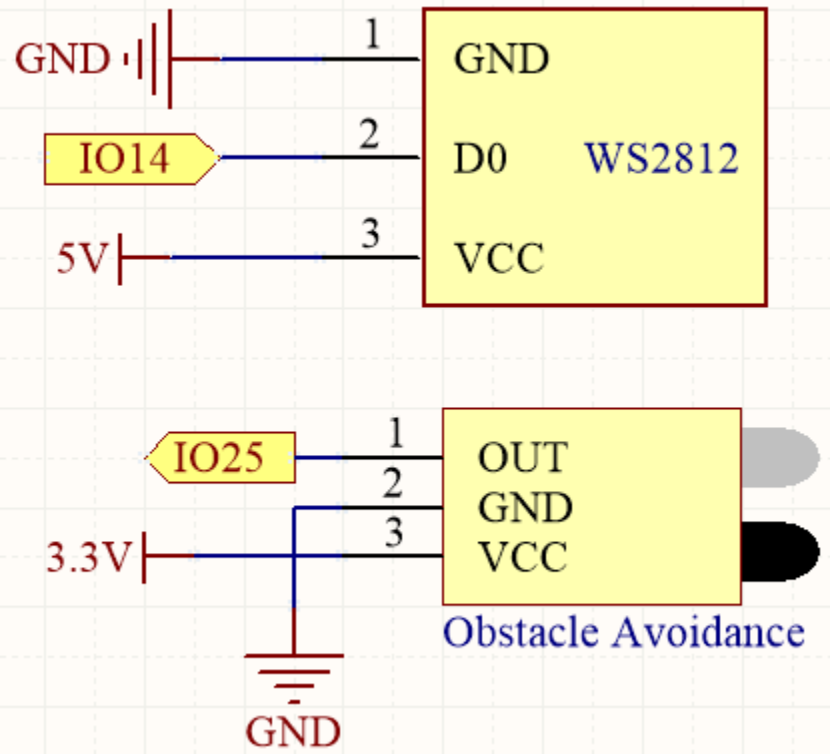
Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

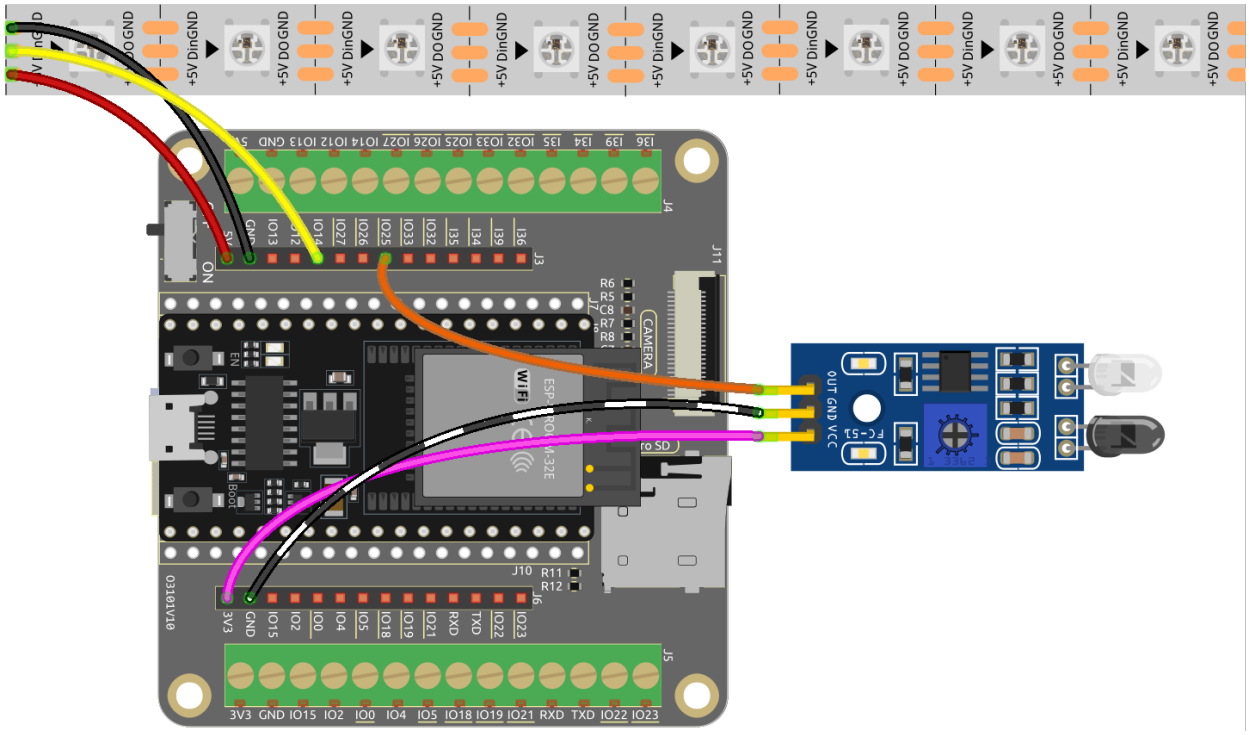
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Evitación de Obstáculos	
Tira de 8 LEDs RGB WS2812	

Diagrama Esquemático



La tira de LED WS2812 está compuesta por una serie de LEDs individuales que pueden ser programados para mostrar diferentes colores y patrones. En este proyecto, la tira está configurada para mostrar una luz corriente que se mueve en una dirección particular y cambia de dirección cuando un obstáculo es detectado por el módulo de evitación de obstáculos.

Cableado



Código

Nota:

- Abre el archivo `6.2_flowing_led.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin
import neopixel
import time
import random

# Set the number of pixels for the running light
num_pixels = 8

# Set the data pin for the RGB LED strip
data_pin = Pin(14, Pin.OUT)

# Initialize the RGB LED strip object
pixels = neopixel.NeoPixel(data_pin, num_pixels)

# Initialize the avoid sensor
avoid = Pin(25, Pin.IN)

# Initialize the direction variable
direction_forward = True

# Initialize the reverse direction flag
reverse_direction = False

# Continuously loop the running light
while True:

# Read the input from the infrared sensor
avoid_value = avoid.value()

# Generate a random color for the current pixel
color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

# If no obstacle is detected
if avoid_value:
      for i in range(num_pixels):

            # Turn on the current pixel with the random color
            pixels[i] = color

            # Update the RGB LED strip display
            pixels.write()

            # Turn off the current pixel
            pixels[i] = (0, 0, 0)
```

(continué en la próxima página)

(proviene de la página anterior)

```
            time.sleep_ms(100)

# If detects an obstacle, change the direction of the LED strip
else:
      for i in range(num_pixels-1, -1, -1):

            pixels[i] = color
            pixels.write()
            pixels[i] = (0, 0, 0)
            time.sleep_ms(100)
```

Los LEDs en la Tira RGB se iluminan uno por uno cuando se ejecuta el script. Tan pronto como se coloca un objeto frente al módulo de evitación de obstáculos, los LEDs en la Tira RGB se iluminan uno por uno en la dirección opuesta.

4.35 6.3 Teremín de Luz

El teremín es un instrumento musical electrónico que no requiere contacto físico. Basado en la posición de la mano del intérprete, produce diferentes tonos.

Su sección de control usualmente está compuesta por dos antenas metálicas que detectan la posición de las manos del tereminista y controlan los osciladores con una mano y el volumen con la otra. Las señales eléctricas del teremín se amplifican y envían a un altavoz.

No podemos reproducir el mismo instrumento a través del ESP32, pero podemos usar un fotorresistor y un zumbador pasivo para lograr una jugabilidad similar.

- Teremín - Wikipedia

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

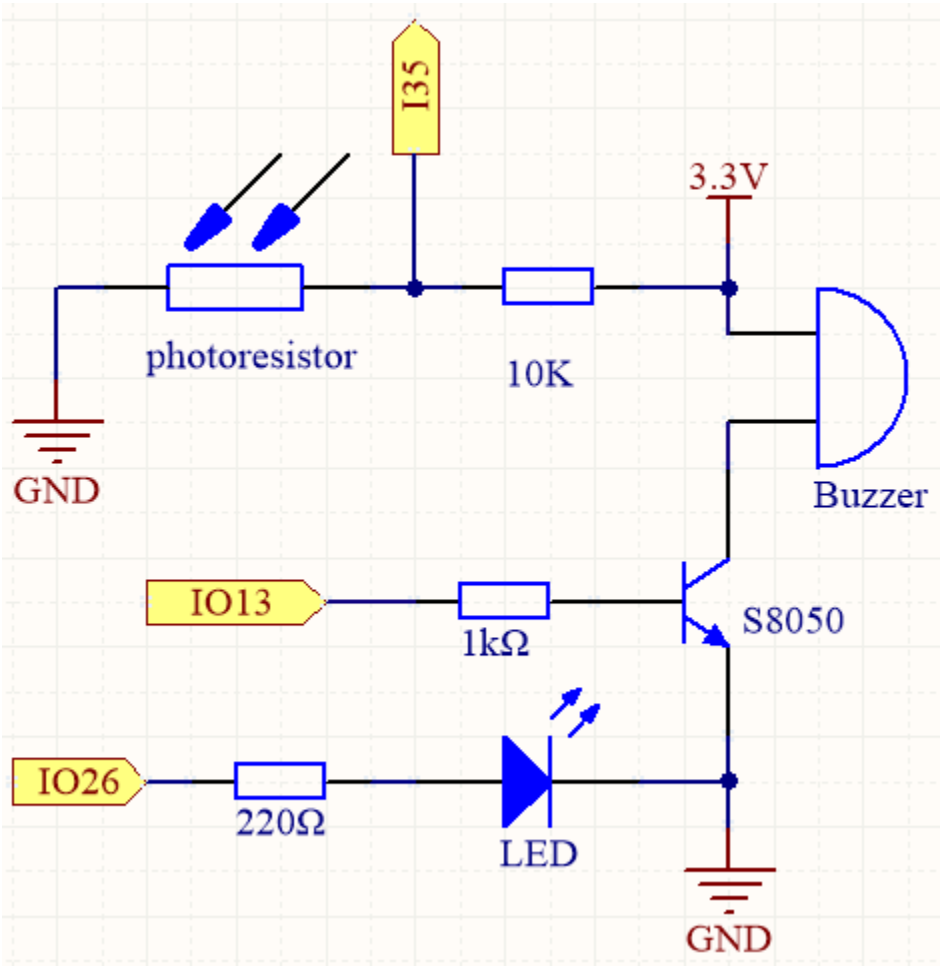
Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	
Fotorresistor	
Zumbador	
Transistor	

Esquemático

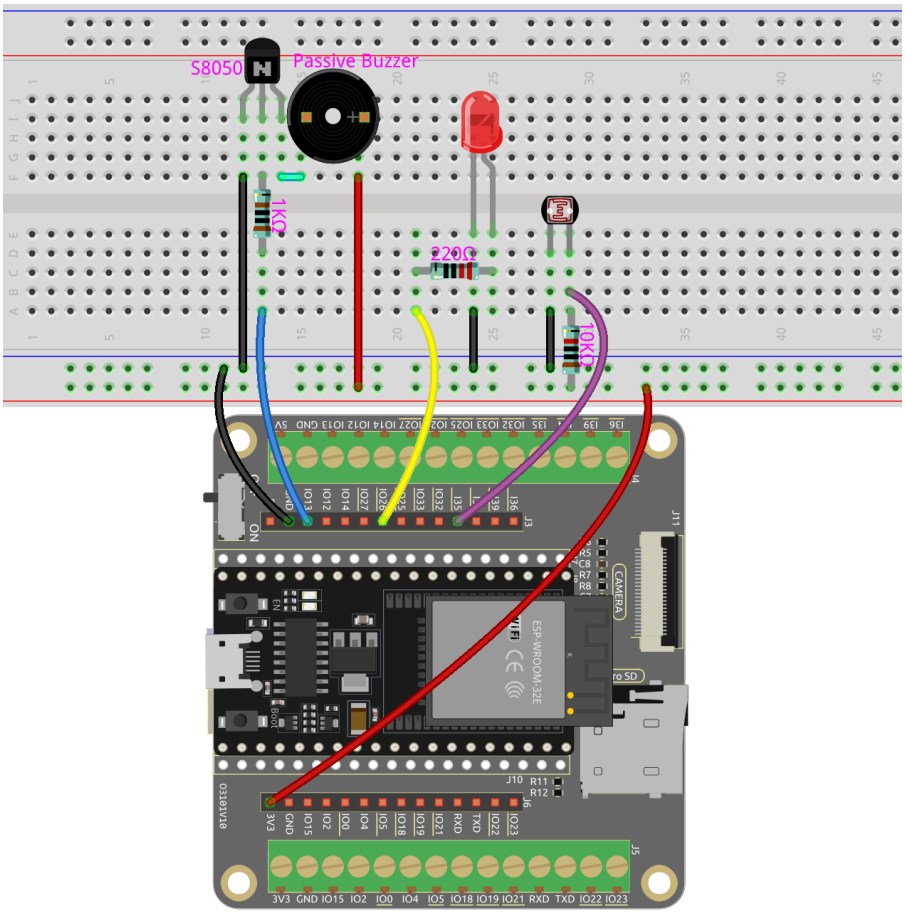


Antes de comenzar el proyecto, calibra el rango de intensidad de luz moviendo tu mano sobre el fotorresistor. El LED

conectado a IO26 se usa como indicador durante el proceso de calibración. Cuando el LED se enciende, significa el inicio de la calibración, y cuando se apaga, indica el fin de la calibración.

A medida que mueves tu mano sobre el fotorresistor, el valor de este cambiará en consecuencia. Utiliza este cambio para controlar el zumbador y reproducir diferentes notas musicales. Cada variación en el valor del fotorresistor puede mapearse a una nota musical específica, permitiendo que el zumbador produzca una melodía mientras mueves tu mano sobre el fotorresistor.

Conexión



Código

Nota:

- Abre el archivo `6.3_light_theremin.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, PWM, ADC
import time

# Inicializar el pin del LED
led = Pin(26, Pin.OUT)
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Inicializar el sensor de luz
sensor = ADC(Pin(35))
sensor.atten(ADC.ATTN_11DB)

# Inicializar el zumbador
buzzer = PWM(Pin(13), freq=440, duty=0)

light_low=4095
light_high=0

# Mapear el intervalo de valores de entrada a valores de salida
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Crear un tono usando el pin especificado, frecuencia y duración
def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty(512)
    time.sleep_ms(duration)
    pin.duty(0)

# Calibrar los valores máximos y mínimos del fotorresistor en 5 segundos.
timer_init_start = time.ticks_ms()
led.value(1) # encender el LED
while time.ticks_diff(time.ticks_ms(), timer_init_start)<5000:
    light_value = sensor.read()
    if light_value > light_high:
        light_high = light_value
    if light_value < light_low:
        light_low = light_value
led.value(0) # apagar el LED

# Reproducir los tonos basados en los valores de luz
while True:
    light_value  = sensor.read()
    pitch = int(interval_mapping(light_value,light_low,light_high,50,6000))
    if pitch > 50 :
        tone(buzzer,pitch,20)
    time.sleep_ms(10)
```

Al iniciar el programa, el LED se enciende, brindándonos una ventana de cinco segundos para calibrar el rango de detección del fotorresistor.

La calibración es un paso crucial ya que tiene en cuenta las diferentes condiciones de iluminación que podemos encontrar mientras usamos el dispositivo, como las variaciones de intensidad de luz durante diferentes momentos del día. Además, el proceso de calibración toma en cuenta la distancia entre nuestras manos y el fotorresistor, lo que determina el rango de juego del instrumento.

Una vez que el período de calibración termina, el LED se apaga, indicando que ahora podemos tocar el instrumento moviendo nuestras manos sobre el fotorresistor. Esta configuración nos permite crear música ajustando la altura de nuestras manos, proporcionando una experiencia interactiva y disfrutable.

4.36 6.4 Asistente de Reversa

Imagina esto: Estás en tu coche, a punto de aparcar en reversa en un lugar estrecho. Con nuestro proyecto, tendrás un módulo ultrasónico montado en la parte trasera de tu vehículo, actuando como un ojo digital. Al meter la marcha atrás, el módulo se activa, emitiendo pulsos ultrasónicos que rebotan en los obstáculos detrás de ti.

La magia ocurre cuando estos pulsos regresan al módulo. Calcula rápidamente la distancia entre tu coche y los objetos, transformando estos datos en retroalimentación visual en tiempo real mostrada en una vibrante pantalla LCD. Serás testigo de indicadores dinámicos y codificados por colores que representan la proximidad de los obstáculos, asegurando que tengas una comprensión cristalina del entorno circundante.

Pero no nos detuvimos ahí. Para sumergirte aún más en esta experiencia de conducción, incorporamos un zumbador animado. A medida que tu coche se acerca a un obstáculo, el tempo del zumbador se intensifica, creando una sinfonía auditiva de advertencias. Es como tener una orquesta personal guiándote a través de las complejidades del estacionamiento en reversa.

Este innovador proyecto combina tecnología de vanguardia con una interfaz de usuario interactiva, haciendo que tu experiencia de reversa sea segura y libre de estrés. Con el módulo ultrasónico, la pantalla LCD y el zumbador animado trabajando en armonía, te sentirás empoderado y confiado mientras maniobras en espacios estrechos, dejándote libre para concentrarte en el placer de conducir.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

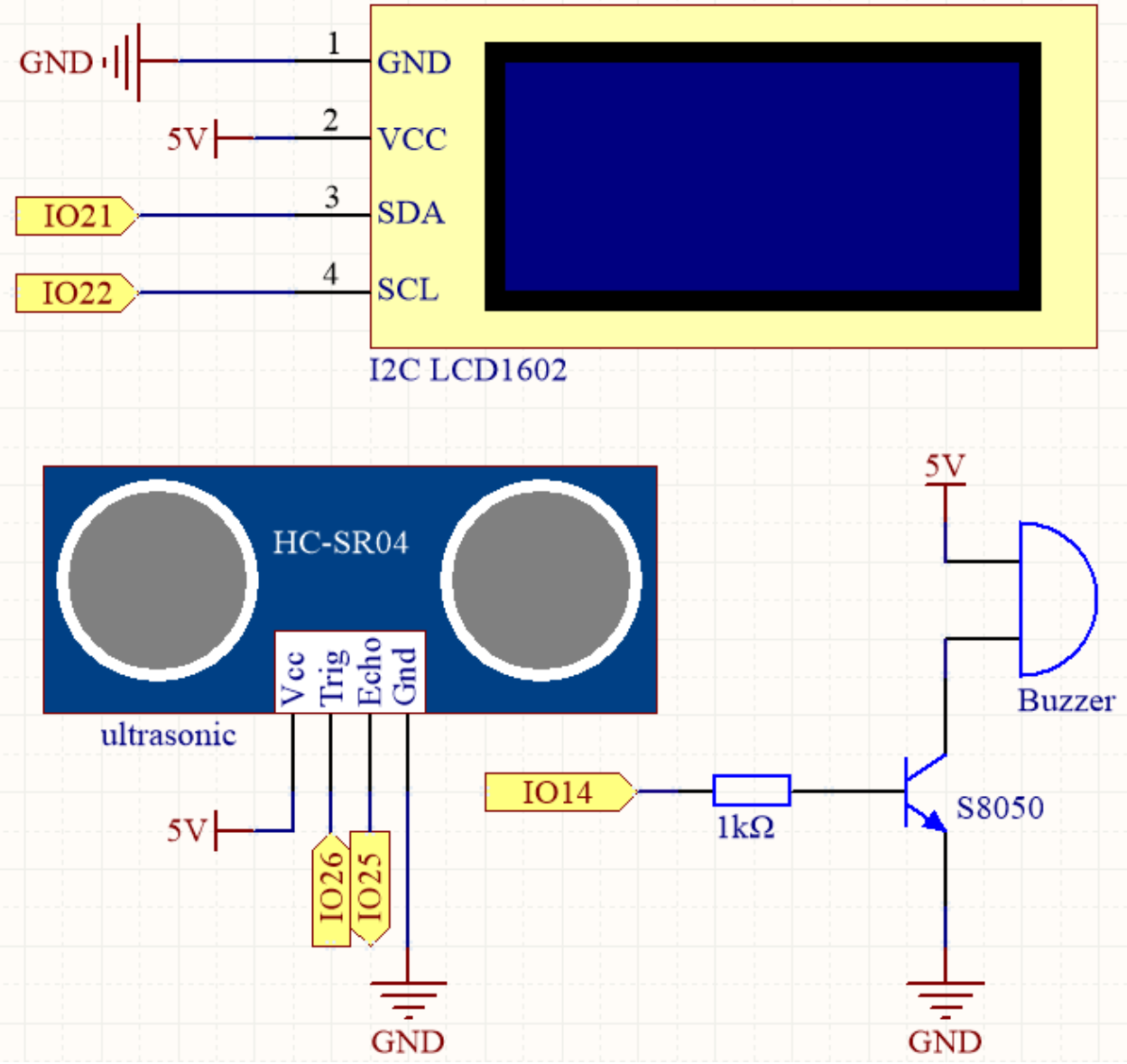
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

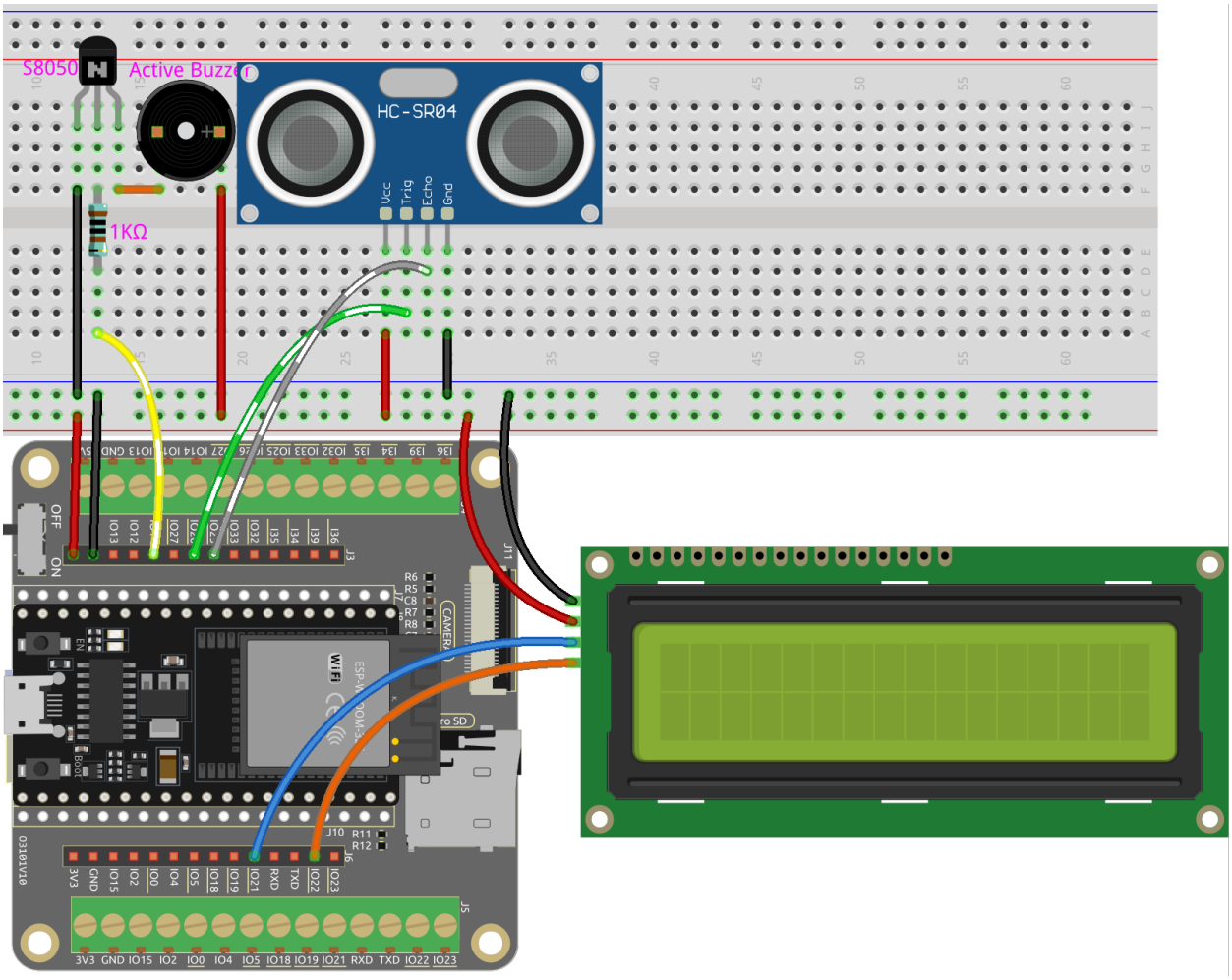
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Módulo Ultrasonido	
Zumbador	-
Transistor	
I2C LCD1602	

Esquemático



El sensor ultrasónico en el proyecto emite ondas de sonido de alta frecuencia y mide el tiempo que tardan en rebotar después de golpear un objeto. Analizando estos datos, se puede calcular la distancia entre el sensor y el objeto. Para proporcionar una advertencia cuando el objeto está demasiado cerca, se utiliza un zumbador para producir una señal audible. Además, la distancia medida se muestra en una pantalla LCD para una fácil visualización.

Conexión



Código

Nota:

- Abra el archivo `6.4_reversing_aid.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes` o copie y pegue el código en Thonny. Luego, haga clic en «Ejecutar script actual» o presione F5 para ejecutarlo.
- Asegúrese de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
# Import required libraries
from machine import Pin
import time
from lcd1602 import LCD
import _thread

# Initialize the buzzer
buzzer = Pin(14, Pin.OUT)

# Initialize the ultrasonic module
TRIG = Pin(26, Pin.OUT)
```

(continué en la próxima página)

(proviene de la página anterior)

```
ECHO = Pin(25, Pin.IN)

# Initialize the LCD1602 display
lcd = LCD()

dis = 100

# Calculate the distance
def distance():
    # Ensure trigger is off initially
    TRIG.off()
    time.sleep_us(2)  # Wait for 2 microseconds

    # Send a 10-microsecond pulse to the trigger pin
    TRIG.on()
    time.sleep_us(10)
    TRIG.off()

    # Wait for the echo pin to go high
    while not ECHO.value():
        pass

    # Record the time when the echo pin goes high
    time1 = time.ticks_us()

    # Wait for the echo pin to go low
    while ECHO.value():
        pass

    # Record the time when the echo pin goes low
    time2 = time.ticks_us()

    # Calculate the time difference between the two recorded times
    during = time.ticks_diff(time2, time1)

    # Calculate and return the distance (in cm) using the speed of sound (340 m/s)
    return during * 340 / 2 / 10000

# Thread to continuously update the ultrasonic sensor reading
def ultrasonic_thread():
    global dis
    while True:
        dis = distance()

        # Clear the LCD screen
        lcd.clear()

        # Display the distance
        lcd.write(0, 0, 'Dis: %.2f cm' % dis)
        time.sleep(0.5)

# Start the ultrasonic sensor reading thread
```

(continué en la próxima página)

(proviene de la página anterior)

```
_thread.start_new_thread(ultrasonic_thread, ())

# Beep function for the buzzer
def beep():
    buzzer.value(1)
    time.sleep(0.1)
    buzzer.value(0)
    time.sleep(0.1)

# Initialize the intervals variable
intervals = 10000000
previousMills = time.ticks_ms()
time.sleep(1)

# Main loop
while True:
    # Update intervals based on distance
    if dis < 0 and dis > 500:
        pass
    elif dis <= 10:
        intervals = 300
    elif dis <= 20:
        intervals = 500
    elif dis <= 50:
        intervals = 1000
    else:
        intervals = 2000

    # Print the distance if it's not -1
    if dis != -1:
        print('Distance: %.2f' % dis)
    time.sleep_ms(100)

    # Check if it's time to beep
    currentMills = time.ticks_ms()
    if time.ticks_diff(currentMills, previousMills) >= intervals:
        beep()
        previousMills = currentMills
```

- Cuando el script esté en ejecución, el módulo ultrasónico detectará continuamente la distancia de los obstáculos frente a él y mostrará la distancia en el Shell y en el LCD I2C 1602.
- A medida que el obstáculo se acerca, la frecuencia de los pitidos del zumbador se volverá más rápida.
- La función `ultrasonic_thread()` se ejecuta en un hilo separado para que pueda actualizar la medición de distancia continuamente sin bloquear el bucle principal.

Nota: Si el código y el cableado son correctos, pero el LCD aún no logra mostrar contenido, puede ajustar el potenciómetro en la parte trasera para aumentar el contraste.

4.37 6.5 Degradado de Color

¿Estás listo para experimentar un mundo de color? Este proyecto te llevará a un viaje mágico donde podrás controlar una tira de LED y lograr transiciones suaves de color. Ya sea que busques agregar algo de color a tu decoración del hogar o busques un proyecto de programación divertido, este proyecto lo tiene todo. ¡Sumérgete juntos en este mundo colorido!

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

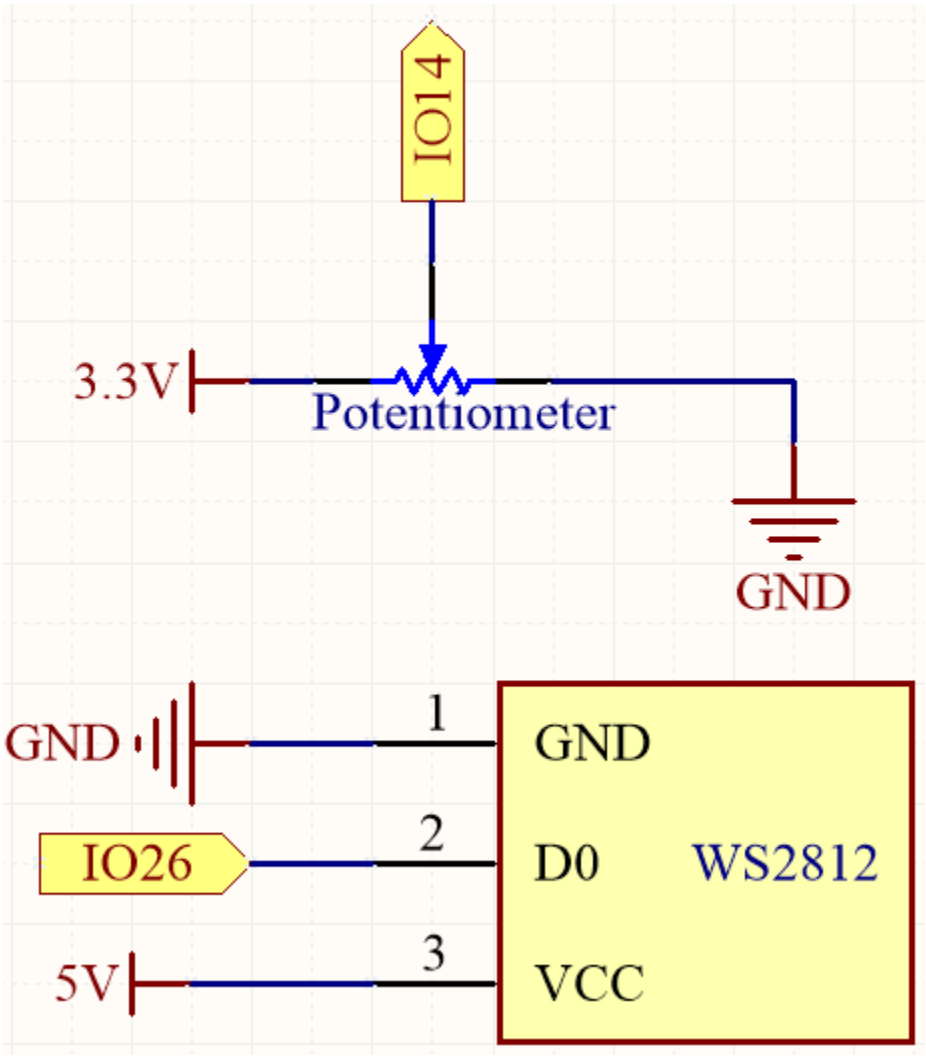
Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

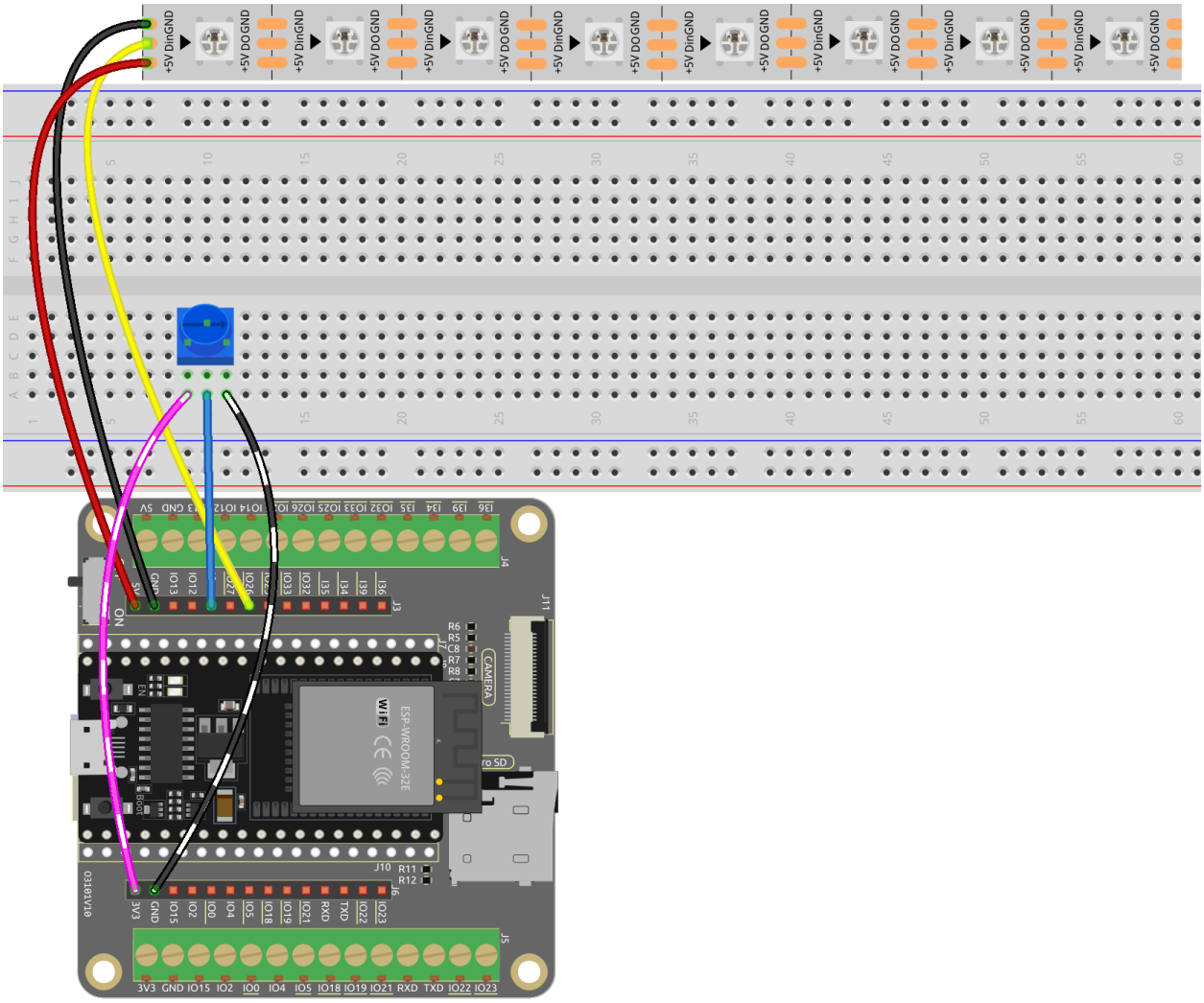
INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Potenciómetro	
Tira de 8 LEDs RGB WS2812	

Esquemático



Este proyecto utiliza una tira de LED y un potenciómetro para crear un efecto de mezcla de colores. El potenciómetro se utiliza para ajustar el valor del tono del LED, que luego se convierte en valores RGB utilizando una función de conversión de color. Los valores RGB se utilizan para actualizar el color del LED.

Cableado



Código

Nota:

- Abre el archivo `6.5_color_gradient.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import Pin, ADC, PWM
import neopixel
import time

NUM_LEDS = 8  # Number of LEDs in the strip
PIN_NUM = 26  # LED strip
POT_PIN = 14  # Potentiometer

# Initialize the potentiometer
```

(continué en la próxima página)

(proviene de la página anterior)

```
potentiometer = ADC(Pin(POT_PIN))
potentiometer.atten(ADC.ATTN_11DB)

# Initialize the NeoPixel LED strip
np = neopixel.NeoPixel(Pin(PIN_NUM), NUM_LEDS)

# Function to convert HSL color space to RGB color space
def hsl_to_rgb(h, s, l):
    # Helper function to convert hue to RGB
    def hue_to_rgb(p, q, t):
        if t < 0:
            t += 1
        if t > 1:
            t -= 1
        if t < 1/6:
            return p + (q - p) * 6 * t
        if t < 1/2:
            return q
        if t < 2/3:
            return p + (q - p) * (2/3 - t) * 6
        return p

    if s == 0:
        r = g = b = l
    else:
        q = l * (1 + s) if l < 0.5 else l + s - l * s
        p = 2 * l - q
        r = hue_to_rgb(p, q, h + 1/3)
        g = hue_to_rgb(p, q, h)
        b = hue_to_rgb(p, q, h - 1/3)

    return (int(r * 255), int(g * 255), int(b * 255))

# Function to set the color of all LEDs in the strip
def set_color(np, color):
    for i in range(NUM_LEDS):
        np[i] = color
    np.write()

# Main loop
while True:
    # Read the potentiometer value and normalize it to the range [0, 1]
    pot_value = potentiometer.read() / 4095.0
    hue = pot_value  # Set hue value based on the potentiometer's position
    saturation = 1  # Set saturation to 1 (fully saturated)
    lightness = 0.5  # Set lightness to 0.5 (halfway between black and white)

    # Convert the HSL color to RGB
    current_color = hsl_to_rgb(hue, saturation, lightness)

    # Set the LED strip color based on the converted RGB value
    set_color(np, current_color)
```

(continué en la próxima página)

(proviene de la página anterior)

```
    # Sleep for a short period to allow for smooth transitions
    time.sleep(0.1)
```

A medida que el código se ejecuta, gira lentamente el potenciómetro y verás cómo el color de la Tira RGB se desvanece de rojo a morado.

4.38 6.6 Dado Digital

Este proyecto se basa en el proyecto *2.5 Visualización de Números* añadiendo un botón para controlar el dígito mostrado en el display de siete segmentos.

Al presionar el botón, el display de siete segmentos recorre los números del 1 al 6, y al soltar el botón, muestra un número aleatorio.

Este ciclo continúa cada vez que se presiona el botón.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

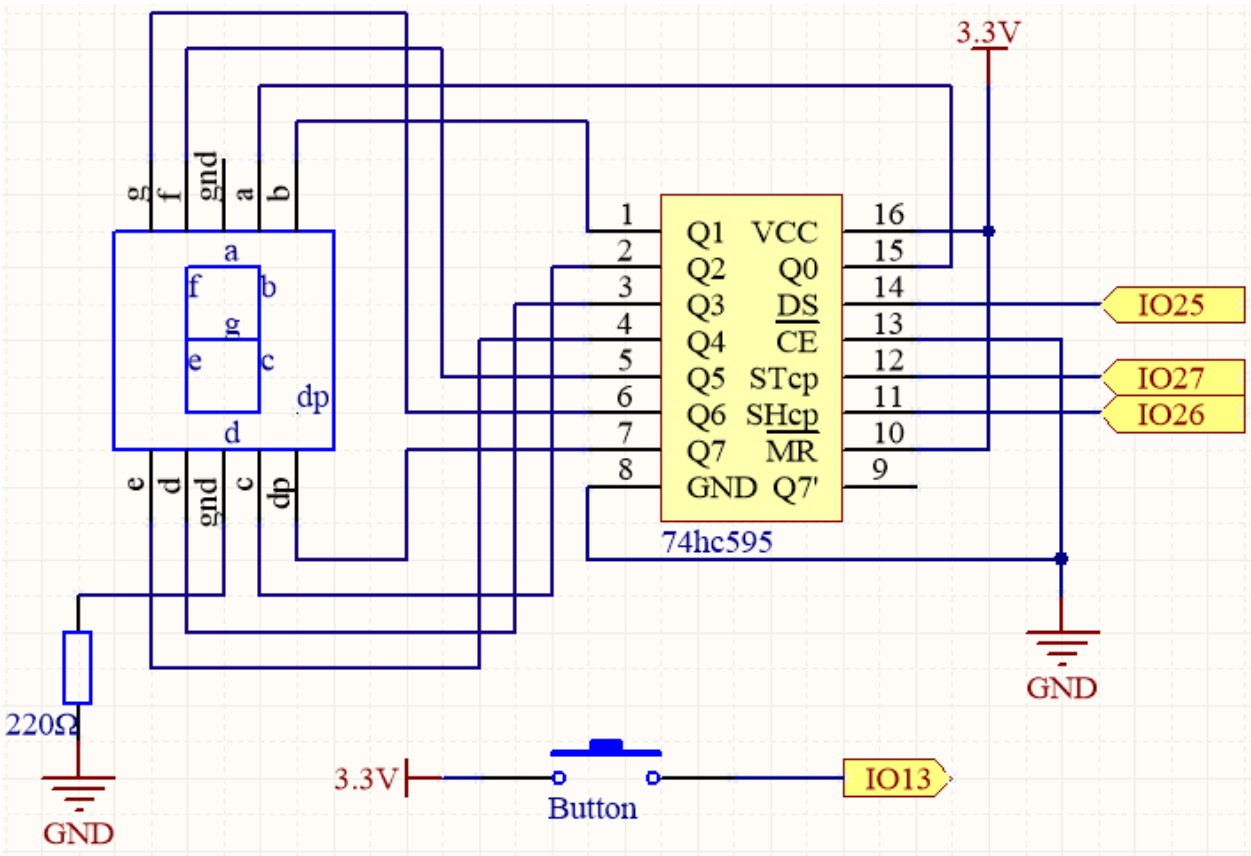
Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
74HC595	
display de 7 Segmentos	
Botón	

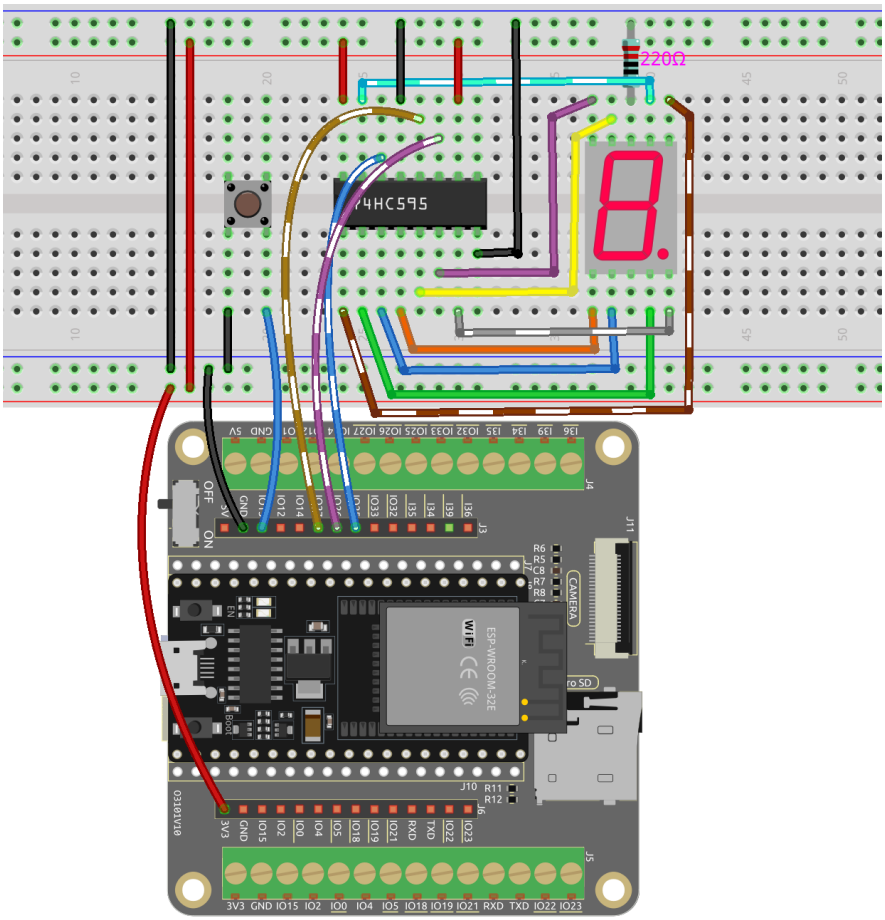
Esquemático



Este proyecto se basa en el proyecto *2.5 Visualización de Números* añadiendo un botón para controlar el dígito mostrado en el display de siete segmentos.

El botón se conecta directamente a IO13 sin una resistencia de pull-up o pull-down externa porque IO13 tiene una resistencia de pull-up interna de 47K, eliminando la necesidad de una resistencia externa adicional.

Cableado



Código

Nota:

- Abre el archivo `6.6_digital_dice.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
import machine
import time
import random

# Define the segment code for a common anode 7-segment display
SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp

button = machine.Pin(13, machine.Pin.IN) # Button pin
```

(continué en la próxima página)

(proviene de la página anterior)

```
# Define the hc595_shift function to shift data into the 74HC595 shift register
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

# Initialize the random seed
random.seed(time.ticks_us())


num = 1
button_state = False

# Define the button callback function to toggle the button state
def button_callback(pin):
    global button_state
    button_state = not button_state

# Attach the button callback function to the falling edge of the button pin
button.irq(trigger=machine.Pin.IRQ_FALLING, handler=button_callback)

# Continuously display the current digit on the 7-segment display, scrolling if button␣
↪is not pressed
while True:

    # Display the current digit on the 7-segment display
    hc595_shift(SEGCODE[num])

    # If the button is pressed and button state is True
    if button_state:
        pass

    # If the button is pressed again and button state i False, generate a new random␣
↪digit
    if not button_state:
```

(continué en la próxima página)

(proviene de la página anterior)

```
        num = random.randint(1, 6)
        time.sleep_ms(10) # Adjust this value to control the display refresh rate
```

Mientras el programa está en ejecución, presionar el botón hará que el display de siete segmentos recorra y muestre aleatoriamente un número entre 1 y 6.

Al presionar el botón nuevamente, el display de siete segmentos se detendrá y revelará un número específico. Presiona el botón una vez más, y el display de siete segmentos reanudará el desplazamiento a través de los dígitos.

4.39 6.7 Adivina el Número

¿Te sientes afortunado? ¿Quieres poner a prueba tu intuición a ver si puedes adivinar el número correcto? ¡Entonces este es el juego para ti!

Con este proyecto, puedes jugar un emocionante juego de azar.

Usando un control remoto IR, los jugadores introducen números entre 0 y 99 para intentar adivinar el número de la suerte generado aleatoriamente. El sistema muestra el número introducido por el jugador en una pantalla LCD, junto con pistas de límites superior e inferior para guiar al jugador hacia la respuesta correcta. Con cada intento, los jugadores se acercan al número de la suerte, hasta que finalmente, alguien acierta el número y gana el juego.

Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

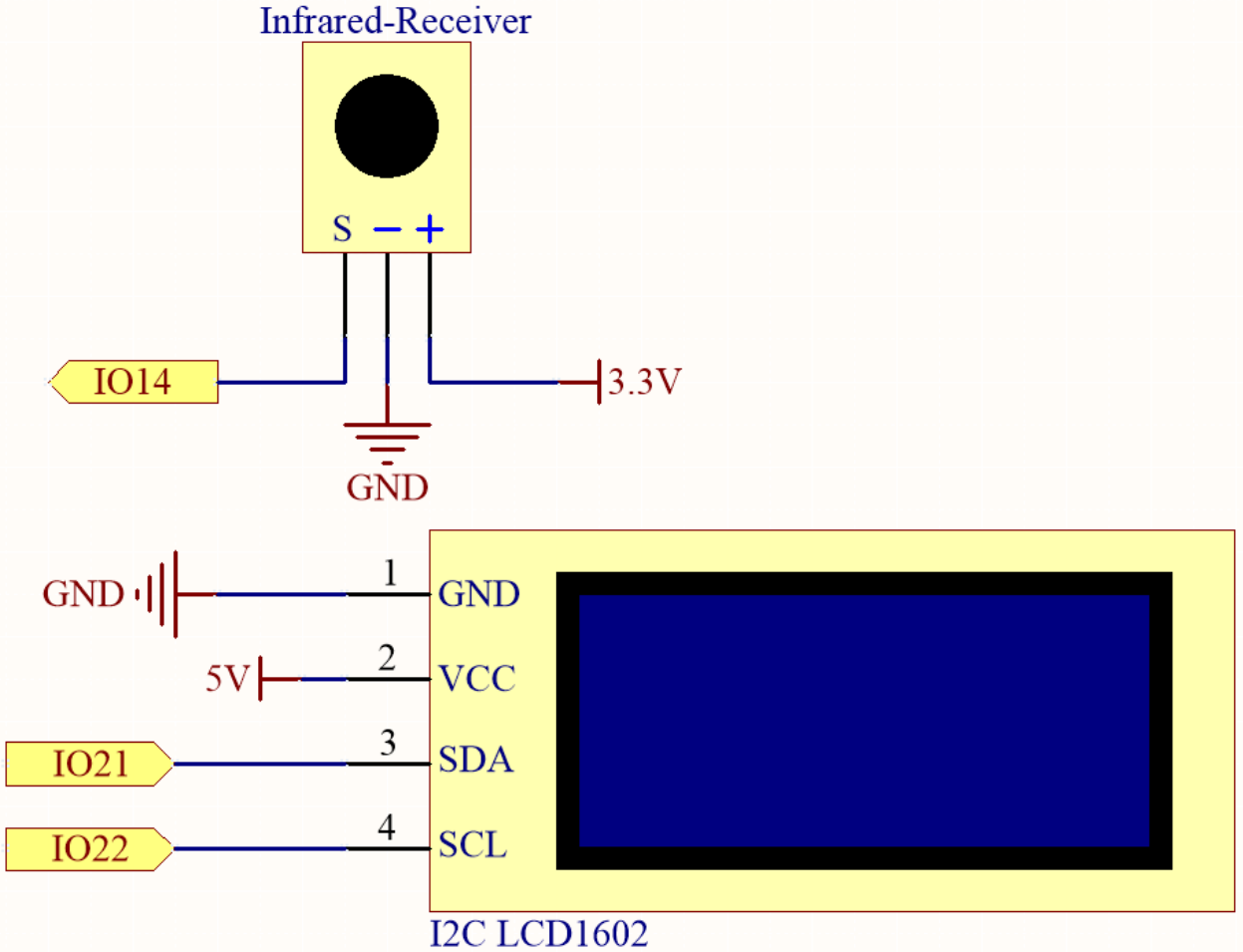
Es definitivamente conveniente comprar un kit completo, aquí tienes el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

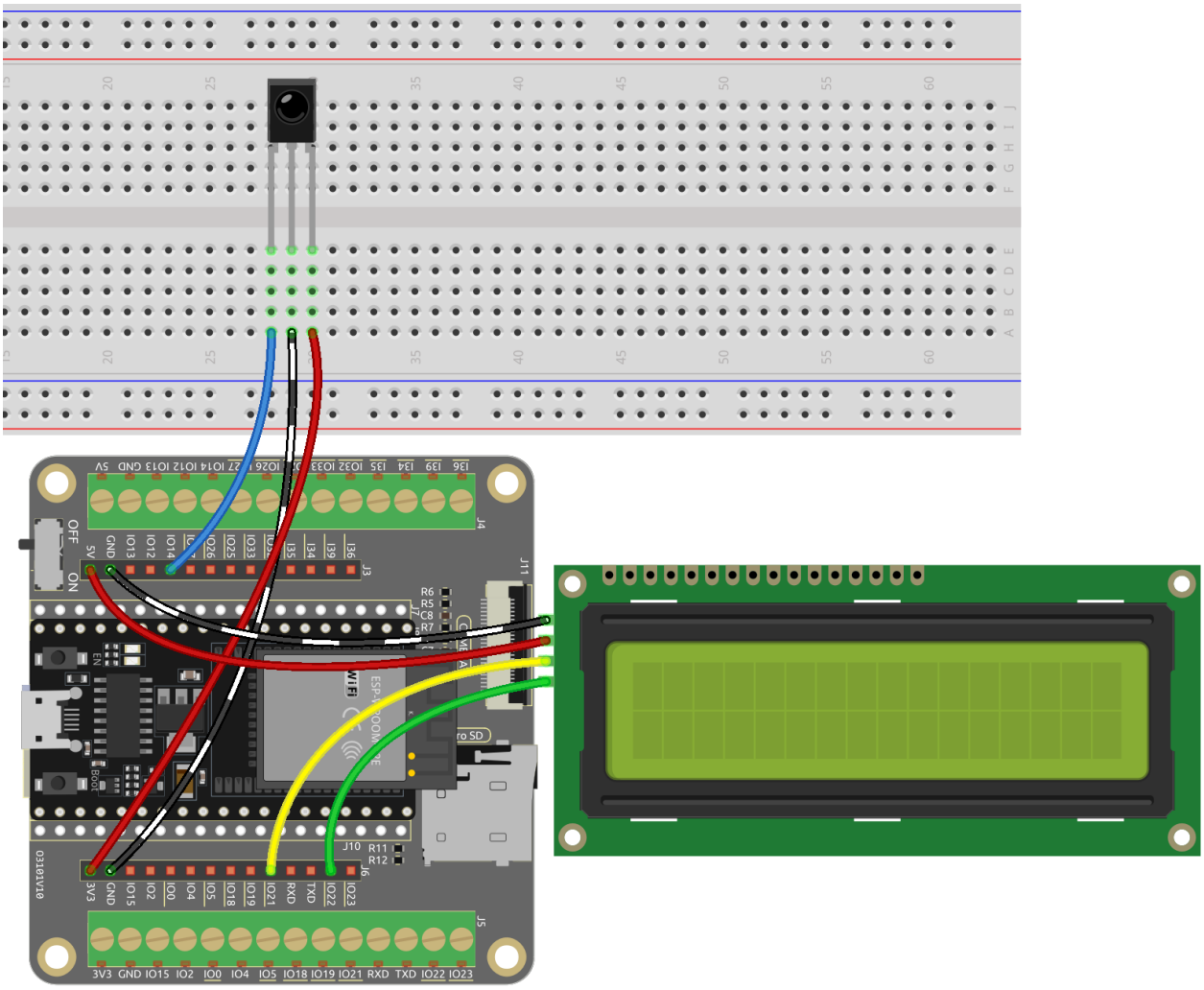
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DE COMPONENTES	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Receptor de Infrarrojos	
I2C LCD1602	

Esquemático


Infrared-Receiver
S
IO14
3.3V
GND
GND
1
GND
5V
2
VCC
IO21
3
SDA
IO22
4
SCL
I2C LCD1602

Cableado



Código

Nota:

- Abre el archivo `6.7_game_guess_number.py` ubicado en la ruta `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar Script Actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.
- Se utilizan las bibliotecas `lcd1602.py` y `ir_rx` aquí, verifica si se han cargado en el ESP32. Consulta *1.4 Subir las Bibliotecas (Importante)* para obtener un tutorial.

```
from lcd1602 import LCD
import machine
import time
import urandom
from machine import Pin
from ir_rx.print_error import print_error
from ir_rx.nec import NEC_8
```

(continué en la próxima página)

(proviene de la página anterior)

```
# IR receiver configuration
pin_ir = Pin(14, Pin.IN)

# Initialize the guessing game variables
lower = 0
upper = 99
pointValue = int(urandom.uniform(lower, upper))
count = 0

# Initialize the LCD1602 display
lcd = LCD()

# Initialize a new random value for the game
def init_new_value():
    global pointValue, upper, lower, count
    pointValue = int(urandom.uniform(lower, upper))
    print(pointValue)
    upper = 99
    lower = 0
    count = 0
    return False

# Display messages on the LCD based on the game state
def lcd_show(result):
    global count
    lcd.clear()
    if result == True:
        string = "GAME OVER!\n"
        string += "Point is " + str(pointValue)
    else:
        string = "Enter number: " + str(count) + "\n"
        string += str(lower) + " < Point < " + str(upper)
    lcd.message(string)
    return

# Process the entered number and update the game state
def number_processing():
    global upper, count, lower
    if count > pointValue:
        if count < upper:
            upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        return True
    count = 0
    return False

# Process the key inputs from the IR remote control
def process_key(key):
```

(continué en la próxima página)

(proviene de la página anterior)

```
    global count, lower, upper, pointValue, result
    if key == "Power":
        init_new_value()
        lcd_show(False)
    elif key == "+":
        result = number_processing()
        lcd_show(result)
        if result:
            time.sleep(5)
            init_new_value()
            lcd_show(False)
        else:
            lcd_show(False)
    elif key.isdigit():
        count = count * 10 + int(key) if count * 10 + int(key) <= 99 else count
        lcd_show(False)

# Decode the received data and return the corresponding key name
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
        return "3"
    if data == 0x08:
        return "4"
    if data == 0x1C:
        return "5"
    if data == 0x5A:
        return "6"
    if data == 0x42:
        return "7"
    if data == 0x52:
        return "8"
    if data == 0x4A:
        return "9"
    if data == 0x09:
        return "+"
    if data == 0x15:
        return "-"
    if data == 0x7:
        return "EQ"
    if data == 0x0D:
        return "U/SD"
    if data == 0x19:
        return "CYCLE"
    if data == 0x44:
        return "PLAY/PAUSE"
    if data == 0x43:
```

(continué en la próxima página)

(proviene de la página anterior)

```
        return "FORWARD"
    if data == 0x40:
        return "BACKWARD"
    if data == 0x45:
        return "POWER"
    if data == 0x47:
        return "MUTE"
    if data == 0x46:
        return "MODE"
    return "ERROR"

def callback(data, addr, ctrl):
    if data < 0:
        pass
    else:
        key = decodeKeyValue(data)
        if key != "ERROR":
            process_key(key)

# Initialize the IR receiver object with the callback function
ir = NEC_8(pin_ir, callback)

# ir.error_function(print_error)

# Initialize the game with a new random value
init_new_value()

# Show the initial game state on the LCD
lcd_show(False)

try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()
```

- Cuando se ejecuta el código, se genera un número secreto que no se muestra en el LCD, y lo que necesitas hacer es adivinarlo.
- Presiona el número que adivinaste en el control remoto, luego presiona la tecla + para confirmar.
- Simultáneamente, el rango mostrado en el LCD I2C 1602 disminuirá, y debes presionar el número adecuado basado en este nuevo rango.
- Si aciertas el número de la suerte, ya sea por suerte o por desgracia, aparecerá `¡JUEGO TERMINADO!`.

Nota: Si el código y el cableado son correctos, pero el LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte trasera para aumentar el contraste.

¿Cómo funciona?

A continuación, se presenta un análisis detallado de parte del código.

1. Inicializar las variables del juego de adivinanzas.

```
lower = 0
upper = 99
pointValue = int(urandom.uniform(lower, upper))
count = 0
```

- `lower` y `upper` son los límites para el número secreto.
- El número secreto (`valorPunto`) se genera aleatoriamente entre los límites `lower` y `upper`.
- La suposición actual del usuario (`count`).

2. Esta función restablece los valores del juego de adivinanzas y genera un nuevo número secreto.

```
def init_new_value():
    global pointValue, upper, lower, count
    pointValue = int(urandom.uniform(lower, upper))
    print(pointValue)
    upper = 99
    lower = 0
    count = 0
    return False
```

3. Esta función muestra el estado actual del juego en la pantalla LCD.

```
def lcd_show(result):
    global count
    lcd.clear()
    if result == True:
        string = "GAME OVER!\n"
        string += "Point is " + str(pointValue)
    else:
        string = "Enter number: " + str(count) + "\n"
        string += str(lower) + " < Point < " + str(upper)
    lcd.message(string)
    return
```

- Si el juego ha terminado (`result=True`), muestra `¡JUEGO TERMINADO!` y el número secreto.
- De lo contrario, muestra la suposición actual (`count`) y el rango de suposición actual (`lower` a `upper`)

4. Esta función procesa la suposición actual del usuario (`count`) y actualiza el rango de suposición.

```
def number_processing():
    global upper, count, lower
    if count > pointValue:
        if count < upper:
            upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        return True
    count = 0
    return False
```

- Si la suposición actual (`count`) es más alta que el número secreto, se actualiza el límite superior.

- Si la suposición actual (`count`) es más baja que el número secreto, se actualiza el límite inferior.
- Si la suposición actual (`count`) es igual al número secreto, la función devuelve `True` (juego terminado).

5. Esta función procesa los eventos de pulsación de teclas recibidos del control remoto IR.

```
def process_key(key):
    global count, lower, upper, pointValue, result
    if key == "Power":
        init_new_value()
        lcd_show(False)
    elif key == "+":
        result = number_processing()
        lcd_show(result)
        if result:
            time.sleep(5)
            init_new_value()
            lcd_show(False)
        else:
            lcd_show(False)
    elif key.isdigit():
        count = count * 10 + int(key) if count * 10 + int(key) <= 99 else␣
↪count
        lcd_show(False)
```

- Si se presiona la tecla `Power`, el juego se reinicia.
- Si se presiona la tecla +, se procesa la suposición actual (`count`) y se actualiza el estado del juego.
- Si se presiona una tecla de dígito, se actualiza la suposición actual (`count`) con el nuevo dígito.

6. Esta función de callback se activa cuando el receptor IR recibe

```
def callback(data, addr, ctrl):
    if data < 0:
        pass
    else:
        key = decodeKeyValue(data)
        if key != "ERROR":
            process_key(key)
```

4.40 6.8 Monitor de Plantas

¡Bienvenido al proyecto Monitor de Plantas!

En este proyecto, utilizaremos una placa ESP32 para crear un sistema que nos ayude a cuidar de nuestras plantas. Con este sistema, podemos monitorear la temperatura, humedad, humedad del suelo y niveles de luz de nuestras plantas, y asegurarnos de que reciban el cuidado y atención que necesitan para prosperar.

Componentes Requeridos

Para este proyecto, necesitamos los siguientes componentes.

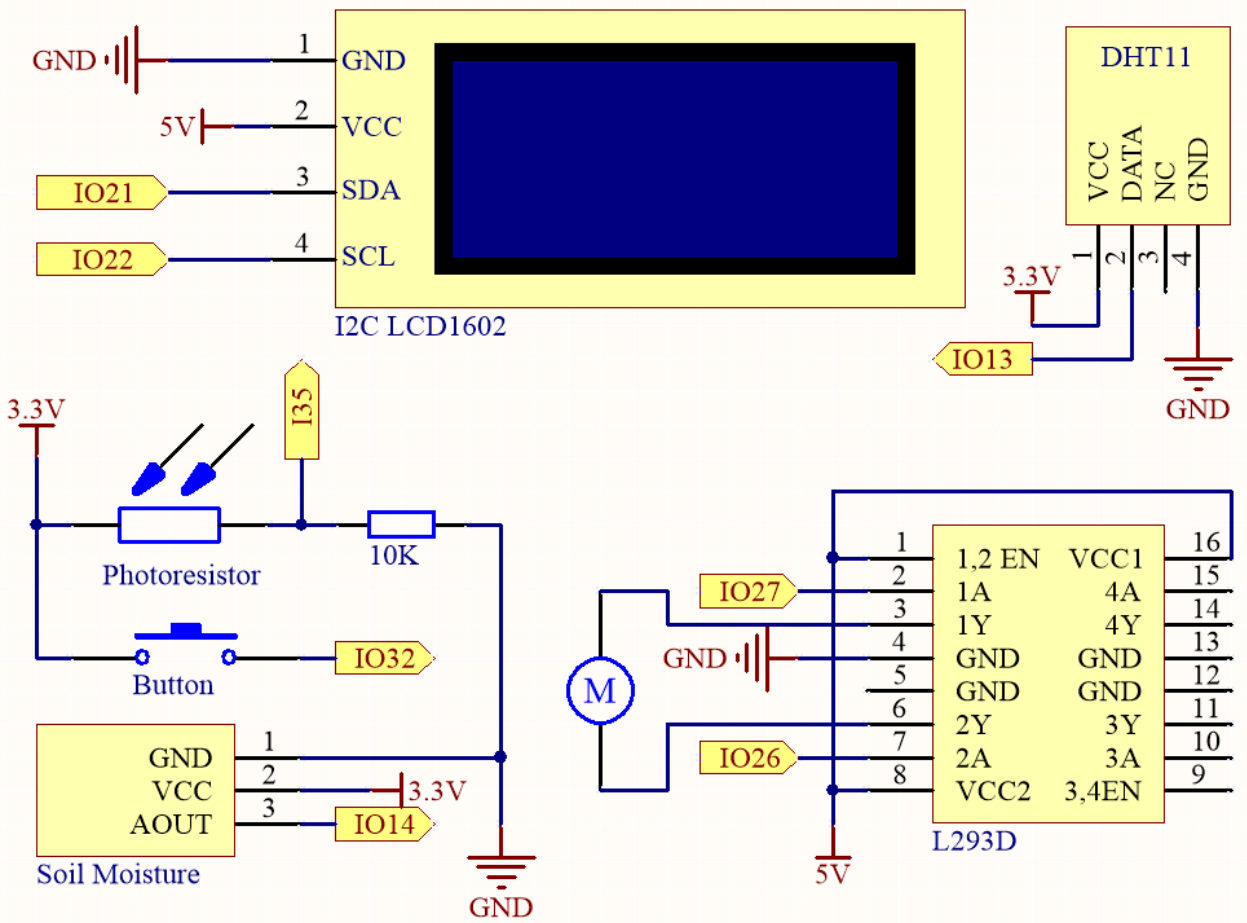
Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Sensor de Humedad y Temperatura DHT11	
I2C LCD1602	
Bomba Centrífuga	-
L293D	-
Botón	
Fotorresistor	
Módulo de Humedad del Suelo	

Esquemático

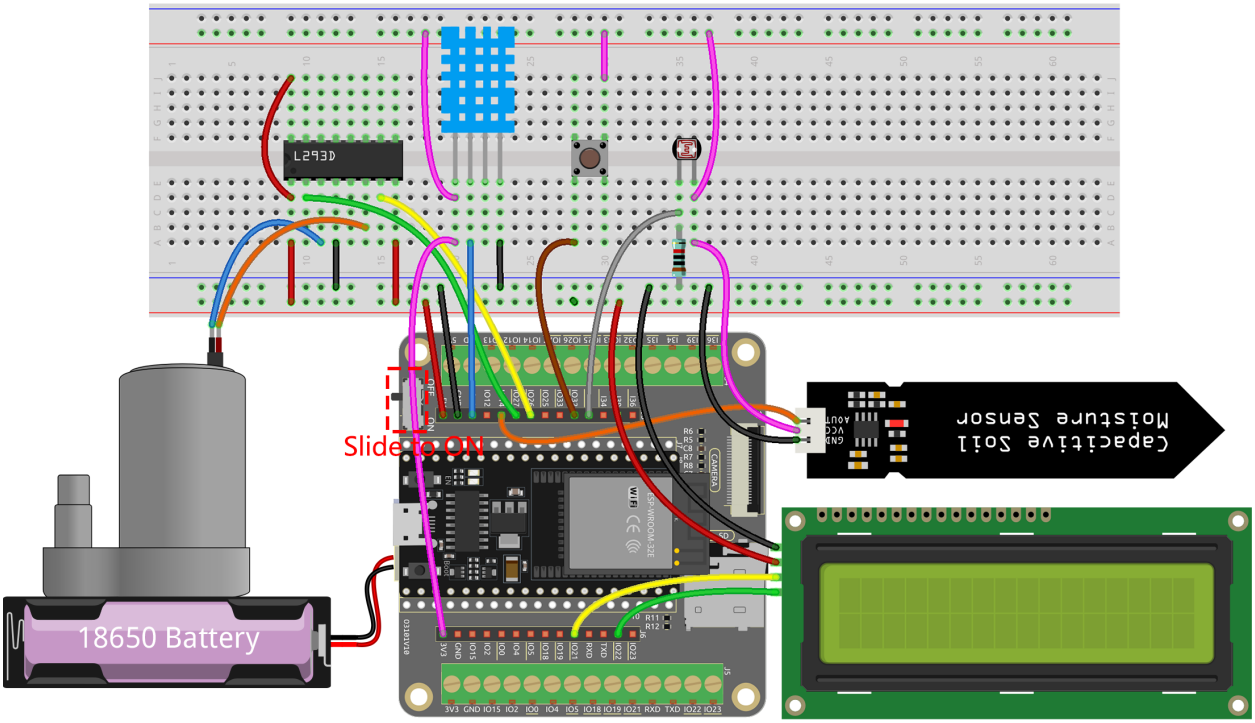


El sistema utiliza un sensor DHT11 para medir los niveles de temperatura y humedad del entorno circundante. Mientras tanto, un módulo de humedad del suelo se utiliza para medir el nivel de humedad del suelo y un fotoresistor se utiliza para medir el nivel de luz. Las lecturas de estos sensores se muestran en una pantalla LCD, y se puede controlar una bomba de agua usando un botón para regar la planta cuando sea necesario.

IO32 tiene una resistencia interna de pull-down de 1K, y por defecto, está en un nivel lógico bajo. Cuando se presiona el botón, se establece una conexión a VCC (alto voltaje), resultando en un nivel lógico alto en IO32.

Conexión

Nota: Se recomienda aquí insertar la batería y luego deslizar el interruptor en la placa de expansión a la posición ON para activar el suministro de la batería.



Código

Nota:

- Abre el archivo `6.8_plant_monitor.py` ubicado en el camino `esp32-starter-kit-main\micropython\codes`, o copia y pega el código en Thonny. Luego, haz clic en «Ejecutar script actual» o presiona F5 para ejecutarlo.
- Asegúrate de seleccionar el intérprete «MicroPython (ESP32).COMxx» en la esquina inferior derecha.

```
from machine import ADC, Pin
import time
import dht
from lcd1602 import LCD

# DHT11
dht11 = dht.DHT11(Pin(13))

# Humedad del suelo
moisture_pin = ADC(Pin(14))
moisture_pin.atten(ADC.ATTN_11DB)

# Fotoresistor
photoresistor = ADC(Pin(35))
photoresistor.atten(ADC.ATTN_11DB)

# Botón y bomba
button = Pin(32, Pin.IN)
```

(continué en la próxima página)

(proviene de la página anterior)

```
motor1A = Pin(27, Pin.OUT)
motor2A = Pin(26, Pin.OUT)

# Configuración del LCD I2C1602
lcd = LCD()

# Rotar la bomba
def rotate():
motor1A.value(1)
motor2A.value(0)

# Detener la bomba
def stop():
motor1A.value(0)
motor2A.value(0)

estado_del_boton = False

# Definir la función de callback del botón para alternar el estado del botón
def button_callback(pin):
global estado_del_boton
estado_del_boton = not estado_del_boton

# Adjuntar la función de callback del botón al borde ascendente del pin del botón
button.irq(trigger=Pin.IRQ_RISING, handler=button_callback)

pagina = 0
temp = 0
humi = 0

try:
while True:

      # Si el botón está presionado y el estado del botón es Verdadero
      if estado_del_boton:
            print("rotar")
            rotate()

      # Si el botón está presionado de nuevo y el estado del botón es Falso
      if not estado_del_boton:
            print("detener")
            stop()
      time.sleep(2)

      # Limpiar la pantalla LCD
      lcd.clear()

      # Alternar el valor de la variable pagina entre 0 y 1
      pagina=(pagina+1)%2

      # Cuando la pagina es 1, mostrar temperatura y humedad en el LCD1602
      if pagina is 1:
```

(continué en la próxima página)

(proviene de la página anterior)

```
            try:
            # Medir temperatura y humedad
            dht11.measure()

            # Obtener valores de temperatura y humedad
            temp = dht11.temperature()
            humi = dht11.humidity()
            except Exception as e:
            print("Error: ", e)

            # Mostrar temperatura y humedad
            lcd.write(0, 0, "Temp: {}\xDFC".format(temp))
            lcd.write(0, 1, "Humi: {}%".format(humi))

      # Si la pagina es 0, mostrar la humedad del suelo y luz
      else:
            luz = photoresistor.read()
            humedad = moisture_pin.read()

            # Limpiar la pantalla LCD
            lcd.clear()

            # Mostrar el valor de la humedad del suelo y luz
            lcd.write(0, 0, f"Humedad: {humedad}")
            lcd.write(0, 1, f"Luz: {luz}")

except KeyboardInterrupt:
# Detener el motor cuando se captura KeyboardInterrupt
stop()
```

- Cuando el código está ejecutando, el LCD I2C1602 muestra alternativamente temperatura y humedad, así como valores analógicos de humedad del suelo e intensidad de luz, con un intervalo de 2 segundos.
- Presiona el botón para iniciar la bomba de agua, y presiónalo de nuevo para detener la bomba de agua.

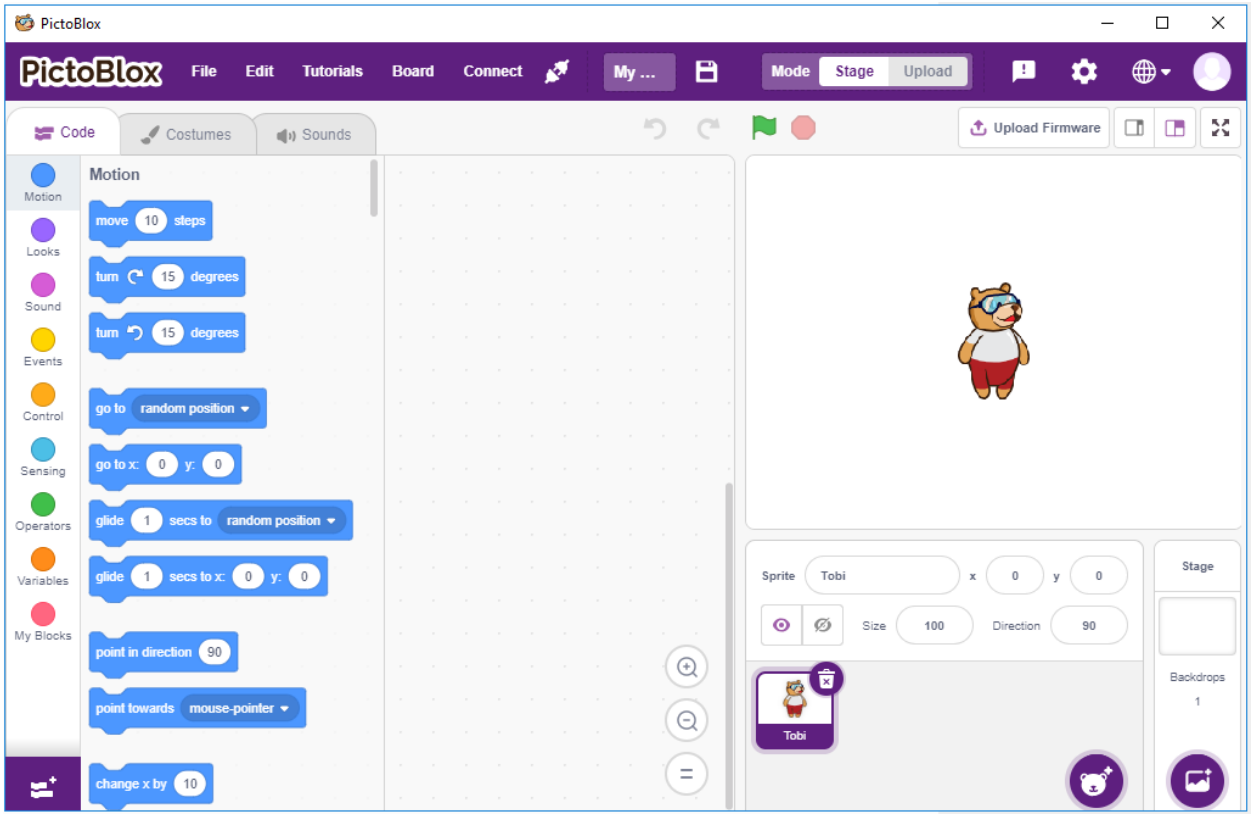
Nota: Si el código y la conexión son correctos, pero el LCD aún no muestra ningún contenido, puedes ajustar el potenciómetro en la parte trasera para aumentar el contraste.

CAPÍTULO 5

Jugar con Scratch

Además de programar en el Arduino IDE o en Thonny IDE, también podemos utilizar la programación gráfica.

Aquí recomendamos programar con Scratch, pero el Scratch oficial actualmente solo es compatible con Raspberry Pi, por lo que hemos establecido una colaboración con una empresa, STEMPedia, que ha desarrollado un software de programación gráfica basado en Scratch 3 para muchas placas - PictoBlox.



Mantiene las funciones básicas de Scratch 3, pero también añade placas de control, como Arduino Uno, Mega, Nano,

ESP32, Microbit y placas principales caseras de STEAMPedia, que pueden utilizar sensores externos y robots para controlar los sprites en el escenario, con sólidas capacidades de interacción con hardware.

Además, cuenta con inteligencia artificial y aprendizaje automático, incluso si no tienes muchos conocimientos de programación, puedes aprender y utilizar estas tecnologías populares y de alta tecnología.

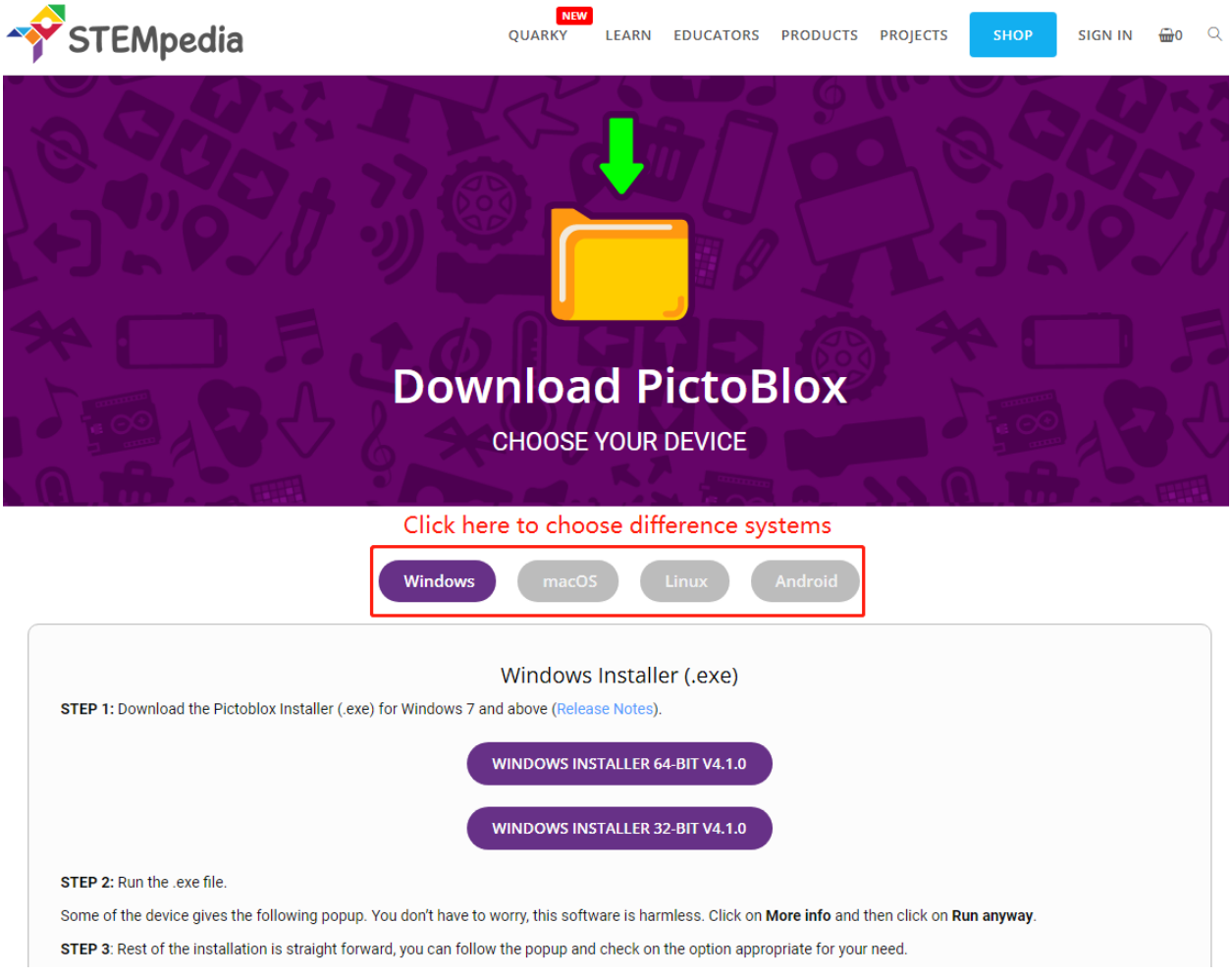
¡Simplemente arrastra y suelta los bloques de código de Scratch y crea juegos geniales, animaciones, proyectos interactivos e incluso controla robots como desees!

¡Ahora vamos a comenzar el viaje de descubrimiento!

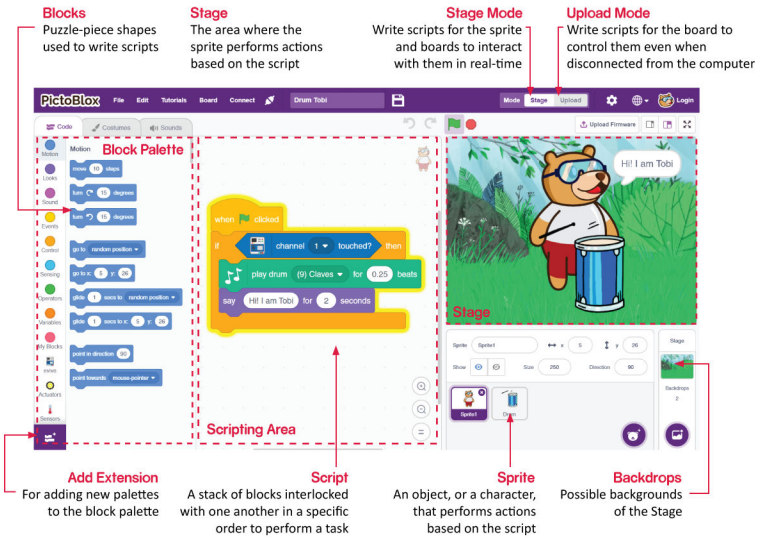
1. Empezar

5.1 1.1 Instalar PictoBlox

Haz clic en este enlace: https://thestempedia.com/product/pictoblox/download-pictoblox/, elige el Sistema Operativo adecuado (Windows, macOS, Linux) y sigue los pasos para instalar.



5.2 1.2 Introducción a la Interfaz



Sprites

Un sprite es un objeto o personaje que realiza diferentes acciones en un proyecto. Entiende y obedece los comandos que se le dan. Cada sprite tiene disfraces y sonidos específicos que también puedes personalizar.

Escenario

El escenario es el área donde el sprite realiza acciones en fondos de acuerdo con tu programa.

Fondos

Los fondos se utilizan para decorar el escenario. Puedes elegir un fondo de PictoBlox, dibujar uno tú mismo o subir una imagen desde tu computadora.

Área de Scripts

Un script es un programa o un código en el lenguaje de PictoBlox/Scratch. Es un conjunto de «bloques» organizados en un orden específico para realizar una tarea o una serie de tareas. Puedes escribir múltiples scripts, todos los cuales pueden ejecutarse simultáneamente. Solo puedes escribir scripts en el área de scripts en el centro de la pantalla.

Bloques

Los bloques son como piezas de un rompecabezas que se utilizan para escribir programas simplemente apilándolos juntos en el área de scripts. Usar bloques para escribir código puede hacer que la programación sea más fácil y reducir la probabilidad de errores.

Paleta de Bloques

Las paletas de bloques están ubicadas en el área izquierda y se nombran por sus funciones, como movimiento, sonido y control. Cada paleta tiene diferentes bloques, por ejemplo, los bloques en la paleta de Movimiento controlarán el movimiento de los sprites, y los bloques en la paleta de Control gestionarán el trabajo del script basado en condiciones específicas.

Hay otros tipos de paletas de bloques que se pueden cargar desde el botón **Añadir Extensión** ubicado en la parte inferior izquierda.

Modos

A diferencia de Scratch, PictoBlox tiene dos modos:

- *Modo Escenario*: En este modo, puedes escribir scripts para el sprite y las placas para interactuar con los sprites en tiempo real. Si desconectas la placa de Pictoblox, ya no puedes interactuar.
- *Modo de Subida*: Este modo te permite escribir scripts y cargarlos en la placa para que puedas usar incluso cuando no está conectada a tu computadora, por ejemplo, necesitas cargar un script para hacer robots móviles.

Para más información, por favor consulta: https://thestempedia.com/tutorials/getting-started-pictoblox

5.3 1.3 Guía Rápida sobre PictoBlox

5.3.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

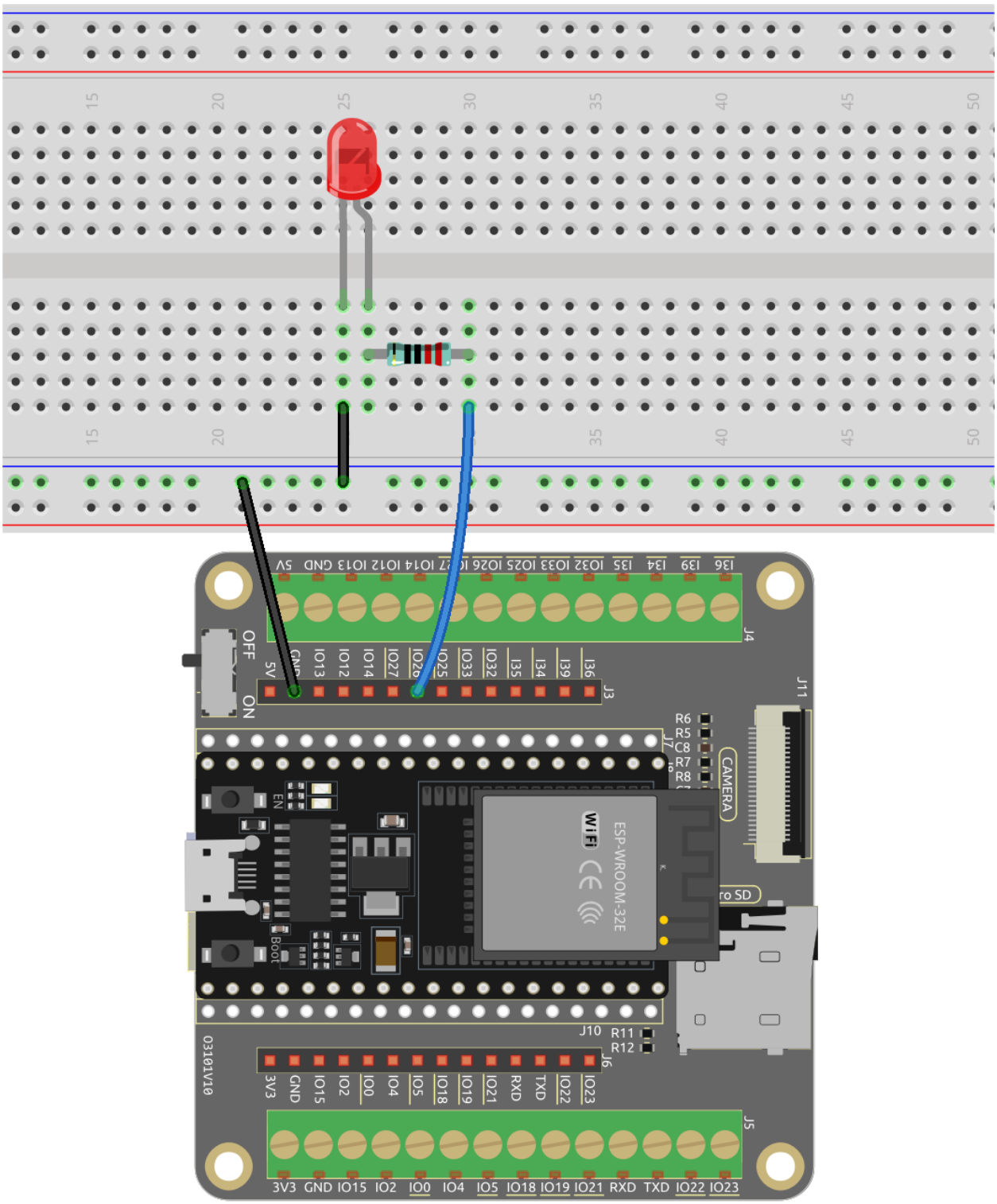
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

Ahora aprendamos a usar PictoBlox en dos modos.

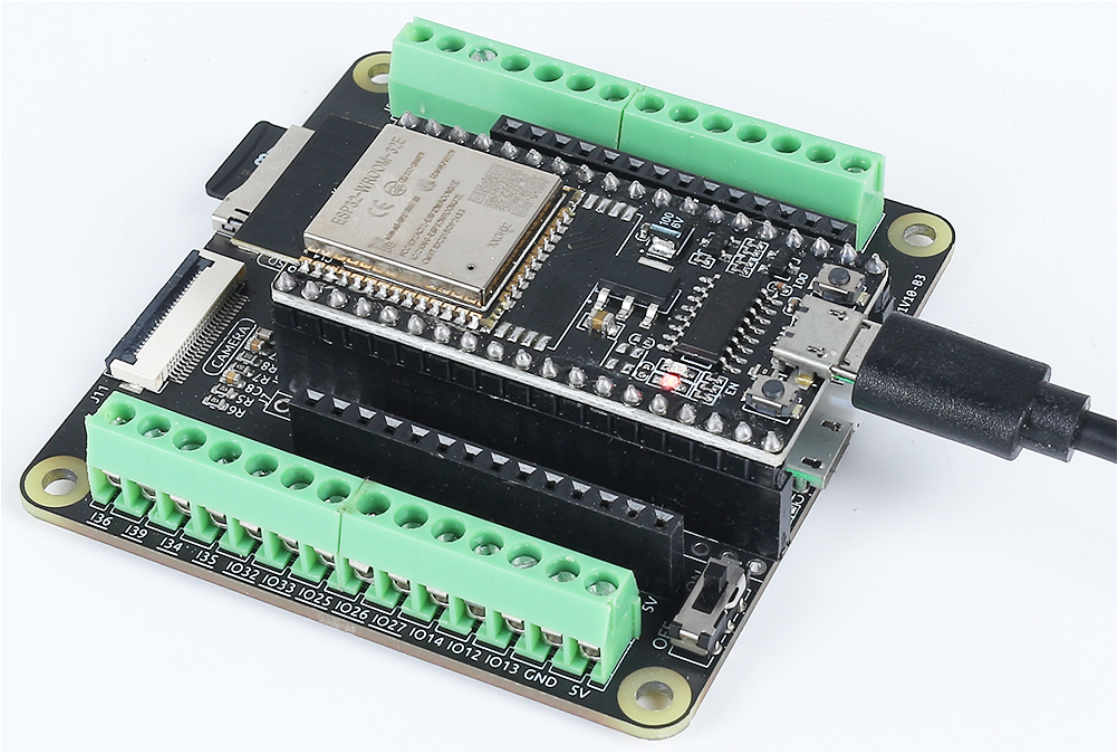
También construiremos un circuito simple para hacer que este LED parpadee en 2 modos diferentes.


5V GND IO13 IO12 IO14 IO27 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J4
OFF
ON
5V GND IO13 IO12 IO14 IO27 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J3
J11
R6
R5
C8
R7
R8
J7
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
J10
R11
R12
J6
03101V10
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23
J5

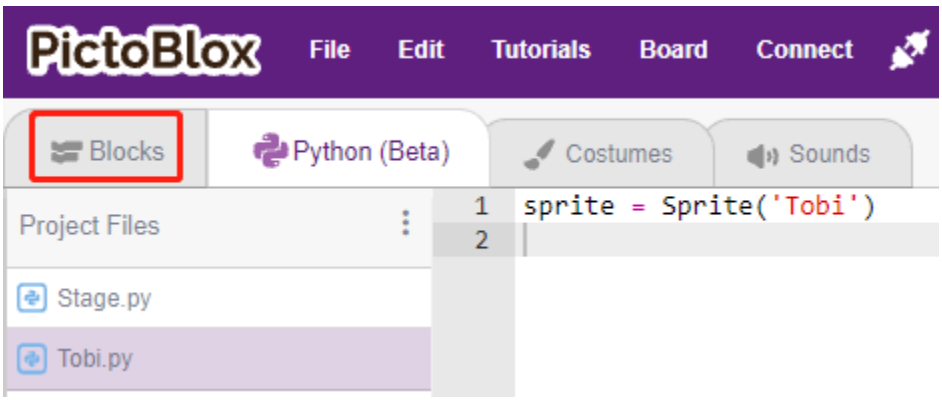
5.3.2 Modo Escenario

1. Conectar con la Placa ESP32

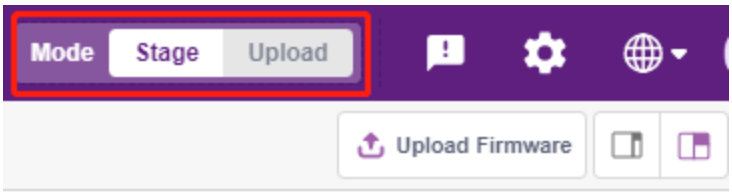
Conecta tu placa ESP32 al ordenador con un cable USB, normalmente el ordenador reconocerá automáticamente tu placa y finalmente asignará un puerto COM.



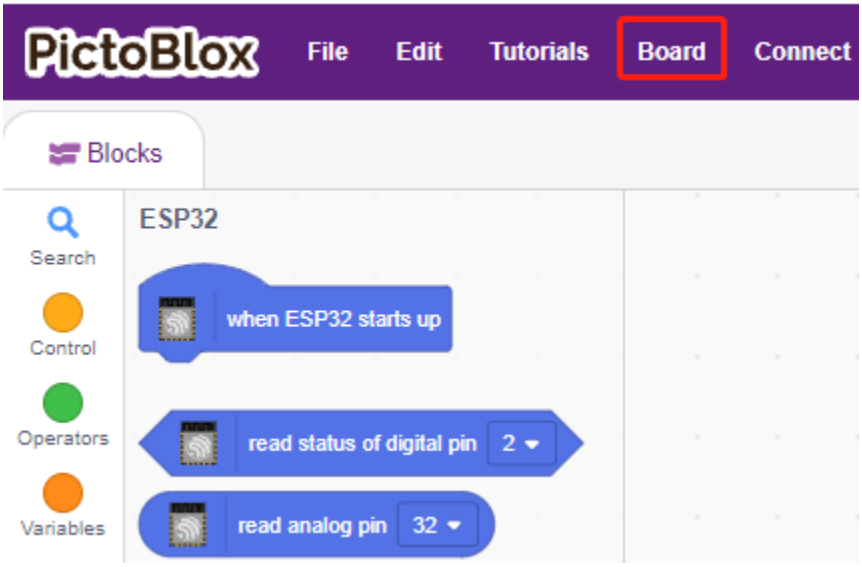
Abre PictoBlox, la interfaz de programación Python se abrirá por defecto. Y necesitamos cambiar a la interfaz de Bloques.



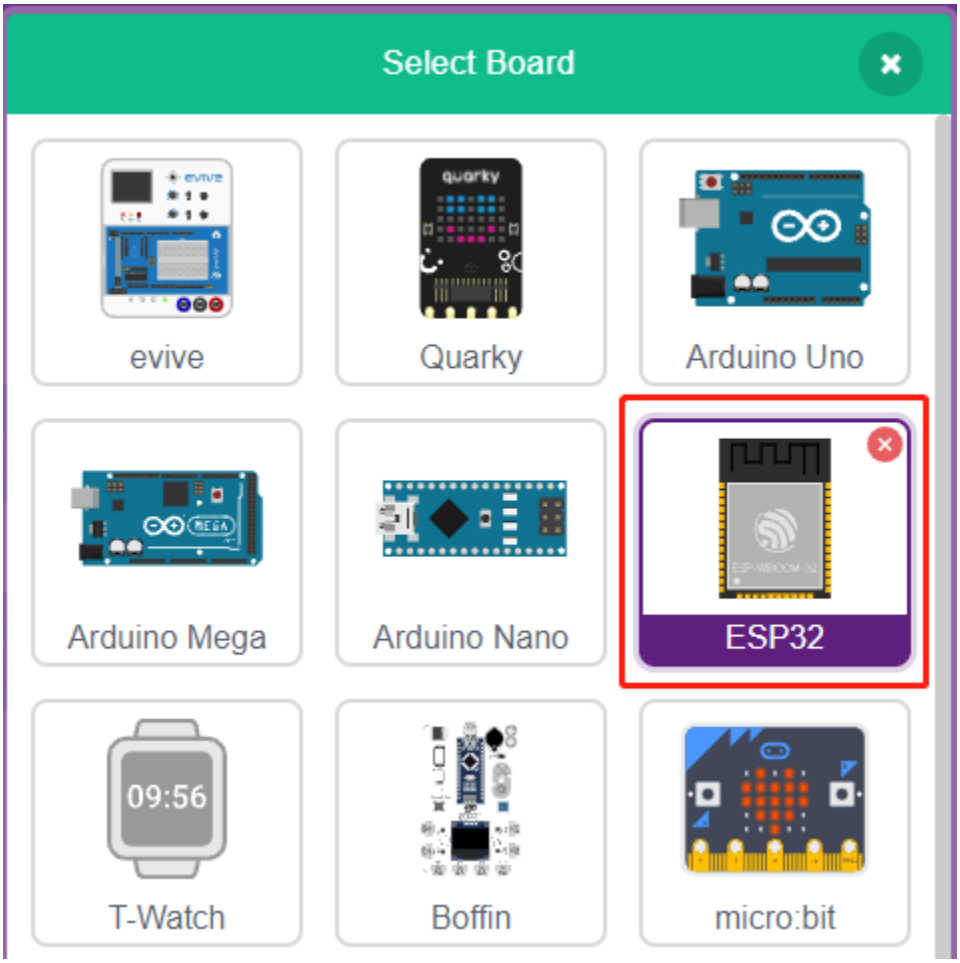
Entonces verás la esquina superior derecha para el cambio de modo. El predeterminado es el modo Escenario, donde Tobi está parado en el escenario.



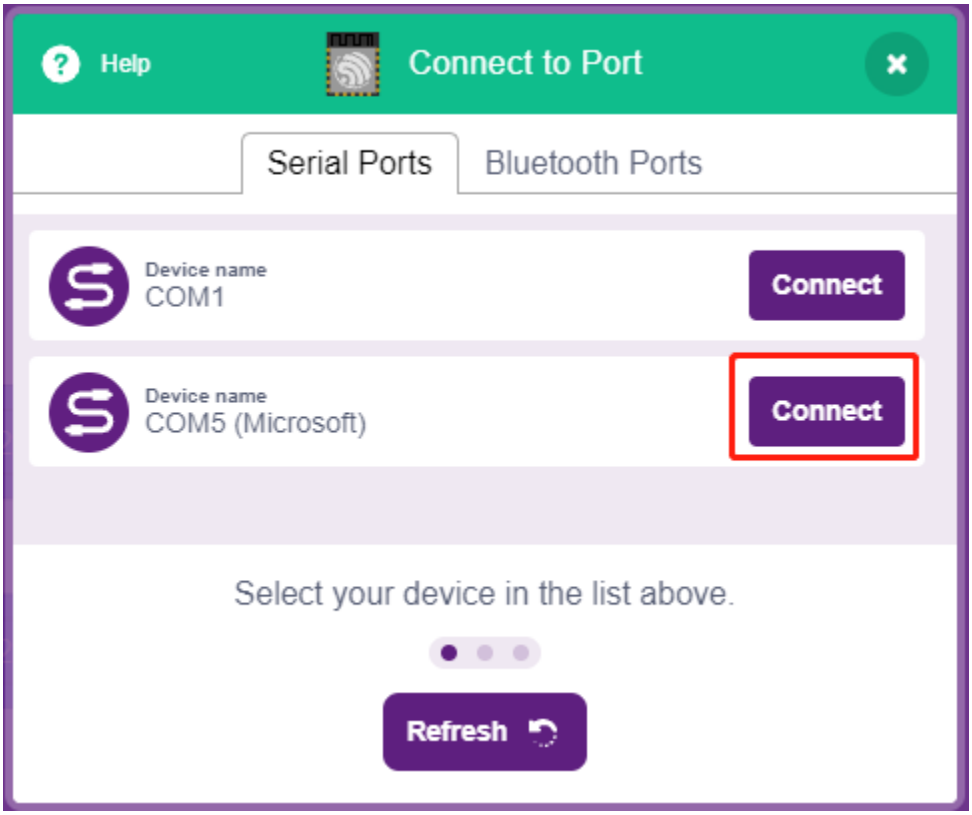
Haz clic en **Placa** en la barra de navegación superior derecha para seleccionar la placa.



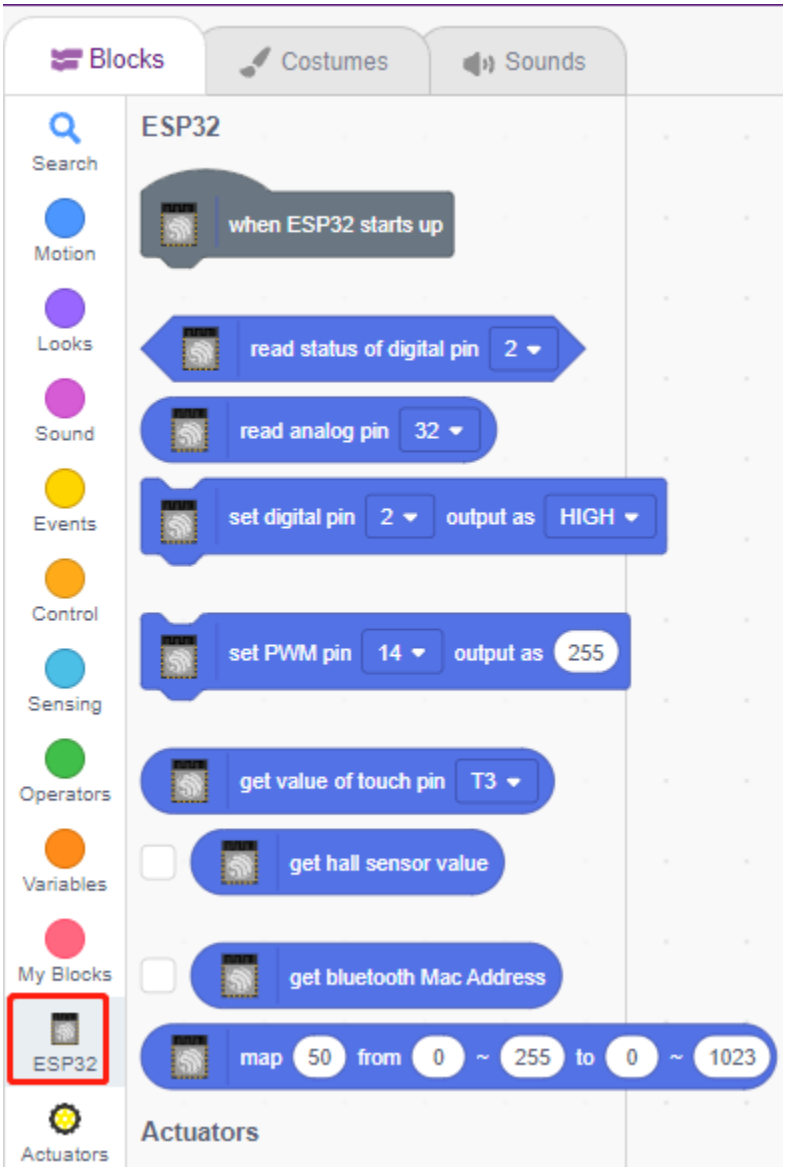
Por ejemplo, elige **ESP32**.



Entonces aparecerá una ventana de conexión para que selecciones el puerto a conectar, y regresarás a la página principal cuando la conexión esté completa. Si rompes la conexión durante el uso, también puedes hacer clic en **Conectar** para reconectar.



Al mismo tiempo, aparecerán en la **Paleta de Bloques** paletas relacionadas con ESP32, como ESP32, Actuadores, etc.

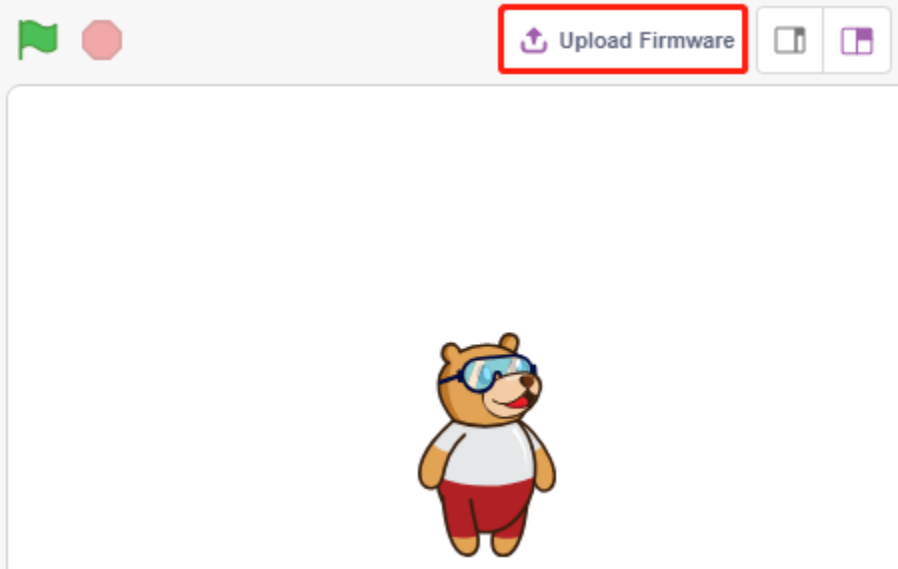


2. Subir Firmware

Dado que vamos a trabajar en el modo Escenario, debemos subir el firmware a la placa. Esto asegurará la comunicación en tiempo real entre la placa y el ordenador. Subir el firmware es un proceso único. Para hacerlo, haz clic en el botón Subir Firmware.

Después de esperar un rato, aparecerá el mensaje de éxito de la subida.

Nota: Si estás usando esta placa en PictoBlox por primera vez, o si esta placa fue previamente subida con el IDE de Arduino. Entonces necesitas tocar **Subir Firmware** antes de que puedas usarla.

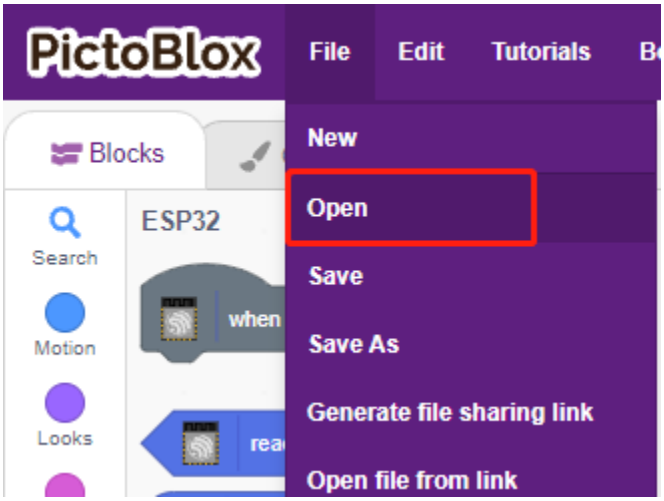


3. Programación

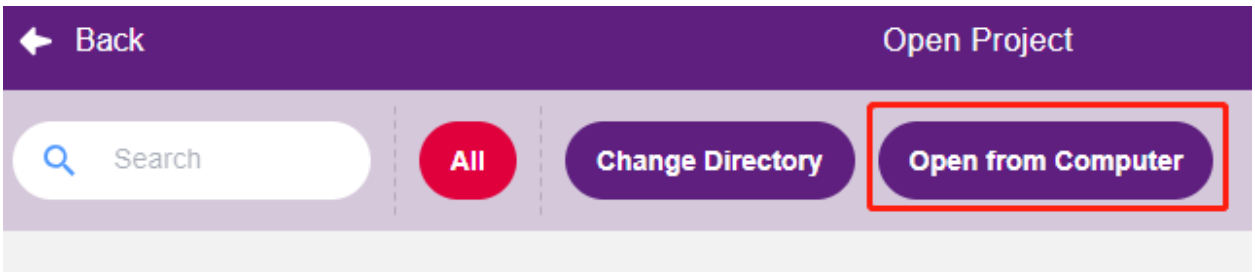
- Abrir y ejecutar el script directamente

Por supuesto, puedes abrir los scripts directamente para ejecutarlos, pero primero descárgalos de github.

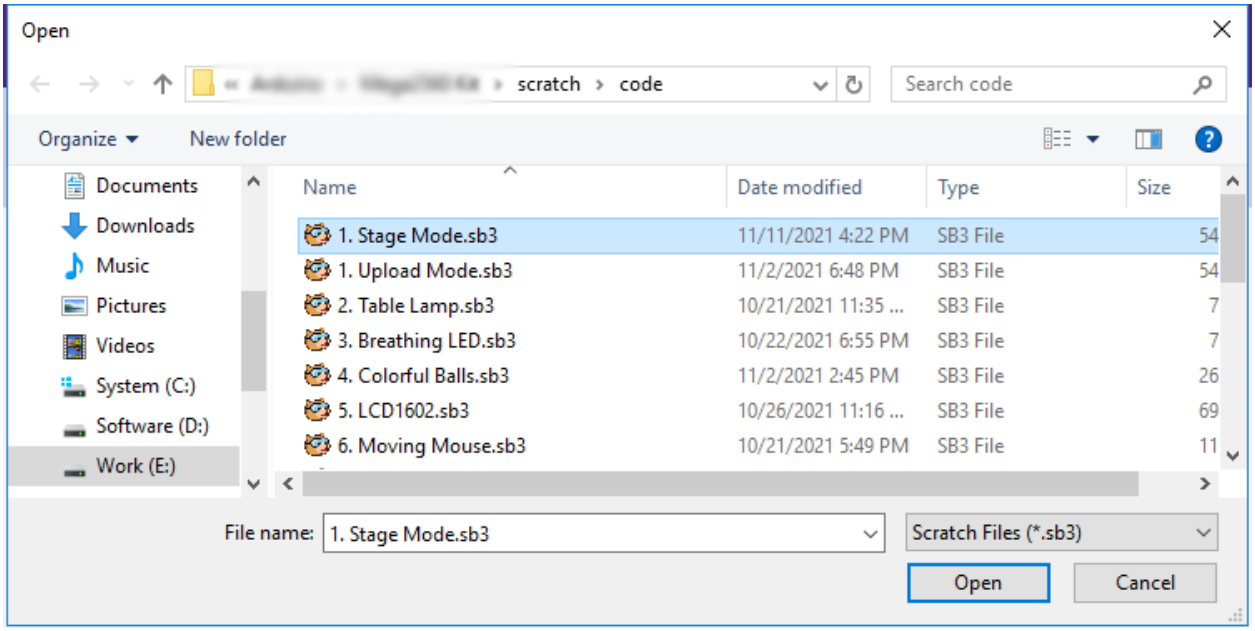
Puedes hacer clic en **Archivo** en la esquina superior derecha y luego elegir **Abrir**.



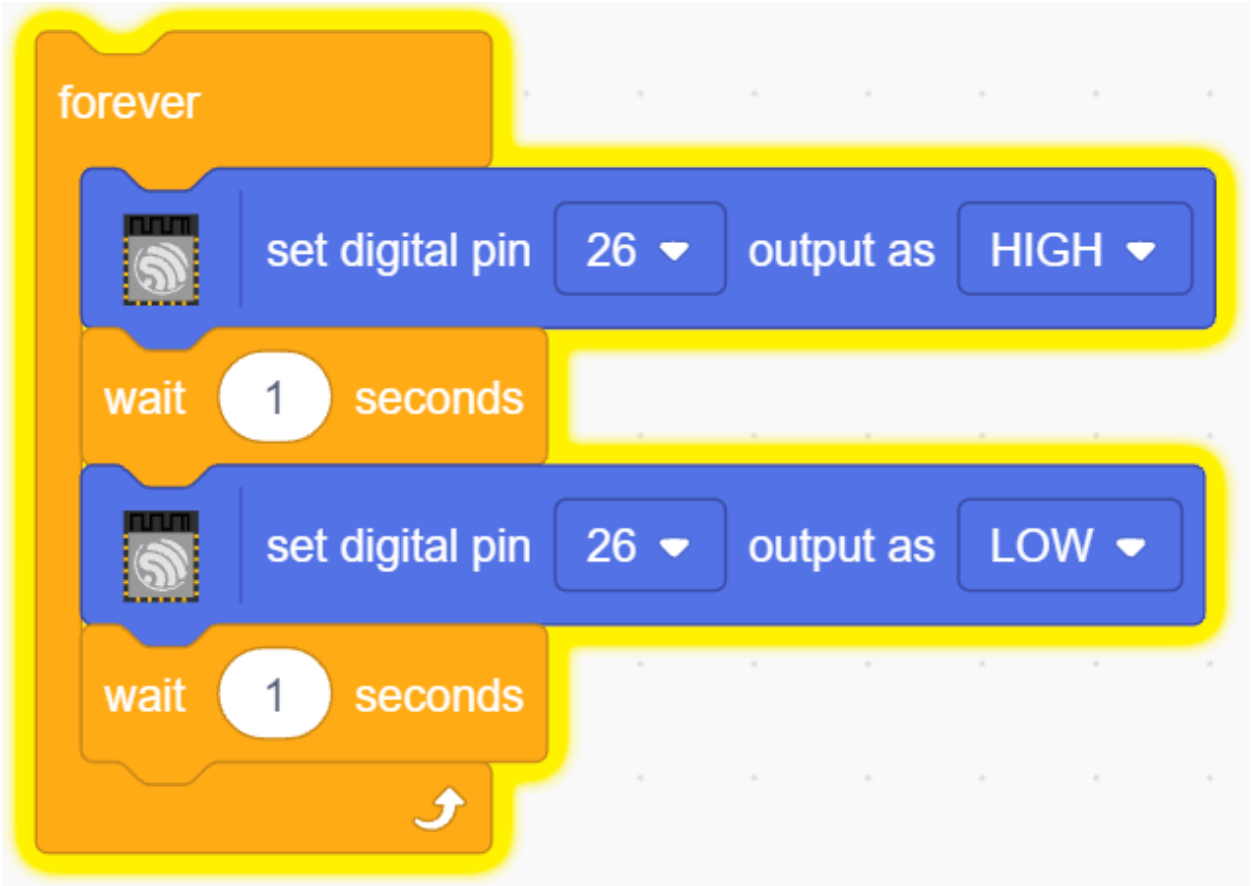
Elige **Abrir desde el Ordenador**.



Luego ve a la ruta de `esp32-starter-kit-main\scratch`, y abre **1. Modo Escenario.sb3**. Asegúrate de haber descargado el código requerido de github.



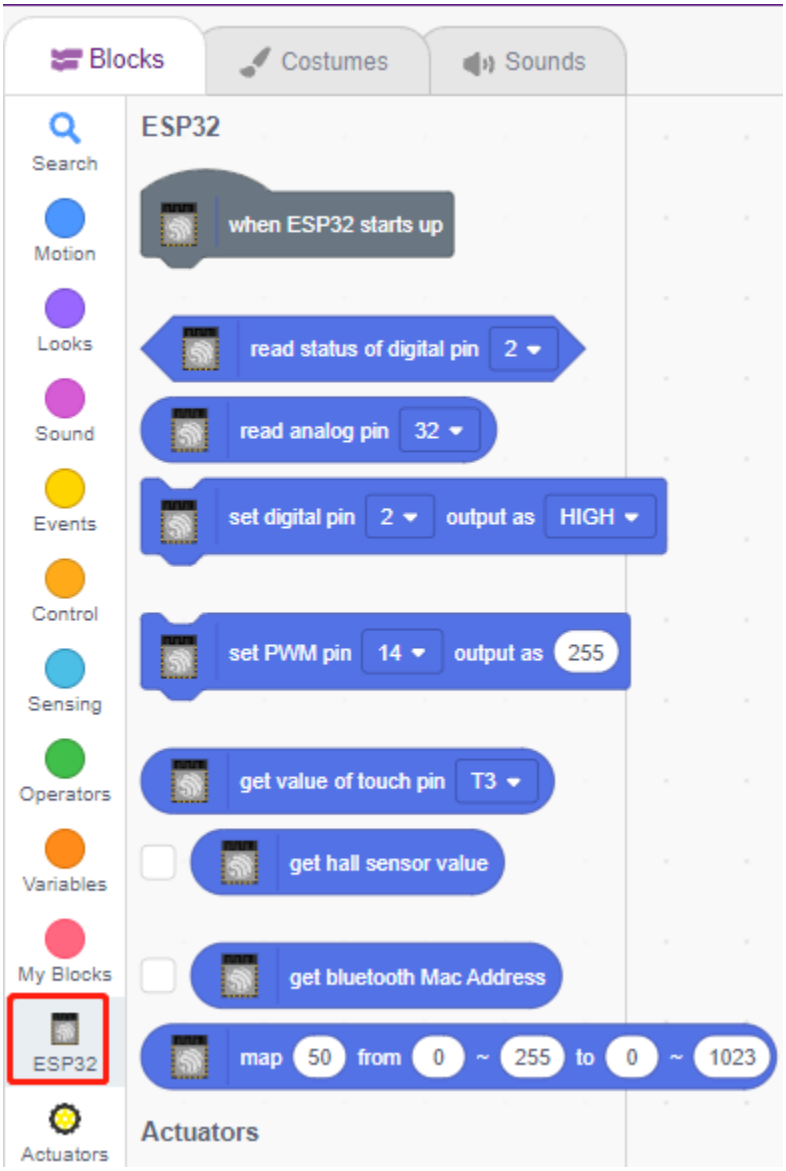
Haz clic directamente en el script para ejecutarlo, algunos proyectos son hacer clic en la bandera verde o hacer clic en el sprite.



- Programar paso a paso

También puedes escribir el script paso a paso siguiendo estos pasos.

Haz clic en la paleta **ESP32**.



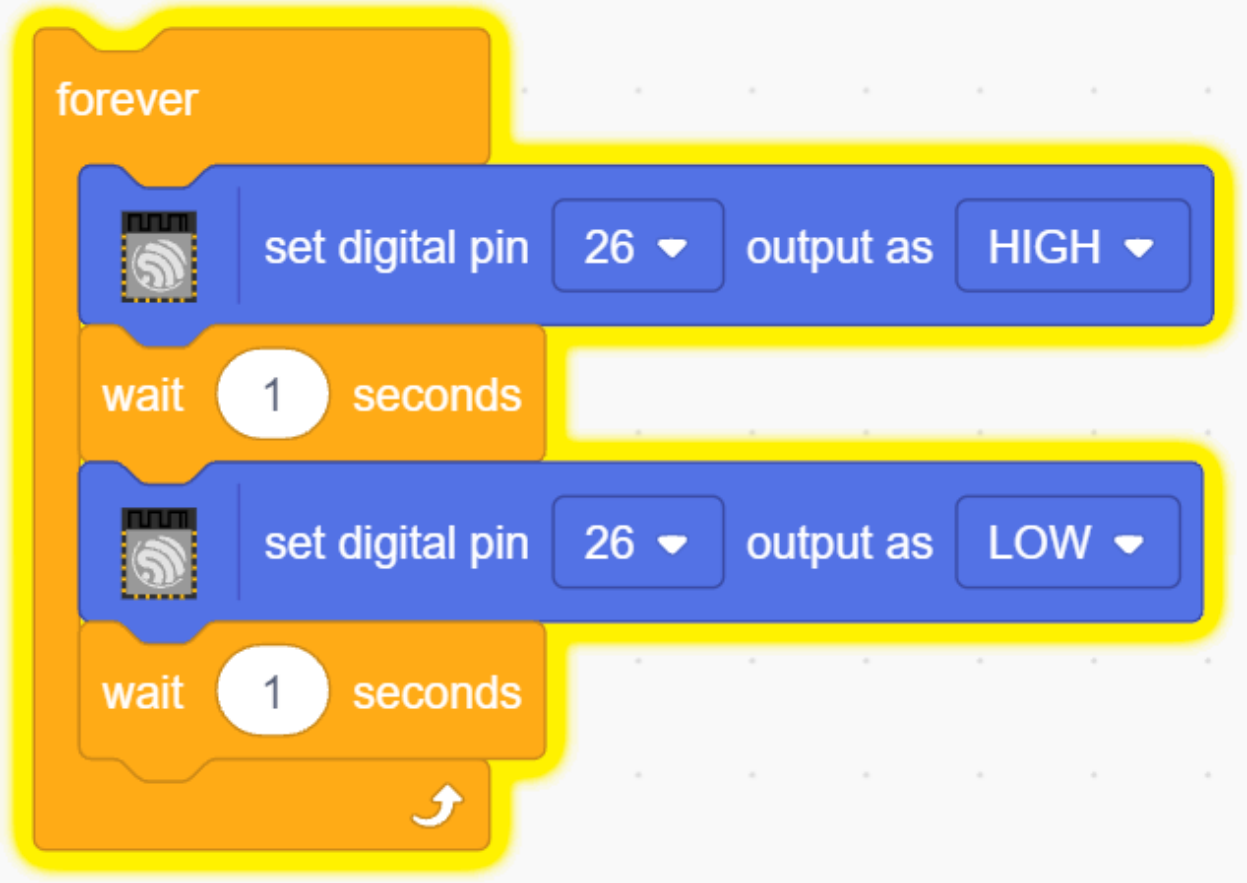
El LED está controlado por el pin digital 26 (solo 2 estados, ALTO o BAJO), así que arrastra el bloque [establecer el pin digital como] al área de script.

Dado que el estado predeterminado del LED es encendido, ahora establece el pin 23 en BAJO y haz clic en este bloque y verás que el LED se apaga.

- [establecer el pin digital como]: Establece el pin digital a nivel (ALTO/BAJO).

Para ver el efecto de un LED parpadeando continuamente, necesitas usar los bloques [Esperar 1 segundos] y [siempre] en la paleta **Control**. Haz clic en estos bloques después de escribir, un halo amarillo significa que está ejecutándose.

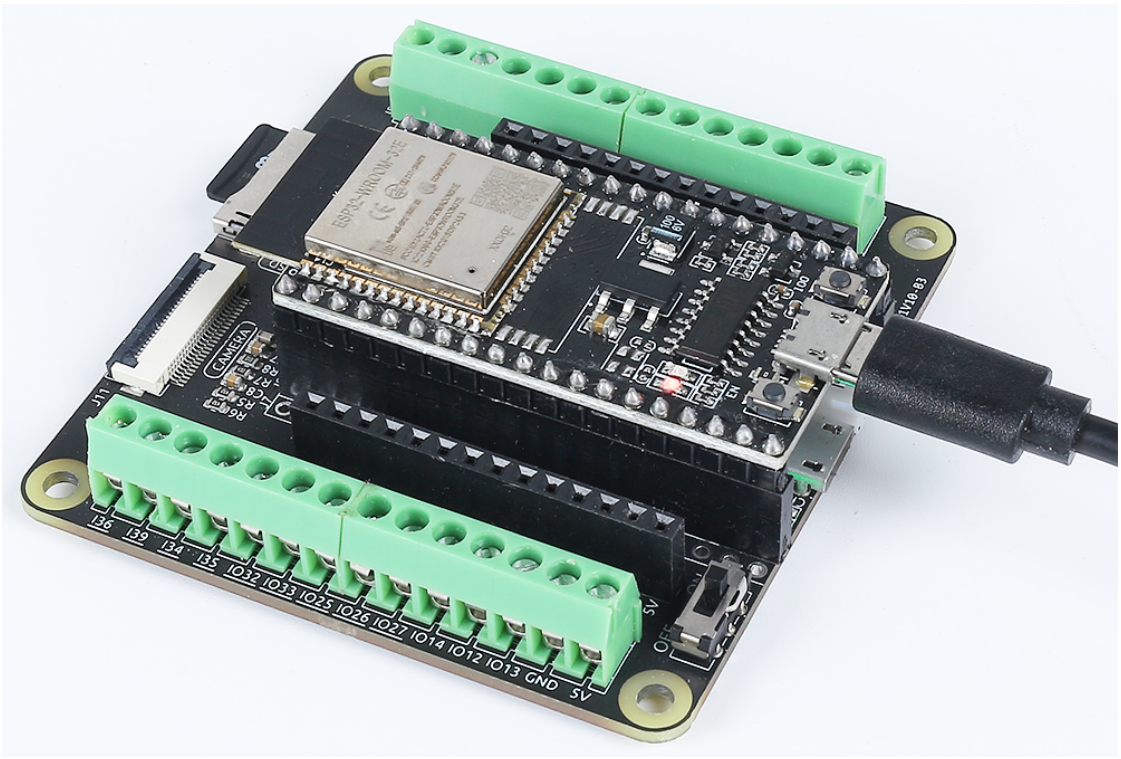
- [Esperar 1 segundos]: de la paleta **Control**, usado para establecer el intervalo de tiempo entre 2 bloques.
- [siempre]: de la paleta **Control**, permite que el script siga ejecutándose a menos que se pause manualmente.



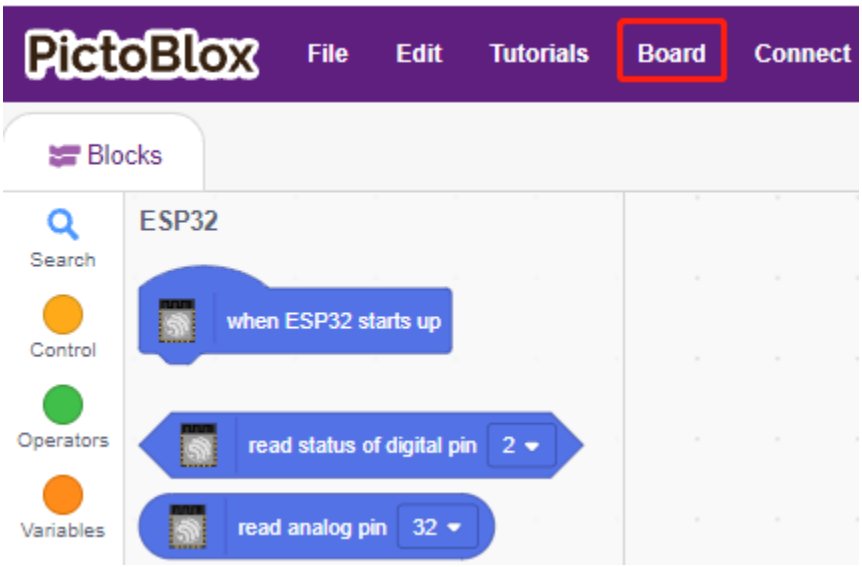
5.3.3 Modo de Subida

1. Conectar con la Placa ESP32

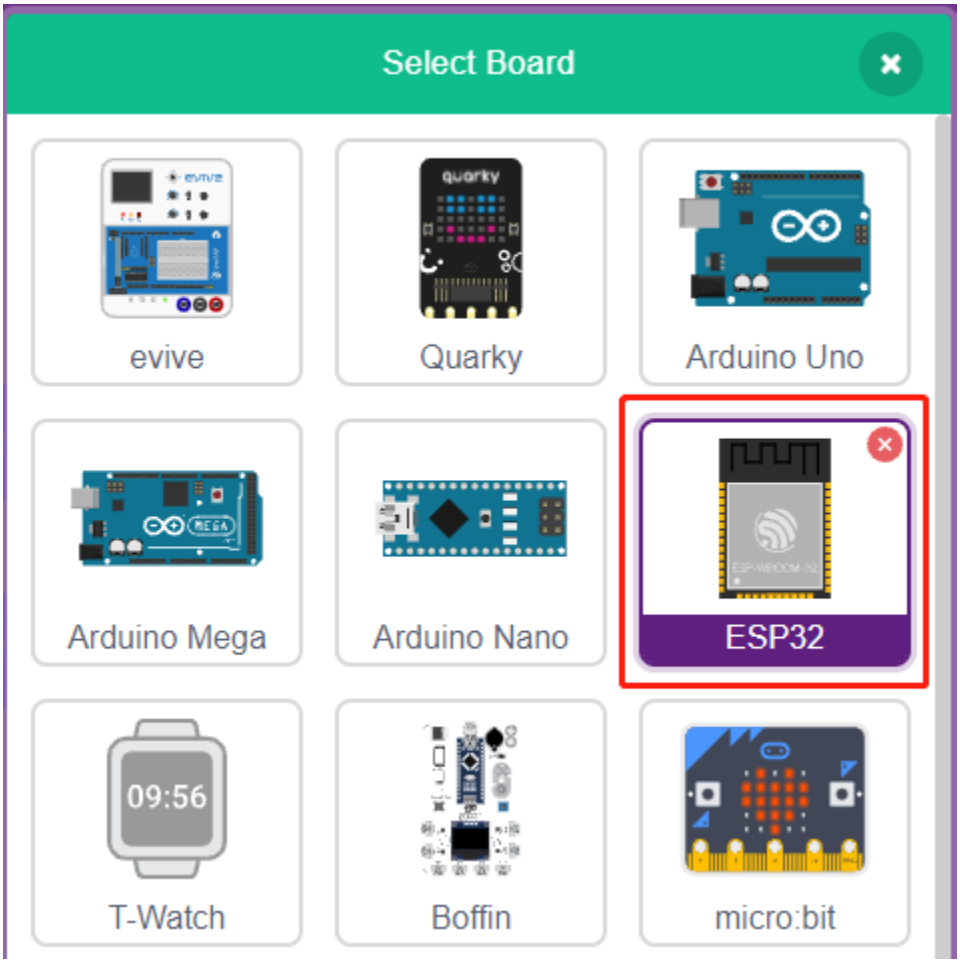
Conecta tu placa ESP32 al ordenador con un cable USB, normalmente el ordenador reconocerá automáticamente tu placa y finalmente asignará un puerto COM.



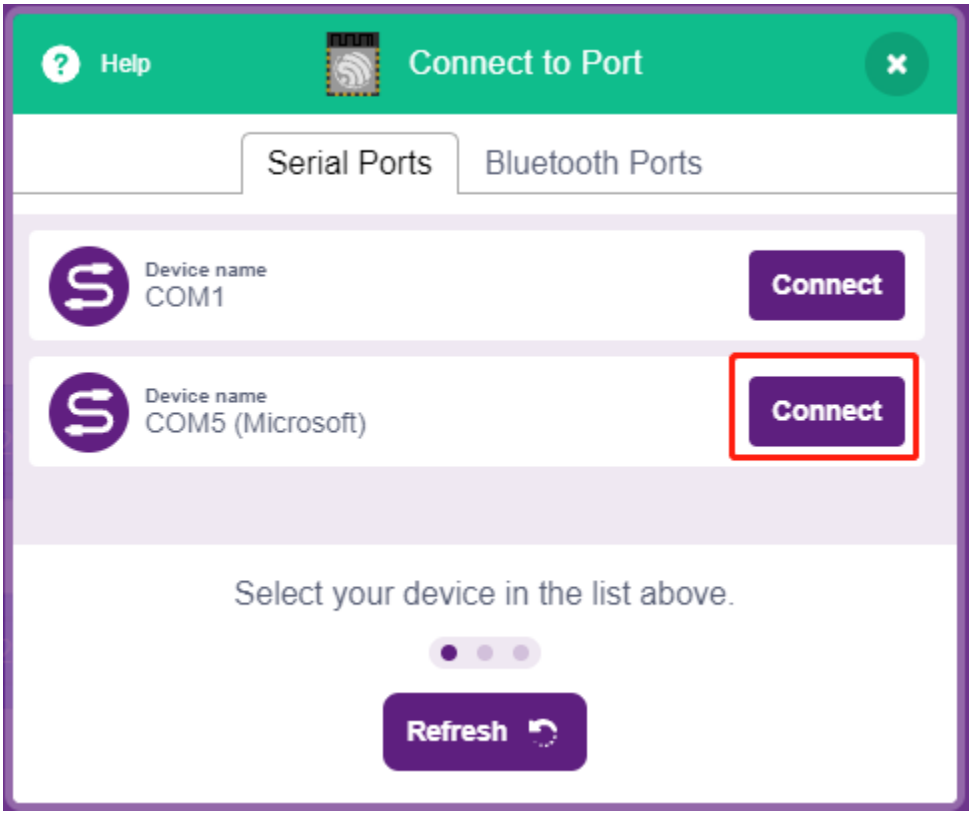
Abre PictoBlox y haz clic en **Placa** en la barra de navegación superior derecha para seleccionar la placa.



Por ejemplo, elige **ESP32**.

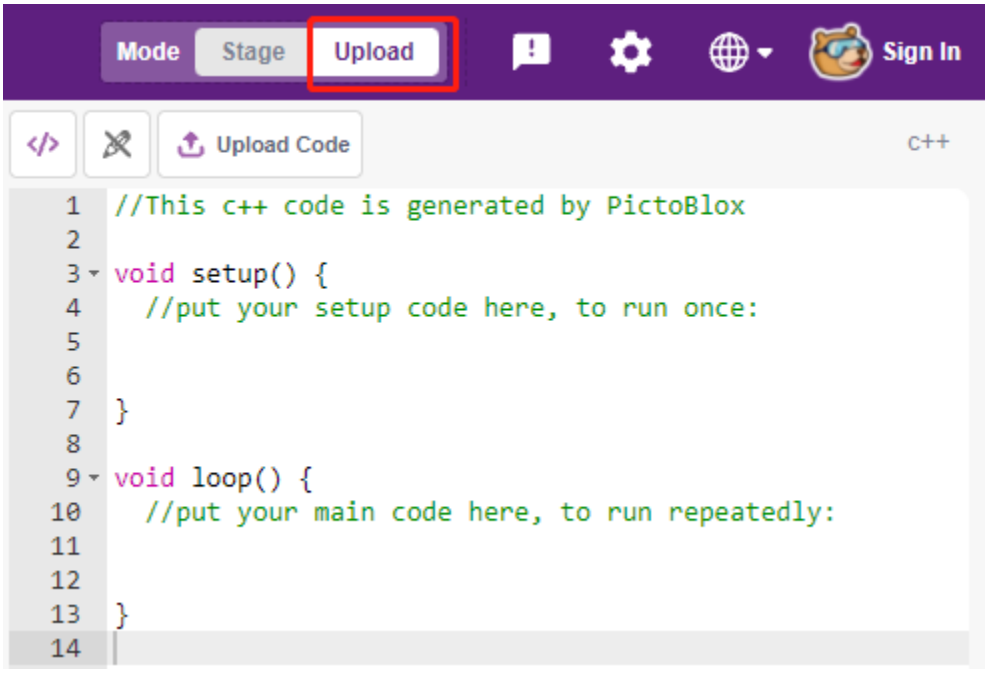


Entonces aparecerá una ventana de conexión para que selecciones el puerto a conectar, y regresarás a la página principal cuando la conexión esté completa. Si rompes la conexión durante el uso, también puedes hacer clic en **Conectar** para reconectar.



Al mismo tiempo, aparecerán en la **Paleta de Bloques** paletas relacionadas con ESP32, como ESP32, Actuadores, etc.

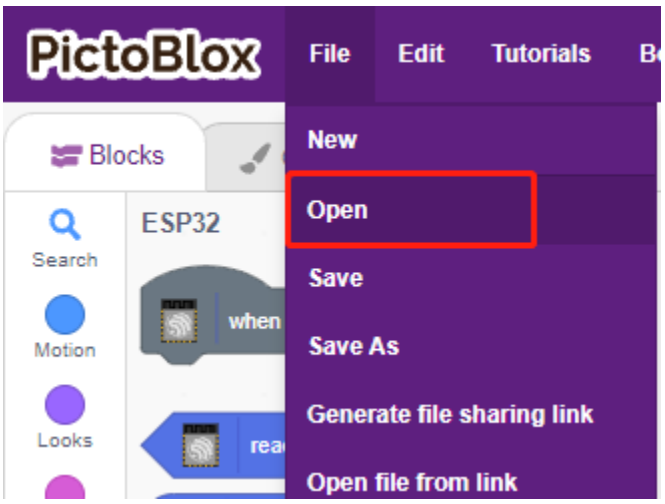
Después de seleccionar el modo de Subida, el escenario cambiará al área de código original.



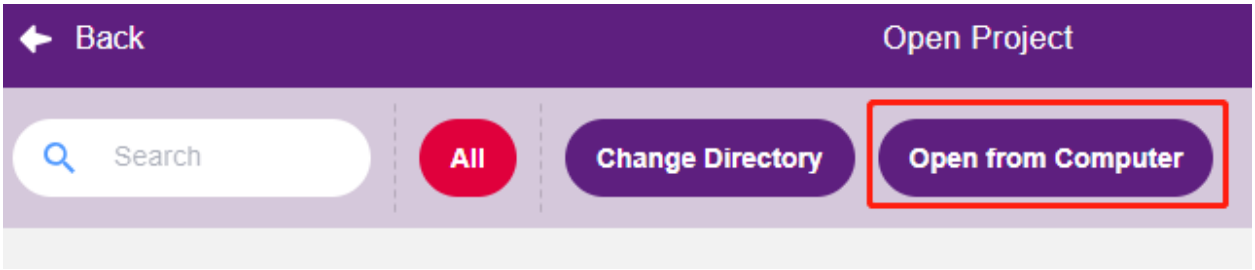
2. Programación

- Abrir y ejecutar el script directamente

Puedes hacer clic en **Archivo** en la esquina superior derecha.

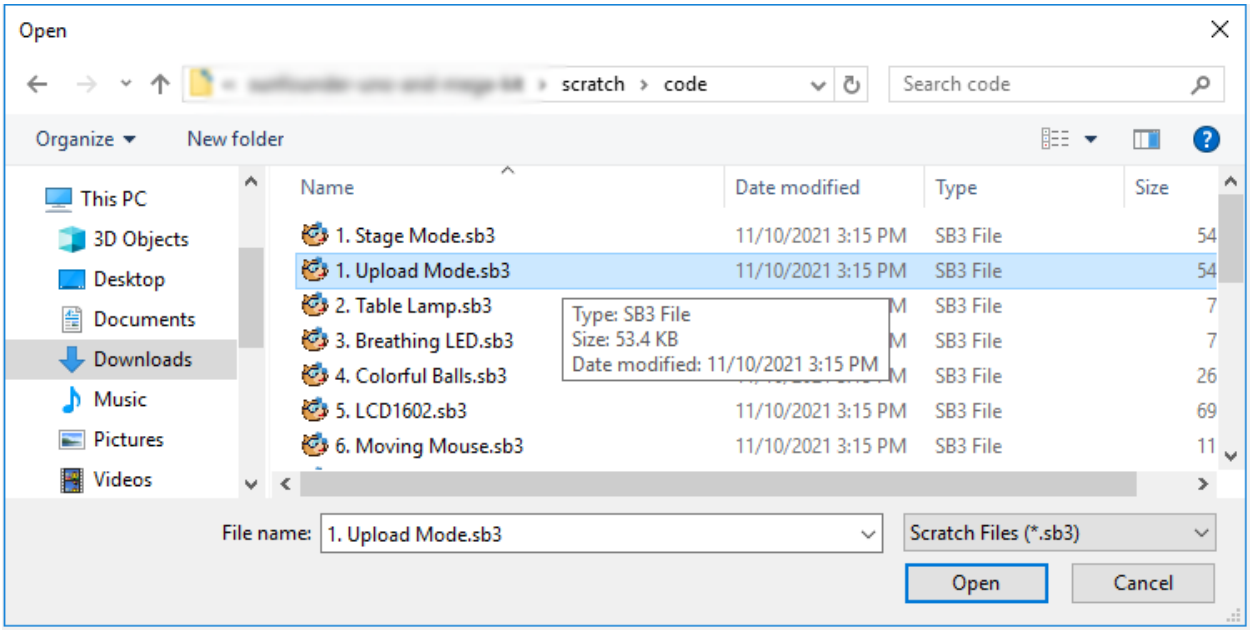


Elige **Abrir desde el Ordenador**.

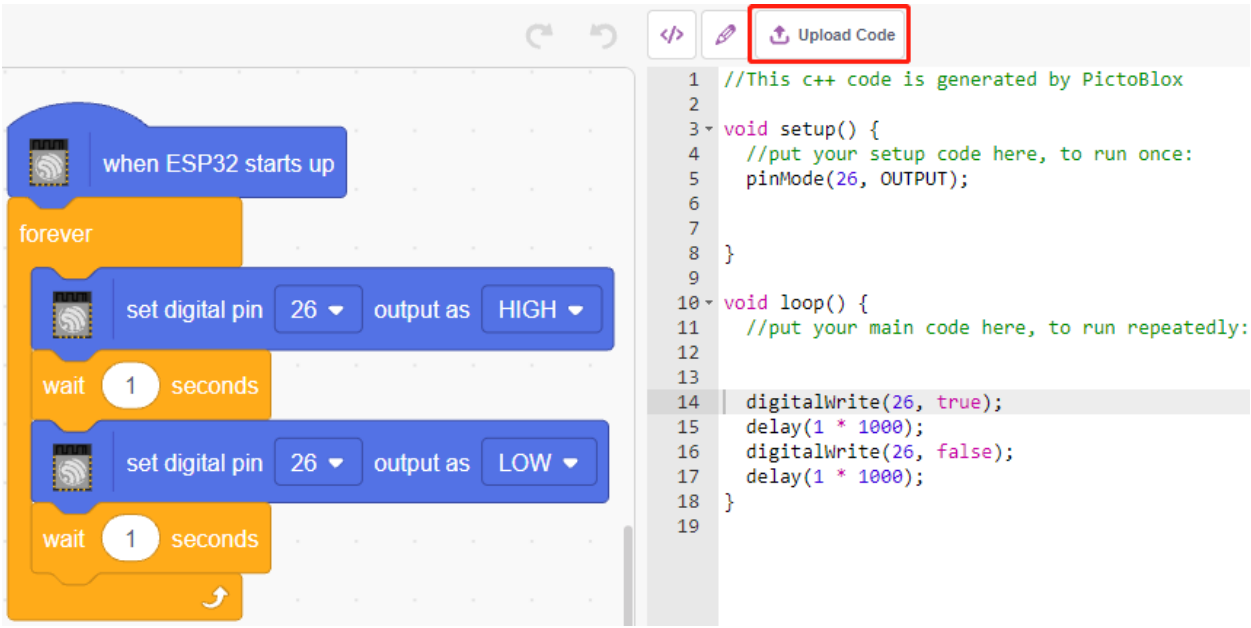


Luego ve a la ruta de `esp32-starter-kit-main\scratch`, y abre **1. Modo de Subida.sb3**. Asegúrate de haber

descargado el código requerido de github.



Finalmente, haz clic en el botón **Subir Código**.

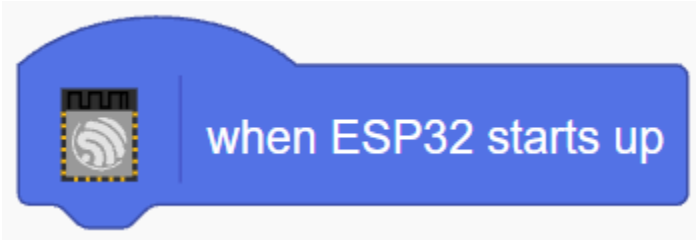


- Programar paso a paso

También puedes escribir el script paso a paso siguiendo estos pasos.

Haz clic en la paleta **ESP32**.

Arrastra [cuando ESP32 se inicia] al área de script, lo cual es necesario para cada script.

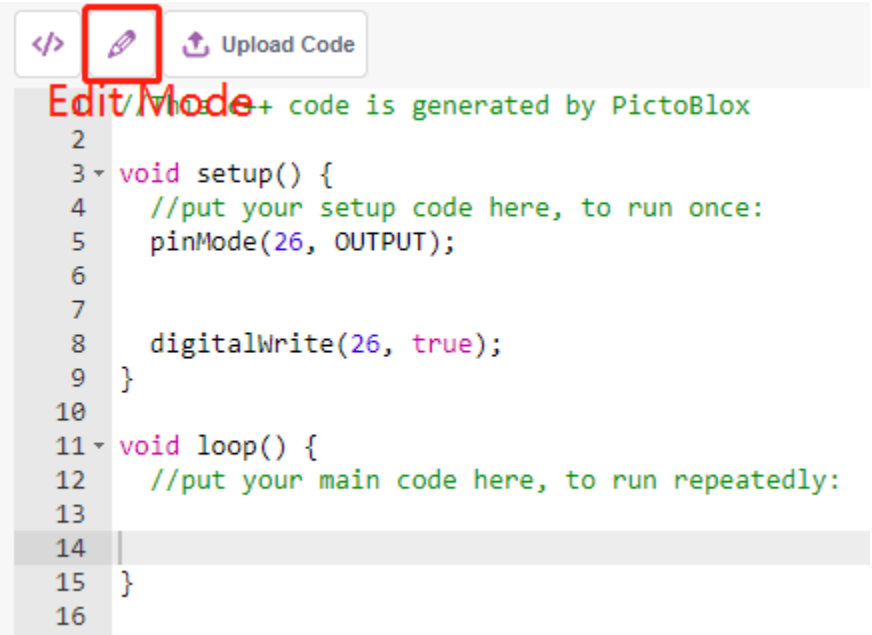


El LED es controlado por el pin digital 26 (solo 2 estados ALTO o BAJO), así que arrastra el bloque [establecer el pin digital como] al área de script.

Dado que el estado predeterminado del LED es encendido, ahora establece el pin 26 en BAJO y haz clic en este bloque y verás que el LED se apaga.

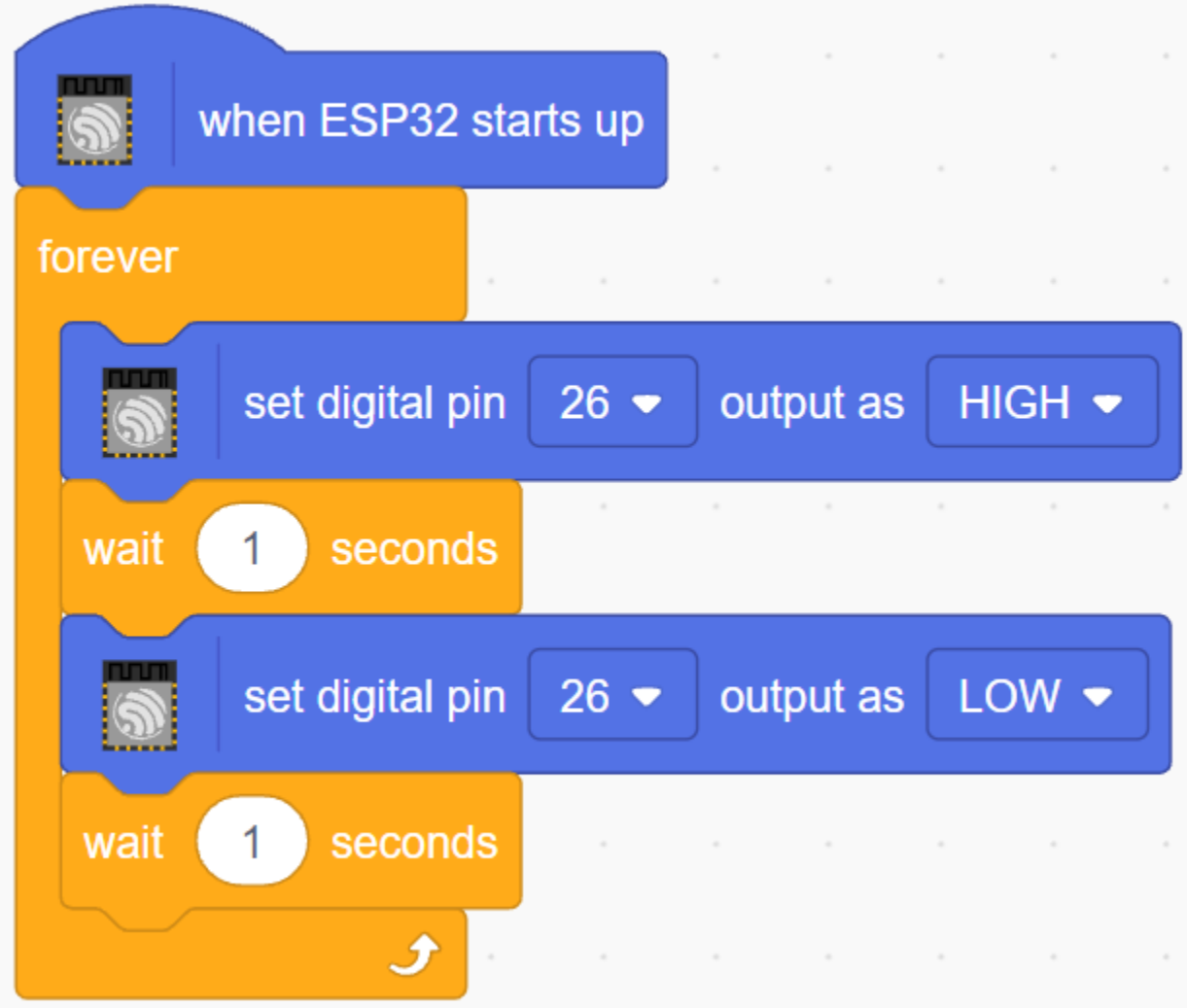
- [establecer el pin digital como]: Establece el pin digital a nivel (ALTO/BAJO).

En este punto verás que el código aparece en el lado derecho, si quieres editar este código, entonces puedes activar el modo Edición.

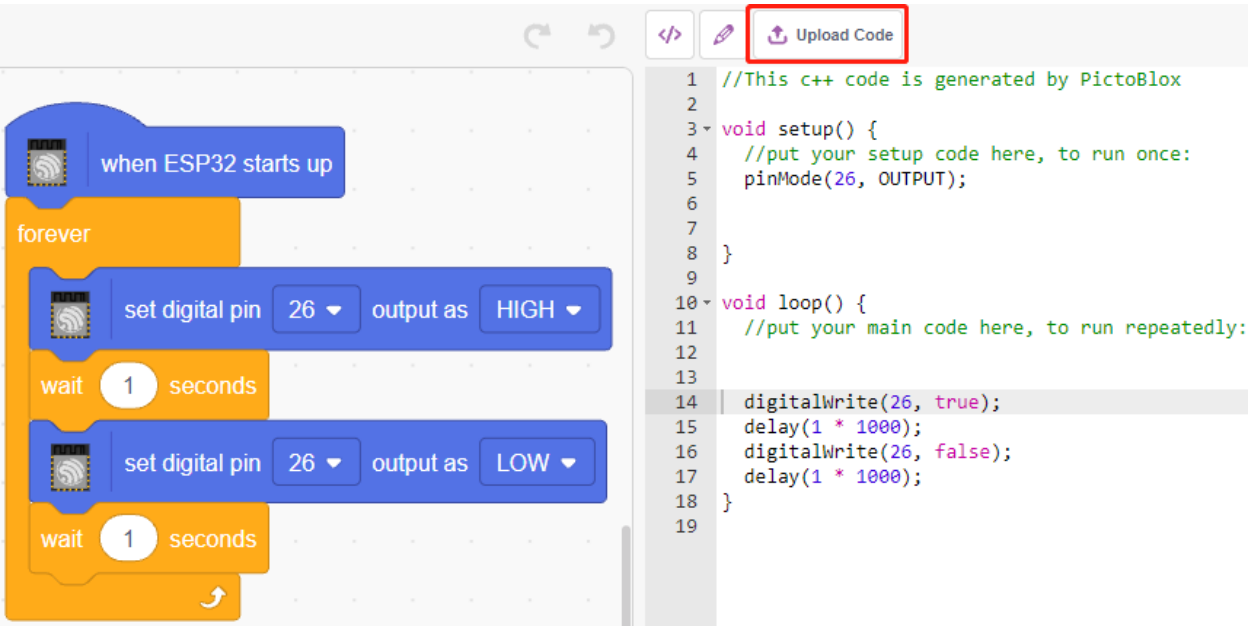


Para ver el efecto de un LED parpadeando continuamente, necesitas usar los bloques [Esperar 1 segundos] y [siempre] en la paleta **Control**. Haz clic en estos bloques después de escribir, un halo amarillo significa que está ejecutándose.

- [Esperar 1 segundos]: de la paleta **Control**, usado para establecer el intervalo de tiempo entre 2 bloques.
- [siempre]: de la paleta **Control**, permite que el script siga ejecutándose a menos que se apague la alimentación.



Finalmente, haz clic en el botón **Subir Código**.



2. Proyectos

Los siguientes proyectos están escritos en orden de dificultad de programación, se recomienda leer estos libros en orden.

En cada proyecto, hay pasos muy detallados para enseñarte cómo construir el circuito y cómo programarlo paso a paso para lograr el resultado final.

Por supuesto, también puedes abrir el script directamente para ejecutarlo, pero debes asegurarte de haber descargado el material relevante desde github.

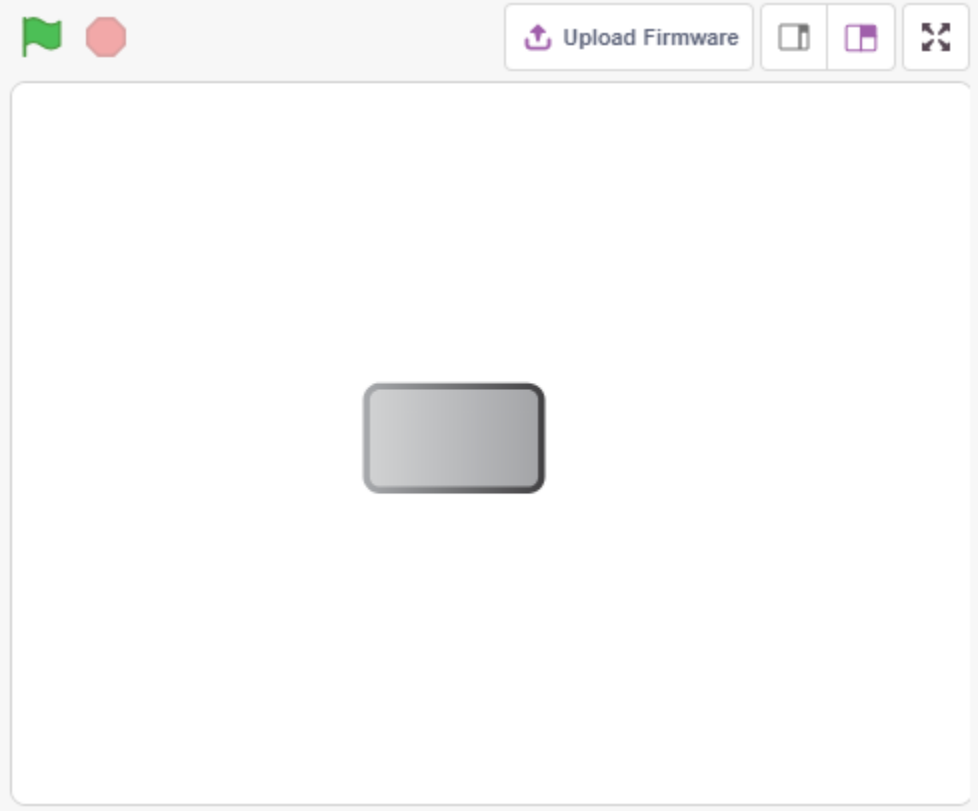
Una vez que la descarga esté completa, descomprímela. Consulta *Modo Escenario* para ejecutar scripts individuales directamente.

Pero el *2.8 Leer Temperatura y Humedad* se utiliza el *Modo de Subida*.

5.4 2.1 Lámpara de Mesa

Aquí, conectamos un LED en la protoboard y hacemos que el sprite controle el parpadeo de este LED.

Cuando se hace clic en el sprite del Botón en el escenario, el LED parpadeará 5 veces y luego se detendrá.



5.4.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

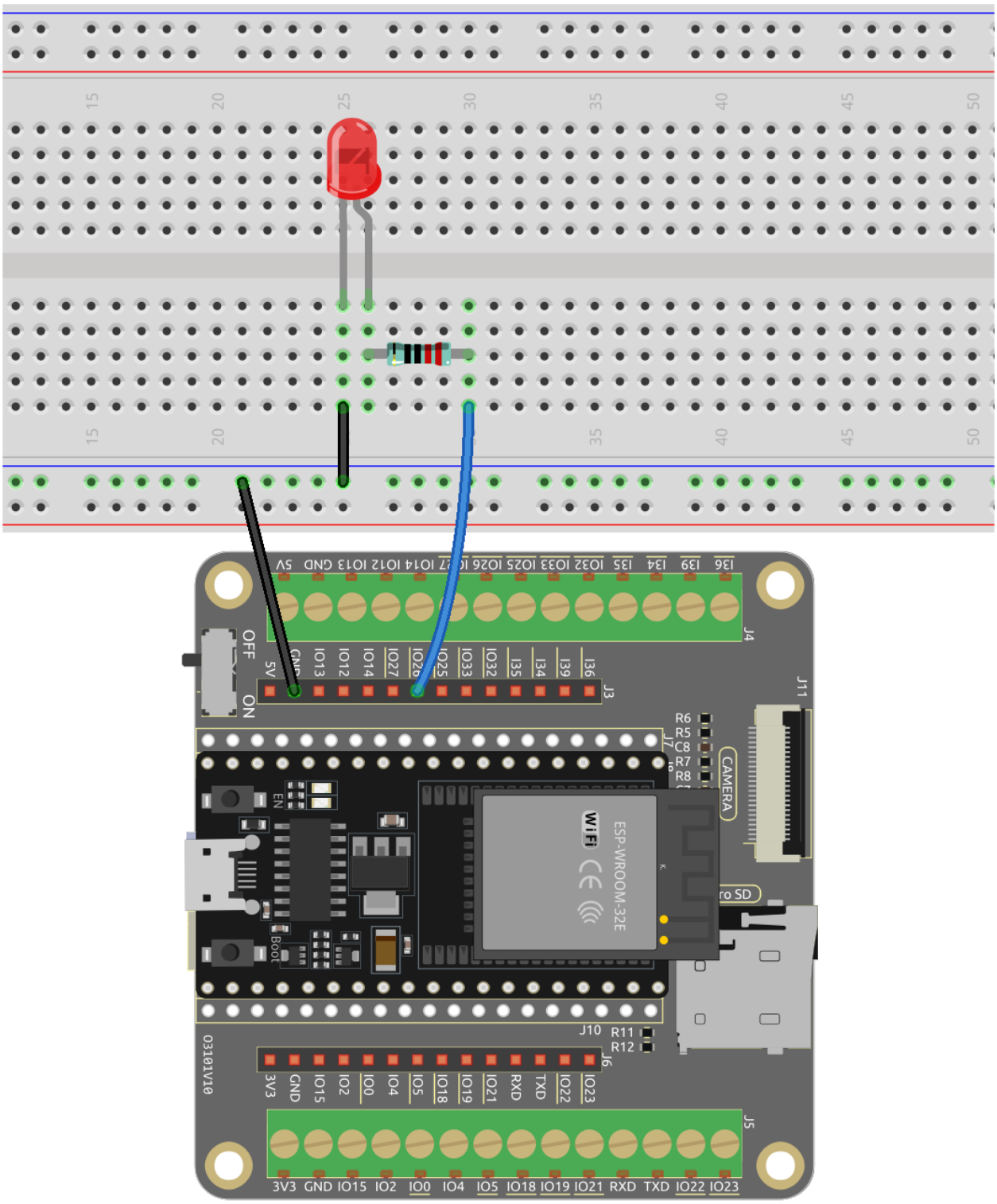
5.4.2 Lo Que Aprenderás

- Protoboard, LEDs y Resistencias
- Construir un circuito en una protoboard
- Eliminar y seleccionar sprites
- Cambiar disfraces
- Establecer un número limitado de bucles de repetición

5.4.3 Construir el Circuito

Sigue el diagrama a continuación para construir el circuito en la protoboard.

Dado que el ánodo del LED (el pin más largo) está conectado al pin 26 a través de una resistencia de 220, y el cátodo del LED está conectado a GND, puedes encender el LED dando un nivel alto al pin 9.

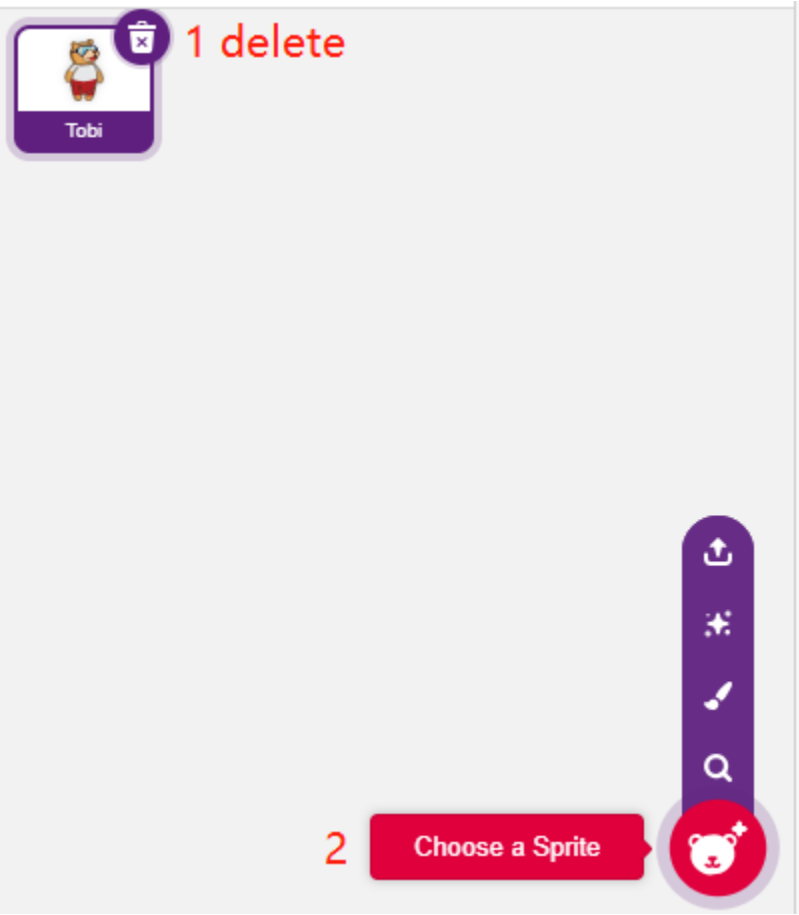

OFF
ON
ESP-WROOM-32E
CAMERA
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

5.4.4 Programación

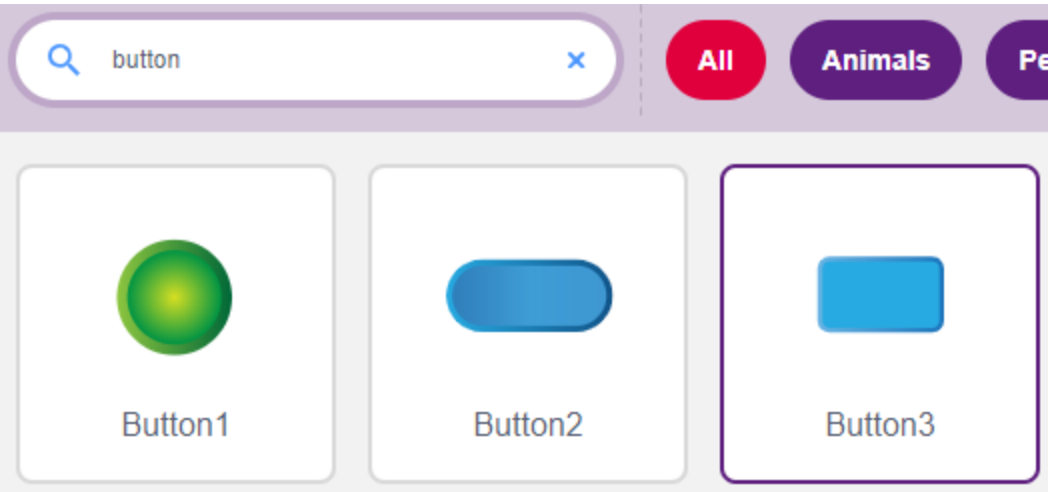
La programación completa se divide en 3 partes, la primera parte es seleccionar el sprite deseado, la segunda parte es cambiar el disfraz del sprite para que parezca clickeable, y la tercera parte es hacer que el LED parpadee.

1. Seleccionar el sprite Button3

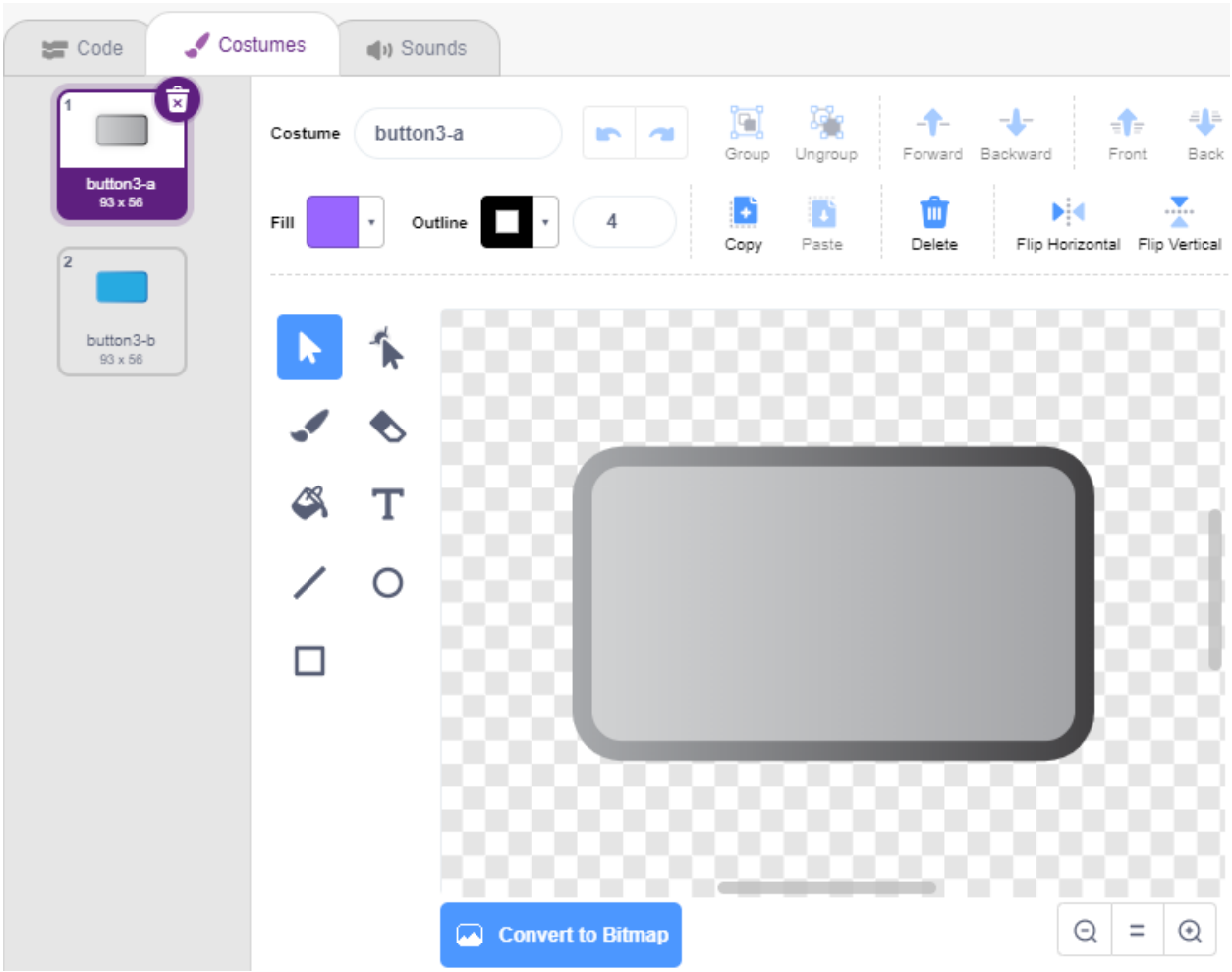
Elimina el sprite de Tobi existente utilizando el botón Eliminar en la esquina superior derecha, y selecciona un sprite de nuevo.



Aquí, seleccionamos el sprite **Button3**.

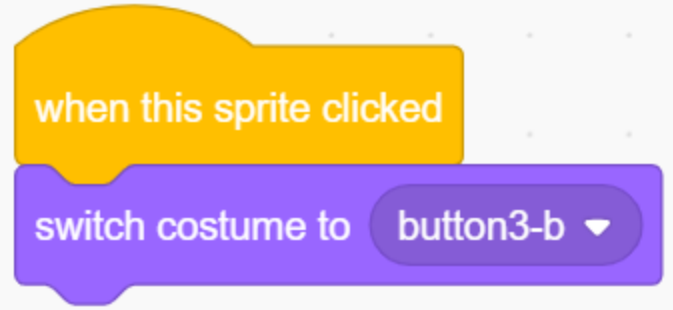


Haz clic en Disfraces en la esquina superior derecha y verás que el sprite Button3 tiene 2 disfraces, establecemos **button3-a** como liberado y **button3-b** como presionado.



2. Cambiando disfraces.

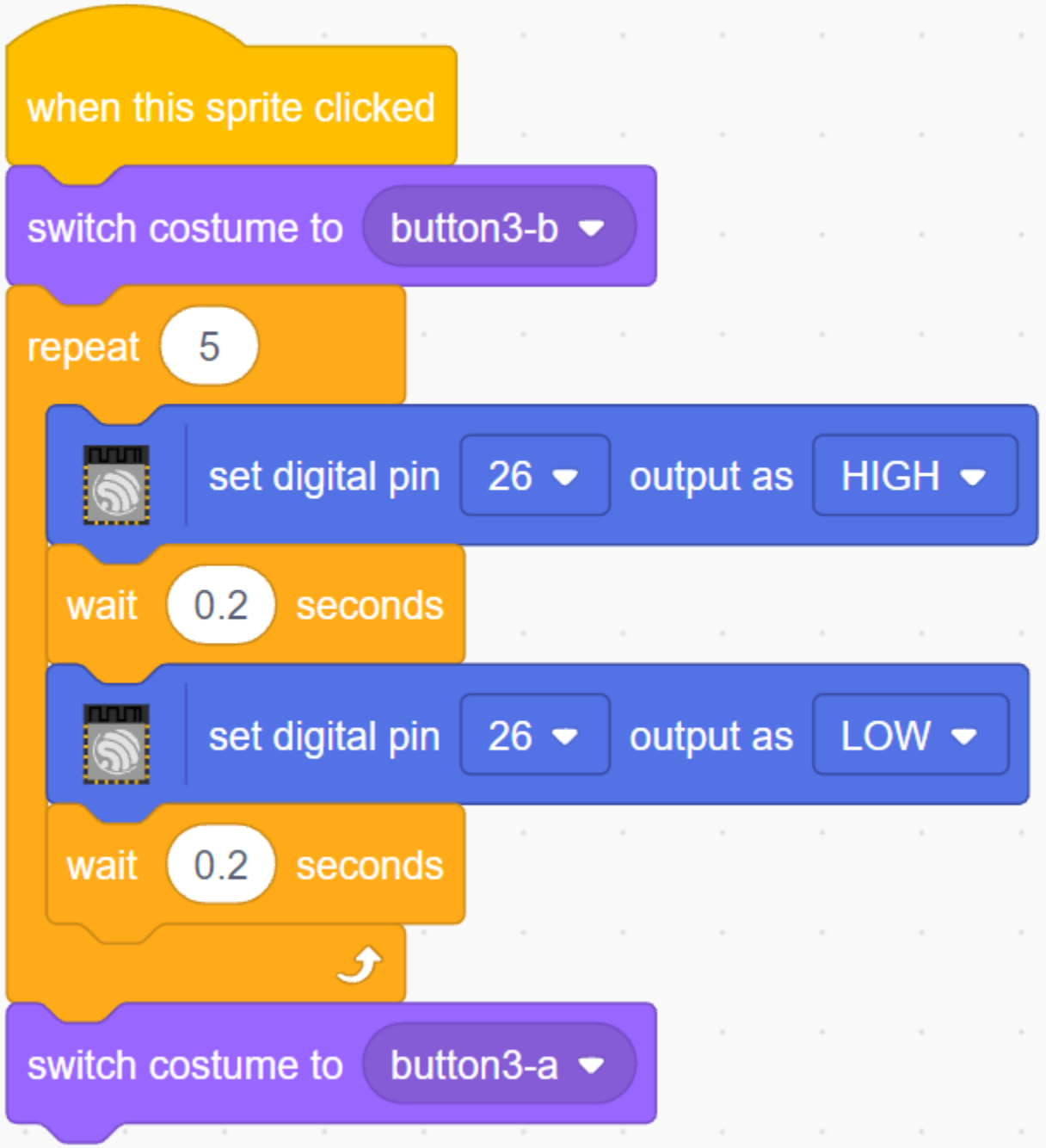
Cuando se hace clic en el sprite (**paleta de Eventos**), cambia al disfraz para **button3-b** (**paleta de Apariencias**).



3. Hacer que el LED parpadee 5 veces

Usa el bloque [Repetir] para hacer que el LED parpadee 5 veces (ciclo Alto -> Bajo) y finalmente cambia el disfraz de vuelta a **button3-a**.

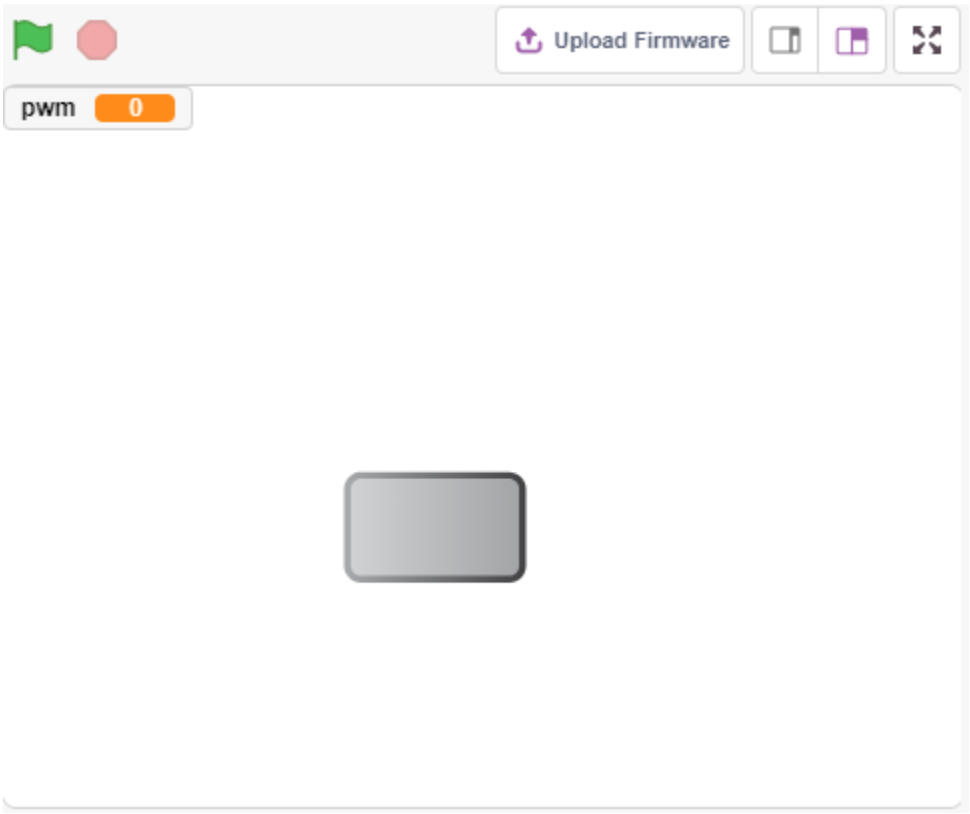
- [Repetir 10]: número limitado de bucles de repetición, puedes establecer el número de repeticiones tú mismo, de la **paleta de Control**.



5.5 2.2 LED Respirando

Ahora usaremos otro método para controlar el brillo del LED. A diferencia del proyecto anterior, aquí el brillo del LED disminuye lentamente hasta desaparecer.

Cuando se hace clic en el sprite en el escenario, el brillo del LED aumenta lentamente y luego se apaga instantáneamente.



5.5.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED	

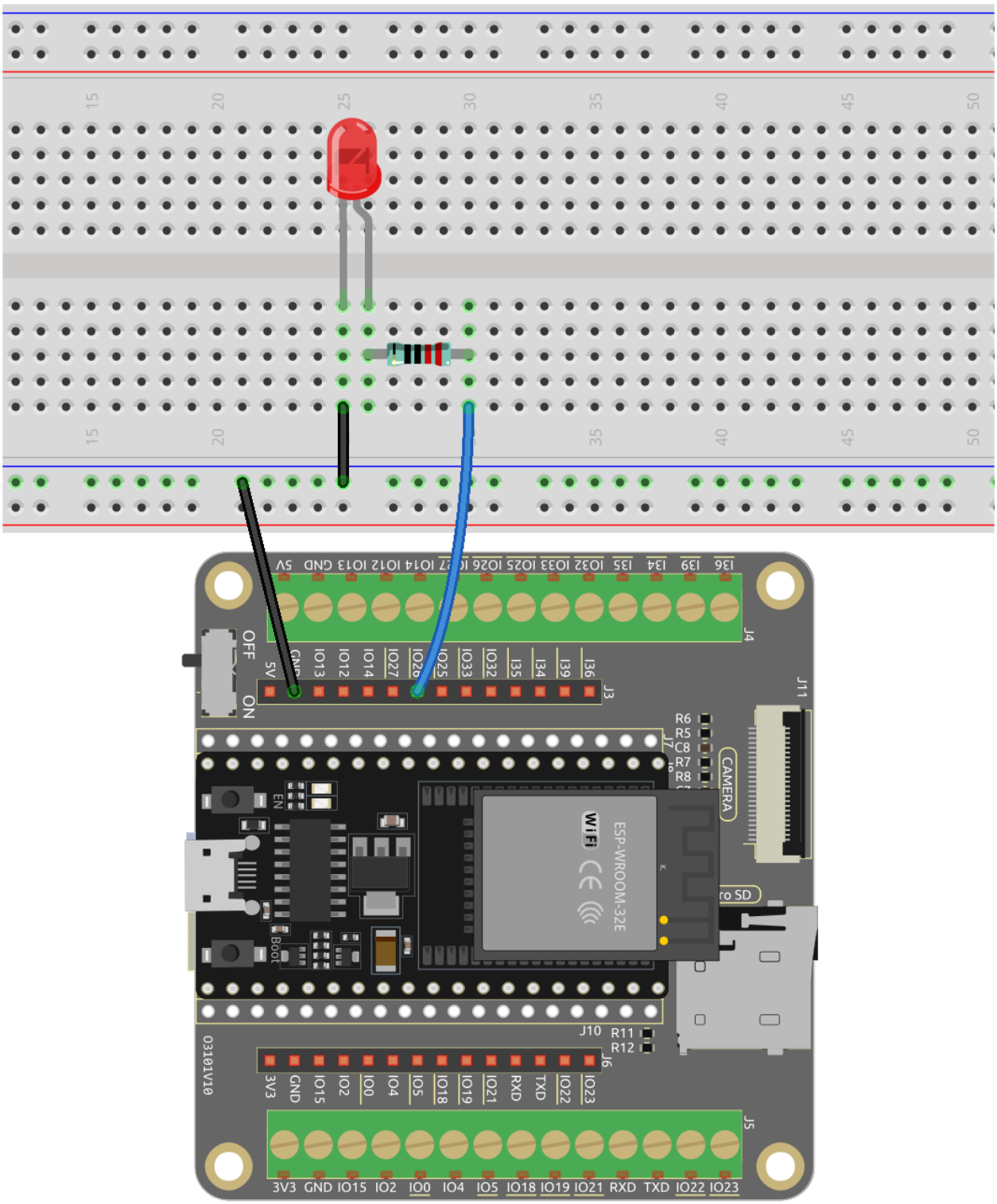
5.5.2 Lo Que Aprenderás

- Establecer el valor de salida del pin PWM
- Crear variables
- Cambiar el brillo del sprite

5.5.3 Construir el Circuito

Este proyecto utiliza el mismo circuito que el proyecto anterior *2.1 Lámpara de Mesa*, pero en lugar de usar ALTO/BAJO para encender o apagar los LEDs, este proyecto utiliza la señal PWM - Wikipedia para iluminar o atenuar lentamente el LED.

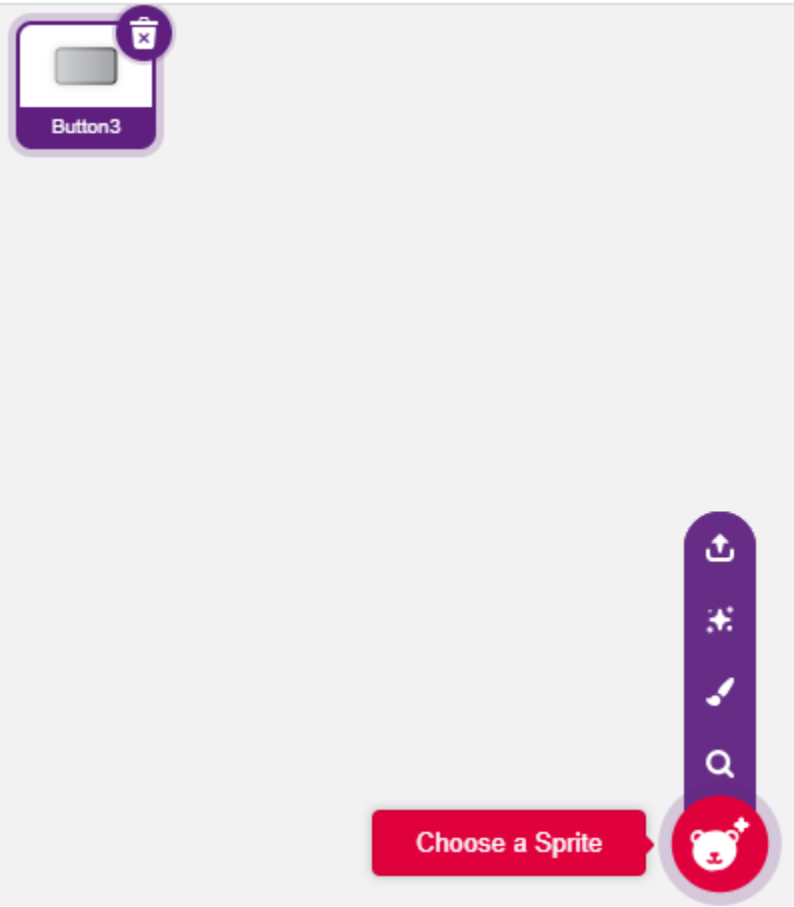
El rango de señal PWM es 0-255, en la placa ESP32, los pines 2, 5, 12~15, 18, 19, 21, 22, 25, 26 y 27 pueden emitir señal PWM.


5V GND IO13 IO12 IO14 IO27 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J4
OFF
ON
5V GND IO13 IO12 IO14 IO27 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J3
J11
R6
R5
C8
R7
R8
J7
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
to SD
J10
R11
R12
J6
03101V10
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23
J5

5.5.4 Programación

1. Seleccionar un sprite

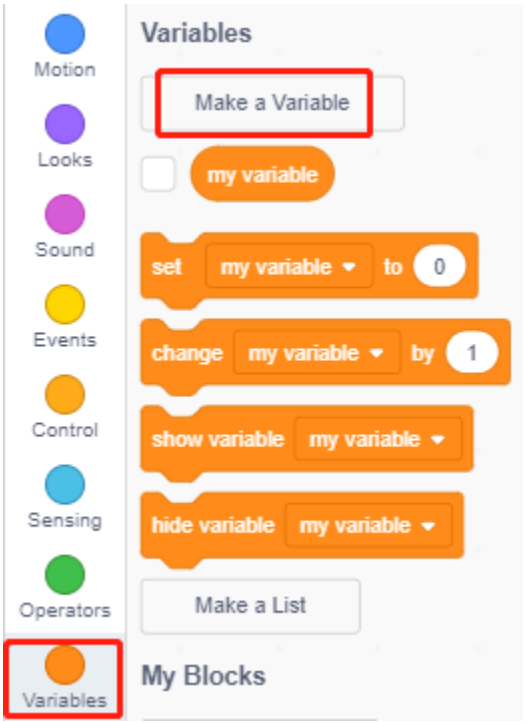
Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprite, introduce **button3** en el cuadro de búsqueda y luego haz clic para añadirlo.



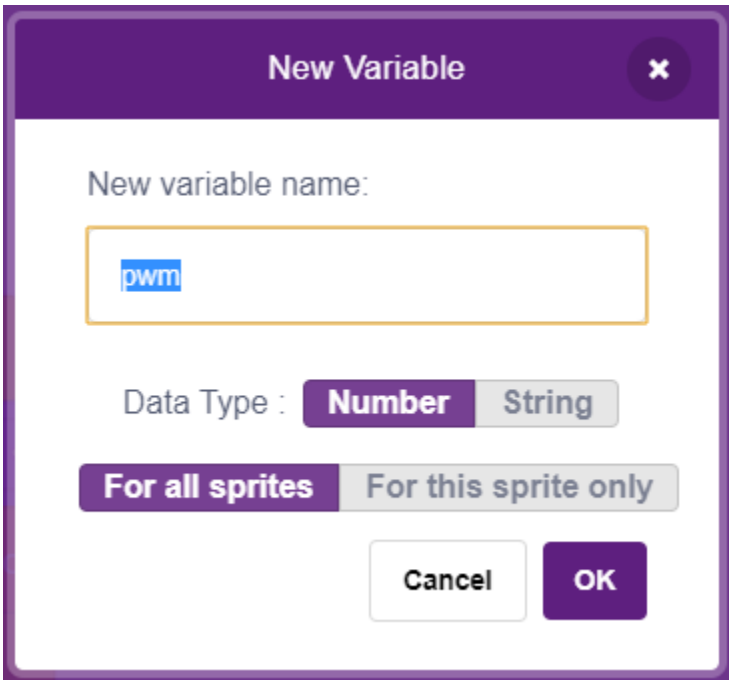
2. Crear una variable.

Crea una variable llamada **pwm** para almacenar el valor del cambio de pwm.

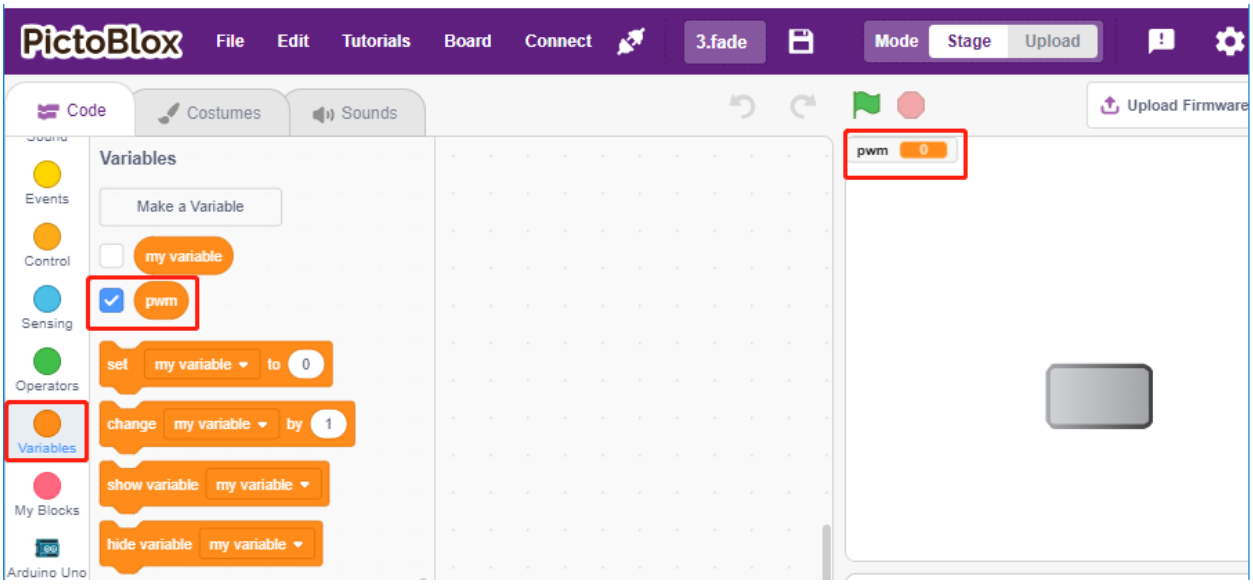
Haz clic en la paleta **Variables** y selecciona **Crear una Variable**.



Introduce el nombre de la variable, puede ser cualquier nombre, pero se recomienda describir su función. El tipo de dato es número y Para todos los sprites.



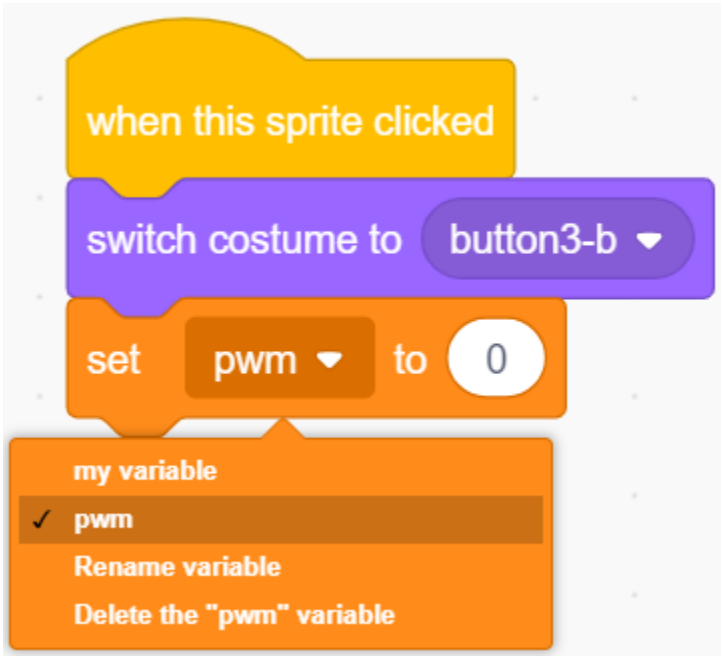
Una vez creada, verás **pwm** dentro de la paleta **Variables** y en estado marcado, lo que significa que esta variable aparecerá en el escenario. Puedes intentar desmarcarla para ver si pwm sigue presente en el escenario.



3. Establecer el estado inicial

Cuando se hace clic en el sprite **button3**, cambia el disfraz a **button-b** (estado de clic), y establece el valor inicial de la variable **pwm** en 0.

- [establecer pwm a 0]: de la paleta **Variables**, usado para establecer el valor de la variable.



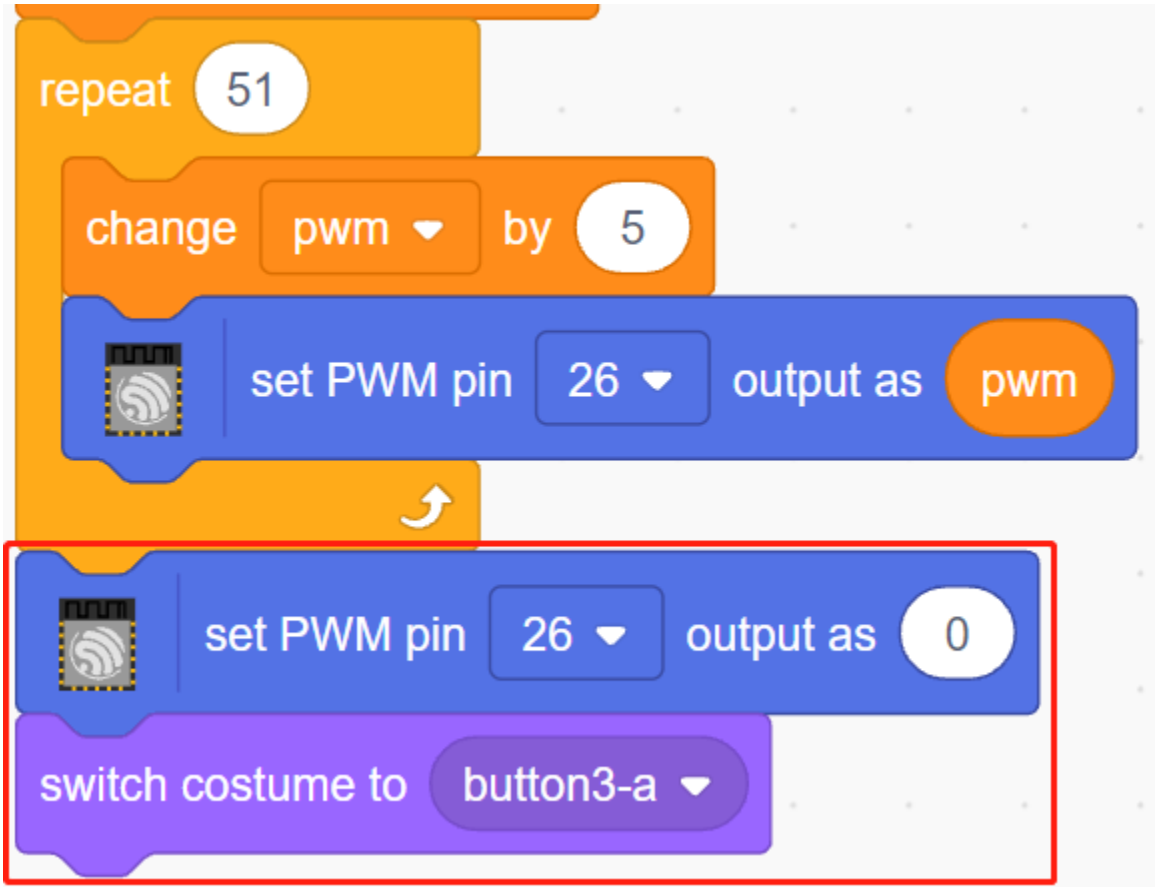
4. Hacer que el LED brille más y más

Dado que el rango de pwm es 255, mediante el bloque [repetir], la variable **pwm** se incrementa en 5 hasta alcanzar 255, y luego se introduce en el bloque [establecer pin PWM], de modo que se puede observar cómo el LED se ilumina lentamente.

- [incrementar pwm en 5]: desde la paleta **Variables**, permite que la variable cambie un número específico cada vez. Puede ser un número positivo o negativo, siendo positivo el incremento en cada ocasión, y negativo la disminución, por ejemplo, aquí la variable pwm se incrementa en 5 en cada ocasión.

- [establecer pin PWM]: desde la paleta **ESP32**, se utiliza para definir el valor de salida del pin pwm.

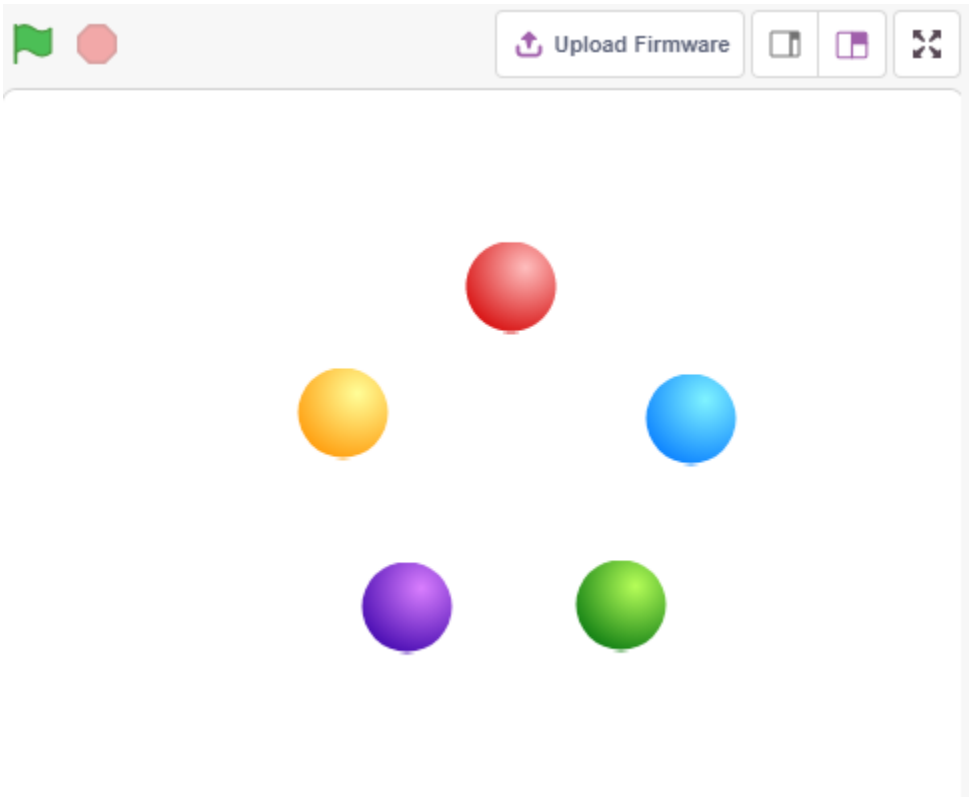
Finalmente, cambia el disfraz del botón3 de nuevo a **botón-a** y haz que el valor del pin PWM sea 0, para que el LED se ilumine lentamente y luego se apague de nuevo.



5.6 2.3 Bolas Coloridas

En este proyecto, haremos que los LED RGB muestren diferentes colores.

Al hacer clic en bolas de diferentes colores en el área de escenario, causará que el LED RGB se ilumine en diferentes colores.



5.6.1 Componentes Necesarios

En este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

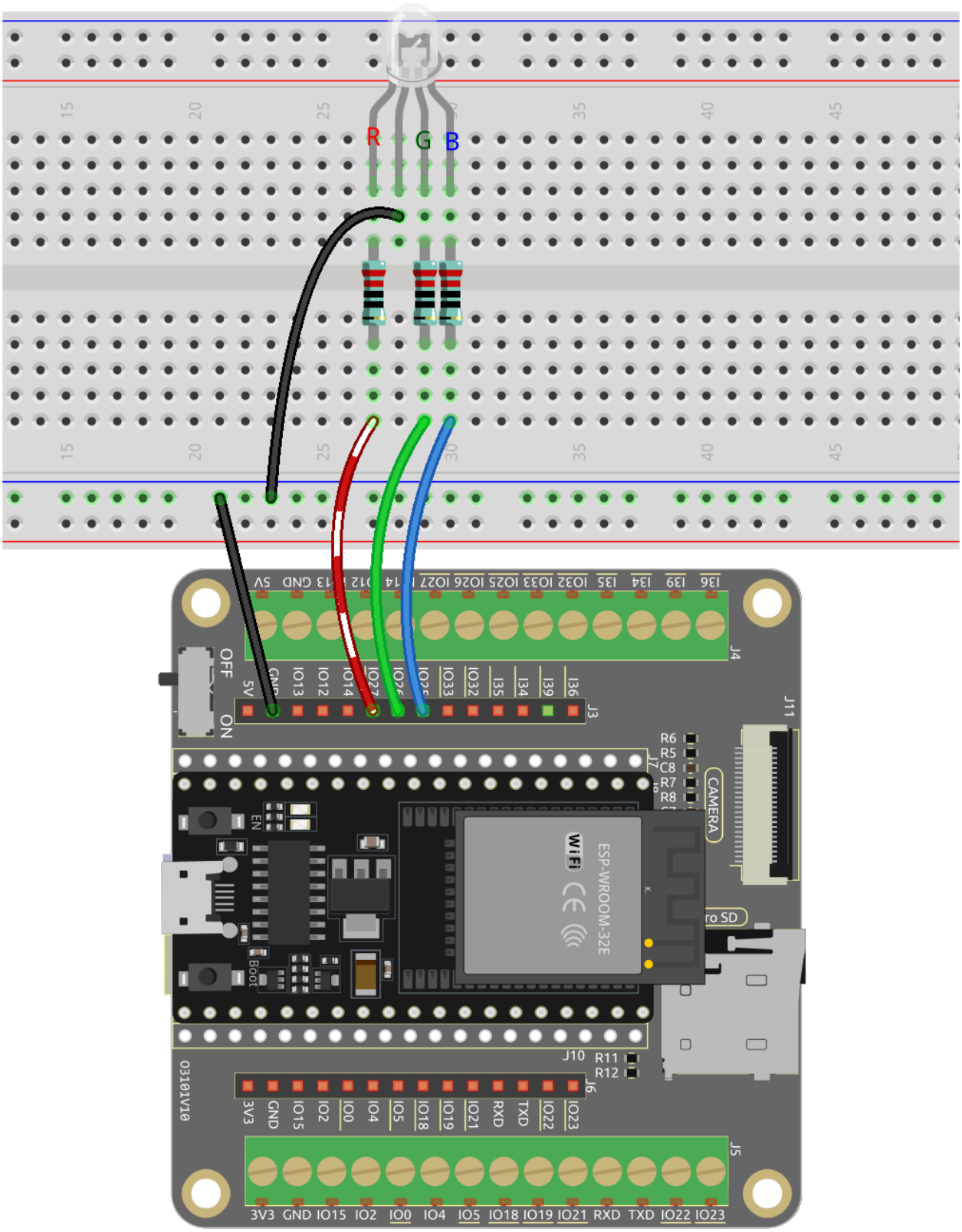
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
LED RGB	

5.6.2 Lo que aprenderás

- El principio del LED RGB
- Copiar sprites y seleccionar diferentes disfraces
- Superposición de los tres colores primarios

5.6.3 Construir el circuito

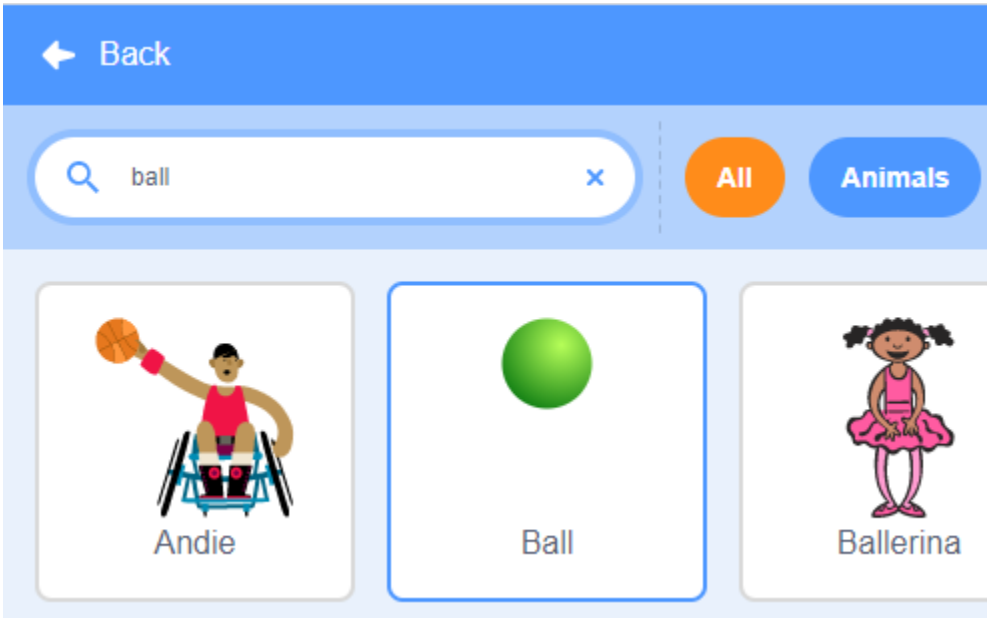
Un LED RGB empaqueta tres LEDs de rojo, verde y azul en una cáscara de plástico transparente o semitransparente. Puede mostrar varios colores cambiando el voltaje de entrada de los tres pines y superponiéndolos, lo que, según las estadísticas, puede crear 16,777,216 colores diferentes.


R
G
B
ESP-WROOM-32E
CAMERA
J4
J3
J11
J10
J6
J5
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

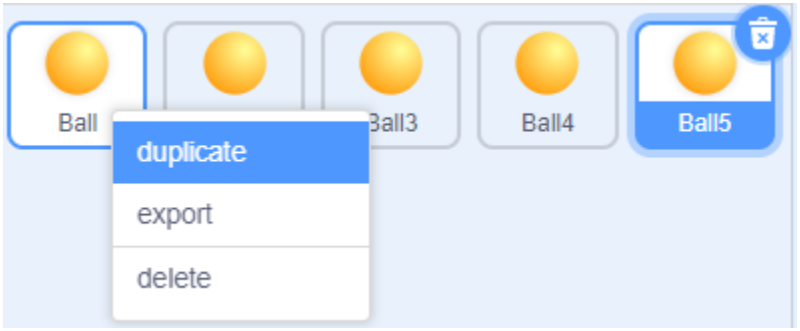
5.6.4 Programación

1. Seleccionar sprite

Elimina el sprite predeterminado, luego elige el sprite **Bola**.

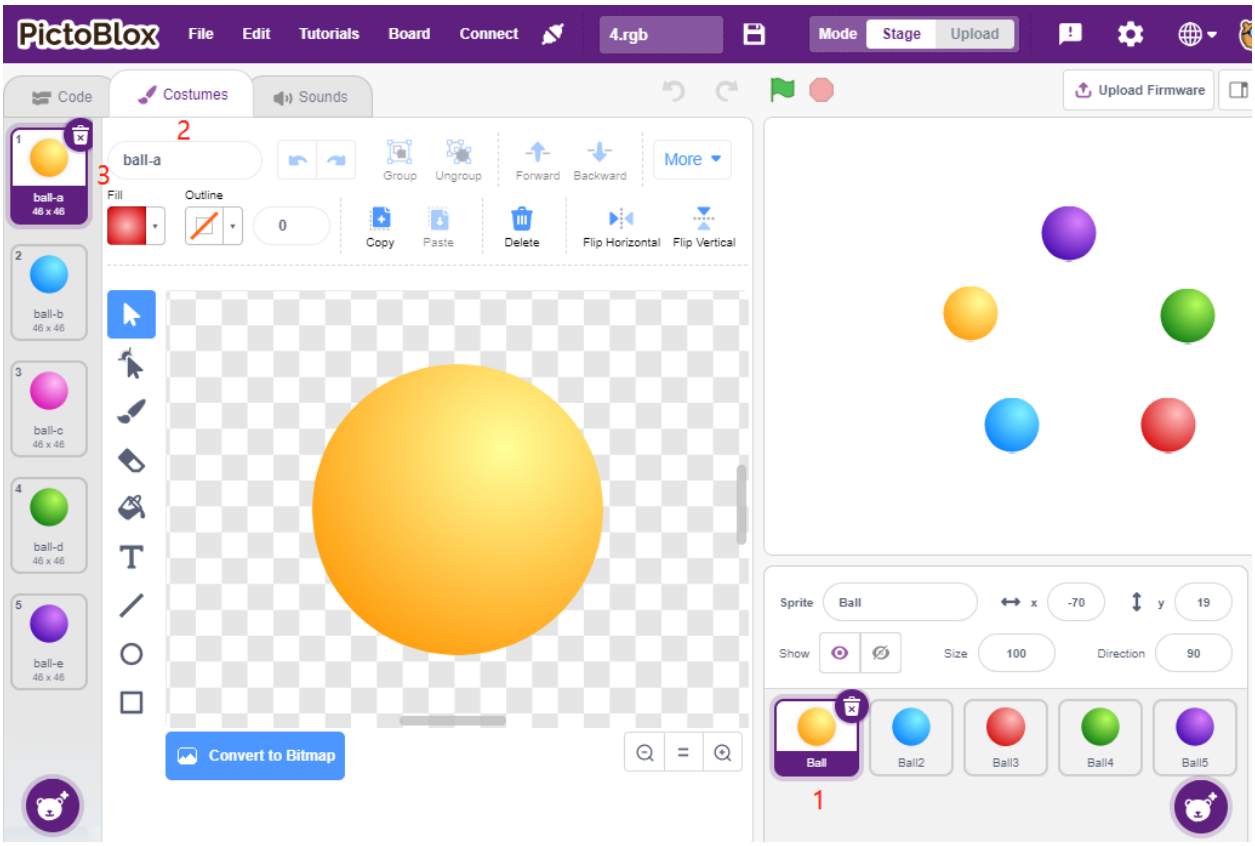


Y duplícalo 5 veces.



Elige diferentes disfraces para estos 5 sprites **Bola** y muévelos a las posiciones correspondientes.

Nota: El color del disfraz del sprite **Bola3** necesita ser cambiado manualmente a rojo.

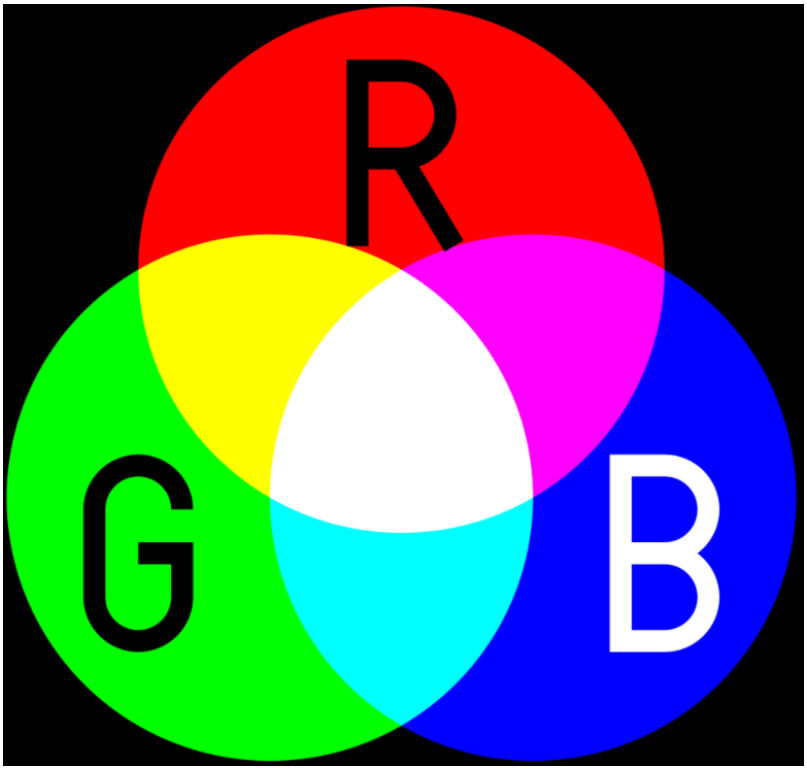


2. Hacer que los LED RGB se iluminen en el color apropiado

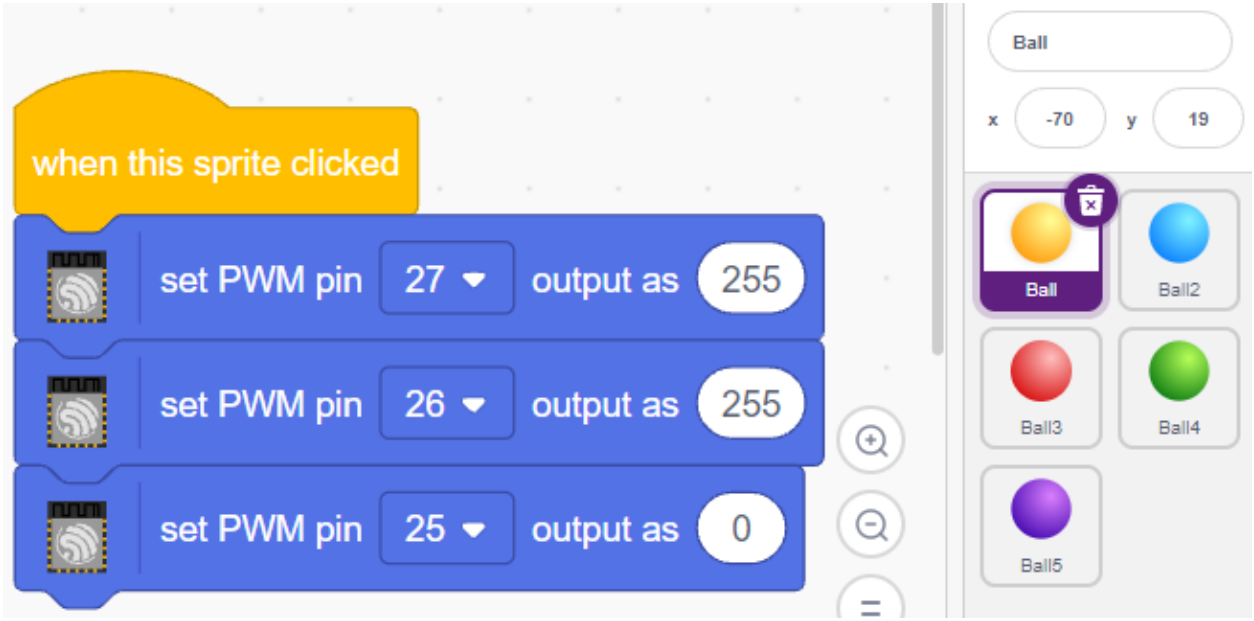
Antes de entender el código, necesitamos entender el modelo de color RGB.

El modelo de color RGB es un modelo de color aditivo en el cual la luz roja, verde y azul se suman de diversas maneras para reproducir una amplia gama de colores.

Mezcla de colores aditiva: añadir rojo a verde produce amarillo; añadir verde a azul produce cian; añadir azul a rojo produce magenta; añadir los tres colores primarios juntos produce blanco.



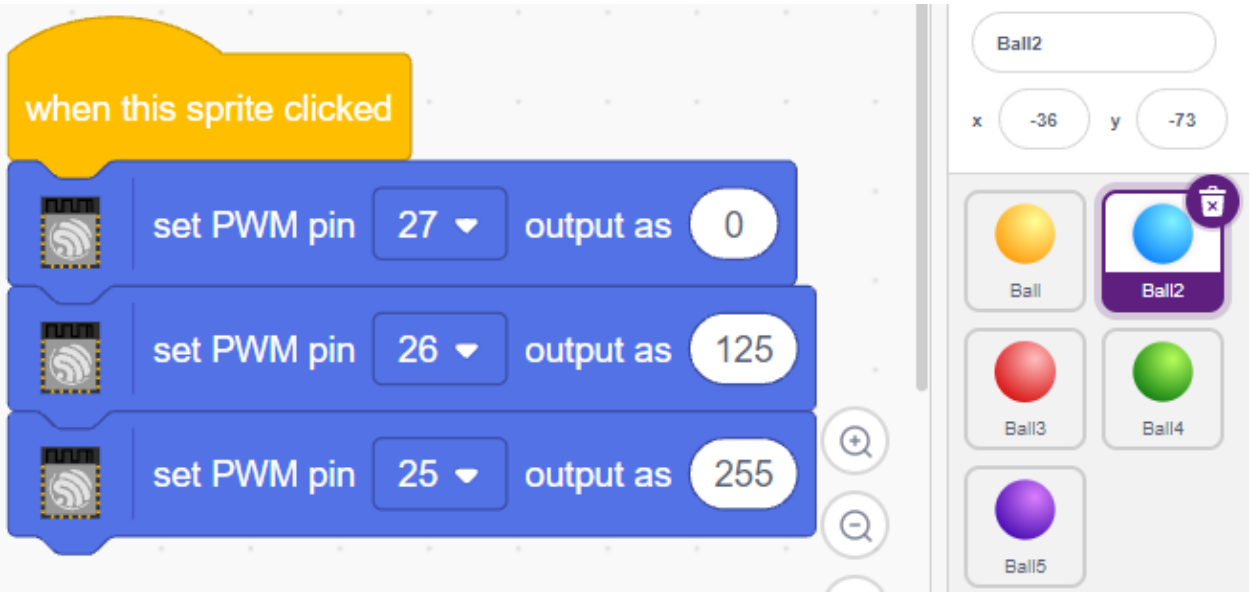
Entonces, el código para hacer que el LED RGB se ilumine de amarillo es el siguiente.



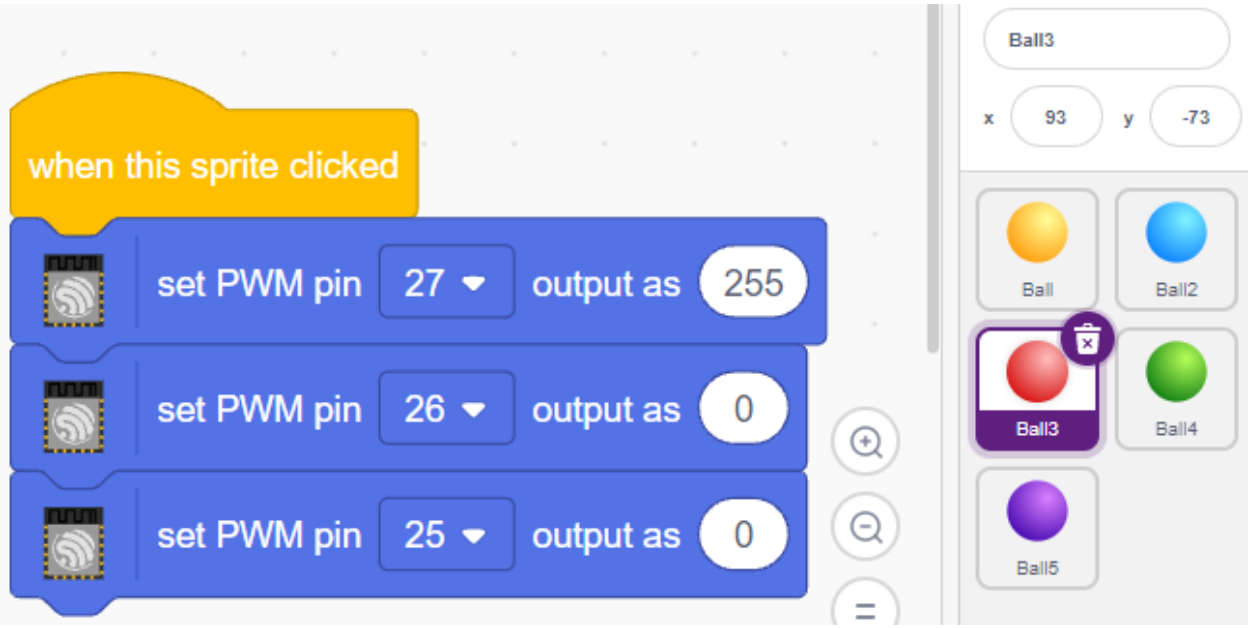
Cuando se hace clic en el sprite Bola (bola amarilla), configuramos el pin 27 en alto (LED rojo encendido), el pin 26 en alto (LED verde encendido) y el pin 25 en bajo (LED azul apagado) para que el LED RGB se ilumine de amarillo.

Puedes escribir códigos para otros sprites de la misma manera para hacer que los LED RGB se iluminen en los colores correspondientes.

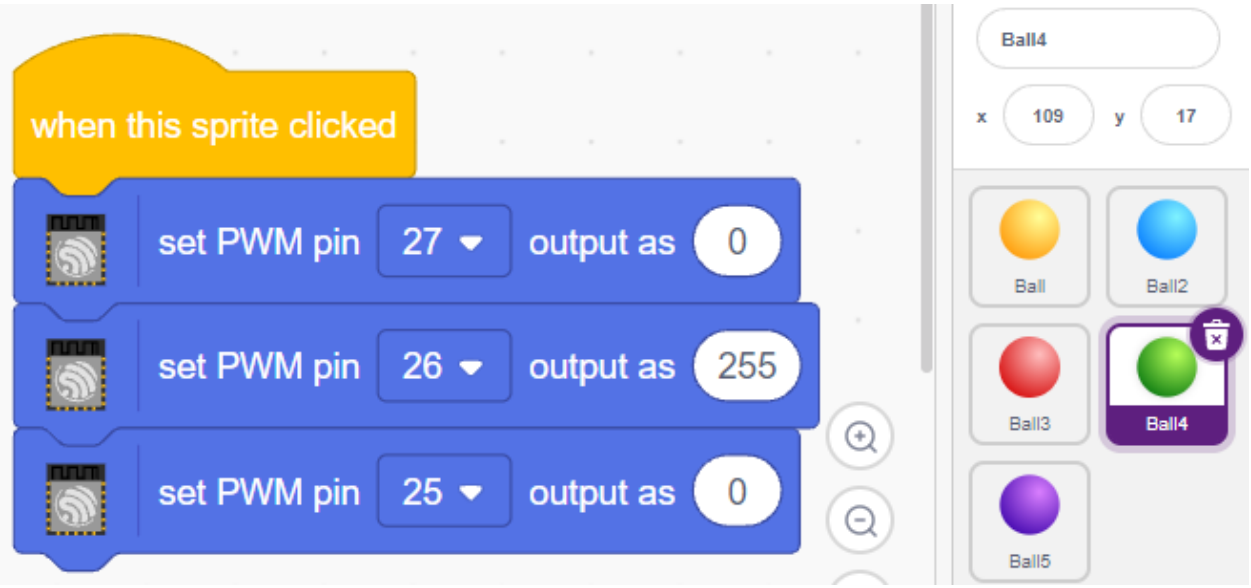
3. Sprite Bola2 (azul claro)



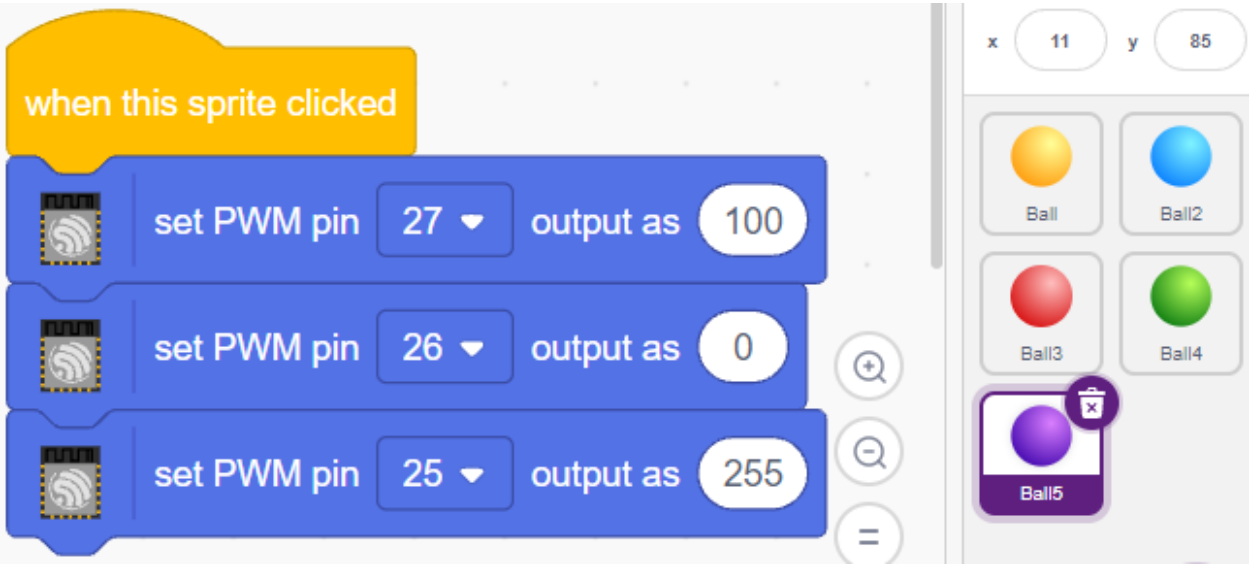
4. Sprite Bola3 (rojo)



5. Sprite Bola4 (verde)



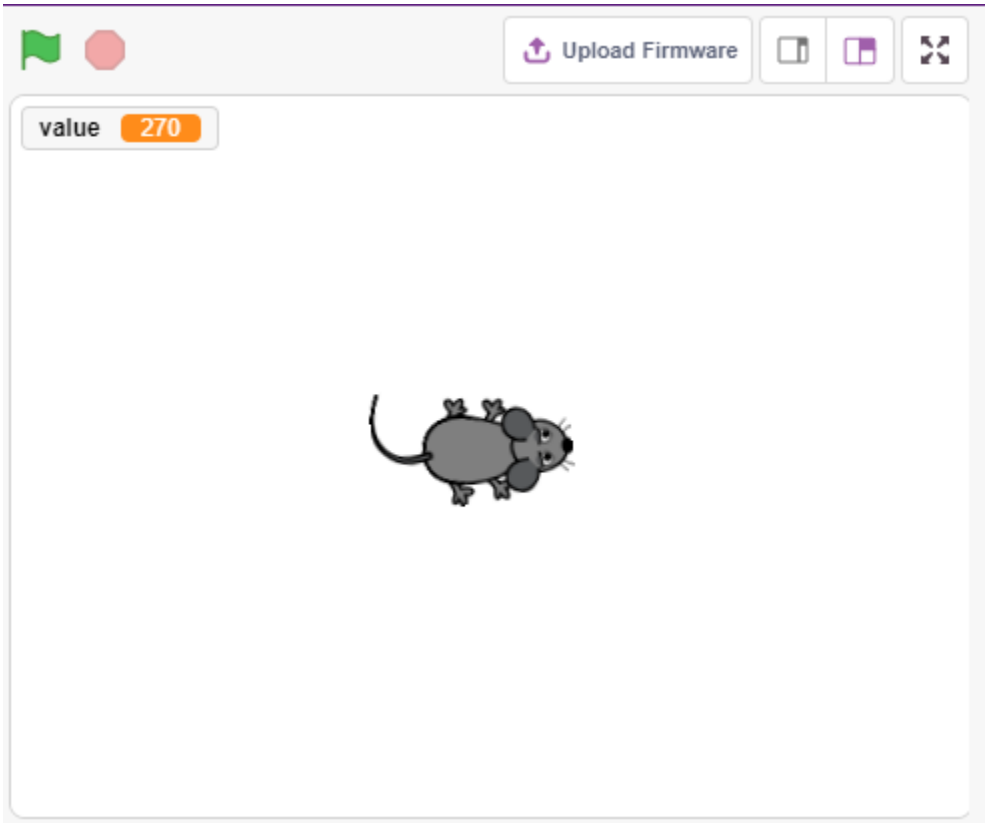
6. Sprite Bola5 (morado)



5.7 2.4 Ratón en Movimiento

Hoy vamos a hacer un juguete de ratón controlado por un potenciómetro.

Cuando se hace clic en la bandera verde, el ratón en el escenario se mueve hacia adelante, y al girar el potenciómetro, el ratón cambiará la dirección de movimiento.



5.7.1 Componentes necesarios

En este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

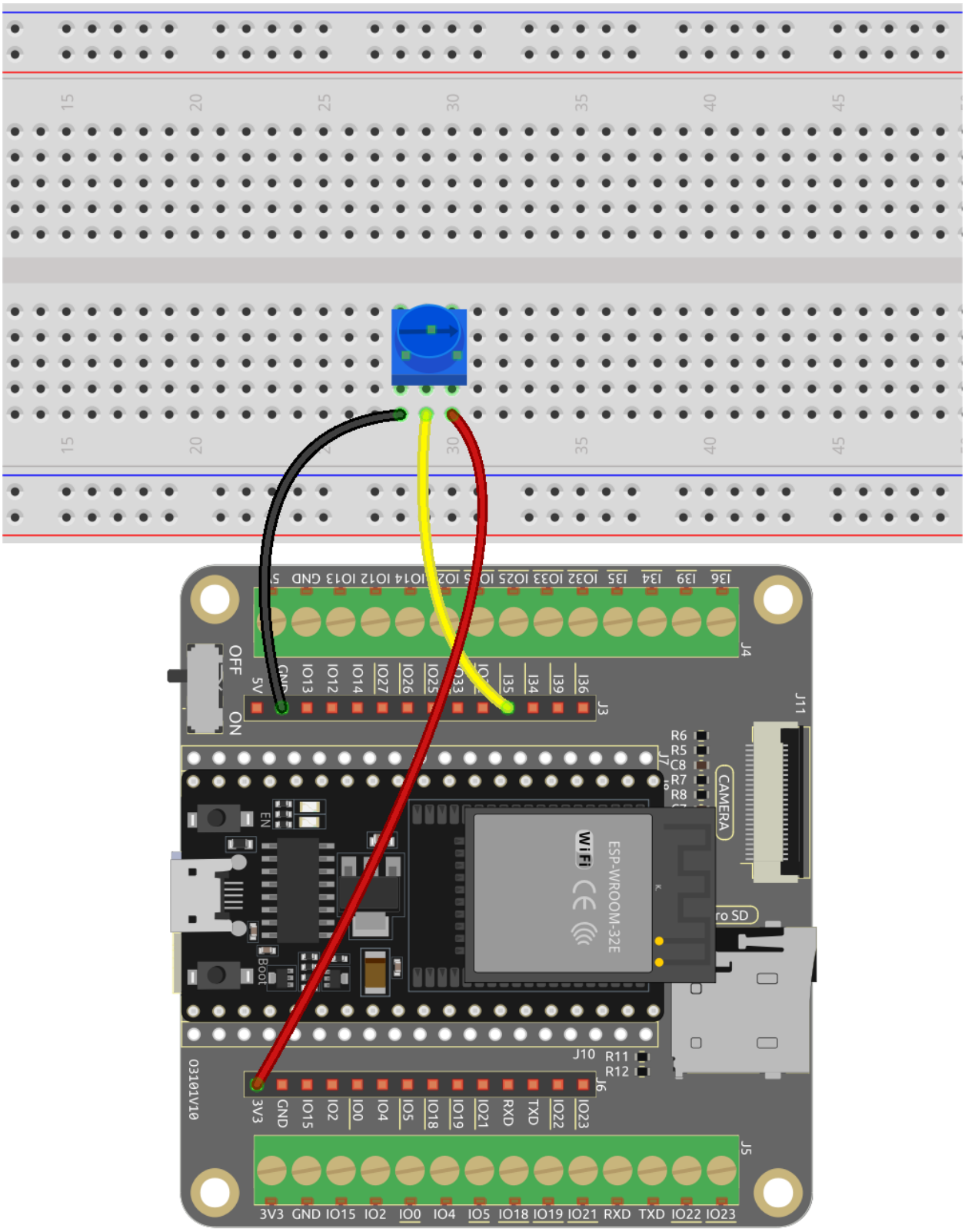
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Potenciómetro	

5.7.2 Lo que aprenderás

- Principio del potenciómetro
- Leer pin analógico y rangos
- Mapear un rango a otro
- Mover y cambiar la dirección del sprite

5.7.3 Construir el circuito

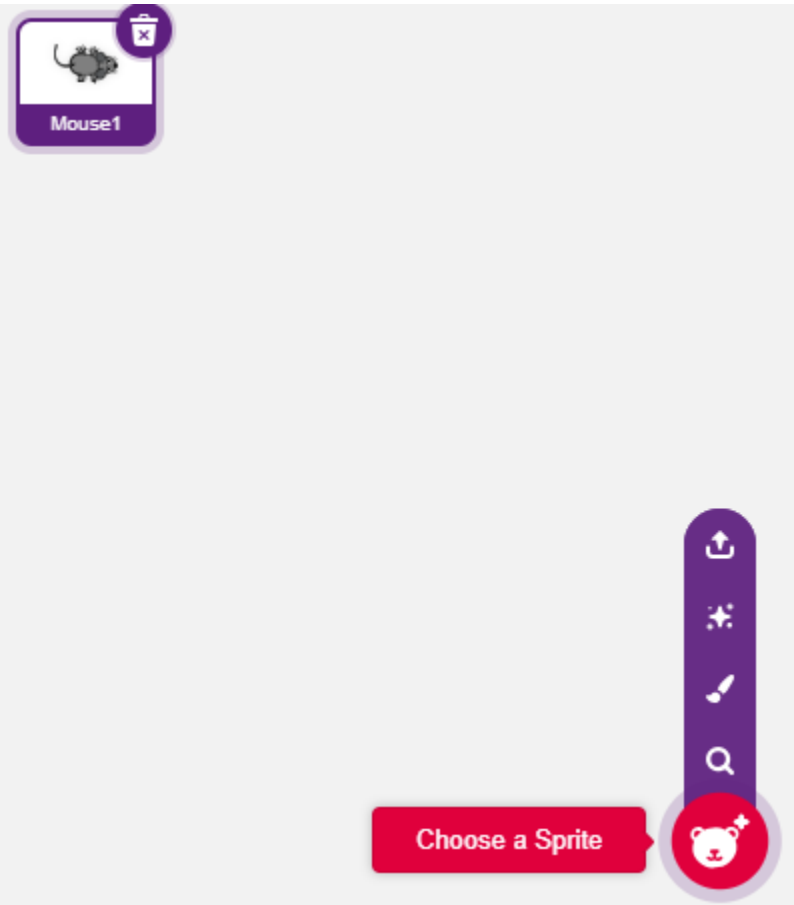
El potenciómetro es un elemento resistivo con 3 terminales, los pines laterales están conectados a 5V y GND, y el pin del medio está conectado al pin35. Después de la conversión por el convertidor ADC del ESP32, el rango de valores es 0-4095.


OFF
ON
5V
GND
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
I35
I34
I39
I36
J4
J3
J11
R6
R5
C8
R7
R8
J7
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
J10
R11
R12
J6
3V3
GND
IO15
IO2
IO0
IO4
IO5
IO18
IO19
IO21
RXD
TXD
IO22
IO23
J5
03101V10

5.7.4 Programación

1. Elegir un sprite

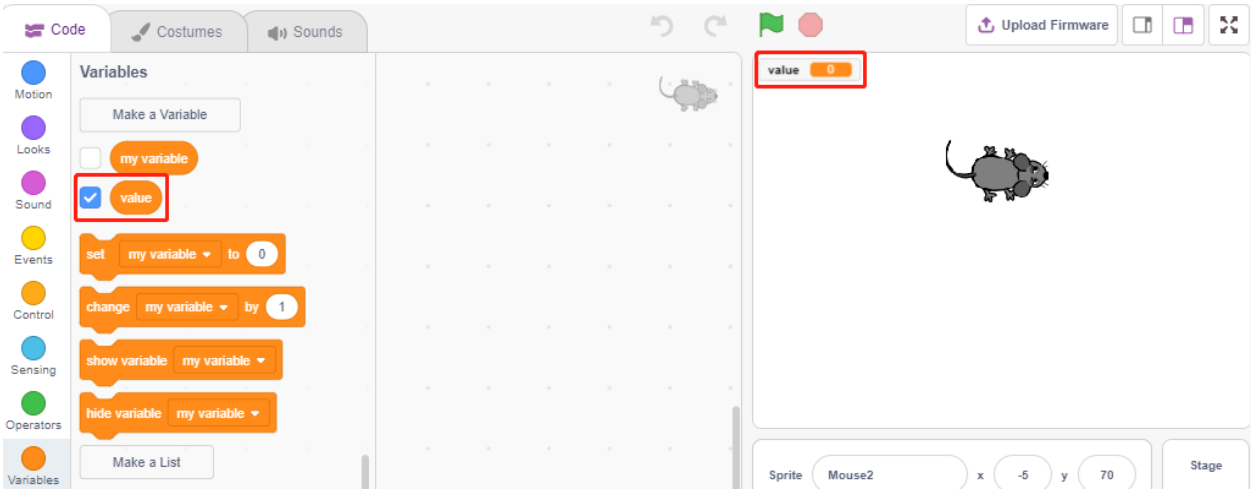
Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprites, ingresa **ratón** en la caja de búsqueda y luego haz clic para añadirlo.



2. Crear una variable.

Crea una variable llamada **valor** para almacenar el valor leído del potenciómetro.

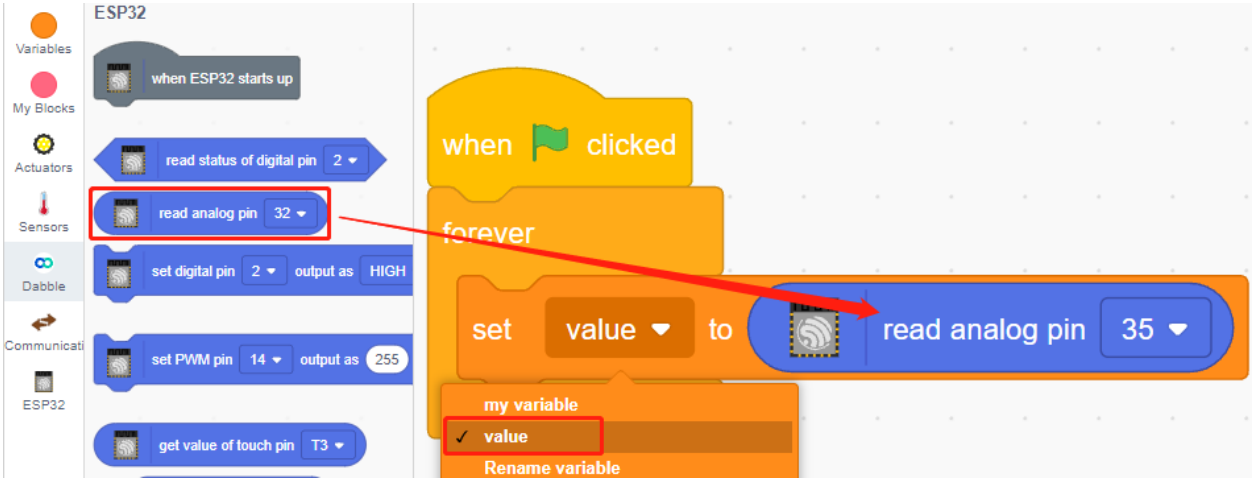
Una vez creada, verás **valor** aparecer dentro de la paleta **Variables** y en estado marcado, lo que significa que esta variable aparecerá en el escenario.



3. Leer el valor del pin35

Almacena el valor leído del pin35 en la variable **valor**.

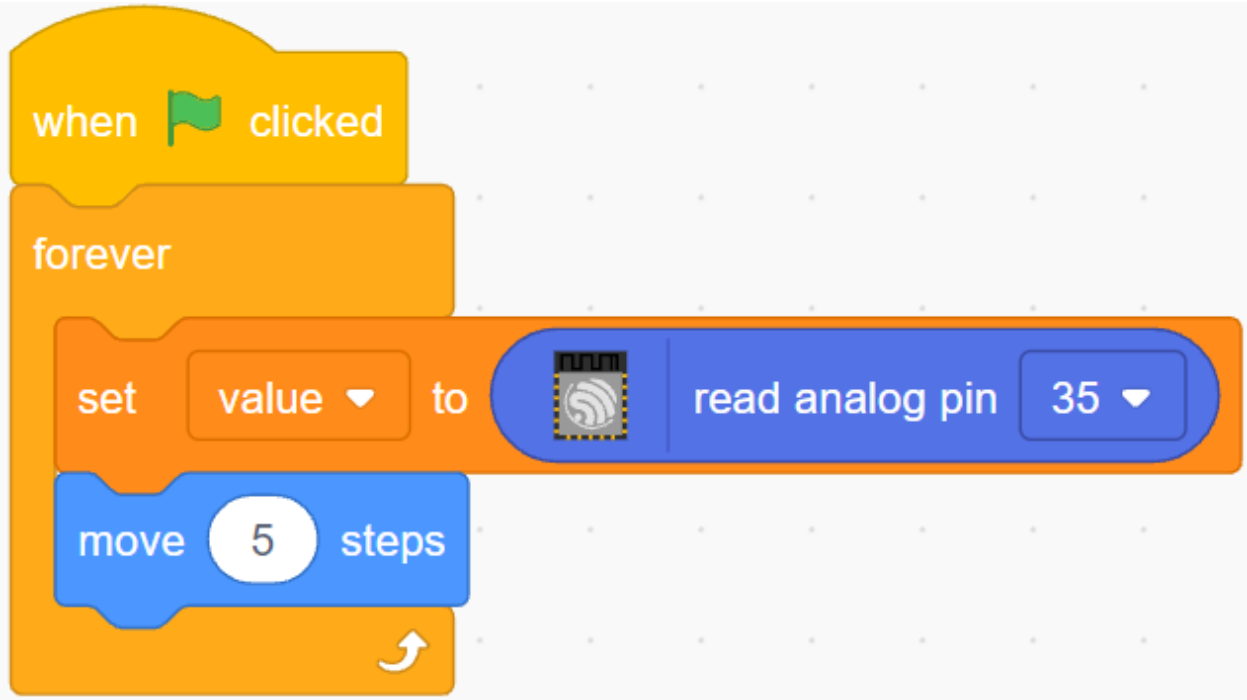
- [establecer mi variable a 0]: Establece el valor de la variable.
- [leer pin analógico ()]: Lee el valor de los pines en el rango de 0-4095.



Para poder leer de manera continua, necesitas usar el bloque [siempre]. Haz clic en este script para ejecutarlo, gira el potenciómetro en ambas direcciones y verás que el rango de valores es 0-1023.

4. Mover el sprite

Usa el bloque [mover pasos] para mover el sprite, ejecuta el script y verás que el sprite se mueve del medio hacia la derecha.

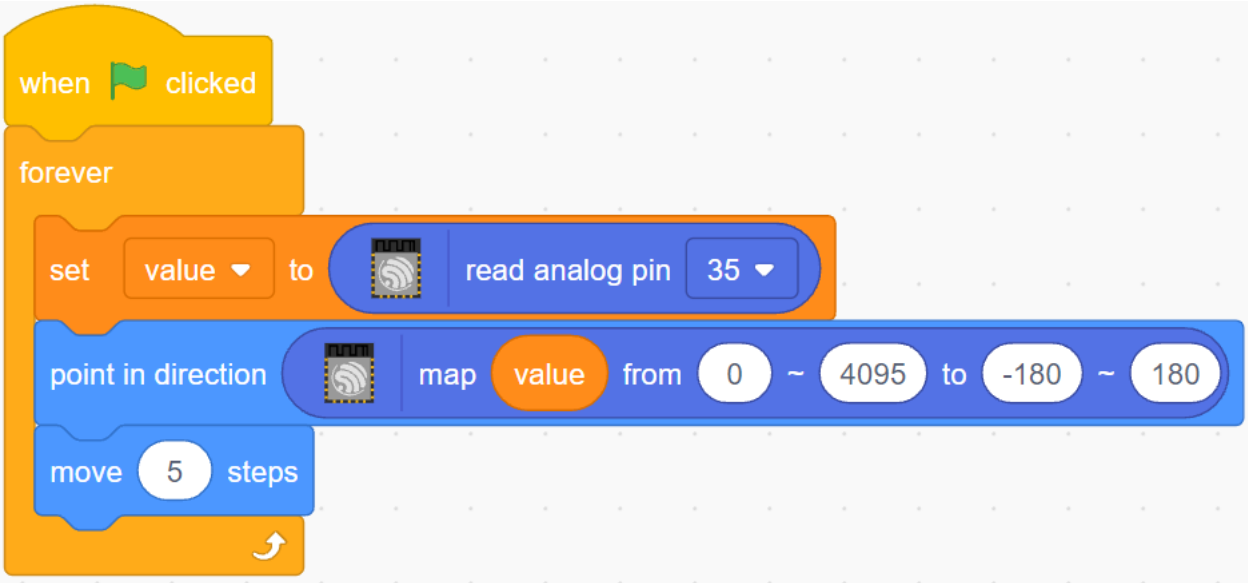


5. Cambiar la dirección del movimiento del sprite

Ahora cambia la dirección del movimiento del sprite por el valor del pin35. Dado que el valor del pin35 varía de 0-4095, pero la dirección de rotación del sprite es de -180~180, se necesita usar un bloque [mapear].

También agrega [cuando se hace clic en la bandera verde] al principio para iniciar el script.

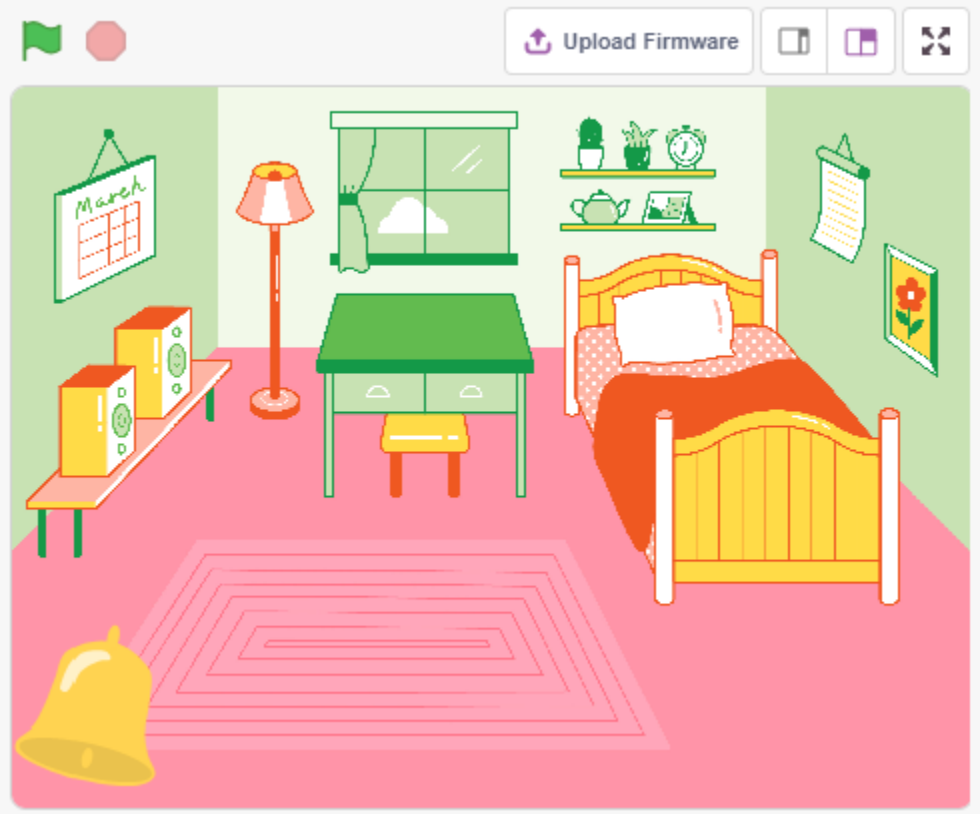
- [apuntar en dirección]: Establece el ángulo de dirección del sprite, desde la paleta **Movimiento**.
- [mapear de a]: Mapea un rango a otro rango.



5.8 2.5 Timbre

Aquí, usaremos el botón y la campana en el escenario para hacer un timbre.

Después de hacer clic en la bandera verde, puedes presionar el botón y la campana en el escenario emitirá un sonido.



5.8.1 Componentes necesarios

En este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Botón	

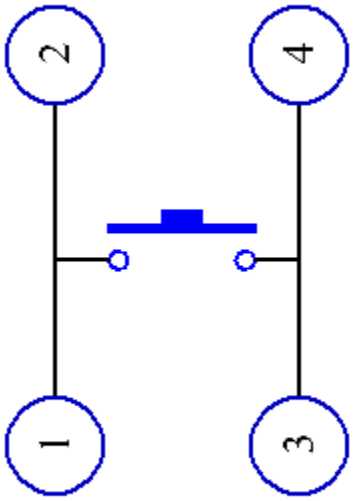
5.8.2 Lo que aprenderás

- Cómo funciona el botón
- Leyendo el pin digital y sus rangos
- Creando un bucle condicional
- Añadiendo un fondo
- Reproduciendo sonido

5.8.3 Construir el circuito

El botón es un dispositivo de 4 pines, ya que el pin 1 está conectado al pin 2, y el pin 3 al pin 4, cuando se presiona el botón, los 4 pines se conectan, cerrando así el circuito.

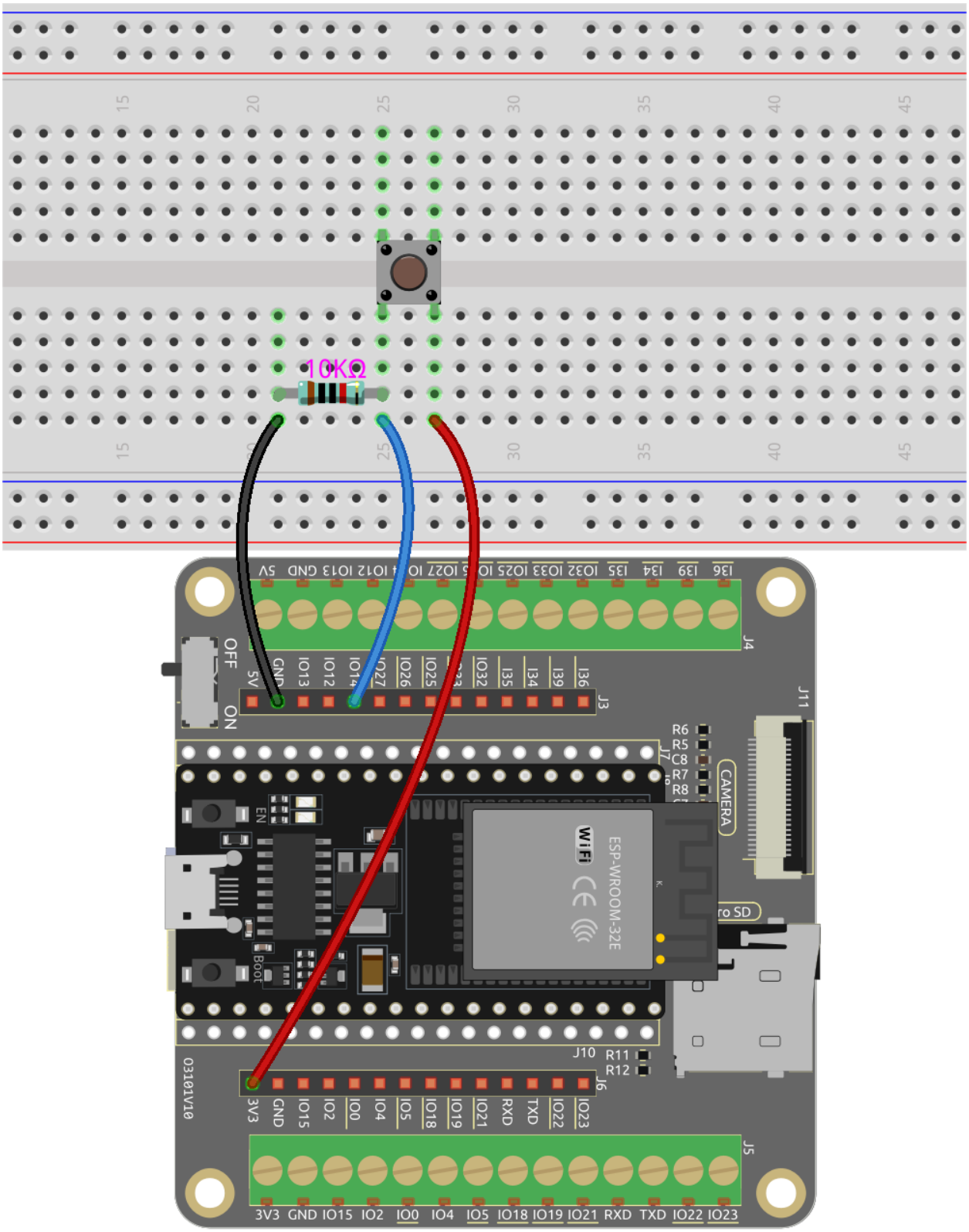
Button



Internal Structure

Construye el circuito según el siguiente diagrama.

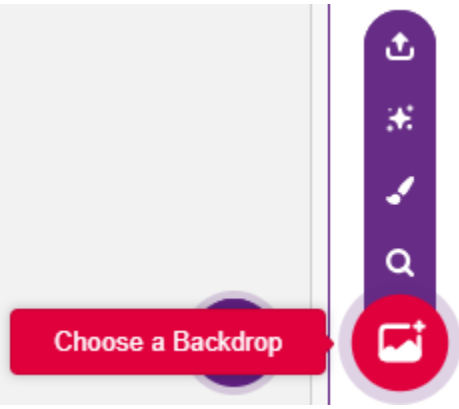
- Conecta uno de los pines del lado izquierdo del botón al pin14, que está conectado a una resistencia de pull-down y un capacitor de 0.1uF (104) (para eliminar el jitter y emitir un nivel estable cuando el botón está funcionando).
- Conecta el otro extremo de la resistencia y el capacitor a GND, y uno de los pines del lado derecho del botón a 5V.


10KΩ
ESP-WROOM-32E
CAMERA
ON
OFF

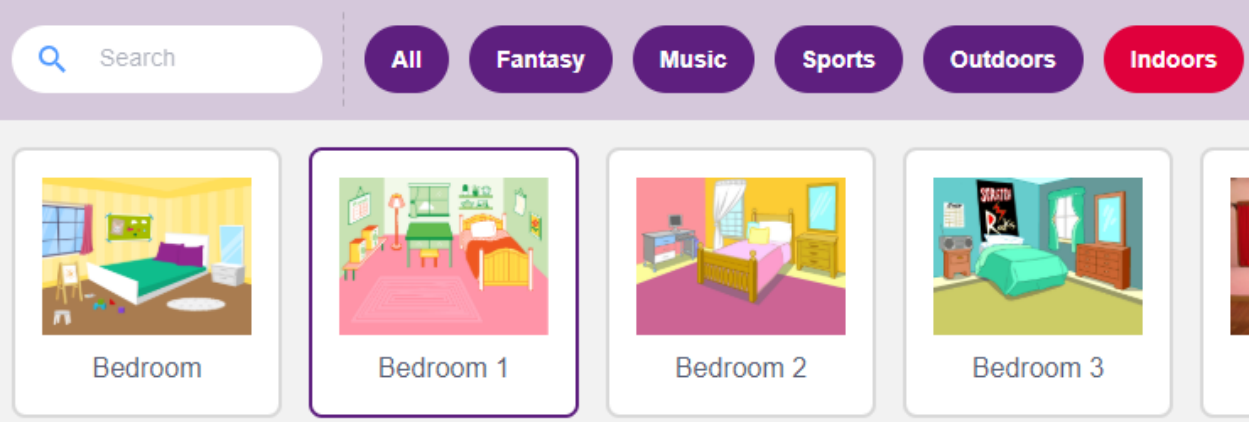
5.8.4 Programación

1. Añadir un fondo

Haz clic en el botón **Elegir un fondo** en la esquina inferior derecha.

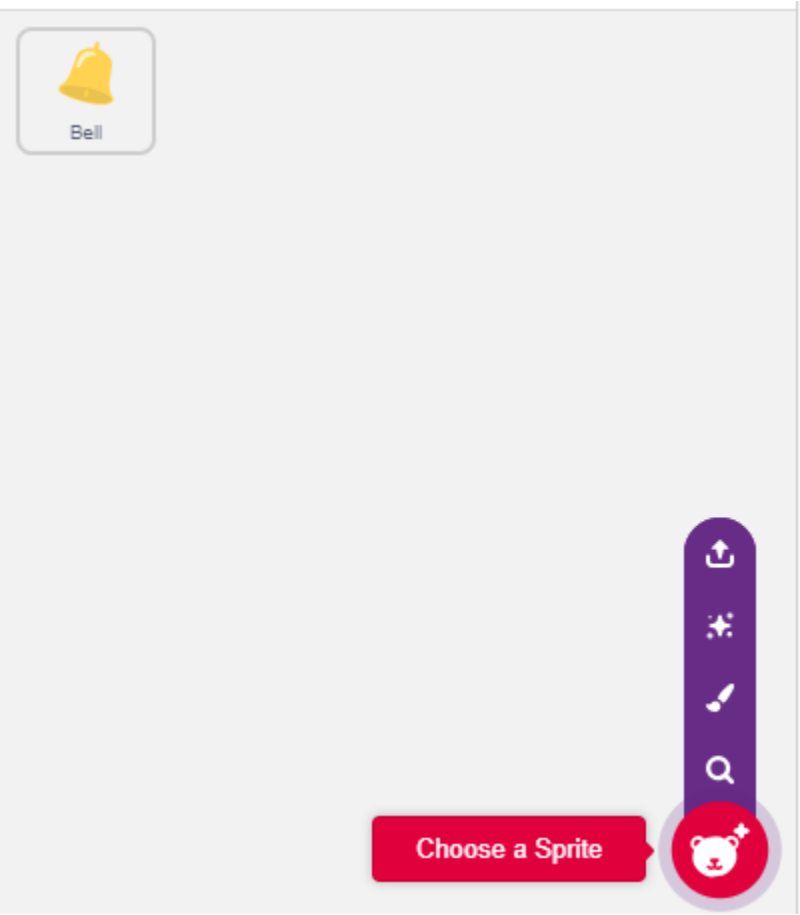


Elige **Dormitorio 1**.

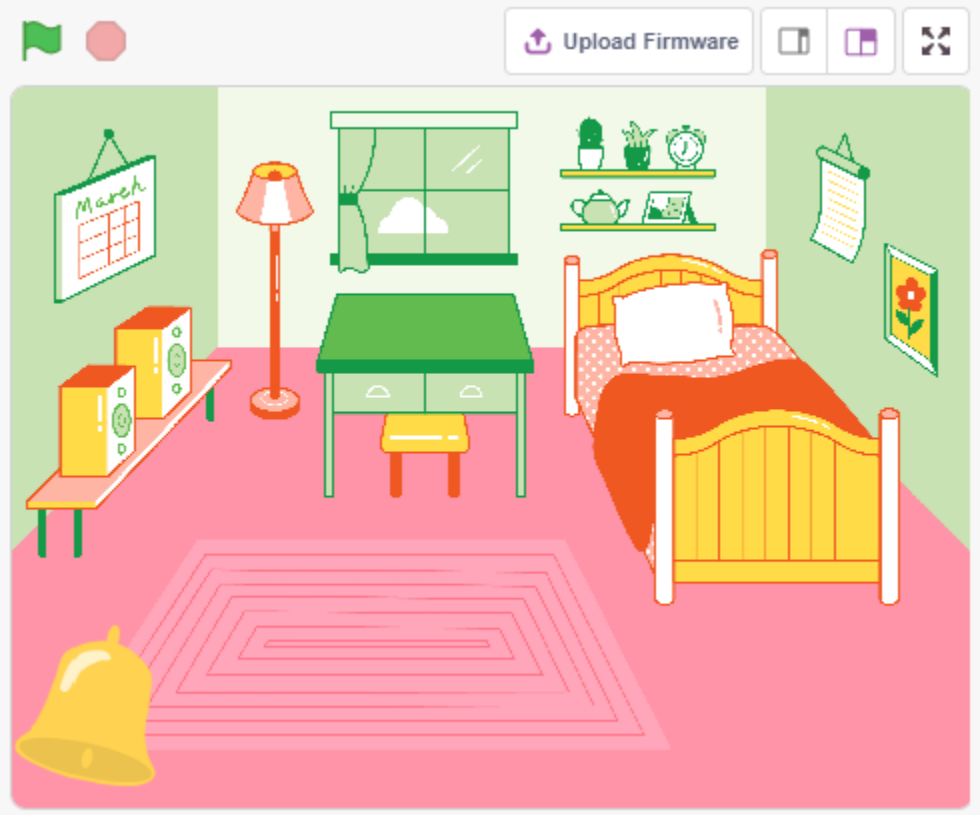


2. Seleccionar el sprite

Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprites, ingresa **campana** en la caja de búsqueda y luego haz clic para añadirla.



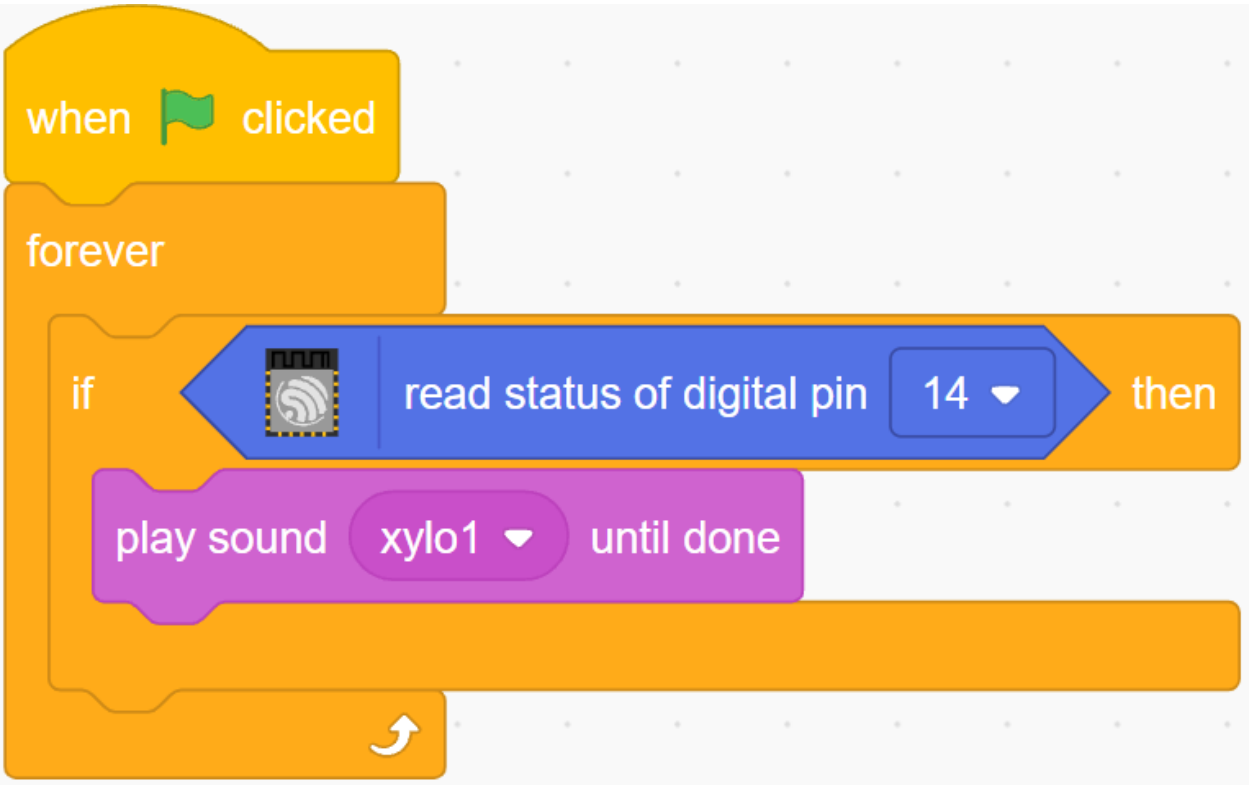
Luego selecciona el sprite **campana** en el escenario y muévelo a la posición correcta.



3. Presionar el botón y la campana hace un sonido

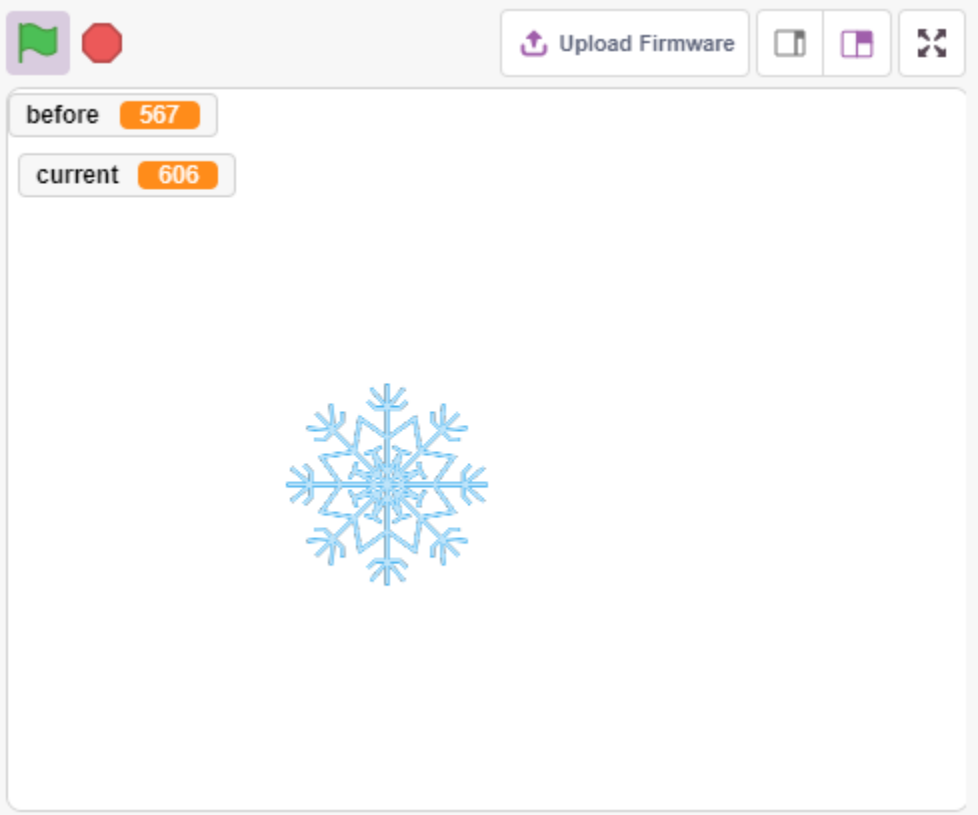
Usa [si entonces] para hacer una declaración condicional que cuando el valor del pin14 leído sea igual a 1 (el botón está presionado), se reproducirá el sonido **xylo1**.

- [leer estado de pin digital]: Este bloque es de la paleta **ESP32** y se usa para leer el valor de un pin digital, el resultado es 0 o 1.
- [si entonces]: Este bloque es un bloque de control y de la paleta **Control**. Si su condición booleana es verdadera, los bloques que contiene se ejecutarán, y luego el script involucrado continuará. Si la condición es falsa, los scripts dentro del bloque serán ignorados. La condición solo se verifica una vez; si la condición se vuelve falsa mientras el script dentro del bloque se está ejecutando, seguirá ejecutándose hasta que haya terminado.
- [reproducir sonido hasta que termine]: Este bloque es de la paleta de Sonido, usado para reproducir sonidos específicos.



5.9 2.6 Alarma de Baja Temperatura

En este proyecto, haremos un sistema de alarma de baja temperatura, cuando la temperatura esté por debajo del umbral, el sprite **Copito de Nieve** aparecerá en el escenario.



5.9.1 Componentes necesarios

En este proyecto, necesitamos los siguientes componentes.

Definitivamente es conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Termistor	

5.9.2 Lo que aprenderás

- Principio de funcionamiento del termistor
- Operaciones multivariables y sustractivas

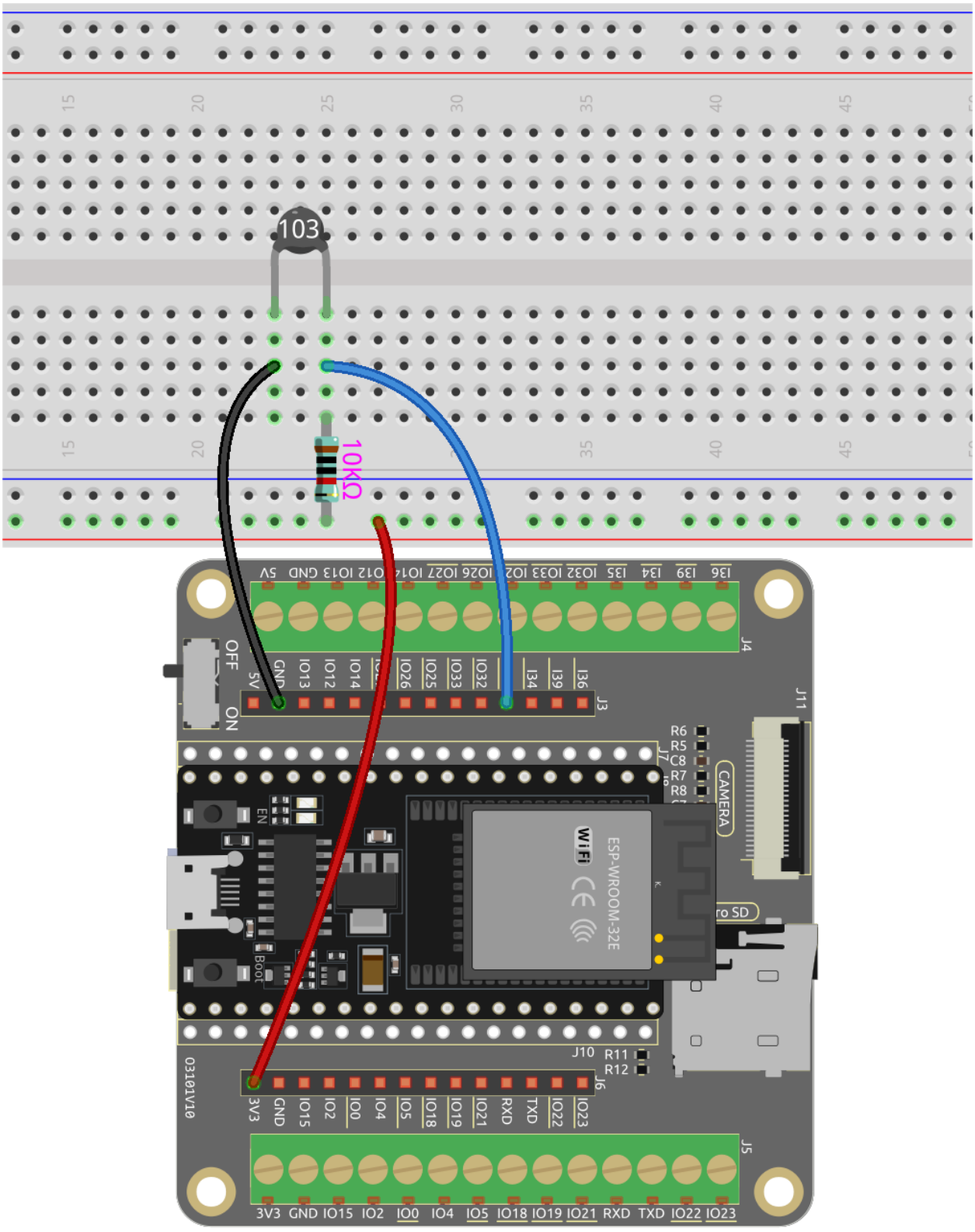
5.9.3 Construir el circuito

Un termistor es un tipo de resistencia cuya resistencia depende fuertemente de la temperatura, más que en las resistencias estándar, y hay dos tipos de resistencias, PTC (la resistencia aumenta a medida que aumenta la temperatura) y PTC (la resistencia disminuye a medida que aumenta la temperatura).

Construye el circuito según el siguiente diagrama.

Un extremo del termistor está conectado a GND, el otro extremo está conectado al pin35, y una resistencia de 10K está conectada en serie a 5V.

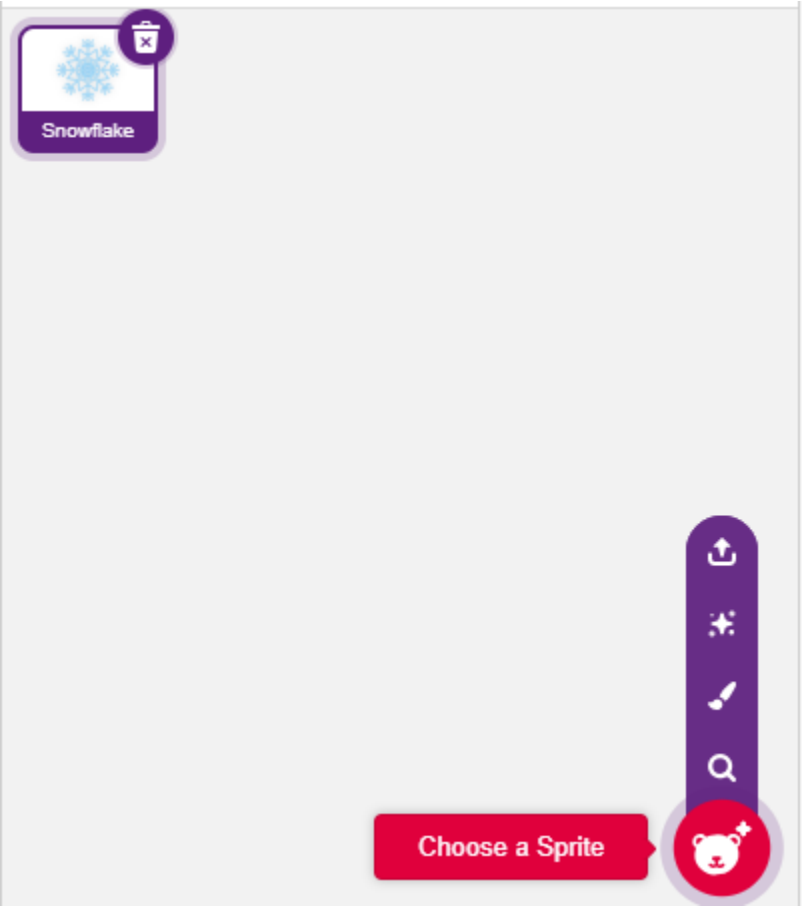
Aquí se utiliza el termistor NTC, así que cuando la temperatura sube, la resistencia del termistor disminuye, la división de voltaje del pin35 disminuye, y el valor obtenido del pin35 disminuye, y viceversa aumenta.


103
10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

5.9.4 Programación

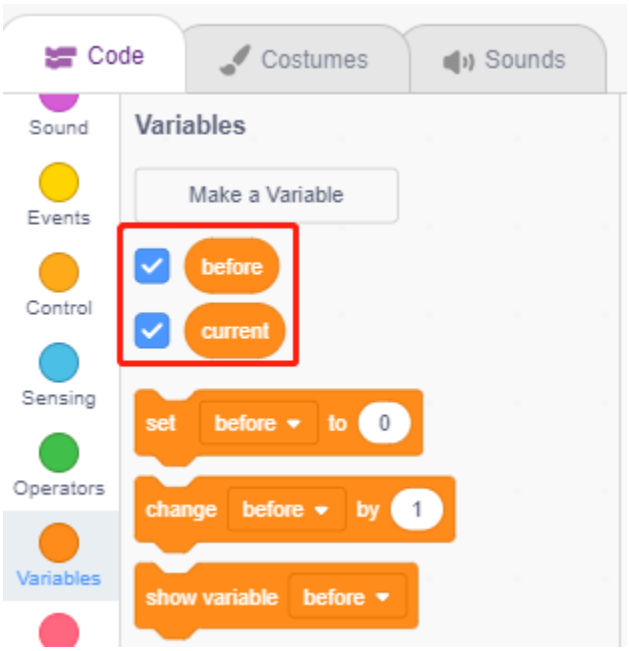
1. Seleccionar un sprite

Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprites, introduce **Copito de Nieve** en la caja de búsqueda y luego haz clic para añadirlo.



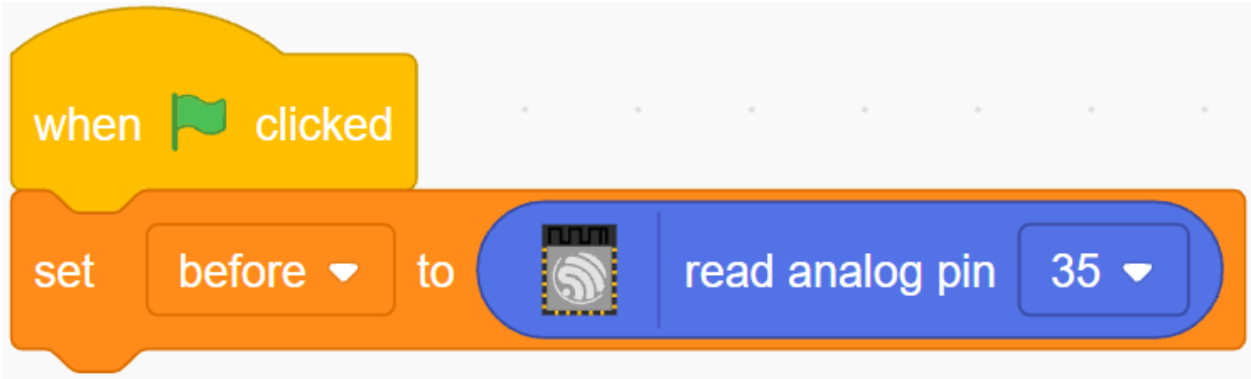
2. Crear 2 variables

Crea dos variables, **antes** y **actual**, para almacenar el valor del pin35 en diferentes casos.



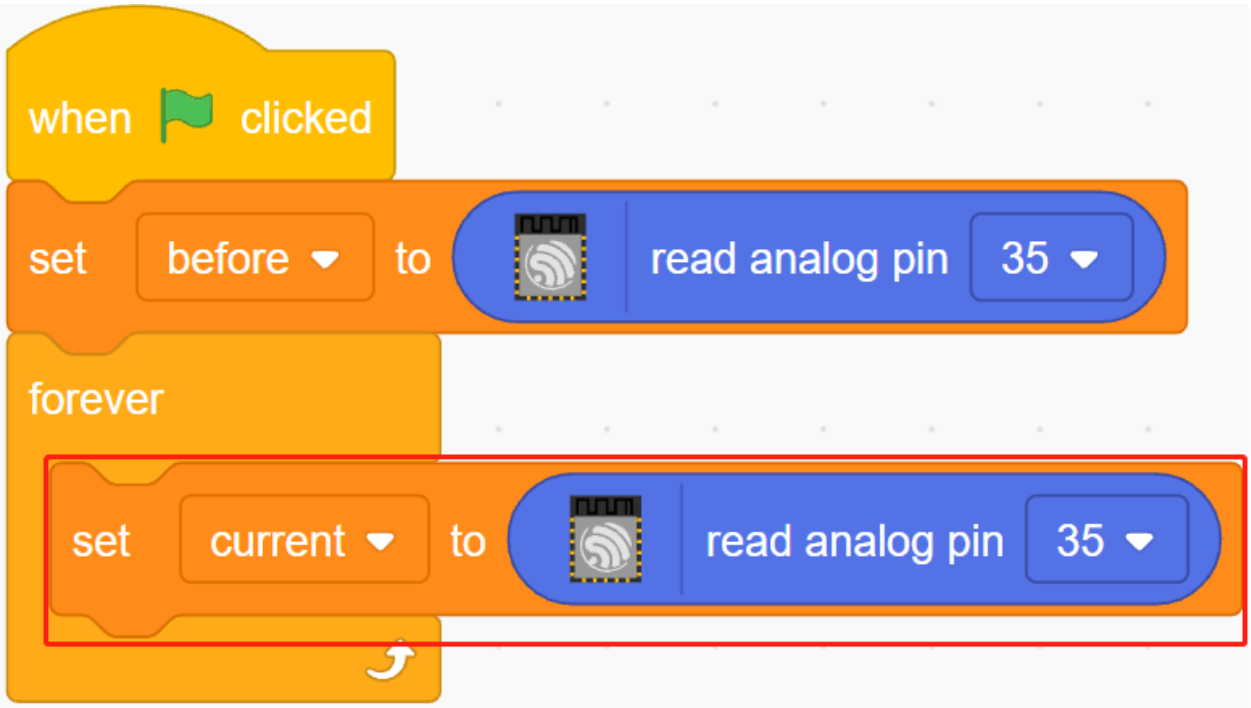
3. Leer el valor del pin35

Cuando se hace clic en la bandera verde, se lee y almacena el valor del pin35 en la variable **antes**.



4. Leer nuevamente el valor del pin35

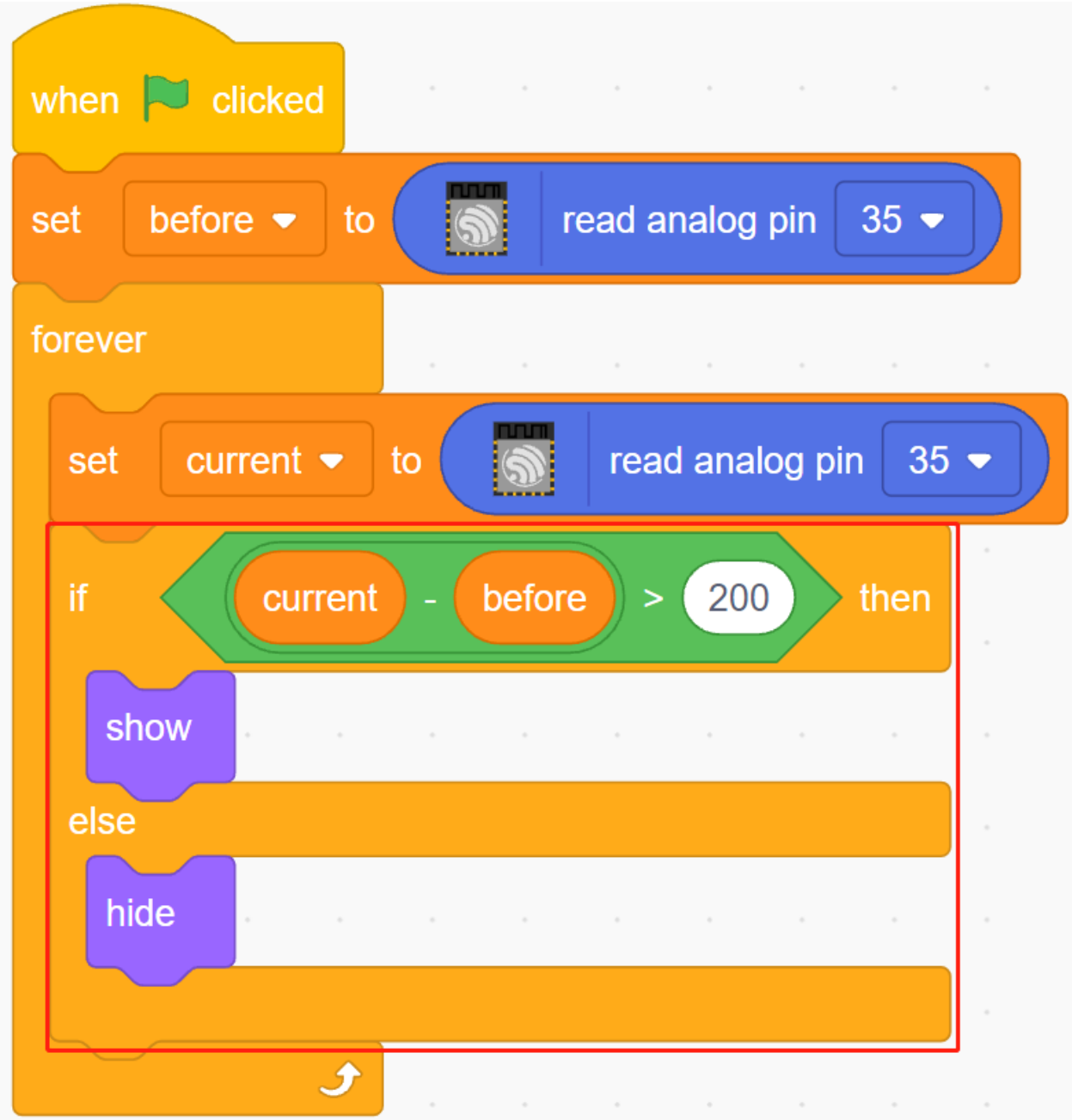
En [siempre], lee nuevamente el valor del pin35 y almacénalo en la variable **actual**.



5. Determinar los cambios de temperatura

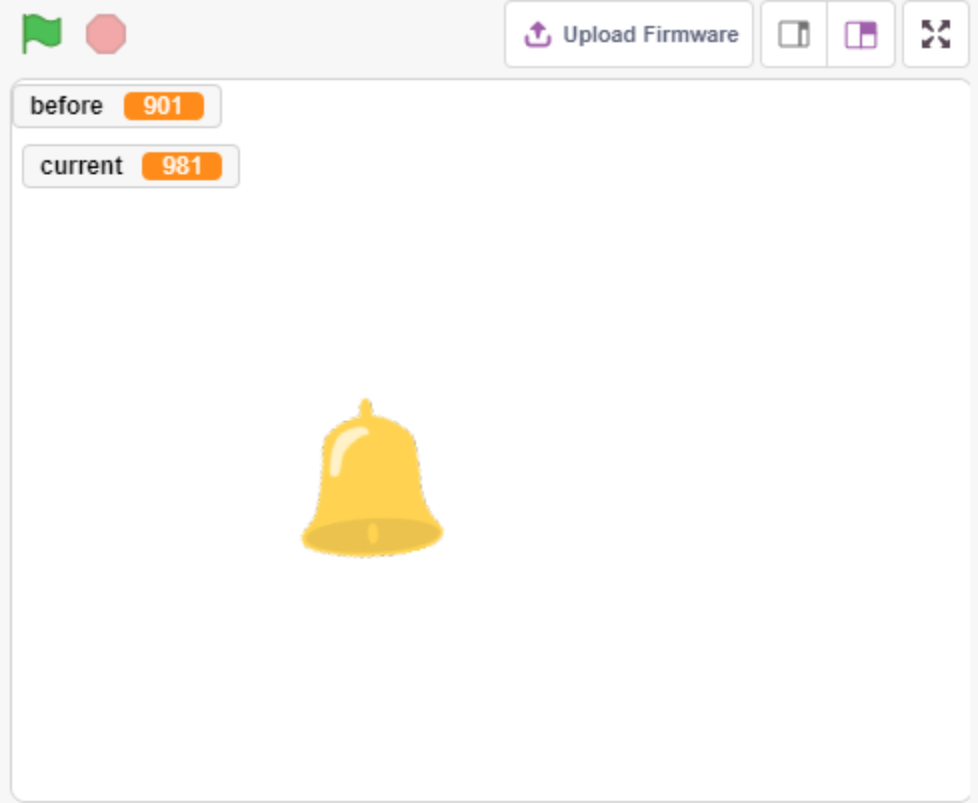
Usando el bloque [si sino], determina si el valor actual del pin35 es 200 mayor que el anterior, lo que representa una disminución de la temperatura. En este punto, deja que el sprite **Copito de Nieve** se muestre, de lo contrario, ocúltalo.

- [-] y [>]: operadores de sustracción y comparación de la paleta **Operadores**.


when clicked
set before to read analog pin 35
forever
set current to read analog pin 35
if current - before > 200 then
show
else
hide

5.10 2.7 Reloj Despertador Luminoso

En la vida, existen varios tipos de relojes despertadores. Ahora, vamos a crear un reloj despertador controlado por la luz. Cuando llega la mañana, la intensidad de la luz aumenta y este reloj despertador controlado por la luz te recordará que es hora de levantarte.



5.10.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Fotorresistor	

5.10.2 Lo Que Aprenderás

- Principio de funcionamiento de la fotorresistencia
- Cómo detener la reproducción de sonido y la ejecución de scripts

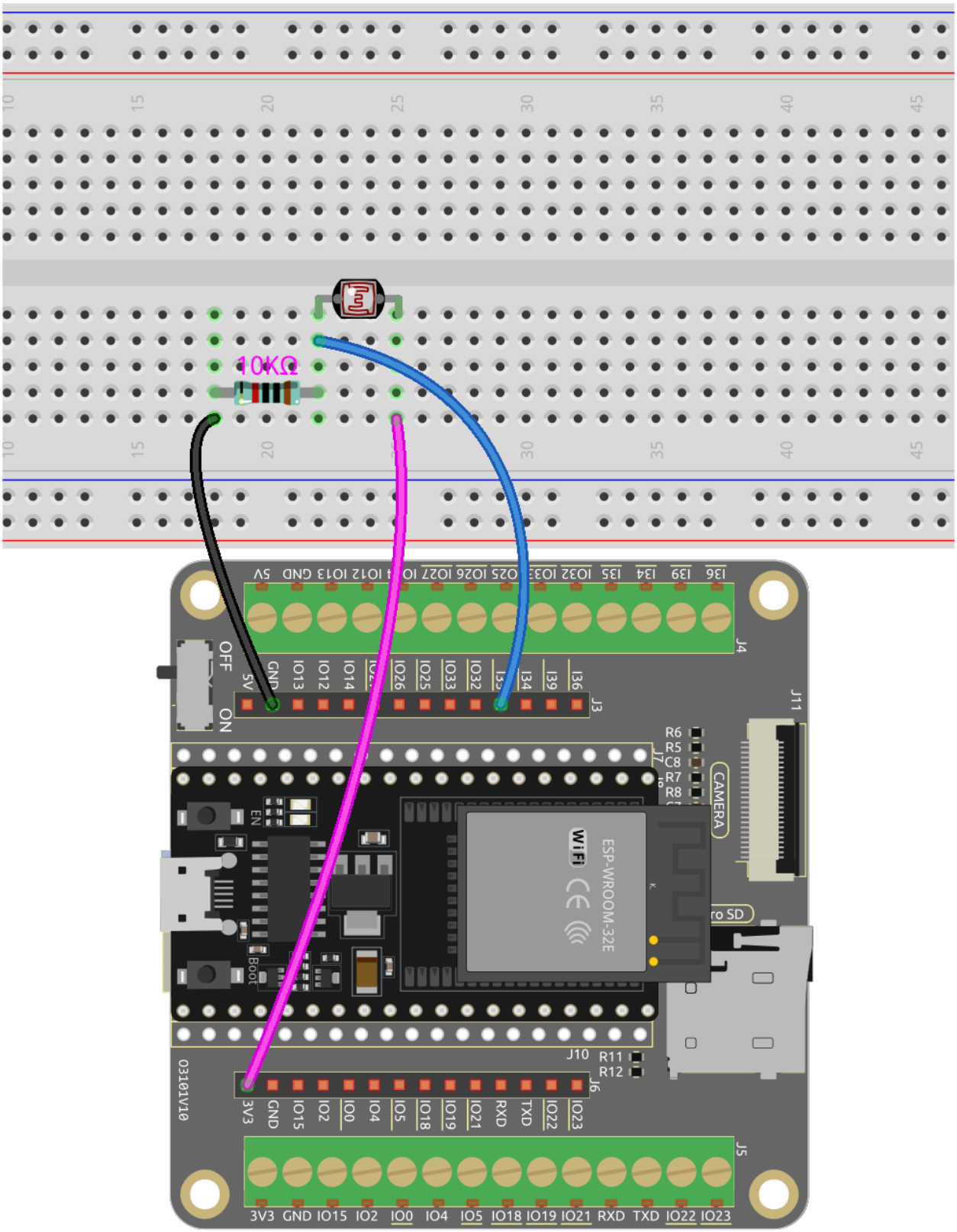
5.10.3 Construye el Circuito

Una fotorresistencia o célula fotoeléctrica es una resistencia variable controlada por la luz. La resistencia de una fotorresistencia disminuye con el aumento de la intensidad de luz incidente.

Construye el circuito según el siguiente diagrama.

Conecta un extremo de la fotorresistencia a 5V, el otro extremo al pin35, y conecta una resistencia de 10K en serie con GND en este extremo.

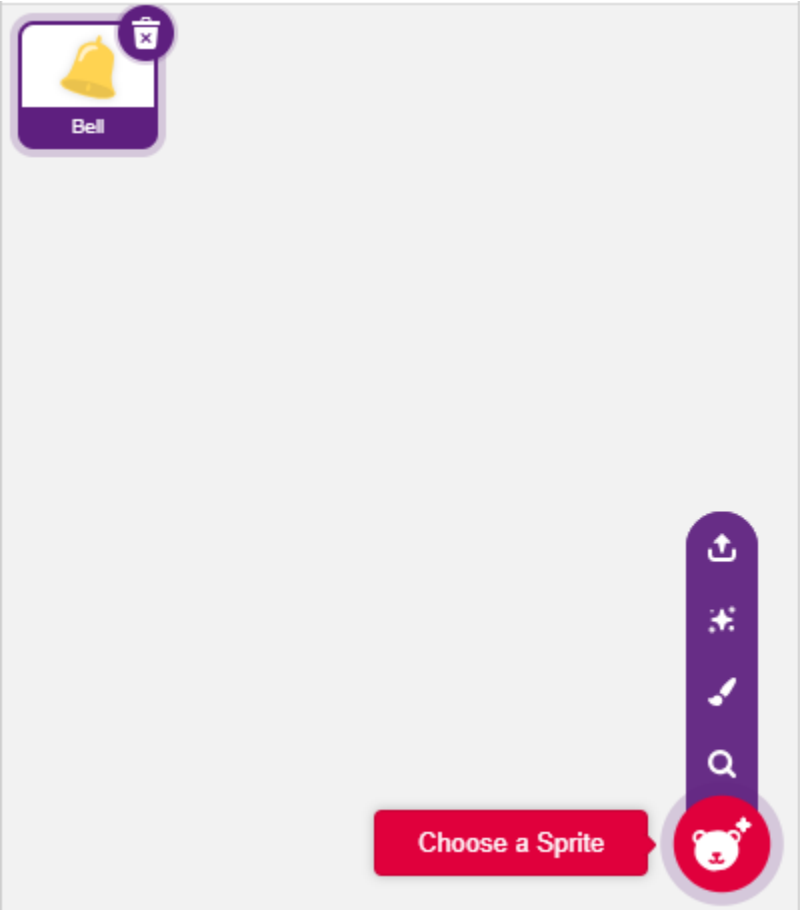
Así que cuando la intensidad de la luz aumenta, la resistencia de la fotorresistencia disminuye, la división de voltaje de la resistencia de 10K aumenta, y el valor obtenido por el pin35 se hace mayor.


10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

5.10.4 Programación

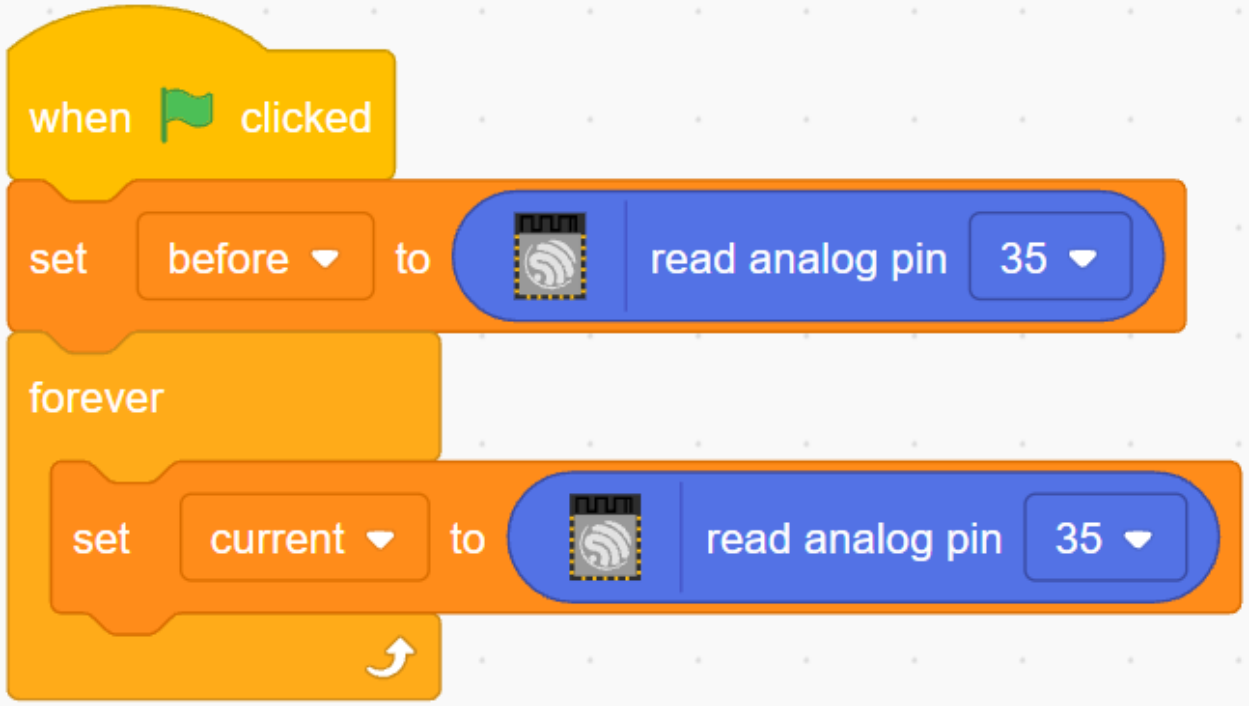
1. Selecciona un sprite

Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprites, introduce **campana** en la caja de búsqueda y luego haz clic para añadirlo.



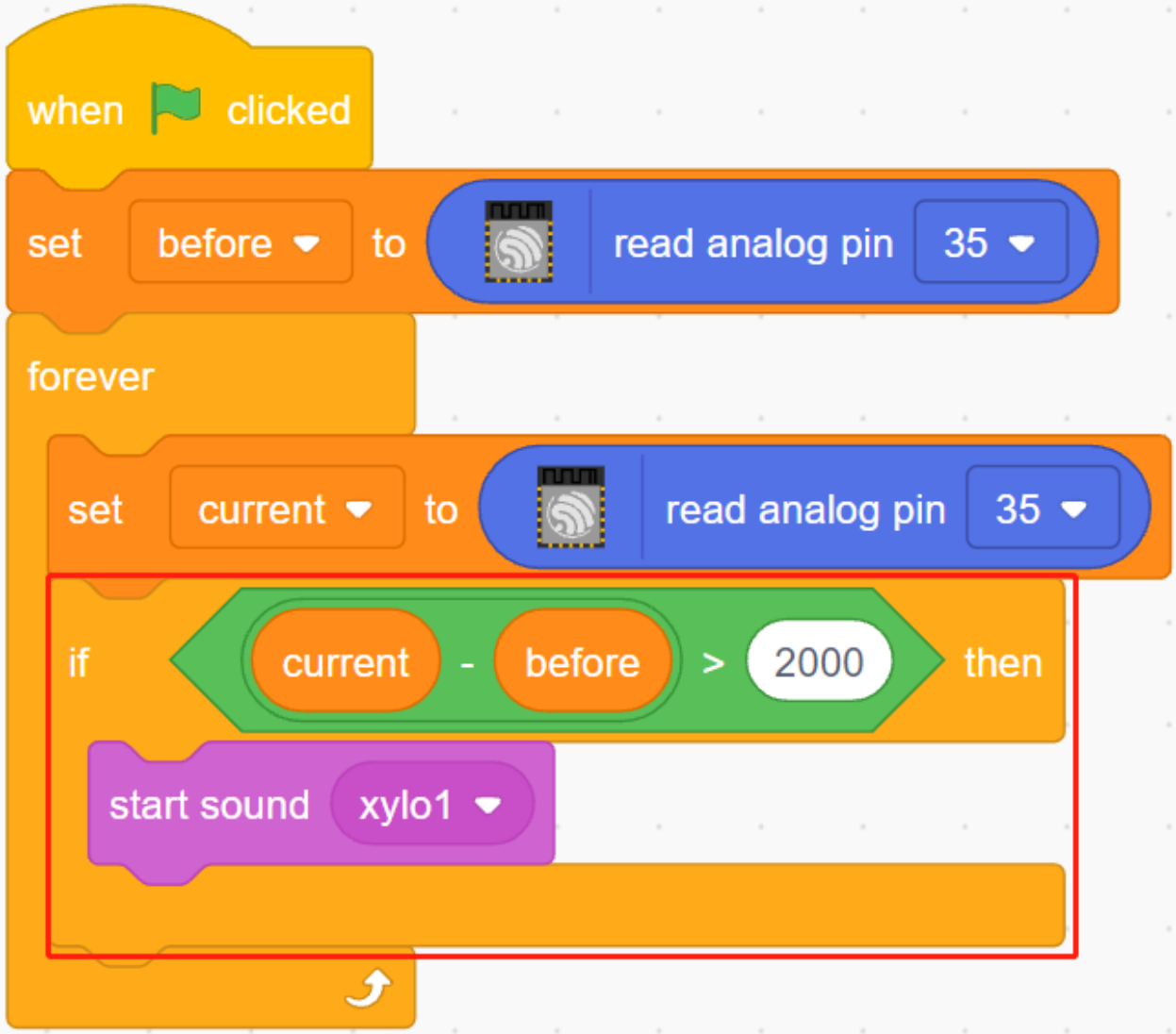
2. Lee el valor del pin35

Crea dos variables **anterior** y **actual**. Al hacer clic en la bandera verde, lee el valor del pin35 y almacénalo en la variable **anterior** como valor de referencia. En [siempre], lee nuevamente el valor del pin35, almacénalo en la variable **actual**.



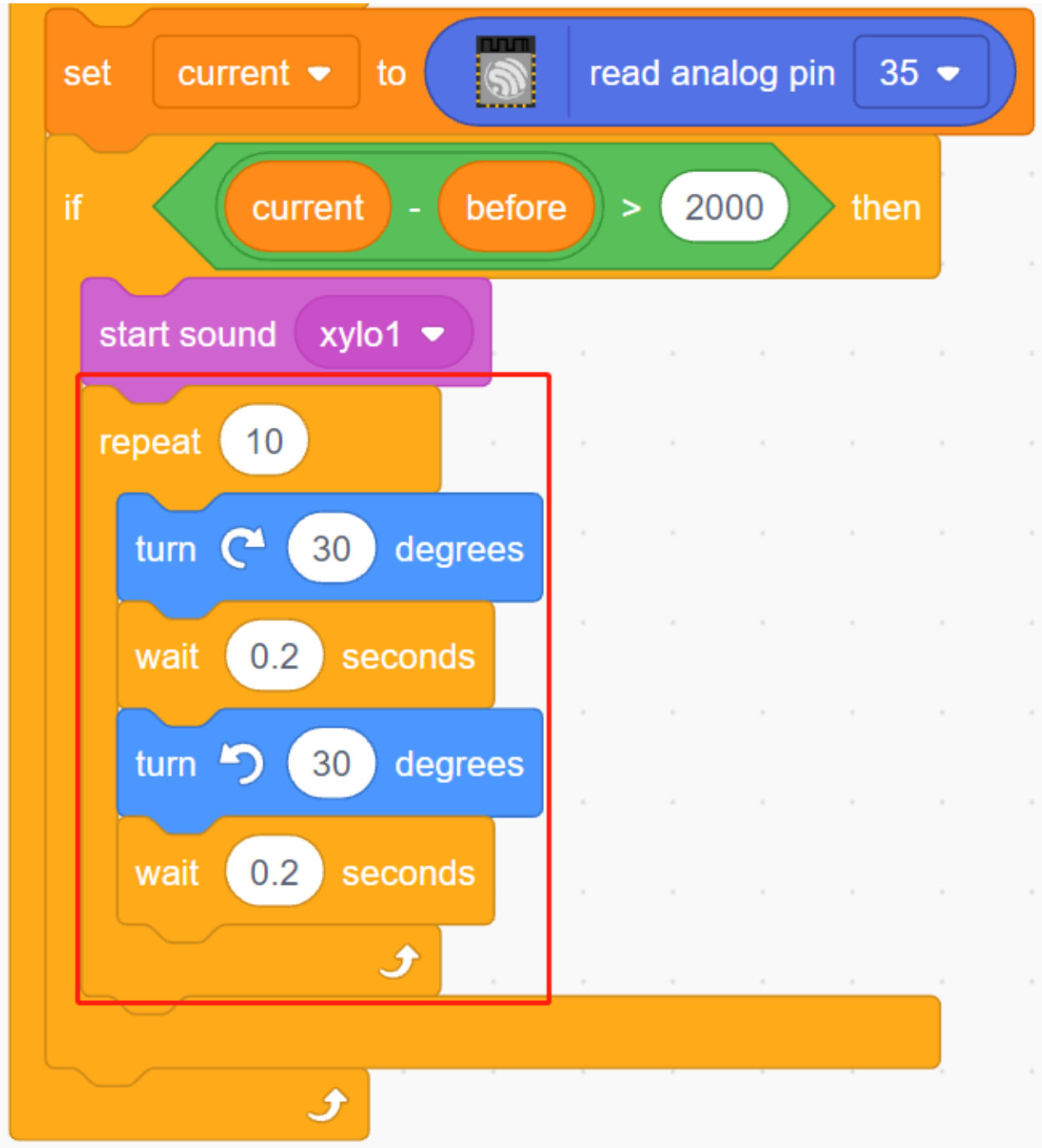
3. Haz un sonido

Cuando el valor del pin35 actual es mayor que el anterior en 50, lo que representa que la intensidad de la luz actual es mayor que el umbral, entonces haz que el sprite emita un sonido.



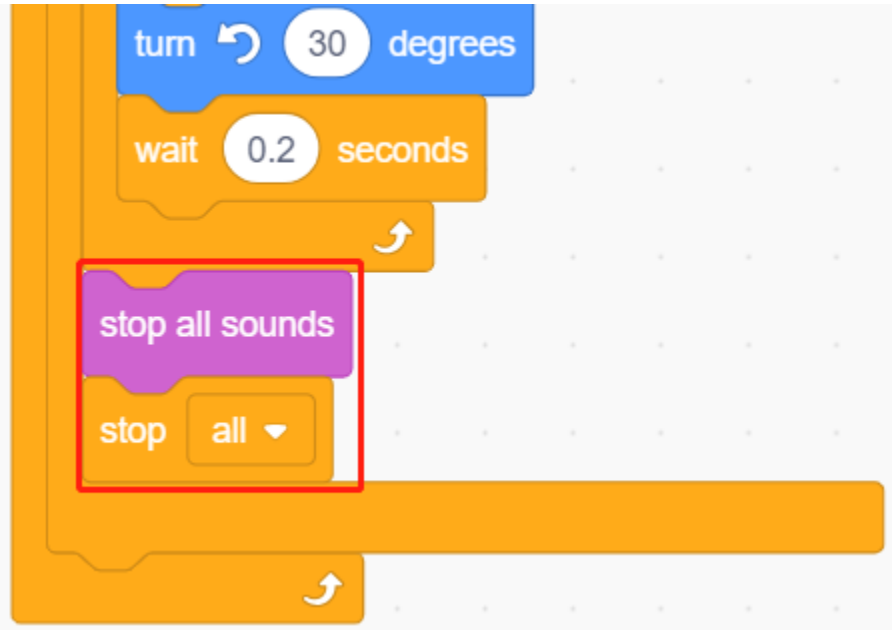
4. Girando el sprite

Usa [bloque de giro] para hacer que el sprite **campana** gire hacia la izquierda y hacia la derecha para lograr el efecto de alarma.



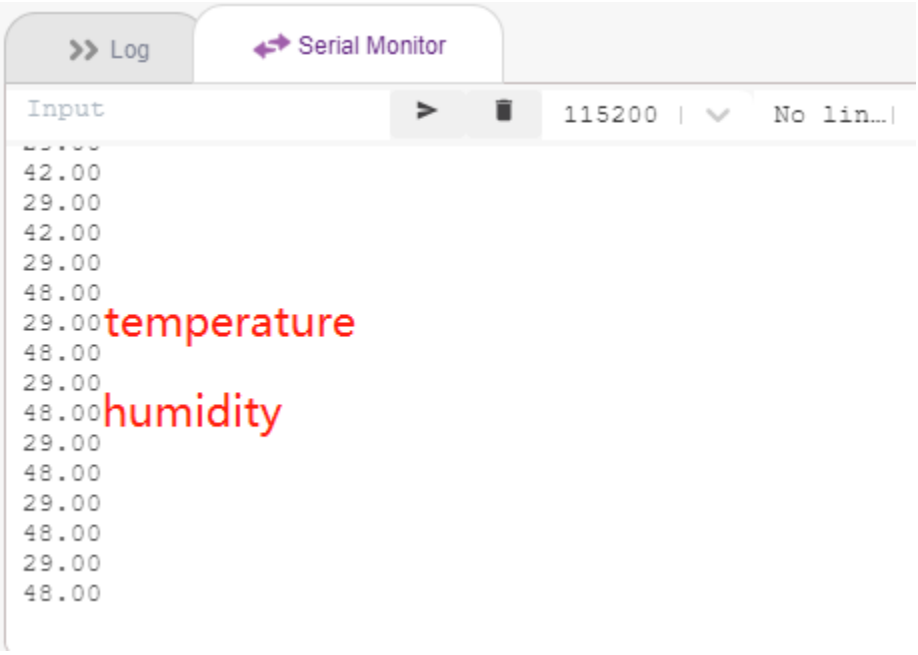
5. detener todo

Detiene la alarma después de que ha estado sonando por un tiempo.



5.11 2.8 Leer Temperatura y Humedad

Los proyectos anteriores han utilizado el modo escenario, pero algunas funciones solo están disponibles en el modo de subida, como la función de comunicación serial. En este proyecto, imprimiremos la temperatura y la humedad del DHT11 utilizando el Monitor Serial en *Modo de Subida*.



5.11.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Sensor de Humedad y Temperatura DHT11	

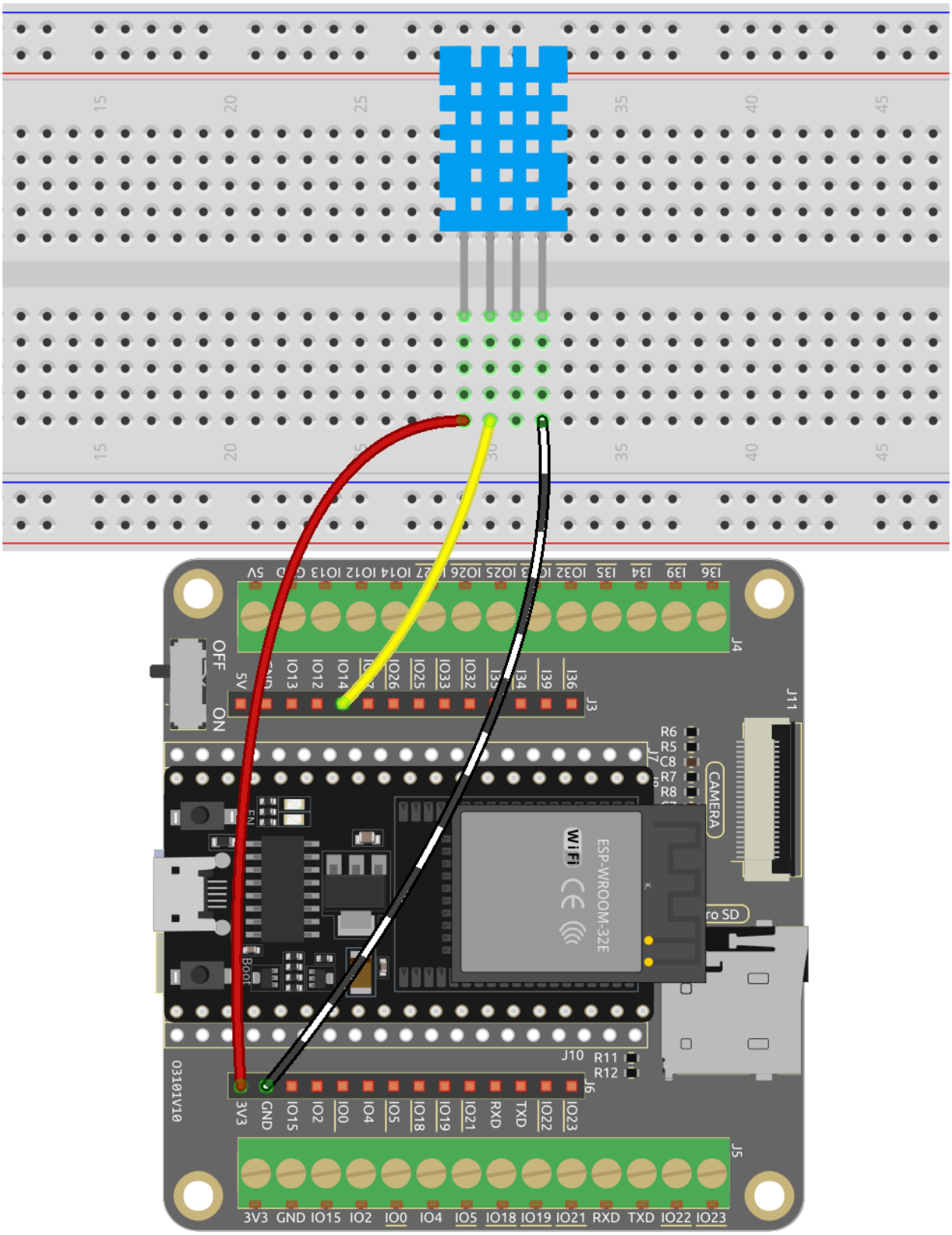
5.11.2 Lo Que Aprenderás

- Obtener la temperatura y la humedad del módulo DHT11
- Monitor Serial para *Modo de Subida*
- Añadir extensión

5.11.3 Construye el Circuito

El sensor digital de temperatura y humedad DHT11 es un sensor compuesto que contiene una salida de señal digital calibrada de temperatura y humedad.

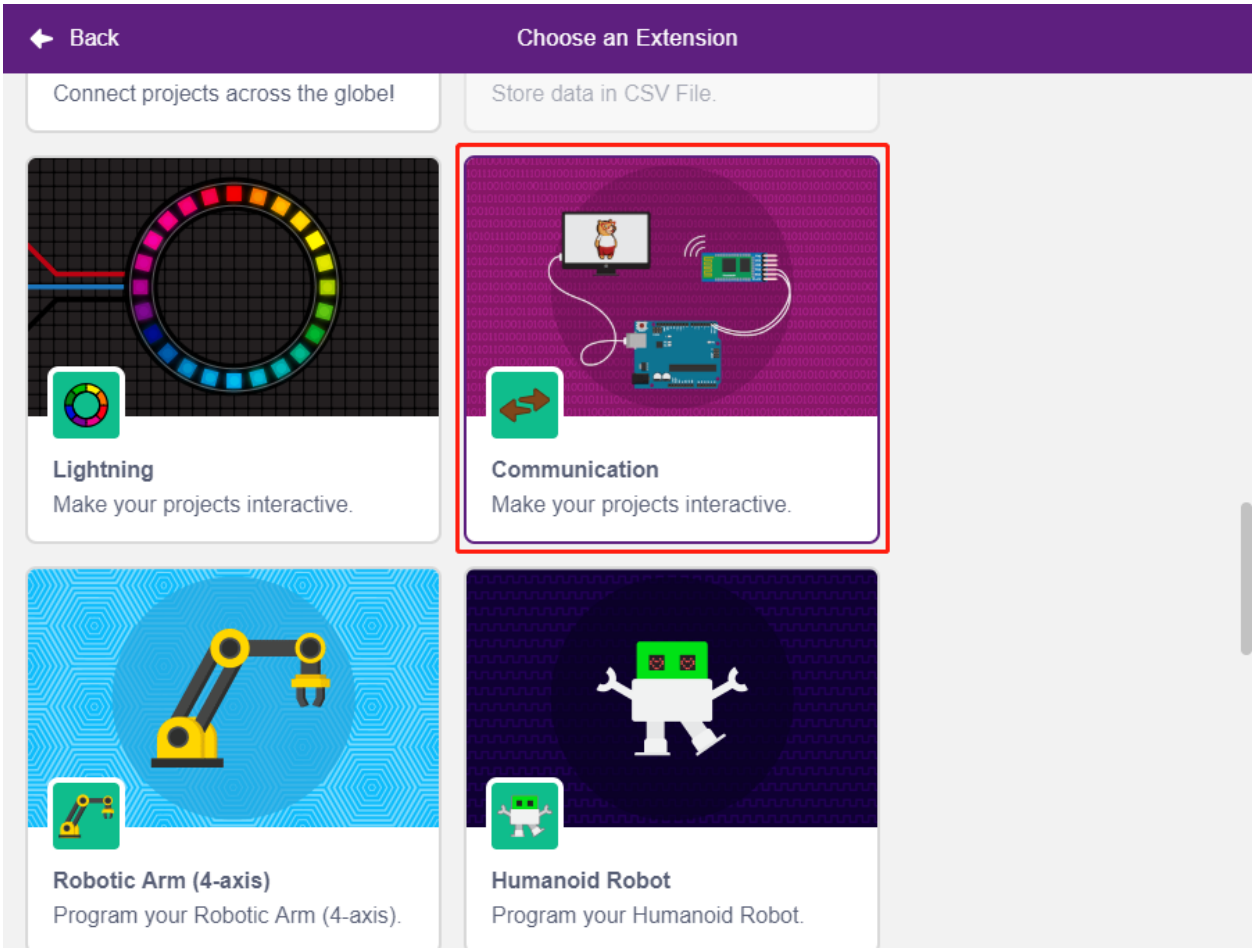
Ahora construye el circuito según el siguiente diagrama.


OFF
ON
J4
J3
J11
J10
J6
J5
R6
R5
C8
R7
R8
R11
R12
CAMERA
ESP-WROOM-32E
Boot
EN
03101V10
5V
GND
IO13
IO12
IO14
IO26
IO25
IO33
IO32
IO35
IO34
IO39
IO36
3V3
GND
IO15
IO2
IO0
IO4
IO5
IO18
IO19
IO21
RXD
TXD
IO22
IO23

5.11.4 Programación

1. Añadiendo Extensiones

Cambia a **Modo de Subida**, haz clic en el botón **Añadir Extensión** en la esquina inferior izquierda, luego selecciona **Comunicación** para añadirla, y aparecerá al final del área de la paleta.



2. Inicializando el ESP32 y el Monitor Serial

En **Modo de Subida**, inicia el ESP32 y luego establece la tasa de baudios del puerto serie.

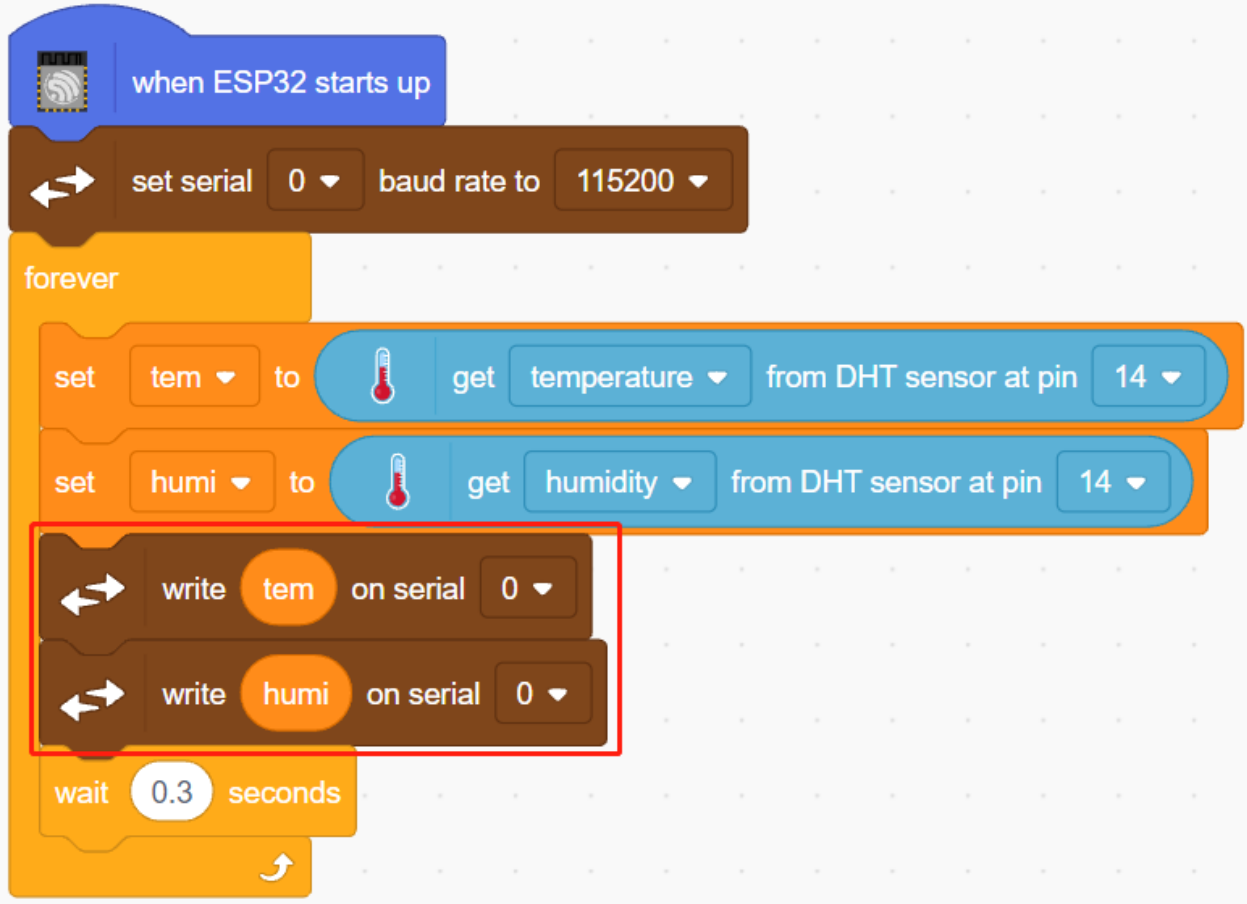
- [cuando ESP32 se inicia]: En **Modo de Subida**, inicia el ESP32.
- [establecer tasa de baudios del serial a]: Desde la paleta **Comunicaciones**, se utiliza para establecer la tasa de baudios del puerto serie 0, el predeterminado es 115200. Si estás usando Mega2560, entonces puedes elegir establecer la tasa de baudios en el puerto serie 0~2.

3. Leer temperatura y humedad

Crea 2 variables **tem** y **humi** para almacenar la temperatura y la humedad respectivamente, el código aparecerá en el lado derecho mientras arrastras y sueltas el bloque.

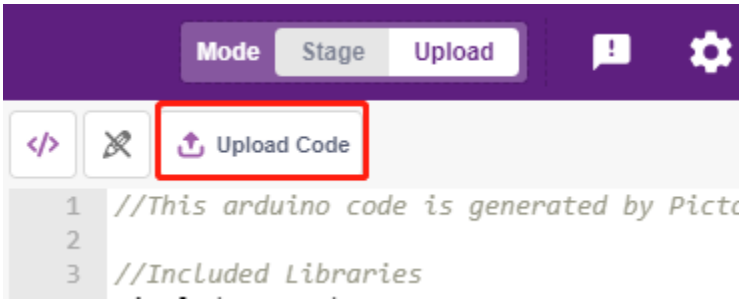
4. Imprimirlas en el Monitor Serial

Escribe la temperatura y la humedad leídas en el Monitor Serial. Para evitar transferir demasiado rápido y causar que PictoBlox se atasque, usa el bloque [esperar segundos], para añadir algún intervalo de tiempo para la próxima impresión.



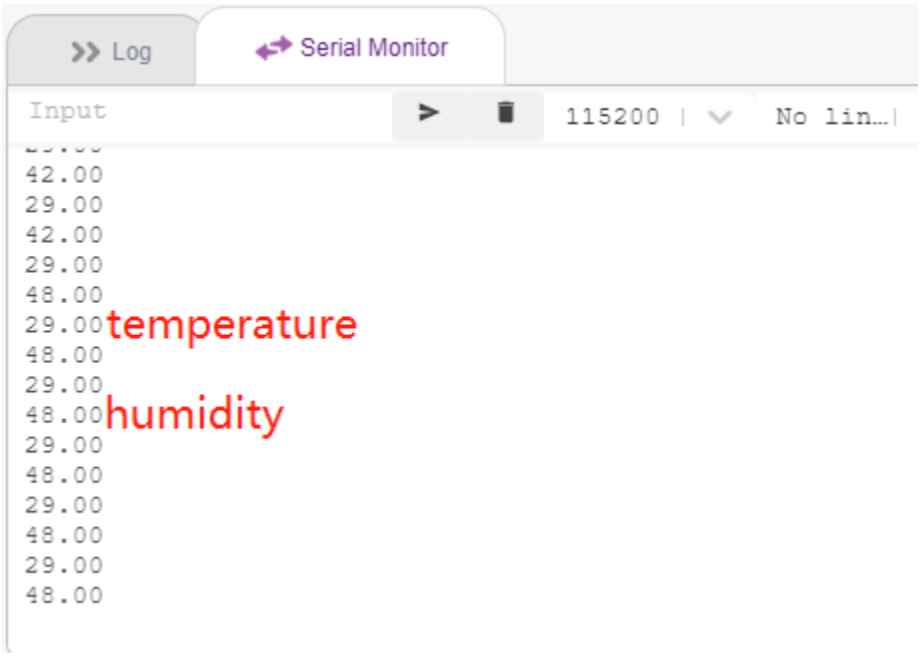
5. Subiendo el código

A diferencia del modo **Escenario**, el código en **Modo de Subida** necesita ser subido a la placa ESP32 usando el botón **Subir Código** para ver el efecto. Esto también te permite desconectar el cable USB y aún tener el programa ejecutándose.



6. Encender el monitor serial

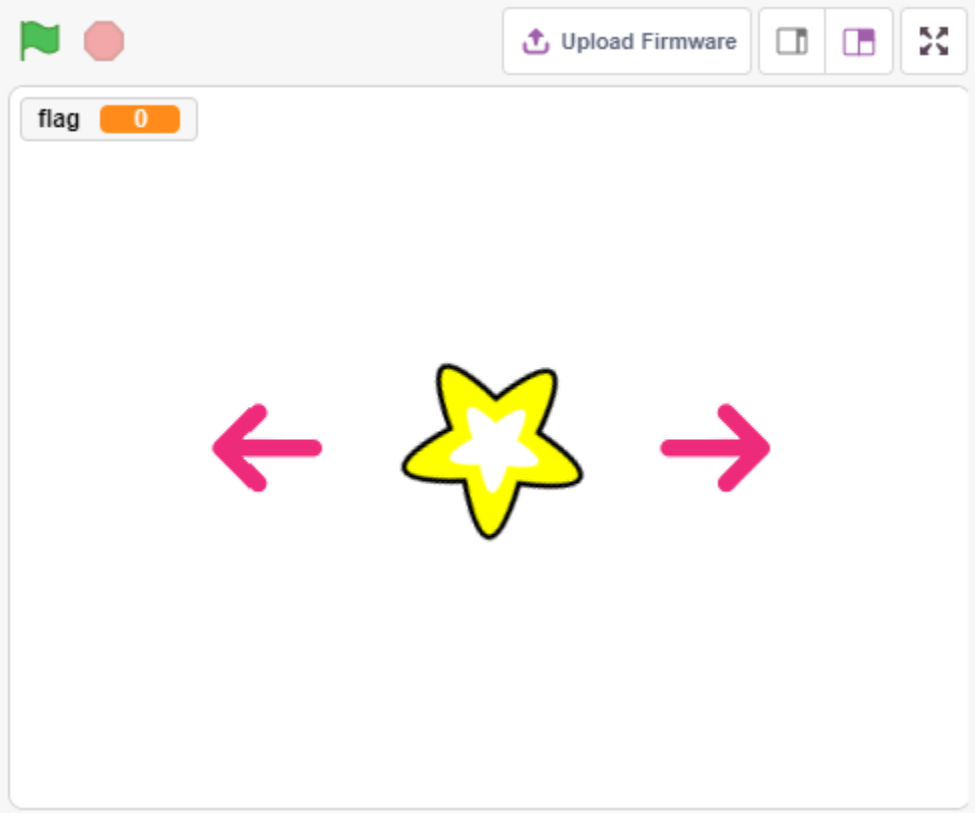
Ahora abre el **Monitor Serial** para ver la temperatura y la humedad.



5.12 2.9 Ventilador Rotativo

En este proyecto, haremos una estrella giratoria y un ventilador.

Al hacer clic en los sprites de flecha izquierda y derecha en el escenario, controlaremos la rotación en el sentido de las agujas del reloj y en sentido contrario del motor y del sprite de la estrella, haz clic en el sprite de la estrella para detener la rotación.



5.12.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

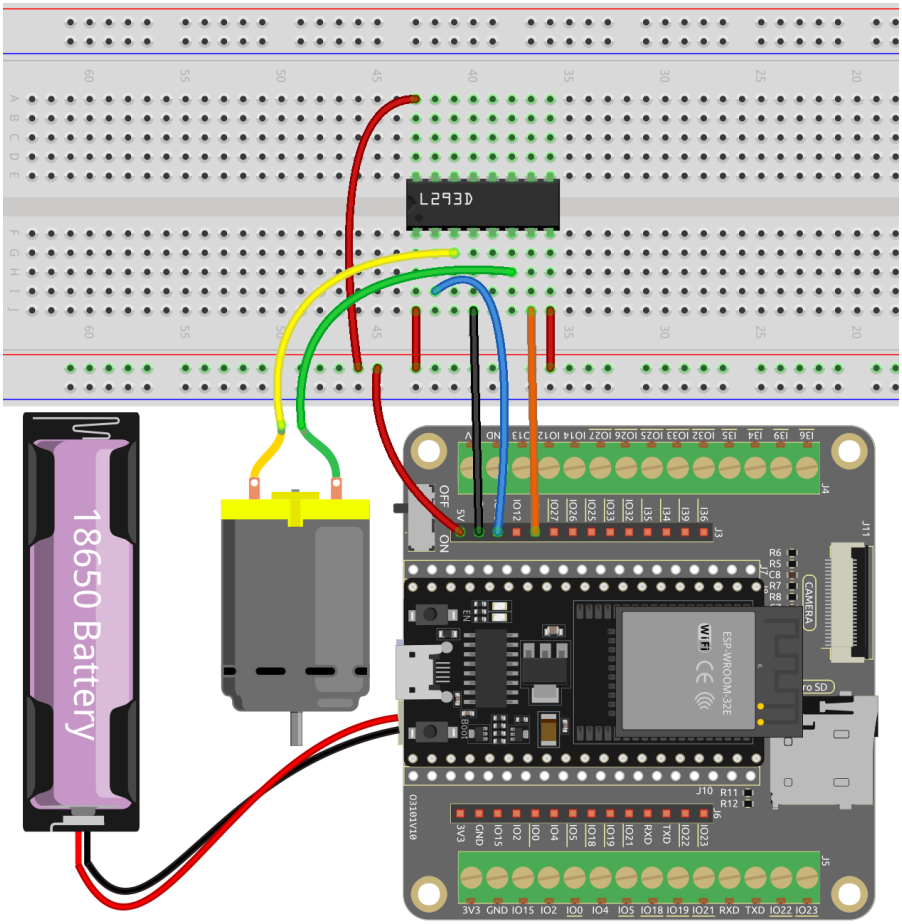
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Motor de Corriente Continua (DC)	
L293D	-

5.12.2 Lo Que Aprenderás

- Principio de funcionamiento del motor
- Función de difusión
- Detener otros scripts en el bloque de sprite

5.12.3 Construye el Circuito

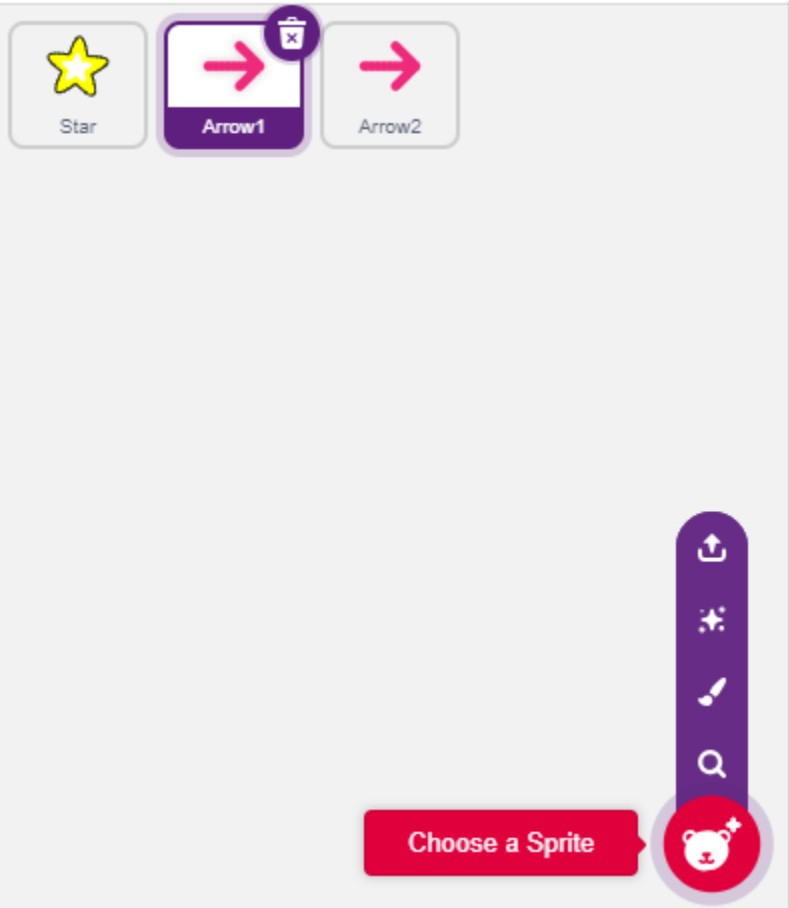


5.12.4 Programación

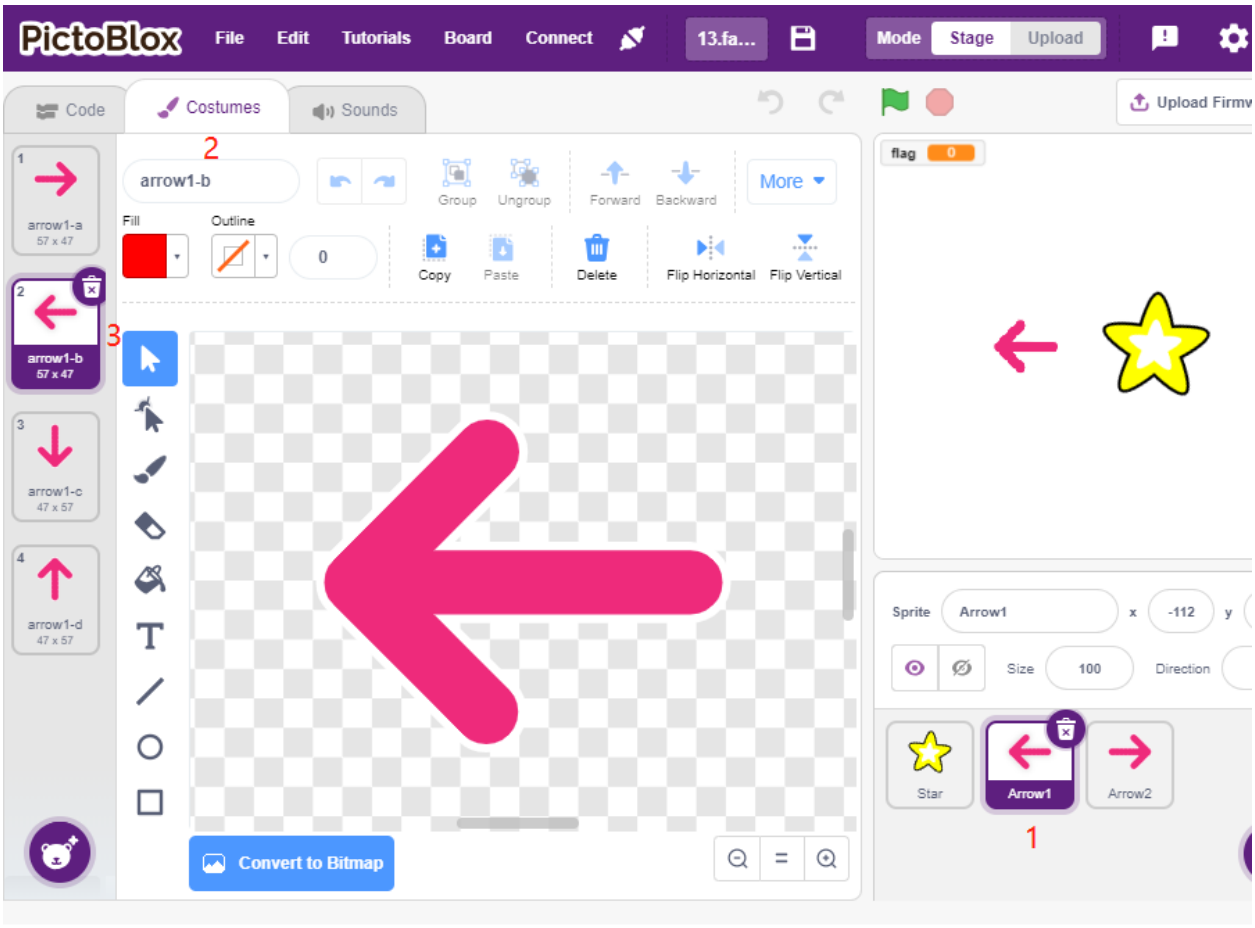
El efecto que queremos lograr es usar 2 sprites de flecha para controlar la rotación en el sentido de las agujas del reloj y en sentido contrario del motor y del sprite de la estrella respectivamente, haciendo clic en el sprite de la estrella se detendrá la rotación del motor.

1. Añadir sprites

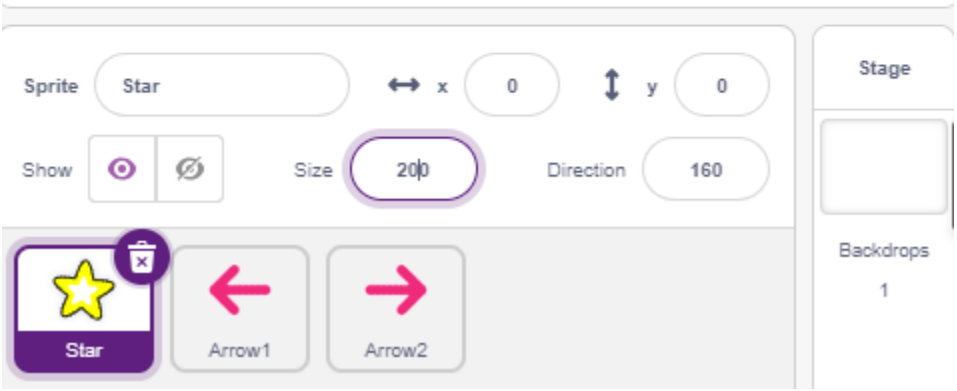
Elimina el sprite predeterminado, luego selecciona el sprite **Estrella** y el sprite **Flecha1**, y copia **Flecha1** una vez.



En la opción **Disfraces**, cambia el sprite **Flecha1** a un disfraz de dirección diferente.



Ajusta el tamaño y la posición del sprite adecuadamente.

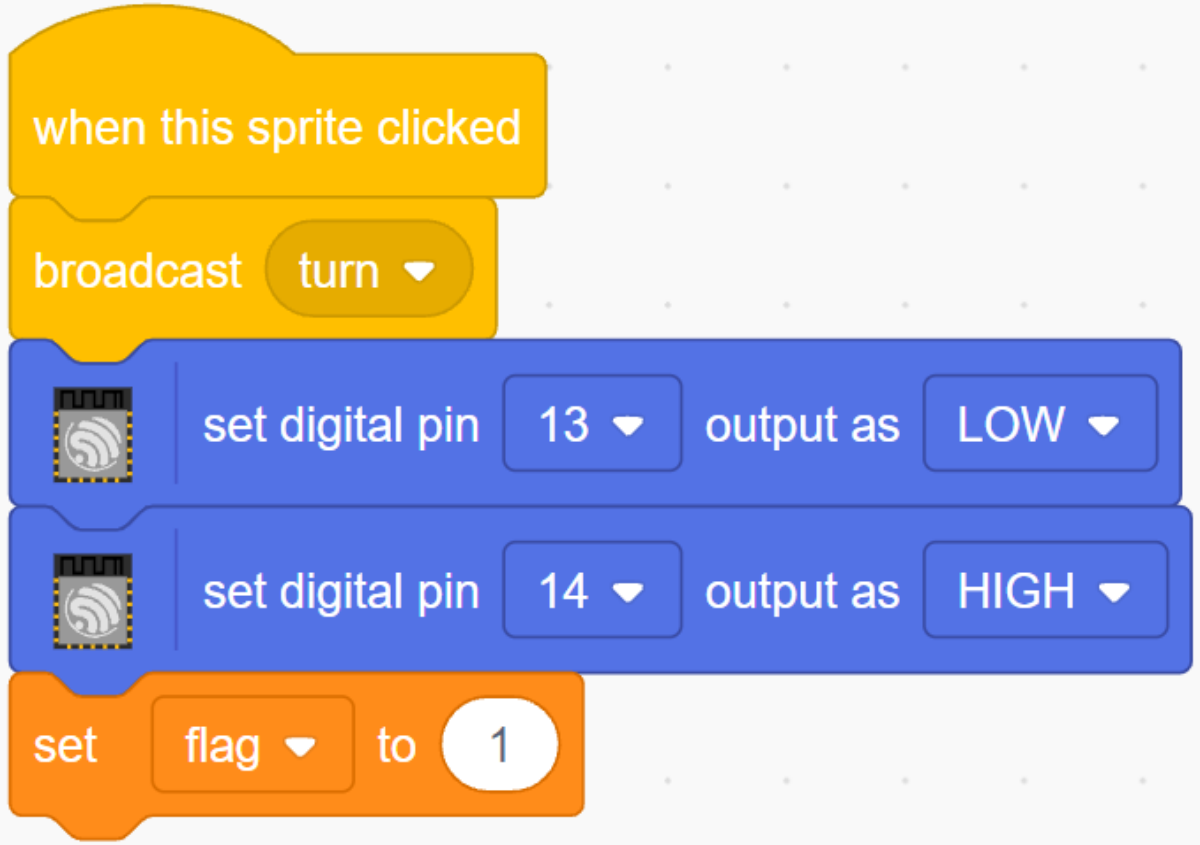


2. Sprite de flecha izquierda

Cuando se hace clic en este sprite, transmite un mensaje - girar, luego establece el pin digital12 en bajo y el pin14 en alto, y establece la variable **flag** en 1. Si haces clic en el sprite de flecha izquierda, encontrarás que el motor gira en sentido antihorario, si tu giro es en sentido horario, entonces intercambia las posiciones de pin12 y pin14.

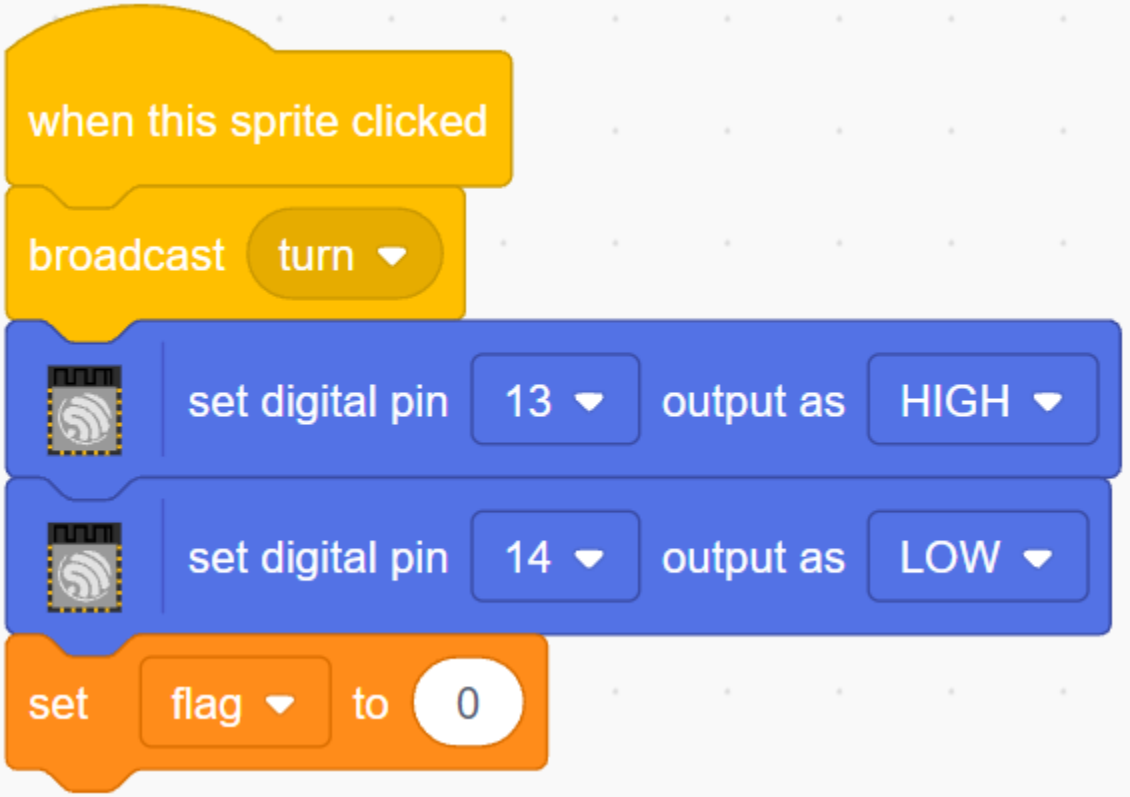
Hay 2 puntos a tener en cuenta aquí.

- [difundir]: de la paleta **Eventos**, se utiliza para transmitir un mensaje a los otros sprites, cuando los otros sprites reciben este mensaje, realizará un evento específico. Por ejemplo, aquí es **girar**, cuando el sprite **estrella** recibe este mensaje, ejecuta el script de rotación.
- variable flag: La dirección de rotación del sprite estrella está determinada por el valor de flag. Por lo tanto, cuando crees la variable **flag**, necesitas hacer que se aplique a todos los sprites.



3. Sprite de flecha derecha

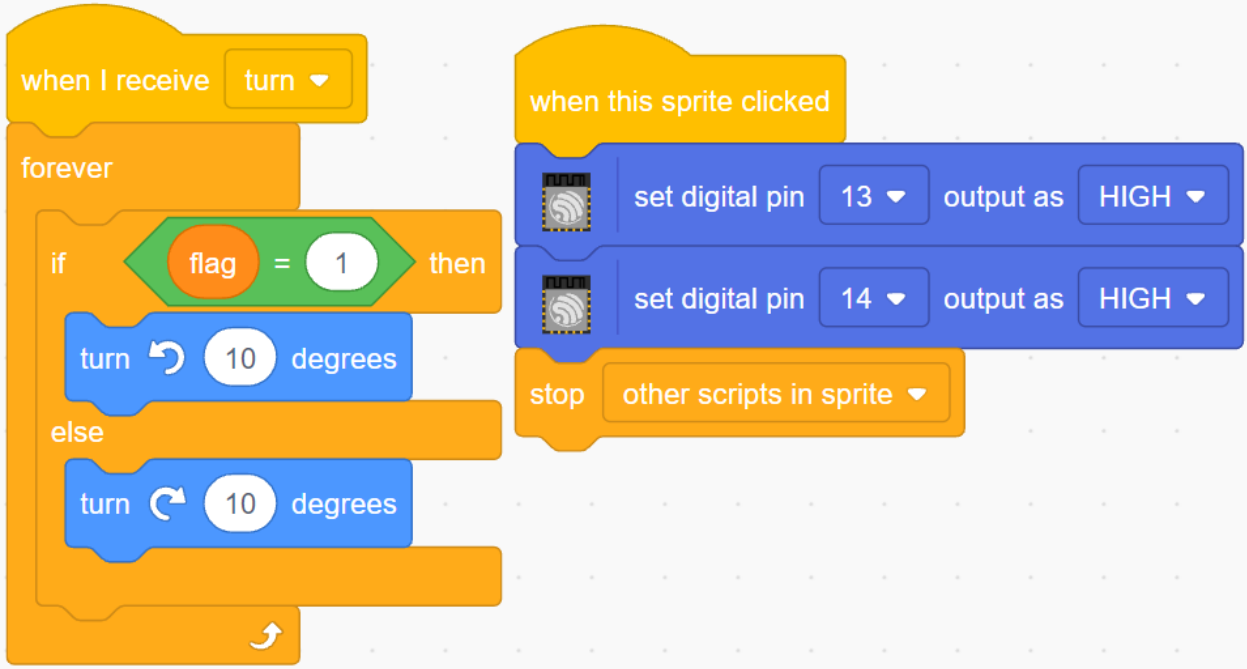
Cuando se hace clic en este sprite, transmite un mensaje girar, luego establece el pin digital12 en alto y el pin14 en bajo para hacer que el motor gire en sentido horario y establece la variable **flag** en 0.



4. Sprite de estrella

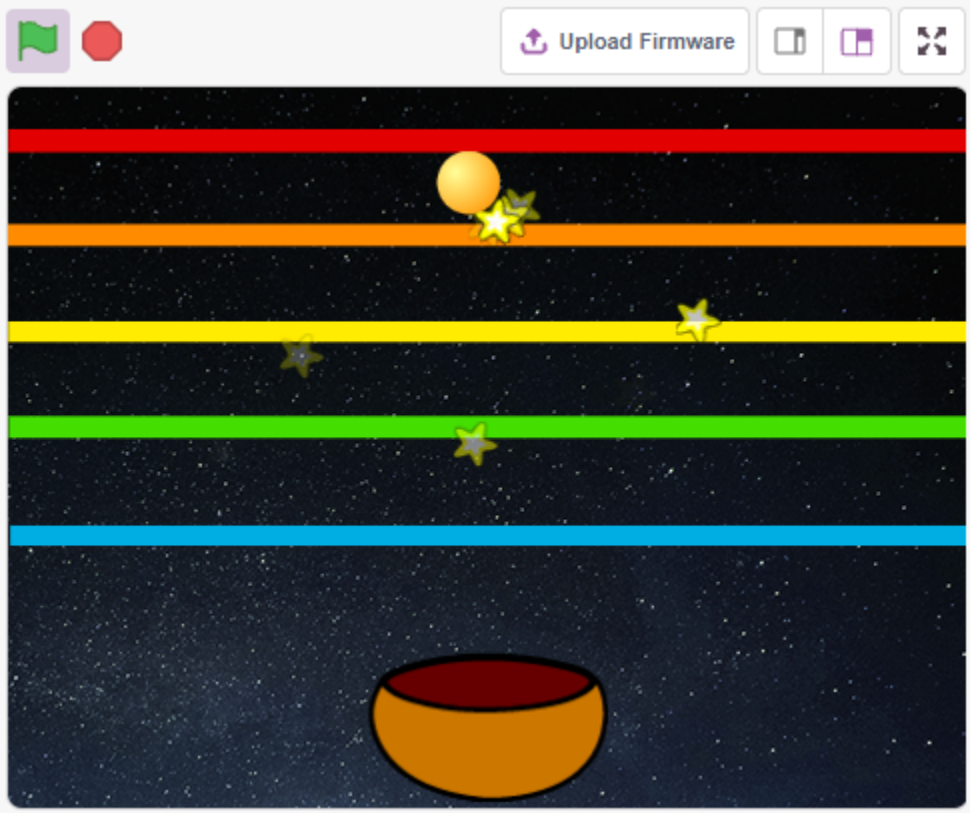
Aquí se incluyen 2 eventos.

- Cuando el sprite **estrella** recibe el mensaje transmitido girar, determina el valor de flag; si flag es 1, gira 10 grados a la izquierda, de lo contrario, se invierte. Dado que está en [SIEMPRE], seguirá girando.
- Cuando se hace clic en este sprite, se establecen ambos pines del motor en alto para hacer que deje de girar y detener los otros scripts en este sprite.



5.13 2.10 Bola Sensible a la Luz

En este proyecto, usaremos una fotorresistencia para hacer que la bola en el escenario vuele hacia arriba. Coloca tu mano sobre la fotorresistencia para controlar la intensidad de luz que recibe. Cuanto más cerca esté tu mano de la fotorresistencia, menor será su valor y más alto volará la bola en el escenario, de lo contrario caerá. Cuando la bola toca la cuerda, produce un sonido agradable así como un parpadeo de estrellas.



5.13.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Fotorresistor	

5.13.2 Lo Que Aprenderás

- Rellenar el sprite con colores
- Toque entre los sprites

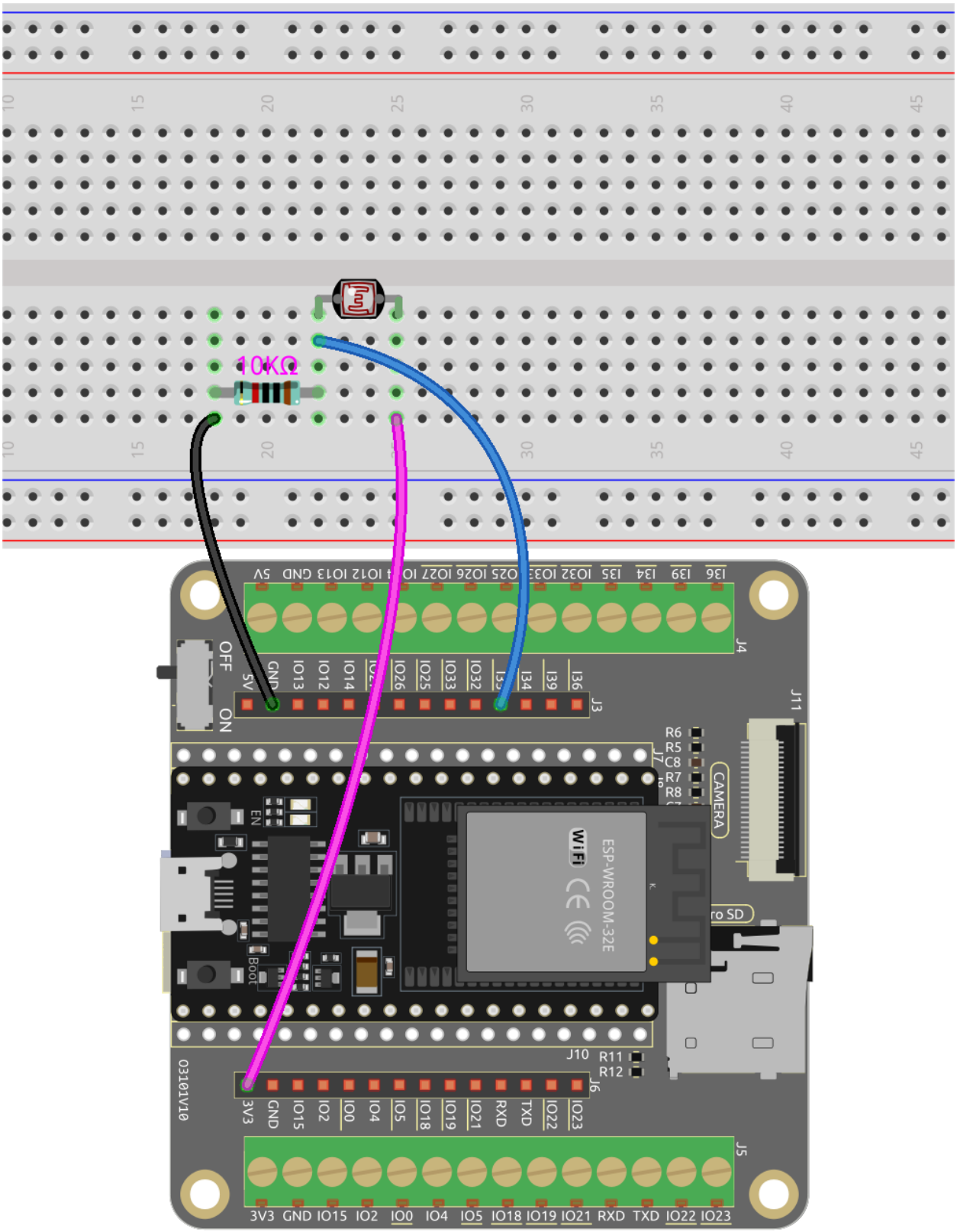
5.13.3 Construye el Circuito

Una fotorresistencia o célula fotoeléctrica es una resistencia variable controlada por la luz. La resistencia de una fotorresistencia disminuye con el aumento de la intensidad de luz incidente.

Construye el circuito según el siguiente diagrama.

Conecta un extremo de la fotorresistencia a 5V, el otro extremo al pin35, y conecta una resistencia de 10K en serie con GND en este extremo.

Así que cuando la intensidad de la luz aumenta, la resistencia de la fotorresistencia disminuye, la división de voltaje de la resistencia de 10K aumenta, y el valor obtenido por el pin35 se hace mayor.

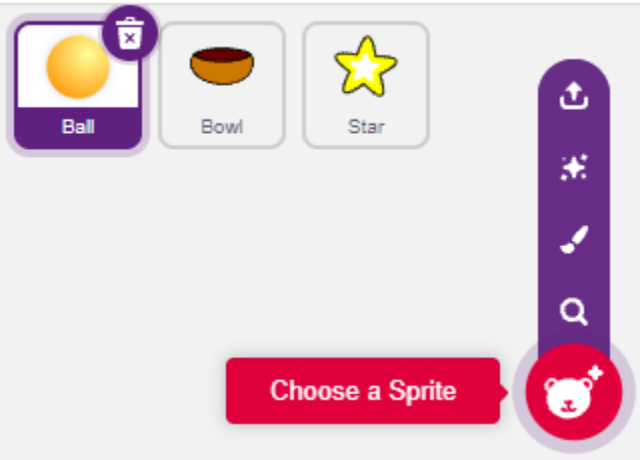

10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
Boot
EN
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

5.13.4 Programación

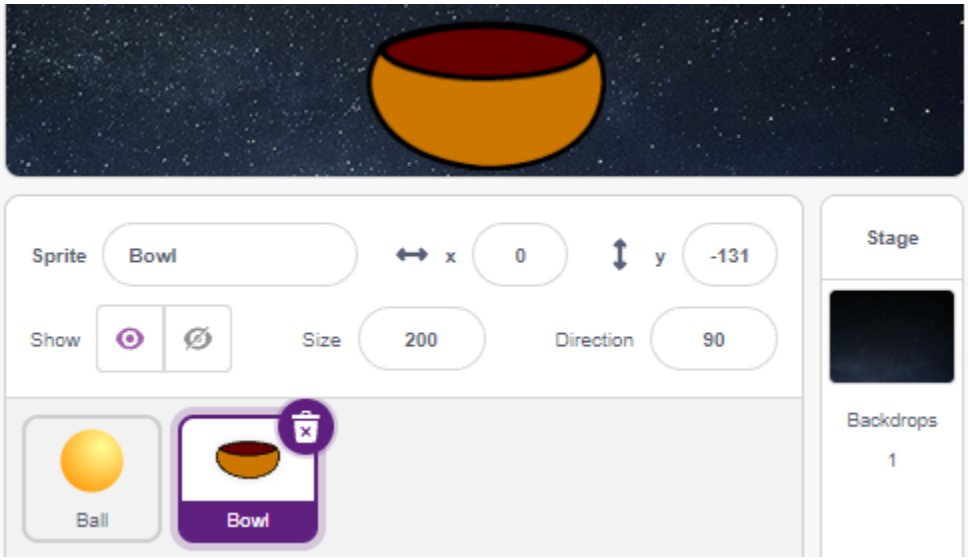
El efecto que queremos obtener es que cuanto más cerca esté tu mano de la fotorresistencia, el sprite de la bola en el escenario sigue subiendo, de lo contrario caerá sobre el sprite del tazón. Si toca el sprite de la Línea mientras sube o cae, emitirá un sonido musical y lanzará sprites de estrella en todas direcciones.

1. Seleccionar sprite y fondo

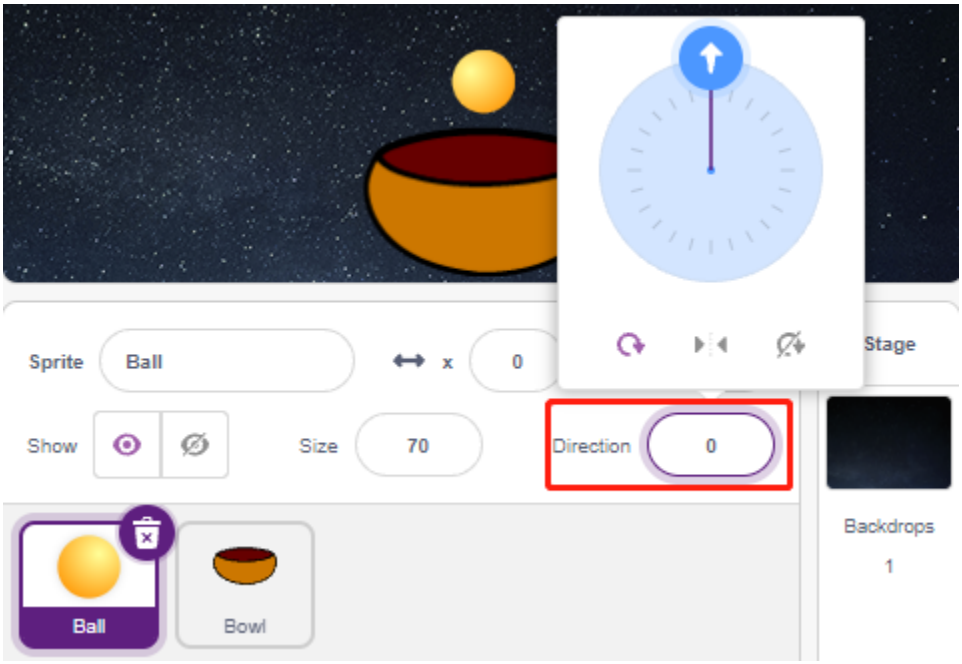
Elimina el sprite predeterminado, selecciona los sprites **Bola**, **Tazón** y **Estrella**.



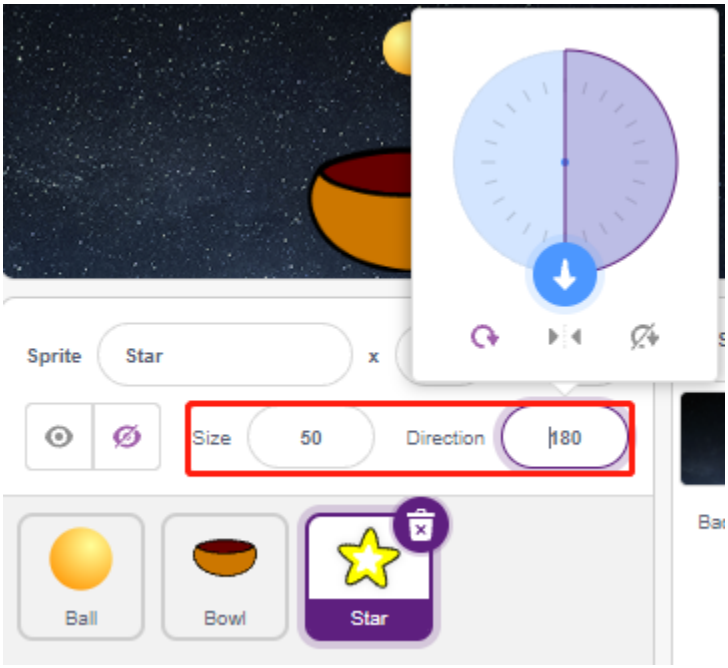
Mueve el sprite **Tazón** al centro inferior del escenario y aumenta su tamaño.



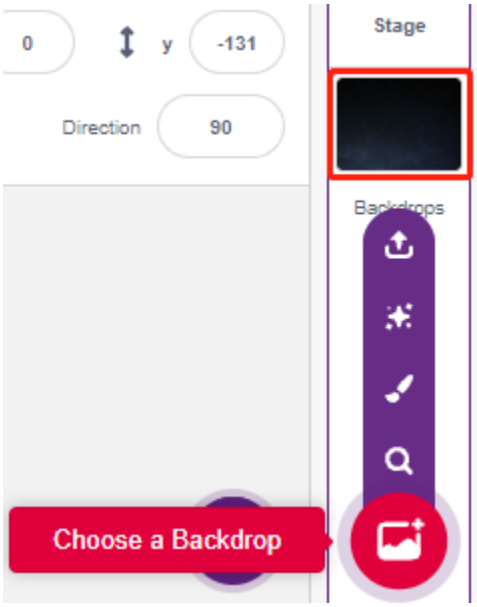
Como necesitamos moverlo hacia arriba, establece la dirección del sprite **Bola** a 0.



Establece el tamaño y la dirección del sprite **Estrella** a 180 porque necesitamos que caiga, o puedes cambiarlo a otro ángulo.

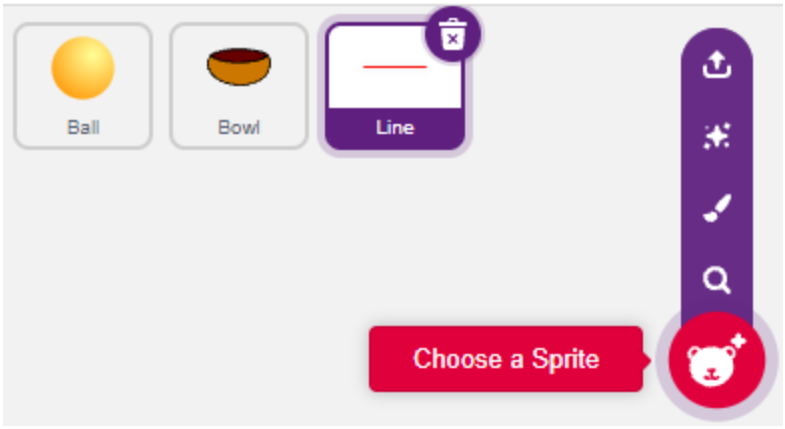


Ahora añade el fondo **Estrellas**.

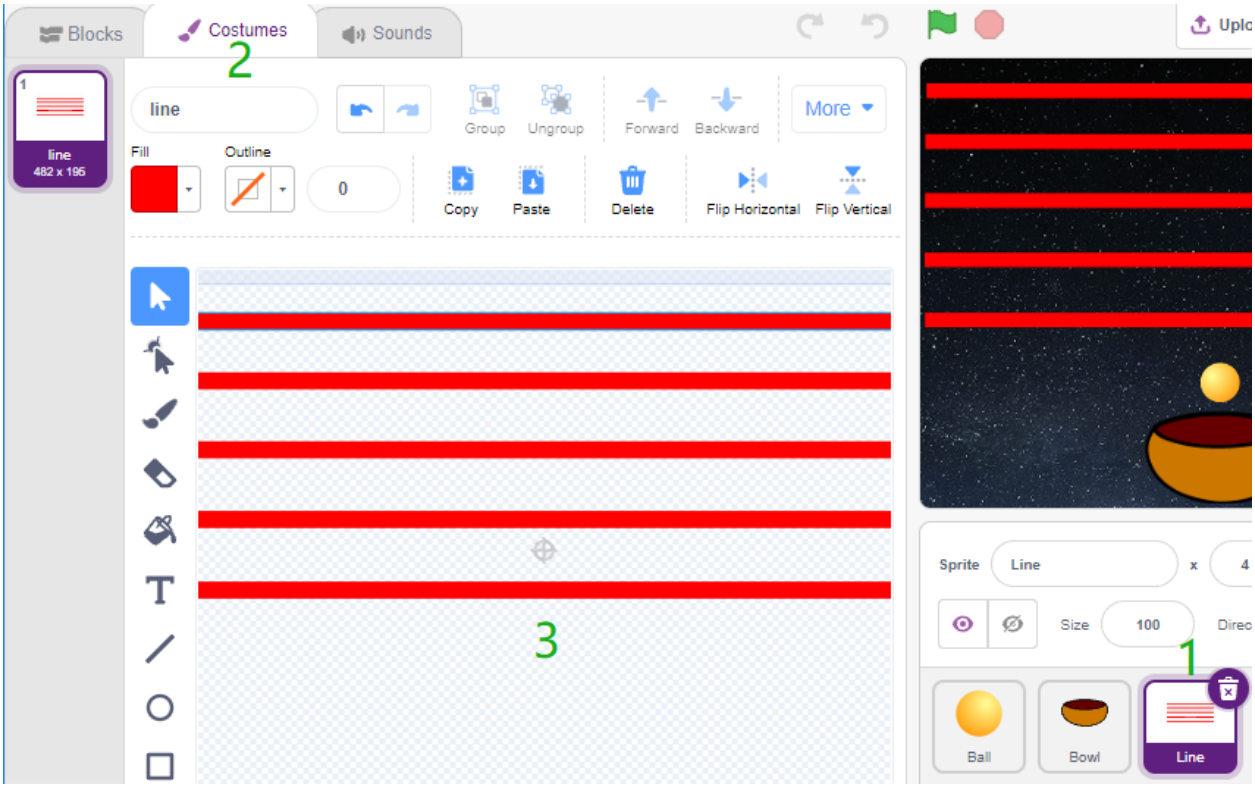


2. Dibujar un sprite de Línea

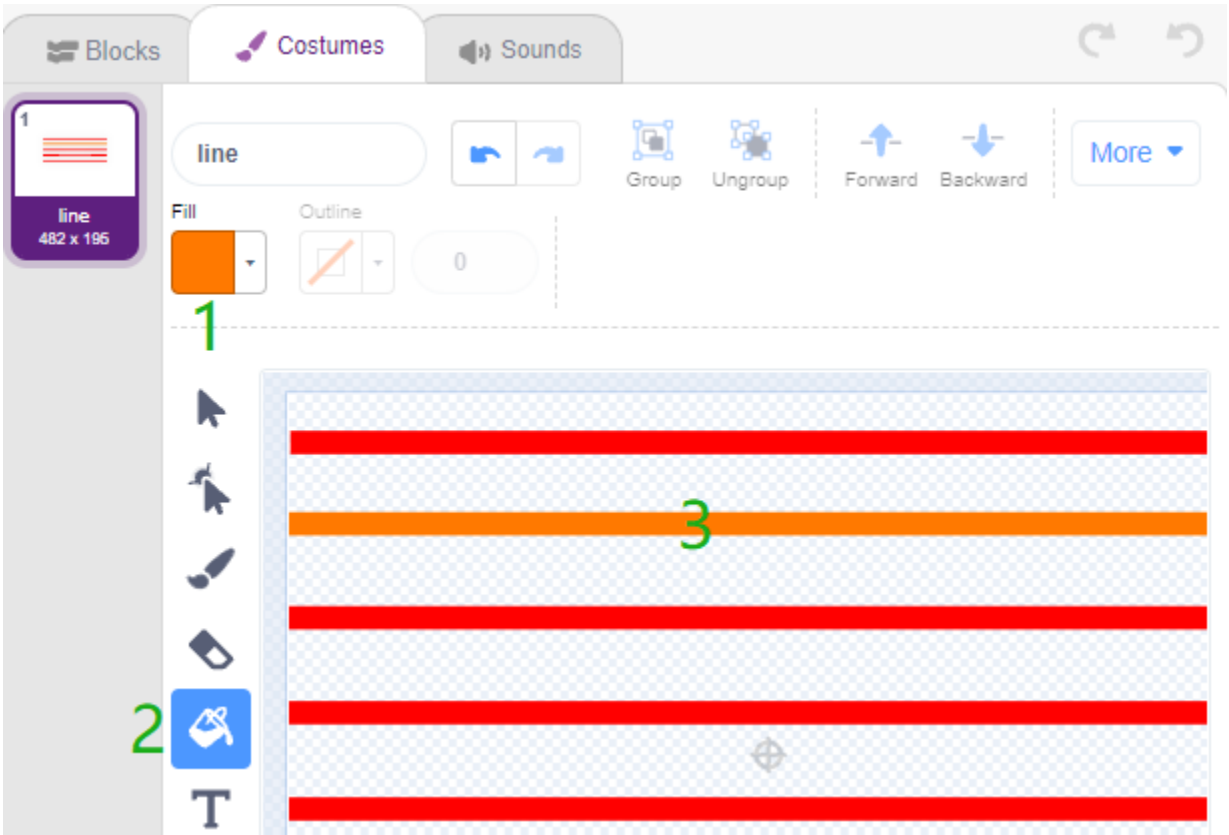
Añade un sprite de Línea.



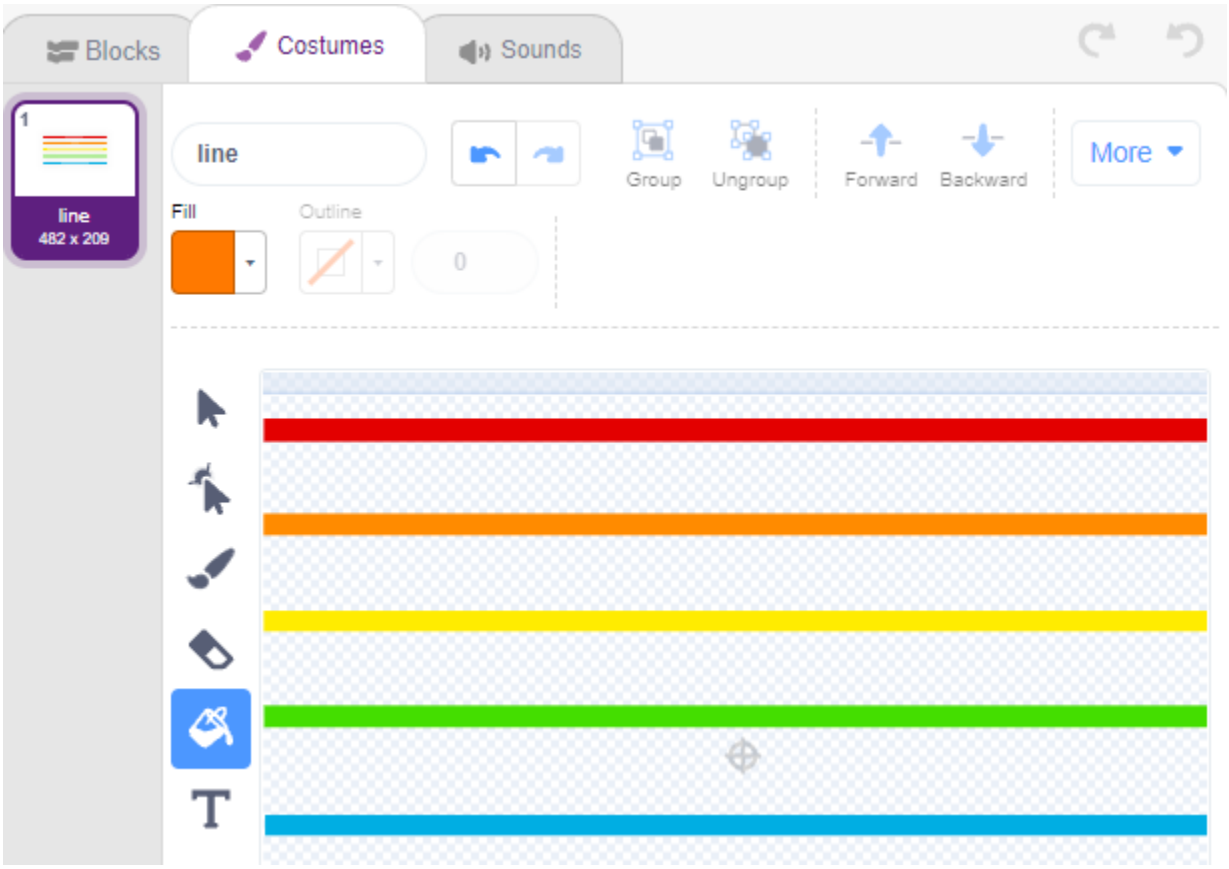
Ve a la página **Disfraces** del sprite **Línea**, reduce ligeramente el ancho de la línea roja en el lienzo, luego cópiala 5 veces y alinea las líneas.



Ahora rellena las líneas con diferentes colores. Primero elige un color que te guste, luego haz clic en la herramienta **Rellenar** y mueve el ratón sobre la línea para llenarla de color.



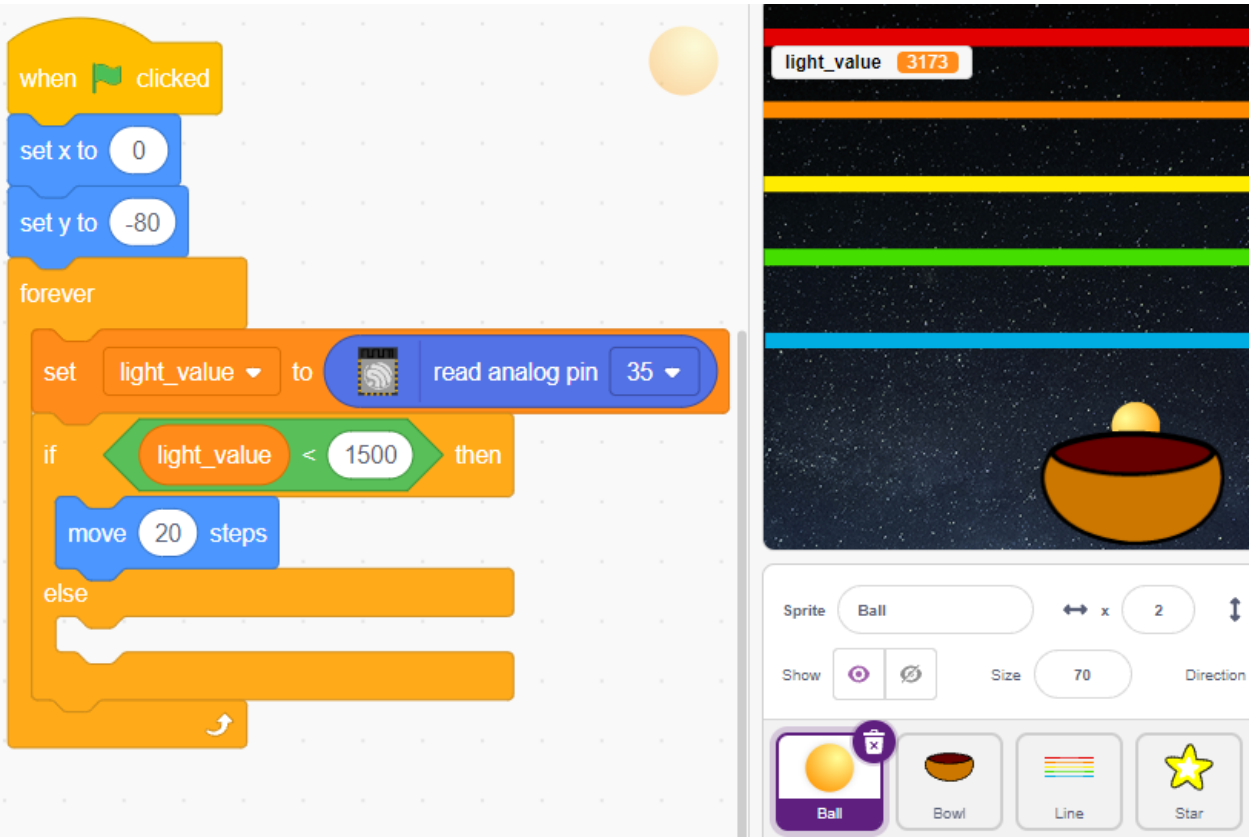
Sigue el mismo método para cambiar el color de las otras líneas.



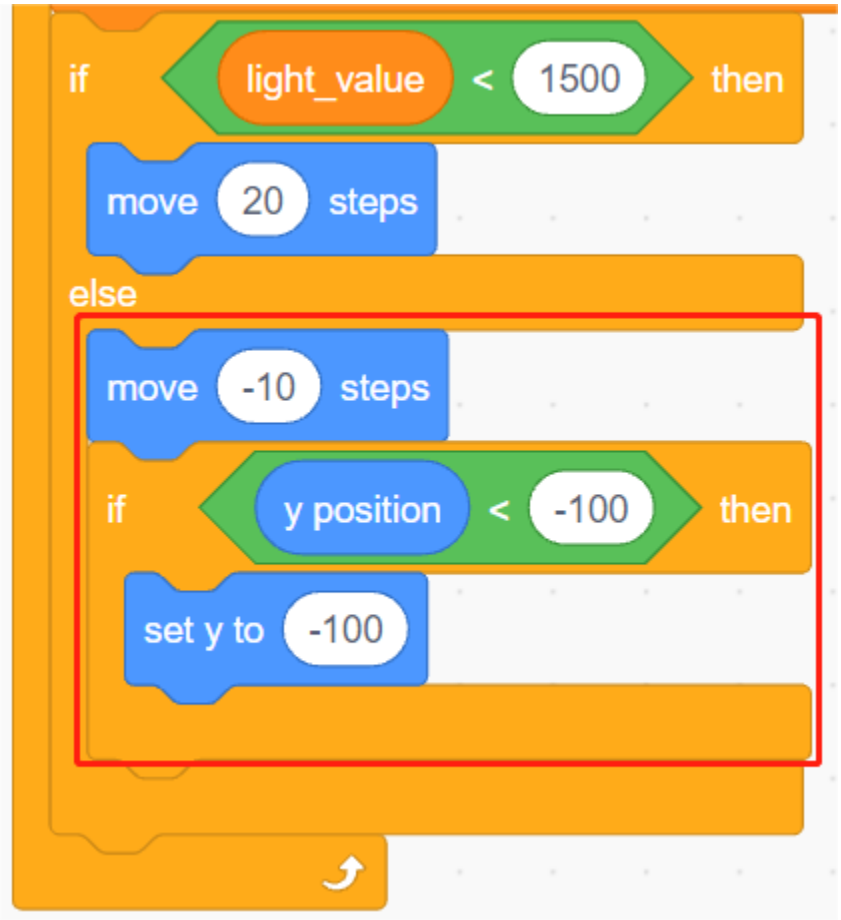
3. Programación del sprite Bola

Establece la posición inicial del sprite **Bola**, luego cuando el valor de luz sea menor a 1500 (puede ser cualquier otro valor, dependiendo de tu ambiente actual.), deja que la Bola se mueva hacia arriba.

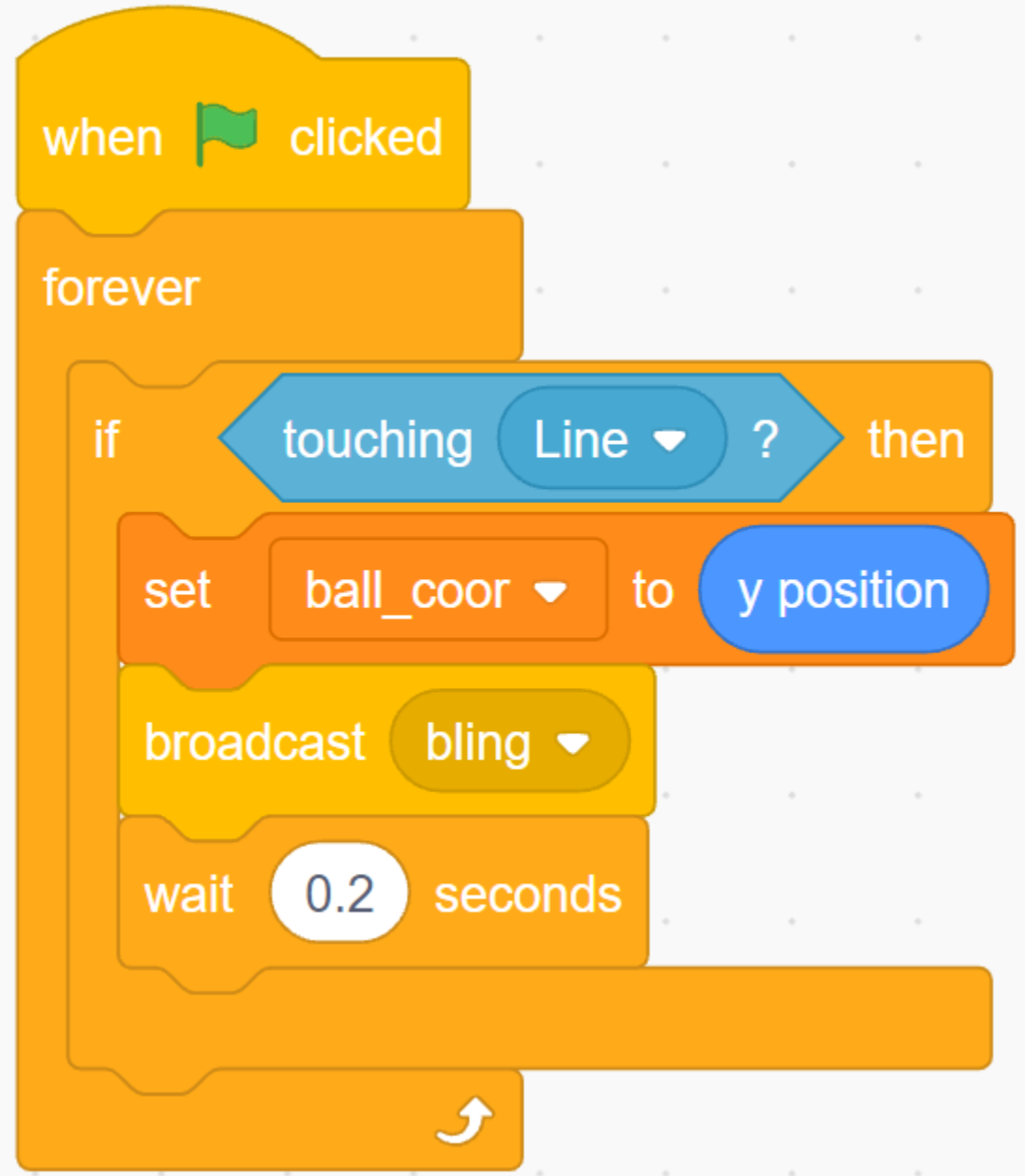
Puedes hacer que la variable valor_luz se muestre en el escenario para observar el cambio de intensidad de luz en cualquier momento.



De lo contrario, el sprite **Bola** caerá y limitará su coordenada Y a un mínimo de -100. Esto se puede modificar para que parezca que está cayendo sobre el sprite **Tazón**.

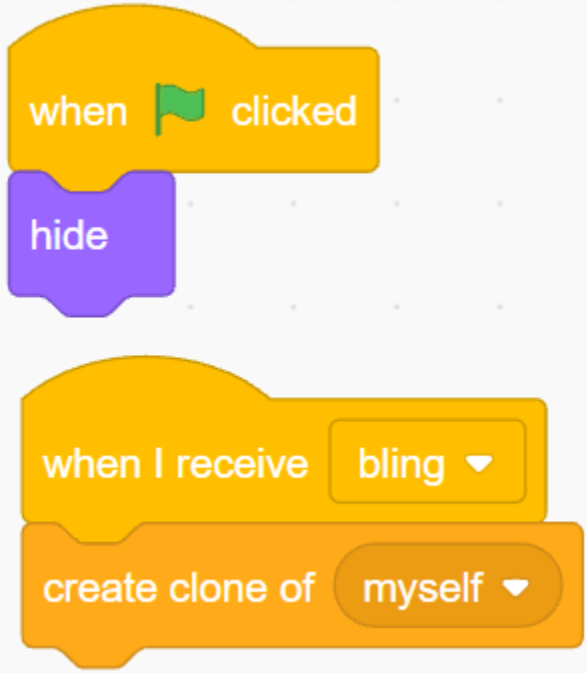


Cuando el sprite **Línea** sea golpeado, la coordenada Y actual se guarda en la variable **coord_bola** y se difunde el mensaje **Bling**.

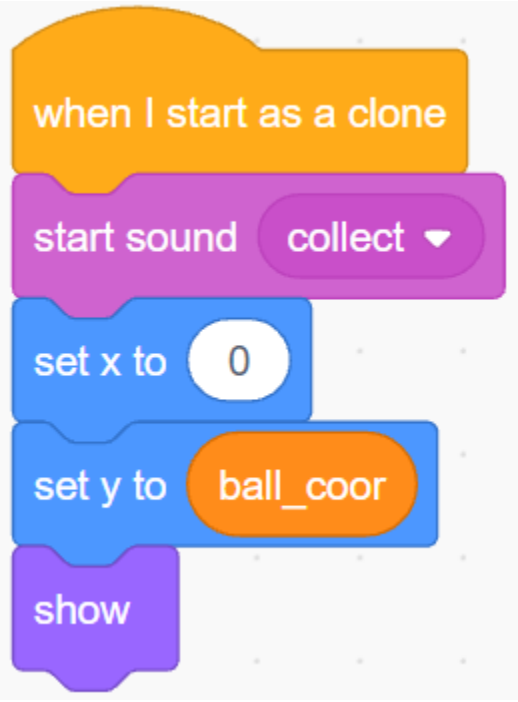


4. Programación del sprite Estrella

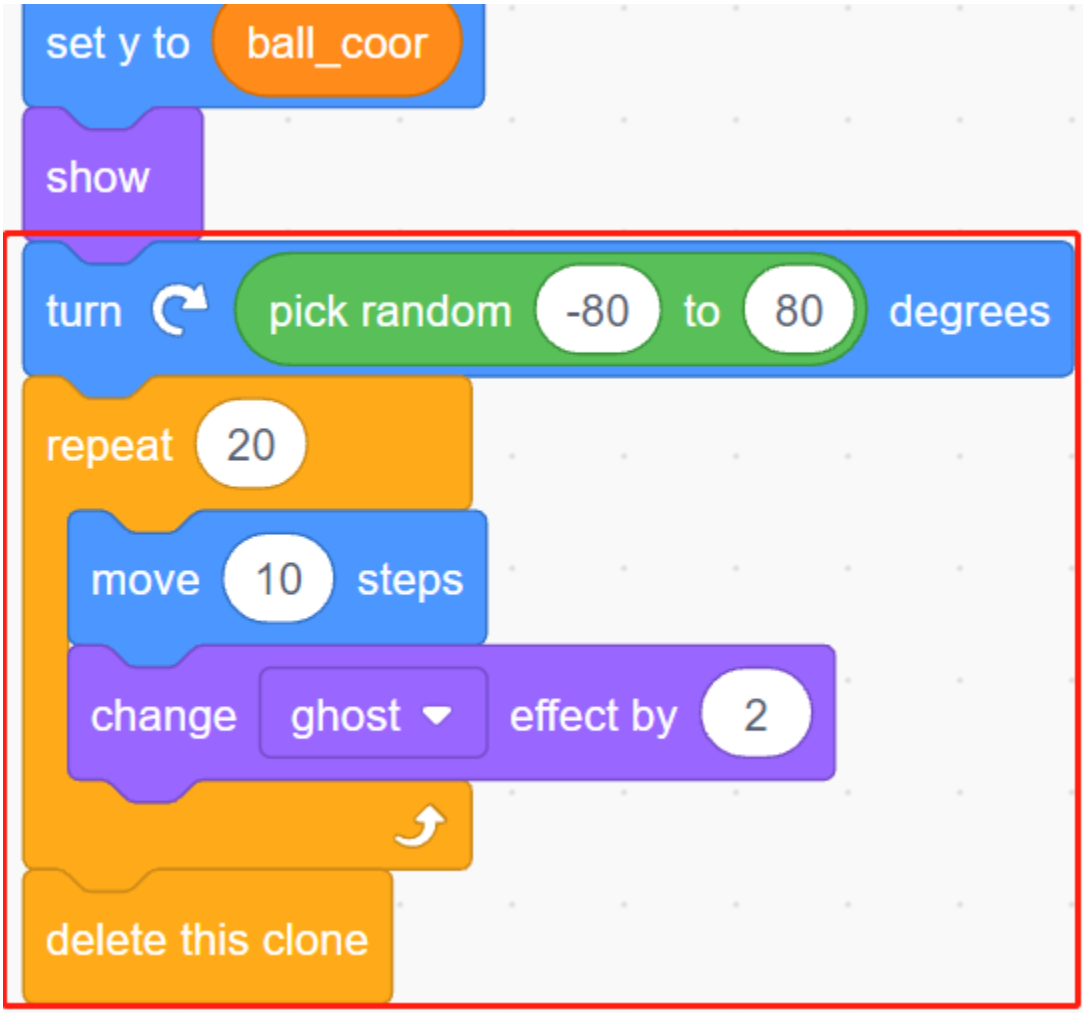
Cuando el script comienza, primero oculta el sprite **Estrella**. Cuando se recibe el mensaje **Bling**, clona el sprite **Estrella**.



Cuando el sprite **Estrella** aparece como un clon, reproduce el efecto de sonido y establece sus coordenadas para que estén sincronizadas con el sprite **Bola**.



Crea el efecto de aparición del sprite **Estrella** y ajústalo según sea necesario.

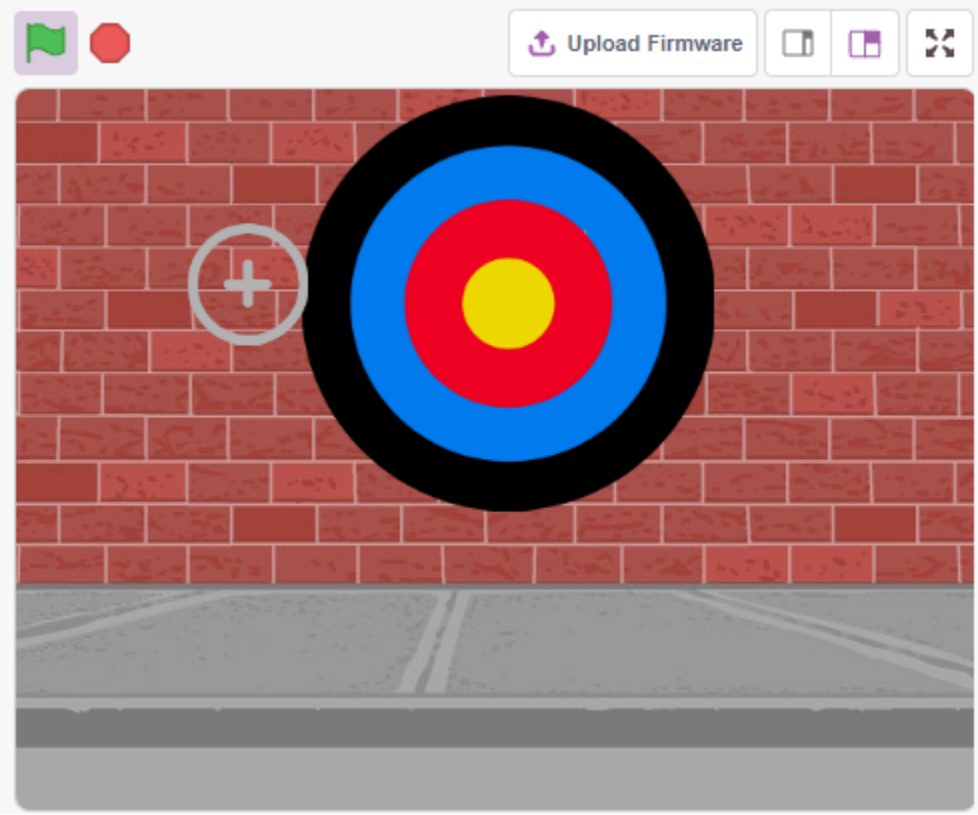


5.14 2.11 JUEGO - Disparos

¿Has visto esos juegos de disparos en la televisión? Cuanto más cerca un concursante dispare una bala al blanco cerca del centro, mayor será su puntuación.

Hoy también haremos un juego de disparos en Scratch. En el juego, permite que la Mira dispare lo más cerca posible al centro para obtener una puntuación más alta.

Haz clic en la bandera verde para empezar. Usa el módulo de Evitación de Obstáculos para disparar una bala.



5.14.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Evitación de Obstáculos	

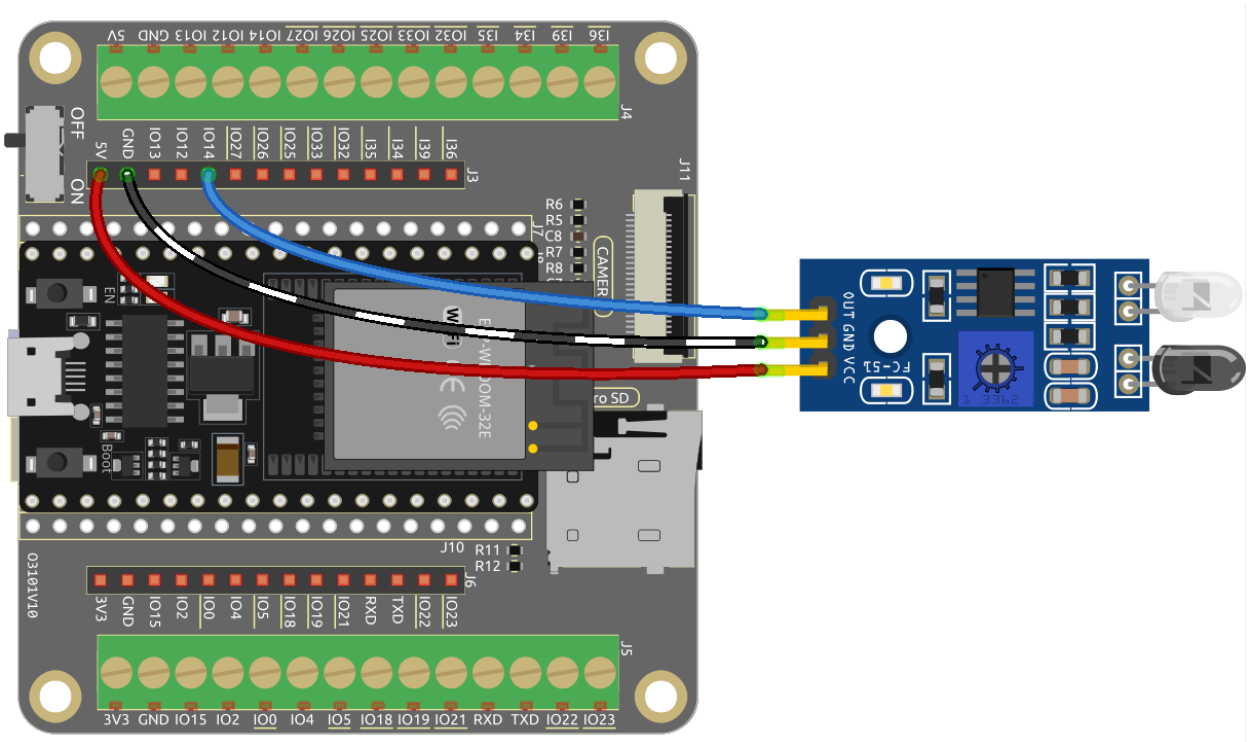
5.14.2 Lo Que Aprenderás

- Cómo funciona el módulo de Evitación de Obstáculos y el rango de ángulo
- Pintar diferentes sprites
- Tocar colores

5.14.3 Construye el Circuito

El módulo de evitación de obstáculos es un sensor de proximidad infrarrojo con distancia ajustable cuya salida es normalmente alta y baja cuando se detecta un obstáculo.

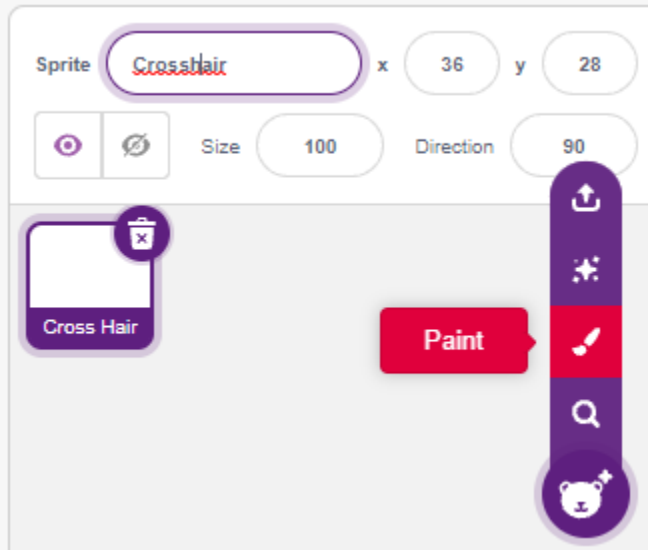
Ahora construye el circuito según el diagrama a continuación.



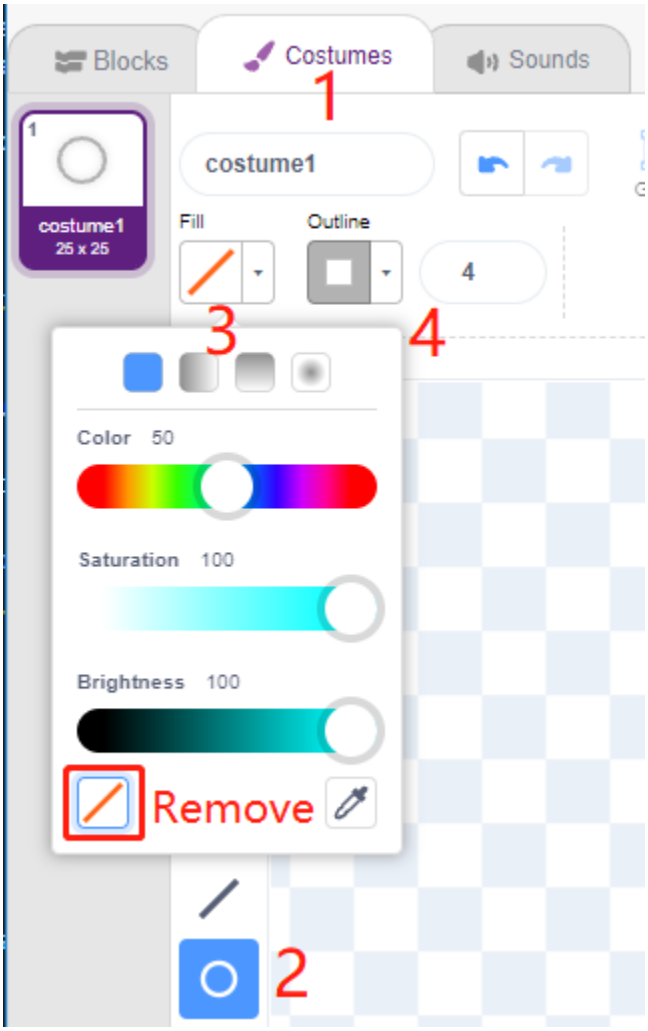
5.14.4 Programación

1. Pintar el sprite de la Mira

Elimina el sprite predeterminado, selecciona el botón **Sprite** y haz clic en **Pintar**, aparecerá un sprite en blanco **Sprite1** y nómbralo **Mira**.

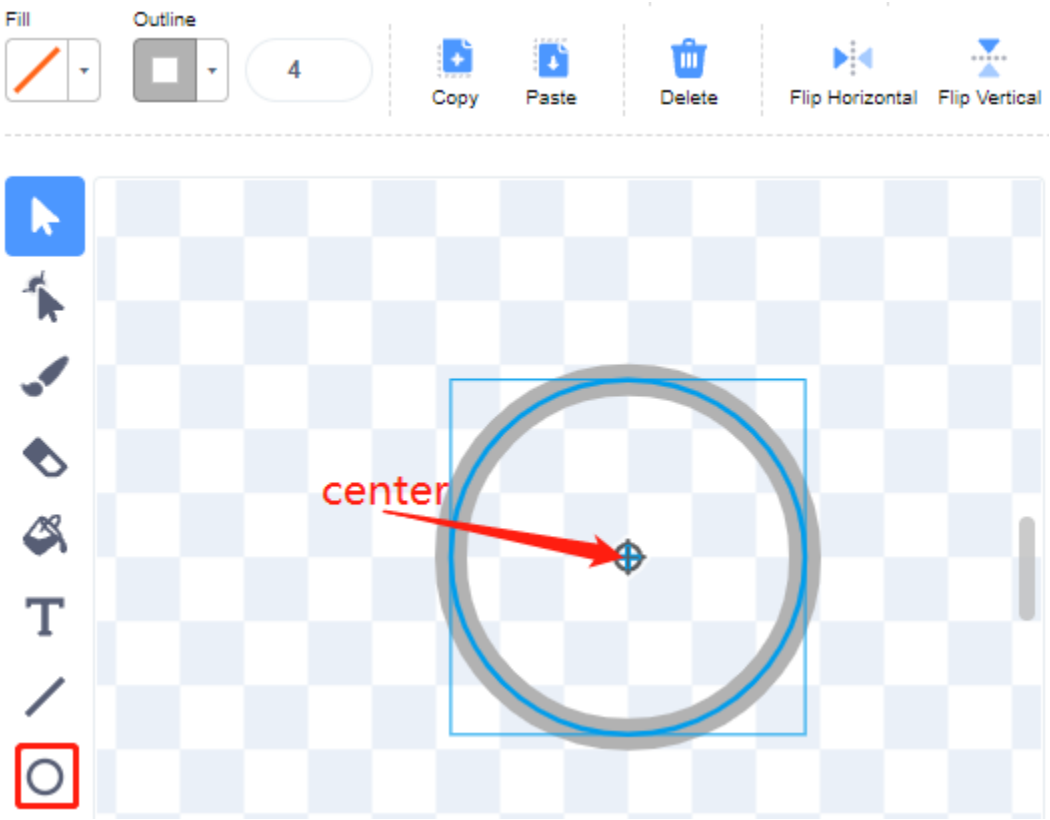


Ve a la página **Disfraces** del sprite **Mira**. Haz clic en la herramienta **Círculo**, elimina el color de relleno y configura el color y el ancho del contorno.

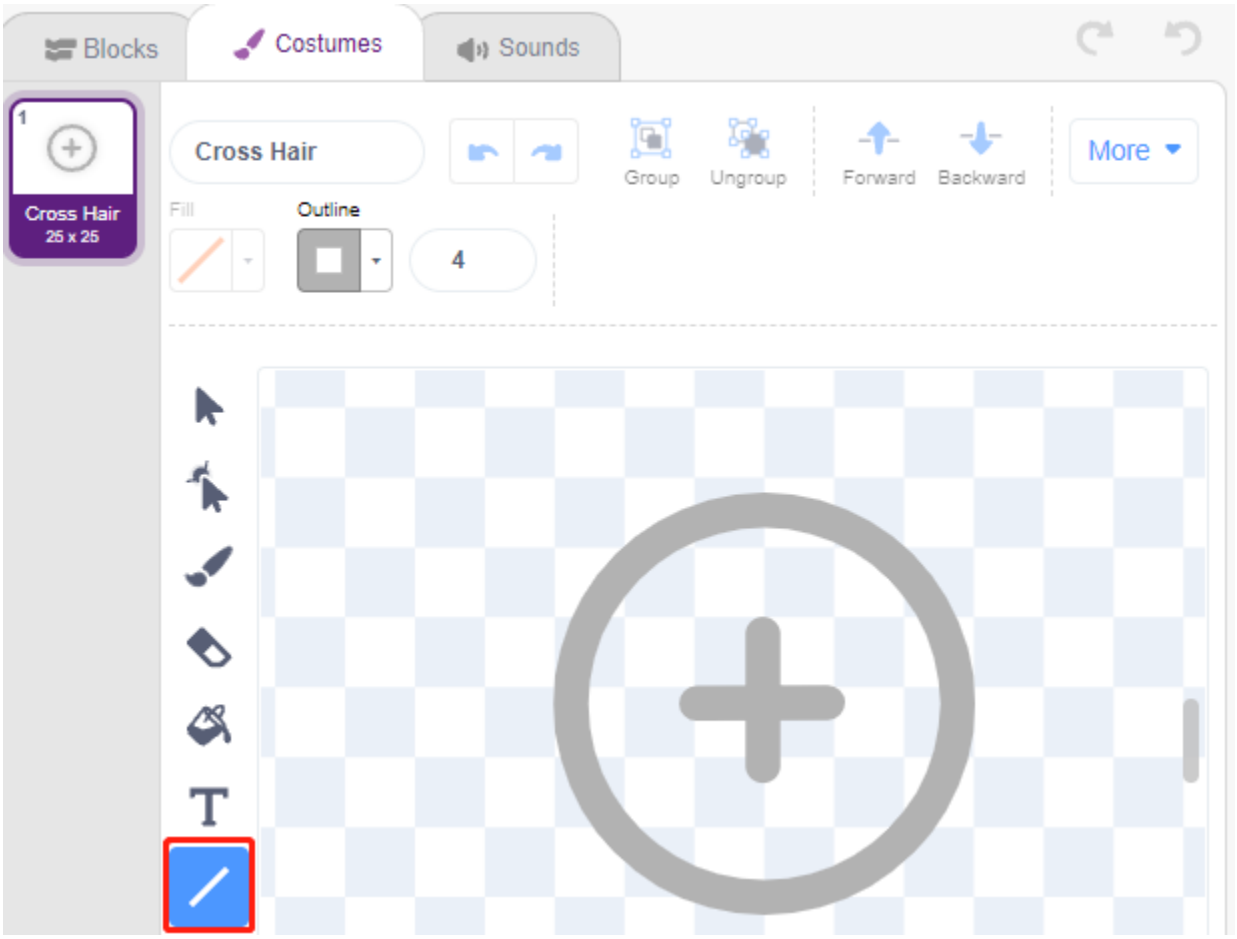


Ahora dibuja un círculo con la herramienta **Círculo**. Después de dibujar, puedes hacer clic en la herramienta **Seleccio-**

nar y mover el círculo para que el punto original se alinee con el centro del lienzo.

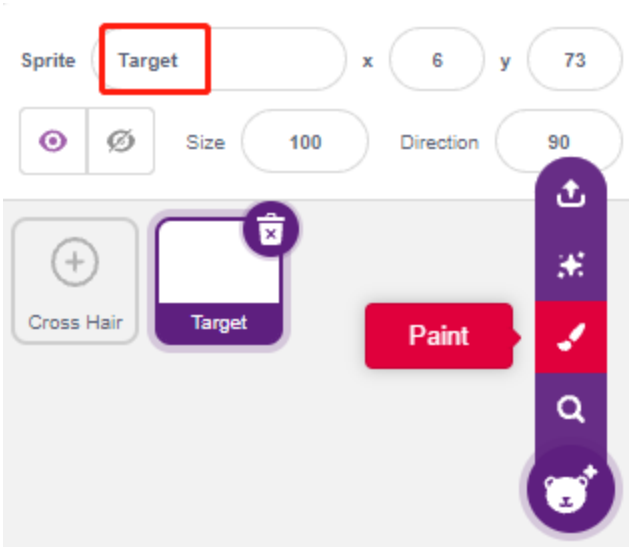


Usando la herramienta **Línea**, dibuja una cruz dentro del círculo.

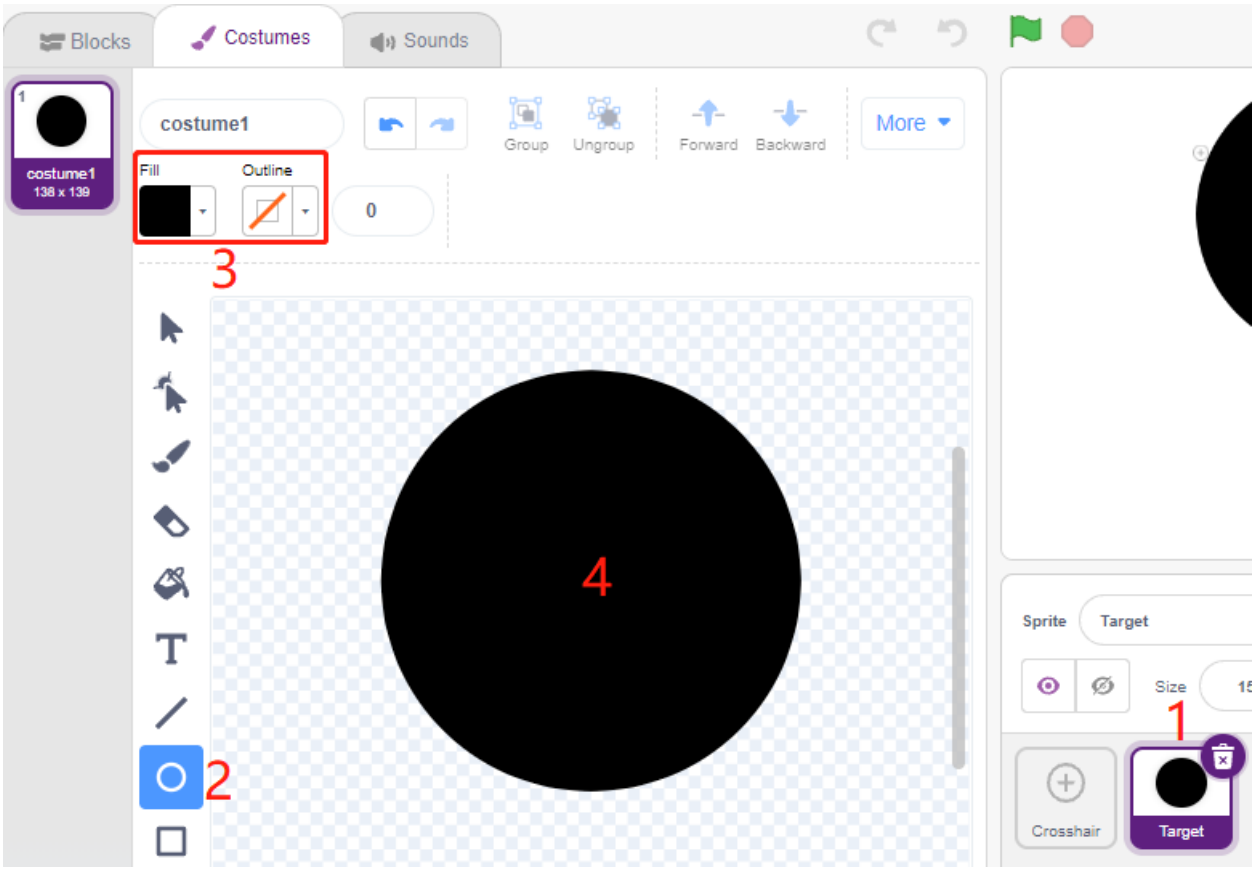


Pintar el sprite del Objetivo

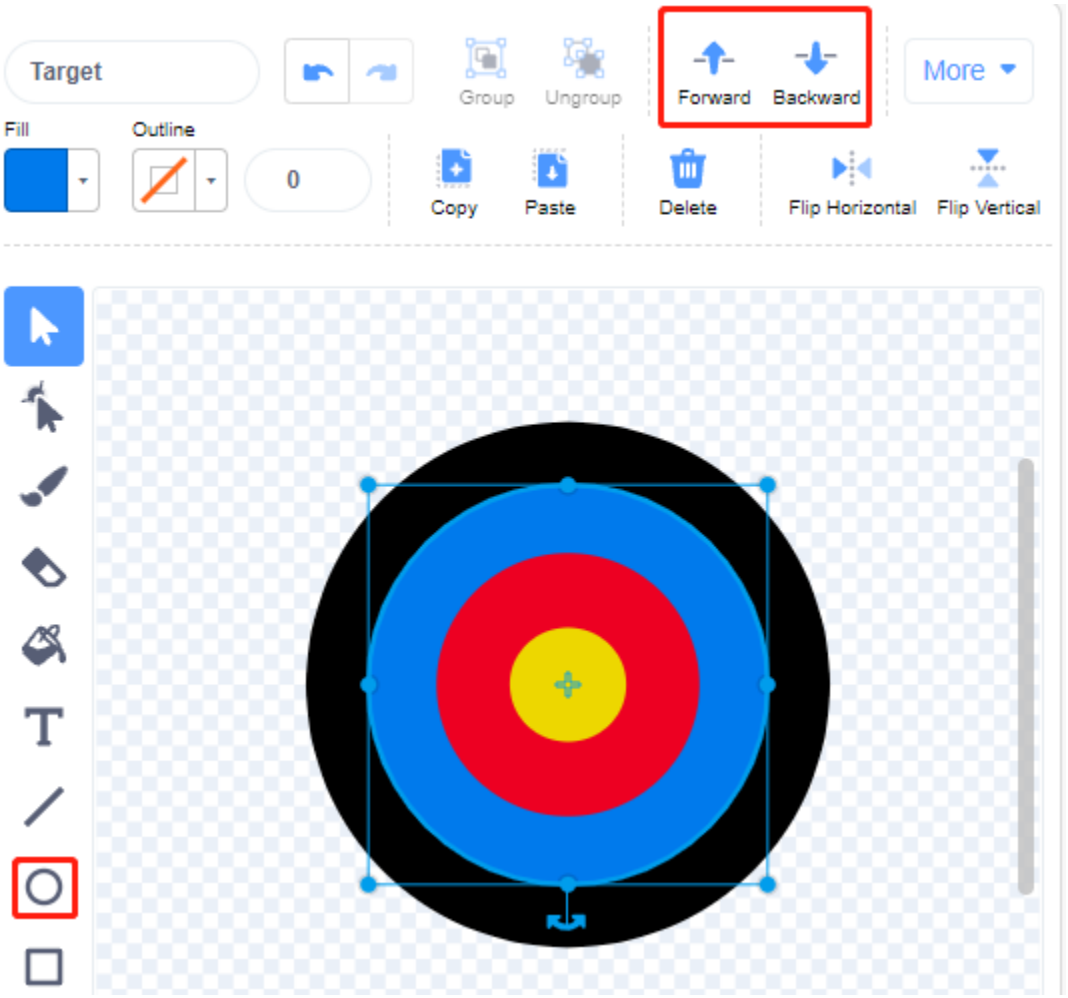
Crea un nuevo sprite llamado **Objetivo**.



Ve a la página de Disfraces del sprite **Objetivo**, haz clic en la herramienta **Círculo**, selecciona un color de relleno y elimina el Contorno y pinta un círculo grande.

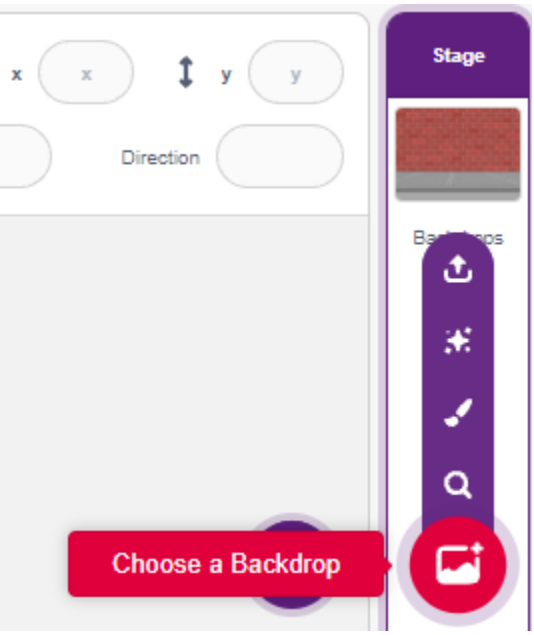


Usa el mismo método para dibujar círculos adicionales, cada uno con un color diferente, y puedes usar la herramienta **Adelante** o **Atrás** para cambiar la posición de los círculos superpuestos. Ten en cuenta que también necesitas seleccionar la herramienta para mover los círculos, para que el origen de todos los círculos y el centro del lienzo estén alineados.



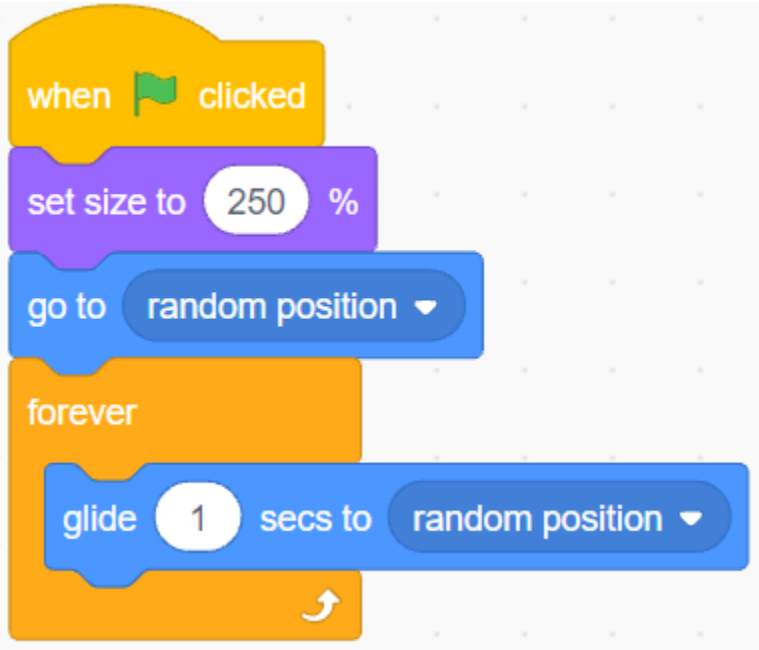
3. Añadir un fondo

Añade un fondo adecuado que preferiblemente no tenga demasiados colores y no coincida con los colores en el sprite **Objetivo**. Aquí he elegido el fondo **Pared1**.

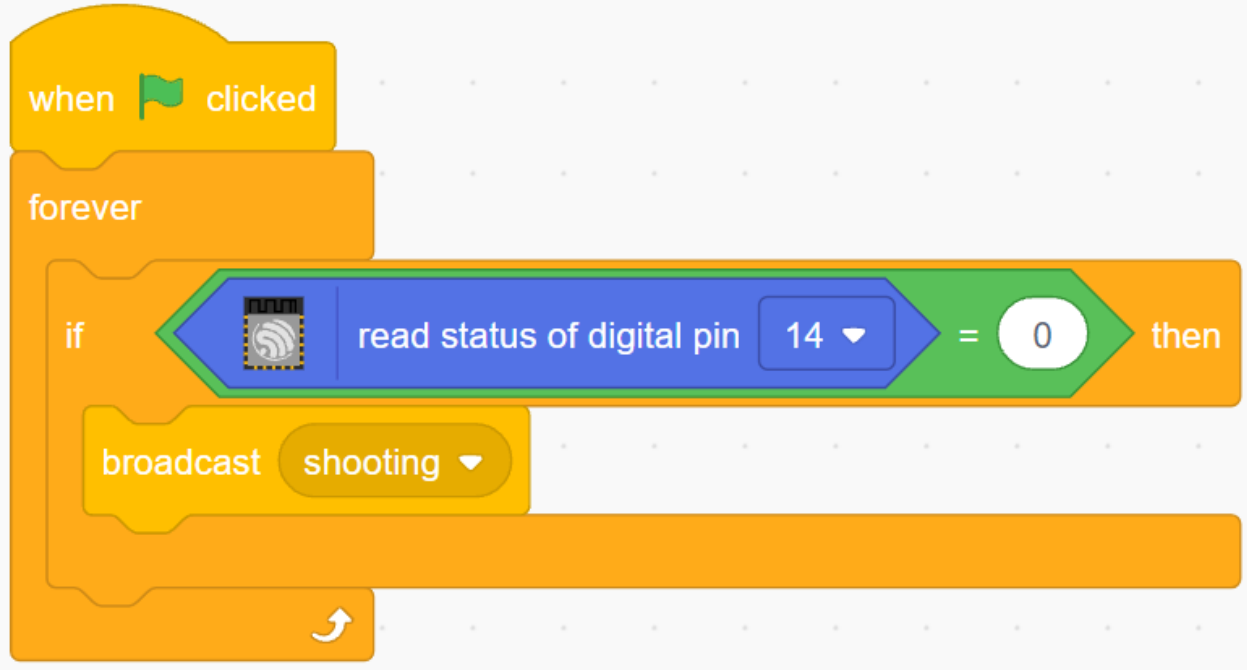


4. Programar el sprite de la Mira

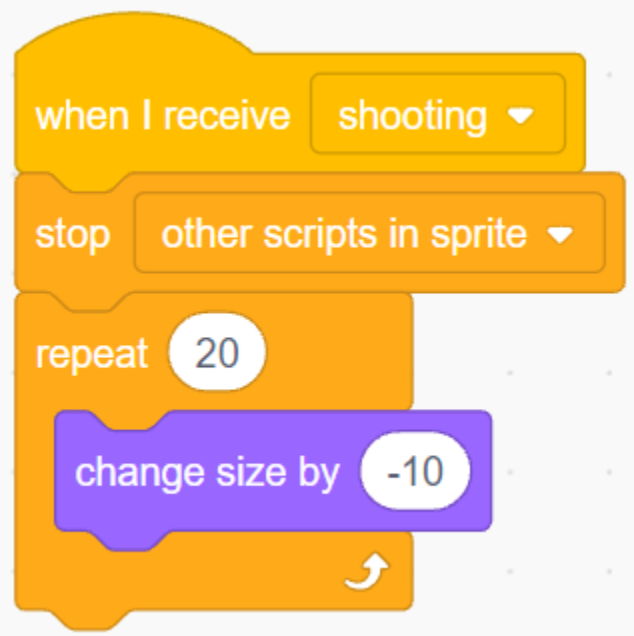
Establece la posición y el tamaño aleatorios del sprite **Mira**, y déjalo moverse aleatoriamente.



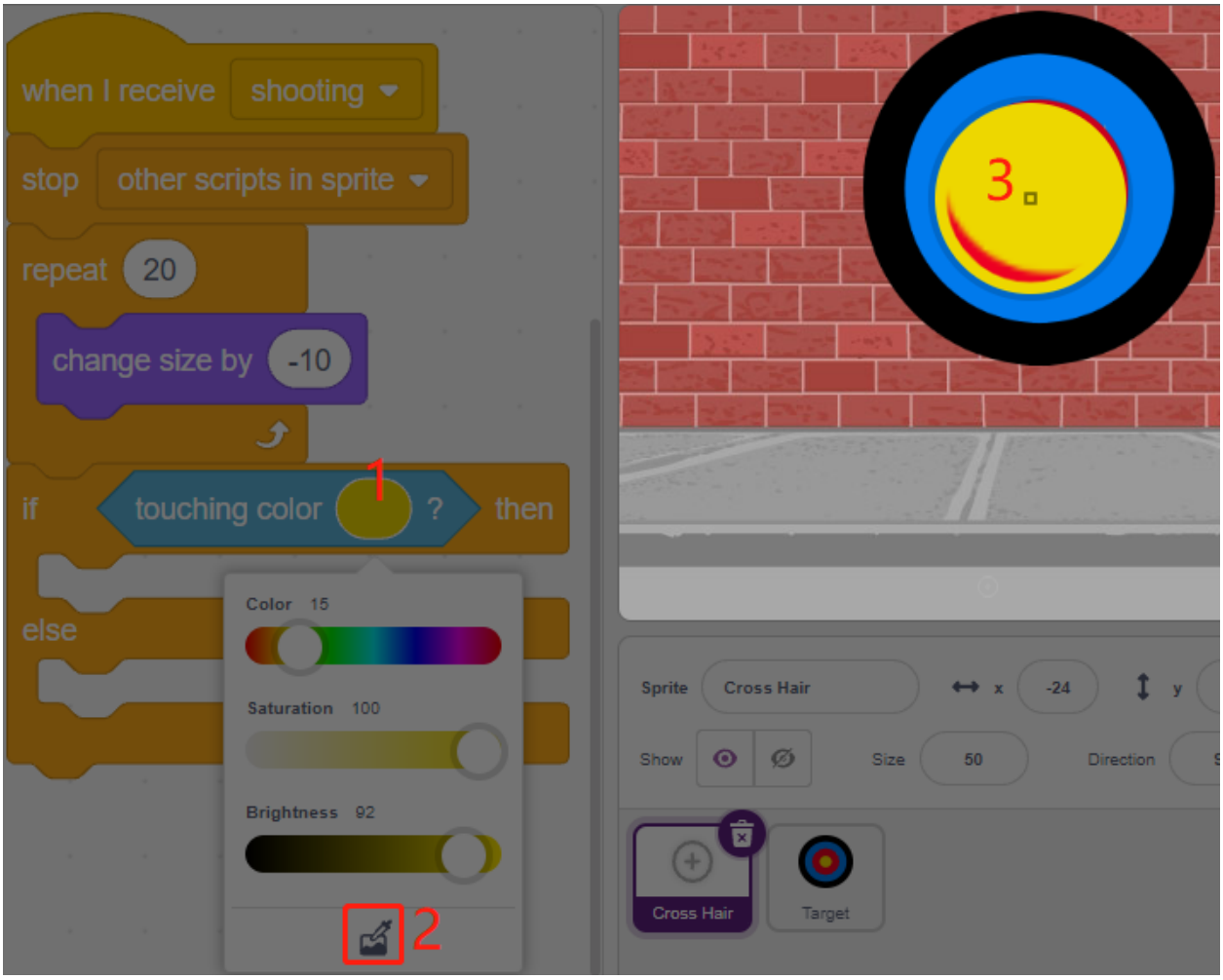
Cuando se coloca una mano frente al módulo de evitación de obstáculos, emitirá un nivel bajo como señal de transmisión.



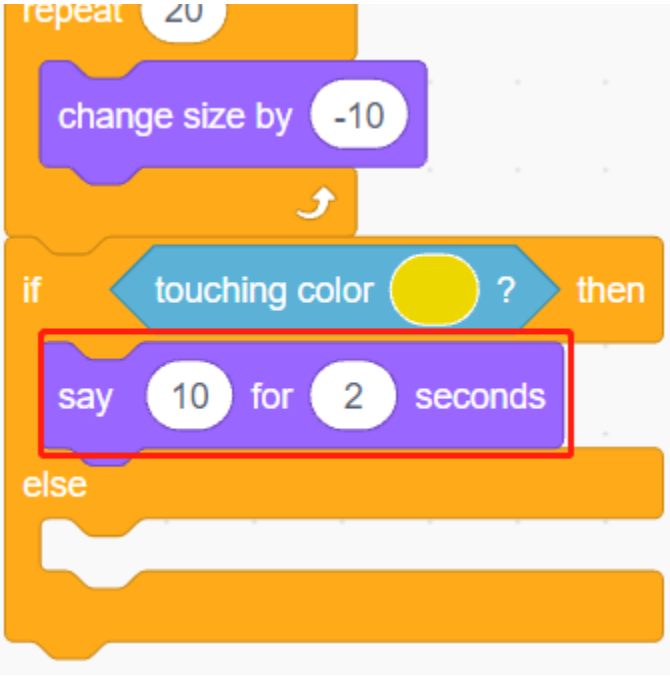
Cuando se recibe el mensaje de **disparo**, el sprite deja de moverse y se encoge lentamente, simulando así el efecto de disparar una bala.



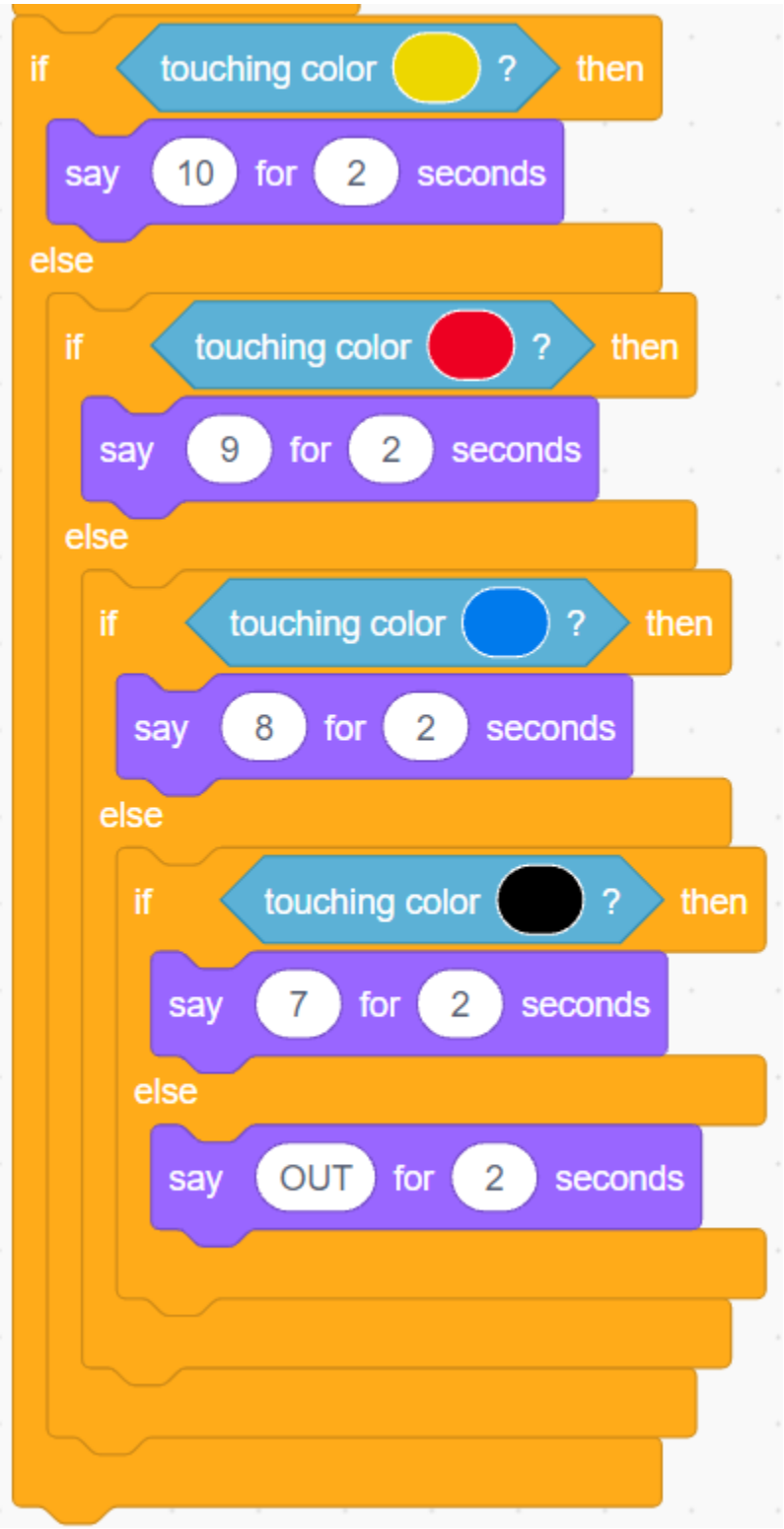
Usa el bloque [Tocar color ()] para determinar la posición del disparo.



Cuando el disparo está dentro del círculo amarillo, se informa 10.



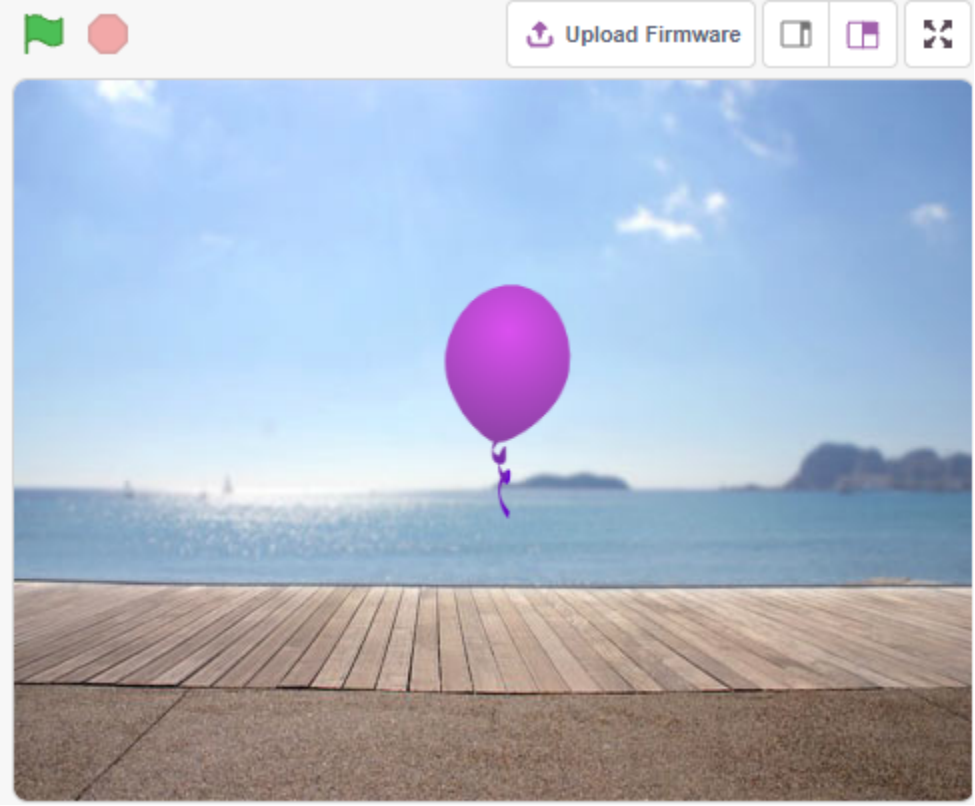
Usa el mismo método para determinar la posición del disparo de la bala, si no se establece en el sprite **Objetivo**, significa que está fuera del círculo.


if touching color ? then
say 10 for 2 seconds
else
if touching color ? then
say 9 for 2 seconds
else
if touching color ? then
say 8 for 2 seconds
else
if touching color ? then
say 7 for 2 seconds
else
say OUT for 2 seconds

5.15 2.12 JUEGO - Inflando el Globo

Aquí, jugaremos un juego de inflar globos.

Después de hacer clic en la bandera verde, el globo se hará más y más grande. Si el globo es demasiado grande, explotará; si el globo es demasiado pequeño, caerá; necesitas juzgar cuándo presionar el botón para hacerlo volar hacia arriba.



5.15.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

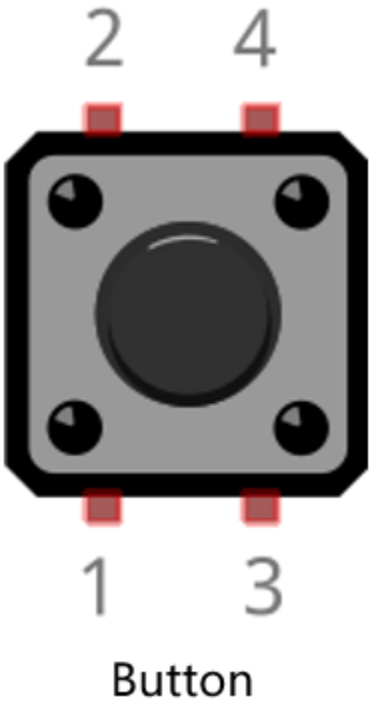
INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Botón	

5.15.2 Lo Que Aprenderás

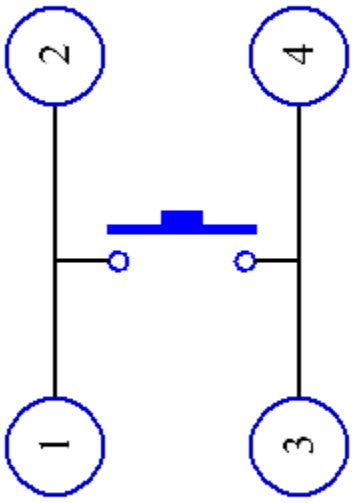
- Pintar disfraz para el sprite

5.15.3 Construye el Circuito

El botón es un dispositivo de 4 pines, ya que el pin 1 está conectado al pin 2, y el pin 3 al pin 4, cuando se presiona el botón, los 4 pines se conectan, cerrando así el circuito.



Button

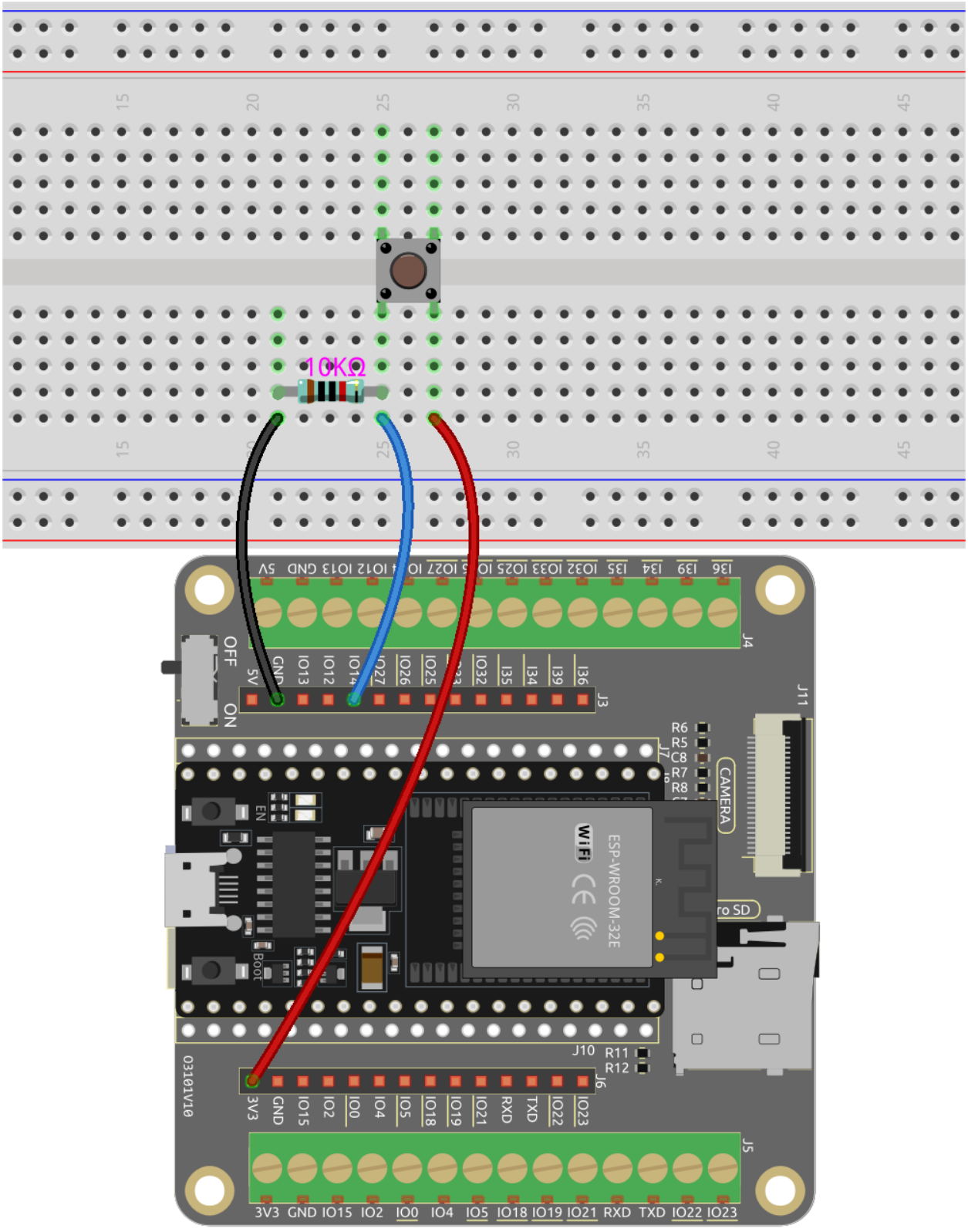


Internal Structure

Construye el circuito según el siguiente diagrama.

- Conecta uno de los pines en el lado izquierdo del botón al pin14, que está conectado a una resistencia de pull-down y un capacitor de 0.1uF (104) (para eliminar la fluctuación y emitir un nivel estable cuando el botón está funcionando).

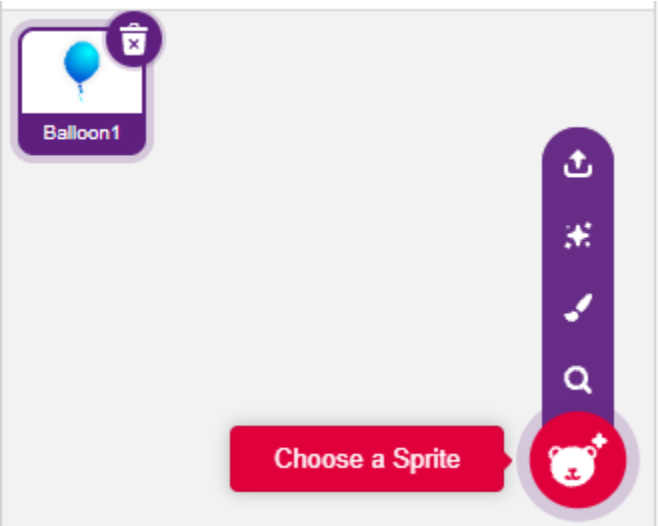
- Conecta el otro extremo de la resistencia y el capacitor a GND, y uno de los pines en el lado derecho del botón a 5V.



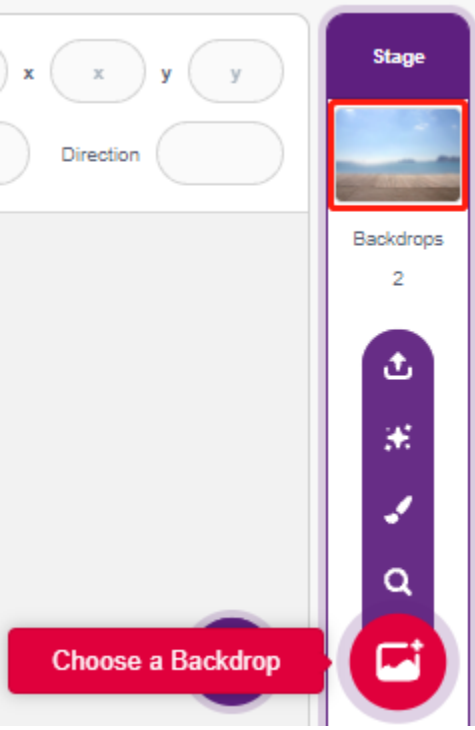
5.15.4 Programación

1. Añadir un sprite y un fondo

Elimina el sprite predeterminado, haz clic en el botón **Elegir un Sprite** en la esquina inferior derecha del área de sprites, luego selecciona el sprite **Globo1**.



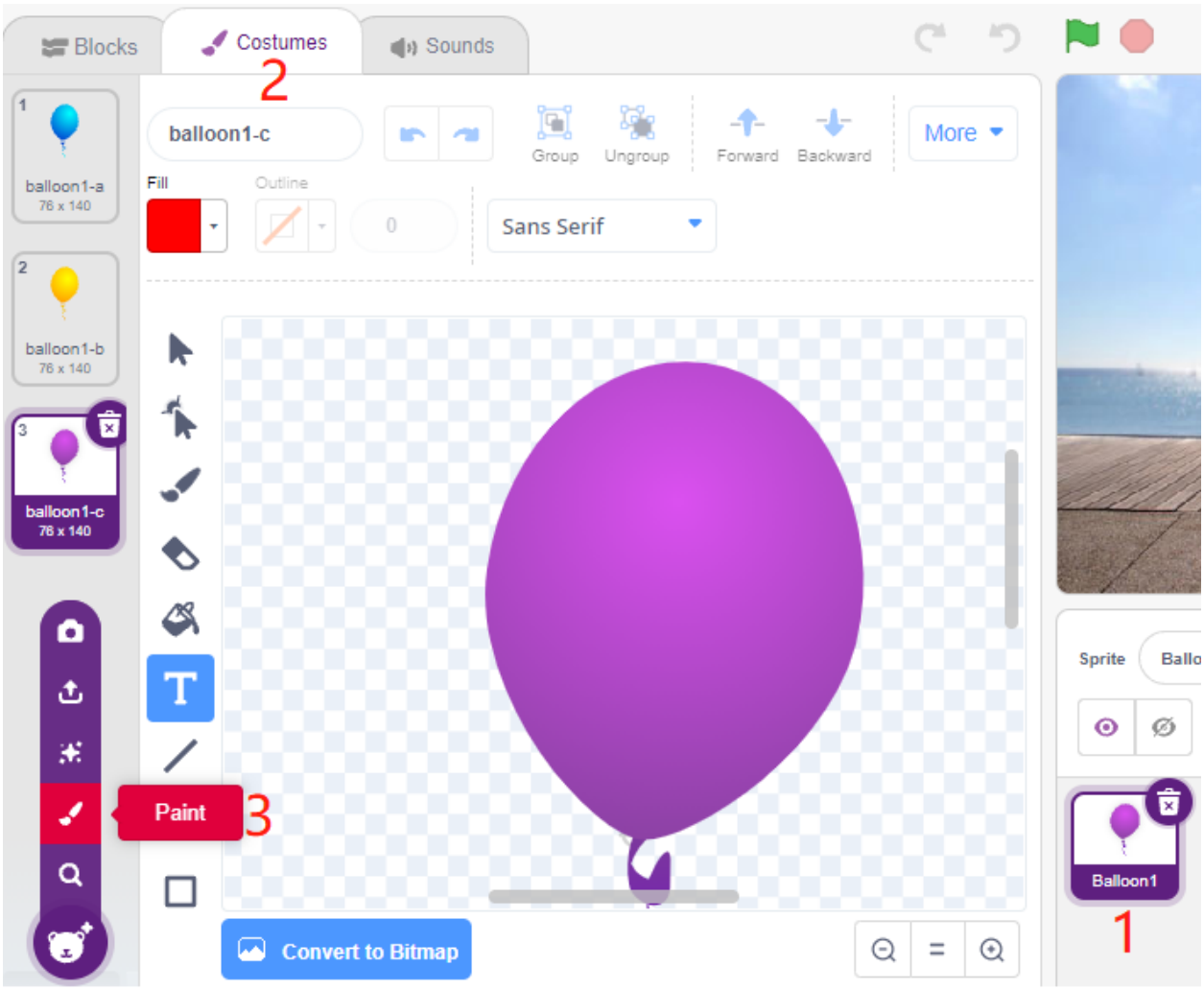
Añade un fondo **Paseo Marítimo** a través del botón **Elegir un fondo**, o cualquier otro fondo que te guste.



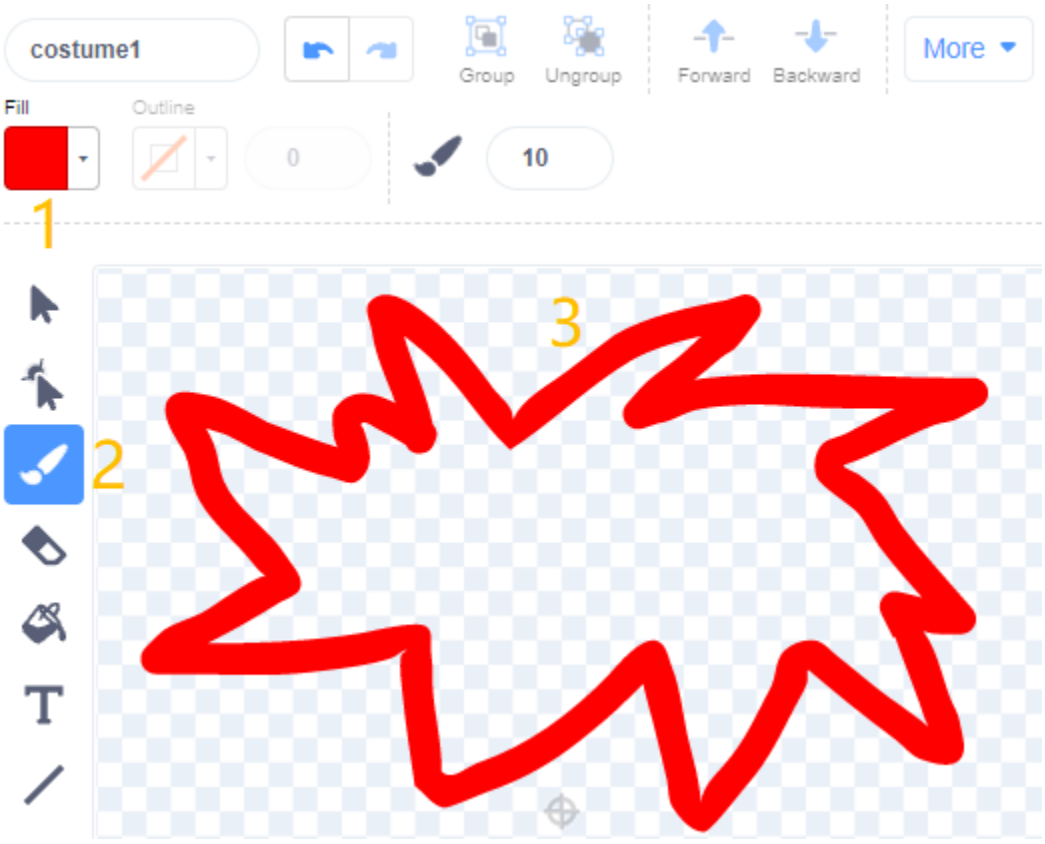
2. Pintar un disfraz para el sprite Globo1

Ahora vamos a dibujar un disfraz de efecto de explosión para el sprite del globo.

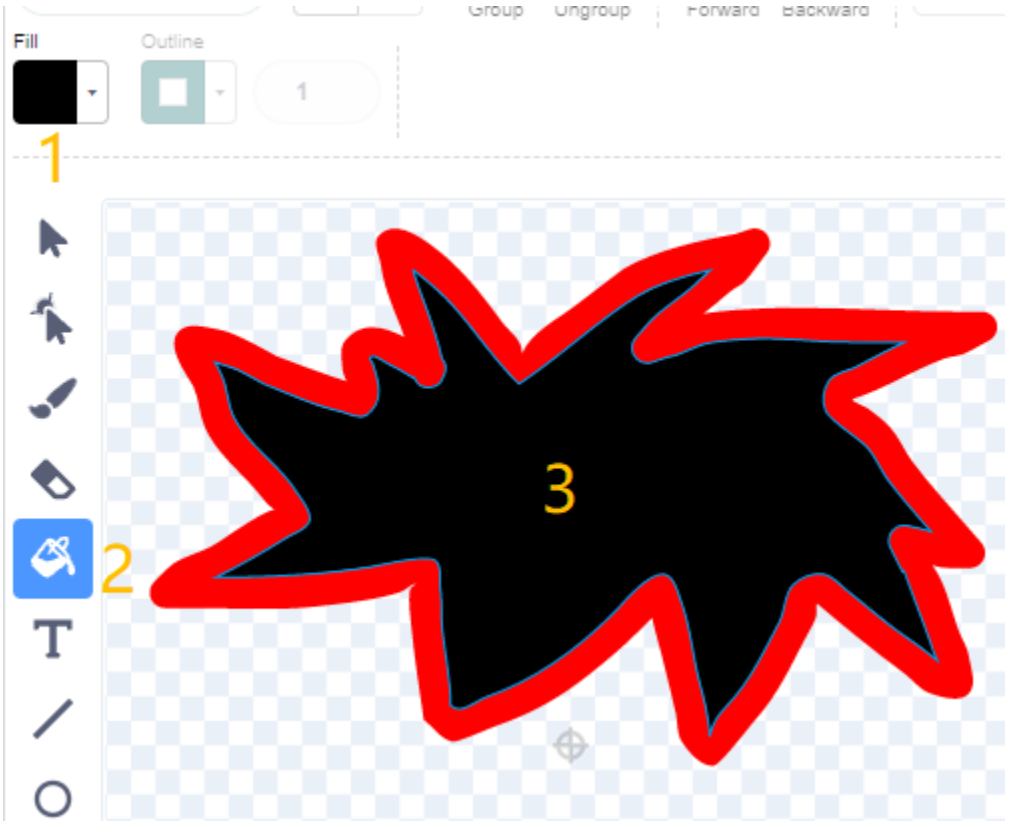
Ve a la página **Disfraces** del sprite **Globo1**, haz clic en el botón **Elegir un Disfraz** en la esquina inferior izquierda, y selecciona **Pintar** para abrir un **Disfraz** en blanco.



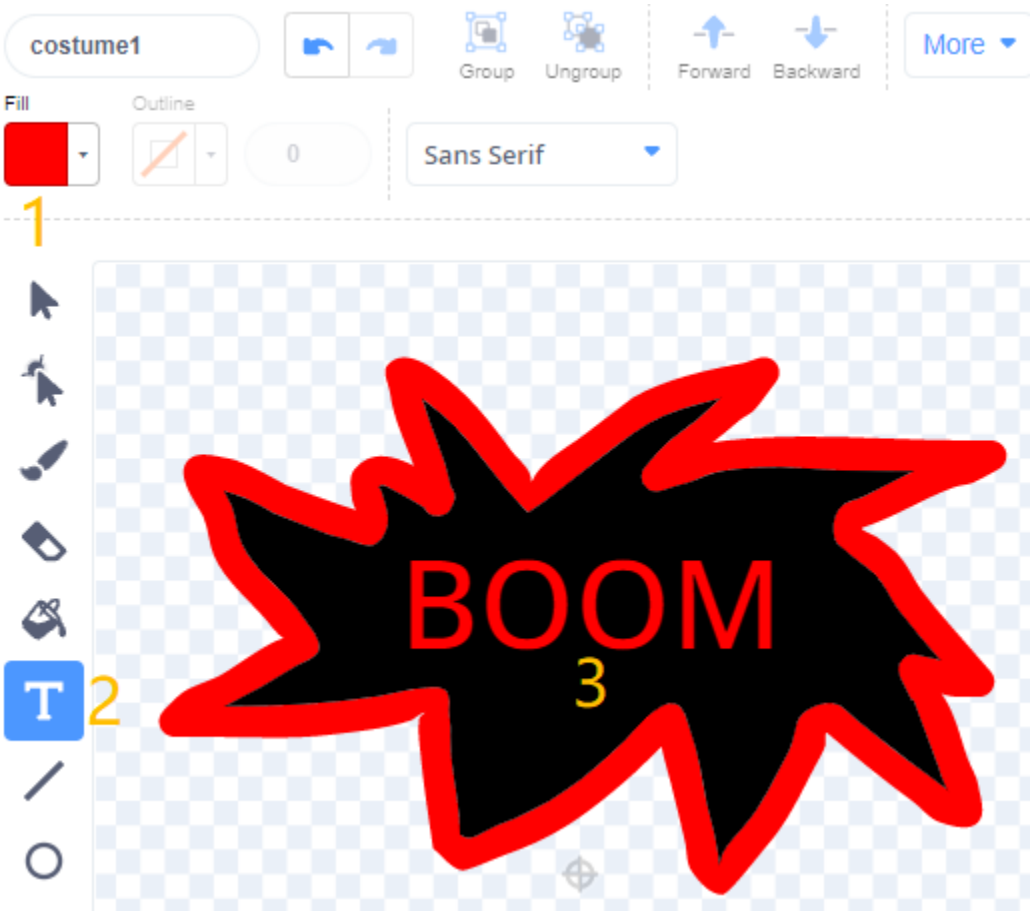
Selecciona un color y luego usa la herramienta **Pincel** para dibujar un patrón.



Selecciona nuevamente un color, haz clic en la herramienta Rellenar y mueve el ratón dentro del patrón para llenarlo de color.

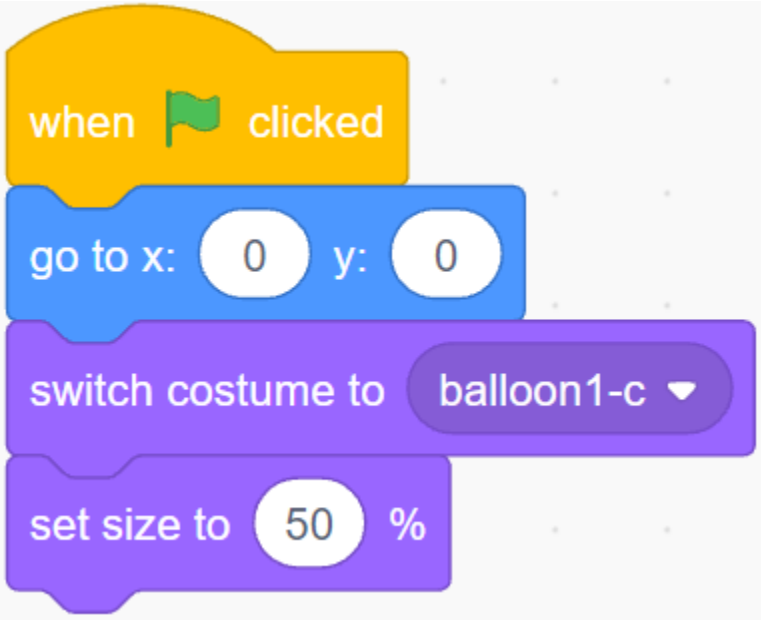


Finalmente, escribe el texto BOOM, para que el disfraz de efecto de explosión esté completo.

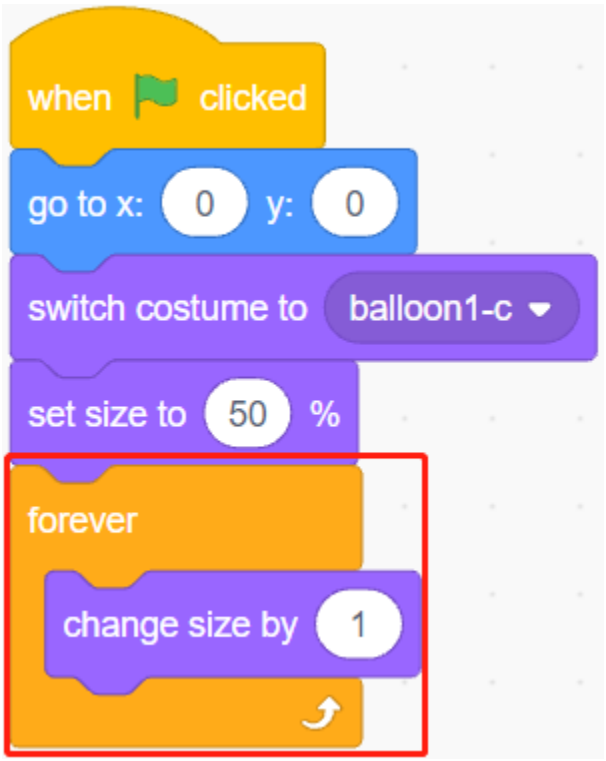


3. Programación del sprite Globo

Establece la posición inicial y el tamaño del sprite **Globo1**.

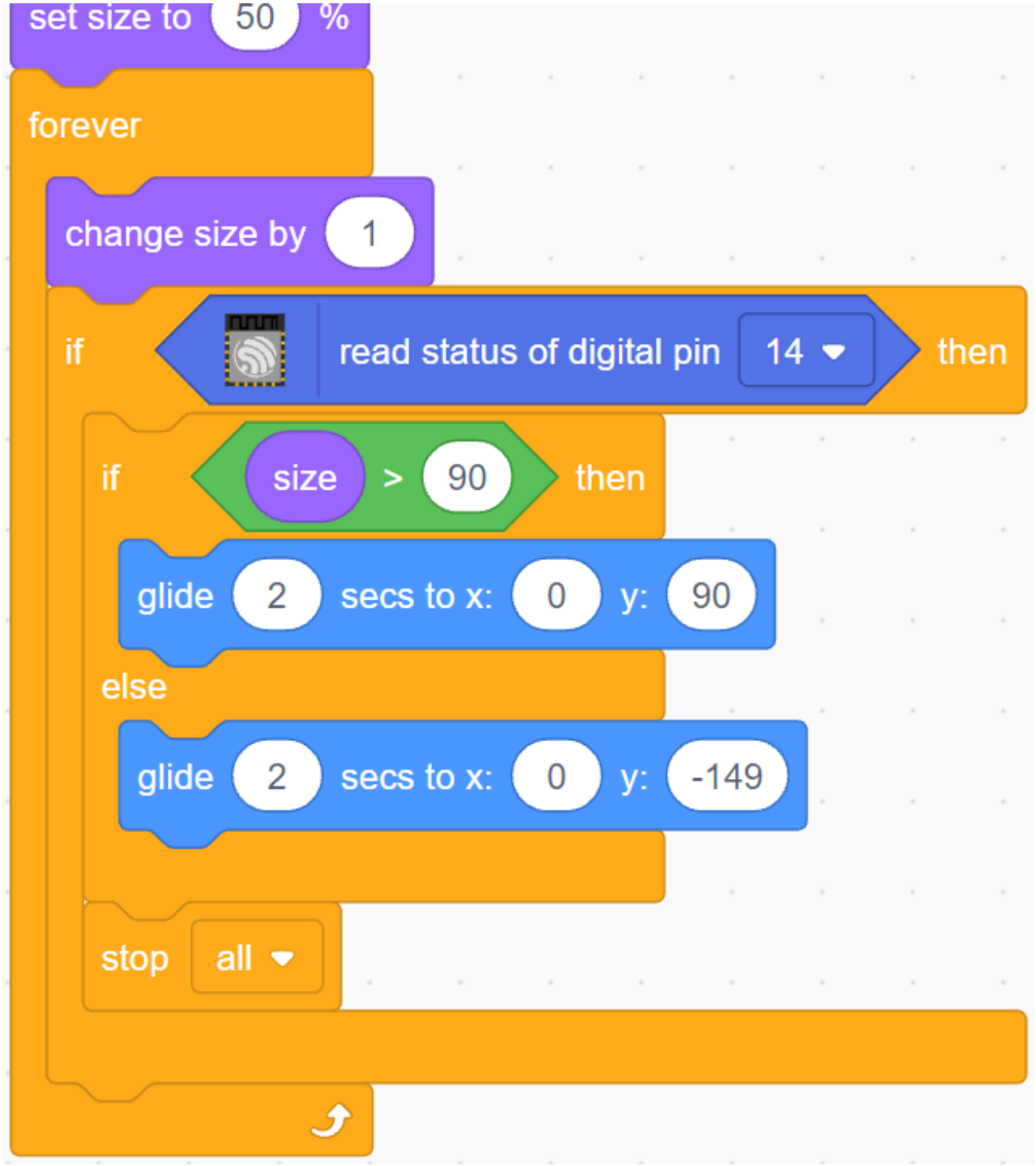


Luego, permite que el sprite **Globo** se haga lentamente más grande.

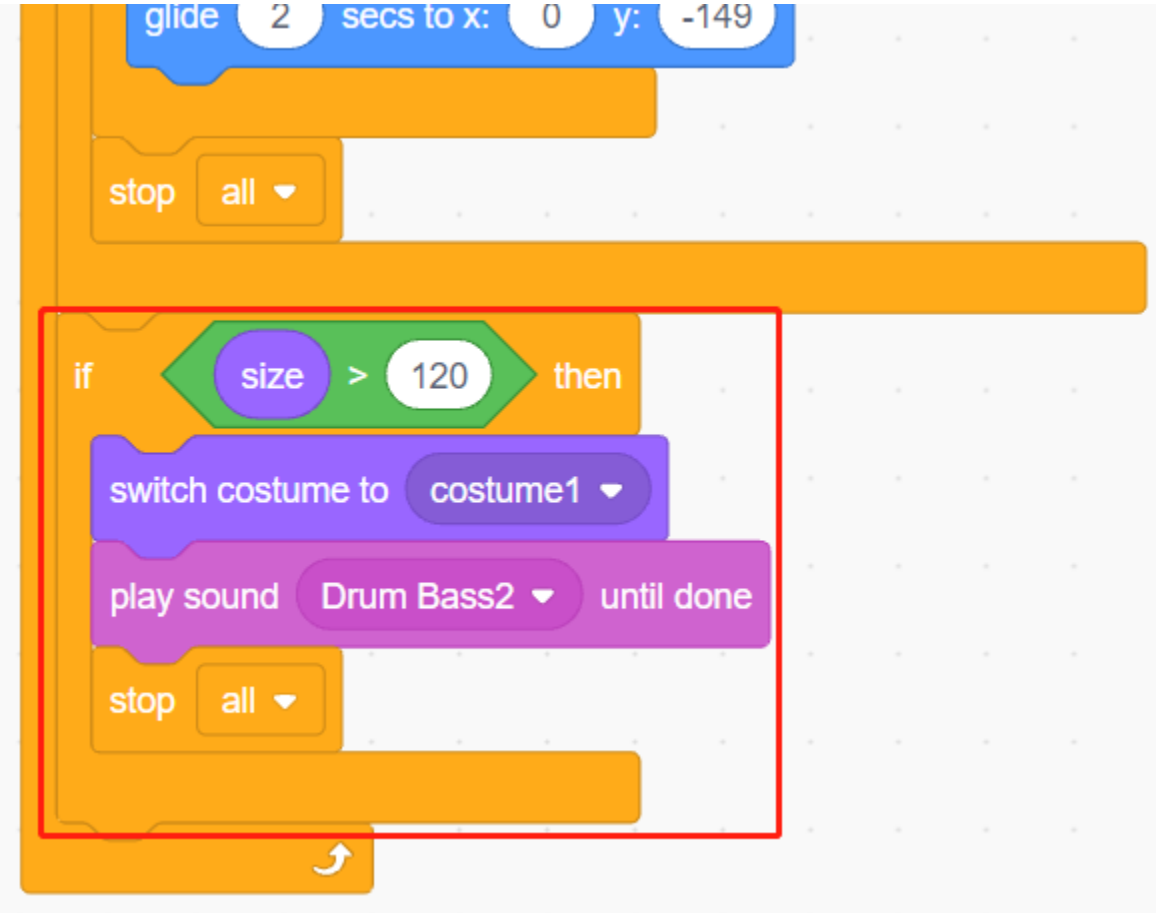


Cuando se presiona el botón (el valor es 1), el tamaño del sprite **Globo1** deja de aumentar.

- Cuando el tamaño es menor a 90, caerá (la coordenada y disminuye).
- Cuando el tamaño es mayor a 90 y menor a 120, volará hacia el cielo (la coordenada y aumenta).



Si el botón no ha sido presionado, el globo se hace lentamente más grande y cuando el tamaño es mayor a 120, explotará (cambia al disfraz de efecto de explosión).

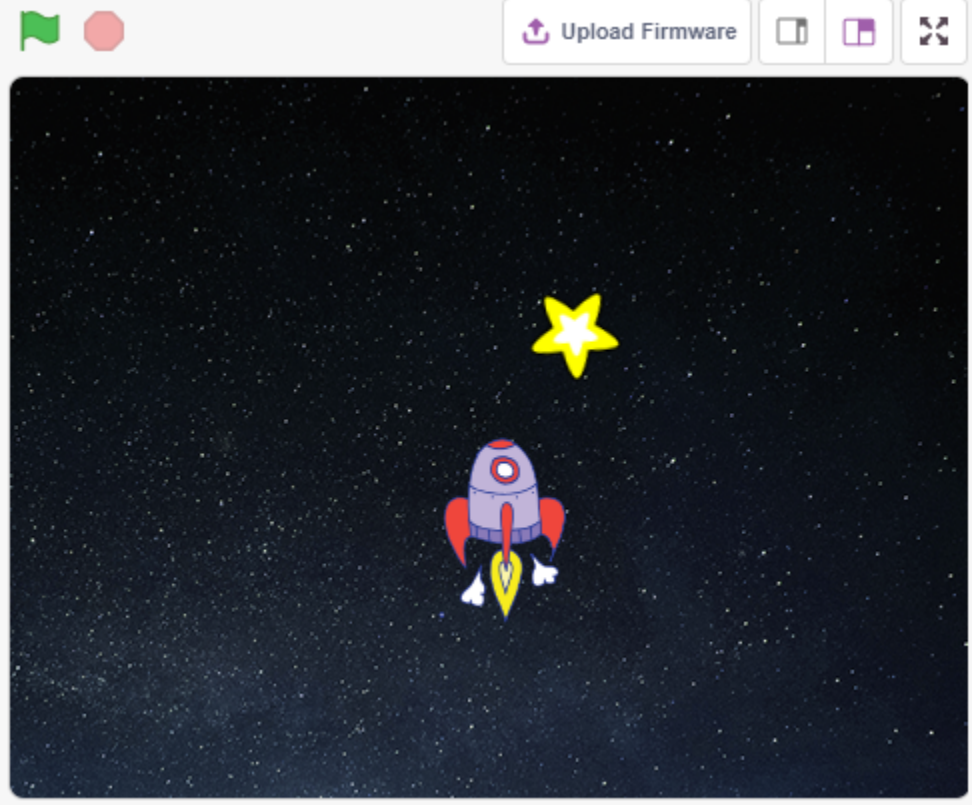


5.16 2.13 JUEGO - Estrellas Cruzadas

En los próximos proyectos, jugaremos algunos minijuegos divertidos en PictoBlox.

Aquí usamos el módulo de Joystick para jugar un juego de Estrellas Cruzadas.

Una vez que se ejecuta el script, las estrellas aparecerán aleatoriamente en el escenario, necesitas usar el Joystick para controlar la Nave Espacial y evitar las estrellas, si las tocas, el juego terminará.



5.16.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Joystick	

5.16.2 Lo Que Aprenderás

- Cómo funciona el módulo de Joystick
- Establecer las coordenadas x y y del sprite

5.16.3 Construye el Circuito

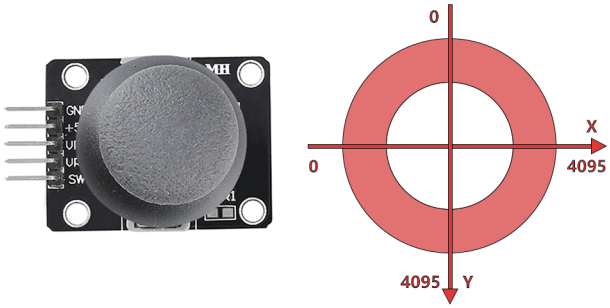
Un joystick es un dispositivo de entrada que consiste en un palo que pivota en una base e informa su ángulo o dirección al dispositivo que está controlando. Los joysticks se utilizan a menudo para controlar videojuegos y robots.

Para comunicar un rango completo de movimiento al ordenador, un joystick necesita medir la posición del palo en dos ejes: el eje X (de izquierda a derecha) y el eje Y (de arriba abajo).

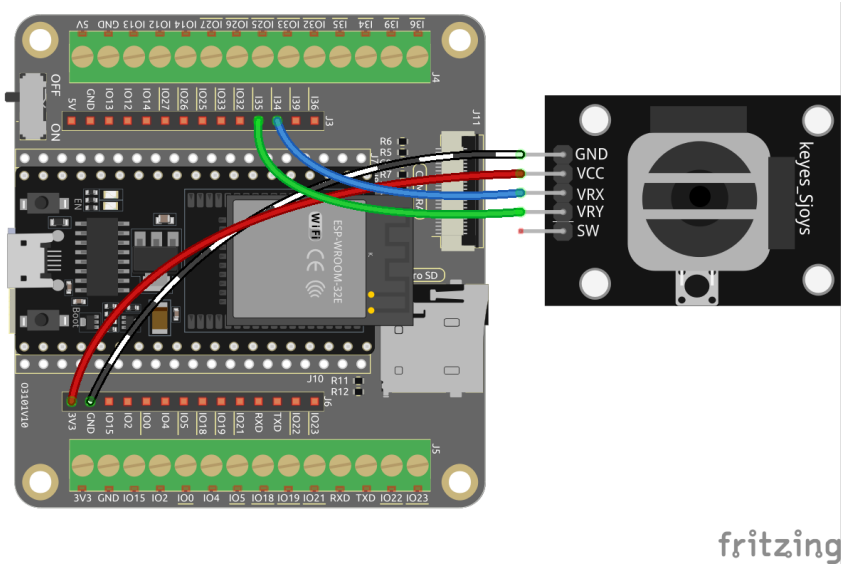
Las coordenadas de movimiento del joystick se muestran en la siguiente figura.

Nota:

- La coordenada x es de izquierda a derecha, el rango es 0-4095.
- La coordenada y es de arriba abajo, el rango es 0-4095.



Ahora construye el circuito según el siguiente diagrama.

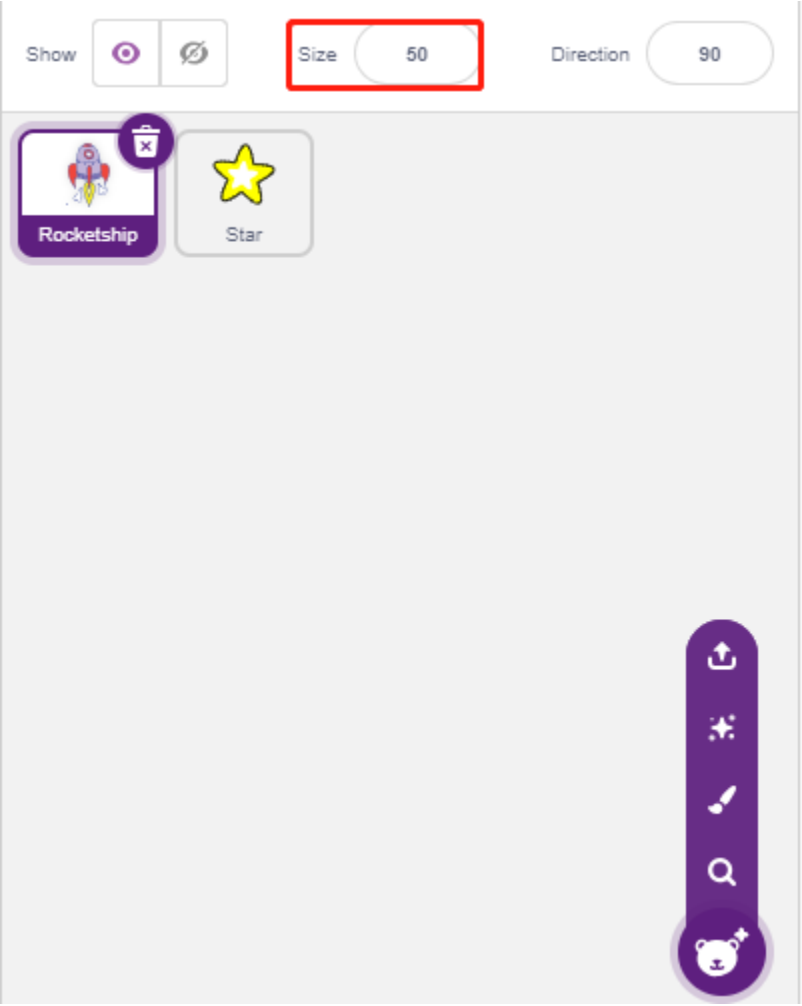


5.16.4 Programación

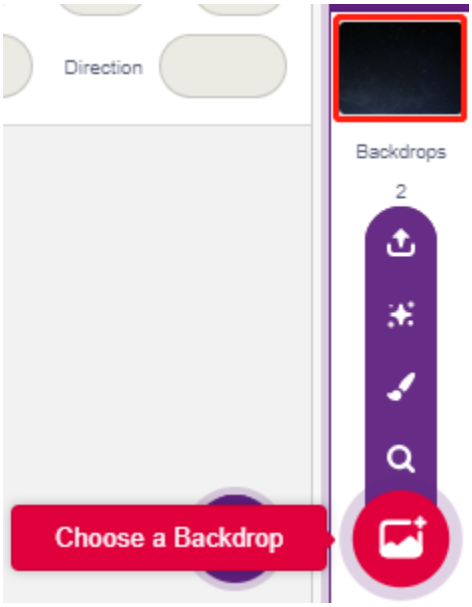
El script completo logra el efecto de que, al hacer clic en la bandera verde, el sprite **Estrellas** se mueve en una curva en el escenario y necesitas usar el joystick para mover la **Nave Espacial**, de modo que no sea tocada por el sprite **Estrella**.

1. Añadir sprites y fondos

Elimina el sprite predeterminado, y usa el botón **Elegir un Sprite** para añadir el sprite **Nave Espacial** y el sprite **Estrella**. Nota que el tamaño del sprite **Nave** se establece al 50 %.



Ahora añade el fondo **Estrellas** mediante **Elegir un Fondo**.

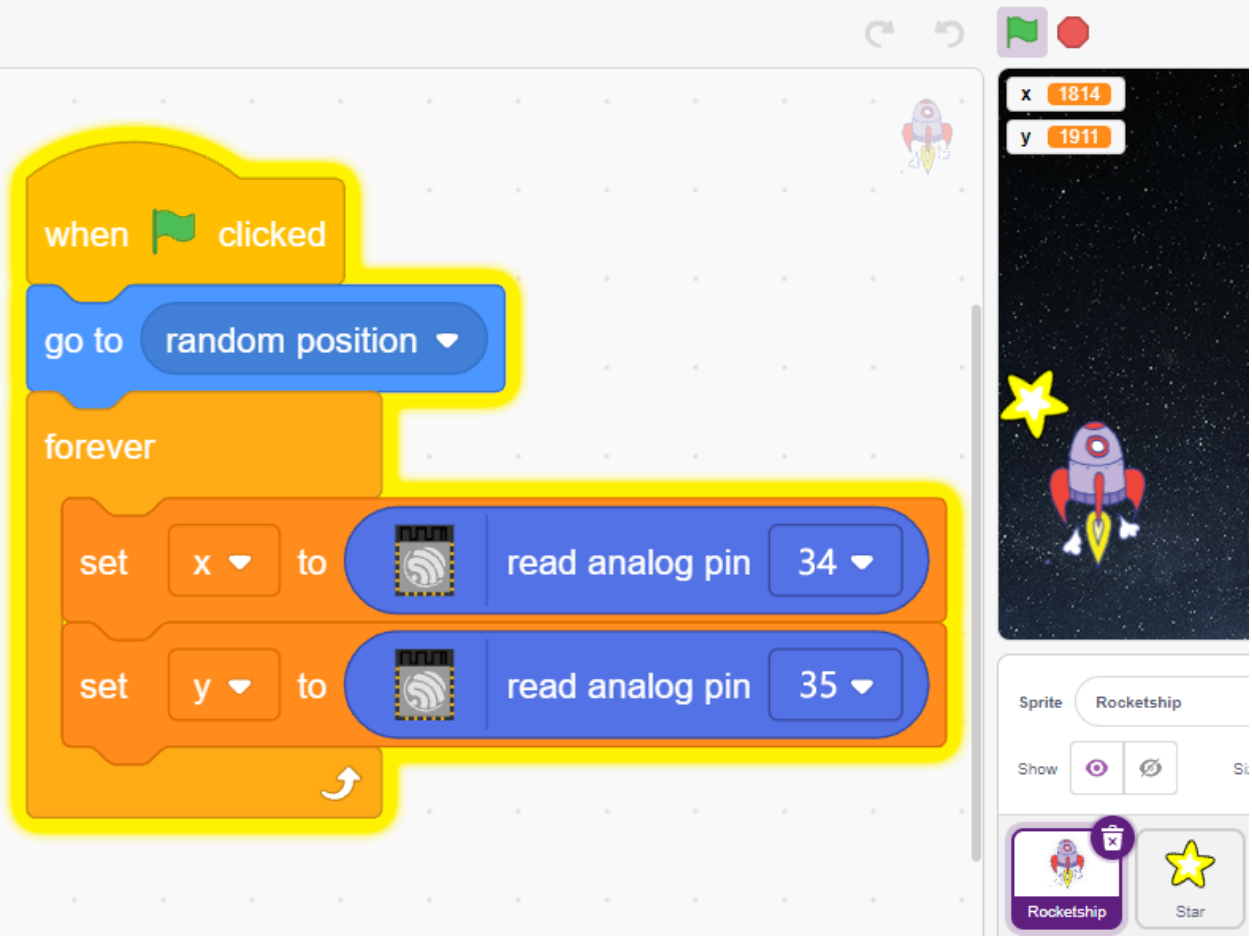


2. Programación para Nave Espacial

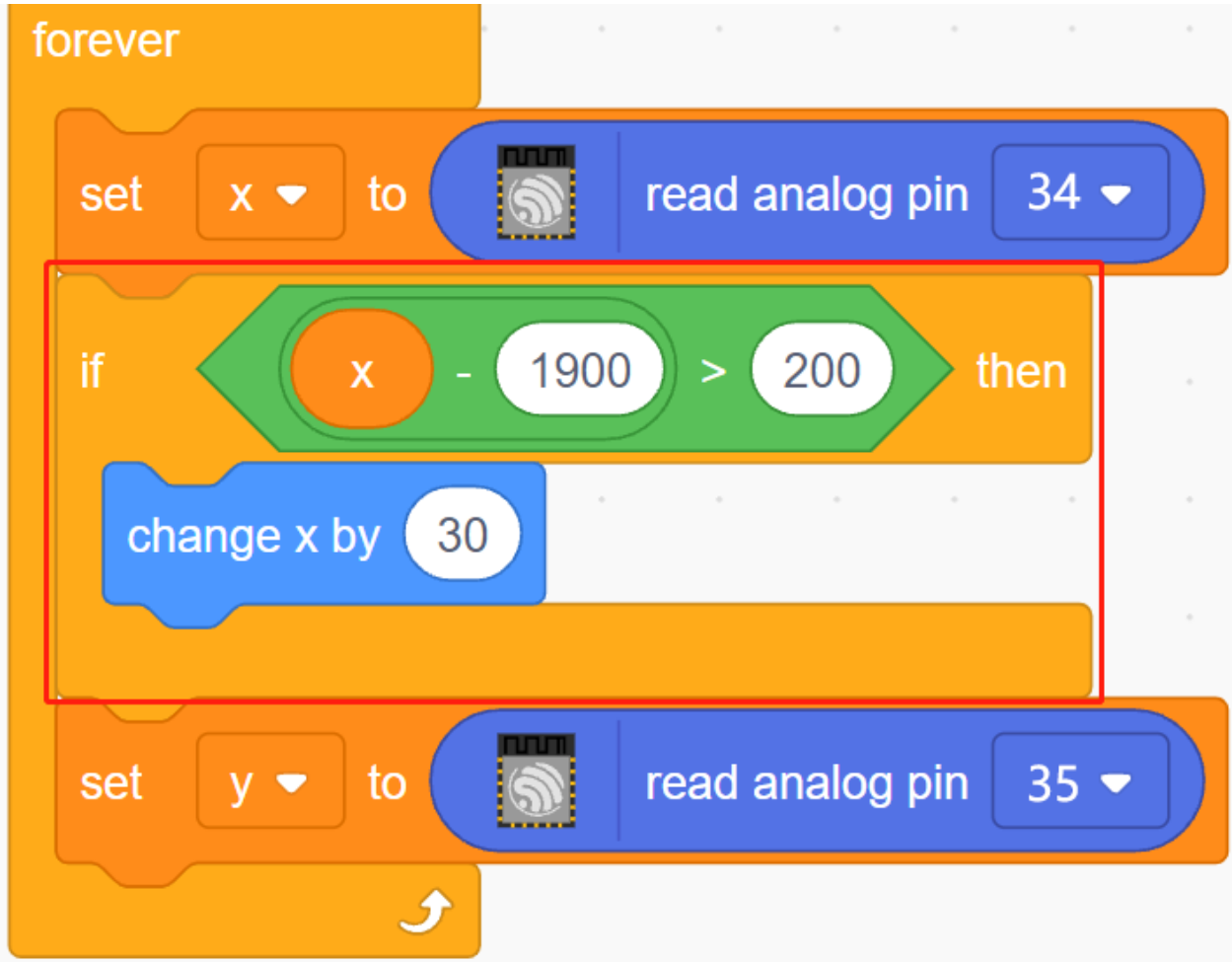
El sprite **Nave Espacial** logra el efecto de aparecer en una posición aleatoria y luego ser controlado por el joystick para moverlo hacia arriba, abajo, izquierda y derecha.

El flujo de trabajo es el siguiente.

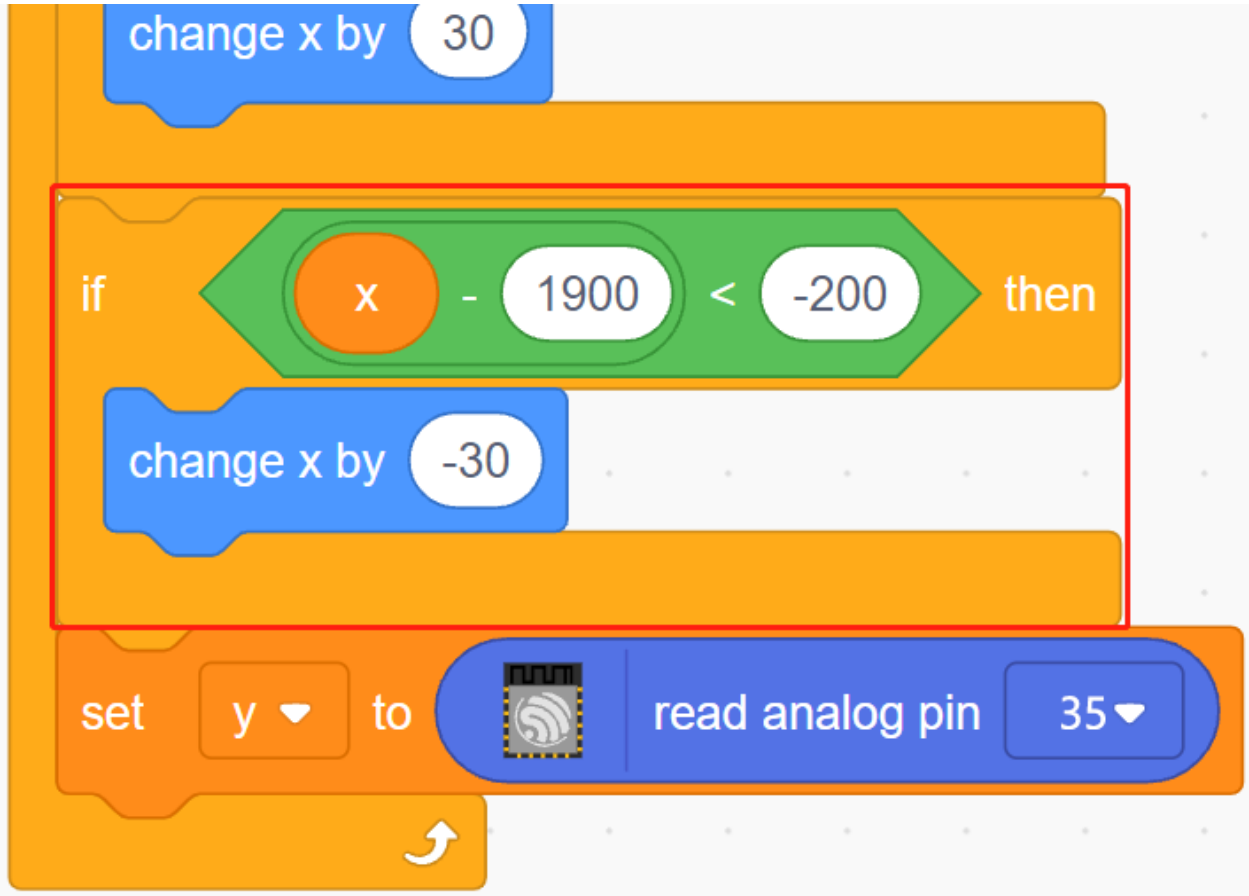
- Al hacer clic en la bandera verde, haz que el sprite vaya a una ubicación aleatoria y crea 2 variables **x** y **y**, que almacenan los valores leídos de pin33 (VRX del Joystick) y pin35 (VRY del Joystick), respectivamente. Puedes dejar que el script se ejecute, moviendo el joystick hacia arriba y hacia abajo, izquierda y derecha, para ver el rango de valores para x y y.



- El valor de pin33 está en el rango 0-4095 (el medio es aproximadamente 1800). Usa `x-1800>200` para determinar si el Joystick se está moviendo hacia la derecha, y si es así, hacer que la coordenada x del sprite +30 (para mover el sprite hacia la derecha).



- Si el Joystick se mueve hacia la izquierda, deja que la coordenada x del sprite sea -30 (para mover el sprite hacia la izquierda).



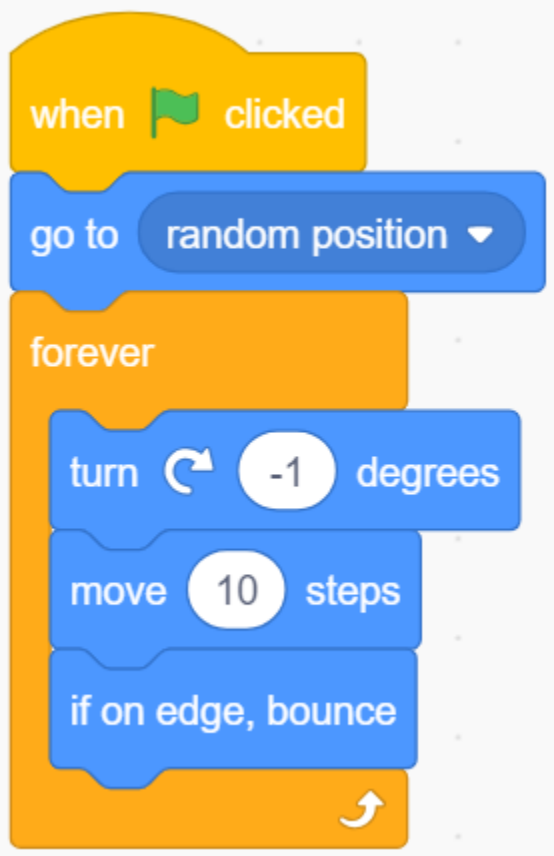
- Dado que la coordenada y del Joystick es de arriba (0) a abajo (4095), y la coordenada y del sprite es de abajo hacia arriba. Entonces, para mover el Joystick hacia arriba y el sprite hacia arriba, la coordenada y debe ser -30 en el script.

- Si el joystick se mueve hacia abajo, la coordenada y del sprite es +30.

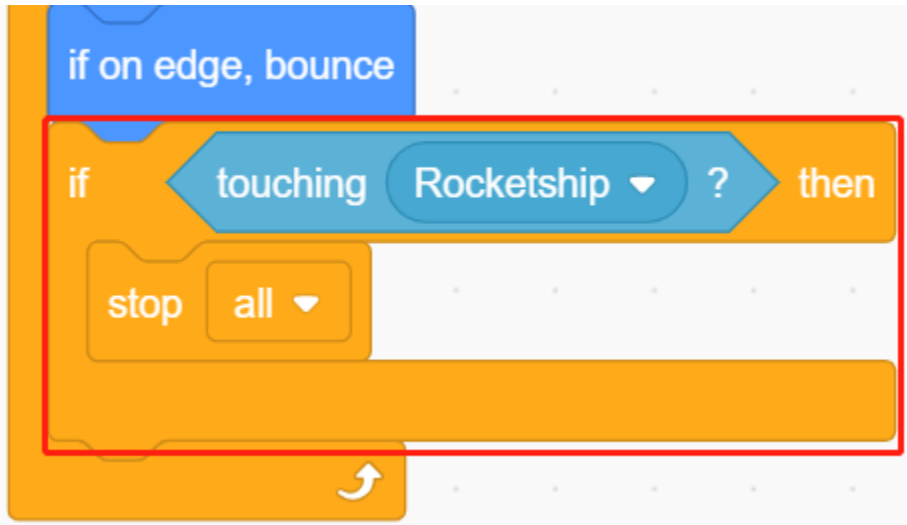
3. Programación para Estrella

El efecto a lograr por el sprite **Estrella** es aparecer en una ubicación aleatoria, y si golpea **Nave Espacial**, el script deja de ejecutarse y el juego termina.

- Al hacer clic en la bandera verde y el sprite va a una ubicación aleatoria, el bloque [girar grados] es para hacer que el sprite **Estrella** se mueva hacia adelante con un cambio de ángulo para que puedas ver que se está moviendo en una curva y si toca el borde, rebota.



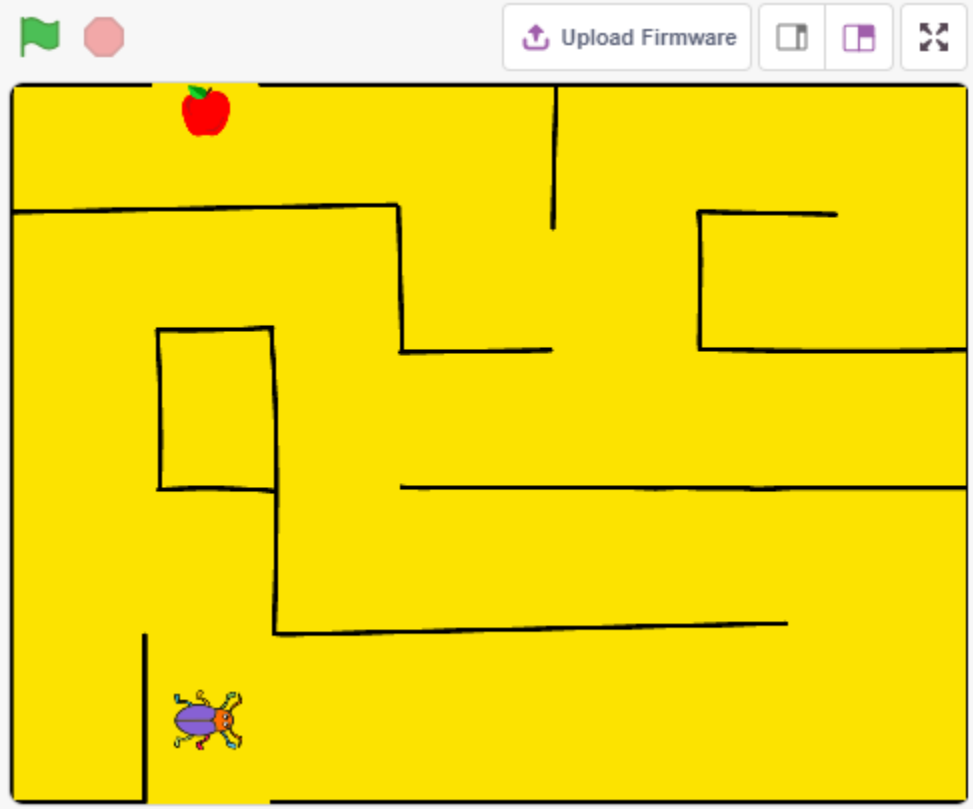
- Si el sprite toca el sprite **Nave Espacial** mientras se mueve, detén la ejecución del script.



5.17 2.14 JUEGO - Comer Manzana

En este proyecto, jugaremos un juego que utiliza un botón para controlar a un Escarabajo para que coma manzanas.

Al hacer clic en la bandera verde, presiona el botón y el Escarabajo girará, presiona el botón nuevamente y el Escarabajo se detiene y avanza en ese ángulo. Necesitas controlar el ángulo del Escarabajo para que avance sin tocar la línea negra en el mapa hasta que coma la manzana. Si toca la línea negra, el juego termina.



5.17.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ARTÍCULOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

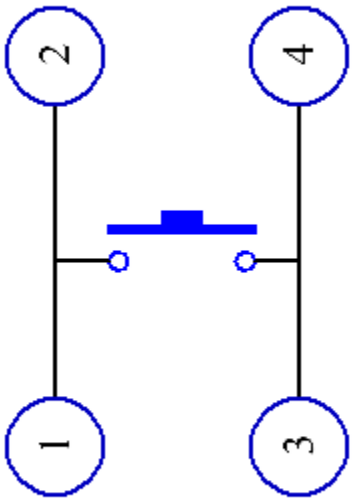
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Botón	

5.17.2 Construye el Circuito

El botón es un dispositivo de 4 pines, ya que el pin 1 está conectado al pin 2, y el pin 3 al pin 4, cuando se presiona el botón, los 4 pines se conectan, cerrando así el circuito.

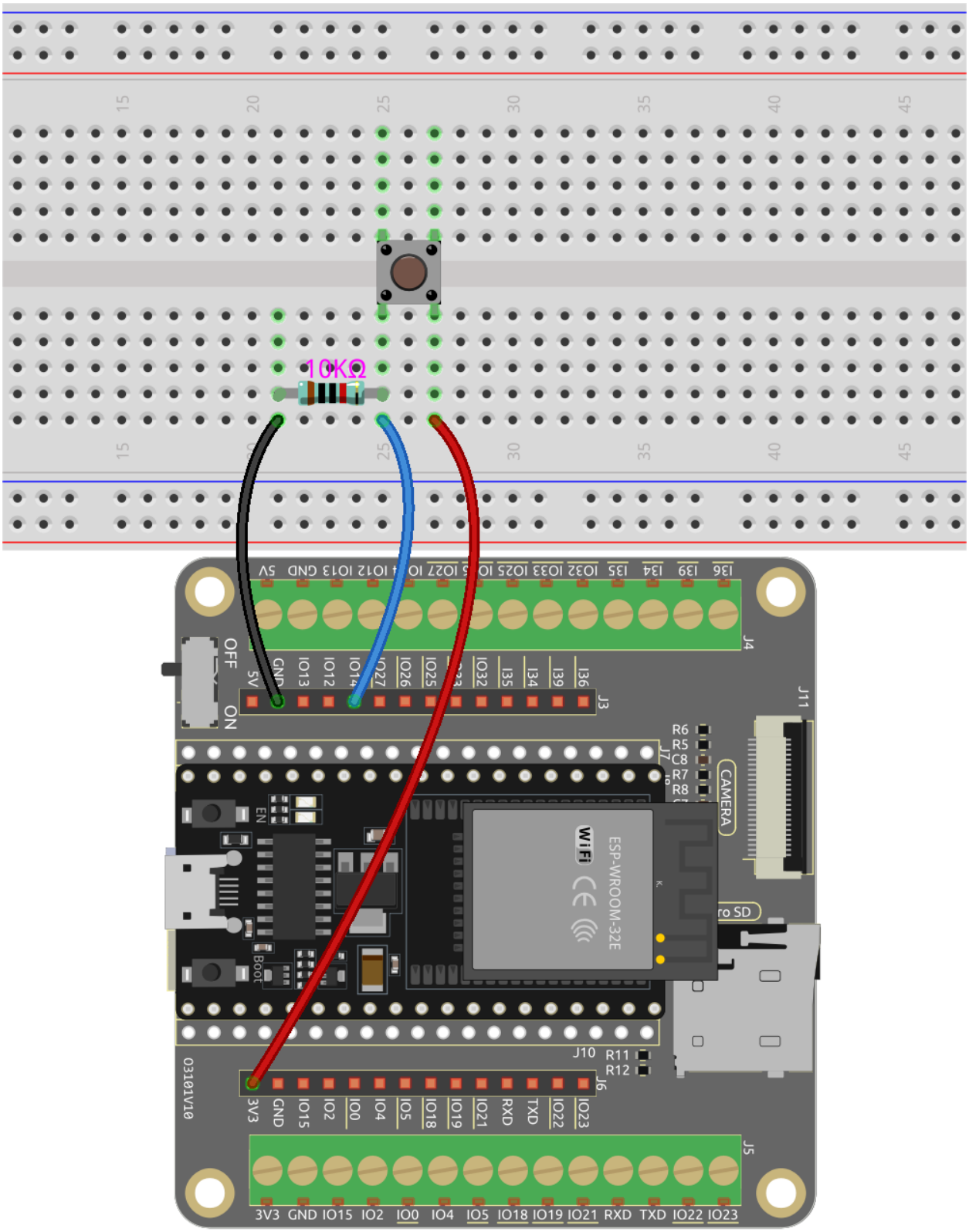
Button



Internal Structure

Construye el circuito según el siguiente diagrama.

- Conecta uno de los pines en el lado izquierdo del botón al pin14, que está conectado a una resistencia de pull-down y un capacitor de 0.1uF (104) (para eliminar la fluctuación y emitir un nivel estable cuando el botón está funcionando).
- Conecta el otro extremo de la resistencia y el capacitor a GND, y uno de los pines en el lado derecho del botón a 5V.


10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

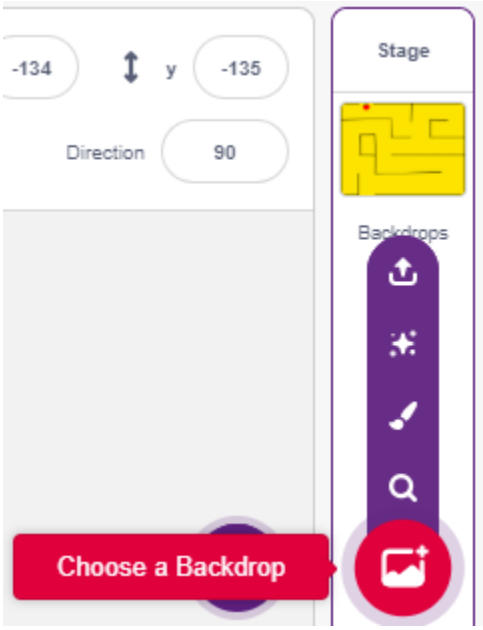
5.17.3 Programación

El efecto que queremos lograr es usar el botón para controlar la dirección del sprite **Escarabajo** para moverse hacia adelante y comer la manzana sin tocar la línea negra en el fondo **Laberinto**, lo cual cambiará el fondo al ser comido.

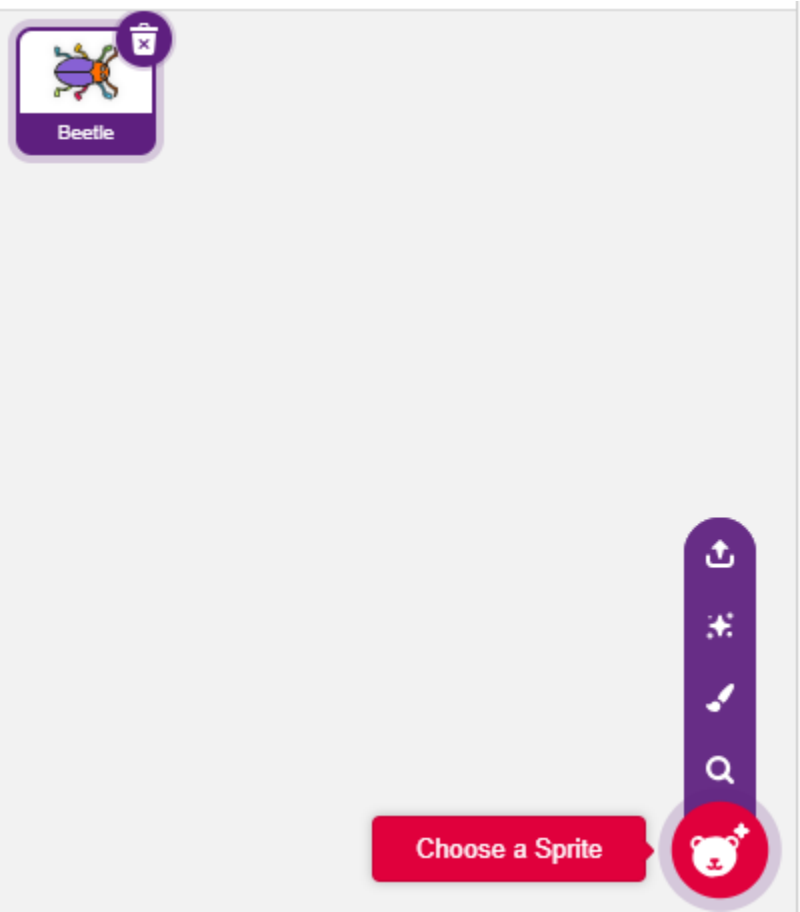
Ahora añade los fondos y sprites relevantes.

1. Añadiendo fondos y sprites

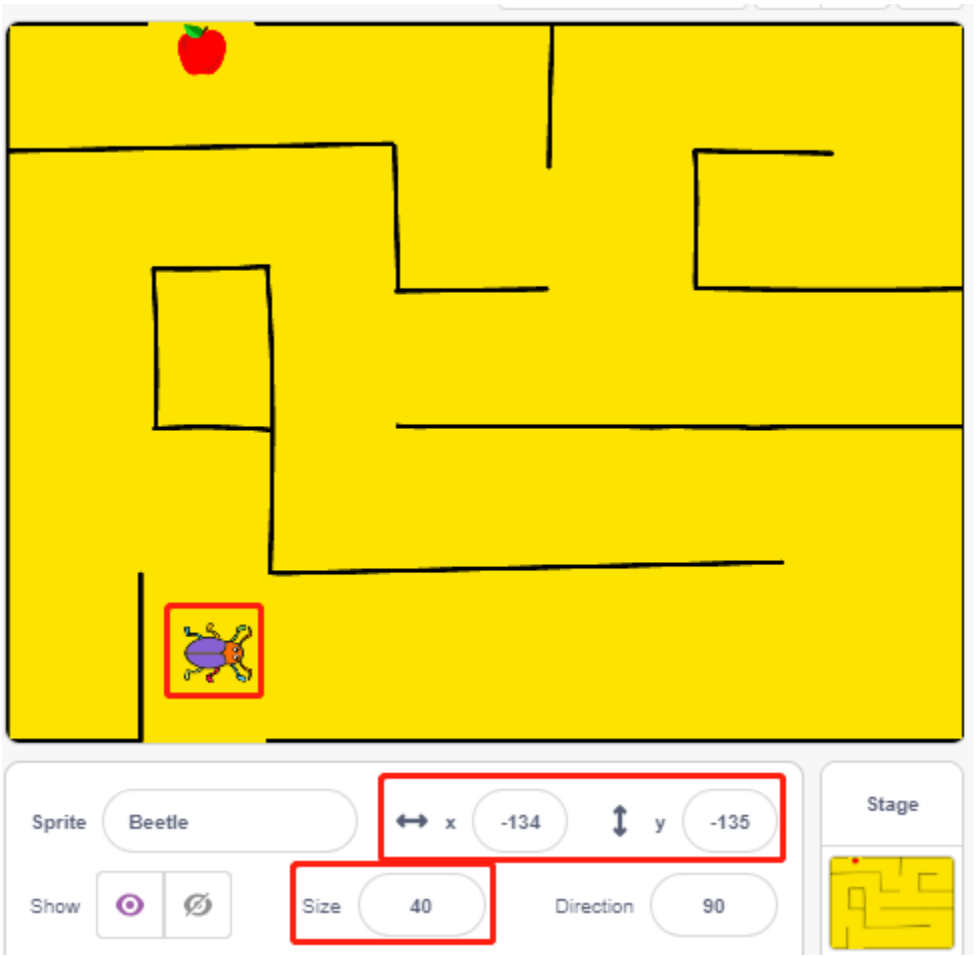
Añade un fondo **Laberinto** mediante el botón **Elegir un fondo**.



Elimina el sprite predeterminado, luego selecciona el sprite **Escarabajo**.



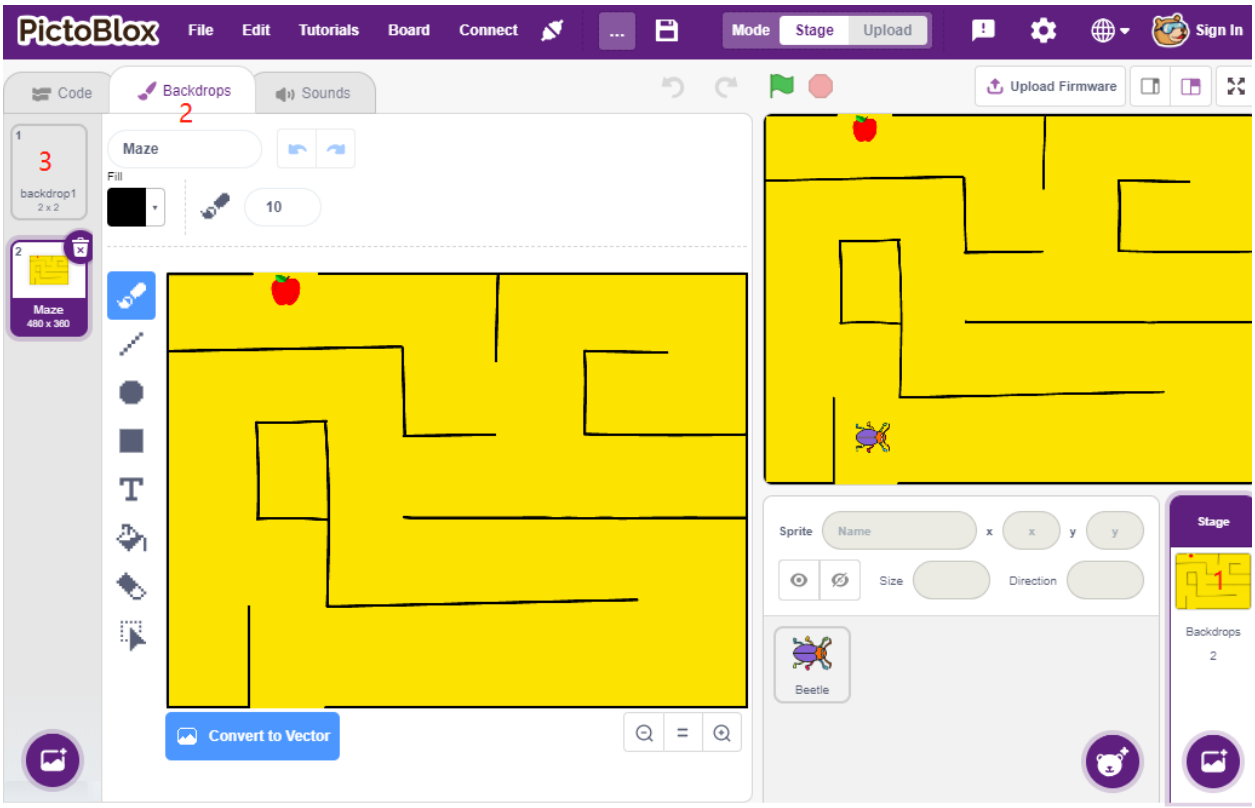
Coloca el sprite **Escarabajo** en la entrada del fondo **Laberinto**, recordando los valores de coordenadas x,y en este punto, y redimensiona el sprite al 40 %.



2. Dibujar un fondo

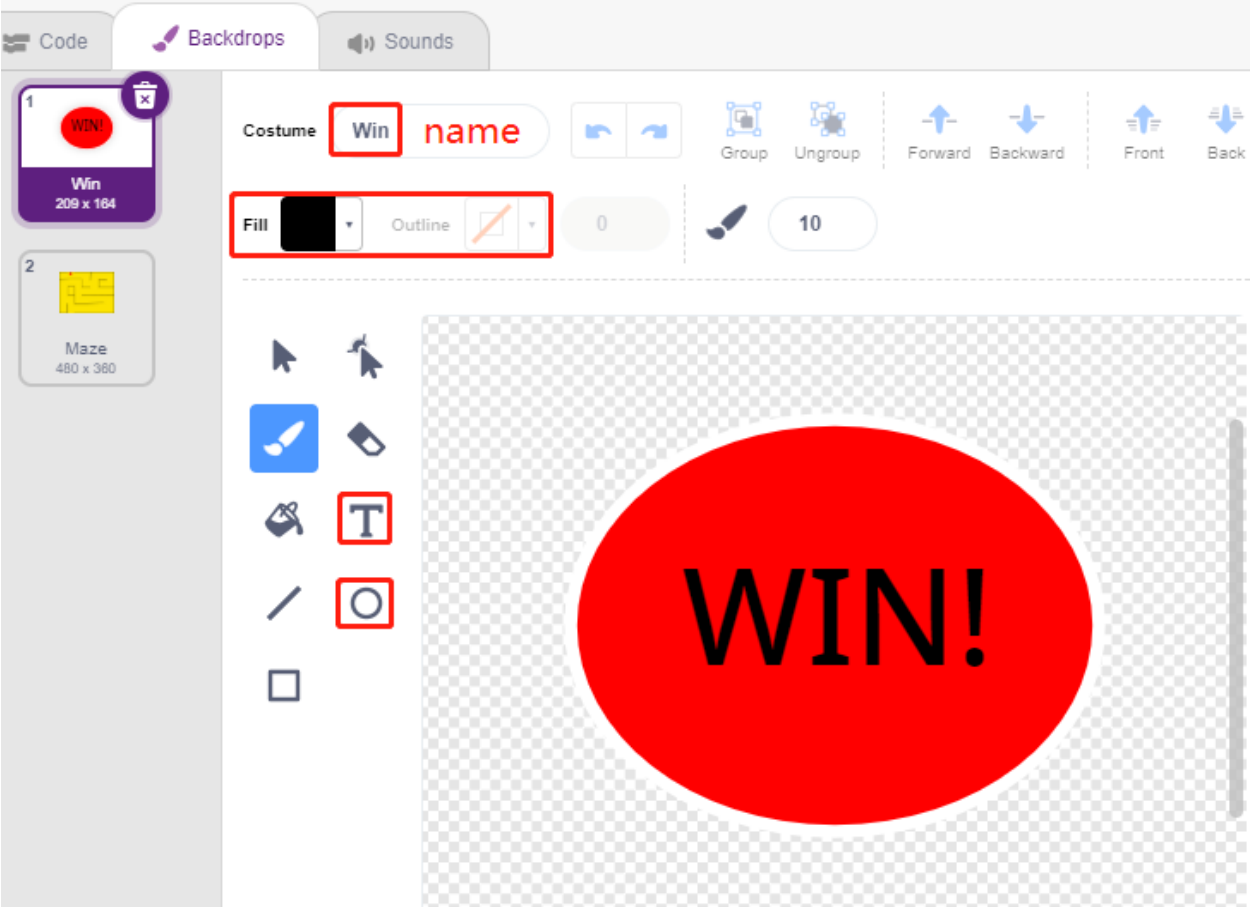
Ahora es el momento de dibujar simplemente un fondo con el personaje WIN! apareciendo en él.

Primero haz clic en la miniatura del fondo para ir a la página **Fondos** y haz clic en el fondo en blanco backdrop1.



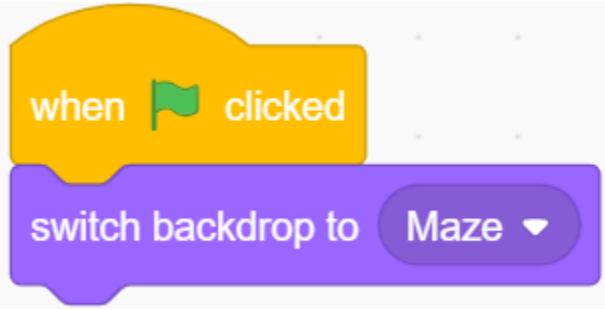
Ahora comienza a dibujar, puedes referirte a la imagen de abajo para dibujar, o también puedes crear tu propio fondo, siempre y cuando la expresión transmita victoria.

- Utilizando la herramienta **Círculo**, dibuja una elipse con el color establecido en rojo y sin contorno.
- Luego, con la herramienta **Texto**, escribe el carácter "¡GANASTE!", establece el color del carácter en negro y ajusta el tamaño y la posición del carácter.
- Nombra el fondo como **Ganar**.



3. Programación para el fondo

El fondo debe cambiarse a **Laberinto** cada vez que comience el juego.



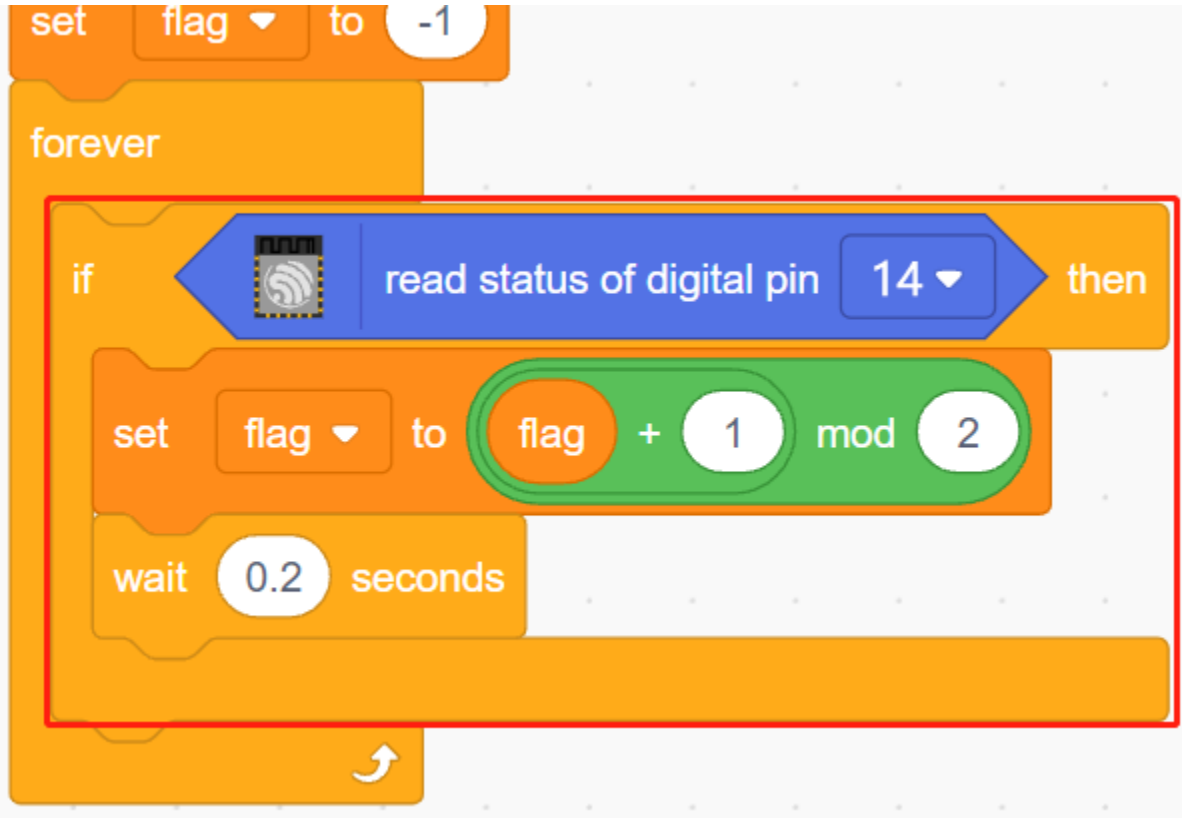
4. Escribir guiones para el sprite Escarabajo

Ahora escribe un guion para el sprite **Escarabajo** para que pueda moverse hacia adelante y cambiar de dirección bajo el control de un botón. El flujo de trabajo es el siguiente.

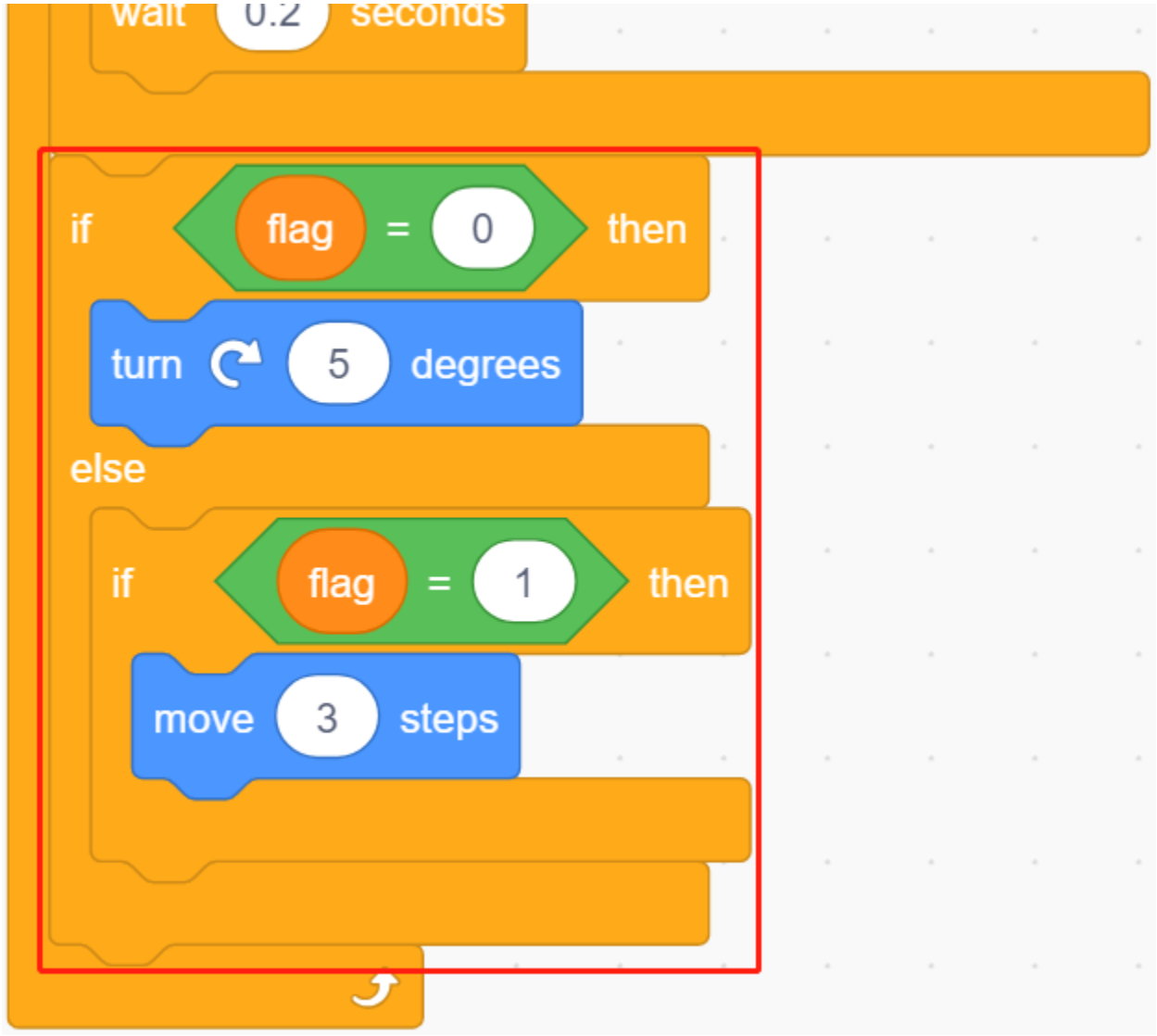
- Al hacer clic en la bandera verde, establece el ángulo del **Escarabajo** a 90, y la posición a (-134, -134), o reemplázalo con el valor de coordenadas de tu propia posición colocada. Crea la variable **bandera** y establece el valor inicial en -1.

A continuación, en el bloque [siempre], se utilizan cuatro bloques [si] para determinar varios escenarios posibles.

- Si el botón es 1 (presionado), usa el bloque [mod] para alternar el valor de la variable **bandera** entre 0 y 1 (alternando entre 0 para esta presión y 1 para la próxima presión).

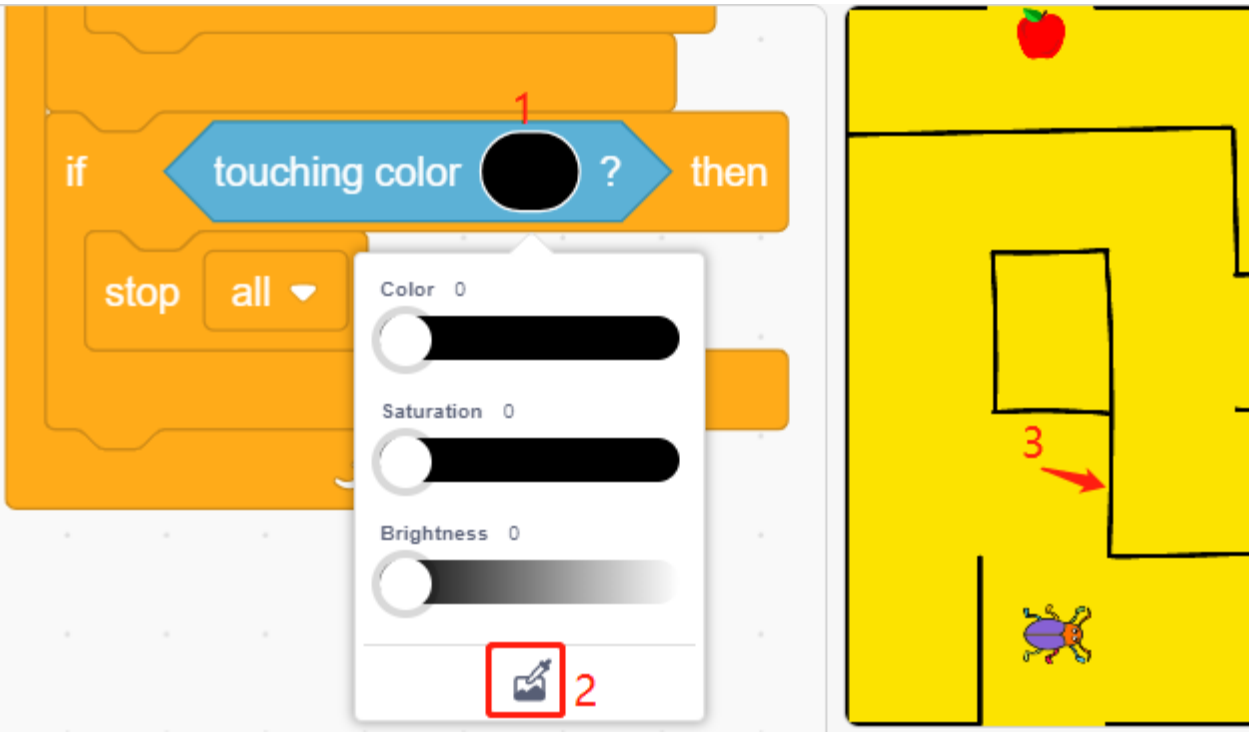


- Si bandera=0 (esta presión de botón), permite que el sprite **Escarabajo** gire en sentido horario. Luego determina si bandera es igual a 1 (botón presionado de nuevo), el sprite **Escarabajo** se mueve hacia adelante. De lo contrario, sigue girando en sentido horario.

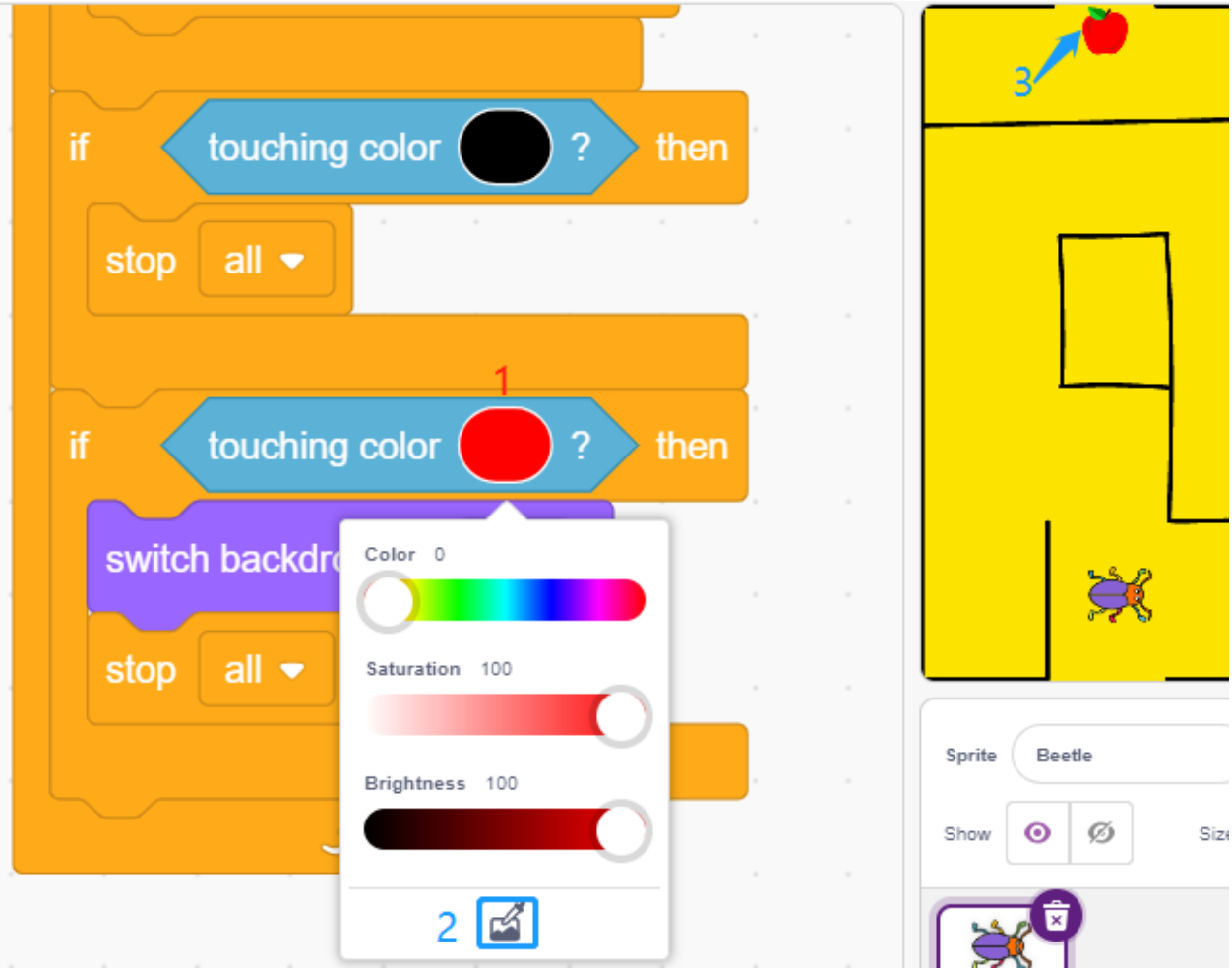


- Si el sprite Escarabajo toca el negro (la línea negra en el fondo **Laberinto**), el juego termina y el guion deja de ejecutarse.

Nota: Necesitas hacer clic en el área de color en el bloque [Tocar color], y luego seleccionar la herramienta cuentagotas para recoger el color de la línea negra en el escenario. Si eliges un negro arbitrariamente, este bloque [Tocar color] no funcionará.



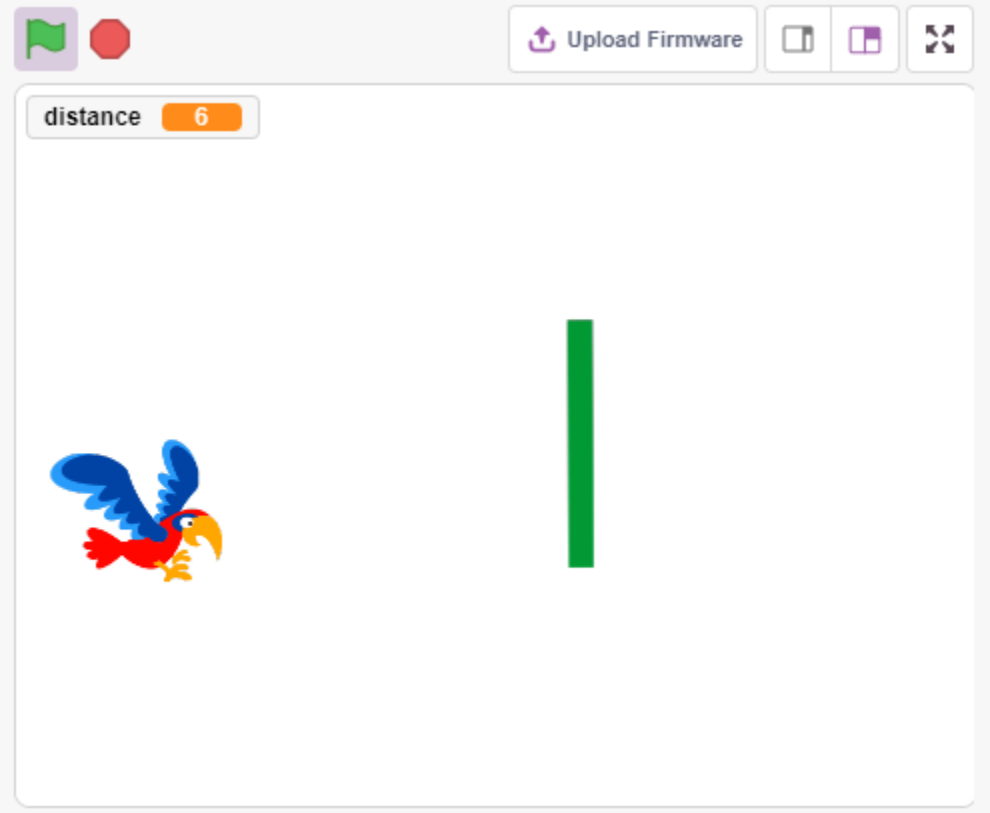
- Si Escarabajo toca rojo (Usa también la herramienta de paja para recoger el color rojo de la manzana), el fondo se cambiará a **Ganar**, lo que significa que el juego tiene éxito y se detiene la ejecución del guion.



5.18 2.15 JUEGO - Loro Flappy

Aquí utilizamos el módulo ultrasónico para jugar un juego del loro flappy.

Después de ejecutar el script, el bambú verde se moverá lentamente de derecha a izquierda a una altura aleatoria. Ahora coloca tu mano sobre el módulo ultrasónico, si la distancia entre tu mano y el módulo ultrasónico es menor a 10, el loro volará hacia arriba, de lo contrario caerá hacia abajo. Necesitas controlar la distancia entre tu mano y el módulo ultrasónico para que el Loro pueda evitar el bambú verde (Paleta), si lo toca, el juego termina.



5.18.1 Componentes Requeridos

En este proyecto, necesitamos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

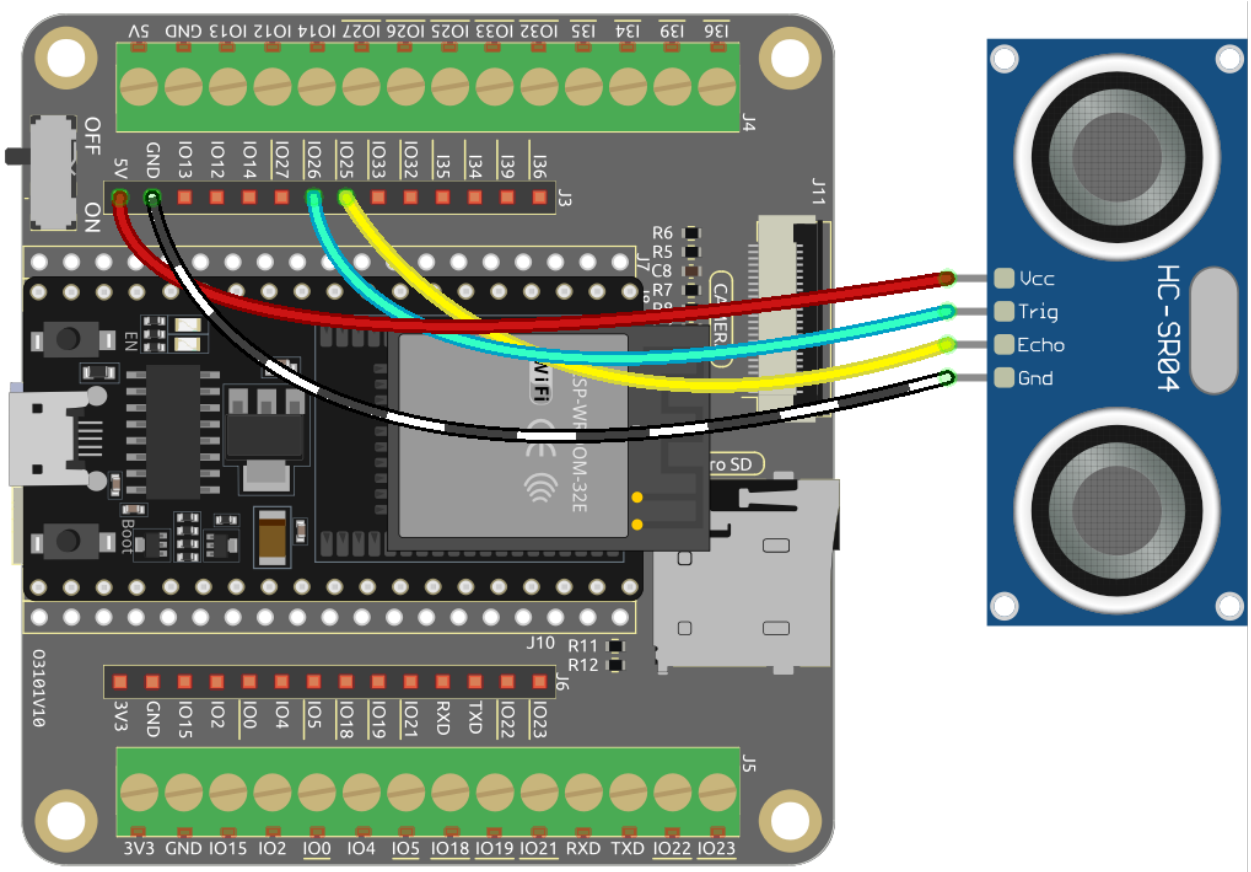
También puedes comprarlos por separado en los siguientes enlaces.

INTRODUCCIÓN DEL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo Ultrasonido	

5.18.2 Construir el Circuito

Un módulo sensor ultrasónico es un instrumento que mide la distancia a un objeto usando ondas sonoras ultrasónicas. Tiene dos sondas. Una es para enviar ondas ultrasónicas y la otra es para recibir las ondas y transformar el tiempo de envío y recepción en una distancia, detectando así la distancia entre el dispositivo y un obstáculo.

Ahora construye el circuito según el siguiente diagrama.

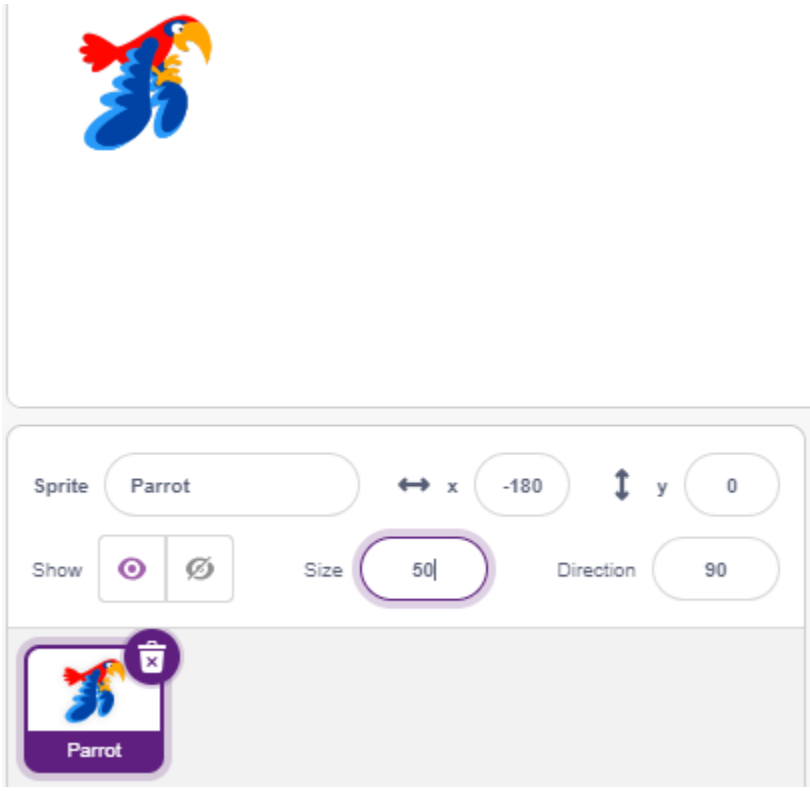


5.18.3 Programación

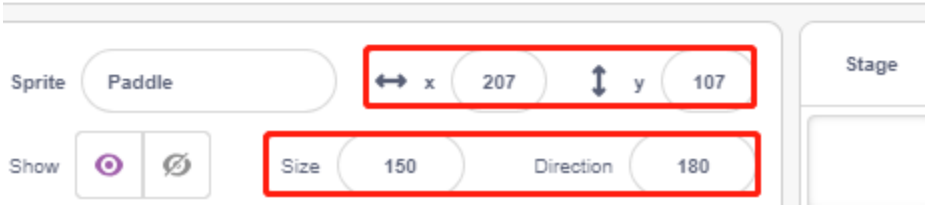
El efecto que queremos lograr es usar el módulo ultrasónico para controlar la altura de vuelo del sprite **Loro**, evitando al mismo tiempo el sprite **Paleta**.

1. Agregar un sprite

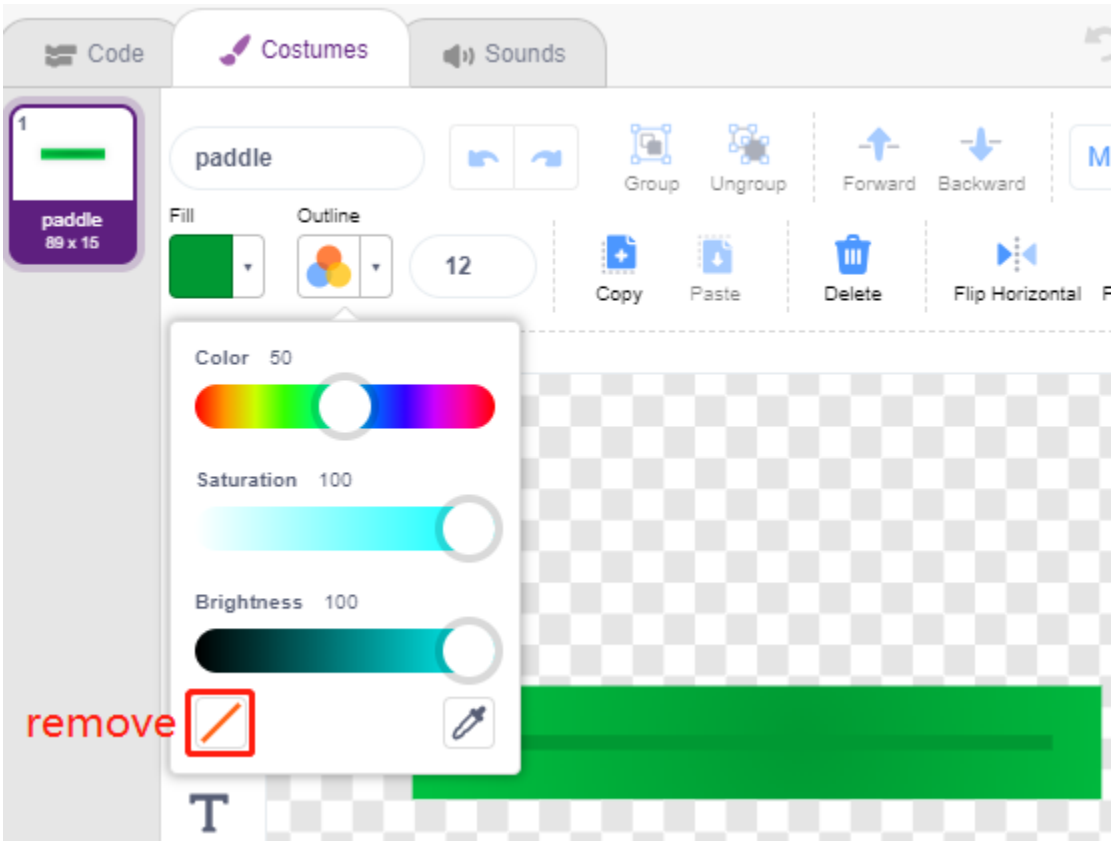
Elimina el sprite predeterminado y utiliza el botón **Elegir un Sprite** para agregar el sprite **Loro**. Establece su tamaño al 50 % y mueve su posición al centro izquierdo.



Ahora agrega el sprite **Paleta**, establece su tamaño al 150 %, su ángulo a 180 y mueve su posición inicial a la esquina superior derecha.



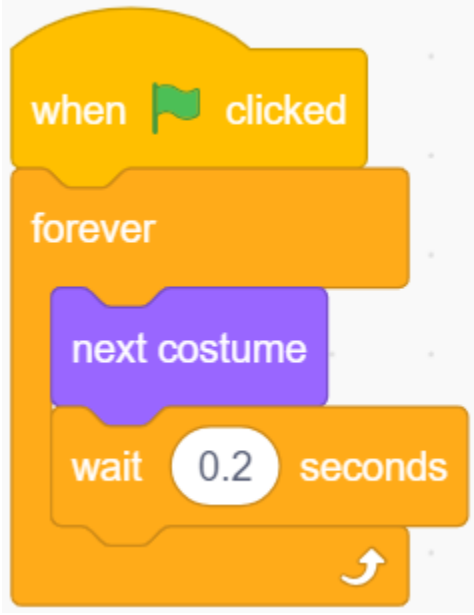
Ve a la página **Disfraces** del sprite **Paleta** y elimina el Contorno.



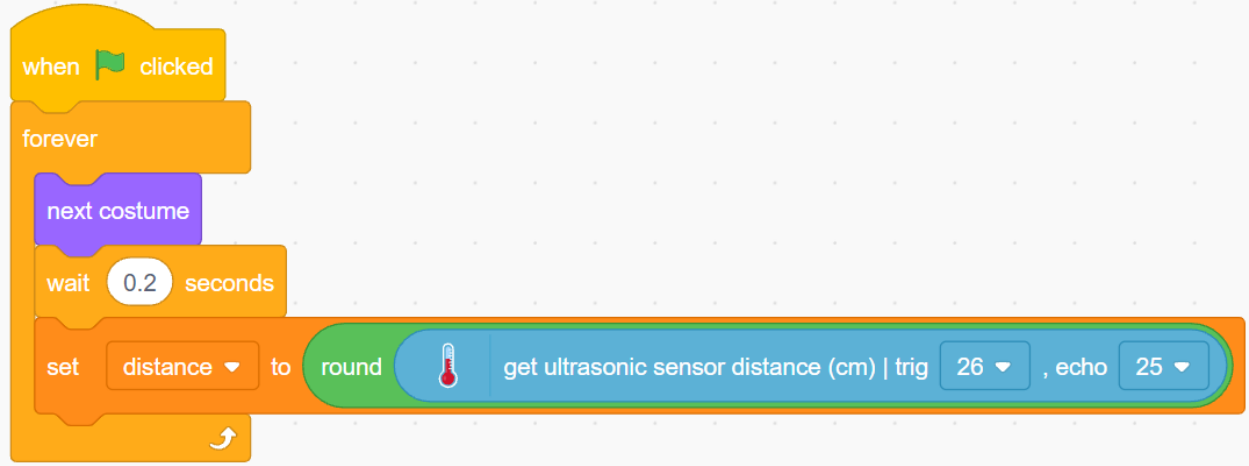
2. Programación para el Sprite Loro

Ahora programa el sprite **Loro**, que está en vuelo y la altitud de vuelo está determinada por la distancia de detección del módulo ultrasónico.

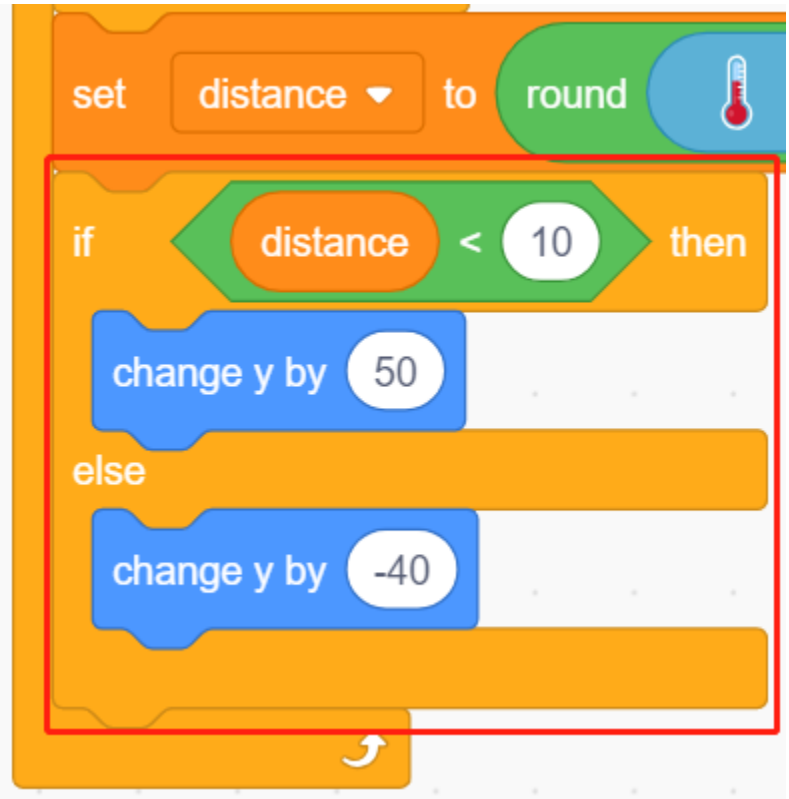
- Cuando se hace clic en la bandera verde, cambia el disfraz cada 0.2s para que siempre esté en vuelo.



- Lee el valor del módulo ultrasónico y almacénalo en la variable **distancia** después de redondearlo con el bloque [redondear].



- Si la distancia de detección ultrasónica es menor a 10cm, deja que la coordenada y aumente en 50, el sprite **Loro** volará hacia arriba. De lo contrario, el valor de la coordenada y disminuye en 40, **Loro** caerá.

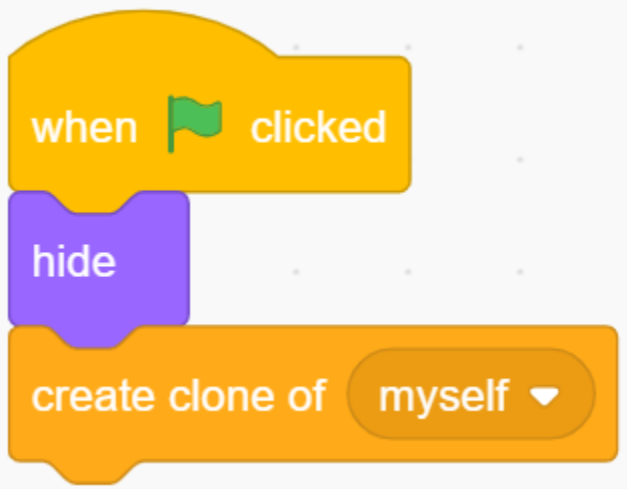


- Si el sprite **Loro** toca el sprite **Paleta**, el juego termina y el script deja de ejecutarse.

3. Scripting para el sprite Paddle

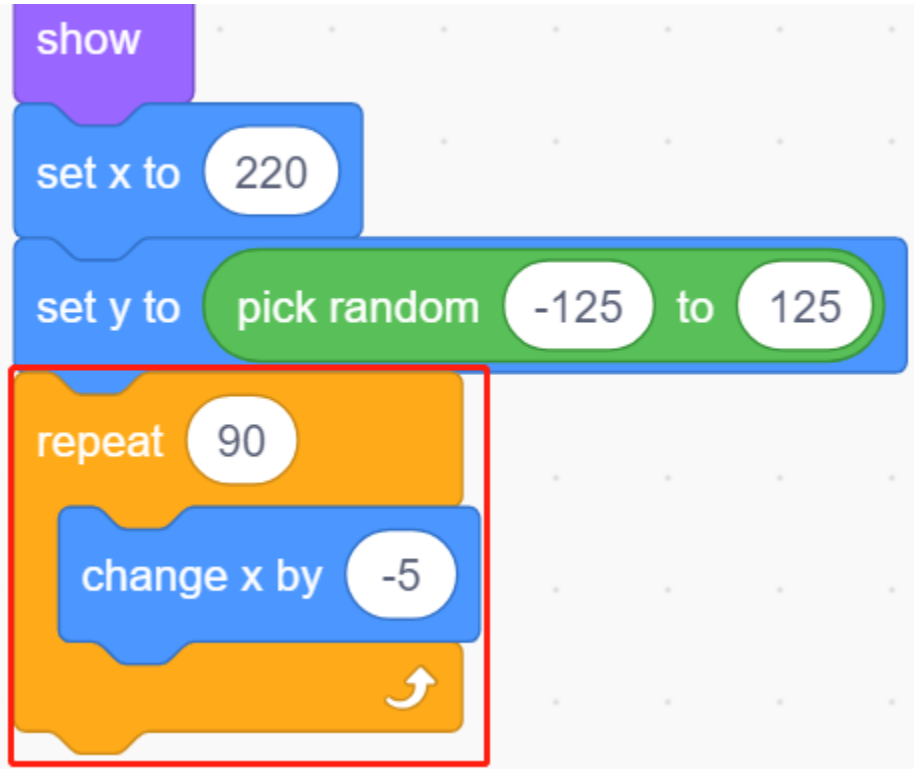
Ahora escribe el script para el sprite **Paddle**, que necesita aparecer aleatoriamente en el escenario.

- Oculta el sprite **Paddle** cuando se haga clic en la bandera verde y clónalo al mismo tiempo. El bloque [*crear clon de <https://en.scratch-wiki.info/wiki/Create_Clone_of_()_(block)>*] es un bloque de control y un bloque de pila. Crea un clon del sprite en el argumento. También puede clonar el sprite en el que se está ejecutando, creando clones de clones, recursivamente.

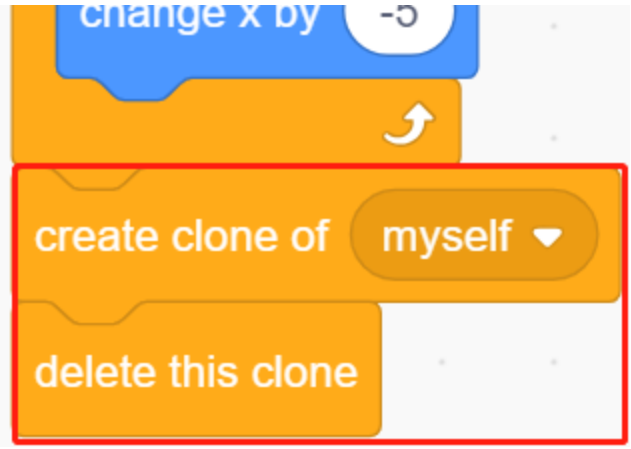


- Cuando **Paddle** se presenta como un clon, su posición es 220 (el más a la derecha) para la coordenada x y su coordenada y en (-125 a 125) aleatorio (altura aleatoria).

- Utiliza el bloque [repetir] para hacer que el valor de su coordenada x disminuya lentamente, para que puedas ver el clon del sprite **Paddle** moviéndose lentamente de la derecha hacia la izquierda hasta que desaparezca.



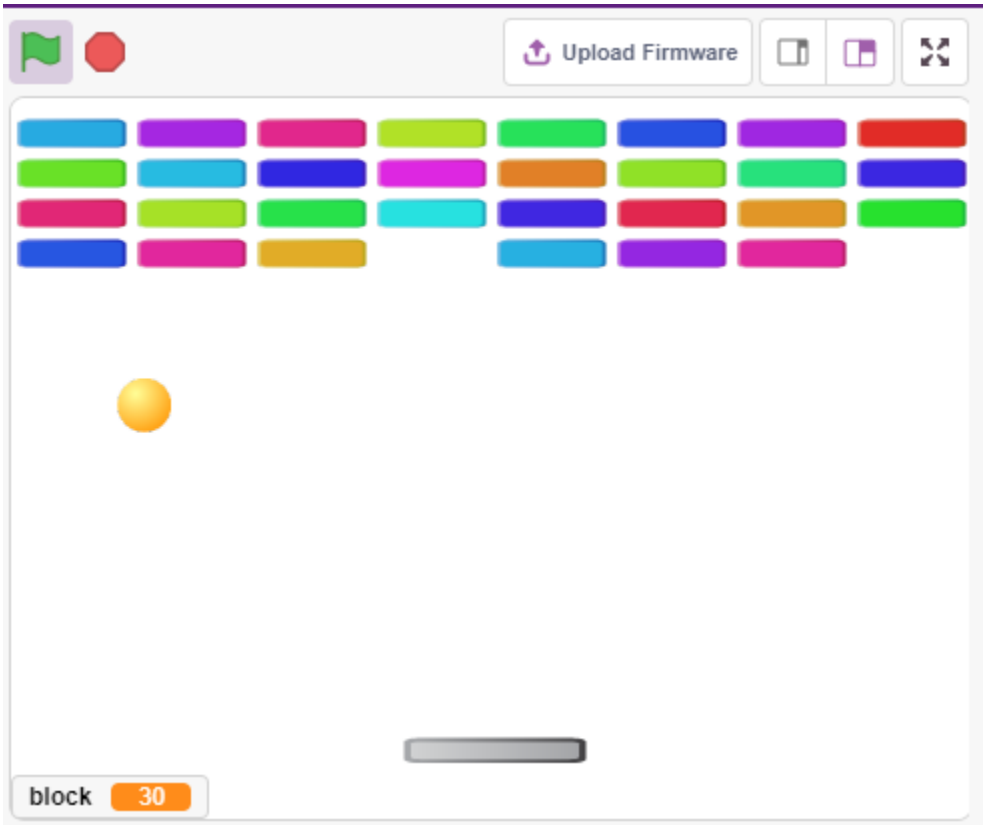
- Reclona un nuevo sprite **Paddle** y elimina el clon anterior.



5.19 2.16 JUEGO - Clon de Breakout

Aquí utilizamos el potenciómetro para jugar a un clon del juego Breakout.

Después de hacer clic en la bandera verde, necesitarás usar el potenciómetro para controlar la paleta en el escenario y atrapar la bola para que pueda subir y golpear los ladrillos. Si desaparecen todos los ladrillos, ganas el juego; si no atrapas la bola, pierdes.



5.19.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

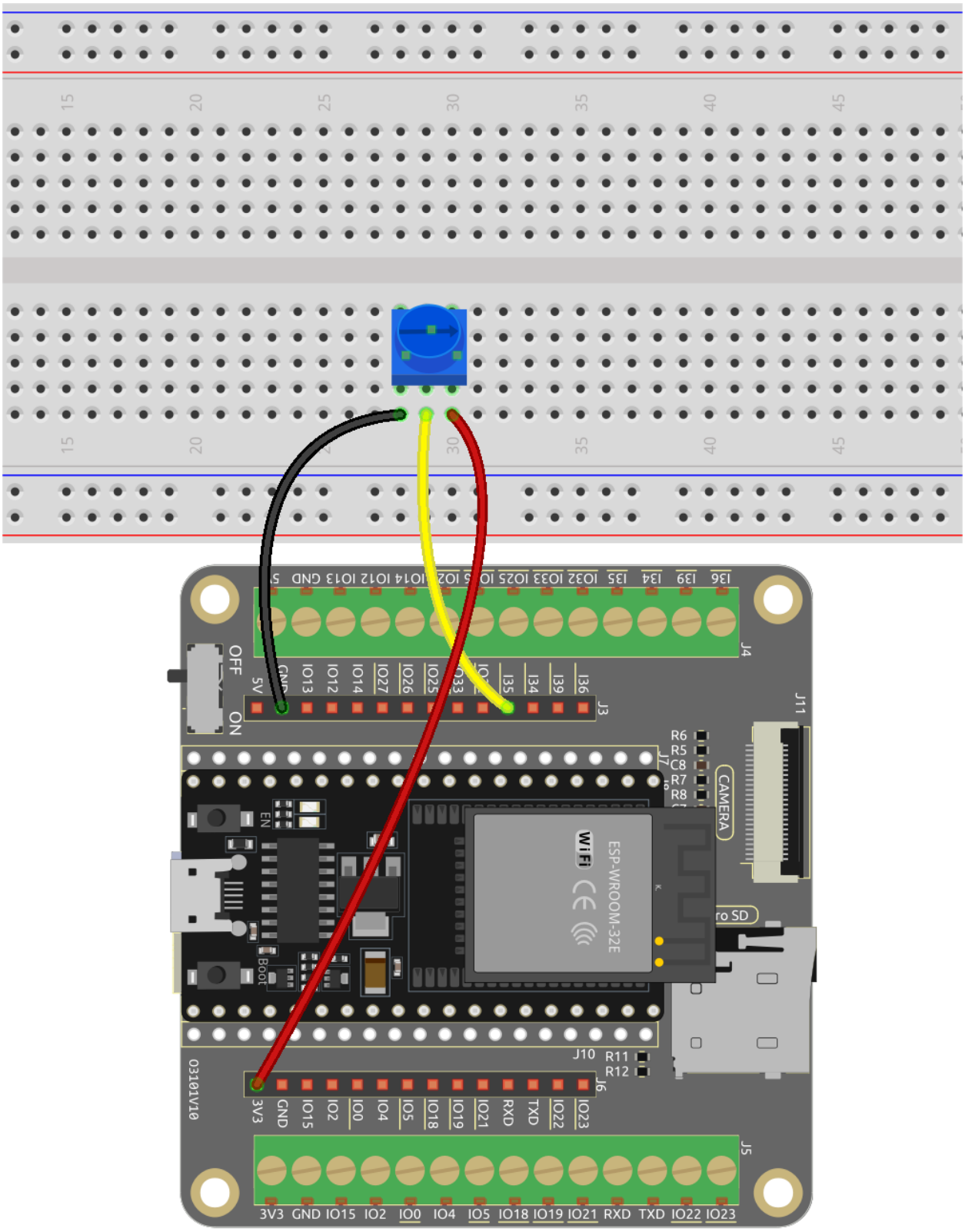
Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Potenciómetro	

5.19.2 Construir el Circuito

El potenciómetro es un elemento resistivo con 3 terminales, los 2 pines laterales están conectados a 5V y GND, y el pin central está conectado al pin35. Después de la conversión por el convertidor ADC de la placa esp32, el rango de valores es de 0-4095.


OFF
ON
5V
GND
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
I35
I34
I39
I36
J4
J3
J11
R6
R5
C8
R7
R8
J7
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
J10
R11
R12
J6
3V3
GND
IO15
IO2
IO0
IO4
IO5
IO18
IO19
IO21
RXD
TXD
IO22
IO23
J5
03101V10

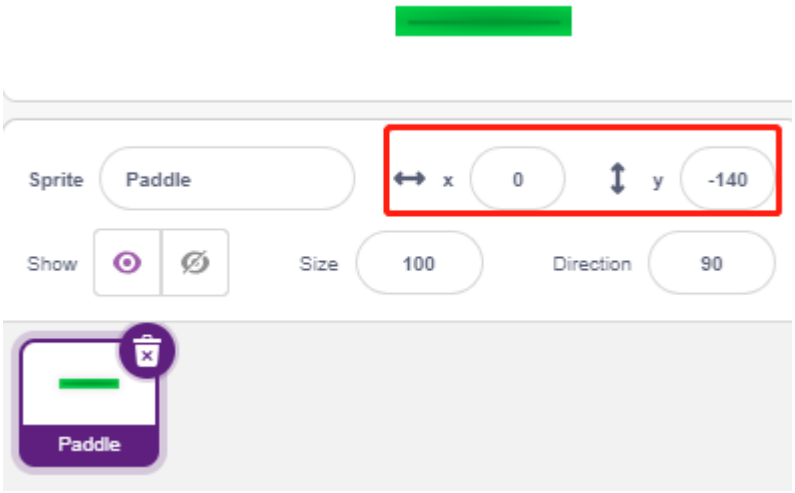
5.19.3 Programación

Hay 3 sprites en el escenario.

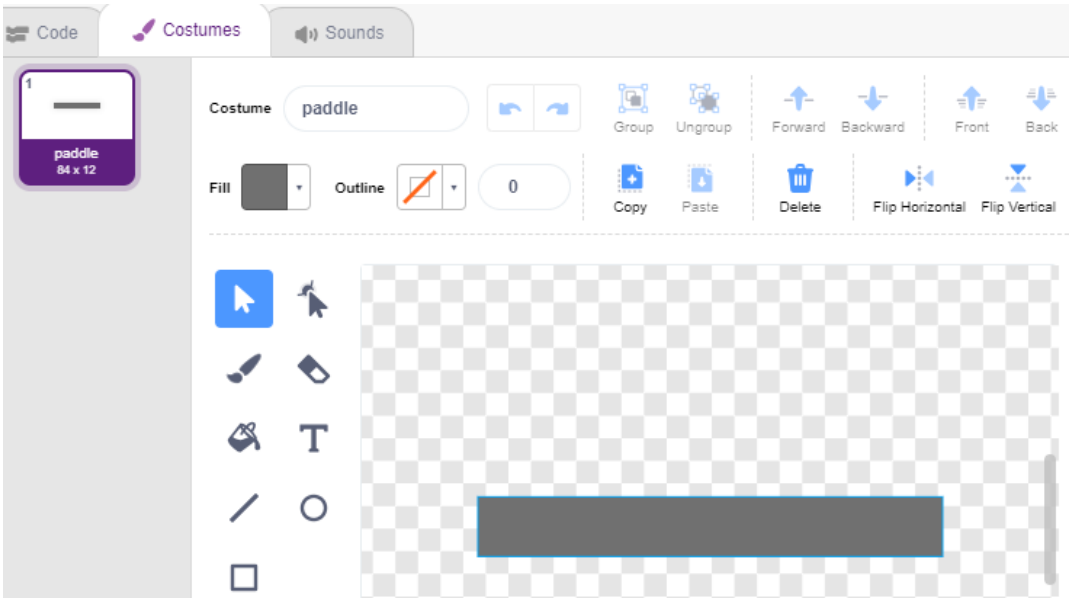
1. Sprite de la Paleta

El efecto que se busca con la **Paleta** es que la posición inicial esté en el medio de la parte inferior del escenario, y sea controlada por un potenciómetro para moverla hacia la izquierda o hacia la derecha.

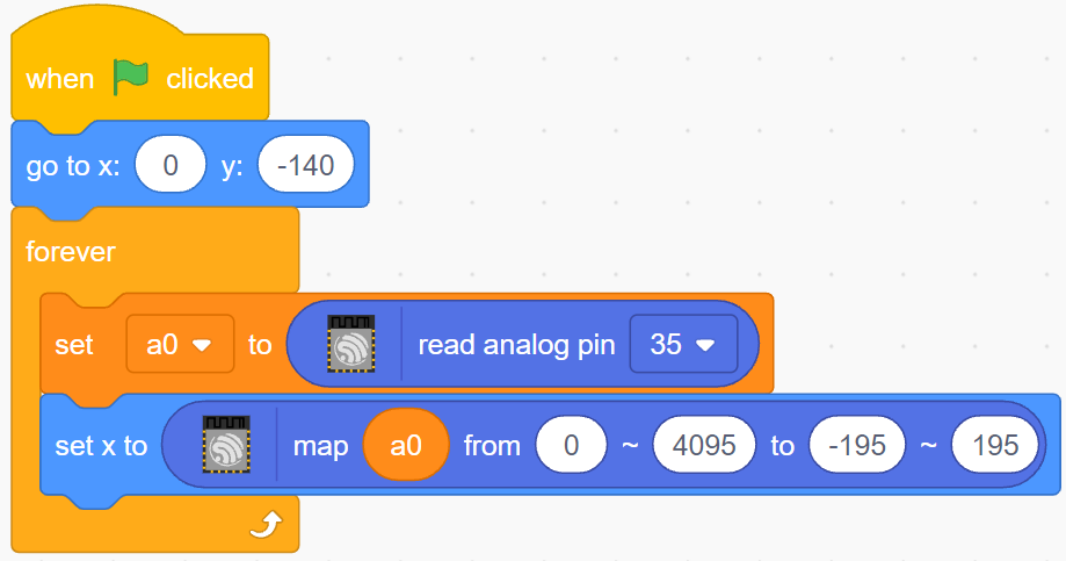
- Elimina el sprite predeterminado, usa el botón **Elegir un Sprite** para añadir el sprite **Paleta**, y ajusta sus coordenadas x e y a (0, -140).



- Ve a la página de **Disfraces**, elimina el contorno y cambia su color a gris oscuro.



- Ahora programa el sprite **Paleta** para establecer su posición inicial a (0, -140) cuando se haga clic en la bandera verde, y lee el valor del pin35 (potenciómetro) en la variable **a0**. Dado que el sprite **Paleta** se mueve de izquierda a derecha en el escenario en las coordenadas x de -195 a 195, necesitas usar el bloque [map] para mapear el rango de la variable **a0** de 0~4095 a -195~195.

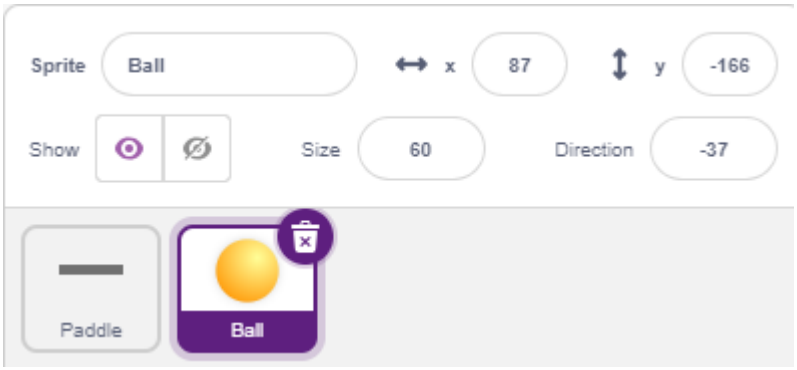


- Ahora puedes girar el potenciómetro para ver si la **Paleta** puede moverse hacia la izquierda y hacia la derecha en el escenario.

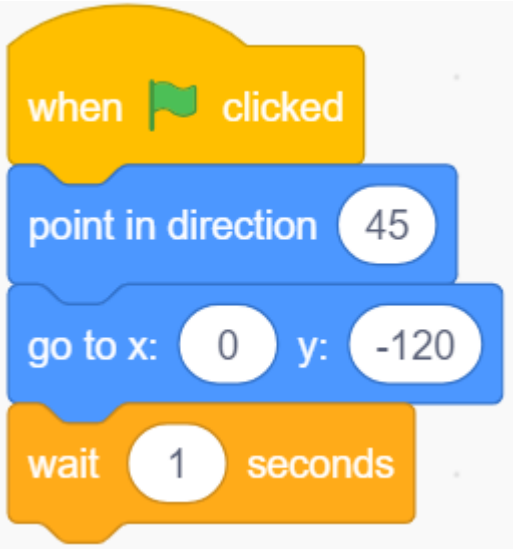
2. Sprite de la Bola

El efecto del sprite de la bola es que se mueva por el escenario y rebote cuando toque el borde; rebota hacia abajo si toca el bloque sobre el escenario; rebota hacia arriba si toca el sprite de la **Paleta** durante su caída; si no, el script deja de ejecutarse y el juego termina.

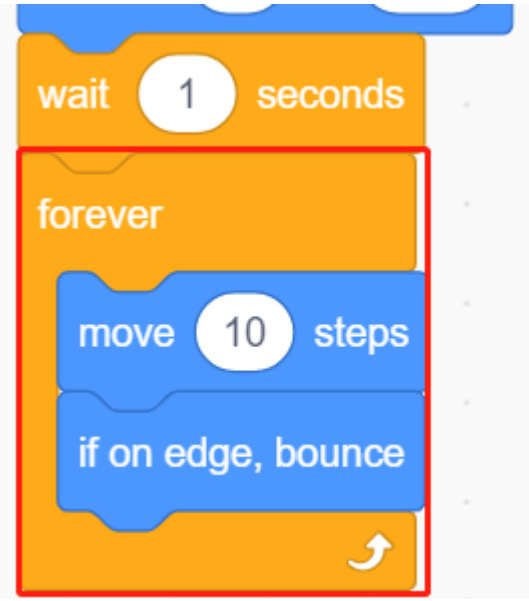
- Añade el sprite **Bola**.



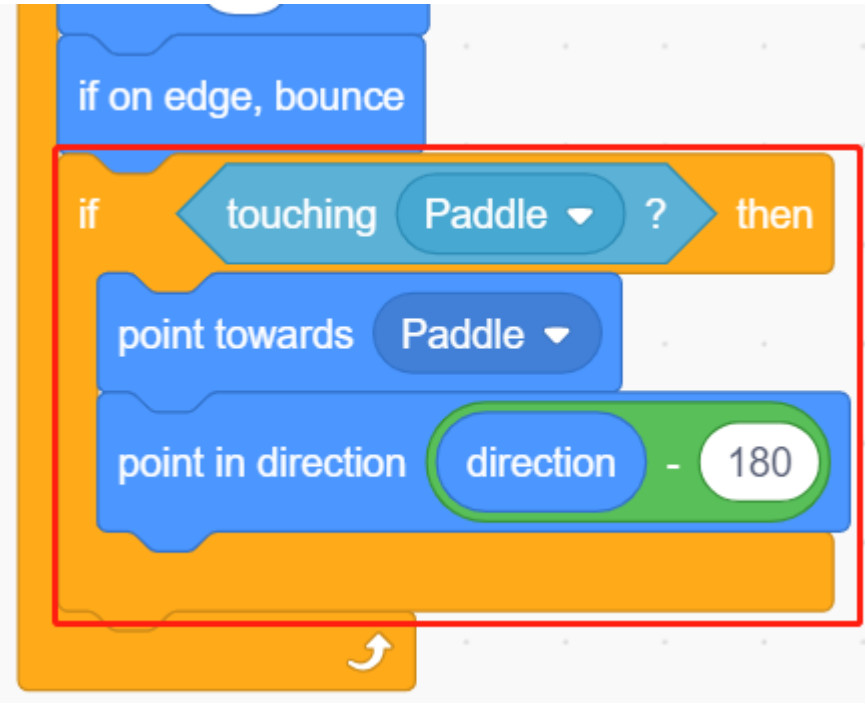
- Cuando se haga clic en la bandera verde, establece el ángulo del sprite **Bola** a 45° y la posición inicial a (0, -120).

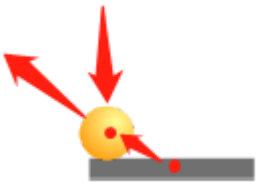


- Ahora deja que el sprite **Bola** se mueva por el escenario y rebote cuando toque el borde, y puedes hacer clic en la bandera verde para ver el efecto.



- Cuando el sprite **Bola** toca el sprite **Paleta**, realiza un reflejo. La forma fácil de hacer esto es dejar que el ángulo se invierta directamente, pero entonces encontrarás que la trayectoria de la bola es completamente fija, lo cual es demasiado aburrido. Por lo tanto, usamos el centro de los dos sprites para calcular y hacer que la bola rebote en la dirección opuesta al centro del baffle.





- Cuando el sprite **Bola** cae al borde del escenario, el script deja de ejecutarse y el juego termina.

3. Sprite del Bloque1

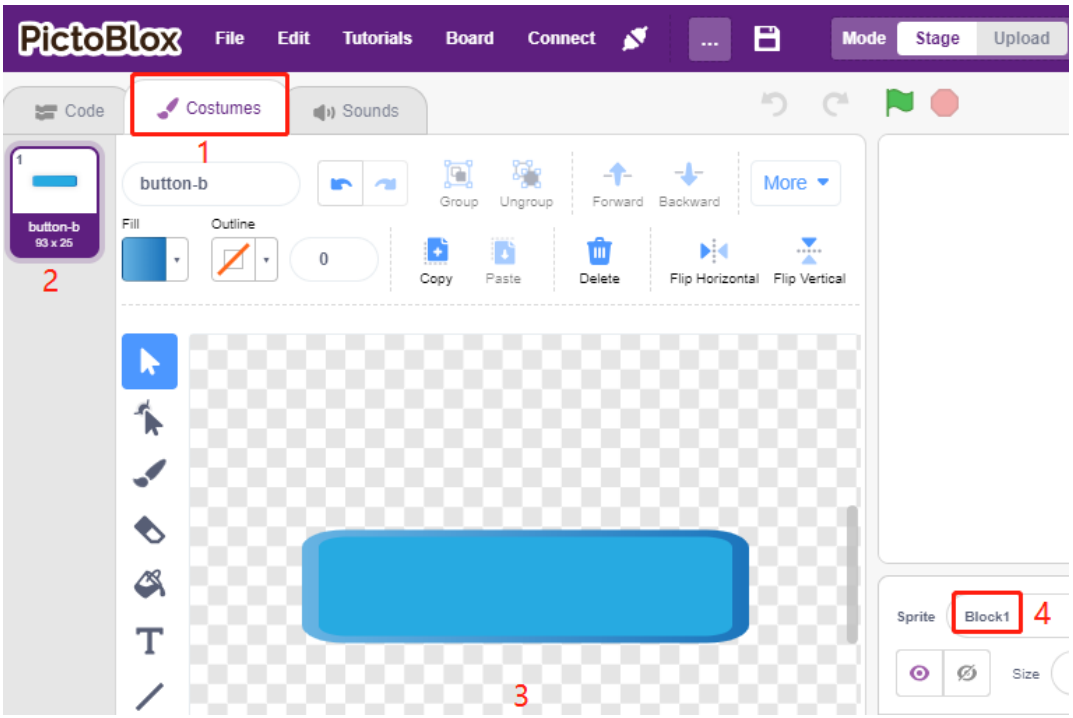
El sprite **Bloque1** aparece con el efecto de clonarse a sí mismo 4x8 veces sobre el escenario en un color aleatorio, y eliminar un clon si es tocado por el sprite **Bola**.

El sprite **Bloque1** no está disponible en la biblioteca **PictoBlox**, necesitas dibujarlo tú mismo o modificarlo con un sprite existente. Aquí vamos a modificarlo con el sprite **Botón3**.

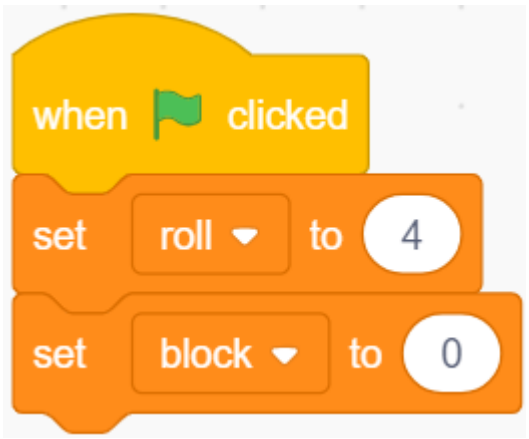
- Después de añadir el sprite **Botón3**, ve a la página de **Disfraces**. Ahora elimina primero **botón-a**, luego reduce tanto el ancho como el alto de **botón-b**, y cambia el nombre del sprite a **Bloque1**, como se muestra en la siguiente imagen.

Nota:

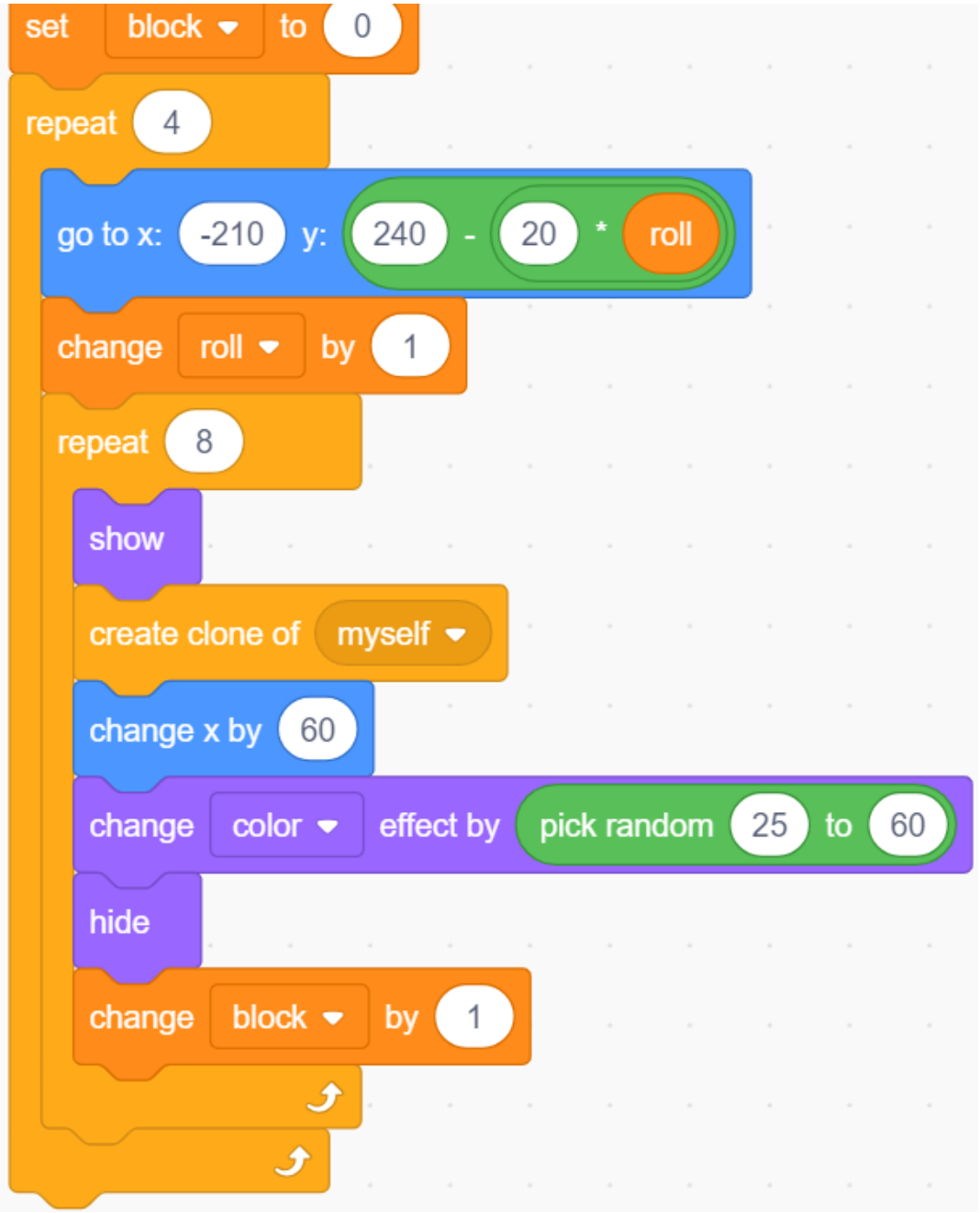
- Para el ancho de **Bloque1**, puedes simularlo en la pantalla para ver si puedes colocar 8 en fila, si no, entonces reduce el ancho apropiadamente.
- En el proceso de reducir el sprite **Bloque1**, necesitas mantener el punto central en el medio del sprite.



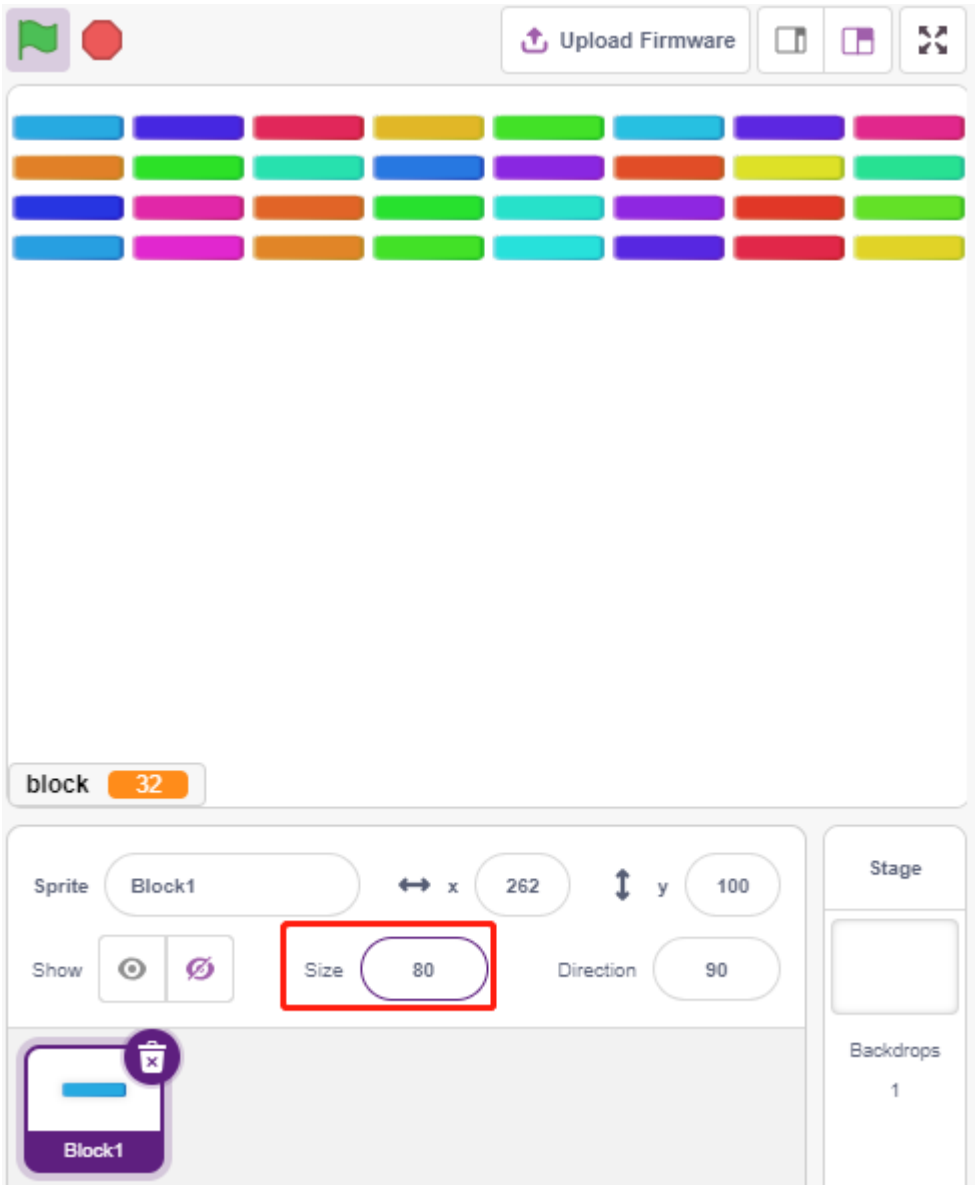
- Ahora crea primero 2 variables, **bloque** para almacenar el número de bloques y **fila** para almacenar el número de filas.



- Necesitamos hacer un clon del sprite **Bloque1** para que se muestre de izquierda a derecha, de arriba abajo, uno por uno, en total 4x8, con colores aleatorios.

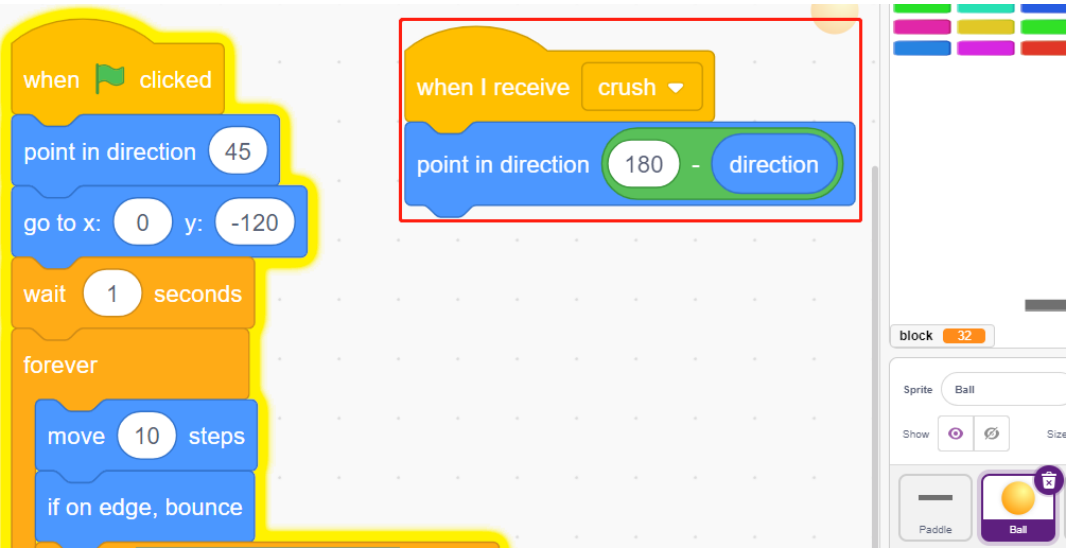


- Después de escribir el script, haz clic en la bandera verde y observa la presentación en el escenario, si es demasiado compacta o pequeña, puedes cambiar el tamaño.



- Ahora escribe el evento desencadenante. Si el clon del sprite **Bloque1** toca el sprite **Bola**, elimina el clon y emite el mensaje **crush**.

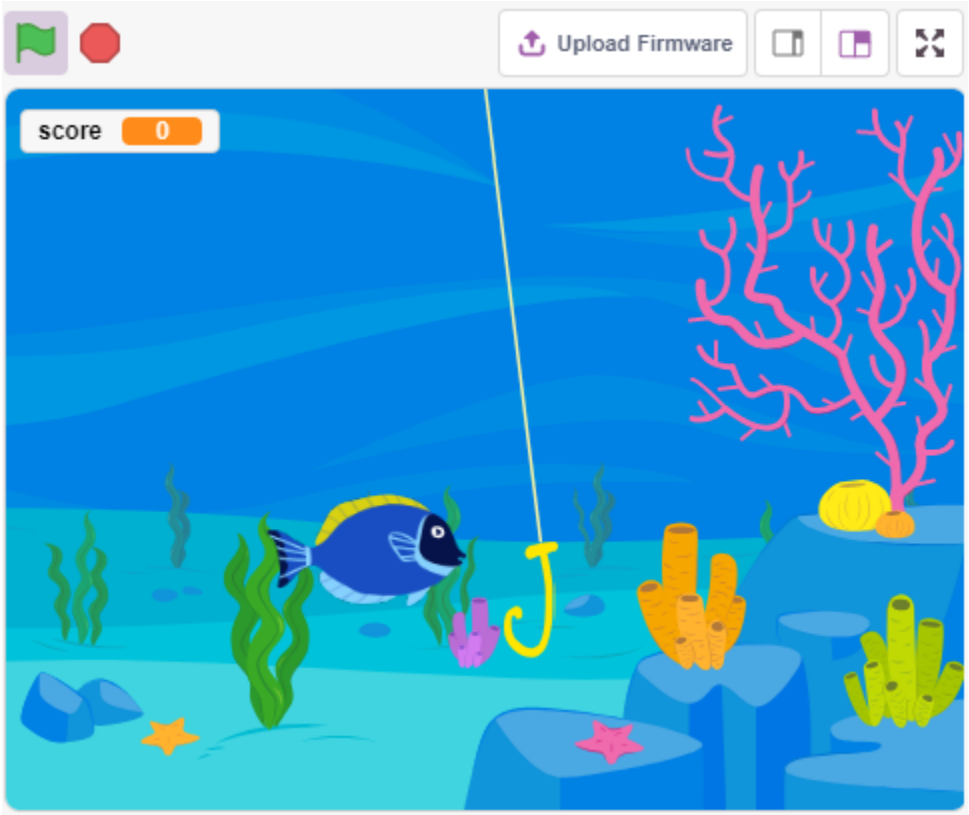
- Vuelve al sprite **Bola**, cuando se recibe la emisión **crush** (el sprite **Bola** toca el clon del sprite **Bloque1**), la **Bola** es expulsada en la dirección opuesta.



5.20 2.17 JUEGO - Pesca

Aquí, jugamos a un juego de pesca con un botón.

Cuando el script está en ejecución, los peces nadan de izquierda a derecha en el escenario. Debes presionar el botón cuando el pez esté casi cerca del anzuelo (se recomienda presionarlo durante más tiempo) para atrapar al pez, y el número de peces capturados se registrará automáticamente.



5.20.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

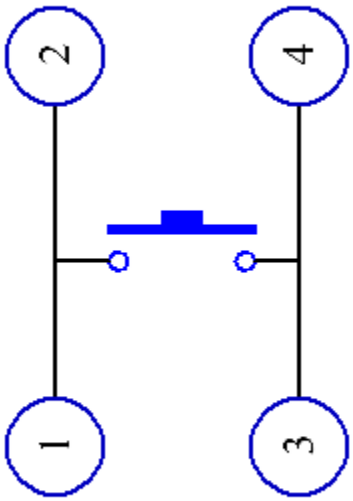
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Resistor	
Botón	

5.20.2 Construir el Circuito

El botón es un dispositivo de 4 pines, ya que el pin 1 está conectado al pin 2, y el pin 3 al pin 4. Cuando se presiona el botón, los 4 pines se conectan, cerrando así el circuito.

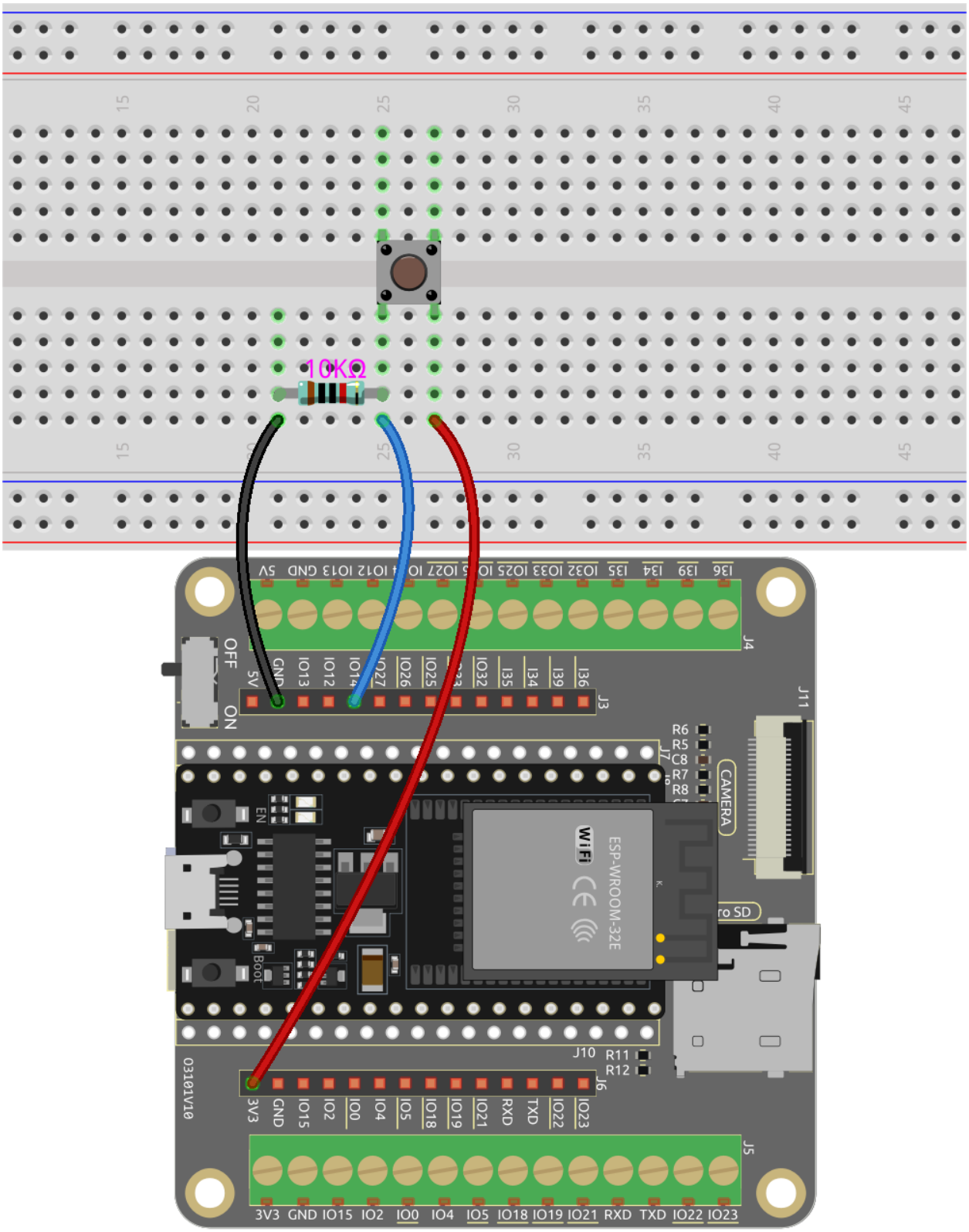
Button



Internal Structure

Construye el circuito según el siguiente diagrama.

- Conecta uno de los pines del lado izquierdo del botón al pin14, que está conectado a una resistencia de pull-down y un condensador de 0.1uF (104) (para eliminar jitter y producir un nivel estable cuando el botón está funcionando).
- Conecta el otro extremo de la resistencia y el condensador a GND, y uno de los pines del lado derecho del botón a 5V.

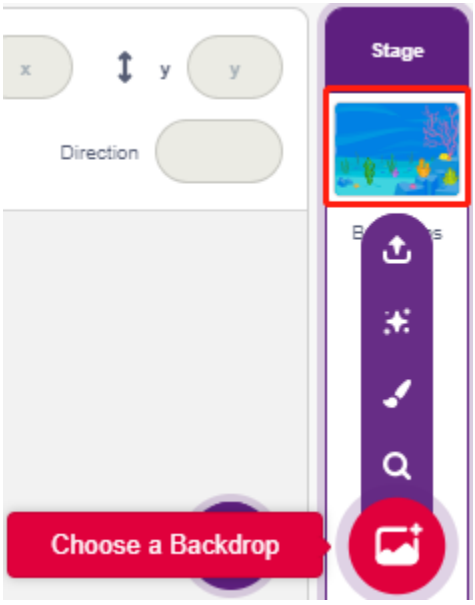

10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

5.20.3 Programación

Primero necesitamos seleccionar un fondo **Subacuático**, luego añadir un sprite **Pez** y permitir que nade de un lado a otro en el escenario. Después, dibujar un sprite **Anzuelo** y controlarlo con un botón para comenzar a pescar. Cuando el sprite **Pez** toque el sprite **Anzuelo** en estado de pesca (se vuelve rojo), quedará enganchado.

1. Añadir un fondo

Usa el botón **Elegir un Fondo** para añadir un fondo **Subacuático**.

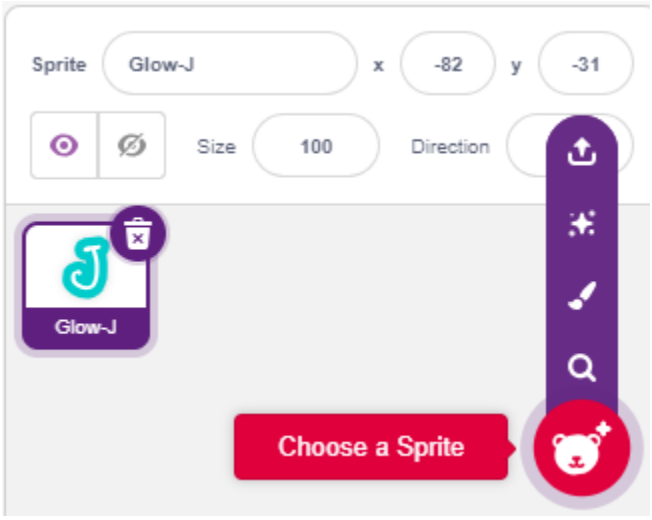


2. Sprite Anzuelo

El sprite **Anzuelo** generalmente permanece bajo el agua en estado amarillo; cuando se presiona el botón, está en estado de pesca (rojo) y se mueve por encima del escenario.

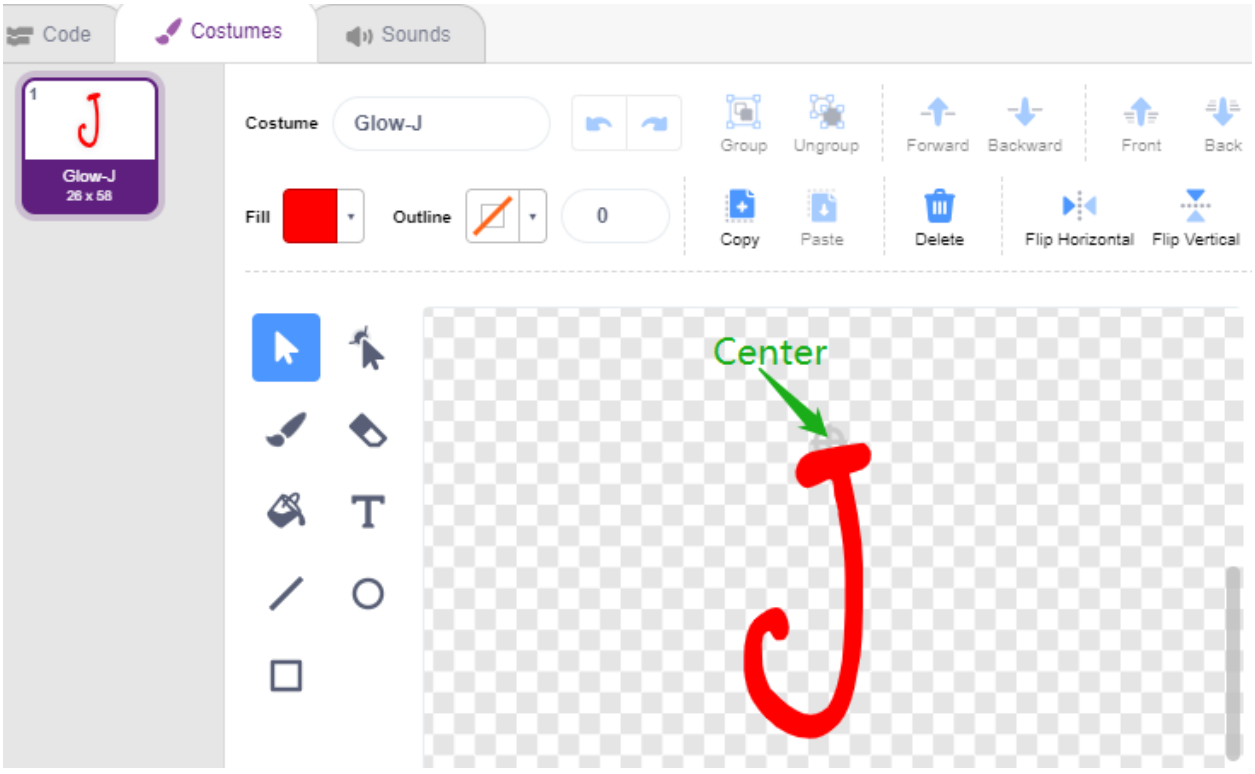
No hay un sprite **Anzuelo** en Pictoblox, podemos modificar el sprite **Glow-J** para que parezca un anzuelo.

- Añade el sprite **Glow-J** a través de **Elegir un Sprite**.

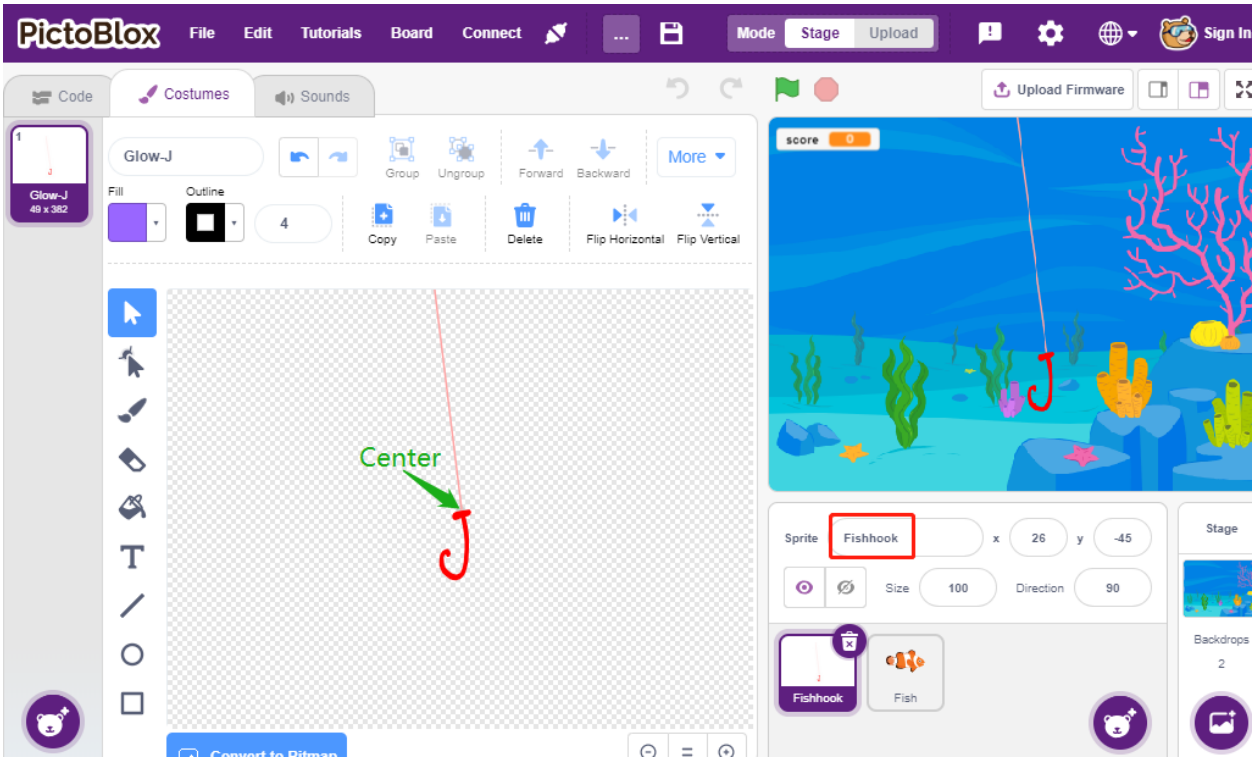


- Ahora ve a la página de **Disfraces** del sprite **Glow-J**, selecciona el relleno Cyan en la pantalla y elimínalo. Luego cambia el color de J a rojo y también reduce su ancho. El punto más importante a tener en cuenta es que necesitas

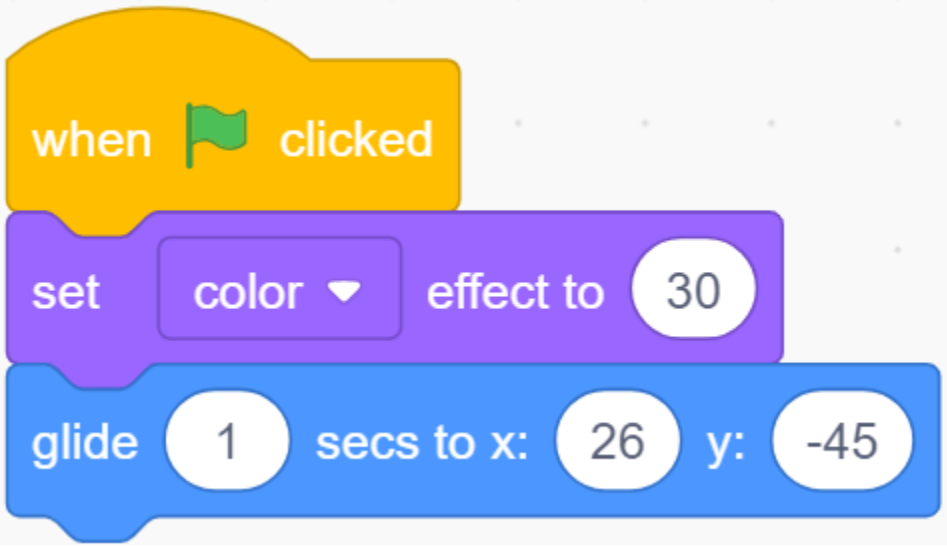
que la parte superior esté justo en el punto central.



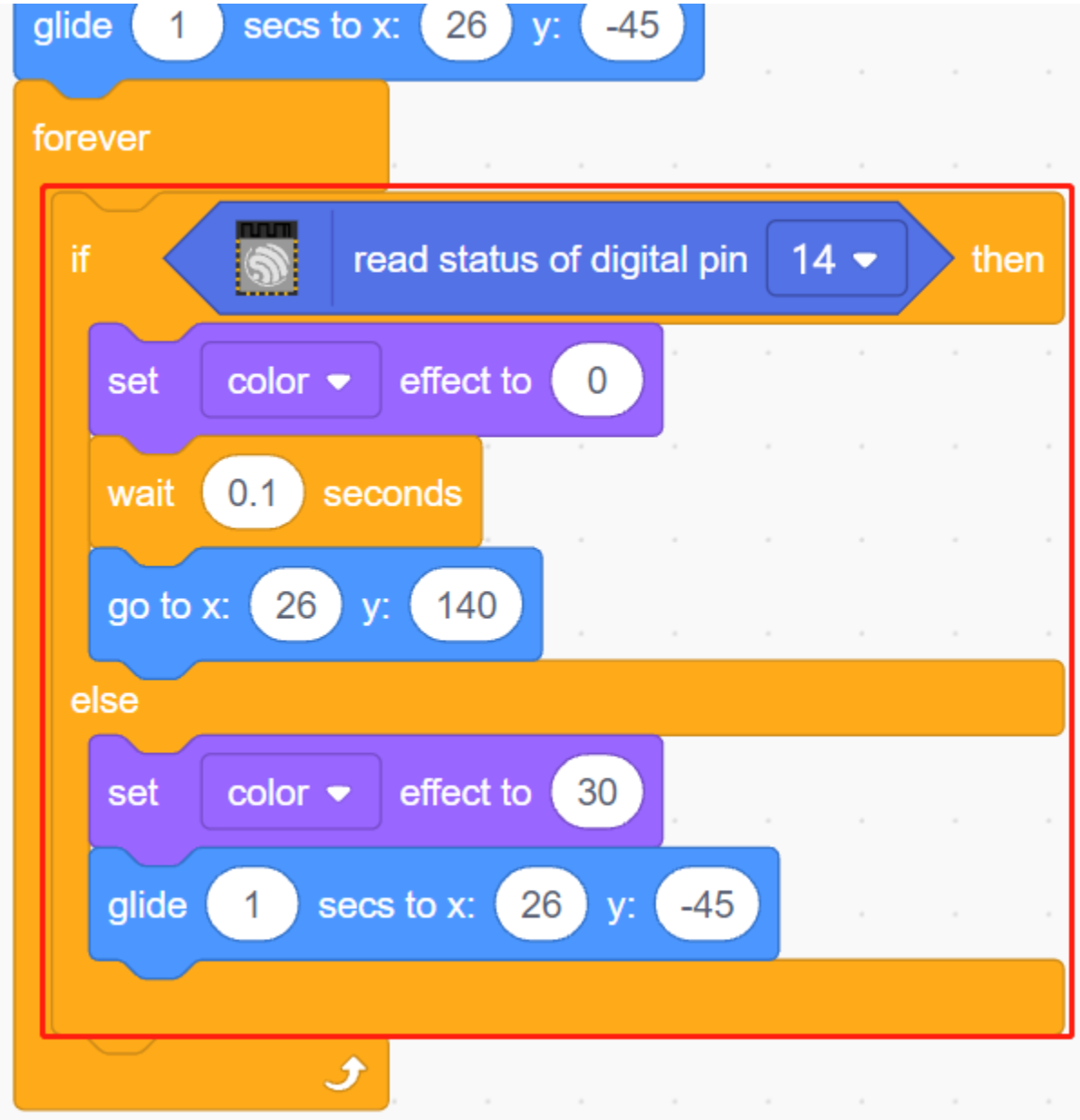
- Usa la **Herramienta de Línea** para dibujar una línea lo más larga posible desde el punto central hacia arriba (línea fuera del escenario). Ahora que el sprite está dibujado, establece el nombre del sprite a **Anzuelo** y muévelo a la posición correcta.



- Cuando se haga clic en la bandera verde, establece el efecto de color del sprite a 30 (amarillo) y establece su posición inicial.



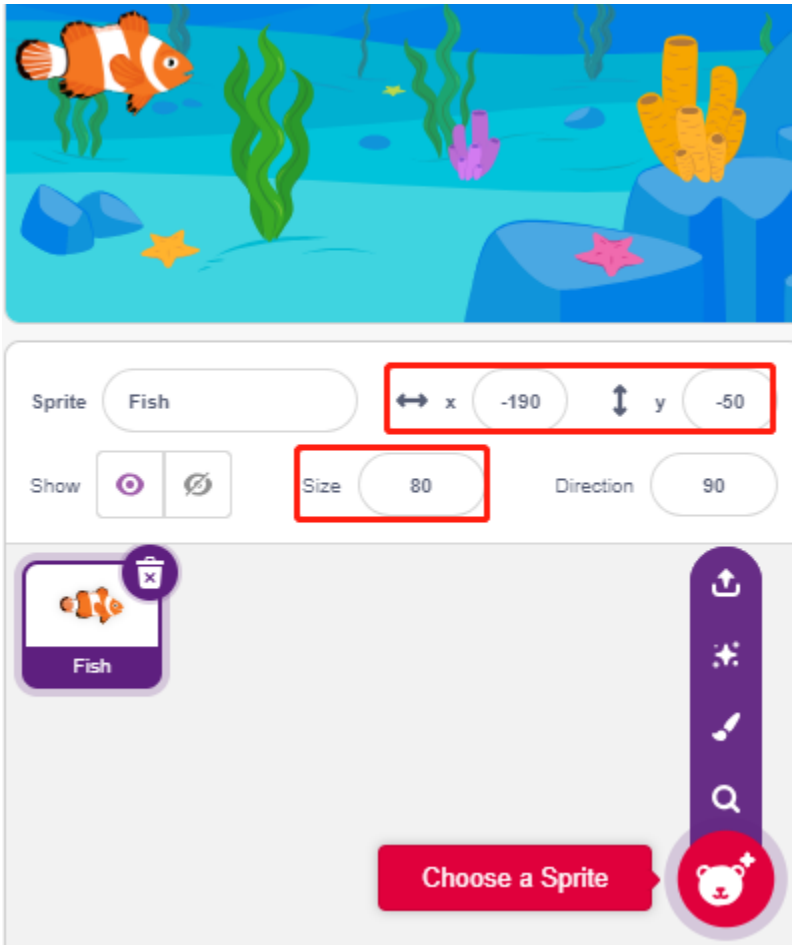
- Si se presiona el botón, establece el efecto de color a 0 (rojo, estado de inicio de pesca), espera 0.1 y luego mueve el sprite **Anzuelo** hacia la parte superior del escenario. Suelta el botón y permite que el **Anzuelo** regrese a su posición inicial.



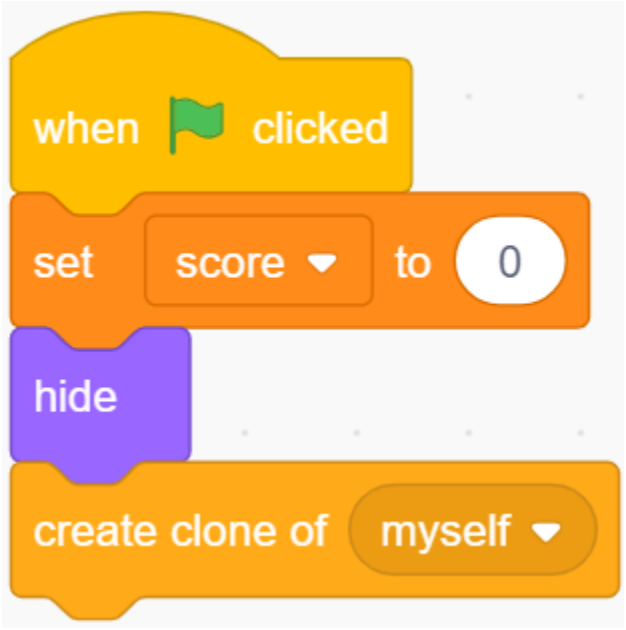
3. Sprite Pez

El efecto que se busca con el sprite **Pez** es moverse de izquierda a derecha en el escenario, y cuando se encuentra con un sprite **Anzuelo** en estado de pesca, se encoge y se mueve a una posición específica y luego desaparece, para luego clonar un nuevo sprite **pez** nuevamente.

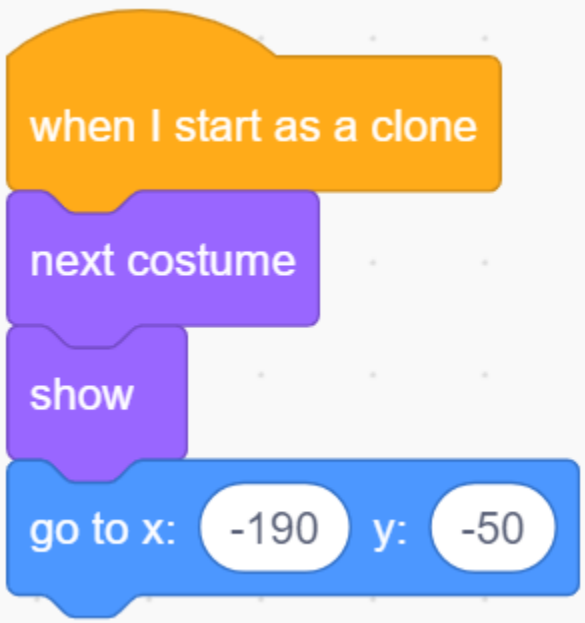
- Ahora añade el sprite **pez** y ajusta su tamaño y posición.



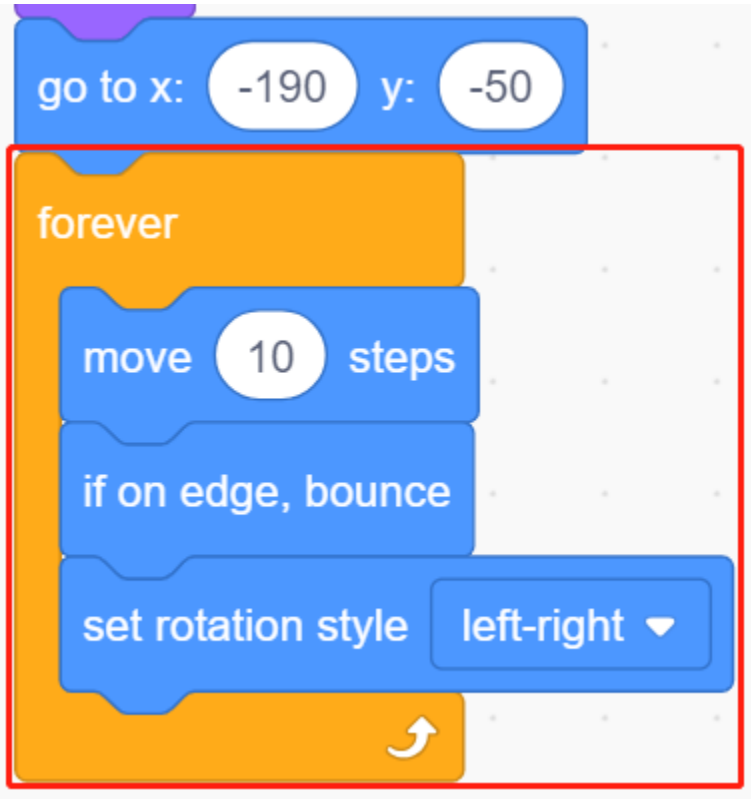
- Crea una variable **puntuación** para almacenar el número de peces capturados, oculta este sprite y clónalo.



- Muestra el clon del sprite **pez**, cambia su disfraz y finalmente establece la posición inicial.



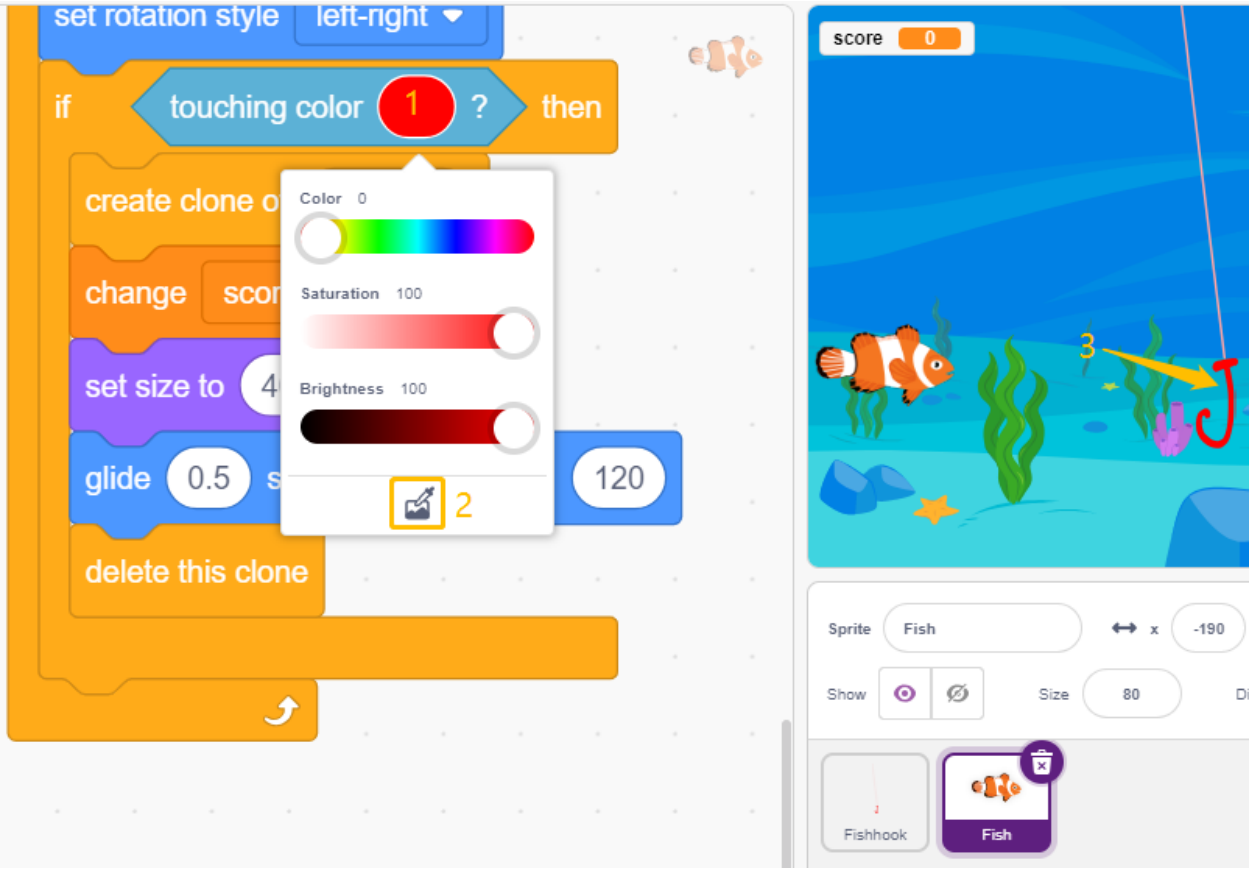
- Haz que el clon del sprite **pez** se mueva de izquierda a derecha y rebote cuando toque el borde.



- El sprite **pez** (del clon) no reaccionará cuando pase el sprite **Anzuelo**; cuando toque el sprite **Anzuelo** en estado de pesca (se vuelve rojo), será capturado, en este punto la puntuación (variable puntuación) +1, y también mostrará una animación de puntuación (se encoge un 40 %, se mueve rápidamente a la posición del marcador y desaparece). Al mismo tiempo, se crea un nuevo pez (un nuevo clon del sprite pez) y el juego continúa.

Nota: Necesitas hacer clic en el área de color en el bloque [Tocar color] y luego seleccionar la herramienta cuentagotas

para recoger el color rojo del sprite **Anzuelo** en el escenario. Si eliges un color arbitrariamente, este bloque [Tocar color] no funcionará.



5.21 2.18 JUEGO - No Toques la Baldosa Blanca

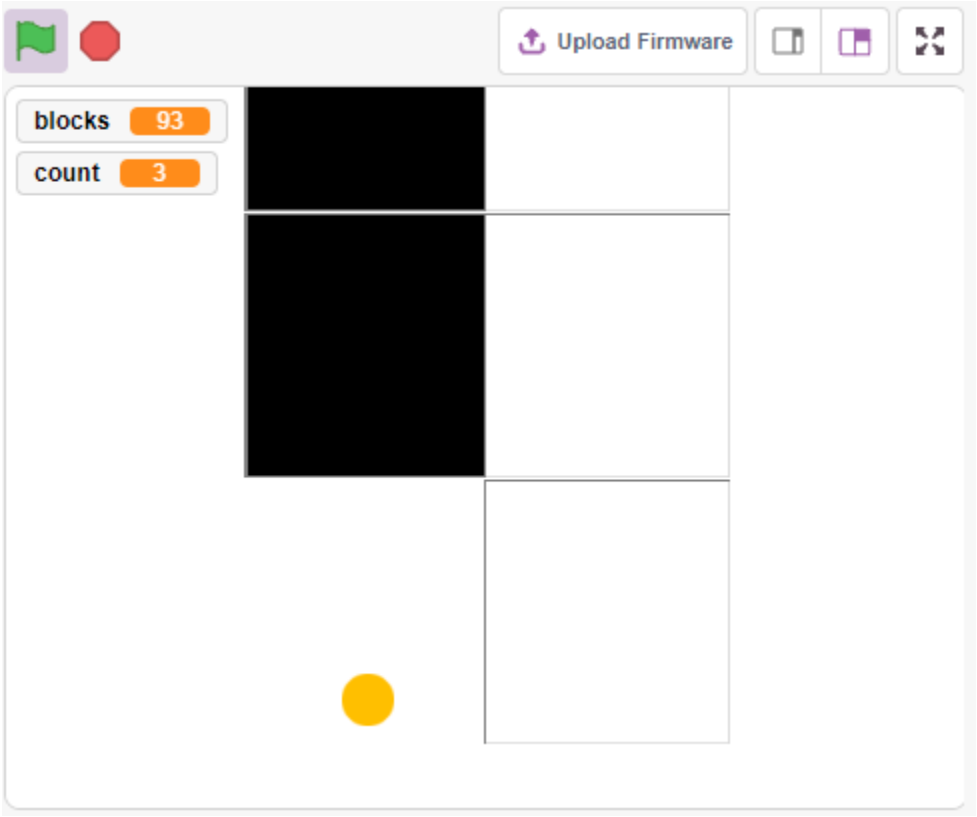
Estoy seguro de que muchos de vosotros habéis jugado a este juego en vuestros teléfonos móviles. Este juego se juega tocando baldosas negras que aparecen al azar para sumar puntos, la velocidad se va incrementando, tocar una baldosa blanca o perder una negra significa el fin del juego.

Ahora utilizaremos PictoBlox para replicarlo.

Inserta dos módulos de evitación de obstáculos IR verticalmente en la placa de pruebas, cuando tu mano está colocada sobre uno de los módulos IR, aparecerá un punto parpadeante en el escenario, representando que se ha hecho un toque.

Si el toque es a la baldosa negra, la puntuación aumenta en 1, tocar la baldosa blanca, la puntuación disminuye en 1.

Necesitas decidir si colocar tu mano sobre el módulo IR de la izquierda o sobre el de la derecha, dependiendo de la posición de la baldosa negra en el escenario.



5.21.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

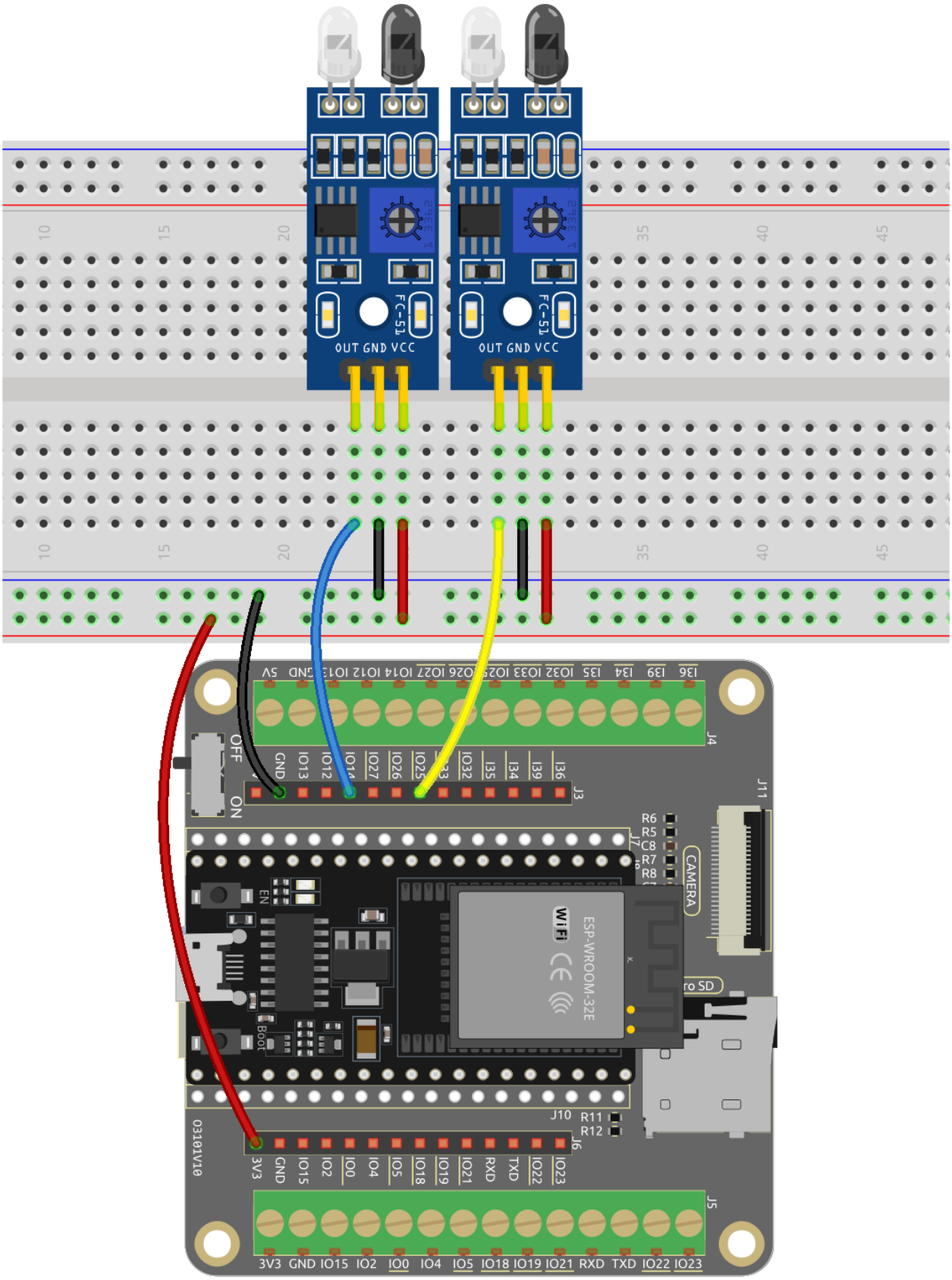
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Protoboard	
Cables Puente	
Módulo de Evitación de Obstáculos	

5.21.2 Construir el Circuito

El módulo de evitación de obstáculos es un sensor de proximidad infrarrojo ajustable en distancia cuya salida es normalmente alta y baja cuando se detecta un obstáculo.

Ahora construye el circuito según el diagrama a continuación.


FC-51
OUT GND VCC
FC-51
OUT GND VCC
ESP-WROOM-32E
CAMERA
OFF
ON
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

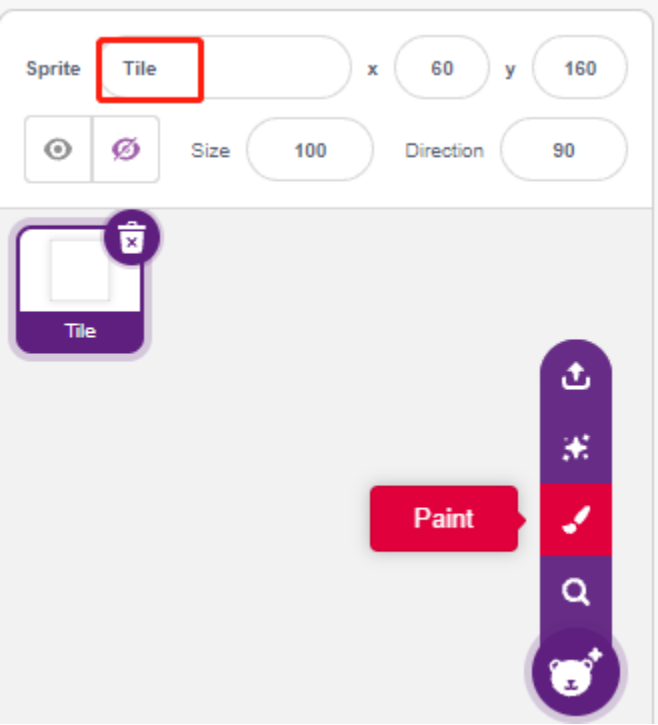
5.21.3 Programación

Aquí necesitamos tener 3 sprites, **Baldosa**, **IR Izquierdo** y **IR Derecho**.

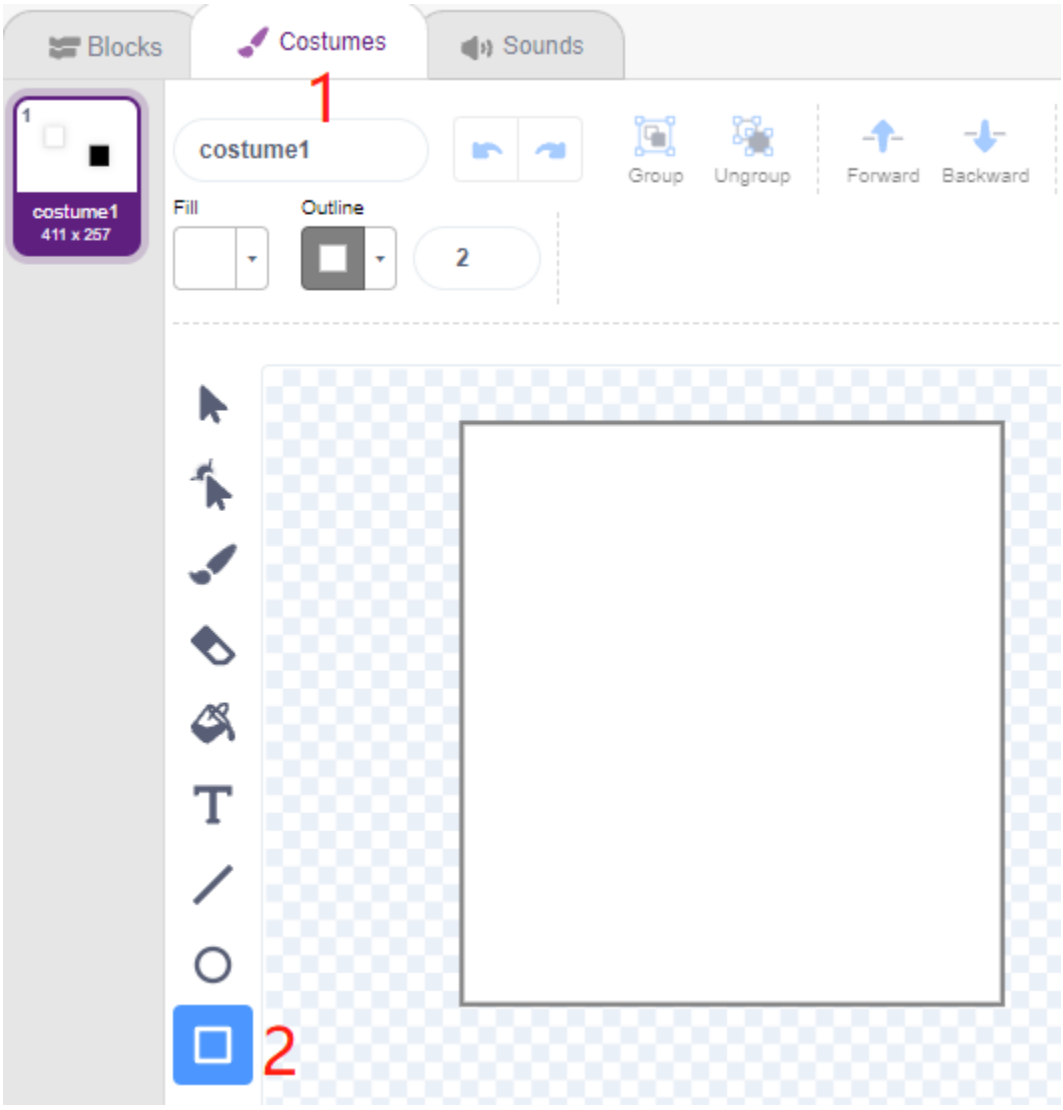
- **Sprite Baldosa**: utilizado para lograr el efecto de baldosas negras y blancas alternas descendiendo, en el teléfono móvil este juego generalmente tiene 4 columnas, aquí solo haremos dos columnas.
- **Sprite IR Izquierdo**: utilizado para lograr el efecto de clic, cuando el módulo IR izquierdo siente tu mano, enviará un mensaje - **izquierda** al sprite **IR Izquierdo**, permitiéndole empezar a funcionar. Si toca la baldosa negra en el escenario, la puntuación aumentará en 1, de lo contrario, disminuirá en 1.
- **Sprite IR Derecho**: La función es básicamente la misma que **IR Izquierdo**, excepto que recibe la información **derecha**.

1. Pinta un sprite Baldosa.

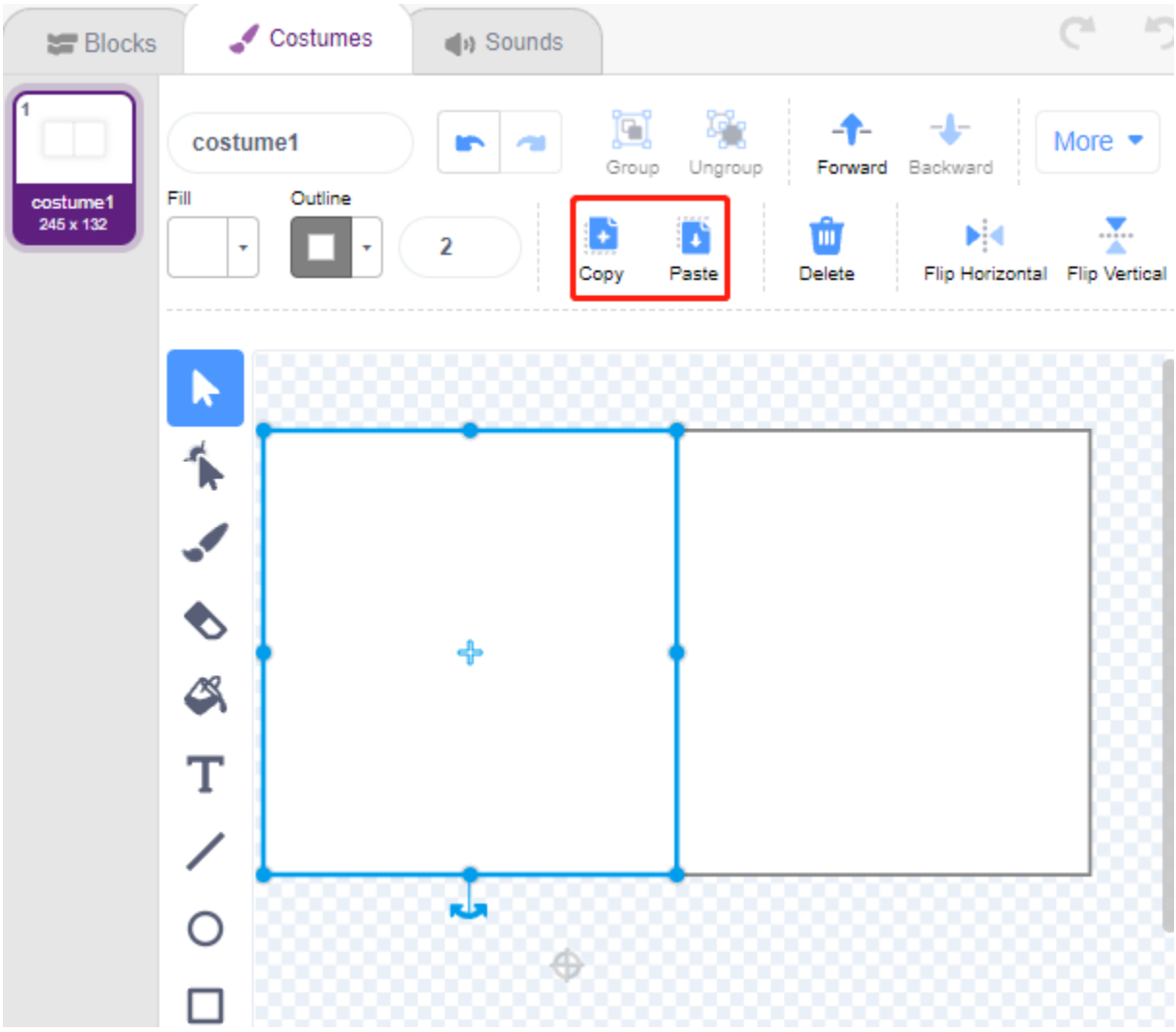
Elimina el sprite predeterminado, pasa el mouse sobre el icono de **Añadir Sprite**, selecciona **Pintar** y aparecerá un sprite en blanco y nómbralo **Baldosa**.



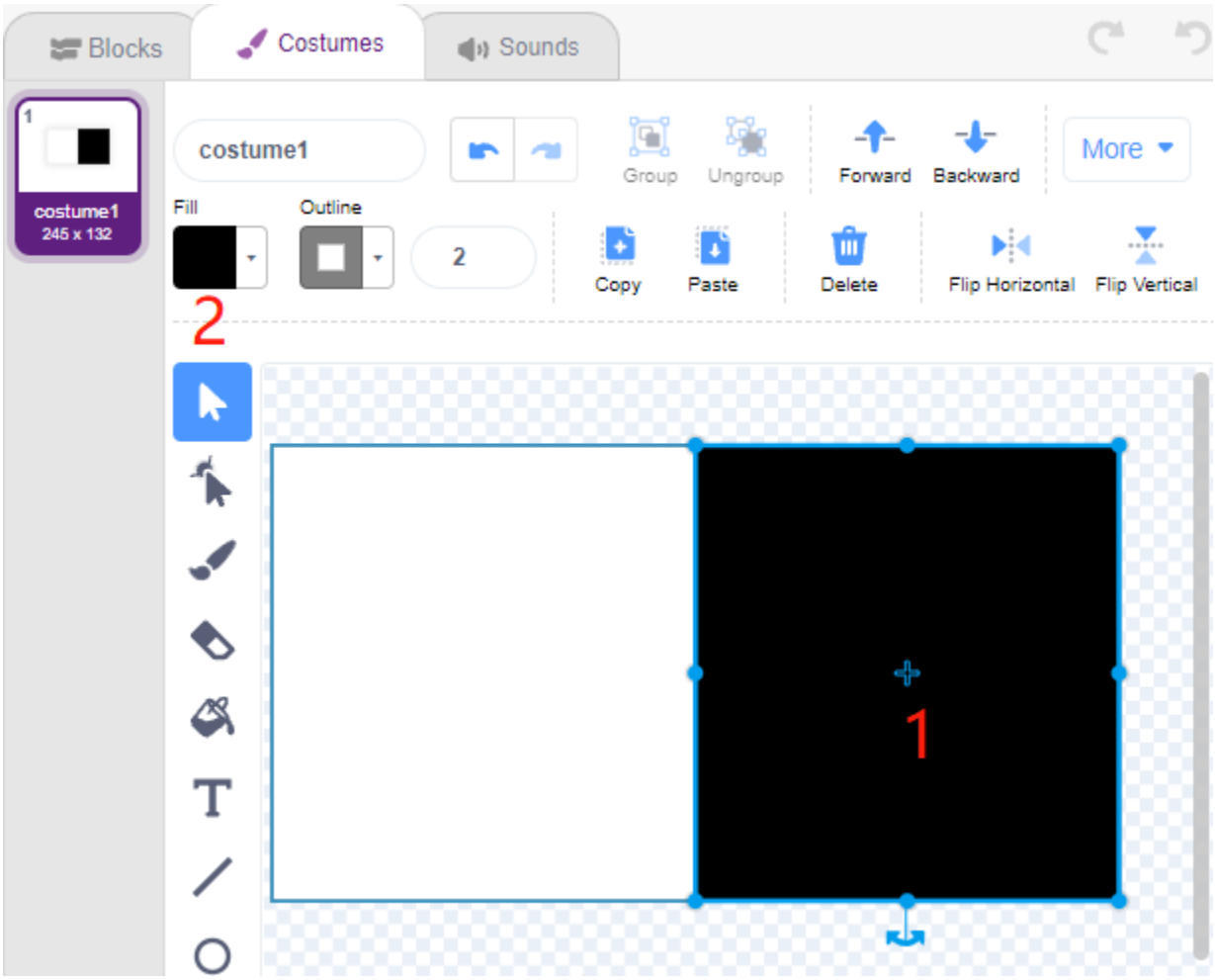
Ve a la página de **Disfraces** y usa la herramienta **Rectángulo** para dibujar un rectángulo.



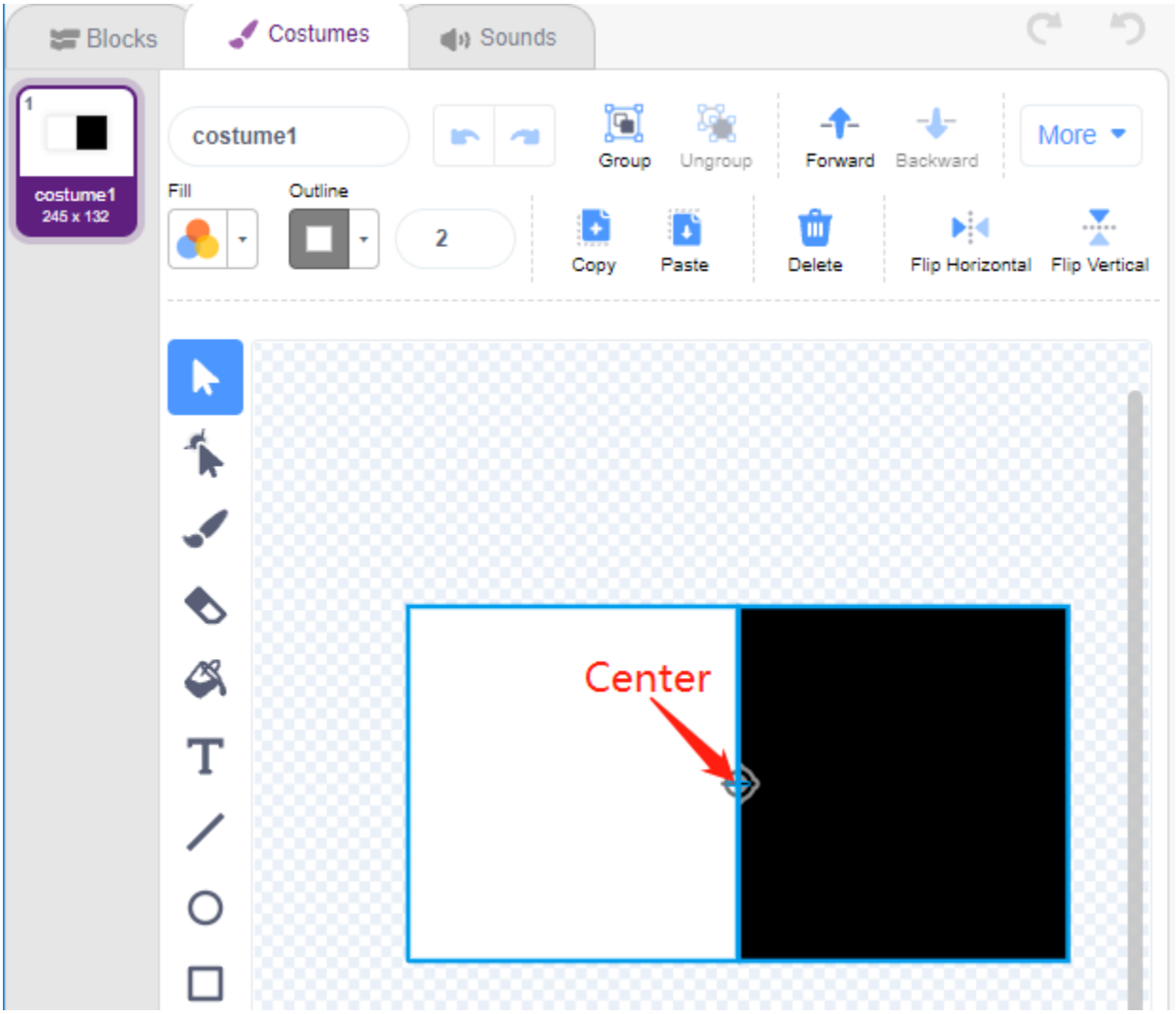
Selecciona el rectángulo y haz clic en **Copiar** -> **Pegar** para hacer un rectángulo idéntico, luego mueve los dos rectángulos a una posición alineada.



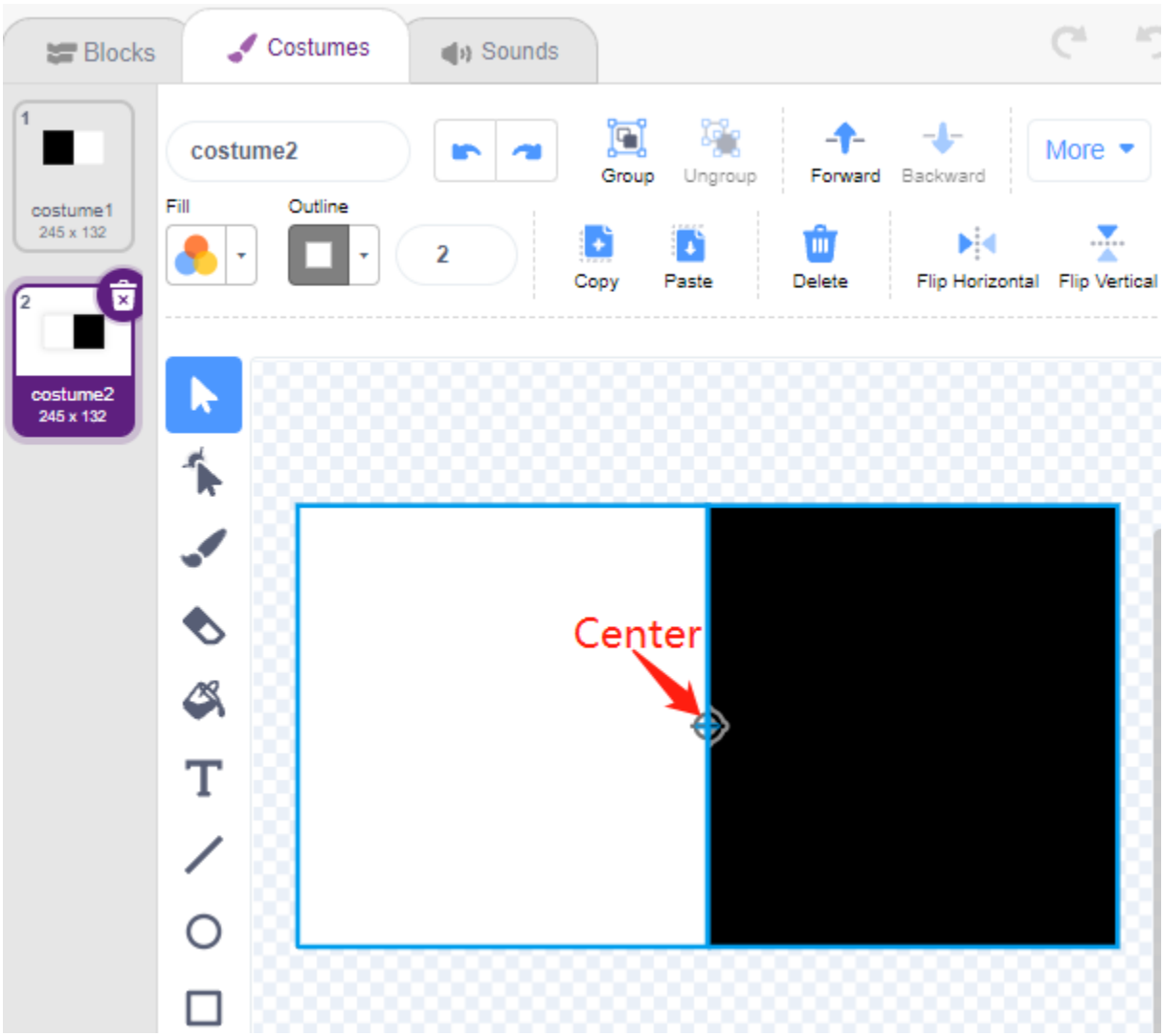
Selecciona uno de los rectángulos y elige un color de relleno negro.



Ahora selecciona ambos rectángulos y muévelos de modo que sus puntos centrales coincidan con el centro del lienzo.

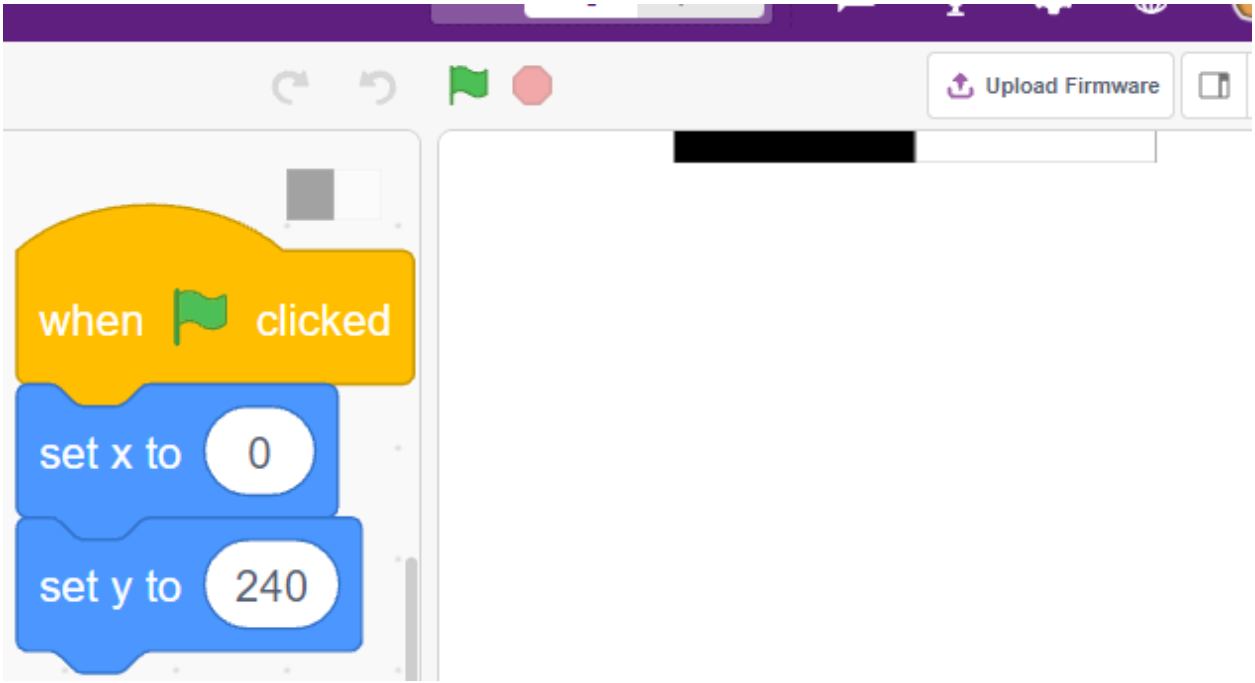


Duplica el disfraz1, alternando los colores de relleno de los dos rectángulos. Por ejemplo, el color de relleno del disfraz1 es blanco a la izquierda y negro a la derecha, y el color de relleno del disfraz2 es negro a la izquierda y blanco a la derecha.



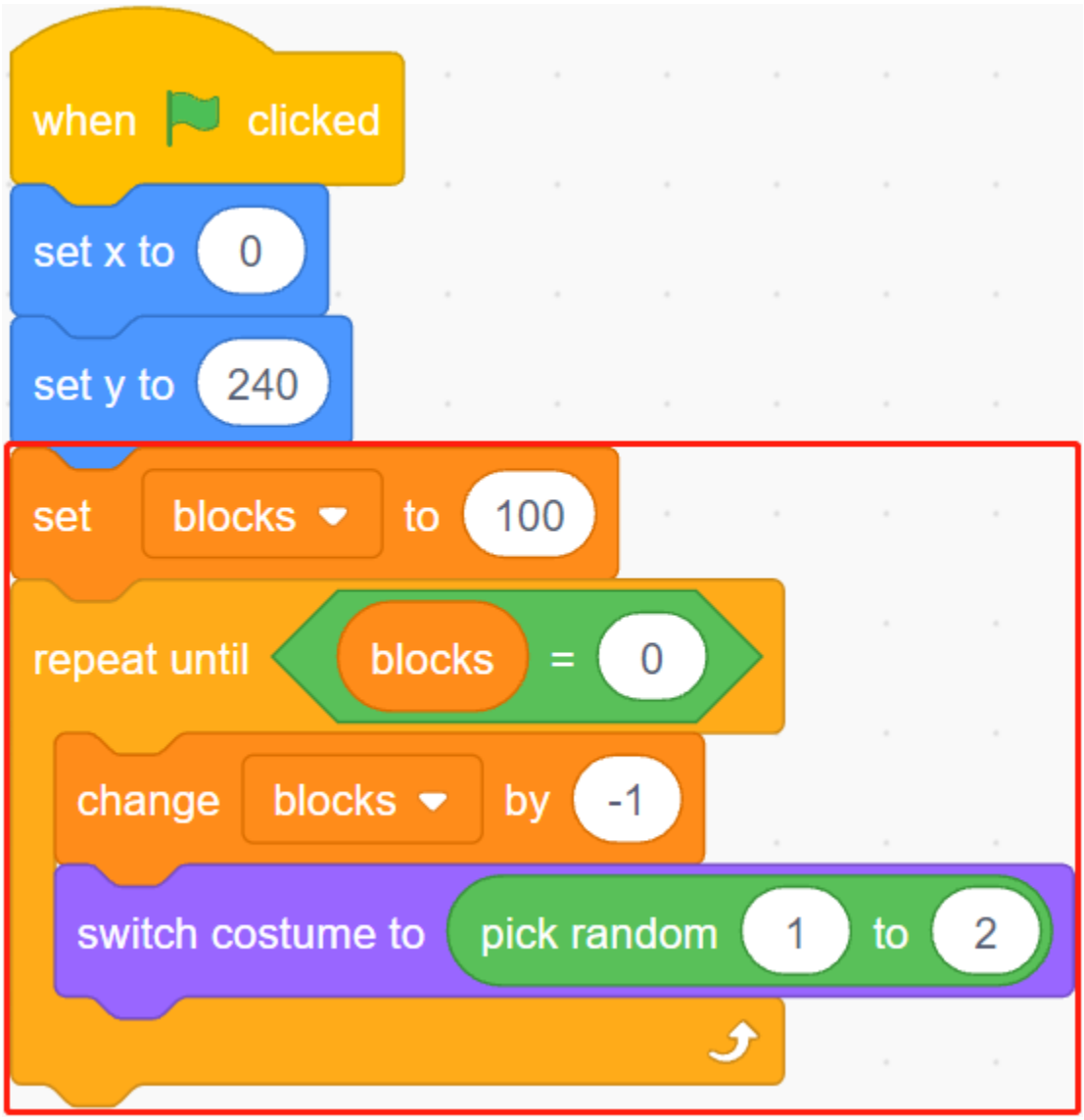
2. Programando el sprite Baldosa

Ahora vuelve a la página de **Bloques** y establece la posición inicial del sprite **Baldosa** para que esté en la parte superior del escenario.

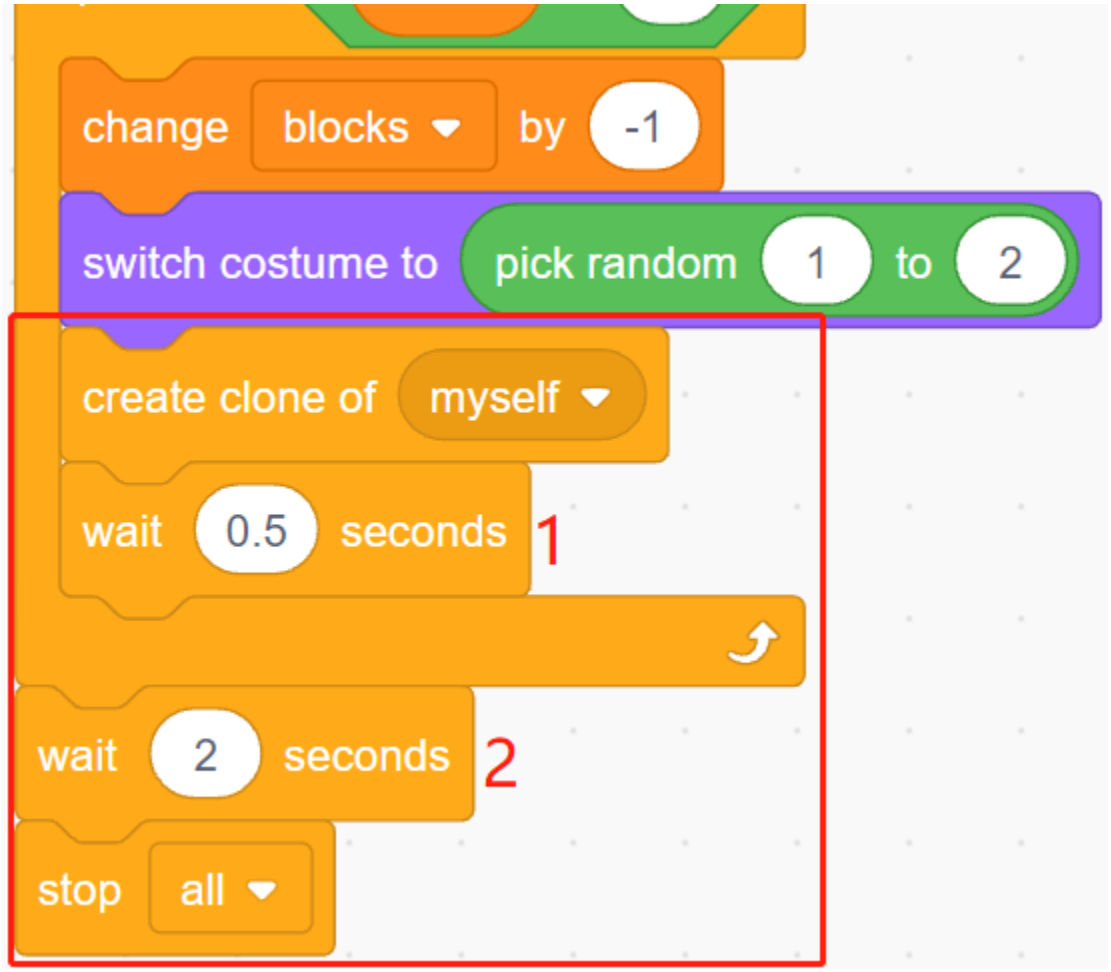


Crea una variable -**bloques** y dale un valor inicial para determinar la cantidad de veces que el sprite **Baldosa** aparecerá. Usa el bloque [repetir hasta] para hacer que la variable **bloques** disminuya gradualmente hasta que **bloques** sea 0. Durante este tiempo, haz que el sprite **Baldosa** cambie aleatoriamente de disfraz.

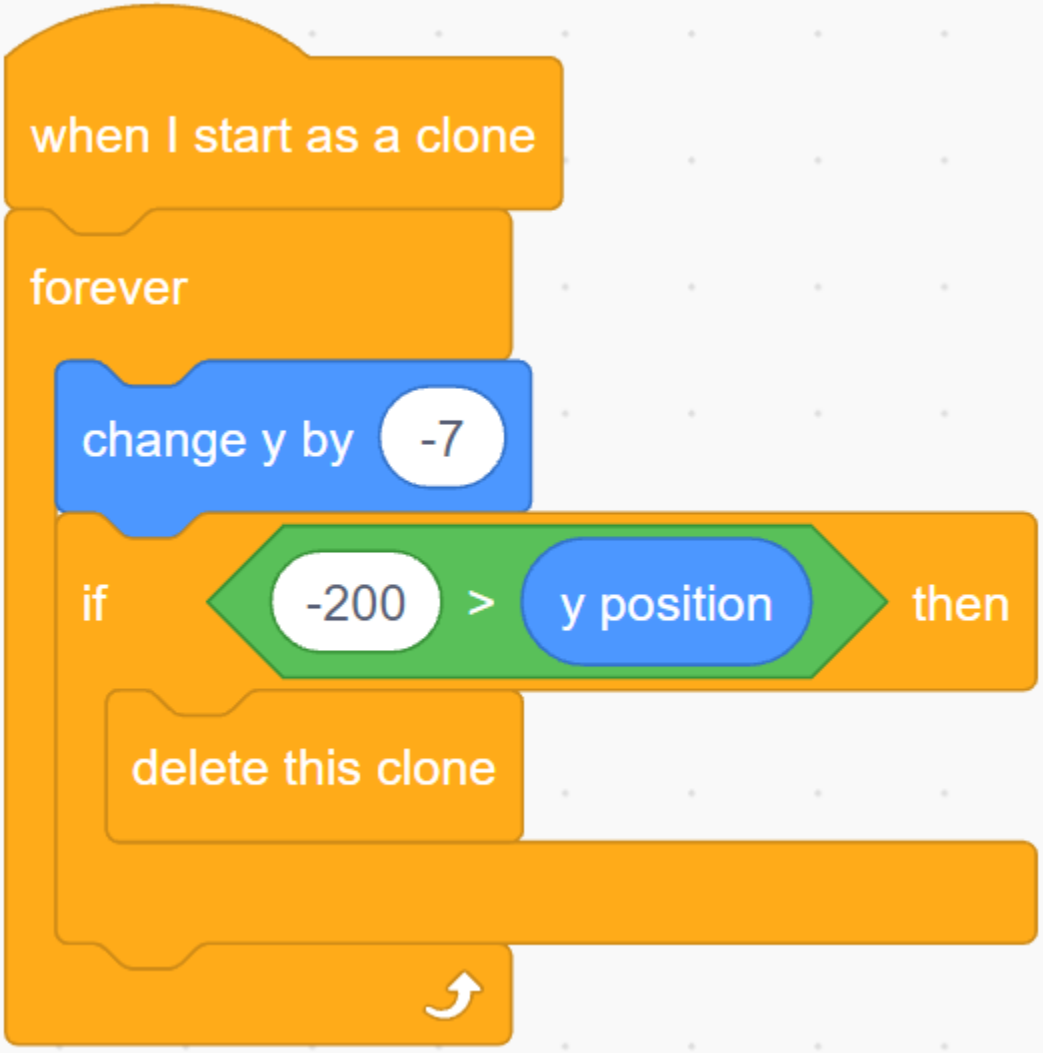
Después de hacer clic en la bandera verde, verás el sprite **Baldosa** en el escenario cambiar rápidamente de disfraces.



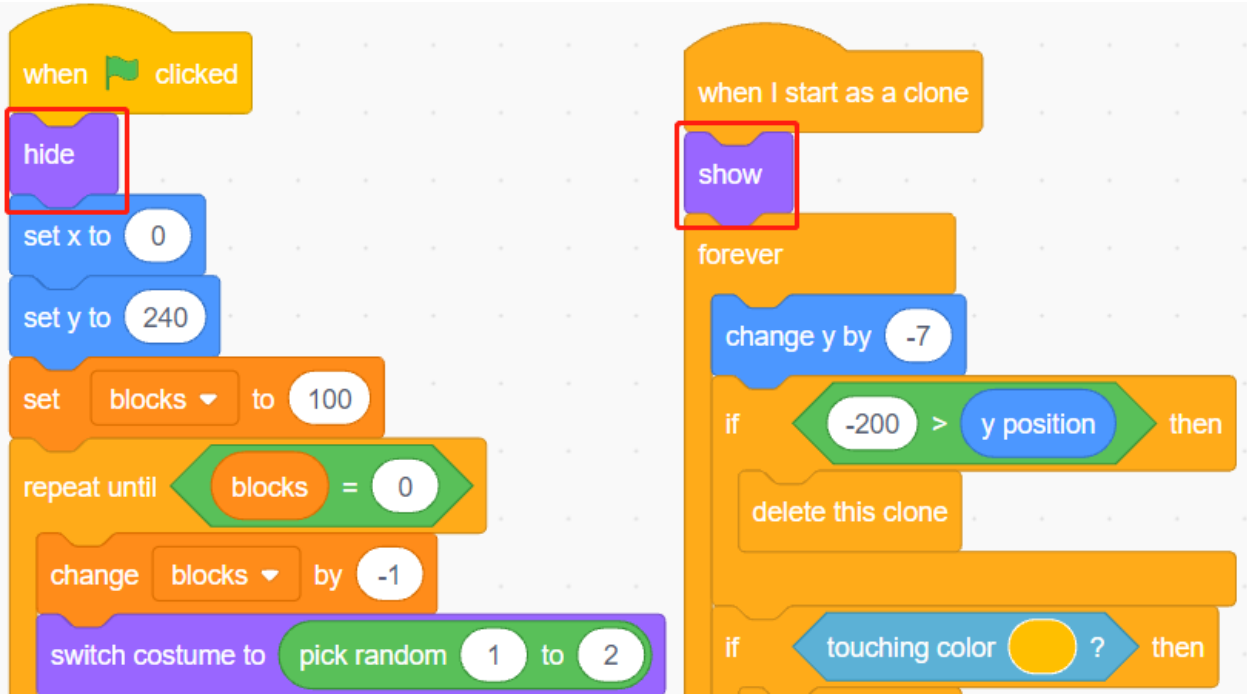
Crea clones del sprite **Baldosa** mientras la variable **bloques** esté disminuyendo, y detén el script cuando bloques sea 0. Aquí se utilizan dos bloques [esperar () segundos], el primero para limitar el intervalo entre los clones de **Baldosa** y el segundo es para permitir que la variable bloques disminuya a 0 sin detener el programa inmediatamente, dando al último sprite baldosa suficiente tiempo para moverse.



Ahora programa el clon del sprite **Baldosa** para que se mueva lentamente hacia abajo y lo elimine cuando llegue a la parte inferior del escenario. El cambio en la coordenada y afecta la velocidad de caída, cuanto mayor sea el valor, más rápida será la velocidad de caída.



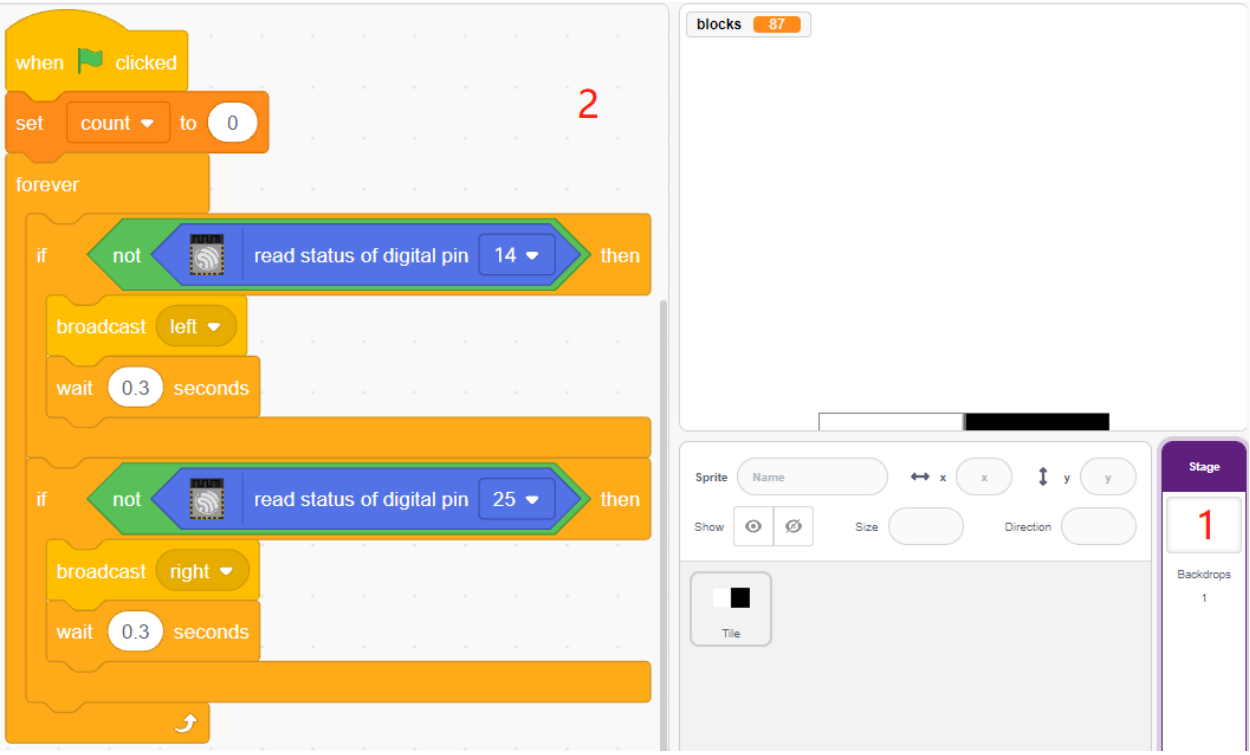
Oculta el cuerpo y muestra el clon.



3. Leer los valores de los 2 módulos IR

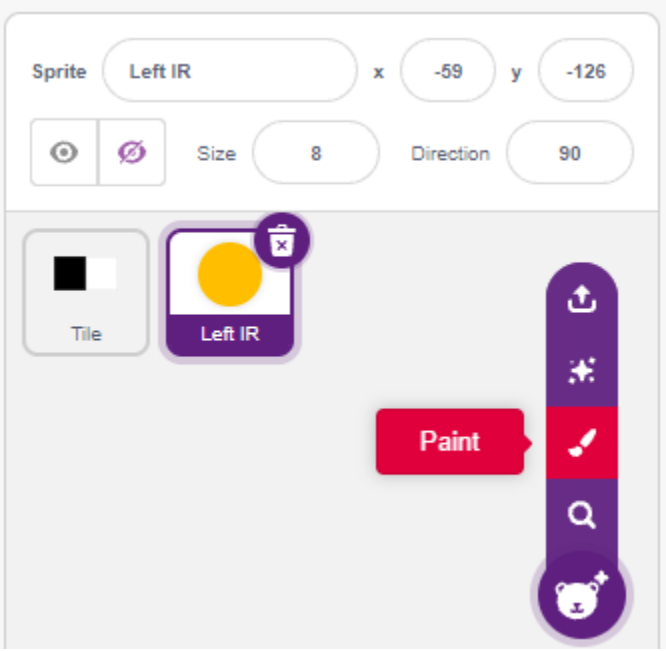
En el fondo, lee los valores de los 2 módulos IR y realiza las acciones correspondientes.

- Si el módulo de evitación de obstáculos IR izquierdo siente tu mano, transmite un mensaje - **izquierda**.
- Si el módulo de evitación de obstáculos IR derecho siente tu mano, transmite un mensaje - **derecha**.

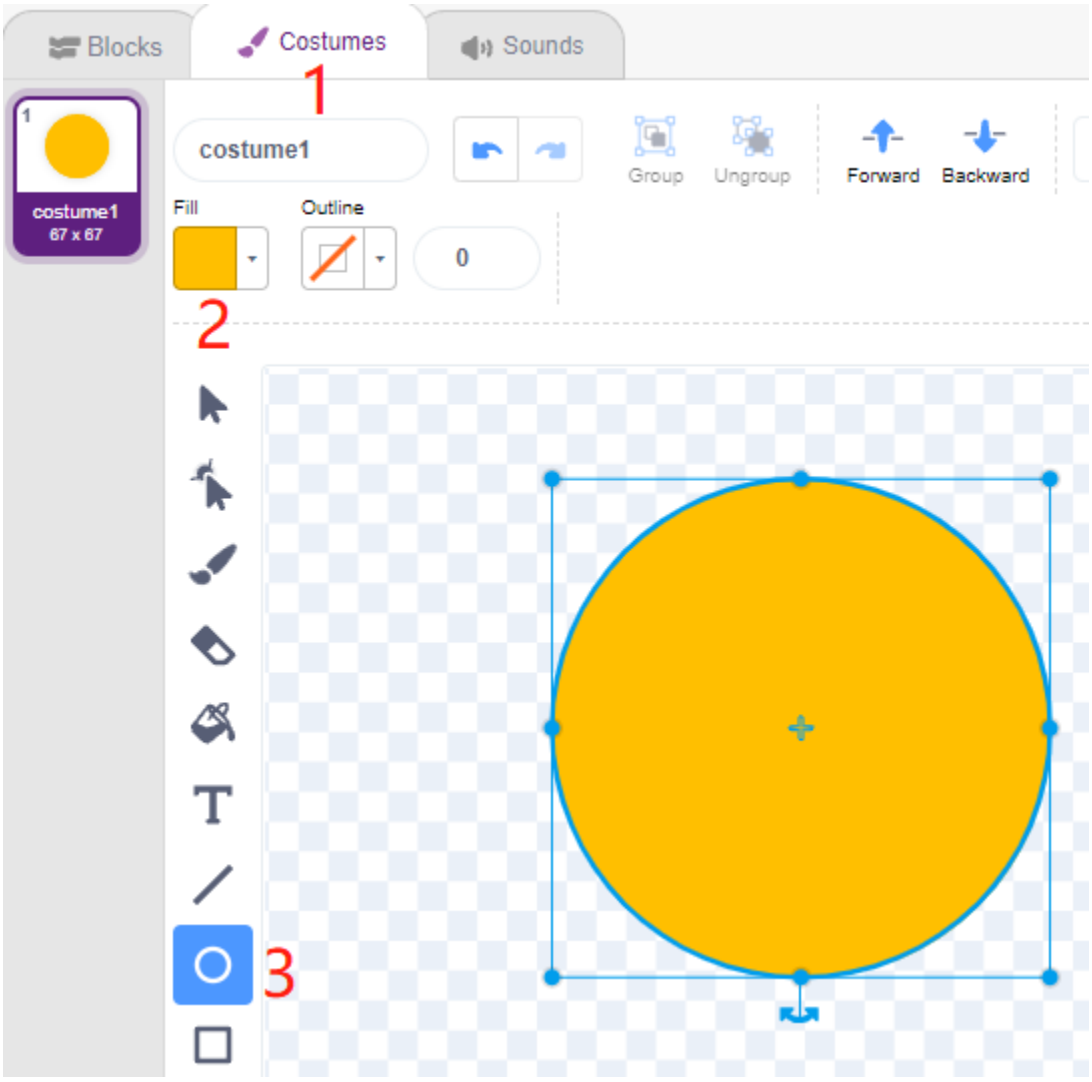


4. Sprite IR Izquierdo

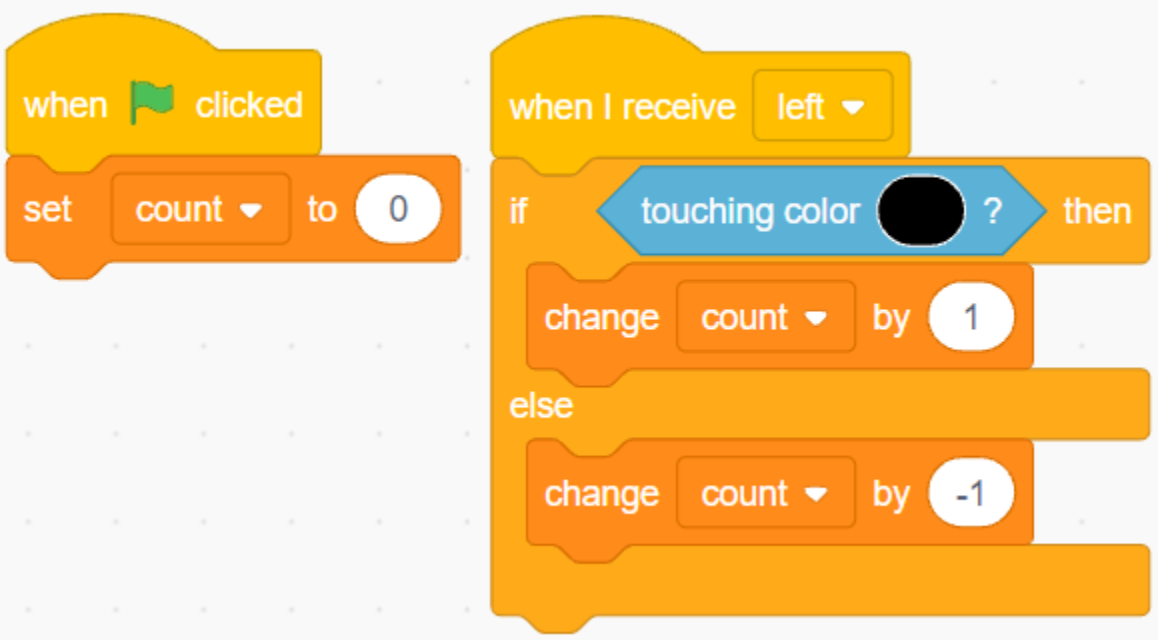
Una vez más, pasa el mouse sobre el icono de **Añadir sprite** y selecciona **Pintar** para crear un nuevo sprite llamado **IR Izquierdo**.



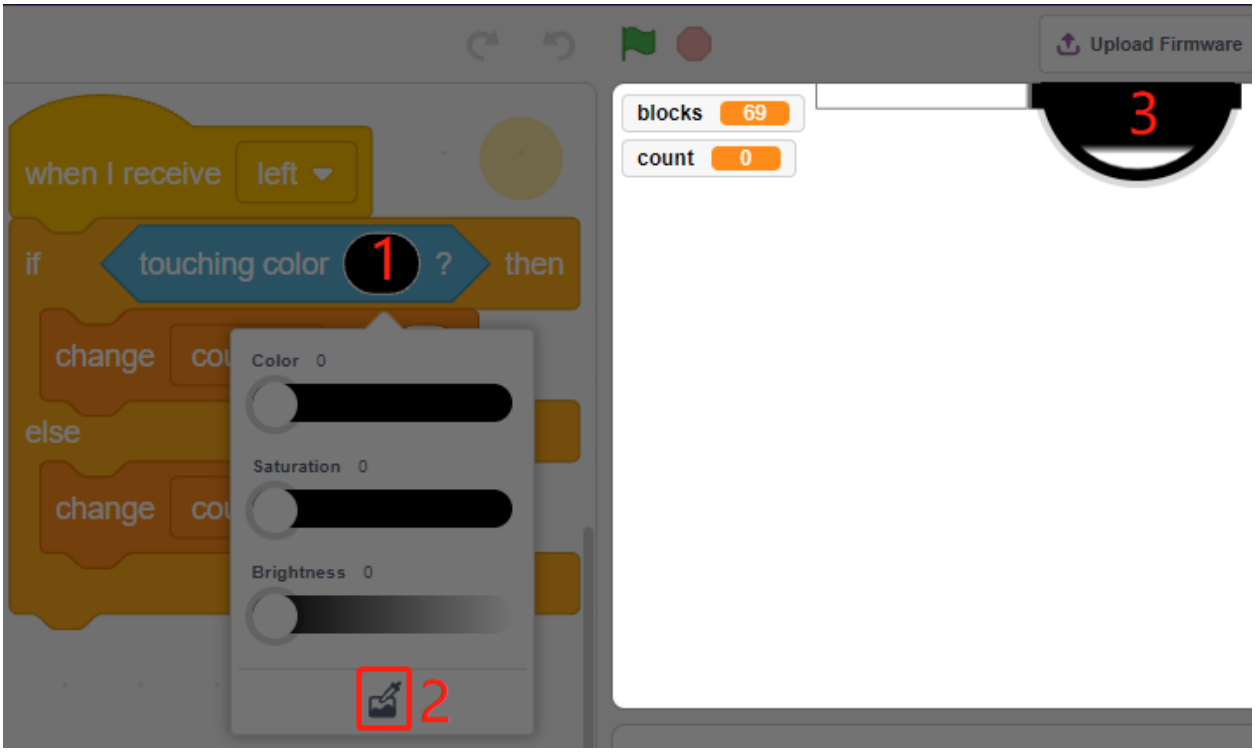
Ve a la página de **Disfraces** del sprite **IR Izquierdo**, selecciona el color de relleno (cualquier color fuera de negro y blanco) y dibuja un círculo.



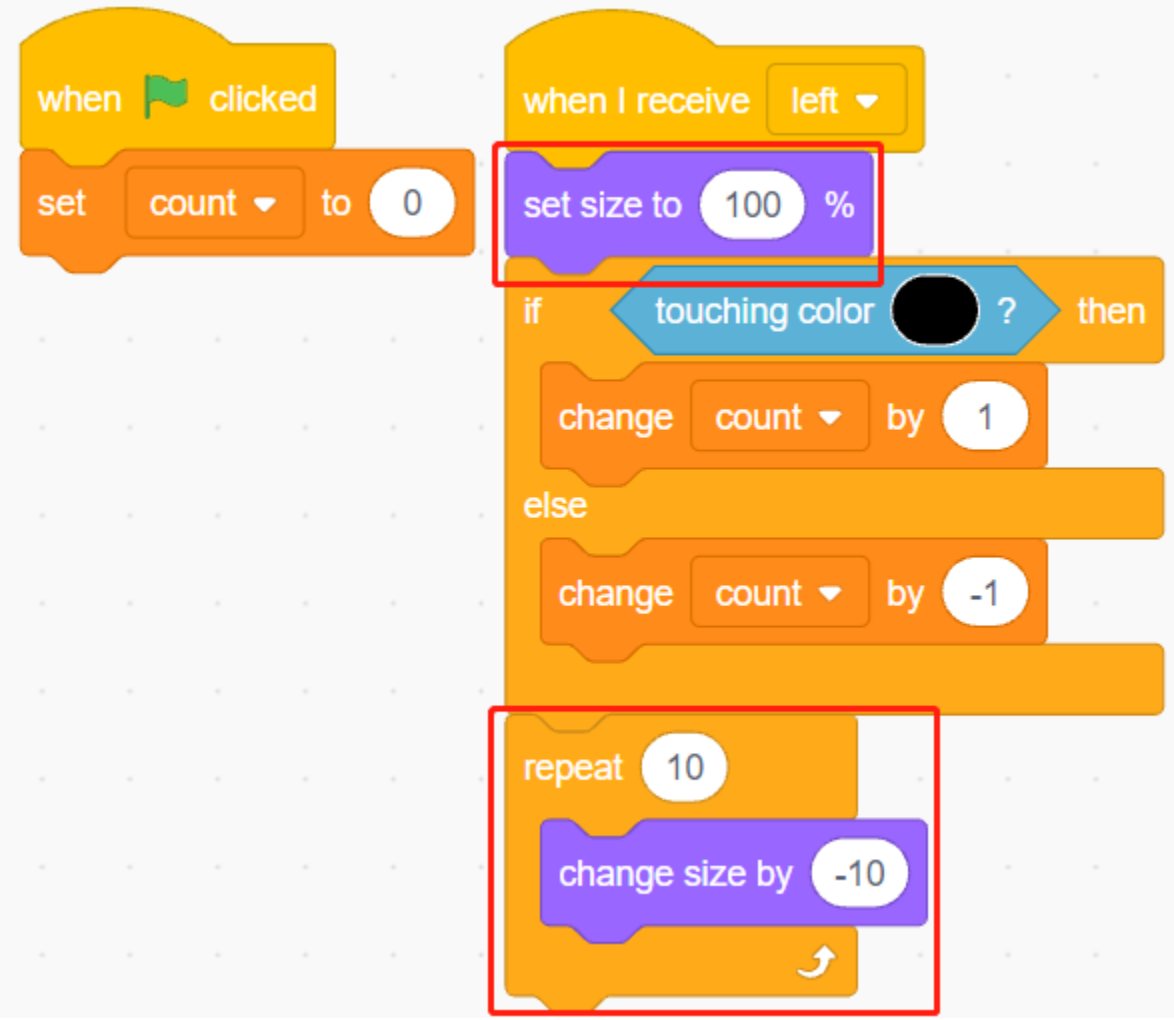
Ahora comienza a programar el sprite **IR Izquierdo**. Cuando se reciba el mensaje - **izquierda** (el módulo receptor IR de la izquierda detecta un obstáculo), entonces determina si se ha tocado el bloque negro del sprite **Baldosa**, y si es así, deja que la variable **cuenta** sume 1, de lo contrario, resta 1.



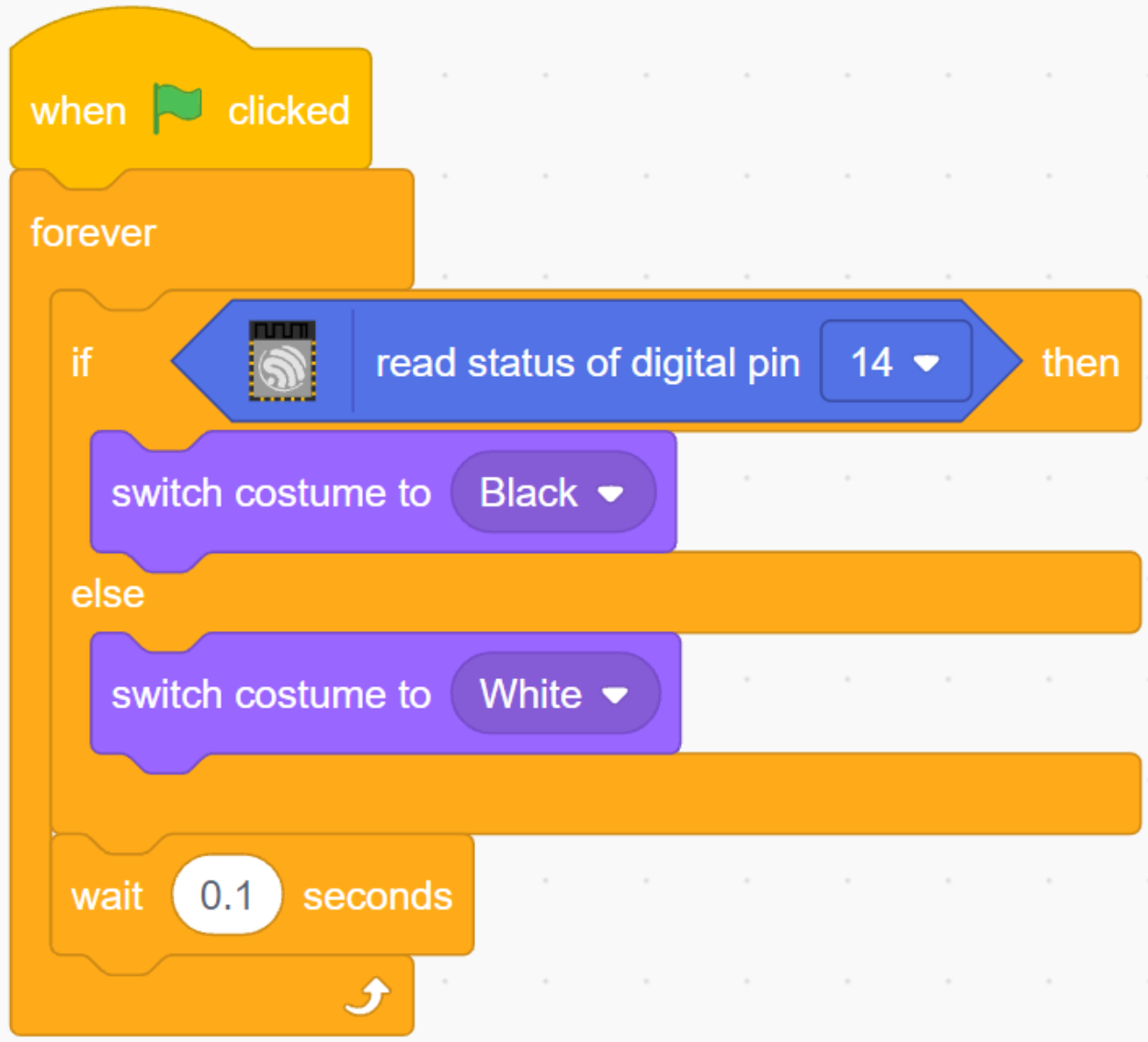
Nota: Necesitas hacer que el sprite **Baldosa** aparezca en el escenario, y luego absorber el color del bloque negro en el sprite **Baldosa**.



Ahora hagamos el efecto de detección (aumentar y disminuir) para **IR Izquierdo**.

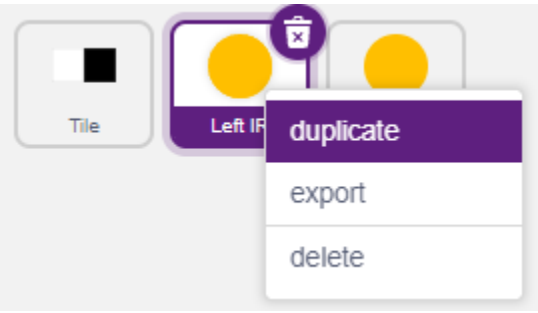


Haz que el sprite **IR Izquierdo** se oculte cuando se haga clic en la bandera verde, se muestre cuando se reciba el mensaje - **izquierda**, y finalmente se oculte de nuevo.

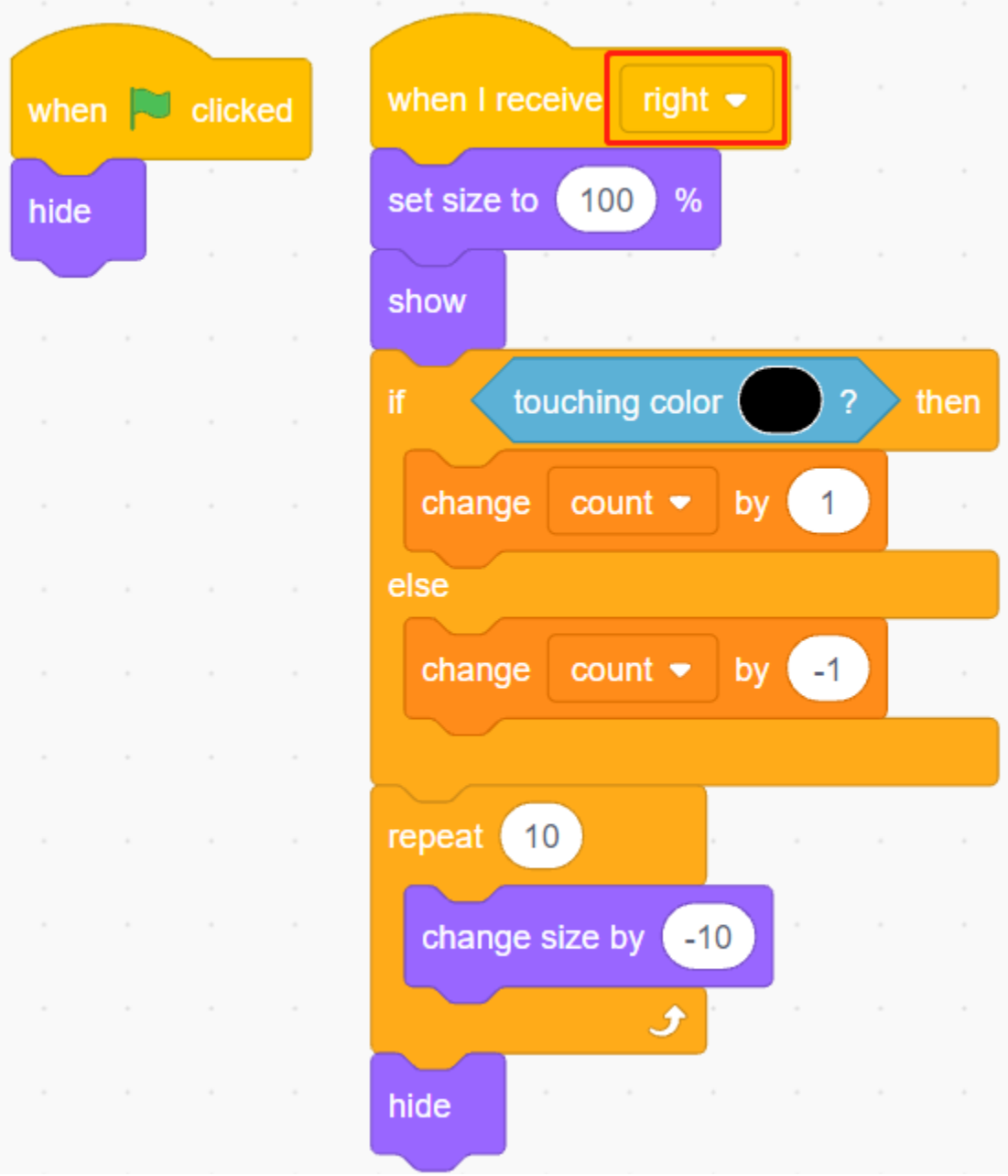


5. Sprite IR Derecho

Copia el sprite **IR Izquierdo** y renómbralo a **IR Derecho**.



Luego cambia el mensaje recibido a - **derecha**.



Ahora toda la programación está completa y puedes hacer clic en la bandera verde para ejecutar el script.

5.22 2.19 JUEGO - Protege Tu Corazón

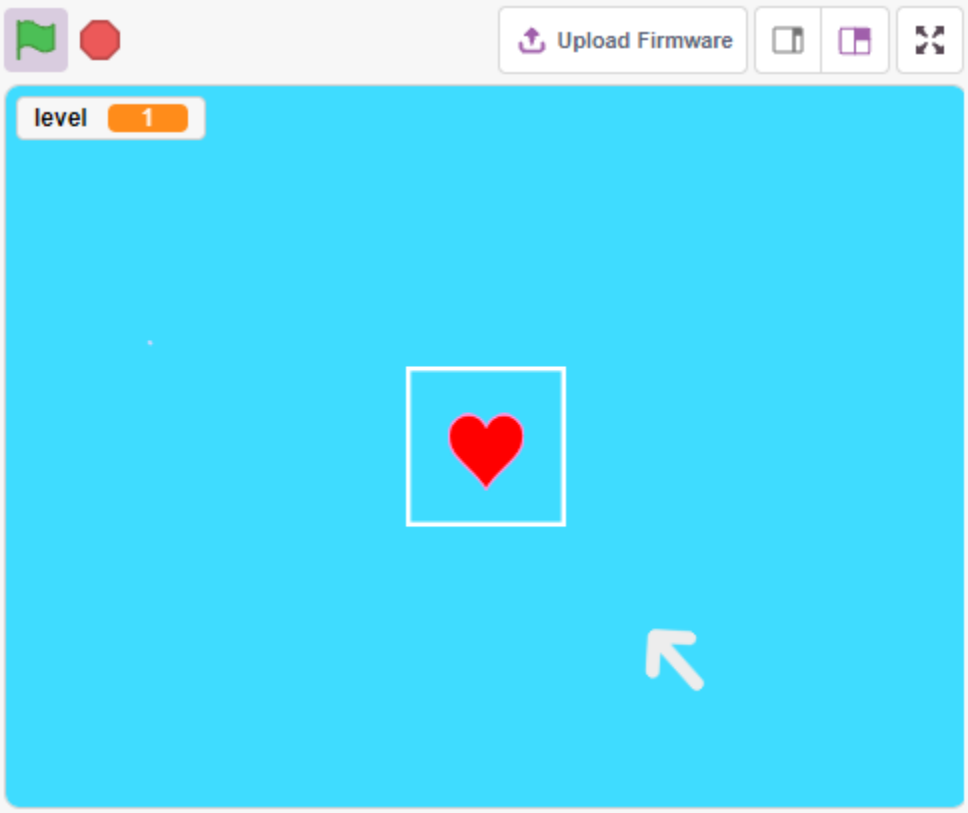
En este proyecto, hagamos un juego que ponga a prueba la velocidad de reacción.

En el escenario, hay un corazón protegido en una caja rectangular, y hay flechas volando hacia este corazón desde cualquier posición en el escenario. El color de la flecha alternará entre negro y blanco al azar y la flecha volará cada vez más rápido.

Si el color de la caja rectangular y el color de la flecha son los mismos, la flecha se bloquea afuera y se suma 1 al nivel; si el color de ambos no es el mismo, la flecha atravesará el corazón y el juego terminará.

Aquí el color de la caja rectangular está controlado por el módulo de Seguimiento de Línea. Cuando el módulo se coloca sobre una superficie negra (una superficie que refleja), el color de la caja rectangular es negro, de lo contrario, es blanco.

Así que necesitas decidir si colocar el módulo de Seguimiento de Línea sobre una superficie blanca o negra según el color de la flecha.



5.22.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

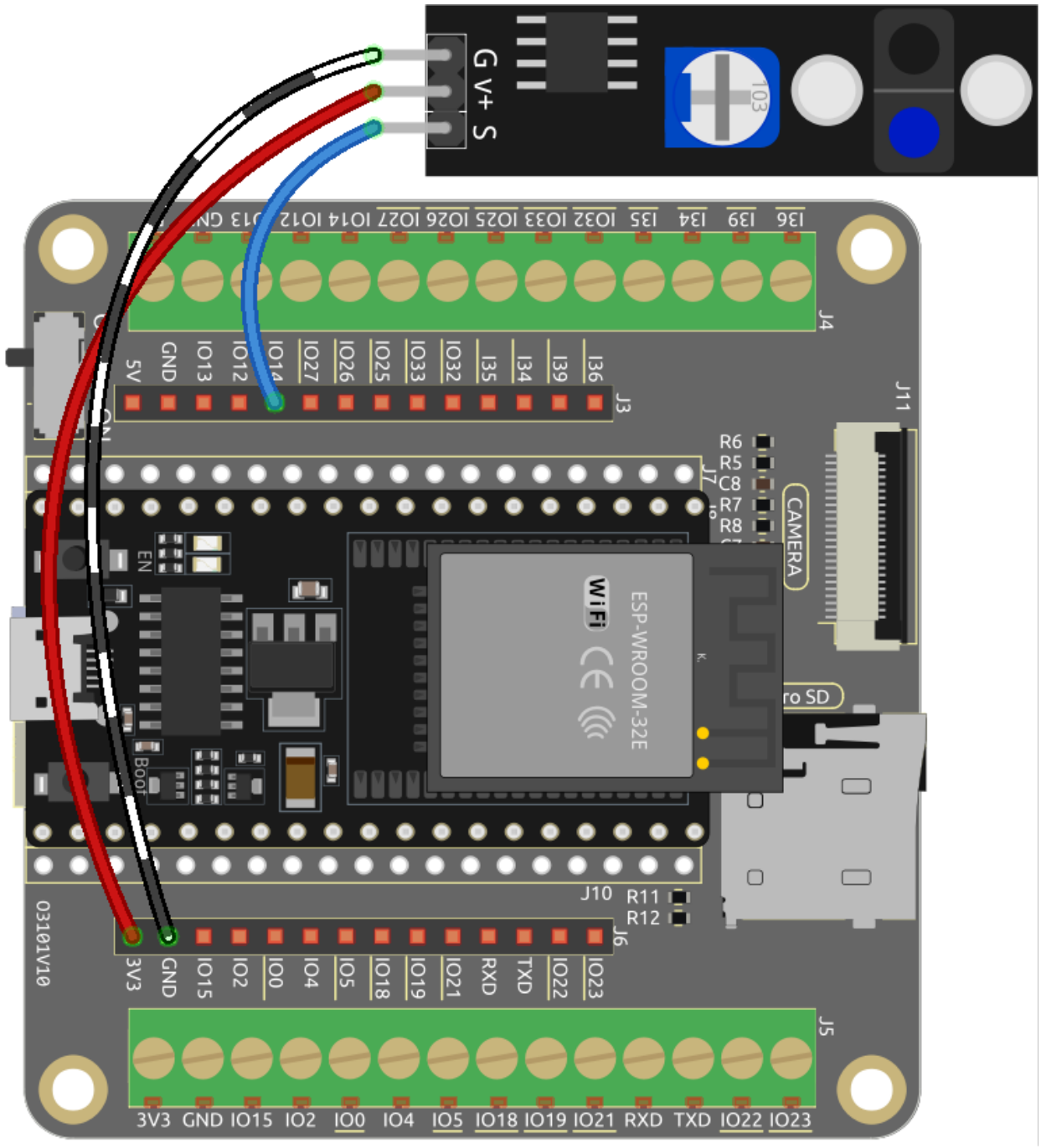
También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Seguimiento de Línea	

5.22.2 Construir el Circuito

Este es un módulo digital de Seguimiento de Línea, cuando detecta una línea negra, emite 1; cuando detecta una línea blanca, emite un valor de 0. Además, puedes ajustar su distancia de detección a través del potenciómetro en el módulo.

Ahora construye el circuito según el diagrama a continuación.



Nota: Antes de empezar el proyecto, necesitas ajustar la sensibilidad del módulo.

Conecta según el diagrama anterior, luego enciende la placa R3 (ya sea directamente en el cable USB o el cable de botón de batería de 9V), sin subir el código.

Ahora pega una cinta eléctrica negra en el escritorio, coloca el módulo de Seguimiento de Línea a una altura de 2cm del escritorio.

Con el sensor mirando hacia abajo, observa el LED de señal en el módulo para asegurarte de que se ilumina en la mesa blanca y se apaga en la cinta negra.

Si no, necesitas ajustar el potenciómetro en el módulo, para que pueda hacer el efecto anterior.

5.22.3 Programación

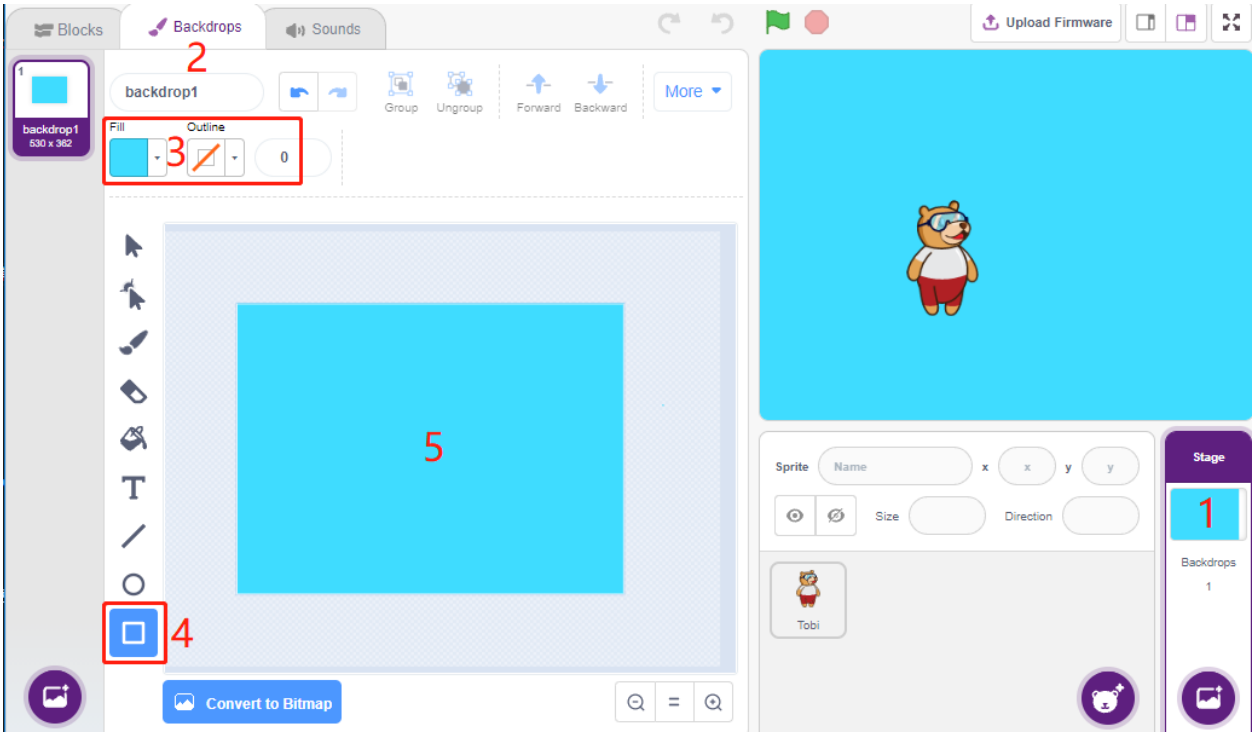
Aquí necesitamos crear 3 sprites, **Corazón**, **Caja Cuadrada** y **Flecha1**.

- **Corazón**: se detiene en medio del escenario, si es tocado por el sprite **Flecha1**, el juego termina.
- **Caja Cuadrada**: Hay dos tipos de disfraces, negro y blanco, y cambiará de disfraz según el valor del módulo de Seguimiento de Línea.
- **Flecha**: vuela hacia el centro del escenario desde cualquier posición en negro/blanco; si su color coincide con el color del sprite **Caja Cuadrada**, se bloquea y vuelve a volar hacia el centro del escenario desde una posición aleatoria; si su color no coincide con el color del sprite **Caja Cuadrada**, pasa a través del sprite **Corazón** y el juego termina.

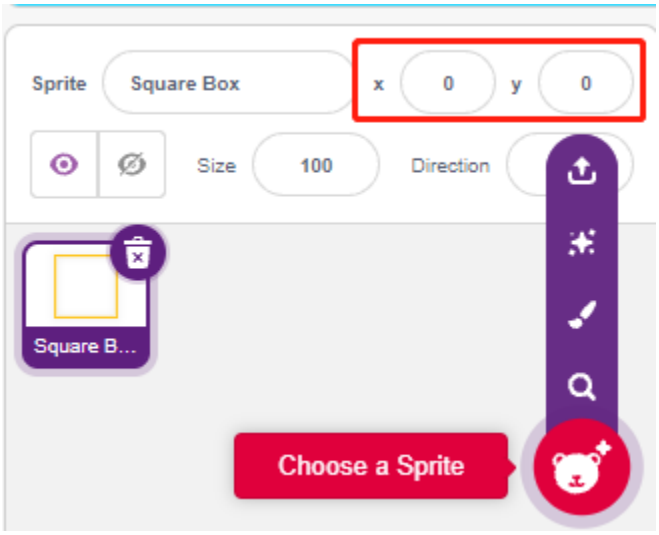
1. Añade el sprite Caja Cuadrada

Dado que el sprite Flecha1 y Caja Cuadrada ambos tienen disfraces blancos, para que se puedan mostrar en el escenario, ahora llena el fondo con un color que puede ser cualquier color excepto negro, blanco y rojo.

- Haz clic en **Backdrop1** para ir a su página de **Fondos**.
- Selecciona el color que quieras llenar.
- Usa la herramienta **Rectángulo** para dibujar un rectángulo del mismo tamaño que el tablero de dibujo.

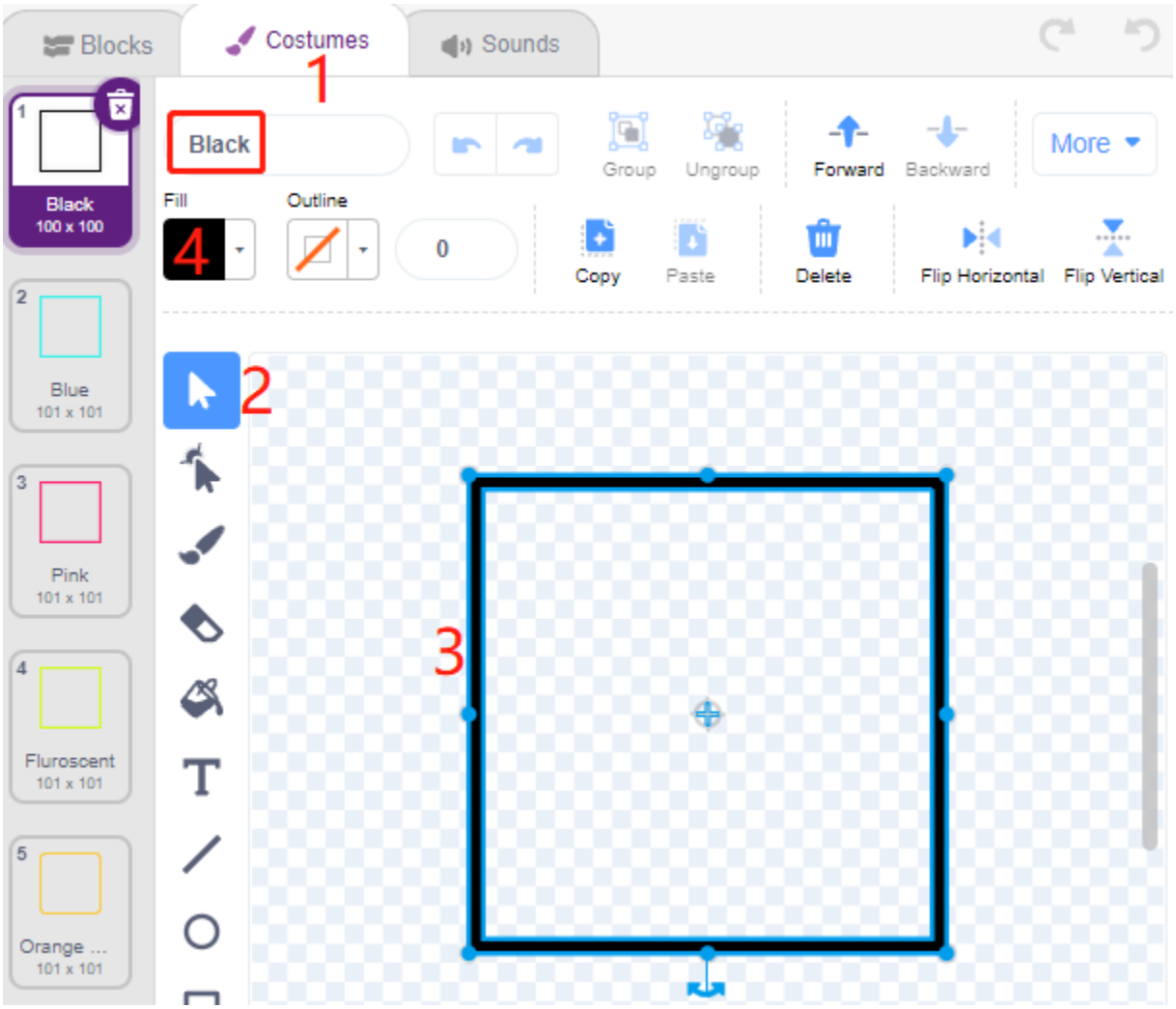


Elimina el sprite predeterminado, usa el botón **Elegir un Sprite** para añadir el sprite **Caja Cuadrada** y ajusta sus coordenadas x e y a (0, 0).

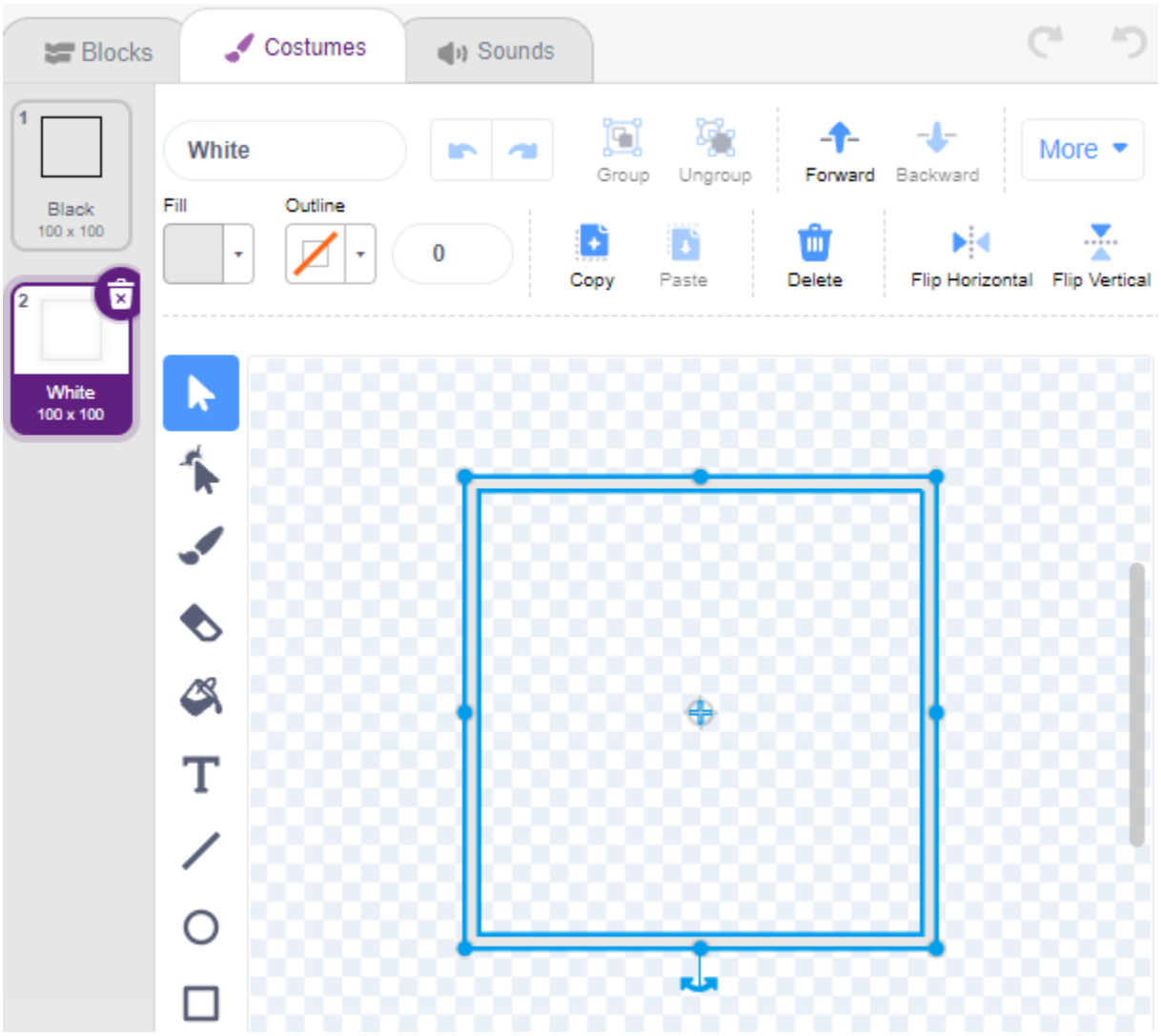


Ve a la página de **Disfraces** del sprite **Caja Cuadrada** y configura los disfraces negro y blanco.

- Haz clic en la herramienta de selección
- Selecciona el rectángulo en el lienzo
- Selecciona el color de relleno como negro
- y nombra el disfraz **Negro**

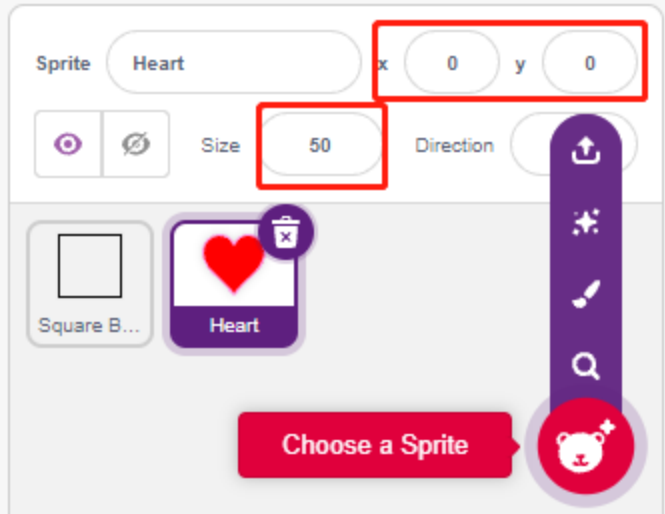


Selecciona el segundo disfraz, configura el color de relleno a blanco, nómbralo Blanco y elimina el resto del disfraz.

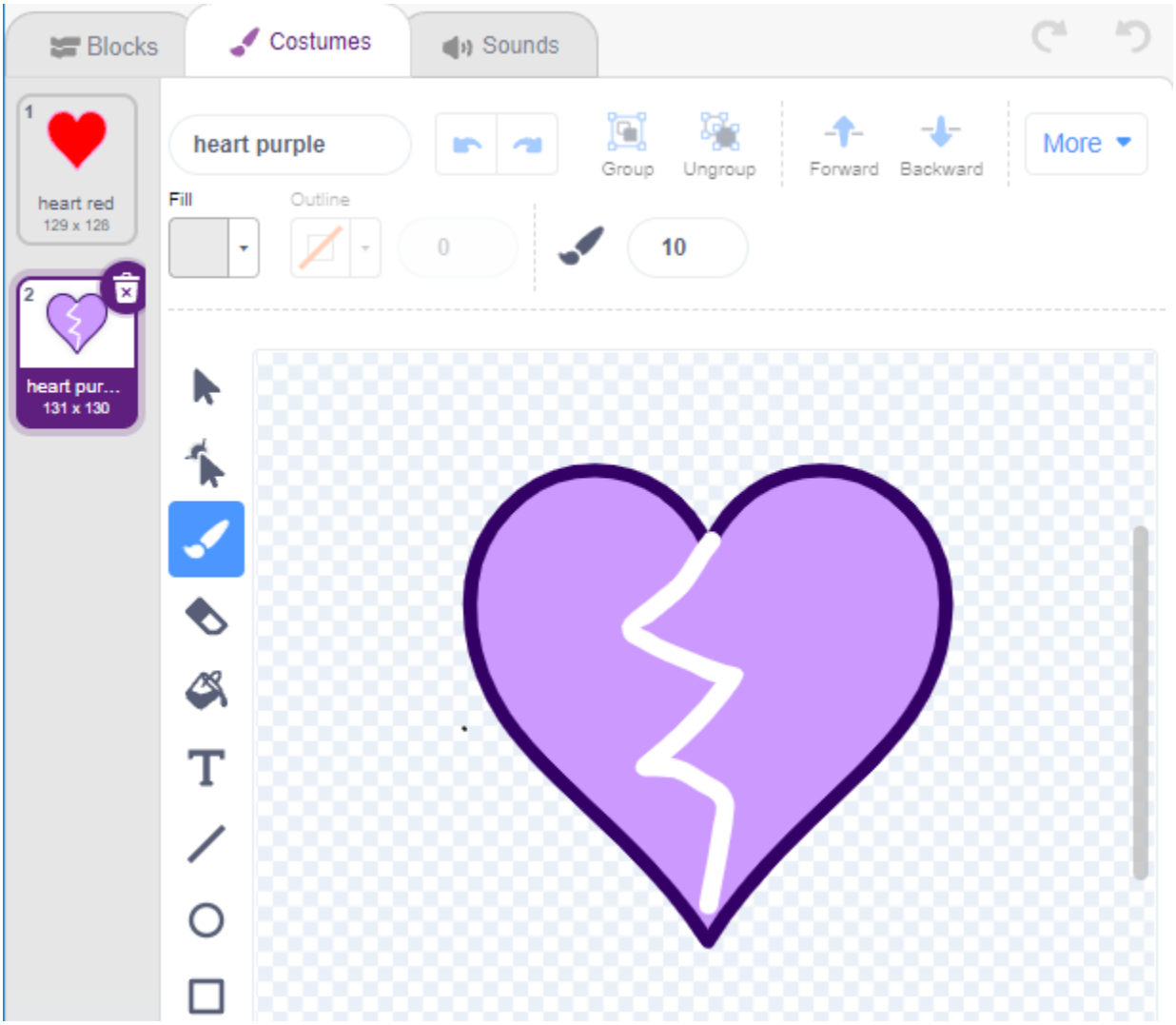


2. Añade el sprite Corazón

También añade un sprite **Corazón**, ajusta su posición a (0, 0) y reduce su tamaño para que parezca estar ubicado dentro de la Caja Cuadrada.

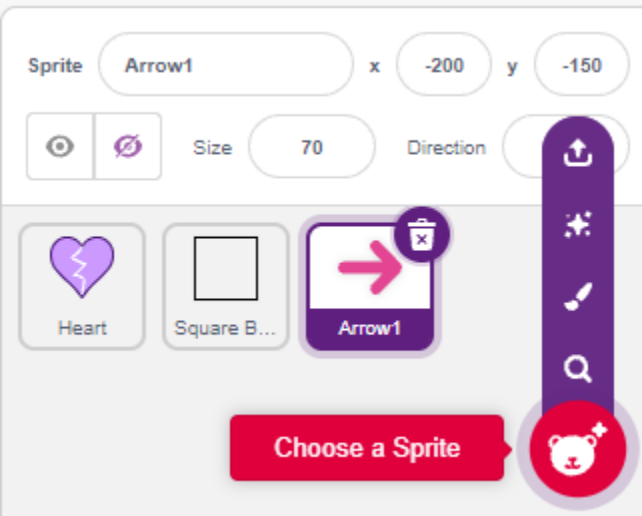


En la página de **Disfraces**, ajusta el disfraz morado del corazón para que parezca estar roto.

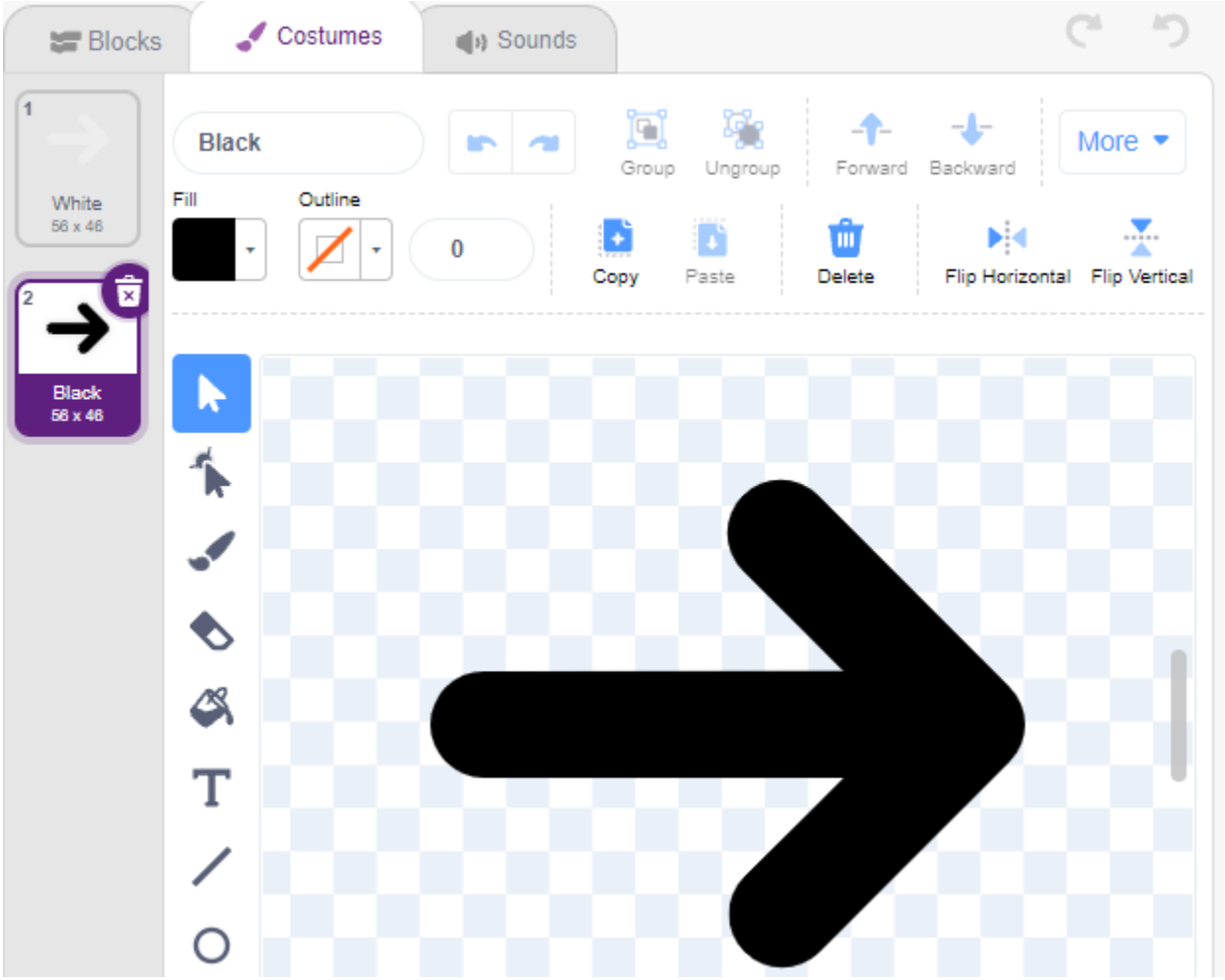


3. Añade el sprite Flecha1

Añade un sprite **Flecha1**.



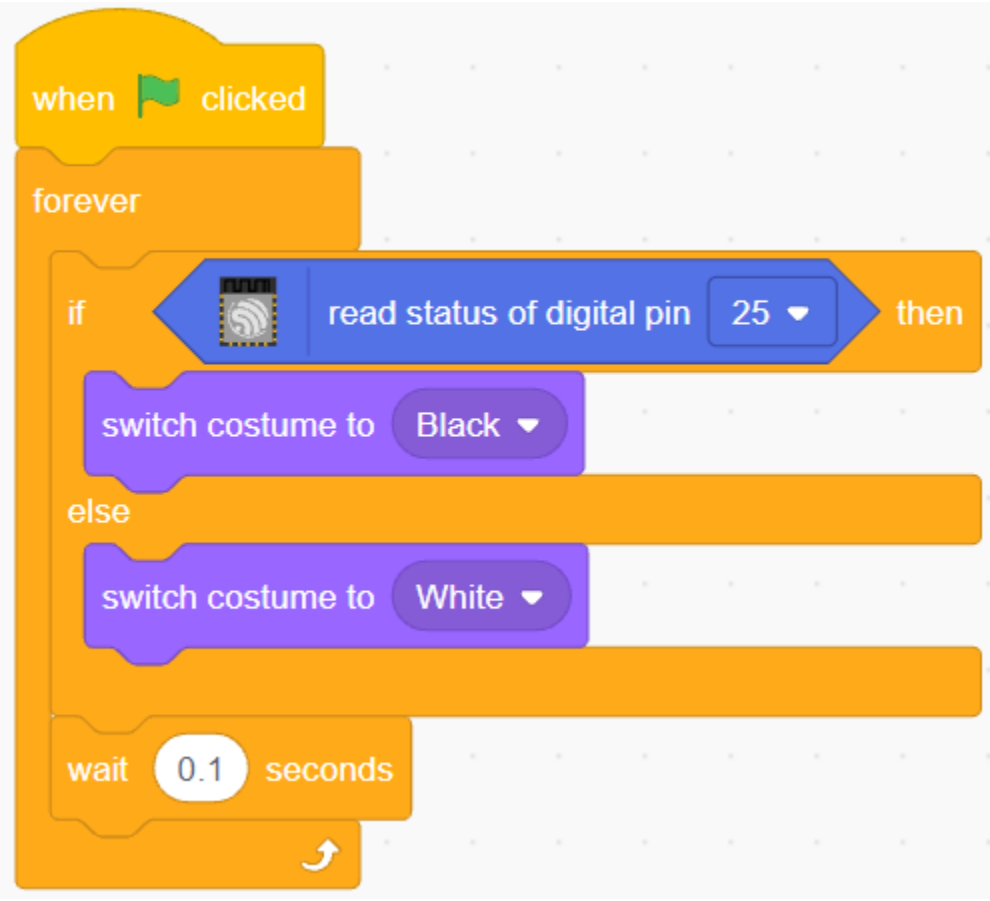
En la página de **Disfraces**, mantén y copia el disfraz que mira hacia la derecha y configura su color a negro y blanco.



4. Programación para el sprite Caja Cuadrada

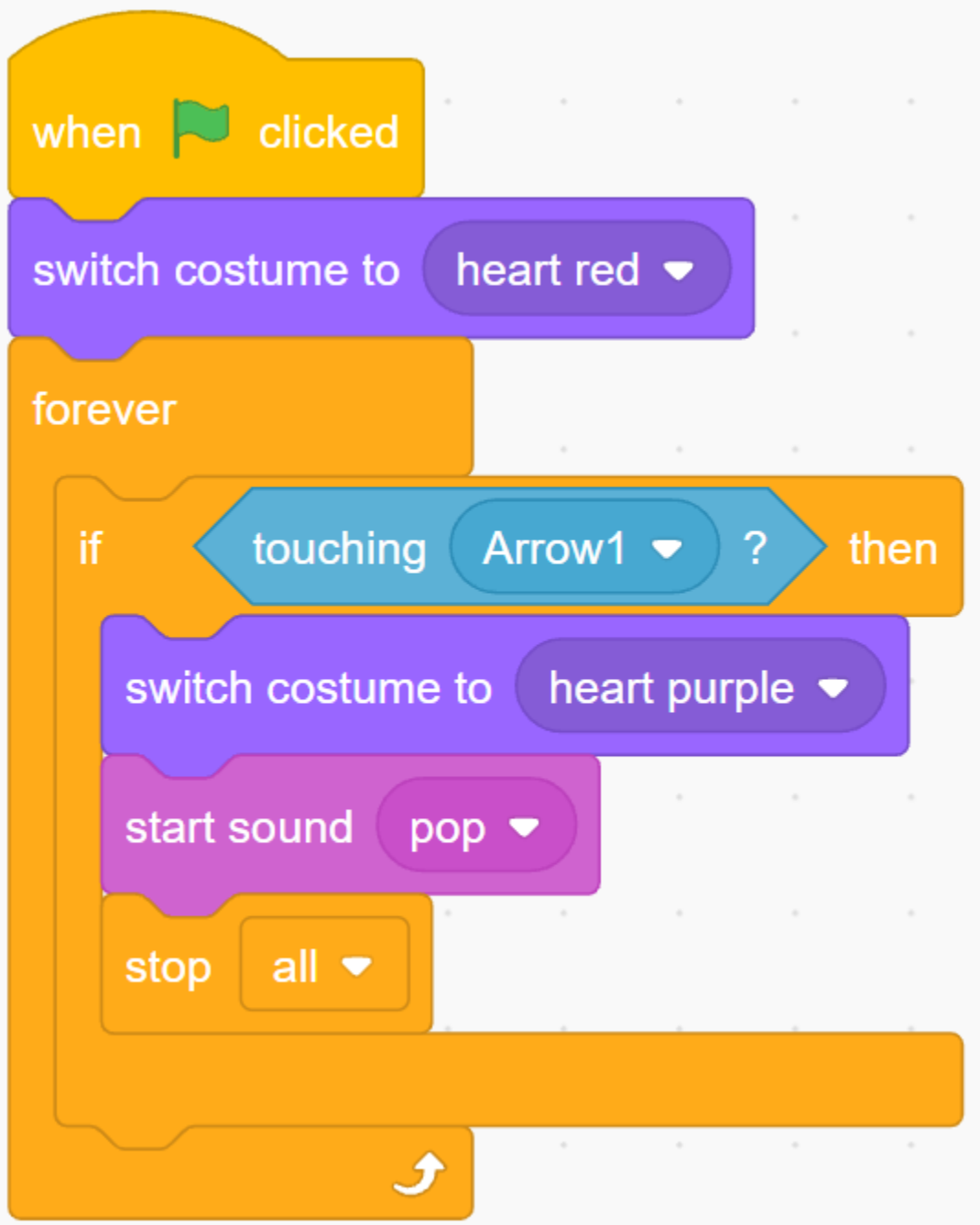
Vuelve a la página de **Bloques** y programa el sprite **Caja Cuadrada**.

- Así que cuando el valor del pin digital 2 (módulo de Seguimiento de Línea) es 1 (línea negra detectada), entonces cambia el disfraz a **Negro**.
- De lo contrario, cambia el disfraz a **Blanco**.



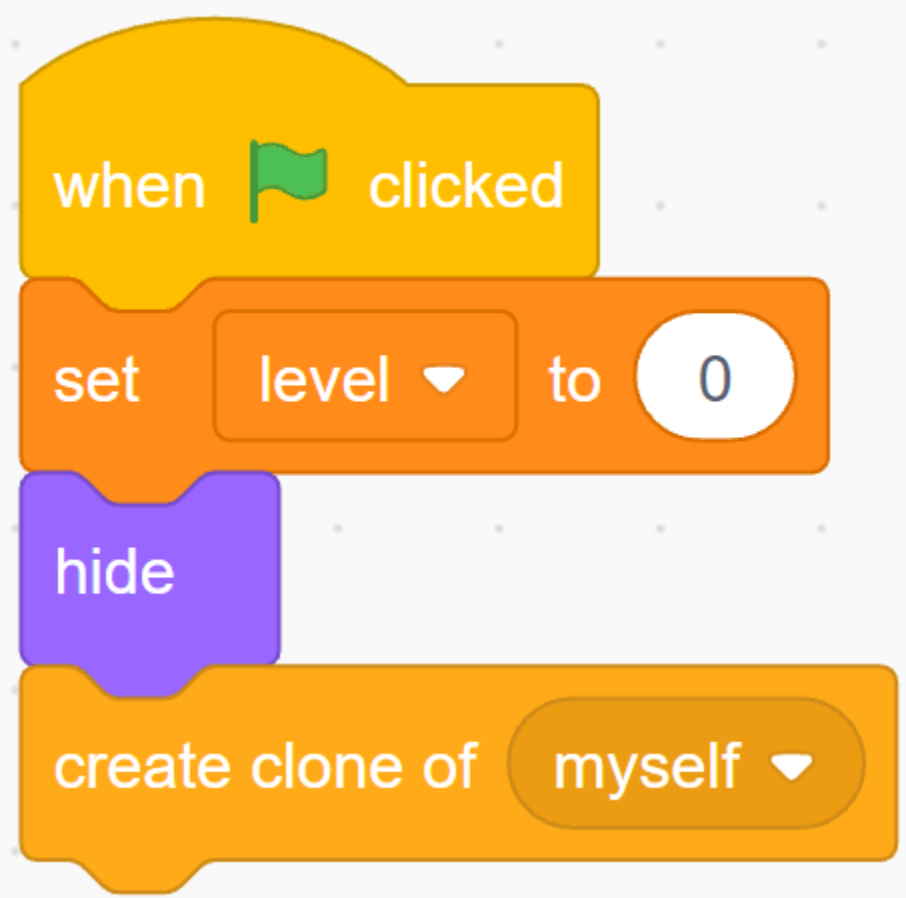
5. Programación para el sprite Corazón

El sprite **Corazón** está protegido dentro de **Caja Cuadrada**, y por defecto es un disfraz rojo. Cuando el sprite Flecha1 lo toca, el juego termina.



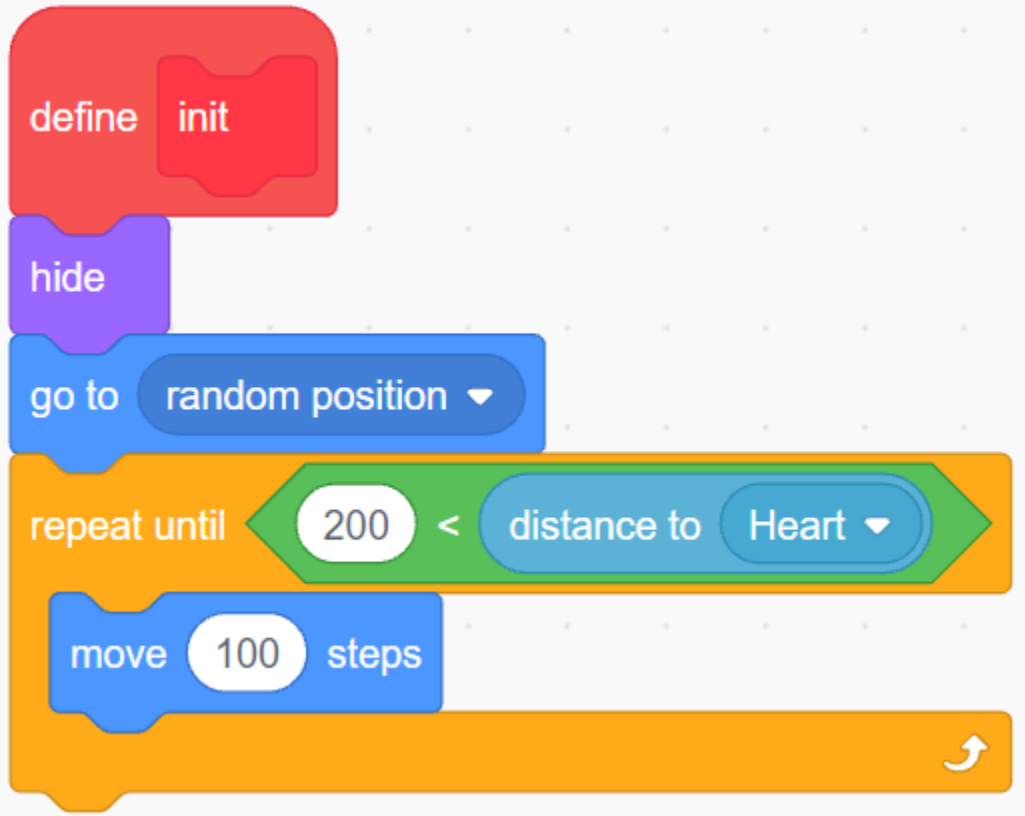
6. Programación para el sprite Flecha1

Haz que el sprite **Flecha1** se oculte y cree un clon cuando se haga clic en la bandera verde.

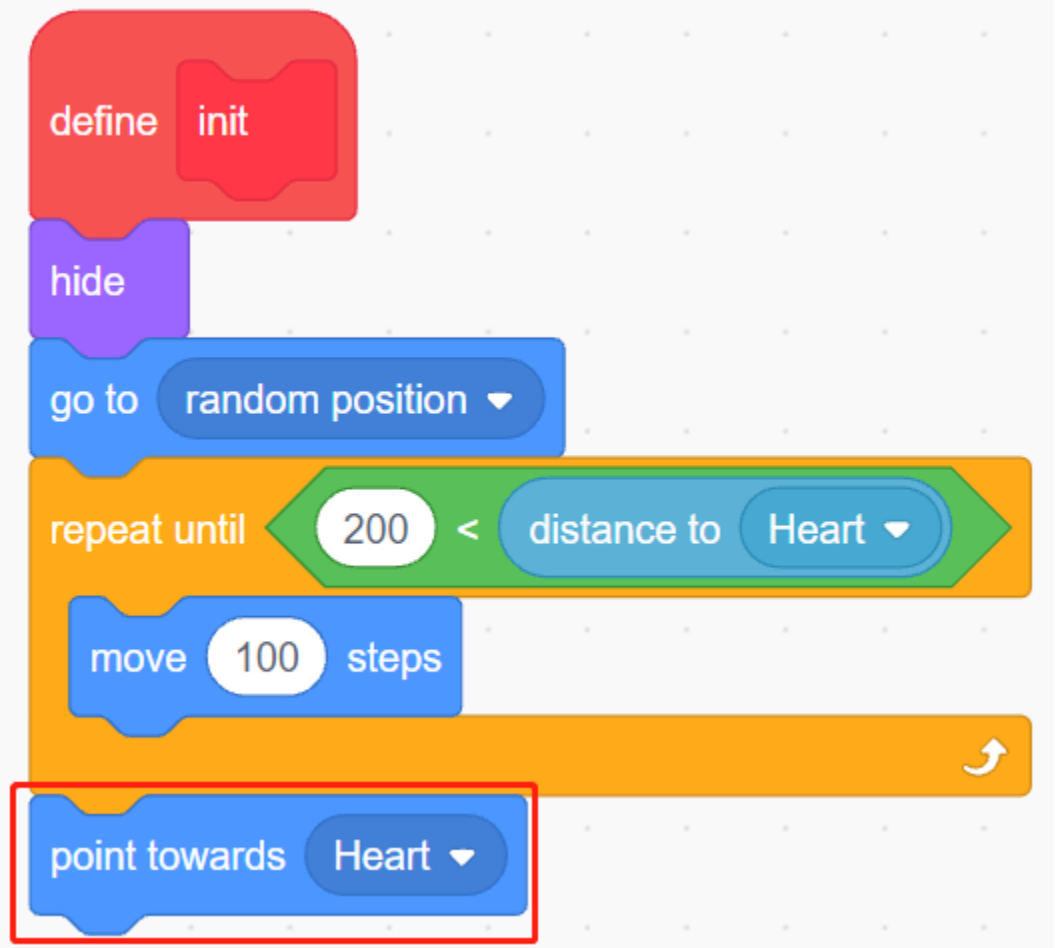


Crea un bloque [init] para inicializar la posición, orientación y color del sprite **Flecha1**.

Aparece en una ubicación aleatoria, y si la distancia entre él y el sprite **Corazón** es menor de 200, se mueve hacia afuera hasta que la distancia sea mayor de 200.

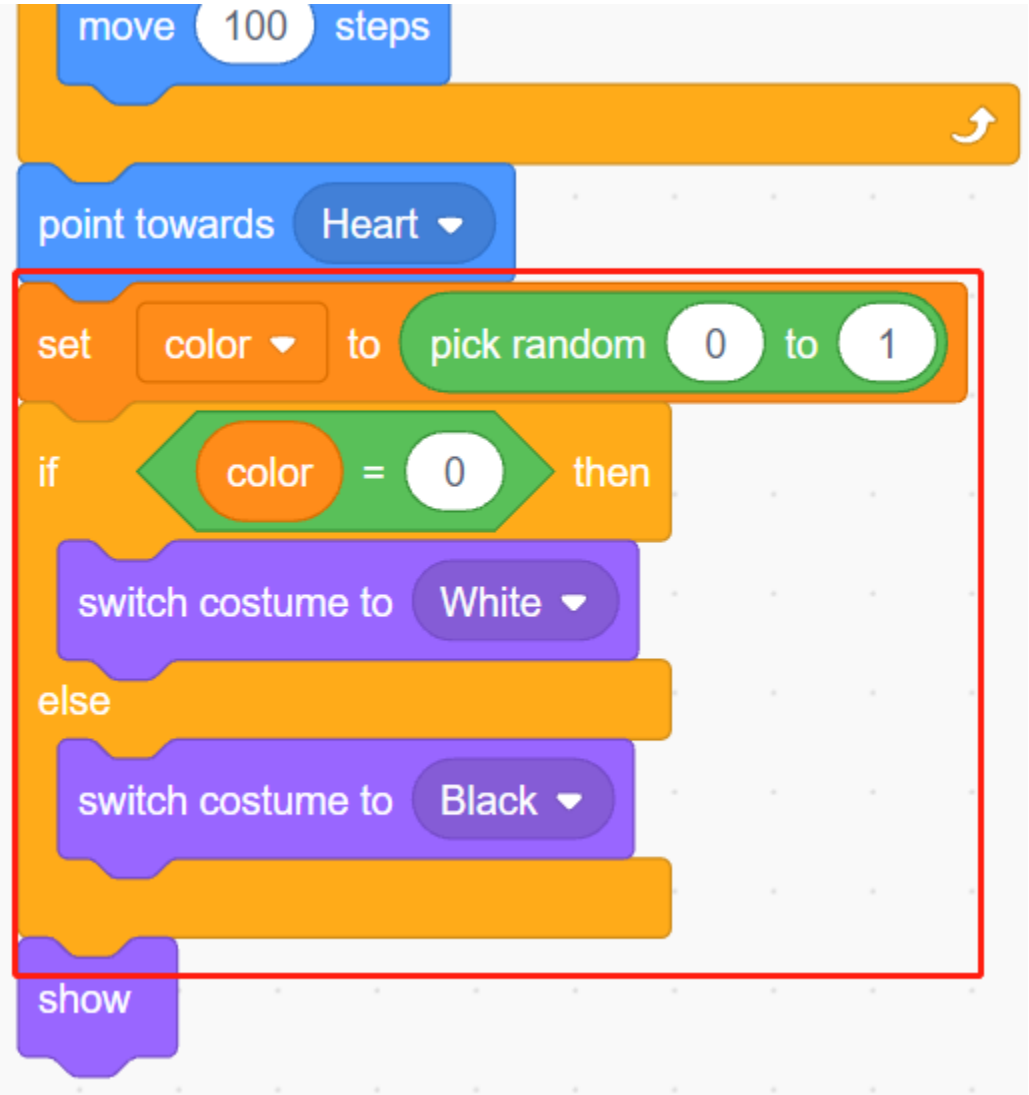


Configura su dirección para enfrentar al sprite **Corazón**.

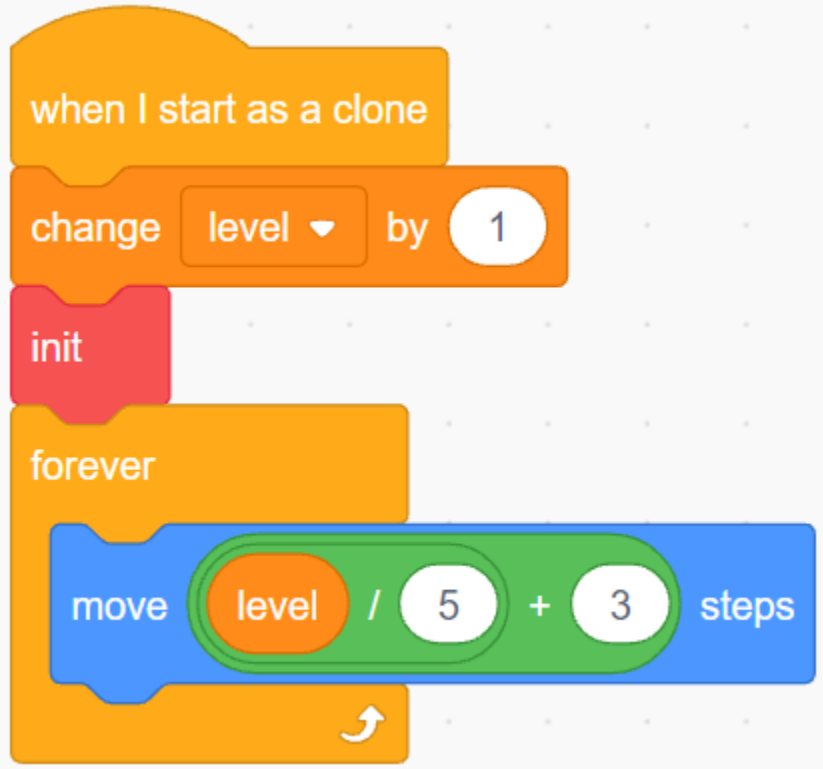


Haz que su color alterne aleatoriamente entre negro/blanco.

- Variable color es 0, cambia el disfraz a **Blanco**.
- Variable color es 1, cambia el disfraz a **Negro**.

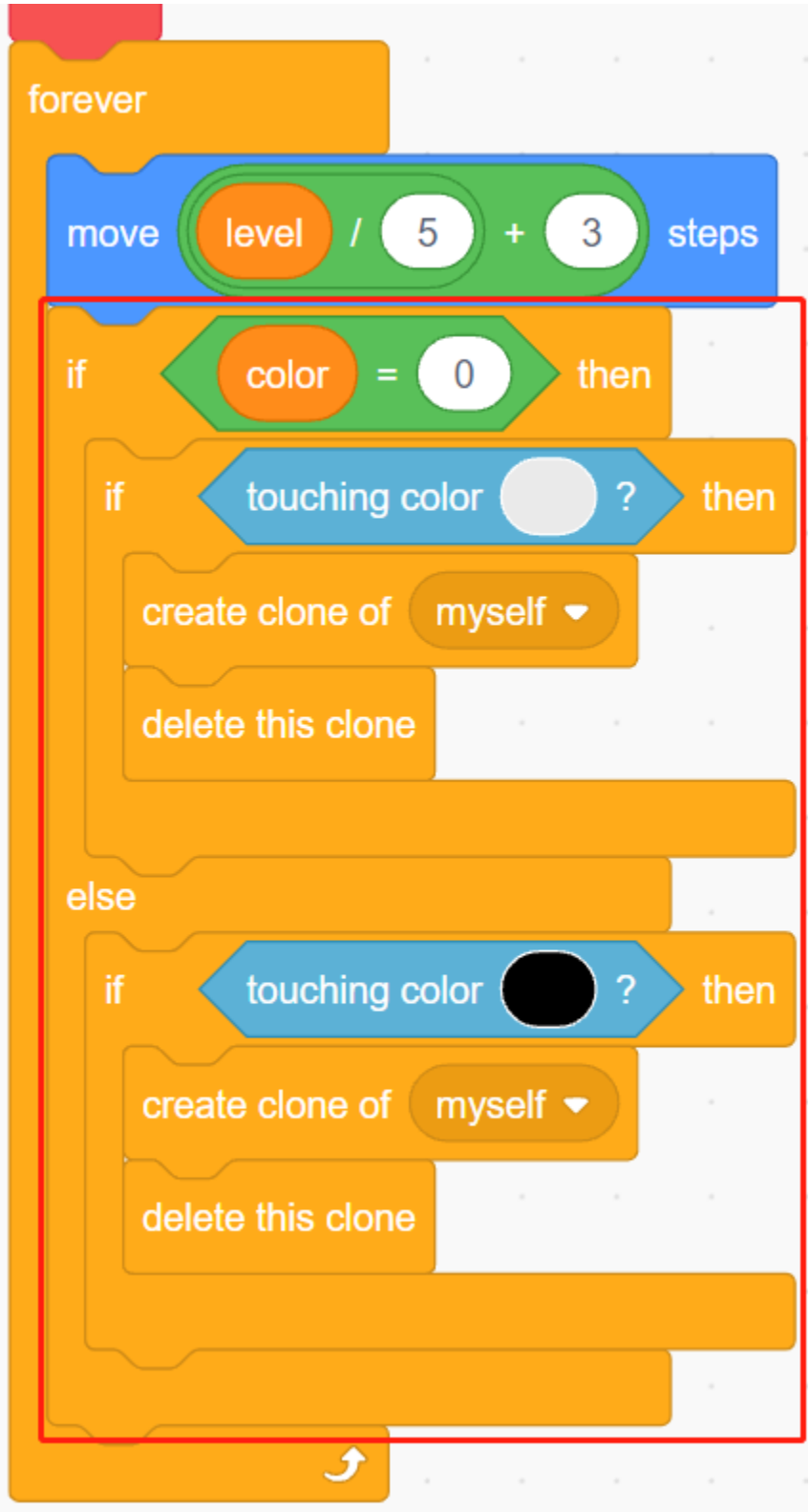


Ahora deja que comience a moverse, se moverá más rápido a medida que aumenta el valor de la variable **nivel**.

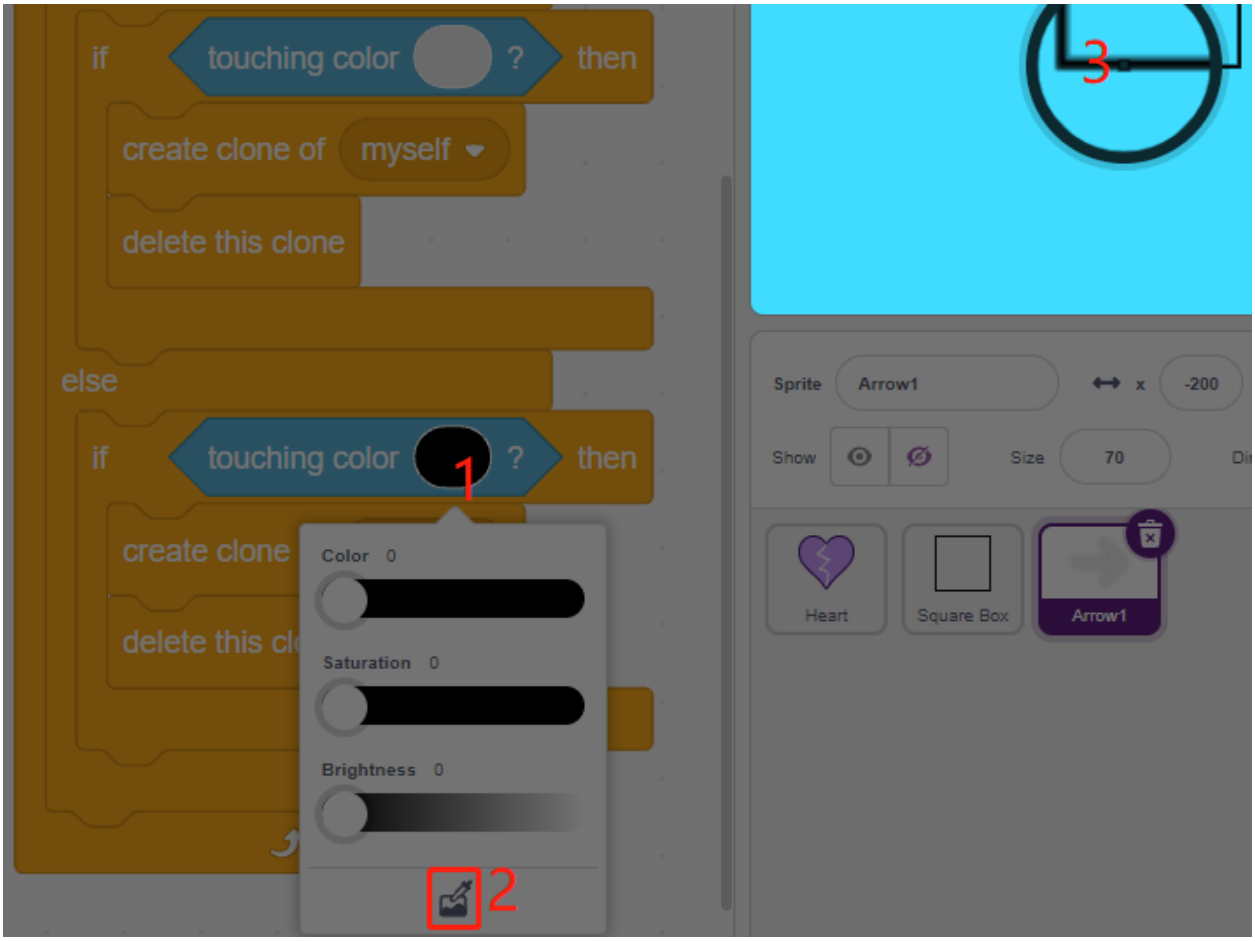


Ahora configura su efecto de colisión con el sprite **Caja Cuadrada**.

- Si el sprite **Flecha1** y el sprite **Caja Cuadrada** tienen el mismo color (que se modificará según el valor del módulo de Seguimiento de Línea), ya sea negro o blanco, se crea un nuevo clon y el juego continúa.
- Si sus colores no coinciden, el sprite **Flecha1** continúa moviéndose y el juego termina cuando golpea al sprite **Corazón**.



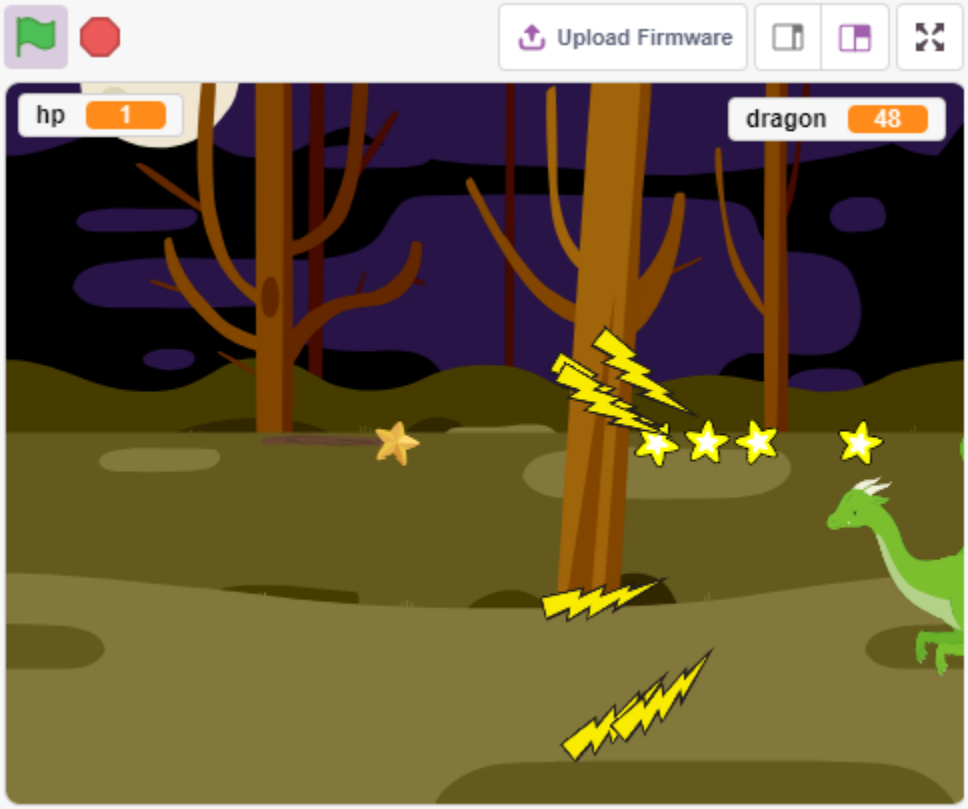
Nota: Los dos bloques [tocar color()] necesitan recoger los disfraces negro/blanco de Caja Cuadrada por separado.



5.23 2.20 JUEGO - Matar al Dragón

Aquí, utilizamos el joystick para jugar a un juego de matar dragones.

Al hacer clic en verde, el dragón flotará arriba y abajo en el lado derecho y lanzará fuego intermitentemente. Necesitas usar el joystick para controlar el movimiento de la varita mágica y lanzar ataques estelares al dragón, evitando las llamas que dispara, y finalmente derrotarlo.



5.23.1 Componentes Necesarios

Para este proyecto, necesitaremos los siguientes componentes.

Es definitivamente conveniente comprar un kit completo, aquí está el enlace:

Nombre	ELEMENTOS EN ESTE KIT	ENLACE
Kit de Inicio ESP32	320+	

También puedes comprarlos por separado en los enlaces a continuación.

INTRODUCCIÓN AL COMPONENTE	ENLACE DE COMPRA
ESP32 WROOM 32E	
Extensión de Cámara ESP32	-
Cables Puente	
Módulo de Joystick	

5.23.2 Construir el Circuito

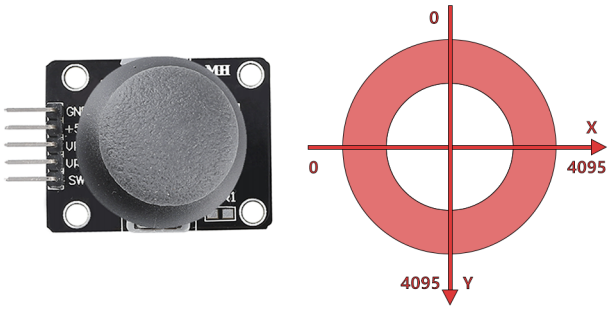
Un joystick es un dispositivo de entrada compuesto por un palo que pivota sobre una base e informa su ángulo o dirección al dispositivo que está controlando. Los joysticks se utilizan a menudo para controlar videojuegos y robots.

Para comunicar un rango completo de movimiento al ordenador, un joystick necesita medir la posición del palo en dos ejes: el eje X (de izquierda a derecha) y el eje Y (de arriba abajo).

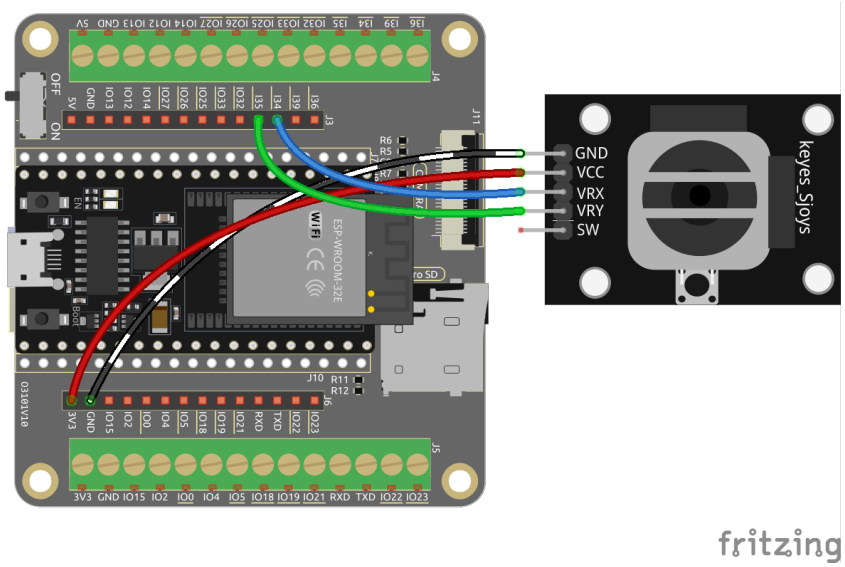
Las coordenadas de movimiento del joystick se muestran en la siguiente figura.

Nota:

- La coordenada x es de izquierda a derecha, el rango es 0-1023.
- La coordenada y es de arriba abajo, el rango es 0-1023.



Ahora construye el circuito según el siguiente diagrama.

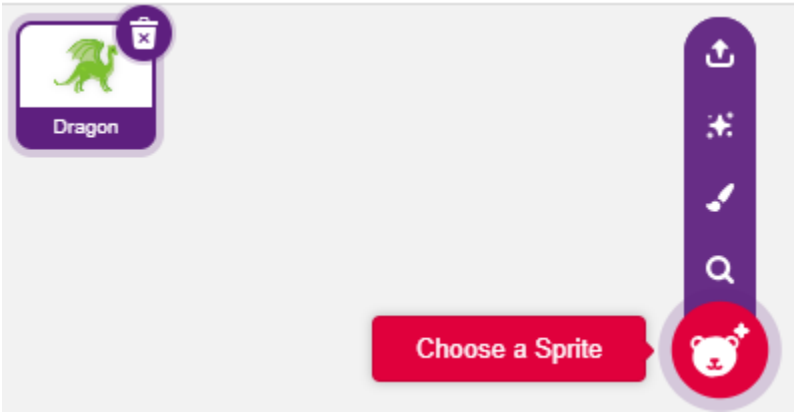


5.23.3 Programación

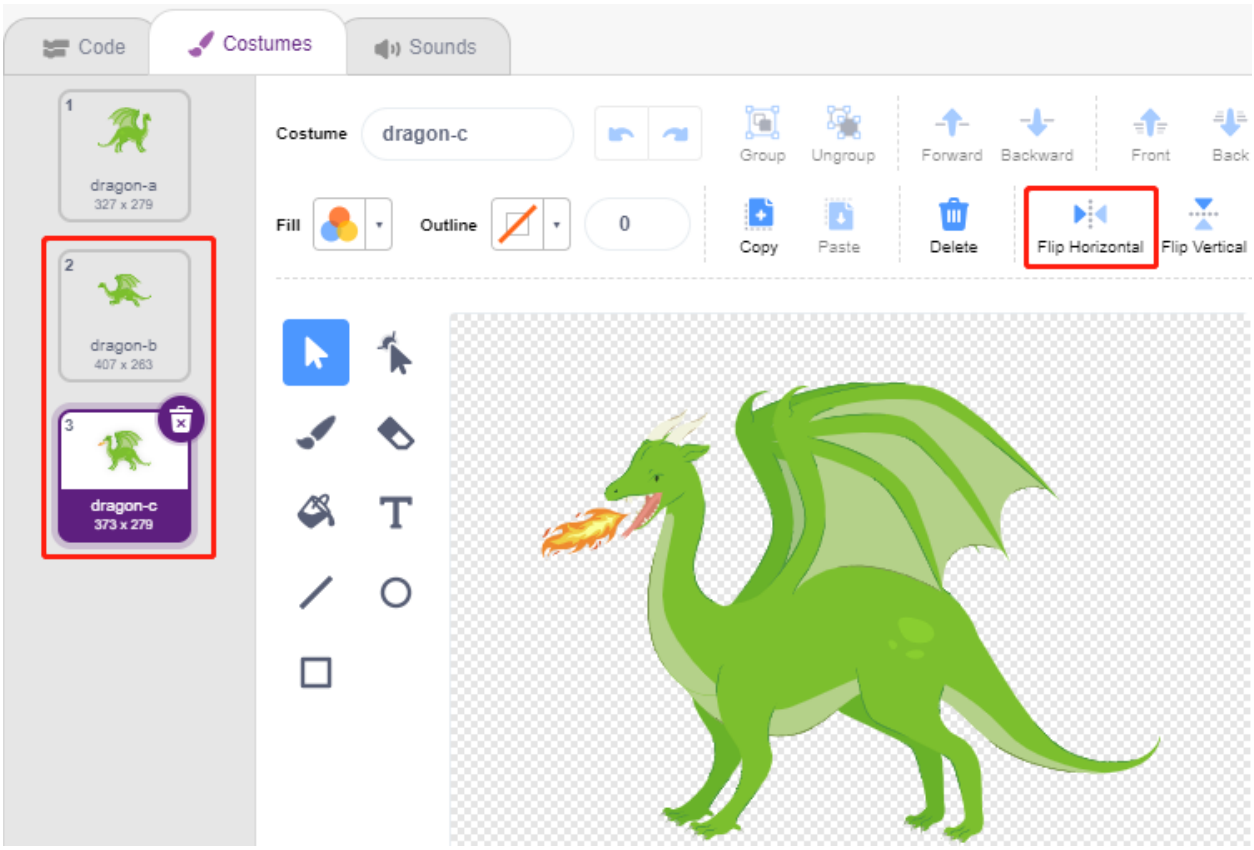
1. Dragón

Se añade el fondo **Bosques** a través del botón **Elegir un Fondo**.

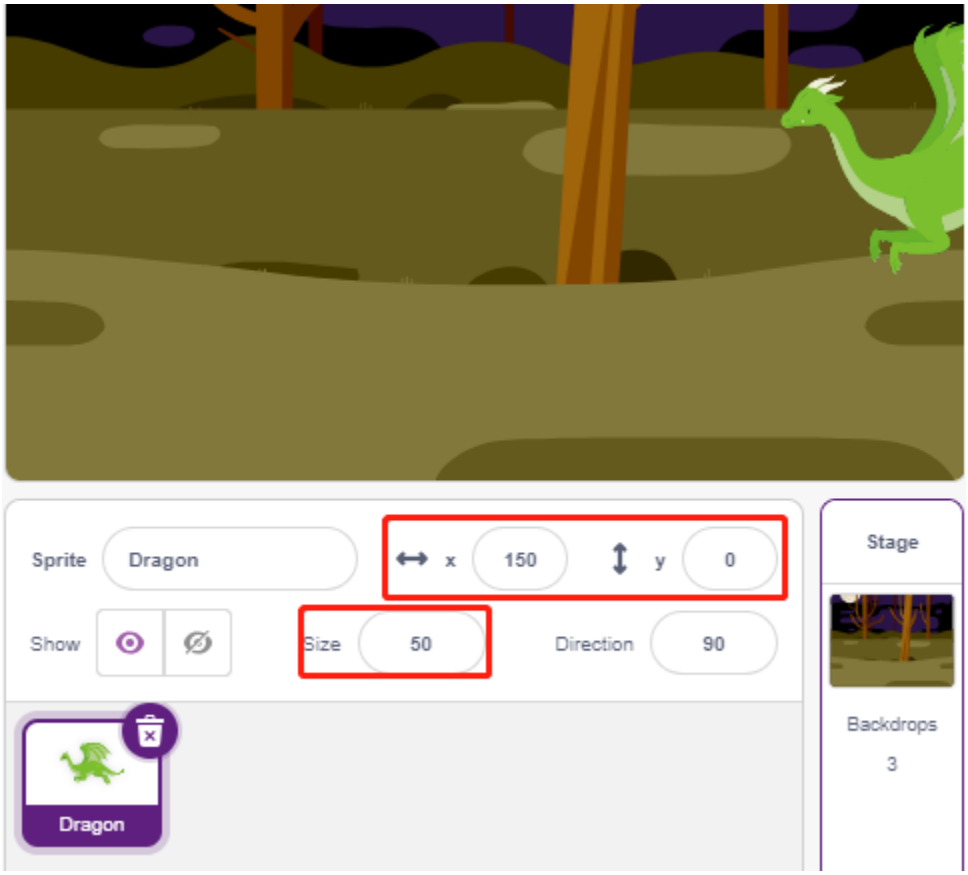
- Elimina el sprite predeterminado y añade el sprite **Dragón**.



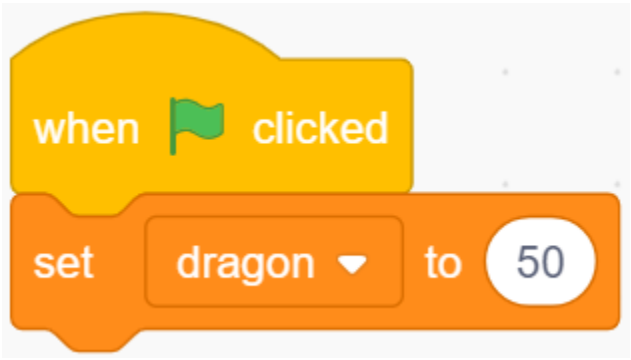
- Ve a la página de **Disfraces** y voltea horizontalmente los disfraces dragón-b y dragón-c.



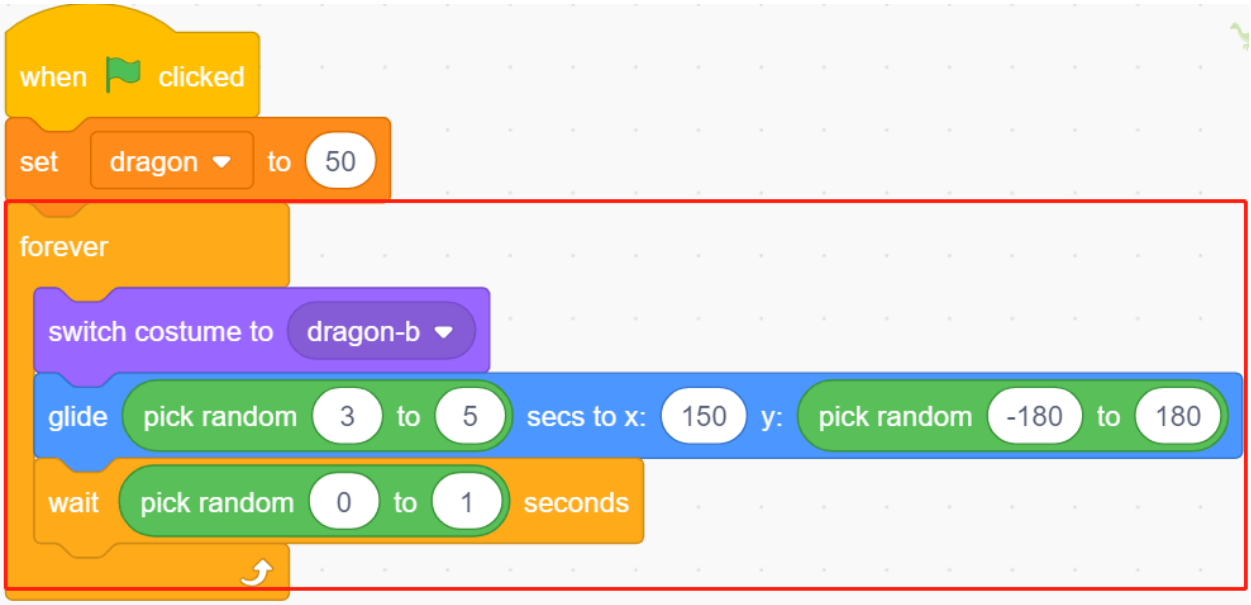
- Establece el tamaño al 50 %.



- Ahora crea una variable - **dragón** para registrar los puntos de vida del dragón, y establece el valor inicial a 50.

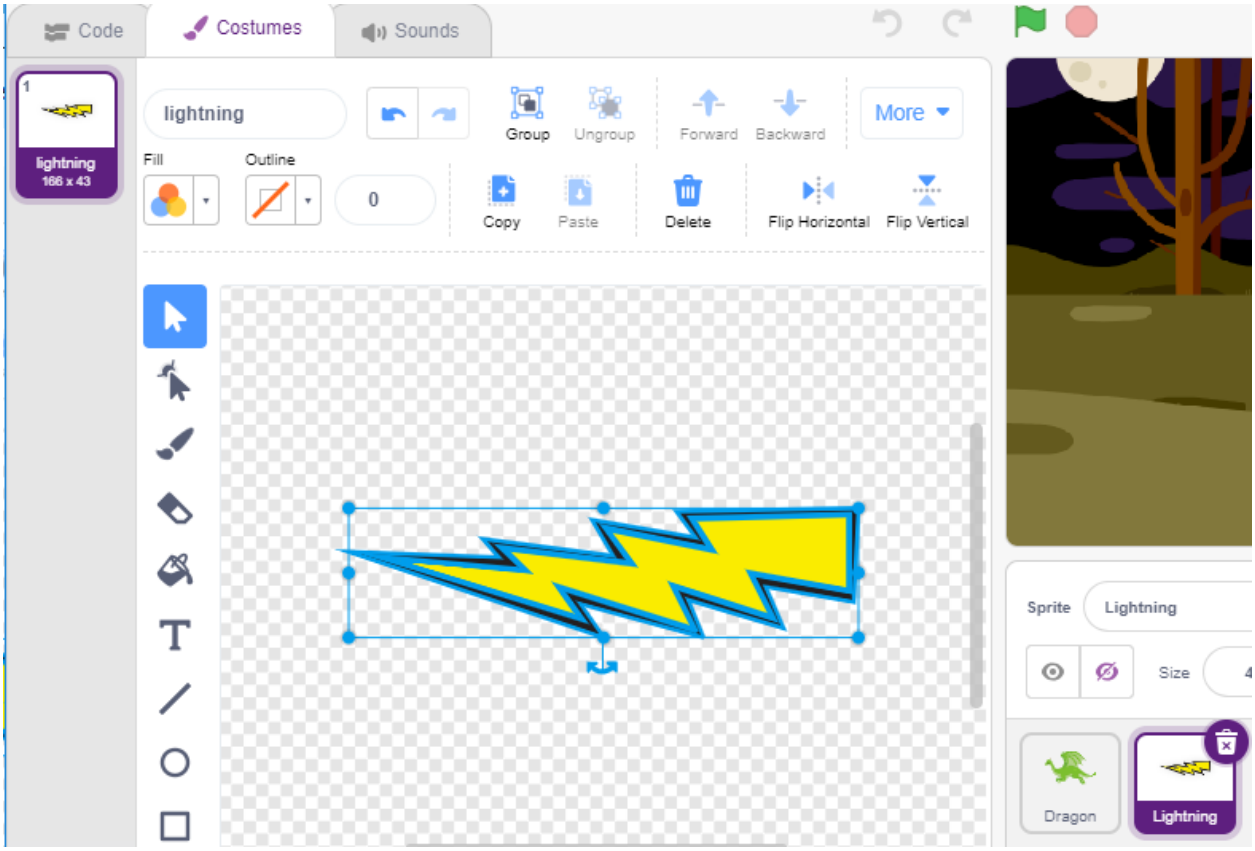


- A continuación, cambia el disfraz del sprite a **dragón-b** y haz que el sprite **Dragón** se mueva arriba y abajo en un rango.

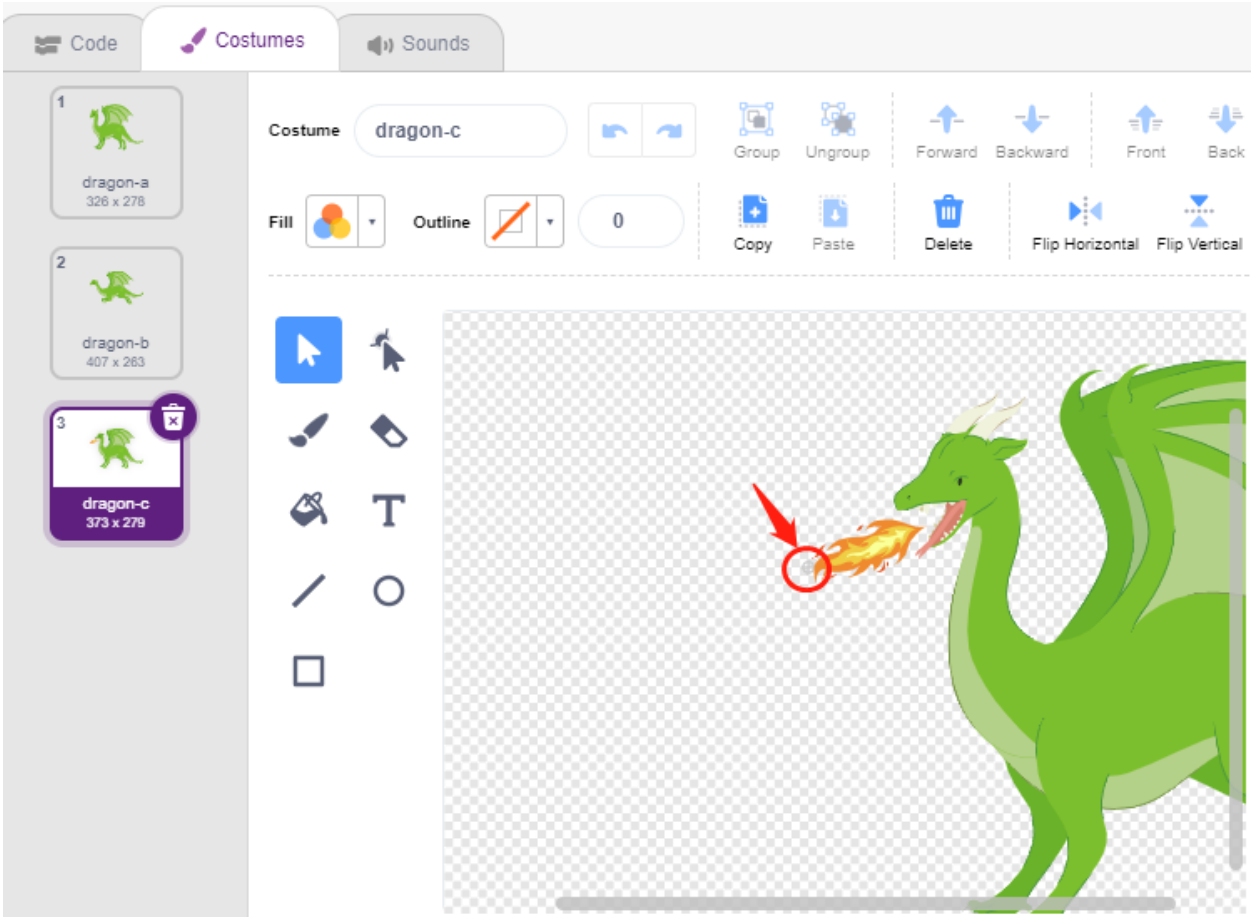


- Añade un sprite **Rayo** como el fuego lanzado por el sprite **Dragón**. Necesitas rotarlo 90° en el sentido de las agujas del reloj en la página de Disfraces, esto es para hacer que el sprite **Rayo** se mueva en la dirección correcta.

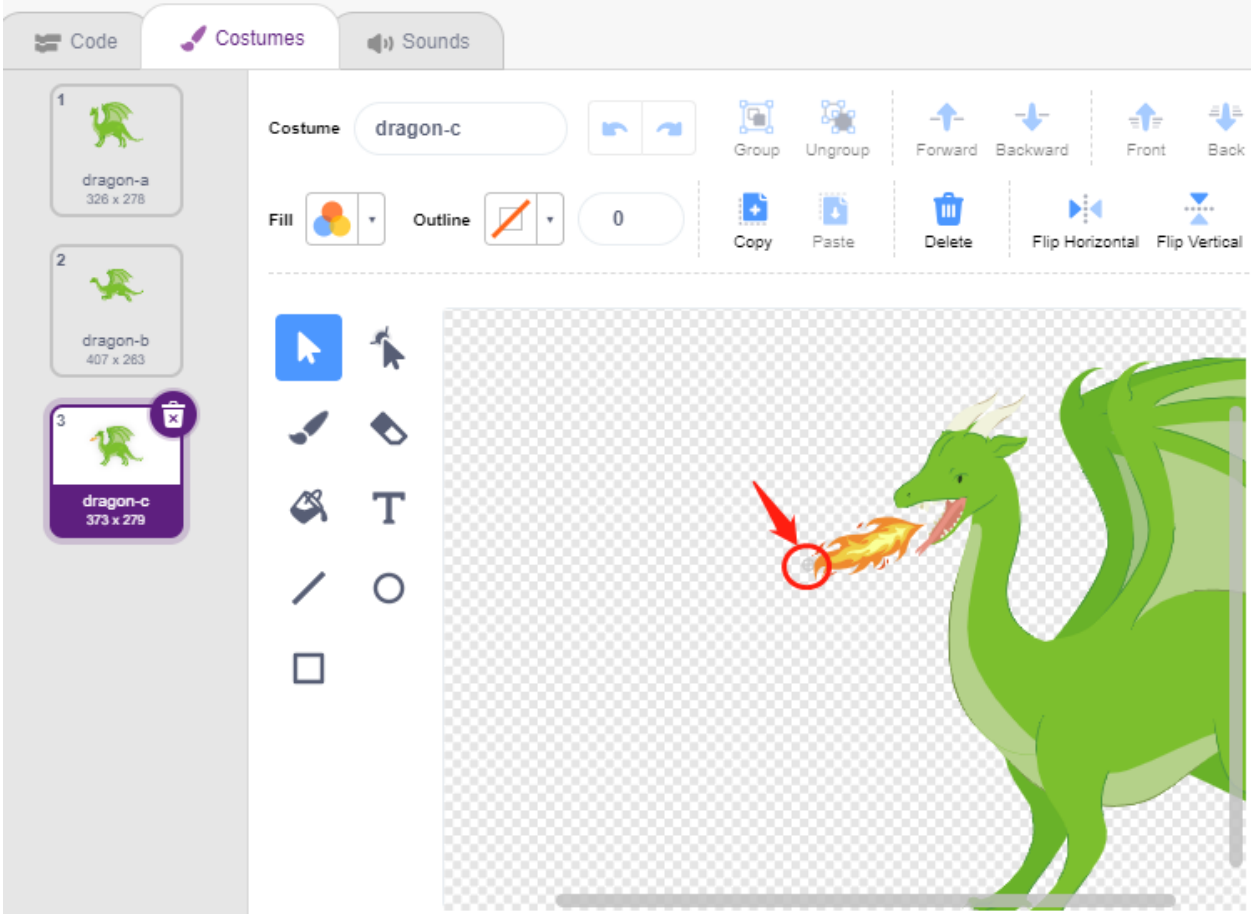
Nota: Al ajustar el disfraz del sprite **Rayo**, puedes moverlo fuera del centro, ¡esto debe evitarse! El punto central debe estar justo en el medio del sprite.



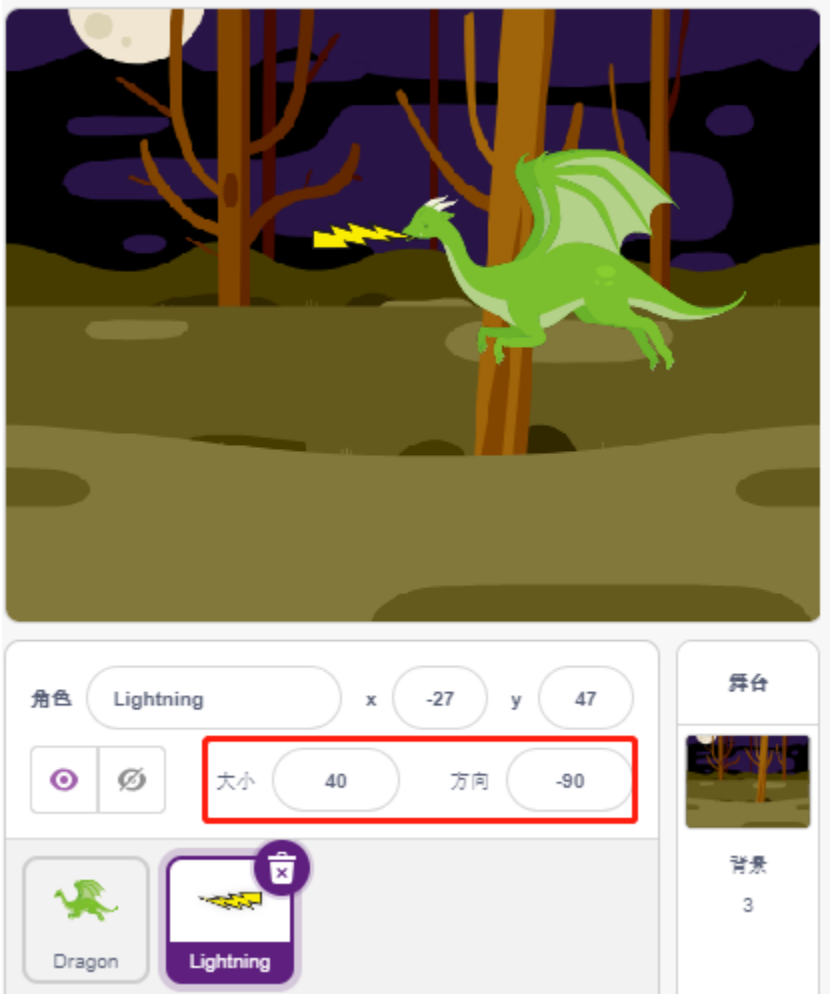
- Ajusta el disfraz de **dragon-c** del sprite **Dragón** para que su punto central esté en la cola del fuego. Esto hará que las posiciones del sprite **Dragón** y del sprite **Relámpago** sean correctas, evitando que el **Relámpago** se lance desde los pies del dragón.



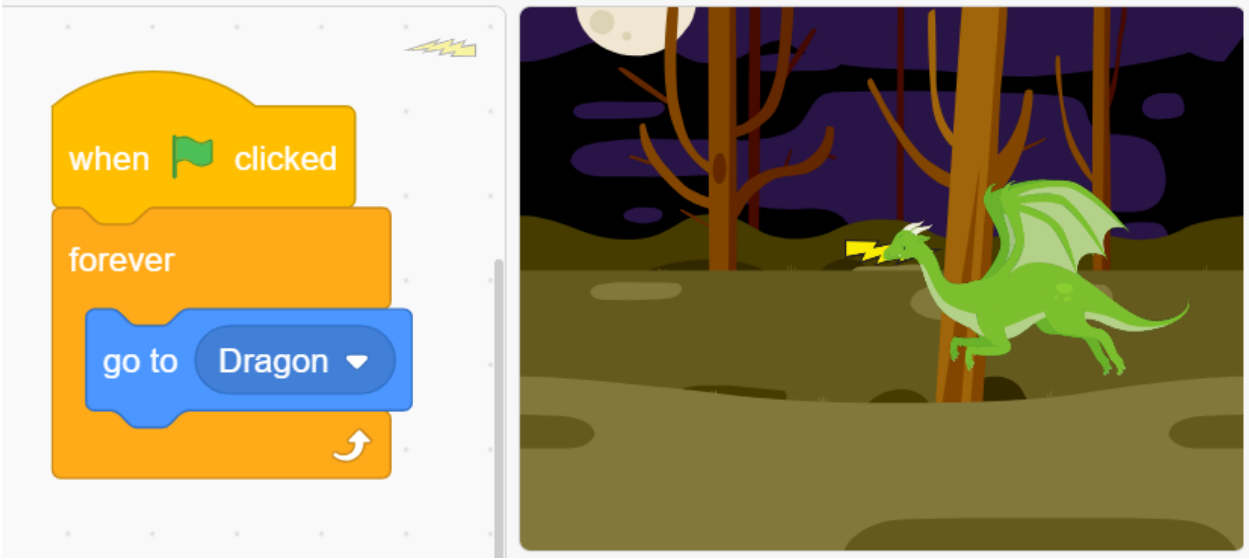
- Correspondientemente, **dragon-b** necesita hacer coincidir la cabeza del dragón con el punto central.



- Ajusta el tamaño y la orientación del sprite **Relámpago** para hacer que la imagen luzca más armoniosa.

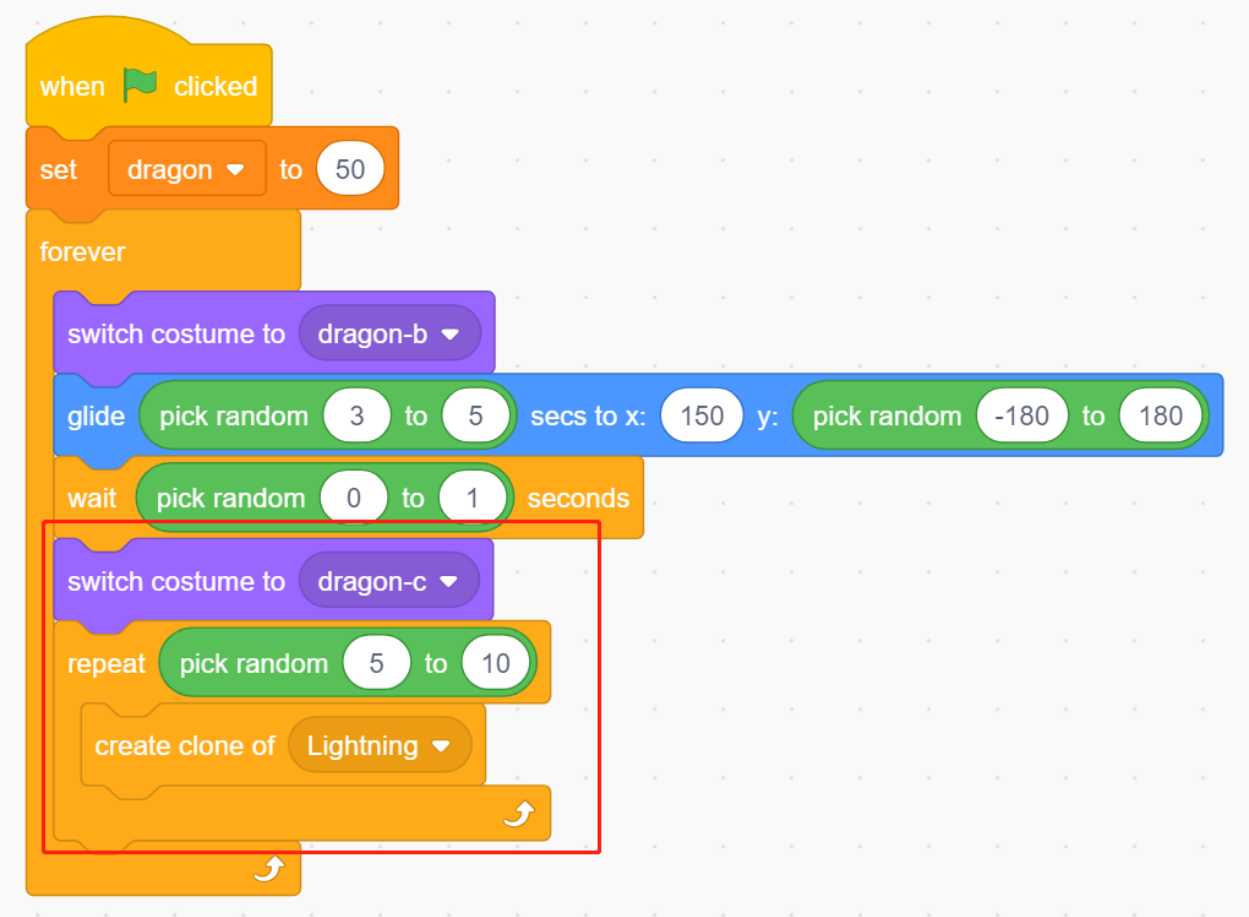


- Ahora programa el sprite **Relámpago**. Es fácil, solo haz que siga al sprite **Dragón** todo el tiempo. En este punto, haz clic en la bandera verde y verás al **Dragón** moviéndose con un relámpago en su boca.

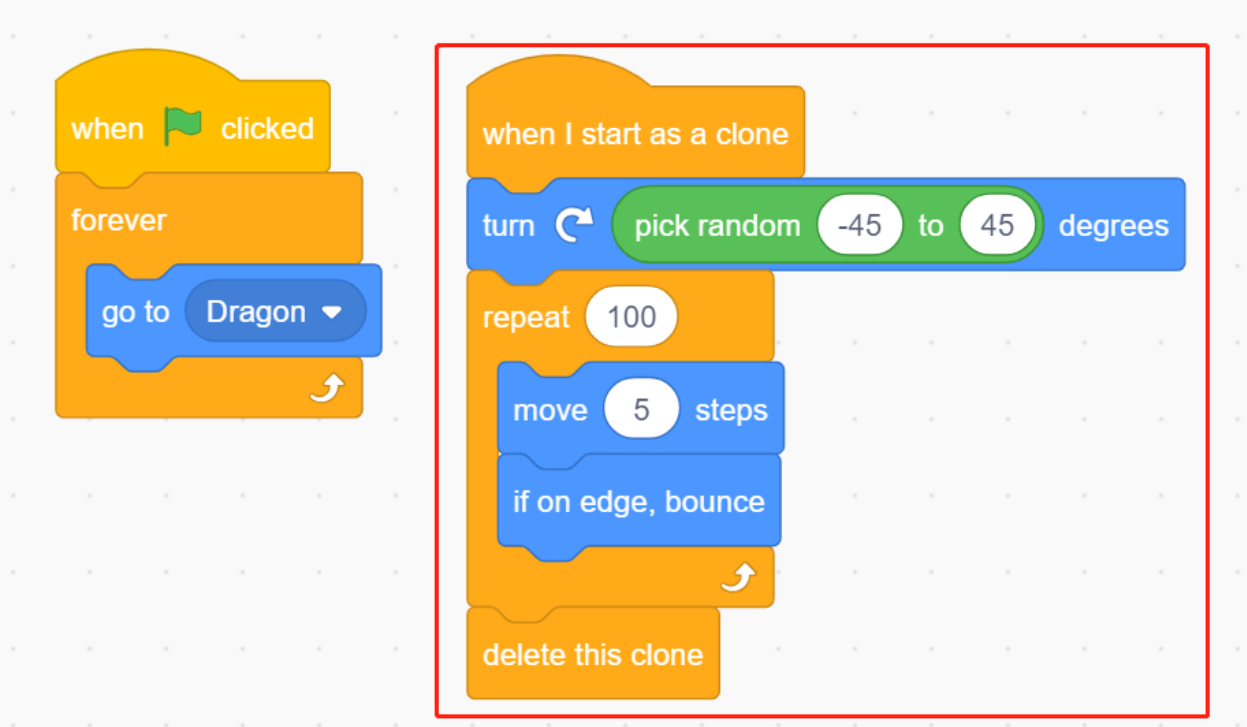


- Vuelve al sprite **Dragón**, ahora haz que sople fuego, teniendo cuidado de que el fuego en su boca no se dispare,

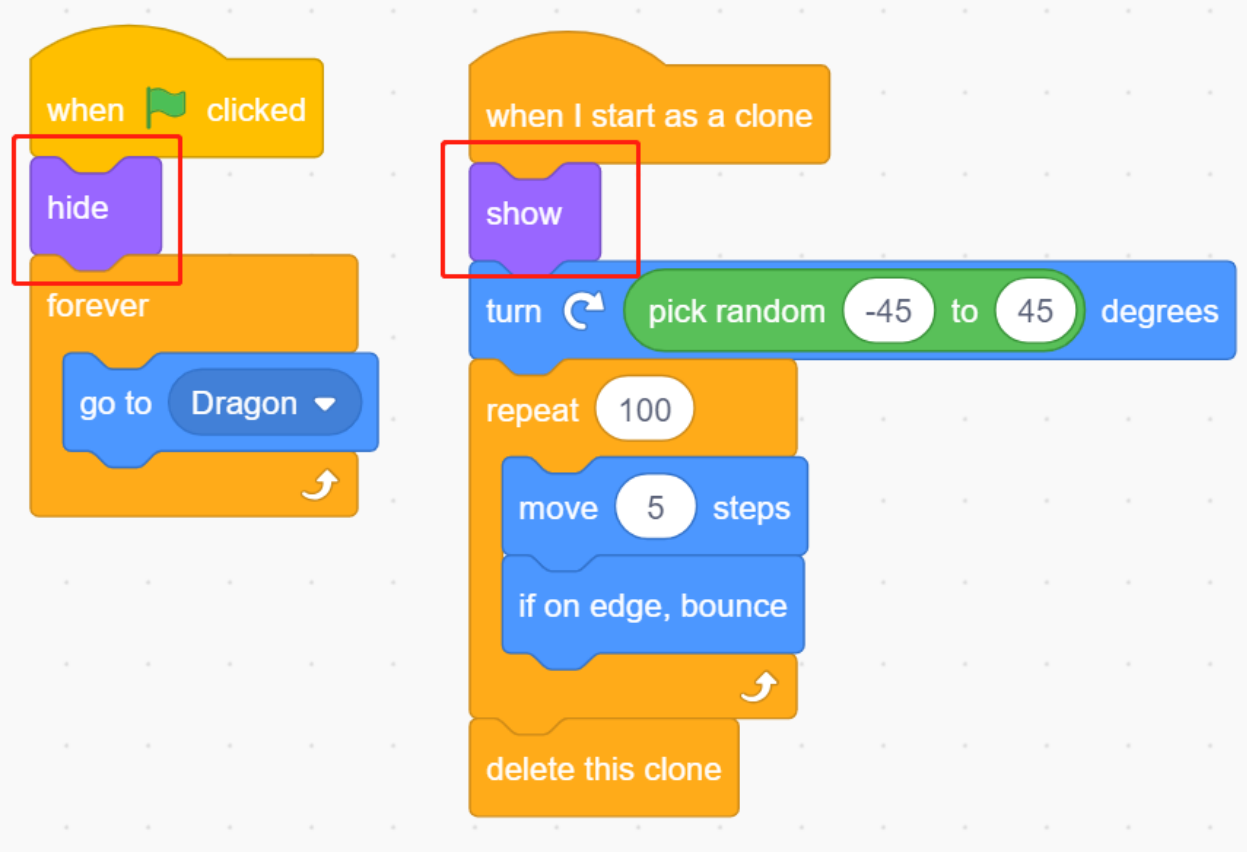
sino que cree un clon para el sprite **Relámpago**.



- Haz clic en el sprite **Relámpago** y permite que el clon de **Relámpago** se dispare en un ángulo aleatorio, rebotará en la pared y desaparecerá después de cierto tiempo.



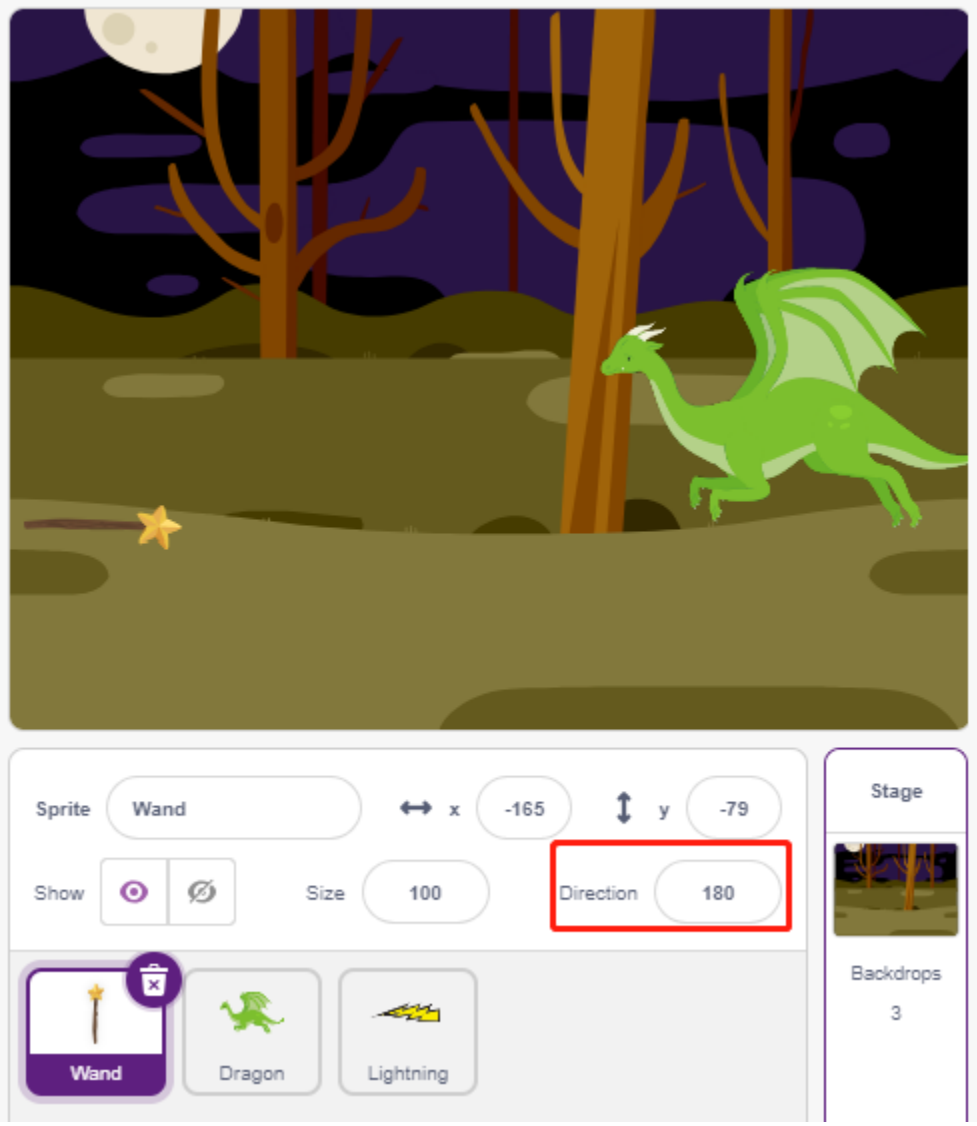
- En el sprite **Relámpago**, oculta su cuerpo y muestra el clon.



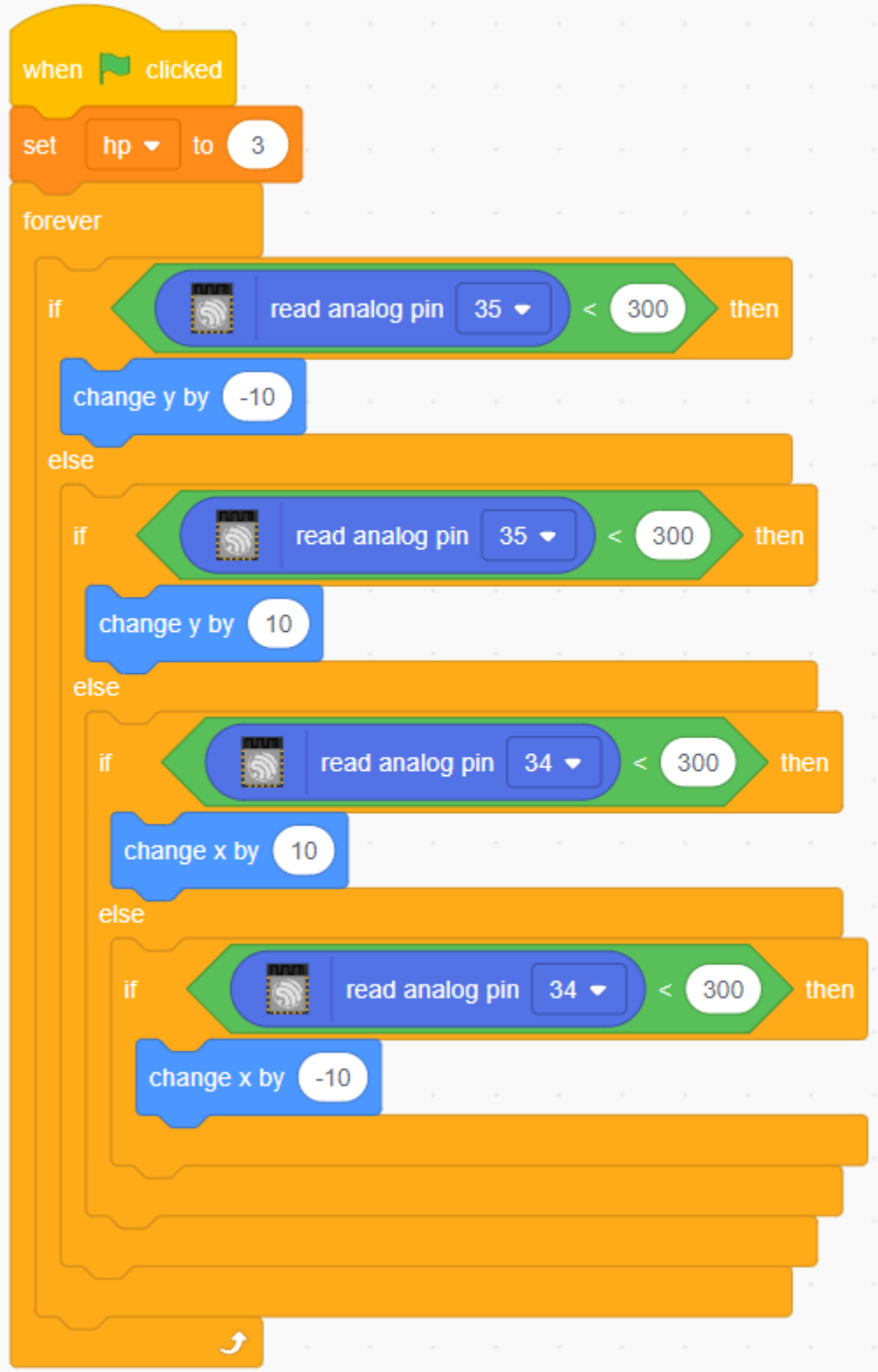
Ahora el dragón puede moverse hacia arriba y hacia abajo y soplar fuego.

2. Varita

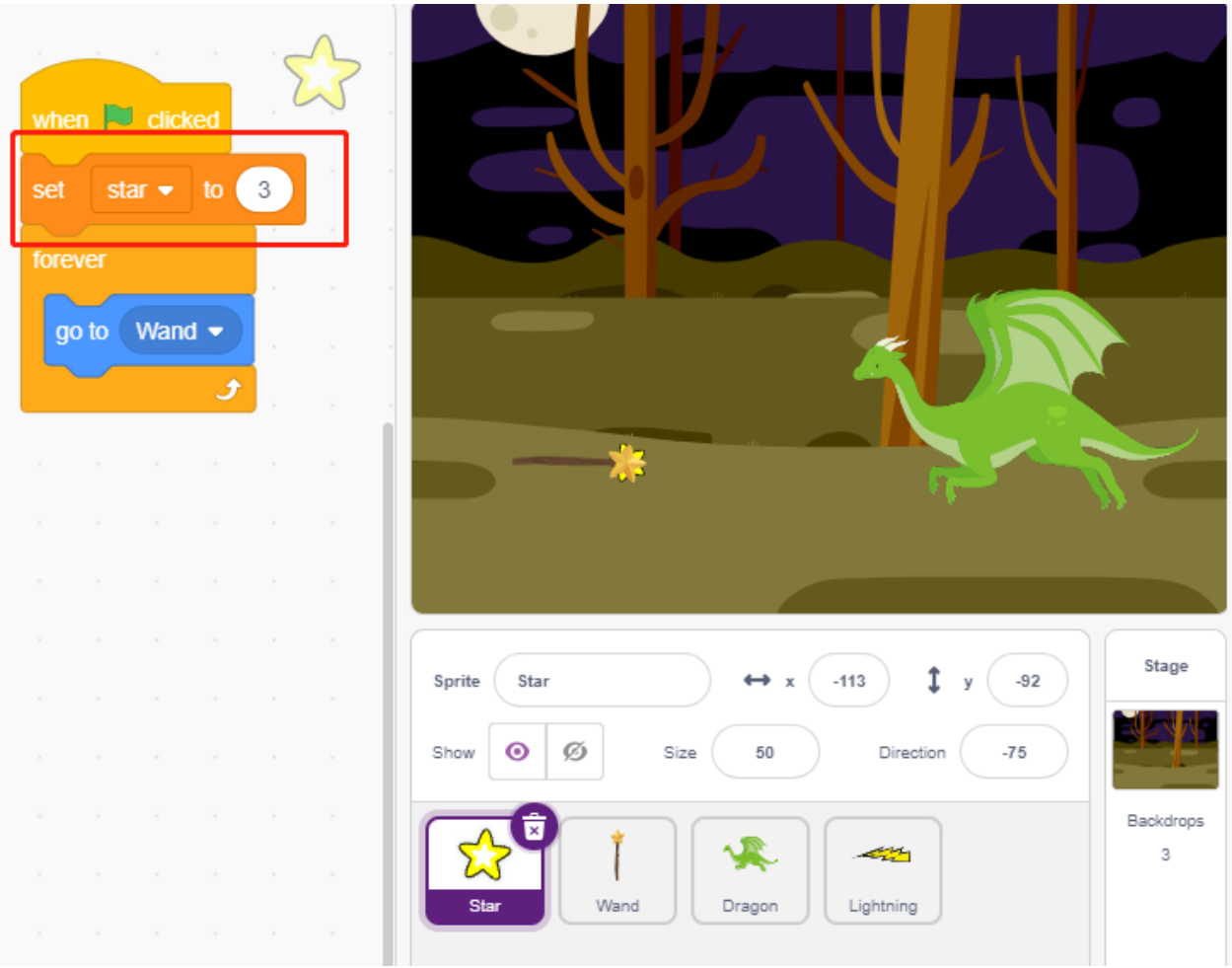
- Crea un sprite **Varita** y rota su dirección a 180 para que apunte hacia la derecha.



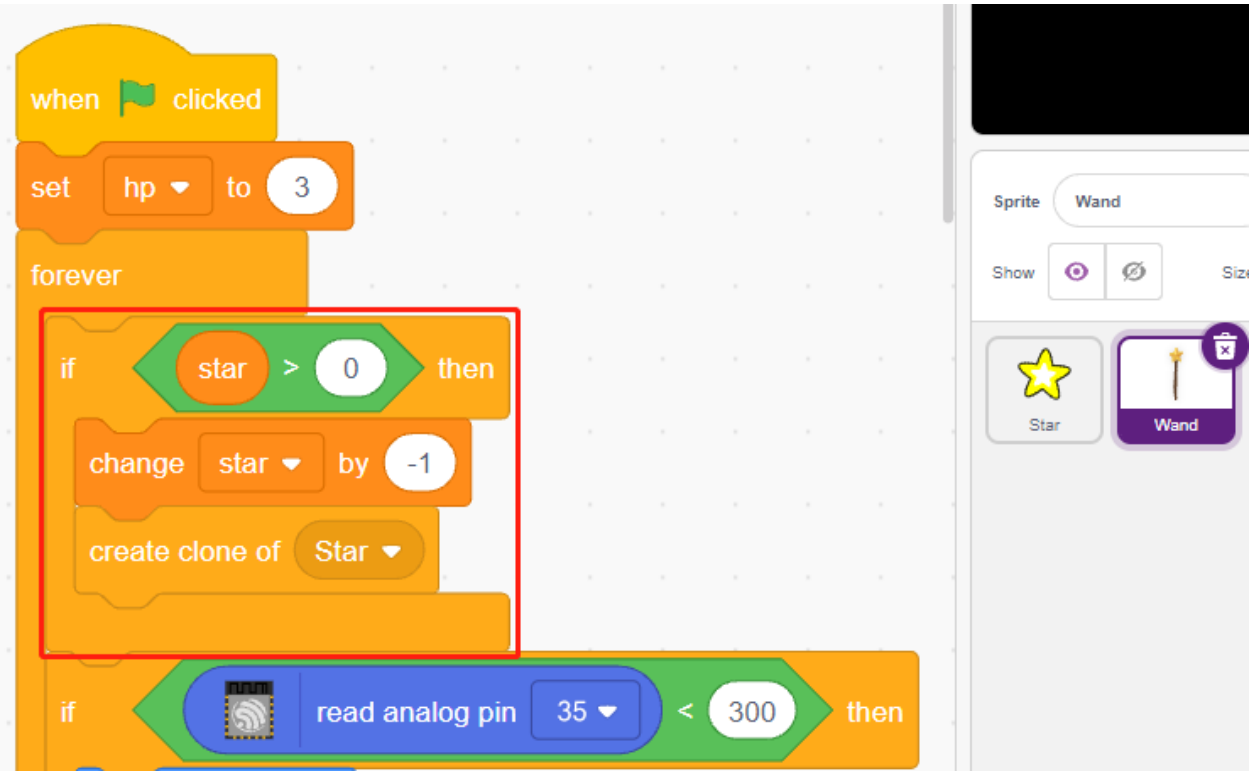
- Ahora crea una variable **hp** para registrar su valor de vida, inicialmente establecido en 3. Luego lee el valor del Joystick, que se utiliza para controlar el movimiento de la varita.



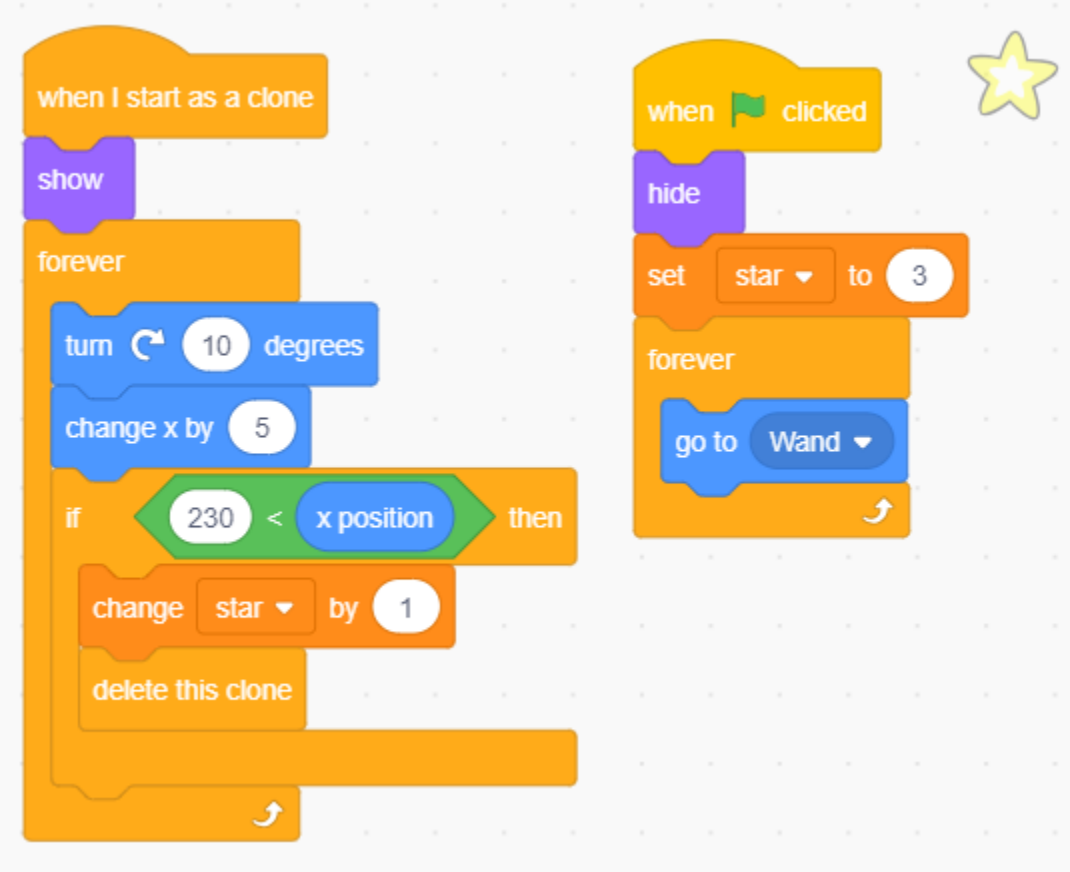
- El dragón tiene relámpagos, ¡y la varita que los destruye tiene su «bala mágica»! Crea un sprite **Estrella**, redimensiónalo y prográmalo para que siempre siga al sprite **Varita**, y limita el número de estrellas a tres.



- Haz que el sprite **Varita** dispare estrellas automáticamente. El sprite **Varita** dispara estrellas de la misma manera que el dragón sopla fuego – creando clones.



- Vuelve al sprite **Estrella** y programa su clon para que gire y dispare hacia la derecha, desaparezca después de ir más allá del escenario y restaurando el número de estrellas. Igual que con el sprite **Relámpago**, oculta el cuerpo y muestra el clon.



Ahora tenemos una varita que dispara balas de estrella.

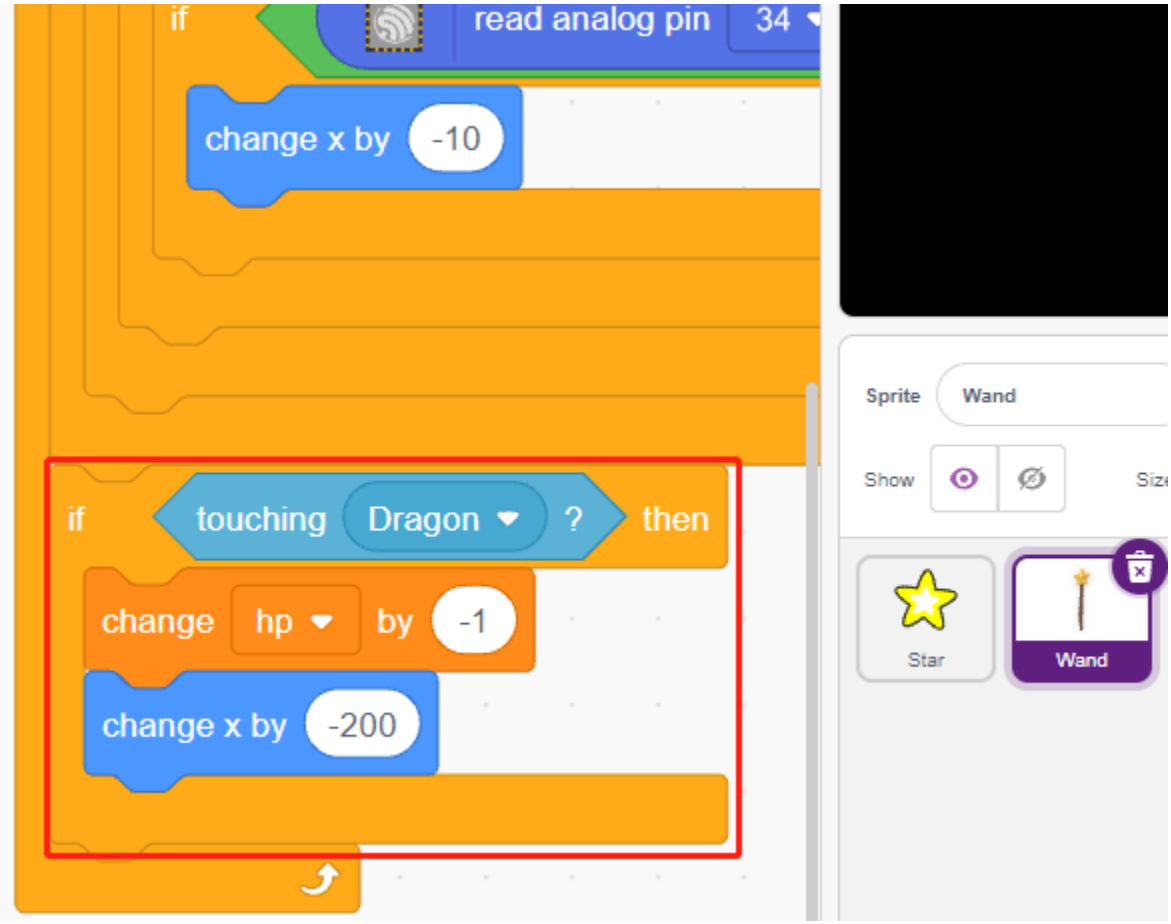
3. ¡Lucha!

La varita y el dragón están actualmente en desacuerdo, y vamos a hacer que luchen. El dragón es fuerte, y la varita es el valiente que se enfrenta al dragón. La interacción entre ellos consiste en las siguientes partes.

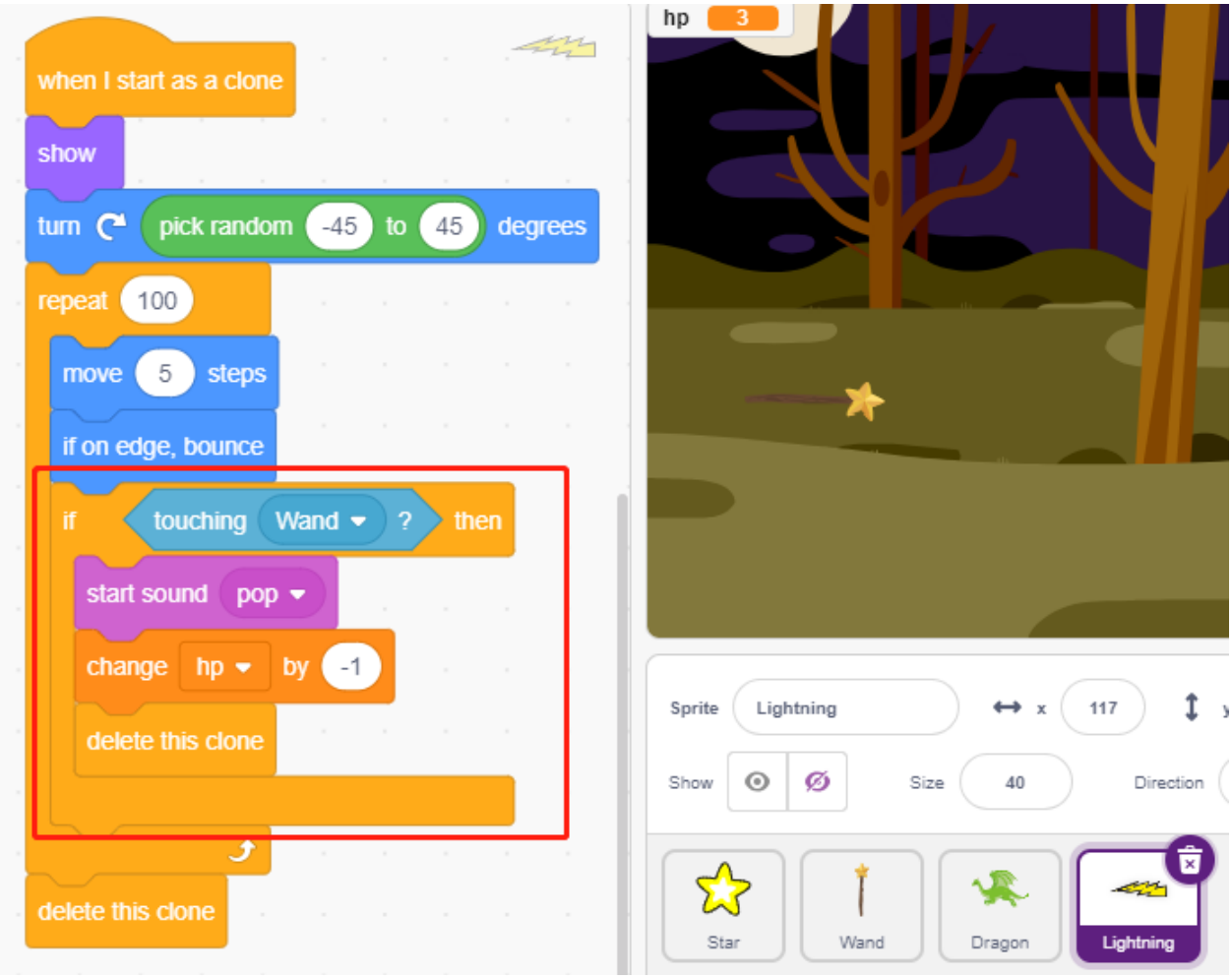
1. si la varita toca al dragón, será repelida hacia atrás y perderá puntos de vida.
2. si un relámpago golpea la varita, la varita perderá puntos de vida.
3. si la bala de estrella golpea al dragón, el dragón perderá puntos de vida.

Una vez que esto esté resuelto, pasemos a cambiar los guiones para cada sprite.

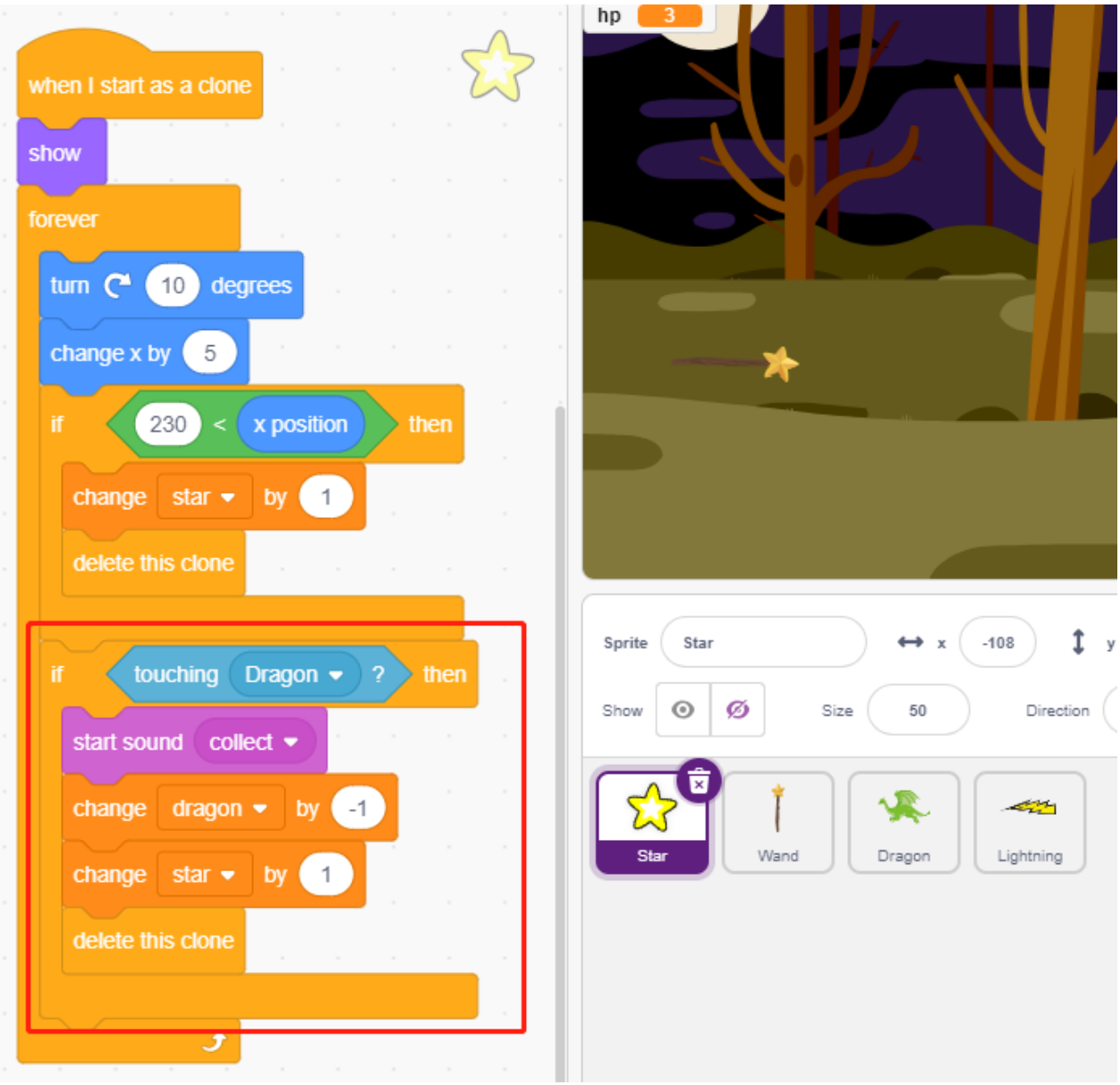
- Si la **Varita** golpea al **Dragón**, será repelida hacia atrás y perderá puntos de vida.



- Si **Relámpago** (un clon del sprite **Relámpago**) golpea al sprite **Varita**, hará un sonido de estallido y desaparecerá, y la **Varita** perderá puntos de vida.



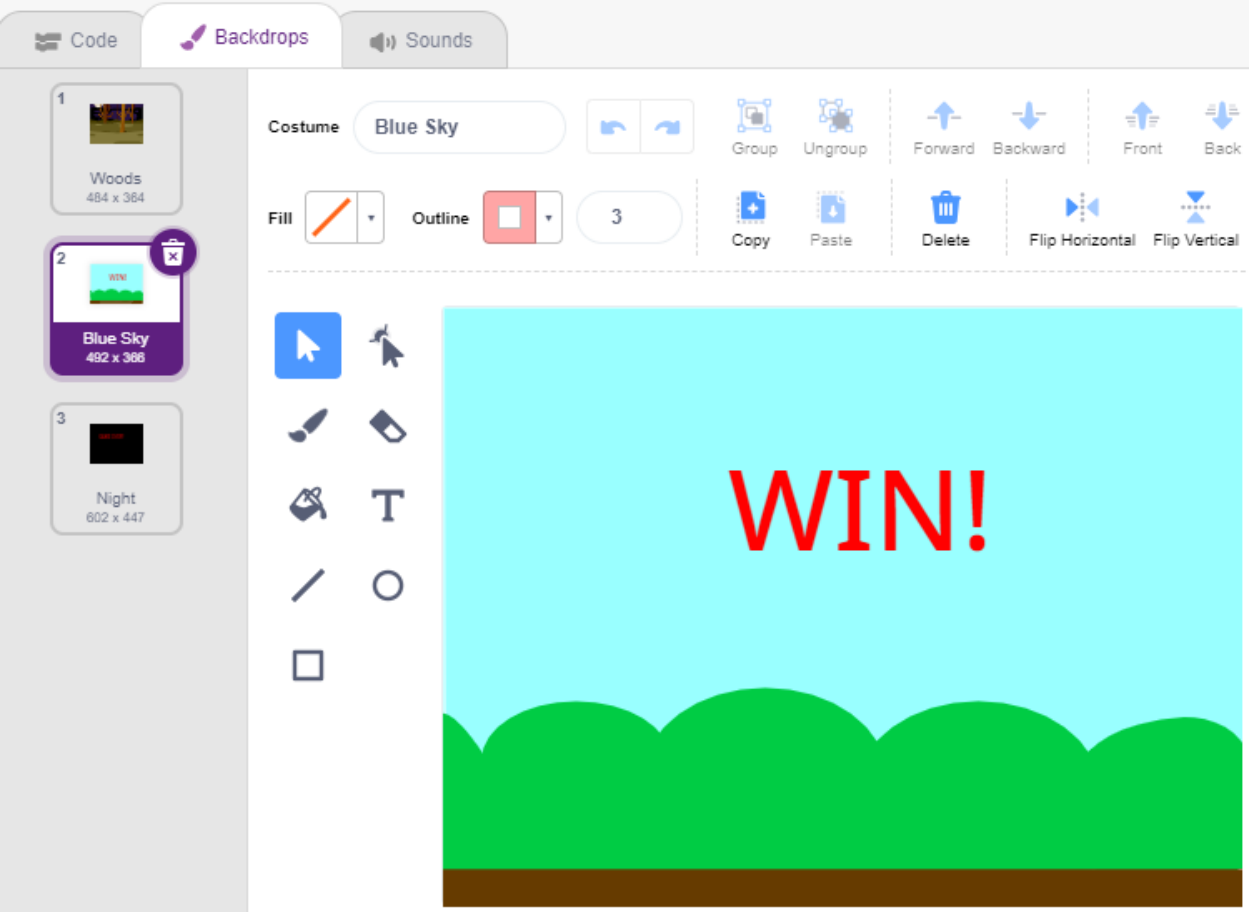
- Si una **Estrella** (clon del sprite **Estrella**) golpea al **Dragón**, emitirá un sonido de recolección y desaparecerá, mientras restaura el conteo de **Estrellas**, y el **Dragón** perderá puntos de vida.



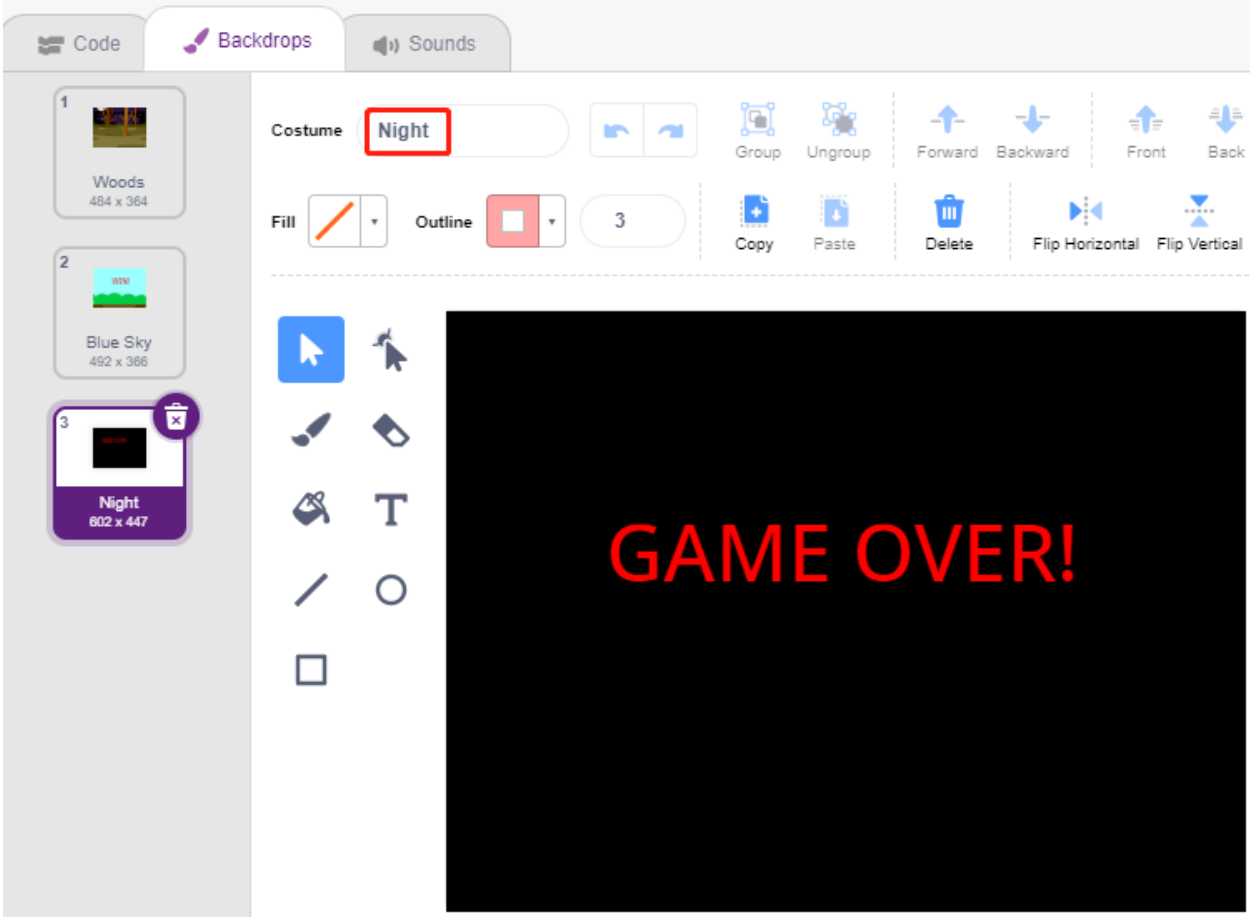
4. Escenario

La batalla entre la **Varita** y el **Dragón** eventualmente se dividirá en ganadores y perdedores, lo que representamos con el escenario.

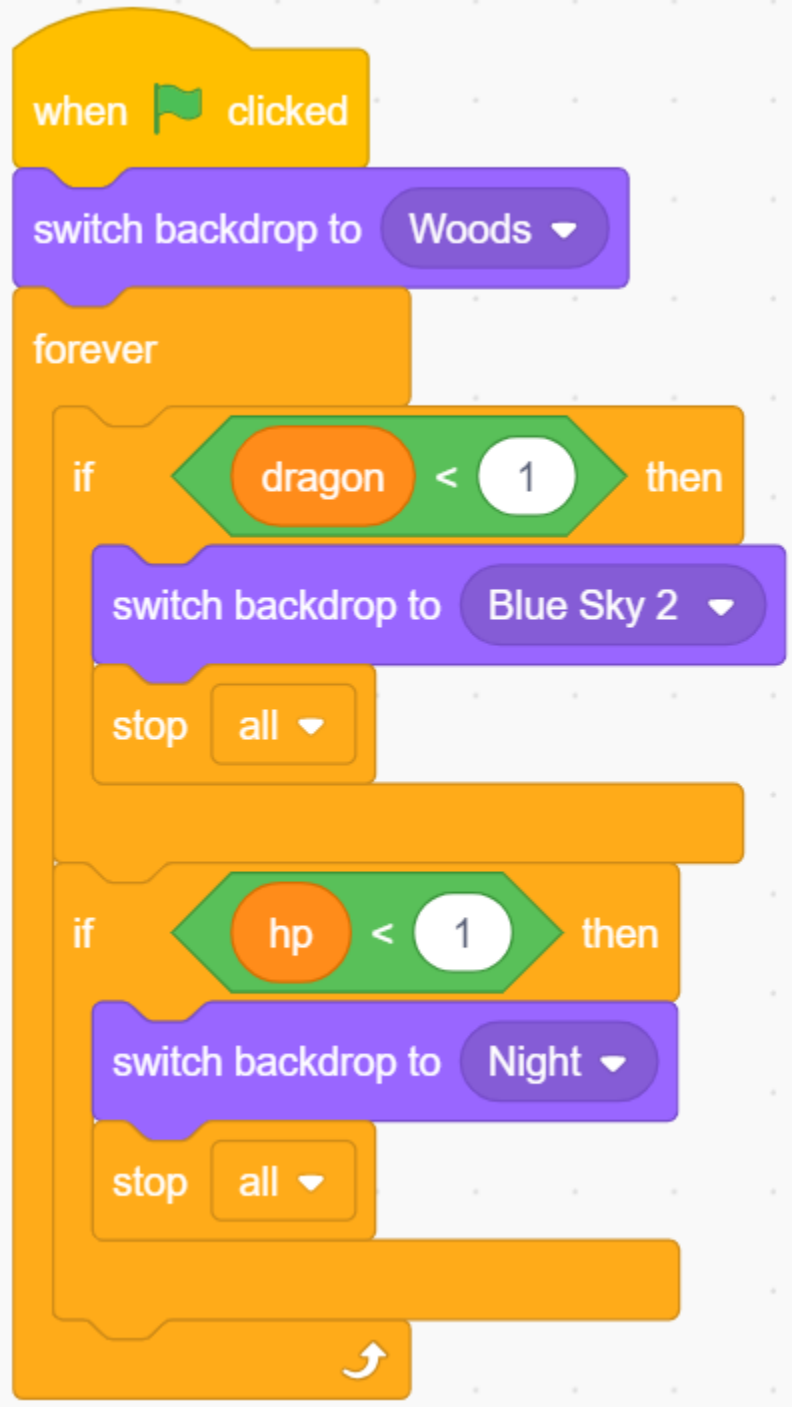
- Añade el fondo **Cielo Azul**, y escribe el personaje «¡GANASTE!» en él para representar que el dragón ha sido derrotado y ha llegado el amanecer.



- Y modifica el fondo en blanco de la siguiente manera, para representar que el juego ha fallado y todo estará en oscuridad.



- Ahora escribe un guion para cambiar estos fondos, cuando se haga clic en la bandera verde, cambia al fondo **Bosque**; si el punto de vida del dragón es menos de 1, entonces el juego tiene éxito y cambia el fondo a **Cielo Azul**; si el valor de vida de la **Varita** es menos de 1, entonces cambia al fondo **Noche** y el juego falla.


when clicked
switch backdrop to Woods
forever
if dragon < 1 then
switch backdrop to Blue Sky 2
stop all
if hp < 1 then
switch backdrop to Night
stop all

CAPÍTULO 6

Aprende sobre los Componentes en tu Kit

Después de abrir el paquete, por favor verifica si la cantidad de componentes cumple con la descripción del producto y si todos los componentes están en buenas condiciones.

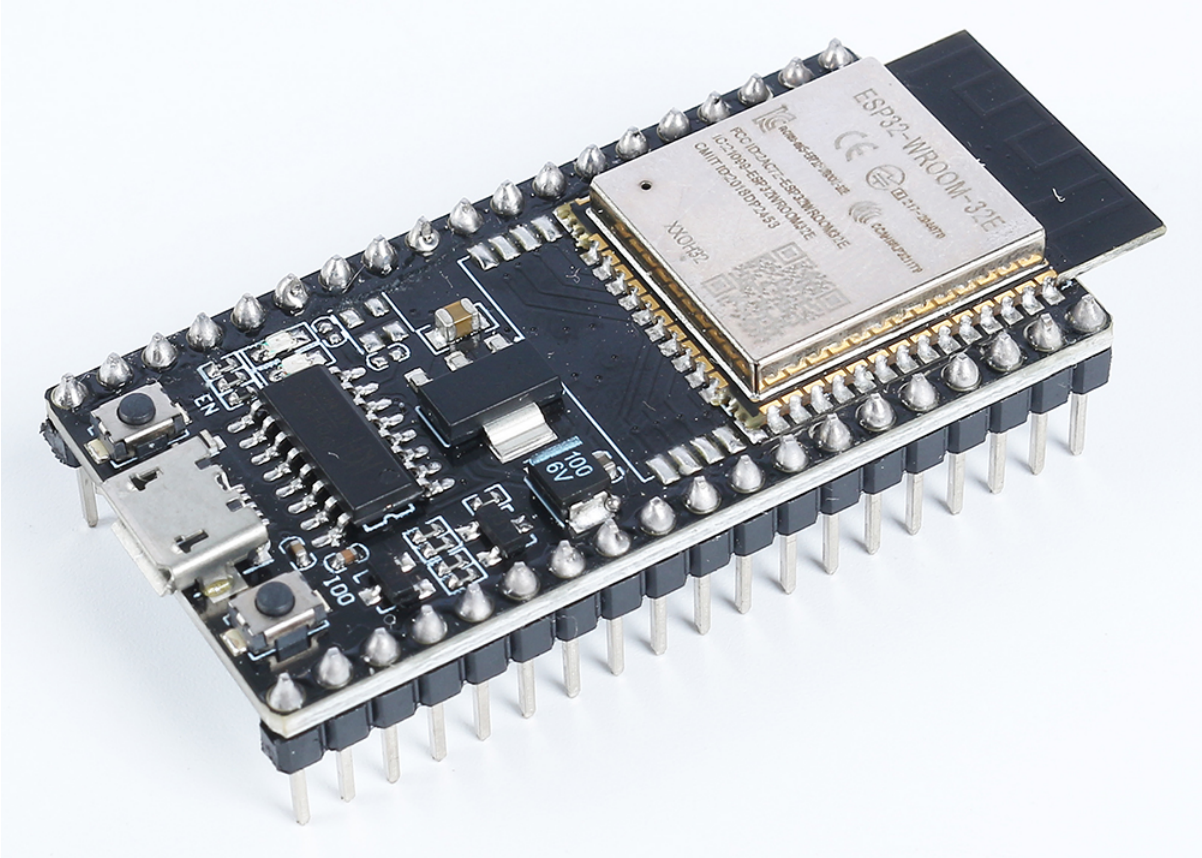
Packing List

A continuación se presenta la introducción de cada componente, que contiene el principio de funcionamiento del componente y los proyectos correspondientes.

Placa de Control

6.1 ESP32 WROOM 32E

El ESP32 WROOM-32E es un módulo versátil y potente construido en torno al chipset ESP32 de Espressif. Ofrece procesamiento de doble núcleo, conectividad integrada Wi-Fi y Bluetooth, y cuenta con una amplia gama de interfaces periféricas. Conocido por su bajo consumo de energía, el módulo es ideal para aplicaciones de IoT, lo que permite una conectividad inteligente y un rendimiento robusto en formatos compactos.



Entre sus características clave se encuentran:

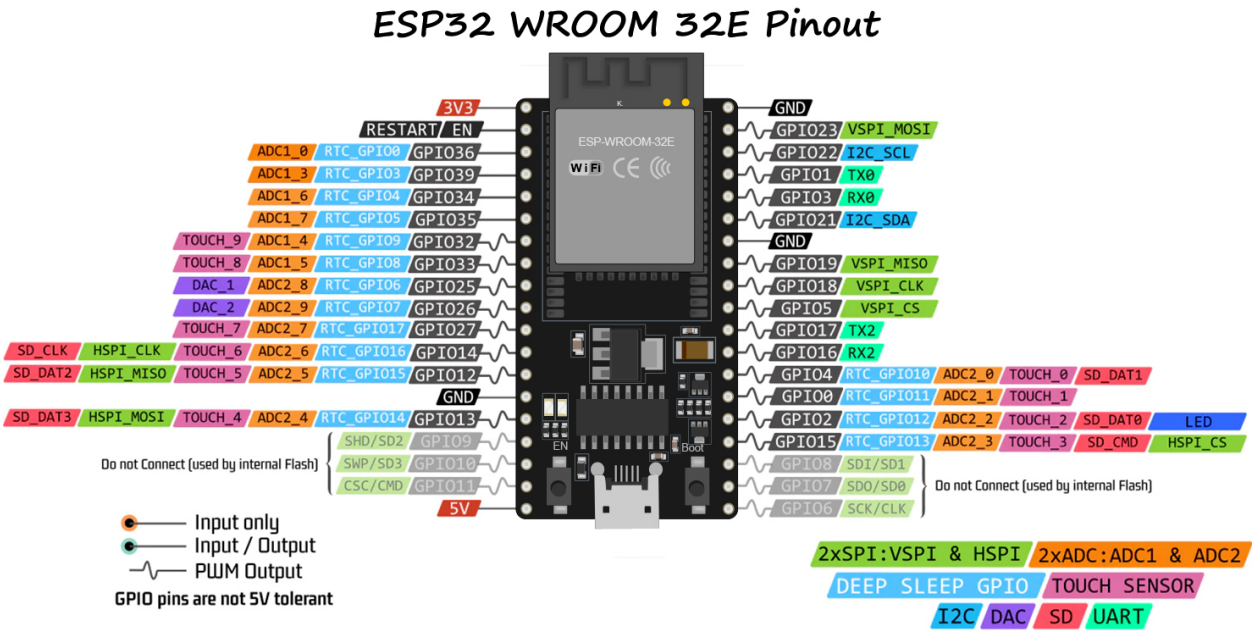
- **Potencia de Procesamiento**: Equipado con un microprocesador de doble núcleo Xtensa® 32-bit LX6, ofrece escalabilidad y flexibilidad.
- **Capacidades Inalámbricas**: Con Wi-Fi integrado de 2.4 GHz y Bluetooth de doble modo, es perfecto para aplicaciones que demandan una comunicación inalámbrica estable.
- **Memoria y Almacenamiento**: Viene con amplio SRAM y almacenamiento flash de alto rendimiento, satisfaciendo las necesidades de programas de usuario y almacenamiento de datos.
- **GPIO**: Ofrece hasta 38 pines GPIO, soportando una variedad de dispositivos y sensores externos.
- **Bajo Consumo de Energía**: Dispone de varios modos de ahorro de energía, haciéndolo ideal para escenarios con batería o eficientes en energía.
- **Seguridad**: Cuenta con encriptación integrada y características de seguridad para asegurar que los datos del usuario y la privacidad estén bien protegidos.
- **Versatilidad**: Desde electrodomésticos simples hasta maquinaria industrial compleja, el WROOM-32E ofrece un rendimiento consistente y eficiente.

En resumen, el ESP32 WROOM-32E no solo ofrece sólidas capacidades de procesamiento y diversas opciones de conectividad, sino que también cuenta con una variedad de características que lo convierten en la opción preferida en los sectores de IoT y dispositivos inteligentes.

-

6.1.1 Diagrama de Pinout

El ESP32 tiene algunas limitaciones de uso de pines debido a que varias funcionalidades comparten ciertos pines. Al diseñar un proyecto, es buena práctica planificar cuidadosamente el uso de pines y verificar posibles conflictos para garantizar un funcionamiento adecuado y evitar problemas.



Aquí hay algunas de las restricciones y consideraciones clave:

- **ADC1 y ADC2**: ADC2 no se puede utilizar cuando el WiFi o el Bluetooth están activos. Sin embargo, ADC1 se puede utilizar sin restricciones.
- **Pines de Bootstrap**: GPIO0, GPIO2, GPIO5, GPIO12 y GPIO15 se utilizan para el arranque durante el proceso de inicio. Se debe tener cuidado de no conectar componentes externos que puedan interferir con el proceso de arranque en estos pines.
- **Pines JTAG**: GPIO12, GPIO13, GPIO14 y GPIO15 se pueden utilizar como pines JTAG para propósitos de depuración. Si la depuración JTAG no es necesaria, estos pines se pueden utilizar como GPIO regulares.
- **Pines de Táctiles**: Algunos pines admiten funcionalidades táctiles. Estos pines deben usarse con cuidado si se planea utilizarlos para sensibilidad táctil.
- **Pines de Alimentación**: Algunos pines están reservados para funciones relacionadas con la alimentación y deben usarse en consecuencia. Por ejemplo, evite extraer corriente excesiva de pines de alimentación como 3V3 y GND.
- **Pines de Solo Entrada**: Algunos pines son solo de entrada y no deben usarse como salidas.

6.1.2 Pines de Estrapeo

ESP32 tiene cinco pines de estrapeo:

Pines de Estrapeo	Descripción
IO5	Por defecto en pull-up, el nivel de voltaje de IO5 y IO15 afecta el Tiempo de SDIO Slave.
IO0	Por defecto en pull-up, si se baja, entra en modo de descarga.
IO2	Por defecto en pull-down, IO0 e IO2 harán que ESP32 entre en modo de descarga.
IO12(MTDI)	Por defecto en pull-down, si se sube, ESP32 no se iniciará correctamente.
IO15(MTDO)	Por defecto en pull-up, si se baja, el registro de depuración no será visible. Además, el nivel de voltaje de IO5 e IO15 afecta el Tiempo de SDIO Slave.

El software puede leer los valores de estos cinco bits del registro «GPIO_STRAPPING». Durante la liberación del reinicio del sistema del chip (reinicio por encendido, reinicio por vigilante RTC y reinicio por caída de tensión), los latches de los pines de estrapeo muestrean el nivel de voltaje como bits de estrapeo de «0» o «1», y mantienen estos bits hasta que el chip se apague o se reinicie. Los bits de estrapeo configuran el modo de arranque del dispositivo, el voltaje de operación de VDD_SDIO y otras configuraciones iniciales del sistema.

Cada pin de estrapeo está conectado a su resistencia interna de pull-up/pull-down durante el reinicio del chip. En consecuencia, si un pin de estrapeo no está conectado o el circuito externo conectado tiene una alta impedancia, la resistencia interna débil de pull-up/pull-down determinará el nivel de entrada predeterminado de los pines de estrapeo.

Para cambiar los valores de bits de estrapeo, los usuarios pueden aplicar resistencias externas de pull-down/pull-up, o utilizar los GPIOs de la MCU host para controlar el nivel de voltaje de estos pines al encender ESP32.

Después de la liberación del reinicio, los pines de estrapeo funcionan como pines de función normal. Consulte la siguiente tabla para obtener una configuración detallada del modo de arranque por pines de estrapeo.

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0

Timing of SDIO Slave					
Pin	Default	FE Sampling FE Output	FE Sampling RE Output	RE Sampling FE Output	RE Sampling RE Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

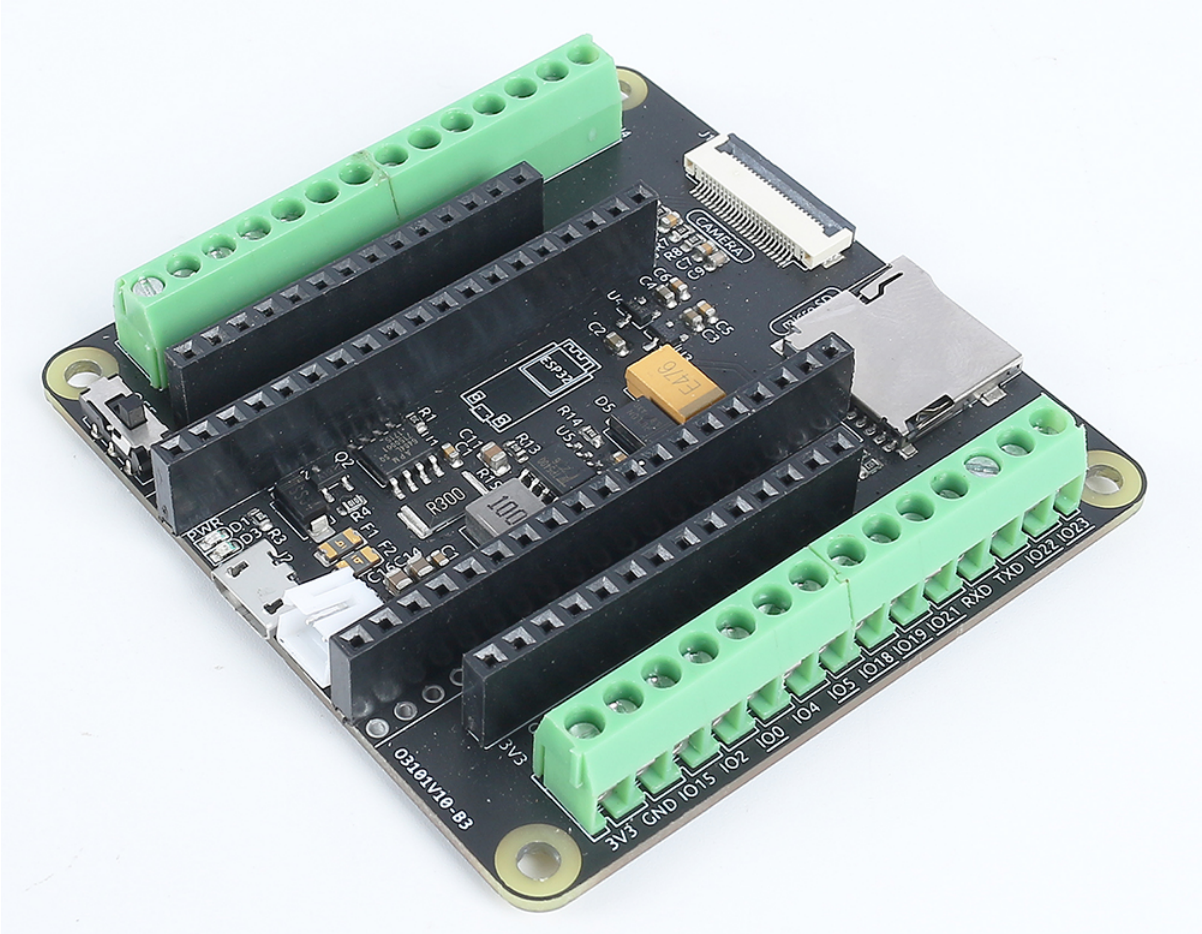
- FE: flanco de bajada, RE: flanco de subida
- El firmware puede configurar los bits del registro para cambiar la configuración de «Voltaje del LDO Interno (VDD_SDIO)» y «Tiempo del Esclavo SDIO», después del arranque.
- El módulo integra una memoria flash SPI de 3.3 V, por lo que el pin MTDI no puede configurarse en 1 cuando el módulo está encendido.

6.2 Extensión de Cámara ESP32

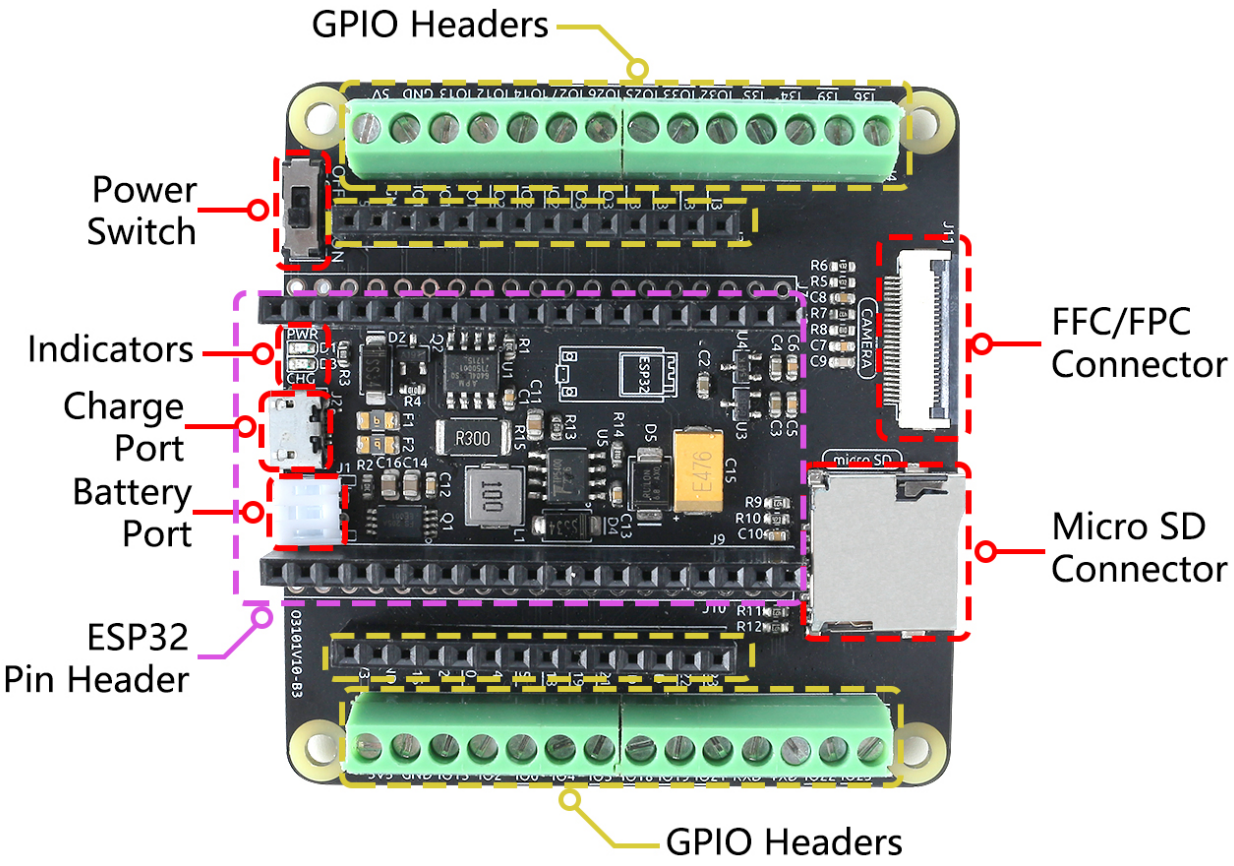
Hemos diseñado una placa de expansión que le permite aprovechar al máximo las funcionalidades de la cámara y la tarjeta SD del ESP32 WROOM 32E. Al combinar la cámara OV2640, Micro SD y ESP32 WROOM 32E, obtiene una placa de expansión todo en uno.

La placa proporciona dos tipos de cabeceras GPIO: una con cabeceras hembra, perfecta para proyectos de prototipado rápido. El otro tipo cuenta con terminales de tornillo, asegurando conexiones de cable estables y haciéndolo adecuado para proyectos de IoT.

Además, puede alimentar su proyecto con una sola batería de 3.7V 18650. Si la batería se agota, puede cargarla convenientemente simplemente conectando un cable Micro USB de 5V. Esto lo convierte en una excelente herramienta para proyectos al aire libre y aplicaciones remotas.



6.2.1 Introducción de la Interfaz



- **Interruptor de Encendido**
  - Controla el suministro de energía de la batería, alternándolo entre encendido y apagado.
- **Puerto de Carga**
  - Al conectar un cable Micro USB de 5V, la batería puede cargarse.
- **Puerto de Batería**
  - Cuenta con una interfaz PH2.0-2P, compatible con baterías de litio 18650 de 3.7V.
  - Proporciona energía tanto al ESP32 WROOM 32E como a la Extensión de Cámara ESP32.
- **Cabeceras de Pines ESP32**
  - Destinadas para el módulo ESP32 WROOM 32E. Preste atención a su orientación; asegúrese de que ambos puertos Micro USB estén orientados hacia el mismo lado para evitar una colocación incorrecta.
- **Cabeceras GPIO**
  - **Hembras**: Utilizadas para conectar varios componentes al ESP32, perfectas para proyectos de prototipado rápido.
  - **Terminal de Tornillo**: Terminal de tornillo de 14 pines con paso de 3.5mm, asegurando conexiones de cable estables y haciéndolo adecuado para proyectos de IoT.
- **Luces Indicadoras**

  - **PWR**: Se enciende cuando la batería está alimentada o cuando se conecta un Micro USB directamente al ESP32.
  - **CHG**: Se ilumina al conectar un Micro USB al puerto de carga de la placa, indicando el inicio de la carga. Se apagará una vez que la batería esté completamente cargada.

- **Conector Micro SD**
  - Ranura de resorte para la inserción y eyección fácil de la tarjeta Micro SD.
- **Conector FFC / FPC de 24 pines de 0.5mm**
  - Diseñado para la cámara OV2640, haciéndolo adecuado para varios proyectos relacionados con la visión.

6.2.2 Pinout de la Extensión de Cámara ESP32

El diagrama de pinout del ESP32 WROOM 32E se puede encontrar en *Diagrama de Pinout*.

Sin embargo, cuando el ESP32 WROOM 32E está insertado en la placa de extensión, algunos de sus pines también se pueden utilizar para manejar la tarjeta Micro SD o una cámara.

Consecuentemente, se han añadido resistencias de pull-up o pull-down a estos pines. Si está utilizando estos pines como entradas, es crucial tener en cuenta estas resistencias ya que pueden afectar los niveles de entrada.

Aquí está la tabla de pinout para los pines del lado derecho:

ESP32 WROOM 32E + Camera Extension Pinout												
	IO13	IO12	IO14	IO27	IO26	IO25	IO33	IO32	I35	I34	I39	I36
PWM Output												
Input/Output												
Input Only												
Analog	ADC2_4	ADC2_5	ADC2_6	ADC2_7	ADC2_9	ADC2_8	ADC1_5	ADC1_4	ADC1_7	ADC1_6	ADC1_3	ADC1_0
Touch Sensor	TOUCH_4	TOUCH_5	TOUCH_6	TOUCH_7			TOUCH_8	TOUCH_9				
DAC					DAC_2	DAC_1						
I2C												
UART												
SPI	H_MOSI	H_MISO	H_CLK									
LED												
Strapping												
SD	DAT3	DAT2	CLK									
Camera												
Pull-up 47K Resistor												
Pull-up 4.7K resistor												
Pull-down 1K resistor												

Aquí está la tabla de pinout para los pines del lado izquierdo:

ESP32 WROOM 32E + Camera Extension Pinout												
	IO15	IO2	IO0	IO4	IO5	IO18	IO19	IO21	RXD	TXD	IO22	IO23
PWM Output												
Input/Output												
Input Only												
Analog	ADC1_0	ADC2_3	ADC2_2	ADC2_1	ADC2_0							
Touch Sensor	TOUCH_3	TOUCH_2	TOUCH_1	TOUCH_0								
DAC												
I2C								SDA			SCL	
UART									RXD	TXD		
SPI	H_CS				V_CS	V_CLK	V_MISO					V_MOSI
LED		LED										
Strapping												
SD	CMD	DAT0		DAT1								
Camera												
Pull-up 47K Resistor												
Pull-up 4.7K resistor												
Pull-down 1K resistor												

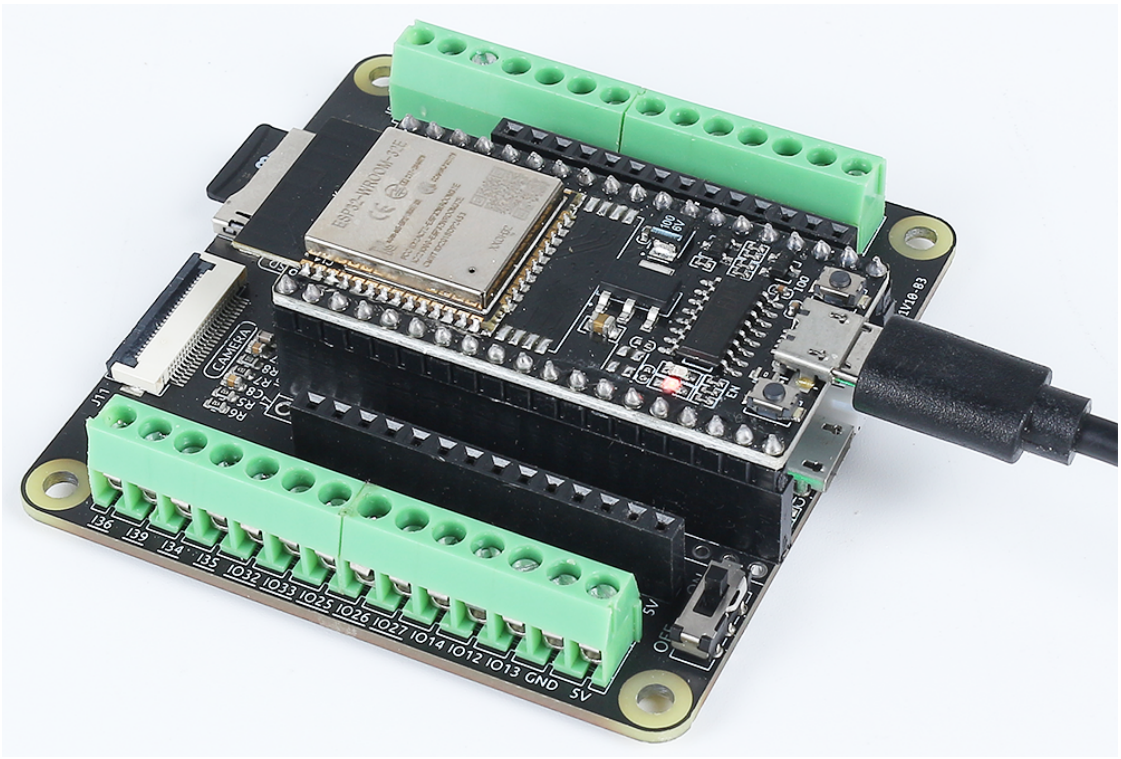
Nota: Hay algunas restricciones específicas:

- **IO33** está conectado a una resistencia de pull-up de 4.7K y un capacitor de filtrado, lo que evita que maneje la tira RGB WS2812.

6.2.3 Guía de Inserción de la Interfaz

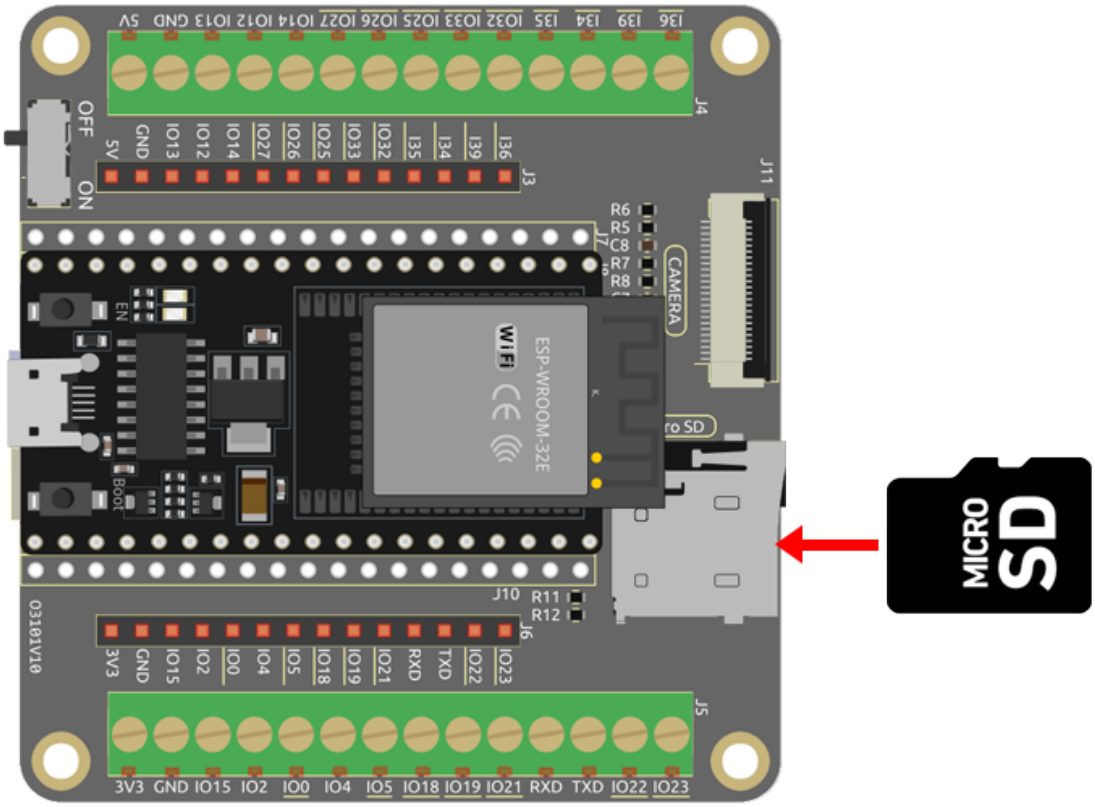
Cargar Código

Cuando necesite cargar código en el ESP32 WROOM 32E, conéctelo a su computadora usando un cable Micro USB.



Insertar la Tarjeta Micro SD

Empuje suavemente la tarjeta Micro SD para asegurarla en su lugar. Presionarla nuevamente la ejectará.

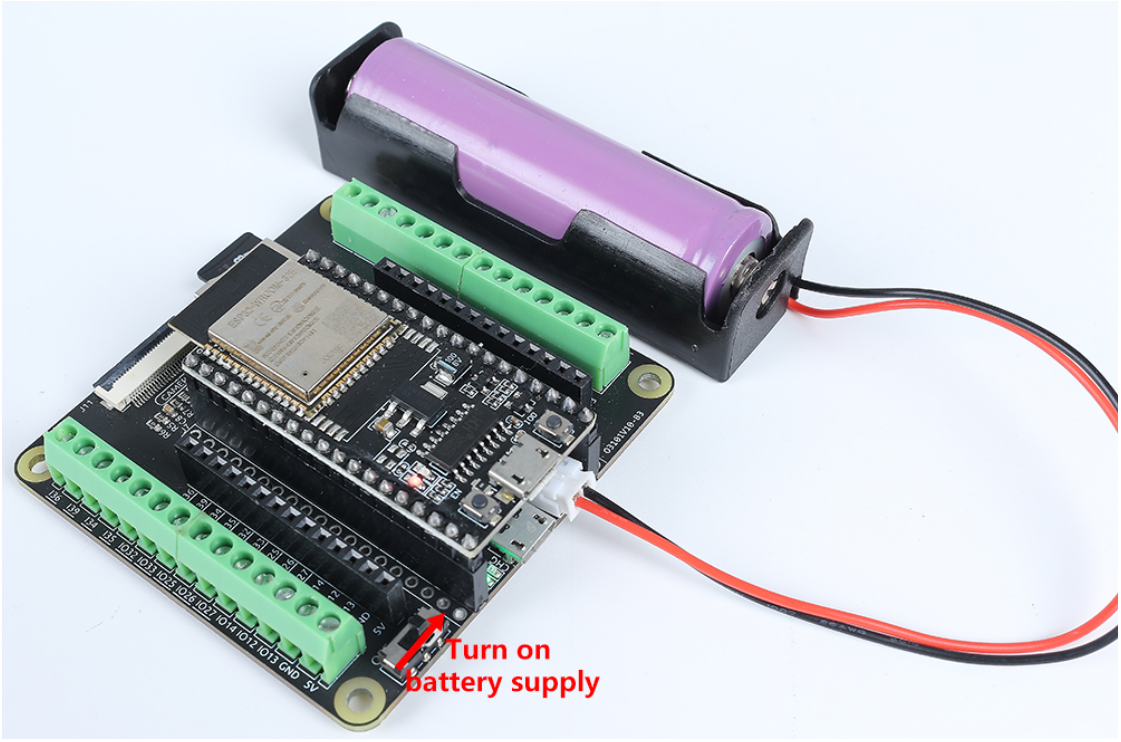


Conexión de la Cámara

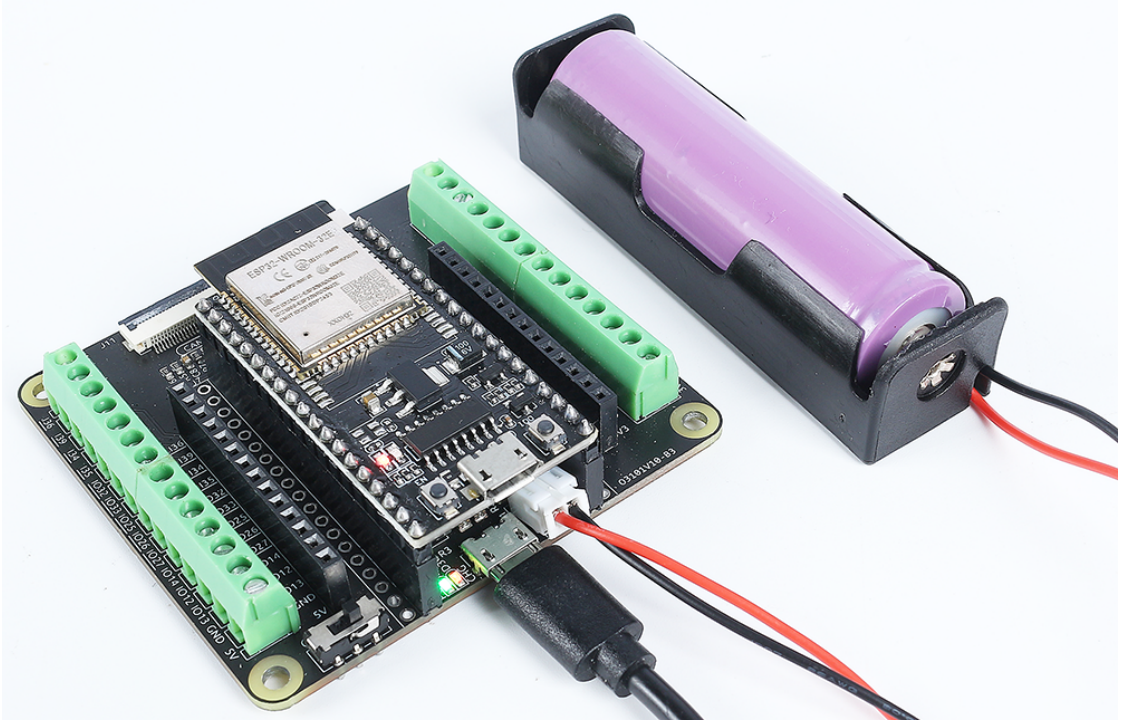
Al conectar la cámara, asegúrese de que la raya negra del cable FPC esté hacia arriba y esté completamente insertada en el conector.

Alimentación de la Batería y Carga

Inserte cuidadosamente el cable de la batería en el puerto de la batería, evitando aplicar demasiada fuerza para evitar empujar hacia arriba el terminal de la batería. Si el terminal se empuja hacia arriba, está bien siempre y cuando los pines no estén rotos; simplemente puede presionarlo nuevamente hacia abajo.



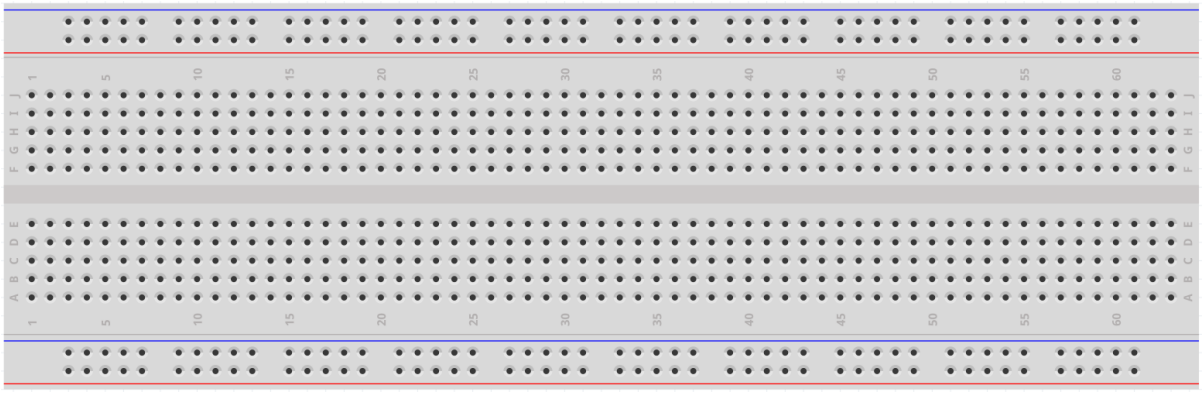
Si la batería se agota, conecte un cable Micro USB de 5V para cargarla.



Básicos

6.3 Protoboard

¿Qué es un protoboard «sin soldadura»?



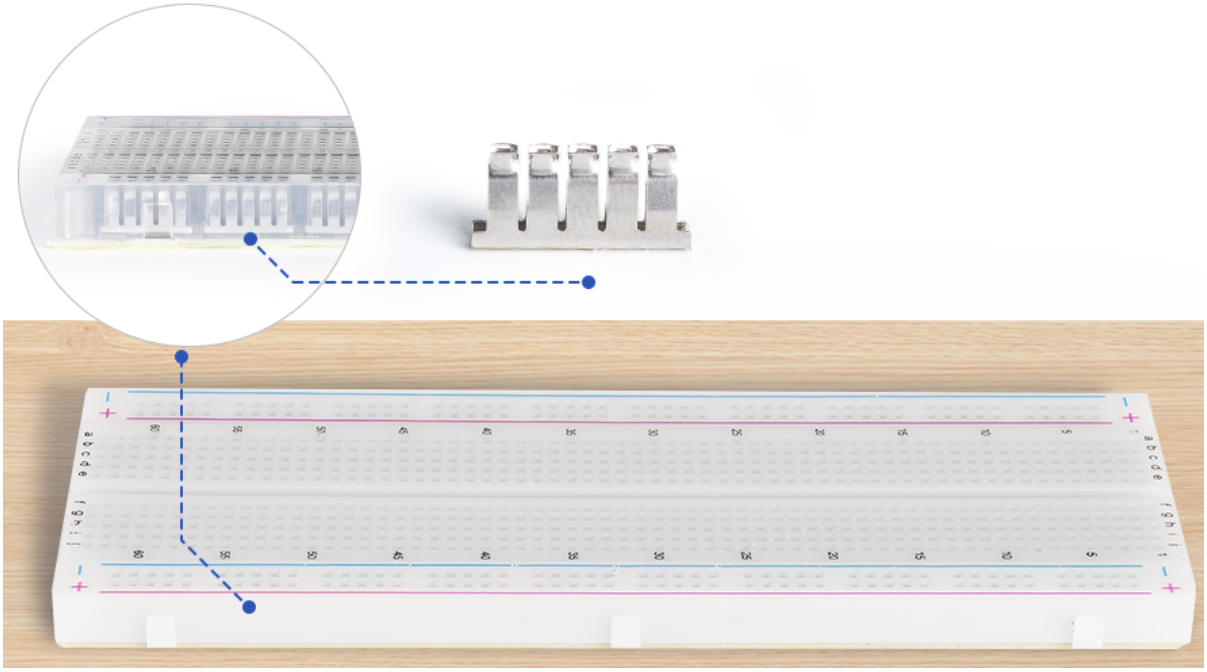
Un protoboard es una placa rectangular de plástico con muchos pequeños agujeros. Estos pequeños agujeros te permiten insertar fácilmente componentes electrónicos para construir circuitos. Técnicamente hablando, estos protoboards son conocidos como protoboards sin soldadura porque no requieren soldadura para hacer conexiones.

Características

- Tamaño: 163 x 54 x 8 mm
- Protoboard de 830 puntos de conexión: área de circuito integrado de 630 puntos de conexión más 2x100 tiras de distribución de puntos de conexión proporcionando 4 rieles de alimentación.
- Tamaño del cable: Adecuado para cables de 20-29 AWG.

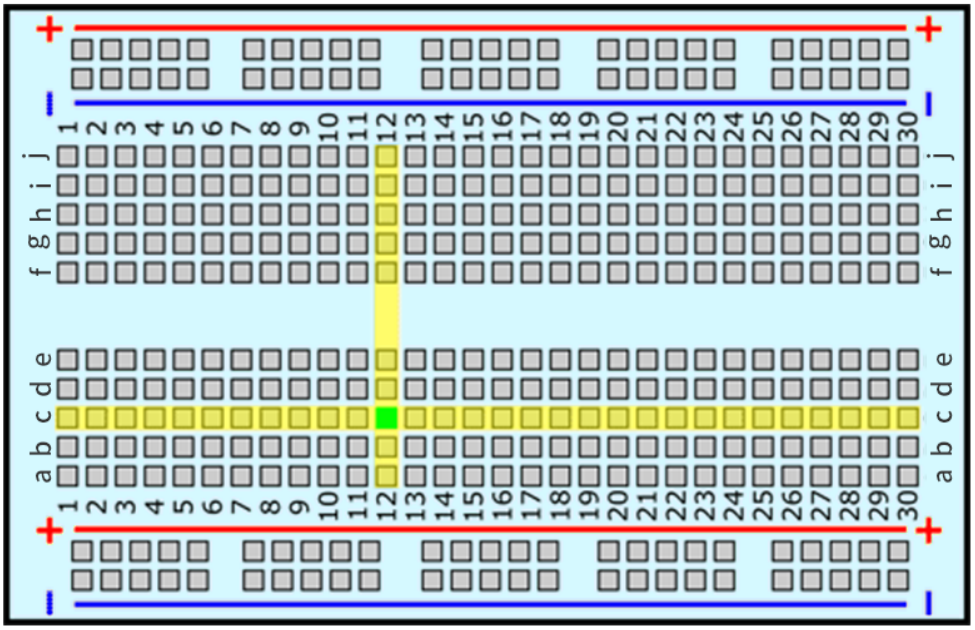
- Material: Panel de plástico ABS, hoja de contacto de bronce fosforado estañado.
- Voltaje / Corriente: 300V/3-5A.
- Con Cinta Adhesiva en la Parte Posterior

¿Qué hay dentro del protoboard?



El interior del protoboard está compuesto por filas de pequeños clips metálicos. Cuando insertas los cables de un componente en los agujeros del protoboard, uno de los clips lo sujeta. Algunos protoboards están hechos de plástico transparente, por lo que puedes ver los clips en su interior.

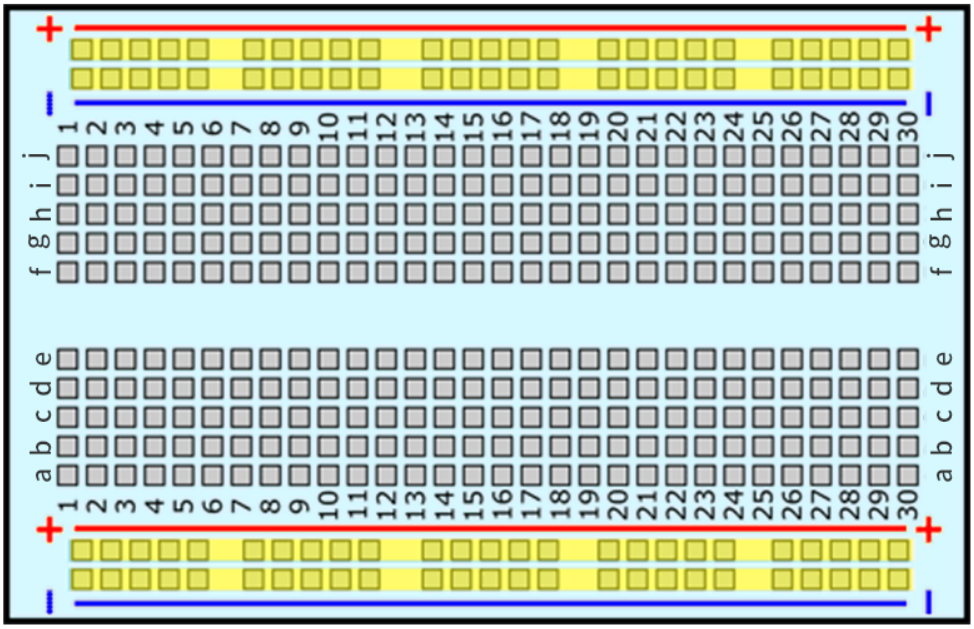
¿Qué significan las letras y números en un protoboard?



La mayoría de los protoboards tienen algunos números, letras y signos más y menos. Aunque las etiquetas variarán de un protoboard a otro, la función es básicamente la misma. Estas etiquetas te permiten encontrar más rápidamente los agujeros correspondientes al construir tu circuito.

Los números de fila y las letras de columna te ayudan a localizar con precisión los agujeros en el protoboard, por ejemplo, el agujero «C12» es donde la columna C se cruza con la fila 12.

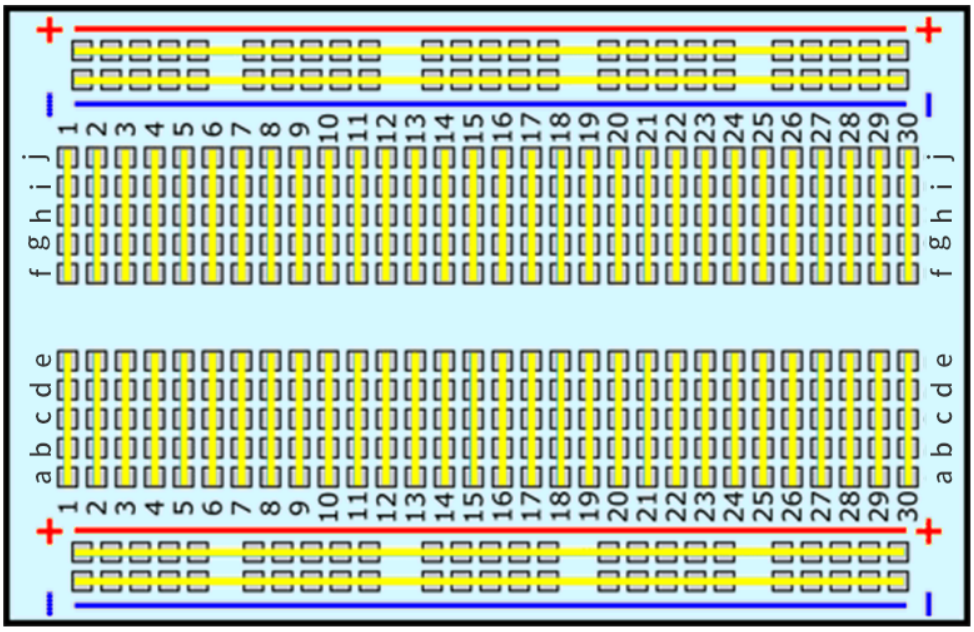
¿Qué significan las líneas de colores y los signos más y menos?



Los lados del protoboard suelen estar diferenciados por rojo y azul (u otros colores), así como por los signos más y menos, y suelen utilizarse para conectarse a la fuente de alimentación, conocida como bus de alimentación.

Al construir un circuito, es común conectar el terminal negativo a la columna azul (-) y el terminal positivo a la columna roja (+).

¿Cómo están conectados los agujeros?

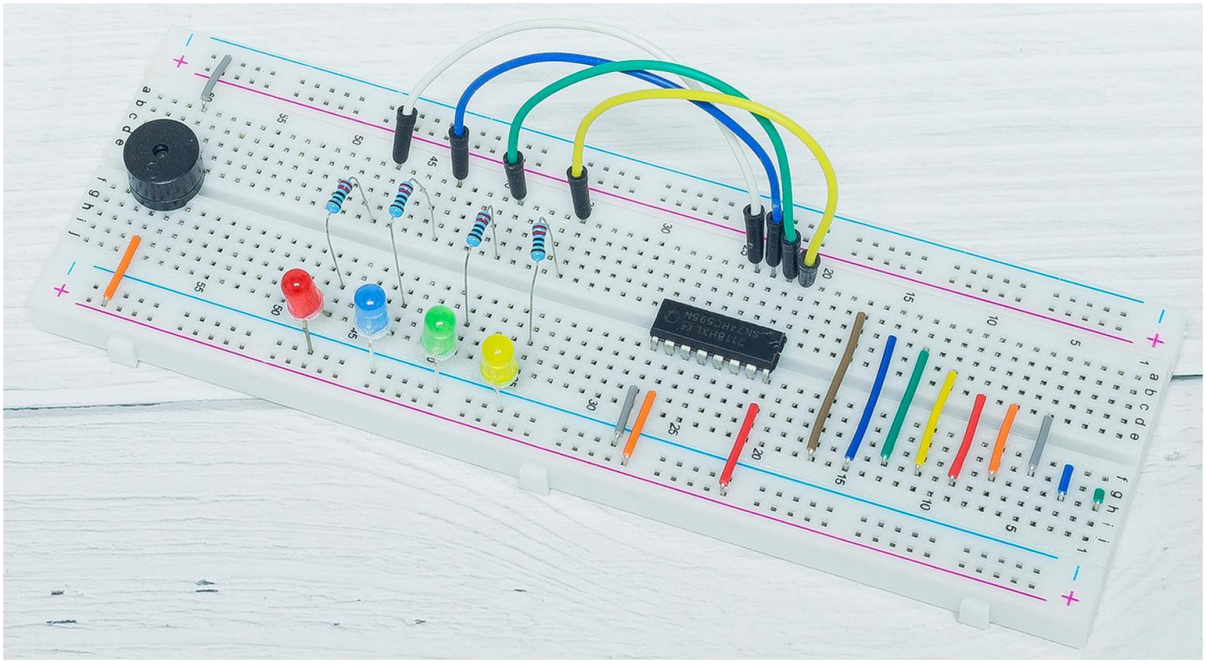


Como se muestra en el diagrama, cada conjunto de cinco agujeros en la sección central, columnas A-E o F-J, está eléctricamente conectado. Esto significa, por ejemplo, que el agujero A1 está eléctricamente conectado a los agujeros B1, C1, D1 y E1.

No está conectado al agujero A2 porque ese agujero está en una fila diferente con un conjunto separado de clips metálicos. Tampoco está conectado a los agujeros F1, G1, H1, I1 o J1 porque están ubicados en la otra «mitad» del protoboard; los clips no están conectados a través del espacio central.

A diferencia de la sección central, que está agrupada por cinco agujeros, los buses en los lados están conectados eléctricamente por separado. Por ejemplo, la columna marcada en azul (-) está eléctricamente conectada en su totalidad, y la columna marcada en rojo (+) también está eléctricamente conectada.

¿Qué partes electrónicas son compatibles con los protoboards?



Muchos componentes electrónicos tienen patas de metal largas llamadas terminales. Casi todos los componentes con terminales funcionarán con un protoboard. Componentes como resistencias, condensadores, interruptores, diodos, etc., se pueden insertar en cualquiera de las filas, pero los IC deben colocarse a través de la brecha central.

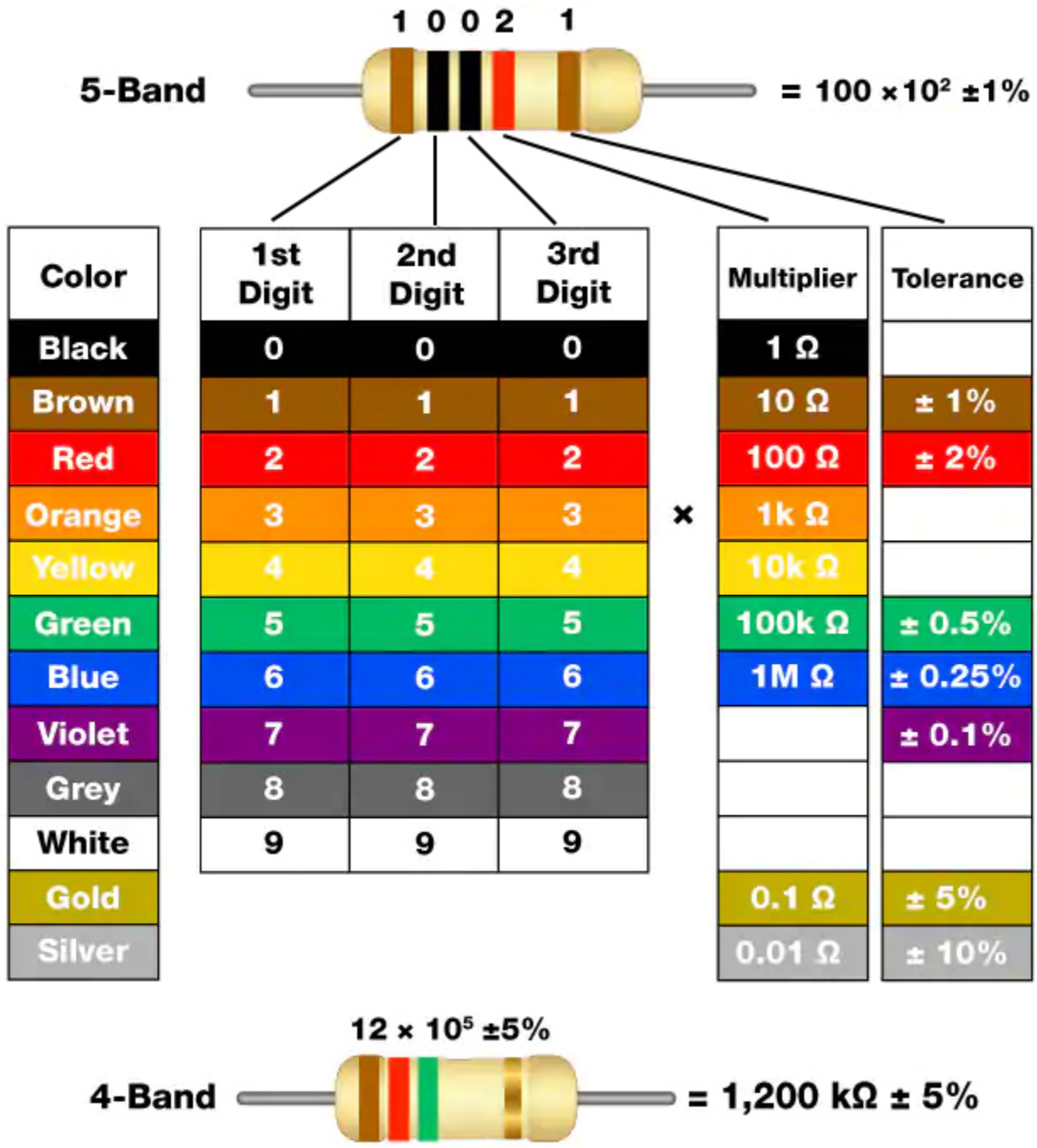
6.4 Resistor

El resistor es un elemento electrónico que puede limitar la corriente de rama. Un resistor fijo es un tipo de resistor cuya resistencia no puede ser cambiada, mientras que la de un potenciómetro o un resistor variable puede ser ajustada.

Existen dos símbolos de circuito generalmente utilizados para representar un resistor. Normalmente, la resistencia está marcada en él. Por lo tanto, si ves estos símbolos en un circuito, representan un resistor.

es la unidad de resistencia y las unidades más grandes incluyen K, M, etc. Su relación se puede mostrar de la siguiente manera: 1 M = 1000 K, 1 K = 1000 . Normalmente, el valor de la resistencia está marcado en él.

Al usar un resistor, primero necesitamos conocer su resistencia. Aquí hay dos métodos: puedes observar las bandas en el resistor, o usar un multímetro para medir la resistencia. Se recomienda utilizar el primer método ya que es más conveniente y rápido.



Como se muestra en la tarjeta, cada color representa un número.

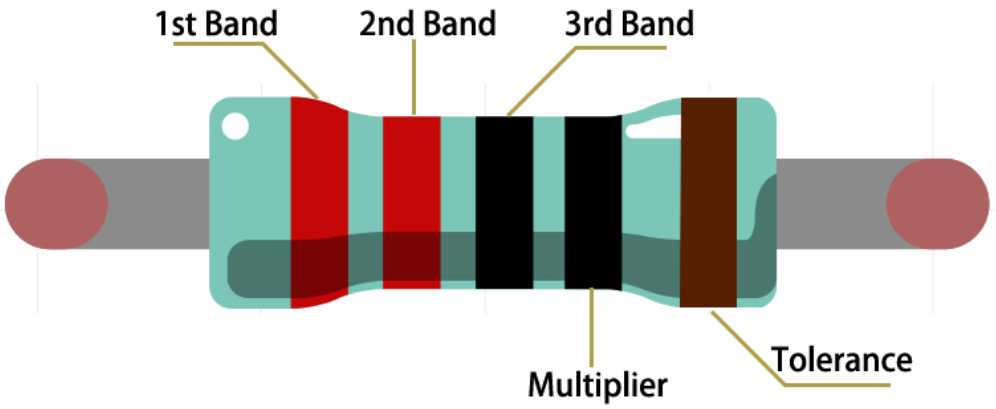
Negro	Marrón	Rojo	Naranja	Amarillo	Verde	Azul	Violeta	Gris	Blanco	Oro	Plata
0	1	2	3	4	5	6	7	8	9	0.1	0.01

Los resistores de 4 y 5 bandas son frecuentemente utilizados, en los cuales hay 4 y 5 bandas cromáticas.

Normalmente, cuando obtienes un resistor, puede resultar difícil decidir por qué extremo comenzar a leer el color. El truco es que el espacio entre la 4ª y la 5ª banda será relativamente más grande.

Por lo tanto, puedes observar el espacio entre las dos bandas cromáticas en un extremo del resistor; si es más grande que cualquier otro espacio entre bandas, entonces puedes leer desde el lado opuesto.

Veamos cómo leer el valor de resistencia de un resistor de 5 bandas como se muestra a continuación.



Así que para este resistor, la resistencia debe leerse de izquierda a derecha. El valor debe estar en este formato: 1ª Banda 2ª Banda 3ª Banda x 10^Multiplicador () y el error permitido es ± Tolerancia %. Por lo tanto, el valor de resistencia de este resistor es 2(rojo) 2(rojo) 0(negro) x 10^0(negro) = 220 , y el error permitido es ± 1 % (marrón).

Tabla 1: Bandas de color comunes de resistores

Resistencia	Banda de Color
10	marrón negro negro plata marrón
100	marrón negro negro negro marrón
220	rojo rojo negro negro marrón
330	naranja naranja negro negro marrón
1k	marrón negro negro marrón marrón
2k	rojo negro negro marrón marrón
5.1k	verde marrón negro marrón marrón
10k	marrón negro negro rojo marrón
100k	marrón negro negro naranja marrón
1M	marrón negro negro verde marrón

Puedes aprender más sobre los resistores en Wikipedia: Resistor - Wikipedia.

6.5 Capacitor

El capacitor, se refiere a la cantidad de almacenamiento de carga bajo una diferencia de potencial dada, denotada como C, y la unidad internacional es el faradio (F). En general, las cargas eléctricas se mueven bajo la fuerza en un campo eléctrico. Cuando hay un medio entre conductores, el movimiento de las cargas eléctricas se ve obstaculizado y las cargas eléctricas se acumulan en los conductores, lo que resulta en la acumulación de cargas eléctricas.

La cantidad de cargas eléctricas almacenadas se llama capacitancia. Debido a que los capacitores son uno de los componentes electrónicos más ampliamente utilizados en equipos electrónicos, se utilizan ampliamente en el aislamiento de corriente continua, acoplamiento, derivación, filtrado, bucles de sintonización, conversión de energía y circuitos de control. Los capacitores se dividen en capacitores electrolíticos, capacitores sólidos, etc.

Según las características del material, los capacitores se pueden dividir en: capacitores electrolíticos de aluminio, capacitores de película, capacitores de tantalio, capacitores cerámicos, supercondensadores, etc.

En este kit, se utilizan capacitores cerámicos y capacitores electrolíticos.

- Capacitor cerámico - Wikipedia
- Capacitor electrolítico - Wikipedia

Hay una etiqueta 103 o 104 en los capacitores cerámicos, que representan el valor de la capacitancia, 103=10x10^3pF, 104=10x10^4pF

Conversión de Unidades

1F=10^3mF=10^6uF=10^9nF=10^12pF

6.6 Cables Puente

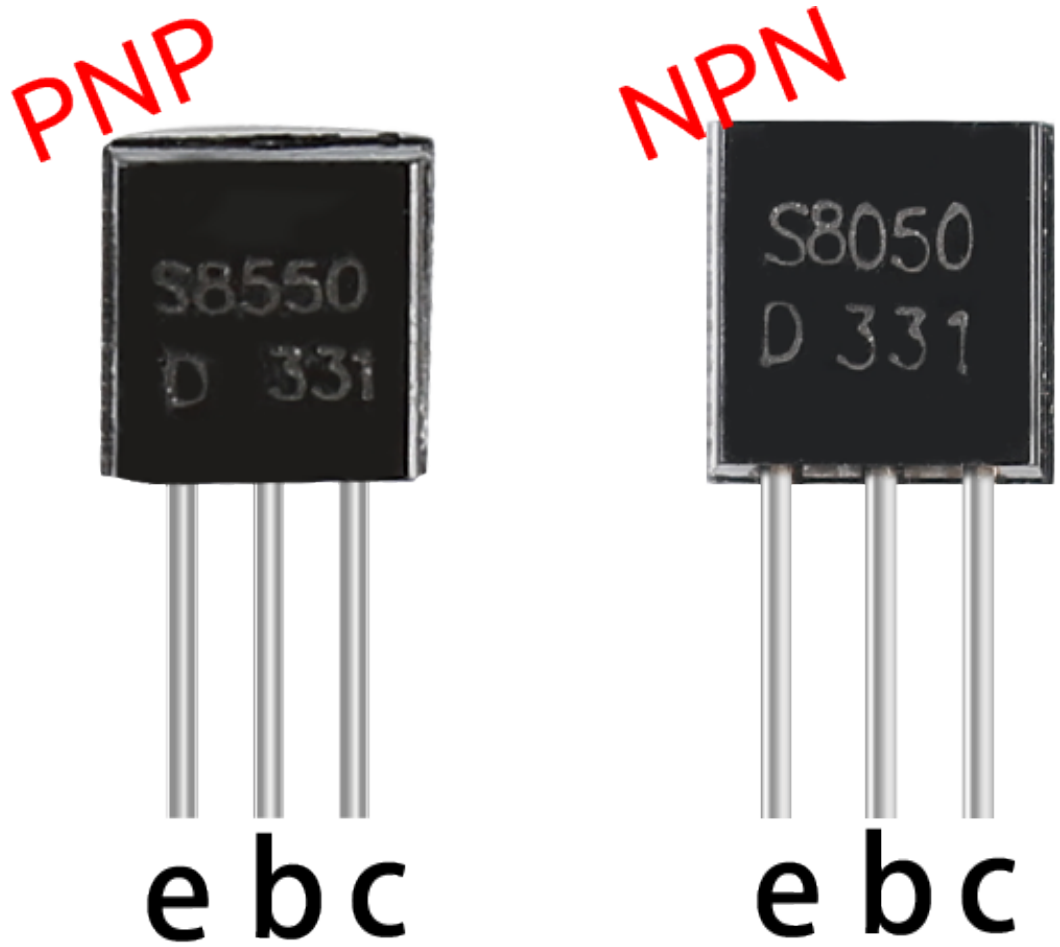
Los cables que conectan dos terminales se llaman cables puente. Hay varios tipos de cables puente. Aquí nos enfocamos en los utilizados en la placa de pruebas. Entre otros usos, se utilizan para transferir señales eléctricas desde cualquier lugar en la placa de pruebas a los pines de entrada/salida de un microcontrolador.

Los cables puente se colocan insertando sus «conectores finales» en las ranuras provistas en la placa de pruebas, debajo de cuya superficie hay unos pocos conjuntos de placas paralelas que conectan las ranuras en grupos de filas o columnas dependiendo del área. Los «conectores finales» se insertan en la placa de pruebas, sin necesidad de soldadura, en las ranuras particulares que necesitan estar conectadas en el prototipo específico.

Hay tres tipos de cables puente: Hembra-Hembra, Macho-Macho y Macho-Hembra. La razón por la que lo llamamos Macho-Hembra es porque tiene la punta destacada en un extremo, así como un extremo hembra hundido. Macho-Macho significa que ambos lados son machos y Hembra-Hembra significa que ambos extremos son hembras.

Más de un tipo de ellos puede ser utilizado en un proyecto. El color de los cables puente es diferente, pero eso no significa que su función sea diferente en consecuencia; está diseñado así para identificar mejor la conexión entre cada circuito.

6.7 Transistor



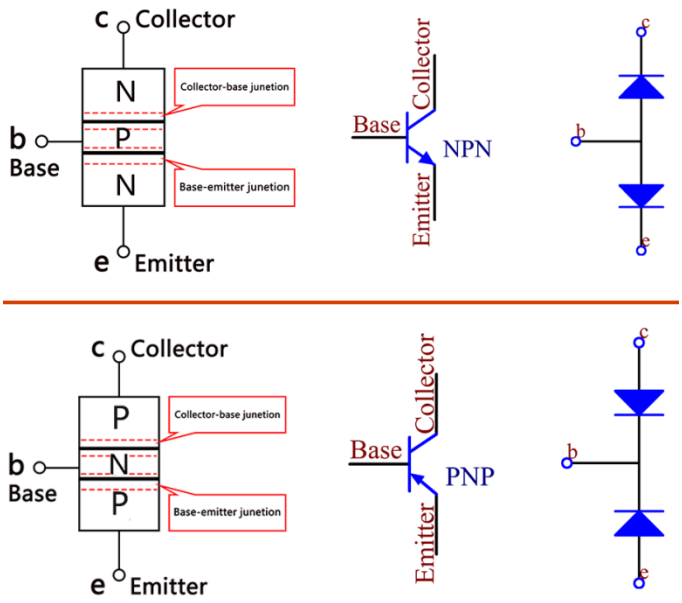
El transistor es un dispositivo semiconductor que controla la corriente mediante corriente. Funciona amplificando una señal débil a una señal de mayor amplitud y también se utiliza como interruptor sin contacto.

Un transistor es una estructura de tres capas compuesta por semiconductores de tipo P y N. Internamente forman las tres regiones. La más delgada en el medio es la región de la base; las otras dos son ambas de tipo N o P: la región más pequeña con portadores de mayoría intensos es la región del emisor, mientras que la otra es la región del colector. Esta composición permite que el transistor sea un amplificador. De estas tres regiones se generan tres polos respectivamente, que son base (b), emisor (e) y colector (c). Forman dos uniones P-N, a saber, la unión del emisor y la unión de recolección. La dirección de la flecha en el símbolo del circuito del transistor indica la dirección de la unión del emisor.

- Unión P-N - Wikipedia

Según el tipo de semiconductor, los transistores se pueden dividir en dos grupos, los NPN y PNP. Por la abreviatura, podemos deducir que el primero está hecho de dos semiconductores de tipo N y uno de tipo P, y que el segundo es lo contrario. Consulte la siguiente figura.

Nota: El s8550 es un transistor PNP y el s8050 es un transistor NPN, se parecen mucho, y debemos verificar cuidadosamente para ver sus etiquetas.



Cuando una señal de nivel alto atraviesa un transistor NPN, se energiza. Pero uno PNP necesita una señal de nivel bajo para gestionarlo. Ambos tipos de transistores se utilizan con frecuencia para interruptores sin contacto, al igual que en este experimento.

- Hoja de datos del transistor S8050
- Hoja de datos del transistor S8550

Coloque el lado de la etiqueta hacia nosotros y los pines hacia abajo. Los pines de izquierda a derecha son emisor (e), base (b) y colector (c).

Nota:

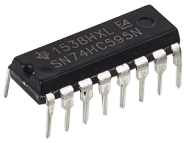
- La base es el dispositivo controlador de puerta para el suministro eléctrico más grande.
- En el transistor NPN, el colector es el suministro eléctrico más grande y el emisor es la salida para ese suministro, en el transistor PNP es justo lo contrario.

Ejemplo

- *5.6 Dos Tipos de Transistores* (Proyecto Arduino)
- *3.1 Beep* (Proyecto Arduino)
- *6.1 Piano de Frutas* (Proyecto Arduino)
- *5.6 Dos Tipos de Transistores* (Proyecto MicroPython)
- *3.2 Tono Personalizado* (Proyecto MicroPython)
- *6.3 Teremín de Luz* (Proyecto MicroPython)

Chip

6.8 74HC595



¿Alguna vez te has encontrado queriendo controlar una gran cantidad de LED, o simplemente necesitas más pines de E/S para controlar botones, sensores y servos al mismo tiempo? Bueno, puedes conectar algunos sensores a los pines de Arduino, pero pronto comenzarás a quedarte sin pines en el Arduino.

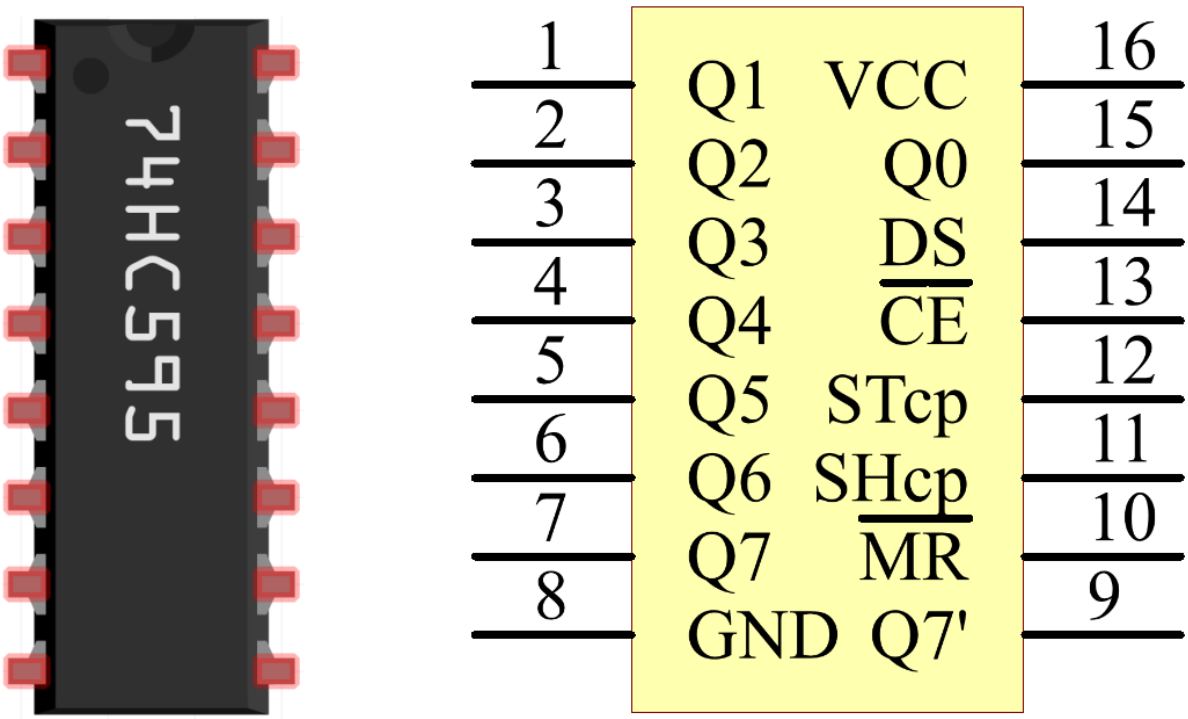
La solución es usar «registros de desplazamiento». Los registros de desplazamiento te permiten expandir el número de pines de E/S que puedes usar desde el Arduino (o cualquier microcontrolador). El registro de desplazamiento 74HC595 es uno de los más famosos.

El 74HC595 básicamente controla ocho pines de salida independientes y utiliza solo tres pines de entrada. Si necesitas más de ocho líneas de E/S adicionales, puedes fácilmente concatenar cualquier número de registros de desplazamiento y crear un gran número de líneas de E/S. Todo esto se hace mediante el llamado desplazamiento.

Características

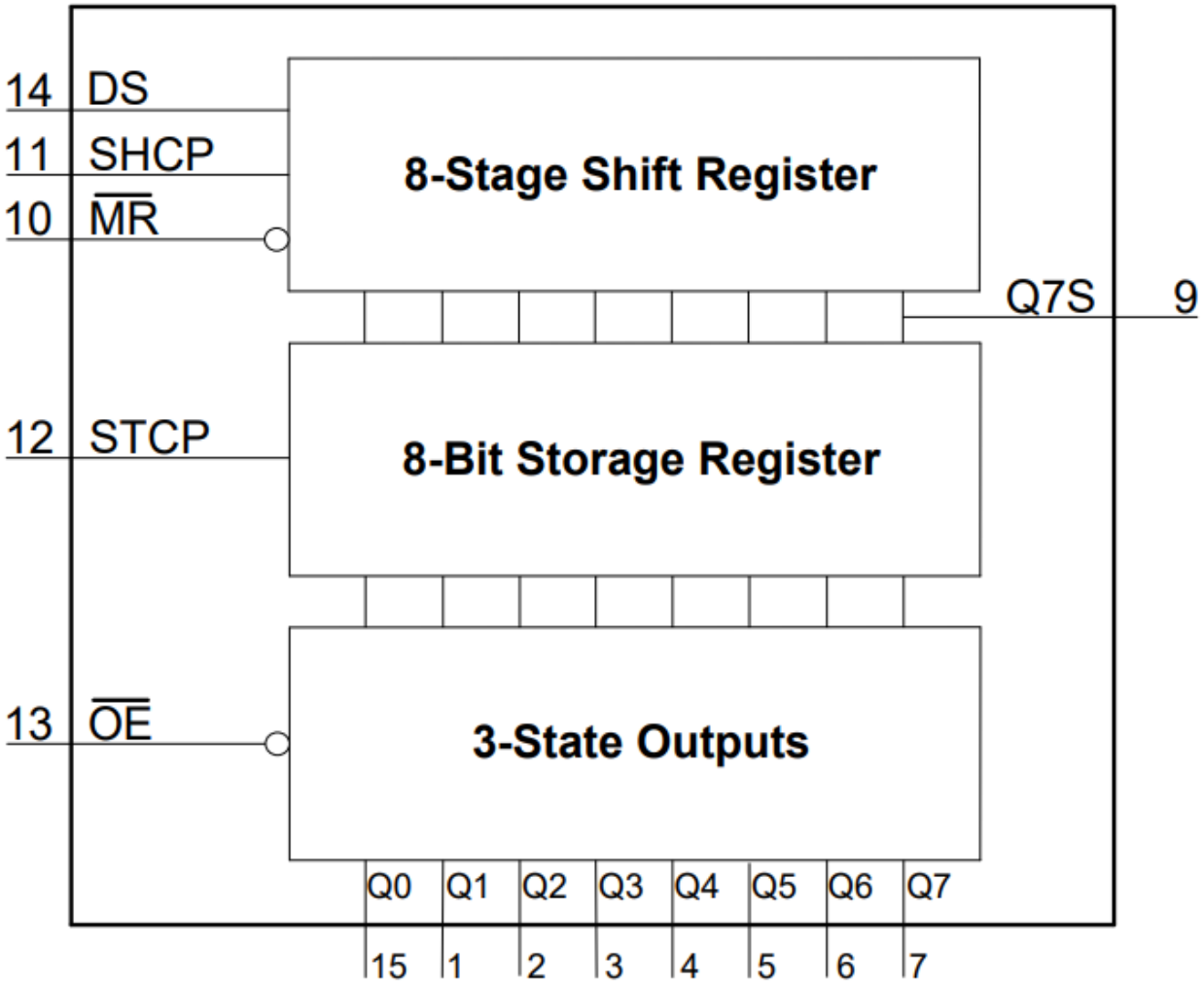
- Registro de desplazamiento serial de 8 bits, paralelo de salida
- Amplio rango de voltaje de operación de 2 V a 6 V
- Las salidas de 3 estados de alta corriente pueden manejar hasta 15 cargas LSTTL
- Bajo consumo de energía, máx. 80 µA de corriente de CC
- tPD típico = 14 ns
- Salida de ±6 mA a 5 V
- Baja corriente de entrada de 1 µA máx.
- Registro de desplazamiento con limpieza directa

Pines de 74HC595 y sus funciones:



- **Q0-Q7**: Pines de salida de datos paralelos de 8 bits, capaces de controlar 8 LED o 8 pines de un display de 7 segmentos directamente.
- **Q7”**: Pin de salida en serie, conectado a DS de otro 74HC595 para conectar múltiples 74HC595 en serie.
- **MR**: Pin de reset, activo en nivel bajo;
- **SHcp**: Entrada de secuencia de tiempo del registro de desplazamiento. En el flanco ascendente, los datos en el registro de desplazamiento se desplazan sucesivamente un bit, es decir, los datos en Q1 se desplazan a Q2, y así sucesivamente. Mientras que en el flanco descendente, los datos en el registro de desplazamiento permanecen sin cambios.
- **STcp**: Entrada de secuencia de tiempo del registro de almacenamiento. En el flanco ascendente, los datos en el registro de desplazamiento se mueven al registro de memoria.
- **CE**: Pin de habilitación de salida, activo en nivel bajo.
- **DS**: Pin de entrada de datos en serie
- **VCC**: Voltaje de alimentación positivo.
- **GND**: Tierra.

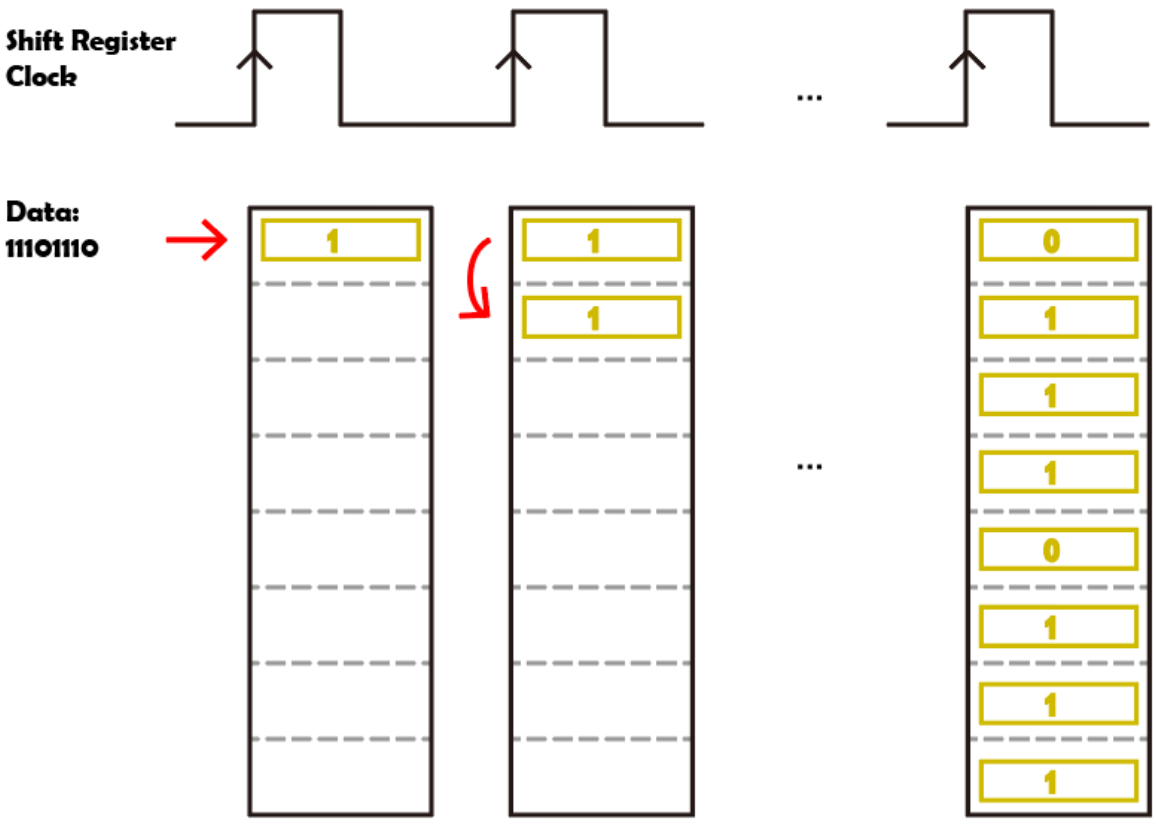
Diagrama Funcional



Principio de Funcionamiento

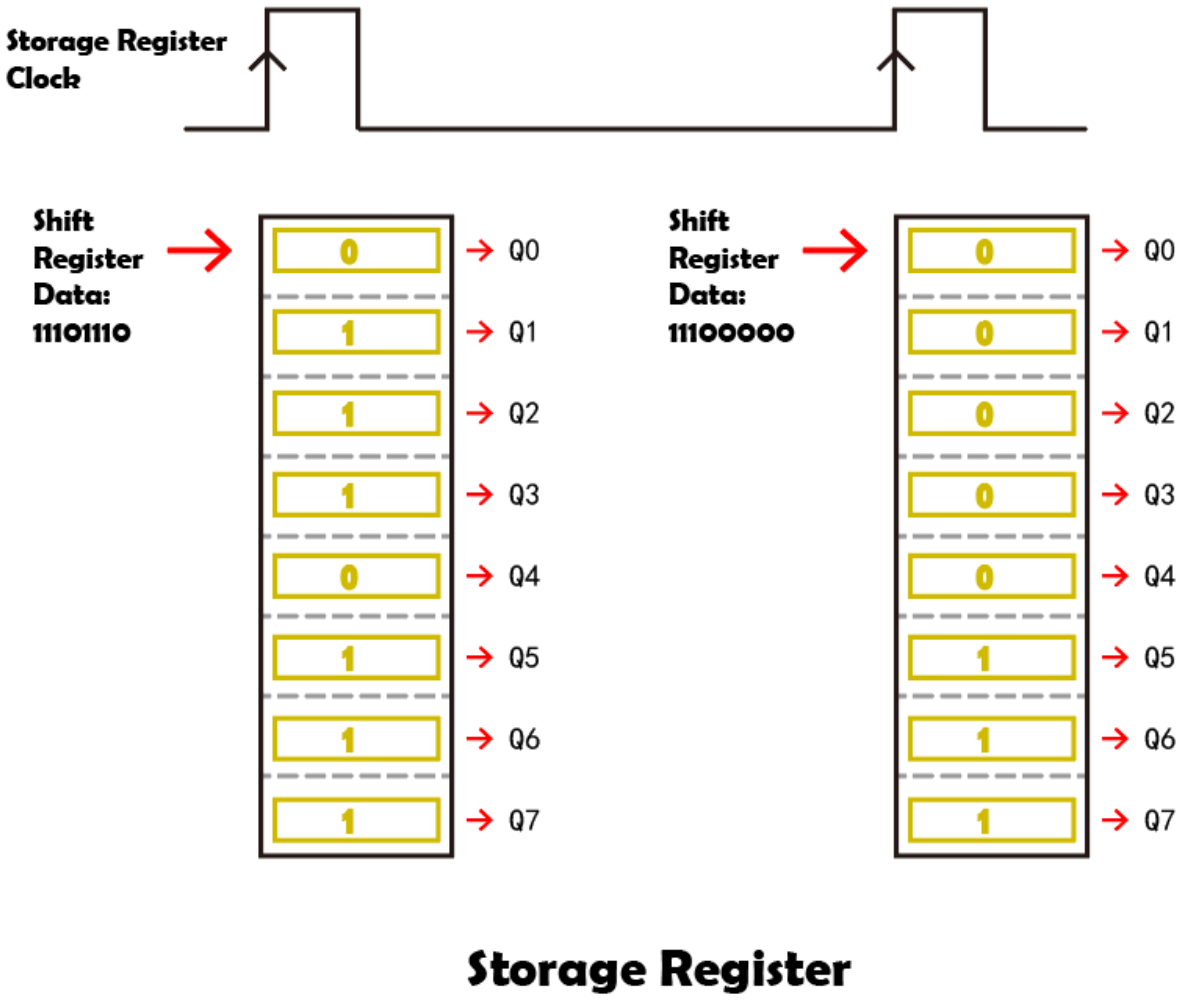
Cuando MR (pin 10) está en nivel alto y OE (pin 13) está en nivel bajo, los datos se ingresan en el flanco ascendente de SHcp y pasan al registro de memoria a través del flanco ascendente de STcp.

- Registro de Desplazamiento
  - Supongamos que queremos ingresar los datos binarios 1110 1110 en el registro de desplazamiento del 74hc595.
  - Los datos se ingresan desde el bit 0 del registro de desplazamiento.
  - Cada vez que el reloj del registro de desplazamiento es un flanco ascendente, los bits en el registro de desplazamiento se desplazan un paso. Por ejemplo, el bit 7 acepta el valor anterior en el bit 6, el bit 6 obtiene el valor del bit 5, etc.



Shift Register

- Registro de Almacenamiento
  - Cuando el registro de almacenamiento está en estado de flanco ascendente, los datos del registro de desplazamiento se transferirán al registro de almacenamiento.
  - El registro de almacenamiento está conectado directamente a los 8 pines de salida, Q0 ~ Q7 podrá recibir un byte de datos.
  - El llamado registro de almacenamiento significa que los datos pueden existir en este registro y no desaparecerán con una salida.
  - Los datos permanecerán válidos e inalterados siempre que el 74HC595 esté alimentado continuamente.
  - Cuando llegan nuevos datos, los datos en el registro de almacenamiento serán sobrescritos y actualizados.



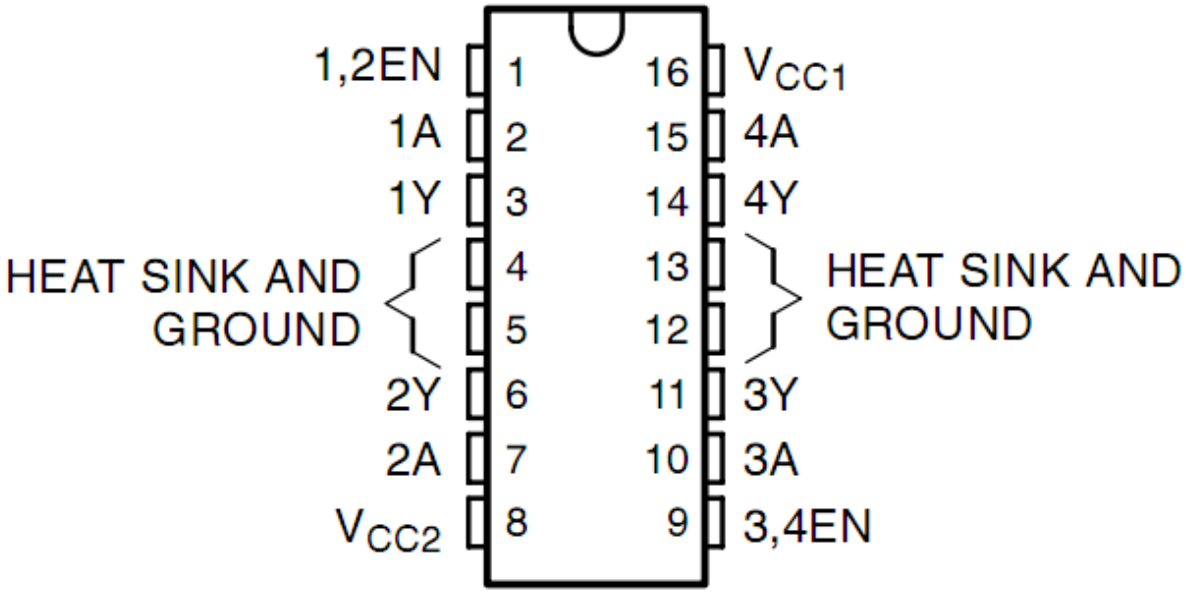
Ejemplo

- *2.4 Microchip - 74HC595* (Proyecto Arduino)
- *2.5 Pantalla de 7 Segmentos* (Proyecto Arduino)
- *6.4 Dados Digitales* (Proyecto Arduino)
- *2.4 Microchip - 74HC595* (Proyecto MicroPython)
- *2.5 Visualización de Números* (Proyecto MicroPython)
- *6.6 Dado Digital* (Proyecto MicroPython)

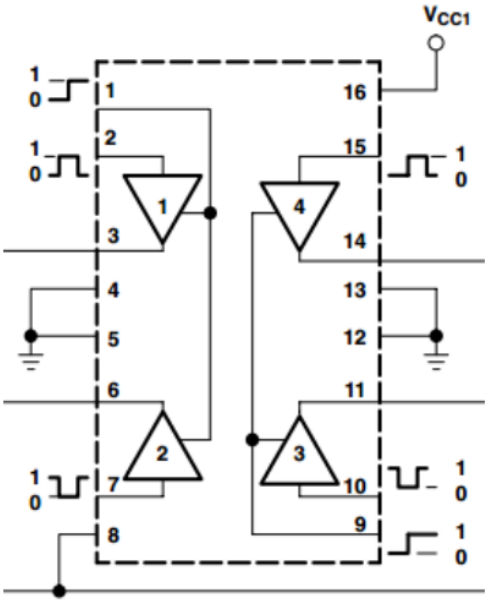
6.9 L293D

L293D es un controlador de motor de 4 canales integrado en un chip con alta tensión y alta corriente. Está diseñado para conectarse a lógica estándar DTL, TTL, y para controlar cargas inductivas (como bobinas de relé, motores de CC, motores paso a paso) y transistores de conmutación de potencia, etc. Los motores de CC son dispositivos que convierten la energía eléctrica de CC en energía mecánica. Se utilizan ampliamente en accionamientos eléctricos por su excelente rendimiento de regulación de velocidad.

Vea la figura de los pines a continuación. L293D tiene dos pines (Vcc1 y Vcc2) para la fuente de alimentación. Vcc2 se utiliza para suministrar energía al motor, mientras que Vcc1 se utiliza para alimentar el chip. Dado que aquí se utiliza un motor de CC de pequeño tamaño, conecte ambos pines a +5V.



A continuación se muestra la estructura interna de L293D. El pin EN es un pin de habilitación y solo funciona con nivel alto; A representa la entrada y Y la salida. Puede ver la relación entre ellos en la esquina inferior derecha. Cuando el pin EN está en nivel alto, si A está en alto, Y emite un nivel alto; si A está en bajo, Y emite un nivel bajo. Cuando el pin EN está en nivel bajo, el L293D no funciona.



INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

- Hoja de datos de L293D

Ejemplo

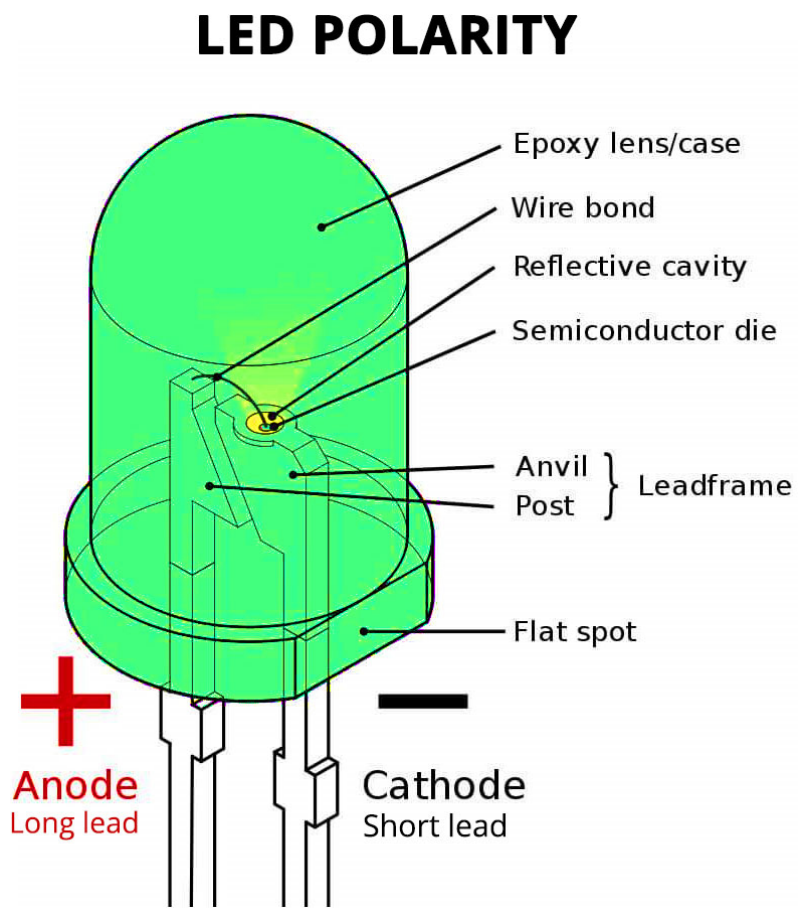
- *4.1 Motor* (Proyecto Arduino)
- *4.2 Bombeo* (Proyecto Arduino)
- *4.1 Pequeño Ventilador* (Proyecto MicroPython)
- *4.2 Bombeo* (Proyecto MicroPython)
- *2.9 Ventilador Rotativo* (Proyecto Scratch)

Display

6.10 LED

¿Qué es un LED?

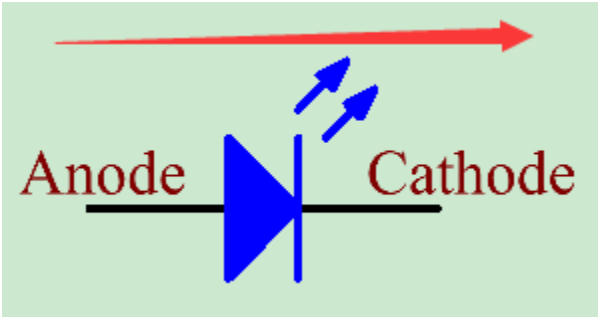

Cathode(-)
Anode(+)
Longer Lead



Los LEDs son dispositivos electrónicos muy comunes que se pueden utilizar para decorar tu habitación durante las festividades, y también puedes usarlos como indicadores para varias cosas, como si la alimentación de tus electrodomésticos está encendida o apagada. Vienen en docenas de formas y tamaños diferentes, y los más comunes son los LEDs con LEDs de agujero pasante, que generalmente tienen cables largos y se pueden enchufar en una placa de pruebas.

El nombre completo de LED es diodo emisor de luz, por lo que tiene las características de un diodo, donde la corriente fluye en una dirección, desde el ánodo (positivo) hasta el cátodo (negativo).

Aquí están los símbolos eléctricos para los LEDs.

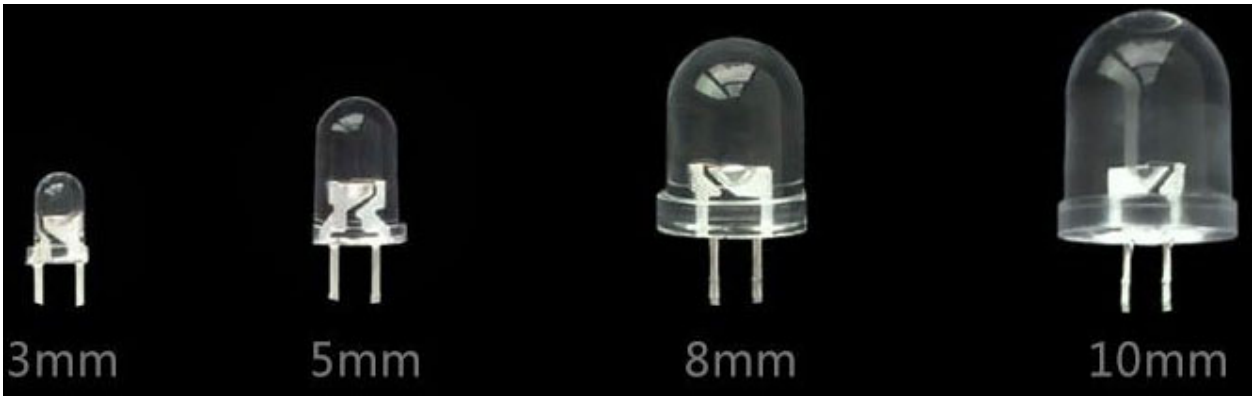


Varios tamaños y colores

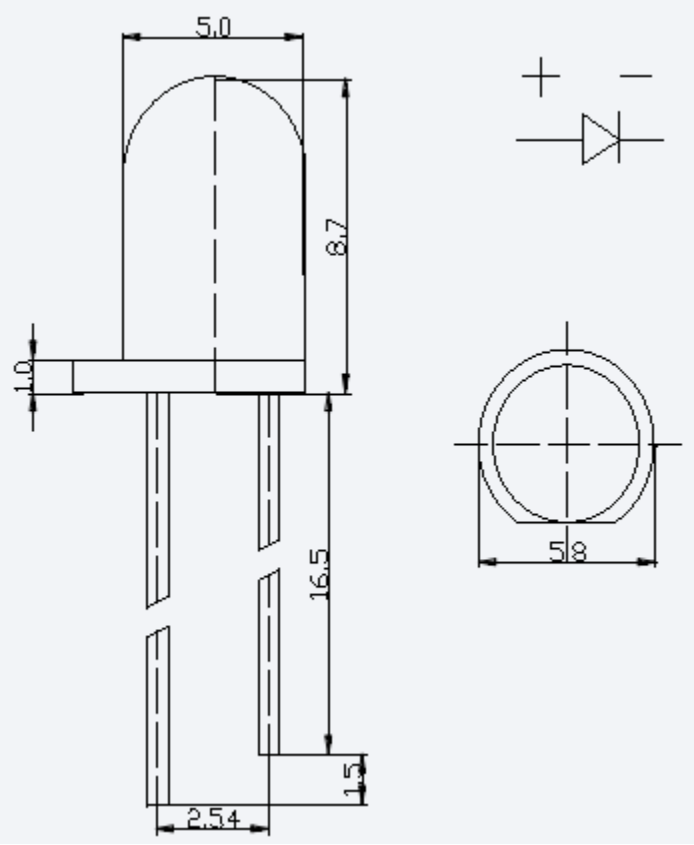
Rojo, amarillo, azul, verde y blanco son los colores de LED más comunes, y la luz emitida suele ser del mismo color que el aspecto.

Raramente usamos LEDs que son transparentes o mate en apariencia, pero la luz emitida puede ser de un color que no sea blanco.

Los LEDs vienen en cuatro tamaños: 3mm, 5mm, 8mm y 10mm, siendo el tamaño más común el de 5mm.



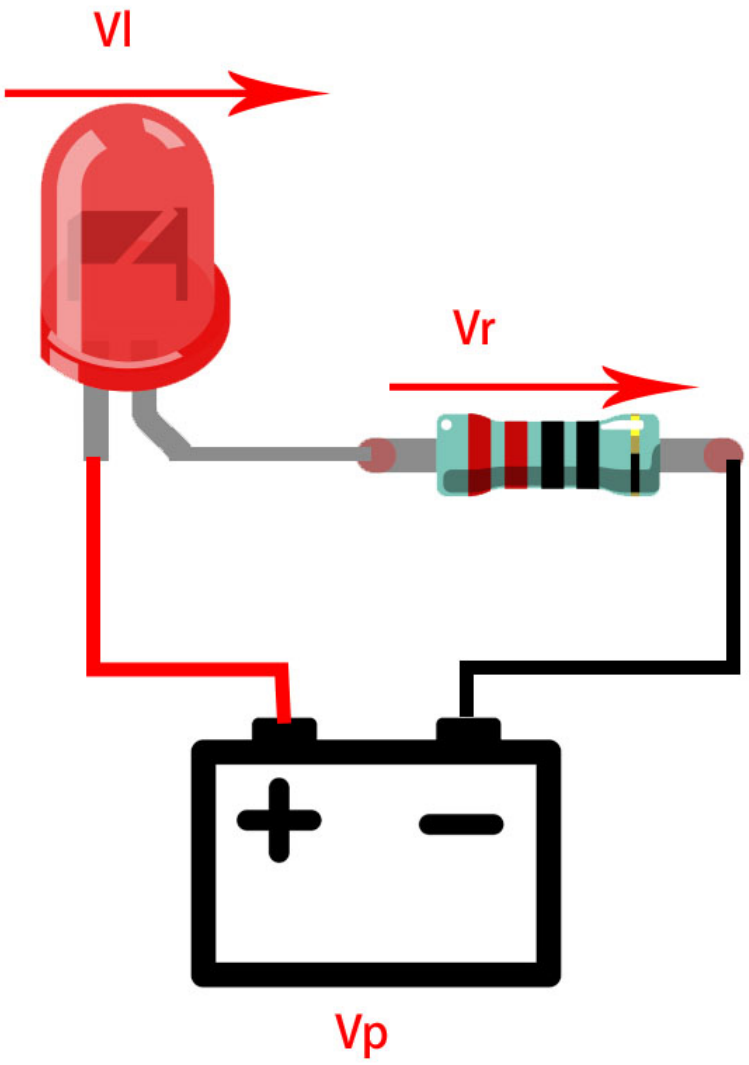
A continuación se muestra el tamaño del LED de 5mm en mm.



Voltaje Directo

El Voltaje Directo es un parámetro muy importante para conocer al usar LEDs, ya que determina cuánta energía se utiliza y cuánto debe ser la resistencia limitadora de corriente.

El Voltaje Directo es el voltaje que el LED necesita para encenderse. Para la mayoría de los LEDs rojos, amarillos, naranjas y verdes claros, generalmente utilizan un voltaje entre 1.9V y 2.1V.



Según la ley de Ohm, la corriente a través de este circuito disminuye a medida que aumenta la resistencia, lo que hace que el LED se atenúe.

I = (Vp-Vl)/R

Para que los LEDs se enciendan de manera segura y con el brillo adecuado, ¿cuánta resistencia deberíamos usar en el circuito?

Para el 99 % de los LEDs de 5mm, la corriente recomendada es de 20mA, como se puede ver en la columna de Condiciones de su hoja de datos.

Electrical / Optical Characteristics at TA=25°C

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Forward Voltage	VF	Lf=20mA	1.9		2.1	V
Reverd Current	IR	VR=5V			10	uA
Vtewing Angle	201/2	IF=20mA		20		deg
Lulninous Intensity	ϕv	IF=20mA	3000		4000	mcd
Chromaticity	Tc	IF=20mA	620		625	K
Chromaticity Coordinates	X.Y	IF=20mA	0. 29, 0. 29			

Ahora convertimos la fórmula anterior como se muestra a continuación.

R = (Vp-Vl)/I

Si `Vp` es 5V, `Vl` (Voltaje Directo) es 2V y `I` es 20mA, entonces `R` es 150.

Por lo tanto, podemos hacer que el LED sea más brillante al reducir la resistencia del resistor, pero no se recomienda bajar de 150 (esta resistencia puede no ser muy precisa, porque diferentes proveedores de LEDs tienen diferencias).

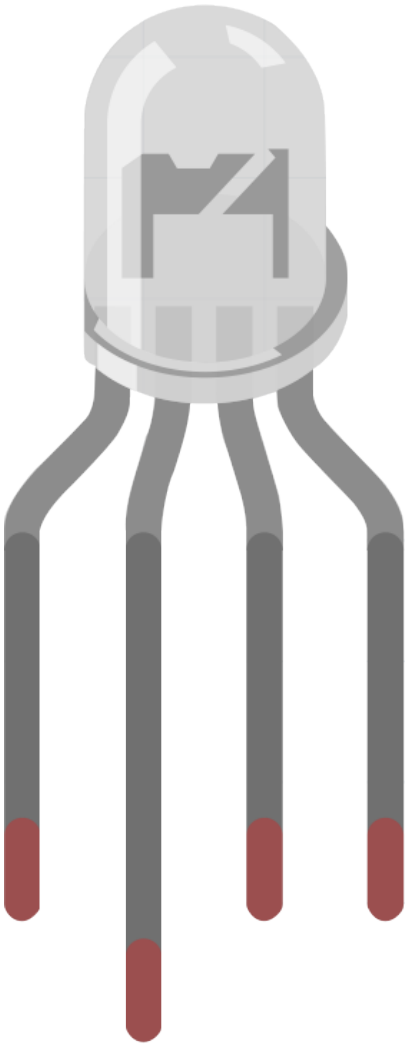
A continuación se muestran los voltajes directos y longitudes de onda de diferentes colores de LEDs que puedes usar como referencia.

Color del LED	Voltaje Directo	Longitud de Onda
Rojo	1.8V ~ 2.1V	620 ~ 625
Amarillo	1.9V ~ 2.2V	580 ~ 590
Verde	1.9V ~ 2.2V	520 ~ 530
Azul	3.0V ~ 3.2V	460 ~ 465
Blanco	3.0V ~ 3.2V	8000 ~ 9000

Ejemplo

- *2.1 ¡Hola, LED!* (Proyecto Arduino)
- *2.2 Desvanecimiento* (Proyecto Arduino)
- *2.1 ¡Hola, LED!* (Proyecto MicroPython)
- *2.2 Atenuación de un LED* (Proyecto MicroPython)
- *2.2 LED Respirando* (Proyecto Scratch)
- *2.1 Lámpara de Mesa* (Proyecto Scratch)

6.11 LED RGB



Los LED RGB emiten luz en varios colores. Un LED RGB combina tres LED de color rojo, verde y azul en una carcasa transparente o semitransparente de plástico. Puede mostrar varios colores cambiando el voltaje de entrada de los tres pines y superponiéndolos, lo que, según estadísticas, puede crear 16,777,216 colores diferentes.

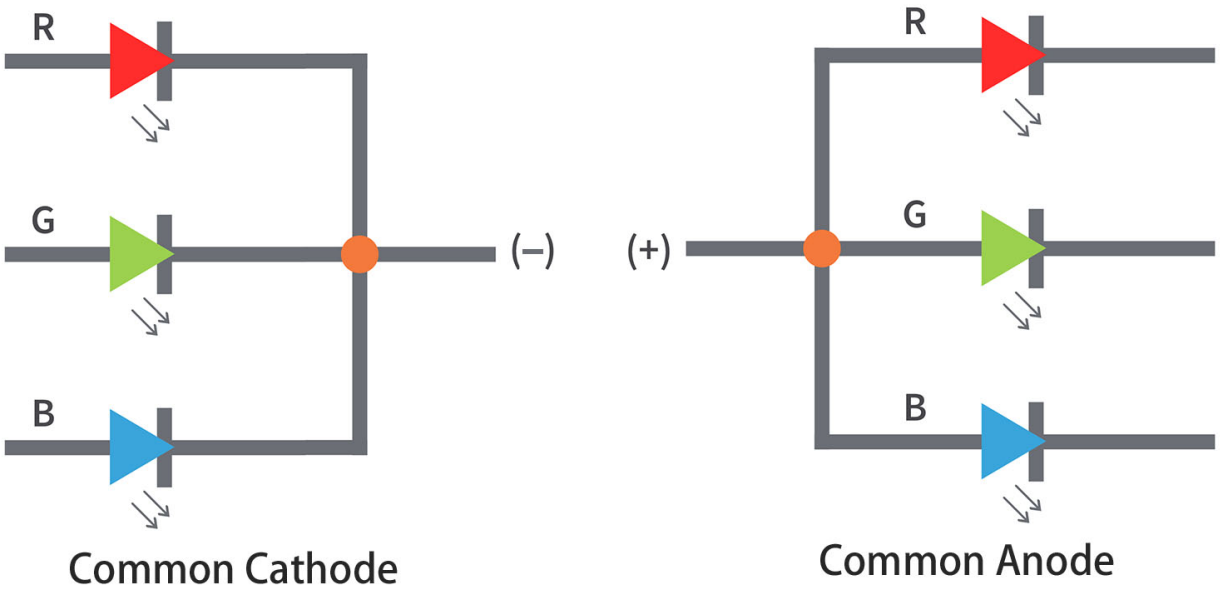
Características

- Color: Tricolor (Rojo/Verde/Azul)
- Cátodo Común
- Lente Redonda Clara de 5mm
- Voltaje Directo: Rojo: DC 2.0 - 2.2V; Azul y Verde: DC 3.0 - 3.2V (IF=20mA)
- LED RGB DIP de 0.06 Watts
- Luminancia Más Brillante Hasta +20 %
- Ángulo de Visión: 30°

Ánodo Común y Cátodo Común

Los LED RGB se pueden categorizar en LED con ánodo común y cátodo común.

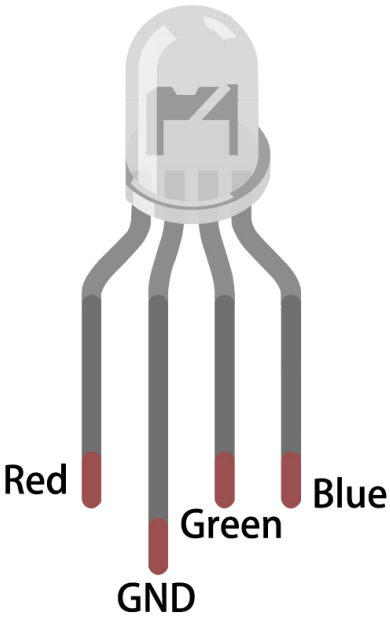
- En un LED RGB con cátodo común, los tres LED comparten una conexión negativa (cátodo).
- En un LED RGB con ánodo común, los tres LED comparten una conexión positiva (ánodo).



Nota: Utilizamos el LED RGB con cátodo común.

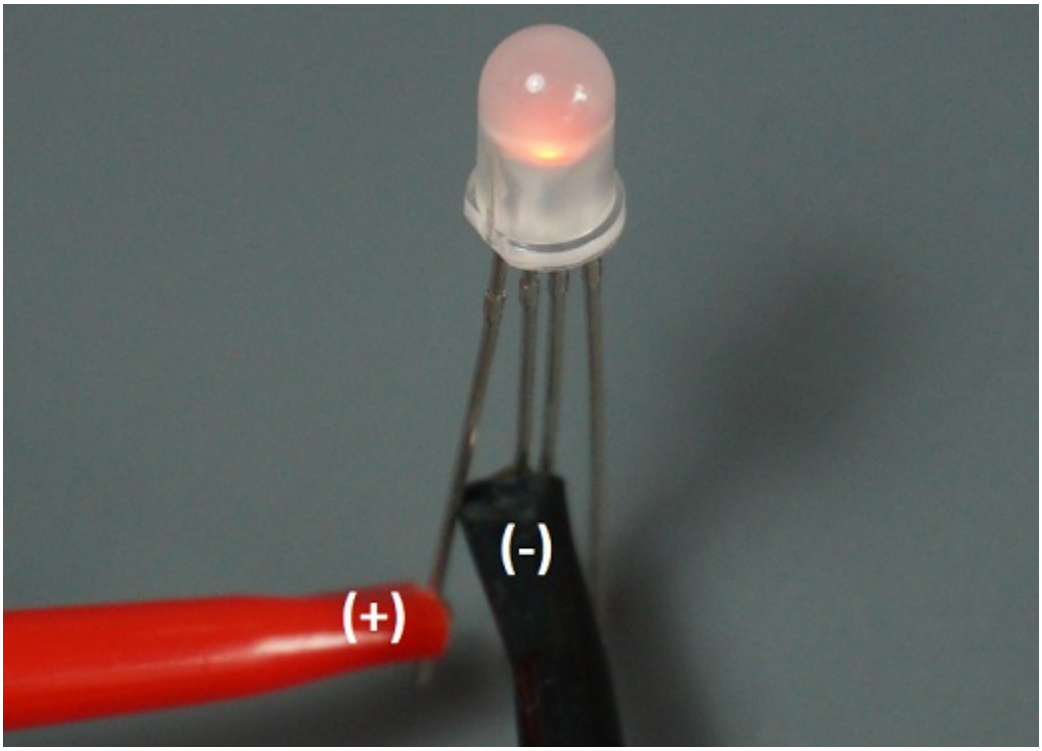
Pines del LED RGB

Un LED RGB tiene 4 pines: el más largo es GND; los otros son Rojo, Verde y Azul. Coloca los LED RGB como se muestra, de modo que el plomo más largo esté segundo desde la izquierda. Entonces los números de pin de los LED RGB deberían ser Rojo, GND, Verde y Azul.



También puedes usar el multímetro para seleccionar el modo de Prueba de Diodo, y luego conectar como se muestra a

continuación para medir el color de cada pin.

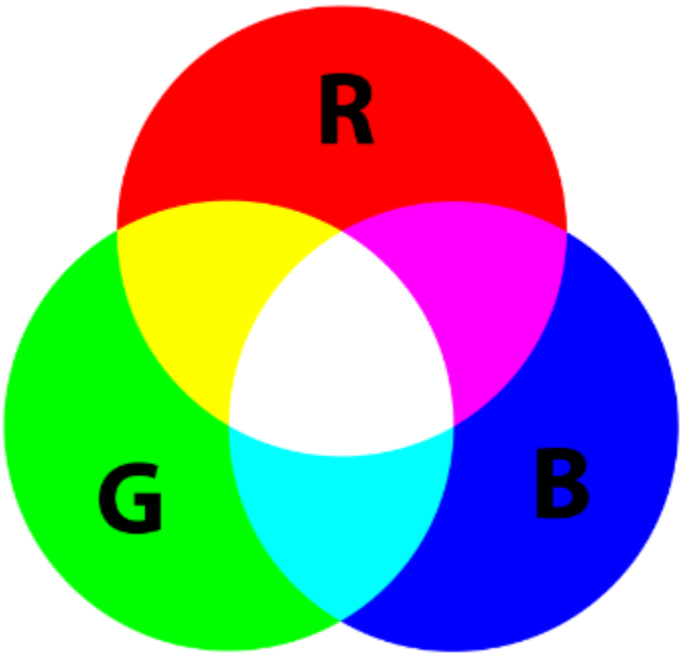


Mezcla de colores

Para generar colores adicionales, puedes combinar los tres colores a diferentes intensidades. Para ajustar la intensidad de cada LED, puedes usar una señal PWM.

Debido a que los LED están tan cerca entre sí, nuestros ojos ven el resultado de la combinación de colores en lugar de los tres colores individualmente.

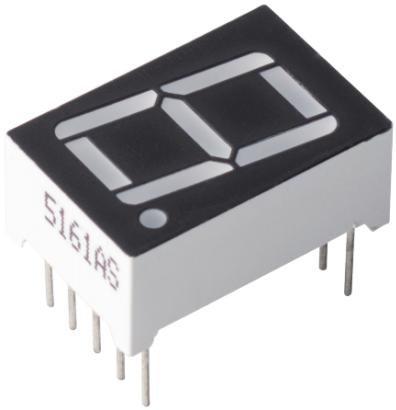
Consulta la tabla a continuación para ver cómo se combinan los colores. Te dará una idea de cómo funciona el gráfico de mezcla de colores y cómo se producen diferentes colores.



Ejemplo

- *2.3 Luz Colorida* (Proyecto Arduino)
- *6.5 Degradado de Color* (Proyecto Arduino)
- *2.3 Luz Colorida* (Proyecto MicroPython)
- *2.3 Bolas Coloridas* (Proyecto Scratch)

6.12 display de 7 Segmentos



Un display de 7 segmentos es un componente con forma de 8 que contiene 7 LED. Cada LED se llama segmento; cuando se energiza, un segmento forma parte de un numeral a ser mostrado.

- Cada uno de los LED en el display tiene un segmento posicional con uno de sus pines de conexión saliendo del paquete plástico rectangular.
- Estos pines LED están etiquetados de «a» a «g» representando cada LED individual.

- Los otros pines LED están conectados juntos formando un pin común.
- Así que al polarizar hacia adelante los pines apropiados de los segmentos LED en un orden particular, algunos segmentos se iluminarán y otros permanecerán tenues, mostrando así el carácter correspondiente en el display.

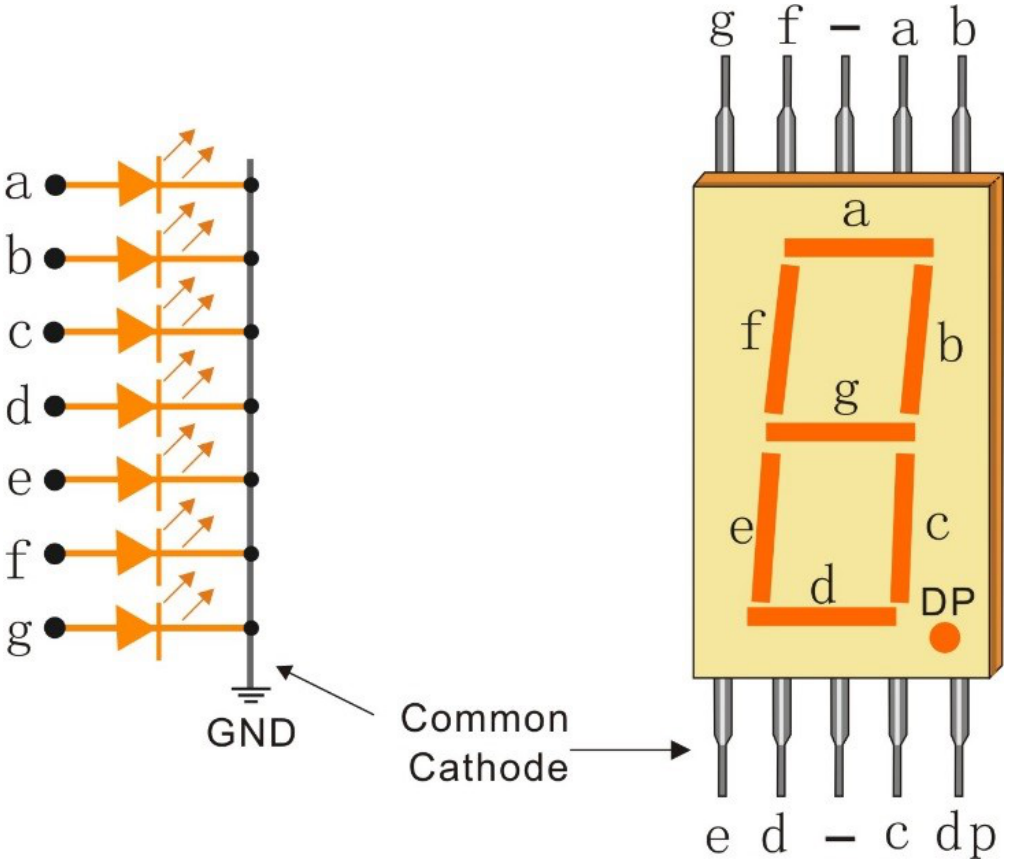
Características

- Tamaño: 19 x 12.7 x 13.8 mm (LxWxH, incluyendo el pin)
- Pantalla: 0.56”””
- Color: rojo
- Cátodo Común
- Voltaje Directo: 1.8V
- 10 pines
- Paso: estándar 0.1» (2.54mm)

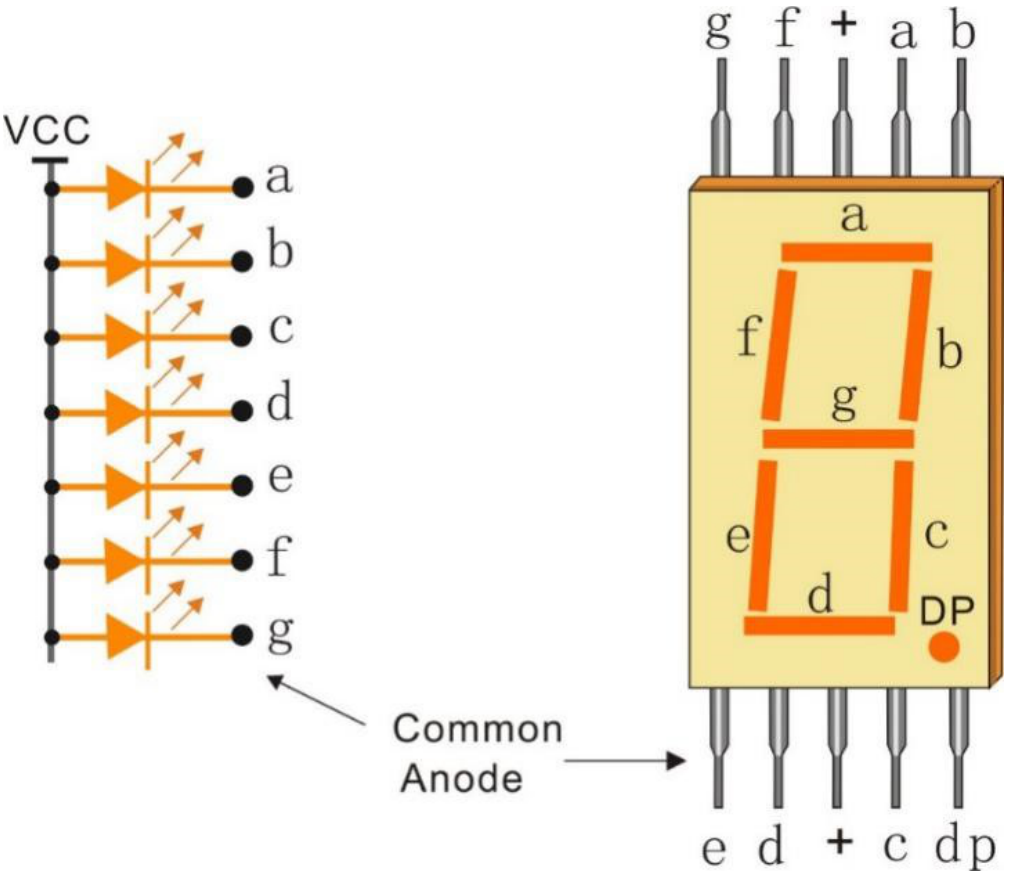
Cátodo Común (CC) o Ánodo Común (CA)

Existen dos tipos de conexión de pines: Cátodo Común (CC) y Ánodo Común (CA). Como su nombre indica, un display CC tiene todos los cátodos de los 7 LEDs conectados cuando un display CA tiene todos los ánodos de los 7 segmentos conectados.

- Display de 7 Segmentos con Cátodo Común

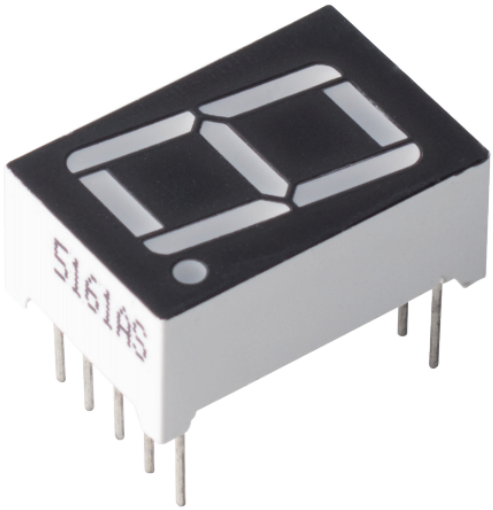


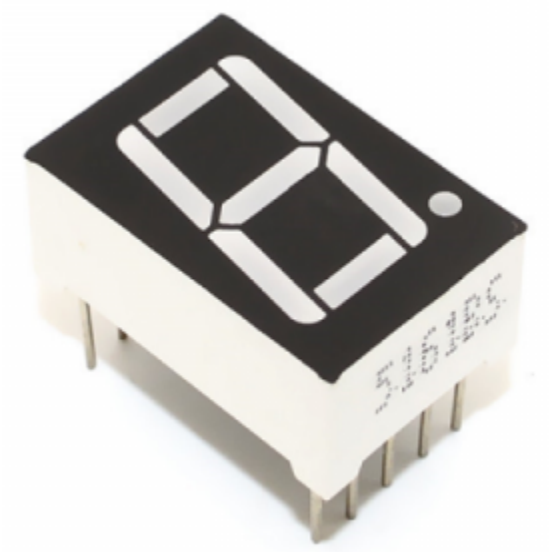
- Display de 7 Segmentos con Ánodo Común



Cómo Saber si es CC o CA?

Por lo general, habrá una etiqueta en el lateral del display de 7 segmentos, xxxAx o xxxBx. En general, xxxAx representa cátodo común y xxxBx representa ánodo común.





También puedes usar un multímetro para verificar el display de 7 segmentos si no hay etiqueta. Configura el multímetro en modo de prueba de diodo y conecta el cable negro al pin central del display de 7 segmentos, y el cable rojo a cualquier otro pin excepto el central. El display de 7 segmentos es de cátodo común si un segmento se ilumina.

Intercambia los cables rojo y negro si no hay ningún segmento iluminado. Cuando un segmento está iluminado, indica un ánodo común.

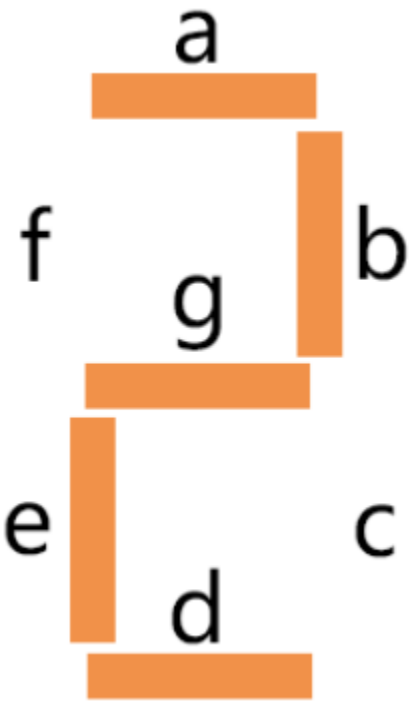
components/img/7_segment_test.JPG

Códigos de Visualización

Para ayudarte a entender cómo los displays de 7 segmentos (Cátodo Común) muestran números, hemos elaborado la siguiente tabla. Los números son el número 0-F mostrado en el display de 7 segmentos; (DP) GFEDCBA se refiere al conjunto LED correspondiente a 0 o 1.

Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00000110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00000111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

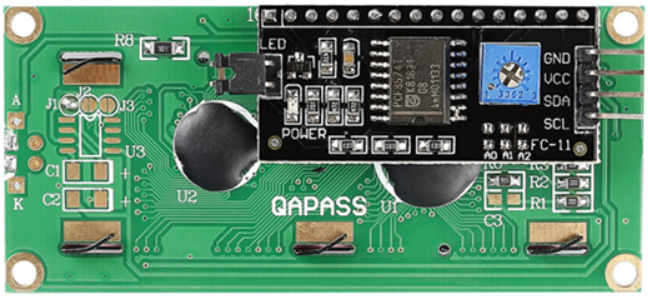
Por ejemplo, 01011011 significa que DP, F y C están configurados en 0, mientras que los otros están configurados en 1. Por lo tanto, el número 2 se muestra en el display de 7 segmentos.



Ejemplo

- *2.5 Pantalla de 7 Segmentos* (Proyecto Arduino)
- *6.4 Dados Digitales* (Proyecto Arduino)
- *2.5 Visualización de Números* (Proyecto MicroPython)
- *6.6 Dado Digital* (Proyecto MicroPython)

6.13 I2C LCD1602



- **GND**: Tierra
- **VCC**: Suministro de voltaje, 5V.
- **SDA**: Línea de datos serial. Conectar a VCC a través de una resistencia pull-up.
- **SCL**: Línea de reloj serial. Conectar a VCC a través de una resistencia pull-up.

Como todos sabemos, aunque las pantallas LCD y algunas otras pantallas enriquecen enormemente la interacción hombre-máquina, comparten una debilidad común. Cuando están conectadas a un controlador, múltiples E/S serán ocupadas del controlador que no tiene tantos puertos externos. También restringe otras funciones del controlador.

Por lo tanto, se desarrolla el LCD1602 con un módulo I2C para resolver el problema. El módulo I2C tiene un chip PCF8574 I2C incorporado que convierte los datos seriales I2C en datos paralelos para la pantalla LCD.

- Hoja de datos de PCF8574

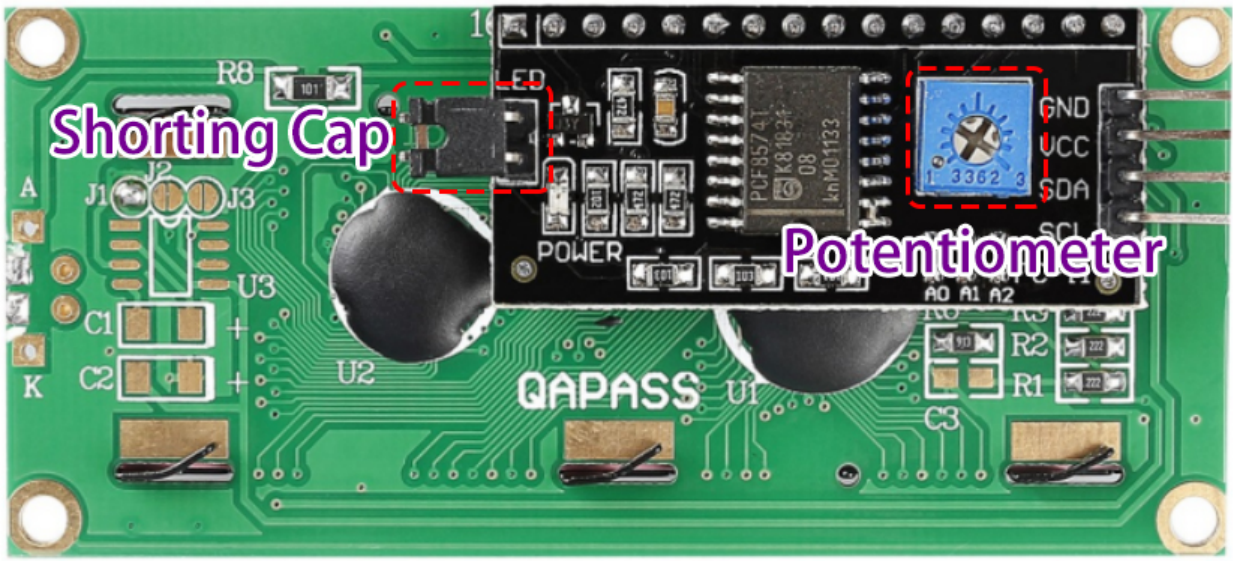
Dirección I2C

La dirección predeterminada es básicamente 0x27, en algunos casos puede ser 0x3F.

Tomando la dirección predeterminada de 0x27 como ejemplo, la dirección del dispositivo se puede modificar cortocircuitando las almohadillas A0/A1/A2; en el estado predeterminado, A0/A1/A2 es 1, y si la almohadilla está cortocircuitada, A0/A1/A2 es 0.

Retroiluminación/Contraste

La retroiluminación se puede habilitar mediante un puente, desconecte el puente para deshabilitar la retroiluminación. El potenciómetro azul en la parte posterior se utiliza para ajustar el contraste (la relación de brillo entre el blanco más brillante y el negro más oscuro).

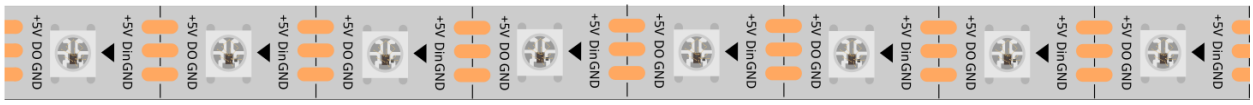


- **Puente Cortocircuitado**: La retroiluminación se puede habilitar mediante este puente, desenchufando este puente se deshabilita la retroiluminación.
- **Potenciómetro**: Se utiliza para ajustar el contraste (la claridad del texto mostrado), que se aumenta en dirección de las agujas del reloj y se disminuye en dirección contraria a las agujas del reloj.

Ejemplo

- *2.6 Mostrar Caracteres* (Proyecto Arduino)
- *6.7 Adivina el Número* (Proyecto Arduino)
- *2.6 Mostrar Caracteres* (Proyecto MicroPython)
- *6.7 Adivina el Número* (Proyecto MicroPython)

6.14 Tira de 8 LEDs RGB WS2812



La tira de 8 LEDs RGB WS2812 está compuesta por 8 LEDs RGB. Solo se requiere un pin para controlar todos los LEDs. Cada LED RGB tiene un chip WS2812, que puede ser controlado de manera independiente. Puede realizar una visualización de brillo de 256 niveles y una visualización de color real de 16,777,216 colores. Al mismo tiempo, el píxel contiene una interfaz digital inteligente de datos de interfaz de bloqueo de señal de amplificador de conformación de señal, y un circuito de conformación de señal está integrado para garantizar efectivamente la altura del color del píxel de luz de punto consistente.

Es flexible, se puede conectar, doblar y cortar a voluntad, y la parte posterior está equipada con cinta adhesiva, que se puede fijar en la superficie irregular a voluntad y se puede instalar en un espacio estrecho.

Características

- Voltaje de trabajo: DC5V
- IC: Un IC controla un LED RGB
- Consumo: 0.3w por cada LED

- Temperatura de trabajo: -15-50
- Color: RGB a todo color
- Tipo de RGB: 5050RGB (IC incorporado WS2812B)
- Grosor de la tira de luz: 2mm
- Cada LED se puede controlar individualmente

Introducción WS2812B

- Hoja de datos WS2812B

WS2812B es una fuente de luz LED de control inteligente en la que el circuito de control y el chip RGB están integrados en un paquete de componentes 5050. Internamente incluye una traba de datos de puerto digital inteligente y un circuito de amplificación de conformación de señal. También incluye un oscilador interno de precisión y una parte de control de corriente constante programable de 12V, lo que garantiza efectivamente la consistencia del color de la luz del punto del píxel.

El protocolo de transferencia de datos utiliza un modo de comunicación NZR único. Después del reinicio de alimentación del píxel, el puerto DIN recibe datos del controlador, el primer píxel recoge datos iniciales de 24 bits y luego los envía al latch de datos interno, los otros datos que son reformados por el circuito de amplificación de conformación de señal interno se envían al siguiente píxel en cascada a través del puerto DO. Después de la transmisión para cada píxel, la señal se reduce en 24 bits. El píxel adopta una tecnología de transmisión de reestructuración automática, lo que hace que el número de cascadas de píxeles no esté limitado a la transmisión de señales, solo depende de la velocidad de transmisión de la señal.

LED con bajo voltaje de conducción, protección del medio ambiente y ahorro de energía, alta luminosidad, ángulo de dispersión grande, buena consistencia, bajo consumo de energía, larga vida útil y otras ventajas. El chip de control integrado en el LED por encima se convierte en un circuito más simple, de pequeño volumen y de instalación conveniente.

Ejemplo

- *2.7 Tira de LEDs RGB* (Proyecto Arduino)
- *6.2 Luz Fluyente* (Proyecto Arduino)
- *2.7 Tira de LED RGB* (Proyecto MicroPython)
- *6.2 Luz Fluyente* (Proyecto MicroPython)
- *6.5 Degradado de Color* (Proyecto MicroPython)

Sonido

6.15 Zumbador

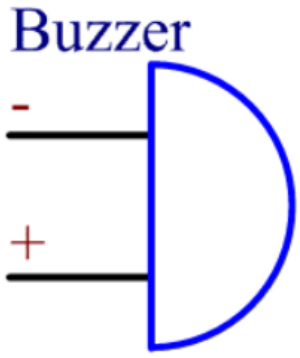
Como un tipo de zumbador electrónico con una estructura integrada, los zumbadores, que son alimentados por corriente continua, son ampliamente utilizados en computadoras, impresoras, fotocopiadoras, alarmas, juguetes electrónicos, dispositivos electrónicos automotrices, teléfonos, temporizadores y otros productos electrónicos o dispositivos de voz.

Los zumbadores pueden clasificarse como activos y pasivos (ver la siguiente imagen). Gira el zumbador para que sus pines estén hacia arriba, y el zumbador con una placa de circuito verde es un zumbador pasivo, mientras que el que está envuelto con una cinta negra es un zumbador activo.

La diferencia entre un zumbador activo y un zumbador pasivo:

Un zumbador activo tiene una fuente de oscilación incorporada, por lo que emitirá sonidos cuando esté electrificado. Pero un zumbador pasivo no tiene tal fuente, por lo que no emitirá pitidos si se usan señales de corriente continua; en su lugar, necesitas usar ondas cuadradas cuya frecuencia esté entre 2K y 5K para conducirlo. El zumbador activo suele ser más caro que el pasivo debido a los múltiples circuitos oscilantes incorporados.

Lo siguiente es el símbolo eléctrico de un zumbador. Tiene dos pines con polos positivos y negativos. Con un + en la superficie representa el ánodo y el otro es el cátodo.



Puedes verificar los pines del zumbador, el más largo es el ánodo y el más corto es el cátodo. Por favor, no los mezcles al conectarlos, de lo contrario, el zumbador no emitirá sonido.

Zumbador - Wikipedia

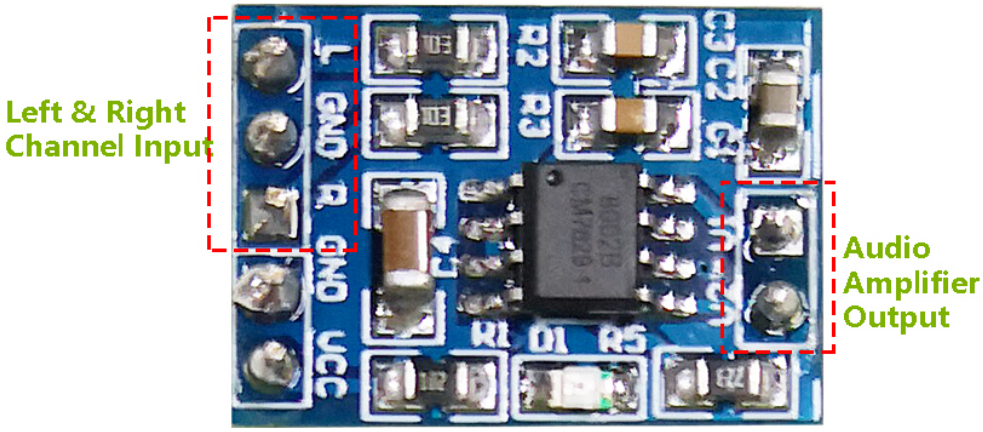
Ejemplo

- *3.1 Beep* (Proyecto Arduino)
- *3.2 Tono Personalizado* (Proyecto Arduino)
- *6.3 Ayuda para Reversa* (Proyecto Arduino)

- *3.2 Tono Personalizado* (Proyecto MicroPython)
- *3.1 Pitido* (Proyecto MicroPython)
- *6.4 Asistente de Reversa* (Proyecto MicroPython)

6.16 Módulo de Audio y Altavoz

Módulo Amplificador de Audio

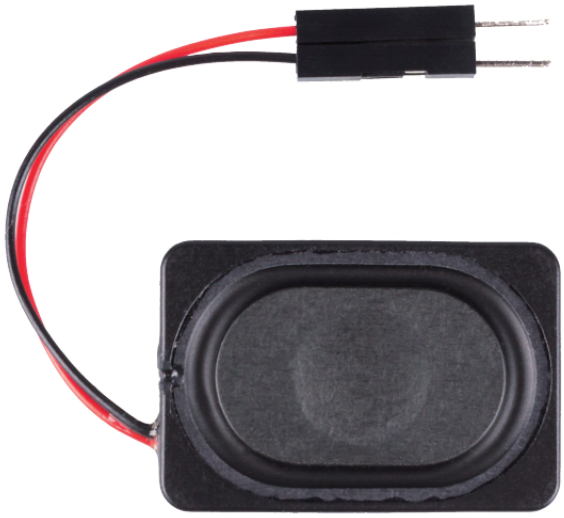


El Módulo Amplificador de Audio contiene un chip amplificador de potencia HXJ8002. Este chip es un amplificador de potencia con bajo consumo, que puede proporcionar una potencia de audio promedio de 3W para una carga BTL de 3Ω con baja distorsión armónica (por debajo del 10 % de distorsión umbral a 1 kHz) desde una fuente de alimentación de 5V DC. Este chip puede amplificar señales de audio sin ningún condensador de acoplamiento o condensadores bootstrap.

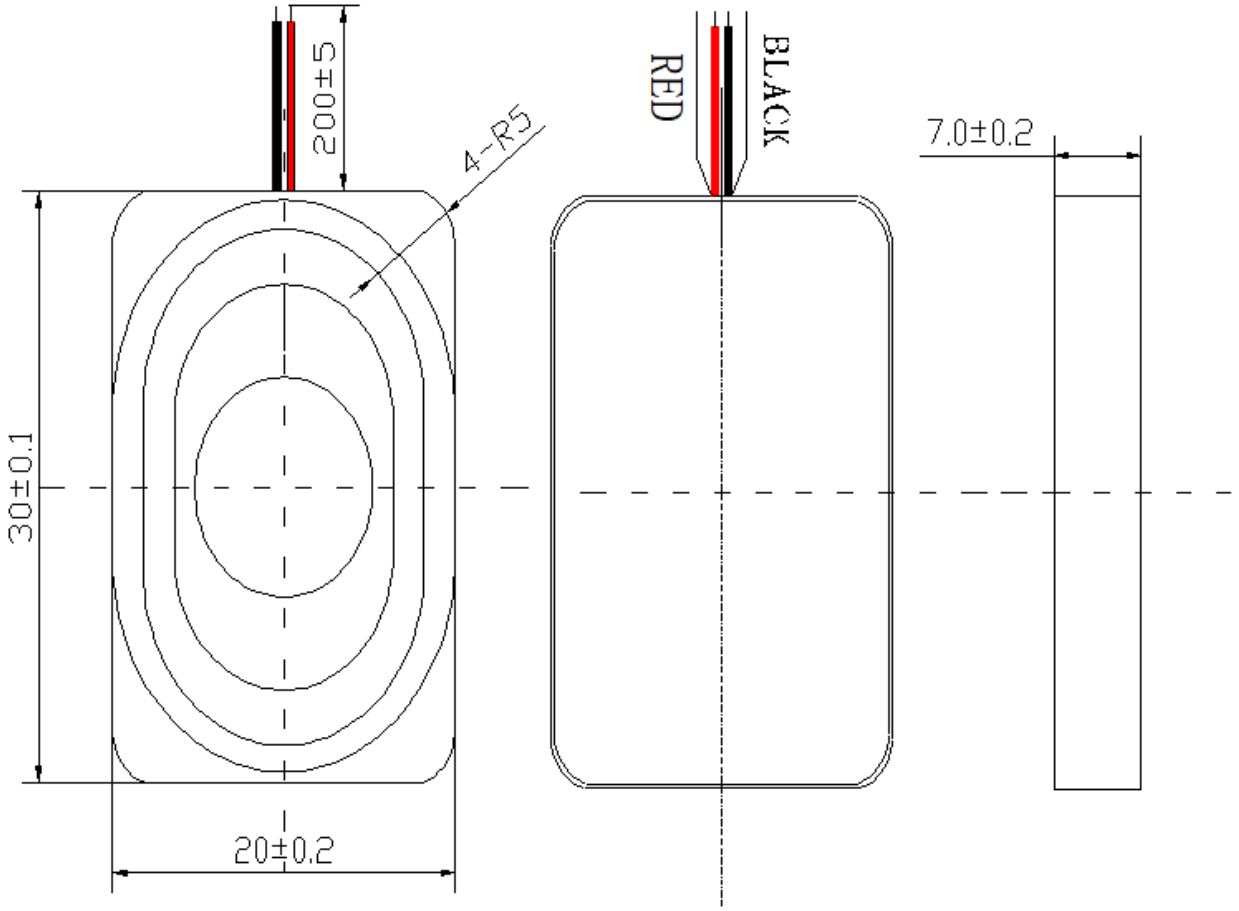
El módulo puede ser alimentado por una fuente de alimentación de 2.0V hasta 5.5V DC con una corriente de funcionamiento de 10mA (0.6uA para la corriente de espera típica) y producir un sonido amplificado potente en un altavoz de impedancia de 3, 4 o 8. Este módulo tiene una circuitería mejorada de pop y clicks para reducir significativamente el ruido de transición en el momento del encendido y apagado. Su tamaño reducido además de su alta eficiencia y baja alimentación lo hacen aplicable en una amplia gama de proyectos portátiles y alimentados por batería y microcontroladores.

- **IC**: HXJ8002
- **Voltaje de Entrada**: 2V ~ 5.5V
- **Corriente en Modo de Espera**: 0.6uA (valor típico)
- **Potencia de Salida**: 3W (carga de 3Ω), 2.5W (carga de 4Ω), 1.5W (carga de 8Ω)
- **Impedancia del Altavoz de Salida**: 3Ω, 4Ω, 8Ω
- **Tamaño**: 19.8mm x 14.2mm

Altavoz



- **Tamaño**: 20x30x7mm
- **Impedancia**: 8ohm
- **Potencia de Entrada Nominal**: 1.5W
- **Potencia de Entrada Máxima**: 2.0W
- **Longitud del Cable**: 10cm



La tabla de tamaños es la siguiente:

- `Hoja de Datos del Altavoz 2030`

Ejemplo

- *7.5 Reproductor MP3 con Soporte de Tarjeta SD* (Proyecto Arduino)
- *7.3 Reproductor de Audio Bluetooth* (Proyecto Arduino)

Driver

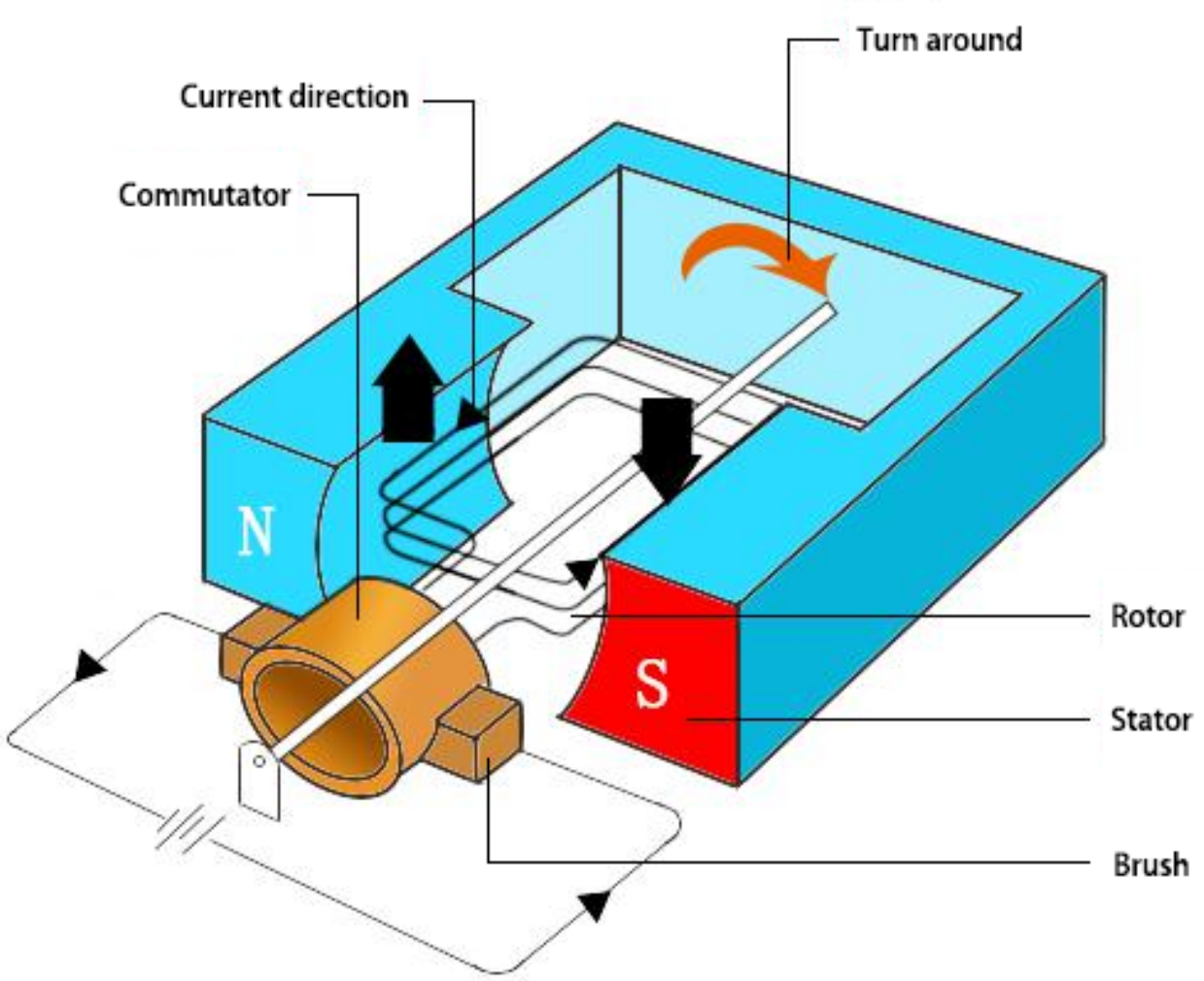
6.17 Motor de Corriente Continua (DC)

Este es un motor de corriente continua (DC) de 3V. Cuando aplicas un nivel alto y un nivel bajo a cada uno de los 2 terminales, girará.

- **Longitud**: 25mm
- **Diámetro**: 21mm
- **Diámetro del Eje**: 2mm
- **Longitud del Eje**: 8mm
- **Voltaje**: 3-6V
- **Corriente**: 0.35-0.4A
- **Velocidad a 3V**: 19000 RPM (Rotaciones Por Minuto)
- **Peso**: Aproximadamente 14g (por unidad)

El motor de corriente continua (DC) es un actuador continuo que convierte la energía eléctrica en energía mecánica. Los motores de corriente continua hacen que funcionen bombas rotativas, ventiladores, compresores, impulsores y otros dispositivos al producir una rotación angular continua.

Un motor de corriente continua consta de dos partes, la parte fija del motor llamada el **estator** y la parte interna del motor llamada el **rotor** (o **inducido** de un motor de corriente continua) que gira para producir movimiento. La clave para generar movimiento es posicionar el inducido dentro del campo magnético del imán permanente (cuyo campo se extiende desde el polo norte hasta el polo sur). La interacción del campo magnético y las partículas cargadas en movimiento (el alambre conductor de corriente genera el campo magnético) produce el par de torsión que hace girar el inducido.



La corriente fluye desde el terminal positivo de la batería a través del circuito, pasando por los cepillos de cobre hasta el conmutador, y luego al inducido. Pero debido a las dos brechas en el conmutador, este flujo se invierte a mitad de cada rotación completa.

Esta inversión continua básicamente convierte la energía eléctrica de corriente continua de la batería en corriente alterna, permitiendo que el inducido experimente un par de torsión en la dirección correcta en el momento adecuado para mantener la rotación.

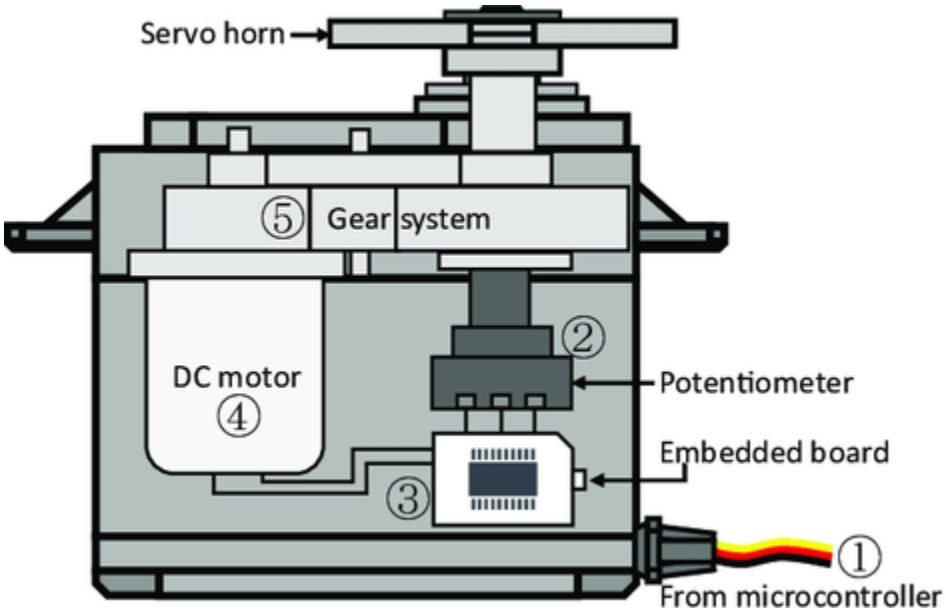
Ejemplo

- *4.1 Motor* (Proyecto Arduino)
- *4.1 Pequeño Ventilador* (Proyecto MicroPython)
- *2.9 Ventilador Rotativo* (Proyecto Scratch)

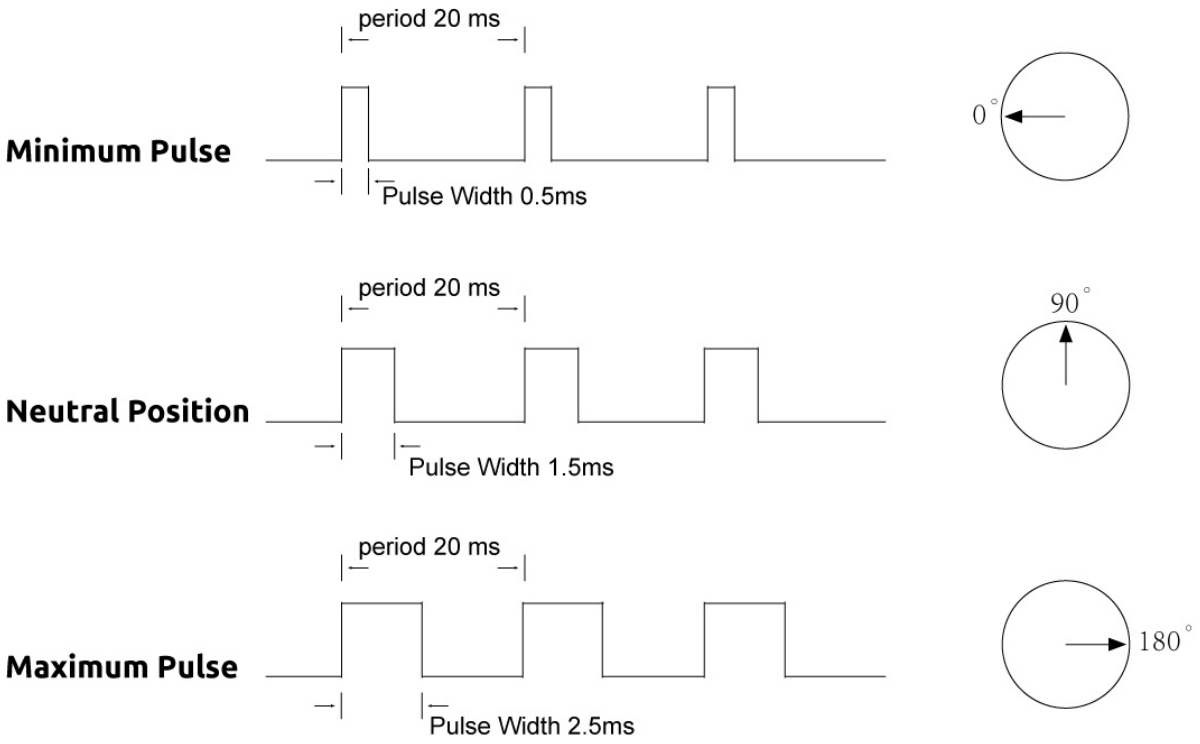
6.18 Servo

Un servo se compone generalmente de las siguientes partes: carcasa, eje, sistema de engranajes, potenciómetro, motor de corriente continua y placa embebida.

Funciona de la siguiente manera: el microcontrolador envía señales PWM al servo, y luego la placa embebida en el servo recibe las señales a través del pin de señal y controla el motor en su interior para girar. Como resultado, el motor impulsa el sistema de engranajes y luego motiva el eje después de la desaceleración. El eje y el potenciómetro del servo están conectados entre sí. Cuando el eje gira, impulsa el potenciómetro, por lo que el potenciómetro emite una señal de voltaje a la placa embebida. Luego, la placa determina la dirección y la velocidad de rotación según la posición actual, por lo que puede detenerse exactamente en la posición correcta definida y mantenerse allí.



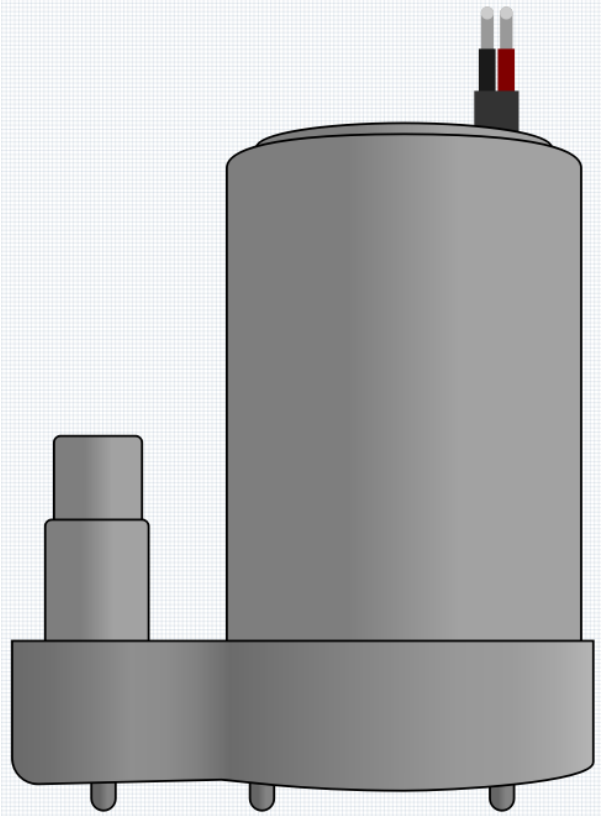
El ángulo está determinado por la duración de un pulso que se aplica al cable de control. Esto se llama Modulación por Ancho de Pulso (PWM, por sus siglas en inglés). El servo espera ver un pulso cada 20 ms. La longitud del pulso determinará cuánto gira el motor. Por ejemplo, un pulso de 1.5 ms hará que el motor gire hasta la posición de 90 grados (posición neutral). Cuando se envía un pulso a un servo que es menor que 1.5 ms, el servo gira a una posición y mantiene su eje de salida un cierto número de grados en sentido antihorario desde el punto neutral. Cuando el pulso es más ancho que 1.5 ms, ocurre lo contrario. El ancho mínimo y el ancho máximo del pulso que ordenará al servo girar a una posición válida son funciones de cada servo. Generalmente, el pulso mínimo tendrá aproximadamente 0.5 ms de ancho y el pulso máximo tendrá 2.5 ms de ancho.



Ejemplo

- *4.3 Balanceo del Servo* (Proyecto Arduino)
- *4.3 Servo Oscilante* (Proyecto MicroPython)

6.19 Bomba Centrífuga



La bomba centrífuga convierte la energía cinética rotacional en energía hidrodinámica para transportar fluidos. La energía de rotación proviene del motor eléctrico. El fluido entra en el impulsor de la bomba a lo largo o cerca del eje rotativo, es acelerado por el impulsor, fluye radialmente hacia afuera en el difusor o cámara voluta, y luego fluye desde allí.

Los usos comunes de las bombas centrífugas incluyen el bombeo de agua, aguas residuales, agrícolas, petroleras y petroquímicas.

- Bomba Centrífuga - Wikipedia

Características

- **Rango de Voltaje**: DC 3 ~ 4.5V
- **Corriente de Operación**: 120 ~ 180mA
- **Potencia**: 0.36 ~ 0.91W
- **Altura Máxima de Agua**: 0.35 ~ 0.55M
- **Tasa Máxima de Flujo**: 80 ~ 100 L/H
- **Vida Laboral Continua**: 100 horas
- **Grado de Protección al Agua**: IP68
- **Modo de Conducción**: DC, Conducción Magnética
- **Material**: Plástico de Ingeniería
- **Diámetro Exterior de Salida**: 7.8 mm

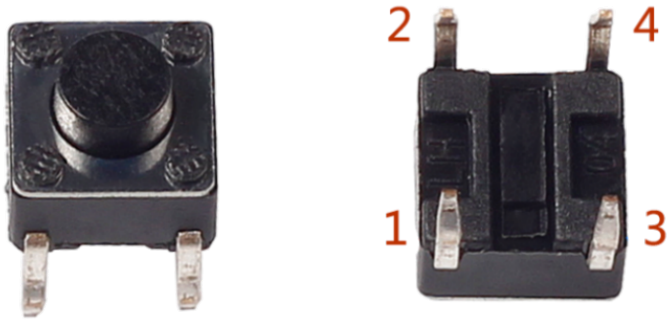
- **Diámetro Interior de Salida**: 6.5 mm
- Es una bomba sumergible y debe usarse de esa manera. Tiende a calentarse demasiado, por lo que existe el riesgo de sobrecalentamiento si se enciende sin estar sumergida.

Ejemplo

- *4.2 Bombeo* (Proyecto Arduino)
- *6.6 Monitor de Plantas* (Proyecto Arduino)
- *4.2 Bombeo* (Proyecto MicroPython)
- *6.8 Monitor de Plantas* (Proyecto MicroPython)

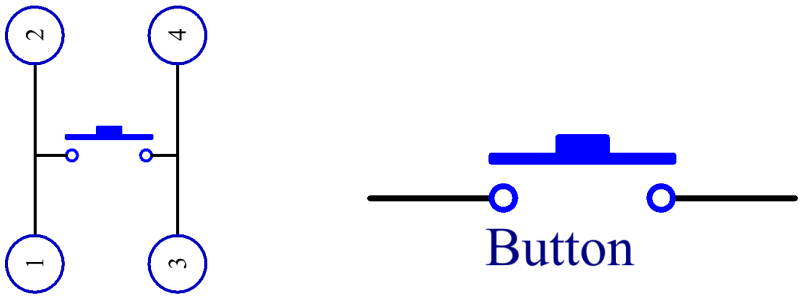
Controlador

6.20 Botón

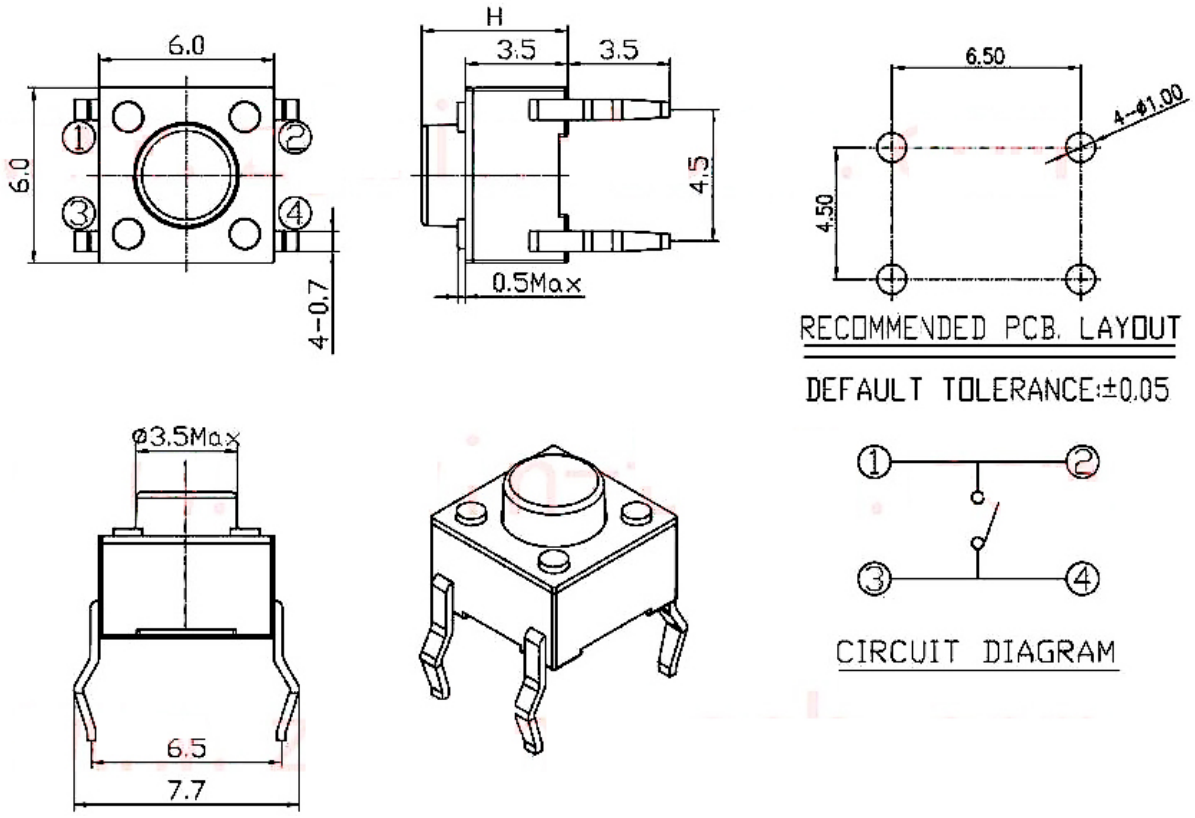


Los botones son componentes comunes utilizados para controlar dispositivos electrónicos. Usualmente se utilizan como interruptores para conectar o interrumpir circuitos. Aunque los botones vienen en una variedad de tamaños y formas, el que se usa aquí es un mini botón de 6 mm como se muestra en las siguientes imágenes. El pin 1 está conectado al pin 2 y el pin 3 al pin 4. Por lo tanto, solo necesitas conectar cualquiera de los pines 1 y 2 al pin 3 o al pin 4.

La siguiente es la estructura interna de un botón. El símbolo en la parte inferior derecha se utiliza generalmente para representar un botón en los circuitos.



Dado que el pin 1 está conectado al pin 2, y el pin 3 al pin 4, cuando se presiona el botón, los 4 pines están conectados, cerrando así el circuito.



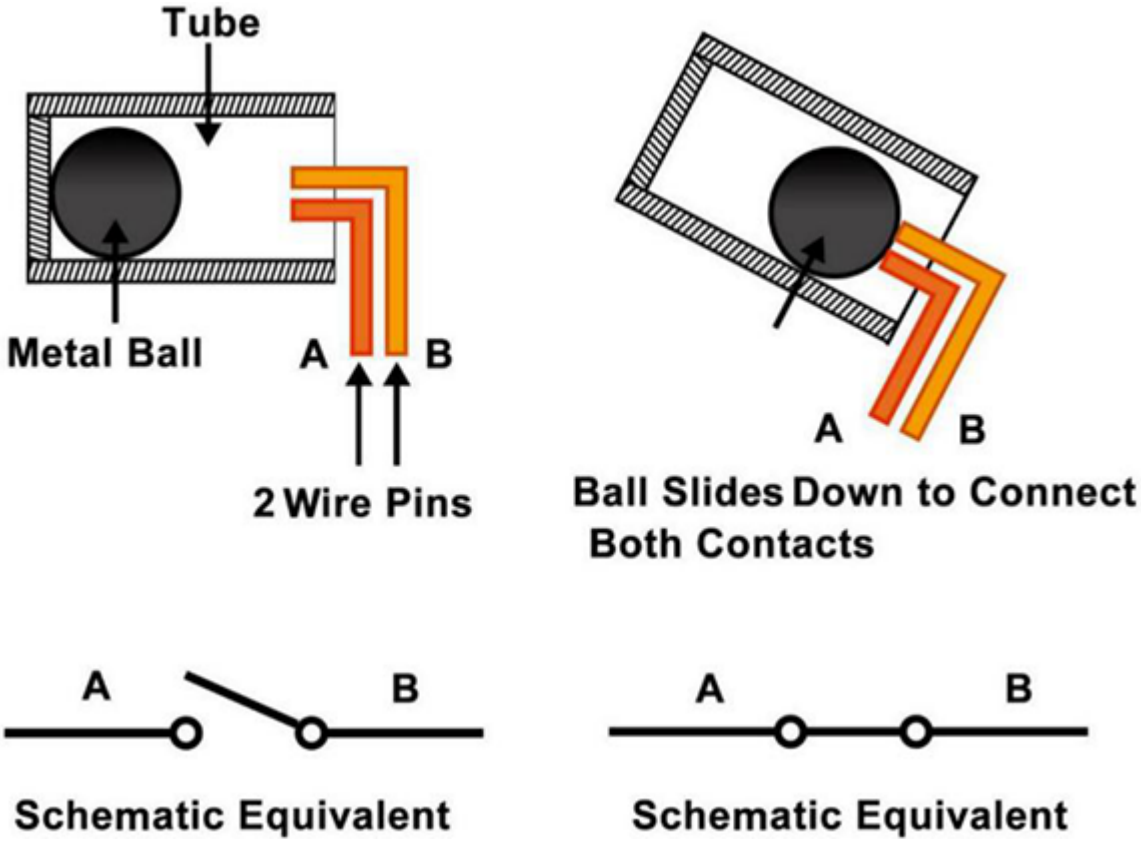
Ejemplo

- *5.1 Lectura del Valor del Botón* (Proyecto Arduino)
- *5.1 Lectura del Valor del Botón* (Proyecto MicroPython)
- *2.5 Timbre* (Proyecto Scratch)
- *2.14 JUEGO - Comer Manzana* (Proyecto Scratch)
- *2.17 JUEGO - Pesca* (Proyecto Scratch)

6.21 Interruptor de Inclinación

El interruptor de inclinación utilizado aquí es una bola con un interruptor de contacto en su interior. Se utiliza para detectar inclinaciones de ángulos pequeños.

El principio es muy simple. Cuando el interruptor se inclina en un cierto ángulo, la bola en su interior rueda hacia abajo y toca los dos contactos conectados a los pines exteriores, activando así los circuitos. De lo contrario, la bola se mantendrá alejada de los contactos, interrumpiendo así los circuitos.

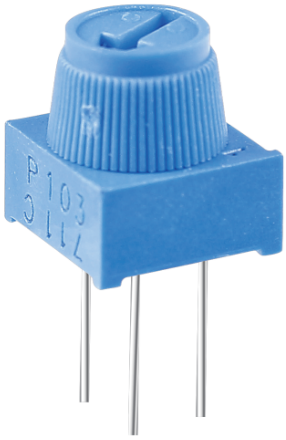


- Hoja de datos del interruptor de inclinación SW520D

Ejemplo

- *5.2 ¡Inclínalo!* (Proyecto Arduino)
- *5.2 ¡Inclínalo!* (Proyecto MicroPython)

6.22 Potenciómetro

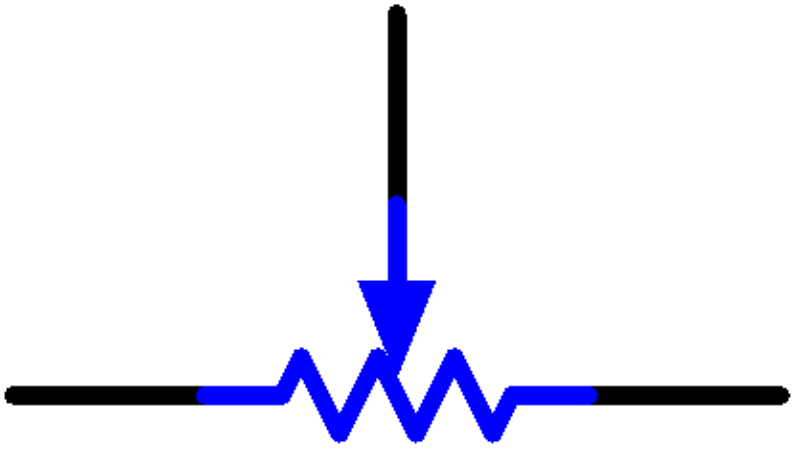


El potenciómetro es también un componente de resistencia con 3 terminales y su valor de resistencia puede ajustarse según alguna variación regular.

Los potenciómetros vienen en varias formas, tamaños y valores, pero todos tienen las siguientes características en común:

- Tienen tres terminales (o puntos de conexión).
- Tienen un botón, tornillo o deslizador que puede moverse para variar la resistencia entre el terminal central y cualquiera de los terminales exteriores.
- La resistencia entre el terminal central y cualquiera de los terminales exteriores varía desde 0 hasta la resistencia máxima del potenciómetro a medida que se mueve el botón, tornillo o deslizador.

Aquí está el símbolo de circuito del potenciómetro.



Las funciones del potenciómetro en el circuito son las siguientes:

1. Servir como divisor de voltaje

   El potenciómetro es un resistor ajustable continuamente. Cuando ajustas el eje o la manija deslizante del potenciómetro, el contacto móvil se deslizará sobre el resistor. En este punto, se puede obtener un voltaje de salida dependiendo del voltaje aplicado al potenciómetro y el ángulo que el brazo móvil ha rotado o el recorrido que ha realizado.

2. Servir como reóstato

   Cuando el potenciómetro se utiliza como un reóstato, conecta el pin central y uno de los otros 2 pines en el circuito. Así puedes obtener un valor de resistencia cambiado suavemente y continuamente dentro del recorrido del contacto móvil.

3. Servir como controlador de corriente

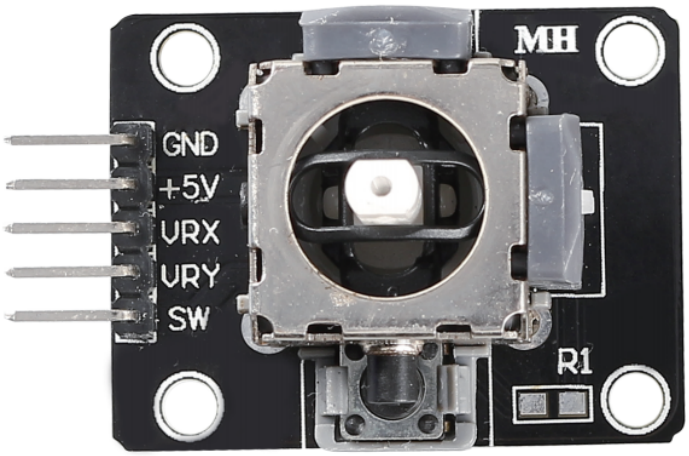
   Cuando el potenciómetro actúa como un controlador de corriente, el terminal de contacto deslizante debe conectarse como uno de los terminales de salida.

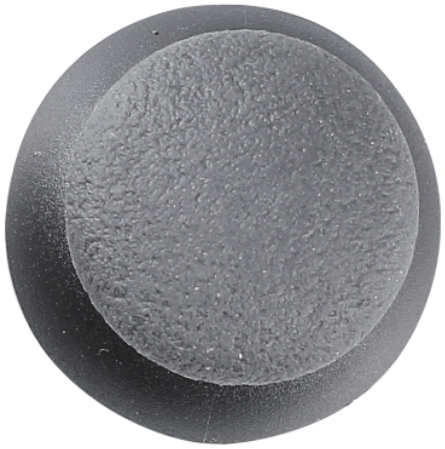
Si deseas saber más sobre el potenciómetro, consulta: Potenciómetro - Wikipedia

Ejemplo

- *5.8 Gira el Potenciómetro* (Proyecto Arduino)
- *5.8 Girar el Pomo* (Proyecto MicroPython)
- *2.4 Ratón en Movimiento* (Proyecto Scratch)
- *2.16 JUEGO - Clon de Breakout* (Proyecto Scratch)

6.23 Módulo de Joystick





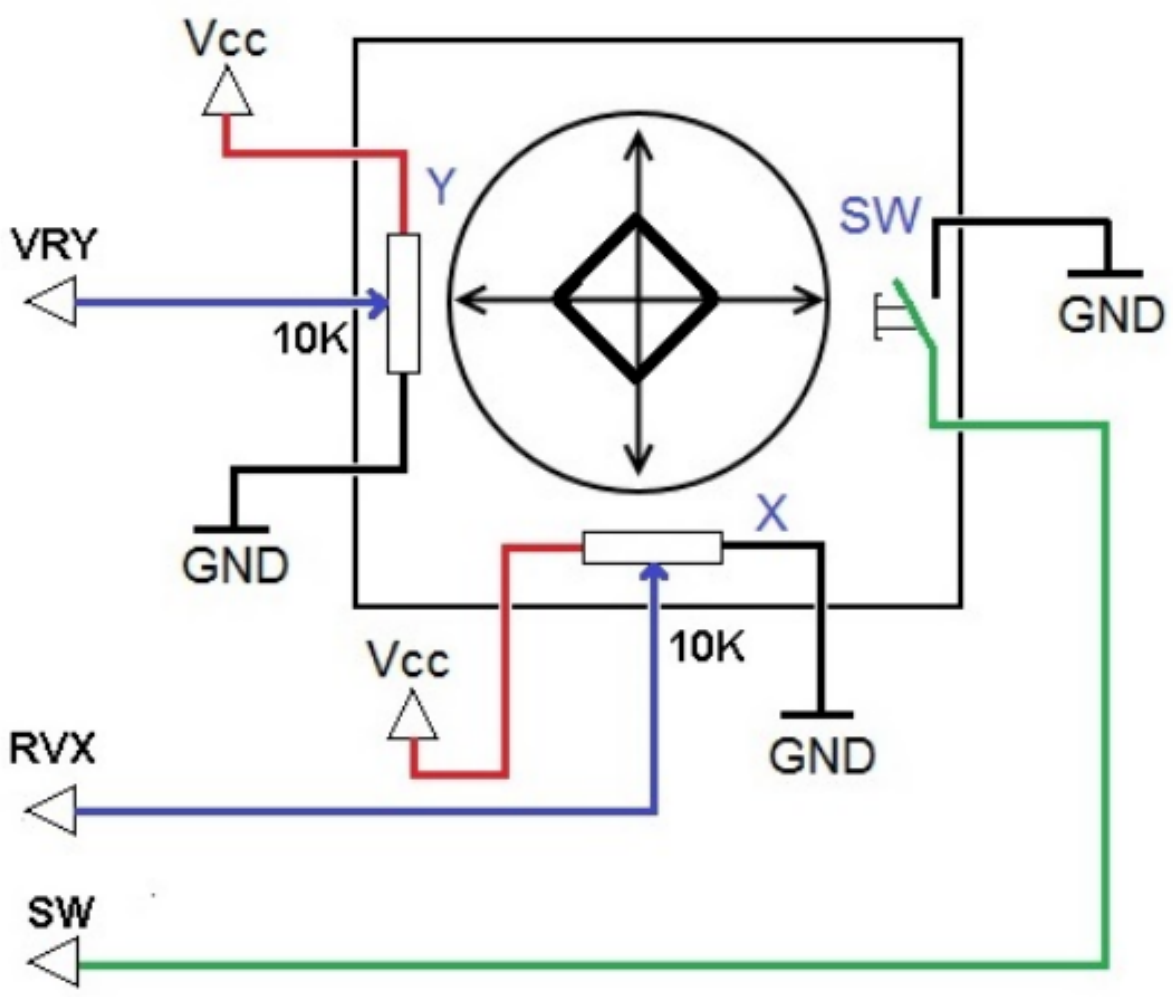
- **GND**: Tierra.
- **+5V**: Suministro de energía, acepta de 3.3V a 5V.
- **VRX**: Salida analógica correspondiente a la posición horizontal (eje X) del joystick.
- **VRY**: Salida analógica correspondiente a la posición vertical (eje Y) del joystick.
- **SW**: Salida del interruptor de botón, activado cuando se presiona el joystick hacia abajo. Para un funcionamiento adecuado, se requiere una resistencia pull-up externa. Con la resistencia en su lugar, el pin SW emite un nivel alto cuando está inactivo y se vuelve bajo cuando se presiona el joystick.

La idea básica de un joystick es traducir el movimiento de un palo en información electrónica que una computadora puede procesar.

Para comunicar un rango completo de movimiento a la computadora, un joystick necesita medir la posición del palo en dos ejes: el eje X (de izquierda a derecha) y el eje Y (arriba y abajo). Al igual que en la geometría básica, las coordenadas X-Y señalan exactamente la posición del palo.

Para determinar la ubicación del palo, el sistema de control del joystick simplemente monitorea la posición de cada eje. El diseño convencional de joystick analógico hace esto con dos potenciómetros, o resistencias variables.

El joystick también tiene una entrada digital que se activa cuando se presiona el joystick hacia abajo.



Ejemplo

- *5.11 Alternar el Joystick* (Proyecto Arduino)
- *5.11 Alternar el Joystick* (Proyecto MicroPython)
- *2.13 JUEGO - Estrellas Cruzadas* (Proyecto Scratch)
- *2.20 JUEGO - Matar al Dragón* (Proyecto Scratch)

6.24 Receptor de Infrarrojos

Receptor de Infrarrojos

- OUT: Salida de señal
- GND: Tierra
- VCC: Fuente de alimentación, 3.3v~5V

El receptor de infrarrojos SL838 es un componente que recibe señales infrarrojas y puede recibir independientemente rayos infrarrojos y emitir señales compatibles con el nivel TTL. Es similar a un transistor normal encapsulado en plástico en tamaño y es adecuado para todo tipo de control remoto infrarrojo y transmisión infrarroja.

La comunicación infrarroja, o IR, es una tecnología de comunicación inalámbrica popular, de bajo costo y fácil de usar. La luz infrarroja tiene una longitud de onda ligeramente más larga que la luz visible, por lo que es imperceptible para el ojo humano, lo que la hace ideal para la comunicación inalámbrica. Un esquema de modulación común para la comunicación infrarroja es la modulación de 38 kHz.

- Puede ser utilizado para control remoto
- Amplio rango de voltaje de funcionamiento: 2.7~5V
- Filtro interno para frecuencia PCM
- Compatibilidad TTL y CMOS
- Fuerte capacidad antiinterferencias
- Cumple con RoHS

Control Remoto

Este es un control remoto inalámbrico de infrarrojos delgado y miniatura con 21 botones de función y una distancia de transmisión de hasta 8 metros, que es adecuado para operar una amplia gama de dispositivos en la habitación de un niño.

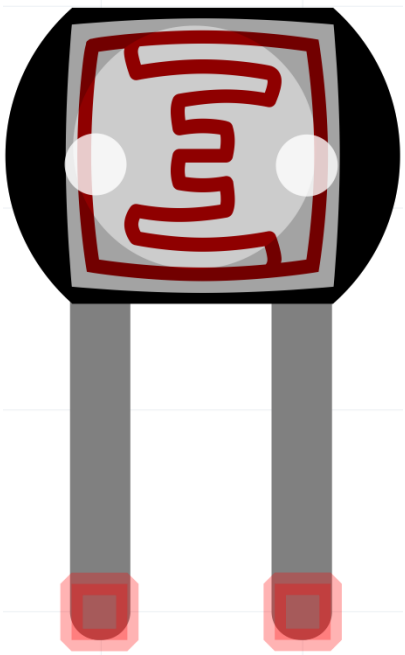
- Tamaño: 85x39x6mm
- Rango de control remoto: 8-10m
- Batería: Pila de litio de tipo botón de 3V
- Frecuencia portadora infrarroja: 38 kHz
- Material de adhesión superficial: PET de 0.125mm
- Vida útil efectiva: más de 20,000 veces

Ejemplo

- *5.14 Receptor IR* (Proyecto Arduino)
- *6.7 Adivina el Número* (Proyecto Arduino)
- *5.14 Control Remoto por Infrarrojos* (Proyecto MicroPython)
- *6.7 Adivina el Número* (Proyecto MicroPython)

Sensor

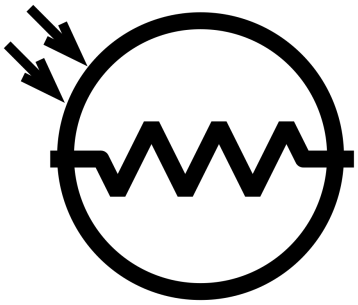
6.25 Fotorresistor



Un fotorresistor o célula fotoeléctrica es un resistor variable controlado por la luz. La resistencia de un fotorresistor disminuye con el aumento de la intensidad de luz incidente; en otras palabras, exhibe fotoconductividad.

Un fotorresistor puede aplicarse en circuitos detectores sensibles a la luz y en circuitos de conmutación activados por la luz y activados por la oscuridad actuando como un semiconductor de resistencia. En la oscuridad, un fotorresistor puede tener una resistencia de hasta varios megaohmios (M), mientras que a la luz, un fotorresistor puede tener una resistencia tan baja como unos pocos cientos de ohmios.

Aquí está el símbolo electrónico del fotorresistor.

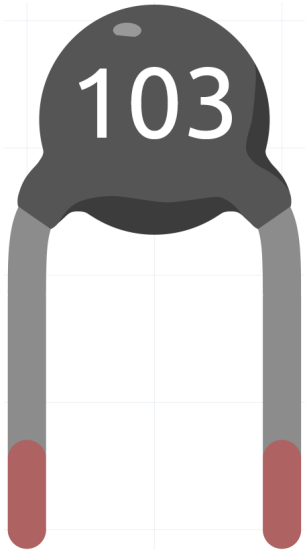


- Fotorresistor - Wikipedia

Ejemplo

- *5.7 Siente la Luz* (Proyecto Arduino)
- *6.6 Monitor de Plantas* (Proyecto Arduino)
- *5.7 Siente la Luz* (Proyecto MicroPython)
- *6.8 Monitor de Plantas* (Proyecto MicroPython)

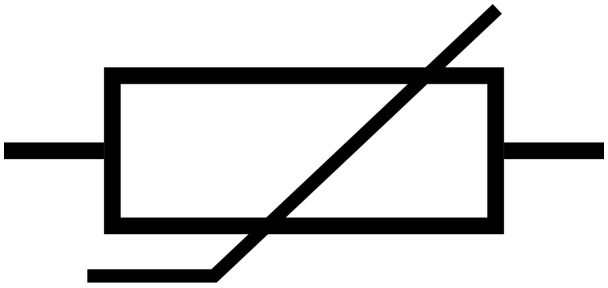
6.26 Termistor



Un termistor es un tipo de resistor cuya resistencia depende fuertemente de la temperatura, más que en los resistores estándar. La palabra es una combinación de térmico y resistor. Los termistores se utilizan ampliamente como limitadores de corriente de arranque, sensores de temperatura (generalmente de coeficiente de temperatura negativo o NTC), protectores de sobrecorriente autorreajustables y elementos calefactores autorregulados (generalmente de coeficiente de temperatura positivo o PTC).

- Termistor - Wikipedia

Aquí está el símbolo electrónico del termistor.



Los termistores son de dos tipos fundamentales opuestos:

- Con los termistores NTC, la resistencia disminuye a medida que aumenta la temperatura, generalmente debido a un aumento de los electrones de conducción estimulados por la agitación térmica desde la banda de valencia. Un NTC se utiliza comúnmente como sensor de temperatura o en serie con un circuito como limitador de corriente de arranque.
- Con los termistores PTC, la resistencia aumenta a medida que aumenta la temperatura, generalmente debido al aumento de las agitaciones térmicas de la red, especialmente las de impurezas e imperfecciones. Los termistores PTC se instalan comúnmente en serie con un circuito y se utilizan para proteger contra condiciones de sobrecorriente, como fusibles rearmables.

En este kit usamos uno NTC. Cada termistor tiene una resistencia normal. Aquí es de 10k ohmios, que se mide a 25 grados Celsius.

Aquí está la relación entre la resistencia y la temperatura:

RT = RN * expB(1/TK - 1/TN)

- **RT** es la resistencia del termistor NTC cuando la temperatura es TK.
- **RN** es la resistencia del termistor NTC bajo la temperatura nominal TN. Aquí, el valor numérico de RN es 10k.
- **TK** es una temperatura en Kelvin y la unidad es K. Aquí, el valor numérico de TK es 273.15 + grados Celsius.
- **TN** es una temperatura nominal en Kelvin; la unidad es también K. Aquí, el valor numérico de TN es 273.15+25.
- Y **B (beta)**, la constante de material del termistor NTC, también se llama índice de sensibilidad al calor con un valor numérico de 3950.
- **exp** es la abreviatura de exponencial, y el número base e es un número natural y es aproximadamente igual a 2.7.

Convertir esta fórmula TK=1/(ln(RT/RN)/B+1/TN) para obtener la temperatura en Kelvin que menos 273.15 es igual a grados Celsius.

Esta relación es una fórmula empírica. Es precisa solo cuando la temperatura y la resistencia están dentro del rango efectivo.

Ejemplo

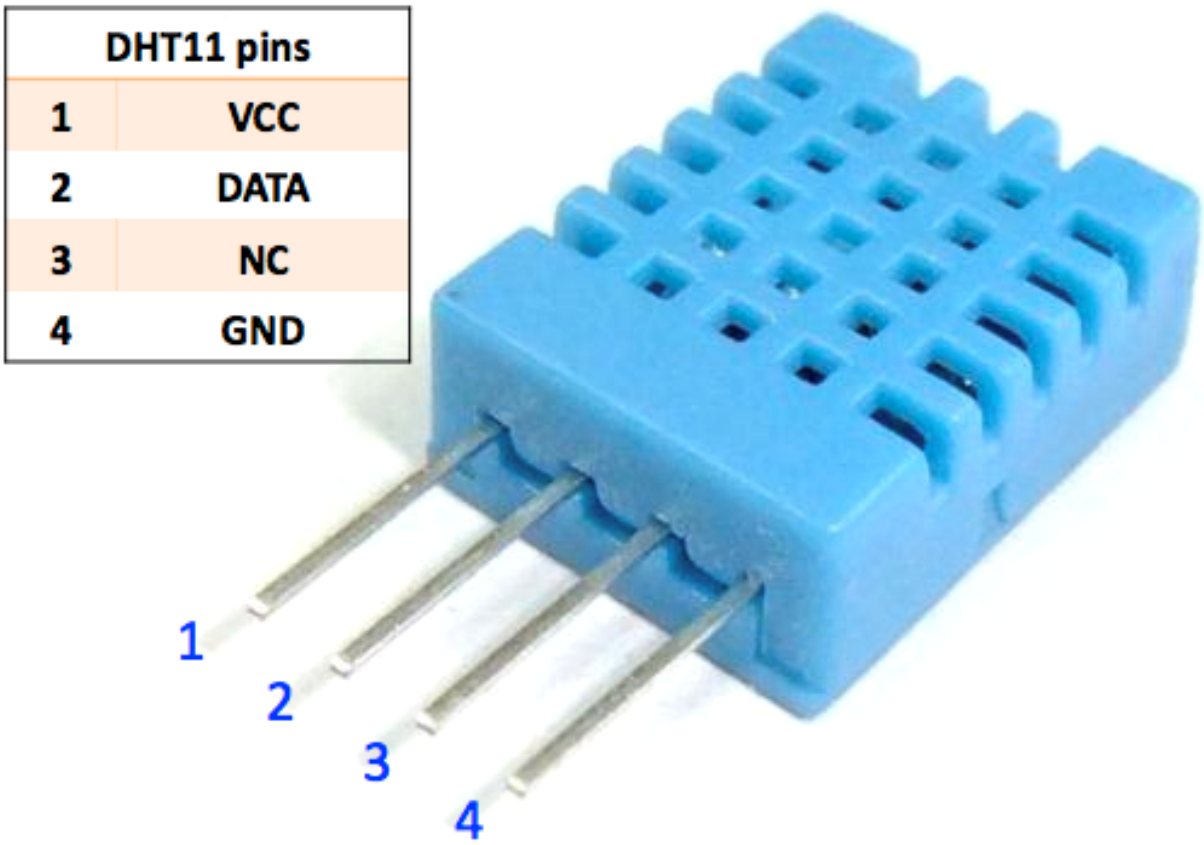
- *5.10 Termómetro* (Proyecto Arduino)
- *8.4 Comunicación IoT con MQTT* (Proyecto Arduino)
- *5.10 Detección de Temperatura* (Proyecto MicroPython)
- *2.6 Alarma de Baja Temperatura* (Proyecto Scratch)

6.27 Sensor de Humedad y Temperatura DHT11

El sensor digital de temperatura y humedad DHT11 es un sensor compuesto que contiene una salida de señal digital calibrada de temperatura y humedad. Se aplica la tecnología de una colección de módulos digitales dedicados y la tecnología de detección de temperatura y humedad para garantizar que el producto tenga una alta fiabilidad y una excelente estabilidad a largo plazo.

El sensor incluye un componente sensible a la humedad resistivo y un dispositivo de medición de temperatura NTC, y está conectado con un microcontrolador de 8 bits de alto rendimiento.

Solo hay tres pines disponibles para su uso: VCC, GND y DATA. El proceso de comunicación comienza con la línea de datos enviando señales de inicio al DHT11, y el DHT11 recibe las señales y devuelve una señal de respuesta. Luego, el host recibe la señal de respuesta y comienza a recibir datos de humedad y temperatura de 40 bits (entero de humedad de 8 bits + decimal de humedad de 8 bits + entero de temperatura de 8 bits + decimal de temperatura de 8 bits + suma de verificación de 8 bits).



Características

1. Rango de medición de humedad: 20 - 90 %RH
2. Rango de medición de temperatura: 0 - 60°C
3. Señales digitales de salida que indican temperatura y humedad
4. Voltaje de trabajo: DC 5V; Tamaño de la placa de circuito impreso: 2.0 x 2.0 cm
5. Precisión de la medición de humedad: ±5 %RH
6. Precisión de la medición de temperatura: ±2°C

- Hoja de datos del DHT11

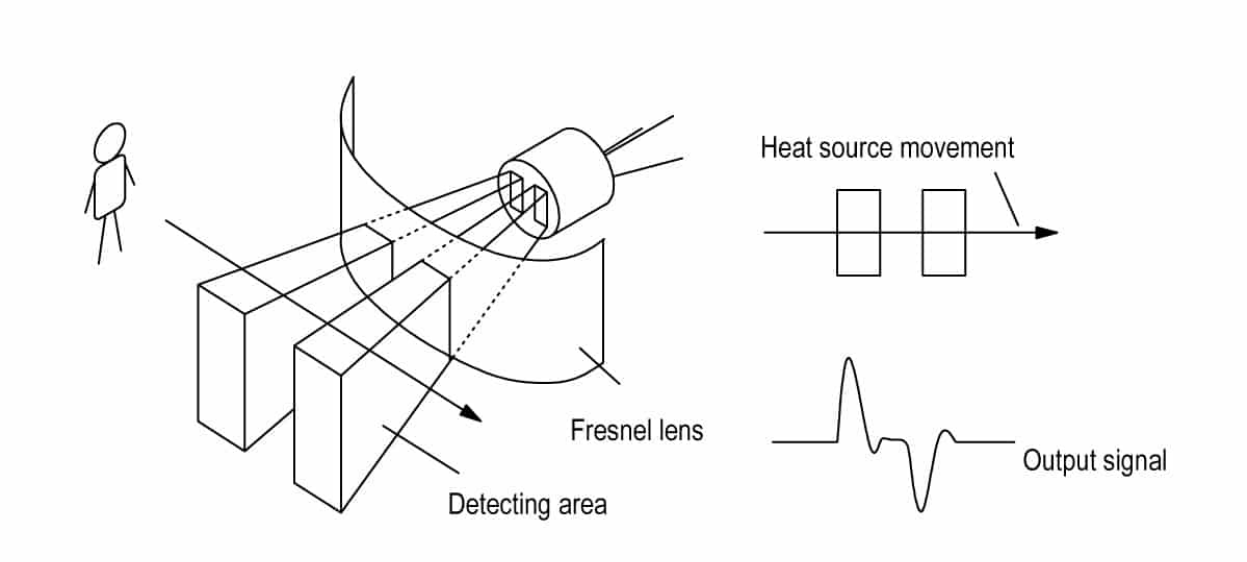
Ejemplo

- *5.13 Temperatura - Humedad* (Proyecto Arduino)
- *6.6 Monitor de Plantas* (Proyecto Arduino)
- *8.6 Monitoreo de Temperatura y Humedad con Adafruit IO* (Proyecto Arduino)
- *5.13 Temperatura - Humedad* (Proyecto MicroPython)
- *6.8 Monitor de Plantas* (Proyecto MicroPython)

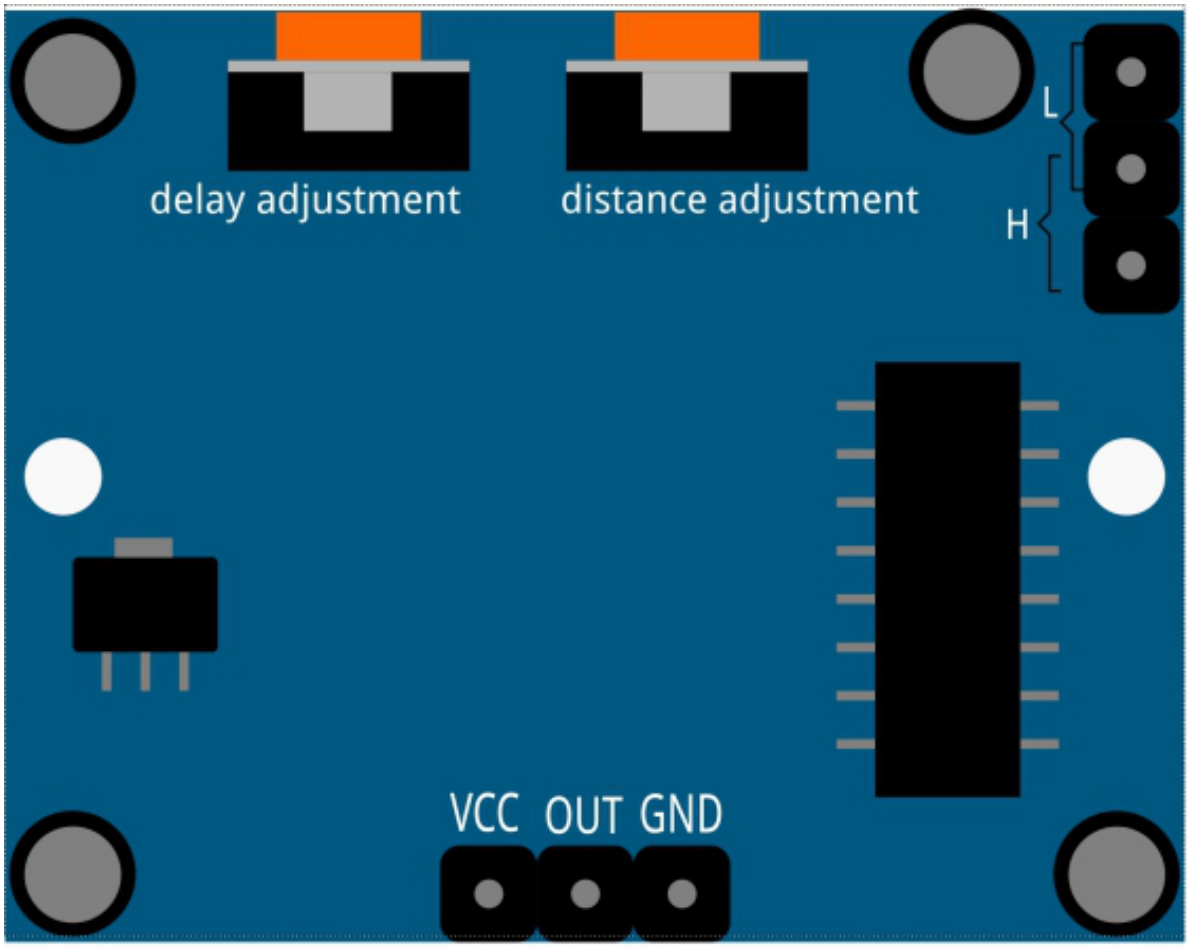
6.28 Módulo Sensor de Movimiento PIR

El sensor PIR detecta radiación de calor infrarrojo que puede utilizarse para detectar la presencia de organismos que emiten radiación de calor infrarrojo.

El sensor PIR está dividido en dos ranuras que están conectadas a un amplificador diferencial. Cuando un objeto estacionario está frente al sensor, las dos ranuras reciben la misma cantidad de radiación y la salida es cero. Cuando un objeto en movimiento está frente al sensor, una de las ranuras recibe más radiación que la otra, lo que hace que la salida fluctúe alta o baja. Este cambio en el voltaje de salida es el resultado de la detección de movimiento.



Después de que el módulo de detección esté cableado, hay una inicialización de un minuto. Durante la inicialización, el módulo emitirá de 0 a 3 veces a intervalos. Luego, el módulo estará en modo de espera. Mantenga la interferencia de fuentes de luz y otras fuentes alejadas de la superficie del módulo para evitar el mal funcionamiento causado por la señal de interferencia. Incluso es mejor utilizar el módulo sin demasiado viento, ya que el viento también puede interferir con el sensor.



Ajuste de Distancia

Girando el botón del potenciómetro de ajuste de distancia en el sentido de las agujas del reloj, aumenta el rango de distancia de detección, y el rango máximo de distancia de detección es de aproximadamente 0 a 7 metros. Si lo gira en sentido contrario a las agujas del reloj, el rango de distancia de detección se reduce, y el rango de distancia de detección mínimo es de aproximadamente 0 a 3 metros.

Ajuste de Retardo

Gire el botón del potenciómetro de ajuste de retardo en el sentido de las agujas del reloj, también puede ver cómo aumenta el retardo de detección. El máximo del retardo de detección puede alcanzar hasta 300s. Por el contrario, si lo gira en sentido contrario a las agujas del reloj, puede acortar el retardo con un mínimo de 5s.

Dos Modos de Disparo

Elija diferentes modos usando la tapa del jumper.

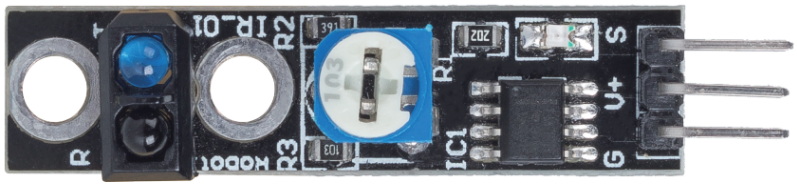
- **H**: Modo de disparo repetible, después de detectar el cuerpo humano, el módulo emite un nivel alto. Durante el período de retardo subsiguiente, si alguien entra en el rango de detección, la salida seguirá siendo de nivel alto.
- **L**: Modo de disparo no repetible, emite un nivel alto cuando detecta el cuerpo humano. Después del retardo, la salida cambiará automáticamente de nivel alto a nivel bajo.

Ejemplo

- *5.5 Detección de Movimiento Humano* (Proyecto Arduino)
- *8.7 Cámara ESP con Bot de Telegram* (Proyecto Arduino)

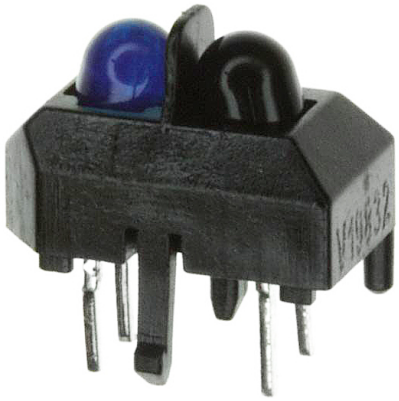
- *5.5 Detección de Movimiento Humano* (Proyecto MicroPython)

6.29 Módulo de Seguimiento de Línea



- S: Normalmente nivel bajo, nivel alto cuando se detecta la línea negra.
- V+: Suministro de energía, 3.3V~5V
- G: Tierra

Este es un módulo de seguimiento de línea de 1 canal que, como sugiere el nombre, sigue líneas negras sobre un fondo blanco o líneas blancas sobre un fondo negro.



El módulo utiliza un sensor infrarrojo TCRT5000, que consta de un LED infrarrojo (azul) y un fototransistor fotosensible (negro).

- El LED infrarrojo azul, cuando se enciende, emite luz infrarroja que es invisible para el ojo humano.
- El fototransistor negro, que se utiliza para recibir luz infrarroja, tiene una resistencia interna cuya resistencia varía con la luz infrarroja recibida; cuanto más luz infrarroja recibe, menor es su resistencia y viceversa.

Hay un comparador LM393 en el módulo, que se utiliza para comparar el voltaje del fototransistor con el voltaje establecido (ajustado por potenciómetro); si es mayor que el voltaje establecido, la salida es 1; de lo contrario, la salida es 0.

Por lo tanto, cuando el tubo emisor infrarrojo brilla sobre una superficie negra, debido a que el negro absorberá la luz, el fototransistor fotosensible recibirá menos luz infrarroja, su resistencia aumentará (aumento de voltaje), después del comparador LM393, la salida será de alto nivel.

De manera similar, cuando brilla sobre una superficie blanca, la luz reflejada será mayor y la resistencia del fototransistor fotosensible disminuirá (disminución de voltaje); por lo tanto, el comparador emite un nivel bajo y el LED indicador se enciende.

- TCRT5000

Características

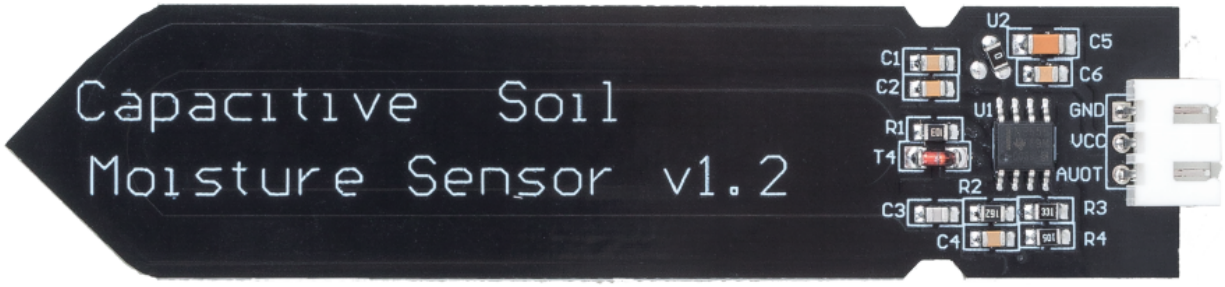
- Utiliza el sensor de emisión infrarroja TCRT5000

- Distancia de detección: 1-8mm, distancia focal de 2.5mm
- Señal de salida del comparador limpia, buena forma de onda, capacidad de conducción mayor a 15mA
- Utiliza un potenciómetro para ajuste de sensibilidad
- Voltaje de funcionamiento: 3.3V-5V
- Salida digital: 0 (blanco) y 1 (negro)
- Utiliza un comparador LM393 de voltaje amplio.
- Tamaño: 42mmx10mm

Ejemplo

- *5.4 Detectar la Línea* (Proyecto Arduino)
- *5.4 Detección de Líneas* (Proyecto MicroPython)
- *2.19 JUEGO - Protege Tu Corazón* (Proyecto Scratch)

6.30 Módulo de Humedad del Suelo

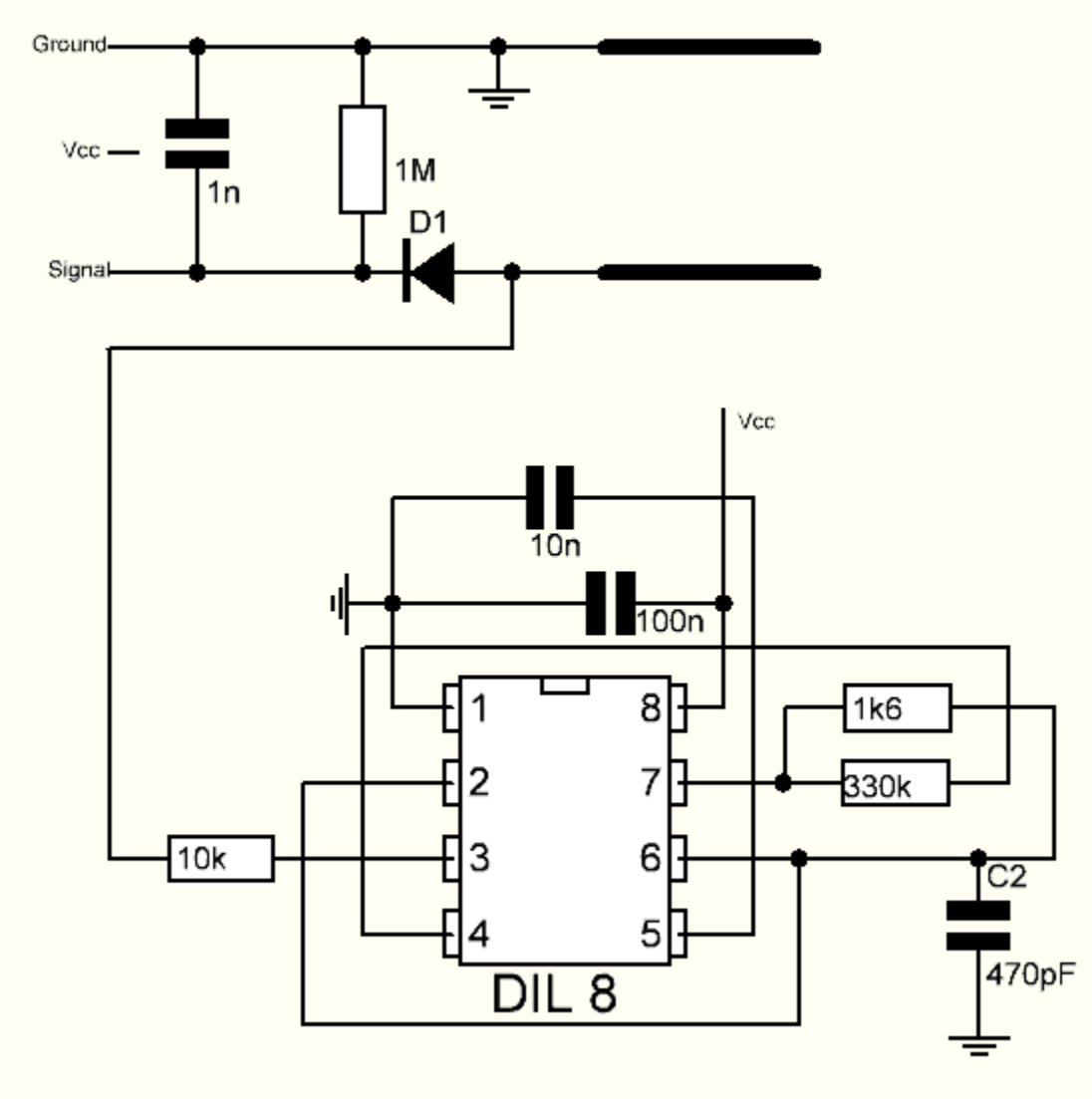


- GND: Tierra
- VCC: Suministro de energía, 3.3v~5V
- AOUT: Devuelve el valor de humedad del suelo, cuanto más húmedo esté el suelo, menor será su valor.

Este sensor de humedad del suelo capacitivo difiere de la mayoría de los sensores resistivos en el mercado, utilizando el principio de inducción capacitiva para detectar la humedad del suelo. Evita el problema de que los sensores resistivos sean altamente susceptibles a la corrosión y extiende considerablemente su vida útil.

Está hecho de materiales resistentes a la corrosión y tiene una excelente vida útil. Inscríbelo en el suelo alrededor de las plantas y monitorea datos de humedad del suelo en tiempo real. El módulo incluye un regulador de voltaje incorporado que le permite operar en un rango de voltaje de 3.3 ~ 5.5 V. Es ideal para microcontroladores de baja tensión con suministros de 3.3 V y 5 V.

El esquemático de hardware del sensor de humedad del suelo capacitivo se muestra a continuación.



Hay un oscilador de frecuencia fija, que está construido con un circuito integrado temporizador 555. La onda cuadrada generada se alimenta luego al sensor como un condensador. Sin embargo, para la señal de onda cuadrada, el condensador tiene cierta reactancia o, para hablar claro, una resistencia con una resistencia ohmica pura (resistencia de 10k en el pin 3) para formar un divisor de voltaje.

Cuanto mayor sea la humedad del suelo, mayor será la capacitancia del sensor. Como resultado, la onda cuadrada tiene menos reactancia, lo que reduce el voltaje en la línea de señal, y menor es el valor de la entrada analógica a través del microcontrolador.

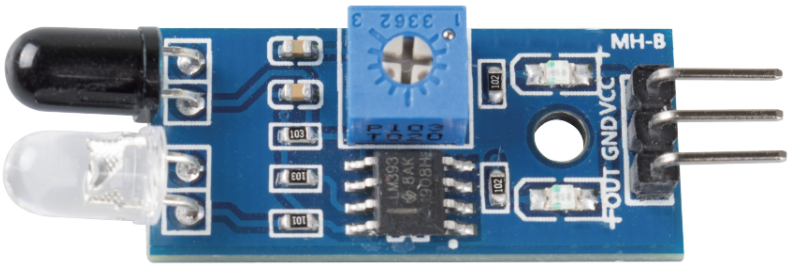
Especificaciones

- Voltaje de Operación: 3.3 ~ 5.5 VDC
- Voltaje de Salida: 0 ~ 3.0VDC
- Corriente de Operación: 5mA
- Interfaz: PH2.0-3P
- Dimensiones: 3.86 x 0.905 pulgadas (L x A)
- Peso: 15g

Ejemplo

- *5.9 Medir la Humedad del Suelo* (Proyecto Arduino)
- *6.6 Monitor de Plantas* (Proyecto Arduino)
- *5.9 Medir la Humedad del Suelo* (Proyecto MicroPython)
- *6.8 Monitor de Plantas* (Proyecto MicroPython)

6.31 Módulo de Evitación de Obstáculos

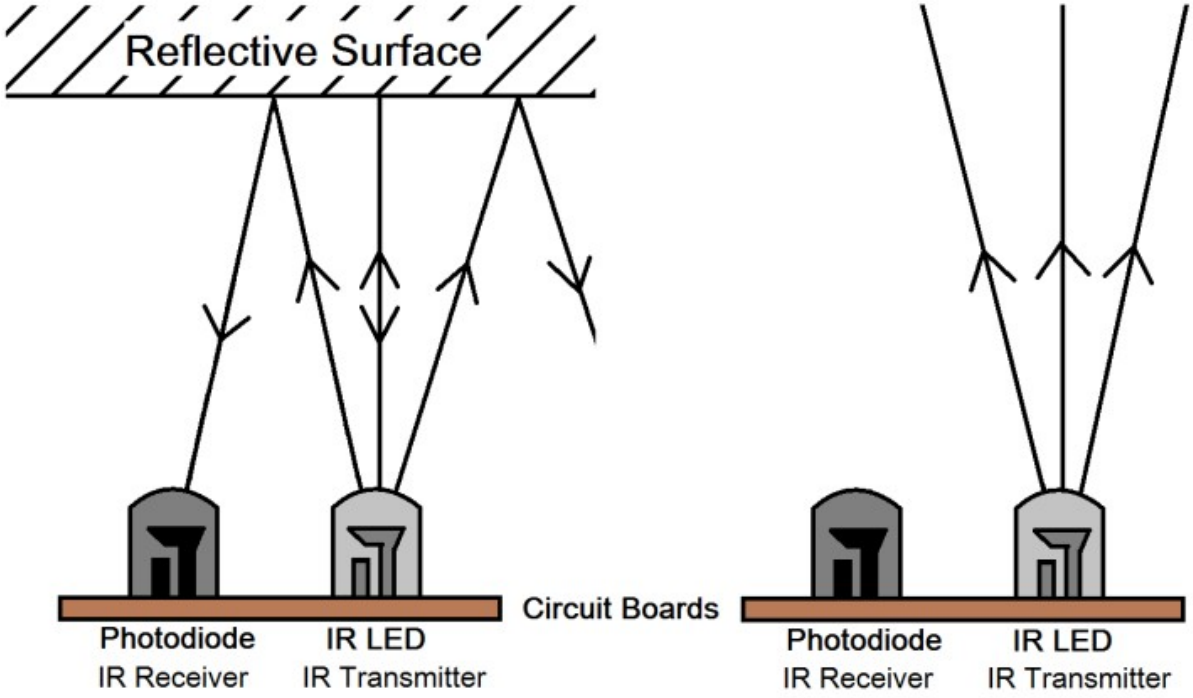


- **VCC**: Alimentación, 3.3 ~ 5V DC.
- **GND**: Tierra
- **OUT**: Pin de señal, normalmente nivel alto, y nivel bajo cuando se detecta un obstáculo.

El módulo de evitación de obstáculos por infrarrojos tiene una gran adaptabilidad a la luz ambiental, cuenta con un par de tubos emisores y receptores de infrarrojos.

El tubo emisor emite frecuencia infrarroja, cuando la dirección de detección encuentra un obstáculo, la radiación infrarroja es recibida por el tubo receptor, después del procesamiento del circuito comparador, el indicador se iluminará y emitirá una señal de nivel bajo.

La distancia de detección se puede ajustar mediante un potenciómetro, el rango de distancia efectiva es de 2 a 30 cm.



Ejemplo

- *5.3 Detectar el Obstáculo* (Proyecto Arduino)
- *5.3 Detección de Obstáculos* (Proyecto MicroPython)
- *2.11 JUEGO - Disparos* (Proyecto Scratch)
- *2.18 JUEGO - No Toques la Baldosa Blanca* (Proyecto Scratch)

6.32 Módulo Ultrasonido

- **TRIG**: Entrada de Pulso de Disparo
- **ECHO**: Salida de Pulso de Eco
- **GND**: Tierra

- **VCC**: Suministro de 5V

Este es el sensor de distancia ultrasónico HC-SR04, que proporciona mediciones sin contacto desde 2 cm hasta 400 cm con una precisión de rango de hasta 3 mm. Incluido en el módulo hay un transmisor ultrasónico, un receptor y un circuito de control.

Solo necesitas conectar 4 pines: VCC (alimentación), Trig (disparador), Echo (recepción) y GND (tierra) para facilitar su uso en tus proyectos de medición.

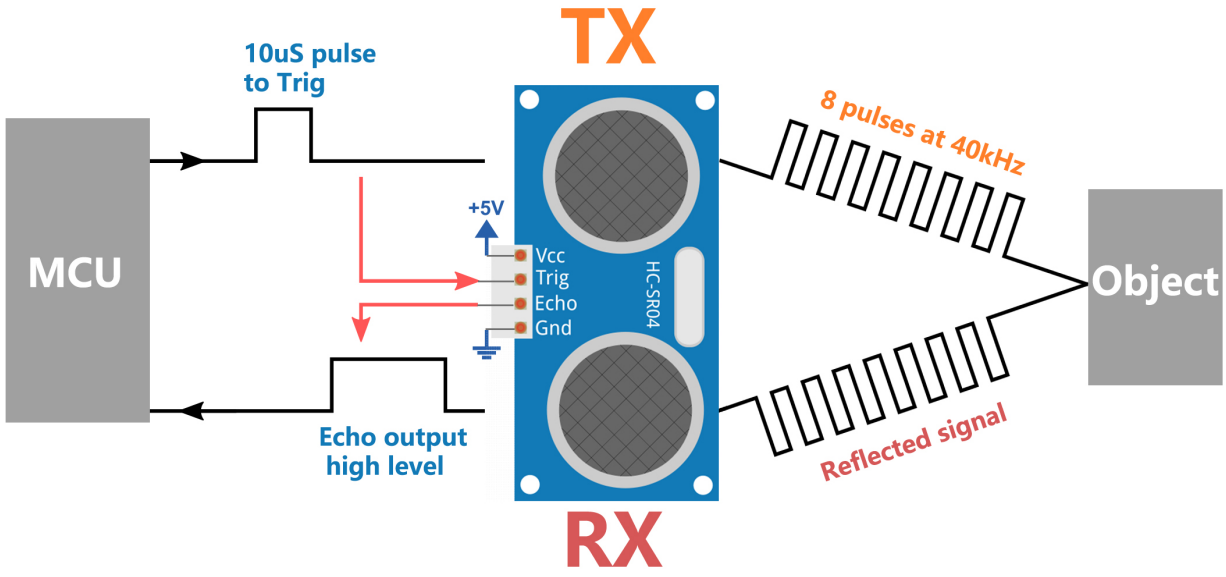
Características

- Voltaje de Trabajo: DC5V
- Corriente de Trabajo: 16mA
- Frecuencia de Trabajo: 40Hz
- Rango Máximo: 500cm
- Rango Mínimo: 2cm
- Señal de Entrada de Disparo: Pulso TTL de 10uS
- Señal de Salida de Eco: Señal de nivel TTL de entrada y el rango en proporción
- Conector: XH2.54-4P
- Dimensiones: 46x20.5x15 mm

Principio

Los principios básicos son los siguientes:

- Usando el disparador de IO durante al menos 10us con una señal de nivel alto.
- El módulo envía una ráfaga de ultrasonido de 8 ciclos a 40 kHz y detecta si se recibe una señal de pulso.
- Echo emitirá un nivel alto si se recibe una señal; la duración del nivel alto es el tiempo desde la emisión hasta el retorno.
- Distancia = (tiempo de nivel alto x velocidad del sonido (340M/S)) / 2



Fórmula:

- us / 58 = distancia en centímetros

- us / 148 = distancia en pulgadas
- distancia = tiempo de nivel alto x velocidad (340M/S) / 2

Nota: Este módulo no debe estar conectado bajo encendido, si es necesario, deja que la GND del módulo se conecte primero. De lo contrario, afectará el funcionamiento del módulo.

El área del objeto a medir debe ser al menos de 0.5 metros cuadrados y lo más plana posible. De lo contrario, afectará los resultados.

Ejemplo

- *5.12 Medición de Distancia* (Proyecto Arduino)
- *6.3 Ayuda para Reversa* (Proyecto Arduino)
- *5.12 Medición de Distancia* (Proyecto MicroPython)
- *6.4 Asistente de Reversa* (Proyecto MicroPython)
- *2.15 JUEGO - Loro Flappy* (Proyecto Scratch)

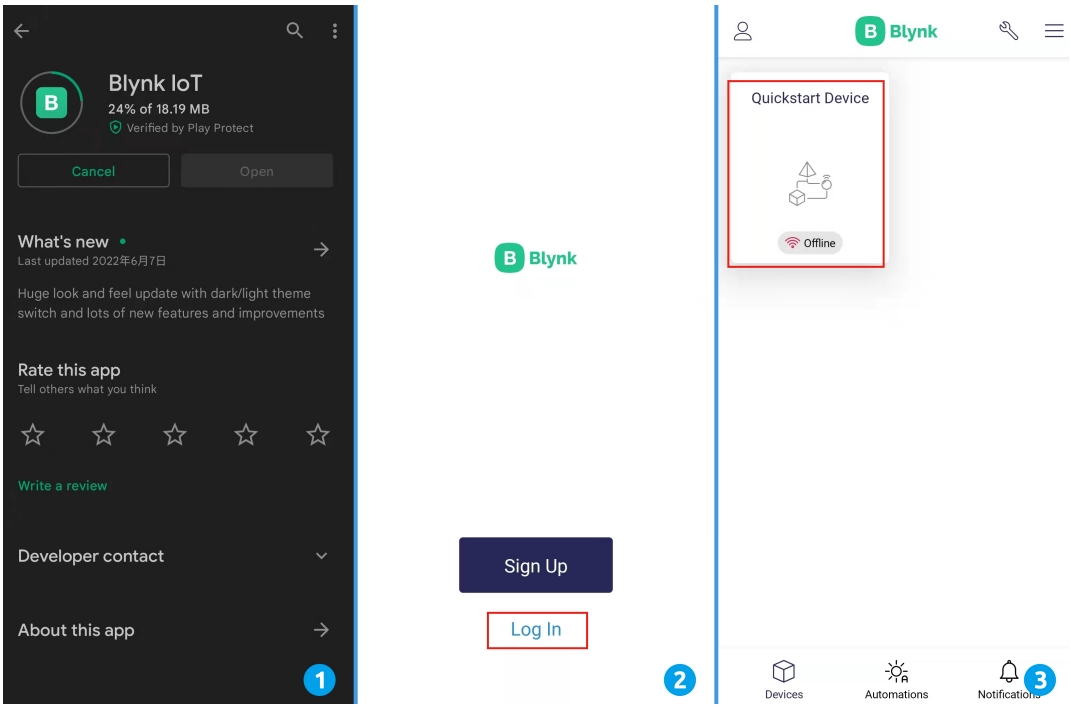
CAPÍTULO 7

FAQ

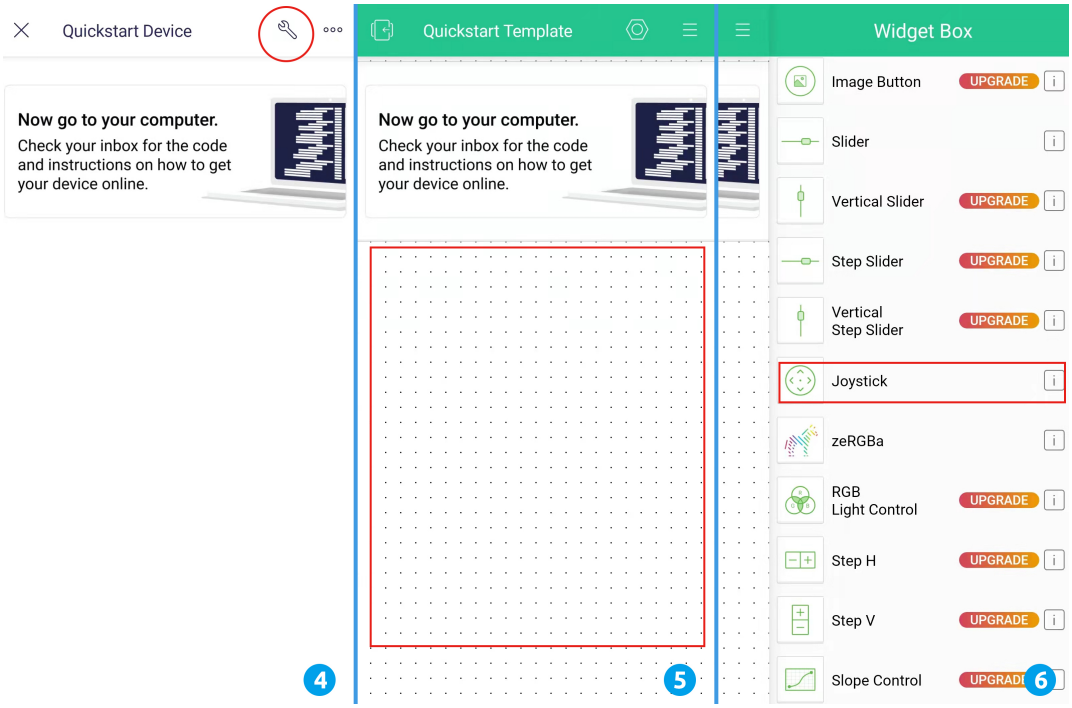
7.1 Cómo usar Blynk en dispositivos móviles?

Nota: Dado que los flujos de datos solo se pueden crear en Blynk en la web, necesitarás referenciar diferentes proyectos para crear flujos de datos en la web, luego sigue el tutorial a continuación para crear widgets en Blynk en tu dispositivo móvil.

1. Abre Google Play o APP Store en tu dispositivo móvil y busca «Blynk IoT» (no Blynk(legacy)) para descargar.
2. Tras abrir la APP, inicia sesión. Esta cuenta debe ser la misma que se usa en el cliente web.
3. Luego, ve al **Tablero de control** (si no tienes uno, créalo) y verás que el **Tablero de control** para móvil y web son independientes uno del otro.

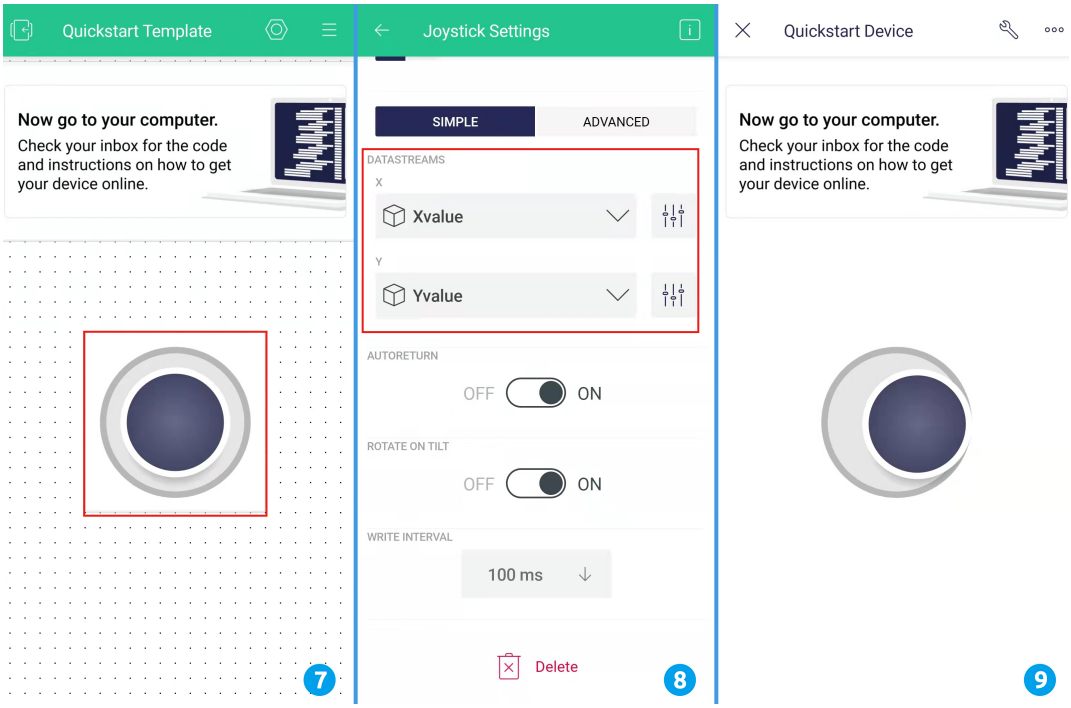


4. Pulsa el ícono **Editar**.
5. Haz clic en el área en blanco.
6. Elige el mismo widget que en la página web, por ejemplo, selecciona un widget **Joystick**.



7. Ahora verás aparecer un widget **Joystick** en el área en blanco, haz clic en él.
8. Aparecerán los ajustes de **Joystick**, selecciona los flujos de datos **Xvalue** y **Yvalue** que acabas de configurar en la página web. Ten en cuenta que cada widget corresponde a un flujo de datos diferente en cada proyecto.

9. Regresa a la página del **Tablero de control** y podrás operar el **Joystick** cuando lo desees.

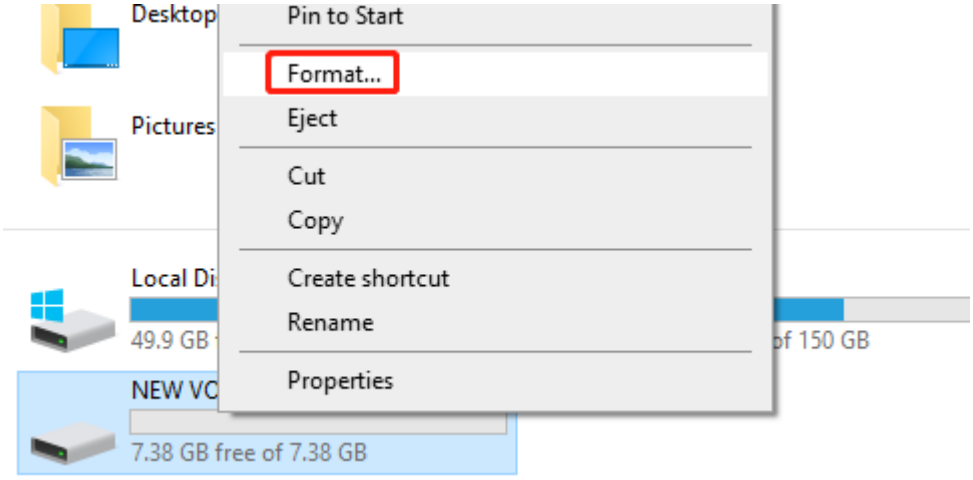


7.2 ¿Cómo formatear la tarjeta SD?

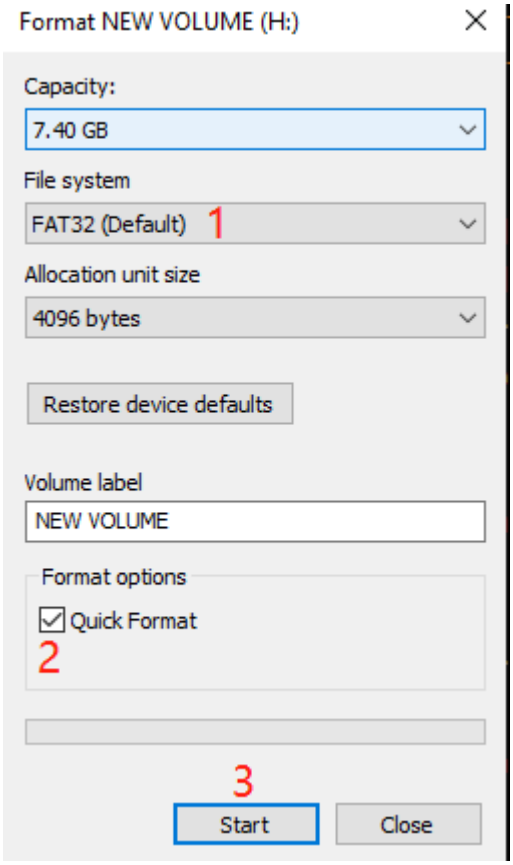
Los pasos para asegurar que tu tarjeta SD esté formateada correctamente pueden variar dependiendo de tu sistema operativo. Aquí te mostramos pasos sencillos sobre cómo formatear una tarjeta SD en Windows, MacOS y Linux:

Windows

1. Inserta tu tarjeta SD en el ordenador, luego abre «Mi PC» o «Este Equipo». Haz clic derecho en tu tarjeta SD y selecciona «Formatear».

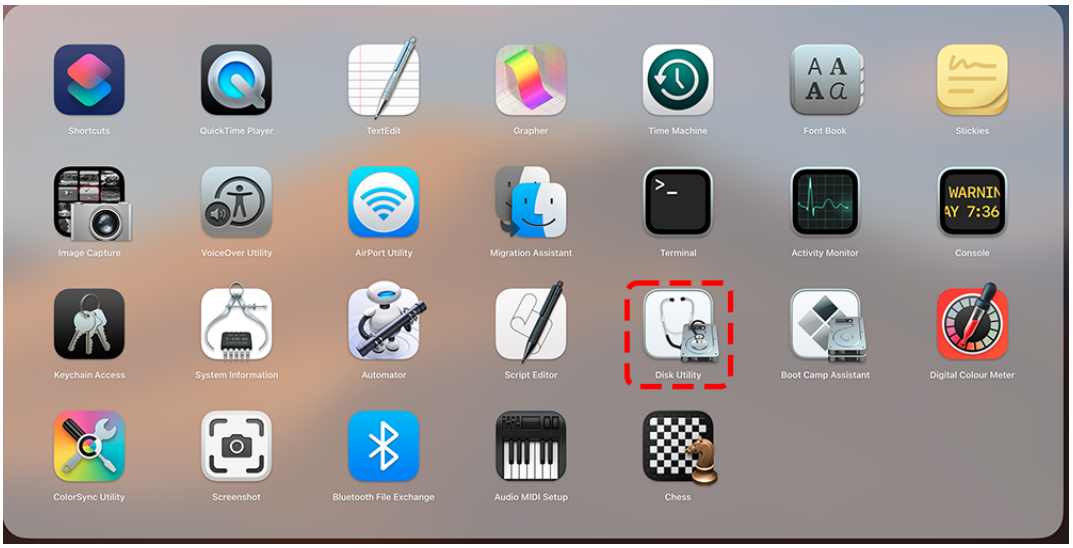


2. En el menú desplegable de sistema de archivos, selecciona el sistema de archivos deseado (usualmente elige FAT32, o para tarjetas SD mayores de 32GB, podrías necesitar elegir exFAT). Marca «Formato rápido» y luego haz clic en «Iniciar».

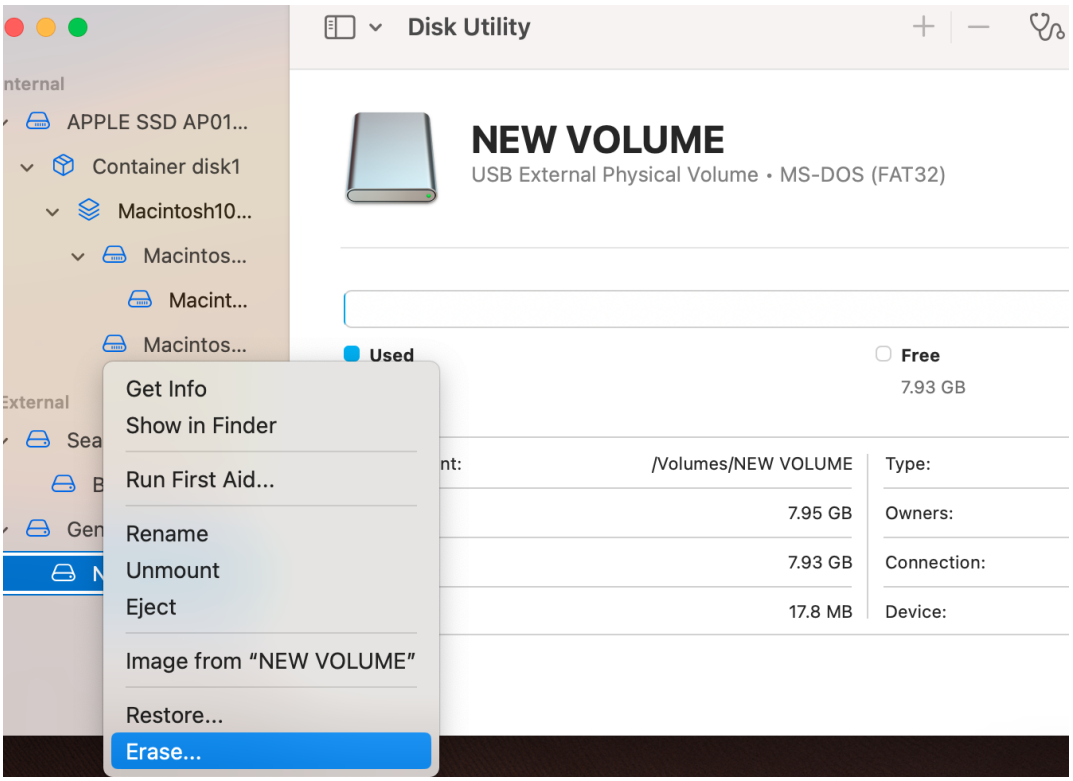


MacOS

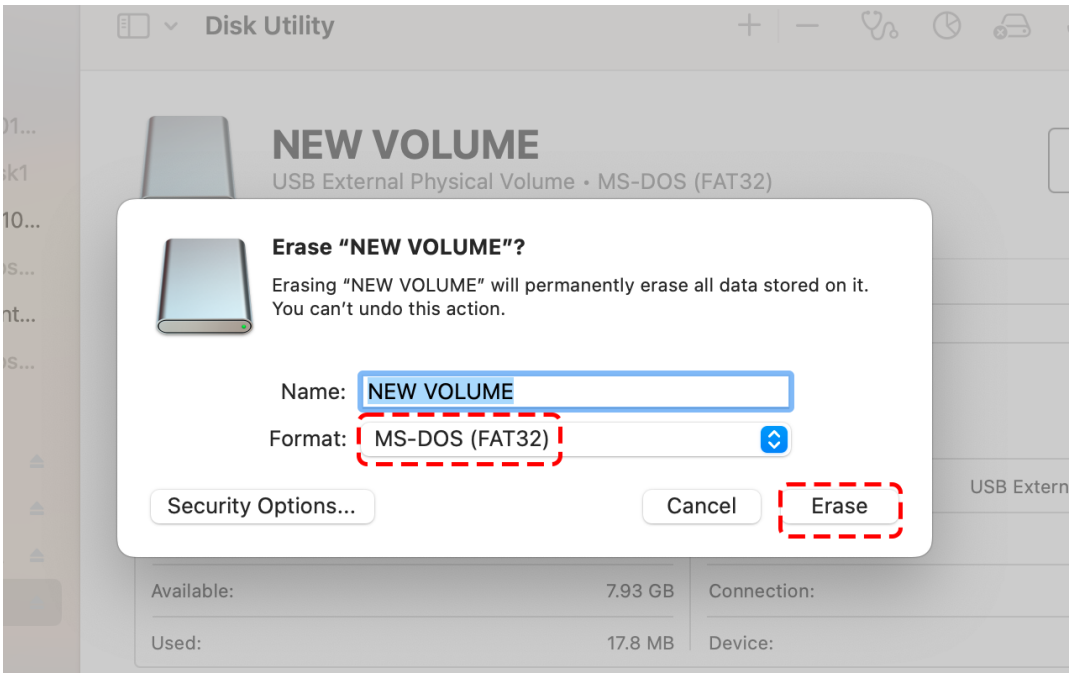
1. Inserta tu tarjeta SD en el ordenador. Abre la aplicación «Utilidad de Discos» (se puede encontrar en la carpeta «Utilidades»).



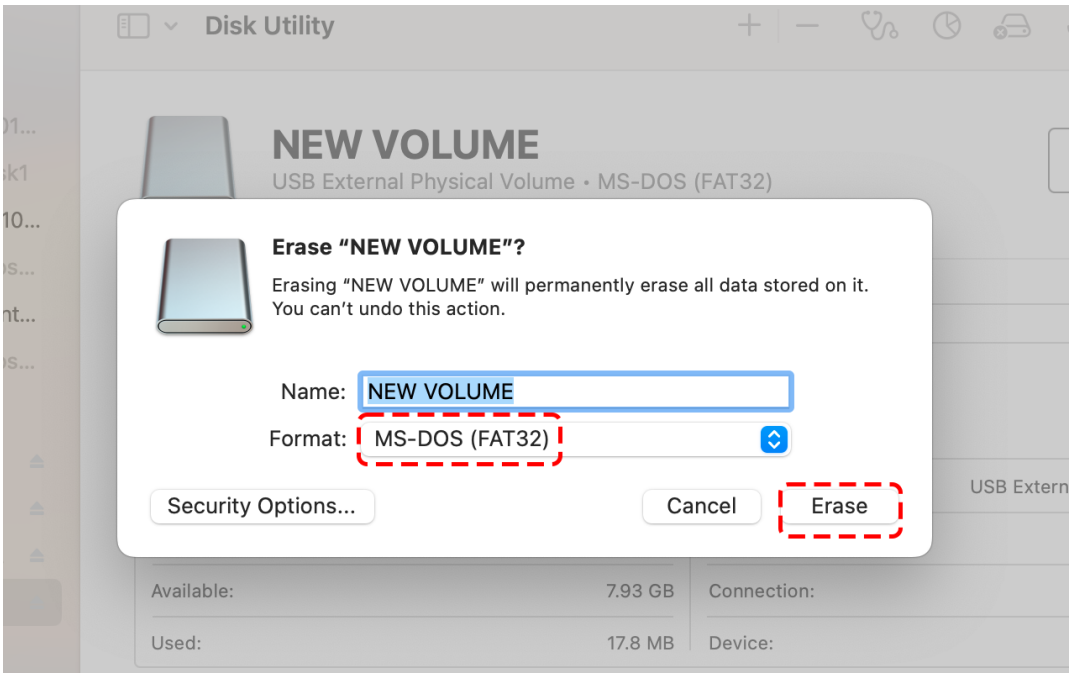
2. Selecciona tu tarjeta SD de la lista a la izquierda y luego haz clic en «Borrar».



3. Del menú desplegable de formato, elige tu sistema de archivos deseado (usualmente elige MS-DOS (FAT) para FAT32, o ExFAT para tarjetas SD mayores de 32GB) y luego haz clic en «Borrar».



4. Finalmente, espera a que se complete el formateo.



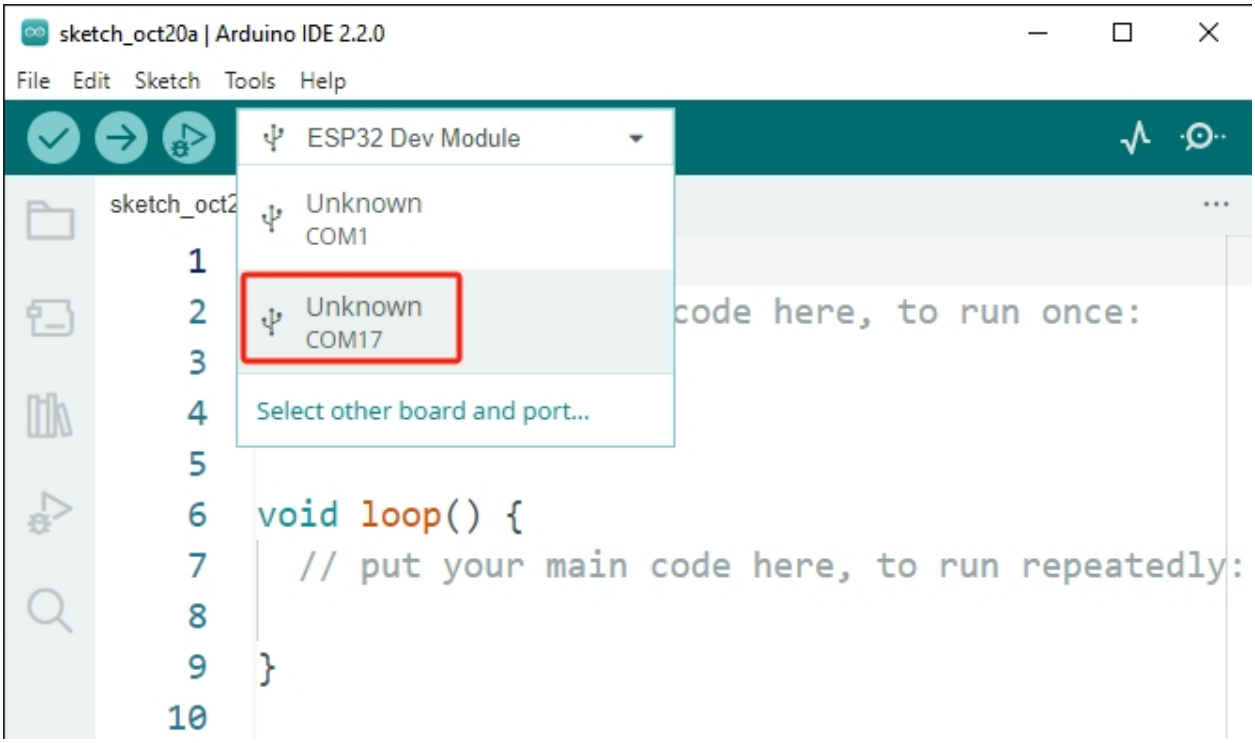
Linux

- Primero, inserta tu tarjeta SD y luego abre una terminal.
- Escribe `lsblk` y encuentra el nombre de tu tarjeta SD en la lista de dispositivos (por ejemplo, podría ser `sdb`).
- Usa el comando `umount` para desmontar la tarjeta SD, como `sudo umount /dev/sdb*`.
- Usa el comando `mkfs` para formatear la tarjeta SD. Por ejemplo, `sudo mkfs.vfat /dev/sdb1` formateará la tarjeta SD a un sistema de archivos FAT32 (para tarjetas SD mayores de 32GB, podrías necesitar usar `mkfs.exfat`).

Antes de formatear tu tarjeta SD, asegúrate de respaldar cualquier dato importante en la tarjeta SD, ya que la operación de formateo borrará todos los archivos en la tarjeta SD.

7.3 ¿Siempre aparece «COMxx desconocido»?

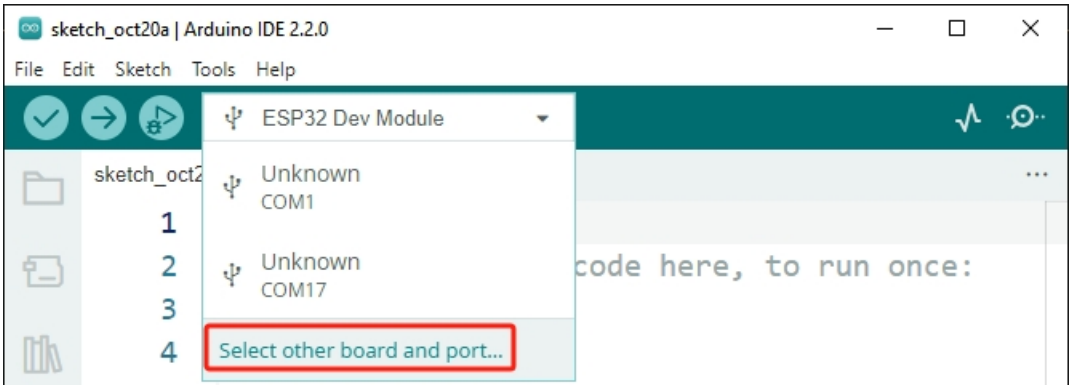
Cuando conectas el ESP32 al ordenador, el IDE de Arduino a menudo muestra `COMxx desconocido`. ¿Por qué sucede esto?



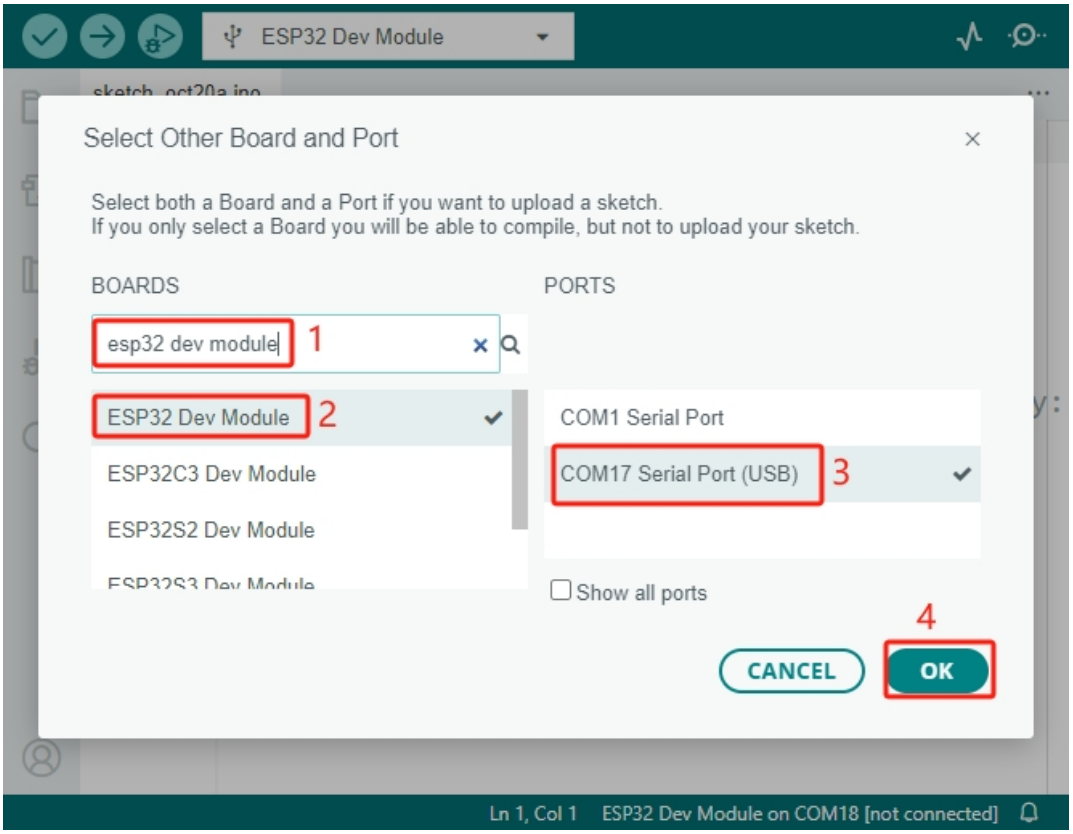
Esto se debe a que el controlador USB del ESP32 es diferente al de las placas Arduino regulares. El IDE de Arduino no puede reconocer automáticamente esta placa.

En tal escenario, necesitas seleccionar manualmente la placa correcta siguiendo estos pasos:

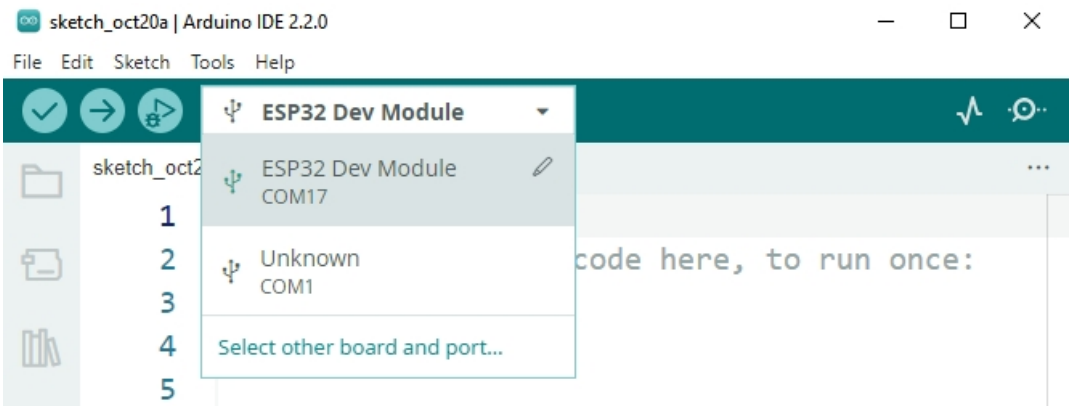
1. Haz clic en **«Seleccionar otra placa y puerto»**.



2. En la búsqueda, escribe **«esp32 dev module»**, luego selecciona la placa que aparece. Después, selecciona el puerto correcto y haz clic en **OK**.



3. Ahora, deberías poder ver tu placa y puerto en esta ventana de vista rápida.



CAPÍTULO 8

Gracias

Agradecemos a los evaluadores que evaluaron nuestros productos, a los veteranos que proporcionaron sugerencias para el tutorial y a los usuarios que han estado siguiéndonos y apoyándonos. ¡Sus valiosas sugerencias para nosotros son nuestra motivación para ofrecer mejores productos!

Agradecimientos Especiales

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Ahora, ¿podrías tomarte un momento para completar este cuestionario?

Nota: Después de enviar el cuestionario, por favor regresa al principio para ver los resultados.

CAPÍTULO 9

Copyright Notice