

---

# SunFounder 3in1 Kit

[www.sunfounder.com](http://www.sunfounder.com)

Feb 19, 2024





# CONTENTS

<b>1</b>	<b>Learn about the Components in Your Kit</b>	<b>3</b>
1.1	Arduino Uno R4 Minima . . . . .	4
1.2	ESP8266 Module . . . . .	7
1.3	Breadboard . . . . .	10
1.4	Resistor . . . . .	11
1.5	Capacitor . . . . .	14
1.6	Jumper Wires . . . . .	16
1.7	74HC595 . . . . .	16
1.8	LED . . . . .	18
1.9	RGB LED . . . . .	19
1.10	7-segment Display . . . . .	21
1.11	I2C LCD1602 . . . . .	24
1.12	Buzzer . . . . .	26
1.13	TT Motor . . . . .	27
1.14	Servo . . . . .	28
1.15	Centrifugal Pump . . . . .	31
1.16	L9110 Motor Driver Module . . . . .	32
1.17	Button . . . . .	34
1.18	Reed Switch . . . . .	35
1.19	Potentiometer . . . . .	36
1.20	Joystick Module . . . . .	38
1.21	IR Receiver . . . . .	39
1.22	Photoresistor . . . . .	42
1.23	Thermistor . . . . .	43
1.24	DHT11 Humiture Sensor . . . . .	44
1.25	Line Tracking Module . . . . .	46
1.26	Soil Moisture Module . . . . .	47
1.27	Obstacle Avoidance Module . . . . .	49
1.28	Ultrasonic Module . . . . .	50
<b>2</b>	<b>Get Started with Arduino</b>	<b>53</b>
2.1	What is Arduino? . . . . .	53
2.2	What can Arduino do? . . . . .	53
2.3	How to build an Arduino Project . . . . .	54
<b>3</b>	<b>Download the Code</b>	<b>83</b>
<b>4</b>	<b>Basic Projects</b>	<b>85</b>
4.1	1. Digital Write . . . . .	85
4.2	2. Analog Write . . . . .	95

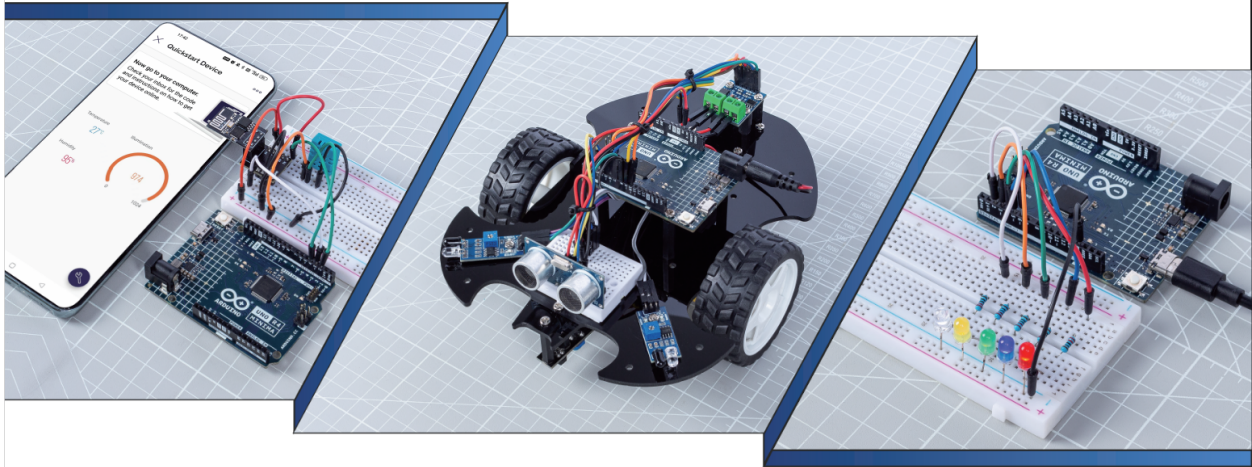
4.3	3. Digital Read . . . . .	103
4.4	4. Analog Read . . . . .	116
4.5	5. More Syntax . . . . .	131
4.6	6. Funny Project . . . . .	195
<b>5</b>	<b>Car Projects</b>	<b>219</b>
5.1	Assemble the Car . . . . .	219
5.2	1. Move . . . . .	230
5.3	2. Move by Code . . . . .	235
5.4	3. Speed Up . . . . .	239
5.5	4. Follow the line . . . . .	240
5.6	5. Play with Obstacle Avoidance Module . . . . .	244
5.7	6. Play with Ultrasonic Module . . . . .	248
5.8	7. Follow Your Hand . . . . .	251
5.9	8. Self-Driving Car . . . . .	253
5.10	9. Remote Control . . . . .	256
5.11	10. One Touch Start . . . . .	261
5.12	11. Speed Calibration . . . . .	263
<b>6</b>	<b>IoT Projects</b>	<b>267</b>
6.1	1. Get Started with Blynk . . . . .	268
6.2	2. Get Data from Blynk . . . . .	281
6.3	3. Push Data to Blynk . . . . .	291
6.4	4. Cloud Music Player . . . . .	297
6.5	5. Home Environment Monitoring . . . . .	303
6.6	6. Plant Monitor . . . . .	311
6.7	7. Current Limiting Gate . . . . .	318
6.8	8. IoT Car . . . . .	324
<b>7</b>	<b>FAQ</b>	<b>333</b>
7.1	How to use Blynk on mobile device? . . . . .	333
7.2	How to re-burn the firmware for ESP8266 module? . . . . .	335
<b>8</b>	<b>Thank You</b>	<b>345</b>
<b>9</b>	<b>Copyright Notice</b>	<b>347</b>

Thanks for choosing our 3 in 1 Starter Kit.

**Note:** This document is available in the following languages.

- 
- 
- 

Please click on the respective links to access the document in your preferred language.



When you bought a learning kit online, did it come with a simple PDF or booklet with only the steps to build the project?

Or maybe you want to build your own smart car, but the ones you find online are pricey and complicated

Or have you seen useful and interesting IoT projects made by others, but have no idea where to start?

All these problems can be solved with our 3 in 1 starter kit.

In the 3-in-1 starter kit, you will find a complete Arduino course to help beginners learn Arduino, as well as a wide variety of interesting projects that other learning kits do not offer, such as smart car projects and IoT projects. You will master Arduino as long as you follow the kit's course step by step, instead of just copying and pasting code, you will write your own code and implement your Arduino project however you like.

Come on! Start programming Arduino with confidence from zero to hero!

If you have any questions or other interesting ideas, feel free to send an email to [service@sunfounder.com](mailto:service@sunfounder.com).



## LEARN ABOUT THE COMPONENTS IN YOUR KIT

After opening the package, please check whether the quantity of components is compliance with product description and whether all components are in good condition.

### Packing List

 Resistor(10Ω) 10 pcs	 Resistor(100Ω) 10 pcs	 Resistor(220Ω) 30 pcs	 Resistor(330Ω) 10 pcs	 Resistor(1KΩ) 10 pcs	 Resistor(2KΩ) 10 pcs	 Resistor(5.1KΩ) 10 pcs	 Resistor(10KΩ) 10 pcs	 Resistor(100KΩ) 10 pcs
 Resistor(1MΩ) 10 pcs	 Reed Switch 2 pcs	 Button 5 pcs	 Active/Passive Buzzer 1+1 pcs	 Capacitor 104 pF 5 pcs	 Potentiometer 1 pc	 74HC595 1 pc	 7-segment Display 1 pc	 IR Receiver 1 pc
 Green LED 5 pcs	 Red LED 5 pcs	 White LED 5 pcs	 Blue LED 5 pcs	 Yellow LED 5 pcs	 RGB LED 1 pc	 Photoresistor/ Thermistor 1+1 pcs	 M2.5x6 Screw 10 pcs	 M3 Nut 8 pcs
 M3x10 Screw 4 pcs	 M3x6 Screw 30 pcs	 M3x30 Screw 6 pcs	 M2.5x11 Standoff 6 pcs	 M3x24 Standoff 7 pcs	 M3x10 Standoff 10 pcs	 Screwdriver 1 pc	 DHT11 1 pc	 ESP8266 Module 1 pc
 Soil Moisture Module 1 pc	 Joystick Module 1 pc	 Ultrasonic Module 1 pc	 IR Obstacle Avoidance Module 2 pcs	 I2C LCD 1602 1 pc	 Arduino UNO R4 Minima 1 pc	 Line Tracking Module 1 pc	 L9110 Module 1 pc	 ESP8266 Adapter Module 1 pc
 USB Cable 1 pc	 TT Wheel 2 pcs	 TT Motor 2 pcs	 1" Universal Wheel 1 pc	 Remote Control 1 pc	 Pump 1 pc	 Acrylic Plate 1 pc	 9G Servo 1 pc	 Mini Breadboard 1 pc
 Breadboard 1 pc	 Velcro 4 pcs	 Tube 1 pc	 Jump Wire F/M 20 pcs	 F/F DuPont Wire 20 pcs	 9V Battery Cable 1 pc	 Jump Wire M/M 65 pcs	 9V Battery 1 pc	Online Tutorials: <a href="https://3in1-kit-r4.rtfid.io">https://3in1-kit-r4.rtfid.io</a>

Below is the introduction to each component, which contains the operating principle of the component and the corresponding projects.

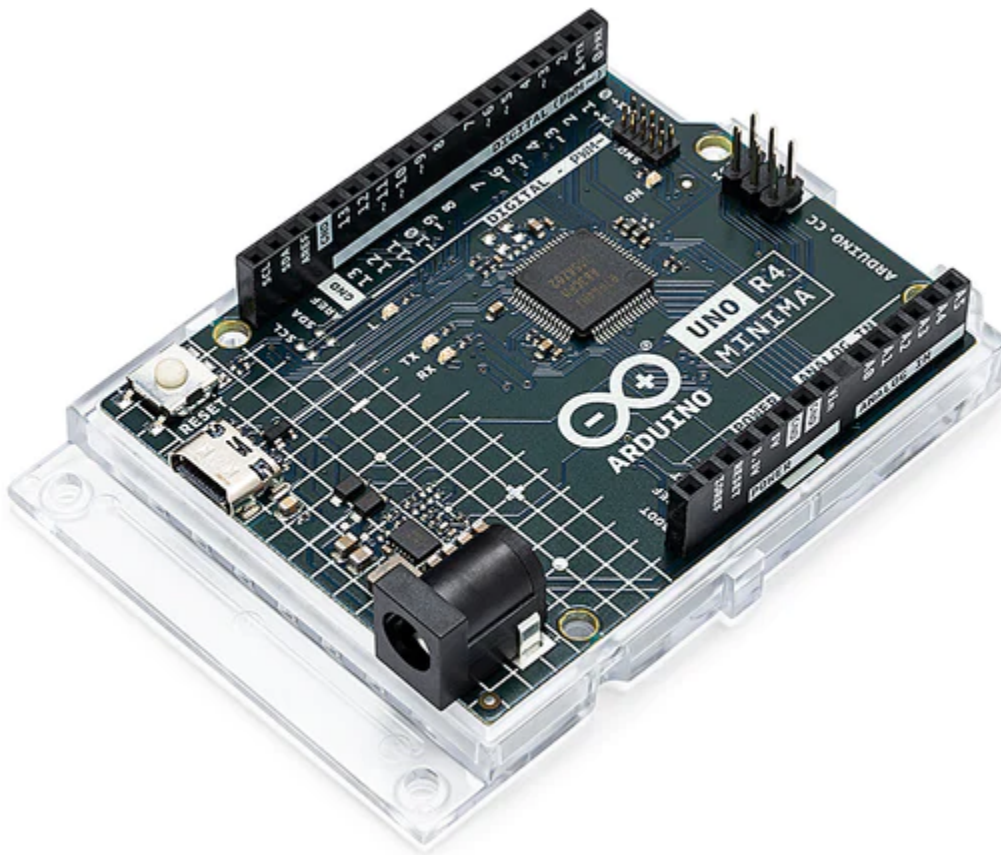
### Control Board

## 1.1 Arduino Uno R4 Minima

### Overview

Enhanced and improved, the Arduino UNO R4 Minima is armed with a powerful 32-bit microcontroller courtesy of Renesas. Brace yourself for increased processing power, expanded memory, and a whole new level of on-board peripherals. The best part? Compatibility with existing shields and accessories remains intact, and there's no need to make any changes to the standard form factor or 5 V operating voltage.

Joining the Arduino ecosystem, the UNO R4 is a trusty addition suitable for both beginners and seasoned electronics enthusiasts. Whether you're just starting out or looking to push the boundaries of your projects, this robust board delivers reliable performance every time.



Here's what the UNO R4 Minima brings to the table:

- **Hardware compatibility with UNO form factor:** The UNO R4 Minima maintains the same form factor, pinout, and 5 V operating voltage as its predecessor, the UNO R3. This ensures a seamless transition for existing shields and projects, leveraging the extensive and unique ecosystem already established for the Arduino UNO.
- **Expanded memory and faster clock:** Prepare for more precise calculations and the ability to handle complex projects with ease. The UNO R4 Minima boasts increased memory and a faster clock speed, empowering you to tackle demanding tasks effortlessly.
- **Extra on-board peripherals:** The UNO R4 Minima introduces a range of on-board peripherals, including a 12-bit DAC, CAN BUS, and OP AMP. These additional components provide you with expanded capabilities and flexibility in your designs.
- **Extended 24 V tolerance:** The UNO R4 Minima now supports a wider input voltage range, allowing power supplies up to 24 V. This enables seamless integration with motors, LED strips, and other actuators, simplifying

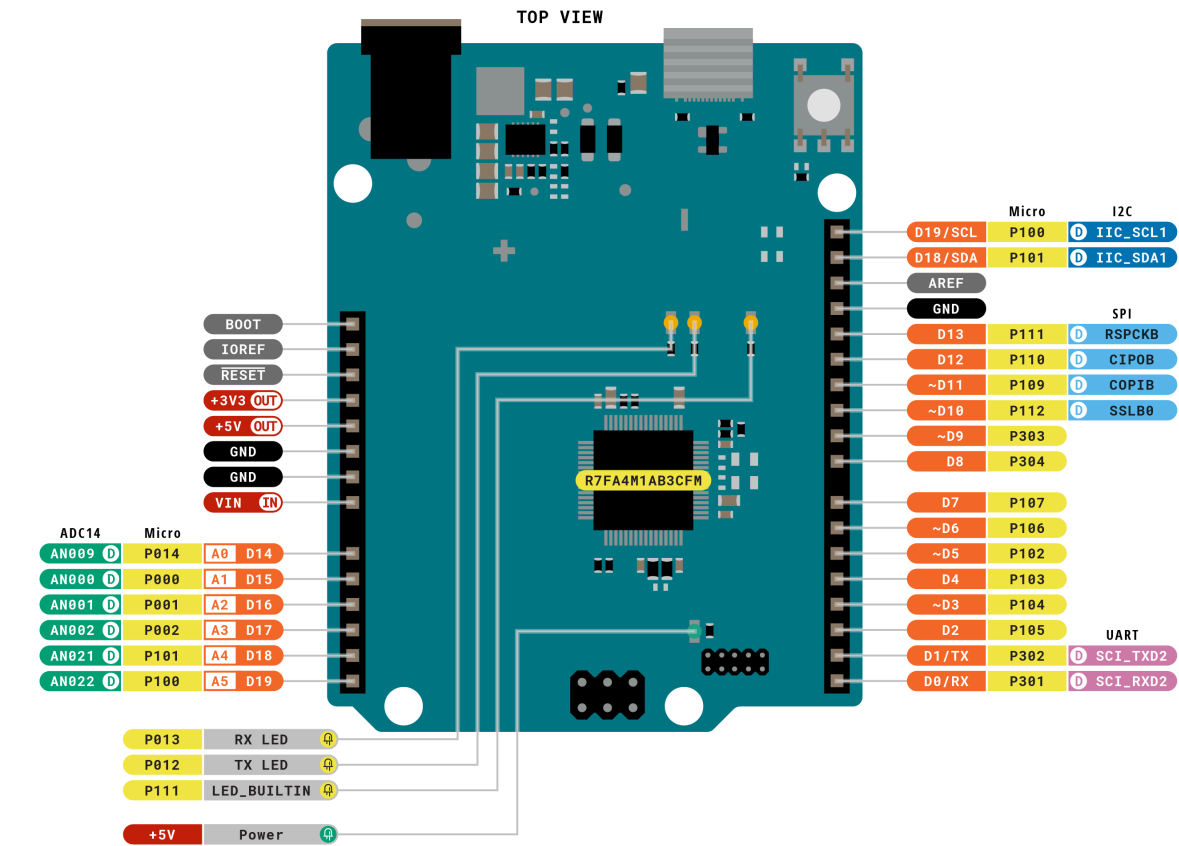
your projects by utilizing a single power source.

- **SWD connector:** Debugging is a critical aspect of any project. Simply connect an external debugger to the UNO R4 Minima and effortlessly monitor the inner workings of your system. Stay in control and gain valuable insights.
- **HID support:** The UNO R4 Minima comes with built-in HID (Human Interface Device) support, enabling it to simulate a mouse or keyboard when connected to a computer via a USB cable. This convenient feature makes it a breeze to send keystrokes and mouse movements to a computer, enhancing usability and functionality.

### **Tech specs**

- **Board:**
  - Name: Arduino® UNO R4 Minima
  - SKU: ABX00080
- **Microcontroller:** Renesas RA4M1 (Arm® Cortex®-M4)
- **USB:** USB-C® Programming Port
- **Pins:**
  - Digital I/O Pins: 14
  - Analog input pin: 6
  - DAC: 1
  - PWM pins: 6
- **Communication**
  - UART: Yes, 1x
  - I2C: Yes, 1x
  - SPI: Yes, 1x
  - CAN: Yes 1 CAN Bus
- **Power**
  - Circuit operating voltage: 5 V
  - Input voltage (VIN): 6-24 V
  - DC Current per I/O Pin: 8 mA
- **Clock speed** Main core 48 MHz
- **Memory** RA4M1 256 kB Flash, 32 kB RAM
- **Dimensions**
  - Width: 68.85 mm
  - Length: 53.34 mm

### **Pinout**



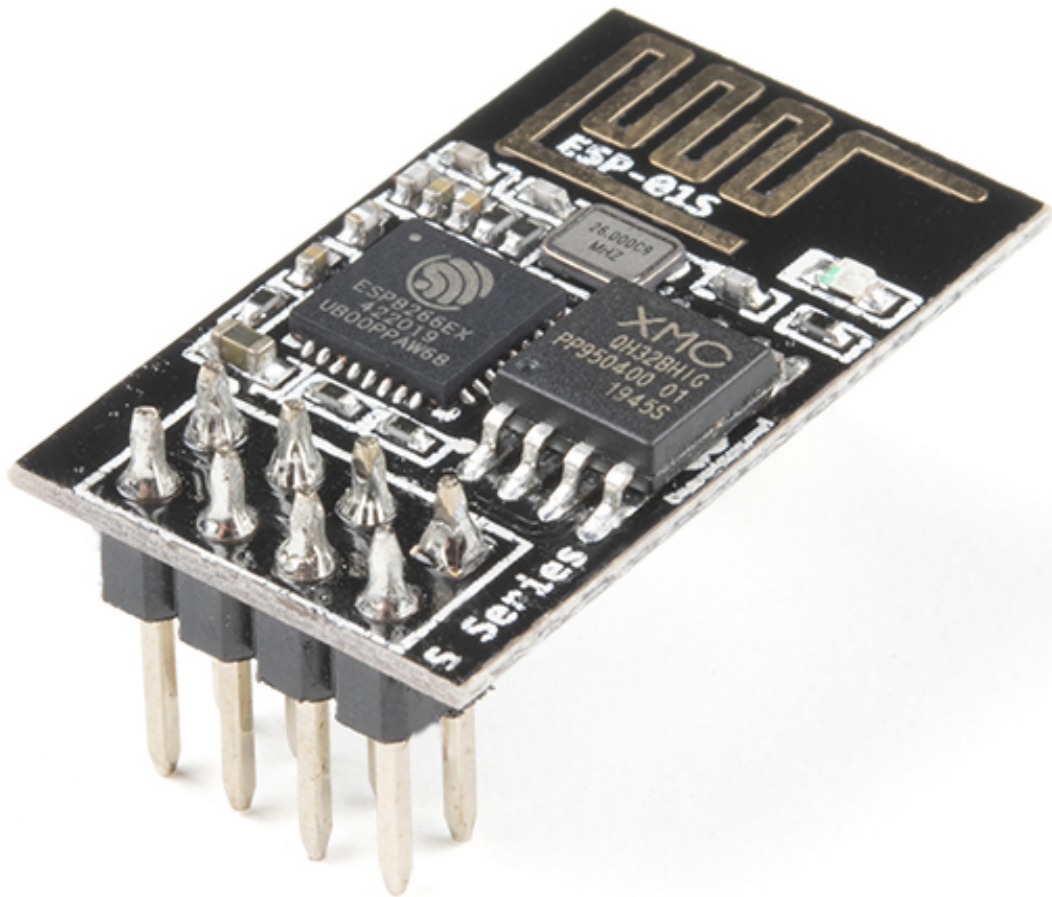
Legend:			
Digital	I2C	Other SERIAL	
Power	Analog	SPI	Analog
Ground	Main Part	UART/USART	PWM/Timer

ARDUINO

ARDUINO UNO R4 MINIMA  
SKU code: ABX00000  
Pinout  
Last update: 16 Jun, 2023



## 1.2 ESP8266 Module



The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems in Shanghai, China.

The chip first came to the attention of Western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first, there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, the chip, and the software on it, as well as to translate the Chinese documentation.

Pins of ESP8266 and their functions:

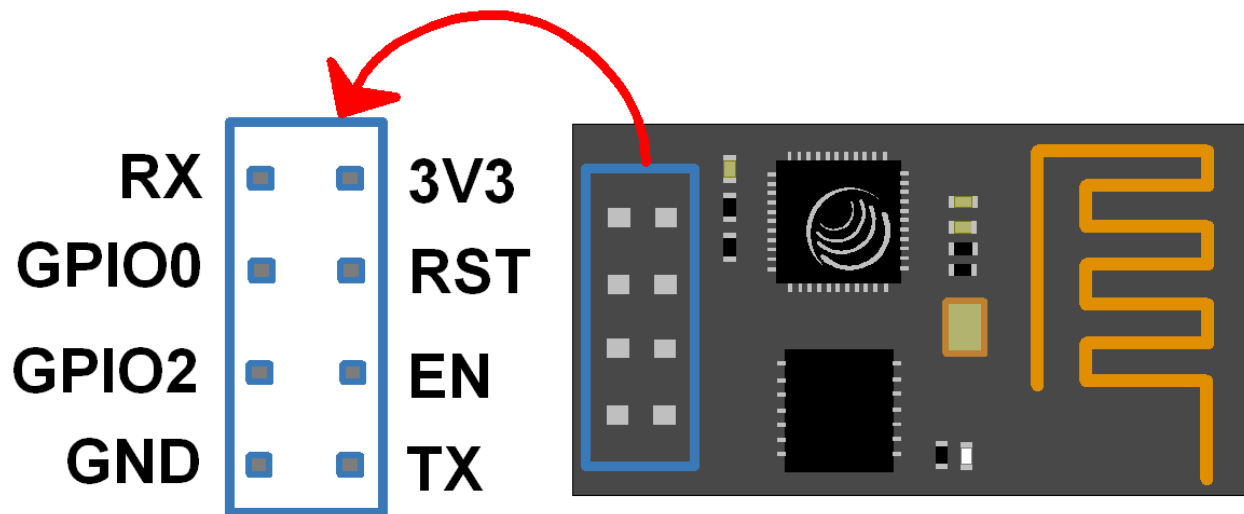
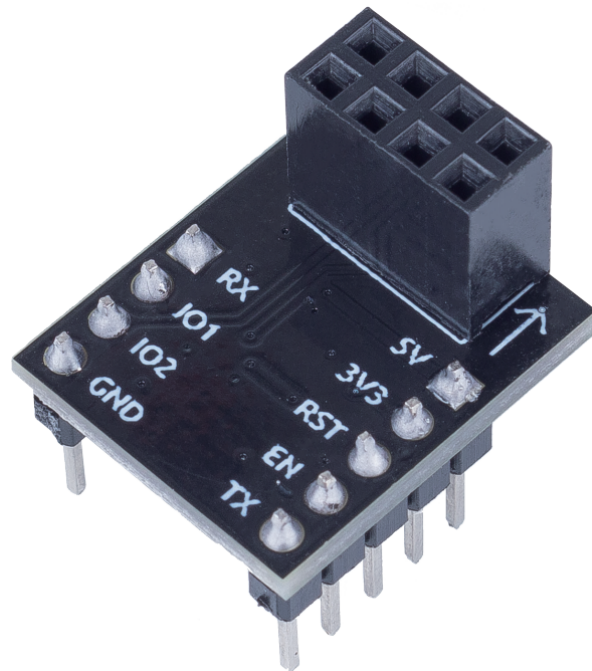


Table 1: ESP8266-01 Pins

Pin	Name	Description
1	TXD	UART_TXD, sending; General Purpose Input/Output: GPIO1; Pull-down is not allowed when startup.
2	GND	GND
3	CU_PD	Working at high level; Power off when low level is supplied.
4	GPIO2	It should be high level when power on, hardware pull-down is not allowed; Pull-up by default;
5	RST	External Reset signal, reset when low level is supplied; work when high level is supplied (high level by default);
6	GPIO0	WiFi Status indicator; Operation mode selection: Pull-up: Flash Boot, operation mode; Pull-down: UART Download, download mode
7	VCC	Power Supply(3.3V)
8	RXD	UART_RXDReceiving; General Purpose Input/Output: GPIO3;

- [ESP8266 - Espressif](#)
- [ESP8266 AT Instruction Set](#)

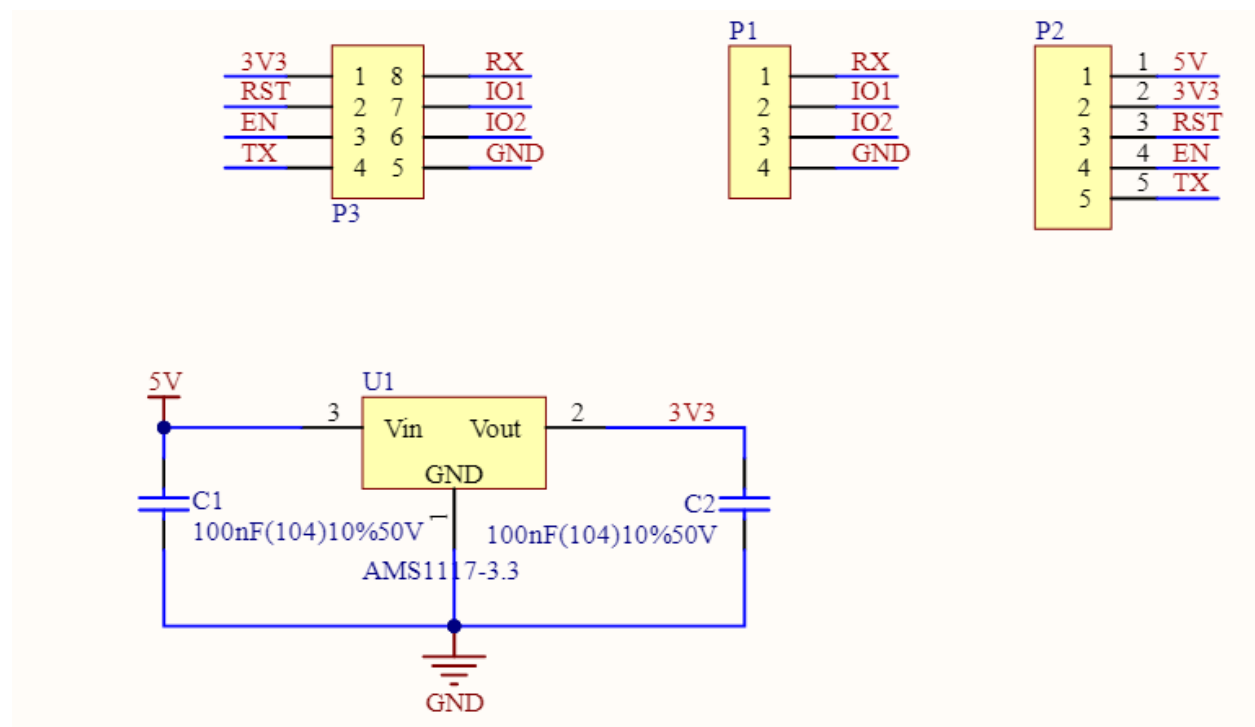
### 1.2.1 ESP8266 Adapter



The ESP8266 adapter is an expansion board that allows the ESP8266 module to be used on a breadboard.

It perfectly matches the pins of the ESP8266 itself, and also adds a 5V pin to receive the voltage from the Arduino board. The integrated AMS1117 chip is used to drive the ESP8266 module after dropping the voltage to 3.3V.

The schematic diagram is as follows:

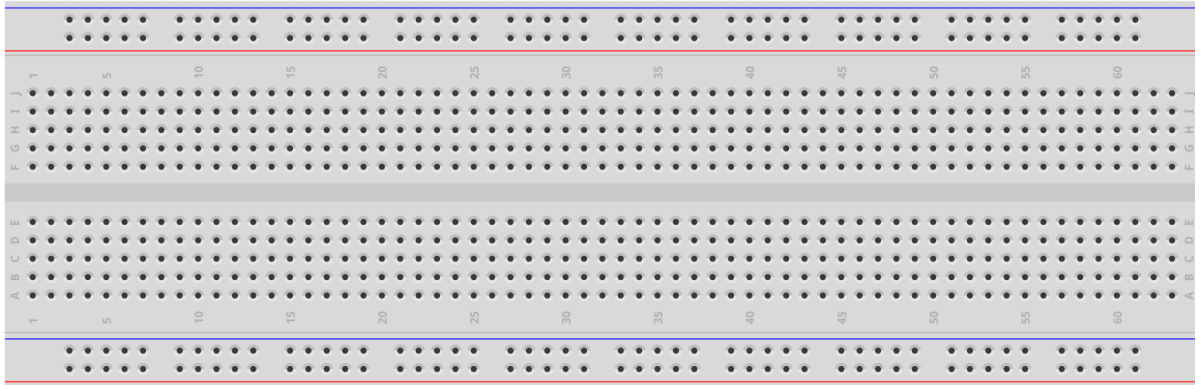


**Example**

- *IoT Projects* (IoT Project)

## Basic

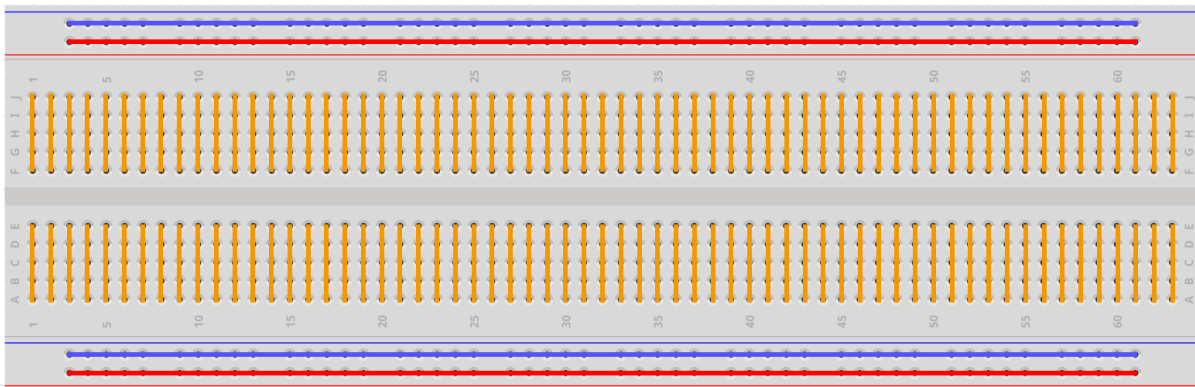
### 1.3 Breadboard



A breadboard is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread.[1] In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term “breadboard” is commonly used to refer to these.

It is used to build and test circuits quickly before finishing any circuit design. And it has many holes into which components mentioned above can be inserted like ICs and resistors as well as jumper wires. The breadboard allows you to plug in and remove components easily.

The picture shows the internal structure of a breadboard. Although these holes on the breadboard appear to be independent of each other, they are actually connected to each other through metal strips internally.



If you want to know more about breadboard, refer to: [How to Use a Breadboard - Science Buddies](#)

## 1.4 Resistor



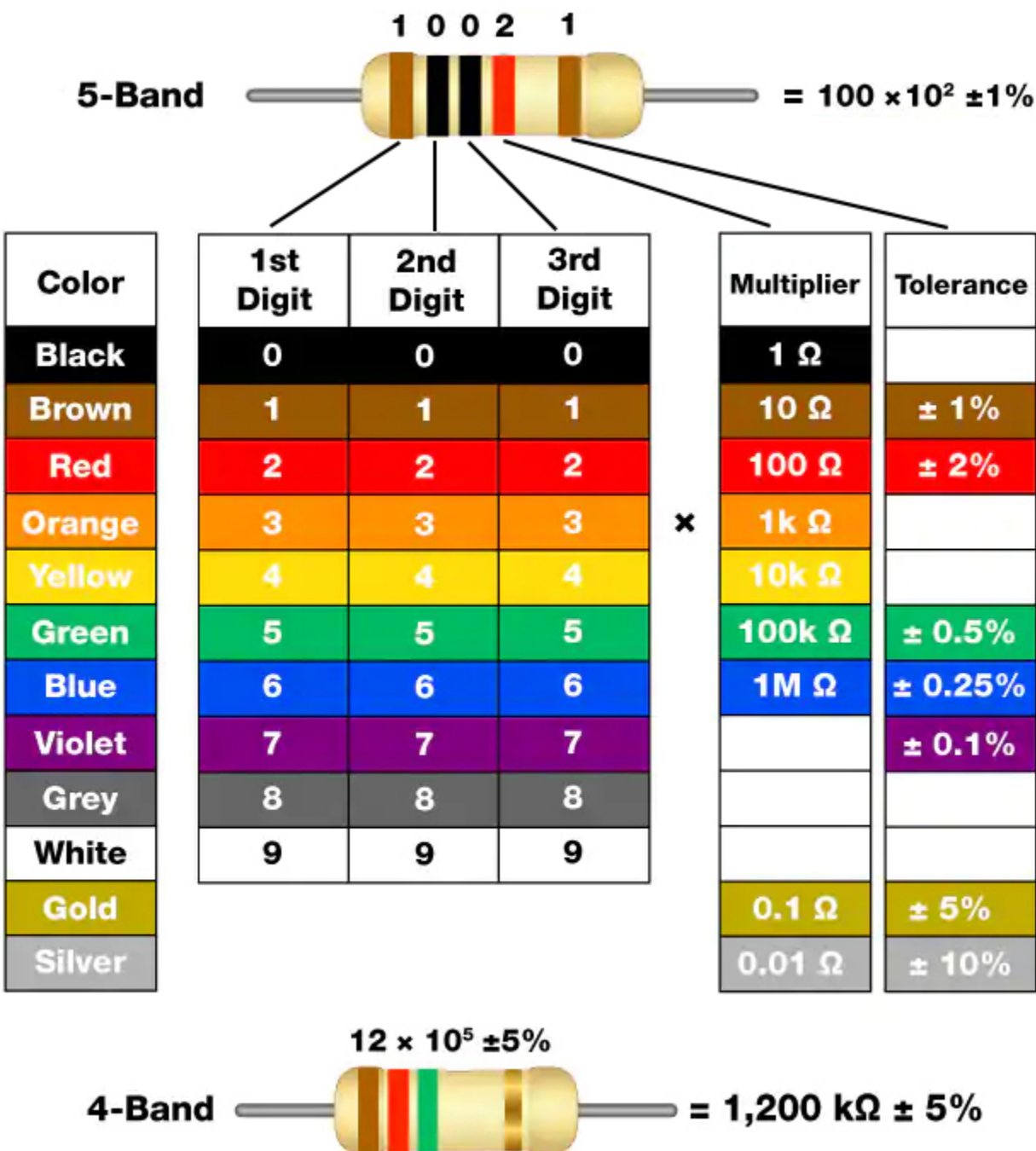
Resistor is an electronic element that can limit the branch current. A fixed resistor is a kind of resistor whose resistance cannot be changed, while that of a potentiometer or a variable resistor can be adjusted.

Two generally used circuit symbols for resistor. Normally, the resistance is marked on it. So if you see these symbols in a circuit, it stands for a resistor.



is the unit of resistance and the larger units include K, M, etc. Their relationship can be shown as follows: 1 M=1000 K, 1 K = 1000 . Normally, the value of resistance is marked on it.

When using a resistor, we need to know its resistance first. Here are two methods: you can observe the bands on the resistor, or use a multimeter to measure the resistance. You are recommended to use the first method as it is more convenient and faster.



As shown in the card, each color stands for a number.

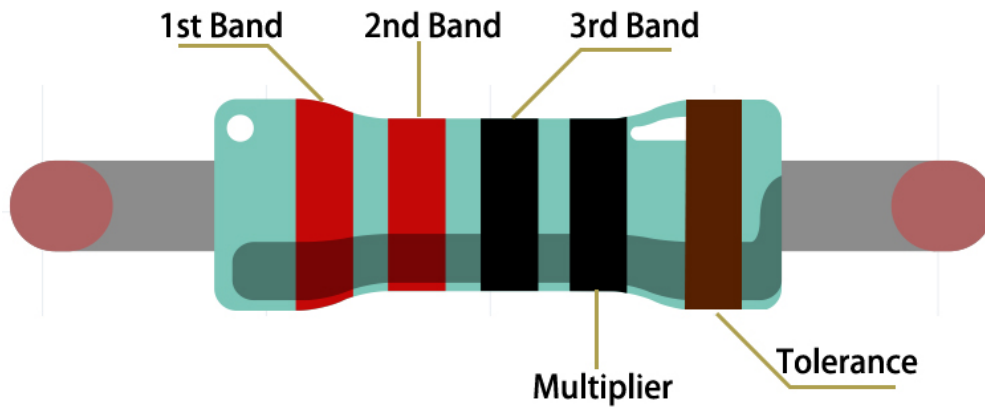
Black	Brown	Red	Orange	Yellow	Green	Blue	Violet	Grey	White	Gold	Silver
0	1	2	3	4	5	6	7	8	9	0.1	0.01

The 4- and 5-band resistors are frequently used, on which there are 4 and 5 chromatic bands.

Normally, when you get a resistor, you may find it hard to decide which end to start for reading the color. The tip is that the gap between the 4th and 5th band will be comparatively larger.

Therefore, you can observe the gap between the two chromatic bands at one end of the resistor; if it's larger than any other band gaps, then you can read from the opposite side.

Let's see how to read the resistance value of a 5-band resistor as shown below.



So for this resistor, the resistance should be read from left to right. The value should be in this format: 1st Band 2nd Band 3rd Band  $\times 10^{\text{Multiplier}}$  and the permissible error is  $\pm \text{Tolerance}\%$ . So the resistance value of this resistor is 2(red) 2(red) 0(black)  $\times 10^0$ (black) = 220 , and the permissible error is  $\pm 1\%$  (brown).

You can learn more about resistor from Wiki: [Resistor - Wikipedia](#).

## 1.5 Capacitor







Capacitor, refers to the amount of charge storage under a given potential difference, denoted as  $C$ , and the international unit is farad (F). Generally speaking, electric charges move under force in an electric field. When there is a medium between conductors, the movement of electric charges is hindered and the electric charges accumulate on the conductors, resulting in accumulation of electric charges.

The amount of stored electric charges is called capacitance. Because capacitors are one of the most widely used electronic components in electronic equipment, they are widely used in direct current isolation, coupling, bypass, filtering, tuning loops, energy conversion, and control circuits. Capacitors are divided into electrolytic capacitors, solid capacitors, etc.

According to material characteristics, capacitors can be divided into: aluminum electrolytic capacitors, film capacitors, tantalum capacitors, ceramic capacitors, super capacitors, etc.

In this kit, ceramic capacitors and electrolytic capacitors are used.

- [Ceramic Capacitor - Wikipedia](#)
- [Electrolytic Capacitor - Wikipedia](#)

There are 103 or 104 label on the ceramic capacitors, which represent the capacitance value, 103= $10 \times 10^3 \text{pF}$ , 104= $10 \times 10^4 \text{pF}$

#### Unit Conversion

$$1\text{F}=10^3\text{mF}=10^6\mu\text{F}=10^9\text{nF}=10^{12}\text{pF}$$

## 1.6 Jumper Wires

Wires that connect two terminals are called jumper wires. There are various kinds of jumper wires. Here we focus on those used in breadboard. Among others, they are used to transfer electrical signals from anywhere on the breadboard to the input/output pins of a microcontroller.

Jump wires are fitted by inserting their “end connectors” into the slots provided in the breadboard, beneath whose surface there are a few sets of parallel plates that connect the slots in groups of rows or columns depending on the area. The “end connectors” are inserted into the breadboard, without soldering, in the particular slots that need to be connected in the specific prototype.

There are three types of jumper wire: Female-to-Female, Male-to-Male, and Male-to-Female. The reason we call it Male-to-Female is because it has the outstanding tip in one end as well as a sunk female end. Male-to-Male means both side are male and Female-to-Female means both ends are female.

Male-to-Female



Male-to-Male



Female-to-Female



More than one type of them may be used in a project. The color of the jump wires is different but it doesn't mean their function is different accordingly; it's just designed so to better identify the connection between each circuit.

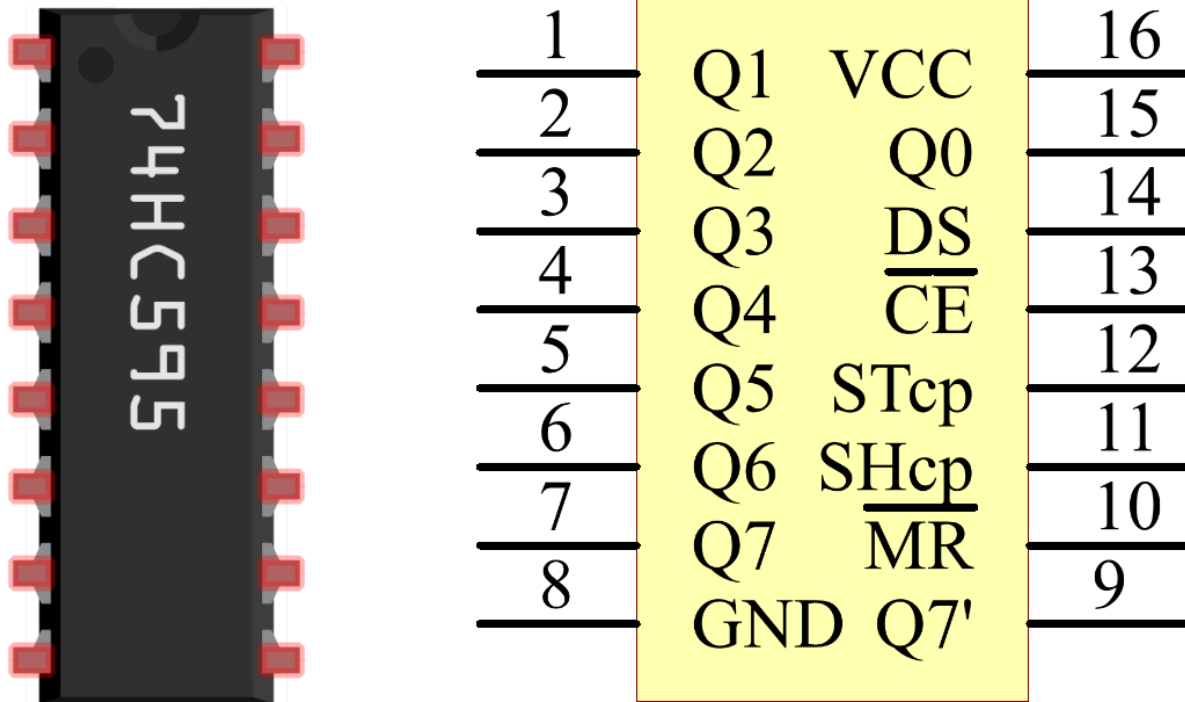
### Chip

## 1.7 74HC595



The 74HC595 consists of an 8bit shift register and a storage register with threestate parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU. When MR (pin10) is high level and OE (pin13) is low level, data is input in the rising edge of SHcp and goes to the memory register through the rising edge of SHcp. If the two clocks are connected together, the shift register is always one pulse earlier than the memory register. There is a serial shift input pin (Ds), a serial output pin (Q) and an asynchronous reset button (low level) in the memory register. The memory register outputs a Bus with a parallel 8-bit and in three states. When OE is enabled (low level), the data in memory register is output to the bus.

- [74HC595 Datasheet](#)



Pins of 74HC595 and their functions:

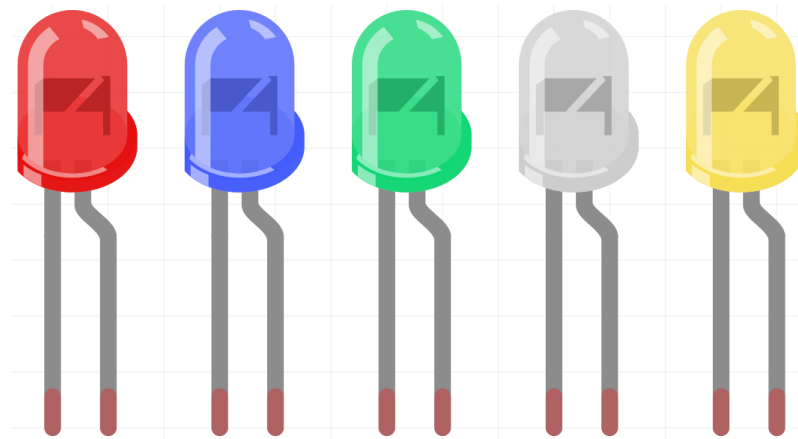
- **Q0-Q7**: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.
- **Q7'**: Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series
- **MR**: Reset pin, active at low level;
- **SHcp**: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.
- **STcp**: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.
- **CE**: Output enable pin, active at low level.
- **DS**: Serial data input pin
- **VCC**: Positive supply voltage.
- **GND**: Ground.

#### Example

- [5.9 ShiftOut\(LED\)](#) (Basic Project)
- [5.10 ShiftOut\(Segment Display\)](#) (Basic Project)
- [7. Current Limiting Gate](#) (IoT Project)

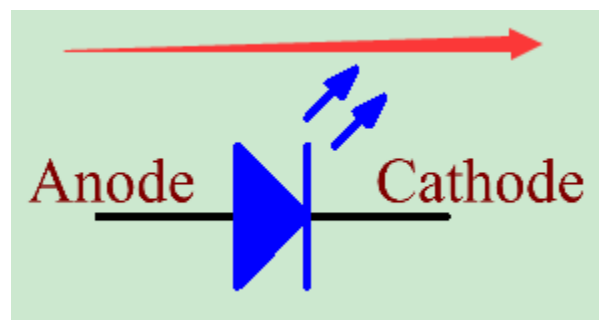
#### Display

## 1.8 LED



Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. By wavelength, it can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which is usually known as light-emitting diode (LED).

Diode has unidirectional conductivity, so the current flow will be as the arrow indicates in figure circuit symbol. You can only provide the anode with a positive power and the cathode with a negative. Thus the LED will light up.



An LED has two pins. The longer one is the anode, and shorter one, the cathode. Pay attention not to connect them inversely. There is fixed forward voltage drop in the LED, so it cannot be connected with the circuit directly because the supply voltage can outweigh this drop and cause the LED to be burnt. The forward voltage of the red, yellow, and green LED is 1.8 V and that of the white one is 2.6 V. Most LEDs can withstand a maximum current of 20 mA, so we need to connect a current limiting resistor in series.

The formula of the resistance value is as follows:

$$R = (V_{\text{supply}} - V_D) / I$$

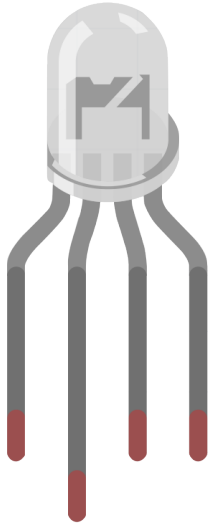
**R** stands for the resistance value of the current limiting resistor, **V<sub>supply</sub>** for voltage supply, **V<sub>D</sub>** for voltage drop and **I** for the working current of the LED.

Here is the detailed introduction for the LED: [LED - Wikipedia](#).

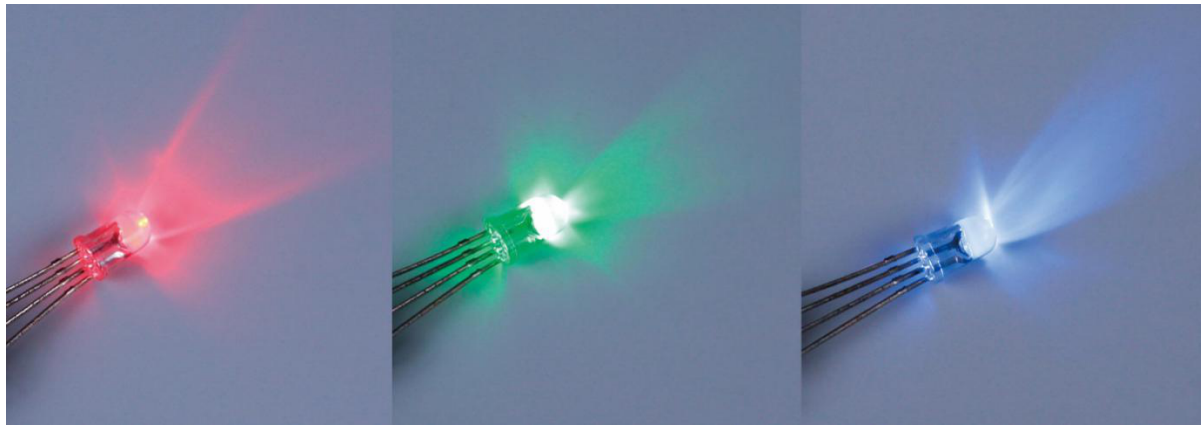
### Example

- [1.1 Hello, LED!](#) (Basic Project)
- [2.1 Fading](#) (Basic Project)
- [2. Get Data from Blynk](#) (IoT Project)

## 1.9 RGB LED

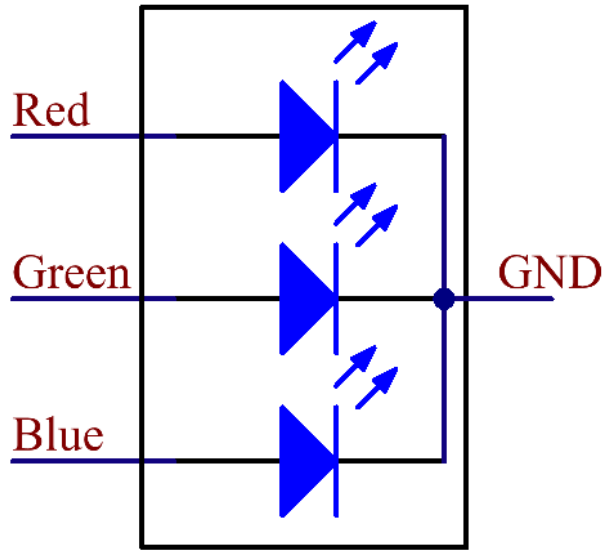


RGB LEDs emit light in various colors. An RGB LED packages three LEDs of red, green, and blue into a transparent or semitransparent plastic shell. It can display various colors by changing the input voltage of the three pins and superimpose them, which, according to statistics, can create 16,777,216 different colors.

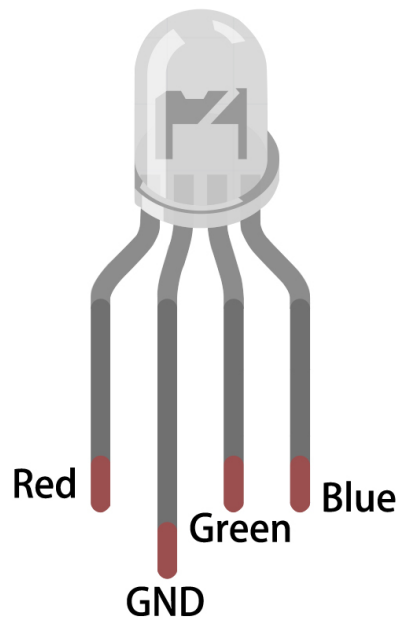


RGB LEDs can be categorized into common anode and common cathode ones. In this kit, the latter is used. The **common cathode**, or CC, means to connect the cathodes of the three LEDs. After you connect it with GND and plug in the three pins, the LED will flash the corresponding color.

Its circuit symbol is shown as figure.



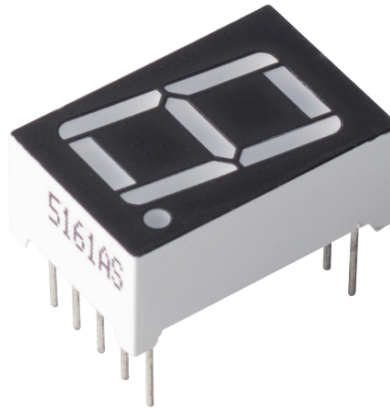
An RGB LED has 4 pins: the longest one is GND; the others are Red, Green and Blue. Touch its plastic shell and you will find a cut. The pin closest to the cut is the first pin, marked as Red, then GND, Green and Blue in turn.



#### Example

- [2.2 Colorful Light](#) (Basic Project)
- [5.2 Threshold](#) (Basic Project)

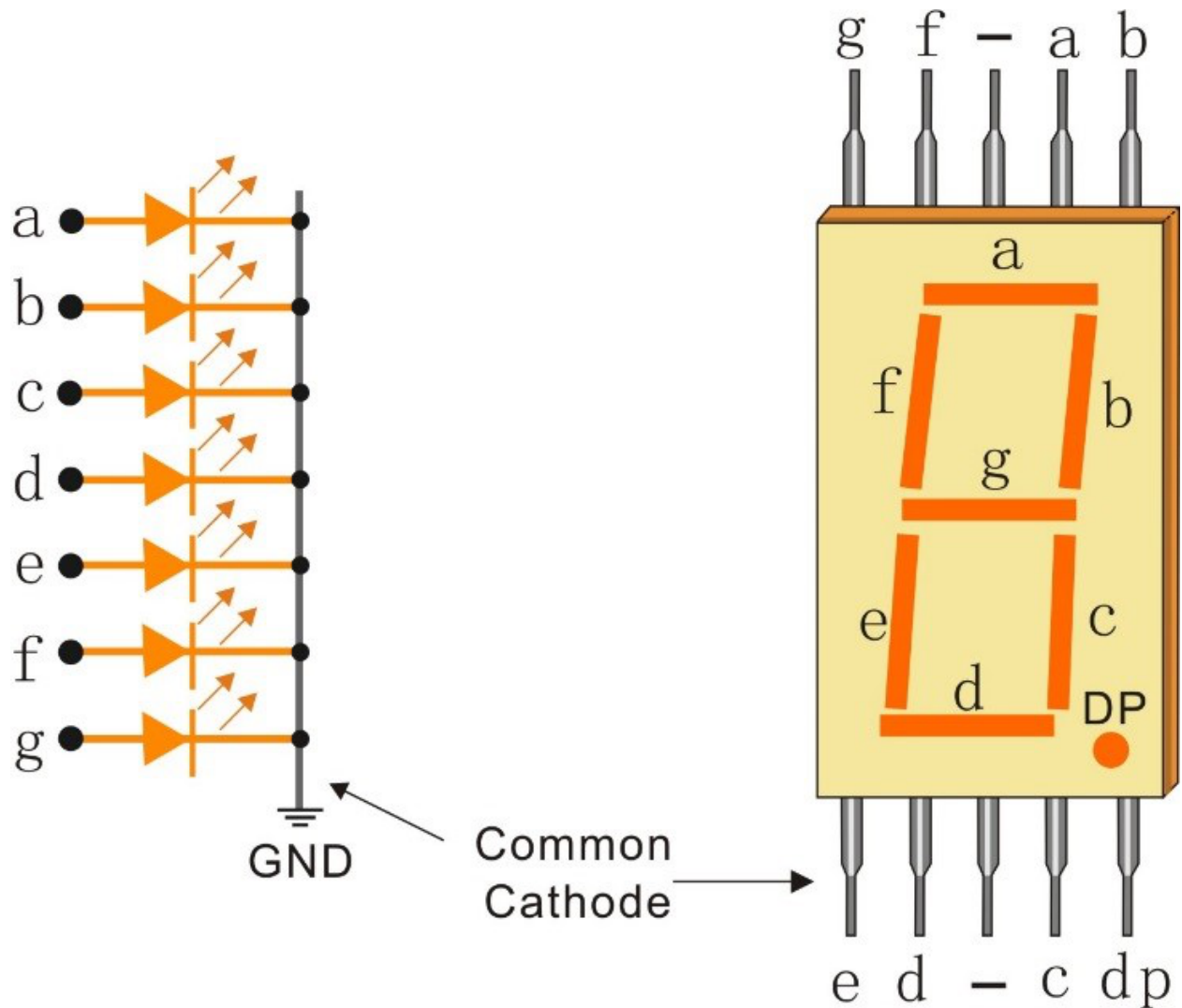
## 1.10 7-segment Display



A 7-segment display is an 8-shaped component which packages 7 LEDs. Each LED is called a segment - when energized, one segment forms part of a numeral to be displayed.

There are two types of pin connection: Common Cathode (CC) and Common Anode (CA). As the name suggests, a CC display has all the cathodes of the 7 LEDs connected when a CA display has all the anodes of the 7 segments connected.

In this kit, we use the Common Cathode 7-segment display, here is the electronic symbol.



Each of the LEDs in the display is given a positional segment with one of its connection pins led out from the rectangular plastic package. These LED pins are labeled from “a” through to “g” representing each individual LED. The other LED pins are connected together forming a common pin. So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will brighten and others stay dim, thus showing the corresponding character on the display.

### Display Codes

To help you get to know how 7-segment displays(Common Cathode) display Numbers, we have drawn the following table. Numbers are the number 0-F displayed on the 7-segment display; (DP) GFEDCBA refers to the corresponding LED set to 0 or 1, For example, 00111111 means that DP and G are set to 0, while others are set to 1. Therefore, the number 0 is displayed on the 7-segment display, while HEX Code corresponds to hexadecimal number.

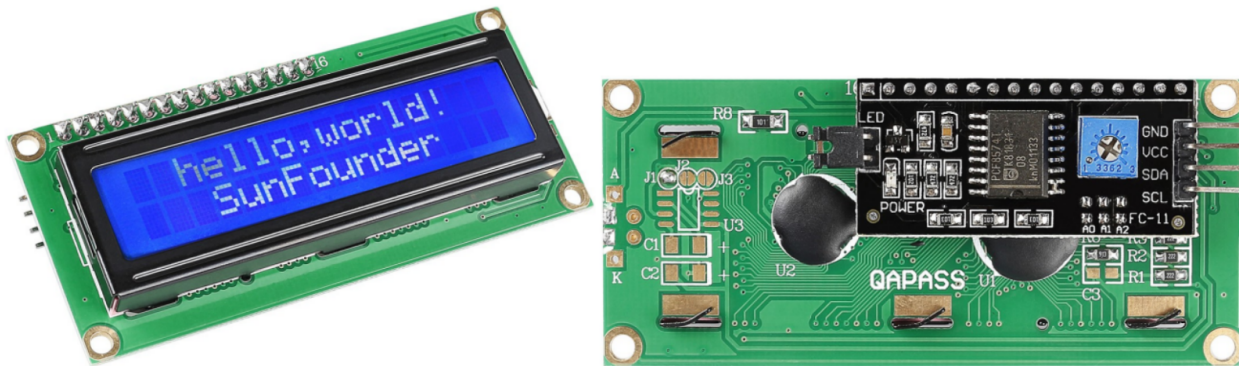


Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA A	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00000110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00000111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

**Example**

- *5.15 EEPROM* (Basic Project)
- *7. Current Limiting Gate* (IoT Project)

## 1.11 I2C LCD1602



- **GND:** Ground
- **VCC:** Voltage supply, 5V.
- **SDA:** Serial data line. Connect to VCC through a pullup resistor.
- **SCL:** Serial clock line. Connect to VCC through a pullup resistor.

As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller.

Therefore, LCD1602 with an I2C module is developed to solve the problem. The I2C module has a built-in PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display.

- [PCF8574 Datasheet](#)

### I2C Address

The default address is basically 0x27, in a few cases it may be 0x3F.

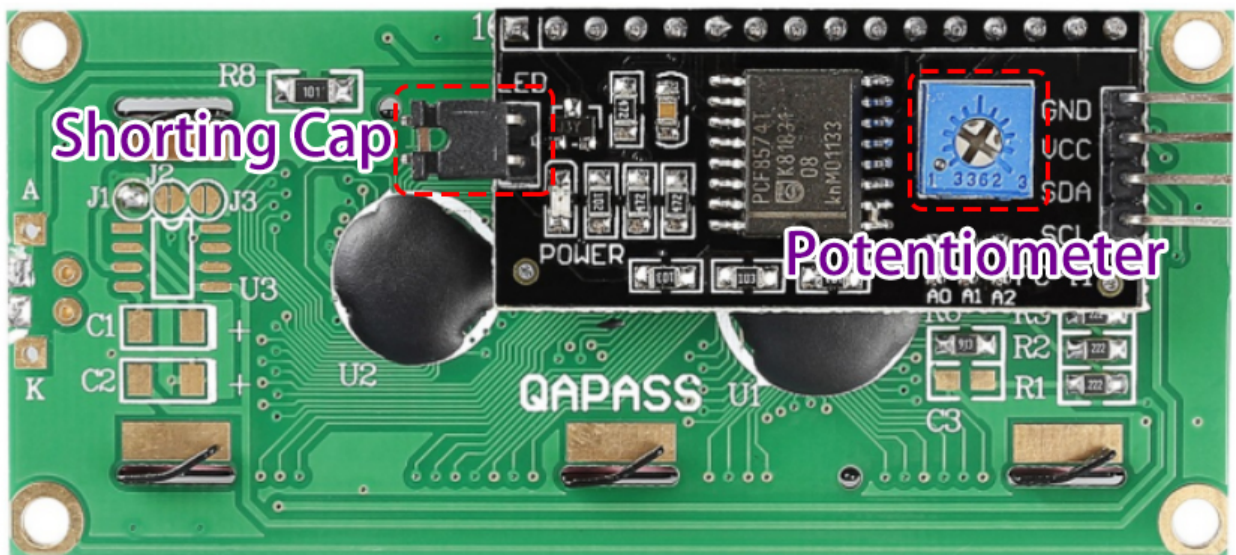
Taking the default address of 0x27 as an example, the device address can be modified by shorting the A0/A1/A2 pads; in the default state, A0/A1/A2 is 1, and if the pad is shorted, A0/A1/A2 is 0.

## Slave Address

0	0	1	0	0	A2	A1	A0	
0	0	1	0	0	1	1	1	0x27
0	0	1	0	0	1	1	0	0x26
0	0	1	0	0	1	0	1	0x25
0	0	1	0	0	0	1	1	0x23
.....								
0	0	1	0	0	0	0	0	0x20

### Backlight/Contrast

Backlight can be enabled by jumper cap, unplug the jumper cap to disable the backlight. The blue potentiometer on the back is used to adjust the contrast (the ratio of brightness between the brightest white and the darkest black).



- **Shorting Cap:** Backlight can be enabled by this cap, unplug this cap to disable the backlight.
- **Potentiometer:** It is used to adjust the contrast (the clarity of the displayed text), which is increased in the clockwise direction and decreased in the counterclockwise direction.

**Example**

- *5.11.1 Liquid Crystal Display* (Basic Project)
- *5.12 Serial Read* (Basic Project)

**Sound**

## 1.12 Buzzer



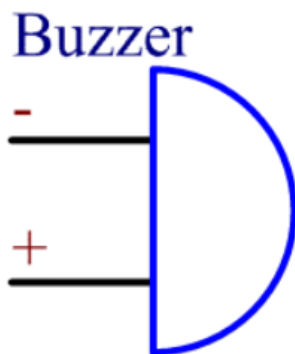
As a type of electronic buzzer with an integrated structure, buzzers, which are supplied by DC power, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products or voice devices.

Buzzers can be categorized as active and passive ones (see the following picture). Turn the buzzer so that its pins are facing up, and the buzzer with a green circuit board is a passive buzzer, while the one enclosed with a black tape is an active one.

The difference between an active buzzer and a passive buzzer:

An active buzzer has a built-in oscillating source, so it will make sounds when electrified. But a passive buzzer does not have such source, so it will not beep if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

The following is the electrical symbol of a buzzer. It has two pins with positive and negative poles. With a + in the surface represents the anode and the other is the cathode.



You can check the pins of the buzzer, the longer one is the anode and the shorter one is the cathode. Please don't mix them up when connecting, otherwise the buzzer will not make sound.

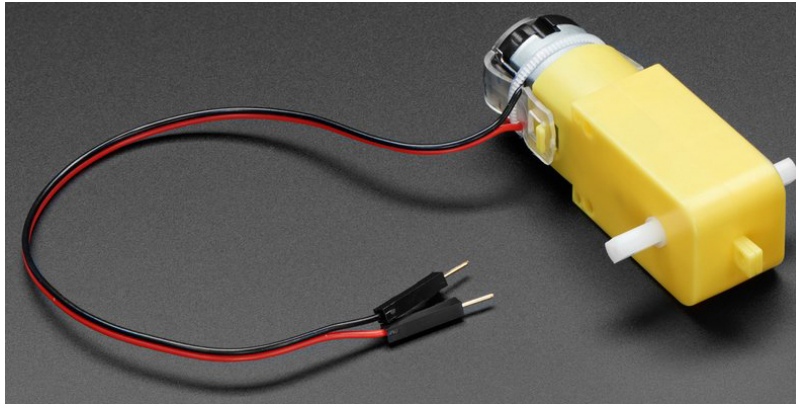
Buzzer - Wikipedia

### Example

- *1.2 Beep* (Basic Project)
- *5.7 Tone() or noTone()* (Basic Project)
- *4. Cloud Music Player* (IoT Project)

### Driver

## 1.13 TT Motor



This is a TT DC gearbox motor with a gear ratio of 1:48, it comes with 2 x 200mm wires with 0.1" male connectors that fit into a breadboard. Perfect for plugging into a breadboard or terminal block.

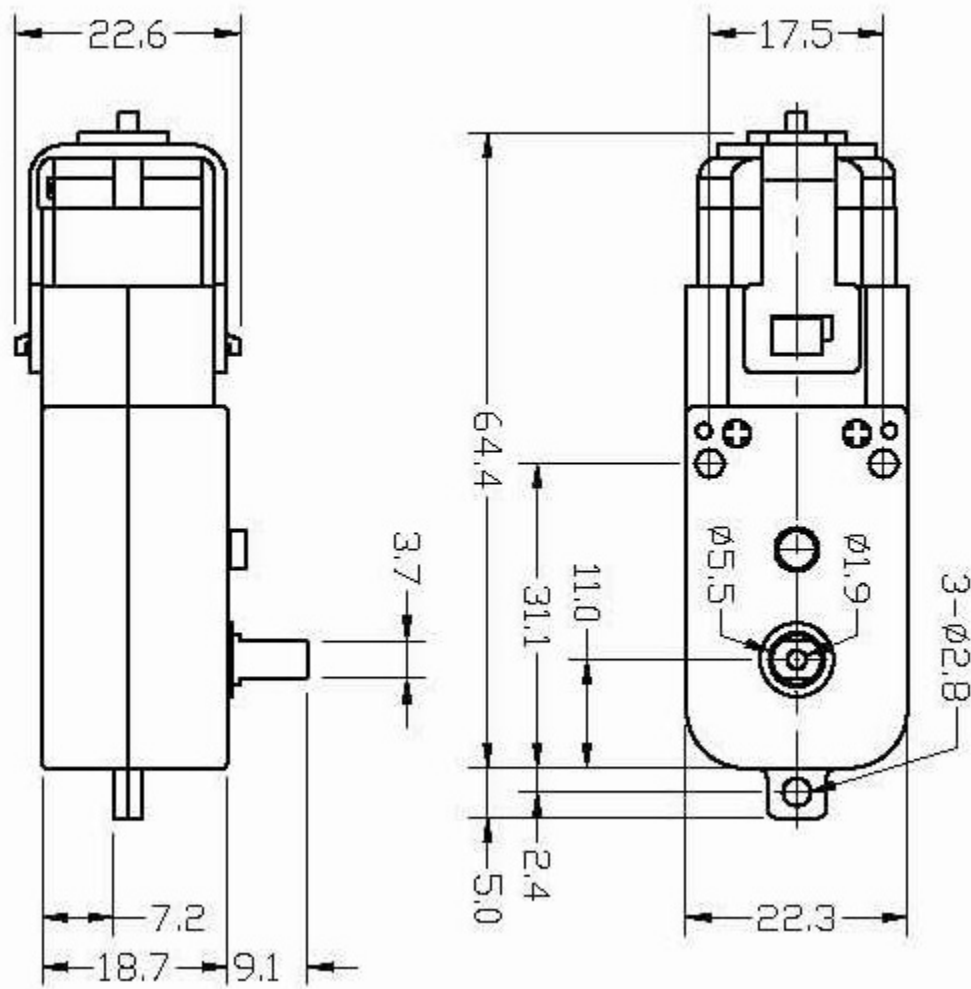
You can power these motors with 3 ~ 6VDC, but of course, they will go a little faster at higher voltages.

Note that these are very basic motors with no built-in encoder, speed control or position feedback. The voltage goes in and the spin comes out. There will be variation from motor to motor, so if you need precise motion, you'll need a separate feedback system.

### Technical Details

- Rated Voltage: 3~6V
- Continuous No-Load Current: 150mA +/- 10%
- Min. Operating Speed (3V): 90+/- 10% RPM
- Min. Operating Speed (6V): 200+/- 10% RPM
- Stall Torque (3V): 0.4kg.cm
- Stall Torque (6V): 0.8kg.cm
- Gear Ratio: 1:48
- Body Dimensions: 70 x 22 x 18mm
- Wires Length: 200mm & 28 AWG
- Weight: 30.6g

### Dimensional Drawing

**Example**

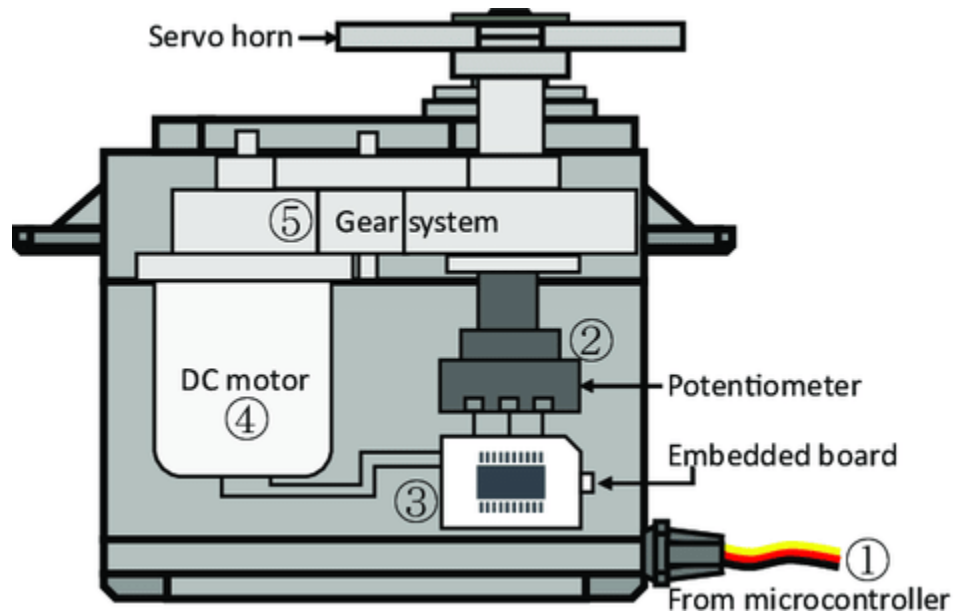
- *1.3 Turn the Wheel* (Basic Project)
- *1. Move* (Car Project)
- *3. Speed Up* (Car Project)
- *8. IoT Car* (IoT Project)

**1.14 Servo**

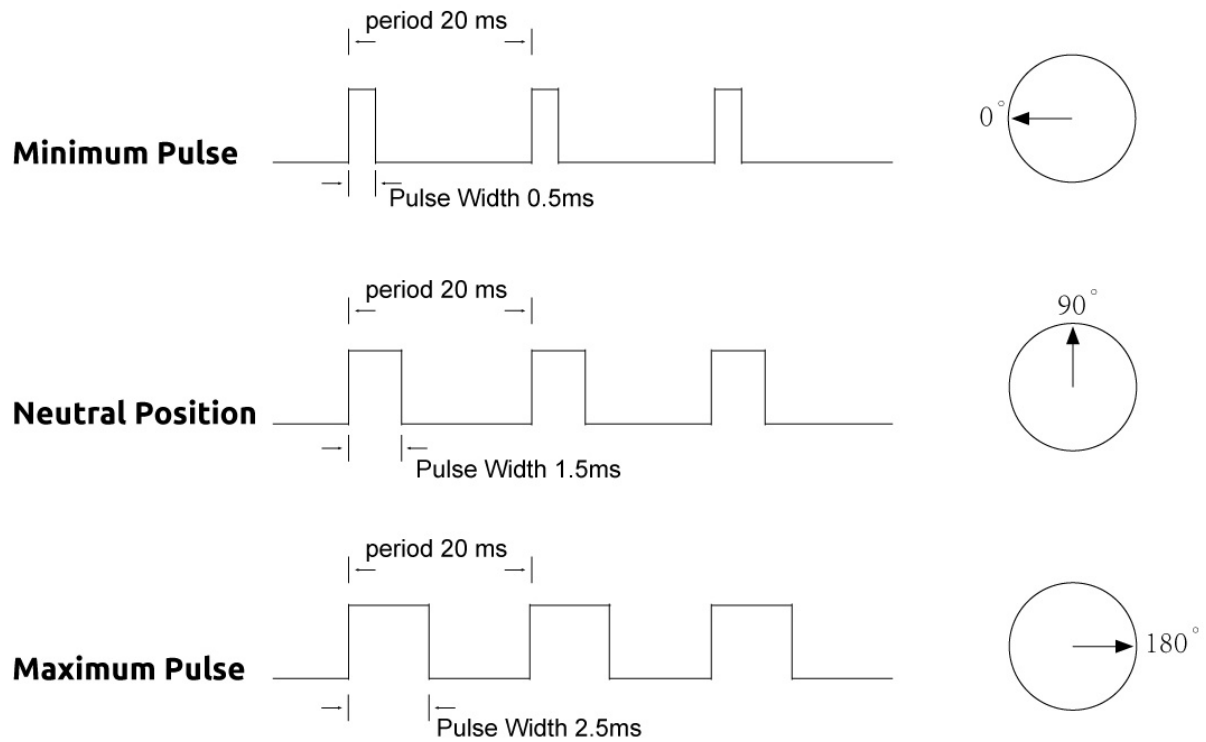
A servo is generally composed of the following parts: case, shaft, gear system, potentiometer, DC motor, and embedded board.



It works like this: The microcontroller sends out PWM signals to the servo, and then the embedded board in the servo receives the signals through the signal pin and controls the motor inside to turn. As a result, the motor drives the gear system and then motivates the shaft after deceleration. The shaft and potentiometer of the servo are connected together. When the shaft rotates, it drives the potentiometer, so the potentiometer outputs a voltage signal to the embedded board. Then the board determines the direction and speed of rotation based on the current position, so it can stop exactly at the right position as defined and hold there.



The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse width Modulation. The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90 degree position (neutral position). When a pulse is sent to a servo that is less than 1.5 ms, the servo rotates to a position and holds its output shaft some number of degrees counterclockwise from the neutral point. When the pulse is wider than 1.5 ms the opposite occurs. The minimal width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo. Generally the minimum pulse will be about 0.5 ms wide and the maximum pulse will be 2.5 ms wide.

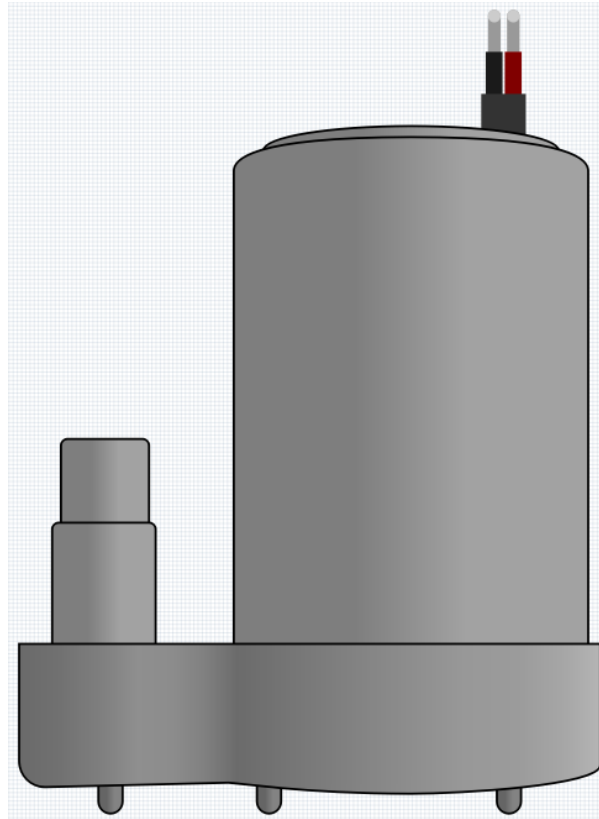


#### Example

- [5.5 Use Internal Library](#) (Basic Project)
- [7. Current Limiting Gate](#) (IoT Project)



## 1.15 Centrifugal Pump



The centrifugal pump converts rotational kinetic energy into hydrodynamic energy to transport fluid. The rotation energy comes from the electric motor. The fluid enters the pump impeller along or near the rotating shaft, is accelerated by the impeller, flows radially outward into the diffuser or volute chamber, and then flows out from there.

Common uses of centrifugal pumps include water, sewage, agricultural, petroleum, and petrochemical pumping.

- [Centrifugal Pump - Wikipedia](#)

### Features

- **Voltage Scope:** DC 3 ~ 4.5V
- **Operating Current:** 120 ~ 180mA
- **Power:** 0.36 ~ 0.91W
- **Max Water Head:** 0.35 ~ 0.55M
- **Max Flow Rate:** 80 ~ 100 L/H
- **Continuous Working Life:** 100 hours
- **Water Fing Grade:** IP68
- **Driving Mode:** DC, Magnetic Driving
- **Material:** Engineering Plastic
- **Outlet Outside Diameter:** 7.8 mm
- **Outlet Inside Diameter:** 6.5 mm

- It is a submersible pump and should be used that way. It tends to heat too much that there's a risk of overheating if you turn it on unsubmerged.

#### Example

- *1.4 Pumping* (Basic Project)
- *6. Plant Monitor* (IoT Project)

## 1.16 L9110 Motor Driver Module

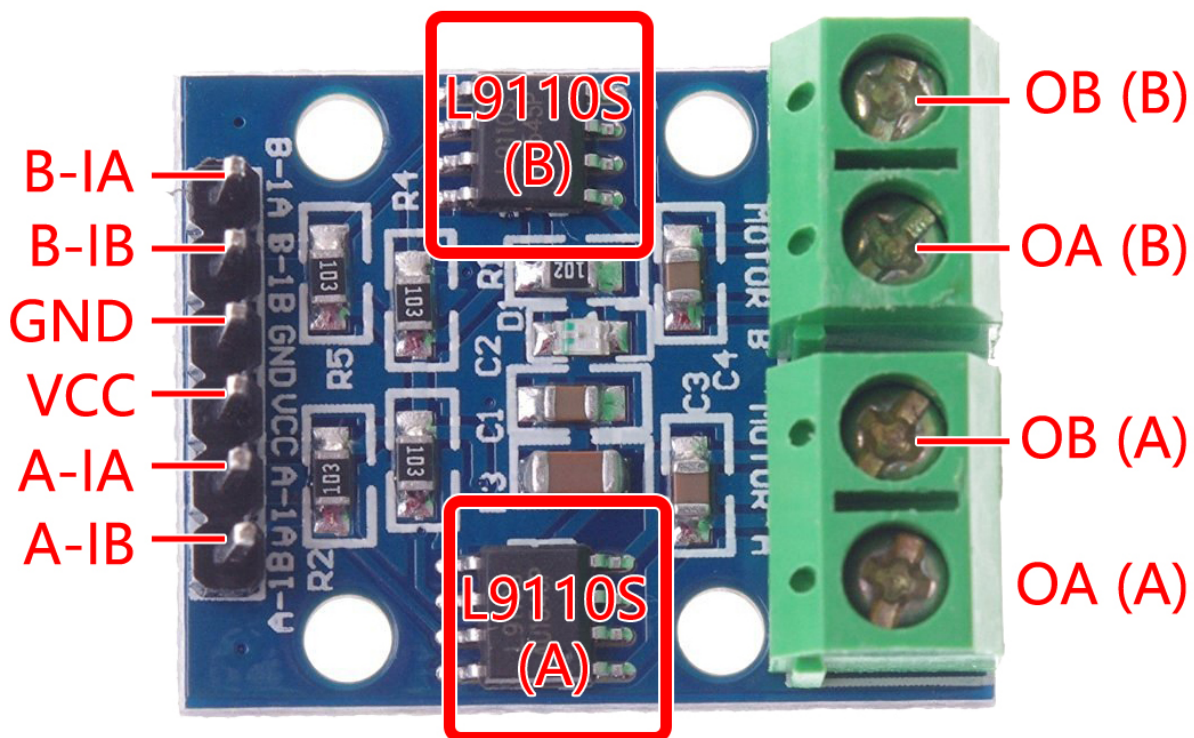
The L9110 motor driver module is adept at driving two motors in tandem. It houses a pair of independent L9110S driver chips, each channel boasting a steady current output of up to 800mA.

Spanning a voltage range from 2.5V to 12V, the module comfortably pairs with both 3.3V and 5V microcontrollers.

Serving as a streamlined solution, the L9110 motor driver module facilitates motor control across a spectrum of applications. Thanks to its dual-channel architecture, it enables the independent orchestration of two motors—ideal for projects where dual motor operations are paramount.

Given its potent continuous current output, this module confidently powers motors from the petite to the moderately sized, paving the way for diverse robotic, automation, and motor-centric endeavors. Its expansive voltage range further injects adaptability, aligning with varied power supply setups.

Designed with user-friendliness in mind, the module offers intuitive input and output terminals, simplifying connections to microcontrollers or akin control devices. Plus, it doesn't skimp on safety—integrated overcurrent and overtemperature safeguards bolster the trustworthiness and security of motor operations.



- **B-1A & B-1B(B-2A)**: Input pins for controlling the spinning direction of Motor B.
- **A-1A & A-1B**: Input pins for controlling the spinning direction of Motor A.
- **0A & OB(A)**: Output pins of Motor A.

- **0A & 0B(B)**: Output pins of Motor B.
- **VCC**: Power input pin (2.5V-12V).
- **GND**: Ground pin.

### Features

- On-board 2 L9110S motor control chip
- Dual-channel motor control.
- Independent motor spinning direction control.
- High current output (800mA per channel).
- Wide voltage range (2.5V-12V).
- Compact design.
- Convenient input and output terminals.
- Built-in protective features.
- Versatile applications.
- PCB Size: 29.2mm x 23mm
- Operating Temperature: -20°C ~ 80°C
- Power-On LED indicator

### Operating Principle

Here is the truth table of Motor B:

This truth table shows the different states of Motor B based on the values of input pins B-1A and B-1B(B-2A). It indicates the direction of rotation (clockwise or counterclockwise), braking, or stopping of Motor B.

B-1A	B-1B(B-2A)	The state of Motor B
1	0	Rotate clockwise
0	1	Rotate counterclockwise
0	0	Brake
1	1	Stop

Here is the truth table of Motor A:

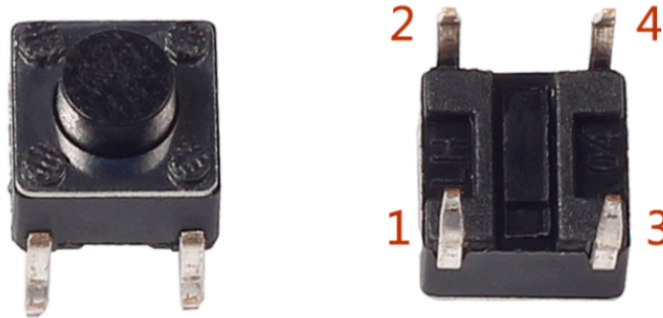
This truth table shows the different states of Motor A based on the values of input pins A-1A and A-1B. It indicates the direction of rotation (clockwise or counterclockwise), braking, or stopping of Motor A.

A-1A	A-1B	The state of Motor B
1	0	Rotate clockwise
0	1	Rotate counterclockwise
0	0	Brake
1	1	Stop

- [1.3 Turn the Wheel](#) (Basic Project)
- [1. Move](#) (Car Project)
- [3. Speed Up](#) (Car Project)
- [8. IoT Car](#) (IoT Project)

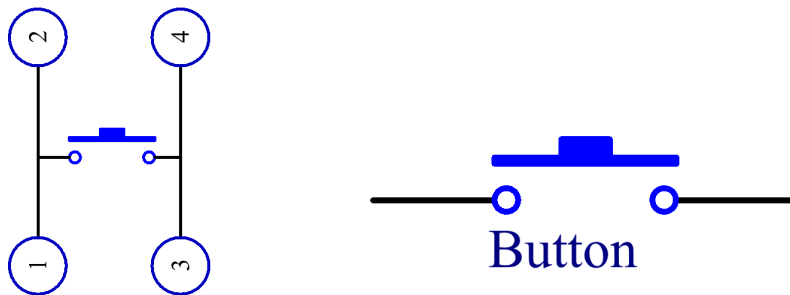
### Controller

## 1.17 Button

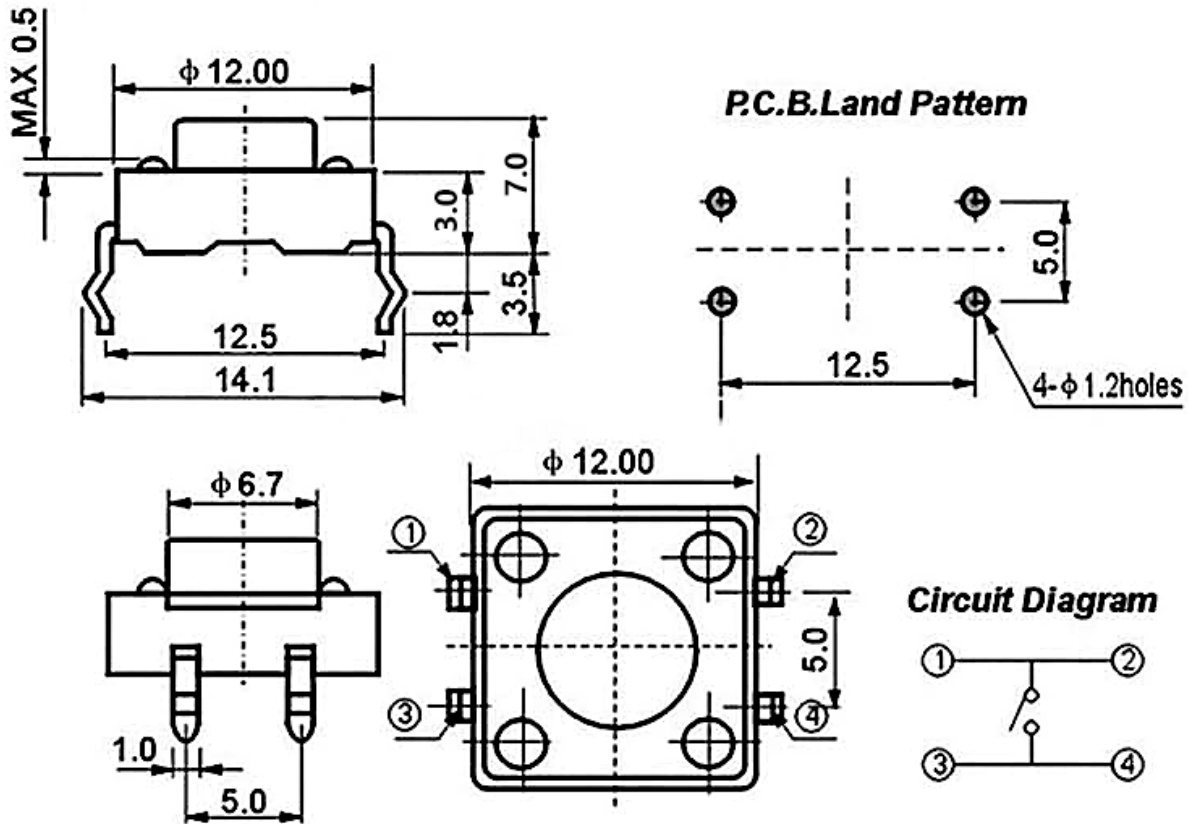


Buttons are a common component used to control electronic devices. They are usually used as switches to connect or break circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pin 1 is connected to pin 2 and pin 3 to pin 4. So you just need to connect either of pin 1 and pin 2 to pin 3 or pin 4.

The following is the internal structure of a button. The symbol on the right below is usually used to represent a button in circuits.



Since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.



### Example

- [3.1 Reading Button Value](#) (Basic Project)

## 1.18 Reed Switch

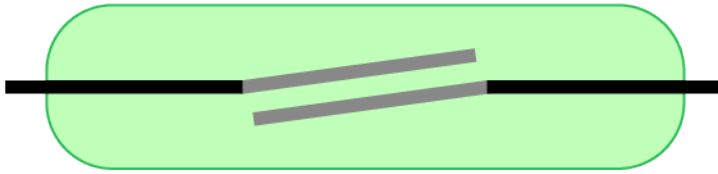


The Reed Switch is an electrical switch that operates by means of an applied magnetic field. It was invented by Walter B. Ellwood of Bell Telephone Laboratories in 1936 and patented in the United States on June 27, 1940, under patent number 2264746.

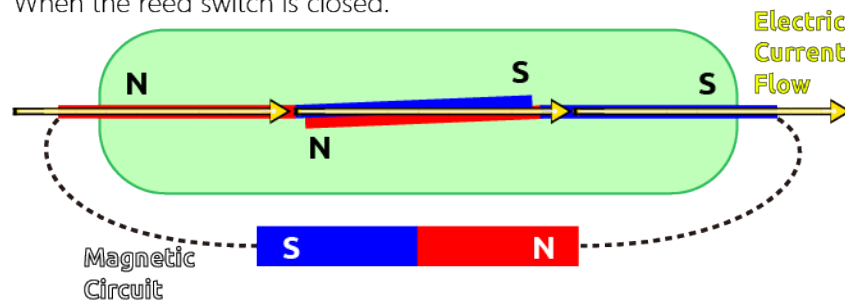
The principle of operation of a reed switch is very simple. Two reeds (usually made of iron and nickel, two metals) that overlap at the end points are sealed in a glass tube, with the two reeds overlapping and separated by a small gap (only about a few microns). The glass tube is filled with a high purity inert gas (such as nitrogen), and some reed switches are made to have a vacuum inside to enhance their high voltage performance.

The reed acts as a magnetic flux conductor. The two reeds are not in contact when not yet in operation; when passing through a magnetic field generated by a permanent magnet or electromagnetic coil, the applied magnetic field causes the two reeds to have different polarities near their endpoints, and when the magnetic force exceeds the spring force of the reeds themselves, the two reeds will be drawn together to conduct the circuit; when the magnetic field weakens or disappears, the reeds are released due to their own elasticity, and the contact surfaces will separate to open the circuit.

When the reed switch is open.



When the reed switch is closed.

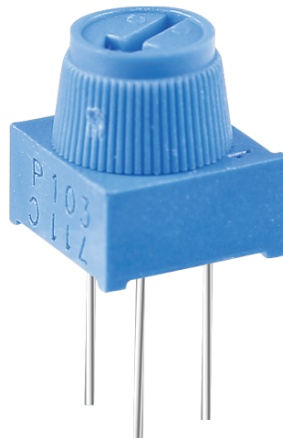


- [Reed Switch - Wikipedia](#)

#### Example

- [3.2 Feel the Magnetism](#) (Basic Project)
- [7. Current Limiting Gate](#) (IoT Project)

## 1.19 Potentiometer



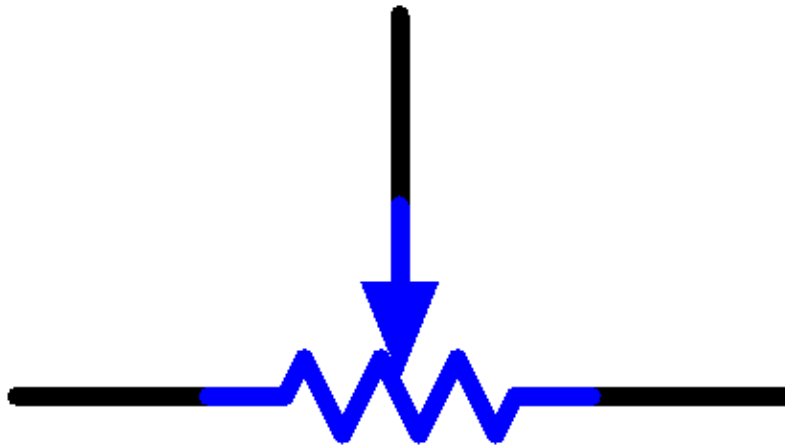
Potentiometer is also a resistance component with 3 terminals and its resistance value can be adjusted according to some regular variation.

Potentiometers come in various shapes, sizes, and values, but they all have the following things in common:

- They have three terminals (or connection points).
- They have a knob, screw, or slider that can be moved to vary the resistance between the middle terminal and either one of the outer terminals.

- The resistance between the middle terminal and either one of the outer terminals varies from 0 to the maximum resistance of the pot as the knob, screw, or slider is moved.

Here is the circuit symbol of potentiometer.



The functions of the potentiometer in the circuit are as follows:

1. Serving as a voltage divider

Potentiometer is a continuously adjustable resistor. When you adjust the shaft or sliding handle of the potentiometer, the movable contact will slide on the resistor. At this point, a voltage can be output depending on the voltage applied onto the potentiometer and the angle the movable arm has rotated to or the travel it has made.

2. Serving as a rheostat

When the potentiometer is used as a rheostat, connect the middle pin and one of the other 2 pins in the circuit. Thus you can get a smoothly and continuously changed resistance value within the travel of the moving contact.

3. Serving as a current controller

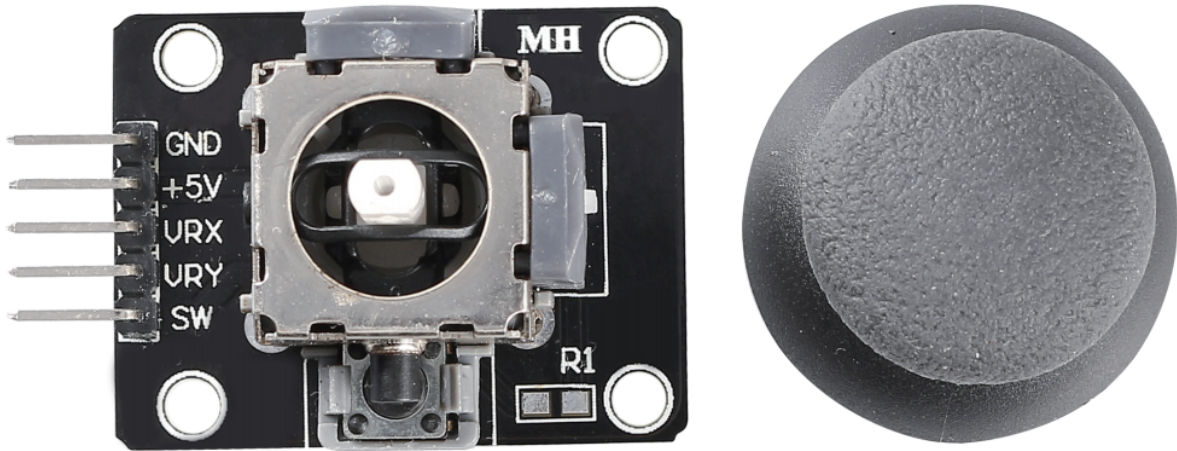
When the potentiometer acts as a current controller, the sliding contact terminal must be connected as one of the output terminals.

If you want to know more about potentiometer, refer to: [Potentiometer - Wikipedia](#)

### Example

- [4.1 Turn the Knob](#) (Basic Project)

## 1.20 Joystick Module



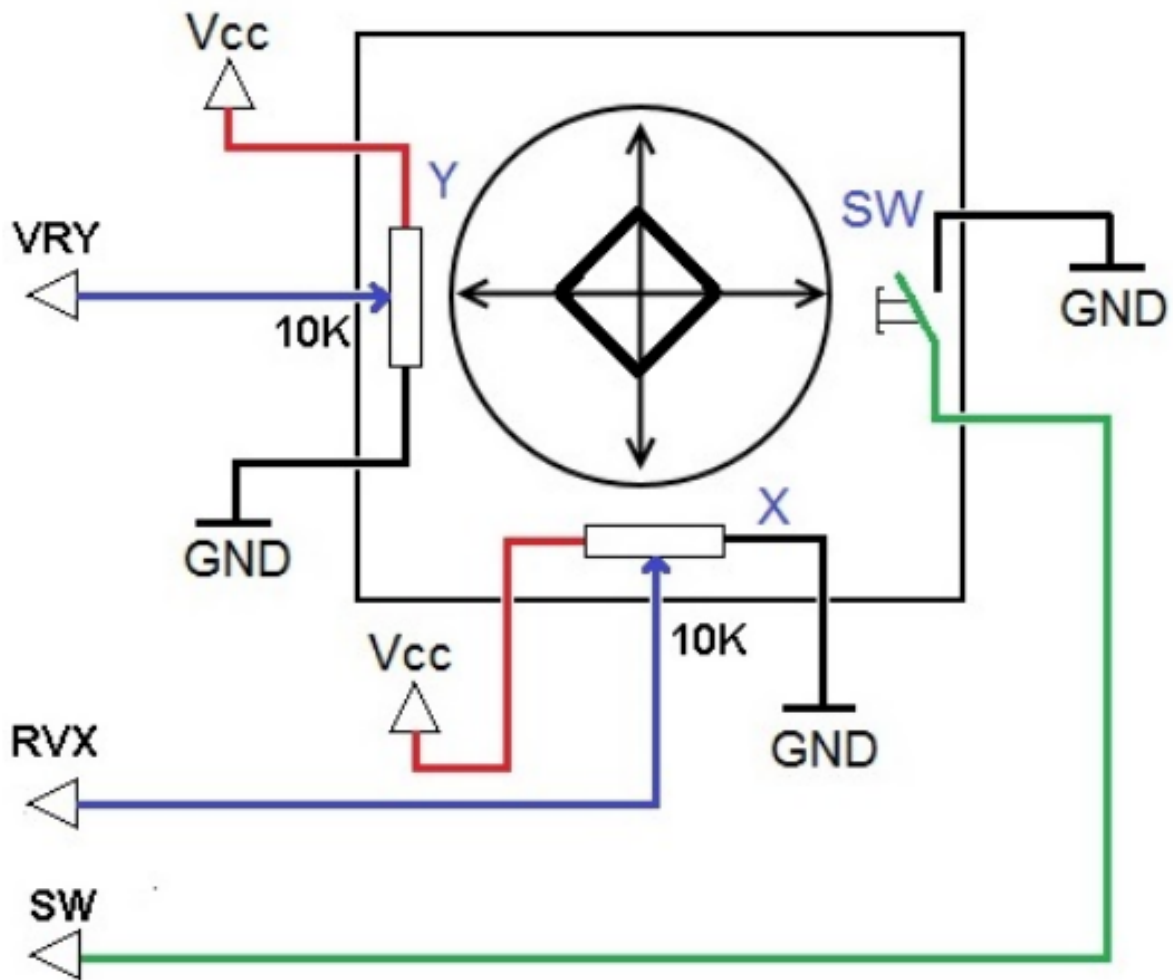
The basic idea of a joystick is to translate the movement of a stick into electronic information that a computer can process.

In order to communicate a full range of motion to the computer, a joystick needs to measure the stick's position on two axes – the X-axis (left to right) and the Y-axis (up and down). Just as in basic geometry, the X-Y coordinates pinpoint the stick's position exactly.

To determine the location of the stick, the joystick control system simply monitors the position of each shaft. The conventional analog joystick design does this with two potentiometers, or variable resistors.

The joystick also has a digital input that is actuated when the joystick is pressed down.



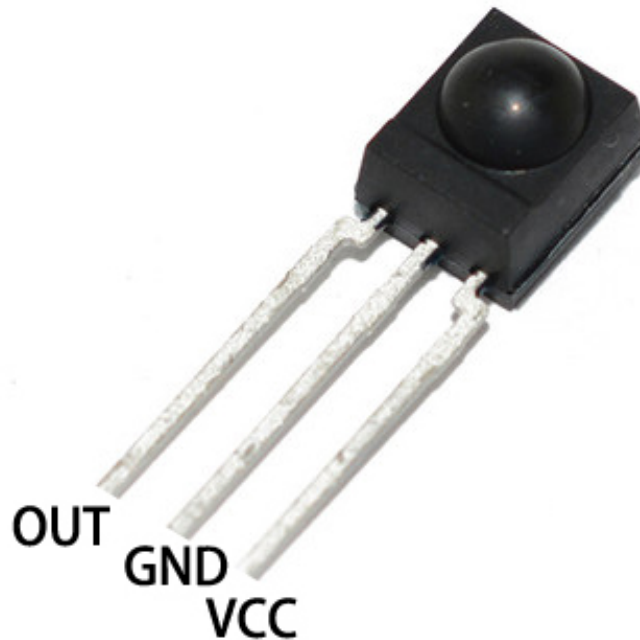


### Example

- [4.3 Toggle the Joystick](#) (Basic Project)

## 1.21 IR Receiver

### IR Receiver



- OUT: Signal output
- GND: GND
- VCC: power supply, 3.3v~5V

An infrared-receiver is a component which receives infrared signals and can independently receive infrared rays and output signals compatible with TTL level. It is similar with a normal plastic-packaged transistor in size and is suitable for all kinds of infrared remote control and infrared transmission.

Infrared, or IR, communication is a popular, low-cost, easy-to-use wireless communication technology. Infrared light has a slightly longer wavelength than visible light, so it is imperceptible to the human eye - ideal for wireless communication. A common modulation scheme for infrared communication is 38KHz modulation.

- Adopted [HS0038B](#) IR Receiver Sensor, high sensitivity
- Can be used for remote control
- Power Supply: 5V
- Interface: Digital
- Modulate Frequency: 38Khz
- Pin Definitions: (1) Output (2) Vcc (3) GND
- Size: 23.5mm x 21.5mm

### Remote Control



This is a Mini thin infrared wireless remote control with 21 function buttons and a transmitting distance of up to 8 meters, which is suitable for operating a wide range of devices in a kid's room.

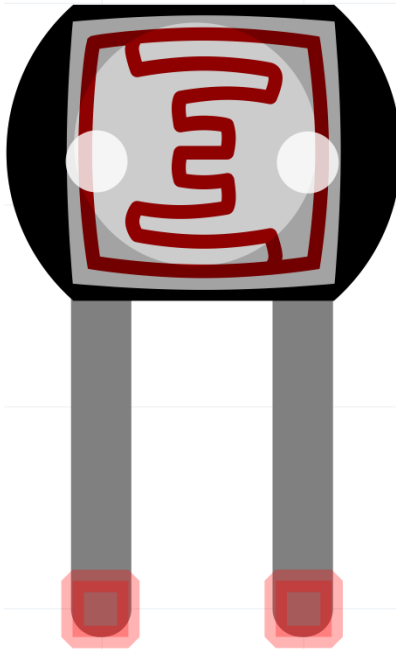
- Size: 85x39x6mm
- Remote control range: 8-10m
- Battery: 3V button type lithium manganese battery
- Infrared carrier frequency: 38KHz
- Surface paste material: 0.125mm PET
- Effective life: more than 20,000 times

#### Example

- [5.11.2 IR Receiver](#) (Basic Project)
- [9. Remote Control](#) (Car Project)
- [10. One Touch Start](#) (Car Project)

#### Sensor

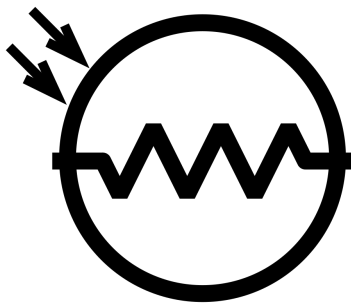
## 1.22 Photoresistor



A photoresistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photo conductivity.

A photoresistor can be applied in light-sensitive detector circuits and light-activated and dark-activated switching circuits acting as a resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megaohms (M), while in the light, a photoresistor can have a resistance as low as a few hundred ohms.

Here is the electronic symbol of photoresistor.

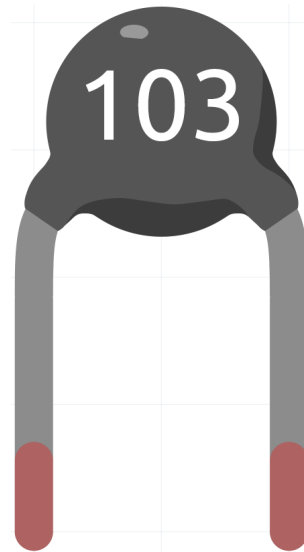


- [Photoresistor - Wikipedia](#)

### Example

- [4.2 Feel the Light](#) (Basic Project)
- [5. Home Environment Monitoring](#) (IoT Project)
- [6. Plant Monitor](#) (IoT Project)

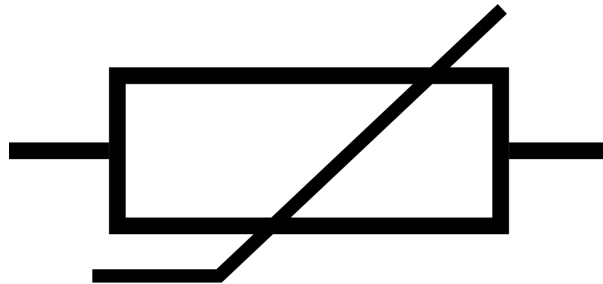
## 1.23 Thermistor



A thermistor is a type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors. The word is a combination of thermal and resistor. Thermistors are widely used as inrush current limiters, temperature sensors (negative temperature coefficient or NTC type typically), self-resetting overcurrent protectors, and self-regulating heating elements (positive temperature coefficient or PTC type typically).

- [Thermistor - Wikipedia](#)

Here is the electronic symbol of thermistor.



Thermistors are of two opposite fundamental types:

- With NTC thermistors, resistance decreases as temperature rises usually due to an increase in conduction electrons bumped up by thermal agitation from valency band. An NTC is commonly used as a temperature sensor, or in series with a circuit as an inrush current limiter.
- With PTC thermistors, resistance increases as temperature rises usually due to increased thermal lattice agitations particularly those of impurities and imperfections. PTC thermistors are commonly installed in series with a circuit, and used to protect against overcurrent conditions, as resettable fuses.

In this kit we use an NTC one. Each thermistor has a normal resistance. Here it is 10k ohm, which is measured under 25 degree Celsius.

Here is the relation between the resistance and temperature:

$$R_T = R_N * \exp(B(1/T_K - 1/T_N))$$

- $R_T$  is the resistance of the NTC thermistor when the temperature is  $T_K$ .

- **RN** is the resistance of the NTC thermistor under the rated temperature **TN**. Here, the numerical value of **RN** is 10k.
- **TK** is a Kelvin temperature and the unit is K. Here, the numerical value of **TK** is 273.15 + degree Celsius.
- **TN** is a rated Kelvin temperature; the unit is K too. Here, the numerical value of **TN** is 273.15+25.
- And **B(beta)**, the material constant of NTC thermistor, is also called heat sensitivity index with a numerical value 3950.
- **exp** is the abbreviation of exponential, and the base number **e** is a natural number and equals 2.7 approximately.

Convert this formula  $TK=1/(\ln(RT/RN)/B+1/TN)$  to get Kelvin temperature that minus 273.15 equals degree Celsius.

This relation is an empirical formula. It is accurate only when the temperature and resistance are within the effective range.

### Example

- [6.3 High Temperature Alarm](#) (Basic Project)
- [4.5 Thermometer](#) (Basic Project)

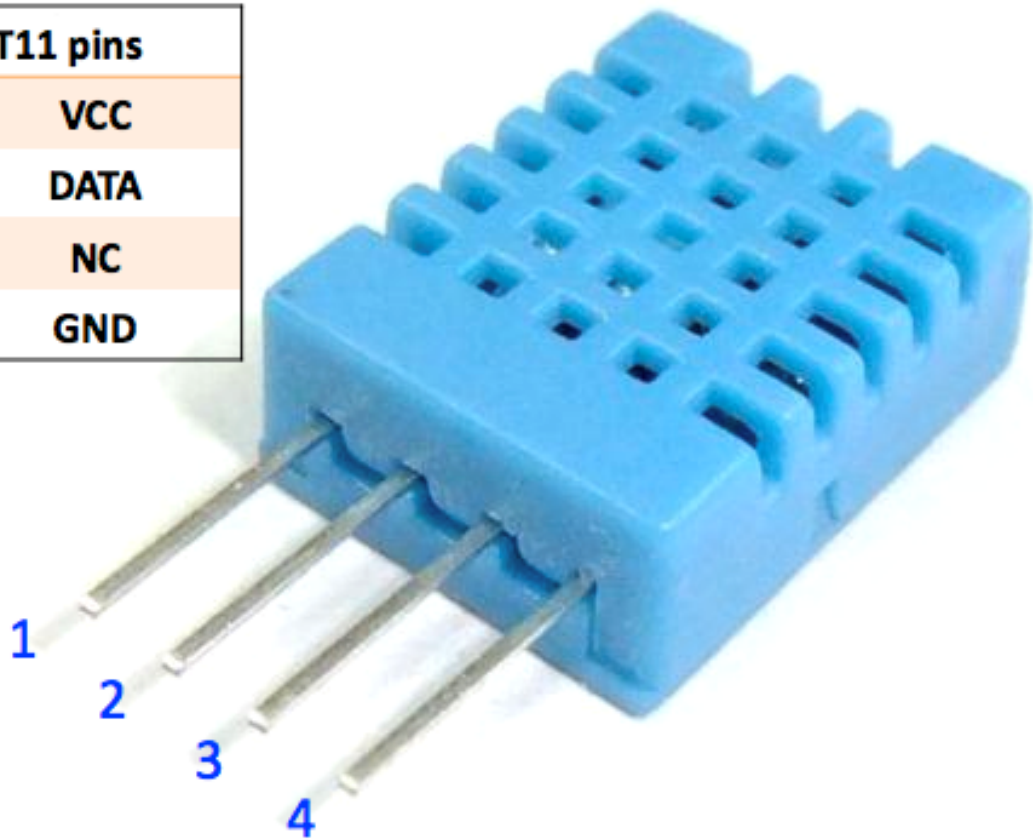
## 1.24 DHT11 Humiture Sensor

The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability.

The sensor includes a resistive sense of wet component and an NTC temperature measurement device, and is connected with a high-performance 8-bit microcontroller.

Only three pins are available for use: VCC, GND, and DATA. The communication process begins with the DATA line sending start signals to DHT11, and DHT11 receives the signals and returns an answer signal. Then the host receives the answer signal and begins to receive 40-bit humiture data (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



### Features

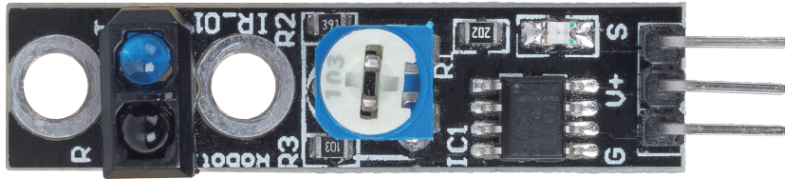
1. Humidity measurement range: 20 - 90%RH
2. Temperature measurement range: 0 - 60°C
3. Output digital signals indicating temperature and humidity
4. Working voltage:DC 5V; PCB size: 2.0 x 2.0 cm
5. Humidity measurement accuracy:  $\pm 5\%$  RH
6. Temperature measurement accuracy:  $\pm 2^\circ\text{C}$

- [DHT11 Datasheet](#)

### Example

- *5.11.3 Temperature - Humidity* (Basic Project)
- *5. Home Environment Monitoring* (IoT Project)
- *6. Plant Monitor* (IoT Project)

## 1.25 Line Tracking Module



- S: Usually low level, high level when the black line is detected.
- V+: Power supply, 3.3v~5V
- G: Ground

This is a 1-channel Line Tracking module which, as the name suggests, tracks black lines on a white background or white lines against a black background.



The module uses a TCRT500 infrared sensor, which consists of an infrared LED (blue) and a photosensitive triplet (black).

- The blue infrared LED, when powered on, emits infrared light that is invisible to the human eye.
- The black phototransistor, which is used to receive infrared light, has an internal resistor whose resistance varies with the infrared light received; the more infrared light received, the lower its resistance decreases and vice versa.

There is a LM393 comparator on the module, which is used to compare the voltage of the phototransistor with the set voltage (adjusted by potentiometer), if it is greater than the set voltage, the output is 1; otherwise the output is 0.

Therefore, when the infrared emitter tube shines on a black surface, because the black will absorb light, the photosensitive transistor receives less infrared light, its resistance will increase (voltage increase), after LM393 comparator, the output high level.

Similarly, when it shines on a white surface, the reflected light will become more and the resistance of the photosensitive transistor will decrease (voltage decreases); therefore, the comparator outputs a low level and the indicator LED lights up.

- [TCRT5000](#)

### Features

- Using infrared emission sensor TCRT5000
- Detection distance: 1-8mm, focal length of 2.5mm
- Comparator output signal clean, good waveform, driving capacity greater than 15mA
- Using potentiometer for sensitivity adjustment



- Operating voltage: 3.3V-5V
- Digital output: 0 (white) and 1 (black)
- Uses wide voltage LM393 comparator.
- Size: 42mmx10mm

#### Example

- *3.4 Detect the Line* (Basic Project)
- *4. Follow the line* (Car Project)

## 1.26 Soil Moisture Module

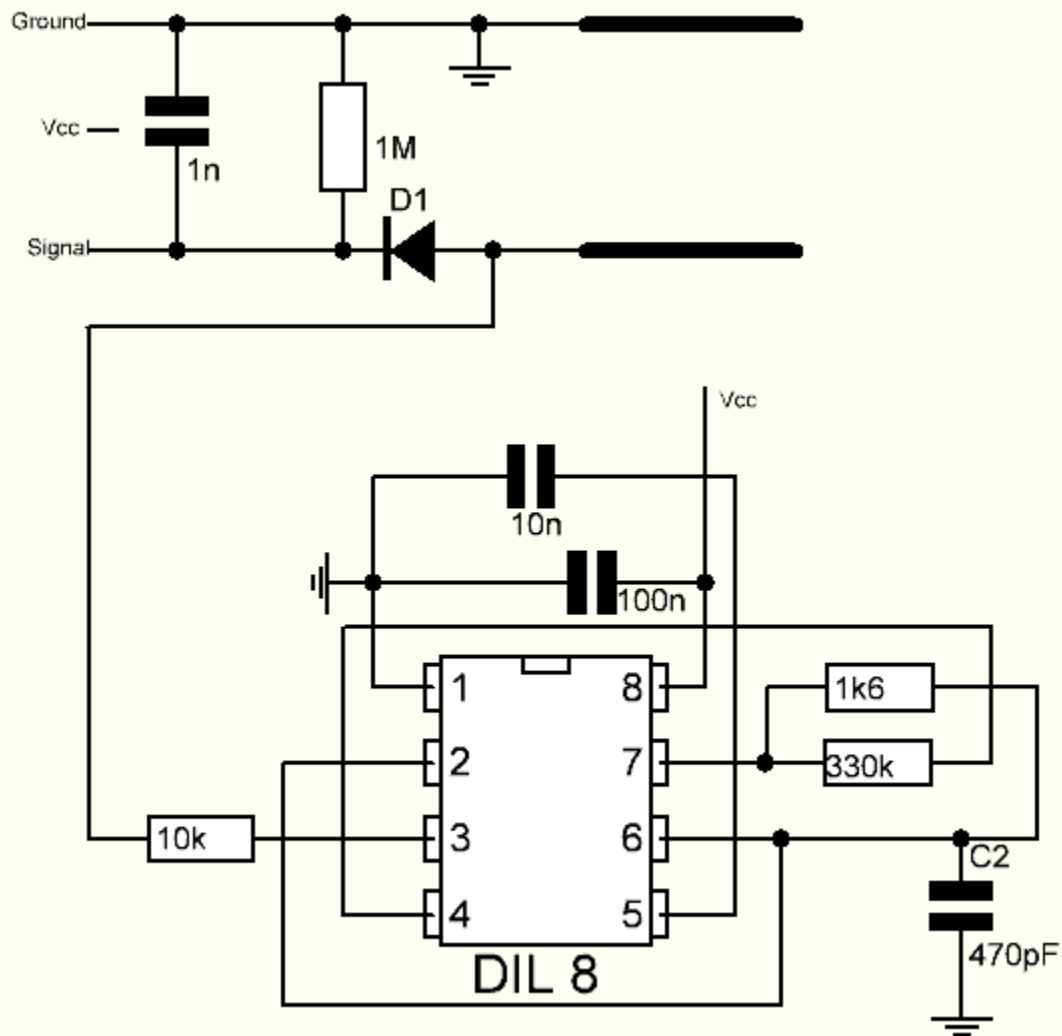


- GND: Ground
- VCC: Power supply, 3.3v~5V
- AU0T: Outputs the soil moisture value, the wetter the soil, the smaller its value.

This capacitive soil moisture sensor is different from most of the resistive sensors on the market, using the principle of capacitive induction to detect soil moisture. It avoids the problem that resistive sensors are highly susceptible to corrosion and greatly extends its working life.

It is made of corrosion-resistant materials and has an excellent service life. Insert it into the soil around plants and monitor real-time soil moisture data. The module includes an on-board voltage regulator that allows it to operate over a voltage range of 3.3 ~ 5.5 V. It is ideal for low-voltage microcontrollers with 3.3 V and 5 V supplies.

The hardware schematic of the capacitive soil moisture sensor is shown below.



There is a fixed frequency oscillator, which is built with a 555 timer IC. The generated square wave is then fed to the sensor like a capacitor. However, for the square wave signal, the capacitor has a certain reactance or, for the sake of argument, a resistor with a pure ohmic resistor (10k resistor on pin 3) to form a voltage divider.

The higher the soil moisture, the higher the capacitance of the sensor. As a result, the square wave has less reactance, which reduces the voltage on the signal line, and the smaller the value of the analog input through the microcontroller.

#### Specification

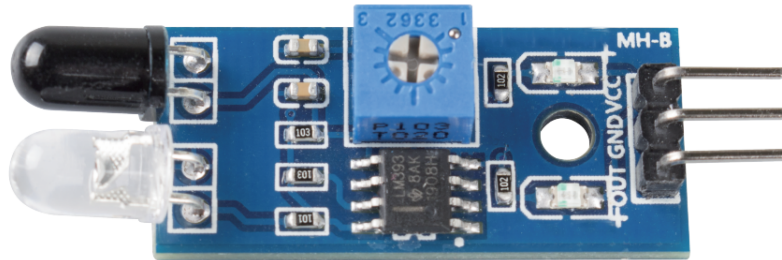
- Operating Voltage: 3.3 ~ 5.5 VDC
- Output Voltage: 0 ~ 3.0VDC
- Operating Current: 5mA
- Interface: PH2.0-3P
- Dimensions: 3.86 x 0.905 inches (L x W)
- Weight: 15g

#### Example

- [4.4 Measure Soil Moisture](#) (Basic Project)

- 6. *Plant Monitor* (IoT Project)

## 1.27 Obstacle Avoidance Module

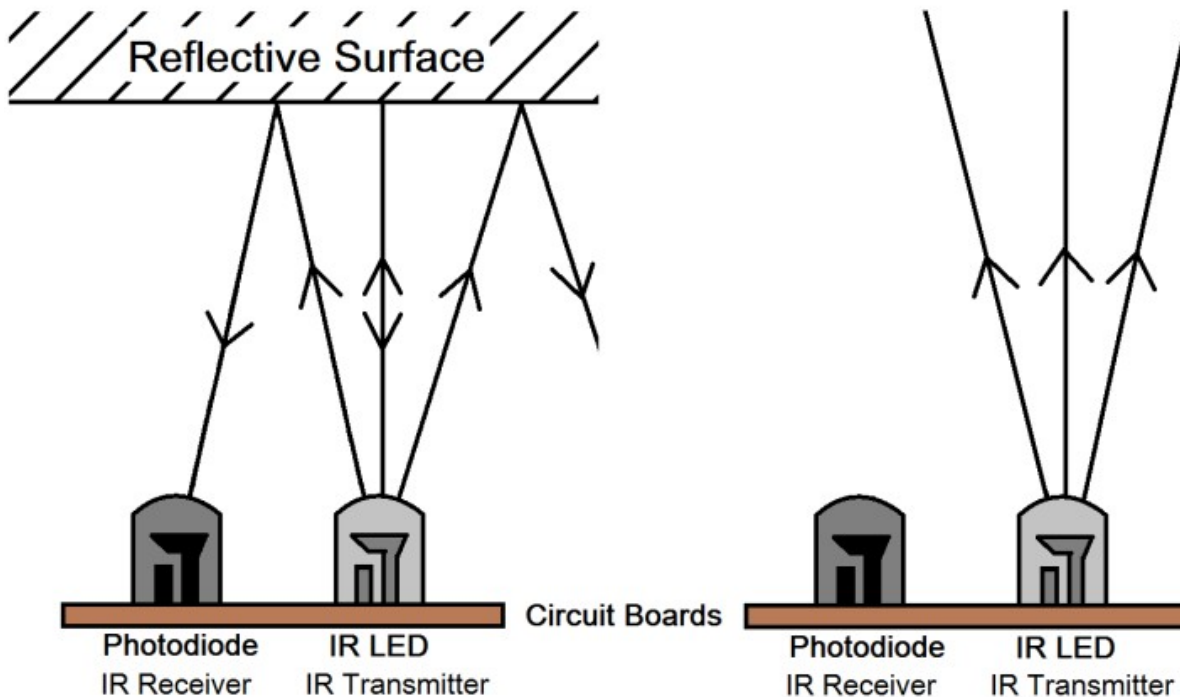


- **VCC:** Power supply, 3.3 ~ 5V DC.
- **GND:** Ground
- **OUT:** Signal pin, usually high level, and low level when an obstacle is detected.

The IR obstacle avoidance module has strong adaptability to environmental light, it has a pair of infrared transmitting and receiving tubes.

The transmitting tube emits infrared frequency, when the detection direction encounters an obstacle, the infrared radiation is received by the receiving tube, after the comparator circuit processing, the indicator will light up and output low level signal.

The detection distance can be adjusted by potentiometer, the effective distance range 2-30cm.

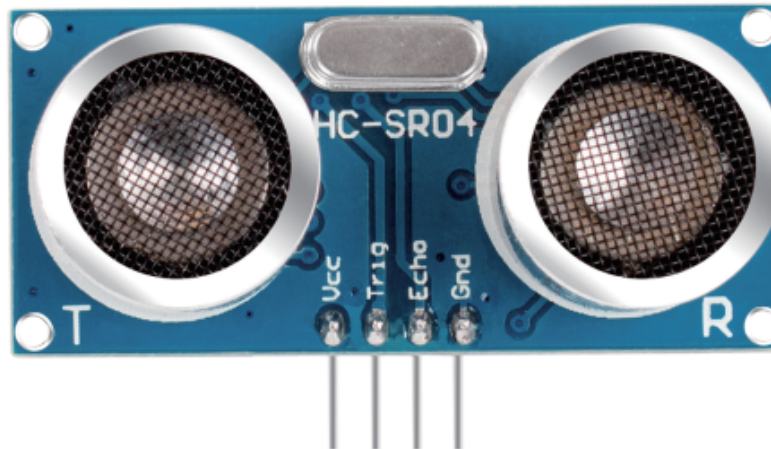


### Example

- 3.3 *Detect the Obstacle* (Basic Project)
- 5. *Play with Obstacle Avoidance Module* (Car Project)

- 8. *Self-Driving Car* (Car Project)
- 7. *Current Limiting Gate* (IoT Project)

## 1.28 Ultrasonic Module

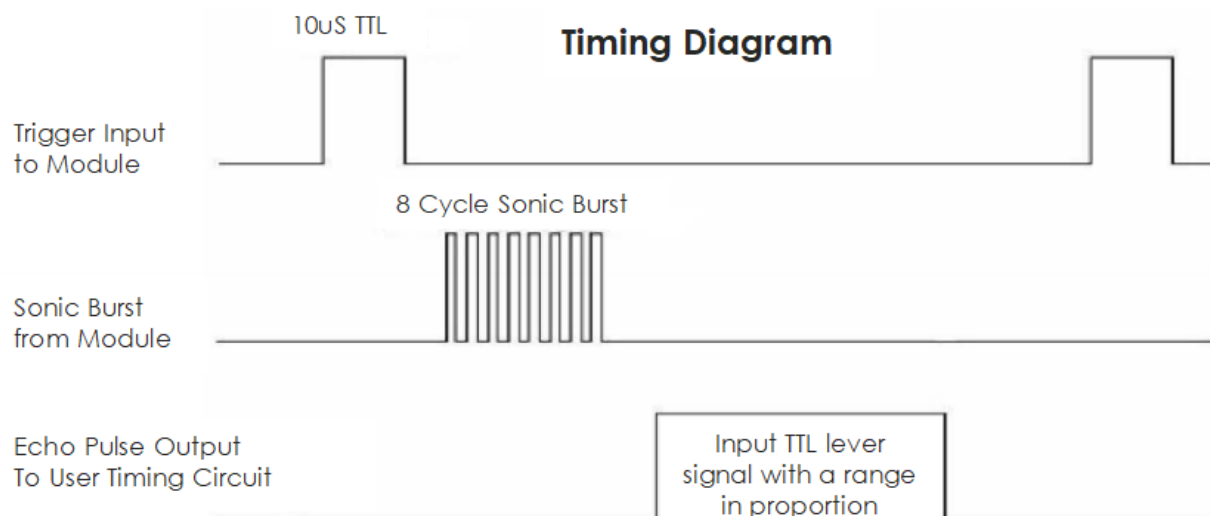


Ultrasonic ranging module provides 2cm - 400cm non-contact measurement function, and the ranging accuracy can reach to 3mm. It can ensure that the signal is stable within 5m, and the signal is gradually weakened after 5m, till the 7m position disappears.

The module includes ultrasonic transmitters, receiver and control circuit. The basic principles are as follows:

1. Use an IO flip-flop to process a high level signal of at least 10 $\mu$ s.
2. The module automatically sends eight 40khz and detects if there is a pulse signal return.
3. If the signal returns, passing the high level, the high output IO duration is the time from the transmission of the ultrasonic wave to the return of it. Here, test distance = (high time x sound speed (340 m / s) / 2.

The timing diagram is shown below.



You only need to supply a short 10us pulse for the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. You can calculate the range through the time interval between sending trigger signal and receiving echo signal.

Formula:  $\text{us} / 58 = \text{centimeters}$  or  $\text{us} / 148 = \text{inch}$ ; or: the range = high level time \* velocity (340M/S) / 2; you are suggested to use measurement cycle over 60ms in order to prevent signal collisions of trigger signal and the echo signal.

**Example**

- *5.8 User-defined Function* (Basic Project)
- *7. Follow Your Hand* (Car Project)
- *6. Play with Ultrasonic Module* (Car Project)



## **GET STARTED WITH ARDUINO**

If you have no idea about Arduino. There are several words I would like to show you: electronics, design, programming, and even Maker. Some of you may think these words are quite far away from us, but in fact, they are not far at all. Because Arduino can take us into the world of programming and help us realize the dream of being a Maker. In this session we will learn:

- What is Arduino?
- what can Arduino do?
- How to build an Arduino Project?

### **2.1 What is Arduino?**

First of all, I will give you a brief introduction to Arduino.

Arduino is a convenient, flexible, and easy-to-use open-source electronic prototyping platform, including hardware Arduino boards of various models and software Arduino IDE. It is not only suitable for engineers for rapid prototyping, but also artists, designers, hobbyists, while it is almost a must-have tool for modern Makers.

Arduino is quite a large system. It has software, hardware, and a very huge online community of people who have never met each other but are able to work together because of a common hobby. Everyone in the Arduino family is using their wisdom, making with their hands, and sharing one great invention after another. And you can also be a part of it.

### **2.2 What can Arduino do?**

Speaking of which, you may have doubts about what Arduino can actually do. Suffice it to say, Arduino will solve all your problems.

Technically speaking, Arduino is a programmable logic controller. It is a development board that can be used to create many exciting and creative electronic creations: such as remote-controlled cars, robotic arms, bionic robots, smart homes, etc.

Arduino boards are straightforward, simple, and powerful, suitable for students, makers and even professional programmers.

To this day, electronics enthusiasts worldwide continue to develop creative electronic creations based on Arduino development boards.

## 2.3 How to build an Arduino Project

Follow these steps to learn how to use Arduino from zero!

### 2.3.1 Download and Install Arduino IDE 2.0

The Arduino IDE, known as Arduino Integrated Development Environment, provides all the software support needed to complete an Arduino project. It is a programming software specifically designed for Arduino, provided by the Arduino team, that allows us to write programs and upload them to the Arduino board.

The Arduino IDE 2.0 is an open-source project. It is a big step from its sturdy predecessor, Arduino IDE 1.x, and comes with revamped UI, improved board & library manager, debugger, autocomplete feature and much more.

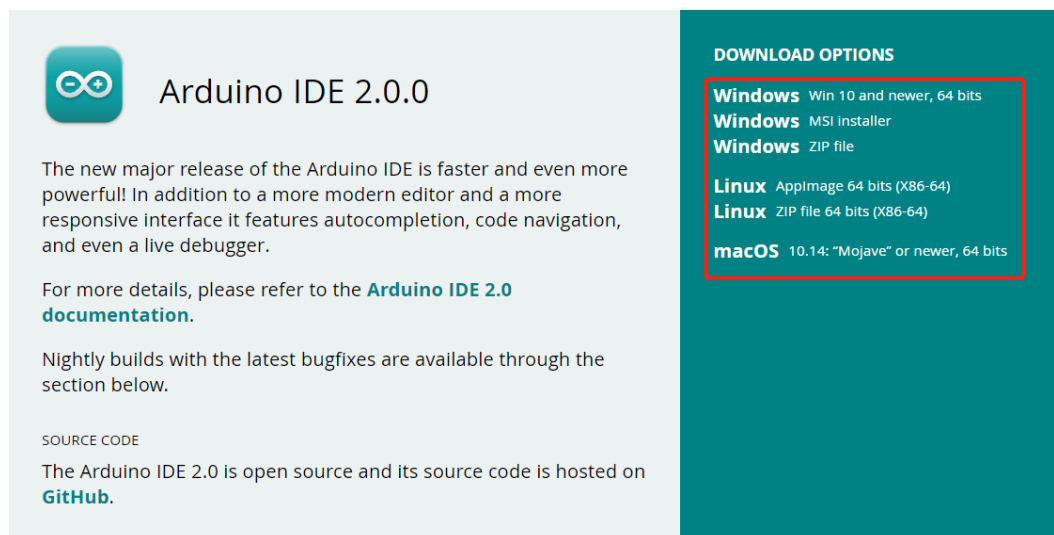
In this tutorial, we will show how to download and install the Arduino IDE 2.0 on your Windows, Mac, or Linux computer.


#### Requirements

- Windows - Win 10 and newer, 64 bits
- Linux - 64 bits
- Mac OS X - Version 10.14: “Mojave” or newer, 64 bits

#### Download the Arduino IDE 2.0

1. Visit .
2. Download the IDE for your OS version.



 **Arduino IDE 2.0.0**

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

**SOURCE CODE**

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

**DOWNLOAD OPTIONS**

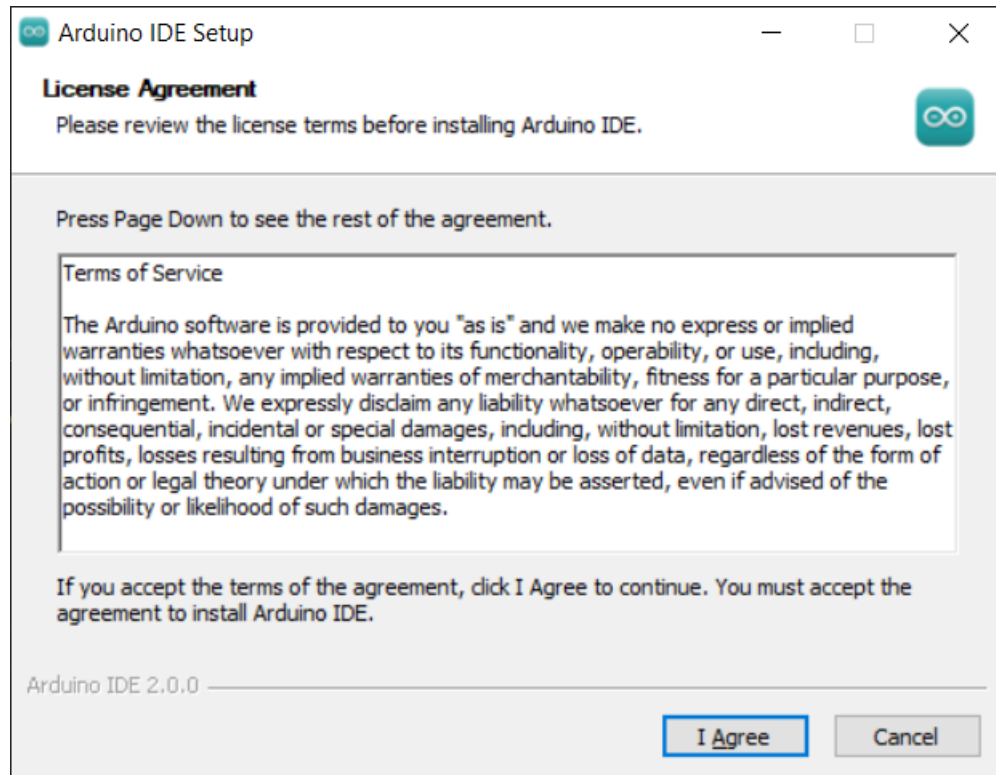
- Windows** Win 10 and newer, 64 bits
- Windows** MSI installer
- Windows** ZIP file
- Linux** AppImage 64 bits (X86-64)
- Linux** ZIP file 64 bits (X86-64)
- macOS** 10.14: “Mojave” or newer, 64 bits



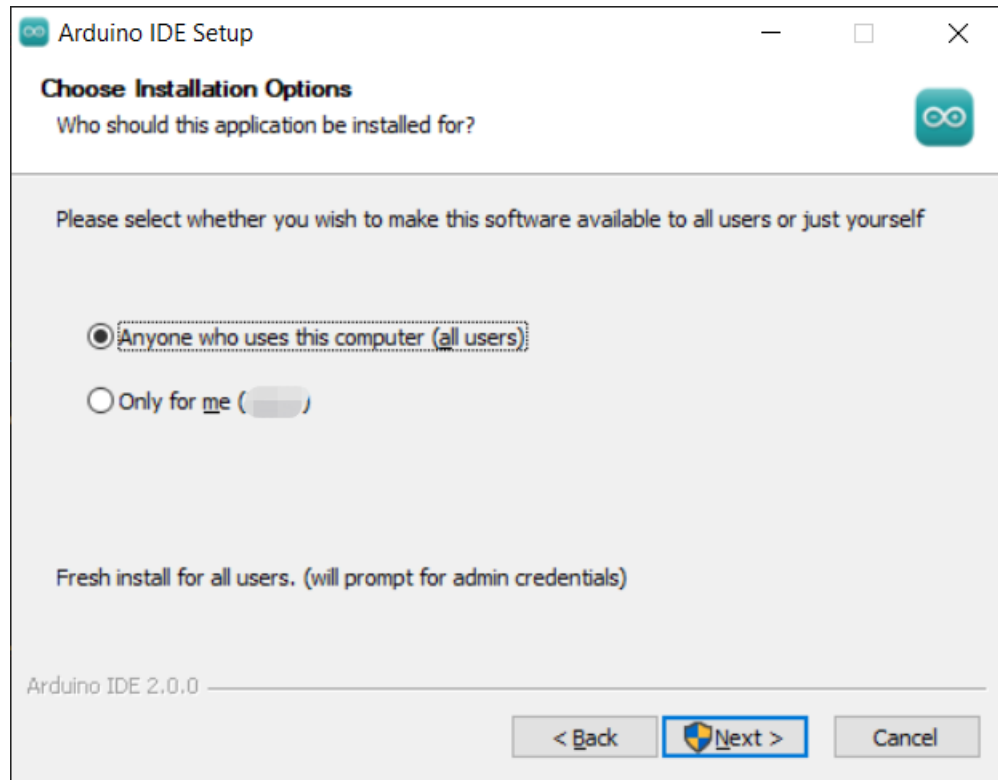
## Installation

### Windows

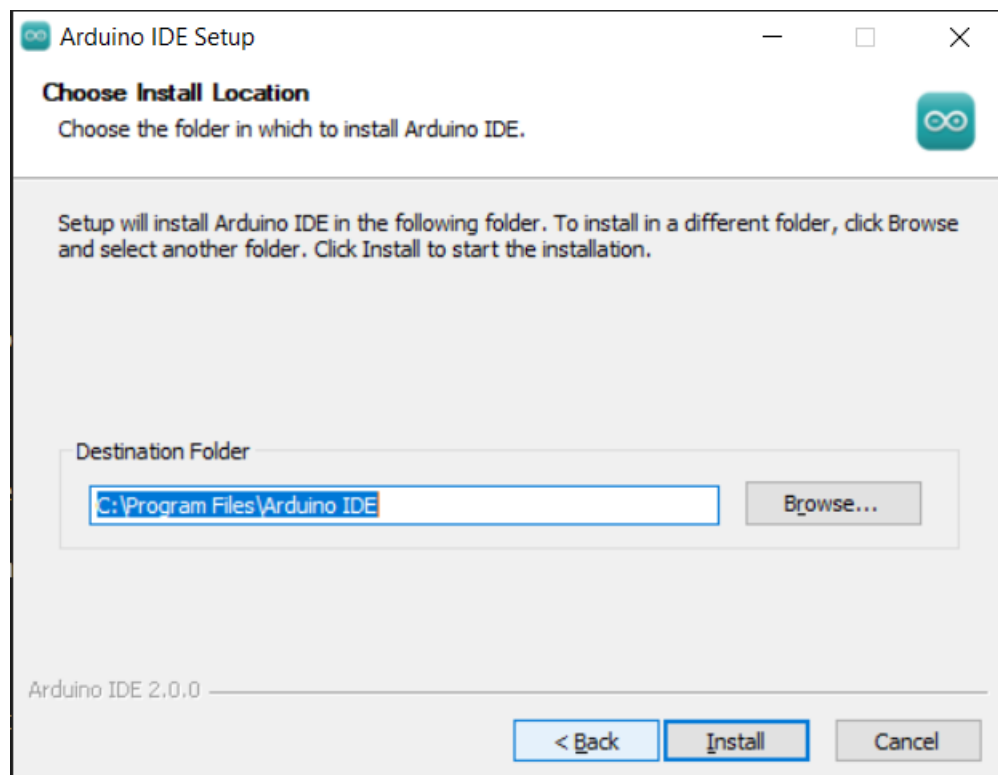
1. Double click the `arduino-ide_xxxx.exe` file to run the downloaded file.
2. Read the License Agreement and agree it.



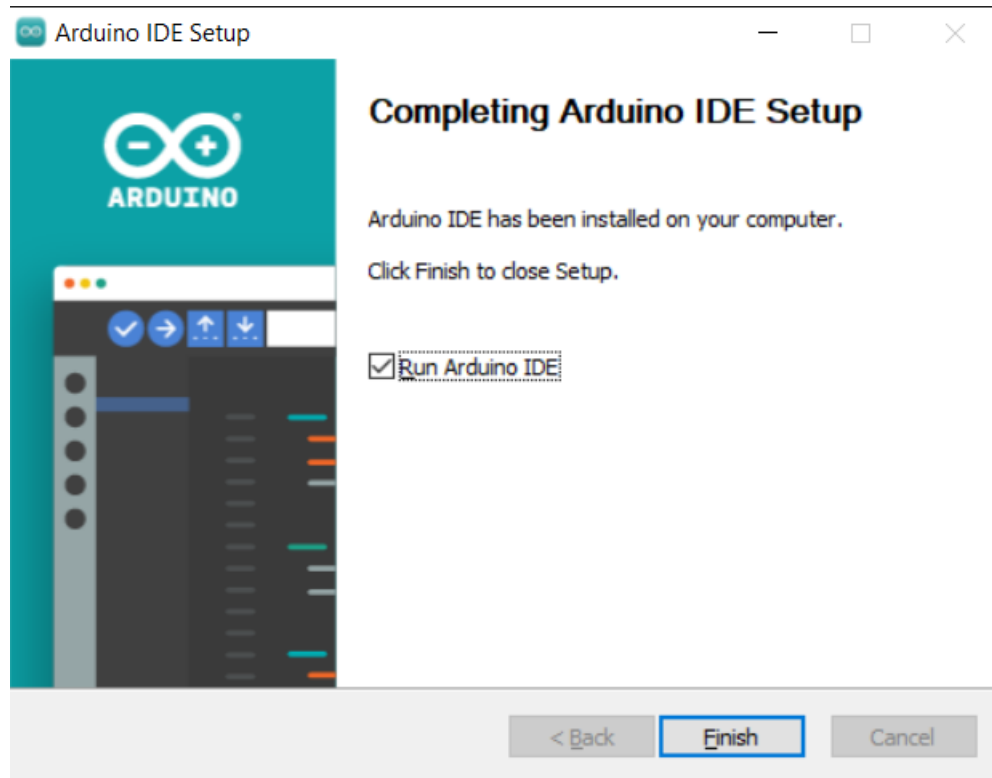
3. Choose installation options.



4. Choose install location. It is recommended that the software be installed on a drive other than the system drive.

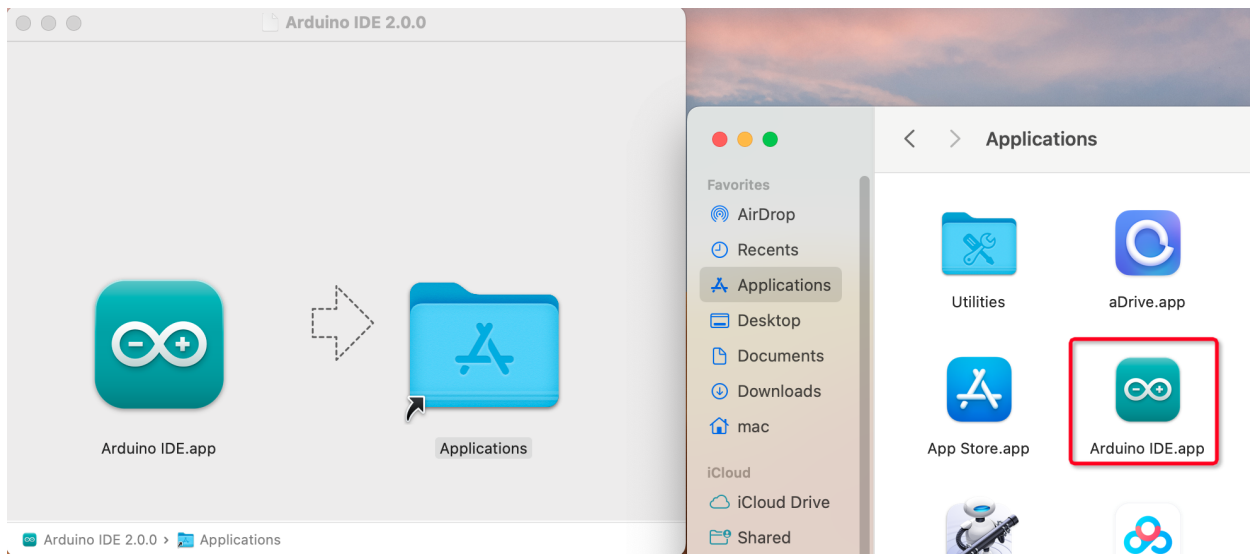


5. Then Finish.



## macOS

Double click on the downloaded `arduino_ide_xxxx.dmg` file and follow the instructions to copy the **Arduino IDE.app** to the **Applications** folder, you will see the Arduino IDE installed successfully after a few seconds.

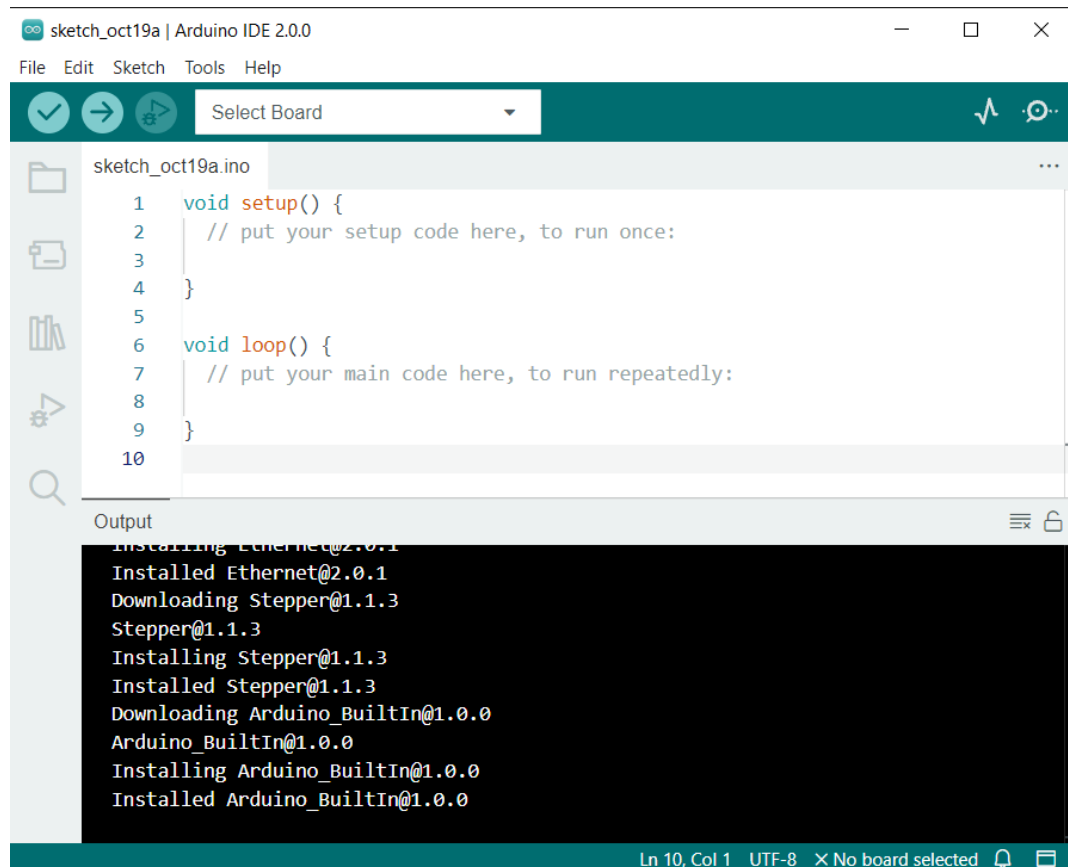


## Linux

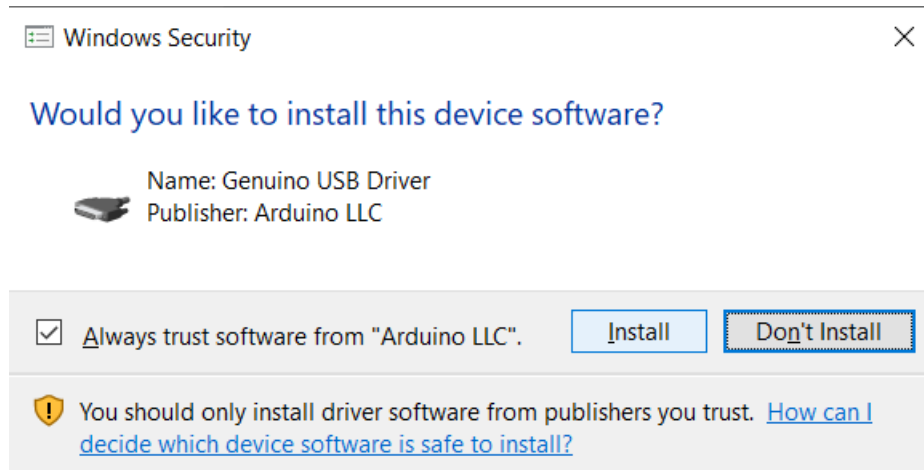
For the tutorial on installing the Arduino IDE 2.0 on a Linux system, please refer to: <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux>

### Open the IDE

1. When you first open Arduino IDE 2.0, it automatically installs the Arduino AVR Boards, built-in libraries, and other required files.



2. In addition, your firewall or security center may pop up a few times asking you if you want to install some device driver. Please install all of them.



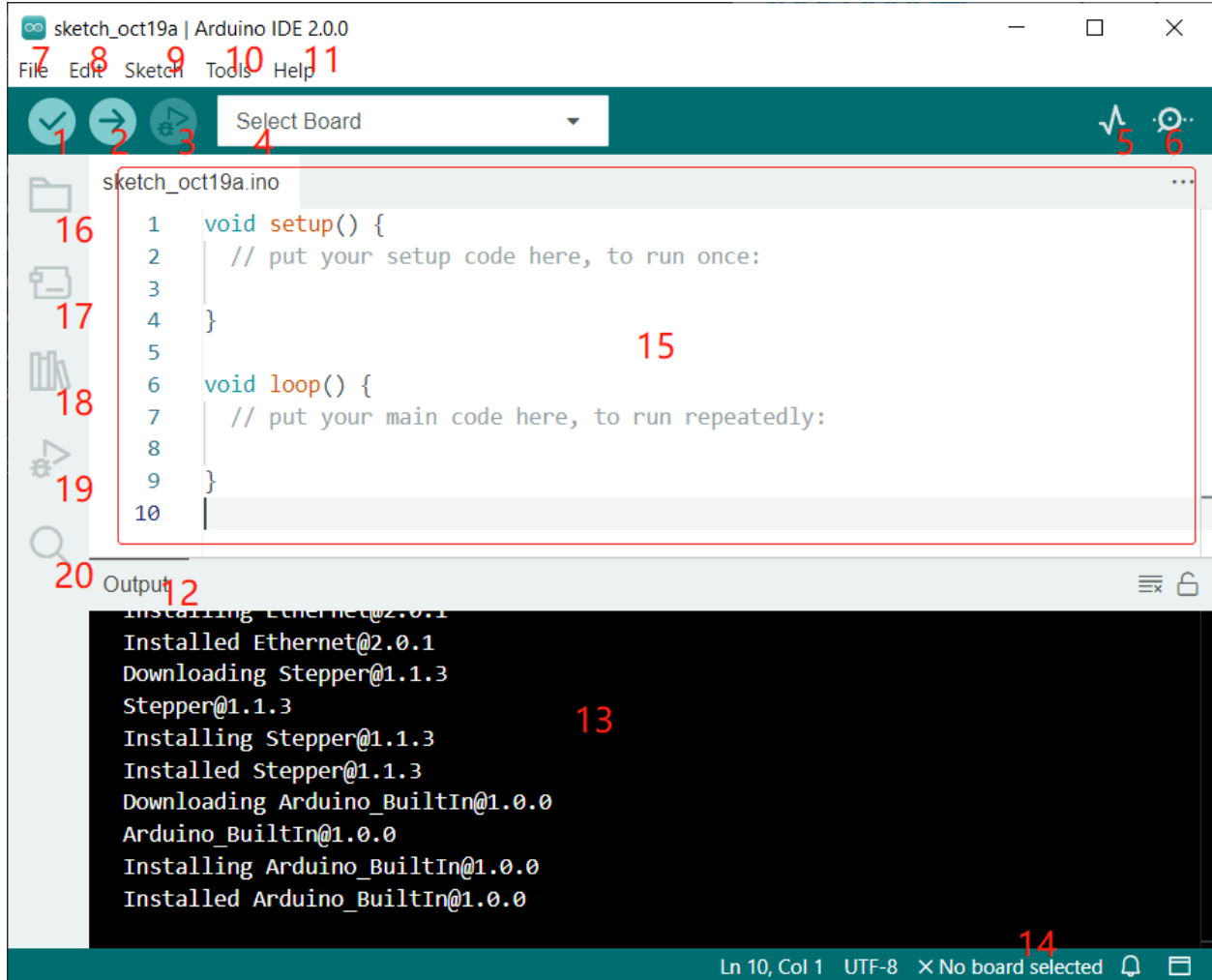
3. Now your Arduino IDE is ready!

---

**Note:** In the event that some installations didn't work due to network issues or other reasons, you can reopen the Arduino IDE and it will finish the rest of the installation. The Output window will not automatically open after all installations are complete unless you click Verify or Upload.

---

### 2.3.2 Introduce of Arduino IDE

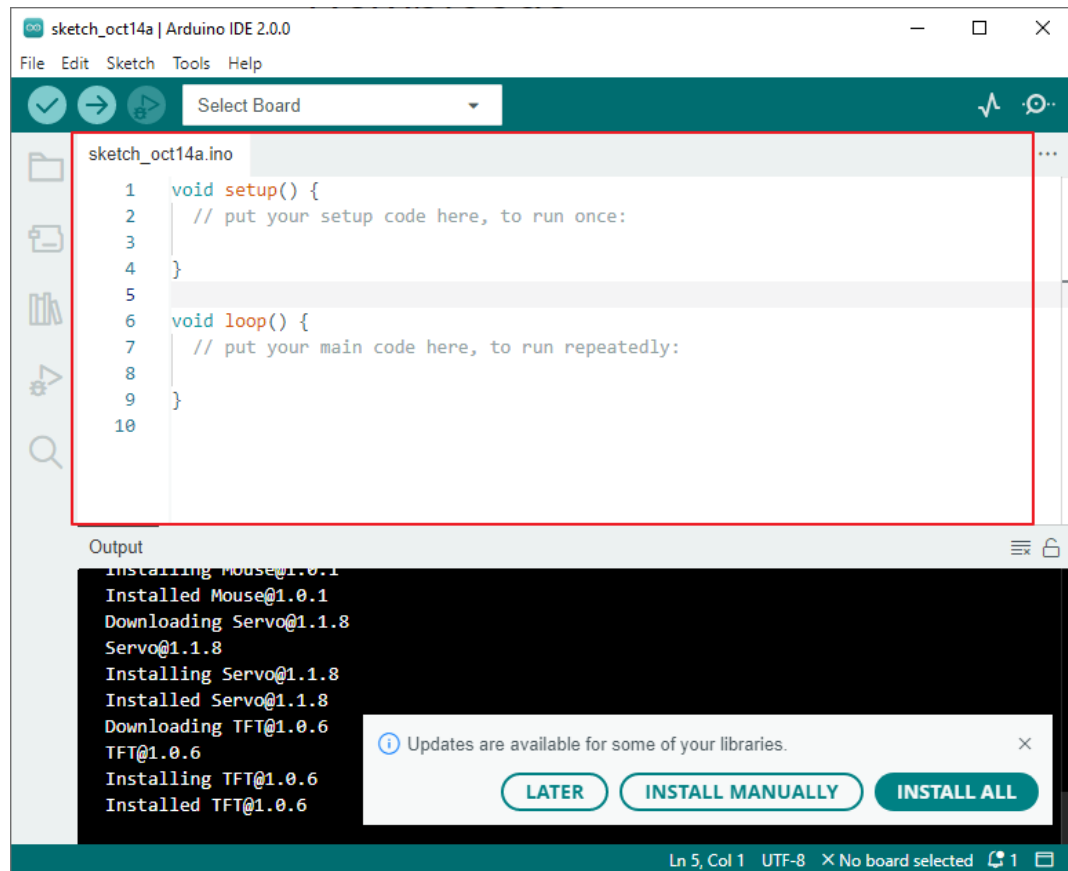


1. **Verify:** Compile your code. Any syntax problem will be prompted with errors.
2. **Upload:** Upload the code to your board. When you click the button, the RX and TX LEDs on the board will flicker fast and won't stop until the upload is done.
3. **Debug:** For line-by-line error checking.
4. **Select Board:** Quick setup board and port.
5. **Serial Plotter:** Check the change of reading value.
6. **Serial Monitor:** Click the button and a window will appear. It receives the data sent from your control board. It is very useful for debugging.
7. **File:** Click the menu and a drop-down list will appear, including file creating, opening, saving, closing, some parameter configuring, etc.
8. **Edit:** Click the menu. On the drop-down list, there are some editing operations like **Cut**, **Copy**, **Paste**, **Find**, and so on, with their corresponding shortcuts.
9. **Sketch:** Includes operations like **Verify**, **Upload**, **Add** files, etc. More important function is **Include Library** – where you can add libraries.

10. **Tool:** Includes some tools – the most frequently used Board (the board you use) and Port (the port your board is at). Every time you want to upload the code, you need to select or check them.
11. **Help:** If you're a beginner, you may check the options under the menu and get the help you need, including operations in IDE, introduction information, troubleshooting, code explanation, etc.
12. **Output Bar:** Switch the output tab here.
13. **Output Window:** Print information.
14. **Board and Port:** Here you can preview the board and port selected for code upload. You can select them again by **Tools** -> **Board** / **Port** if any is incorrect.
15. The editing area of the IDE. You can write code here.
16. **Sketchbook:** For managing sketch files.
17. **Board Manager:** For managing board driver.
18. **Library Manager:** For managing your library files.
19. **Debug:** Help debugging code.
20. **Search:** Search the codes from your sketches.

### 2.3.3 How to create, open or Save the Sketch?

1. When you open the Arduino IDE for the first time or create a new sketch, you will see a page like this, where the Arduino IDE creates a new file for you, which is called a “sketch”.



These sketch files have a regular temporary name, from which you can tell the date the file was created. `sketch_oct14a.ino` means October 14th first sketch, `.ino` is the file format of this sketch.

2. Now let's try to create a new sketch. Copy the following code into the Arduino IDE to replace the original code.

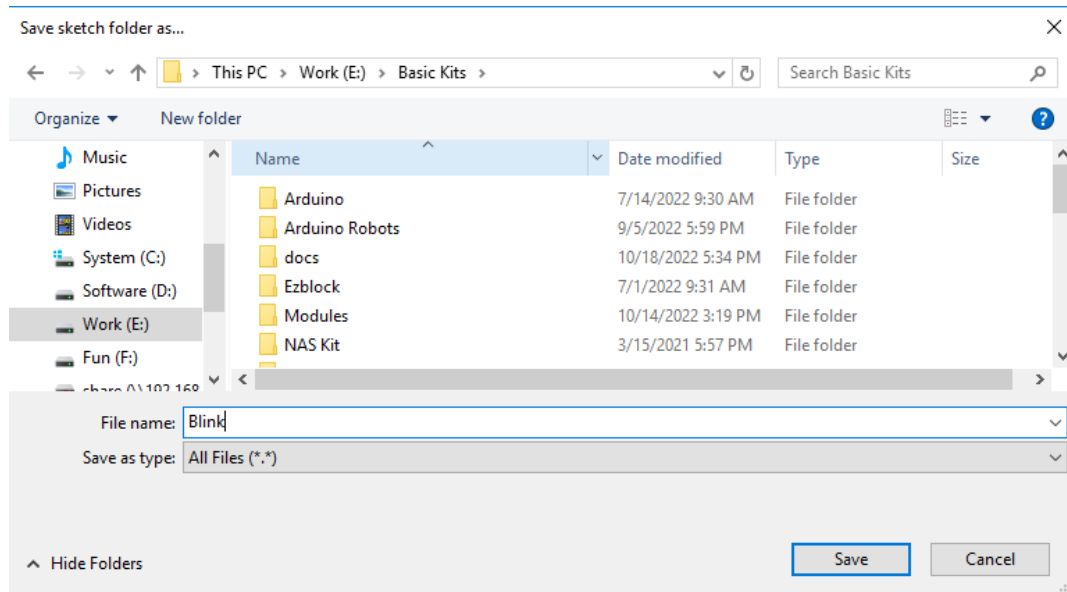


```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

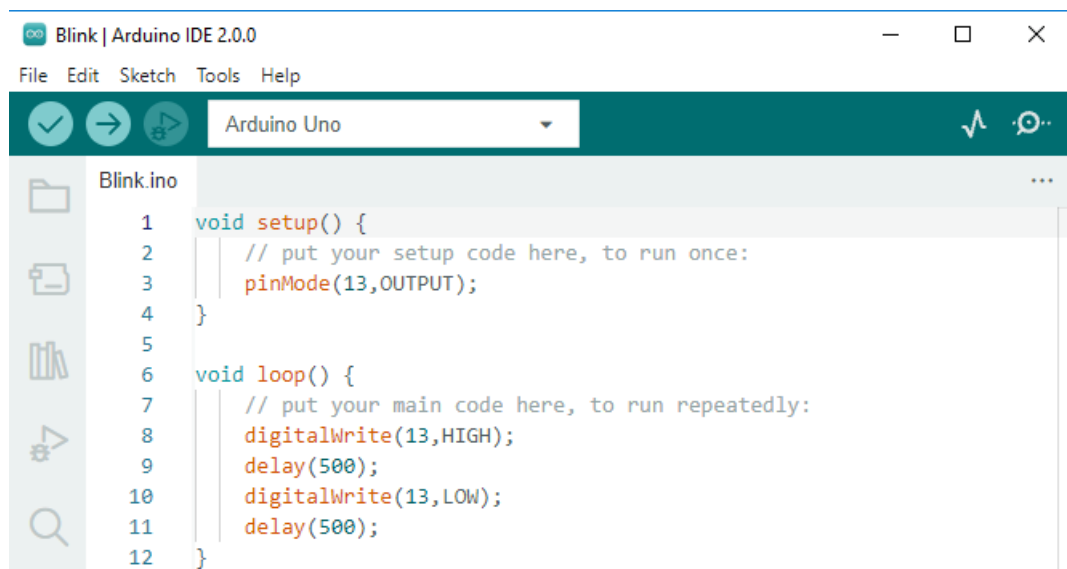
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(13,LOW);
  delay(500);
}
```

3. Press `Ctrl+S` or click **File** -> **Save**. The Sketch is saved in: `C:\Users\{your_user}\Documents\Arduino` by default, you can rename it or find a new path to save it.





4. After successful saving, you will see that the name in the Arduino IDE has been updated.



Please continue with the next section to learn how to upload this created sketch to your Arduino board.

## 2.3.4 How to upload Sketch to the Board?

In this section, you will learn how to upload the sketch created previously to the Arduino board, as well as learn about some considerations.

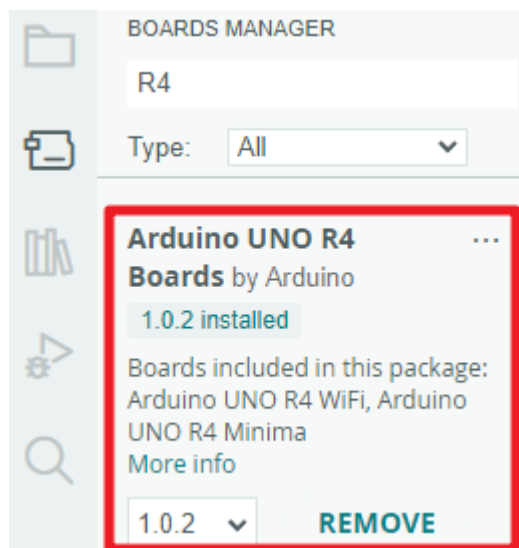
### 1. Choose Board and port

Arduino development boards usually come with a USB cable. You can use it to connect the board to your computer.

Select the correct **Board** and **Port** in the Arduino IDE. Normally, Arduino boards are recognized automatically by the computer and assigned a port, so you can select it here.



If your board is already plugged in, but not recognized, check if the **INSTALLED** logo appears in the **Arduino UNO R4 Boards** section of the **Boards Manager**, if not, please scroll down a bit and click on **INSTALL**.



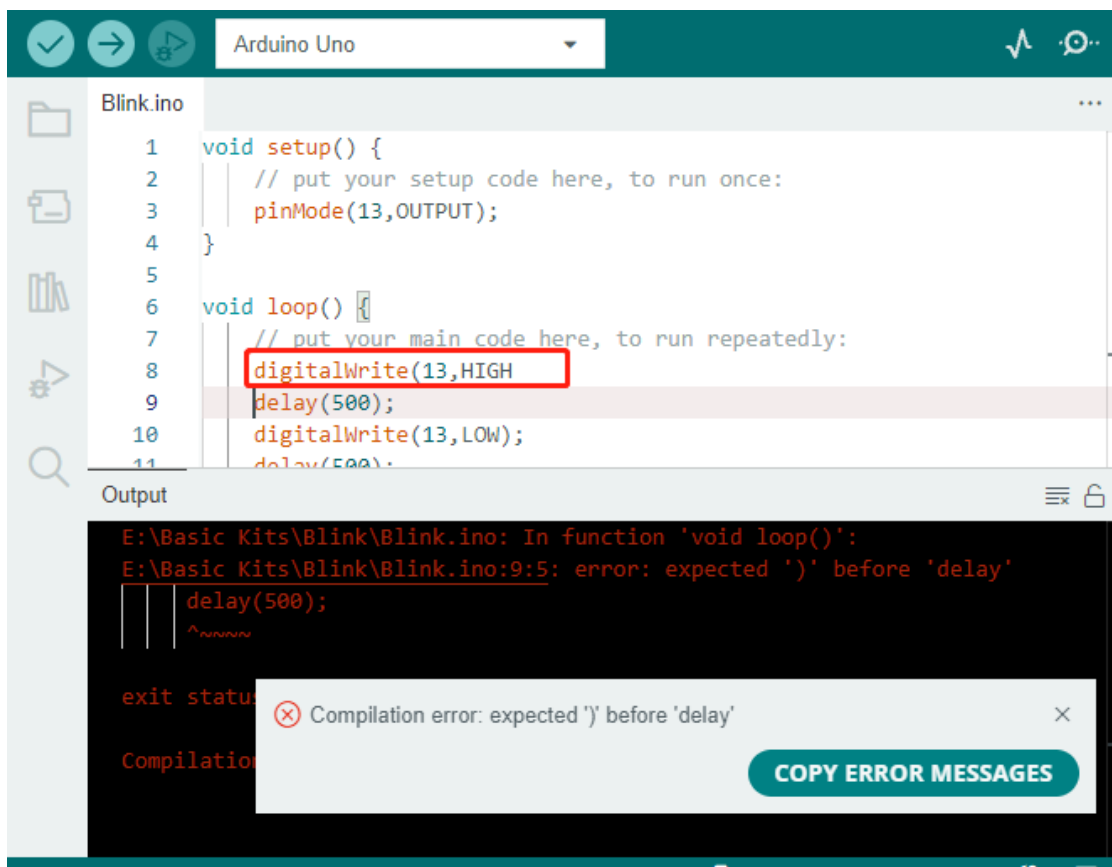
Reopening the Arduino IDE and re-plugging the Arduino board will fix most of the problems. You can also click **Tools** -> **Board** or **Port** to select them.

## 2. Verify the Sketch

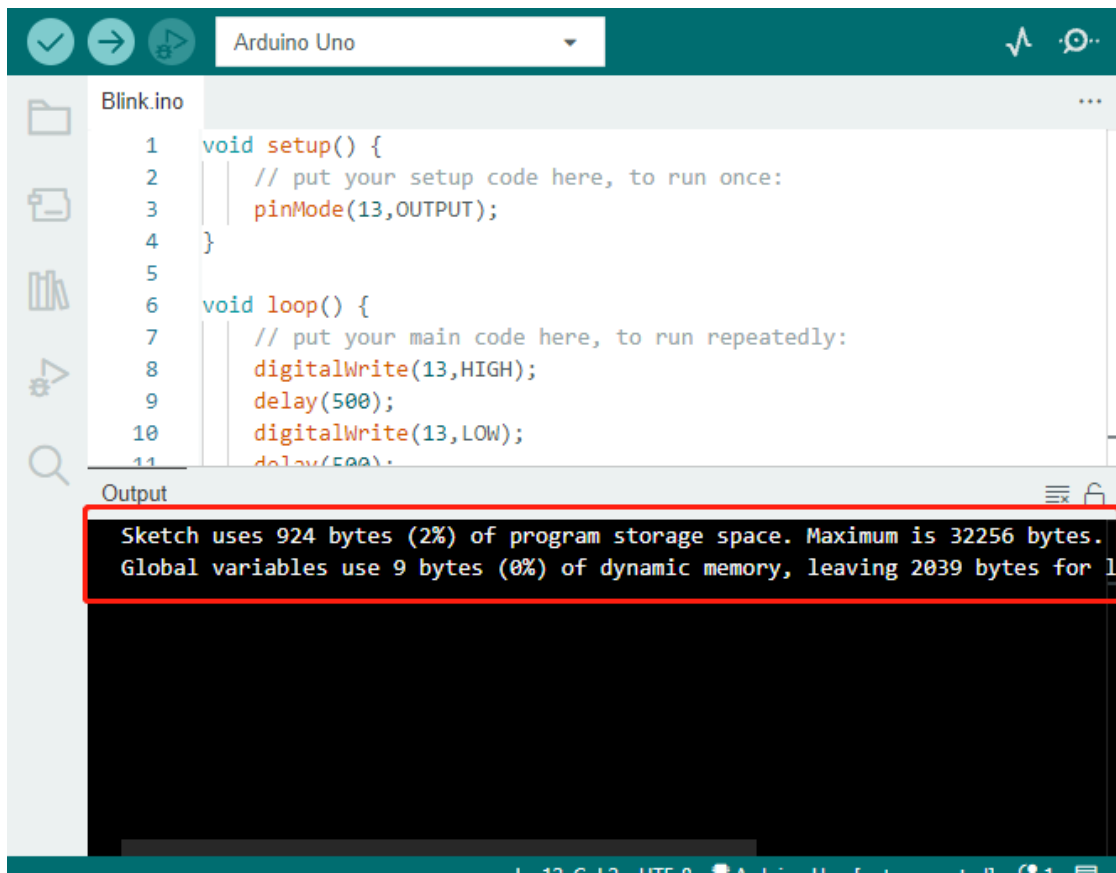
After clicking the Verify button, the sketch will be compiled to see if there are any errors.



You can use it to find mistakes if you delete some characters or type a few letters by mistake. From the message bar, you can see where and what type of errors occurred.

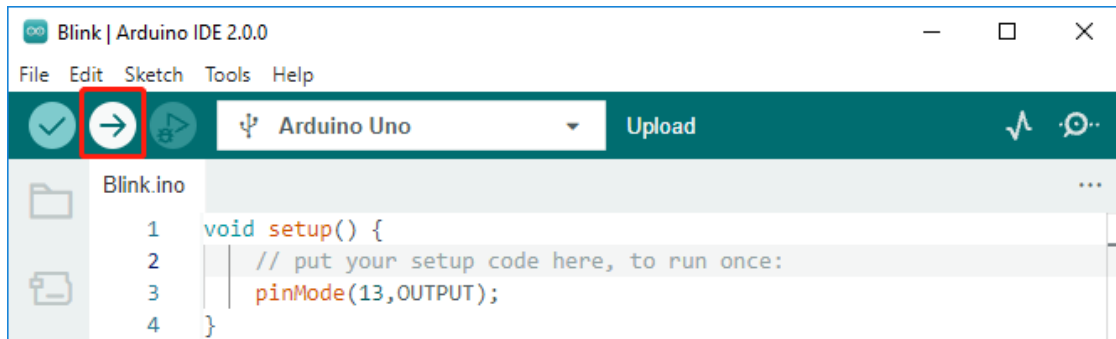


If there are no errors, you will see a message like the one below.

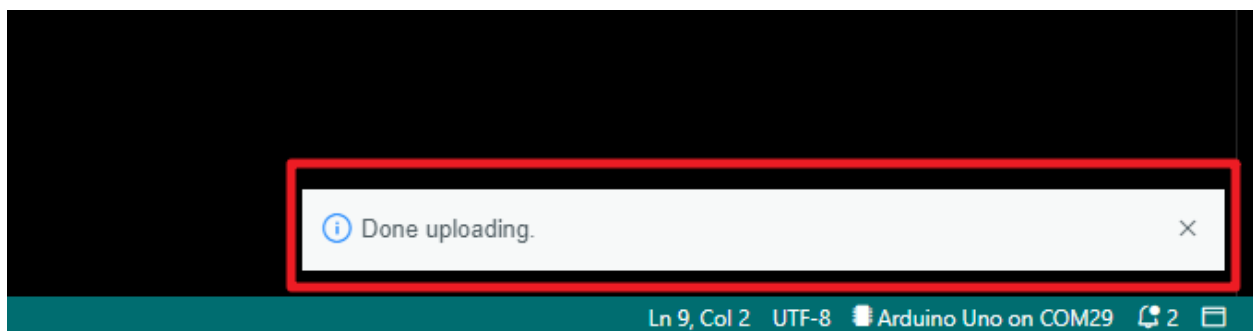


### 3. Upload sketch

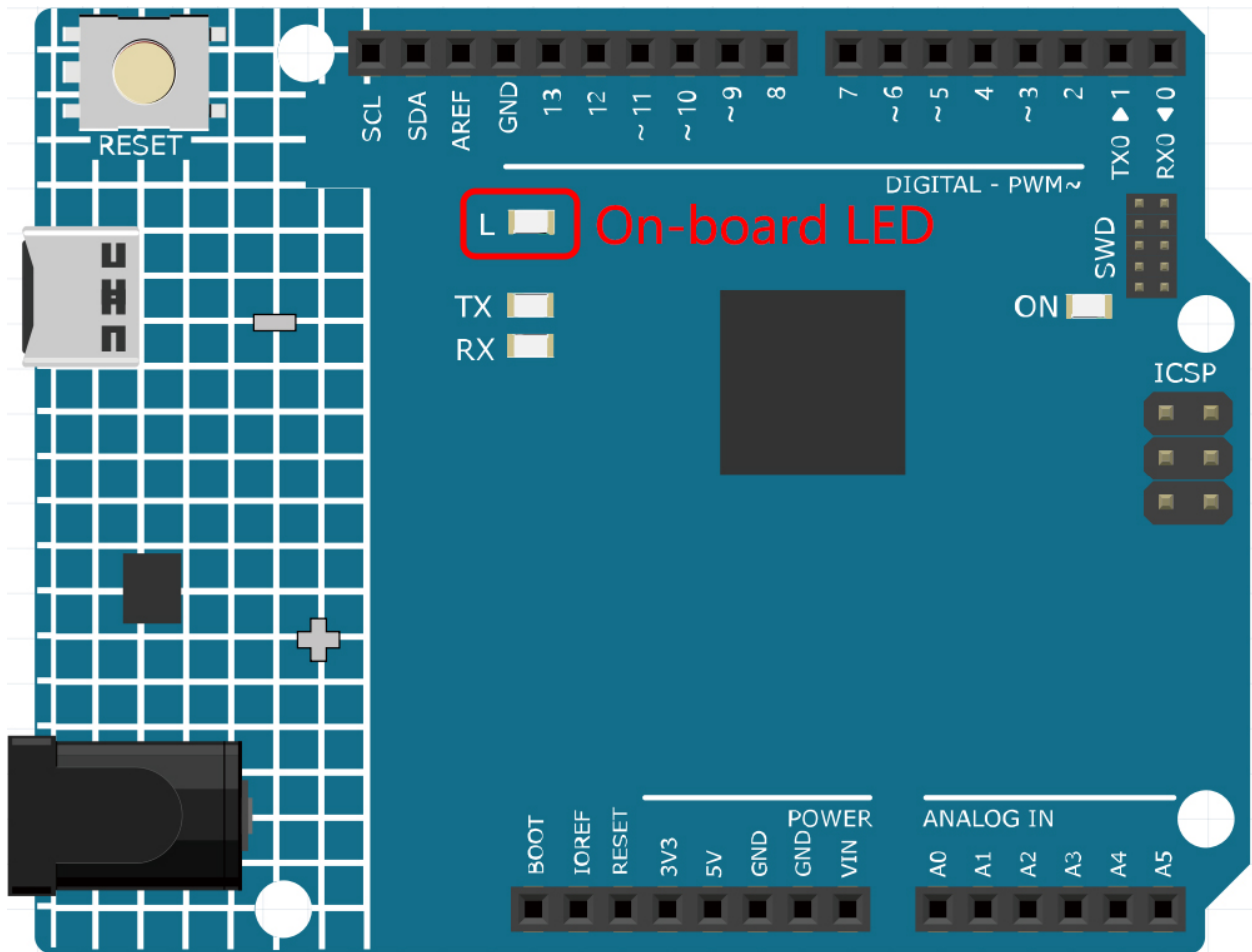
After completing the above steps, click the **Upload** button to upload this sketch to the board.



If successful, you will be able to see the following prompt.



At the same time, the on-board LED blink.



The Arduino board will automatically run the sketch after power is applied after the sketch is uploaded. The running program can be overwritten by uploading a new sketch.

### 2.3.5 Arduino Program Structure

Let's take a look at the new sketch file. Although it has a few lines of code itself, it is actually an “empty” sketch. Uploading this sketch to the development board will cause nothing to happen.

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

If we remove `setup()` and `loop()` and make the sketch a real blank file, you will find that it does not pass the verification. They are the equivalent of the human skeleton, and they are indispensable.

During sketching, `setup()` is run first, and the code inside it (inside `{}`) is run after the board is powered up or reset

and only once. `loop()` is used to write the main feature, and the code inside it will run in a loop after `setup()` is executed.

To better understand `setup()` and `loop()`, let's use four sketches. Their purpose is to make the on-board LED of the Arduino blink. Please run each experiment in turn and record them specific effects.

- Sketch 1: Make the on-board LED blink continuously.

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(13,OUTPUT);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    digitalWrite(13,HIGH);  
    delay(500);  
    digitalWrite(13,LOW);  
    delay(500);  
}
```

- Sketch 2: Make the on-board LED blink only once.

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(13,OUTPUT);  
    digitalWrite(13,HIGH);  
    delay(500);  
    digitalWrite(13,LOW);  
    delay(500);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```

- Sketch 3: Make the on-board LED blink slowly once and then blink quickly.

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(13,OUTPUT);  
    digitalWrite(13,HIGH);  
    delay(1000);  
    digitalWrite(13,LOW);  
    delay(1000);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    digitalWrite(13,HIGH);  
    delay(200);  
    digitalWrite(13,LOW);  
    delay(200);  
}
```

- Sketch 4: Report an error.

```

void setup() {
    // put your setup code here, to run once:
    pinMode(13,OUTPUT);
}

digitalWrite(13,HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);

void loop() {
    // put your main code here, to run repeatedly:
}

```

With the help of these sketches, we can summarize several features of `setup-loop`.

- `loop()` will be run repeatedly after the board is powered up.
- `setup()` will run only once after the board is powered up.
- After the board is powered up, `setup()` will run first, followed by `loop()`.
- The code needs to be written within the `{}` scope of `setup()` or `loop()`, out of the framework will be an error.

**Note:** Statements such as `digitalWrite(13,HIGH)` are used to control the on-board LED, and we will talk about their usage in detail in later chapters.

## 2.3.6 Sketch Writing Rule

If you ask a friend to turn on the lights for you, you can say “Turn on the lights.”, or “Lights on, bro.”, you can use any tone of voice you want.

However, if you want the Arduino board to do something for you, you need to follow the Arduino program writing rules to type in the commands.

This chapter contains the basic rules of the Arduino language and will help you understand how to translate natural language into code.

Of course, this is a process that takes time to get familiar with, and it is also the most error-prone part of the process for newbies, so if you make mistakes often, it’s okay, just try a few more times.

### Semicolon ;

Just like writing a letter, where you write a period at the end of each sentence as the end, the Arduino language requires you to use `;` to tell the board the end of the command.

Take the familiar “onboard LED blinking” example. A healthy sketch should look like this.

Example:

```

void setup() {
    // put your setup code here, to run once:
    pinMode(13,OUTPUT);
}

```

(continues on next page)

(continued from previous page)

```
void loop() {  
  // put your main code here, to run repeatedly:  
  digitalWrite(13,HIGH);  
  delay(500);  
  digitalWrite(13,LOW);  
  delay(500);  
}
```

Next, let's take a look at the following two sketches and guess if they can be correctly recognized by Arduino before running them.

Sketch A:

```
void setup() {  
  // put your setup code here, to run once:  
  pinMode(13,OUTPUT);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  digitalWrite(13,HIGH)  
  delay(500)  
  digitalWrite(13,LOW)  
  delay(500)  
}
```

Sketch B:

```
void setup() {  
  // put your setup code here, to run once:  
  pinMode(13,OUTPUT);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  digitalWrite(13,  
HIGH); delay  
  (500  
  );  
  digitalWrite(13,  
  
  LOW);  
      delay(500)  
  ;  
}
```

The result is that **Sketch A** reports an error and **Sketch B** runs.

- The errors in **Sketch A** are missing `;` and although it looks normal, the Arduino can't read it.
- **Sketch B**, looks anti-human, but in fact, indentation, line breaks and spaces in statements are things that do not exist in Arduino programs, so to the Arduino compiler, it looks the same as in the example.

However, please don't write your code as **Sketch B**, because it is usually natural people who write and view the code,



so don't get yourself into trouble.

## Curlybraces {}

{ } is the main component of the Arduino programming language, and they must appear in pairs. A better programming convention is to insert a structure that requires curly braces by typing the right curly brace directly after typing the left curly brace, and then moving the cursor between the curly braces to insert the statement.

## Comment //

Comment is the part of the sketch that the compiler ignores. They are usually used to tell others how the program works.

If we write two adjacent slashes in a line of code, the compiler will ignore anything up to the end of the line.

If we create a new sketch, it comes with two comments, and if we remove these two comments, the sketch will not be affected in any way.

```
void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

Comment is very useful in programming, and several common uses are listed below.

- Usage A: Tell yourself or others what this section of code does.

```
void setup() {
    pinMode(13,OUTPUT); //Set pin 13 to output mode, it controls the onboard LED
}

void loop() {
    digitalWrite(13,HIGH); // Activate the onboard LED by setting pin 13 high
    delay(500); // Status quo for 500 ms
    digitalWrite(13,LOW); // Turn off the onboard LED
    delay(500); // Status quo for 500 ms
}
```

- Usage B: Temporarily invalidate some statements (without deleting them) and uncomment them when you need to use them, so you don't have to rewrite them. This is very useful when debugging code and trying to locate program errors.

```
void setup() {
    pinMode(13,OUTPUT);
    // digitalWrite(13,HIGH);
    // delay(1000);
    // digitalWrite(13,LOW);
    // delay(1000);
}
```

(continues on next page)

(continued from previous page)

```
void loop() {  
    digitalWrite(13,HIGH);  
    delay(200);  
    digitalWrite(13,LOW);  
    delay(200);  
}
```

**Note:** Use the shortcut Ctrl+/ to help you quickly comment or uncomment your code.

### Comment `/**/`

Same as `//` for comments. This type of comment can be more than one line long, and once the compiler reads `/*`, it ignores anything that follows until it encounters `*/`.

Example 1:

```
/* Blink */  
  
void setup() {  
    pinMode(13,OUTPUT);  
}  
  
void loop() {  
    /*  
    The following code will blink the onboard LED  
    You can modify the number in delay() to change the blinking frequency  
    */  
    digitalWrite(13,HIGH);  
    delay(500);  
    digitalWrite(13,LOW);  
    delay(500);  
}
```

### `#define`

This is a useful C++ tool.

```
#define identifier token-string
```

The compiler automatically replaces `identifier` with `token-string` when it reads it, which is usually used for constant definitions.

As an example, here is a sketch that uses `define`, which improves the readability of the code.

```
#define ONBOARD_LED 13  
#define DELAY_TIME 500  
  
void setup() {  
    pinMode(ONBOARD_LED,OUTPUT);
```

(continues on next page)

(continued from previous page)

```

}

void loop() {
    digitalWrite(ONBOARD_LED,HIGH);
    delay(DELAY_TIME);
    digitalWrite(ONBOARD_LED,LOW);
    delay(DELAY_TIME);
}

```

To the compiler, it actually looks like this.

```

void setup() {
    pinMode(13,OUTPUT);
}

void loop() {
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}

```

We can see that the identifier is replaced and does not exist inside the program. Therefore, there are several caveats when using it.

1. A token-string can only be modified manually and cannot be converted into other values by arithmetic in the program.
2. Avoid using symbols such as ;. For example.

```

#define ONBOARD_LED 13;

void setup() {
    pinMode(ONBOARD_LED,OUTPUT);
}

void loop() {
    digitalWrite(ONBOARD_LED,HIGH);
}

```

The compiler will recognize it as the following, which is what will be reported as an error.

```

void setup() {
    pinMode(13;,OUTPUT);
}

void loop() {
    digitalWrite(13;,HIGH);
}

```

**Note:** A naming convention for #define is to capitalize identifier to avoid confusion with variables.

### 2.3.7 Variable

The variable is one of the most powerful and critical tools in a program. It helps us to store and call data in our programs.

The following sketch file uses variables. It stores the pin numbers of the on-board LED in the variable `ledPin` and a number “500” in the variable `delayTime`.

```
int ledPin = 13;
int delayTime = 500;

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

void loop() {
  digitalWrite(ledPin,HIGH);
  delay(delayTime);
  digitalWrite(ledPin,LOW);
  delay(delayTime);
}
```

Wait, is this a duplicate of what `#define` does? The answer is NO.

- The role of `#define` is to simply and directly replace text, it is not considered by the compiler as part of the program.
- A variable, on the other hand, exists within the program and is used to store and call value. A variable can also modify its value within the program, something that a define cannot do.

The sketch file below self-adds to the variable and it will cause the on-board LED to blink longer after each blink.

```
int ledPin = 13;
int delayTime = 500;

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

void loop() {
  digitalWrite(ledPin,HIGH);
  delay(delayTime);
  digitalWrite(ledPin,LOW);
  delay(delayTime);
  delayTime = delayTime+200; //Each execution increments the value by 200
}
```

## Declare a variable

Declaring a variable means creating a variable.

To declare a variable, you need two things: the data type, and the variable name. The data type needs to be separated from the variable by a space, and the variable declaration needs to be terminated by a ;.

Let's use this variable as an example.

```
int delayTime;
```

### Data Type

Here `int` is a data type called integer type, which can be used to store integers from -32768 to 32766. It can also not be used to store decimals.

Variables can hold different kinds of data other than integers. The Arduino language (which, remember, is C++) has built-in support for a few of them (only the most frequently used and useful are listed here):

- `float`: Store a decimal number, for example 3.1415926.
- `byte`: Can hold numbers from 0 to 255.
- `boolean`: Holds only two possible values, `True` or `False`, even though it occupies a byte in memory.
- `char`: Holds a number from -127 to 127. Because it is marked as a `char` the compiler will try to match it to a character from the .
- `string`: Can stores a string of characters, e.g. Halloween.

### Variable Name

You can set the variable to any name you want, such as `i`, `apple`, `Bruce`, `R2D2`, `Sectumsempra`, but there are some basic rules to follow.

1. describe what it is used for. Here, I named the variable `delayTime`, so you can easily understand what it does. It works fine if I name the variable `barryAllen`, but it confuses the person looking at the code.
2. Use regular nomenclature. You can use CamelCase like I did, with the initial T in `delayTime` so that it is easy to see that the variable consists of two words. Also, you can use UnderScoreCase to write the variable as `delay_time`. It doesn't affect the program's running, but it would help the programmer to read the code if you use the nomenclature you prefer.
3. Don't use keywords. Similar to what happens when we type "`int`", the Arduino IDE will color it to remind you that it is a word with a special purpose and cannot be used as a variable name. Change the name of the variable if it is colored.
4. Special symbols are not allowed. For example, space, `#`, `$`, `/`, `+`, `%`, etc. The combination of English letters (case sensitive), underscores, and numbers (but numbers cannot be used as the first character of a variable name) is rich enough.

### Assign a value to a variable

Once we have declared the variable, it is time to store the data. We use the assignment operator (i.e. `=`) to put value into the variable.

We can assign values to the variable as soon as we declare it.

```
int delayTime = 500;
```

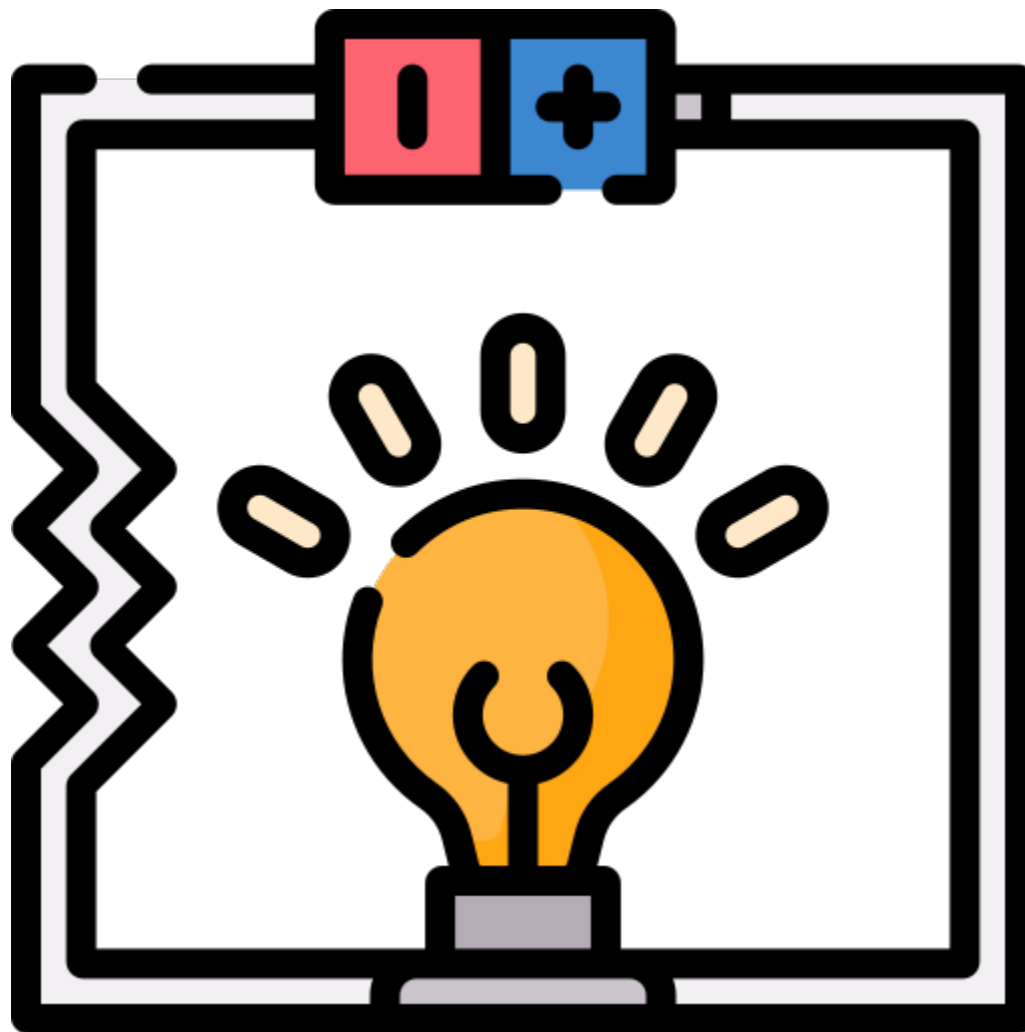
It is also possible to assign a new value to it at some time.

```
int delayTime; // no value
delayTime = 500; // value is 500
delayTime = delayTime +200; // value is 700
```

### 2.3.8 How to Build the Circuit

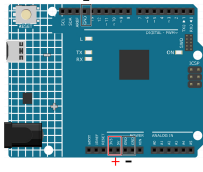
Many of the things you use every day are powered by electricity, like the lights in your house and the computer you're reading.

To use electricity, you must build an electrical circuit. Basically, a circuit is a path through which electricity flows, or an electronic circuit, and is made up of electrical devices and components (appliances) that are connected in a certain way, such as resistors, capacitors, power supplies, and switches.



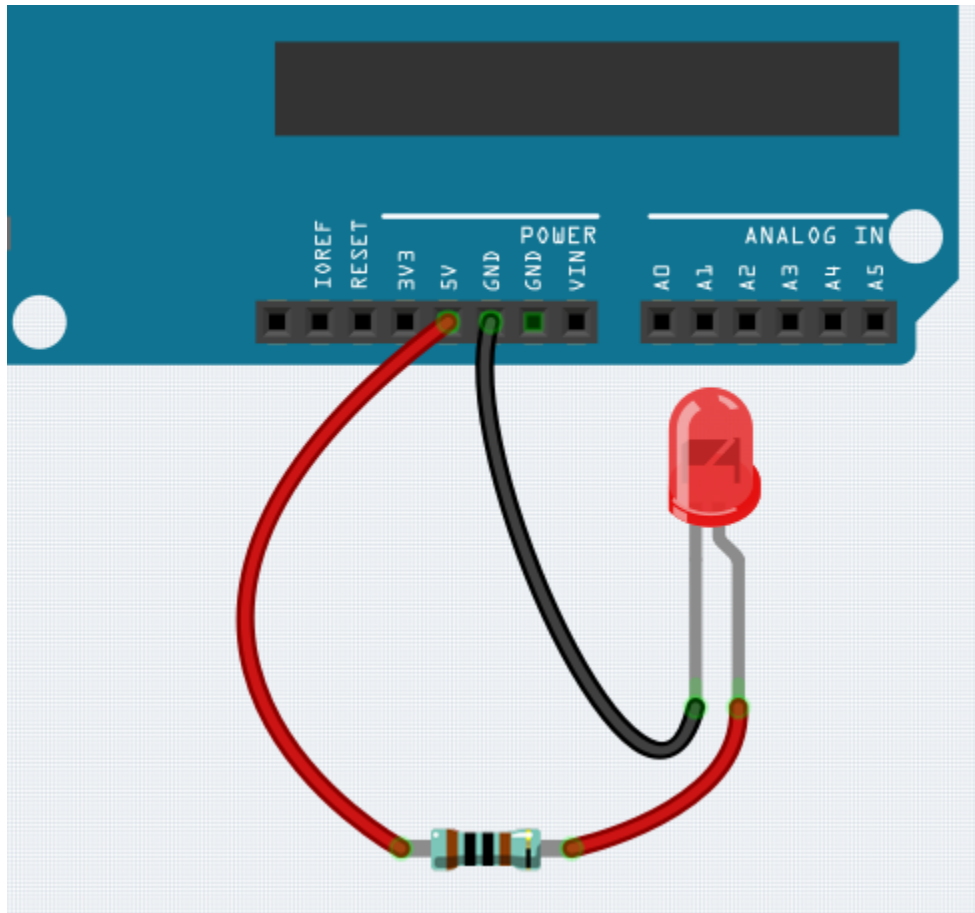
A circuit is a closed path in which electrons move to create an electric current. To flow current, there must be a conducting path between the positive terminal of the power supply and the negative terminal, which is called a closed circuit (if it is broken, it is called an open circuit.) .

The Arduino Board has some power output pins (positive) and some ground pins (negative). You can use these pins as the positive and negative sides of the power supply by plugging the power source into the board.



With electricity, you can create works with light, sound, and motion. You can light up an LED by connecting the long pin to the positive terminal and the short pin to the negative terminal. The LED will break down very quickly if you do this, so you need to add a 1k\* resistor inside the circuit to protect it.

The circuit they form is shown below.



You may have questions this time: how do I build this circuit? Hold the wires by hand, or tape the pins and wires?

In this situation, solderless breadboards will be your strongest allies.

### Hello, Breadboard!

A breadboard is a rectangular plastic plate with a bunch of small holes. These holes allow us to easily insert electronic components and build electronic circuits. Breadboards do not permanently fix electronic components, so we can easily repair a circuit and start over if something goes wrong.

**Note:** There is no need for special tools to use breadboards. However, many electronic components are very small, and a pair of tweezers can help us to pick up small parts better.

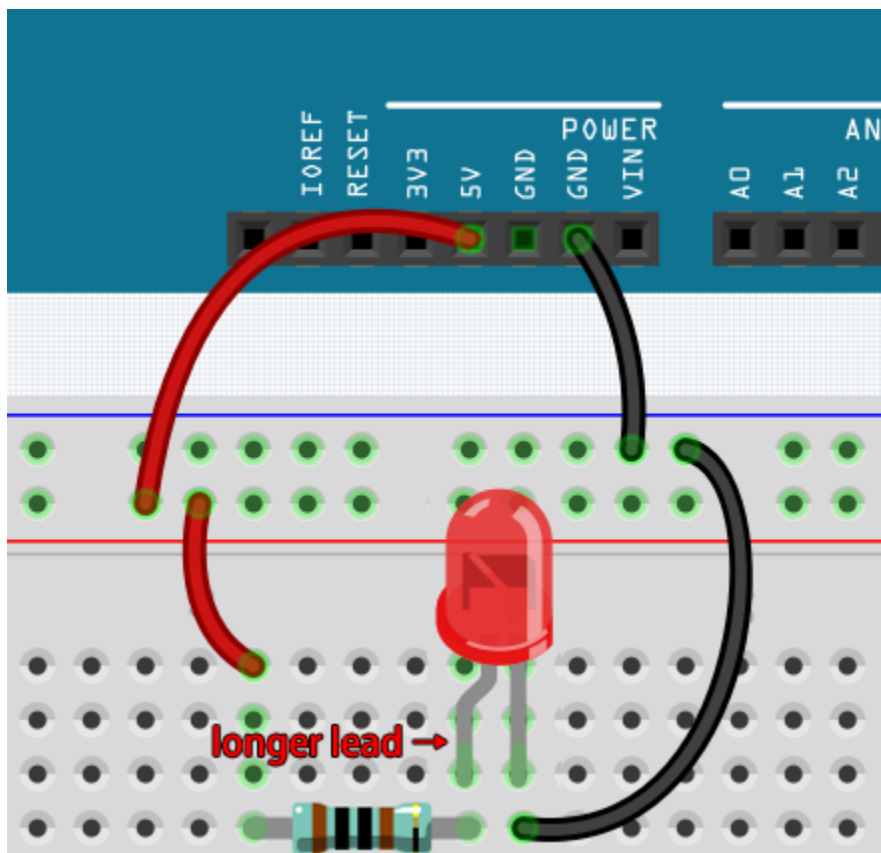
On the Internet, we can find a lot of information about breadboards.

- [How to Use a Breadboard - Science Buddies](#)
- [What is a BREADBOARD? - Makezine](#)

Here are some things you should know about breadboards.

1. Each half-row group (such as column A-E in row 1 or column F-J in row 3) is connected. Therefore, if an electrical signal flows in from A1, it can flow out from B1, C1, D1, E1, but not from F1 or A2.
2. In most cases, both sides of the breadboard are used as power buses, and the holes in each column (about 50 holes) are connected together. As a general rule, positive power supplies are connected to the holes near the red wire, and negative power supplies are connected to the holes near the blue wire.
3. In a circuit, current flows from the positive pole to the negative pole after passing through the load. In this case, a short circuit may occur.

Let us follow the direction of the current to build the circuit!





1. In this circuit, we use the 5V pin of the board to power the LED. Use a male-to-male (M2M) jumper wire to connect it to the red power bus.
2. To protect the LED, the current must pass through a 1k ohm resistor. Connect one end (either end) of the resistor to the red power bus, and the other end to the free row of the breadboard.

---

**Note:** The color ring of the 1k ohm resistor is brown, black, black, brown and brown.

---

3. If you pick up the LED, you will see that one of its leads is longer than the other. Connect the longer lead to the same row as the resistor, and the shorter lead to the other row.

---

**Note:** The longer lead is the anode, which represents the positive side of the circuit; the shorter lead is the cathode, which represents the negative side.

The anode needs to be connected to the GPIO pin through a resistor; the cathode needs to be connected to the GND pin.

---

4. Using a male-to-male (M2M) jumper wire, connect the LED short pin to the breadboard's negative power bus.
5. Connect the GND pin of board to the negative power bus using a jumper.

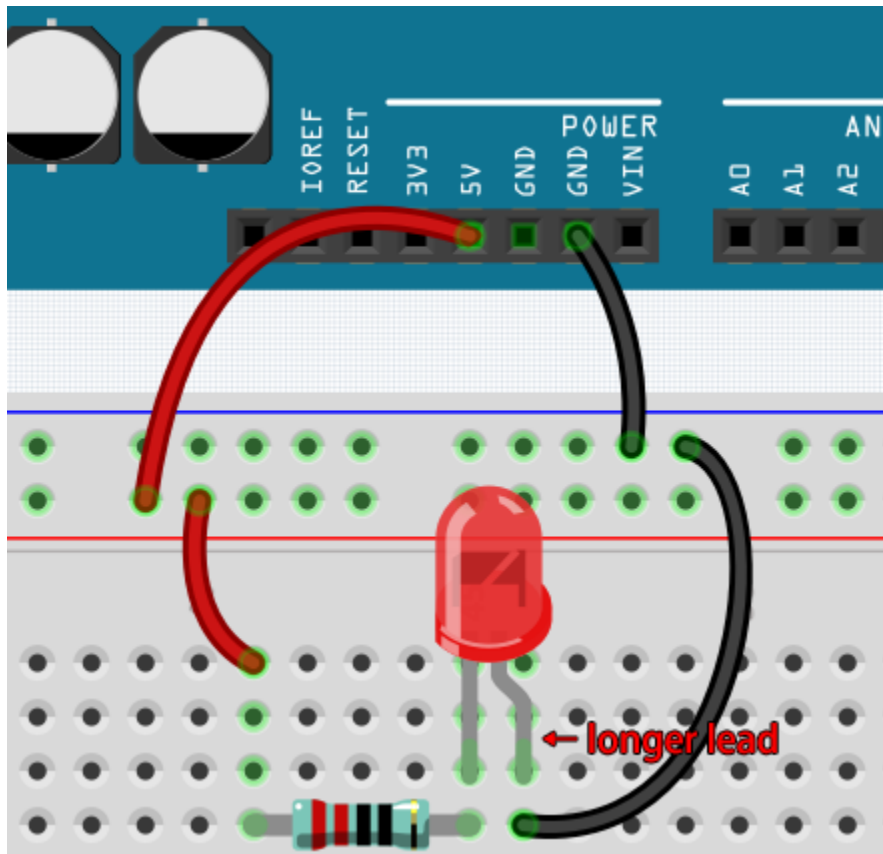
### **Beware of short circuits**

Short circuits can occur when two components that shouldn't be connected are "accidentally" connected. This kit includes resistors, transistors, capacitors, LEDs, etc. that have long metal pins that can bump into each other and cause a short. Some circuits are simply prevented from functioning properly when a short occurs. Occasionally, a short circuit can damage components permanently, especially between the power supply and the ground bus, causing the circuit to get very hot, melting the plastic on the breadboard and even burning the components!

Therefore, always make sure that the pins of all the electronics on the breadboard are not touching each other.

### **Direction of the circuit**

There is an orientation to circuits, and the orientation plays a significant role in certain electronic components. There are some devices with polarity, which means they must be connected correctly based on their positive and negative poles. Circuits built with the wrong orientation will not function properly.



If you reverse the LED in this simple circuit that we built earlier, you will find that it no longer works.

In contrast, some devices have no direction, such as the resistors in this circuit, so you can try inverting them without affecting the LEDs' normal operation.

Most components and modules with labels such as “+”, “-”, “GND”, “VCC” or have pins of different lengths must be connected to the circuit in a specific way.

### Protection of the circuit

Current is the rate at which electrons flow past a point in a complete electrical circuit. At its most basic, current = flow. An ampere (AM-pir), or amp, is the international unit used for measuring current. It expresses the quantity of electrons (sometimes called “electrical charge”) flowing past a point in a circuit over a given time.

The driving force (voltage) behind the flow of current is called voltage and is measured in volts (V).

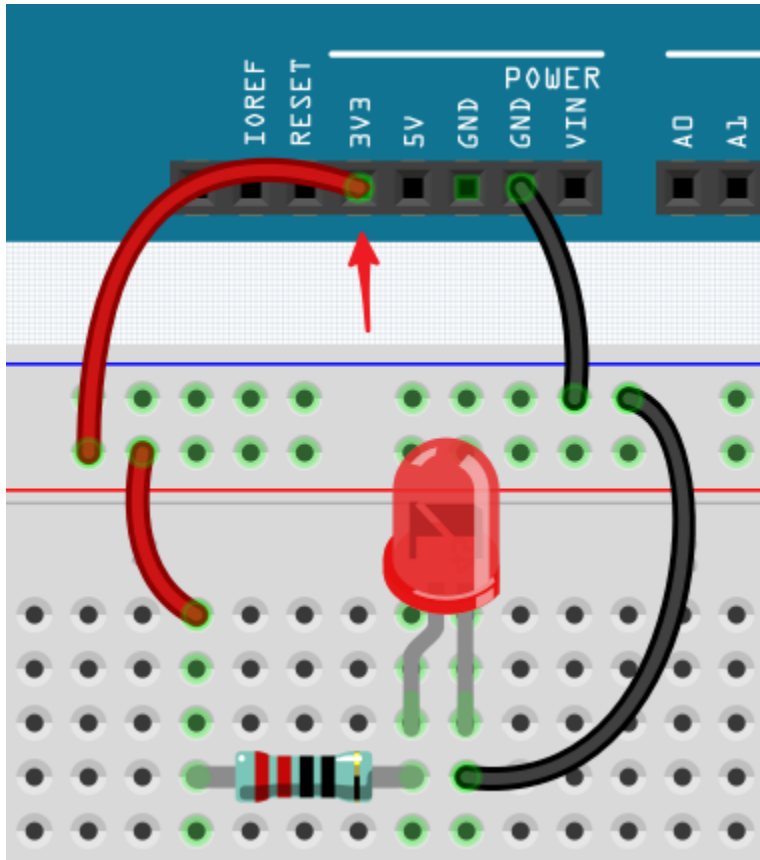
Resistance (R) is the property of the material that restricts the flow of current, and it is measured in ohms ( $\Omega$ ).

According to Ohm's law (as long as the temperature remains constant), current, voltage, and resistance are proportional. A circuit's current is proportional to its voltage and inversely proportional to its resistance.

Therefore, current (I) = voltage (V) / resistance (R).

- [Ohm's law - Wikipedia](#)

About Ohm's law we can do a simple experiment.



By changing the wire connecting 5V to 3.3V, the LED gets dimmer. If you change the resistor from 1kohm to 2kohm, you will notice that the LED becomes dimmer than before. The larger the resistor, the dimmer the LED.

---

**Note:** For an introduction to resistors and how to calculate resistance values, see [Resistor](#).

---

Most packaged modules only require access to the proper voltage (usually 3.3V or 5V), such as ultrasonic module.

However, in your self-built circuits, you need to be aware of the supply voltage and resistor usage for electrical devices.

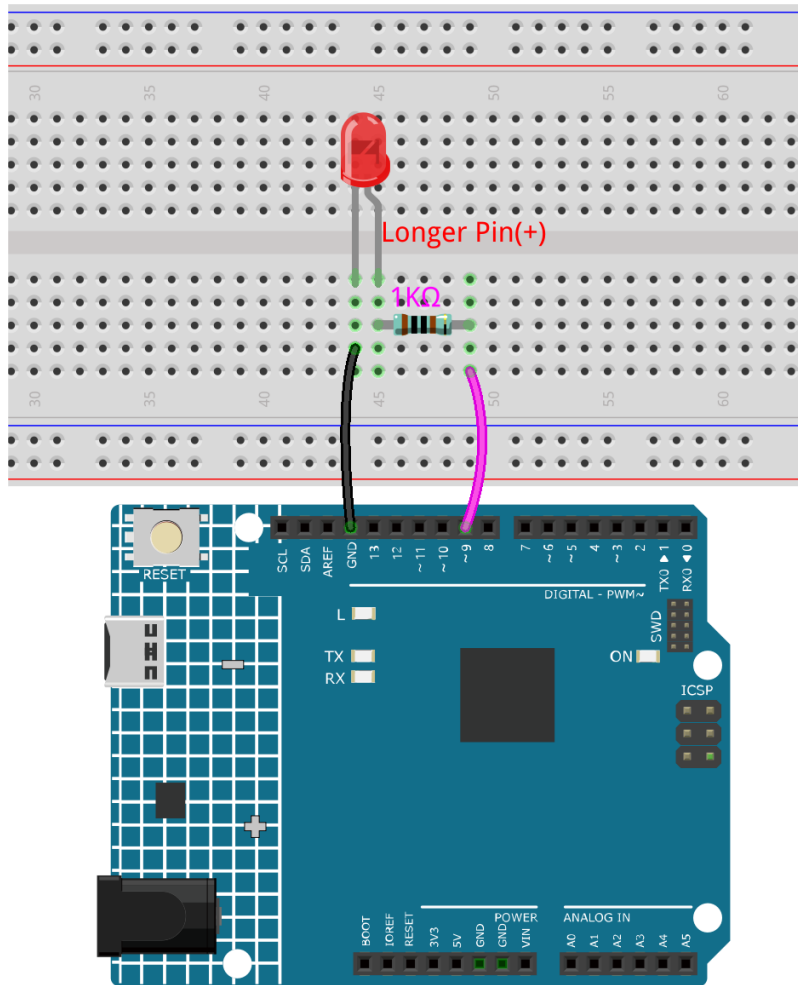
As an example, LEDs usually consume 20mA of current, and their voltage drop is about 1.8V. According to Ohm's law, if we use 5V power supply, we need to connect a minimum of 160ohm  $((5-1.8)/20\text{mA})$  resistor in order not to burn out the LED.

### Control circuit with Arduino

Now that we have a basic understanding of Arduino programming and electronic circuits, it's time to face the most critical question: How to control circuits with Arduino.

Simply put, the way Arduino controls a circuit is by changing the level of the pins on the board. For example, when controlling an on-board LED, it is writing a high or low level signal to pin 13.

Now let's try to code the Arduino board to control the blinking LED on the breadboard. Build the circuit so that the LED is connected to pin 9.



Next, upload this sketch to the Arduino development board.

```
int ledPin = 9;
int delayTime = 500;

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

void loop() {
  digitalWrite(ledPin,HIGH);
  delay(delayTime);
  digitalWrite(ledPin,LOW);
  delay(delayTime);
}
```

This sketch is very similar to the one we used to control the blinking of the on-board LED, the difference is that the value of `ledPin` has been changed to 9. This is because we are trying to control the level of pin 9 this time.

Now you can see the LED on the breadboard blinking.

## DOWNLOAD THE CODE

Download the relevant code from the link below.

- SunFounder 3 in 1 Kit for Arduino
- Or check out the code at [SunFounder 3 in 1 Kit for Arduino - GitHub](#)



## BASIC PROJECTS

This chapter is used to learn how to control electronic circuits using Arduino.

Depending on the components, the basic control methods of Arduino can be divided into four types:

- *1. Digital Write*: Set the output voltage of the pin to be high or low, which can be used to turn the light on and off.
- *2. Analog Write*: Write the analog value (*PWM wave*) to the pin, which can be used to adjust the brightness of the light.
- *3. Digital Read*: Read the level signal of the digital pin, which can be used to read the working condition of the switch.
- *4. Analog Read*: Read the voltage of the analog pin, which can be used to read the working condition of the knob.

There are also some components that require additional libraries for use, and these are grouped under the section *5.11 Install External Libraries*.

Finally, the kit also provides some *6. Funny Project*, which includes many simple and useful manipulations. Try this section of code and you will understand how most simple projects work.

### 4.1 1. Digital Write

**Digital Write** is to output or write a digital signal to a digital pin. The digital signal has only two states, 0 or 1, 0V or 5V, so it allows some components, such as the LED and buzzer, to be on or off.

On the Arduino R4 board, there are 14 digital I/O pins from 0 to 13, now use the `pinMode()` and `digitalWrite()` functions to write a high or low level to these digital pins.

- `pinMode(pin, mode)`: Configure the specific pin as INPUT or OUTPUT, here it needs to be set as OUTPUT.

#### Syntax

```
pinMode(pin, mode)
```

#### Parameters

- `pin`: the Arduino pin number to set the mode of.
- `mode`: INPUT, OUTPUT, or INPUT\_PULLUP.
- `digitalWrite(pin, value)`: Write a high level (5V) or a low level (0V) to a digital pin to change the operating state of the component. If the pin has been configured as an OUTPUT with `pinMode()`, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

#### Syntax

```
digitalWrite(pin, value)
```

**Parameters**

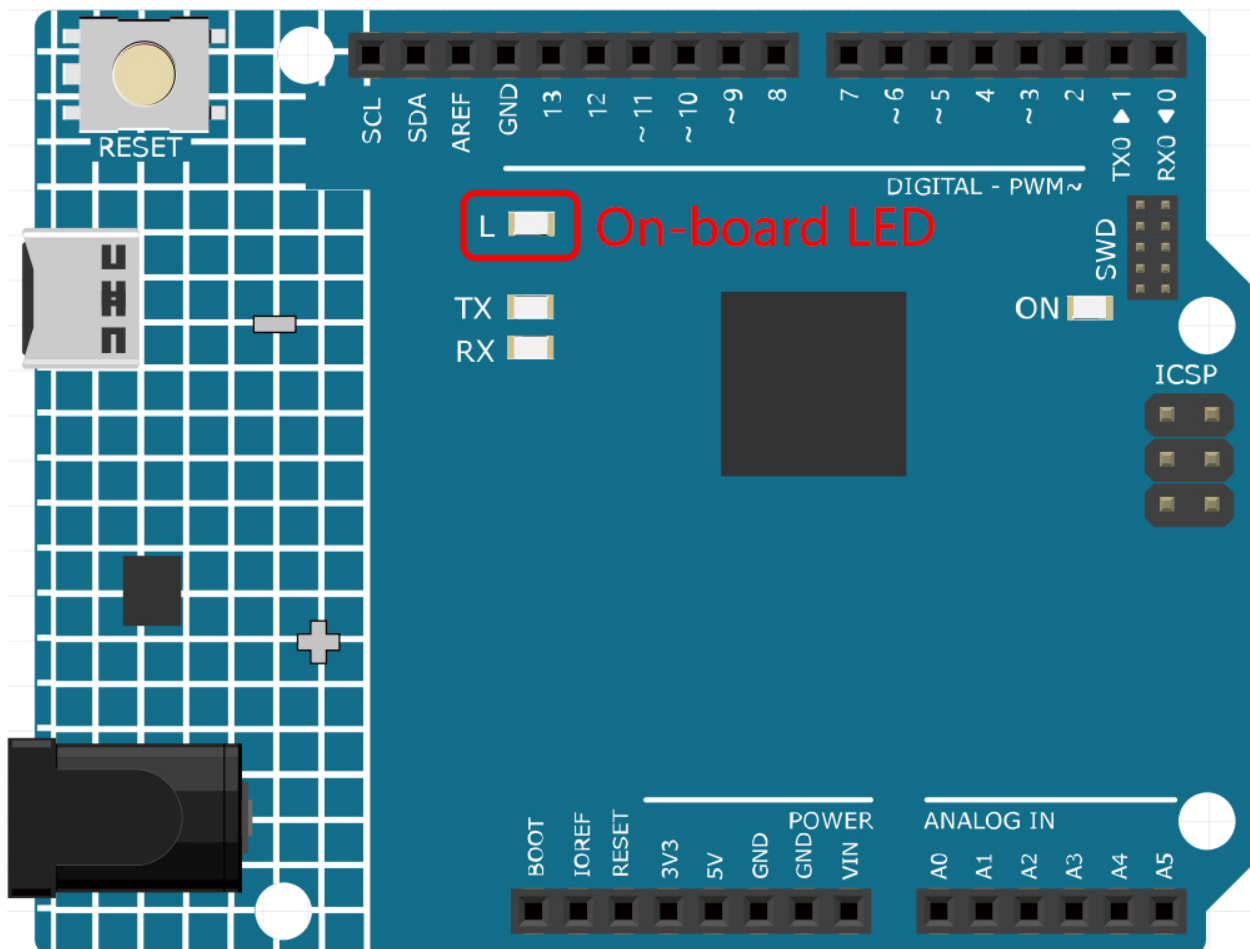
- pin: the Arduino pin number.
- value: HIGH or LOW.

**Example of Digital Write:**

```
const int pin = 13;

void setup() {
  pinMode(pin, OUTPUT);    // sets the digital pin as output
}

void loop() {
  digitalWrite(pin, HIGH); // sets the digital pin on
  delay(1000);             // waits for a second
  digitalWrite(pin, LOW);  // sets the digital pin off
  delay(1000);             // waits for a second
}
```

**Notes and Warnings**

- The pins 0~13 are all digital pins.
- Do not use pins 0 and 1, as they are used to communicate with the computer. Connecting anything to these pins



will interfere with communication, including causing the upload board to fail.

- If the digital pins are used up, the analog pins (A0-A5) can also be used as digital pins.

### Related Components

Below are the related components, you can click in to learn how to use them.

#### 4.1.1 1.1 Hello, LED!

Just as printing “Hello, world!” is the first step in learning to program, using a program to drive an LED is the traditional introduction to learning physical programming.

### Required Components

In this project, we need the following components.

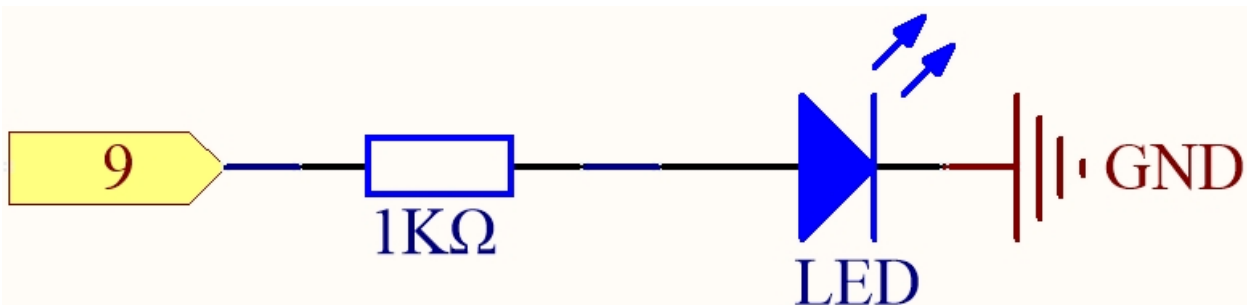
It’s definitely convenient to buy a whole kit, here’s the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

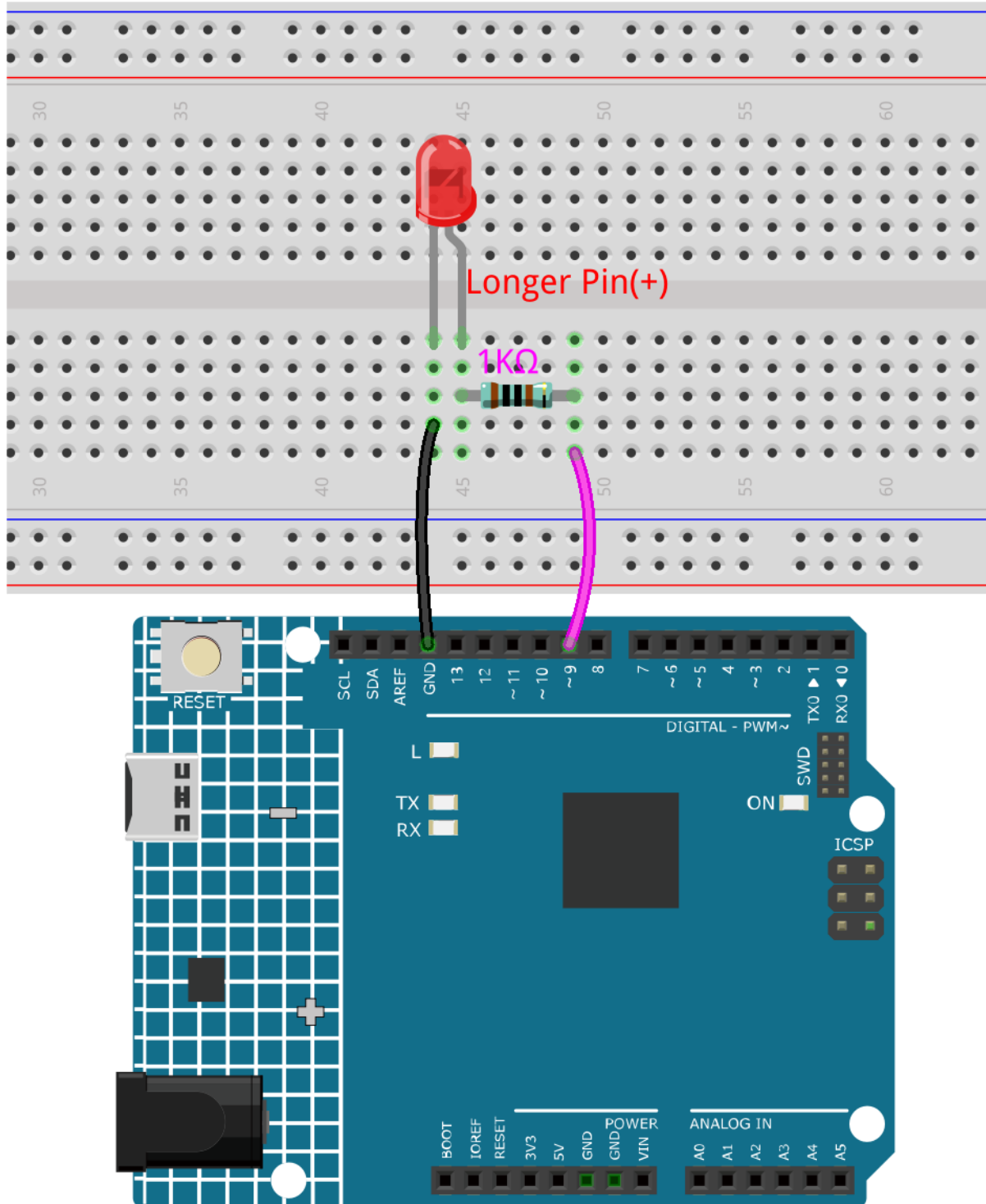
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	

### Schematic



The principle of this circuit is simple and the current direction is shown in the figure. When pin 9 outputs high level(5V), the LED will light up after the 1kohm current limiting resistor. When pin 9 outputs low level (0v), the LED will turn off.

### Wiring



## Code

### Note:

- You can open the file `1.1.hello_led.ino` under the path of `3in1-kit\learning_project\1.1.hello_led`.

- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, you will see the LED blinking.

#### How it works?

Here, we connect the LED to the digital pin 9, so we need to declare an int variable called ledpin at the beginning of the program and assign a value of 9.

```
const int ledPin = 9;
```

Now, initialize the pin in the setup() function, where you need to initialize the pin to OUTPUT mode.

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

In loop(), digitalWrite() is used to provide 5V high level signal for ledpin, which will cause voltage difference between LED pins and light LED up.

```
digitalWrite(ledPin, HIGH);
```

If the level signal is changed to LOW, the ledPin's signal will be returned to 0 V to turn LED off.

```
digitalWrite(ledPin, LOW);
```

An interval between on and off is required to allow people to see the change, so we use a delay(1000) code to let the controller do nothing for 1000 ms.

```
delay(1000);
```

### 4.1.2 1.2 Beep

The active buzzer is a typical digital output device that is as easy to use as lighting up an LED!

Two types of buzzers are included in the kit. We need to use active buzzer. Turn them around, the sealed back (not the exposed PCB) is the one we want.



#### Required Components

In this project, we need the following components.

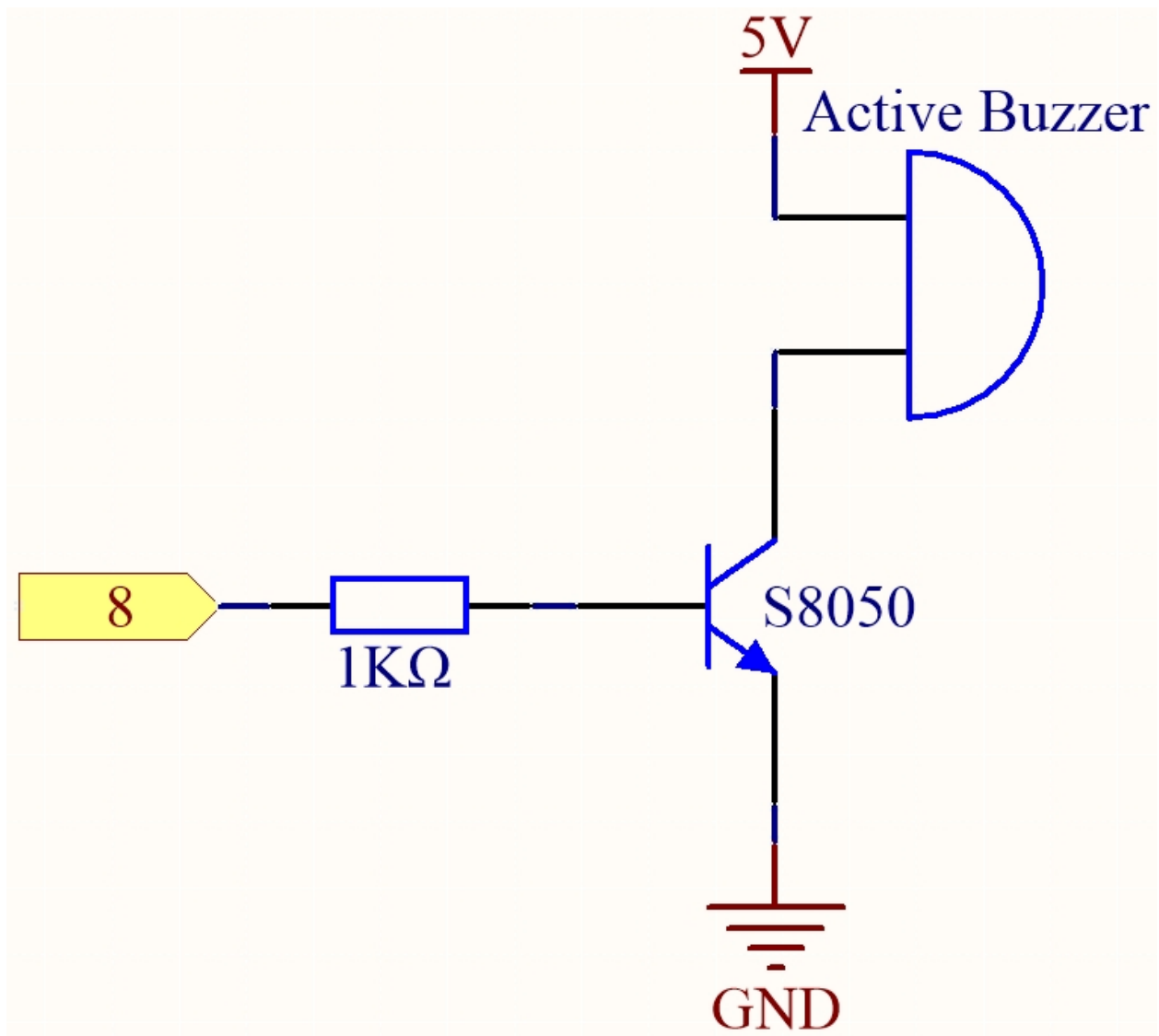
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

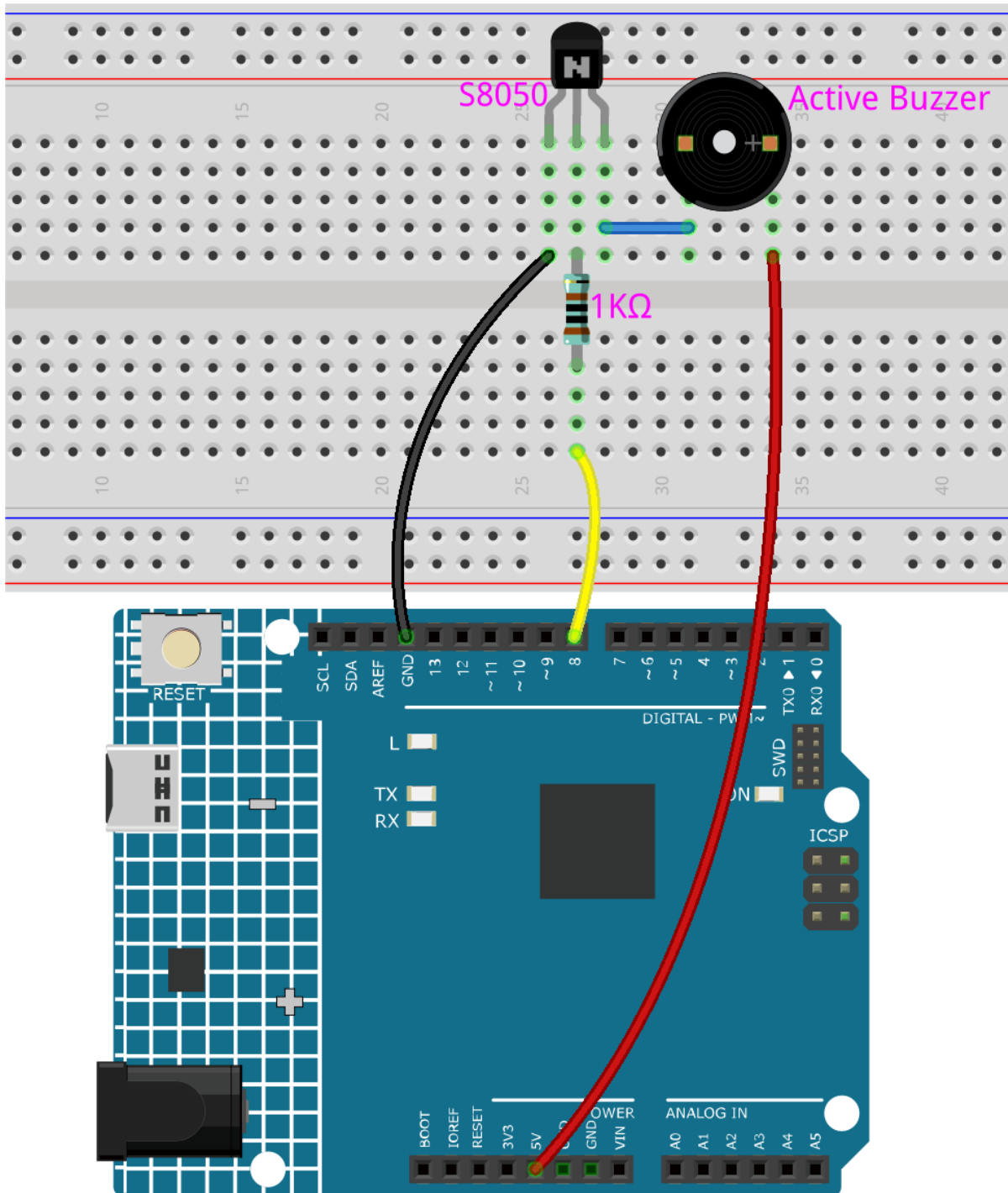
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Buzzer</i>	-

### Schematic



## Wiring



## Code

### Note:

- You can open the file 1.2.beep.ino under the path of 3in1-kit\learning\_project\1.2.beep.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, you will hear a beep every second.

### 4.1.3 1.3 Turn the Wheel

A motor is a typical digital output device, and it is used in the same way as an LED. However, the motor needs to be driven with a large current, and the large current may damage the main control board such as R4 board. Therefore, an motor driver module is used in this occasion, which is a good helper for the R4 board to control the motor safely.

#### Required Components

In this project, we need the following components.

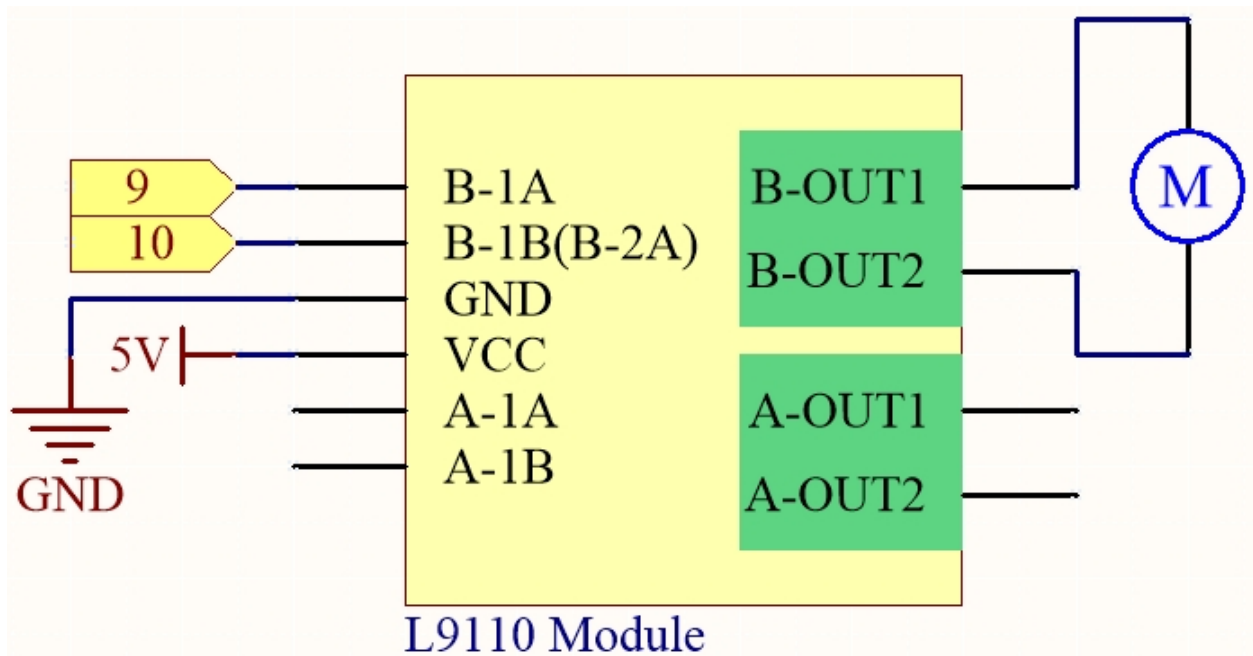
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

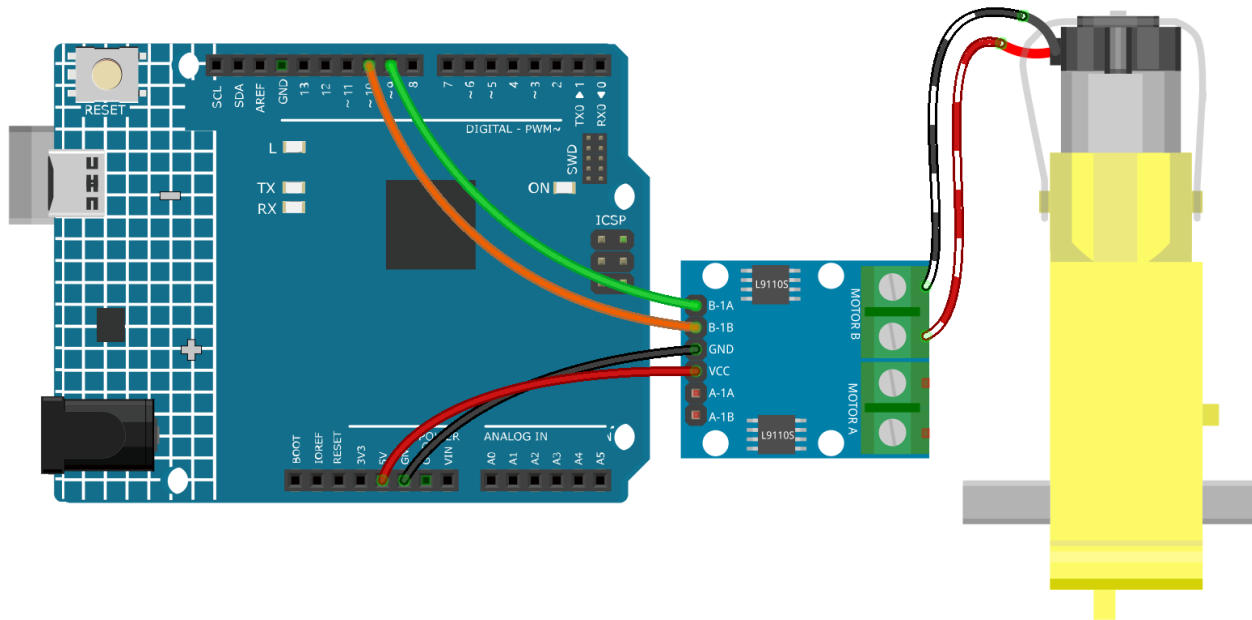
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>TT Motor</i>	-
<i>L9110 Motor Driver Module</i>	-

#### Schematic



#### Wiring

L9110S	R4 Board	Motor
VCC	5V	
GND	GND	
B-1B(B-2A)	9	
B-1A	10	
OA		one wire of the motor
OB		one wire of the motor



## Code

### Note:

- You can open the file 1.3.turn\_the\_wheel.ino under the path of 3in1-kit\learning\_project\1.3.turn\_the\_wheel.
- Or copy this code into **Arduino IDE**.

## 4.1.4 1.4 Pumping

The water pump is also a motor, which converts the mechanical energy of the motor or other external energy through a special structure to transport the liquid.

### Required Components

In this project, we need the following components.

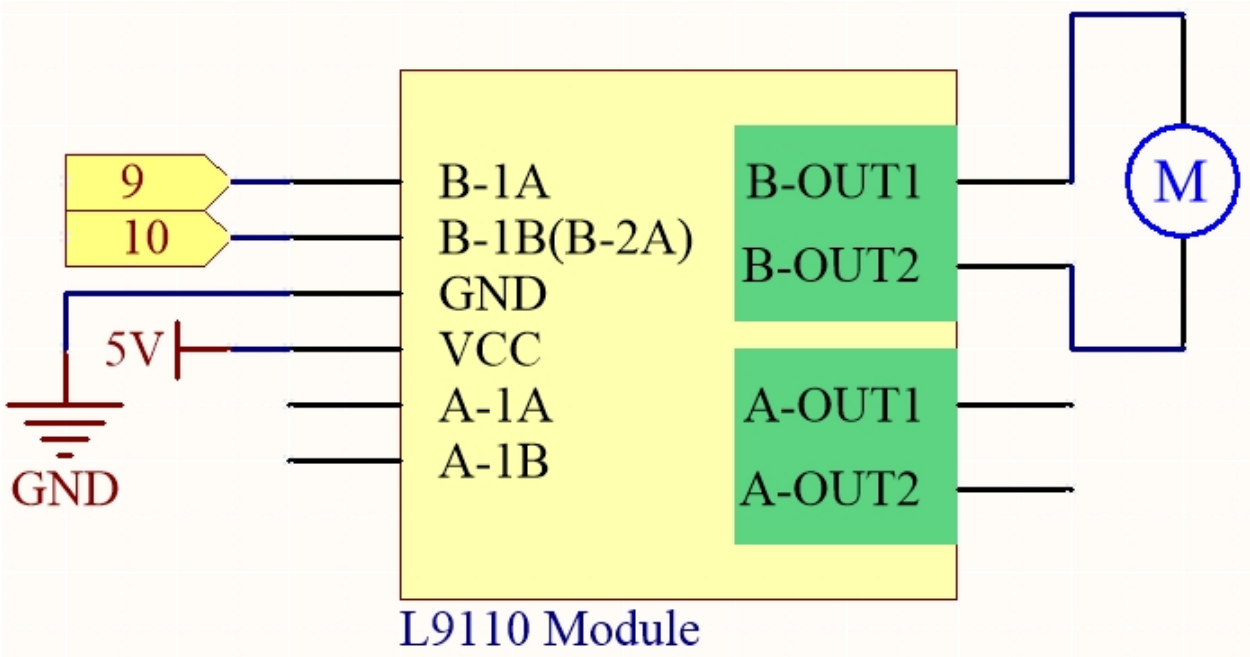
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
Arduino Uno R4 Minima	-
Jumper Wires	
L9110 Motor Driver Module	-
Centrifugal Pump	-

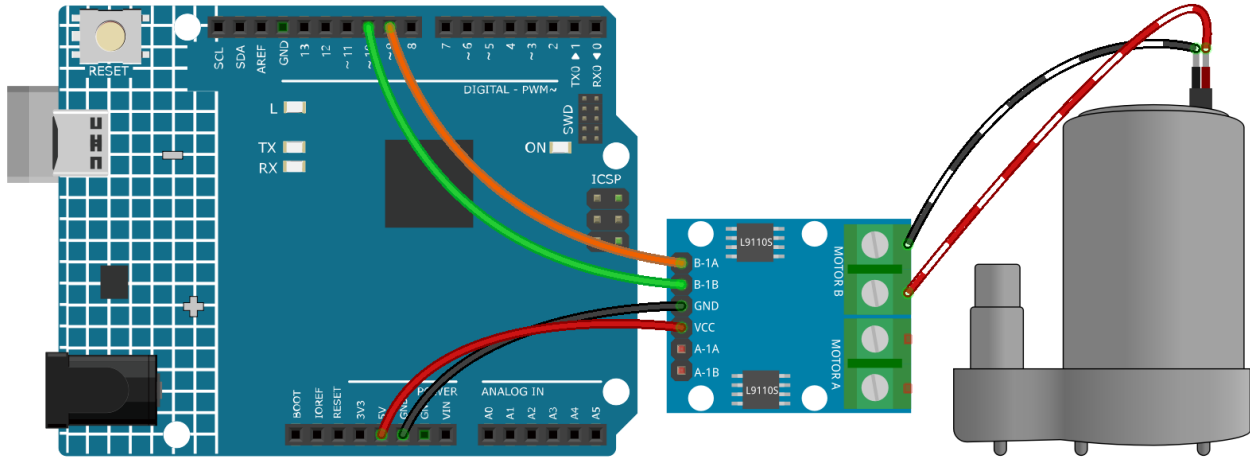
Schematic



Wiring

L9110S	R4 Board	Motor
VCC	5V	
GND	GND	
B-1B(B-2A)	9	
B-1A	10	
OA		one wire of the motor
OB		one wire of the motor





### Code

#### Note:

- You can open the file `1.4.pumping.ino` under the path of `3in1-kit\learning_project\1.4.pumping`.
- Or copy this code into **Arduino IDE**.

Add the tubing to the pump and place it in the basin. After the code is uploaded successfully, you can see that the water in the basin is drained after a while. When doing this experiment, please keep the circuit away from water to avoid short circuit!

## 4.2 2. Analog Write

6 of the Arduino's 14 digital pins also have PWM out function. Therefore, in addition to writing digital signals to these 6 pins, you can also write analog signals (PWM wave signals) to them. This way you can make the LEDs show different brightness or make the motor rotate at different speeds.

Pulse Width Modulation, or **PWM**, is a technique for getting analog results with digital means. Since it may be hard to grasp the literal meaning, here is an example of controlling the intensity of an LED to help you better understand.

A digital signal consisting of high and low levels is called a pulse. The pulse width of these pins can be adjusted by changing the ON/OFF speed. Simply put, when we turn the LED on, off, and on again for a short period of time (like 20ms, the visual dwell time of most people), We won't see that it has gone out, but the brightness of the light will be slightly weaker. During this period, the longer the LED is on, the brighter the LED will be. That is to say, within a period, the wider the pulse, the greater the "electrical signal strength" output by the microcontroller.

This is the function needed to write the PWM wave.

- `analogWrite(pin, value)`

Writes an analog value (PWM wave) to a pin. Different output voltages (0-5V) can be simulated by generating a specified pulse signal. The pin will hold this signal until it is called by a new read or write statement.

#### Syntax

`analogWrite(pin, value)`

#### Parameters

- `pin`: the Arduino pin to write to. Allowed data types: `int`.

- value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

### Example of Analog Write

```
int pin = 9;           //connect to pwm pin

void setup() {
  pinMode(pin, OUTPUT); // sets the pin as output
}

void loop() {
  for (int i = 0 ;i<255 ; i++){
    analogWrite(pin, i); //analogWrite values from 0 to 255
    delay(30);
  }
}
```

### Notes and Warnings

- Looking closely at the R4 board, the pins marked with the “~” symbol have analog output function.
- The PWM outputs generated on pins 5 and 6 will have higher-than-expected duty cycles. This is because of interactions with the `millis()` and `delay()` functions, which share the same internal timer used to generate those PWM outputs. This will be noticed mostly on low duty-cycle settings (e.g. 0 - 10) and may result in a value of 0 not fully turning off the output on pins 5 and 6.

### Related Components

Below are the related components, you can click in to learn how to use them.

## 4.2.1 2.1 Fading

This project is similar to [1.1 Hello, LED!](#), the difference is the signal type. The former is to make the LED light on or off by outputting a digital signal (0&1), this project is to control the brightness of the LED by outputting an analog signal.

### Required Components

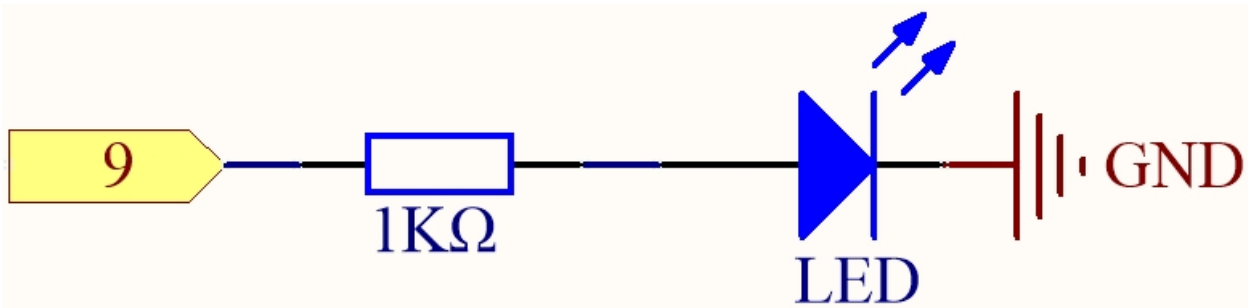
In this project, we need the following components.

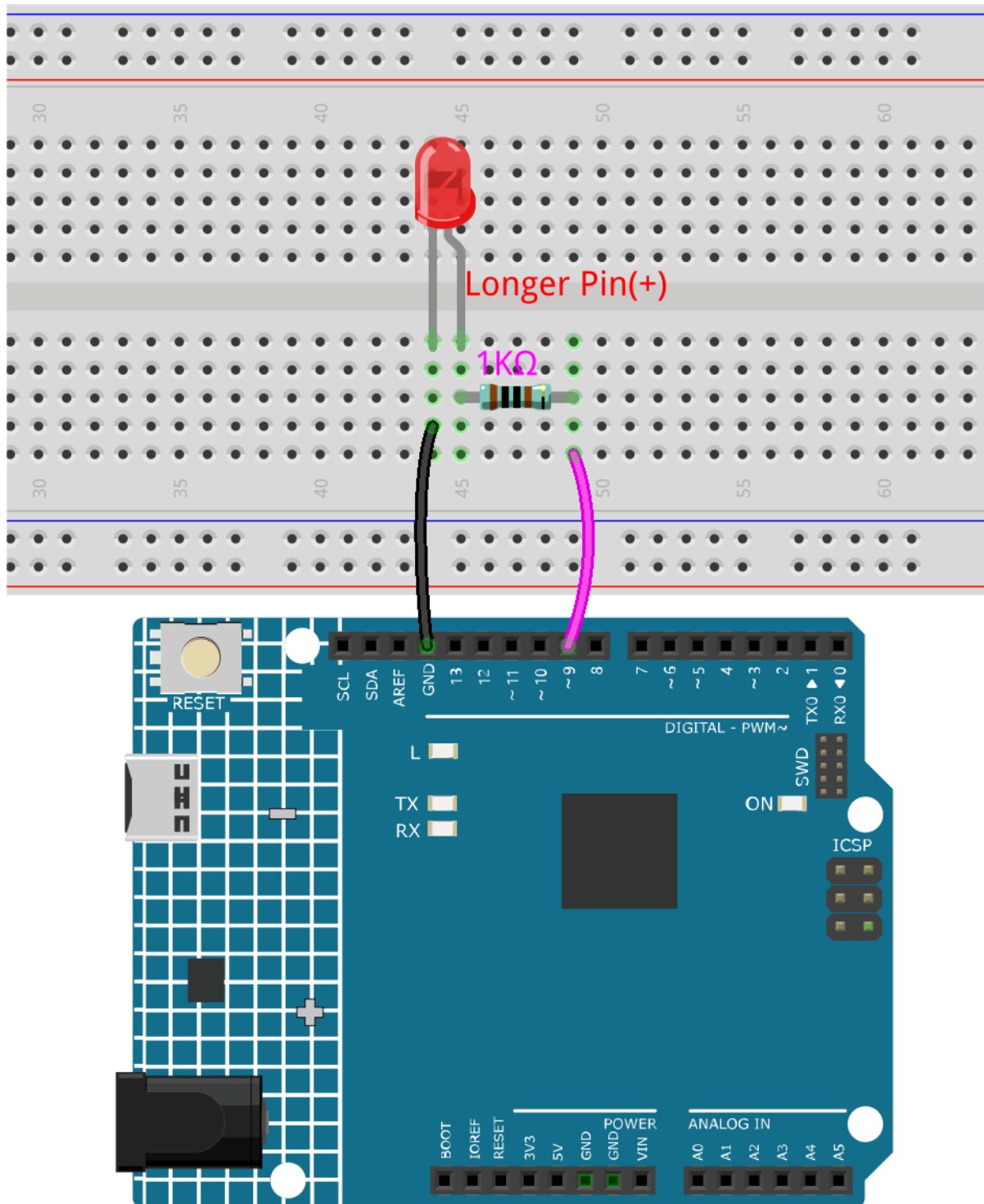
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	

**Schematic****Wiring**



## Code

### Note:

- You can open the file `2.1.fading.ino` under the path of `3in1-kit\learning_project\2.analogWrite\2.1.fading`.

- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, you can see the LED breathing.

### 4.2.2 2.2 Colorful Light

As we know, light can be superimposed. For example, mix blue light and green light give cyan light, red light and green light give yellow light. This is called “The additive method of color mixing”.

- [Additive color - Wikipedia](#)

Based on this method, we can use the three primary colors to mix the visible light of any color according to different specific gravity. For example, orange can be produced by more red and less green.

In this chapter, we will use RGB LED to explore the mystery of additive color mixing!

RGB LED is equivalent to encapsulating Red LED, Green LED, Blue LED under one lamp cap, and the three LEDs share one cathode pin. Since the electric signal is provided for each anode pin, the light of the corresponding color can be displayed. By changing the electrical signal intensity of each anode, it can be made to produce various colors.

#### Required Components

In this project, we need the following components.

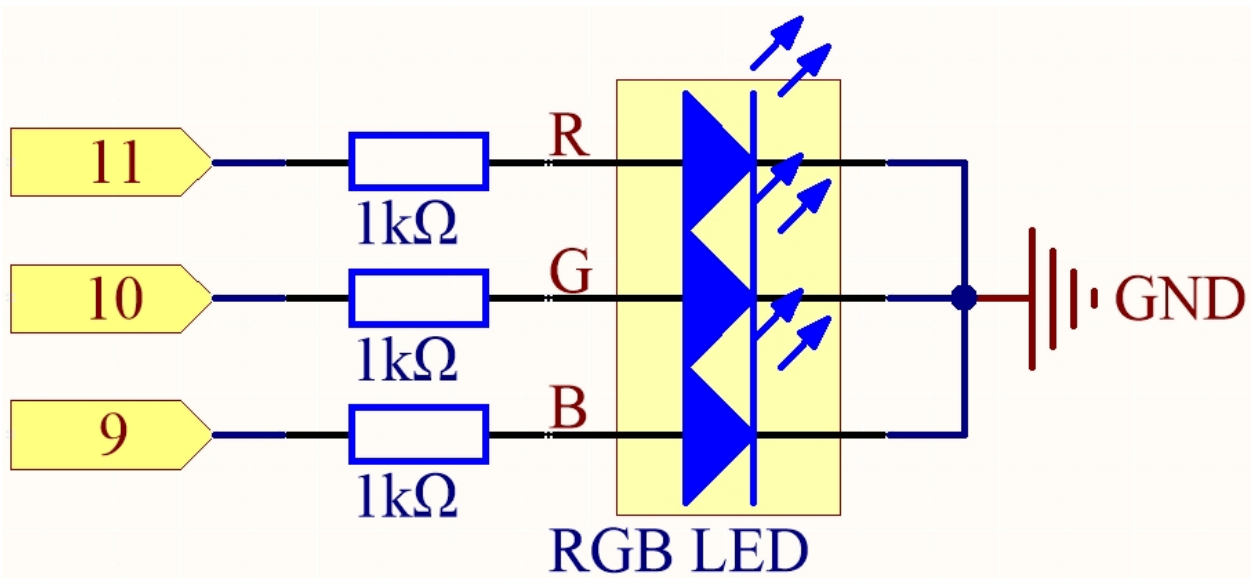
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

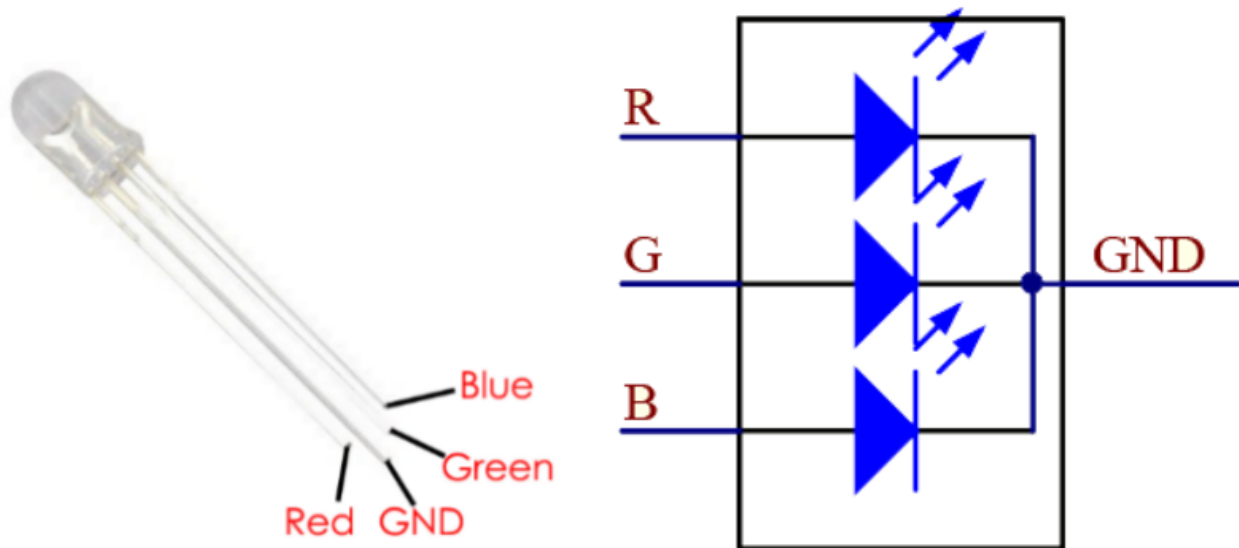
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>RGB LED</i>	

#### Schematic

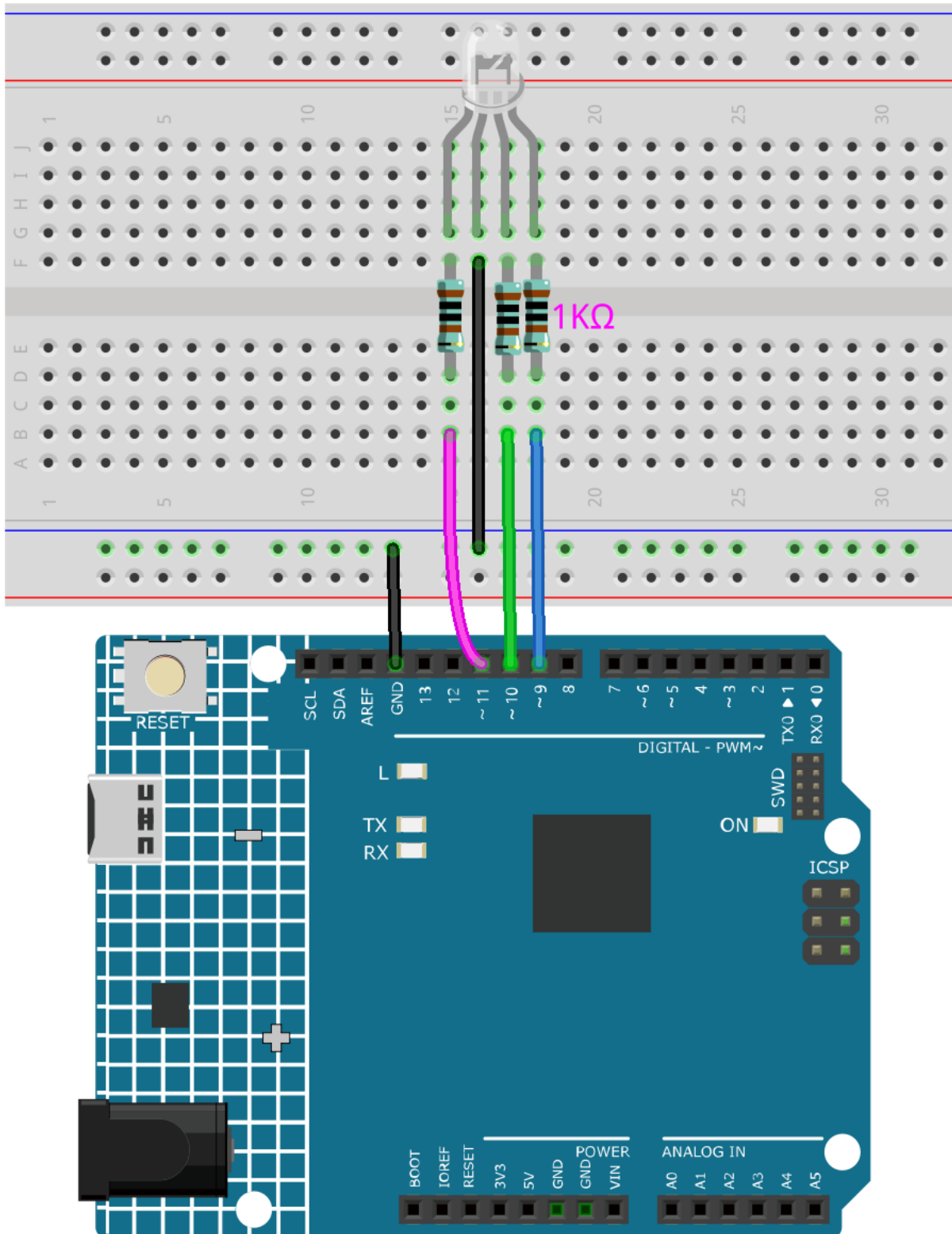


The PWM pins 11, 10 and 9 control the Red, Green and Blue pins of the RGB LED respectively, and connect the common cathode pin to GND. This allows the RGB LED to display a specific color by superimposing light on these pins with different PWM values.

#### Wiring



An RGB LED has 4 pins: the longest pin is the common cathode pin, which is usually connected to GND, the left pin next to the longest pin is Red, and the 2 pins on the right are Green and Blue.

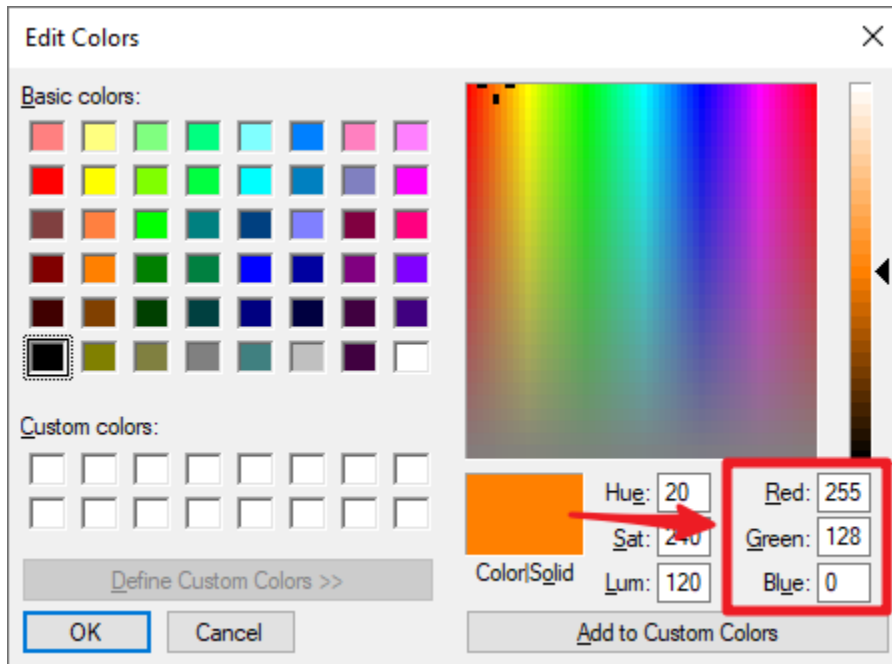


### Code

Here, we can choose our favorite color in drawing software (such as paint) and display it with RGB LED.

**Note:**

- You can open the file 2.2.colorful\_light.ino under the path of 3in1-kit\learning\_project\2.analogWrite\2.2.colorful\_light.
- Or copy this code into **Arduino IDE**.



Write the RGB value into `color_set()`, you will be able to see the RGB light up the colors you want.

**How it works?**

In this example, the function used to assign values to the three pins of RGB is packaged in an independent subfunction `color()`.

```
void color (unsigned char red, unsigned char green, unsigned char blue)
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

In `loop()`, RGB value works as an input argument to call the function `color()` to realize that the RGB can emit different colors.

```
void loop()
{
    color(255, 0, 0); // red
    delay(1000);
    color(0, 255, 0); // green
    delay(1000);
    color(0, 0, 255); // blue
    delay(1000);
}
```



## 4.3 3. Digital Read

Sensors capture real-world information, which is then communicated to the main board via pins (some digital, some analog) so that the computer can know the reality of the situation.

Therefore, the Arduino board can know the working status of digital sensors by reading the value of digital pins like buttons, IR obstacle avoidance module.

Here are the required functions.

- `pinMode(pin, mode)`: Configure the specific pin as INPUT or OUTPUT, here it needs to be set as INPUT.

### Syntax

```
pinMode(pin, mode)
```

### Parameters

- `pin`: the Arduino pin number to set the mode of.
- `mode`: INPUT, OUTPUT, or INPUT\_PULLUP.

- `digitalRead(pin)`: Read the value (level state) from the specified digital pin.

### Syntax

```
digitalRead(pin)
```

### Parameters

- `pin`: the Arduino pin number you want to read

### Returns

HIGH or LOW

### Example of Digital Read

```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output
  pinMode(inPin, INPUT);   // sets the digital pin 7 as input
}

void loop() {
  val = digitalRead(inPin); // read the input pin
  digitalWrite(ledPin, val); // sets the LED to the button's value
}
```

### Notes and Warnings

1. Pull Up & Pull Down.

`digitalRead()` may produce random, indeterminate values if the pin is not getting a level signal. So directing the input pins to a known state can make the project more reliable. When using an input component such as a button, it is usually necessary to connect a pull-up or pull-down resistor in parallel to the digital input pin.

Apart from connecting a pull-up resistor, you can also set the pin mode to `INPUT_PULLUP` in the code, for example `pinMode(pin, INPUT_PULLUP)`. In this case, the pin will access the Atmega's built-in pull-up resistor via software, and it will have the same effect as connecting a pull-up resistor.

## 2. About Pin13.

All digital pins (1-13) on the R4 board can be used as `digitalRead()`. But digital pin 13 is more difficult to use as a digital input than other digital pins. Because it connects an LED and resistor, it is soldered on most boards. If you enable its internal 20k pull-up resistor, it will hang around 1.7V instead of the expected 5V because the onboard LED and series resistor pull the voltage level low, which means it always returns LOW. If you must use pin 13 as a digital input, set its `pinMode()` to INPUT and use an external pull-down resistor.

## 3. Analog pins.

If the digital pins are not enough, the analog pins (A0-A5) can also be used as digital pins. It needs to be set to INPUT with `pinMode(pin,mode)`.

### Related Components

Below are the related components, you can click in to learn how to use them.

## 4.3.1 3.0 Serial Monitor

In the Arduino IDE, there is a serial monitor that allows you to send messages from your computer to the Arduino board (via USB) and also to receive messages from the Arduino.

So in this project we will learn how to receive data from the Arduino board.

---

**Note:** On Uno, Nano, Mini, and Mega, pins 0 and 1 are used for communication with the computer. Connecting anything to these pins can interfere with that communication, including causing failed uploads to the board.

---

### Using the Serial Monitor

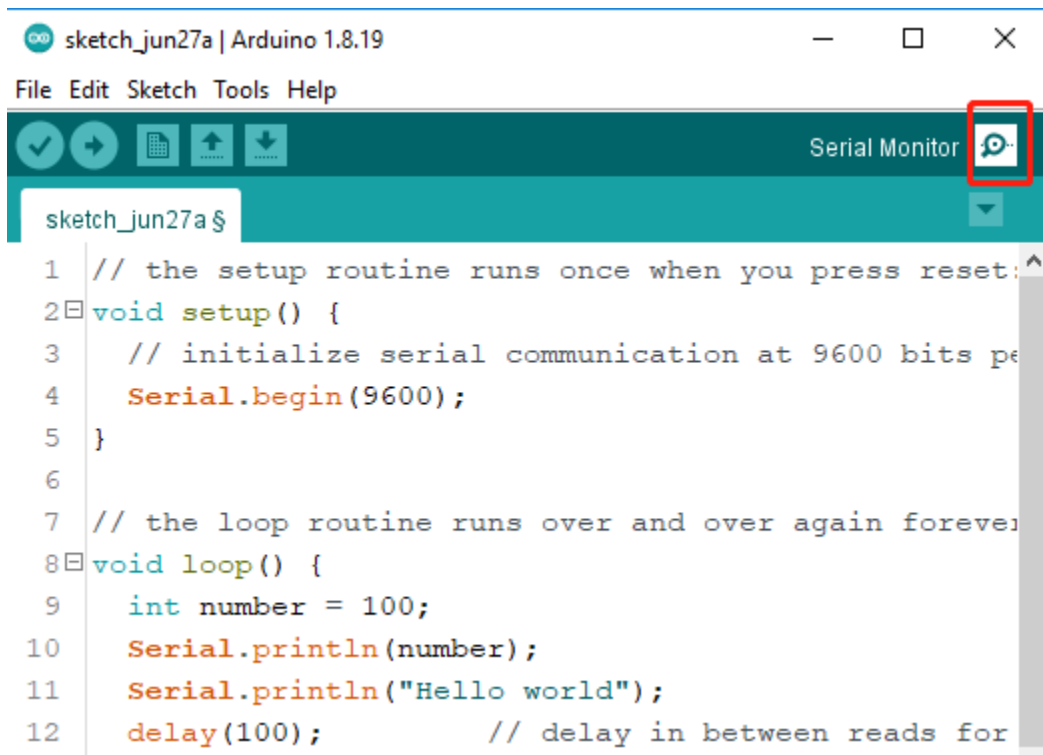
1. Open the Arduino IDE, and paste the following code in it.

```
// the setup routine runs once when you press reset:
void setup() {
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}

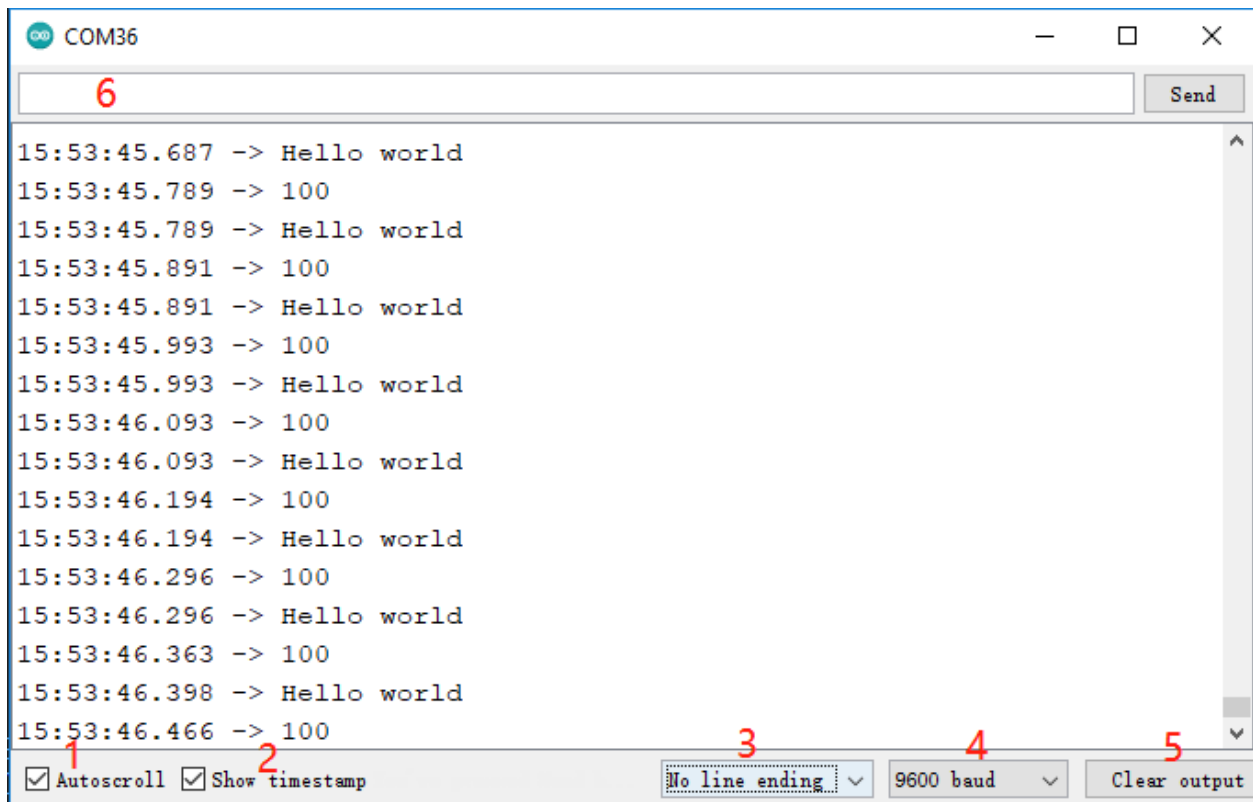
// the loop routine runs over and over again forever:
void loop() {
    int number = 100;
    Serial.println(number);
    Serial.println("Hello world");
    delay(100);           // delay in between reads for stability
}
```

- `Serial.begin()`: Sets the data rate in bits per second (baud) for serial data transmission, here set to 9600.
- `Serial.println()`: Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or 'r') and a newline character (ASCII 10, or 'n'). This command takes the same forms as `Serial.print()`.

2. Select the correct board and port to upload the code.
3. In the toolbar, click the magnifying glass icon to turn on Serial Monitor.



4. Here is the Serial Monitor.



- 1: Option to select between automatically scroll and not scroll.
- 2: Option to show timestamp prior to data displayed on Serial Monitor.

- **3:** Ending selection, select the ending characters appended to data sent to Arduino. Selection includes:
  - **No line Ending** just sends what you type;
  - **Newline** is `\n` and will send an ASCII new line code after what you type
  - **Carriage Return** is `\r`, which will send an ASCII carriage return character after what you type;
  - **Both NL & CR** is `\r\n` which will send both a carriage return and a new line character after what you type.
- **4:** Select communication speed between Arduino board and PC. This value **MUST** be the same as the value set in `Serial.begin()`.
- **5:** Clear all text on the output console.
- **6:** A textbox to send characters to the Arduino board, see [5.12 Serial Read](#) for a tutorial.

### 4.3.2 3.1 Reading Button Value

In the previous projects, we used the output function, in this chapter we will use the input function to input read the button value.

#### Required Components

In this project, we need the following components.

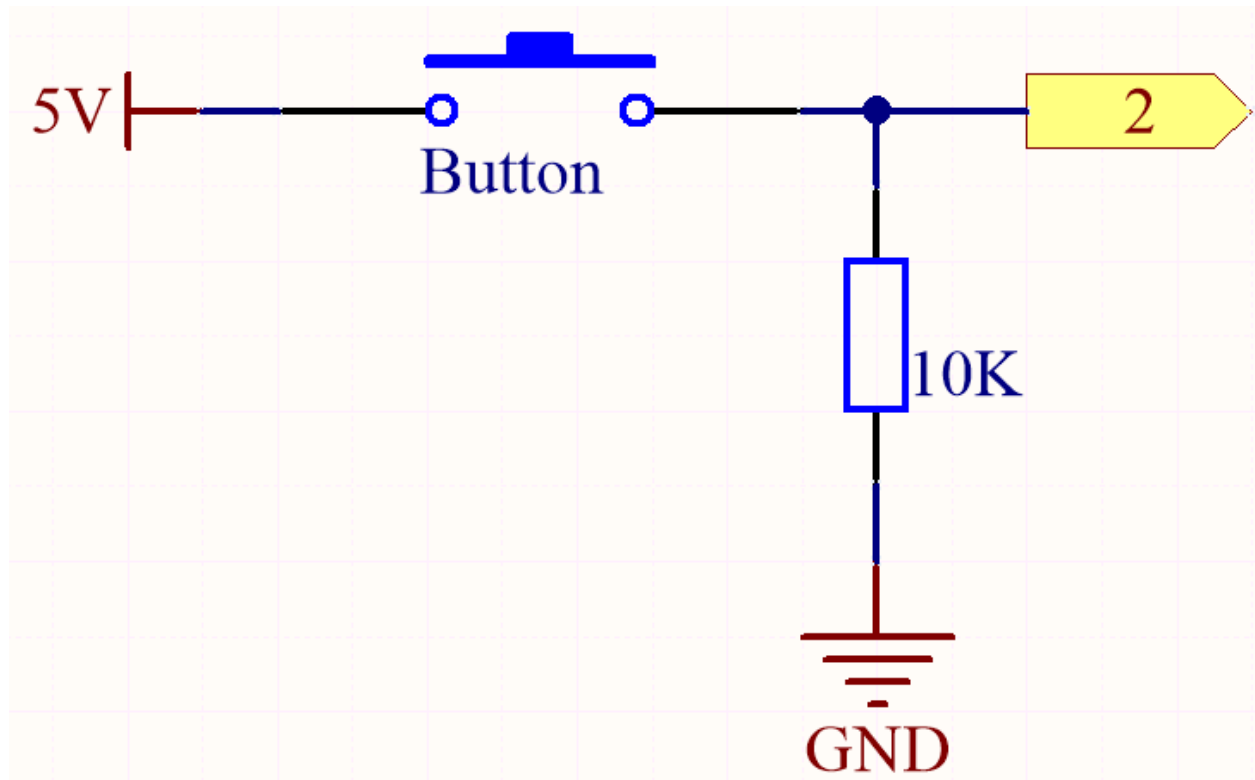
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

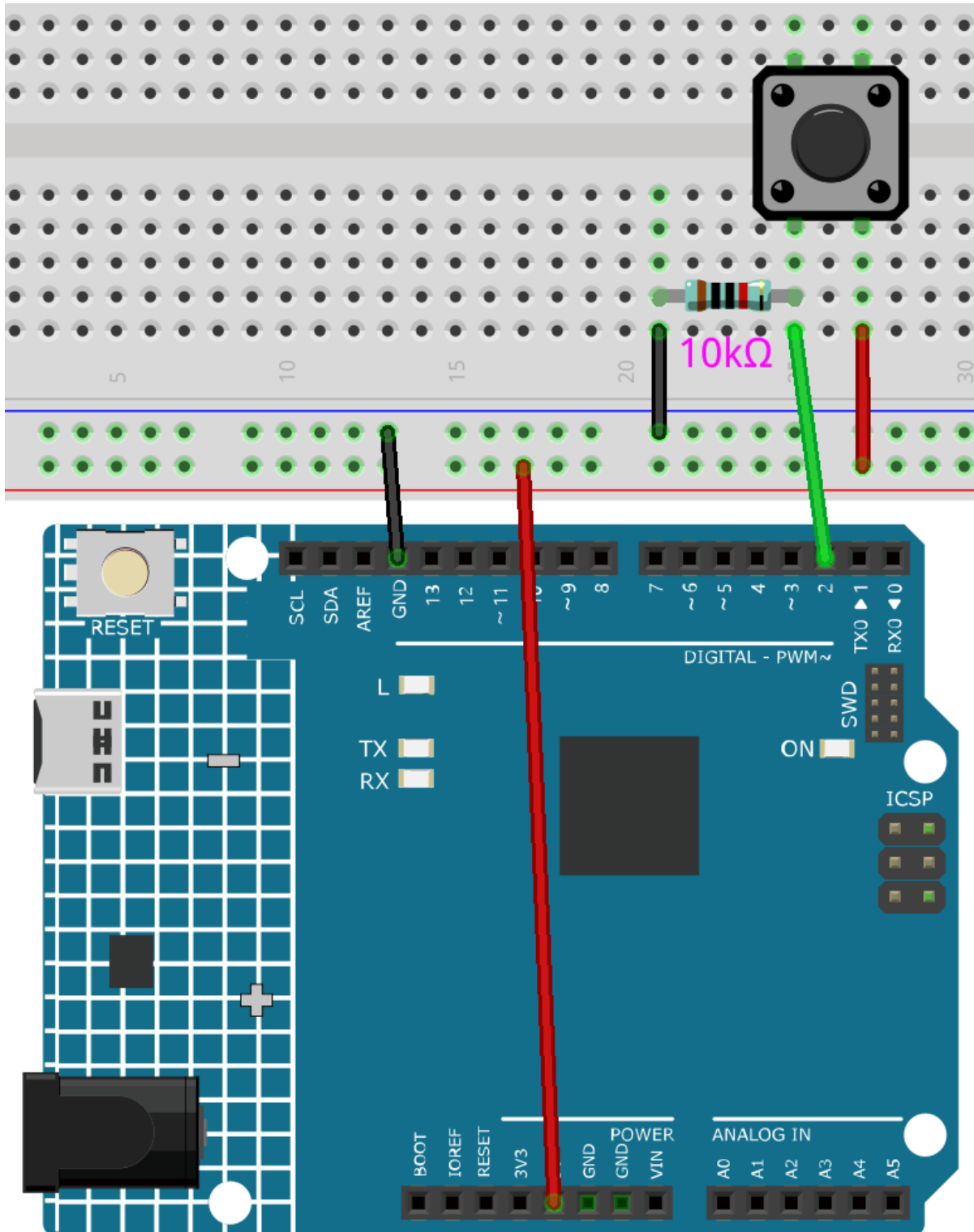
COMPONENT INTRODUCTION	PURCHASE LINK
<a href="#">Arduino Uno R4 Minima</a>	-
<a href="#">Breadboard</a>	
<a href="#">Jumper Wires</a>	
<a href="#">Resistor</a>	
<a href="#">Button</a>	

#### Schematic



One side of the button pin is connected to 5V, and the other side pin is connected to pin 2, so when the button is pressed, pin 2 will be high. However, when the button is not pressed, pin 2 is in a suspended state and may be high or low. In order to get a stable low level when the button is not pressed, pin 2 needs to be reconnected to GND through a 10K pull-down resistor.

#### Wiring



## Code

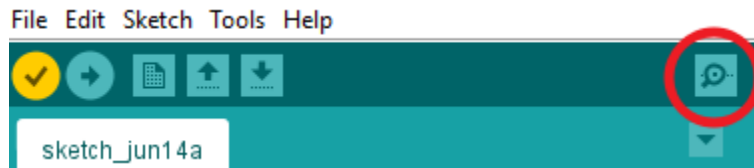
### Note:

- You can open the file `3.1.read_button_value.ino` under the path of `3in1-kit\learning_project\3.`

```
1.read_button_value.
```

- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, click the magnifying glass icon in the upper right corner of the Arduino IDE (Serial Monitor).



When you press the button, the Serial Monitor will print “1”.

### 4.3.3 3.2 Feel the Magnetism

The most common type of reed switch contains a pair of magnetizable, flexible, metal reeds whose end portions are separated by a small gap when the switch is open.

A magnetic field from an electromagnet or a permanent magnet will cause the reeds to attract each other, thus completing an electrical circuit. The spring force of the reeds causes them to separate, and open the circuit, when the magnetic field ceases.

A common example of a reed switch application is to detect the opening of a door or windows, for a security alarm.

#### Required Components

In this project, we need the following components.

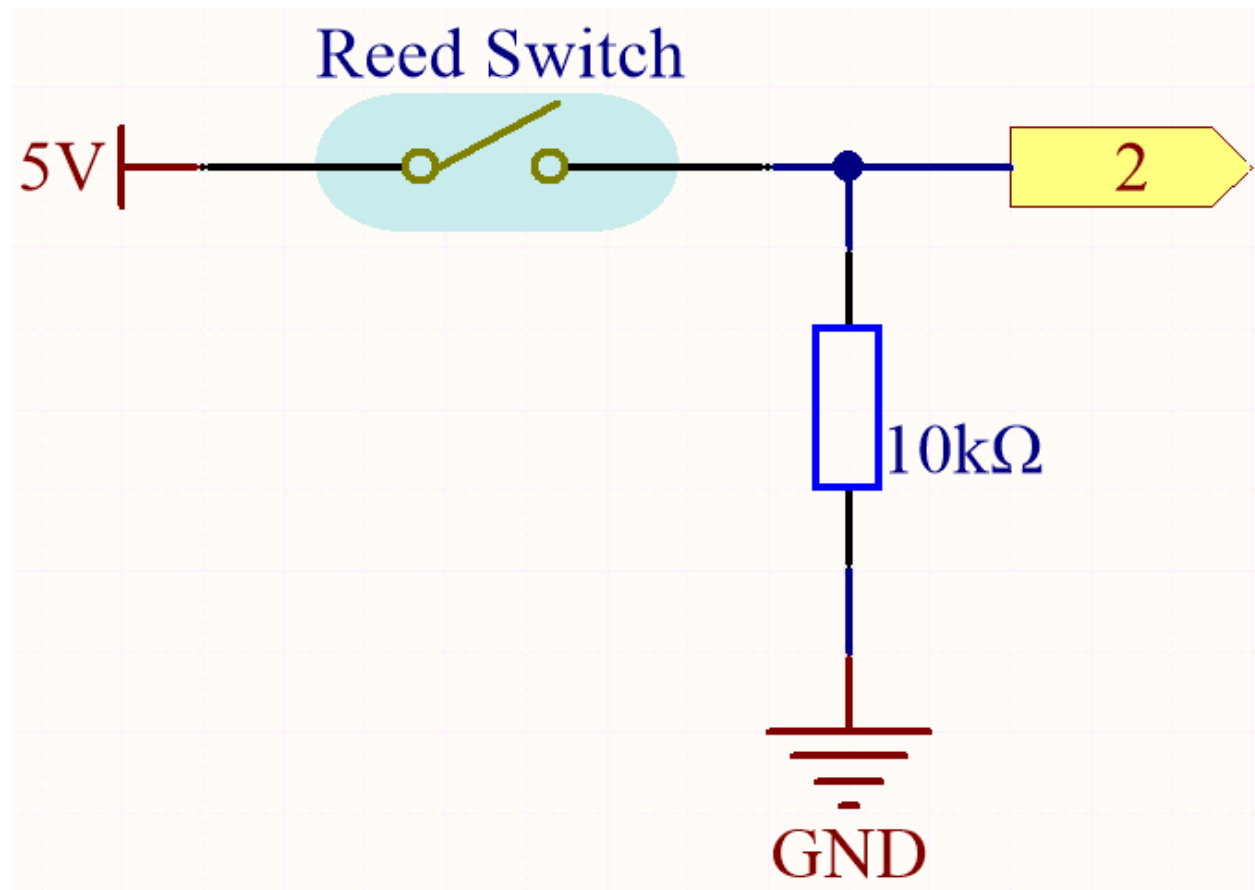
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Reed Switch</i>	-

#### Schematic

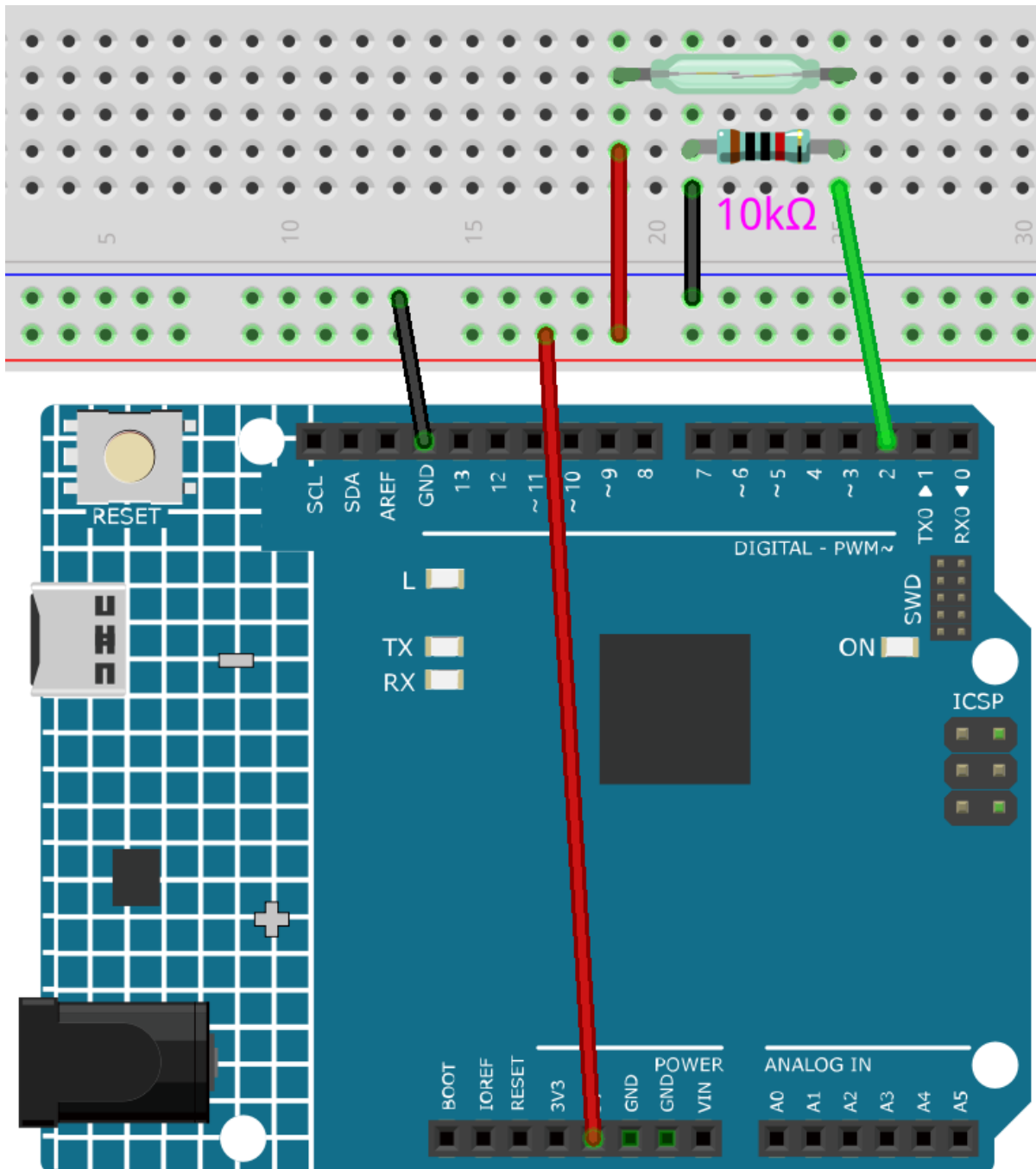


By default, pin 2 is low; and will go high when the magnet is near the reed switch.

The purpose of the 10K resistor is to keep the pin 2 at a steady low level when no magnet is near.

#### Wiring





### Code

#### Note:

- You can open the file `3.2.feel_the_magnetism.ino` under the path of `3in1-kit\learning_project\3.2.feel_the_magnetism`.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, when a magnet is near the reed switch, the serial monitor will print 1.

### 4.3.4 3.3 Detect the Obstacle

This module is commonly installed on the car and robot to judge the existence of the obstacles ahead. Also it is widely used in hand held device, water faucet and so on.

#### Required Components

In this project, we need the following components.

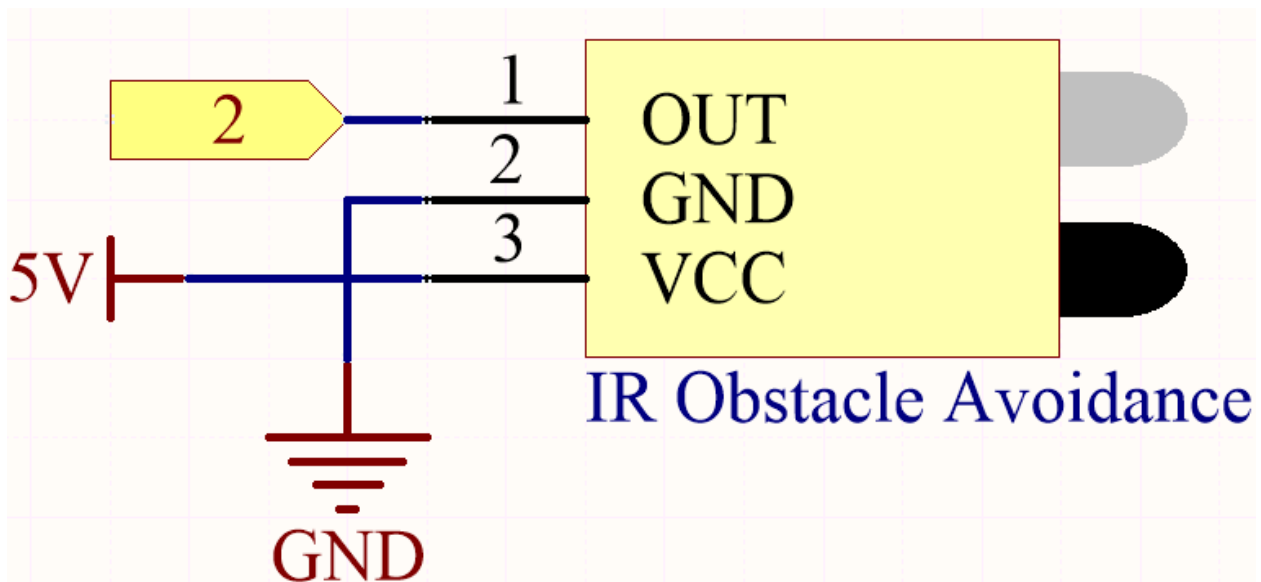
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

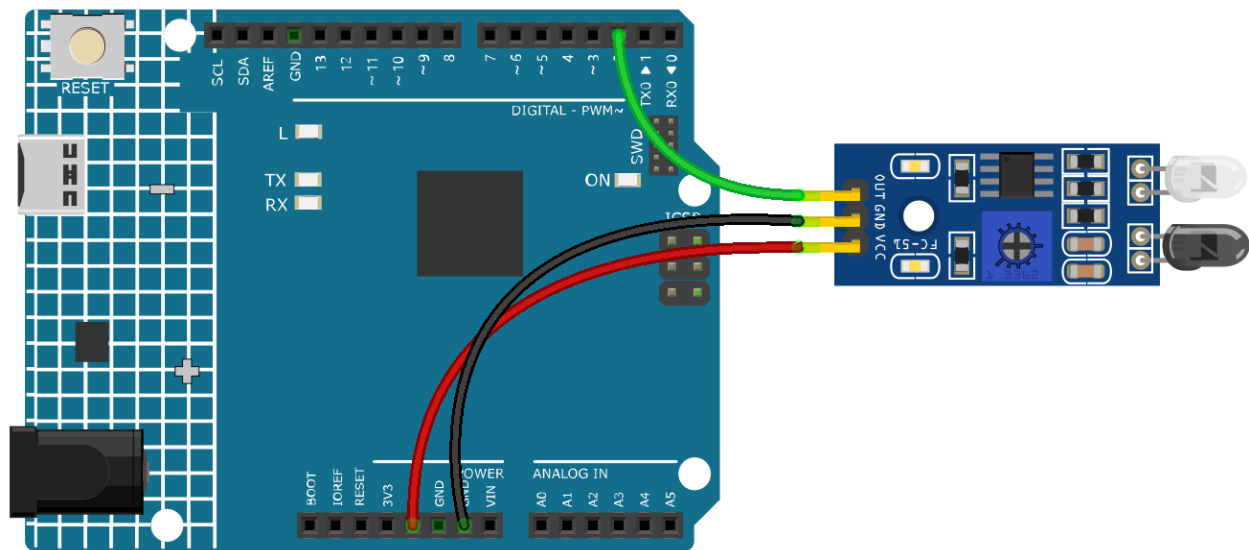
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>Obstacle Avoidance Module</i>	

#### Schematic



The digital pin 2 is used to read the signal of IR Obstacle Avoidance Module. We get the VCC of IR Sensor Module connected to 5V, GND to GND, OUT to digital pin 2.

#### Wiring



Code

Note:

- You can open the file 3.3.detect\_the\_obstacle.ino under the path of 3in1-kit\learning\_project\3.3.detect\_the\_obstacle.
- Or copy this code into **Arduino IDE**.

When the IR obstacle avoidance module detects something blocking in front of it, [0] will appear on the serial monitor, otherwise [1] will be displayed.

4.3.5 3.4 Detect the Line

The line tracking module is used to detect whether there are black areas on the ground, such as black lines pasted with electrical tape.

One of its LEDs emits appropriate infrared light to the ground, and the black surface has a relatively strong ability to absorb light and a weaker reflection ability. White surfaces are the opposite. If it detects reflected light, it means the ground is currently white. If not detected, it means black.

That’s how it works.

Required Components

In this project, we need the following components.

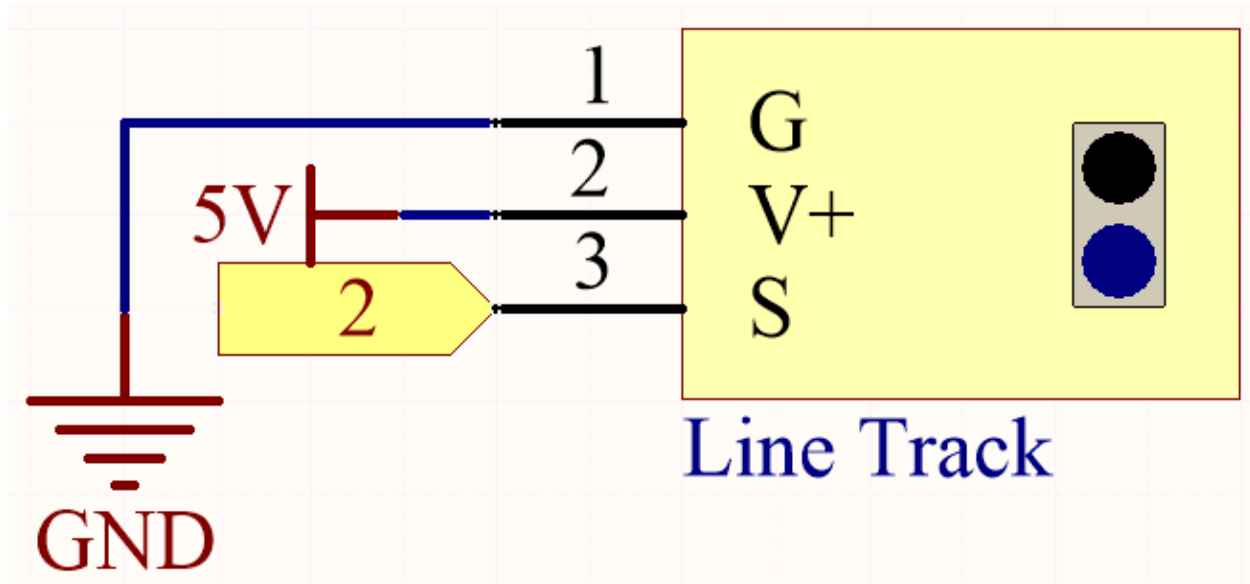
It’s definitely convenient to buy a whole kit, here’s the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

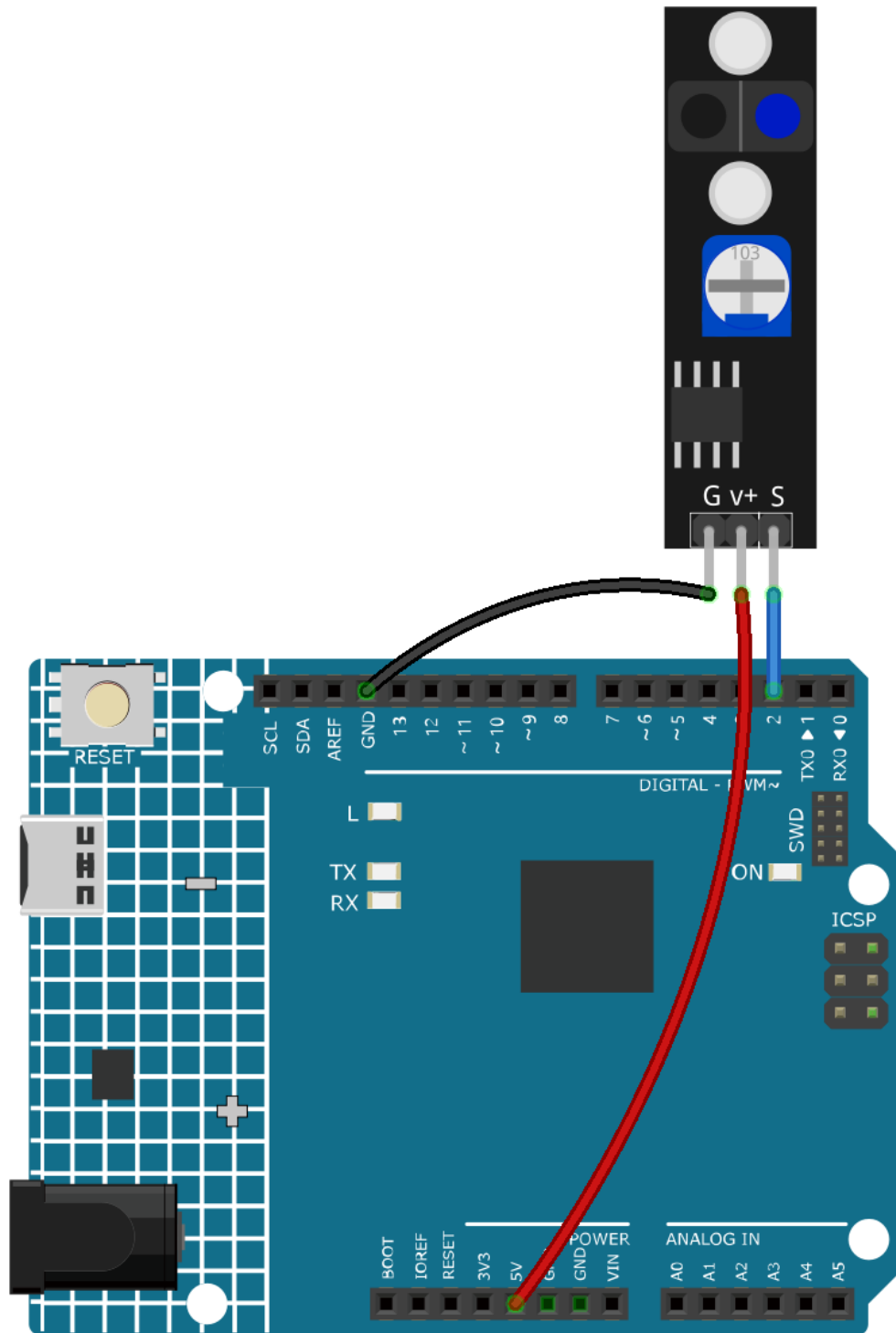
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>Line Tracking Module</i>	

### Schematic



The digital pin 2 is used to read the signal of line track module. We get the VCC of the module connected to 5V, GND to GND, OUT to digital pin 2.

### Wiring



### Code

#### Note:

- You can open the file `3.4.detect_the_line.ino` under the path of `3in1-kit\learning_project\3.4.detect_the_line`.
- Or copy this code into **Arduino IDE**.

When the line tracking module detects there is black line, there appears [1] on the Serial Monitor; otherwise, [0] is displayed.

## 4.4 4. Analog Read

The Arduino can read the connected analog sensors through the analog pins.

The R4 board contains a multi-channel, 10-bit analog-to-digital converter. This means it maps the input voltage between 0 and the operating voltage (5V or 3.3V) to an integer value between 0 and 1023.

You need the `analogRead(pin)` function to read the value of the analog pin.

- `analogRead(pin)`: Read the value from the specified analog pin.

### Syntax

`analogRead(pin)`

### Parameters

- `pin`: the name of the analog input pin to read from (A0 to A5).

### Returns

0-1023. Data type: `int`.

### Example of Analog Read

```
int analogPin = A0; // device connected to analog pin A0
                    // outside leads to ground and +5V
int val = 0; // variable to store the value read

void setup() {
  Serial.begin(9600); // setup serial
}

void loop() {
  val = analogRead(analogPin); // read the input pin
  Serial.println(val); // debug value
}
```

### Notes and Warnings

- The analog pins are A0-A5.
- You don't need to call `pinMode()` before calling the analog pin, but if the pin was previously set to OUTPUT, the function `analogRead()` will not work properly, in which case you need to call `pinMode()` to set it back to INPUT.

### Related Components

Below are the related components, you can click in to learn how to use them.

### 4.4.1 4.1 Turn the Knob

Potentiometer is a resistor component with 3 terminals and its resistance value can be adjusted according to some regular variation.

#### Required Components

In this project, we need the following components.

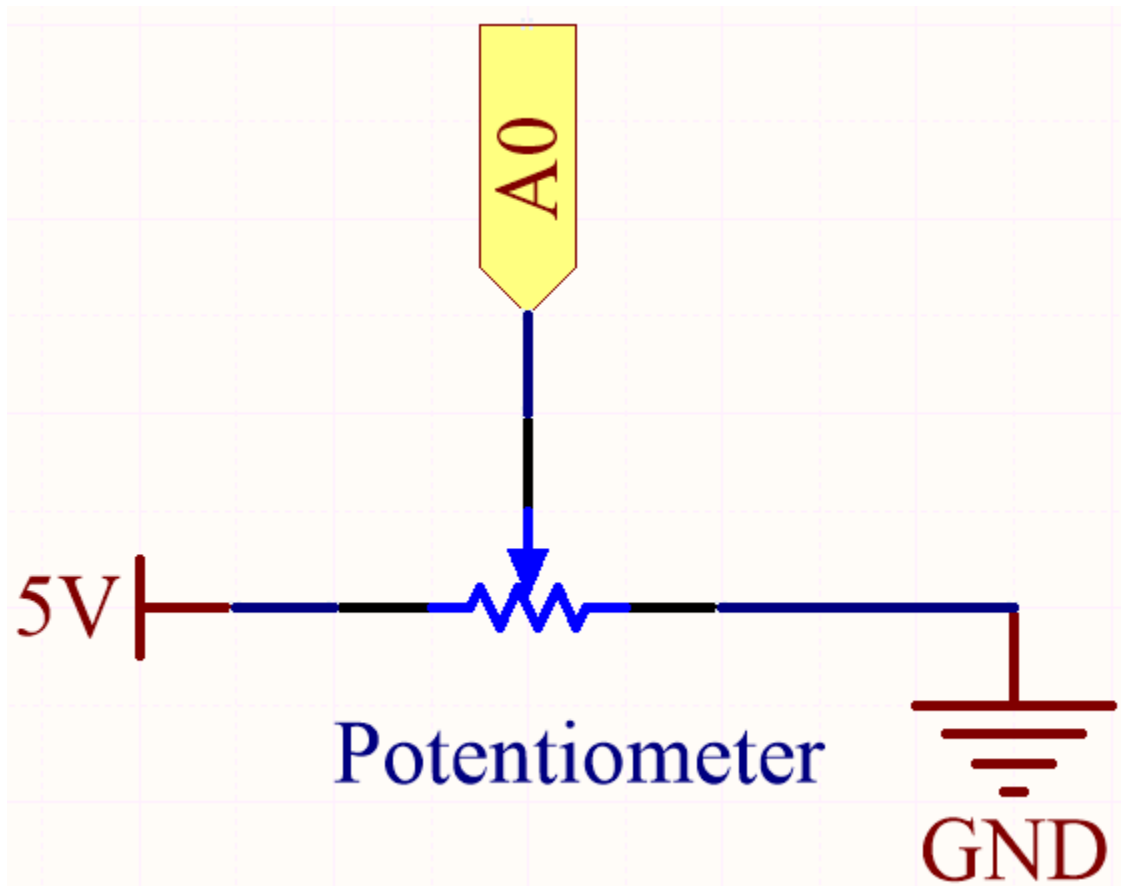
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Potentiometer</i>	

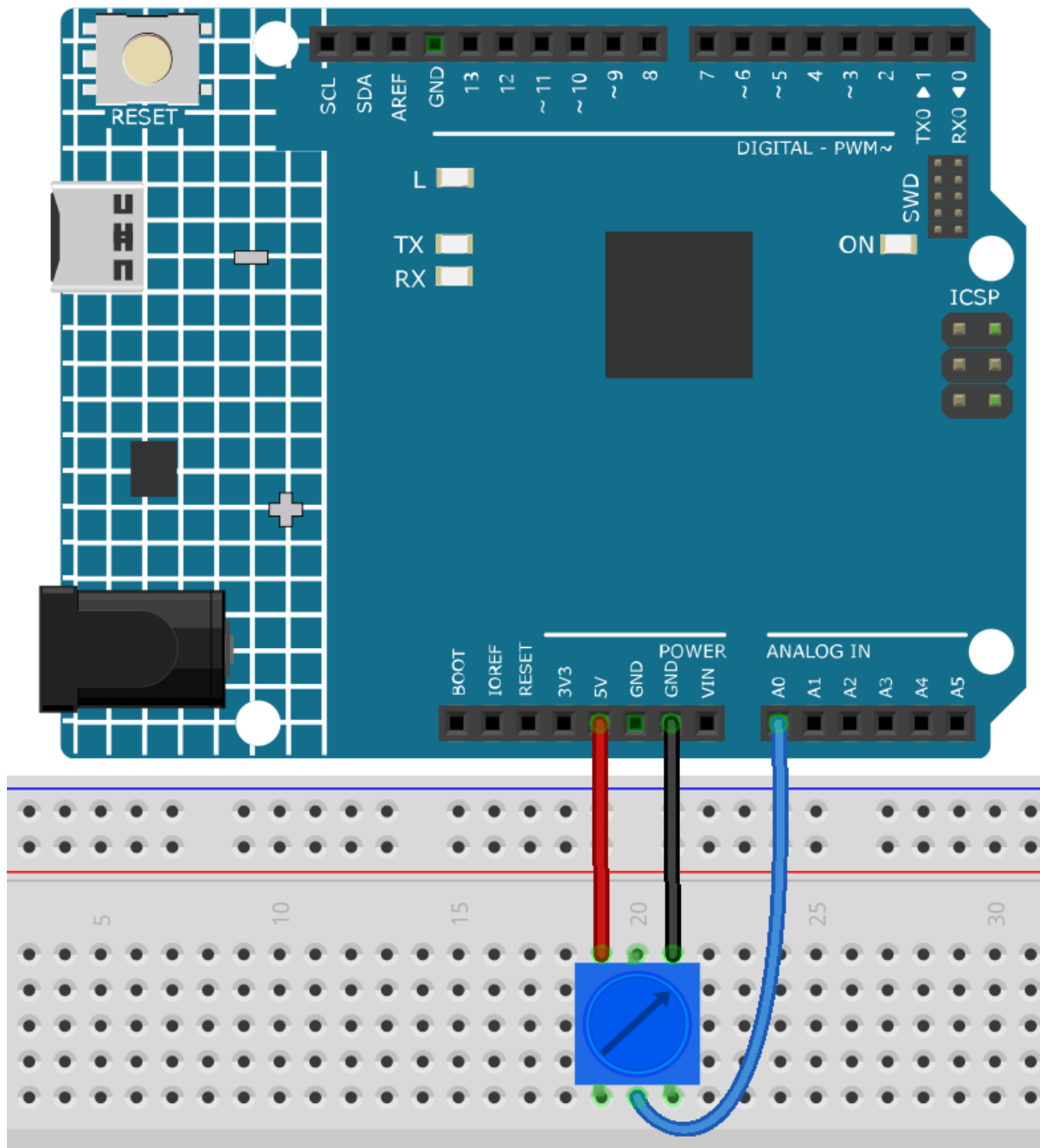
#### Schematic



In this example, we use the analog pin (A0) to read the value of the potentiometer. By rotating the axis of the potentiometer, you can change the distribution of resistance among these three pins, changing the voltage on the middle pin. When the resistance between the middle and a outside pin connected to 5V is close to zero (and the resistance between the middle and the other outside pin is close to 10k), the voltage at the middle pin is close to 5V. The reverse operation (the resistance between the middle and a outside pin connected to 5V is close to 10k) will make the voltage at the middle pin be close to 0V.

#### Wiring





### Code

#### Note:

- You can open the file 4.1.turn\_the\_knob.ino under the path of 3in1-kit\learning\_project\4.1.turn\_the\_knob.
- Or copy this code into **Arduino IDE**.

After uploading the codes to the board, you can open the serial monitor to see the reading value of the pin. When rotating the axis of the potentiometer, the serial monitor will print the value 0~1023.

### 4.4.2 4.2 Feel the Light

The photoresistor is a typical device for analog inputs and it is used in a very similar way to a potentiometer. Its resistance value depends on the intensity of the light, the stronger the irradiated light, the smaller its resistance value; conversely, it increases.

#### Required Components

In this project, we need the following components.

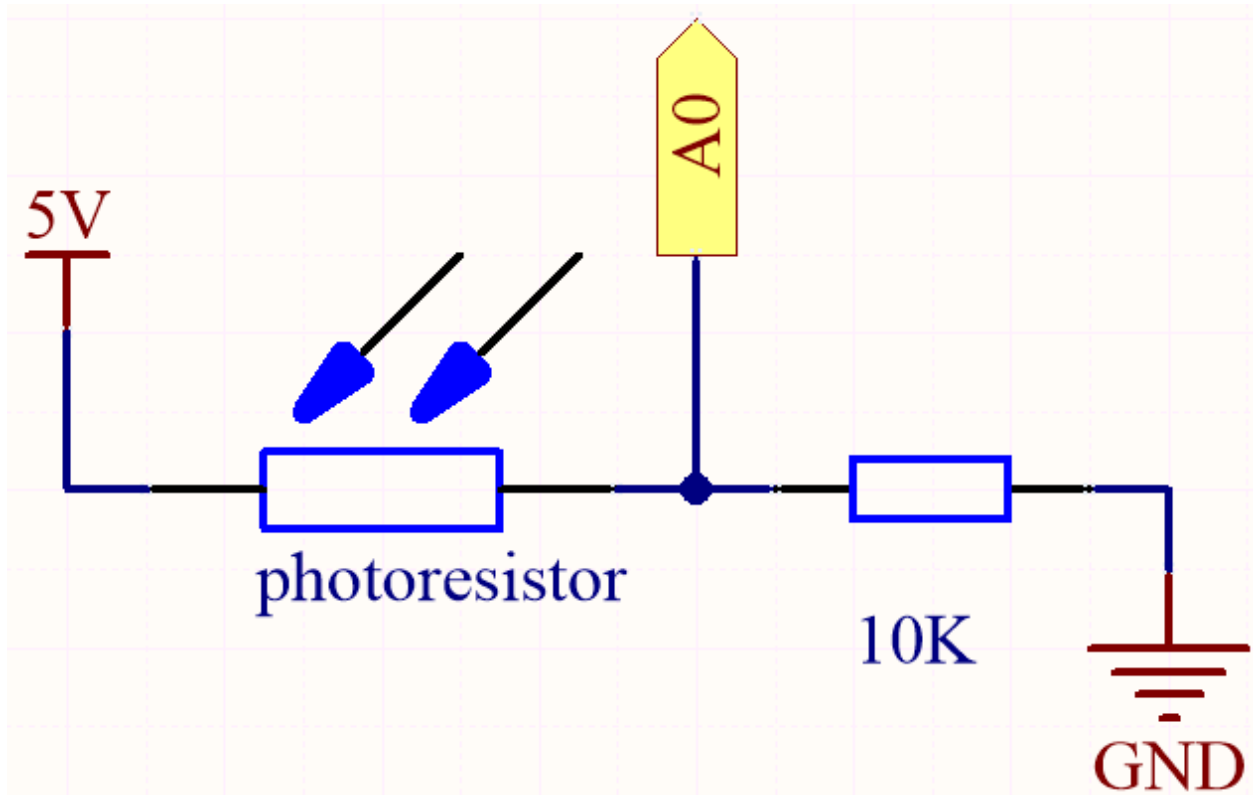
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Photoresistor</i>	

#### Schematic

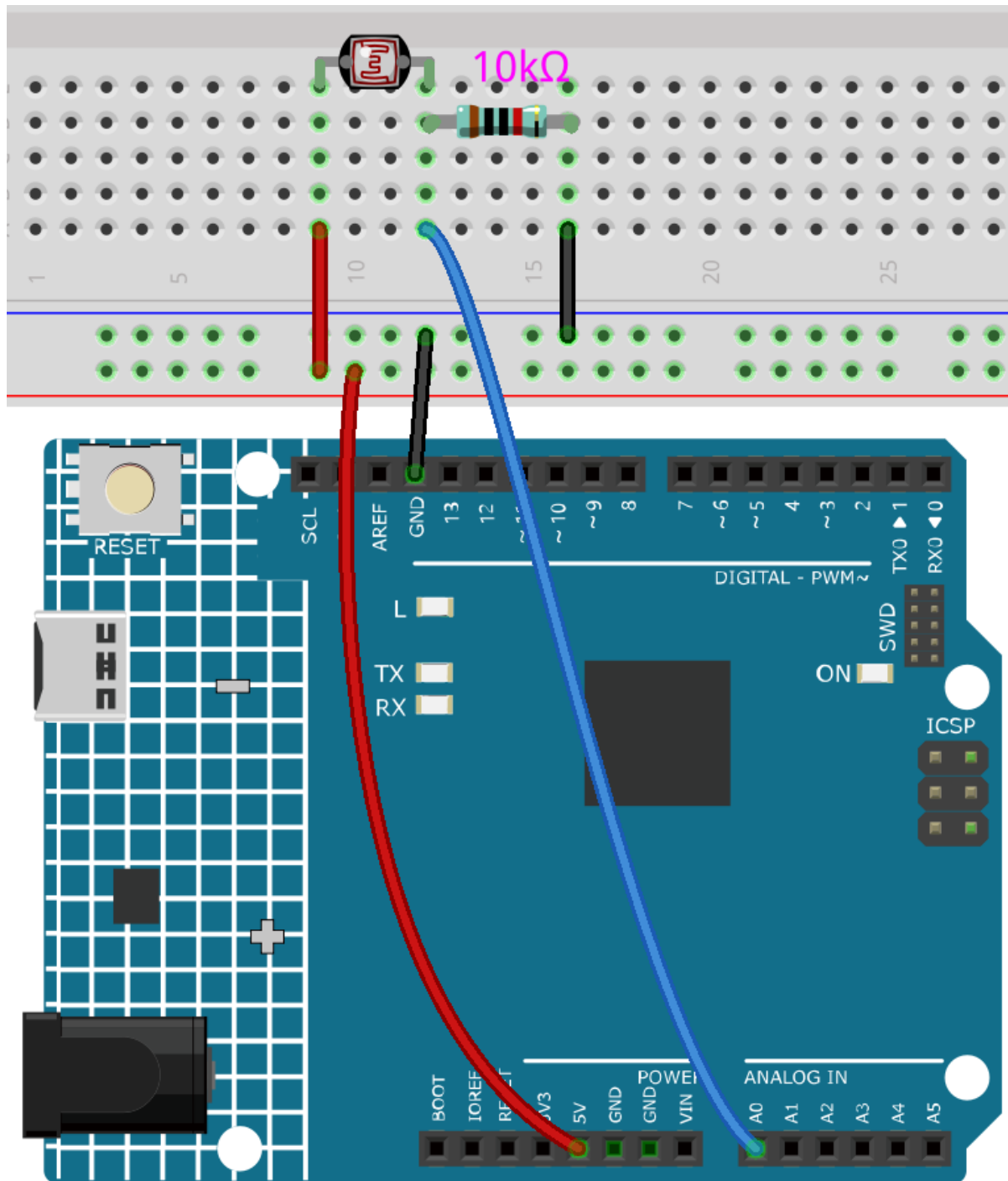


In this circuit, the 10K resistor and the photoresistor are connected in series, and the current passing through them is the same. The 10K resistor acts as a protection, and the pin A0 reads the value after the voltage conversion of the photoresistor.

When the light is enhanced, the resistance of the photoresistor decreases, then its voltage decreases, so the value from pin A0 will increase; if the light is strong enough, the resistance of the photoresistor will be close to 0, and the value of pin A0 will be close to 1023. At this time, the 10K resistor plays a protective role, so that 5V and GND are not connected together, resulting in a short circuit.

If you place the photoresistor in a dark situation, the value of pin A0 will decrease. In a dark enough situation, the resistance of the photoresistor will be infinite, and its voltage will be close to 5V (the 10K resistor is negligible), and the value of pin A0 will be close to 0.

#### Wiring



## Code

**Note:**

- Open the `4.2.feel_the_light.ino` file under the path of `3in1-kit\learning_project\4.2.feel_the_light`.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, the serial monitor prints out the photoresistor values. The stronger the current ambient brightness, the larger the value displayed on the serial monitor.

### 4.4.3 4.3 Toggle the Joystick

The joystick should be very familiar to anyone who plays video games regularly. It is usually used to move characters or rotate the screen.

Our movements can be read by the Joystick, which works on a very simple principle. It consists of two potentiometers that are perpendicular to each other. These two potentiometers measure the analog value of the joystick in both vertical and horizontal directions, producing a value (x,y) in a planar right-angle coordinate system.

This kit also includes a joystick with a digital input. It is activated when the joystick is pressed.

#### Required Components

In this project, we need the following components.

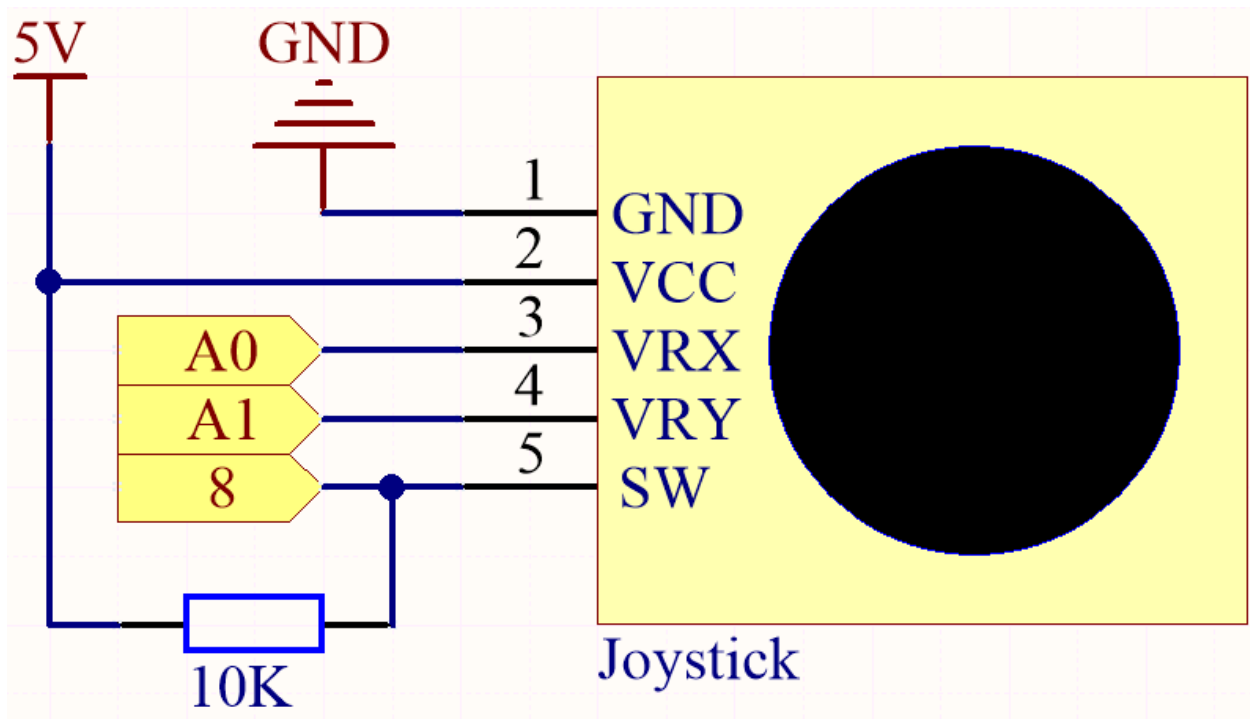
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Joystick Module</i>	-

#### Schematic



**Note:** The SW pin is connected to a 10K pull-up resistor, the reason is to be able to get a stable high level on the SW pin (Z axis) when the joystick is not pressed; otherwise the SW is in a suspended state and the output value may vary between 0/1.

## Wiring



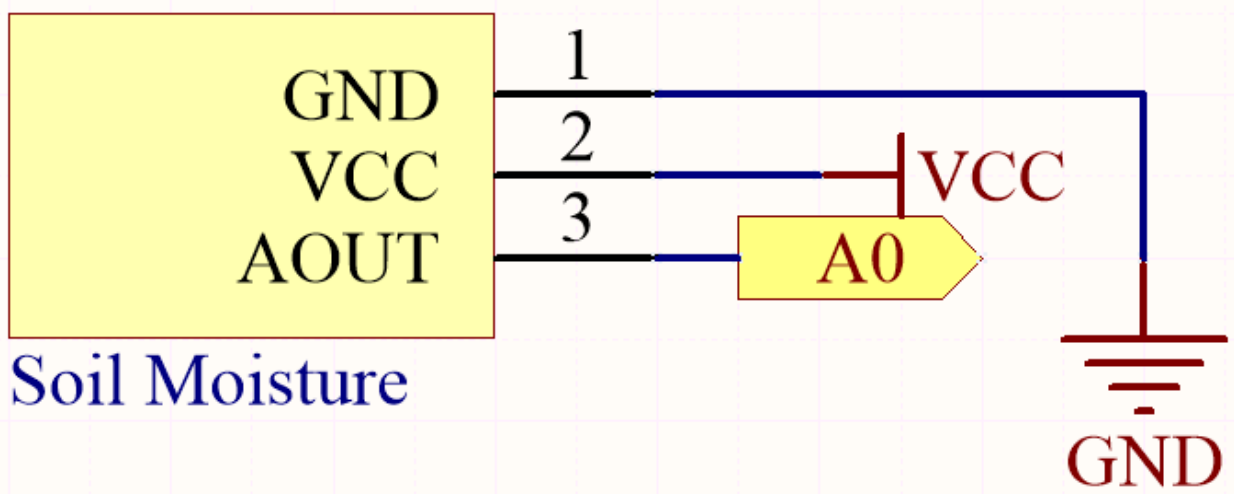
## SunFounder 3in1 Kit

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

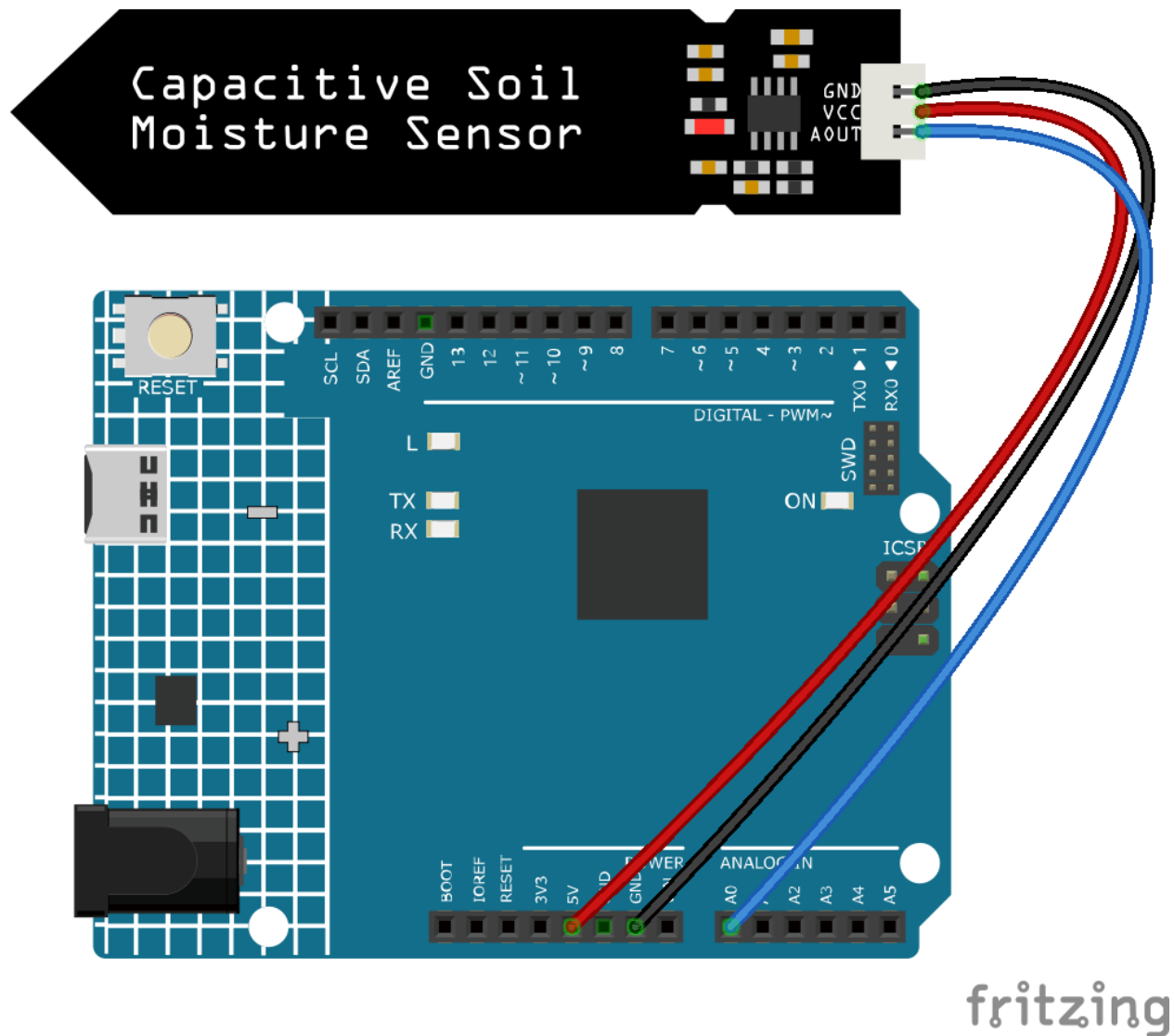
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>Soil Moisture Module</i>	

### Schematic



### Wiring





### Code

#### Note:

- Open the 4.4.measure\_soil\_moisture.ino file under the path of 3in1-kit\learning\_project\4.4.measure\_soil\_moisture.
- Or copy this code into **Arduino IDE**.

Once the code is successfully uploaded, the serial monitor will print out the soil moisture value.

By inserting the module into the soil and watering it, the value of the soil moisture sensor will become smaller.

### 4.4.5 4.5 Thermometer

A thermometer is a device that measures temperature or a temperature gradient (the degree of hotness or coldness of an object). A thermometer has two important elements: (1) a temperature sensor (e.g. the bulb of a mercury-in-glass thermometer or the pyrometric sensor in an infrared thermometer) in which some change occurs with a change in temperature; and (2) some means of converting this change into a numerical value (e.g. the visible scale that is marked on a mercury-in-glass thermometer or the digital readout on an infrared model). Thermometers are widely used in technology and industry to monitor processes, in meteorology, in medicine, and in scientific research.

A thermistor is a type of temperature sensor whose resistance is strongly dependent on temperature, and it has two types: Negative Temperature Coefficient (NTC) and Positive Temperature Coefficient (PTC), also known as NTC and PTC. The resistance of PTC thermistor increases with temperature, while the condition of NTC is opposite to the former.

In this experiment we use an **NTC thermistor** to make a thermometer.

#### Required Components

In this project, we need the following components.

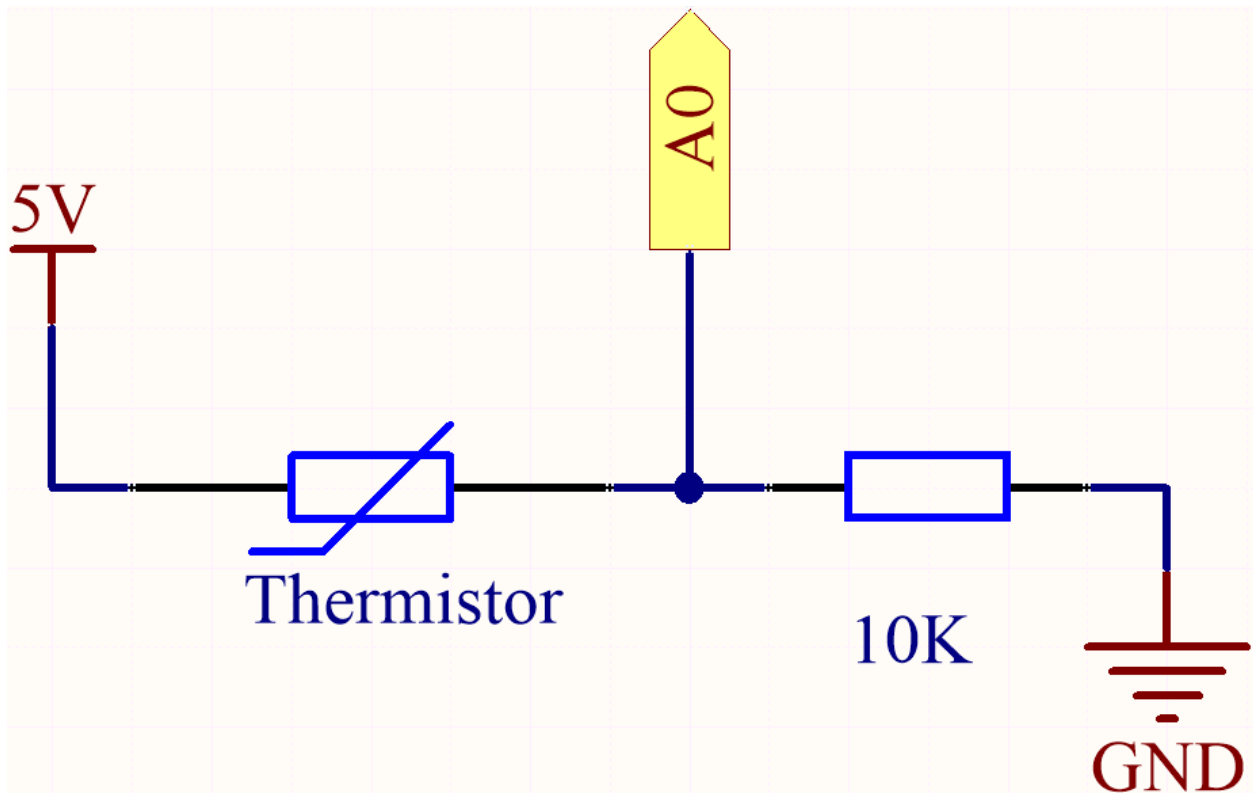
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Thermistor</i>	

#### Schematic



Each thermistor has a normal resistance. Here it is 10k ohm, which is measured under 25 degree Celsius.

When the temperature gets higher, the resistance of the thermistor decreases. Then the voltage data is converted to digital quantities by the A/D adapter.

The temperature in Celsius or Fahrenheit is output via programming.

Here is the relation between the resistance and temperature:

$$RT = RN \exp B(1/TK - 1/TN)$$

- **RT** is the resistance of the NTC thermistor when the temperature is **TK**.
- **RN** is the resistance of the NTC thermistor under the rated temperature **TN**. Here, the numerical value of **RN** is 10k.
- **TK** is a Kelvin temperature and the unit is K. Here, the numerical value of **TK** is  $273.15 + \text{degree Celsius}$ .
- **TN** is a rated Kelvin temperature; the unit is K too. Here, the numerical value of **TN** is  $273.15 + 25$ .
- And **B(beta)**, the material constant of NTC thermistor, is also called heat sensitivity index with a numerical value 3950.
- **exp** is the abbreviation of exponential, and the base number **e** is a natural number and equals 2.7 approximately.

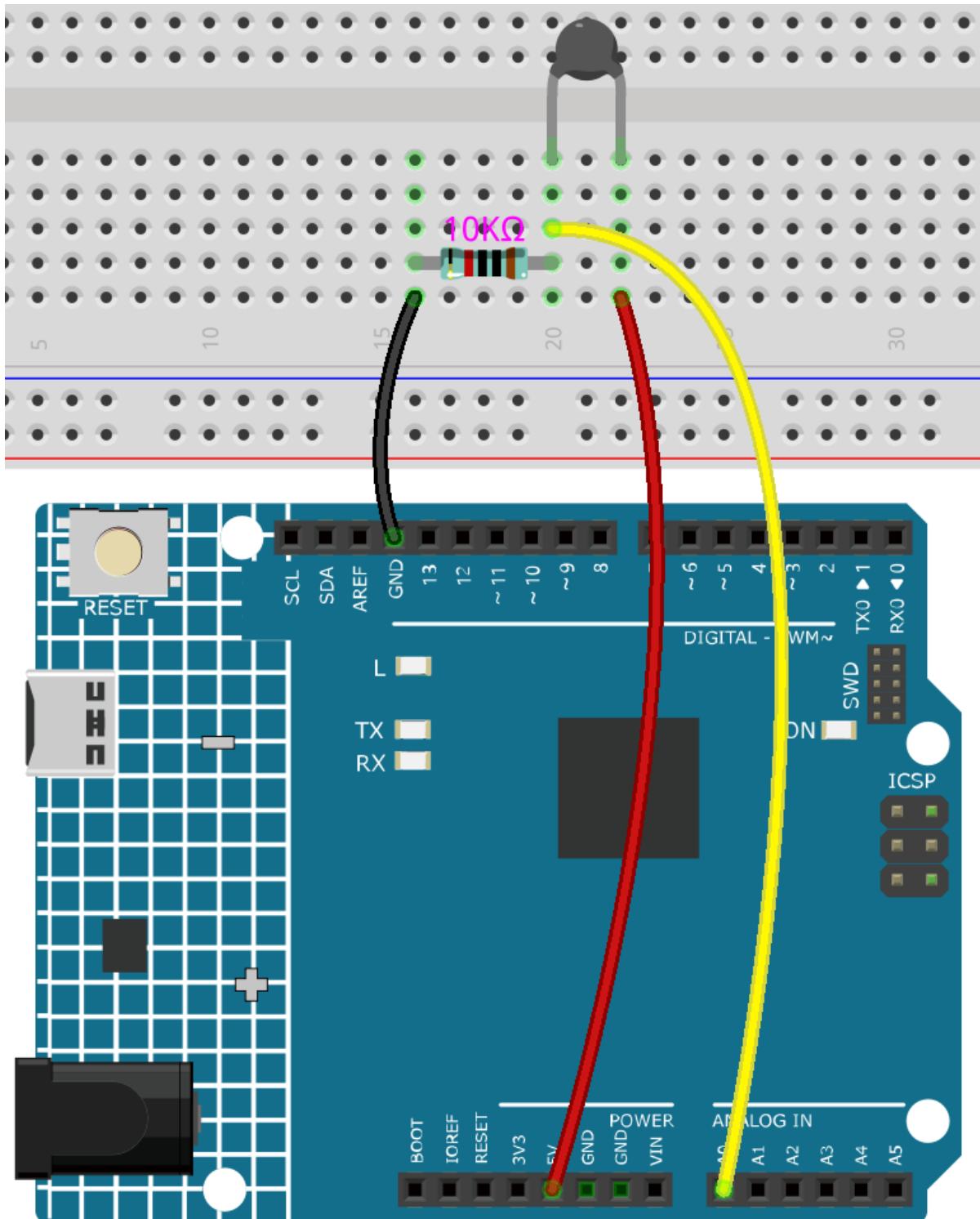
Convert this formula  $TK = 1 / (\ln(RT/RN) / B + 1/TN)$  to get Kelvin temperature that minus 273.15 equals degree Celsius.

This relation is an empirical formula. It is accurate only when the temperature and resistance are within the effective range.

## Wiring

**Note:**

- The thermistor is black or green and marked 103.



## Code

**Note:**

- You can open the file `4.5_thermometer.ino` under the path of `euler-kit/arduino/4.5_thermometer`.
- Or copy this code into **Arduino IDE**.

Don't forget to select the Raspberry Pi Pico board and the correct port before clicking the Upload button.

After the code is successfully uploaded, the Serial Monitor will print out the Celsius and Fahrenheit temperatures.

## 4.5 5. More Syntax

In this chapter, you will find some examples that illustrate the basic logic of how most programs interact with reality. This will help you become familiar with Arduino programming. When you have a creative idea in mind, programming will no longer be challenging for you.

### 4.5.1 5.1 If else

Usually we use conditional judgment to complete the most basic reality interaction projects. Here, we build a door detection system with reed switch and LED to show this logic.

Fix the magnet on one side of the door and the reed switch (with circuit) on the other side of the door. When the door is closed, the magnet is close to the reed switch, which will turn it on.

#### Required Components

In this project, we need the following components.

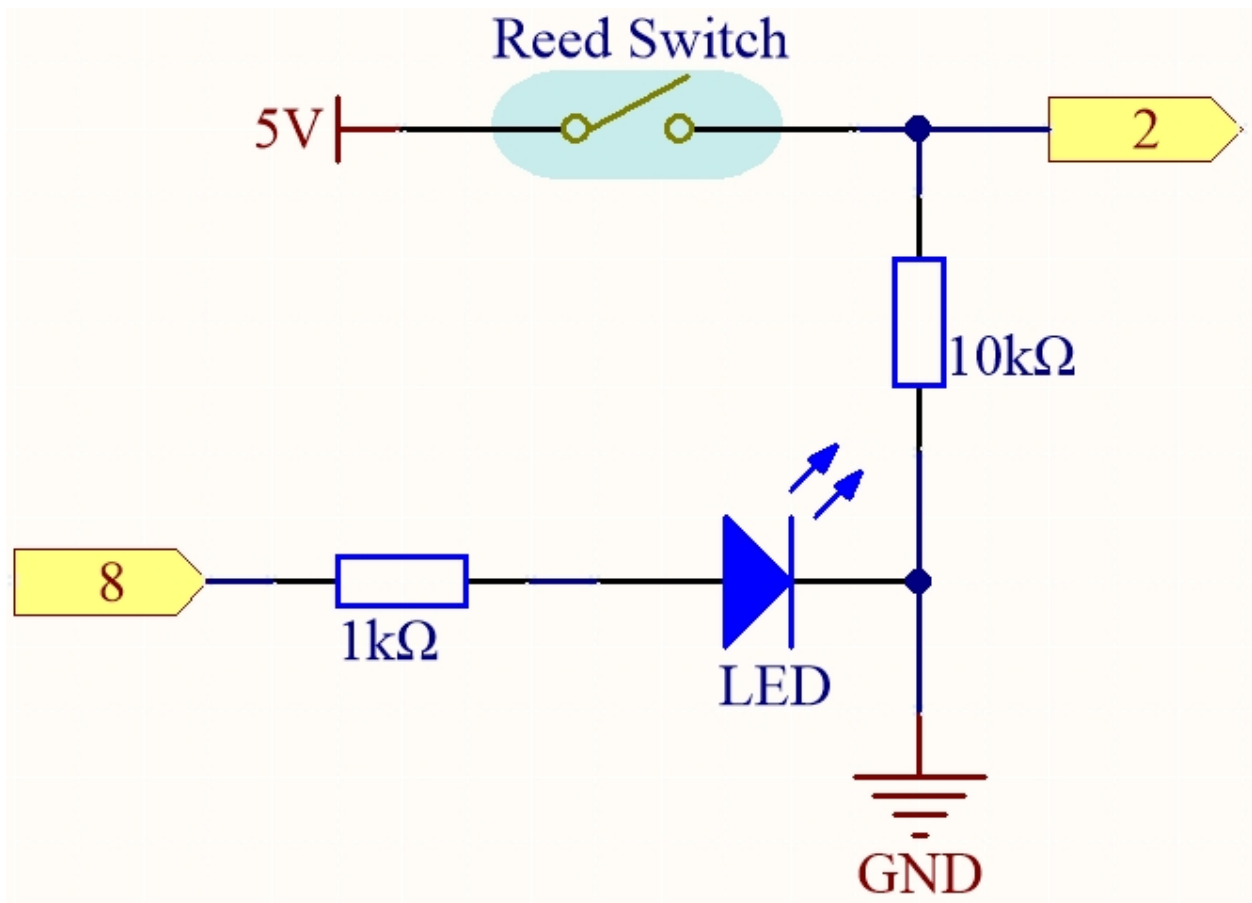
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Reed Switch</i>	-

#### Schematic



Wiring



---

### Note:

- Open the `5.1.if_else.ino` file under the path of `3in1-kit\learning_project\5.1.if_else`.
  - Or copy this code into **Arduino IDE**.
- 

After the code is uploaded successfully, if you do not close the door, the LED will light up, prompting you to close the door.

By the way, if we need the opposite effect (lighting up the LED when the door is closed), we just need to modify the condition in the `if`.

- `if else`

The `if else` allows greater control over the flow of code than the basic `if` statement, by allowing multiple tests to be grouped.

## 4.5.2 5.2 Threshold

In many projects, you will encounter such a need. “When xxx reaches a certain level, then...”

For example, in a smart home, when the light intensity is lower than 50Lux, turn on the light; Another example is in the computer motherboard, if the operating temperature of the CPU is higher than 65 degrees Celsius, turn on the fan, and so on.

In these requirements, the keyword “threshold” is reflected.

We can adjust the value of the threshold to make the circuit operate more in line with individual needs. For example, if I like a brighter living environment, I can increase the threshold of the automatic lights of the smart home to 80Lux. Another example is that the ventilation environment of my studio is not very good, and the heat dissipation demand is higher, then the threshold value of automatic fan opening can be adjusted to 50 degrees Celsius.

Here we use soil moisture sensor and 2 LEDs to make a pot monitor. If the soil is too dry, the red LED will light up; if the soil is moist enough, the green LED will light up. You need to manually adjust the thresholds for determining the dryness and wetness of the soil.

### Required Components

In this project, we need the following components.

It’s definitely convenient to buy a whole kit, here’s the link:

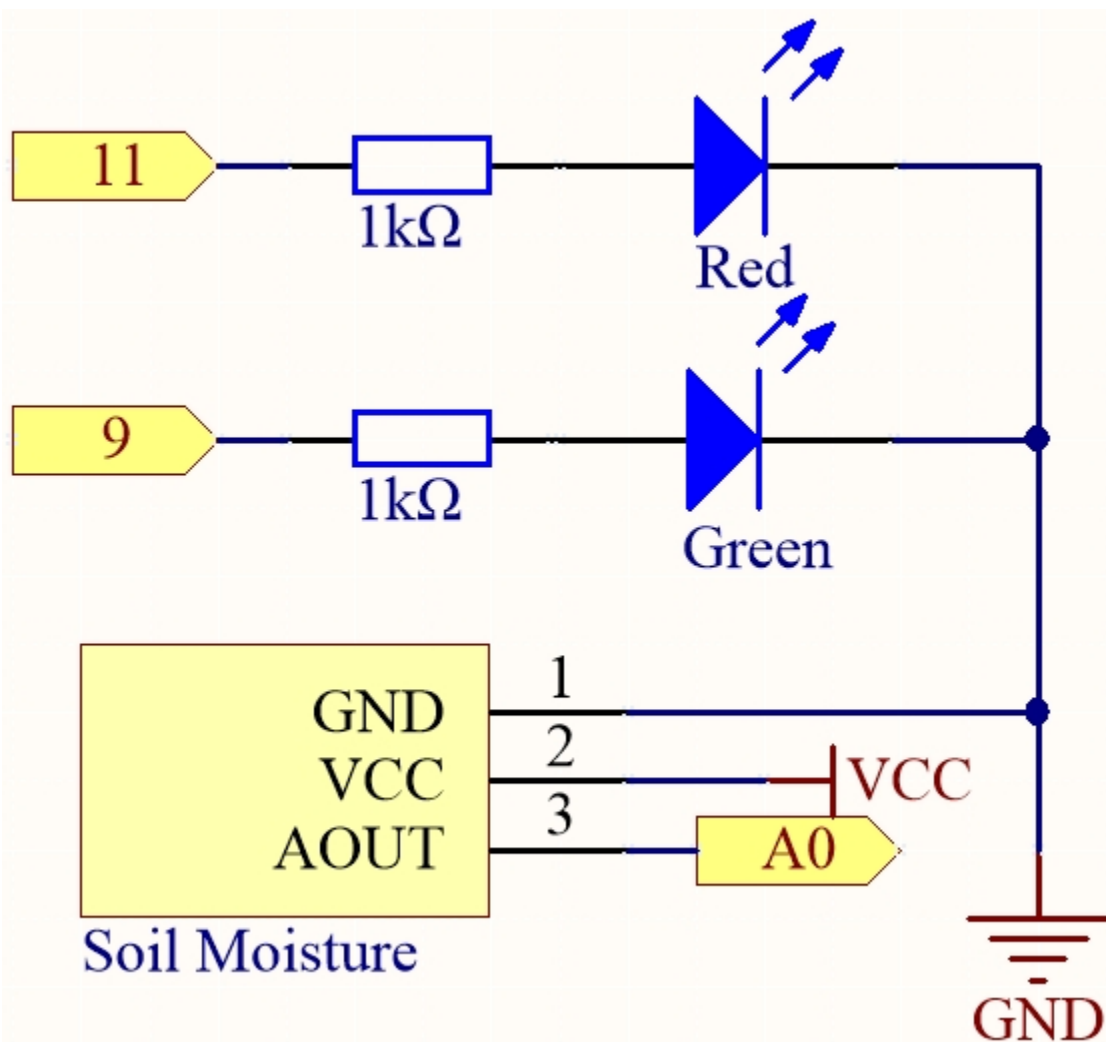
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

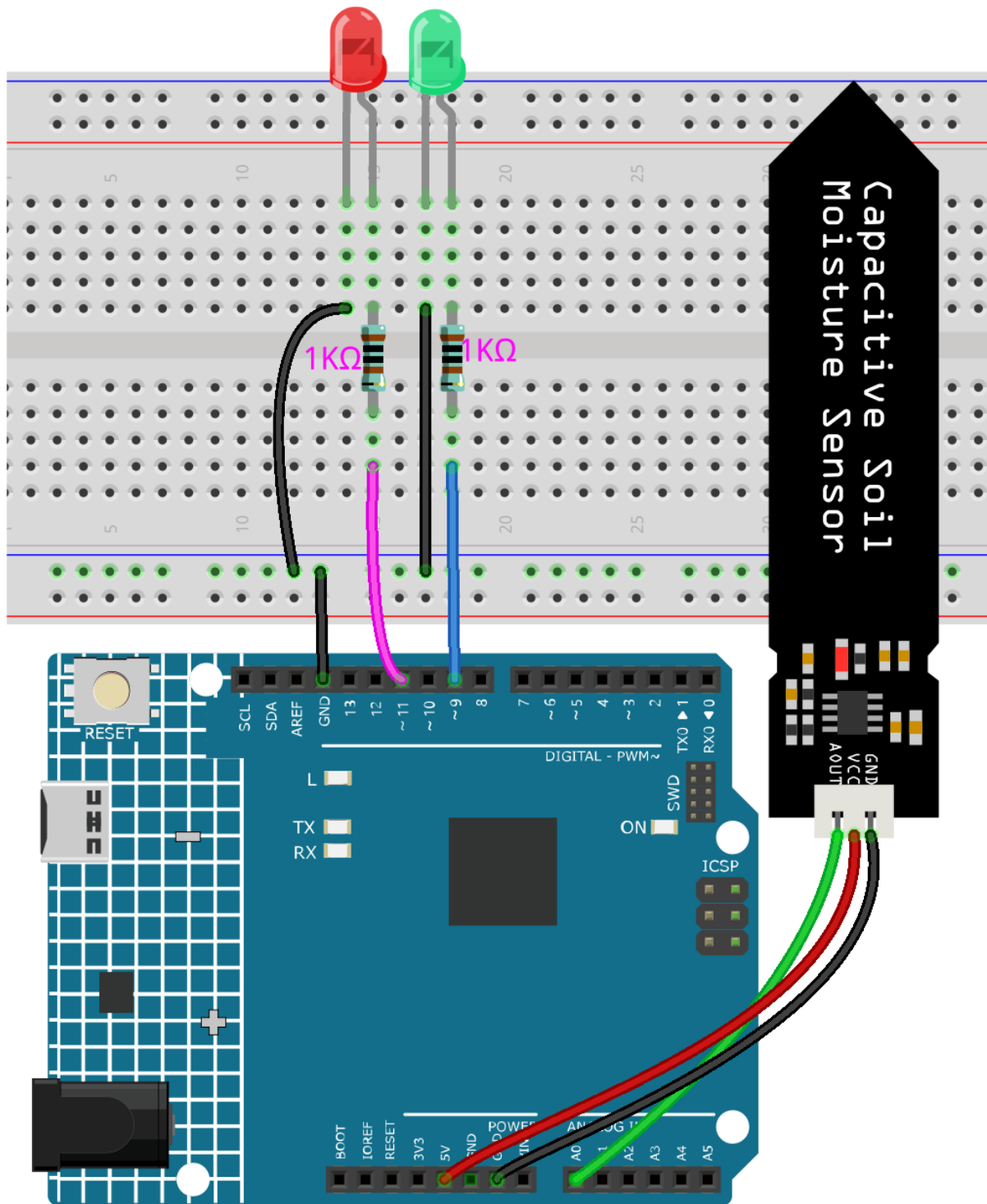


COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Soil Moisture Module</i>	

### Schematic



### Wiring



## Code

### Note:

- Open the `5.2.threshold.ino` file under the path of `3in1-kit\learning_project\5.2.threshold`.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, if your threshold is set correctly, you will see the red LED light up when the soil is dry to remind you that you need to water; after watering, the green LED will light up.

#### How it works

```
...

void loop() {
  int sensorValue = analogRead(soilMoisture);
  Serial.println(sensorValue);
  if (sensorValue > threshold) {
    digitalWrite(redPin, HIGH); // Turn the red LED
    digitalWrite(greenPin, LOW); // green
  } else {
    digitalWrite(greenPin, HIGH); // Turn on the green LED
    digitalWrite(redPin, LOW); // red
  }
}

...
```

First set a **threshold** value and then read the value of the soil moisture module, its value decreases as the moisture level increases. If the value currently read is greater than the set **threshold**, then let the red LED light up, otherwise it will turn on the green LED.

This **threshold** value needs to be adjusted according to the actual situation, you can upload the code first, then open the serial monitor to check the value, record the value in both wet and dry conditions, and then choose a middle value as the **threshold** value.

### 4.5.3 5.3 State Change Detection

When the button controls other devices, it can not only work when it is pressed, but stop when it is released. It is also possible to switch the working state each time the button is pressed.

In order to achieve this effect, you need to know how to toggle the working state between off and on when the button is pressed, That is “state change detection”.

In this project, we will use the button to control the motor.

#### Required Components

In this project, we need the following components.

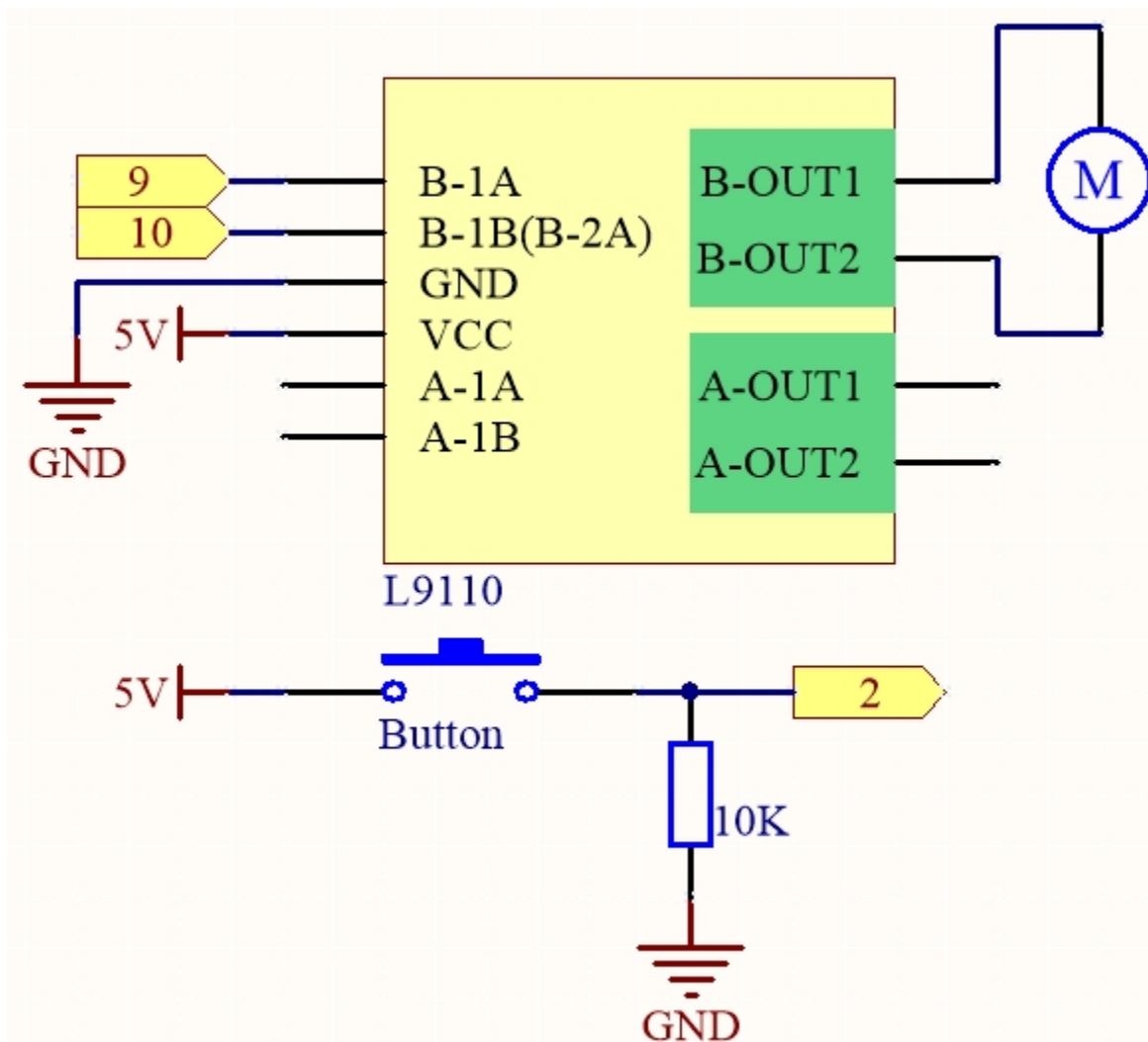
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

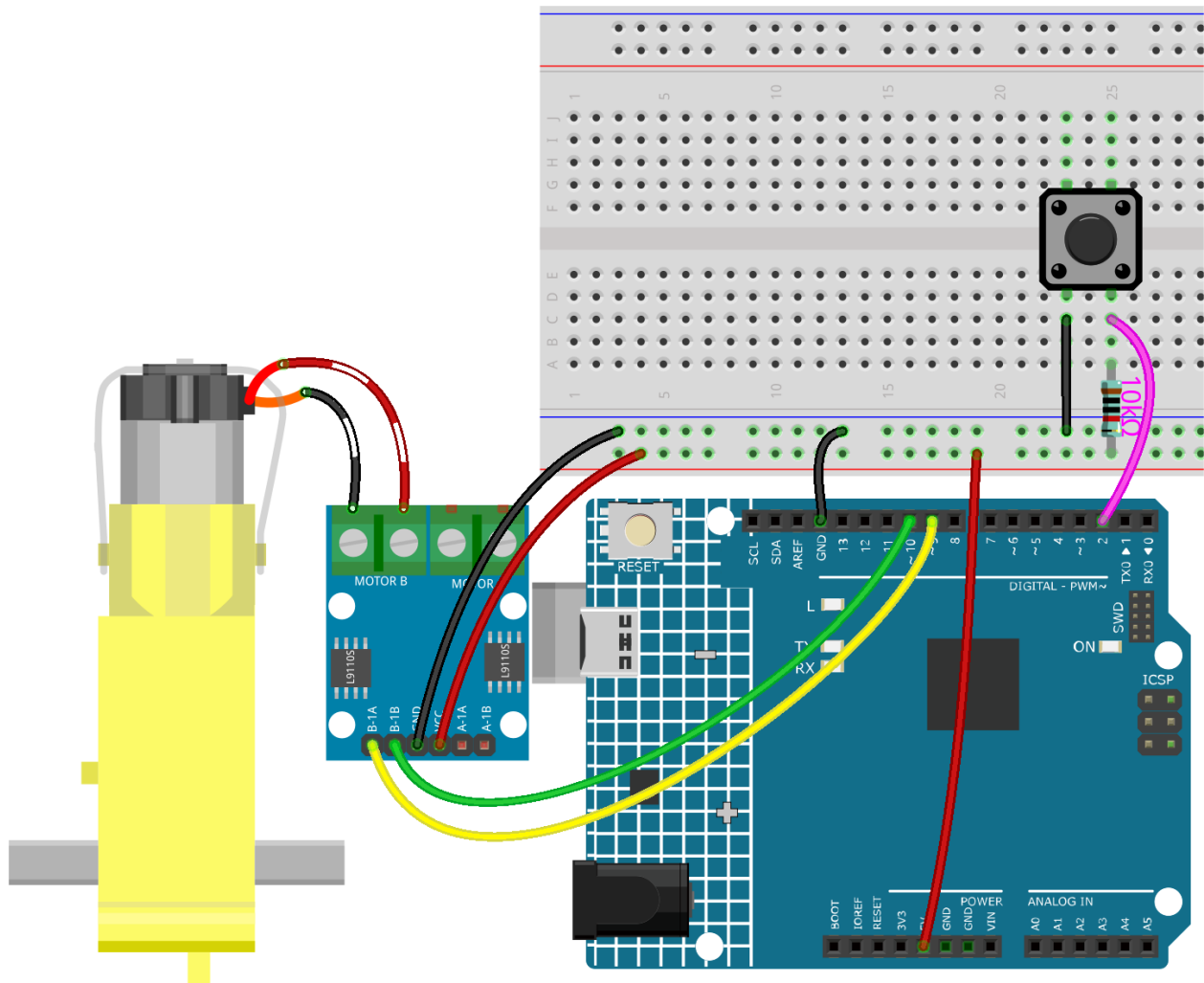
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Button</i>	
<i>TT Motor</i>	-
<i>L9110 Motor Driver Module</i>	-

### Schematic



### Wiring



## Code

### Note:

- Open the 5.3.state\_change\_detection.ino file under the path of 3in1-kit\learning\_project\5.3.state\_change\_detection.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, you press the button and the motor will turn; until you press the button again, the motor will stop.

### How it works?

1. Create variables and define pins for the motor and button.

```
...
int detectionState = 0;
int buttonState = 0;
int lastButtonState = 0;
```

- detectionState is a flag whose value changes each time the button is pressed, e.g., 0 this time,

1 the next, and so on alternately.

- `buttonState` and `lastButtonState` are used to record the state of the button this time and the last time, to compare whether the button was pressed or released.

2. Initialize each pin and set the baud rate of the serial monitor.

```
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
  pinMode(A_1A, OUTPUT);
  pinMode(A_1B, OUTPUT);
}
```

3. First read the state of the button, and if the button is pressed, the variable `detectionState` will switch its value from 0 to 1 or 1 to 0. When `detectionState` is 1, the motor will be turned. It has the effect that this time the button is pressed, the motor turns, the next time the button is pressed, the motor stops, and so on alternately.

```
void loop() {
  // Toggle the detectionState each time the button is pressed
  buttonState = digitalRead(buttonPin);
  if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      detectionState=(detectionState+1)%2;
      Serial.print("The detection state is: ");
      Serial.println(detectionState);
    }
    delay(50);
  }
  lastButtonState = buttonState;

  // According to the detectionState, start the motor
  if(detectionState==1){
    digitalWrite(A_1A,HIGH);
    digitalWrite(A_1B,LOW);
  }else{
    digitalWrite(A_1A,LOW);
    digitalWrite(A_1B,LOW);
  }
}
```

The entire workflow is as follows.

- Read the button value.

```
buttonState = digitalRead(buttonPin);
```

- If `buttonState` and `lastButtonState` are not equal, it means that the button state has changed, continue with the next judgment, and store the button state at this time into the variable `lastButtonState`. `delay(50)` is used to eliminate jitter.

```
if (buttonState != lastButtonState) {
  ...
  delay(50);
}
lastButtonState = buttonState;
```

- When the button is pressed, its value is HIGH. Here, when the button is pressed, the value of the variable `detectionState` is changed, e.g., from 0 to 1 after an operation.

```
if (buttonState == HIGH) {
  detectionState=(detectionState+1)%2;
  Serial.print("The detection state is: ");
  Serial.println(detectionState);
}
```

- When the variable `detectionState` is 1, let the motor rotate, otherwise stop.

```
if(detectionState==1){
  digitalWrite(A_1A,HIGH);
  digitalWrite(A_1B,LOW);
}else{
  digitalWrite(A_1A,LOW);
  digitalWrite(A_1B,LOW);
}
```

#### 4.5.4 5.4 Interval

Sometimes you need to do two things at once. For example you might want to blink an LED while reading a button press. In this case, you can't use `delay()`, because Arduino pauses your program during the `delay()`. If the button is pressed while Arduino is paused waiting for the `delay()` to pass, your program will miss the button press.

An analogy would be warming up a pizza in your microwave, and also waiting some important email. You put the pizza in the microwave and set it for 10 minutes. The analogy to using `delay()` would be to sit in front of the microwave watching the timer count down from 10 minutes until the timer reaches zero. If the important email arrives during this time you will miss it.

What you would do in real life would be to turn on the pizza, and then check your email, and then maybe do something else (that doesn't take too long!) and every so often you will come back to the microwave to see if the timer has reached zero, indicating that your pizza is done.

This sketch demonstrates how to tone an buzzer without using `delay()`. It turns the buzzer on and then makes note of the time. Then, each time through `loop()`, it checks to see if the desired interval time has passed. If it has, it tone the buzzer and makes note of the new time. In this way the buzzer tones continuously while the sketch execution never lags on a single instruction.

Based on this condition, we can add the code of the button to control the LED, it will not be disturbed by the buzzer playing music.

#### Required Components

In this project, we need the following components.

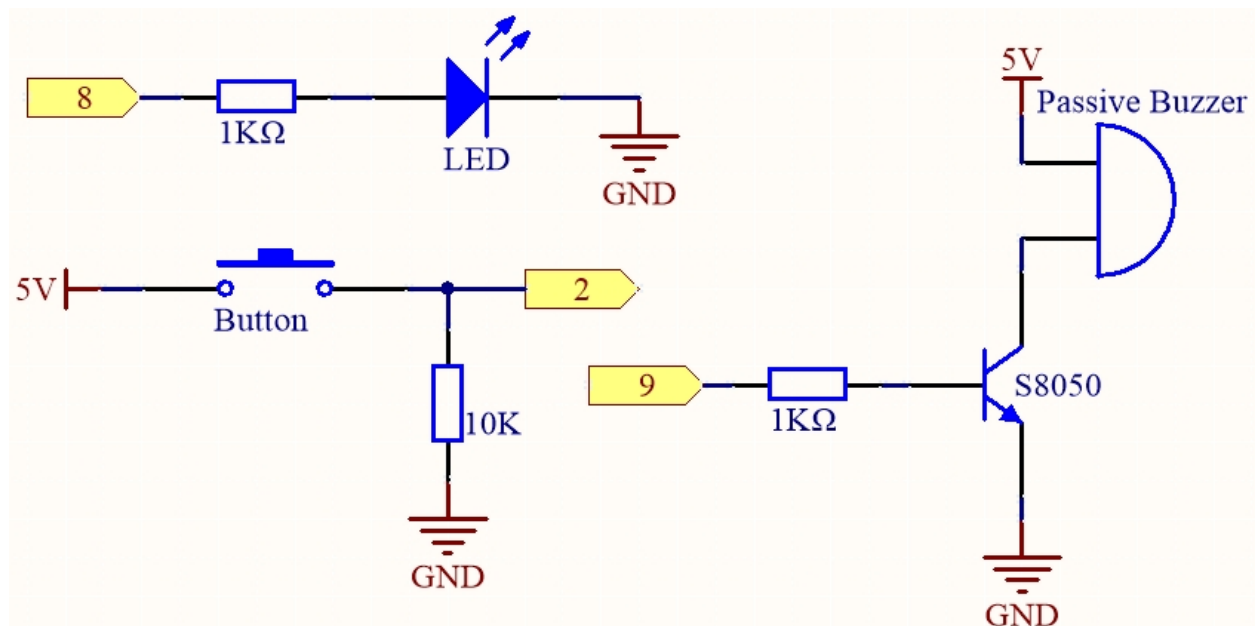
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

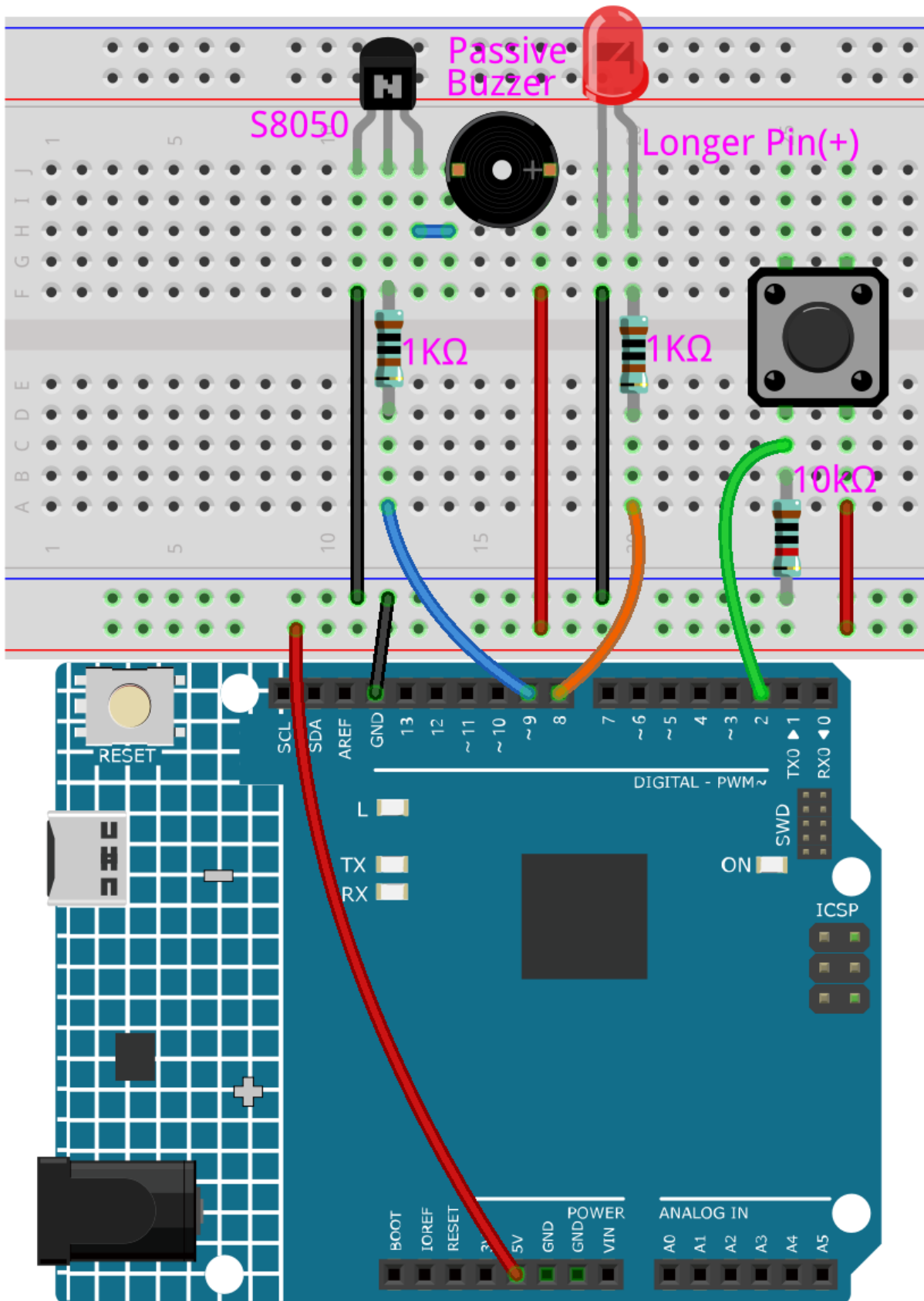
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Button</i>	
<i>Buzzer</i>	

### Schematic



### Wiring





### Note:

- Open the 5.4.interval.ino file under the path of 3in1-kit\learning\_project\5.4.interval.
  - Or copy this code into **Arduino IDE**.
- 

After the code is uploaded successfully, the buzzer will play music; whenever you press the button, the LED will light up. The work of LED and buzzer does not interfere with each other.

### How it works?

Initial a variable named `previousMillis` to store previous operating time of microcontroller.

```
unsigned long previousMillis = 0;
```

Mark which note is played.

```
int thisNote=0;
```

The interval time of each note.

```
long interval = 1000;
```

In `loop()`, declare `currentMillis` to store the current time.

```
unsigned long currentMillis = millis();
```

When the interval between the current operating time and last updating time is larger than 1000ms, certain functions are triggered. Meanwhile, update the `previousMillis` to the current time for the next triggering that is to happen 1 second latter.

```
if (currentMillis - previousMillis >= interval) {  
    previousMillis = currentMillis; // save the last time of the last tone  
    //...  
}
```

Play the notes in the melody one by one.

```
tone(buzzerPin,melody[thisNote],100);  
interval=1000/noteDurations[thisNote]; // interval at which to tone  
thisNote=(thisNote+1)%(sizeof(melody)/2); //iterate over the notes of the melody
```

The button control the LED.

```
// play button & led  
digitalWrite(ledPin,digitalRead(buttonPin));
```

### 4.5.5 5.5 Use Internal Library

In the Arduino IDE, you can use many built-in libraries by adding the corresponding .h file directly to your code. This project uses the Servo library to drive the Servo, so that it can rotate between 0° and 180°.

#### Required Components

In this project, we need the following components.

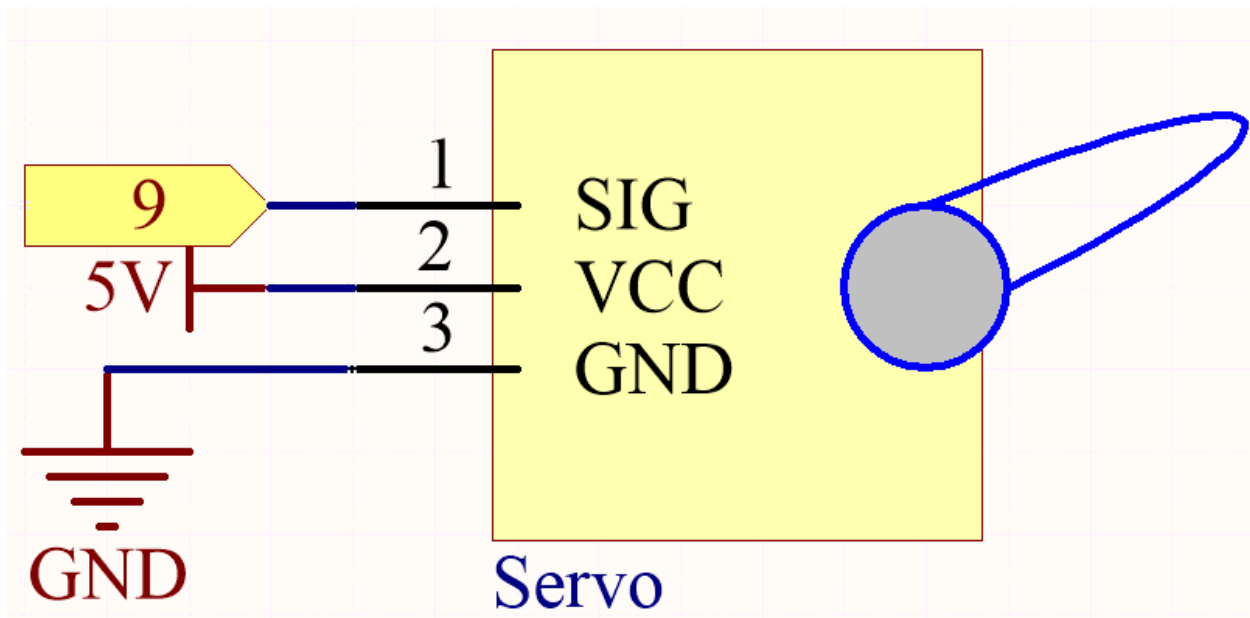
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

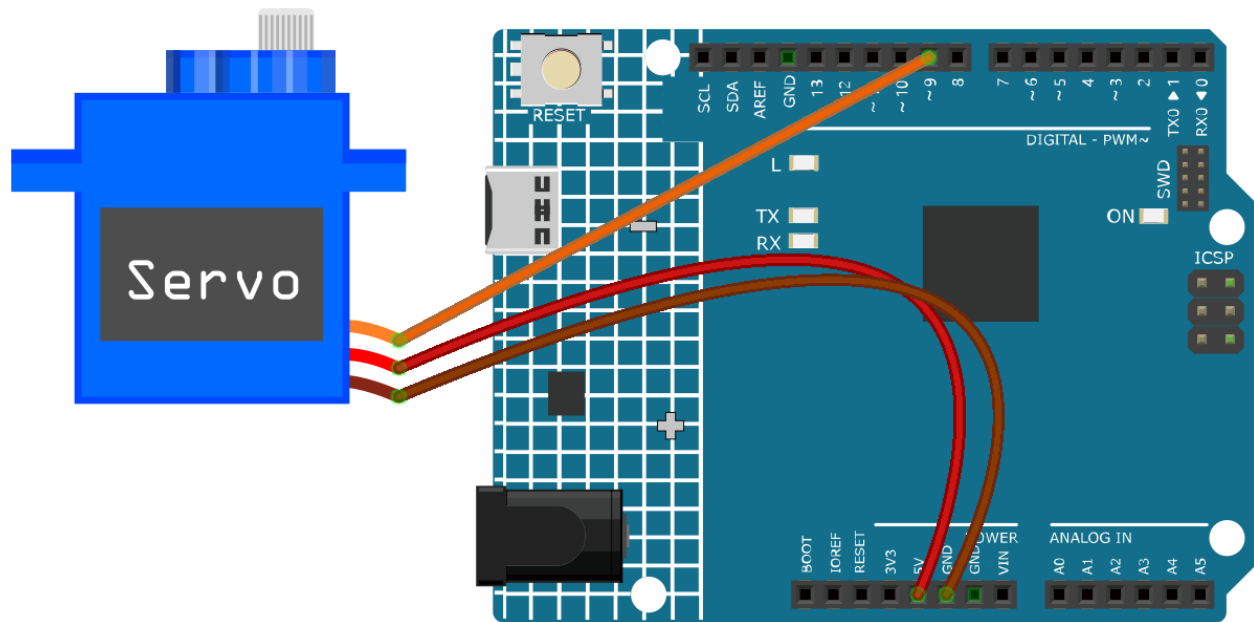
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>Servo</i>	

#### Schematic



In this project, we use PWM pin 9 to drive the Servo, and get the orange wire of the servo connected to the PWM pin 9, the red one to 5V, and the brown one to GND.

#### Wiring



### Code

---

#### Note:

- Open the `5.5.use_internal_library.ino` file under the path of `3in1-kit\learning_project\5.5.use_internal_library`.
  - Or copy this code into **Arduino IDE**.
- 

Once you finish uploading the codes to the R4 board, you can see the servo arm rotating in the range  $0^{\circ}\sim 180^{\circ}$ .

#### How it works?

By calling the library `Servo.h`, you can drive the servo easily.

```
#include <Servo.h>
```

Library Functions:

```
Servo
```

Create **Servo** object to control a servo.

```
uint8_t attach(int pin);
```

Call `pinMode()` to turn a pin into a servo driver and return 0 on failure.

```
void detach();
```

Release a pin from servo driving.

```
void write(int value);
```

Set the angle of the servo in degrees, 0 to 180.

```
int read();
```

Return that value set with the last `write()`.

```
bool attached();
```

Return 1 if the servo is currently attached.

## 4.5.6 5.6 Map

If you observe carefully, you will notice that many values have different ranges in programming. For example, the range of values for analog inputs is (0~1023). The value range for the analog output is (0~255). The output angle of the servo is (0~180).

This means that if we want to use the potentiometer to control the brightness of the LED or the angle of the servo, we need to go through a mapping operation.

Now let's see how to achieve it.

### Required Components

In this project, we need the following components.

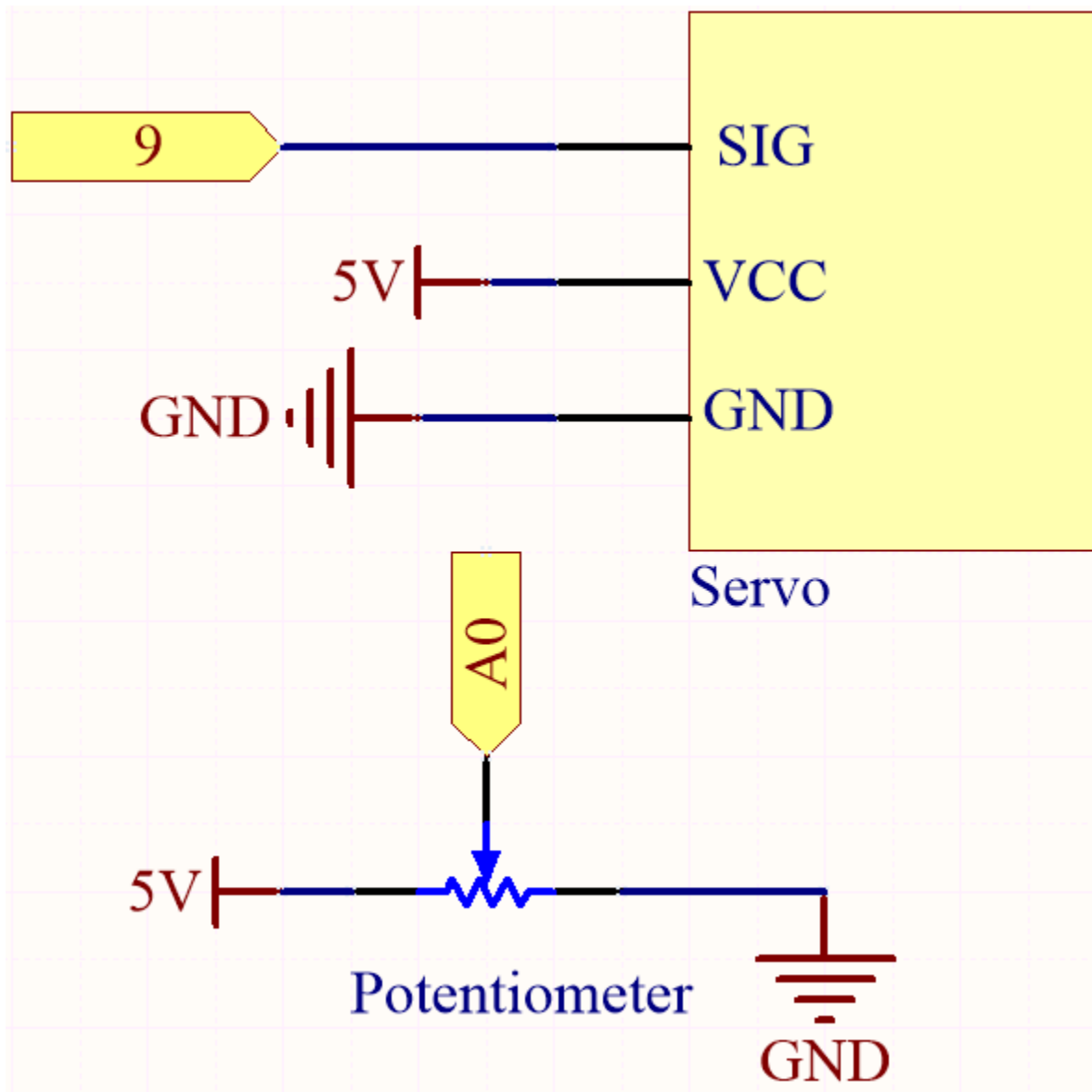
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

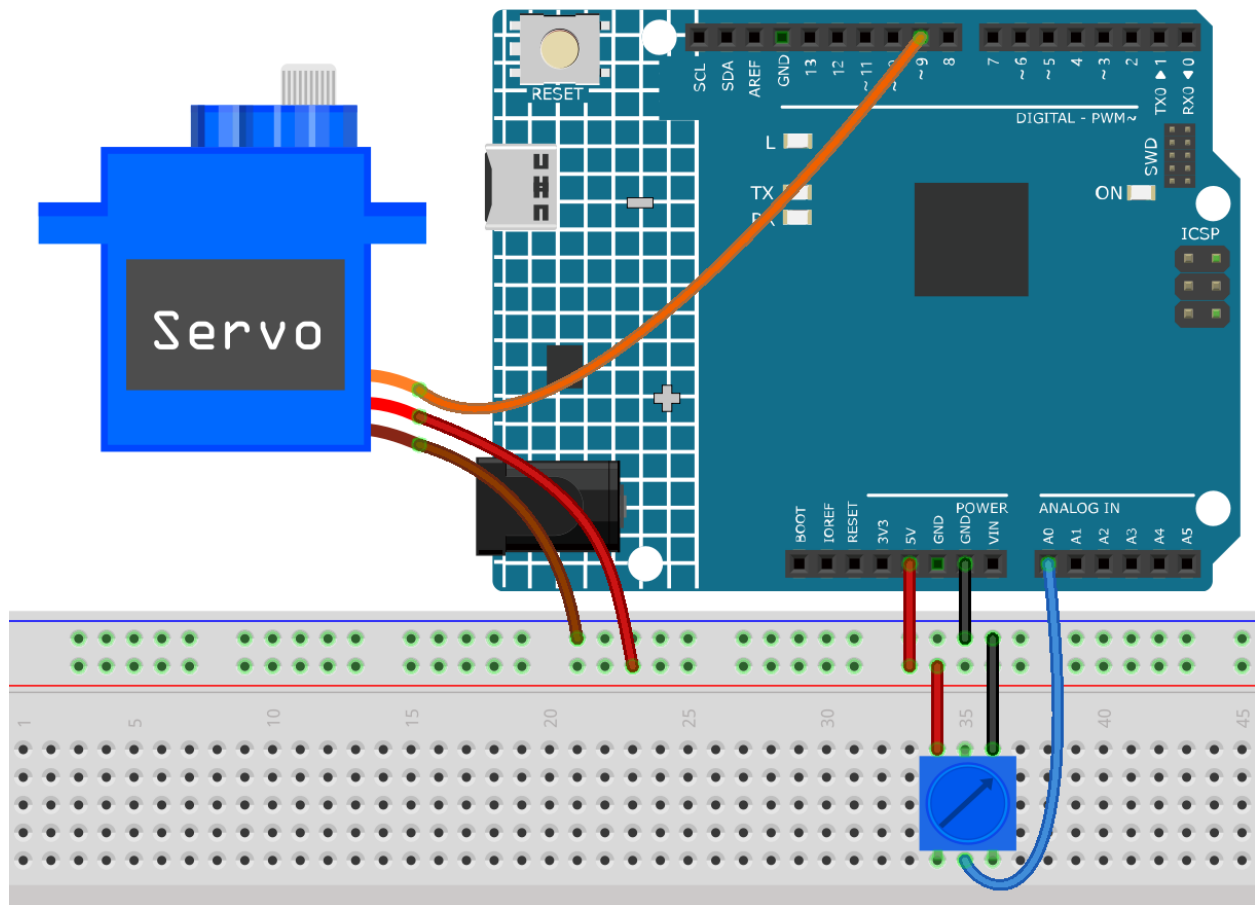
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Servo</i>	
<i>Potentiometer</i>	

### Schematic



Wiring



## Code

### Note:

- Open the 5.6.map.ino file under the path of 3in1-kit\learning\_project\5.6.map.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, you can rotate the potentiometer back and forth, and the output shaft of the servo will rotate back and forth.

### How it works?

`map(value, fromLow, fromHigh, toLow, toHigh)`: Map a number from one range to another. That is, a from-Low value is mapped to toLow, and a fromHigh value is mapped to toHigh.

#### Syntax

`map(value, fromLow, fromHigh, toLow, toHigh)`

#### Parameters

- **value**: the number to map.
- **fromLow**: the lower bound of the value's current range.
- **fromHigh**: the upper bound of the value's current range.
- **toLow**: the lower bound of the value's target range.

- `toHigh`: the upper bound of the value's target range.

If the potentiometer controls the LED, you can also use the `map` to complete the task.

```
int x = analogRead(knob);
int y = map(x,0,1023,0,255);
analogWrite(led,y);
```

### Notes and Warnings

- The “lower bound” of both ranges may be larger or smaller than the “upper bound”, which means that the `map()` function can be used to reverse a range of numbers.

```
y = map(x,0,180,180,0);
```

- Mapping also works well for negative numbers.

```
y = map(x,0,1023,-90,90);
```

- The mapping uses integers, and the decimal places of floats are discarded.

## 4.5.7 5.7 Tone() or noTone()

`Tone()` is used to generate a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to `noTone()`.

In this project, use this two functions to make the passive buzzer vibrate to make sound. Like the active buzzer, the passive buzzer also uses the phenomenon of electromagnetic induction to work. The difference is that a passive buzzer does not have oscillating source, so it will not beep if DC signals are used. But this allows the passive buzzer to adjust its own oscillation frequency and can emit different notes such as “doh, re, mi, fa, sol, la, ti”.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

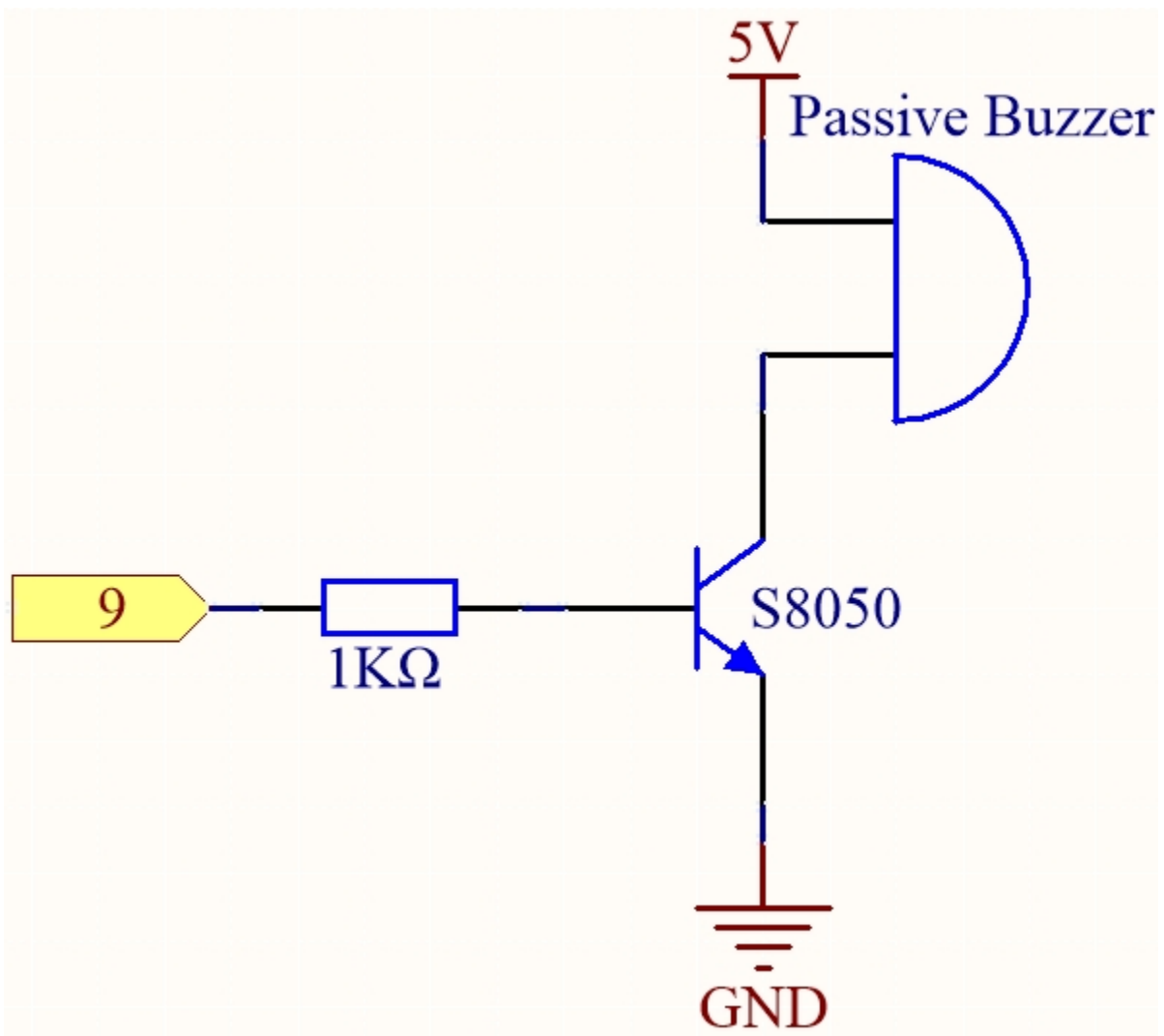
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Buzzer</i>	

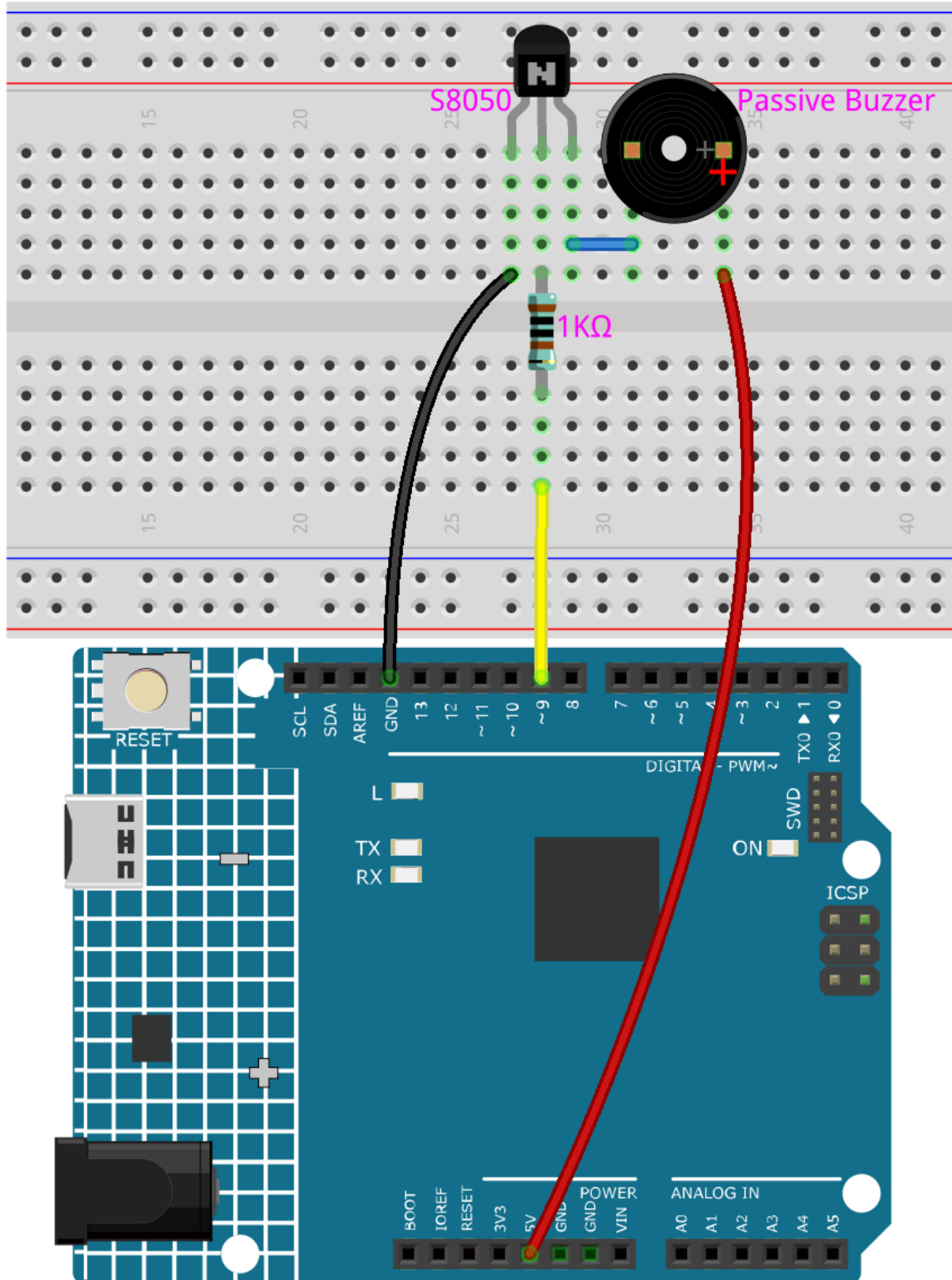
### Schematic





Connect the cathode of the Buzzer to GND, and the anode to the digital pin 9.

#### Wiring



Code

**Note:**

- Open the 5.7.tone\_notone.ino file under the path of 3in1-kit\learning\_project\5.7.tone\_notone.
  - Or copy this code into **Arduino IDE**.
- 

At the time when you finish uploading the codes to the R4 board, you can hear a melody containing seven notes.

**How it works?**

There are two points needing your attention:

1. **tone()** & **noTone()**: This function is used to control the sound of the passive buzzer directly and its prototype is as follows:

**Syntax**

```
void tone(int pin, unsigned int frequency)
```

```
void tone(int pin, unsigned int frequency, unsigned long duration)
```

**Parameters**

- **pin**: The Arduino pin on which to generate the tone.
- **frequency**: The frequency of the tone in hertz.
- **duration**: The duration of the tone in milliseconds (optional)

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin (so as to make the passive buzzer vibrate to make sound). A duration can be specified, otherwise the wave continues until a call to **noTone()**. The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to **tone()** will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the **tone()** function will interfere with PWM output on pins 3 and 11.

It is not possible to generate tones lower than 31Hz.

**Syntax**

```
void noTone(int pin)
```

**Parameters**

**pin**: The Arduino pin on which to generate the tone.

Stops the generation of a square wave triggered by **tone()**. Has no effect if no tone is being generated.

Having known the two functions, you may grasp the codes—the installation of the array **melody[]** and the array **noteDurations[]** is the preparation of the subsequently several times of calling of the function **tone()** and the changing of tone and duration in the loop for better effect of music play.

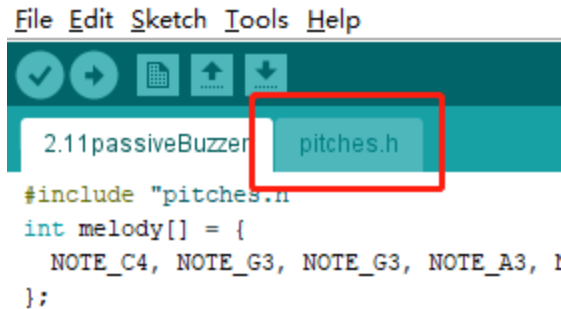
2. **pitches.h**: The code uses an extra file, **pitches.h**. This file contains all the pitch values for typical notes. For example, **NOTE\_C4** is middle C. **NOTE\_FS4** is F sharp, and so forth. This note table was originally written by Brett Hagman, on whose work the **tone()** command was based. You may find it useful whenever you want to make musical notes.

```
#include "pitches.h"
```

---

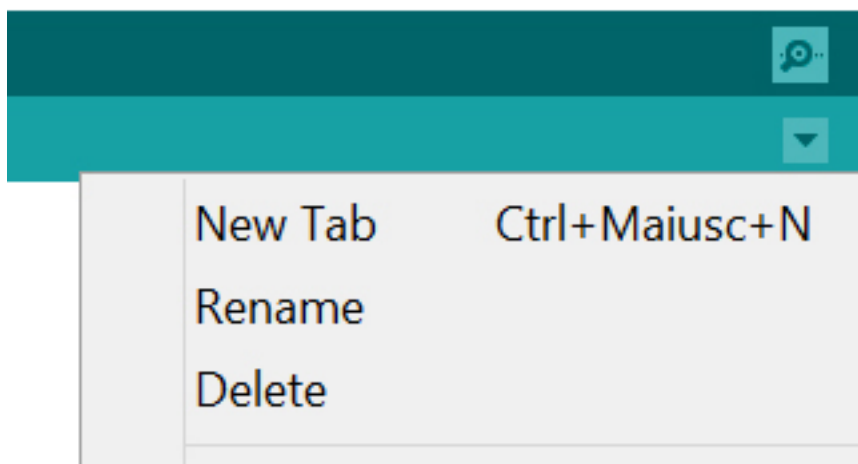
**Note:** There is already a **pitches.h** file in this sample program. If we put it together with the main code in one folder, the successive steps of installing **pitches.h** can be omitted.

---



After you open the code file, if you cannot open the `pitches.h` code, you can just create one manually. The steps are as follows:

To make the `pitches.h` file, either click on the button just below the serial monitor icon and choose **New Tab**, or use **Ctrl+Shift+N**.



Then paste in the following code and save it as `pitches.h`:

```

/*****
Public Constants
*****/
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78

```

(continues on next page)

(continued from previous page)

```
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
```

(continues on next page)

(continued from previous page)

```

#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 49

```

### 4.5.8 5.8 User-defined Function

In c, we can divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the C program. In other words, we can say that the collection of functions creates a program. The function is also known as procedure or subroutine in other programming languages.

There are the following advantages of functions.

- By using functions, we can avoid rewriting same logic/code again and again in a program.
- We can call C functions any number of times in a program and from any place in a program.
- We can track a large C program easily when it is divided into multiple functions.
- Reusability is the main achievement of C functions.
- However, Function calling is always a overhead in a C program.

There are two types of functions in C programming:

- **Library Functions:** the functions which are declared in the C header files.
- **User-defined functions:** the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.

In this project, define a function to read the value of the ultrasonic module.

#### Required Components

In this project, we need the following components.

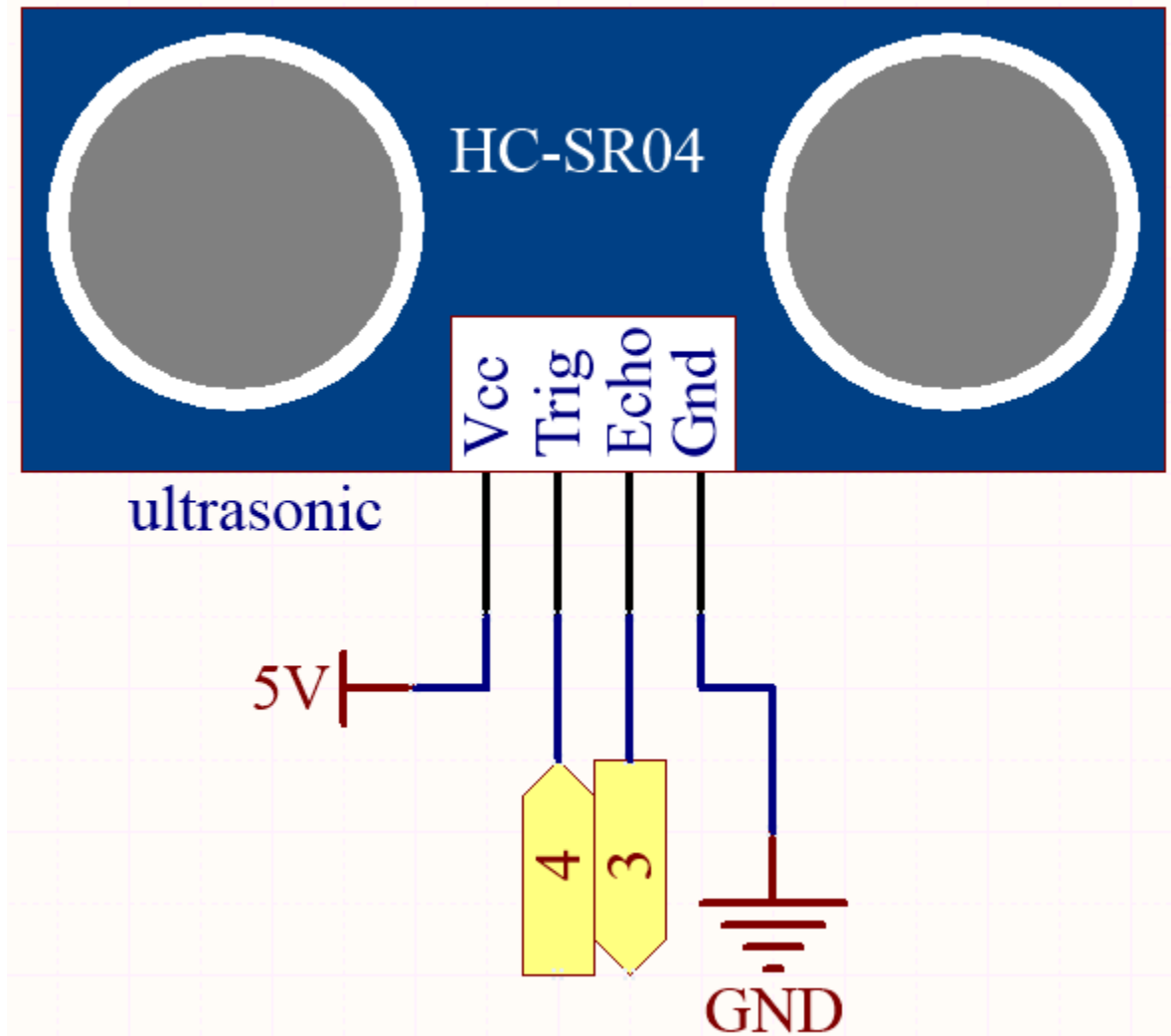
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

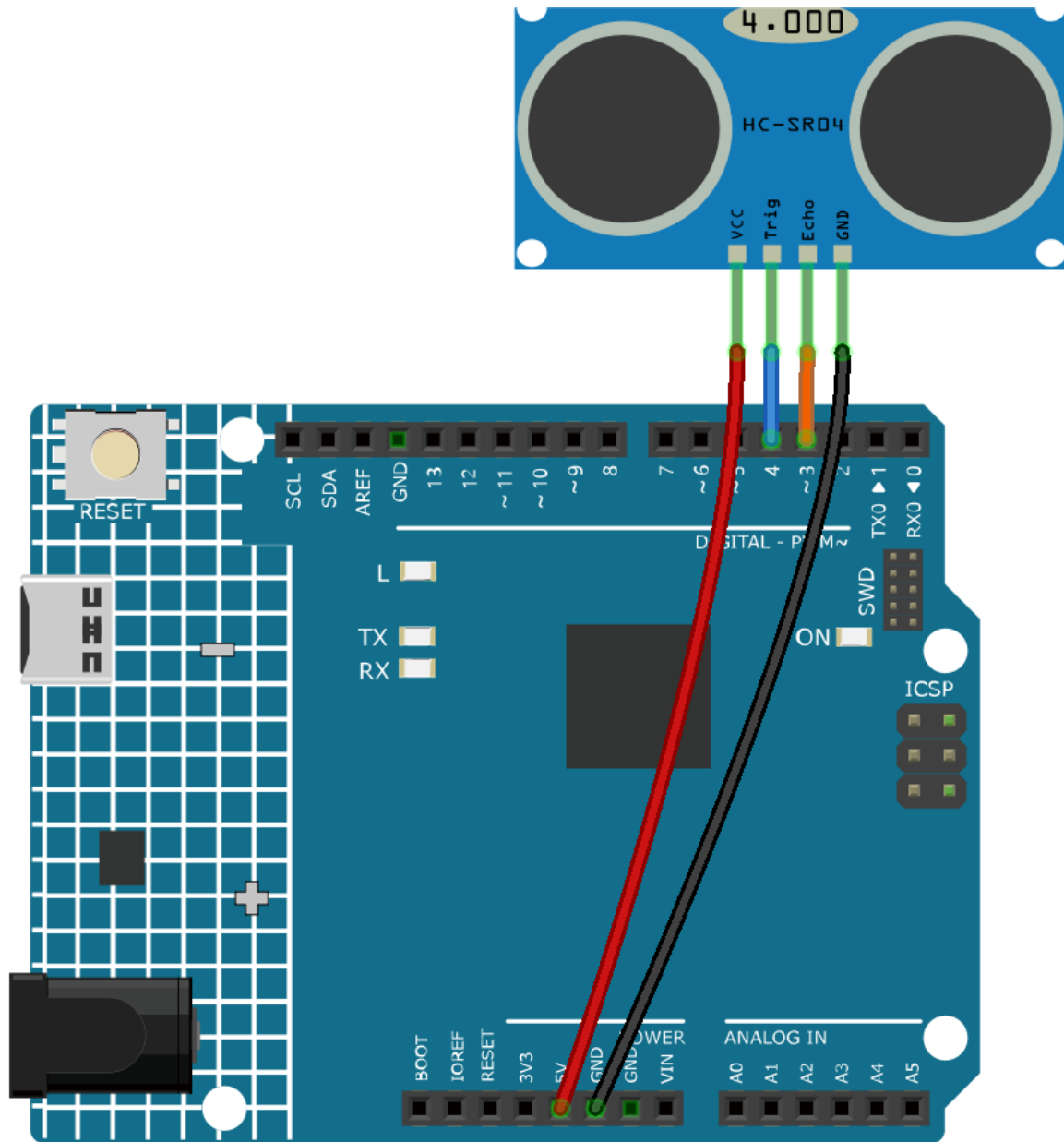
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>Ultrasonic Module</i>	

### Schematic



### Wiring



## Code

### Note:

- Open the 5.8.user\_function.ino file under the path of 3in1-kit\learning\_project\5.8.user\_function.
- Or copy this code into **Arduino IDE**.

After the code is successfully uploaded, the serial monitor will print out the distance between the ultrasonic sensor and the obstacle ahead.

### How it works?

About the application of ultrasonic sensor, we can directly check the subfunction.



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

The trigPin of the ultrasonic module transmits a 10us square wave signal every 2us

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

The echoPin receives a high level signal if there is an obstacle within the range and use the pulseIn() function to record the time from sending to receiving.

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

The speed of sound is 340 m/s or 29 microseconds per centimeter.

This gives the distance travelled by the square wave, outbound and return, so we divide by 2 to get the distance of the obstacle.

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

Note that the ultrasonic sensor will pause the program when it is working, which may cause some lagging when writing complex projects.

### 4.5.9 5.9 ShiftOut(LED)

shiftOut() will make 74HC595 output 8 digital signals. It outputs the last bit of the binary number to Q0, and the output of the first bit to Q7. In other words, writing the binary number “00000001” will make Q0 output high level and Q1~Q7 output low level.

In this project, you will learn how to use 74HC595. 74HC595 consists of an 8bit shift register and a storage register with threestate parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU.

Specifically, 74hc595 can replace 8 pins for digital signal output by writing an 8-bit binary number.

- [Binary number - Wikipedia](#)

#### Required Components

In this project, we need the following components.

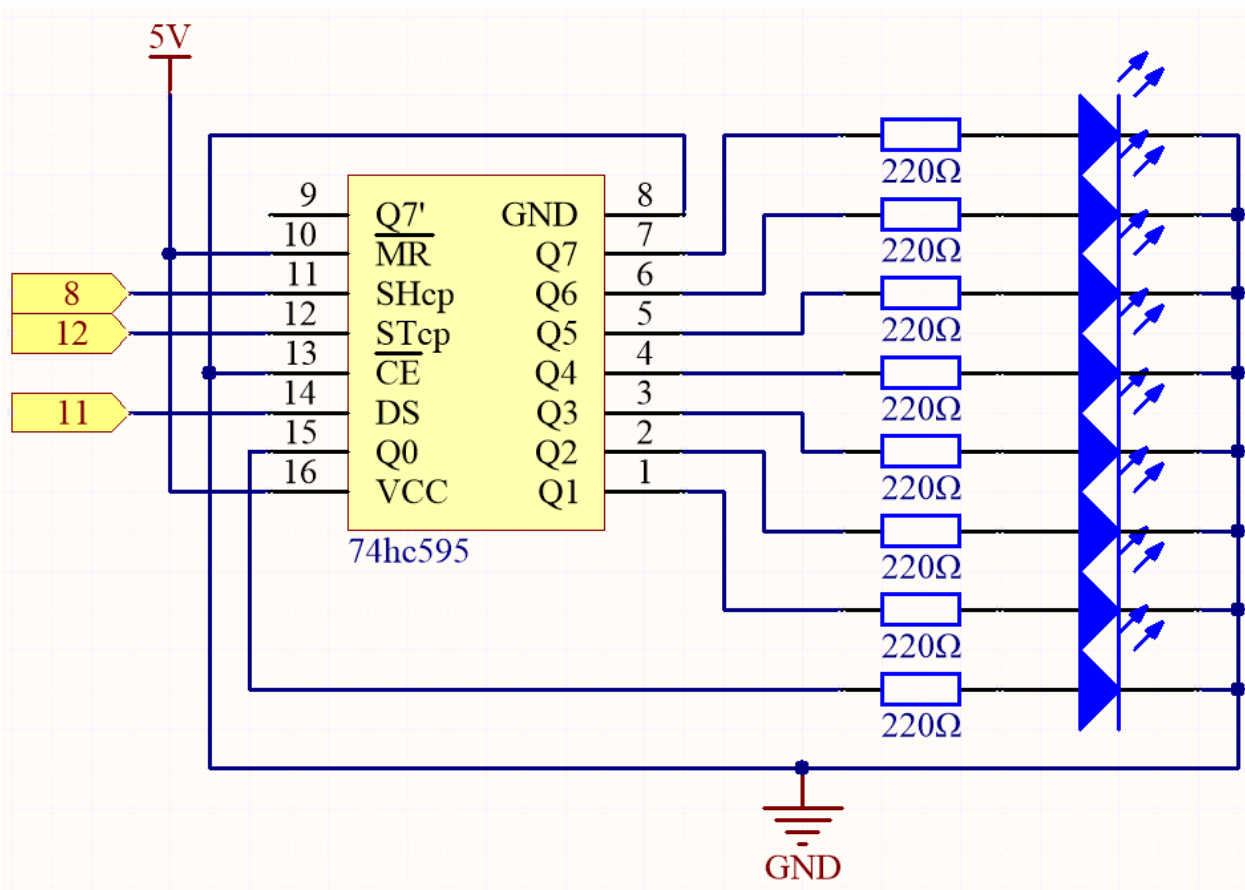
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

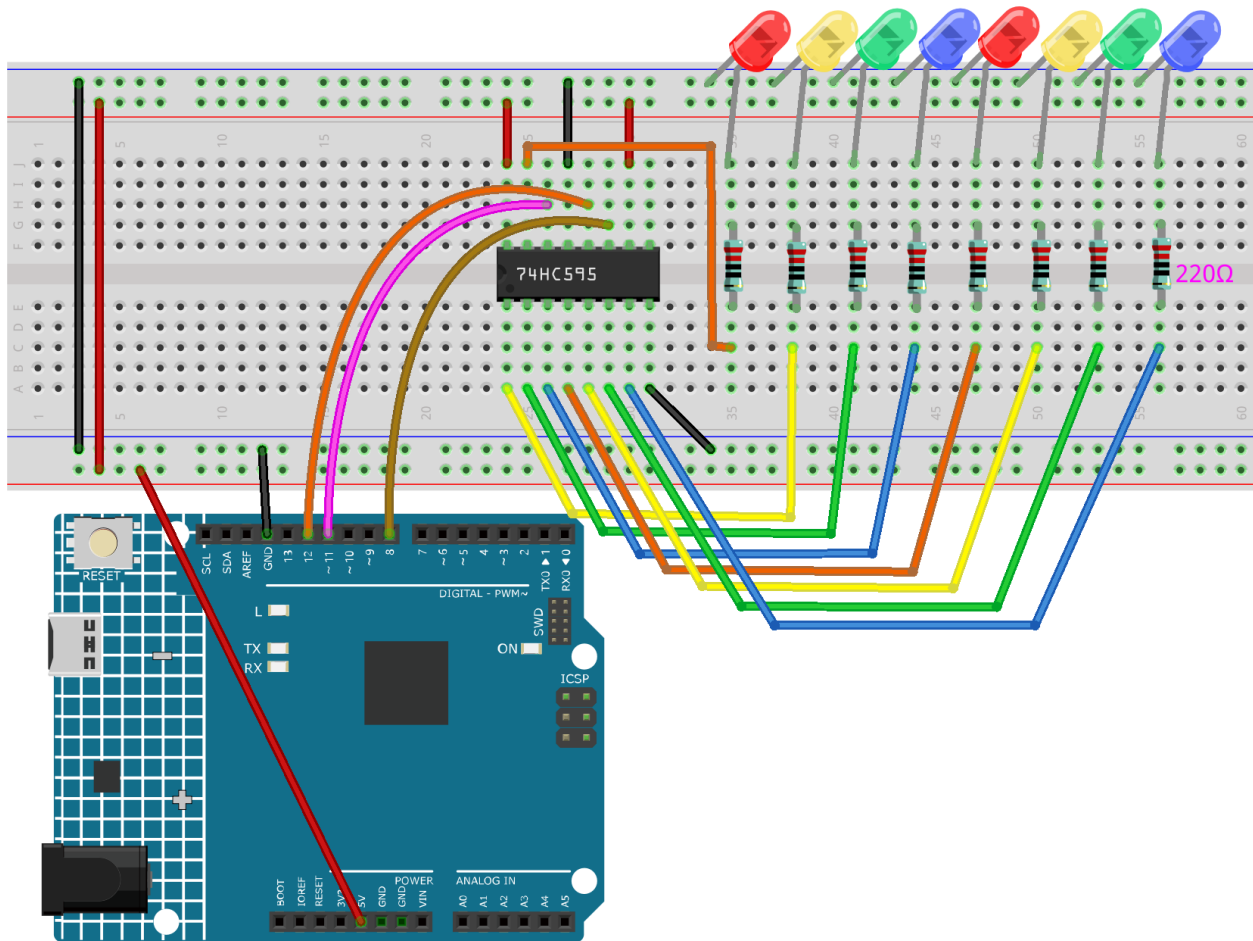
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>74HC595</i>	

## Schematic



- When MR (pin10) is high level and OE (pin13) is low level, data is input in the rising edge of SHcp and goes to the memory register through the rising edge of SHcp.
- If the two clocks are connected together, the shift register is always one pulse earlier than the memory register.
- There is a serial shift input pin (Ds), a serial output pin (Q) and an asynchronous reset button (low level) in the memory register.
- The memory register outputs a Bus with a parallel 8-bit and in three states.
- When OE is enabled (low level), the data in memory register is output to the bus(Q0 ~ Q7).

## Wiring



## Code

### Note:

- Open the 5.9.shiftout\_led.ino file under the path of 3in1-kit\learning\_project\5.9.shiftout\_led.
- Or copy this code into **Arduino IDE**.

When you finish uploading the codes to the R4 board, you can see the LEDs turning on one after another.

### How it works?

Declare an array, store several 8 bit binary numbers that are used to change the working state of the eight LEDs controlled by 74HC595.

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

Set STcp to low level first and then high level. It will generate a rising edge pulse of STcp.

```
digitalWrite(STcp, LOW);
```

`shiftOut()` is used to shift out a byte of data one bit at a time, which means to shift a byte of data in `dataArray[num]` to the shifting register with the DS pin. **MSBFIRST** means to move from high bits.

```
shiftOut(DS, SHcp, MSBFIRST, dataArray[num]);
```

After `digitalWrite(STcp, HIGH)` is run, the STcp will be at the rising edge. At this time, the data in the shift register will be moved to the memory register.

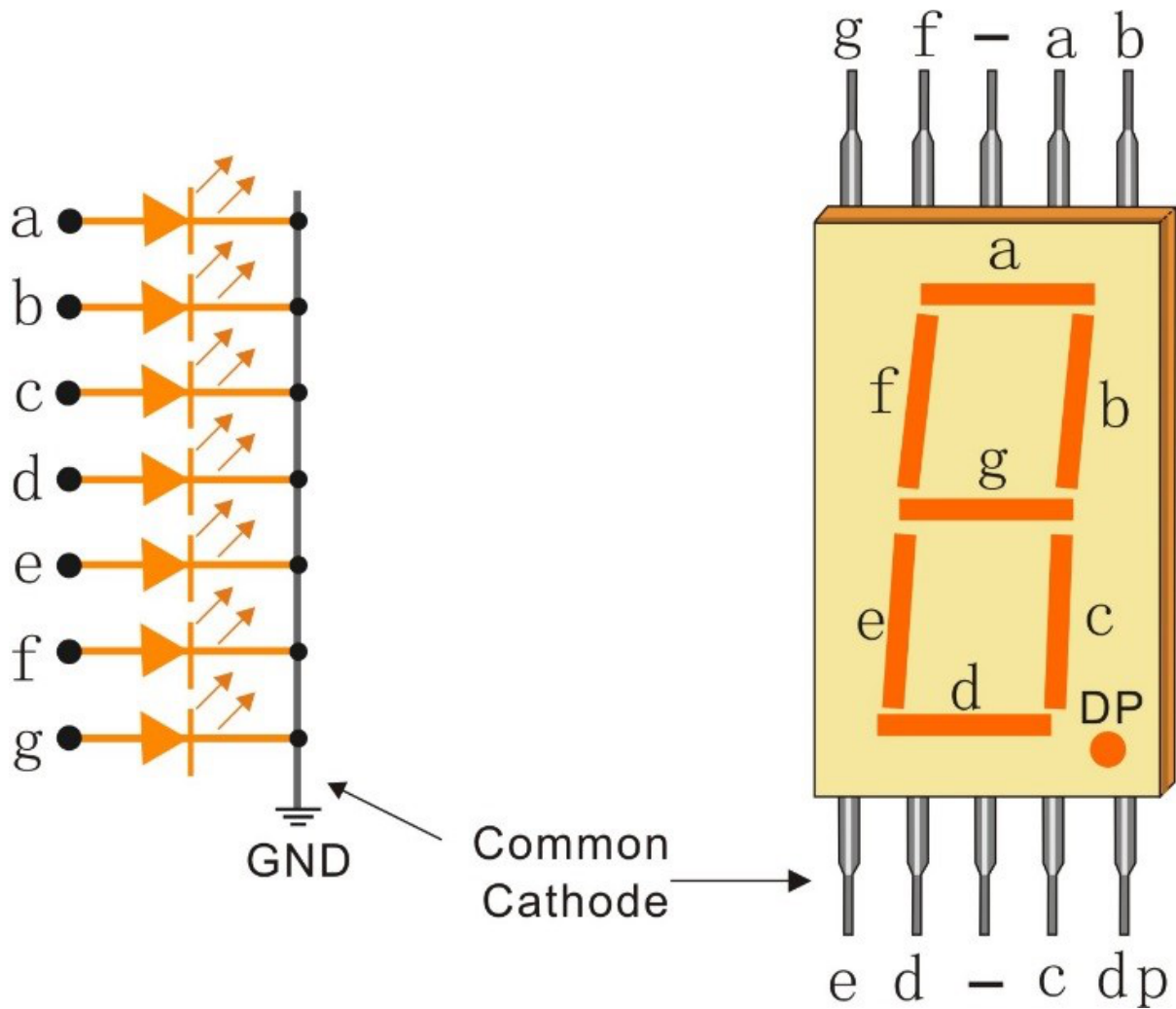
```
digitalWrite(STcp, HIGH);
```

A byte of data will be transferred into the memory register after 8 times. Then the data of memory register are output to the bus (Q0-Q7). For example, shiftout `B00000001` will light up the LED controlled by Q0 and turn off the LED controlled by Q1~Q7.

#### 4.5.10 5.10 ShiftOut(Segment Display)

Previously, we used the `shiftout()` function to light up eight LEDs; here we use it to display 0-9 on the 7-segment Display.

The 7-segment Display is essentially a device packaged by 8 LEDs, of which 7 strip-shaped LEDs form an “8” shape, and there is a slightly smaller dotted LED as a decimal point. These LEDs are marked as a, b, c, d, e, f, g, and dp. They have their own anode pins and share cathodes. Their pin locations are shown in the figure below.



### Required Components

In this project, we need the following components.

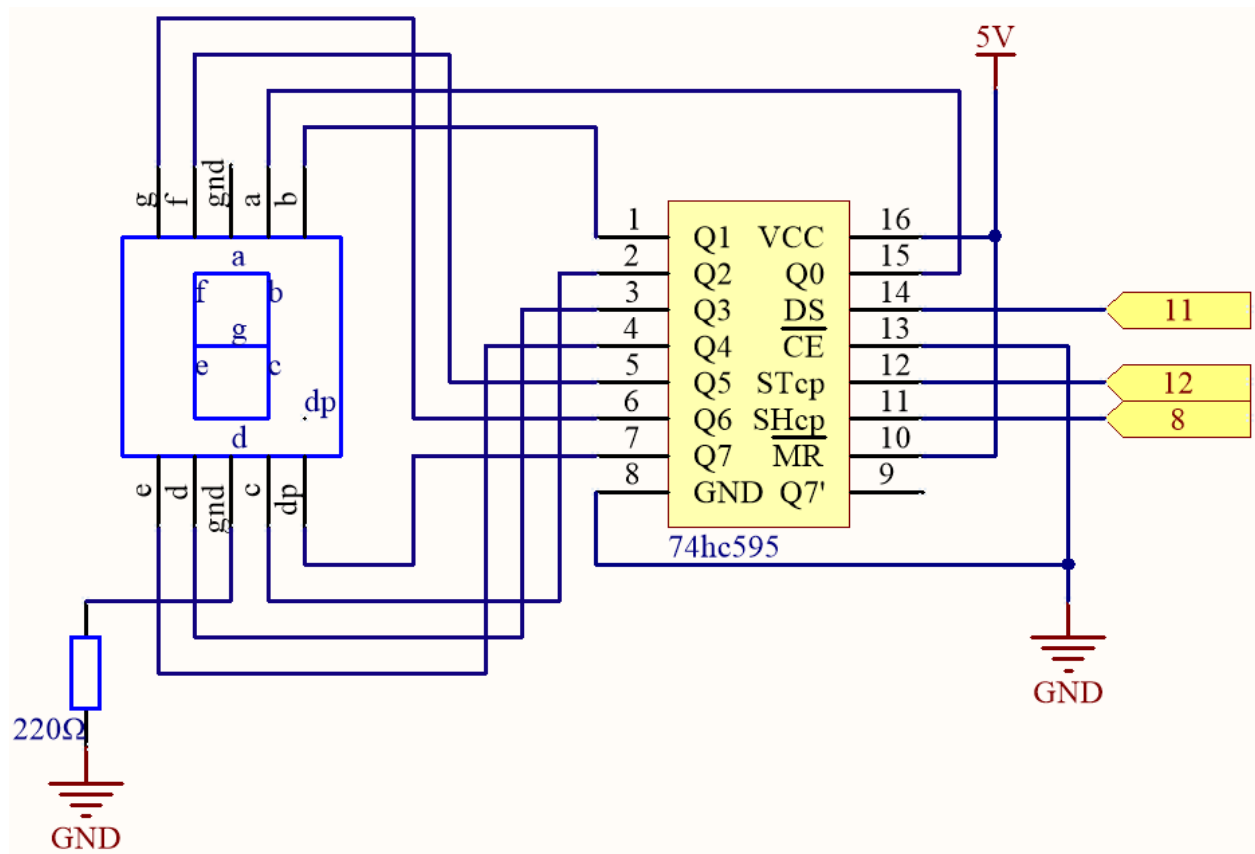
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>7-segment Display</i>	
<i>74HC595</i>	

### Schematic



### Wiring

Table 1: Wiring

74HC595	LED Segment Display
Q0	a
Q1	b
Q2	c
Q3	d
Q4	e
Q5	f
Q6	g
Q7	dp





---

## Code

---

### Note:

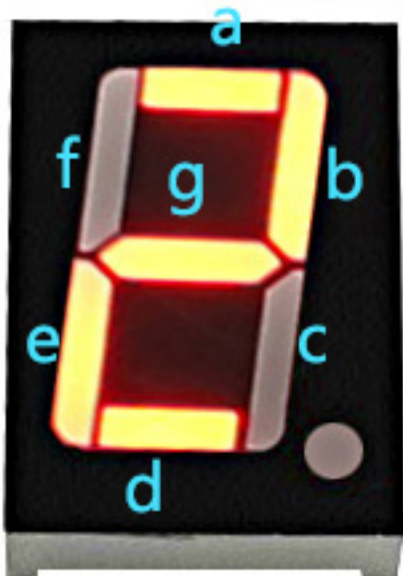
- Open the 5.10.shiftout\_segment.ino file under the path of 3in1-kit\learning\_project\5.10.shiftout\_segment.
  - Or copy this code into **Arduino IDE**.
- 

After the code is uploaded successfully, you will be able to see the LED Segment Display display 0~9 in sequence.

### How it works?

shiftOut() will make 74HC595 output 8 digital signals. It outputs the last bit of the binary number to Q0, and the output of the first bit to Q7. In other words, writing the binary number "00000001" will make Q0 output high level and Q1~Q7 output low level.

Suppose that the 7-segment Display display the number "2", we need to write a high level for a, b, d, e and g, and write a low level for c, f and dp. That is, the binary number "01011011" needs to be written. For readability, we will use hexadecimal notation as "0x5b".



- [Hexadecimal](#)
- [BinaryHex Converter](#)

Similarly, we can also make the 7-Segment Display display other numbers in the same way. The following table shows the codes corresponding to these numbers.

Table 2: Glyph Code

Numbers	Binary Code	Hex Code
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

Write these codes into `shiftOut()` to make the LED Segment Display display the corresponding numbers.

#### 4.5.11 5.11 Install External Libraries

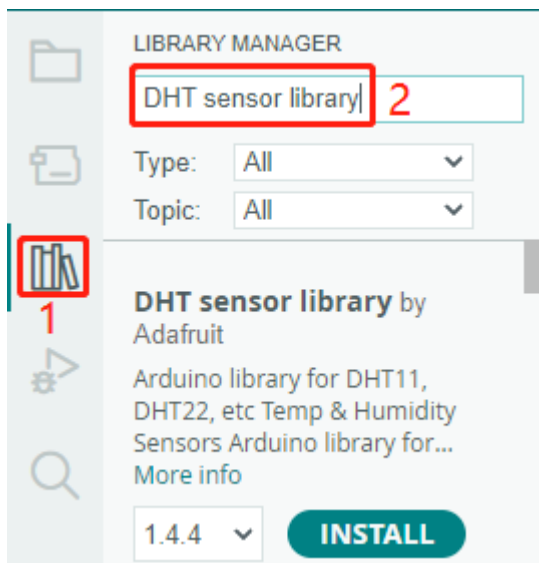
A library is a collection of pre-written code or functions that extend the capabilities of the Arduino IDE. Libraries provide ready-to-use code for various functionalities, allowing you to save time and effort in coding complex features.

There are two main ways to install libraries:

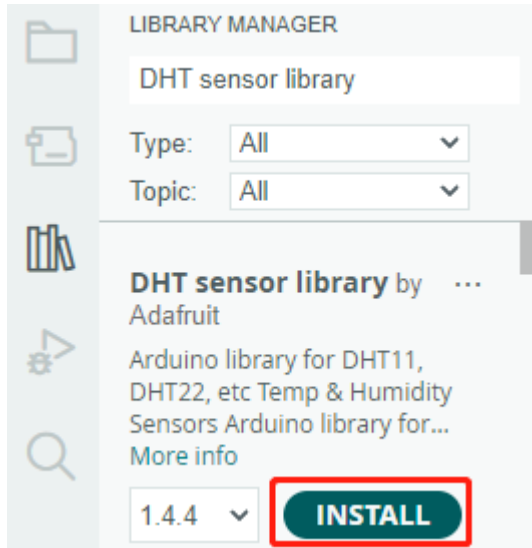
Many libraries are available directly through the Arduino **Library Manager**. You can access the **Library Manager** by following these steps:

1. In the **Library Manager**, you can search for the desired library by name or browse through different categories.

**Note:** In projects where library installation is required, there will be prompts indicating which libraries to install. Follow the instructions provided, such as “The DHT sensor library is used here, you can install it from the **Library Manager**.” Simply install the recommended libraries as prompted.



2. Once you find the library you want to install, click on it and then click the **Install** button.



3. The Arduino IDE will automatically download and install the library for you.

### Related Components

Below are the related components, you can click in to learn how to use them.

#### 5.11.1 Liquid Crystal Display

An I2C LCD1602 is composed of an LCD1602 and an I2C module, LCD1602 can be used to display characters, numbers, etc., but need to take up a lot of pins of the main control, after configuring an I2C module, only 2 I/O pins are needed to drive this LCD1602.

Now look at how to make this I2C CDL1602 work.

### Required Components

In this project, we need the following components.

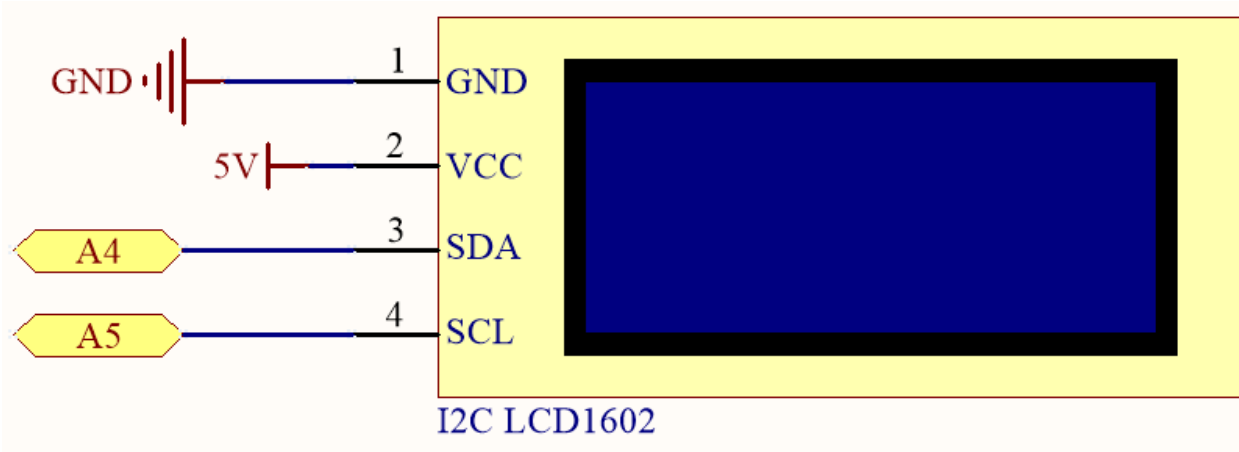
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

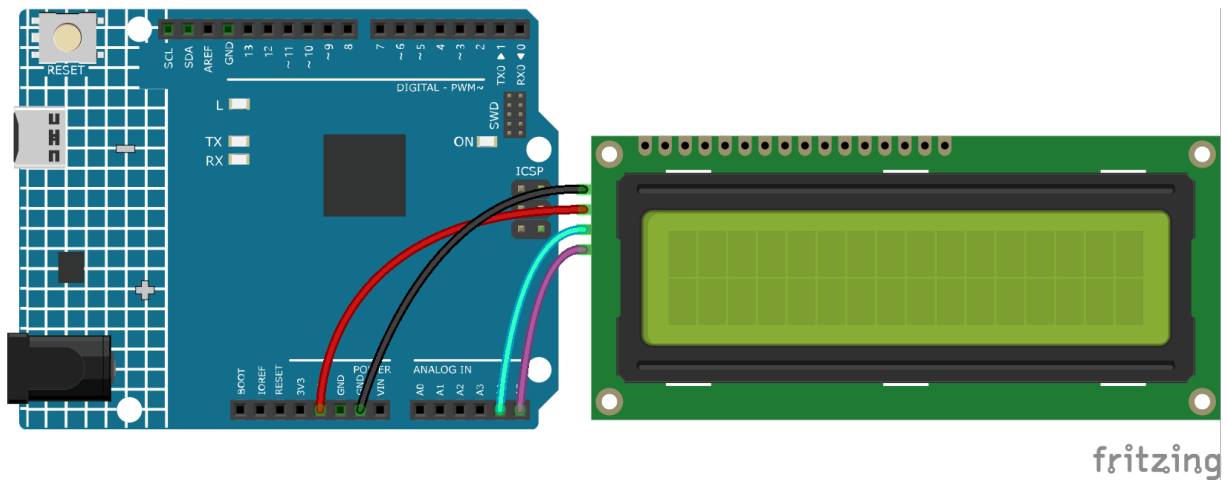
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>I2C LCD1602</i>	

### Schematic



### Wiring

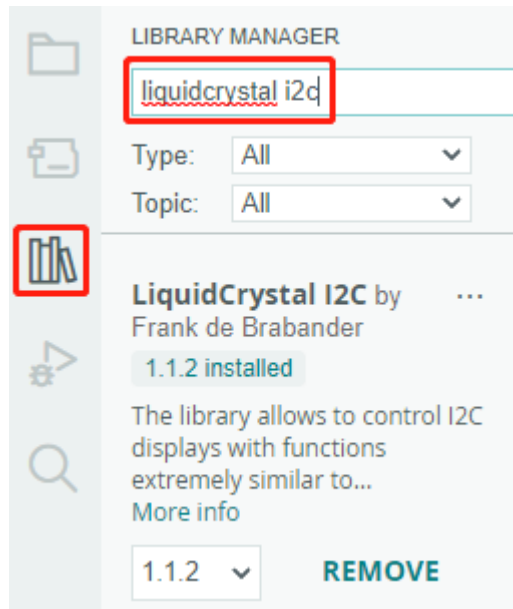


**Note:** The SDA and SCL of the R4 board are the pins A4 and A5.

### Code

#### Note:

- Open the 5.11.liquid\_crystal\_display.ino file under the path of 3in1-kit\learning\_project\5.11.liquid\_crystal\_display.
- Or copy this code into **Arduino IDE**.
- The LiquidCrystal I2C library is used here, you can install it from the **Library Manager**.



After the code is uploaded successfully, you will see “SunFounder”, “Hello World” on the I2C LCD1602.

**Note:** If the code and wiring are fine, but the LCD still does not display content, you can turn the potentiometer on the back.

### How it works?

By calling the library `LiquidCrystal_I2C.h`, you can easily drive the LCD.

```
#include "LiquidCrystal_I2C.h"
```

Library Functions:

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

Creates a new instance of the `LiquidCrystal_I2C` class that represents a particular LCD attached to your Arduino board.

- `lcd_Addr`: The address of the LCD defaults to 0x27.
- `lcd_cols`: The LCD1602 has 16 columns.
- `lcd_rows`: The LCD1602 has 2 rows.

```
void init()
```

Initialize the lcd.

```
void backlight()
```

Turn the (optional) backlight on.

```
void nobacklight()
```

Turn the (optional) backlight off.

```
void display()
```

Turn the LCD display on.

```
void nodisplay()
```

Turn the LCD display off quickly.

```
void clear()
```

Clear display, set cursor position to zero.

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

Set the cursor position to col,row.

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

Prints text to the LCD.

- **data:** The data to print (char, byte, int, long, or string).
- **BASE (optional):** The base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

### 5.11.2 IR Receiver

In this project, you will learn to use IR Receiver.

An infrared-receiver is a component which receives infrared signals and can independently receive infrared rays and output signals compatible with TTL level. It is similar with a normal plastic-packaged transistor in size and is suitable for all kinds of infrared remote control and infrared transmission.

#### Required Components

In this project, we need the following components.

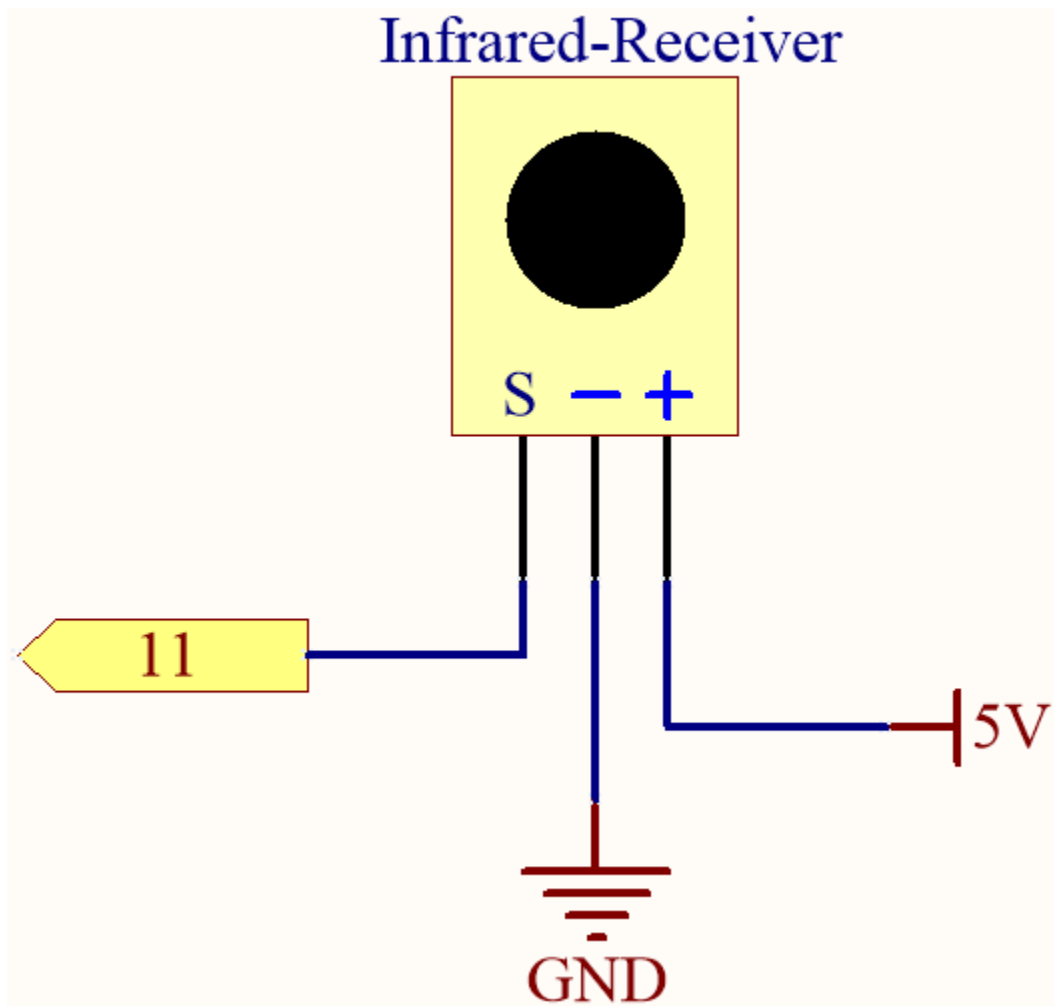
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

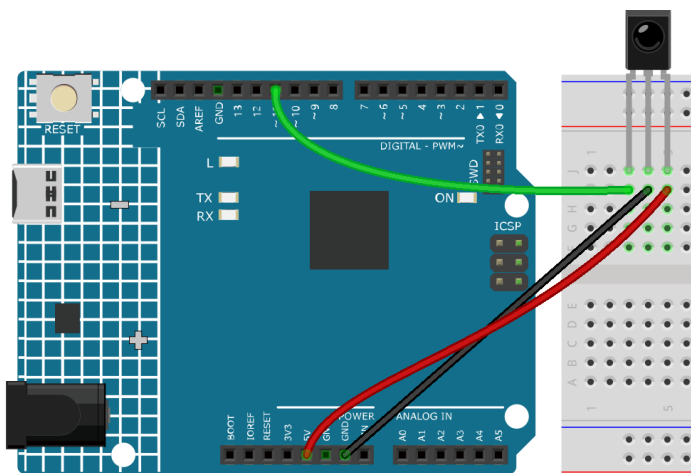
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>IR Receiver</i>	-

#### Schematic



### Wiring

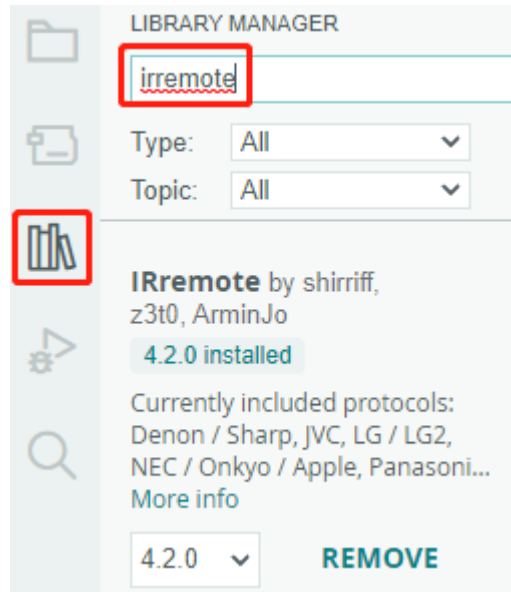
In this example, we wire up the left pin of IR Receiver to pin 11, the middle pin to GND, and the right pin to 5V.



### Code

**Note:**

- Open the 5.11.ir\_receiver.ino file under the path of 3in1-kit\learning\_project\5.11.ir\_receiver.
- Or copy this code into **Arduino IDE**.
- The IRremote library is used here, you can install it from the **Library Manager**.



After uploading the codes to the R4 board, you can see that the current value of the pressed button of IR Remote Controller displays on the serial monitor.

**How it works?**

This code is designed to work with an infrared (IR) remote control using the IRremote library. Here's the breakdown:

1. Include Libraries: This includes the IRremote library, which provides functions to work with IR remote controls.

```
#include <IRremote.h>
```

2. Defines the Arduino pin to which the IR sensor's signal pin is connected.

```
const int IR_RECEIVE_PIN = 11; // Define the pin number for the IR Sensor
```

3. Initializes serial communication at a baud rate of 9600. Initializes the IR receiver on the specified pin (IR\_RECEIVE\_PIN) and enables LED feedback (if applicable).

```
void setup() {
  Serial.begin(9600); // Start serial
  // communication at 9600 baud rate
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK); // Start the IR
  // receiver
}
```

4. The loop runs continuously to process incoming IR remote signals.



```

void loop() {
  if (IrReceiver.decode()) {
    String decodedValue = decodeKeyValue(IrReceiver.decodedIRData.
    ←command);
    if (decodedValue != "ERROR") {
      Serial.println(decodedValue);
      delay(100);
    }
    IrReceiver.resume(); // Enable receiving of the next value
  }
}

```

- Checks if an IR signal is received and successfully decoded.
- Decodes the IR command and stores it in decodedValue using a custom decodeKeyValue() function.
- Checks if the decoded value is not an error.
- Prints the decoded IR value to the serial monitor.
- Resumes IR signal reception for the next signal.

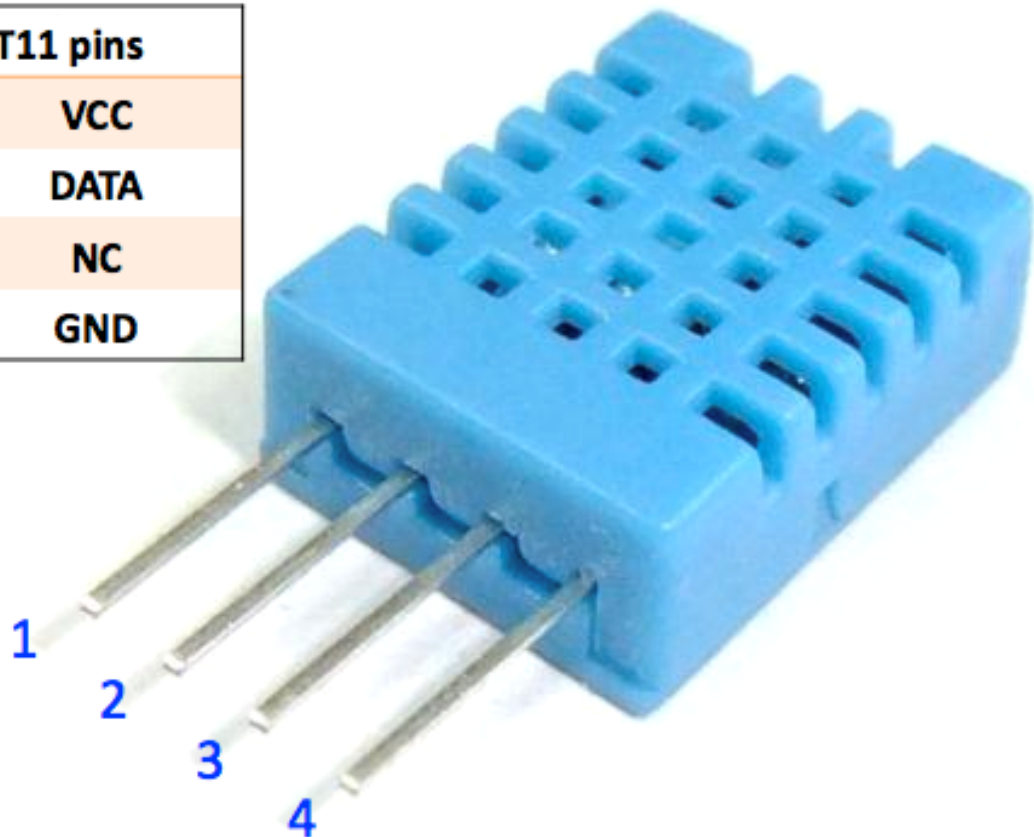
### 5.11.3 Temperature - Humidity

Humidity and temperature are closely related from the physical quantity itself to the actual people's life. The temperature and humidity of human environment will directly affect the thermoregulatory function and heat transfer effect of human body. It will further affect the thinking activity and mental state, thus affecting the efficiency of our study and work.

Temperature is one of the seven basic physical quantities in the International System of Units, which is used to measure the degree of hot and cold of an object. Celsius is one of the more widely used temperature scales in the world, expressed by the symbol "°C".

Humidity is the concentration of water vapor present in the air. The relative humidity of air is commonly used in life and is expressed in %RH. Relative humidity is closely related to temperature. For a certain volume of sealed gas, the higher the temperature, the lower the relative humidity, and the lower the temperature, the higher the relative humidity.

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



The dht11, a digital temperature and humidity sensor, is provided in this kit. It uses a capacitive humidity sensor and thermistor to measure the surrounding air and outputs a digital signal on the data pin.

### Required Components

In this project, we need the following components.

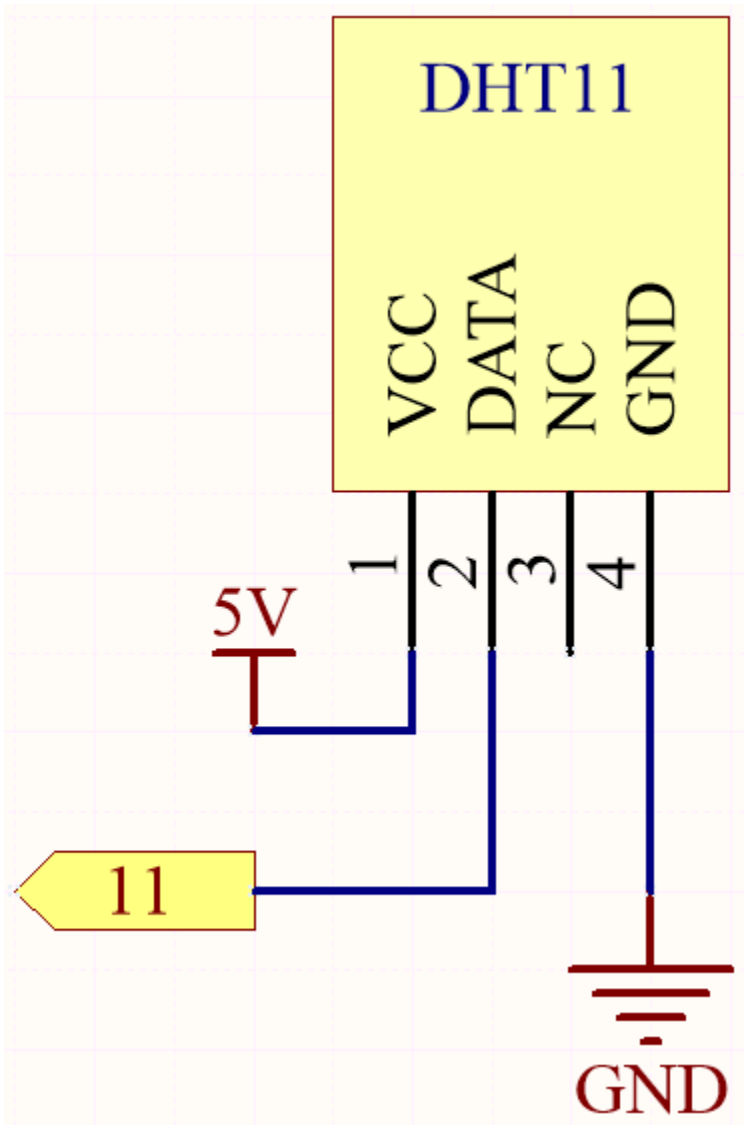
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

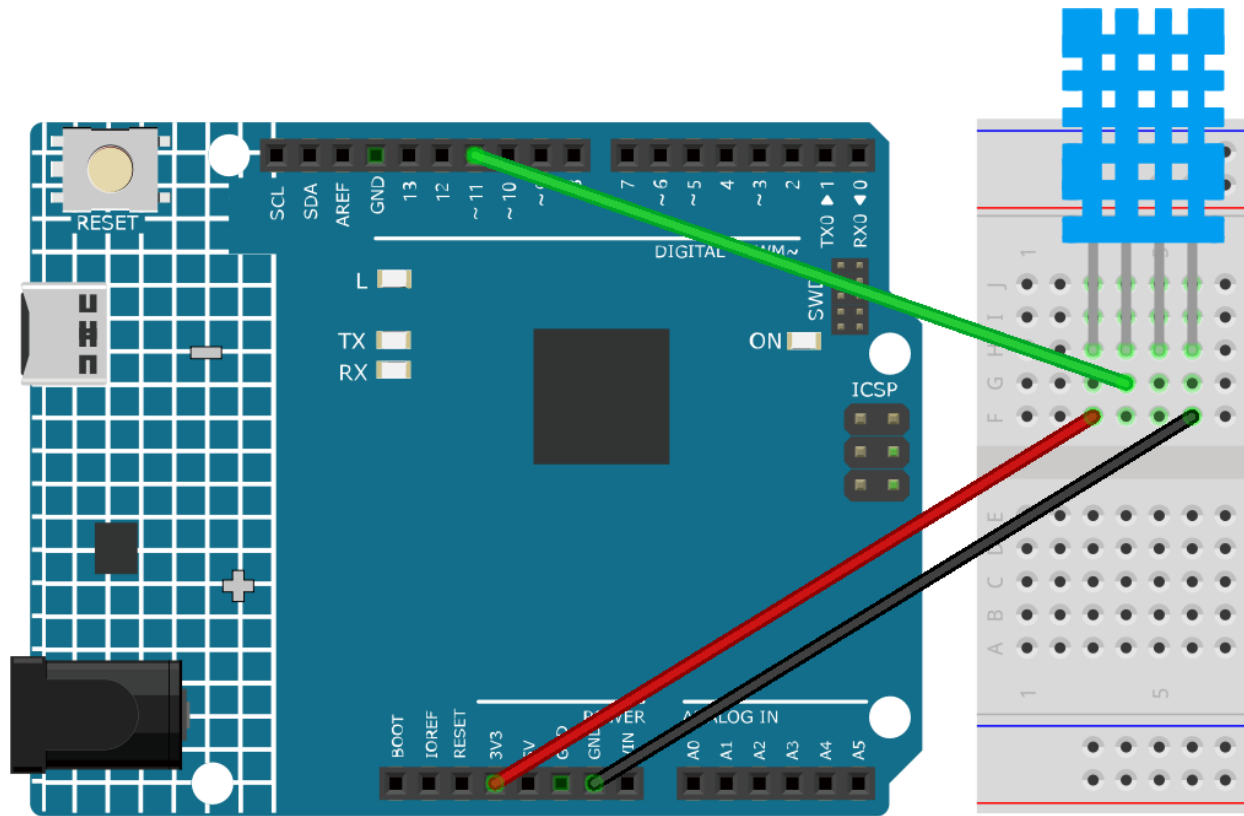
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>DHT11 Humiture Sensor</i>	-

### Schematic



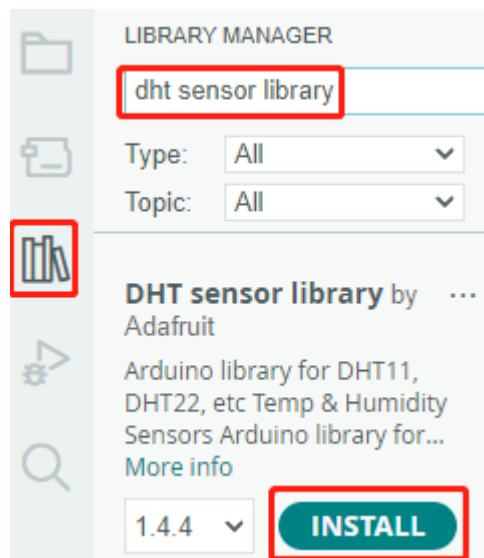
Wiring



## Code

### Note:

- Open the 5.11.temperature\_humidity.ino file under the path of 3in1-kit\learning\_project\5.11.temperature\_humidity.
- Or copy this code into **Arduino IDE**.
- The DHT sensor library is used here, you can install it from the **Library Manager**.



After the code is uploaded successfully, you will see the Serial Monitor continuously print out the temperature and humidity, and as the program runs steadily, these two values will become more and more accurate.

### How it works?

1. Includes the DHT.h library, which provides functions to interact with the DHT sensors. Then, set the pin and type for the DHT sensor.

```
#include "DHT.h"

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

DHT dht(DHTPIN, DHTTYPE);
```

2. Initializes serial communication at a baud rate of 115200 and initializes the DHT sensor.

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

3. In the loop() function, read temperature and humidity values from the DHT11 sensor, and print them to the serial monitor.

```
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");
}
```

- The `dht.readHumidity()` function is called to read the humidity value from the DHT sensor.
- The `dht.readTemperature()` function is called to read the temperature value from the DHT sensor.

- The `isnan()` function is used to check if the readings are valid. If either the humidity or temperature value is NaN (not a number), it indicates a failed reading from the sensor, and an error message is printed.

## 4.5.12 5.12 Serial Read

You may have noticed this when using the `Serial.print()` function. Since there is printing, is there reading? What is the text box on the Serial Monitor used for? Yes, you guessed it, there are ways to control programs and circuits by entering information through the text box on the Serial Monitor.

In this project, we will use the I2C LCD1602 to display the text entered in the Serial Monitor in order to experience the usage of `Serial.read()`.

### Required Components

In this project, we need the following components.

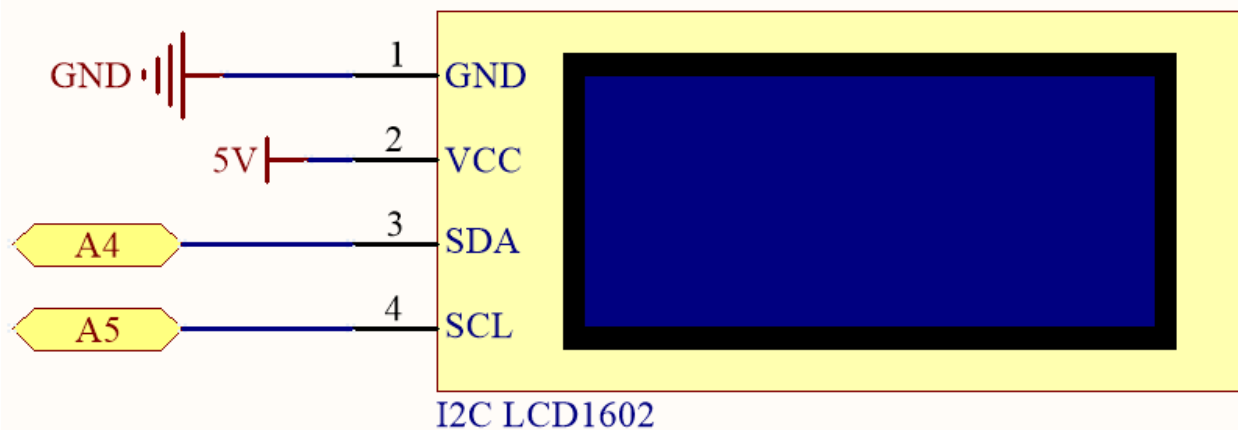
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

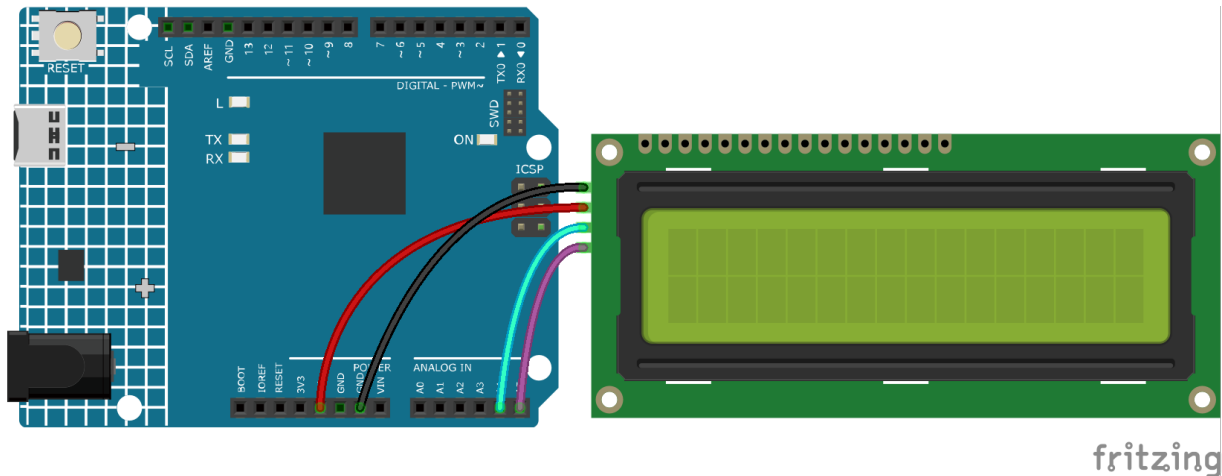
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Jumper Wires</i>	
<i>I2C LCD1602</i>	

### Schematic



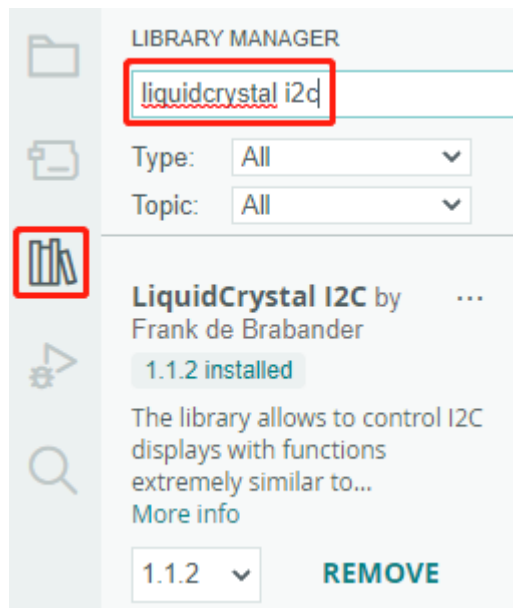
### Wiring



## Code

### Note:

- Open the 5.12.serial\_read.ino file under the path of 3in1-kit\learning\_project\5.12.serial\_read.
- Or copy this code into **Arduino IDE**.
- The LiquidCrystal I2C library is used here, you can install it from the **Library Manager**.



After the code is uploaded successfully, you can enter text in the text box on the serial monitor, and the LCD will display the information.

### How it works?

```
void loop()
{
  // when characters arrive over the serial port...
```

(continues on next page)

(continued from previous page)

```
if (Serial.available()) {  
    // wait a bit for the entire message to arrive  
    delay(100);  
    // clear the screen  
    lcd.clear();  
    // read all the available characters  
    while (Serial.available() > 0) {  
        // display each character to the LCD  
        lcd.write(Serial.read());  
    }  
}
```

- `Serial.available()` can get the number of characters available in the incoming stream when you type something from the textbox. Since there are two terminators in the input, you actually have 3 characters when you type A, and 4 characters when you type AB.
- `Serial.read()` will take the first character from the incoming stream. For example, if you typed AB, calling `Serial.read()` only once, will get the character A; The second call, you will get B; the third and fourth call, you will get two end symbols; calling this function when the input stream has no characters available will result in an error.

To sum up, it is common to combine the above two, using a `while` loop to read all characters entered each time.

```
while (Serial.available() > 0) {  
    Serial.print(Serial.read());  
}
```

By the way, if you don't use `Serial.read()` to get characters from the incoming stream, the characters from the incoming stream will be stacked together. For example, if you type A followed by AB, the incoming stream will accumulate 7 characters.

### 4.5.13 5.13 Interrupt

If you use some `delay()` in a project that uses sensors, you may find that when you trigger these sensors, the program may have no effect. This is because the delay statement will cause the program to suspend, and the program will not be able to obtain the signal sent by the sensor to the main control board.

In this case, interrupt can be used. Interrupt allows the program not to miss a pulse.

In this chapter, we use the active buzzer and buttons to experience the process of using interrupt.

In the `loop()` function, `delay(1000)` is used to count seconds. Put the button to control the buzzer into the ISR, so that it will not be disturbed by the delay and complete the task smoothly.

---

**Note:** ISRs are special kinds of functions that have some unique limitations most other functions do not have. An ISR cannot have any parameters, and they shouldn't return anything. Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be executed after the current one finishes in an order that depends on the priority they have.

---

#### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

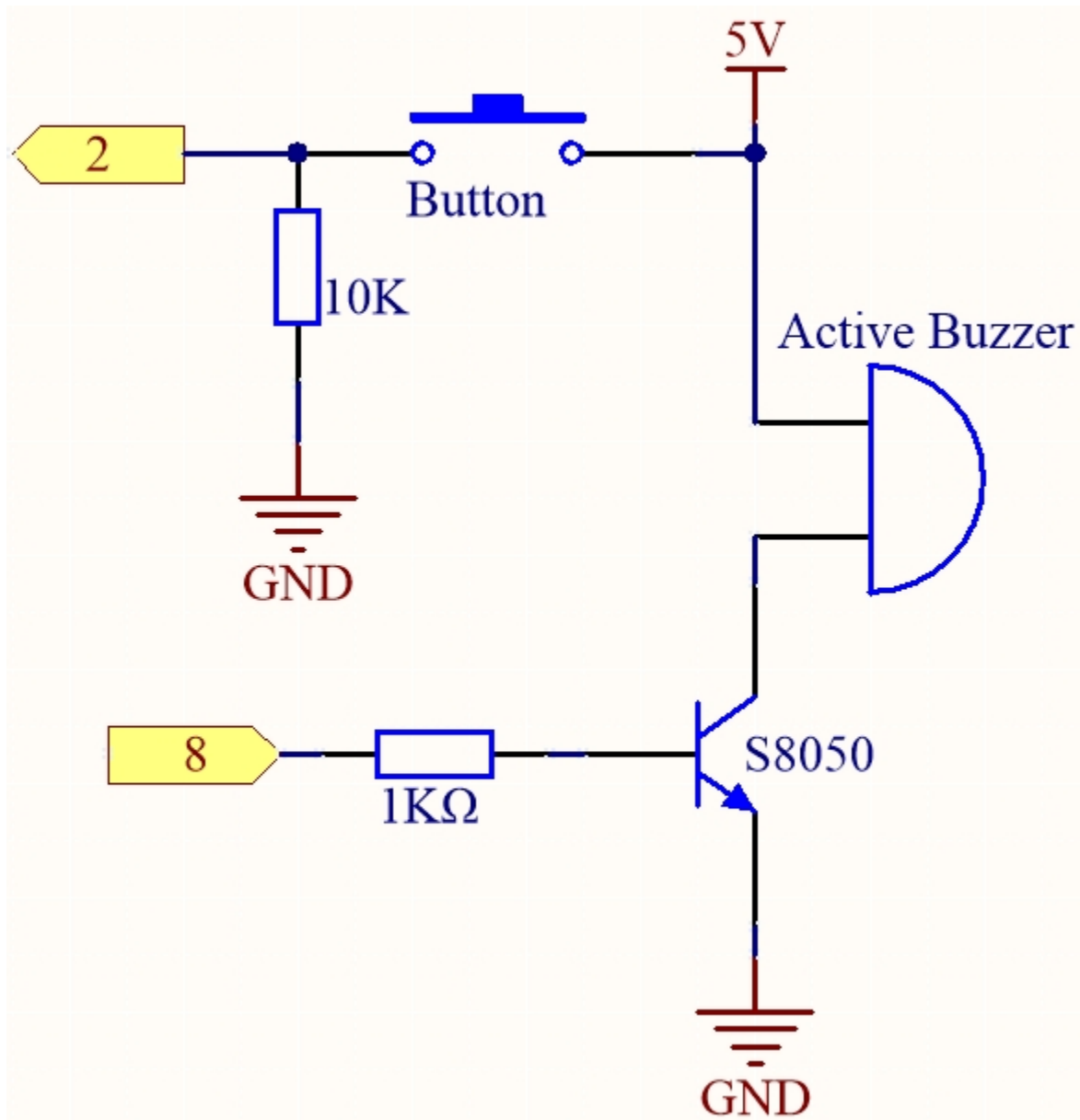


Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

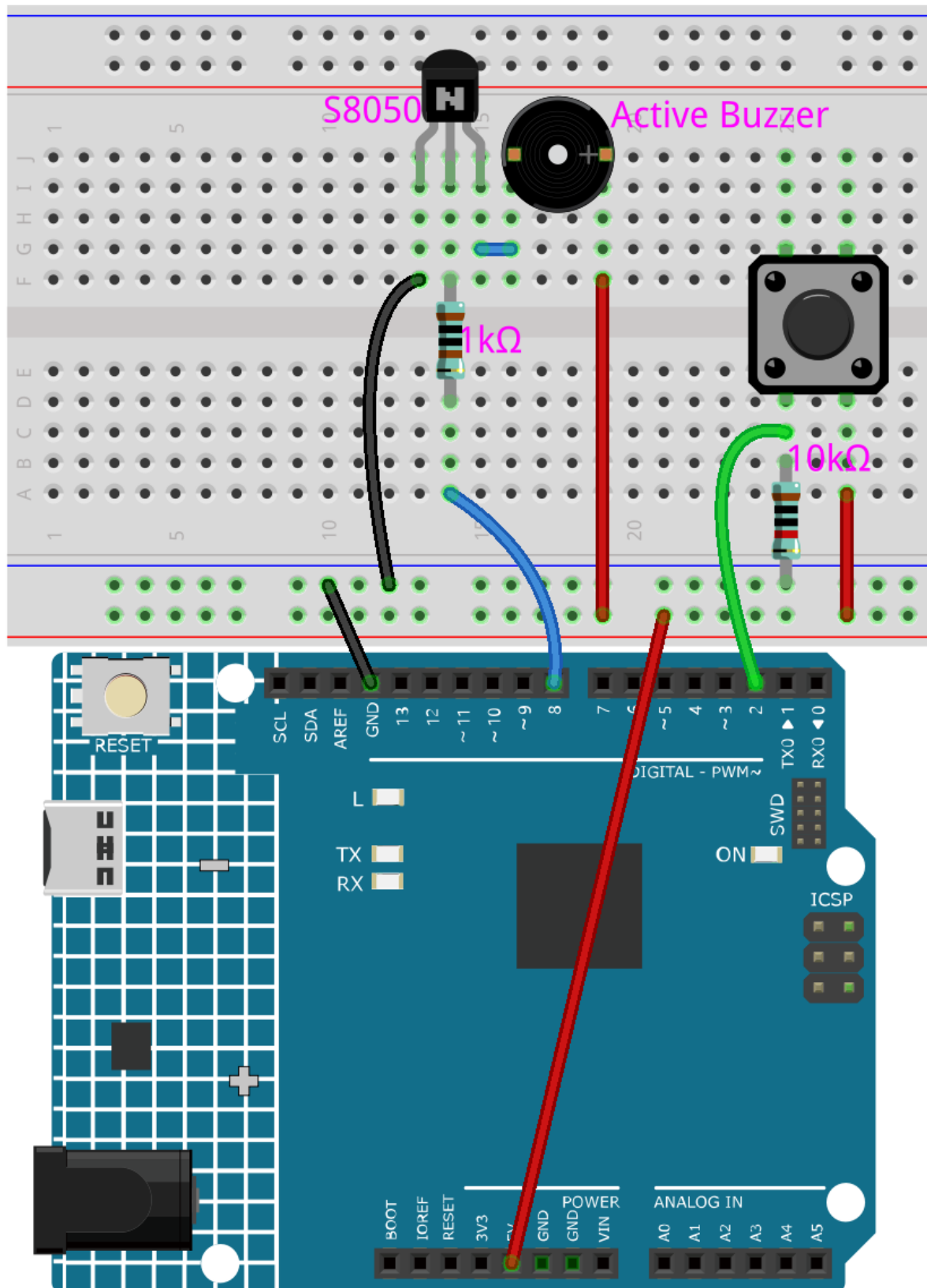
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Button</i>	
<i>Buzzer</i>	-

## Schematic



Wiring



Code

---

### Note:

- Open the `5.13.interrupt.ino` file under the path of `3in1-kit\learning_project\5.13.interrupt`.
  - Or copy this code into **Arduino IDE**.
- 

After the code is successfully uploaded, turn on the Serial Monitor and you will see an auto-incrementing number printed out every second. If you press the button, the buzzer will sound. The button-controlled buzzer function and the timing function do not conflict with each other.

### How it works?

- `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`: Add an interrupt.

#### Syntax

`attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`

#### Parameters

- `pin`: the Arduino pin number. You should use `digitalPinToInterrupt(pin)` to convert the actual digital pin to a specific interrupt number. For example, if you connect to pin 3, use its `digitalPinToInterrupt(3)` as the first parameter.
- `ISR`: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.
- `mode`: defines when the interrupt should be triggered. Four constants are predefined as valid values:
  - \* `LOW` to trigger the interrupt whenever the pin is low,
  - \* `CHANGE` to trigger the interrupt whenever the pin changes value.
  - \* `RISING` to trigger when the pin goes from low to high.
  - \* `FALLING` for when the pin goes from high to low.

---

**Note:** Different main control boards can use interrupt pins differently. On your board, only pin 2 and pin 3 can use interrupt.

---

## 4.5.14 5.14 Calibration

When you use analog input components, such as photoresistors, soil moisture sensors, etc., you may find that their reading range is not 0 to 1023, but rather a range like 0 to 800 or 600 to 1000, because it is impossible to reach the limits of these devices with normal use.

In this case, a technique for calibrating the sensor inputs can be used. During startup, have the control board measure the sensor readings for five seconds and record the highest and lowest readings. This five-second reading defines the minimum and maximum expected values of the readings taken during the cycle.

In this project, we use a photoresistor and a passive buzzer to implement a [theremin](#) -like game using the calibration technique described above.

---

**Note:** The [theremin](#) is an electronic musical instrument that requires no physical contact. It generates different tones by sensing the position of the player's hands.

---

### Required Components

In this project, we need the following components.

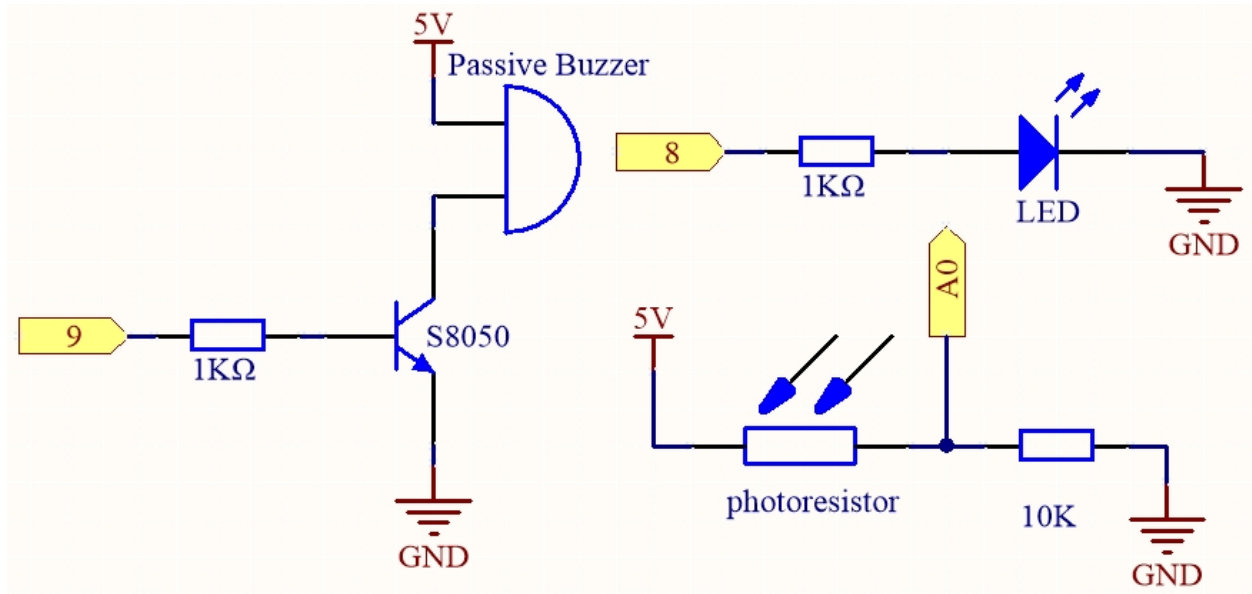
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Buzzer</i>	
<i>LED</i>	
<i>Photoresistor</i>	

### Schematic



### Wiring



**Note:**

- Open the 5.14.calibration.ino file under the path of 3in1-kit\learning\_project\5.14.calibration.
- Or copy this code into **Arduino IDE**.

After the code is uploaded successfully, the LED will light up, and we will have 5 seconds to calibrate the detection range of the photoresistor. This is because we may be in a different light environment each time we use it (e.g. the light intensity is different between midday and dusk).

At this time, we need to swing our hands up and down on top of the photoresistor, and the movement range of the hand will be calibrated to the playing range of this instrument.

After 5 seconds, the LED will go out and we can wave our hands on the photoresistor to play.

**How it works?**

1. Set the initial values and pins of all components.

```
const int buzzerPin = 9;
const int ledPin = 8;
const int photocellPin = A0; //photoresistor attach to A2

int lightLow = 1023;
int lightHigh = 0;

int sensorValue = 0; // value read from the sensor
int pitch = 0; // sensor value converted into LED 'bars'

unsigned long previousMillis = 0;
const long interval = 5000;
```

2. Set up a calibration process in setup().

```
void setup()
{
  pinMode(buzzerPin, OUTPUT); // make buzzer output
  pinMode(ledPin, OUTPUT); // make the LED pin output

  /* calibrate the photoresistor max & min values */
  previousMillis = millis();
  digitalWrite(ledPin, HIGH);
  while (millis() - previousMillis <= interval) {
    sensorValue = analogRead(photocellPin);
    if (sensorValue > lightHigh) {
      lightHigh = sensorValue;
    }
    if (sensorValue < lightLow) {
      lightLow = sensorValue;
    }
  }
  digitalWrite(ledPin, LOW);
}
```

The work flow is as follows.

- using `millis()` for timing with an interval of 5000ms.

```
previousMillis = millis();
...
while (millis() - previousMillis <= interval) {
...
}
```

- During these five seconds, wave a hand around the photoresistor, the maximum and minimum values of the detected light are recorded and assigned to `lightHigh` and `lightLow` respectively.

```
sensorValue = analogRead(photocellPin);
if (sensorValue > lightHigh) {
    lightHigh = sensorValue;
}
if (sensorValue < lightLow) {
    lightLow = sensorValue;
}
```

- Now you can start playing this Thermin. Read the value of the photoresistor to `sensorValue` and map it from the small range to the large range to be used as the frequency of the buzzer.

```
void loop()
{
    /* play*/
    sensorValue = analogRead(photocellPin); //read the value of A0
    pitch = map(sensorValue, lightLow, lightHigh, 50, 6000); // map to the
    ↪ buzzer frequency
    if (pitch > 50) {
        tone(buzzerPin, pitch, 20);
    }
    delay(10);
}
```

#### 4.5.15 5.15 EEPROM

**EEPROM** is a memory, so the data it stores will not be erased when the main control board is turned off. You can use it to record some data and read it the next time you turn it on.

As an example, you can make a sports counter that keeps track of how many rope skippings you do every day.

You can also write data to it in one program and read it in another. For example, when you are working on a car project, the speeds of the two motors are inconsistent. You can write a calibration program to record the compensation value of the motor speed.

#### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

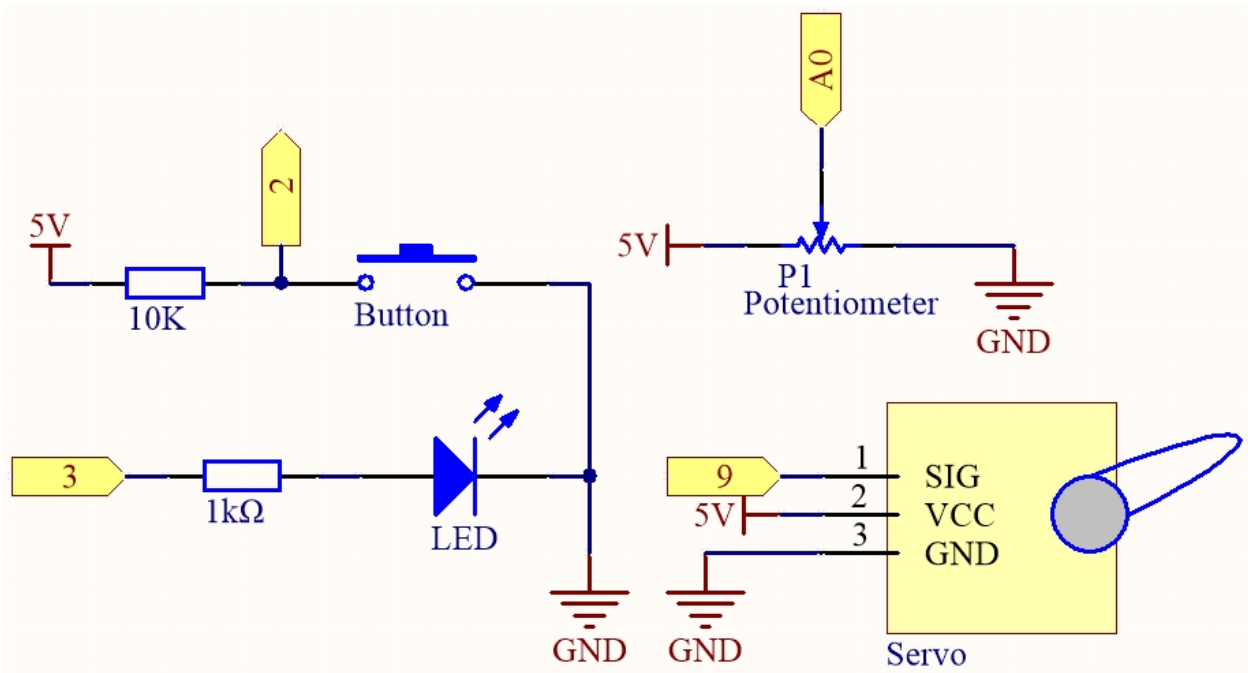
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	



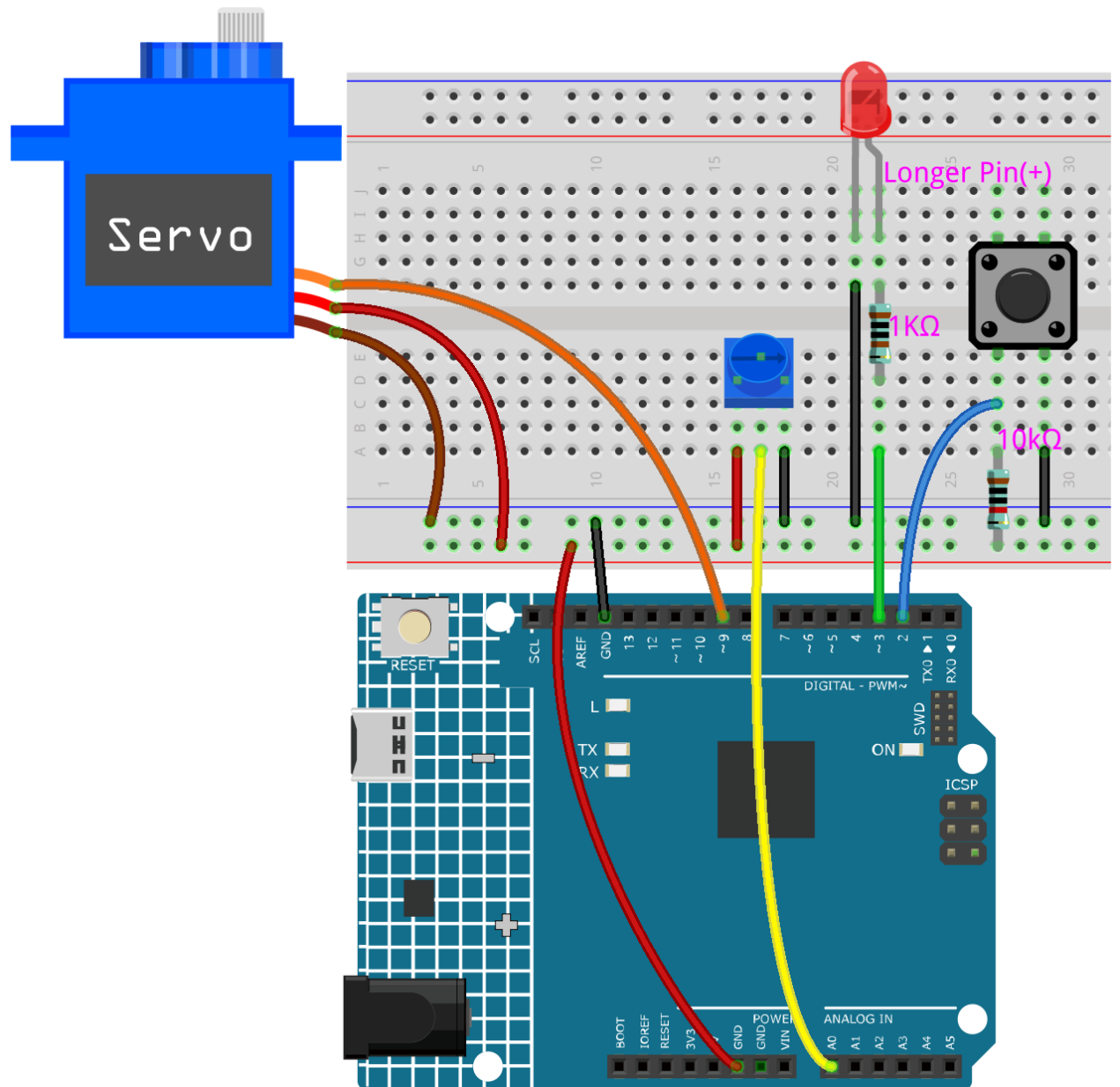
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Servo</i>	
<i>Button</i>	
<i>Potentiometer</i>	

### Schematic



### Wiring



## Code

**Note:**

- Open the 5.15.eeprom.ino file under the path of 3in1-kit\learning\_project\5.15.eeprom.
- Or copy this code into **Arduino IDE**.

To use this circuit, you simply press the button to begin recording and input the desired information through a potentiometer. Now, the board will repeat your actions endlessly (and it blinks an led each iteration) until you press the button again to record new actions. You may also vary the amount of time recorded by changing the values of resolution and recordTime.

## How it works?

1. Import the `EEPROM.h` library, and initialize the EEPROM memory.

```

...
#include <EEPROM.h> //used to store recorded values

...
float resolution = 1000; //MUST be less than EEPROM.length()
float recordTime = 5; //delay time
bool recording = false;
...

```

Please note that /MUST be less than EEPROM.length(), in setup() it will print the memory of your board's EEPROM, which should be 1024 for your board. If you are using a different board, you can change the value of the variable resolution.

2. Print the EEPROM memory of your board.

```

void setup() {
    ...
    Serial.begin(9600);
    //Serial.println(EEPROM.length());
}

```

To find the size of your board's EEPROM memory, uncomment the line Serial.println(EEPROM.read(i)). This will print the size of EEPROM in the serial monitor, and you can change the value of the variable resolution accordingly.

3. As soon as a button press is detected, then recording begins and the required information is entered via a potentiometer. Now the board repeats your action endlessly (and flashes an LED for each repetition) until you press the button again, recording a new action.

```

void loop() {
    if (recording == true) { //record
        for (int i = 1; i <= resolution; i++) {
            digitalWrite(ledPin, HIGH); //light status led
            int val = map(analogRead(A0), 0, 1023, 0, 180);
            EEPROM.write(i, val);
            //Serial.println(EEPROM.read(i));
            myServo.write(val);
            delay(recordTime);
        }
        digitalWrite(ledPin, LOW); //turn off status led
        delay(1000); //give time for person
        recording = false;
    }
    else {
        for (int i = 1; i <= resolution; i++) { //playback
            if (digitalRead(buttonPin) == 0) { //stop playback and record
                ↪ new values
                recording = true;
                break;
            }
            int readval = EEPROM.read(i);
            myServo.write(readval);
            //Serial.println(readval);
            delay(recordTime);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```
    }  
    digitalWrite(ledPin, HIGH); //show a new repeat  
    delay(100);  
    digitalWrite(ledPin, LOW);  
  }  
}
```

- Make the variable `recording` true when the button is pressed.
- When the variable `recording` is true, start recording the action in the memory range.
- Read the value of the potentiometer and map it to 0-180 to store it in EEPROM and control the rotation of the servo.
- The LED lights up at the start of recording and goes off at the end.
- Repeat the recorded action with a quick flash of the LED to remind you of a new repeat.

#### 4. About the EEPROM library.

Here are some of its functions.

- `write(address, value)`: Write a byte to the EEPROM.
  - `address`: the location to write to, starting from 0 (int)
  - `value`: the value to write, from 0 to 255 (byte)
  - An EEPROM write takes 3.3 ms to complete. The EEPROM memory has a specified life of 100,000 write/erase cycles, so you may need to be careful about how often you write to it.
- `Read(address)`: Reads a byte from the EEPROM. Locations that have never been written to have the value of 255.
- `update(address, value)`: Write a byte to the EEPROM. The value is written only if differs from the one already saved at the same address.
  - An EEPROM write takes 3.3 ms to complete. The EEPROM memory has a specified life of 100,000 write/erase cycles, so using this function instead of `write()` can save cycles if the written data does not change often
- `EEPROM.put(address, data)`: Write any data type or object to the EEPROM.
  - `address`: the location to read from, starting from 0 (int).
  - `data`: the data to read, can be a primitive type (eg. float) or a custom struct.
  - This function uses `EEPROM.update()` to perform the write, so does not rewrites the value if it didn't change.
- `EEPROM.get(address, data)`: Read any data type or object from the EEPROM.
  - `address`: the location to read from, starting from 0 (int).
  - `data`: the data to read, can be a primitive type (eg. float) or a custom struct.

## 4.6 6. Funny Project

In this chapter, you will find some examples that illustrate the basic logic of how most programs interact with reality. This will help you become familiar with Arduino programming. When you have a creative idea in mind, programming will no longer be challenging for you.

### 4.6.1 6.1 Light-sensitive Array

A photoresistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and darkness-activated switching circuits.

The resistance of a photoresistor changes with incident light intensity. If the light intensity gets higher, the resistance decreases; if it gets lower, the resistance increases. In this experiment, we will use eight LEDs to show the light intensity. The higher the light intensity is, the more LEDs will light up. When the light intensity is high enough, all the LEDs will be on. When there is no light, all the LEDs will go out.

#### Required Components

In this project, we need the following components.

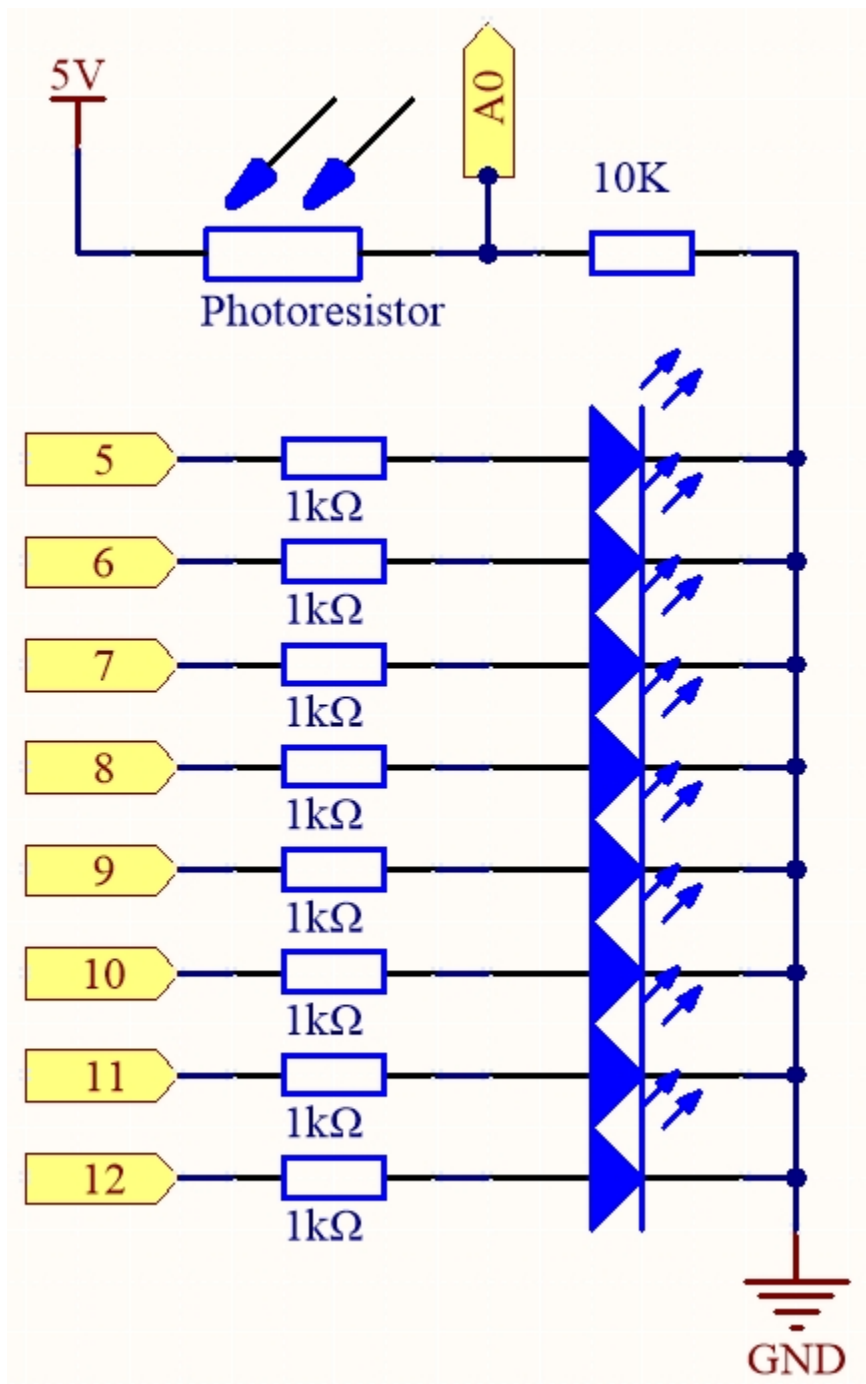
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

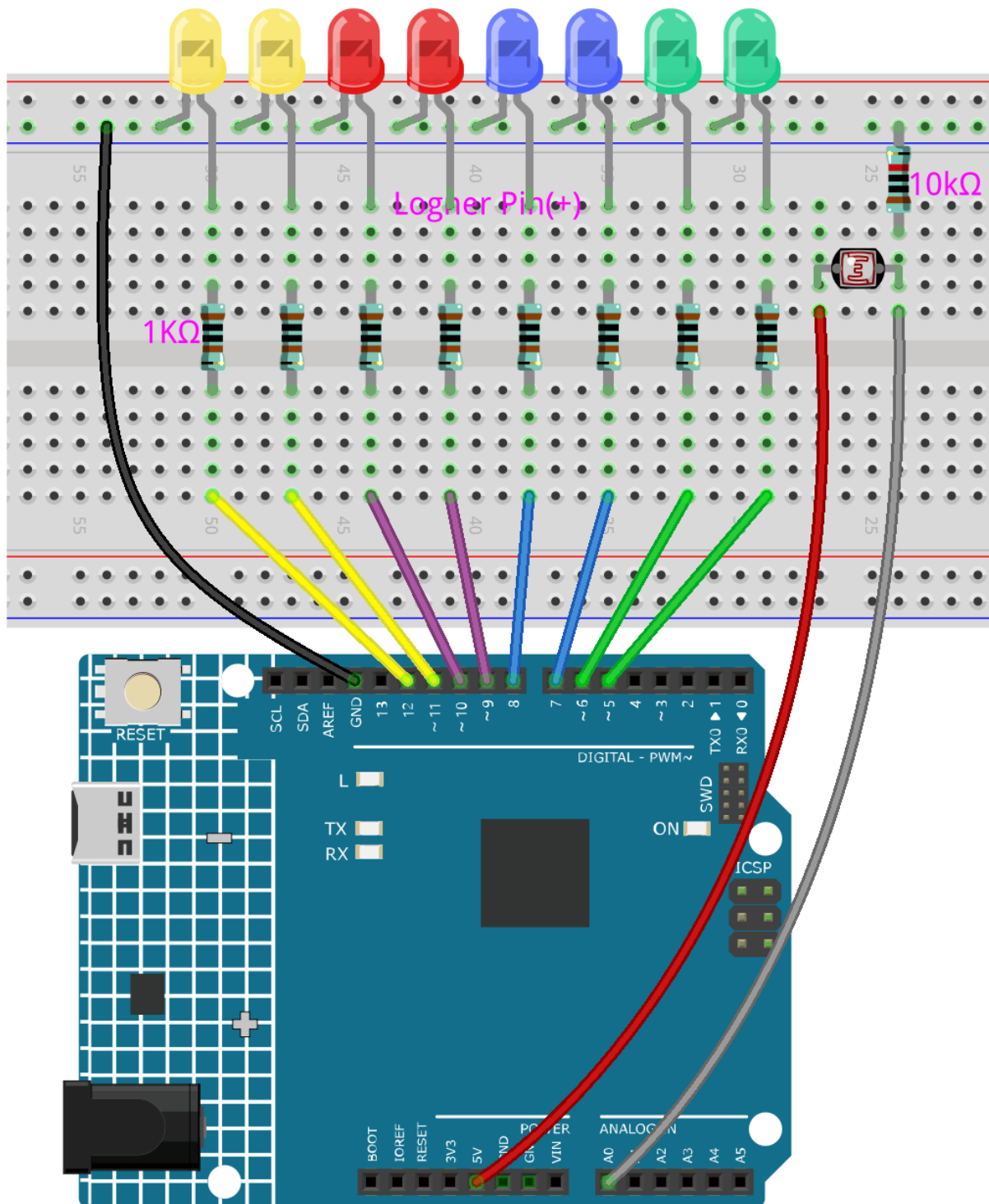
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Photoresistor</i>	

#### Schematic



Wiring



## Code

### Note:

- Open the 6.1.light\_control\_led.ino file under the path of 3in1-kit\learning\_project\6.1.light\_control\_led.
- Or copy this code into **Arduino IDE**.

Now, shine some light on the photoresistor, and you will see several LEDs light up. Shine more light and you will see more LEDs light up. When you place it in a dark environment, all the LEDs will go out.

#### How it works?

```
void loop()
{
    sensorValue = analogRead(photocellPin); //read the value of A0
    ledLevel = map(sensorValue, 300, 1023, 0, NbrLEDs); // map to the number of LEDs
    for (int led = 0; led < NbrLEDs; led++)//
    {
        if (led < ledLevel ) //When led is smaller than ledLevel, run the following code.
        {
            digitalWrite(ledPins[led], HIGH); // turn on pins less than the level
        }
        else
        {
            digitalWrite(ledPins[led],LOW); // turn off pins higher than
        }
    }
}
```

By using the map() function, you can map the photoresistor value to the 8 LEDs, for example, if sensorValue is 560, then ledLevel is 4, so at this point, ledPins[0] to ledPins[4] should be lit, and ledPins[5] to ledPins[7] should be off.

## 4.6.2 6.2 Digital Dice

Here we use button, 7-segment and 74hc595 to make an electronic dice. Each time the button is pressed, a random number ranging from 1 to 6 is generated and displayed on the 7-segment Display.

#### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

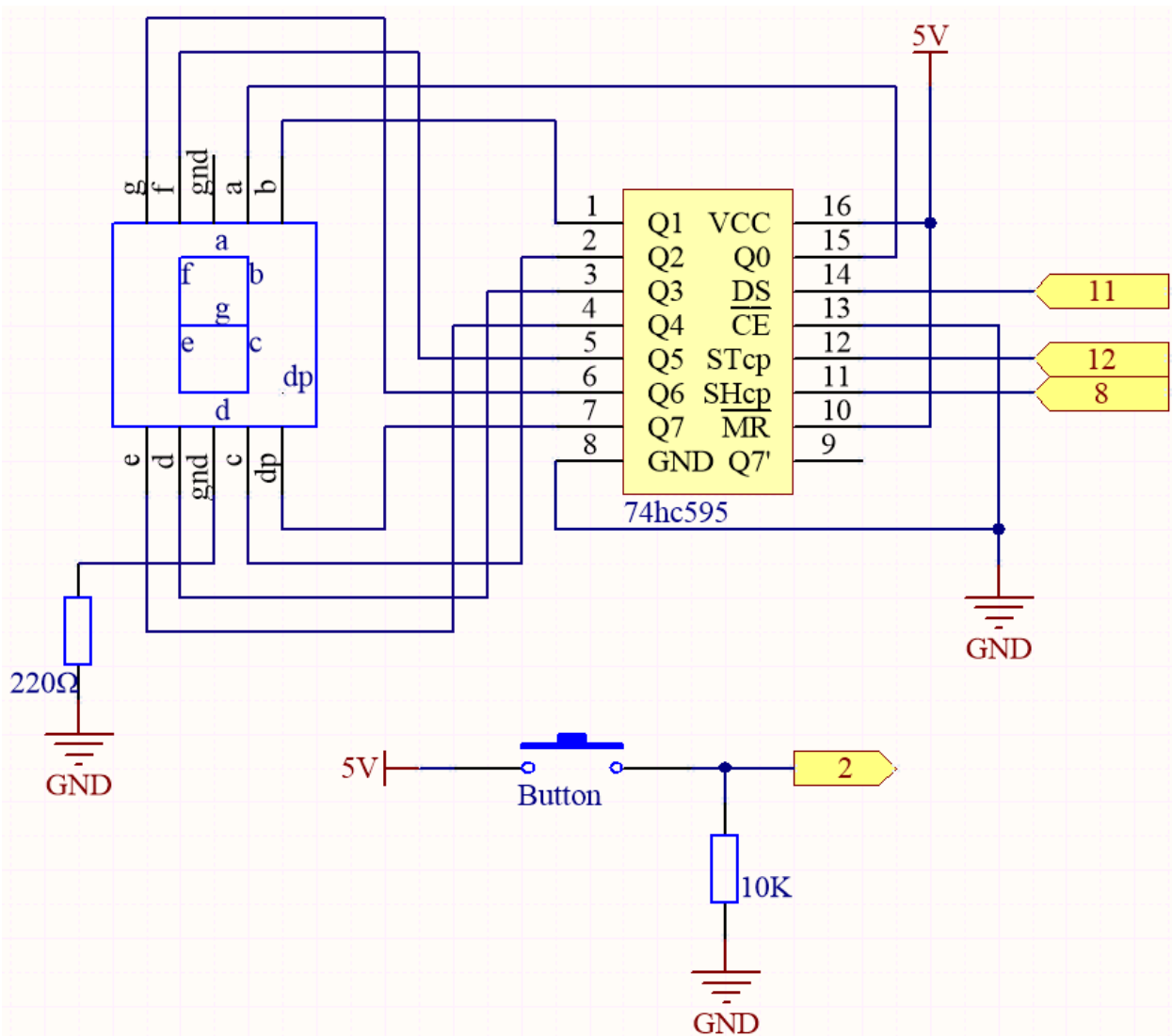
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

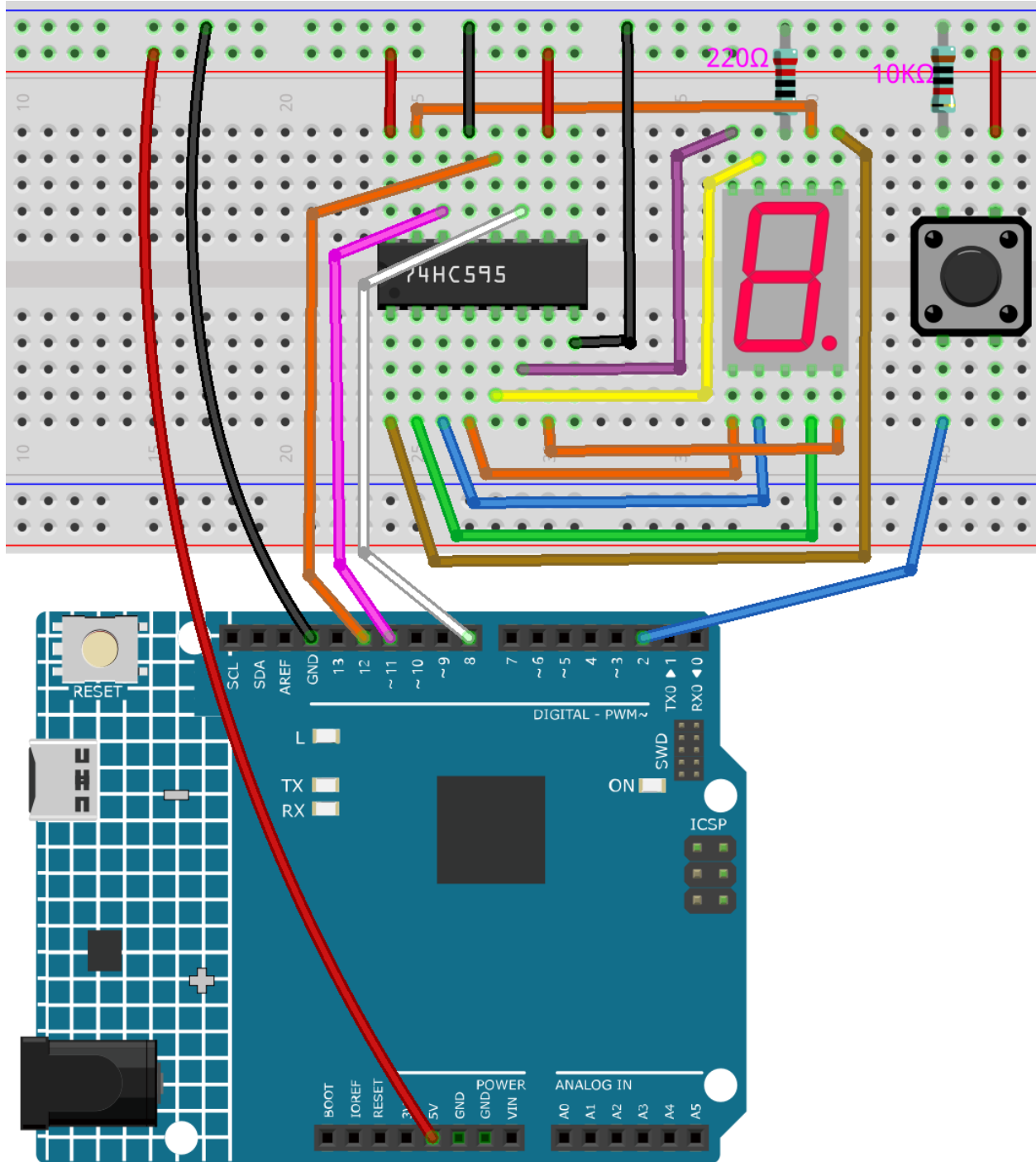


COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Button</i>	
<i>74HC595</i>	
<i>7-segment Display</i>	

### Schematic



### Wiring



## Code

### Note:

- Open the 6.2.electronic\_dice.ino file under the path of 3in1-kit\learning\_project\6.2.electronic\_dice.
- Or copy this code into **Arduino IDE**.

When the code is uploaded successfully, the 7-segment Display will display 0-7 in a fast scroll, and when you press the button, it will display a random number and stop scrolling. The scrolling display starts again when you press the button again.

### How it works?

This project is based on *5.10 ShiftOut(Segment Display)* with a button to start/pause the scrolling display on the 7-segment Display.

1. Initialize each pin and read the value of the button.

```
void setup ()
{
    ...
    attachInterrupt(digitalPinToInterrupt(buttonPin), rollDice, FALLING);
}
```

- The interrupt is used here to read the state of the button. The default value of `buttonPin` is low, which changes from low to high when the button is pressed.
- `rollDice` represents the function to be called when the interrupt is triggered, it is used to toggle the value of the variable `state`.
- `FALLING` means the interrupt is triggered when the `buttonPin` goes from low to high.

2. When the variable `state` is 0, the function `showNumber()` is called to make the 7-segment Display randomly display a number between 1 and 7.

```
void loop()
{
    if (state == 0) {
        showNumber((int)random(1, 7));
        delay(50);
    }
}
```

3. About `rollDice()` function.

```
void rollDice() {
    state = !state;
}
```

When this function is called, it toggles the value of `state`, such as 1 last time and 0 this time.

4. About `showNumber()` function.

```
void showNumber(int num) {
    digitalWrite(STcp, LOW); //ground ST_CP and hold low for as long as you
    //are transmitting
    shiftOut(DS, SHcp, MSBFIRST, dataArray[num]);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(STcp, HIGH); //pull the ST_CPST_CP to save the data
}
```

This is the code inside `loop()` in the project *5.10 ShiftOut(Segment Display)* into the function `showNumber()`.

### 4.6.3 6.3 High Temperature Alarm

Next, we will make a high temperature alarm device using thermistor, push button, potentiometer and LCD. The LCD1602 shows the temperature detected by the thermistor and the high temperature threshold value, which can be adjusted using a potentiometer. The threshold value is stored on EEPROM at the same time, so if the current temperature exceeds the threshold value, the buzzer will sound.

#### Required Components

In this project, we need the following components.

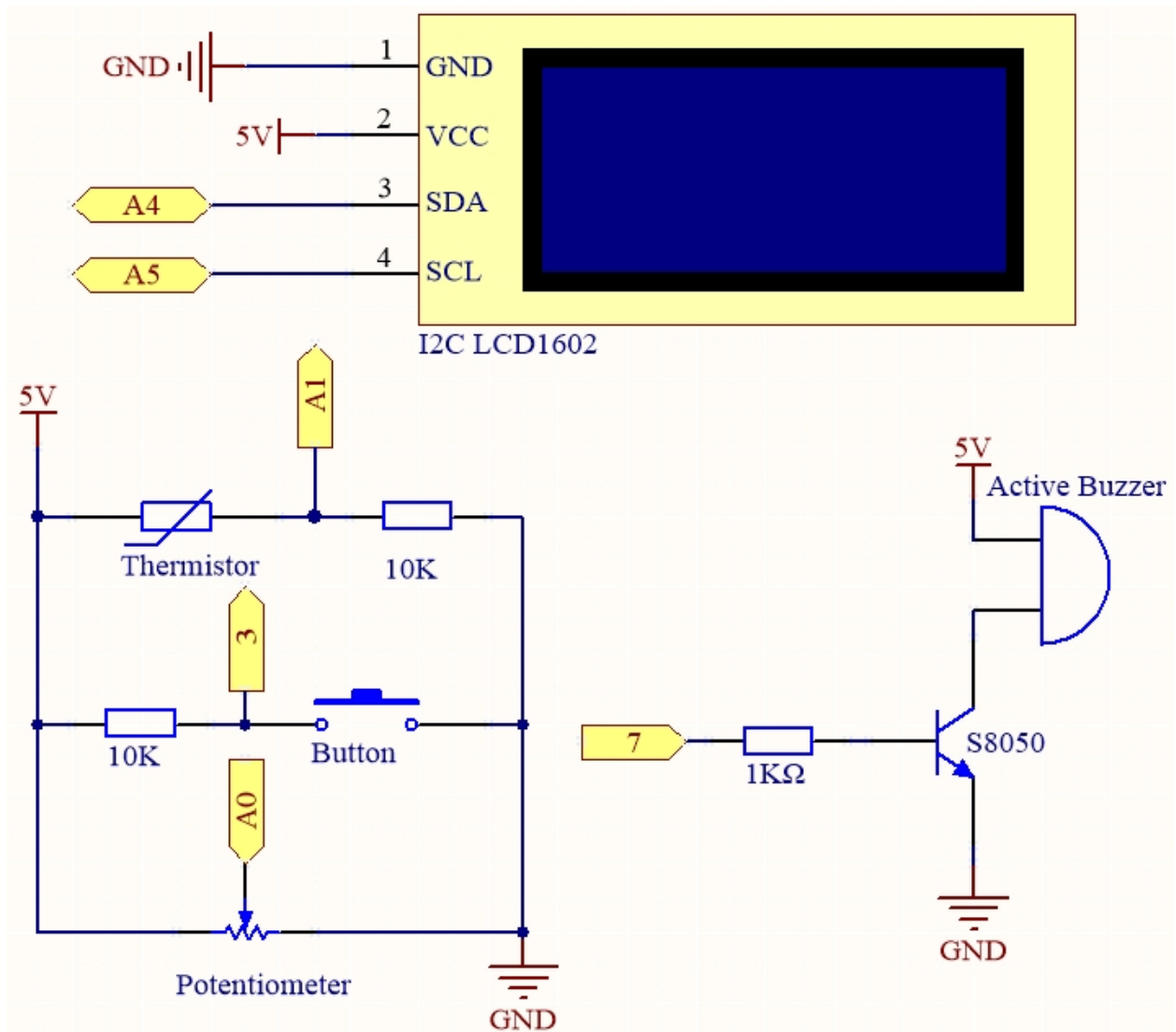
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

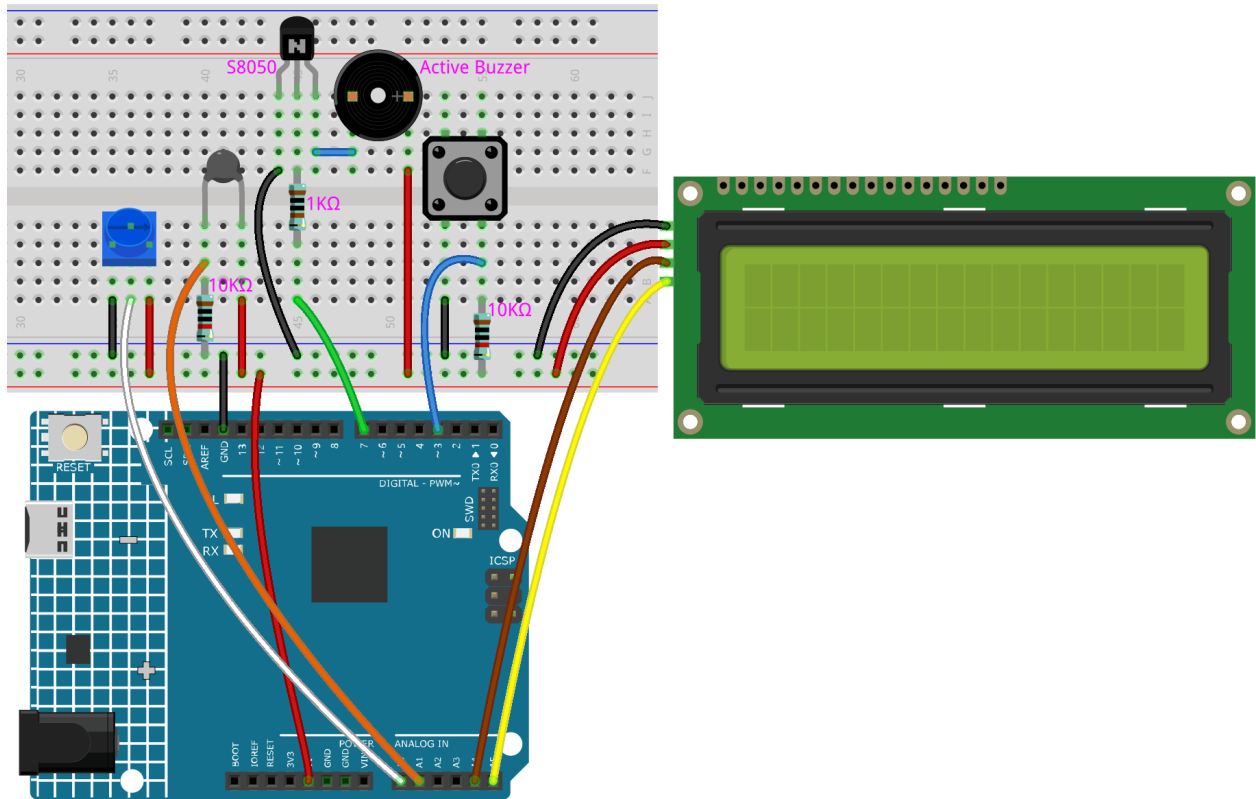
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Buzzer</i>	-
<i>Button</i>	
<i>I2C LCD1602</i>	
<i>Thermistor</i>	
<i>Potentiometer</i>	

#### Schematic



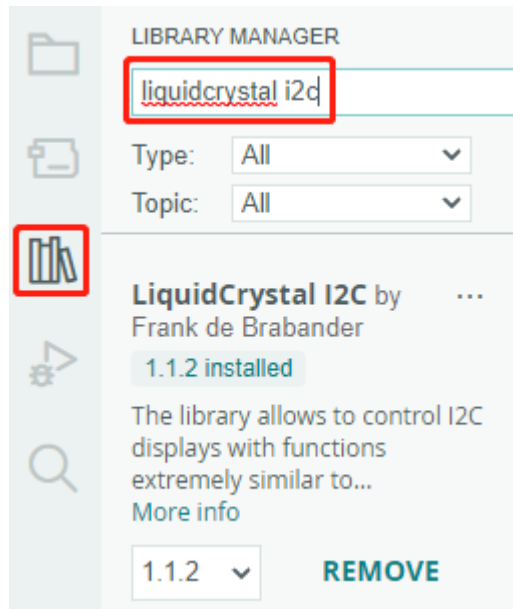
### Wiring



### Code

#### Note:

- You can open the file `6.3.high_tem_alarm.ino` under the path of `3in1-kit\learning_project\6.3.high_tem_alarm` directly.
- Or copy this code into Arduino IDE.
- The LiquidCrystal I2C library is used here, you can install it from the **Library Manager**.



After the code is successfully uploaded, The LCD1602 shows the temperature detected by the thermistor and the high temperature threshold value, which can be adjusted using a potentiometer. The threshold value is stored on EEPROM at the same time, so if the current temperature exceeds the threshold value, the buzzer will sound.

**Note:** If the code and wiring are fine, but the LCD still does not display content, you can turn the potentiometer on the back.

### How it works?

1. Initialize the button, buzzer and I2C LCD1602, and read the EEPROM values. An interrupt is also used here to read the button status.

```
void setup()
{
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  lcd.init();
  lcd.backlight();
  upperTem = EEPROM.read(0);
  delay(1000);
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonState, FALLING);
}
```

- The interrupt is used here to read the button's state. When the button is pressed, buttonPin changes from low to high.
  - The function buttonState is called when the interrupt triggers, and it toggles the value of the variable state.
  - FALLING means the interrupt occurs when buttonPin goes from low to high.
2. To set the high temperature threshold, the function upperTemSetting() is called when state is 1 (state switches between 0 and 1 with button press) in the main program, otherwise monitoringTemp() is called to display the current temperature and the set threshold.

```

void loop()
{
    if (state == 1)
    {
        upperTemSetting();
    }
    else {
        monitoringTemp();
    }
}

```

### 3. About upperTemSetting() function.

```

void upperTemSetting()
{
    int setTem = 0;

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Adjusting...");
    lcd.setCursor(0, 1);
    lcd.print("Upper Tem: ");

    while (1) {
        lcd.setCursor(11, 1);
        setTem = map(analogRead(potPin), 0, 1023, 0, 100);
        lcd.print(setTem);
        if (state == 0)
        {
            EEPROM.write(0, setTem);
            upperTem = setTem;
            lcd.clear();
            return;
        }
    }
}

```

- A threshold can be set with this function. When you enter this function, the LCD1602 displays the current threshold value, which can be modified using the potentiometer. This threshold value will be stored in EEPROM and exited when the button is pressed again.

### 4. About monitoringTemp() function.

```

void monitoringTemp()
{
    long a = analogRead(temPin);
    float tempC = beta / (log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) -
    ↪ 273.0;
    float tempF = 1.8 * tempC + 32.0;
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.print(tempC);
    lcd.print(char(223));
    lcd.print("C  ");
}

```

(continues on next page)



(continued from previous page)

```

lcd.setCursor(0, 1);
lcd.print("Upper: ");
lcd.print(upperTem);
lcd.print(char(223));
lcd.print("C  ");
delay(300);
if (tempC >= upperTem)
{
    digitalWrite(buzzerPin, HIGH);
    delay(50);
    digitalWrite(buzzerPin, LOW);
    delay(10);
}
else
{
    digitalWrite(buzzerPin, LOW);
}
}

```

- Using this function, you can display temperature and set an alarm.
- The thermistor value is read and then converted to Celsius temperature by the formula and displayed on the LCD1602.
- The set threshold is also displayed on the LCD.
- If the current temperature is greater than the threshold, the buzzer will sound an alarm.

#### 4.6.4 6.4 Reversing Aid

With the development of science and technology, a lot of high-tech products have been installed in cars, among which the reversing assist system is one of them. Here we use ultrasonic module, LCD, LED and buzzer to make a simple ultrasonic reversing assist system.

##### Required Components

In this project, we need the following components.

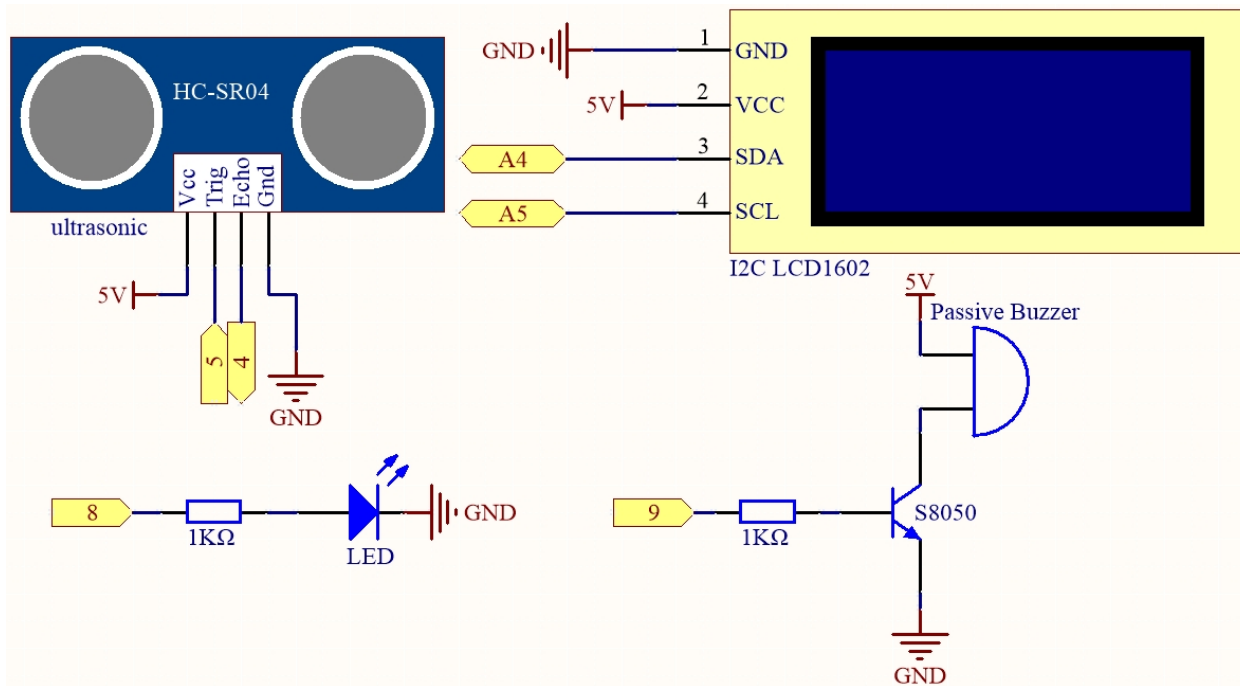
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

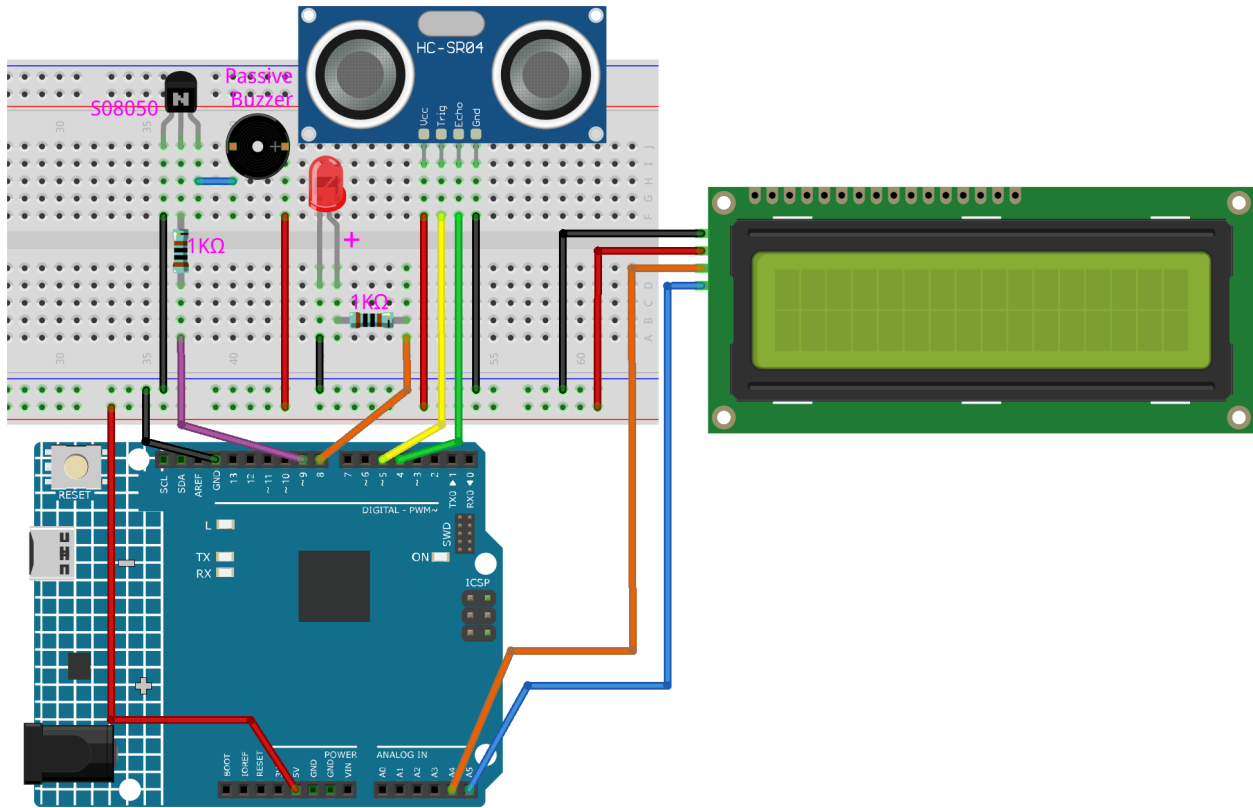
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Buzzer</i>	
<i>I2C LCD1602</i>	
<i>Ultrasonic Module</i>	

### Schematic



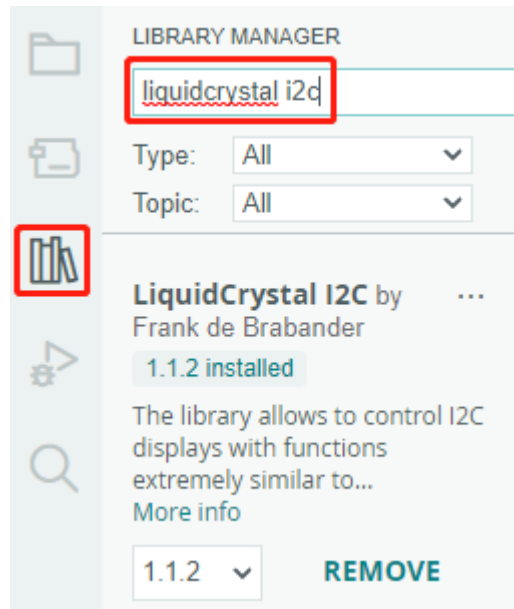
### Wiring



## Code

### Note:

- You can open the file 6.4\_reversingAid.ino under the path of 3in1-kit\learning\_project\6.4\_reversingAid directly.
- Or copy this code into Arduino IDE.
- The LiquidCrystal I2C library is used here, you can install it from the **Library Manager**.



After the code is successfully uploaded, the current detected distance will be displayed on the LCD. Then the buzzer will change the sounding frequency according to different distances.

**Note:** If the code and wiring are fine, but the LCD still does not display content, you can turn the potentiometer on the back.

### How it works?

This code helps us create a simple distance measuring device that can measure the distance between objects and provide feedback through an LCD display and a buzzer.

The `loop()` function contains the main logic of the program and runs continuously. Let's take a closer look at the `loop()` function.

#### 1. Loop to read distance and update parameters

In the loop, the code first reads the distance measured by the ultrasonic module and updates the interval parameter based on the distance.

```
// 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. Check if it's time to beep

The code calculates the difference between the current time and the previous beep time, and if the difference is greater than or equal to the interval time, it triggers the buzzer and updates the previous beep time.

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

### 3. Update LCD display

The code clears the LCD display and then displays “Dis:” and the current distance in centimeters on the first line.

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

delay(100);
```

## 4.6.5 6.5 Reaction Game

Our body has many reaction times, such as audio RT, visual RT, touch RT, etc.

Reaction times have many effects on our daily life, for example, slower than normal reaction times when driving can lead to serious consequences.

In this project, we use 3 buttons and 2 LEDs to measure our visual reaction time.

The serial monitor of the Arduino displays the message “waiting...” After pressing the Ready button, one of the two LEDs must light up randomly after a random time interval. It is important that the testee pushes the corresponding button as soon as possible. The Arduino records the time difference between when the LED lights up and when the person presses the corresponding button, and prints the measured response time on the Arduino serial monitor.

### Required Components

In this project, we need the following components.

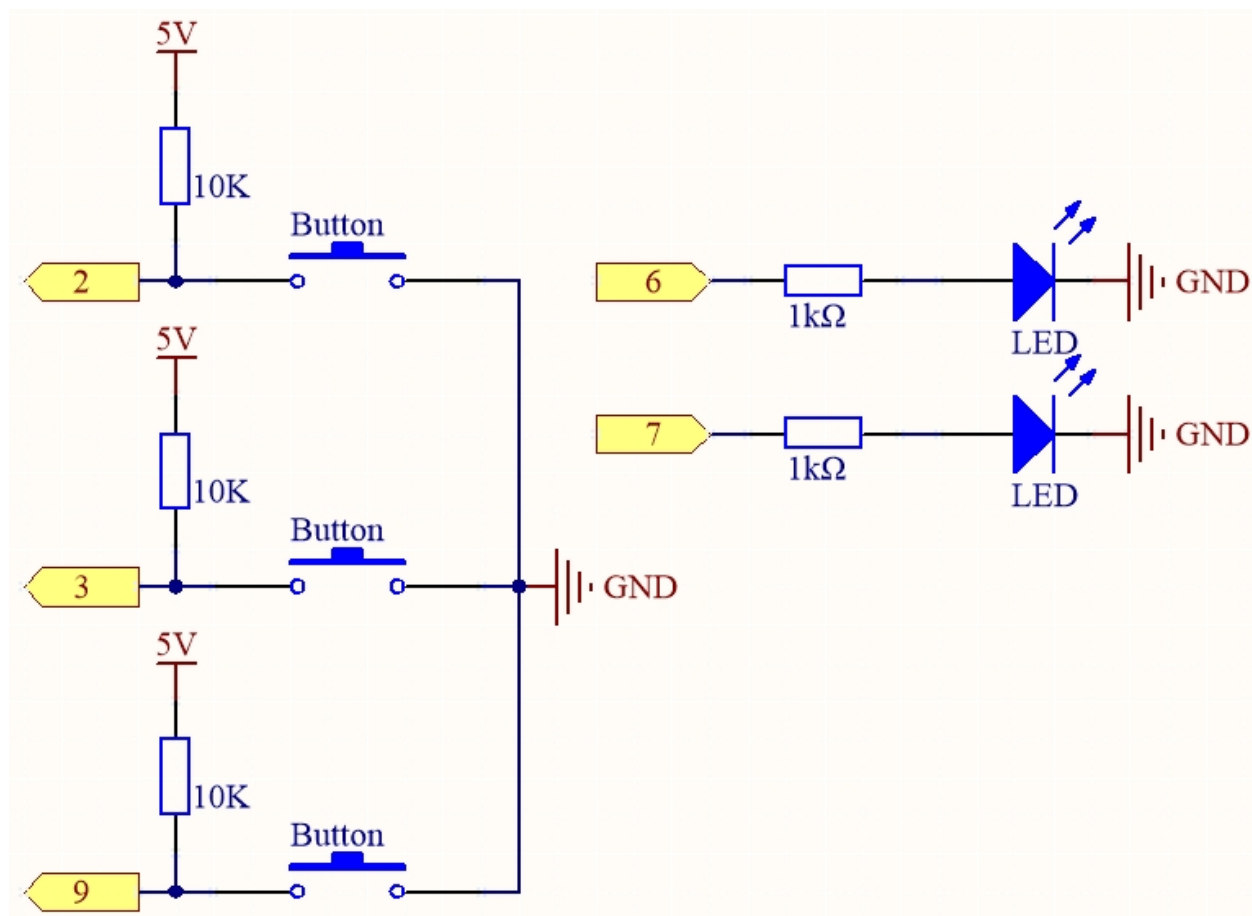
It’s definitely convenient to buy a whole kit, here’s the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

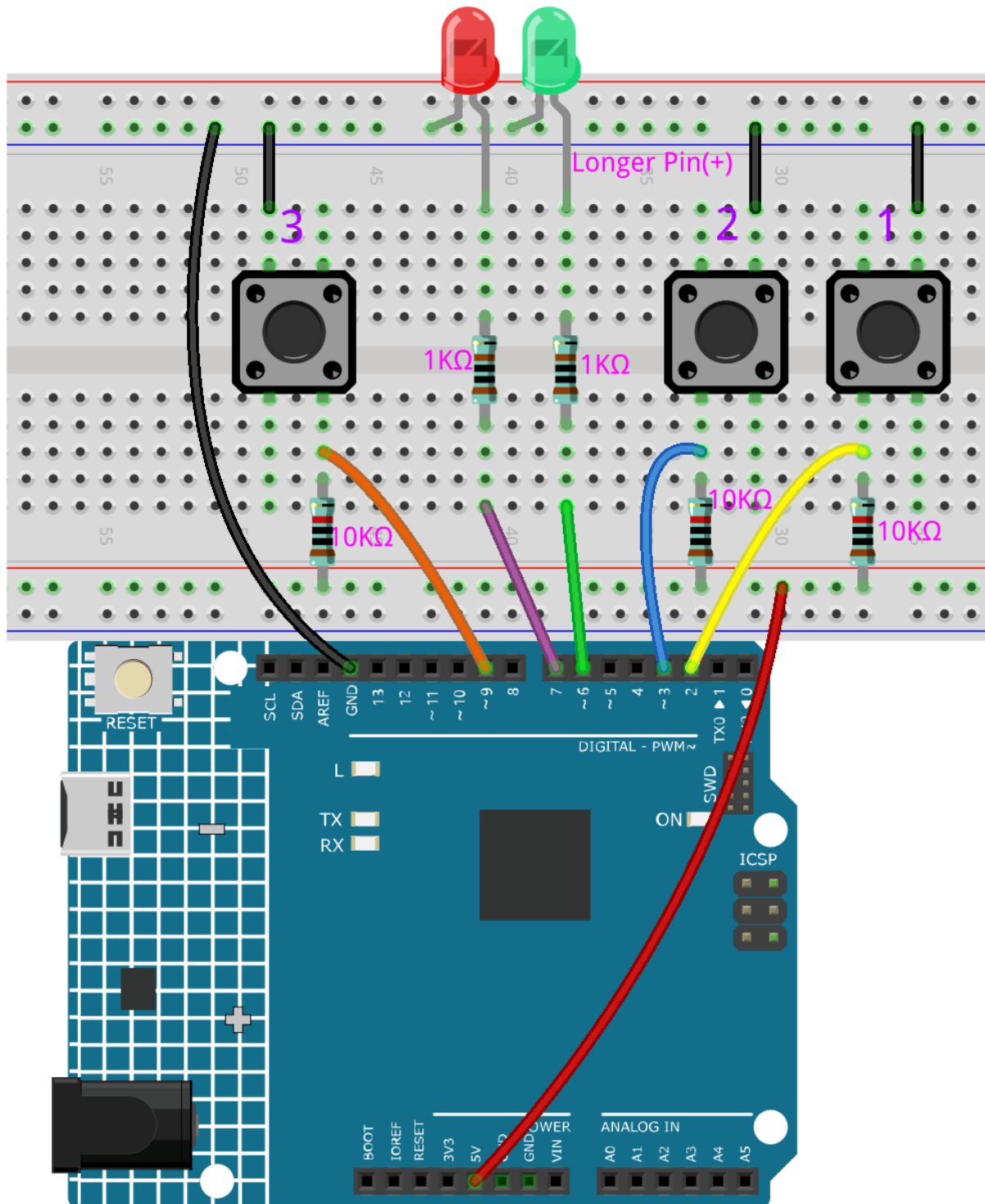
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	
<i>Button</i>	

### Schematic



### Wiring



## Code

### Note:

- You can open the file `6.5_reaction_time.ino` under the path of `3in1-kit\learning_project\6.5_reversingAid` directly.

- Or copy this code into Arduino IDE.
  - Please make sure you have added the `LiquidCrystal_I2C` library, detailed tutorials refer to [5.11 Install External Libraries](#).
- 

### How it works?

1. Initialize the buttons and LEDs, 2 interrupt are used here to read the button status.

```
void setup()
{
    ...

    attachInterrupt(digitalPinToInterrupt(buttonPin1), pressed1, FALLING);
    attachInterrupt(digitalPinToInterrupt(buttonPin2), pressed2, FALLING);
    ...
}
```

2. If the `rstBtn` button is pressed, the game starts again. At a random time between 2 and 5ms, make one of the LEDs light up.

```
void loop()
{
    if (flag == -1 && digitalRead(rstBtn) == LOW) {
        digitalWrite(ledPin1, LOW);
        digitalWrite(ledPin2, LOW);
        Serial.println("Waiting...");
        int randomTime = random(2000, 5000);
        delay(randomTime);

        timer = millis();
        flag = randomTime % 2;
        Serial.println("Light!");

        if (flag == 0) {
            digitalWrite(ledPin1, HIGH);
        } else if (flag == 1) {
            digitalWrite(ledPin2, HIGH);
        }
    }
    delay(200);
}
```

- When flag is -1 and `rstBtn` button is pressed, use `random()` function to generate a random time of 2-5s.
  - This time is then used to control the lighting of the LEDs.
  - Also the lighting of 2 LEDs is randomly generated by `randomTime % 2` with 0 and 1. If flag is 0, then LED1 is lit; if 1, then LED2 is lit.
3. About `pressed1()` function

```
void pressed1() {
    if (flag == -1) {
```

(continues on next page)



(continued from previous page)

```

        return;
    }
    if (flag == 0) {
        int currentTime = millis();
        Serial.print("Correct! You reaction time is : ");
        Serial.print(currentTime - timer);
        Serial.println(" ms");
    } else if (flag == 1) {
        Serial.println("Wrong Click!");
    }
    flag = -1;
}

```

This is the function that will be triggered when button 1 is pressed. When button 1 is pressed, if the flag is 0 at this time, the response time will be printed, otherwise the press error will be prompted.

#### 4. About pressed2() function

```

void pressed2() {
    if (flag == -1) {
        return;
    }
    if (flag == 1) {
        int currentTime = millis();
        Serial.print("Correct! You reaction time is : ");
        Serial.print(currentTime - timer);
        Serial.println(" ms");
    } else if (flag == 0) {
        Serial.println("Wrong Click!");
    }
    flag = -1;
}

```

This is the function that will be triggered when button 2 is pressed. When button 2 is pressed, if the flag is 1 at this time, the response time will be printed, otherwise the press error will be prompted.

### 4.6.6 6.6 Guess Number

Guessing Numbers is a fun party game where you and your friends take turns inputting a number (0~99). The range will be smaller with the inputting of the number till a player answers the riddle correctly. Then the player is defeated and punished. For example, if the lucky number is 51 which the players cannot see, and the player 1 inputs 50, the prompt of number range changes to 50~99; if the player 2 inputs 70, the range of number can be 50~70; if the player 3 inputs 51, he or she is the unlucky one. Here, we use IR Remote Controller to input numbers and use LCD to output outcomes.

#### Required Components

In this project, we need the following components.

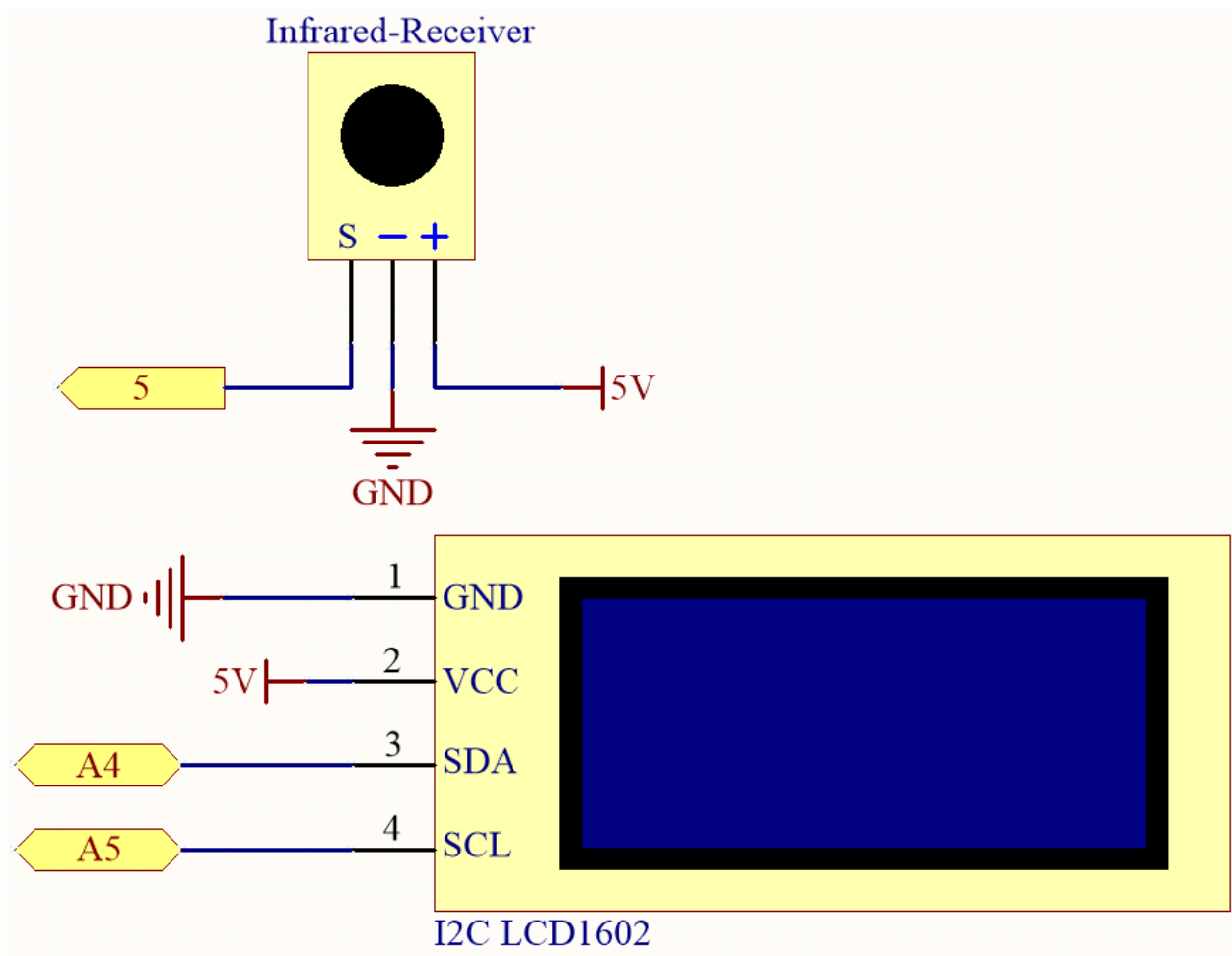
It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

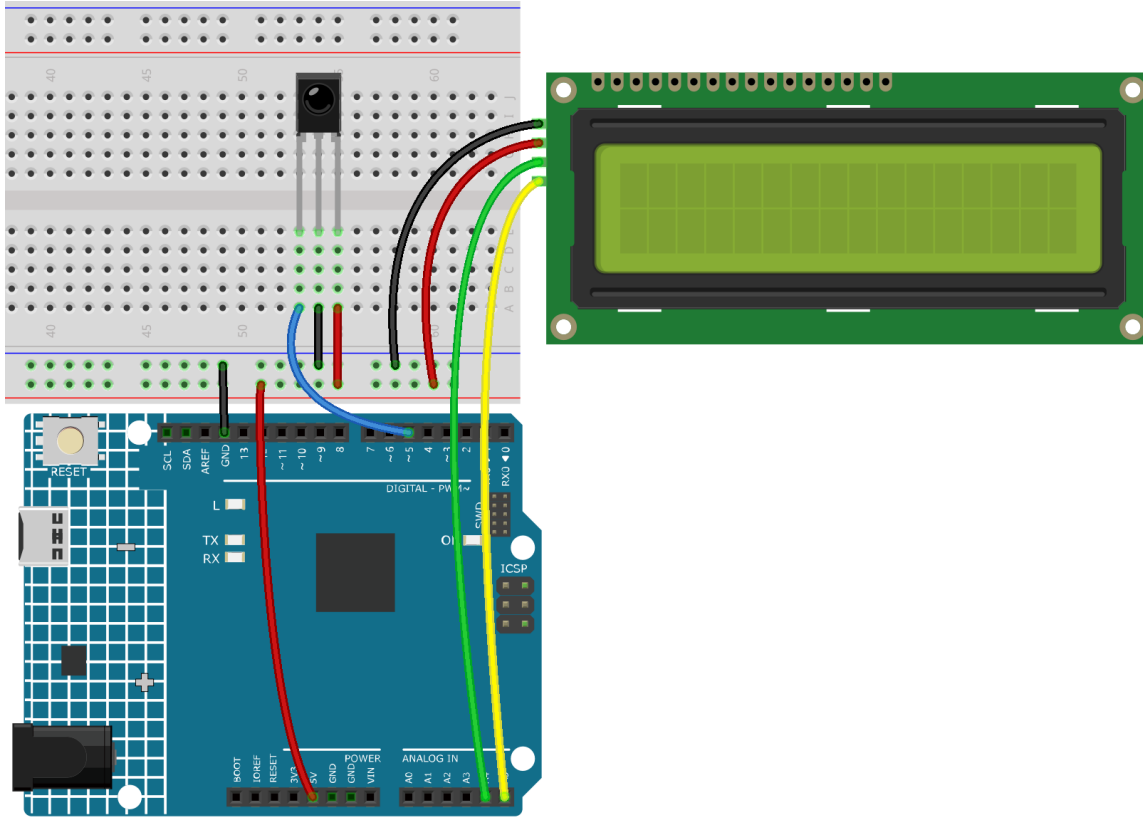
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>Jumper Wires</i>	
<i>I2C LCD1602</i>	
<i>IR Receiver</i>	-

### Schematic



### Wiring

In this example, the wiring of LCD1602 and infrared receiving module is as follows.



## Code

### Note:

- You can open the file 6.6.guess\_number.ino under the path of 3in1-kit\learning\_project\6.6.guess\_number directly.
- Or copy this code into Arduino IDE.
- The LiquidCrystal I2C and IRremote libraries are used here, you can install them from the **Library Manager**.

After the code is successfully uploaded, the welcome characters will appear on the LCD1602. Now press the number according to the range prompt on the screen, the display will get smaller and smaller unless you guess that lucky number.

**Note:** If the code and wiring are fine, but the LCD still does not display content, you can turn the potentiometer on the back to increase the contrast.

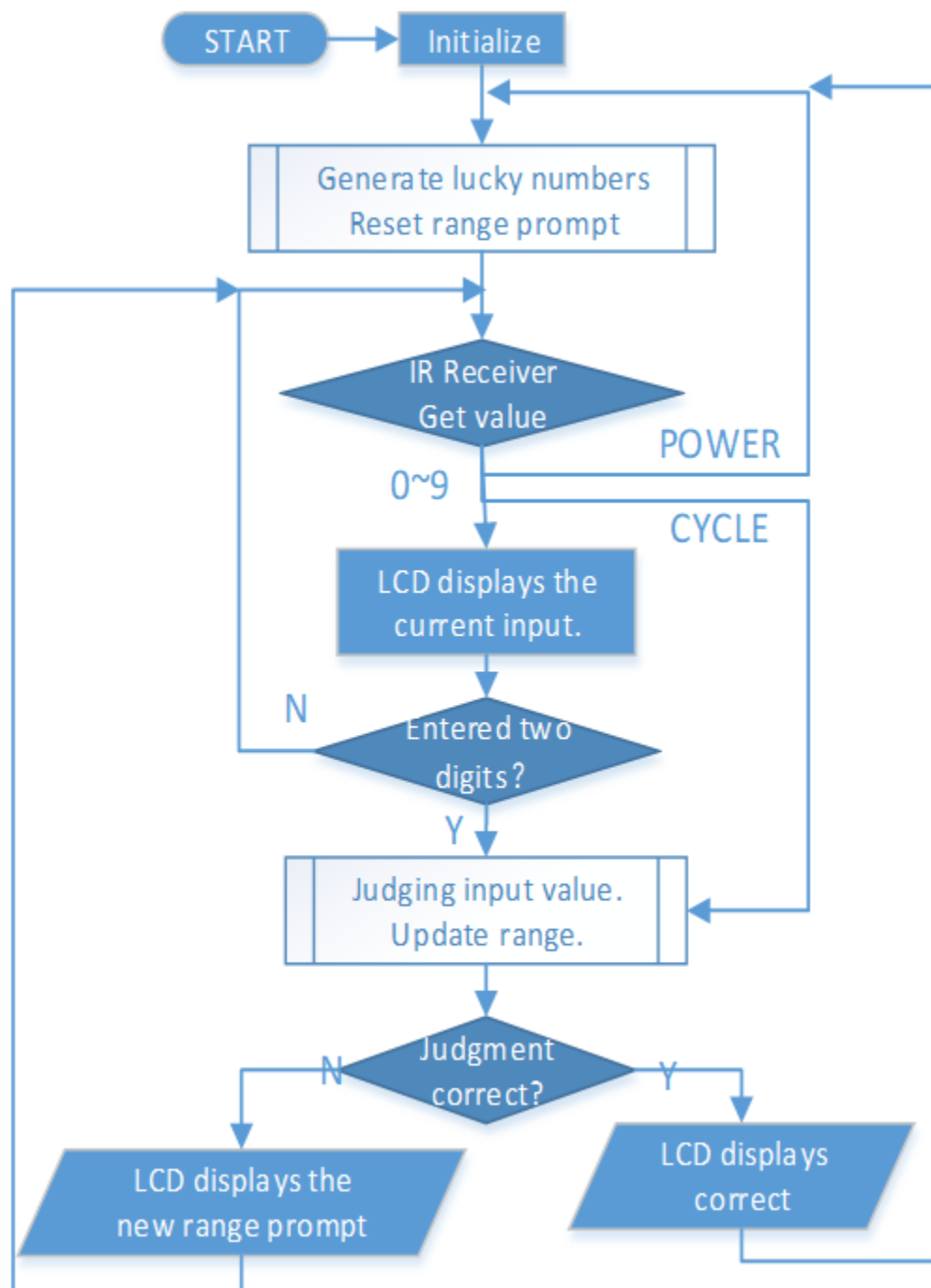
### How it works?

In order to make the number guessing game become vivid and funny, we need to achieve the following functions:

1. The lucky number will be displayed when we start and reset the game, and the number range prompt is reset to 0 ~ 99.
2. LCD will display the number being input and the number range prompt.
3. After inputting two digits, there appears result judgment automatically.

4. If you input a single digit, you can press the CYCLE key (the key at the center of the Controller) to start the result judgment.
5. If the answer is not guessed, the new number range prompt will be displayed (if the lucky number is 51 and you enter 50, the number range prompt will change to 50~99).
6. The game is automatically reset after the lucky number is guessed, so that the player can play a new round.
7. The game can be reset by directly pressing the POWER button (the button in the upper left corner).

In conclusion, the work flow of the project is shown in the flow chart.



## CAR PROJECTS

I believe you have seen a lot of different smart robot car, their basic functions are similar, basic movement, obstacle avoidance, line following, following and control by remote control, etc..

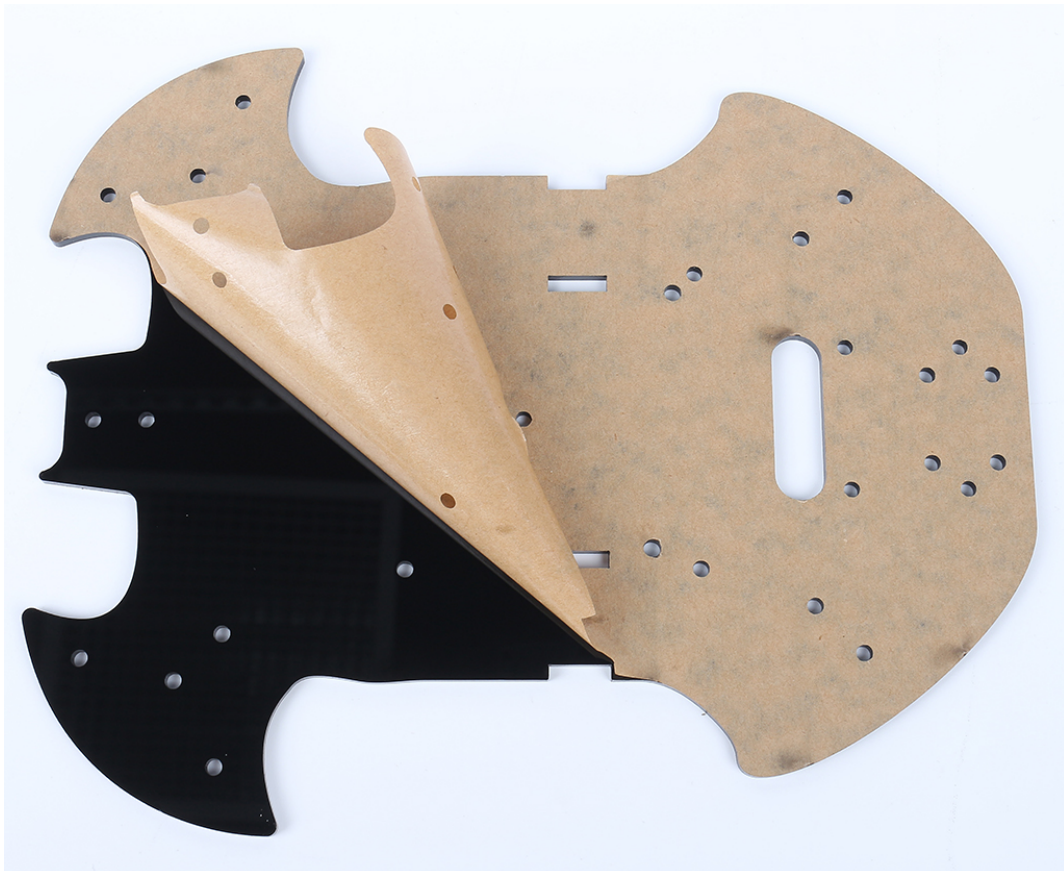
Here, we use the simplest structure to build a smart robot car, which can also achieve all the above functions. In addition you can control it with your cell phone, please refer to [8. IoT Car](#) for the tutorial.

### assembly instructions

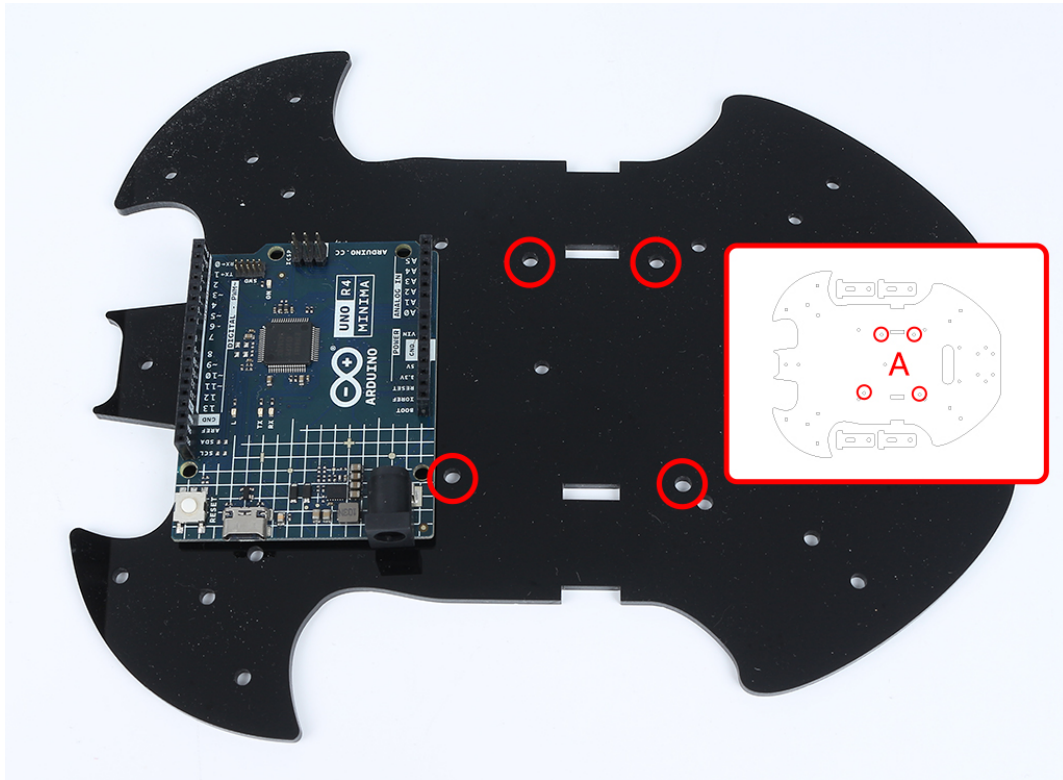
## 5.1 Assemble the Car

Please follow the steps below to complete the assembly of the car.

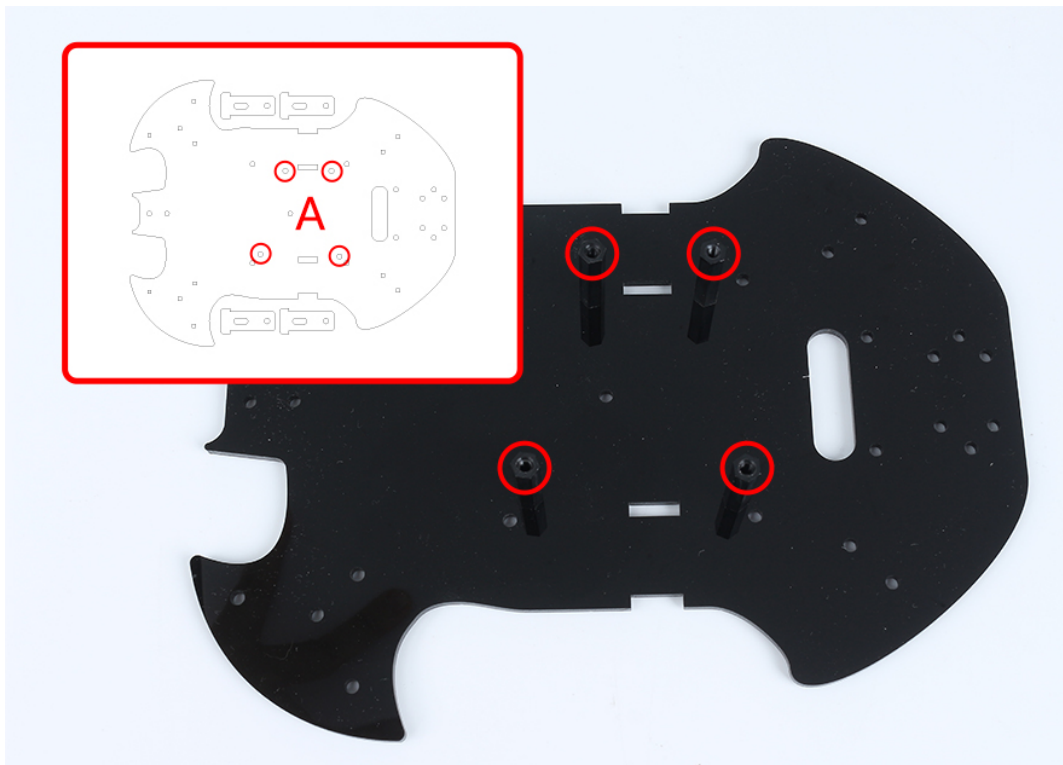
1. Remove the protective film on the acrylic.



2. Place the board on the table as shown in the picture, the side with the same hole as the R4 board, we call A; the back is B. This will help you avoid mistakes during assembly.

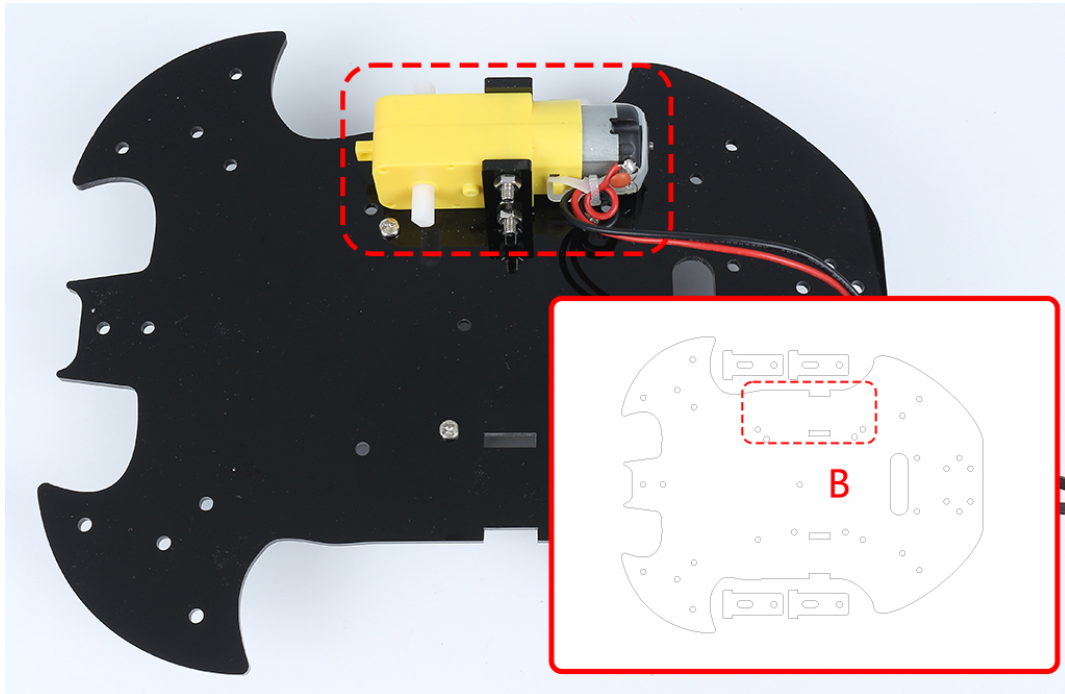


3. Mount the **M3x24mm standoff** with **M3x6mm screws** in the position as shown below.

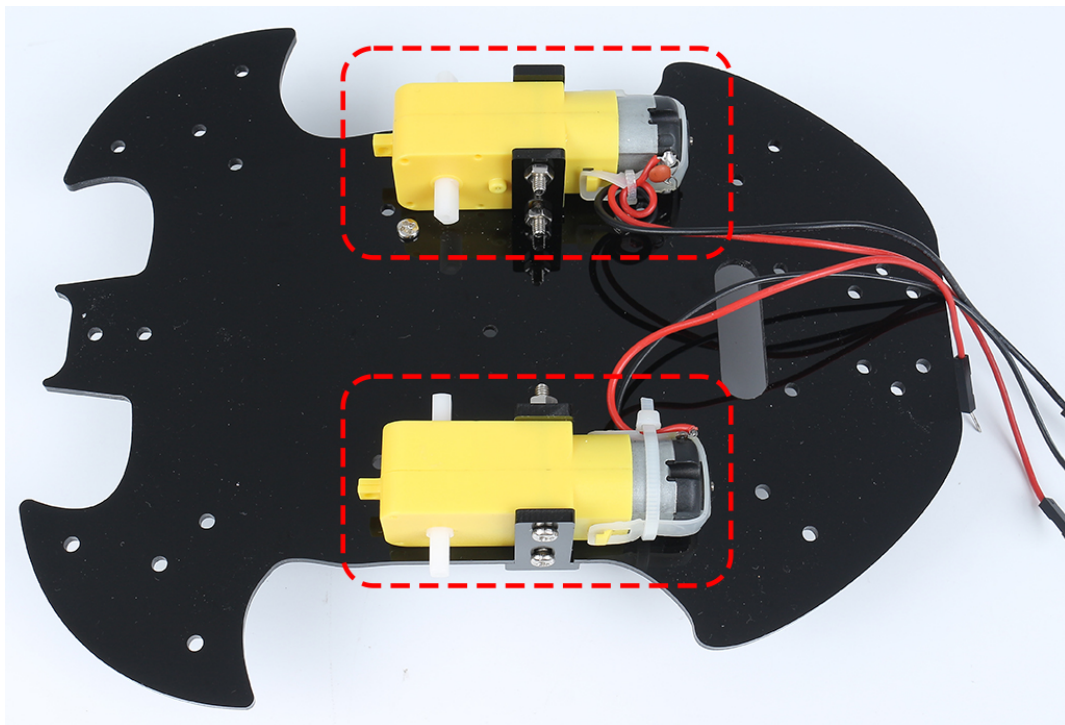




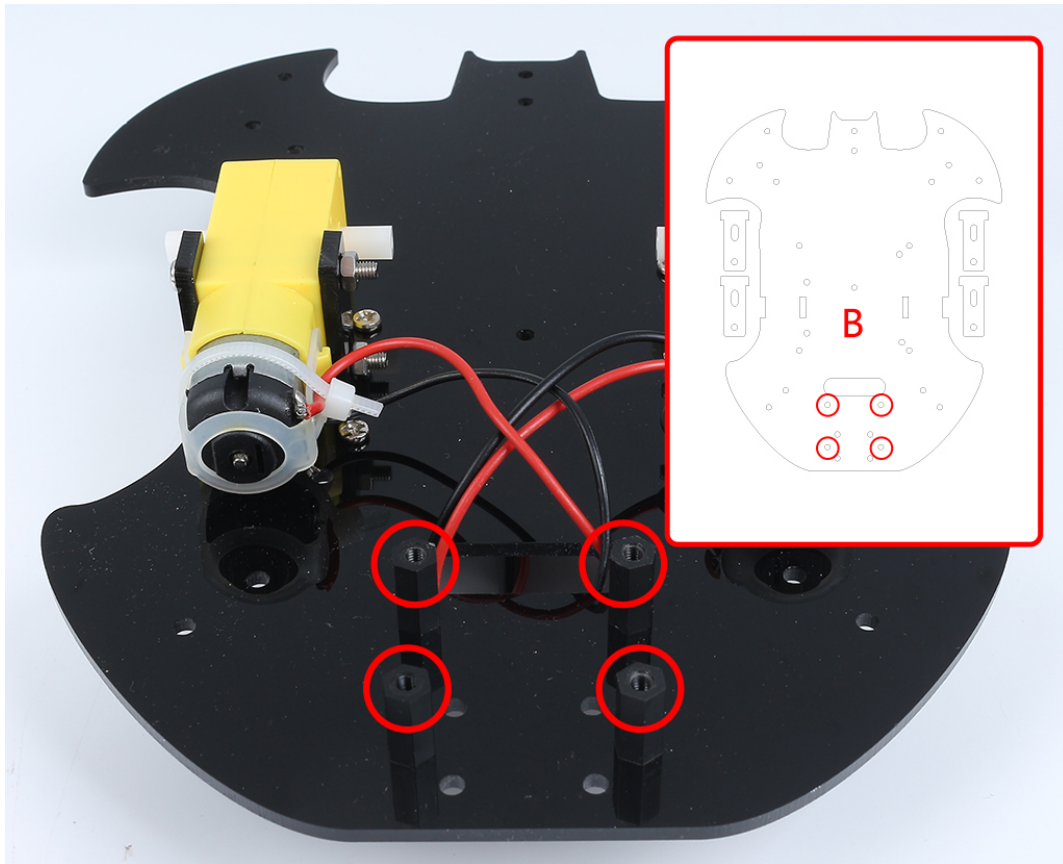
4. Turn to the B side, use **M3x30mm screws** and **M3 nuts** to attach the TT Motor. 2 details here: 1 - the output shaft is facing the bat-shaped side; 2 - the motor cable is facing the inside.



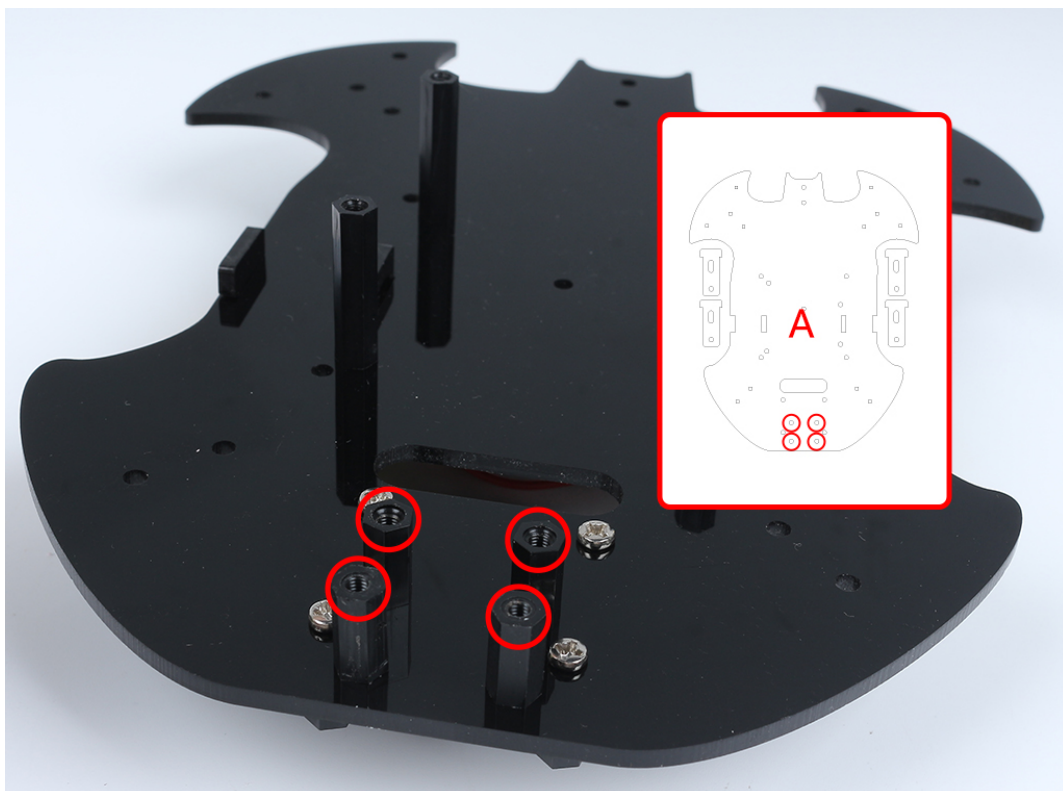
5. Mount another TT Motor, the same attention needs to be paid to the direction of the output shaft and the direction of the cable.



6. Use **M3x6mm screws** to mount the **M3x10mm standoff** in the position as shown below.

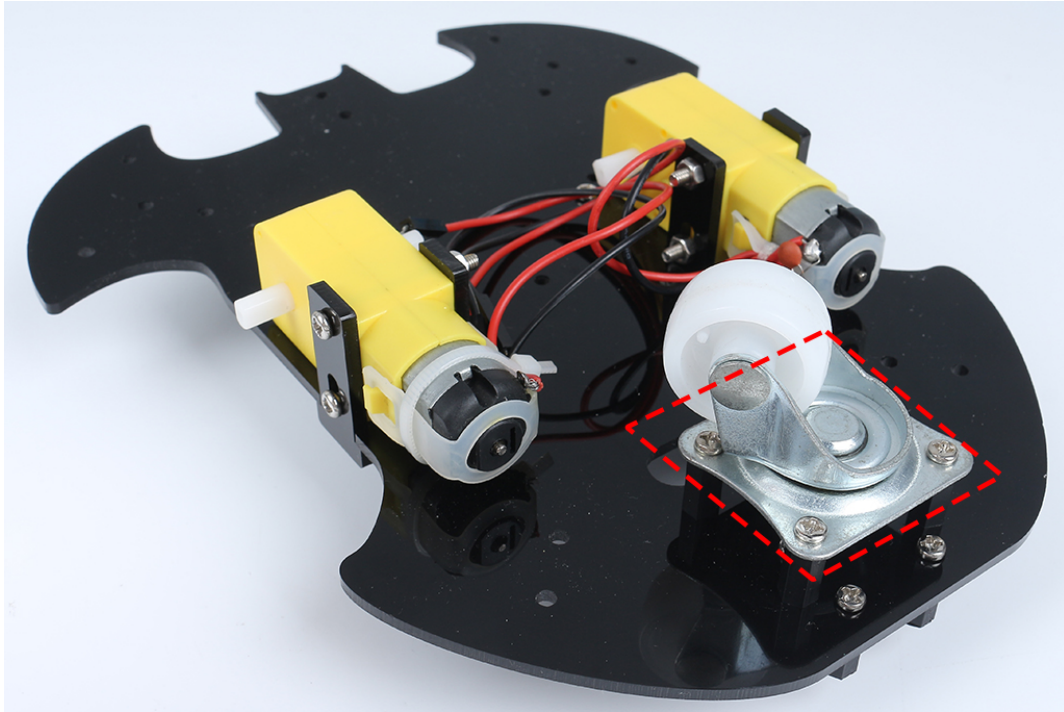


7. Attach the **M2.5x11mm standoff** to the rear of the car with **M2.5x6mm screws**.

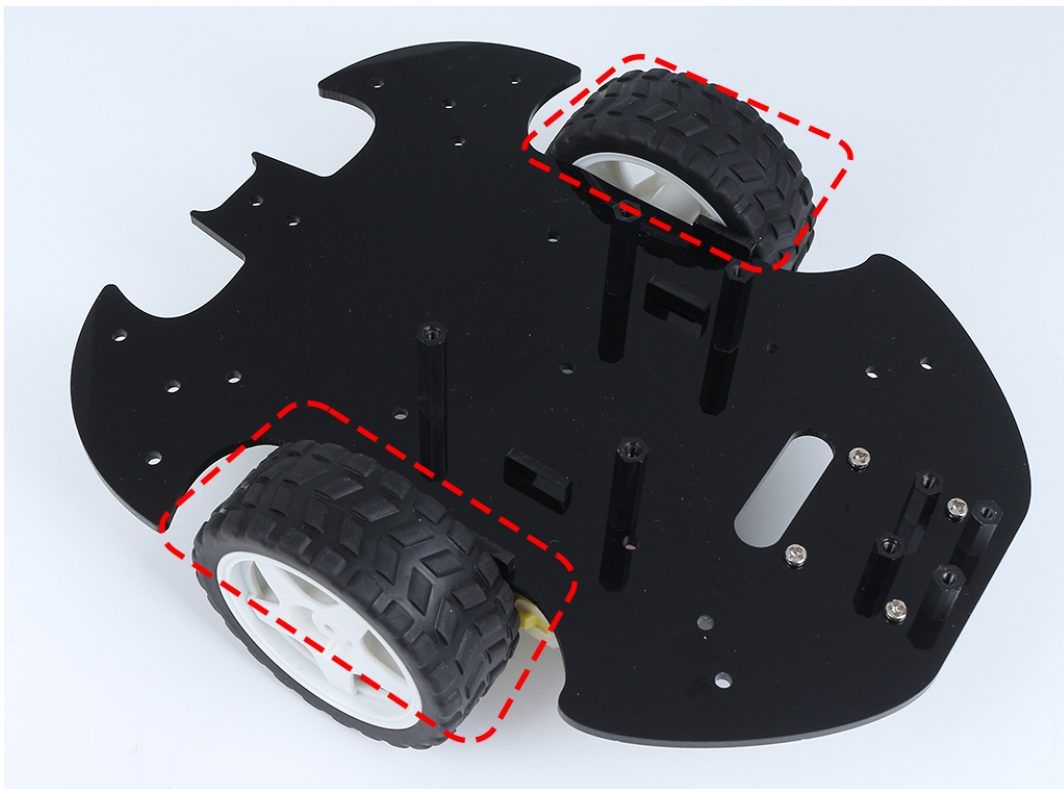




8. Use **M3x6mm** screws to mount the universal wheel.



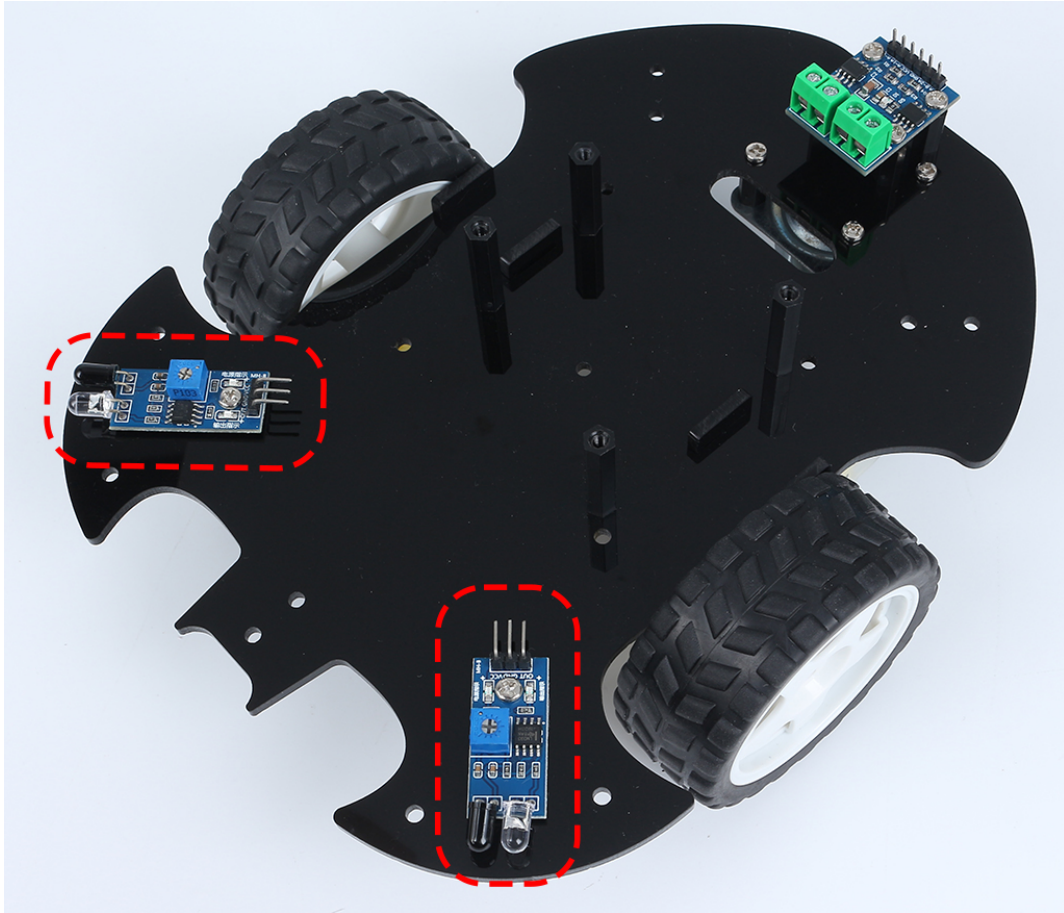
9. Putting on the 2 wheels and the car's basic structure has been completed.



10. Attach the L9110 module with **M2.5x6mm** screws.



11. Assemble the two IR Obstacle Modules with **M3x10mm** screws and **M3** nuts.



12. Turn to side B and attach the Line Track module with four **M3x6mm** screws and two **M3x24mm** standoffs.

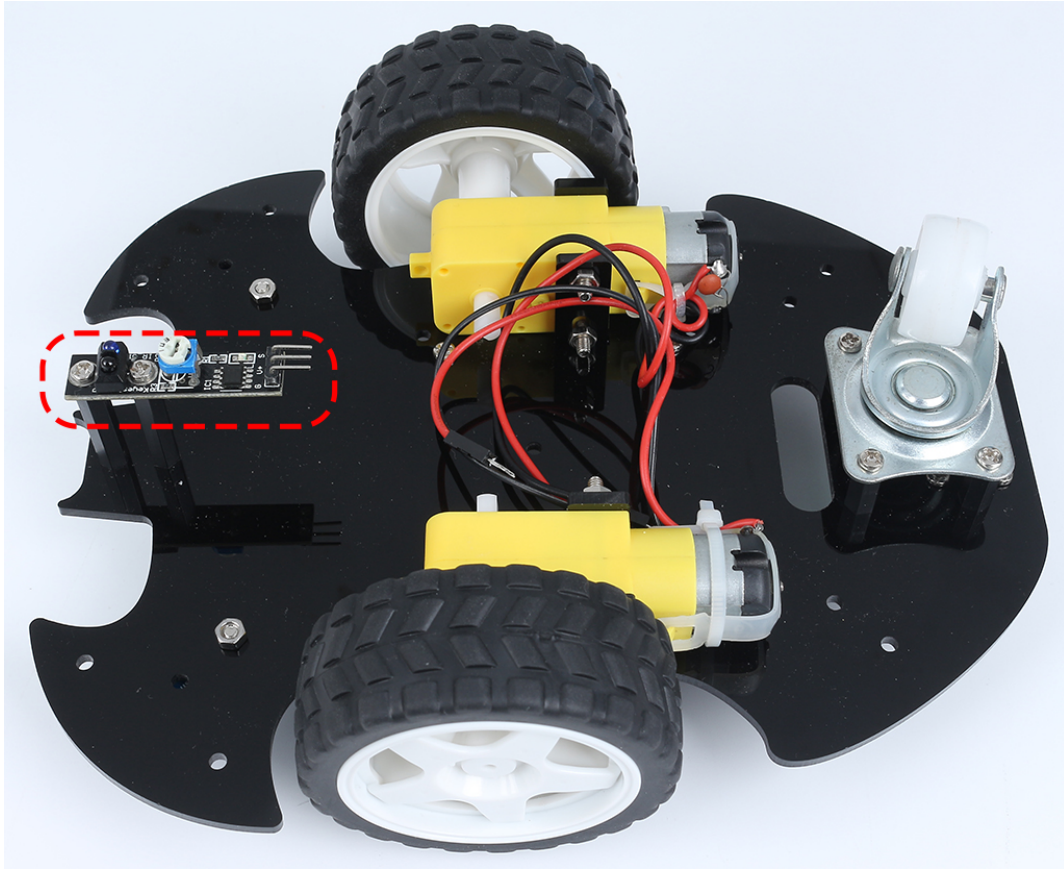
---

**Note:** It's advisable to first secure the **M3x24mm** standoffs onto the Line Track module.

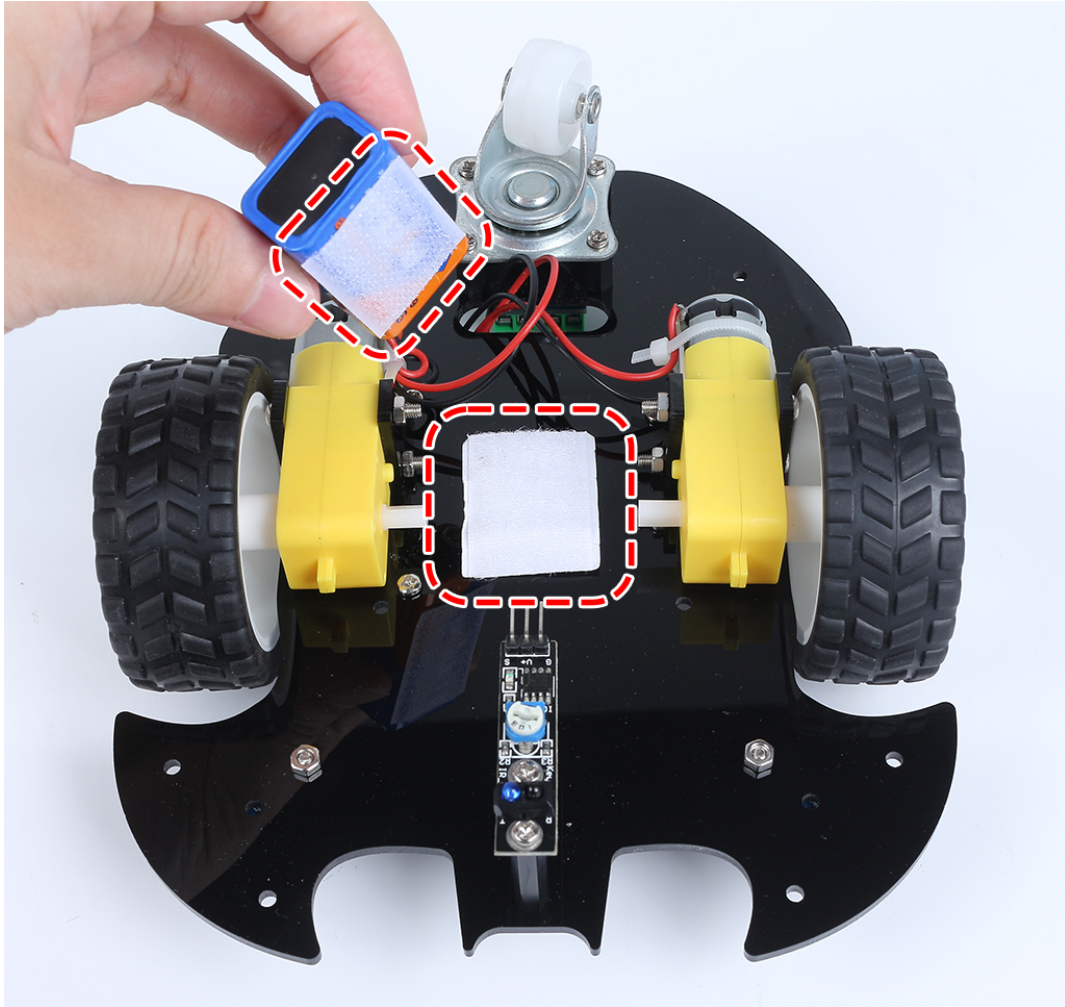
One important note to keep in mind: the pins of the line sensor are slightly soft and protrude a bit towards the holes. When screwing in the **M3x24mm** standoffs, apply gentle pressure to push aside the sensor pins slightly.

---

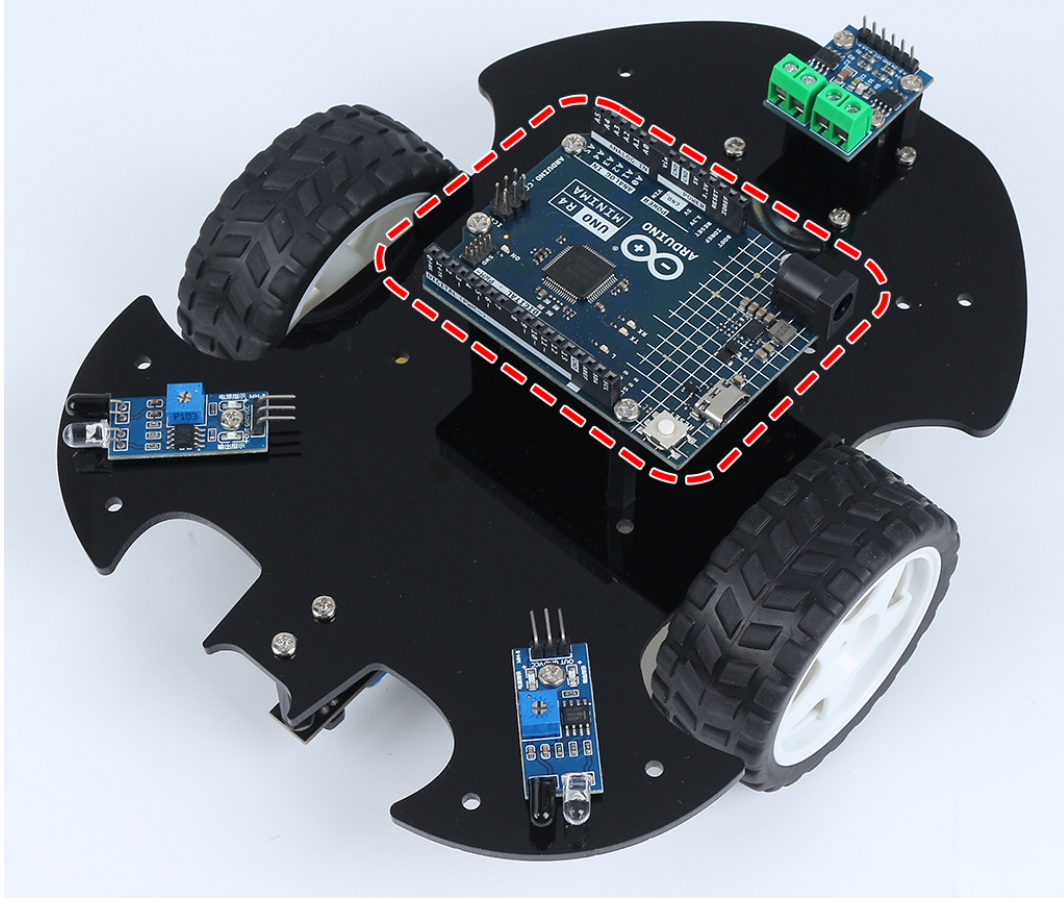




13. Stick the velcro on the 9V battery and put on the battery clip. Stick the other section of the Velcro on the car to secure the battery.

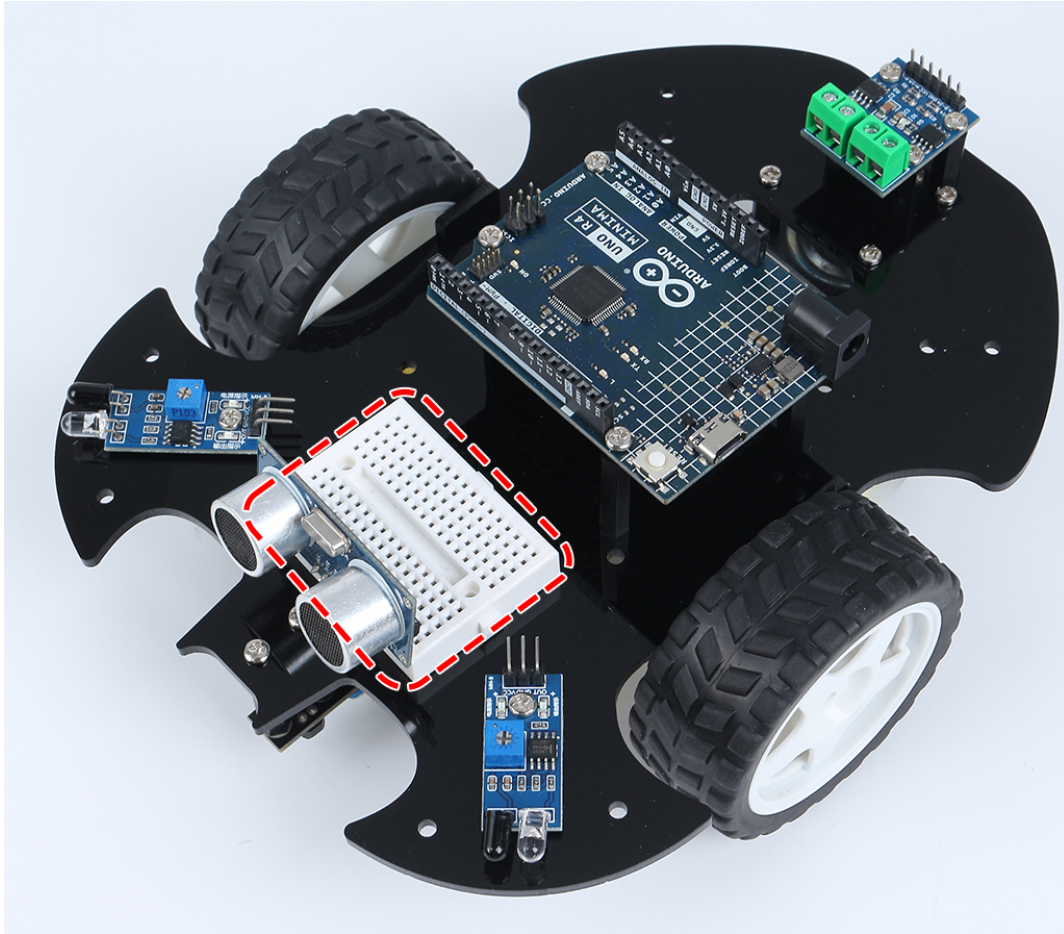


14. Turn over to side A and mount the R4 board with **M3x6mm** screws.



15. Attach the breadboard to the front of the car. Thereafter, you will be able to add different components (e.g. ultrasonic module) to the breadboard as required for your project.





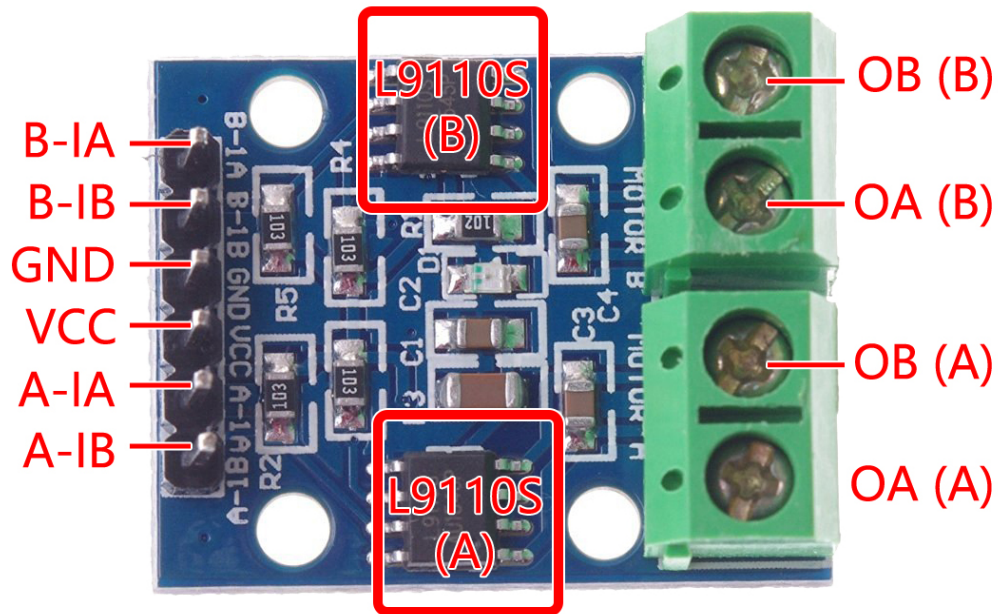
16. Getting the car running also requires wiring it up and writing code, which will be written in subsequent sections.

### Projects

Here are some projects for the car, programmed in C with the Arduino IDE, if you are not particularly skilled with Arduino, you can refer to *Get Started with Arduino*.

The following projects are written in order of programming difficulty, it is recommended to read these books in order.

## 5.2 1. Move



Before we start programming, let's review the working principle of L9110 module.

Here is the truth table of Motor B:

B-1A	B-1B(B-2A)	The state of Motor B
1	0	Rotate clockwise
0	1	Rotate counterclockwise
0	0	Brake
1	1	Stop

Here is the truth table of Motor A:

A-1A	A-1B	The state of Motor B
1	0	Rotate clockwise
0	1	Rotate counterclockwise
0	0	Brake
1	1	Stop

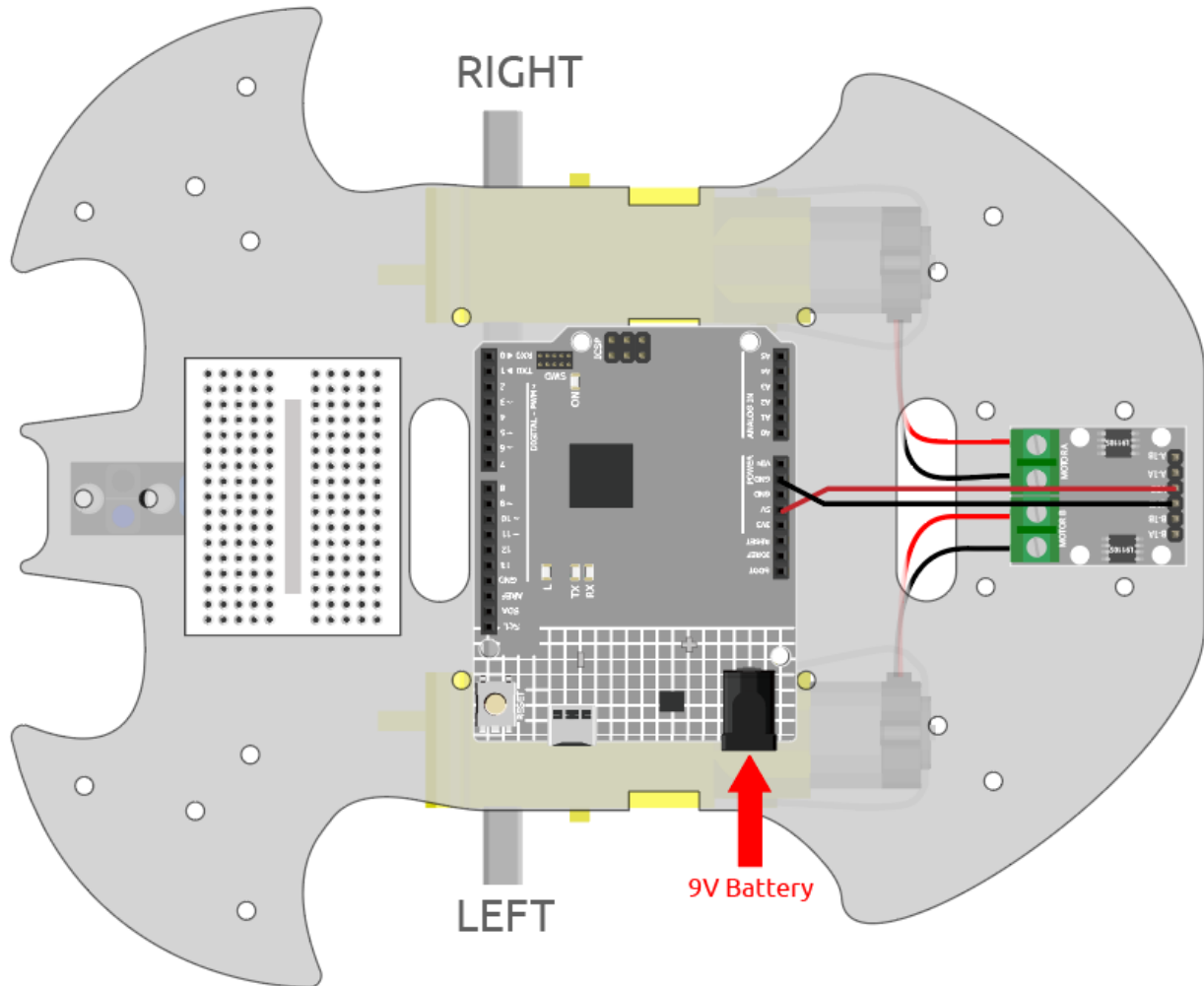
- *L9110 Motor Driver Module*

### Forward

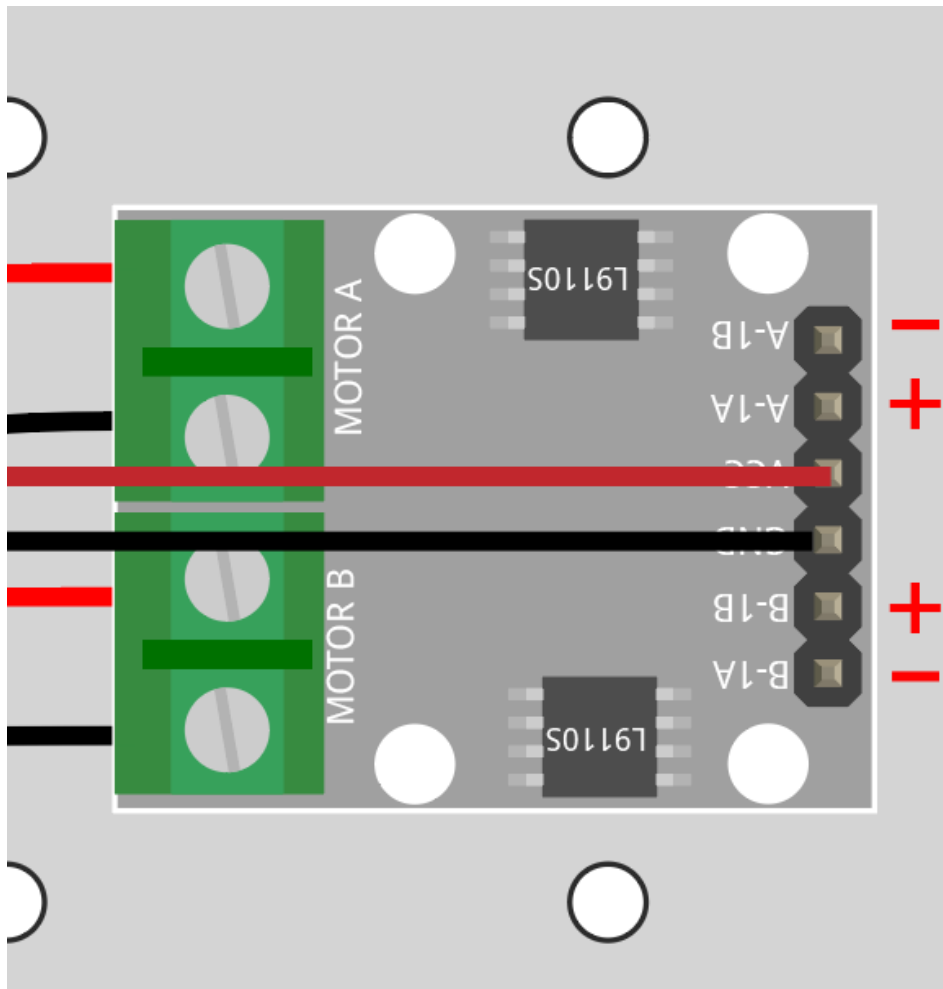
Now let's connect the input of L9110 module directly to 12V and GND respectively to make the car move.

1. Connect R4 board, L9110 module and 2 motors.





2. Connect B-1B(B-2A) and A-1A to VCC, and B-1A and A-1B to GND, then you will be able to see the car moving forward.

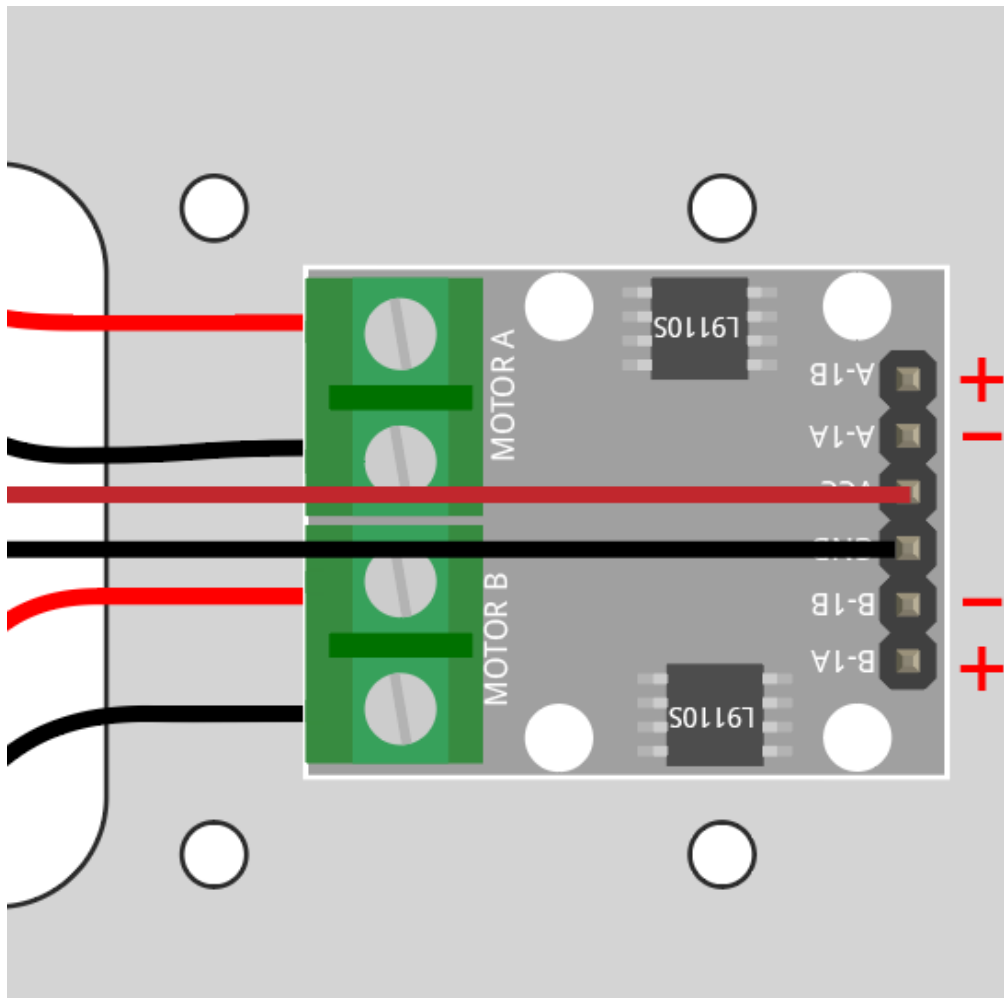


If not both turn forward, but the following situations occur, you need to readjust the wiring of the two motors.

- If both motors turn backward at the same time (left motor turns clockwise, right motor turns counterclockwise), swap the wiring of the left and right motors at the same time, OA(A) and OB(A) swap, OA(B) and OB(B) swap.
- If the left motor turns backward (clockwise rotation), exchange the wiring of OA(B) and OB(B) of the left motor.
- If the right motor turns backward (counterclockwise rotation), swap the wiring of OA(A) and OB(A) of the right motor.

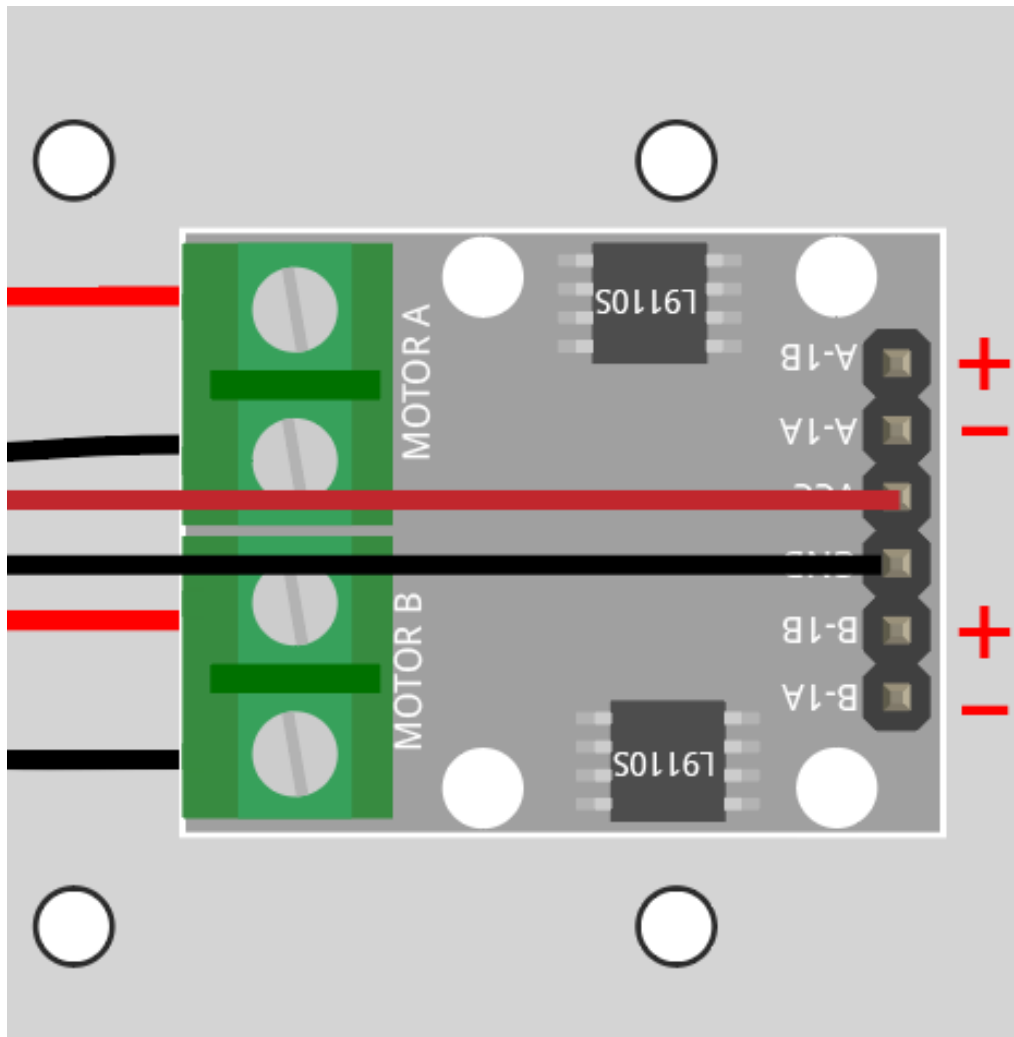
### Backward

Connect B-1B(B-2A) and A-1A to GND, and B-1A and A-1B to VCC, then you will be able to see the car moving backward.



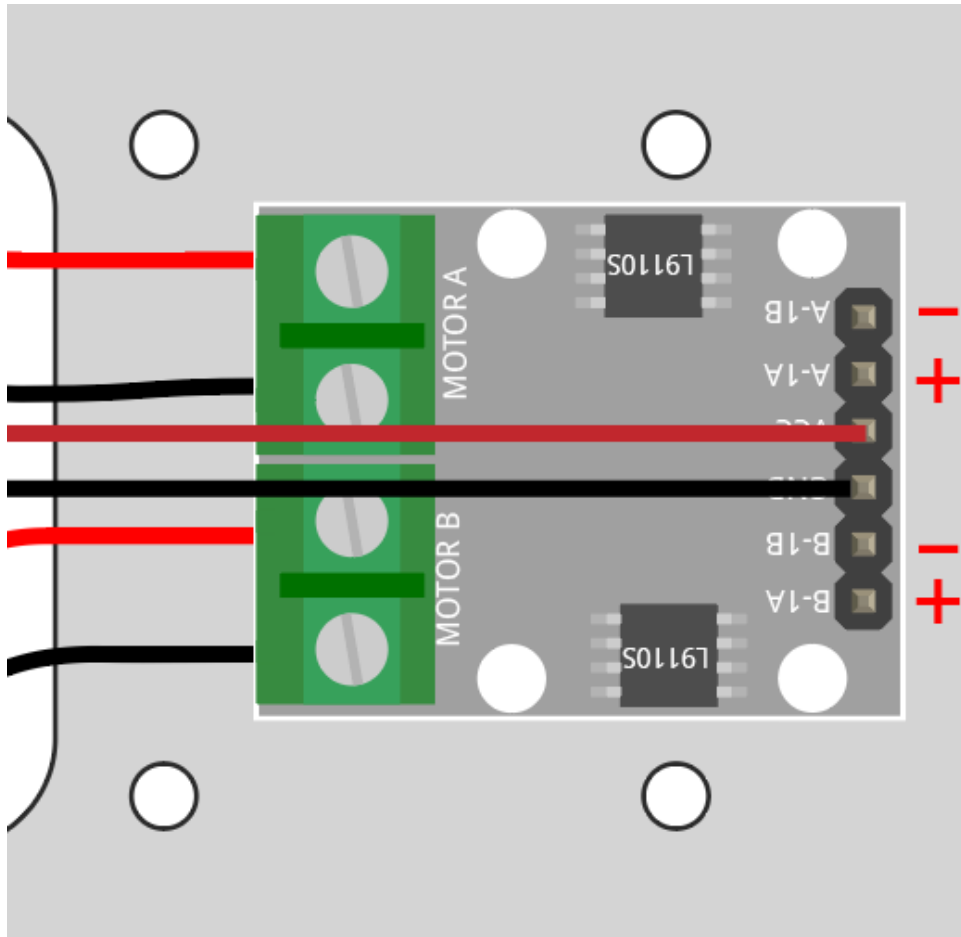
### Turn Left

If you want to make the car turn left, that is, make both motors turn clockwise. You need to connect B-1A and A-1A to GND, and B-1B(B-2A) and A-1B to VCC.



### Turn Right

Conversely, if you want to turn the car to the right, that is, make both motors turn counterclockwise. You need to connect B-1A and A-1A to VCC and B-1B(B-2A) and A-1B to GND.



### Stop

To stop the motor, connect the inputs on the same side to 12V or GND at the same time, e.g. connect B-1A and B-1B(B-2A) to 12V or 5V at the same time, and the same for A-1A and A-1B.

This is of course theoretical and needed later on when controlling with code. Here remove the power supply to the car can stop it.

## 5.3 2. Move by Code

In the previous project, we have tried to control the operation of the motor by using different level signals for the input of the L9110 module.

If we modify the level signals through the program, then we can control the movement of the car in a flexible way.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

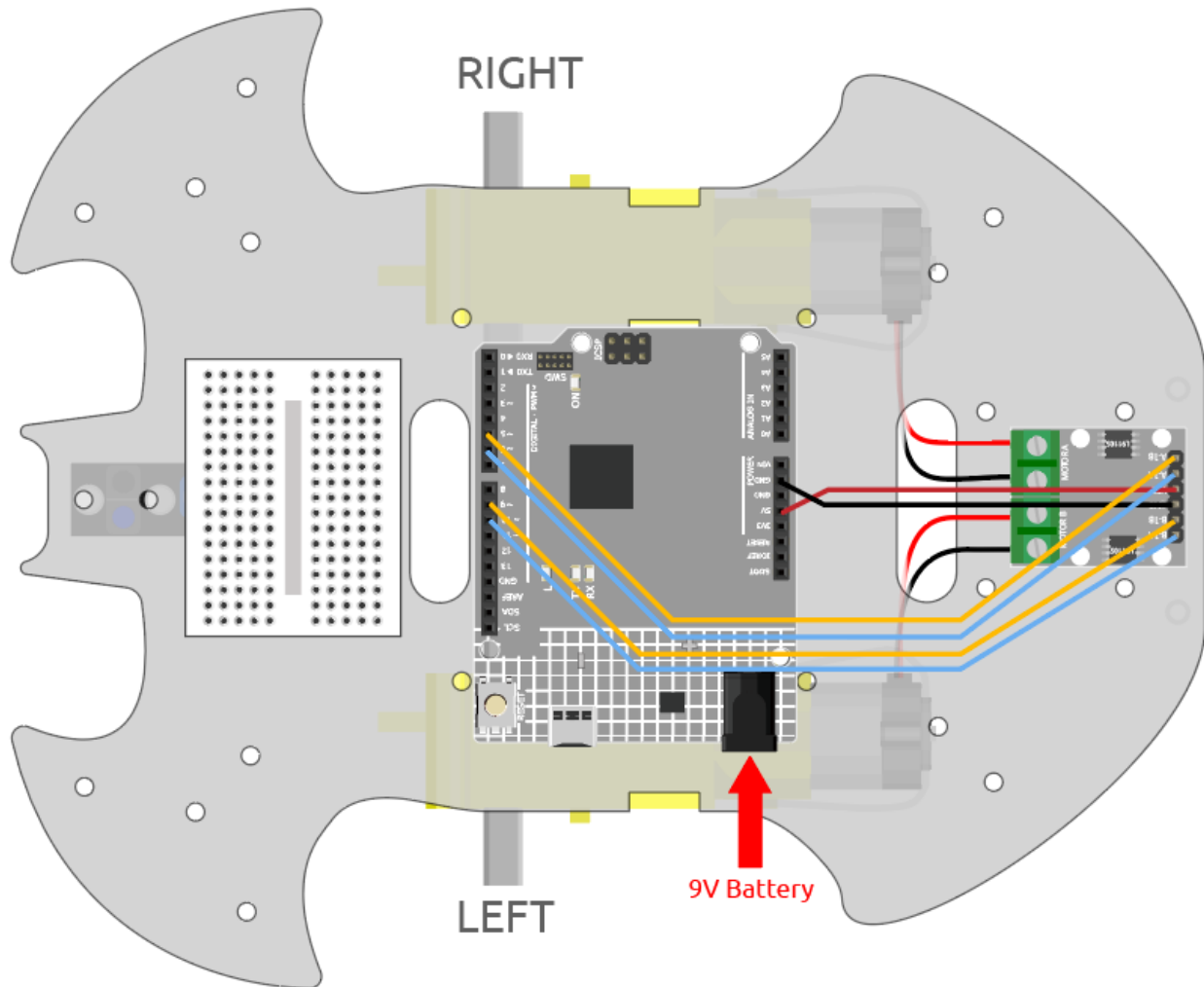
You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-

### Wiring

Connect the wires between the L9110 module and the R4 board according to the diagram below.

L9110 Module	R4 Board	Motor
A-1B	5	
A-1A	6	
B-1B(B-2A)	9	
B-1A	10	
OB(B)		Black wire of right motor
OA(B)		Red wire of right motor
OB(A)		Black wire of left motor
OA(A)		Red wire of left motor



## Code

### Note:

- Open the 2.move.ino file under the path of 3in1-kit\car\_project\2.move.
- Or copy this code into **Arduino IDE**.

After the code is uploaded, the car will move forward, backward, left and right for two seconds respectively.

### How it works?

This project is essentially the same as the previous one, involving making the car move forward, backward, left, and right, as well as stopping by providing different signal levels to the input pins of the L9110 module.

1. Initialize the pins of L9110 module.

```
const int A_1B = 5;
const int A_1A = 6;
const int B_1B = 9;
const int B_1A = 10;
```

(continues on next page)

(continued from previous page)

```
void setup() {  
    pinMode(A_1B, OUTPUT);  
    pinMode(A_1A, OUTPUT);  
    pinMode(B_1B, OUTPUT);  
    pinMode(B_1A, OUTPUT);  
}
```

2. Set the input pins to different high or low levels to control the rotation of the left and right motors, and then encapsulate them in individual functions.

```
void moveForward() {  
    digitalWrite(A_1B, LOW);  
    digitalWrite(A_1A, HIGH);  
    digitalWrite(B_1B, HIGH);  
    digitalWrite(B_1A, LOW);  
}  
  
void moveBackward() {  
    digitalWrite(A_1B, HIGH);  
    digitalWrite(A_1A, LOW);  
    digitalWrite(B_1B, LOW);  
    digitalWrite(B_1A, HIGH);  
}  
...
```

3. Call these functions in loop().

```
void loop() {  
    moveForward();  
    delay(2000);  
    stopMove();  
    delay(500);  
  
    moveBackward();  
    delay(2000);  
    stopMove();  
    delay(500);  
    ...  
}
```

- `digitalWrite(pin, value)`

- pin: the Arduino pin number.
- value: HIGH or LOW.

Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with `pinMode()`, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

- `pinMode(pin, mode)`

- pin: the Arduino pin number to set the mode of.
- mode: INPUT, OUTPUT, or INPUT\_PULLUP.

Configures the specified pin to behave either as an input or an output.



- `delay(ms)`

- ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

## 5.4 3. Speed Up

In addition to the digital signal (HIGH/LOW), the input of L9110 module can also receive PWM signal to control the speed of the output.

In other words, we can use `AnalogWrite()` to control the moving speed of the car.

In this project, we let the car gradually change its forward speed, first accelerating and then decelerating.

### Wiring

This project is the same wiring as 2. *Move by Code*.

### Code

#### Note:

- Open the `3.speed_up.ino` file under the path of `3in1-kit\car_project\3.speed_up`.
- Or copy this code into **Arduino IDE**.
- Or upload the code through the [Arduino Web Editor](#).

After the program runs, the car will gradually accelerate and then gradually decelerate.

### How it works?

The purpose of this project is to write different PWM values to the input pins of the L9110 module to control the forward speed of the car.

1. Use the `for()` statement to give speed in steps of 5, writing values from 0 to 255 so you can see the change in the car's forward speed.

```
void loop() {
  for(int i=0;i<=255;i+=5){
    moveForward(i);
    delay(500);
  }
  for(int i=255;i>=0;i-=5){
    moveForward(i);
    delay(500);
  }
}
```

2. About the `moveForward()` function.

As opposed to 2. *Move by Code* which directly gives high/low levels to the input pins of the L9110 module, here we pass a parameter `speed` to where we need to give high levels.

```
void moveForward(int speed) {
  analogWrite(A_1B, 0);
  analogWrite(A_1A, speed);
  analogWrite(B_1B, speed);
  analogWrite(B_1A, 0);
}
```

- for

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop.

```
for (initialization; condition; increment) {
  // statement(s);
}
```

- **initialization:** happens first and exactly once.
- **condition:** each time through the loop, condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.
- **increment:** executed each time through the loop when condition is true.

## 5.5 4. Follow the line

The car is equipped with a Line Track module, which can be used to make the car follow the black line.

When the line following module detects the black line, the right motor rotates while the left motor does not, so that the car moves one step to the left front. As the car moves, the line module will be moved out of the line, then the left motor turns and the right motor does not turn, the car will move one step to the right to return to the line. Repeat the above two steps, the car can move along the black line.

Before starting the project, you need to build a curve map with black line tape, the recommended line width is between 0.8-1.5cm and the angle of the turn should not be less than 90 degrees.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

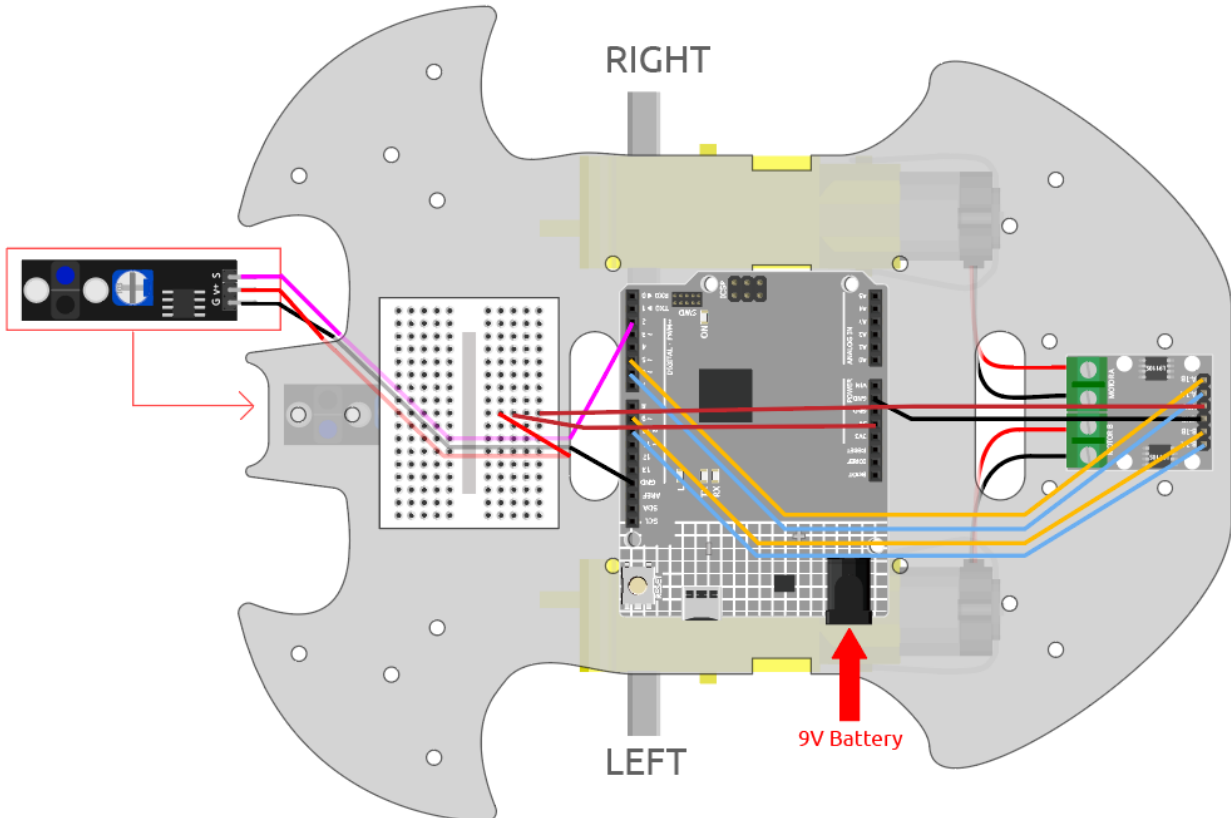
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>Line Tracking Module</i>	

## Wiring

This is a digital Line Tracking module, when a black line is detected, it outputs 1; when a white line is detected, it outputs a value of 0. In addition, you can adjust its sensing distance through the potentiometer on the module.

Build the circuit according to the following diagram.

Line Tracking Module	R4 Board
S	2
V+	5V
G	GND



## Adjust the Module

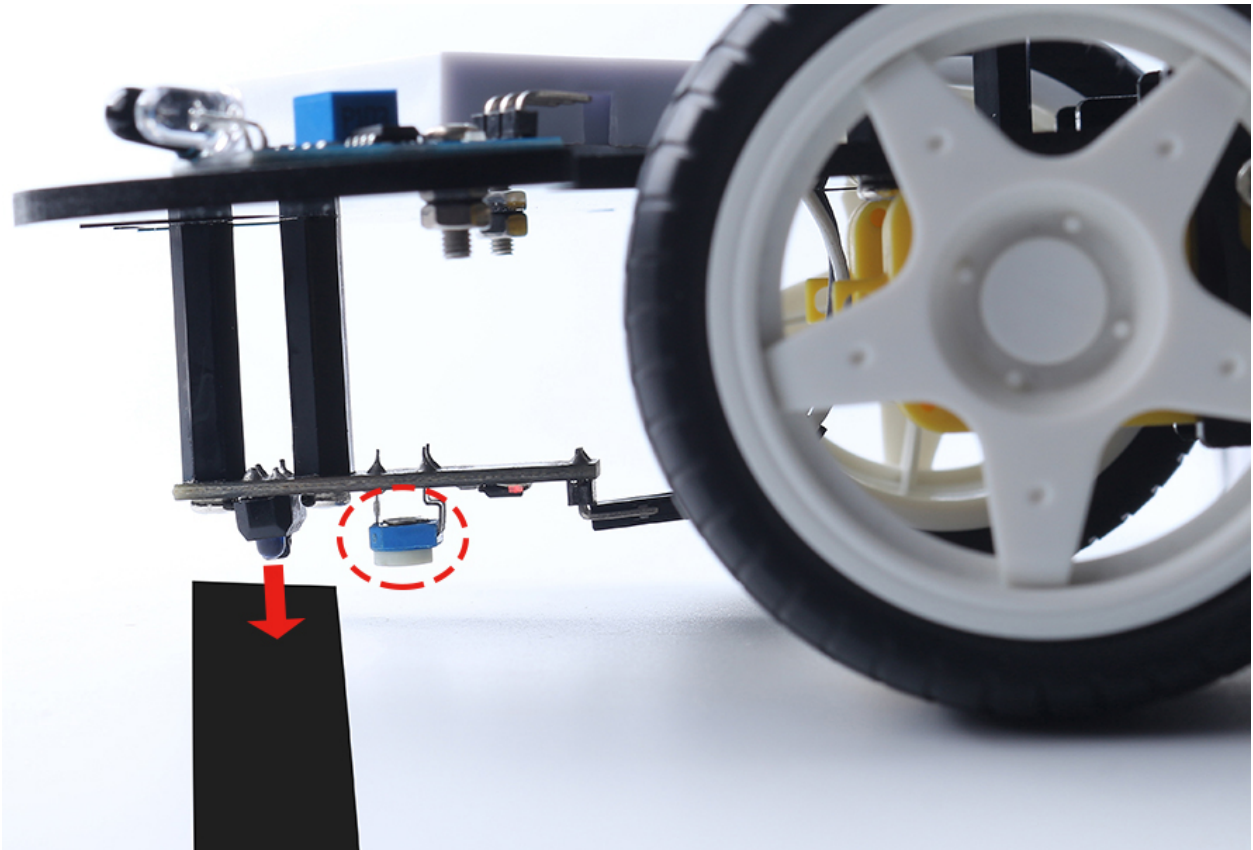
Before starting the project, you need to adjust the sensitivity of the module.

Wiring according to the above diagram, then power up the R4 board (either directly into the USB cable or the 9V battery button cable), without uploading the code.

Stick a black electrical tape on the table and put the cart on it.

Observe the signal LED on the module to make sure it lights up on the white table and goes off on the black tape.

If not, you need to adjust the potentiometer on the module, so that it can do the above effect.



## Code

---

### Note:

- Open the 4.follow\_the\_line.ino file under the path of 3in1-kit\car\_project\4.follow\_the\_line.
  - Or copy this code into **Arduino IDE**.
  - Or upload the code through the [Arduino Web Editor](#).
- 

After uploading the code to the R4 board, then align the Line Tracking module under the car with the black line, and you will see the car following the line.

### How it works?

In this code, it is letting the two motors micro-rotate left and right according to the value of the Line Track module so that you can see the car following the black line.

1. Add the pin definition for the Line Tracking module, here it is set to INPUT. Here also initialize the serial monitor and set the baud rate to 9600bps.

```
...  
const int lineTrack = 2;  
Serial.begin(9600);  
void setup() {  
    ...  
    pinMode(lineTrack, INPUT);  
}
```

2. Read the value of the Line Tracking module, if it is 1, then let the car go forward to the left; otherwise go forward to the right. Also you can open the serial monitor by clicking the magnifying glass icon in the upper right corner to see the change of the Line Tracking module value on the black and white line before unplugging the USB cable.

```
void loop() {

    int speed = 150;

    int lineColor = digitalRead(lineTrack); // 0:white    1:black
    Serial.println(lineColor);
    if (lineColor) {
        moveLeft(speed);
    } else {
        moveRight(speed);
    }
}
```

3. About the moveLeft() and moveRight() functions.

Unlike the left-right turn function in project 2. *Move by Code*, only small left-right turns are needed here, so you only need to adjust the value of A\_1A or B\_1B each time. For example, if you move to the left front (moveLeft()), you only need to set the speed to A\_1A and all others to 0, it will make the right motor turn clockwise and the left motor not move.

```
void moveLeft(int speed) {
    analogWrite(A_1B, 0);
    analogWrite(A_1A, speed);
    analogWrite(B_1B, 0);
    analogWrite(B_1A, 0);
}

void moveRight(int speed) {
    analogWrite(A_1B, 0);
    analogWrite(A_1A, 0);
    analogWrite(B_1B, speed);
    analogWrite(B_1A, 0);
}
```

- **Serial**

Used for communication between the Arduino board and a computer or other devices

- Serial.begin(): Sets the data rate in bits per second (baud) for serial data transmission.
- Serial.println(): Prints data to the serial port as human-readable ASCII text followed by a car return character (ASCII 13, or 'r') and a newline character (ASCII 10, or 'n').

- **if else**

The if else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped.

## 5.6 5. Play with Obstacle Avoidance Module

Two infrared obstacle avoidance modules are mounted on the front of the car, which can be used to detect some close obstacles.

In this project, the car is allowed to move forward freely, and when it encounters an obstacle it is able to avoid it and continue to move in other directions.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>Obstacle Avoidance Module</i>	

### Wiring

The obstacle avoidance module is a distance-adjustable infrared proximity sensor whose output is normally high and low when an obstacle is detected.

Now build the circuit according to the diagram below.

Left IR Module	R4 Board
OUT	8
GND	GND
VCC	5V

Right IR Module	R4 Board
OUT	7
GND	GND
VCC	5V

Before starting the project, you need to adjust the detection distance of the module.

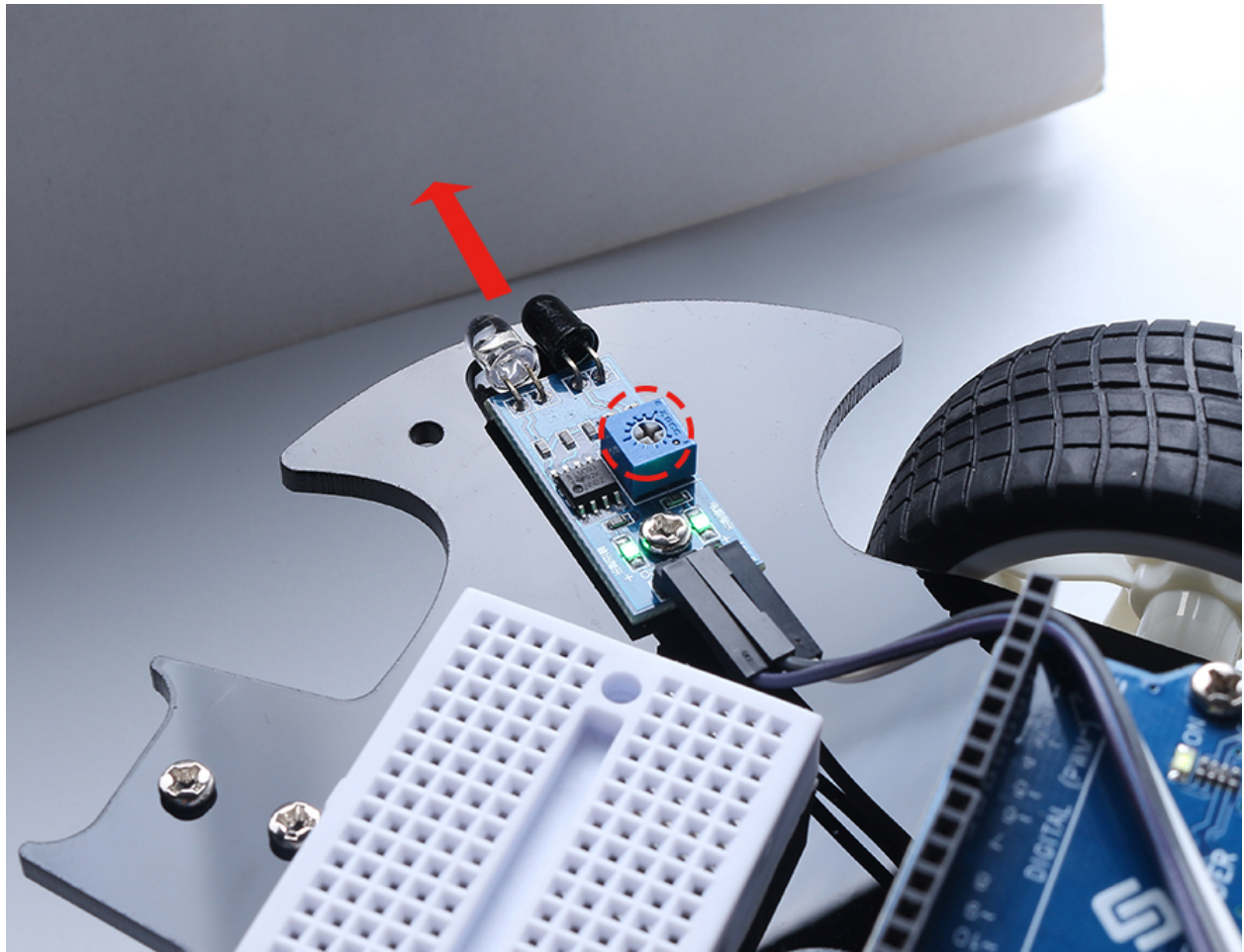
Wiring according to the above diagram, power up the R4 board (either by plugging in the USB cable directly or by snapping the 9V battery cable), without uploading the code.

Place a notebook or any other flat object about 5cm in front of the IR obstacle avoidance.

Then use a screwdriver to rotate the potentiometer on the module until the signal indicator on the module just lights up, so as to adjust its maximum detection distance of 5cm.

Follow the same method to adjust another infrared module.





### Code

#### Note:

- Open the 5.obstacle\_avoidance\_module.ino file under the path of 3in1-kit\car\_project\5.obstacle\_avoidance\_module.
- Or copy this code into **Arduino IDE**.
- Or upload the code through the [Arduino Web Editor](#).

The car will move forward once the code has been successfully uploaded. When the left infrared module detects an obstacle, it will go backwards to the left; when the right infrared module detects an obstacle, it will go backwards to the right; if both sides detect an obstacle, it will go backwards squarely.

#### How it works?

This project is based on the value of the left and right infrared obstacle avoidance modules to make the car make the appropriate action.

1. Add the pin definition for the 2 obstacle avoidance modules, here they are set to INPUT.

```
...  
const int rightIR = 7;  
const int leftIR = 8;
```

(continues on next page)



(continued from previous page)

```

void setup() {
  ...

  //IR obstacle
  pinMode(leftIR, INPUT);
  pinMode(rightIR, INPUT);
}

```

2. Read the values of the left and right infrared modules and let the car to make the corresponding action.

```

void loop() {

  int left = digitalRead(leftIR);  // 0: Obstructed  1: Empty
  int right = digitalRead(rightIR);
  int speed = 150;

  if (!left && right) {
    backLeft(speed);
  } else if (left && !right) {
    backRight(speed);
  } else if (!left && !right) {
    moveBackward(speed);
  } else {
    moveForward(speed);
  }
}

```

- If the left IR module is 0 (obstacle detected) and the right IR module is 1, let the car back up to the left.
- If the right IR module is 0 (obstacle detected), let the car go back up to the right.
- If 2 IR modules detect the obstacle at the same time, the car will go backward.
- Otherwise the car will keep going forward.

3. About the backLeft() function.

When the right motor is turning counterclockwise and the left motor is not turning, the car will go backward to the left.

```

void backLeft(int speed) {
  analogWrite(A_1B, speed);
  analogWrite(A_1A, 0);
  analogWrite(B_1B, 0);
  analogWrite(B_1A, 0);
}

```

4. About the backRight() function.

When the left motor is turning clockwise and the right motor is not turning, the car will go backward to the right.

```

void backRight(int speed) {
  analogWrite(A_1B, 0);

```

(continues on next page)

(continued from previous page)

```

analogWrite(A_1A, 0);
analogWrite(B_1B, 0);
analogWrite(B_1A, speed);
}

```

- `&&`: Logical AND results in true only if both operands are true.
- `!`: Logical NOT results in a true if the operand is false and vice versa.

## 5.7 6. Play with Ultrasonic Module

In *5. Play with Obstacle Avoidance Module* project, the 2 infrared obstacle avoidance modules are used for obstacle avoidance, but the detection distance of IR obstacle avoidance module is short, which may make the car too late to avoid the obstacles.

In this project, we use ultrasonic module to do some long-distance detection, so that the car can sense obstacles at a farther distance to make a judgment.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

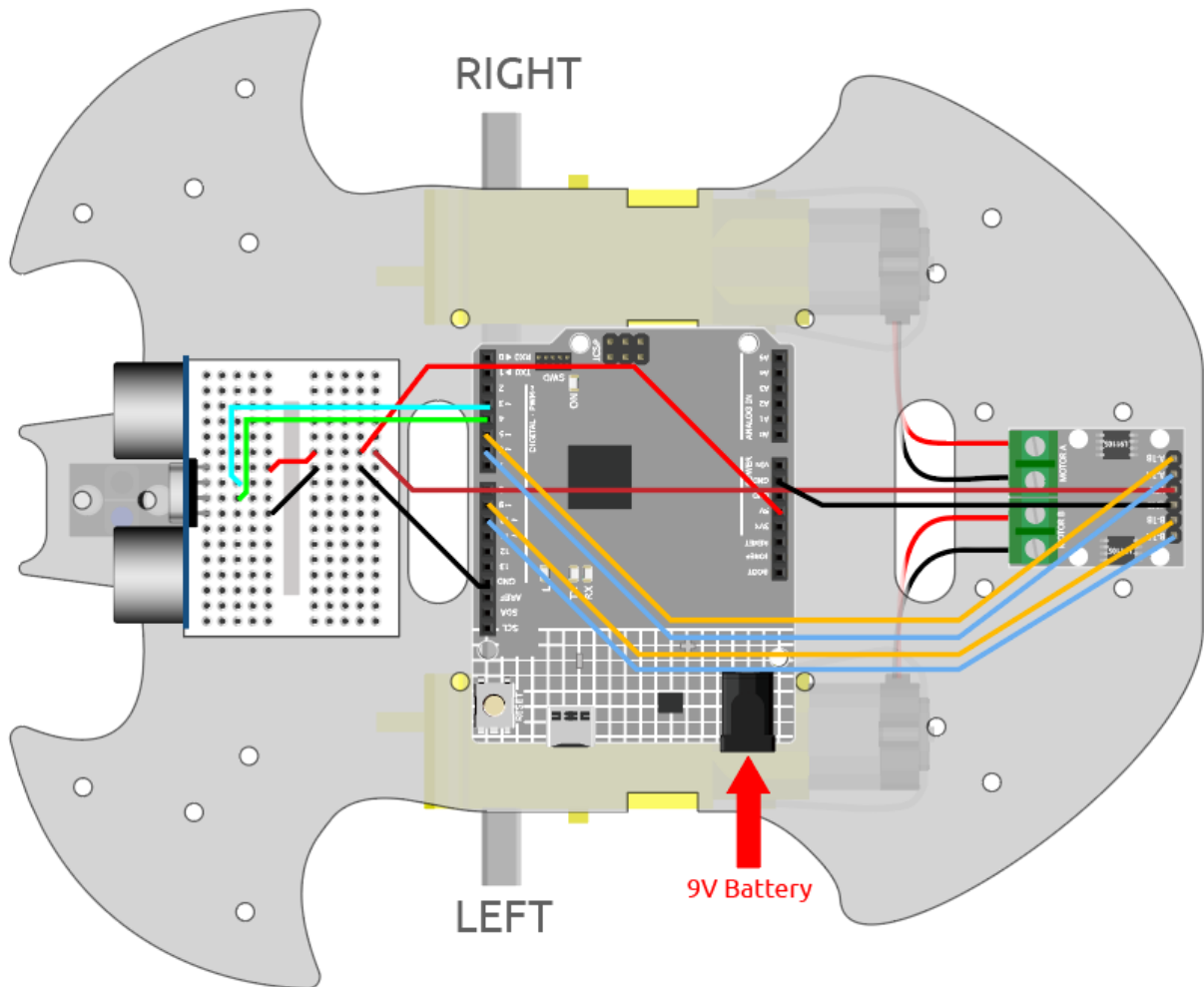
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>Ultrasonic Module</i>	

### Wiring

An ultrasonic sensor module is an instrument that measures the distance to an object using ultrasonic sound waves. It has two probes. One is to send ultrasonic waves and the other is to receive the waves and transform the time of sending and receiving into a distance, thus detecting the distance between the device and an obstacle.

Now build the circuit according to the following diagram.

Ultrasonic Module	R4 Board
Vcc	5V
Trig	3
Echo	4
Gnd	GND



## Code

### Note:

- Open the 6.ultrasonic\_module.ino file under the path of 3in1-kit\car\_project\6.ultrasonic\_module.
- Or copy this code into **Arduino IDE**.
- Or upload the code through the [Arduino Web Editor](#).

After the code is successfully uploaded, turn the car towards the wall. If the distance is too far, it will move forward; if it is too close, it will move backward; if it is at a safe distance, it will stop.

### How it works?

This project is based on the distance read by the ultrasonic module to make the car move accordingly.

1. Add the pin definition for the ultrasonic module, trigPin is used to transmit ultrasonic wave, so set it to OUTPUT; echoPin is set to INPUT to receive ultrasonic wave.

```
...
const int trigPin = 3;
```

(continues on next page)

(continued from previous page)

```

const int echoPin = 4;

void setup() {
  ...

  //ultrasonic
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
}

```

2. First read the distance value obtained from the ultrasonic module, if the distance is greater than 25, let the car move forward; if the distance is between 2-10cm, let the car move backward, otherwise (between 10~25) stop.

```

void loop() {
  float distance = readSensorData();
  if (distance > 25) {
    moveForward(200);
  }
  else if (distance < 10 && distance > 2) {
    moveBackward(200);
  } else {
    stopMove();
  }
}

```

3. About readSensorData() function.

The transmitter of the ultrasonic module transmits a 10us square wave signal every 2us, and the receiver receives a high level signal if there is an obstacle within the range. Use the pulseIn() function to record the time from sending to receiving, divide by the speed of sound 340m/s, and then divide by 2, the result is the distance between this module and the obstacle with units: cm.

```

float readSensorData() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  float distance = pulseIn(echoPin, HIGH) / 58.00; //Equivalent to (340m/
↪ S*1us)/2
  return distance;
}

```

- pulseIn(pin, value)
  - pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
  - value: type of pulse to read: either HIGH or LOW. Allowed data types: int.

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.

## 5.8 7. Follow Your Hand

Think of this car as your pet here, and when you will wave to him, it comes running to you.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>Ultrasonic Module</i>	
<i>Obstacle Avoidance Module</i>	

### Wiring

Connect the ultrasonic module and the 2 IR obstacle avoidance modules at the same time.

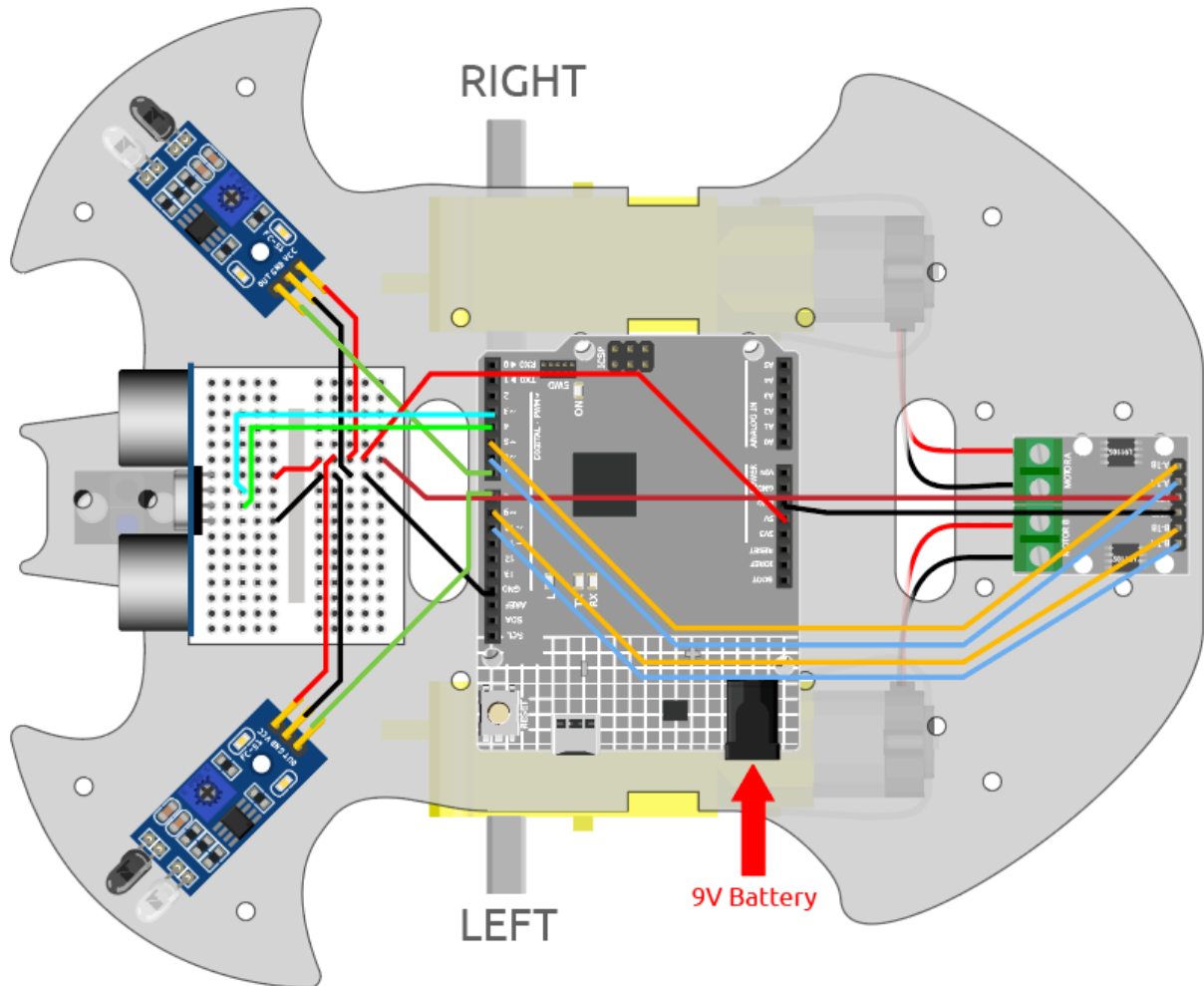
Wire the ultrasonic to the R4 board as follows.

Ultrasonic Module	R4 Board
Vcc	5V
Trig	3
Echo	4
Gnd	GND

The wiring of the 2 IR obstacle avoidance modules to the R4 board is as follows.

Left IR Module	R4 Board
OUT	8
GND	GND
VCC	5V

Right IR Module	R4 Board
OUT	7
GND	GND
VCC	5V



## Code

### Note:

- Open the `7. follow_your_hand.ino` file under the path of `3in1-kit\car_project\7. follow_your_hand`.
- Or copy this code into **Arduino IDE**.
- Or upload the code through the [Arduino Web Editor](#).

Place the car on the ground after the code has been uploaded successfully. Place your hand close to 5\*10cm in front of the car, and it will follow your hand forward. If you put your hand close to the IR Obstacle module on both sides, it will also turn to the corresponding direction.

### How it works?

This project is a combination of the previous two projects [6. Play with Ultrasonic Module](#) and [5. Play with Obstacle Avoidance Module](#), but the implemented effect is different. The previous 2 projects are detecting an obstacle backwards, but here it is detecting that your hand will follow the forward or turn direction. The workflow of this project is as follows.

- Read the distance detected by the ultrasonic module and the value of both infrared modules.
- If the distance is 5~10cm, let the car move with your hand.
- If the left IR module detects your hand, turn left.

- If the right IR module detects your hand, turn right.
- If neither the infrared module nor the ultrasonic module detects your hand, let the car stop.

```
void loop() {
    float distance = readSensorData();

    int left = digitalRead(leftIR);    // 0: Obstructed  1: Empty
    int right = digitalRead(rightIR);
    int speed = 150;

    if (distance>5 && distance<10){
        moveForward(speed);
    }
    if(!left&&right){
        turnLeft(speed);
    }else if(left&&!right){
        turnRight(speed);
    }else{
        stopMove();
    }
}
```

## 5.9 8. Self-Driving Car

This project is a combination of the two projects *6. Play with Ultrasonic Module* and *5. Play with Obstacle Avoidance Module*. 2 infrared obstacle avoidance modules do short distance or edge detection, and ultrasonic modules do long distance detection to confirm that the car does not hit an obstacle during the free driving process.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>Ultrasonic Module</i>	
<i>Obstacle Avoidance Module</i>	

### Wiring

Connect the ultrasonic module and the 2 IR obstacle avoidance modules at the same time.

Wire the ultrasonic to the R4 board as follows.

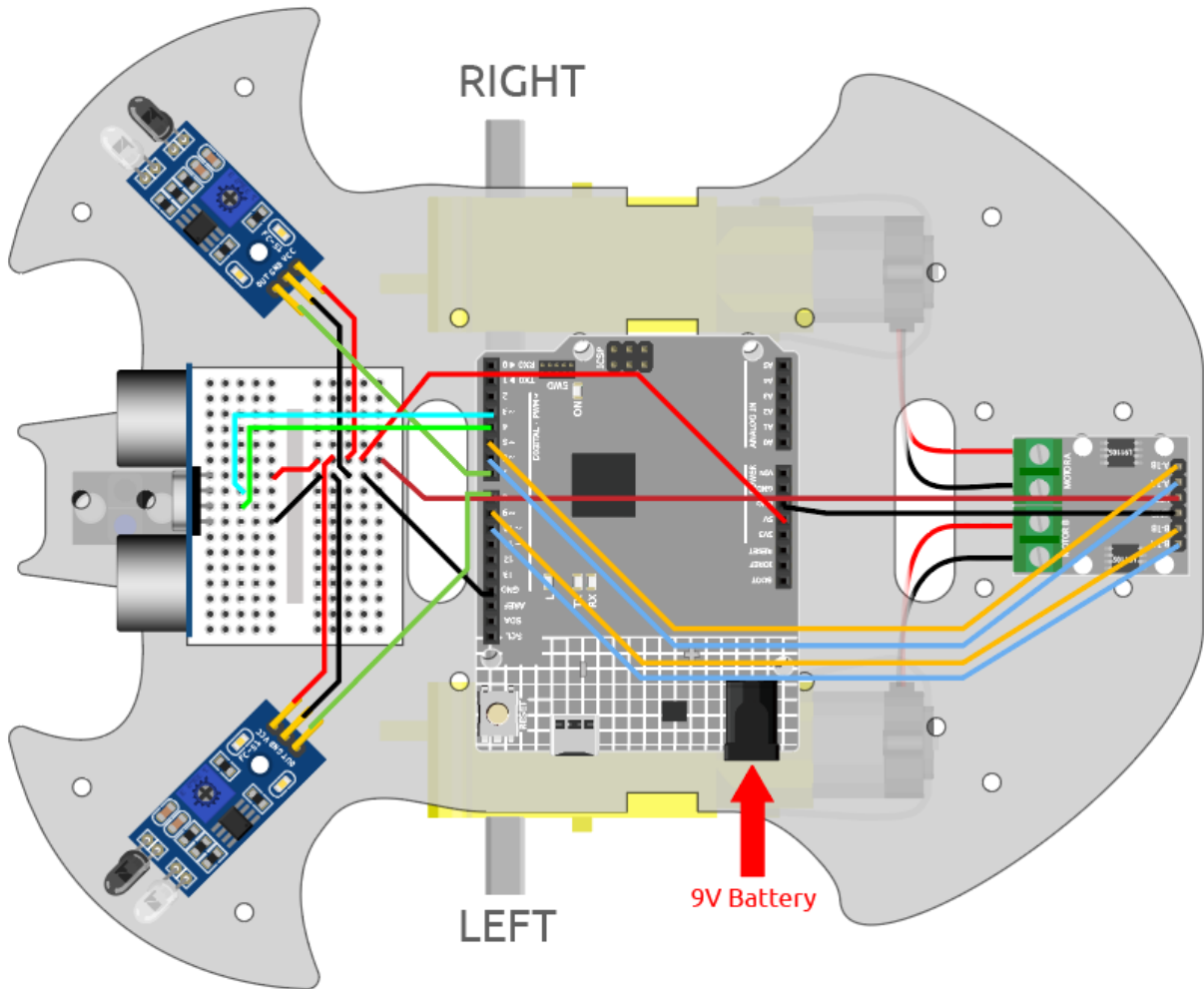
Ultrasonic Module	R4 Board
Vcc	5V
Trig	3
Echo	4
Gnd	GND

The wiring of the 2 IR obstacle avoidance modules to the R4 board is as follows.

Left IR Module	R4 Board
OUT	8
GND	GND
VCC	5V

Right IR Module	R4 Board
OUT	7
GND	GND
VCC	5V





## Code

### Note:

- Open the `8.self_driving_car.ino` file under the path of `3in1-kit\car_project\8.self_driving_car`.
- Or copy this code into **Arduino IDE**.
- Or upload the code through the [Arduino Web Editor](#).

The car will drive freely once the code has been uploaded successfully. When the IR obstruction module on both sides detects an obstacle, it will move in the opposite direction for emergency evasion; if there is an obstacle within 2~10cm directly in front of the car, it will back up to the left, adjust its direction, and then move forward.

### How it works?

The workflow of this project is as follows.

- Priority read the value of left and right IR obstacle avoidance module.
- If the left IR module is 0 (obstacle detected), the right IR module is 1, let the car back up to the left.
- If the right IR module is 0 (obstacle detected), let the car back up to the right.
- If 2 IR modules detect the obstacle at the same time, the car will back up.

- Otherwise read the distance detected by the ultrasonic module.
- If the distance is greater than 50cm, let the car go forward.
- If the distance is between 2-10cm, let the car backward before turning.
- If the distance is between 10-50cm, let the car go forward at low speed.

```
void loop() {

  int left = digitalRead(leftIR);  // 0: Obstructed  1: Empty
  int right = digitalRead(rightIR);

  if (!left && right) {
    backLeft(150);
  } else if (left && !right) {
    backRight(150);
  } else if (!left && !right) {
    moveBackward(150);
  } else {
    float distance = readSensorData();
    Serial.println(distance);
    if (distance > 50) { // Safe
      moveForward(200);
    } else if (distance < 10 && distance > 2) { // Attention
      moveBackward(200);
      delay(1000);
      backLeft(150);
      delay(500);
    } else {
      moveForward(150);
    }
  }
}
```

## 5.10 9. Remote Control

This kit comes with an IR receiver, which allows you to use an IR remote control to control the movement of the car.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

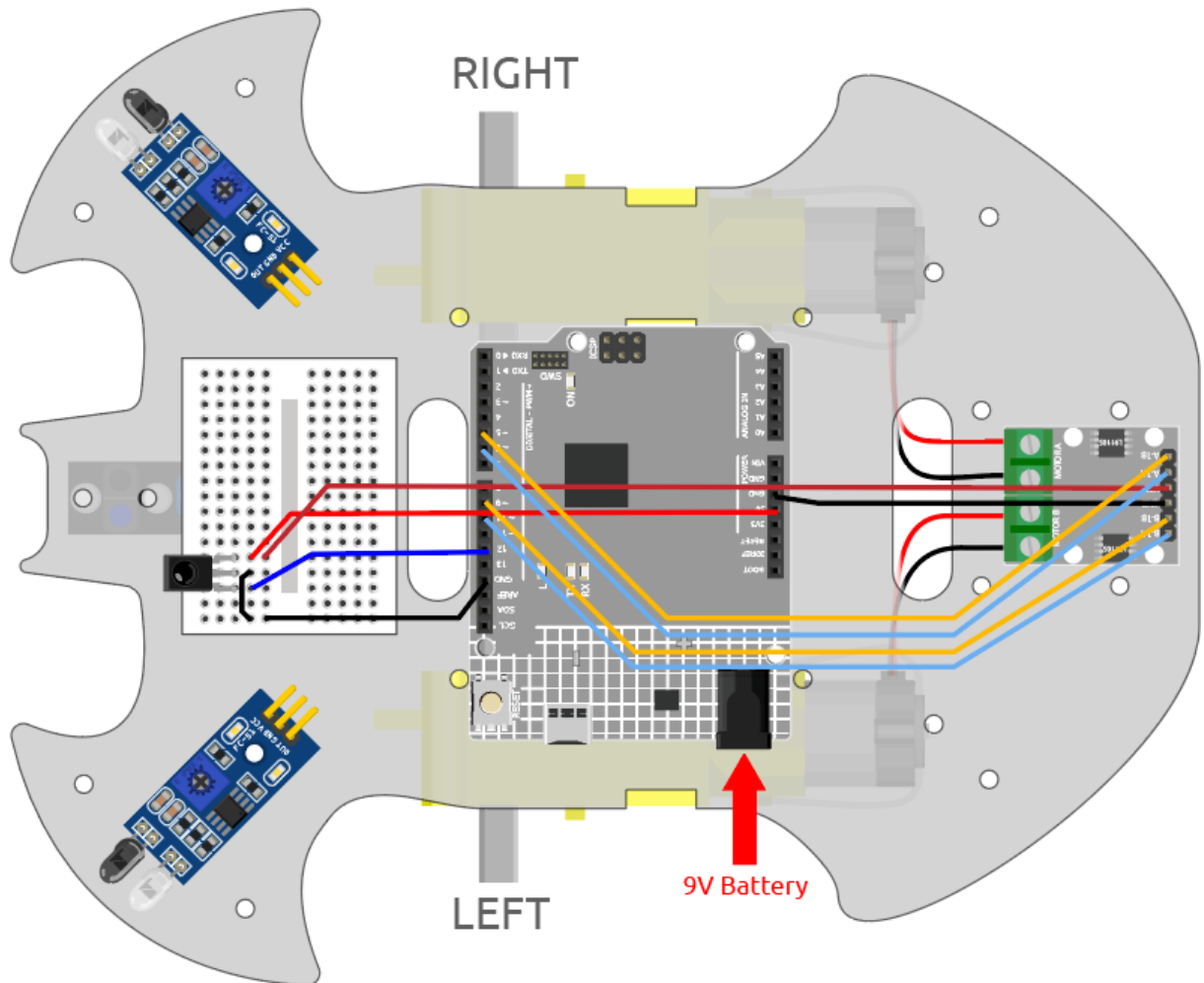
COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	
<i>L9110 Motor Driver Module</i>	-
<i>TT Motor</i>	-
<i>LED</i>	
<i>IR Receiver</i>	-

### Wiring

Now build the circuit according to the diagram below.

IR Receiver	R4 Board
OUT	12
GND	GND
VCC	5V

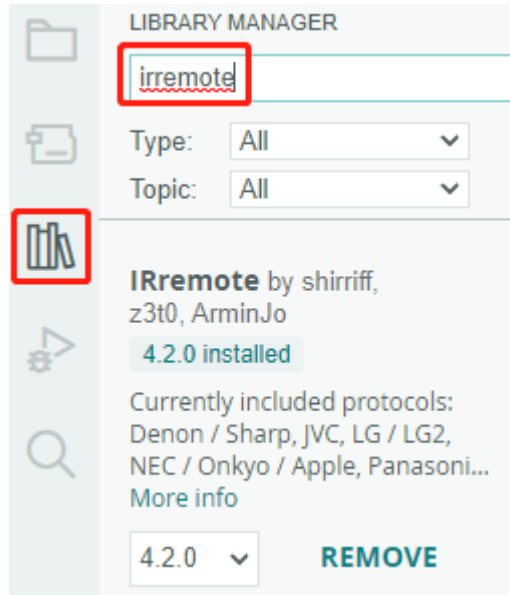
LED	R4 Board
Anode(The longer pin)	13
Cathode	GND



## Code

### Note:

- Open the `9.remote_control.ino` file under the path of `3in1-kit\car_project\9.remote_control`.
- Or copy this code into **Arduino IDE**.
- The `IRremote` library is used here, you can install it from the **Library Manager**.



After the code is uploaded successfully, press the button on the remote control, the LED will blink once to represent that the signal has been received, and the car will move according to the button you pressed. You can press the following keys to control the car.

- +: Accelerate
- -: Decelerate
- 1: Forward to the left
- 2: Forward
- 3: Forward to the right
- 4: Turn left
- 6: Turn right
- 7: Backward to the left
- 8: Backward
- 9: Backward to the right

#### How it works?

The effect of this project is to make the car move by reading the key value of the IR remote control. In addition, an LED is added to indicate that the IR signal has been successfully received.

1. Import the IRremote library, you can install it from the **Library Manager**.

```
#include <IRremote.h>

const int IR_RECEIVE_PIN = 12;  // Define the pin number for the IR Sensor
```

2. Initializes serial communication at a baud rate of 9600. Initializes the IR receiver on the specified pin (IR\_RECEIVE\_PIN) and enables LED feedback (if applicable).

```
...
```

(continues on next page)

(continued from previous page)

```

void setup() {
    ...
    //IR remote
    IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK); // Start the IR
    receiver // Start the receiver
    Serial.println("REMOTE CONTROL START");
}

```

3. When you press the keys on the remote control, the infrared receiver will know which key is pressed, and then the car will move according to the corresponding key value.

```

void loop() {
    if (IrReceiver.decode()) {
        // Serial.println(results.value, HEX);
        String key = decodeKeyValue(IrReceiver.decodedIRData.command);
        if (key != "ERROR") {
            Serial.println(key);

            if (key == "+") {
                speed += 50;
            } else if (key == "-") {
                speed -= 50;
            } else if (key == "2") {
                moveForward(speed);
                delay(1000);
            }
            ...
        }
        IrReceiver.resume(); // Enable receiving of the next value
    }
}

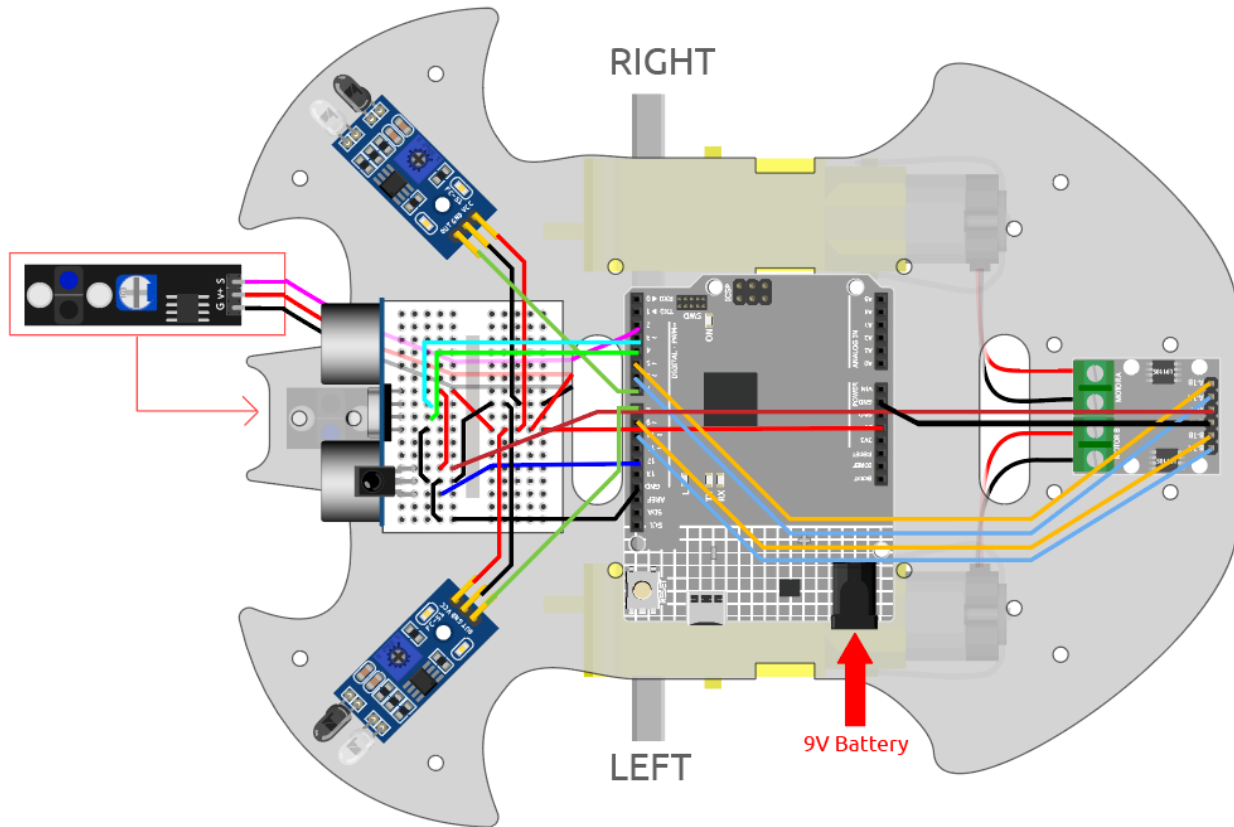
```

- Checks if an IR signal is received and successfully decoded.
- Decodes the IR command and stores it in `key` using a custom `decodeKeyValue()` function.
- Checks if the decoded value is not an error.
- Prints the decoded IR value to the serial monitor.
- Resumes IR signal reception for the next signal.

## 5.11 10. One Touch Start

In this project, we have integrated the previous projects - line following, following, obstacle avoidance, self-driving, etc. together. They can be switched by buttons on the remote control, so you can start the car and experience all functions at once.

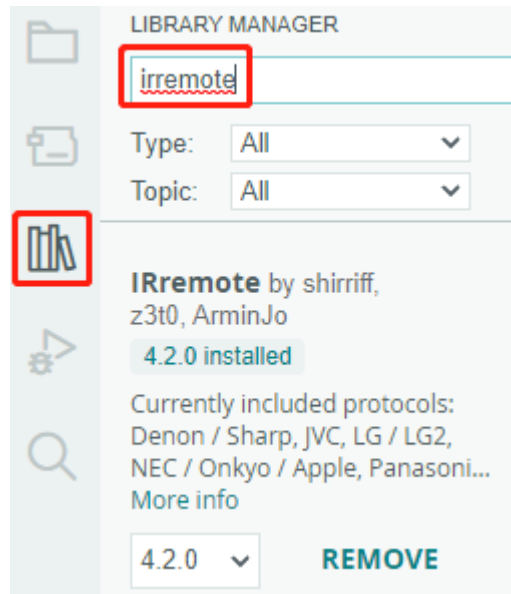
### Wiring



### Code

#### Note:

- Open the `10.one_touch_start.ino` file under the path of `3in1-kit\car_project\10.one_touch_start`.
- Or copy this code into **Arduino IDE**.
- The `IRremote` library is used here, you can install it from the **Library Manager**.



After the code is uploaded successfully, the LED will flash rapidly 3 times for every signal received from the remote control by the IR receiver. You can press the following keys to operate the cart.

- **+**: Accelerate
- **-**: Decelerate
- **1**: Move to the left front
- **2**: Forward
- **3**: Move to the right.
- **4**: Turn left
- **6**: Turn right
- **7**: Backward to the left
- **8**: Backward.
- **9**: Backward to the right
- **CYCLE**: Follow the line
- **U/SD**: Self-driving
- **|**: Obstacle avoidance with ultrasonic module
- **|**: Obstacle avoidance with IR Obstacle module
- **EQ**: Follow your hand
- **0**: Stop



## 5.12 11. Speed Calibration

In getting the car to move forward, you may find that the car does not travel in a straight line. This is because the two motors may not have the same speed at the factory. But we can write offset to the two motors to make their rotational speeds converge.

In this project, we will learn to store the offset into **EEPROM**, the point of this is that after each calibration, all projects can get the offset value directly from the EEPROM, so that the car can go in a straight line smoothly.

### Wiring

This project is the same wiring as 2. *Move by Code*.

### How to play?

1. Open the 11.speed\_calibration.ino file under the path of 3in1-kit\car\_project\11.speed\_calibration. Or copy this code into **Arduino IDE**.
2. After the code is uploaded successfully, connect the car with 9V battery, put it on the ground and let it move forward to see which side it is offset to.
  - If the car moves to the left front, it means the right motor speed is too fast and needs to be reduced.

```
EEPROM.write(1, 100) // 1 means the right motor, 100 means 100% speed, can
↳ be set to 90, 95, etc., depending on the actual situation.
```

- If the car moves to the right, it means the left motor speed is too fast and needs to be reduced.

```
EEPROM.write(0, 100) // 0 means the right motor, 100 means the speed is 100
↳ %, can be set to 90, 95, etc., depending on the actual situation. 3.
```

3. After modifying the code, upload the code to R4 board to see the effect. Repeat the above steps until the car is almost straight.
4. This offset will be recorded in **EEPROM**, you only need to read this offset when you use it in other projects, take 5. *Play with Obstacle Avoidance Module* as an example.

```
#include <EEPROM.h>

float leftOffset = 1.0;
float rightOffset = 1.0;

const int A_1B = 5;
const int A_1A = 6;
const int B_1B = 9;
const int B_1A = 10;

const int rightIR = 7;
const int leftIR = 8;

void setup() {
  Serial.begin(9600);

  //motor
  pinMode(A_1B, OUTPUT);
  pinMode(A_1A, OUTPUT);
  pinMode(B_1B, OUTPUT);
```

(continues on next page)

(continued from previous page)

```

pinMode(B_1A, OUTPUT);

//IR obstacle
pinMode(leftIR, INPUT);
pinMode(rightIR, INPUT);

leftOffset = EEPROM.read(0) * 0.01;//read the offset of the left motor
rightOffset = EEPROM.read(1) * 0.01;//read the offset of the right motor
}

void loop() {

    int left = digitalRead(leftIR);    // 0: Obstructed  1: Empty
    int right = digitalRead(rightIR);
    int speed = 150;

    if (!left && right) {
        backLeft(speed);
    } else if (left && !right) {
        backRight(speed);
    } else if (!left && !right) {
        moveBackward(speed);
    } else {
        moveForward(speed);
    }
}

void moveForward(int speed) {
    analogWrite(A_1B, 0);
    analogWrite(A_1A, int(speed * leftOffset));
    analogWrite(B_1B, int(speed * rightOffset));
    analogWrite(B_1A, 0);
}

void moveBackward(int speed) {
    analogWrite(A_1B, speed);
    analogWrite(A_1A, 0);
    analogWrite(B_1B, 0);
    analogWrite(B_1A, speed);
}

void backLeft(int speed) {
    analogWrite(A_1B, speed);
    analogWrite(A_1A, 0);
    analogWrite(B_1B, 0);
    analogWrite(B_1A, 0);
}

void backRight(int speed) {
    analogWrite(A_1B, 0);
    analogWrite(A_1A, 0);
    analogWrite(B_1B, 0);
}

```

(continues on next page)

(continued from previous page)

```
    analogWrite(B_1A, speed);  
}
```

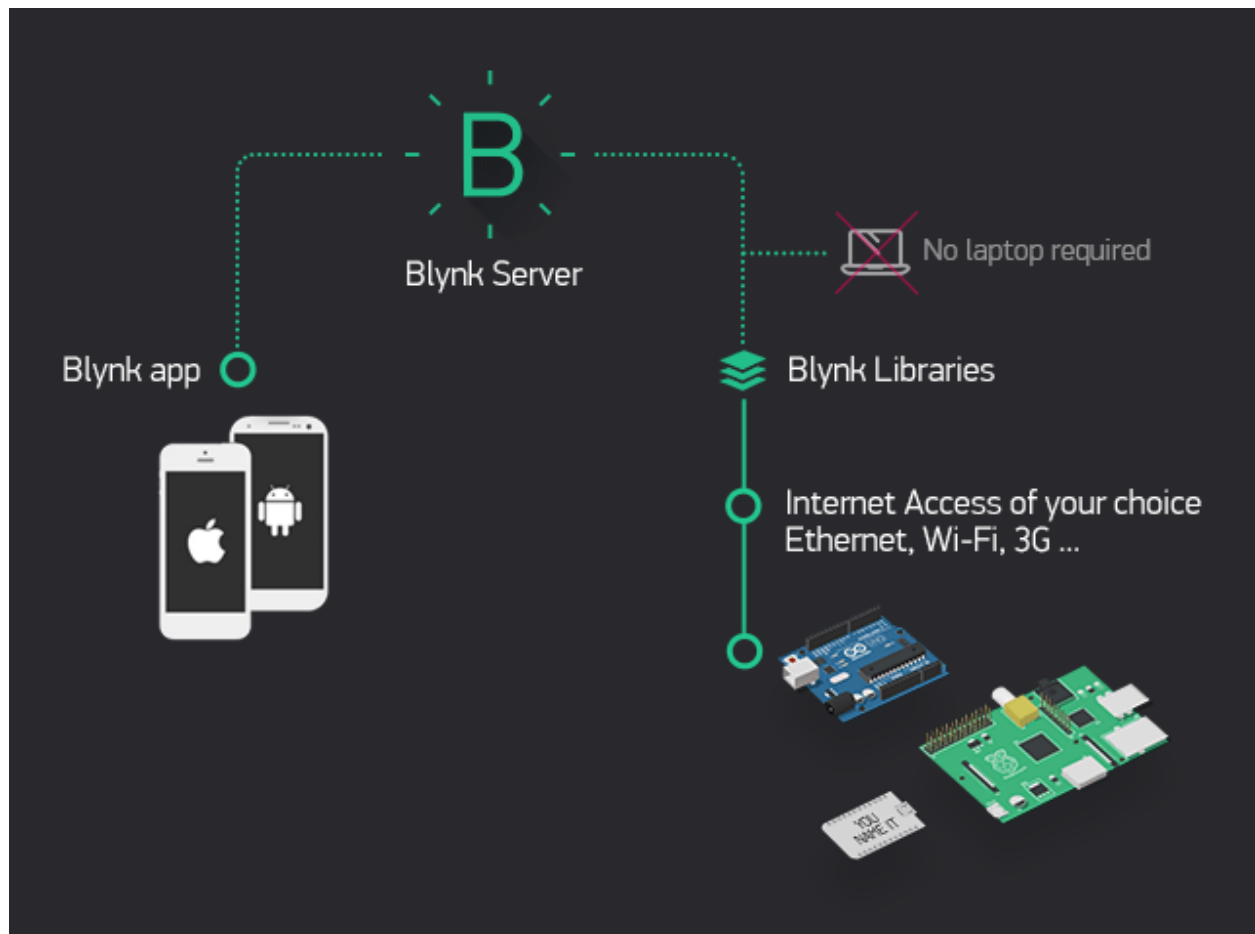


## IOT PROJECTS

This kit has the ESP8266 Wifi module which allows Arduino to connect to the internet for IoT experiments.

Here we will guide you how to make Arduino connect to [BLYNK](#) platform with the help of ESP8266 Wifi module to do some interesting IoT projects. Also you can use the Blynk APP on your cell phone to control the smart car.

Blynk is a full suite of software required to prototype, deploy, and remotely manage connected electronic devices at any scale: from personal IoT projects to millions of commercial connected products. With Blynk anyone can connect their hardware to the cloud and build a no-code iOS, Android, and web applications to analyze real-time and historical data coming from devices, control them remotely from anywhere in the world, receive important notifications, and much more...



## 6.1 1. Get Started with Blynk

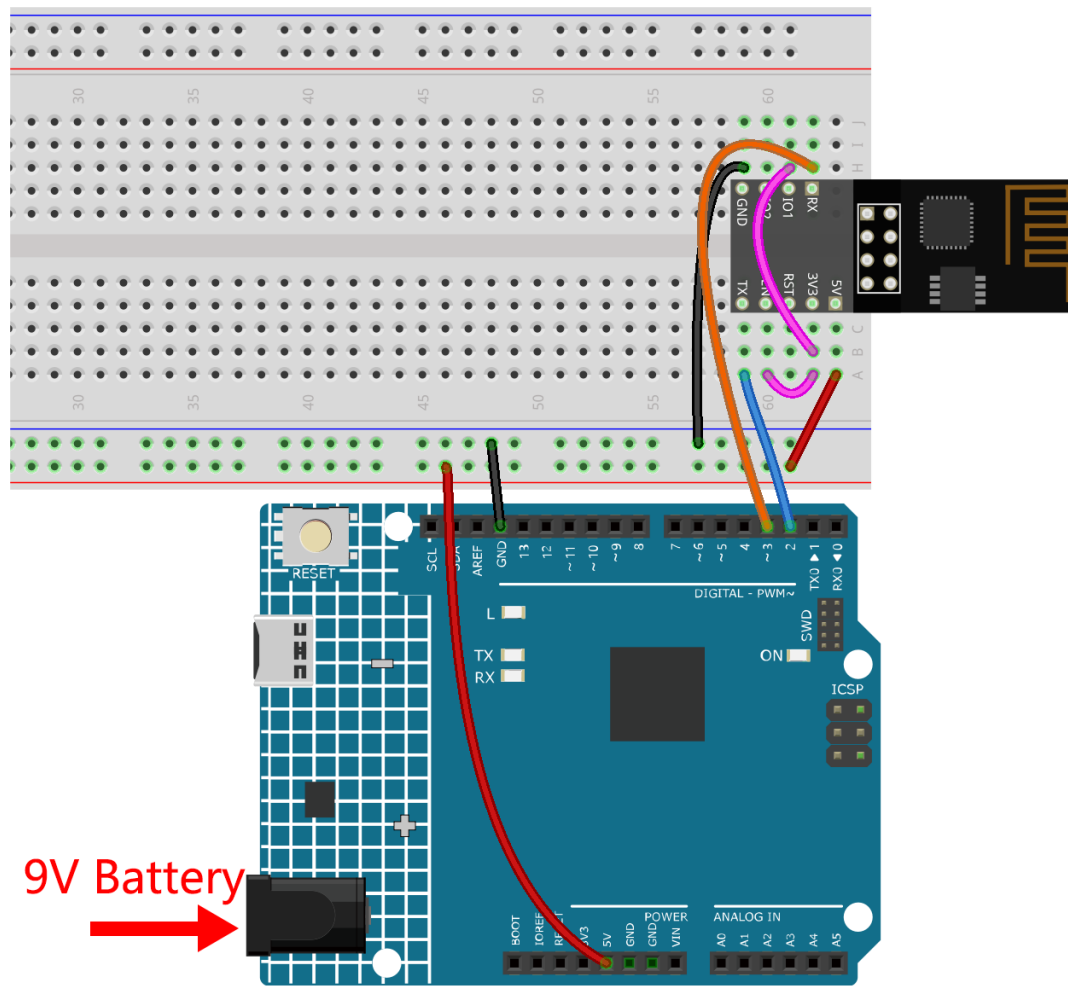
Getting the R4 board to communicate with Blynk requires some configuration when you first use Blynk.

Follow the steps below, and note that you must do them in order and not skip any chapters.

### 6.1.1 1.1 Configuring the ESP8266

The ESP8266 module that comes with the kit is already pre-burned with AT firmware, but you still need to modify its configuration by following the steps below.

1. Build the circuit.



2. Open the 1.set\_software\_serial.ino file under the path of 3in1-kit\iot\_project\1.set\_software\_serial. Or copy this code into **Arduino IDE**.

```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(2, 3); //Rx,Tx

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}
```

(continues on next page)

(continued from previous page)

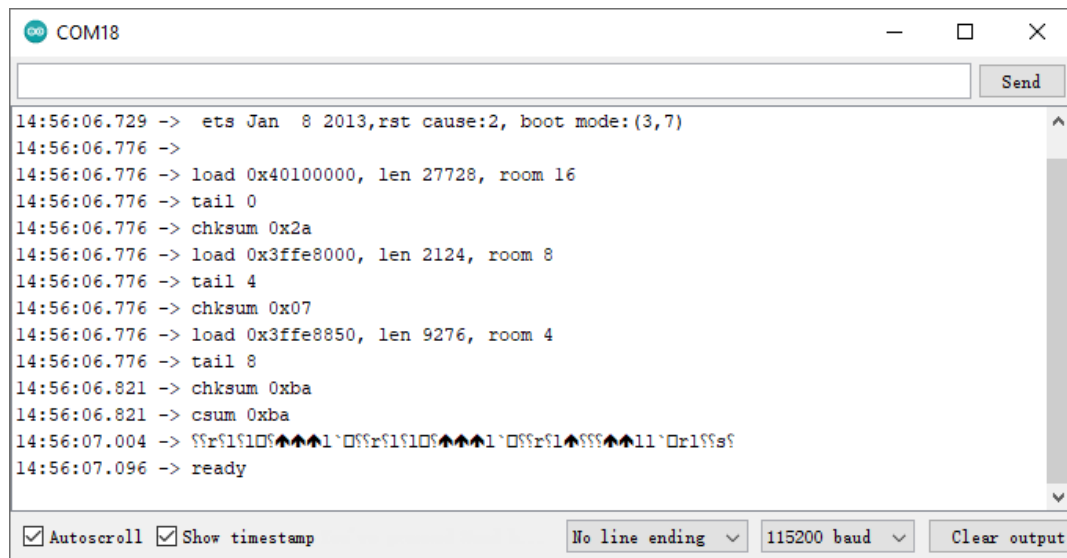
```

    espSerial.begin(115200);
}

void loop() {
    if (espSerial.available()) {
        Serial.write(espSerial.read());
    }
    if (Serial.available()) {
        espSerial.write(Serial.read());
    }
}

```

3. Click the magnifying glass icon (Serial Monitor) in the upper right corner and set the baud rate to **115200**. (You may have some printed information like me, or you may not, it doesn't matter, just go to the next step.)

**Warning:**

- If ready doesn't appear, you can try to reset the ESP8266 module(connect RST to GND) and re-open the Serial Monitor.
- In addition, if the result is OK, you may need to re-burn the firmware, please refer to [How to re-burn the firmware for ESP8266 module?](#) for details. If you still can't solve it, please take a screenshot of the serial monitor and send it to [sevice@sunfounder.com](mailto:sevice@sunfounder.com), we will help you solve the problem as soon as possible.

4. Click on **NEWLINE DROPDOWN BOX**, select both NL & CR in the drop down option, enter AT, if it returns OK, it means ESP8266 has successfully established connection with your board.

```

COM18
AT2
14:56:06.776 -> load 0x40100000, len 27728, room 16
14:56:06.776 -> tail 0
14:56:06.776 -> chksum 0x2a
14:56:06.776 -> load 0x3ffe8000, len 2124, room 8
14:56:06.776 -> tail 4
14:56:06.776 -> chksum 0x07
14:56:06.776 -> load 0x3ffe8850, len 9276, room 4
14:56:06.776 -> tail 8
14:56:06.821 -> chksum 0xba
14:56:06.821 -> csum 0xba
14:56:07.004 -> $$$r$1$10$####1`0$$r$1$10$####1`0$$r$1$10$####1`0r1$$s$
14:56:07.096 -> ready
15:00:05.119 -> AT
15:00:05.119 ->
15:00:05.119 -> OK
☒ Autoscroll ☒ Show timestamp Both NL & CR 115200 baud Clear output

```

5. Enter AT+CWMODE=3 and the managed mode will be changed to **Station and AP** coexistence.

```

COM18
AT+CWMODE=3
14:56:06.776 -> tail 4
14:56:06.776 -> chksum 0x07
14:56:06.776 -> load 0x3ffe8850, len 9276, room 4
14:56:06.776 -> tail 8
14:56:06.821 -> chksum 0xba
14:56:06.821 -> csum 0xba
14:56:07.004 -> $$$r$1$10$####1`0$$r$1$10$####1`0$$r$1$10$####1`0r1$$s$
14:56:07.096 -> ready
15:00:05.119 -> AT
15:00:05.119 ->
15:00:05.119 -> OK
15:01:08.226 -> AT+CWMODE=3
15:01:08.319 ->
15:01:08.319 -> OK
☒ Autoscroll ☒ Show timestamp Both NL & CR 115200 baud Clear output

```

6. In order to use the software serial later, you must input AT+UART=9600,8,1,0,0 to modify the ESP8266's baud rate to 9600.



```

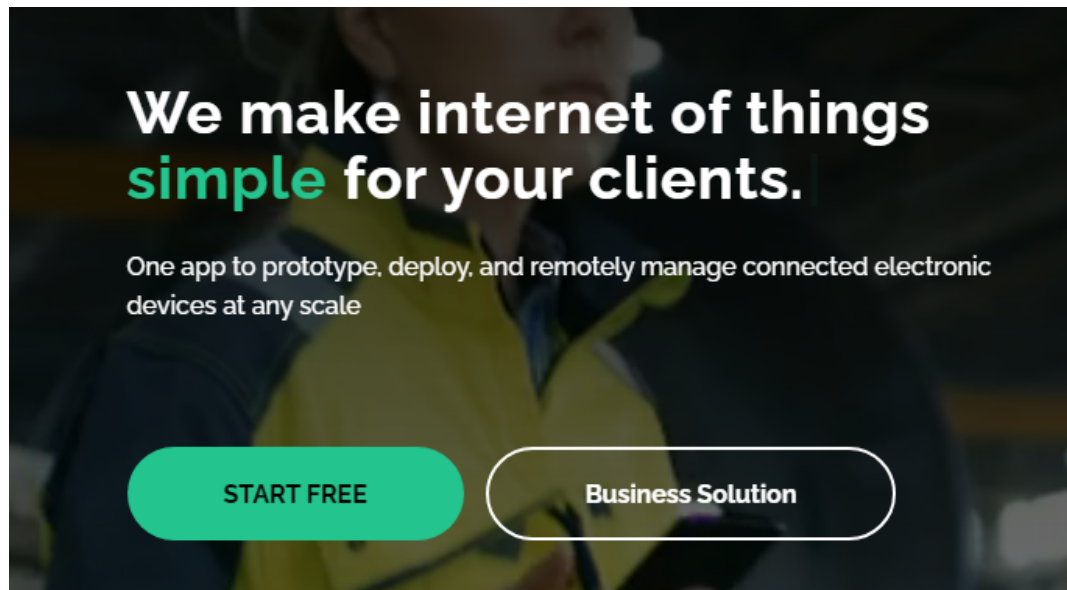
COM18
AT+UART=9600,8,1,0,0
14:56:06.776 -> tail 8
14:56:06.821 -> chksum 0xba
14:56:06.821 -> csum 0xba
14:56:07.004 -> ??r$1$10$1111`0$?r$1$10$1111`0$?r$111111`0r1$1$1
14:56:07.096 -> ready
15:00:05.119 -> AT
15:00:05.119 ->
15:00:05.119 -> OK
15:01:08.226 -> AT+CWMODE=3
15:01:08.319 ->
15:01:08.319 -> OK
15:03:06.853 -> AT+UART=9600,8,1,0,0
15:03:06.853 ->
15:03:06.853 -> OK

```

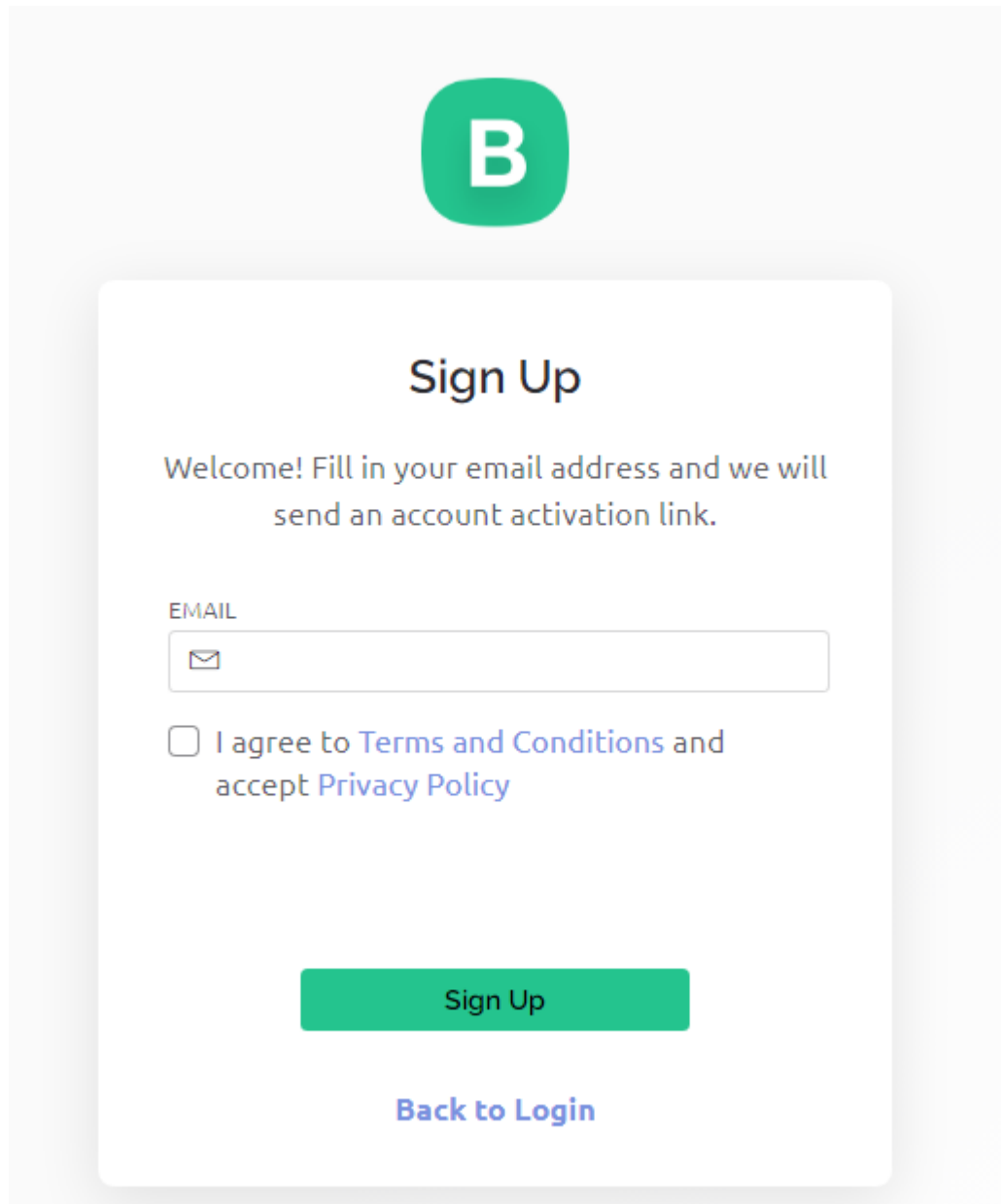
☒ Autoscroll
 ☒ Show timestamp
 Both NL & CR
 115200 baud
 Clear output

### 6.1.2 1.2 Configuring the Blynk

1. Go to the [BLYNK](#) and click **START FREE**.



2. Fill in your email address to register an account.



The image shows a 'Sign Up' form for a platform with a green circular logo containing a white letter 'B'. The form is centered on a light gray background. It includes a title 'Sign Up', a welcome message, an email input field with an envelope icon, a checkbox for terms and conditions, a green 'Sign Up' button, and a blue 'Back to Login' link.

**B**

## Sign Up

Welcome! Fill in your email address and we will send an account activation link.

EMAIL

☐ I agree to [Terms and Conditions](#) and accept [Privacy Policy](#)

**Sign Up**

[Back to Login](#)

3. Go to your email address to complete your account registration.

Welcome!

We're excited to see you on board.

To get started, you'll need to create a password for your account.

Create Password

The link will expire in 30 days.

4. Afterwards, **Blynk Tour** will appear and you can read it to learn the basic information about the Blynk.

### Blynk Tour

1 Welcome 2 Platform 3 Modes 4 Devices 5 Template 6 Template components 7 Features 8 Business

#### Hi Blynker!

You've just joined the community of more than 500,000+ developers building amazing IoT products and projects.

With Blynk you can connect your devices to the Internet and create mobile and web dashboards to control your devices from anywhere in the world.

Let's save your learning time with a few quick steps.



Skip

Let's go!

5. Next we need to create a template and device with this **Quick Start**, click **Let's go**.

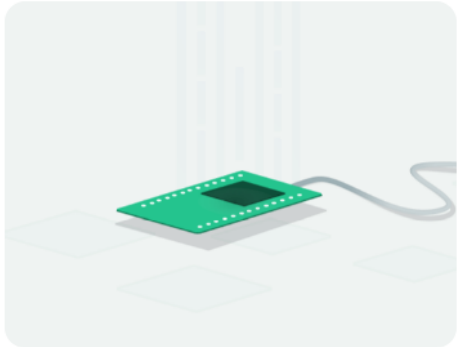
## Quickstart

This is a step by step guide to get your first device online and start controlling it from anywhere in the world in **less than 5 minutes**

**What you will need:**

- Supported hardware. Check the full list of supported hardware [here](#).
- IDE. You can use Arduino IDE or PlatformIO or any other editor.
- Blynk Library
- It will be beneficial if you already know how to upload code to your hardware.

CancelLet's go!



6. Select the hardware and connection type.

## Quickstart

1

 Hardware — 

2

 IDE — 

3

 Blynk Library — 

4

 Code — 

5

 Device activation

**Which hardware are you using?**

We will help you prepare the code for you board

ESP8266

▼

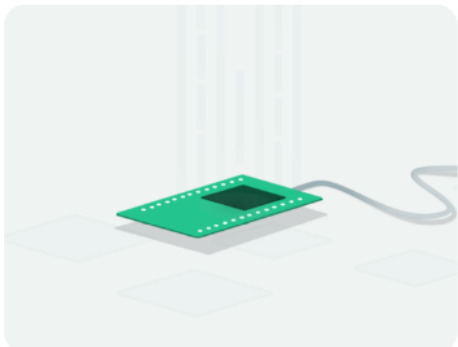
**What is your device connectivity type**

Blynk supports various connection types (BLE is not supported yet).

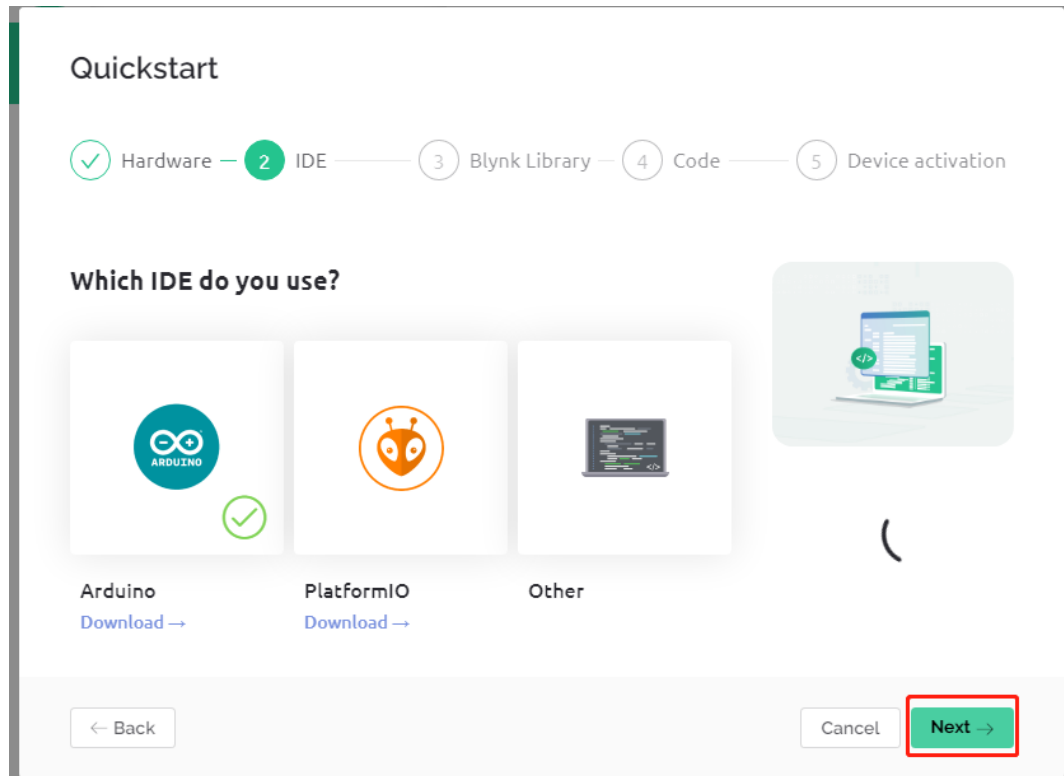
WiFi

▼

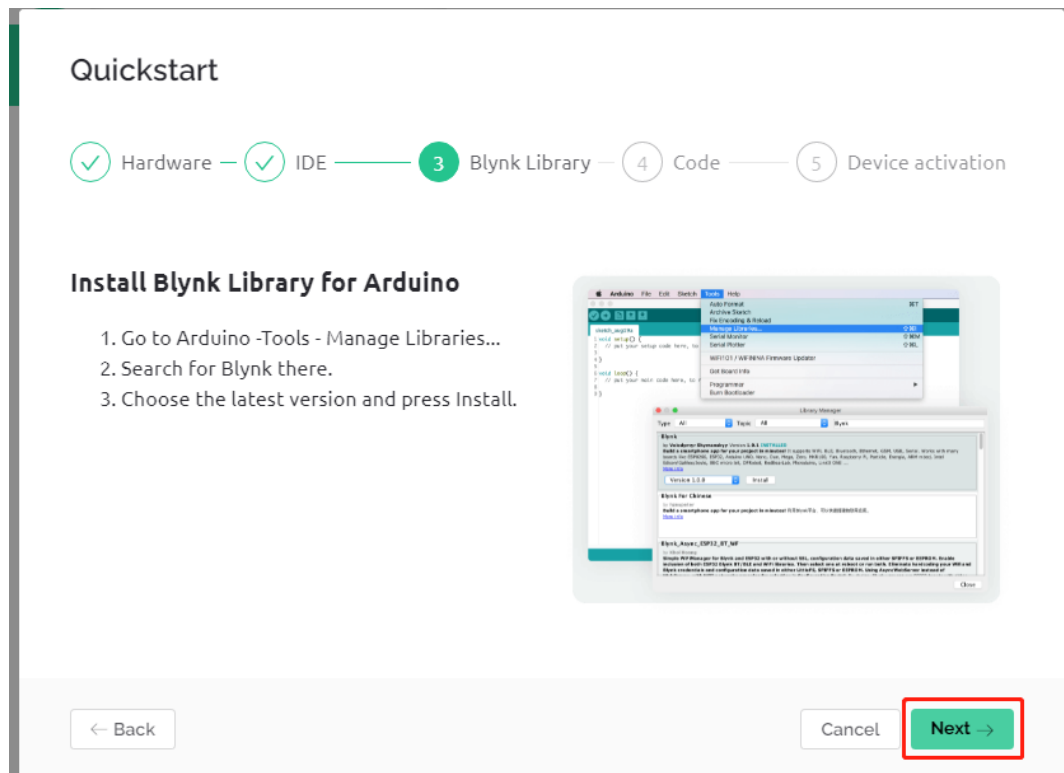
CancelNext →



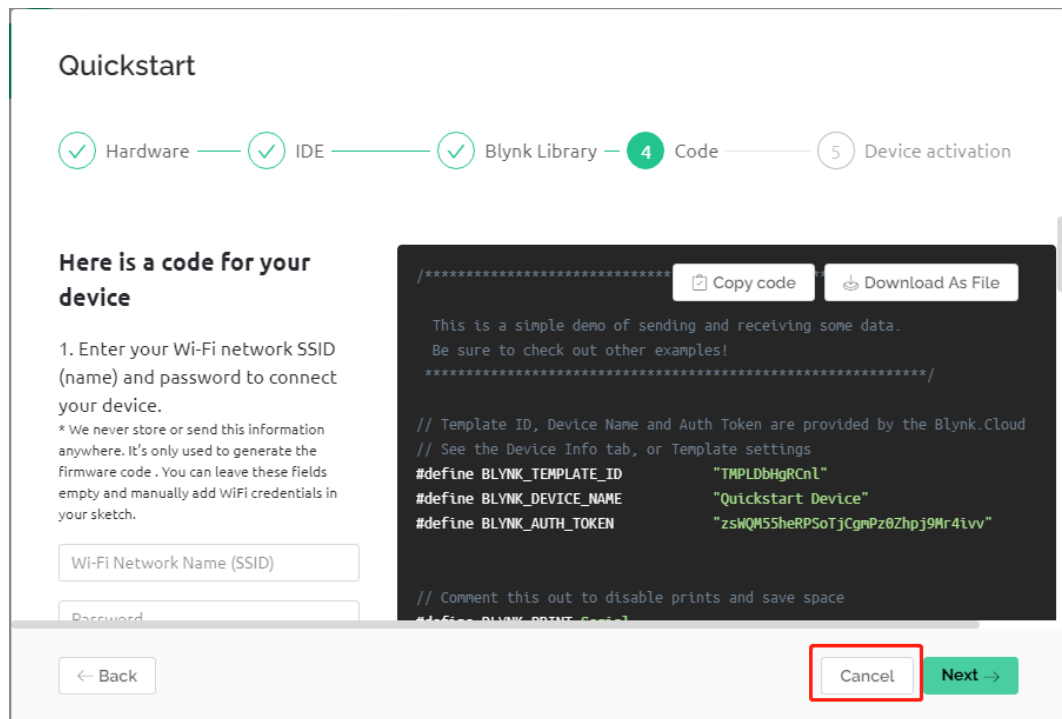
7. Here you are told which IDE you need to prepare, we recommend the **Arduino IDE**.



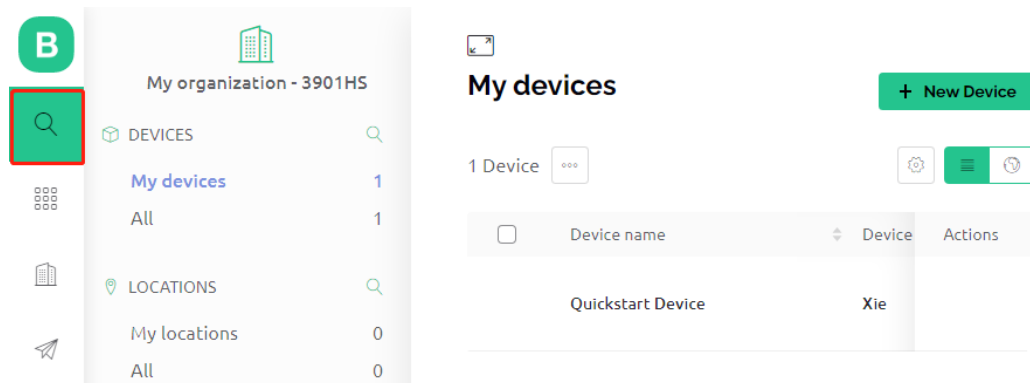
8. Here is the library you need to add, but the recommended library here is a bit problematic, we need to add other libraries manually (we will mention it later). Click **Next** here, and a new template and device will be created.



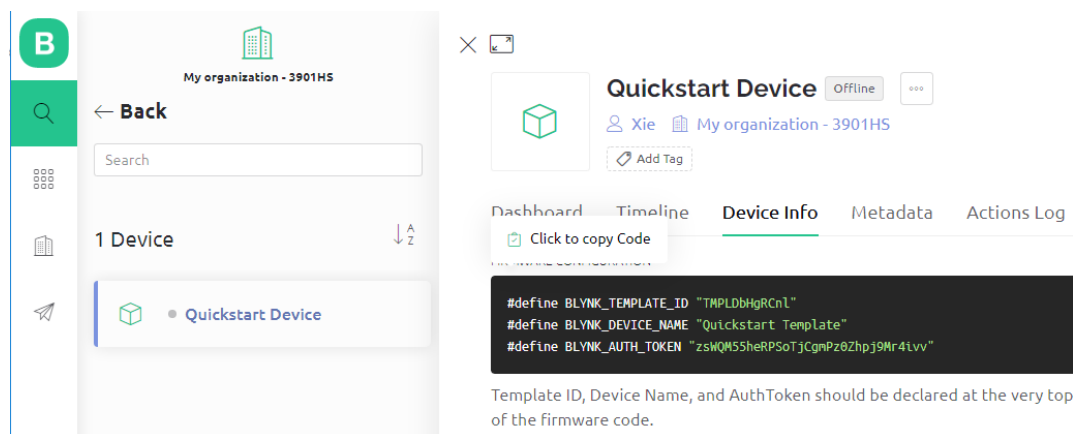
9. The next steps are to upload the relevant code and connect your board to Blynk, but since there is a problem with the library provided earlier, you need to add other libraries again. So click **Cancel** here to stop **Quick Start**.



10. Click the **Search** button and you will see the new device you just created.



11. Go to this **Quickstart Device** and you will see TEMPLATE\_ID, DEVICE\_NAME and AUTH\_TOKEN on the **Device info** page, and you will need to copy them later.






### 6.1.3 1.3 Adding the required libraries

You need to add the correct libraries for the Arduino IDE to use Blynk.

1. Click [HERE](#), scroll down to the bottom of the page and download the first .zip file.

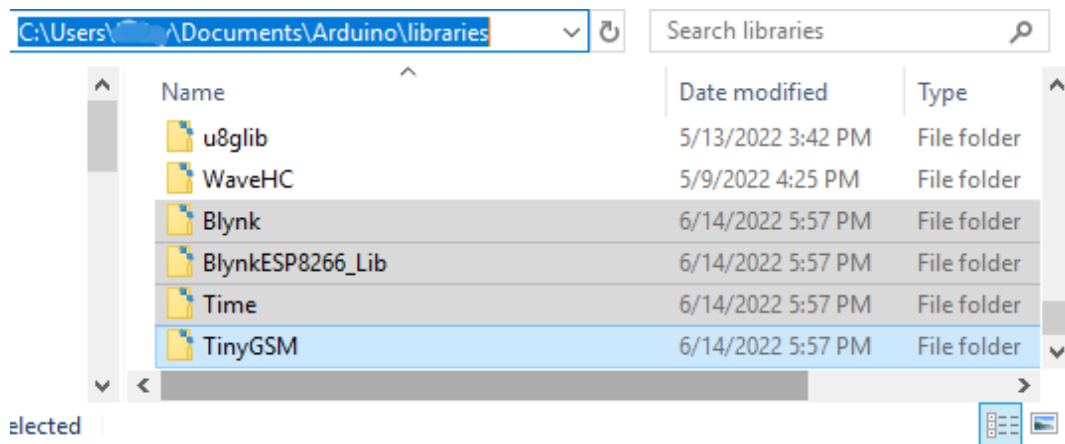
#### ▼ Assets 3

 Blynk_Release_v1.1.0.zip	779 KB	22 days ago
 Source code (zip)		22 days ago
 Source code (tar.gz)		22 days ago

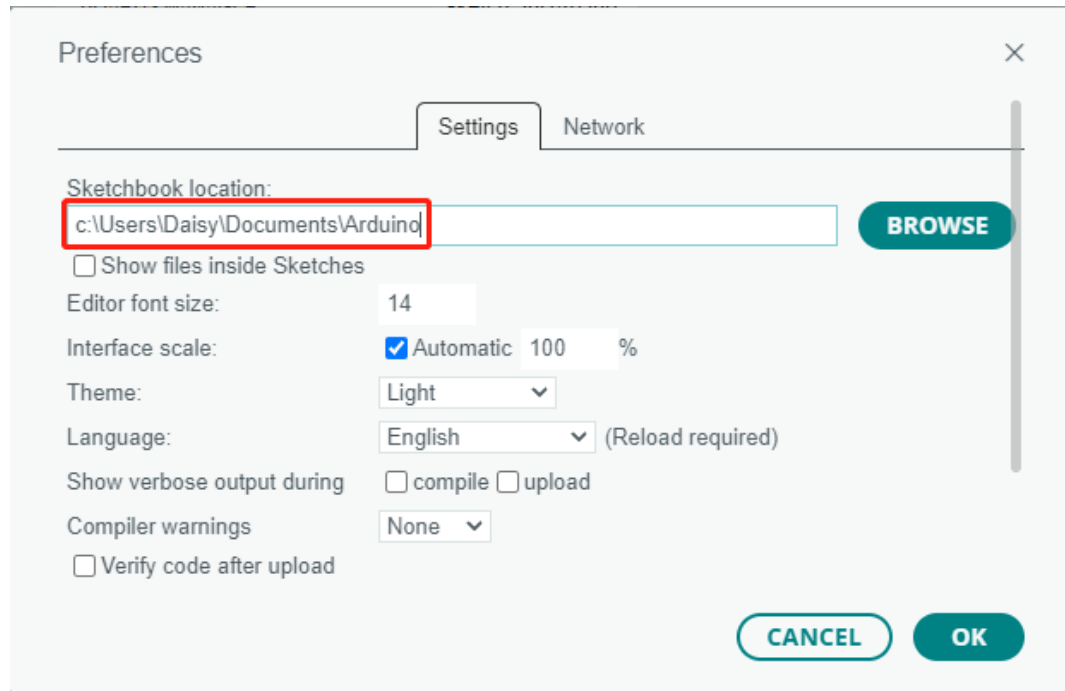
2. Unzip this file and you can see the following folders.

..			File folder	
Blynk	1,060,482	459,412	File folder	7/16/2021 1:13 ...
BlynkESP8266_Lib	57,090	11,208	File folder	5/25/2021 5:12 ...
Time	63,774	25,512	File folder	5/25/2021 5:12 ...
TinyGSM	602,241	172,967	File folder	5/25/2021 5:12 ...

3. Copy them all and paste them to the default libraries directory of the Arduino IDE, which is usually located at C:\Users\xxx\Documents\Arduino\libraries.



4. If your libraries directory is different, you can check it by going to **File -> Preferences**.



### 6.1.4 1.4 Connecting the R4 board to Blynk

1. Reconnect the ESP8266 module and R4 board, here the software serial is used, so TX and RX are connected to pins 2 and 3 of R4 board respectively.

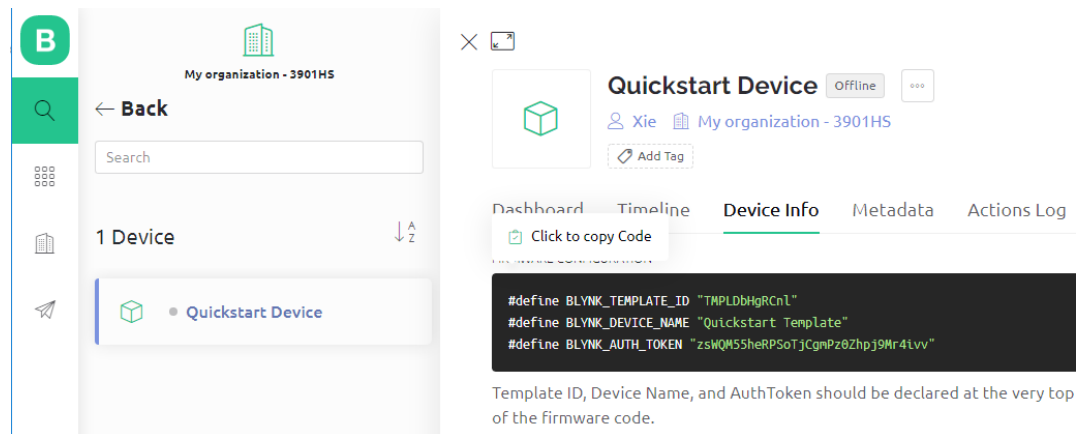
---

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.

---



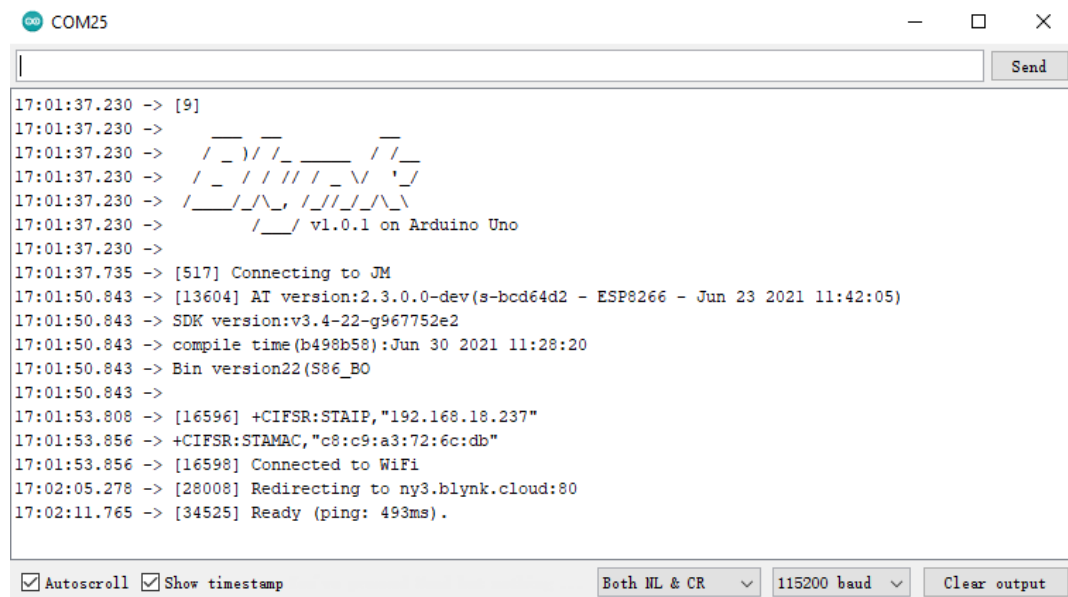




4. Fill in the ssid and password of the WiFi you are using.

```
char ssid[] = "ssid";
char pass[] = "password";
```

5. Upload the code to the R4 board, then open the serial monitor and set the baud rate to 115200. when the R4 board communicates with Blynk successfully, the serial monitor will show the ready character.

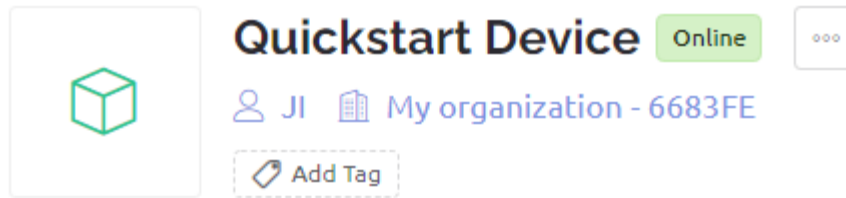


**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

6. The status of Blynk will change from **offline** to **online**.



## 6.2 2. Get Data from Blynk

You will learn how to control the circuit with Blynk in this chapter. Let's light up the LEDs over the Internet!

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

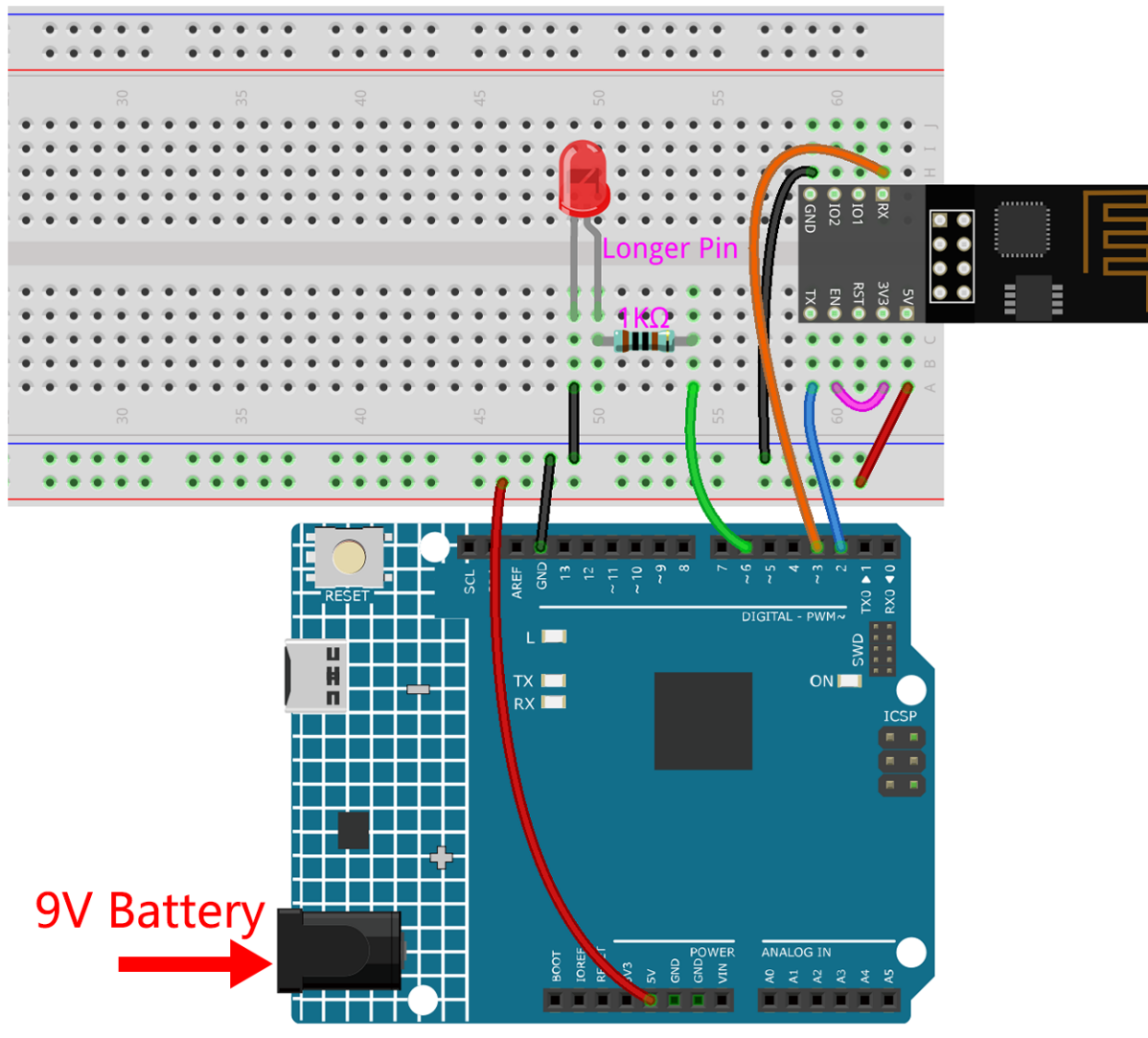
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>ESP8266 Module</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>LED</i>	

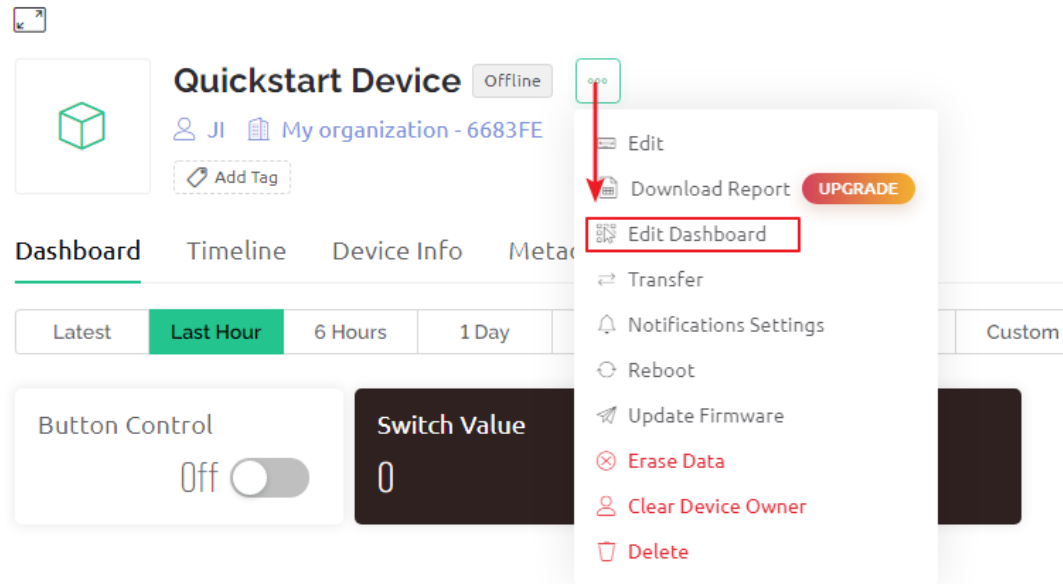
### 1. Build the Circuit

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.

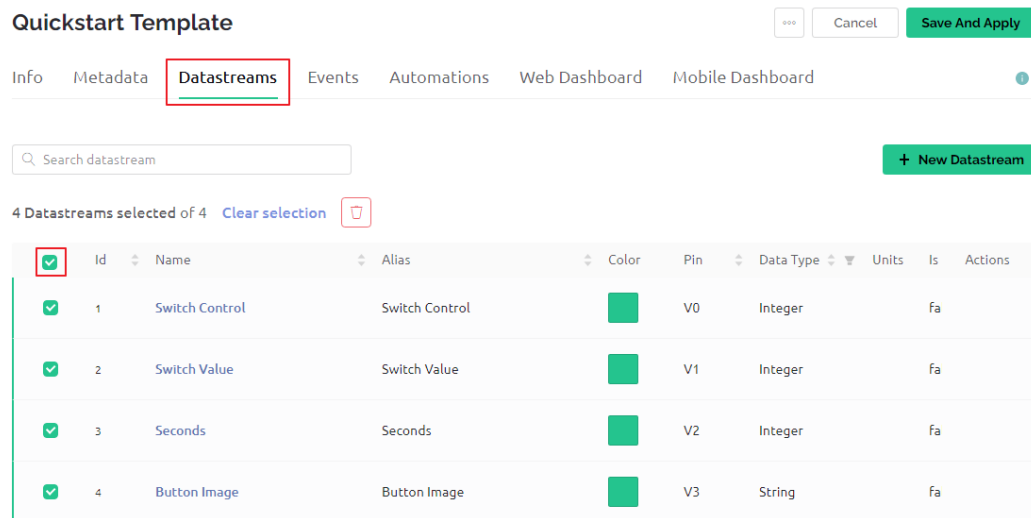


## 2. Edit Dashboard

1. Go to the **Quickstart Device** you created earlier, click on the menu icon in the upper right corner and select **edit dashboard**.




2. Datastreams allow the widgets on Blynk and the code on your board to recognize each other. To experience the complete configuration process, remove all Datastreams from the Datastreams page.



3. Please read the warning carefully and confirm it is correct before deleting the Datastreams.

## DANGER ZONE

 Deleting Datastream(s) can lead to breaking changes.

- All widgets using this Datastream(s) will stop working
- All Automation scenarios using this Datastream(s) will stop working
- All active rules in Rules Engine using this Datastream(s) will stop working

This can not be undone.

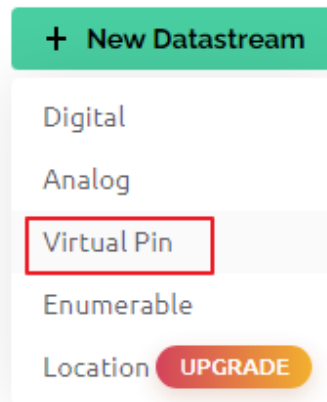
Type DELETE in the field below to proceed with the deletion.

☒ I fully understand that this action is critical and will lead to data loss

4. Create a Datastream of type **Virtual Pin**, which will be used to control the LED using Blynk's switch.

## Datastreams

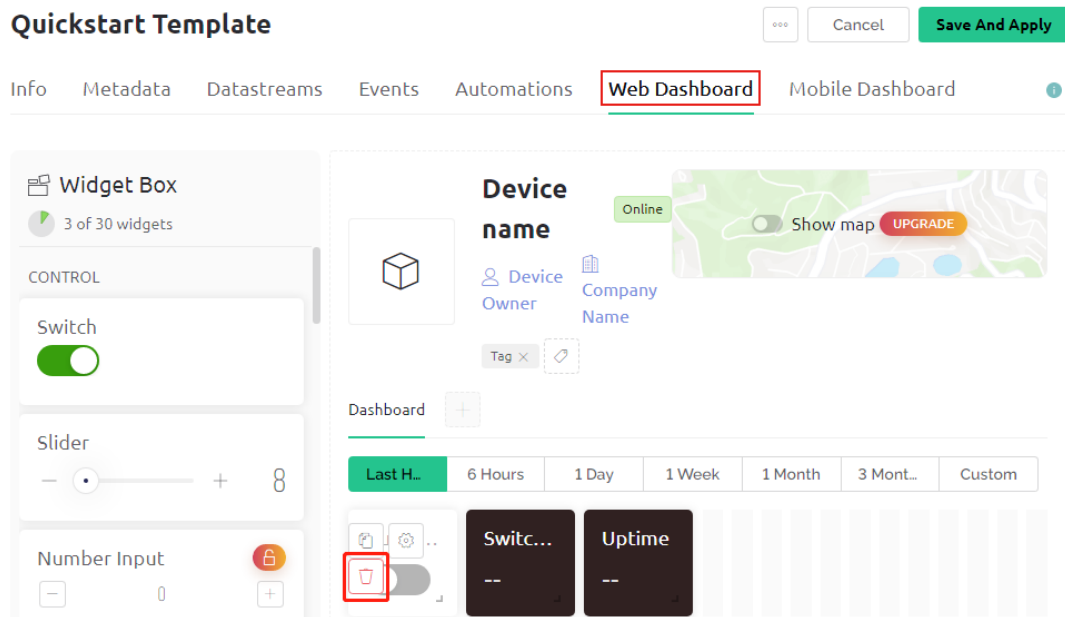
Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.



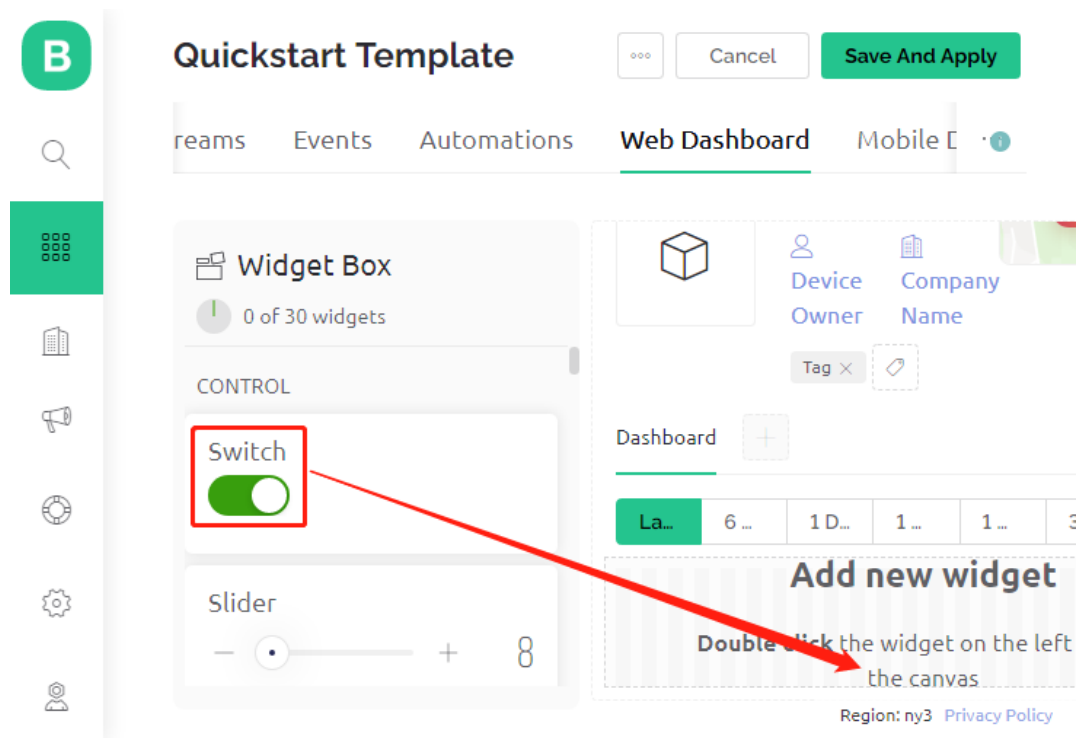
5. Configure the **Virtual Pin**. As the button and LED only need to be ON and OFF, set DATA TYPE to Integer and MIN and MAX to 0 and 1.

A screenshot of a mobile application interface showing the 'Virtual Pin Datastream' configuration screen. The screen has a white background with a green header bar. The title 'Virtual Pin Datastream' is at the top. Below the title, there are several input fields and dropdown menus. The 'NAME' field contains 'Button Contro' and is highlighted with a red rectangular box. The 'ALIAS' field contains 'Button Control'. The 'PIN' field contains 'V0' and is highlighted with a red rectangular box. The 'DATA TYPE' dropdown menu is set to 'Integer'. The 'UNITS' dropdown menu is set to 'None'. The 'MIN' field contains '0', the 'MAX' field contains '1', and the 'DEFAULT VALUE' field contains '0'. At the bottom, there is a green button labeled 'Save' and a white button labeled 'Cancel'. There is also a link for 'ADVANCED SETTINGS' and a footer with 'Region: ny3' and 'Privacy Policy'.

6. Go to the **Web Dashboard** page and delete the existing widgets.

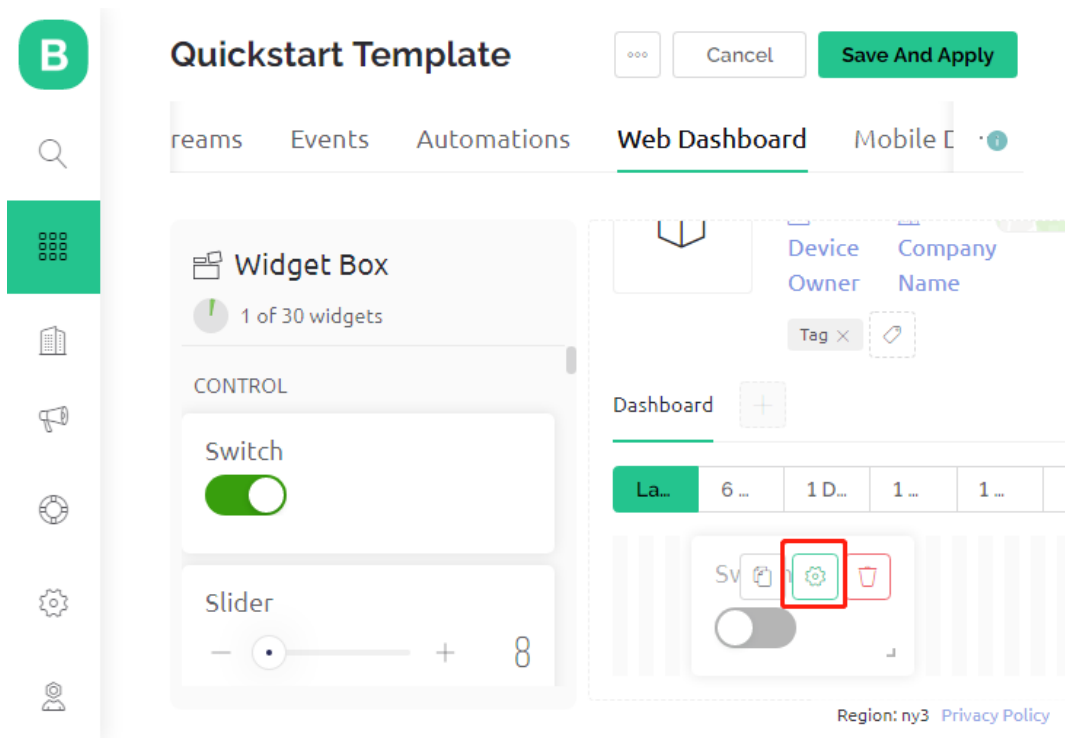


7. Drag and drop a **switch** widget from the **Widget Box** on the left.

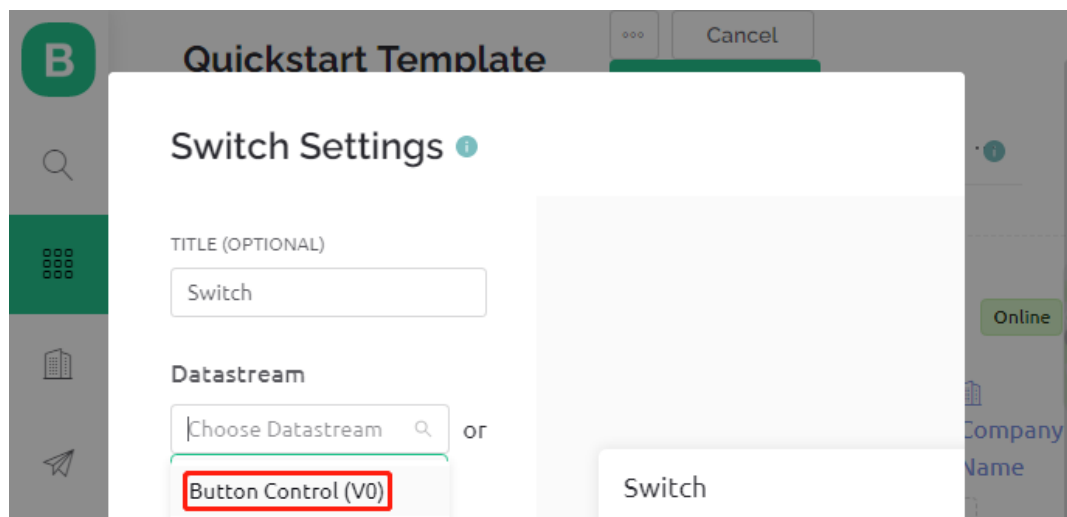


8. Now to set it up.

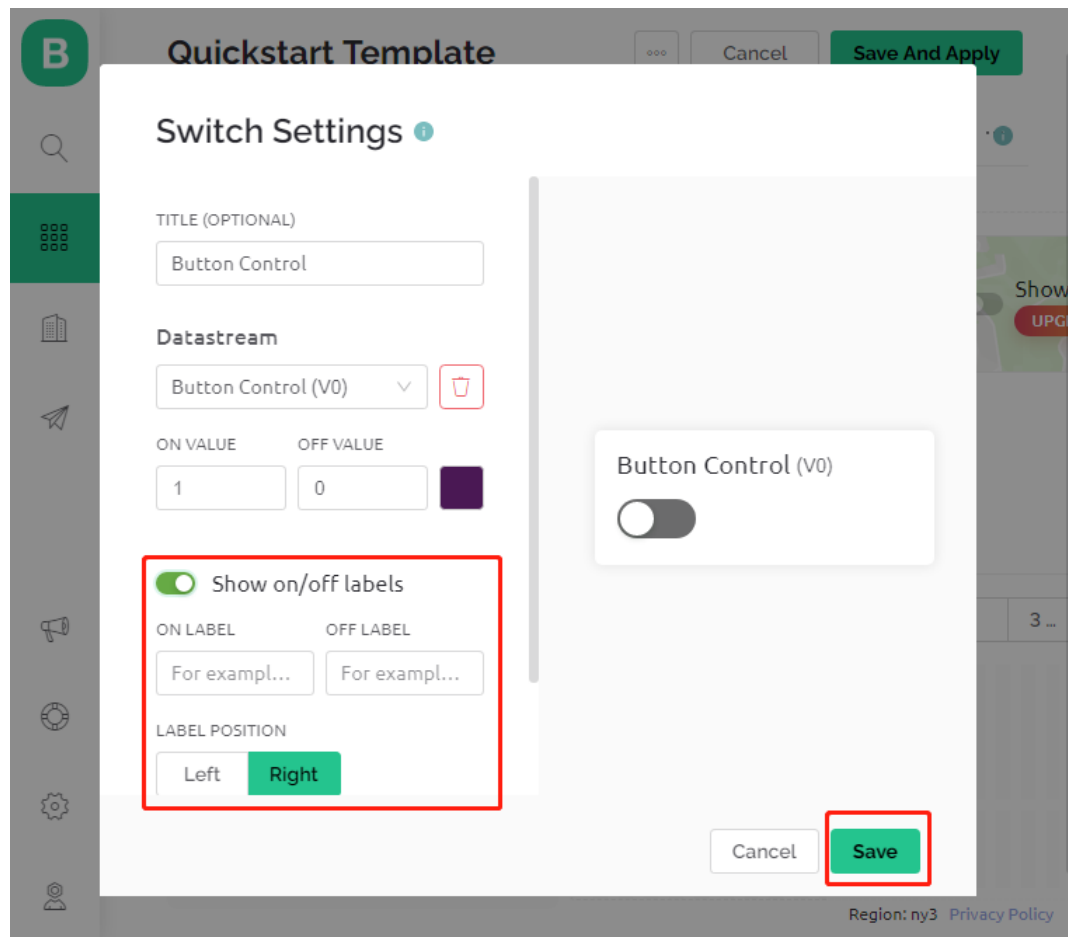




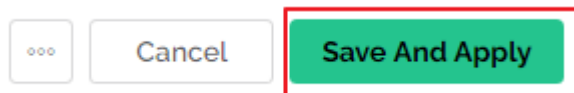
9. Select **Datastream** as the one you set earlier.



10. After selecting Datastream, you will see a few custom settings, then press Save.

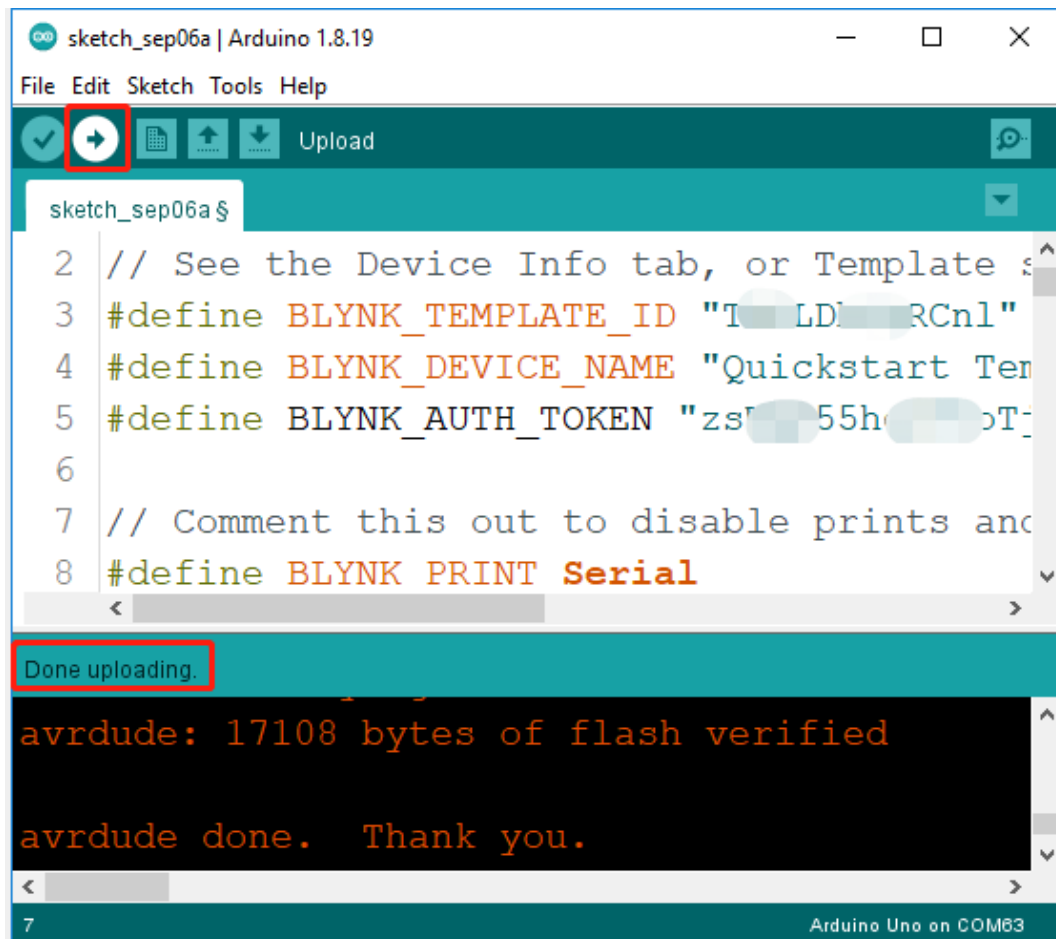


11. Finally, click **Save And Apply**.

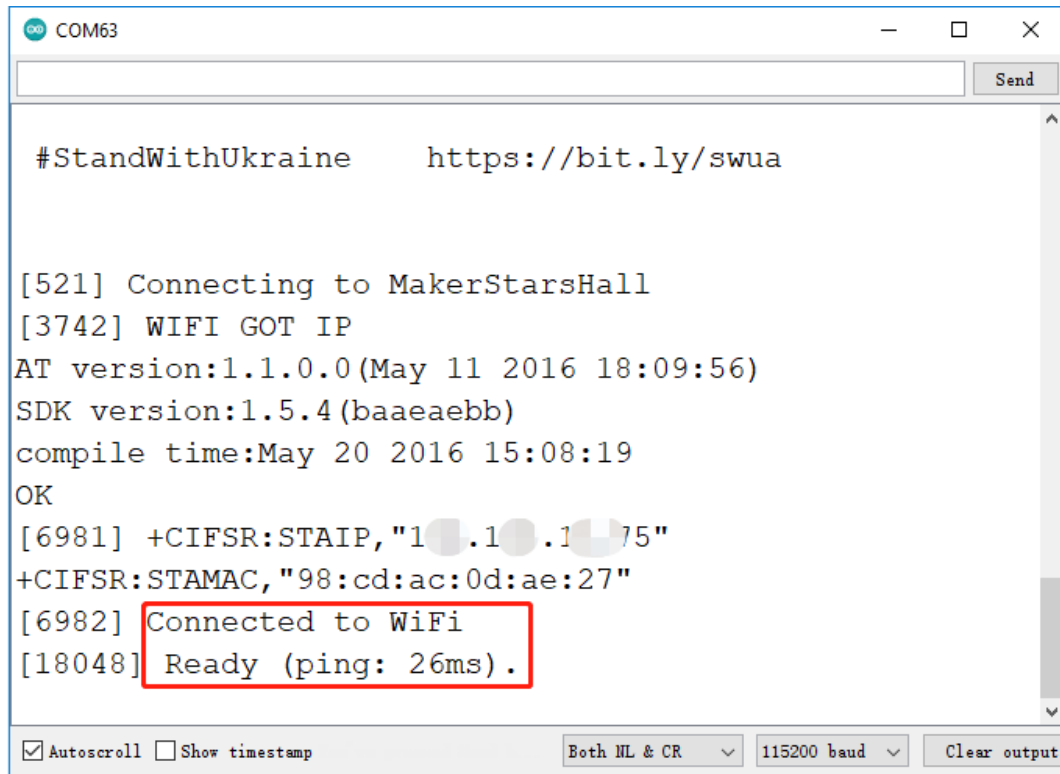


### 3. Run the Code

1. Open the `2.get_data_from_blynk.ino` file under the path of `3in1-kit\iot_project\2.get_data_from_blynk`, or copy this code into **Arduino IDE**.
2. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
3. After selecting the correct board and port, click the **Upload** button.



4. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.

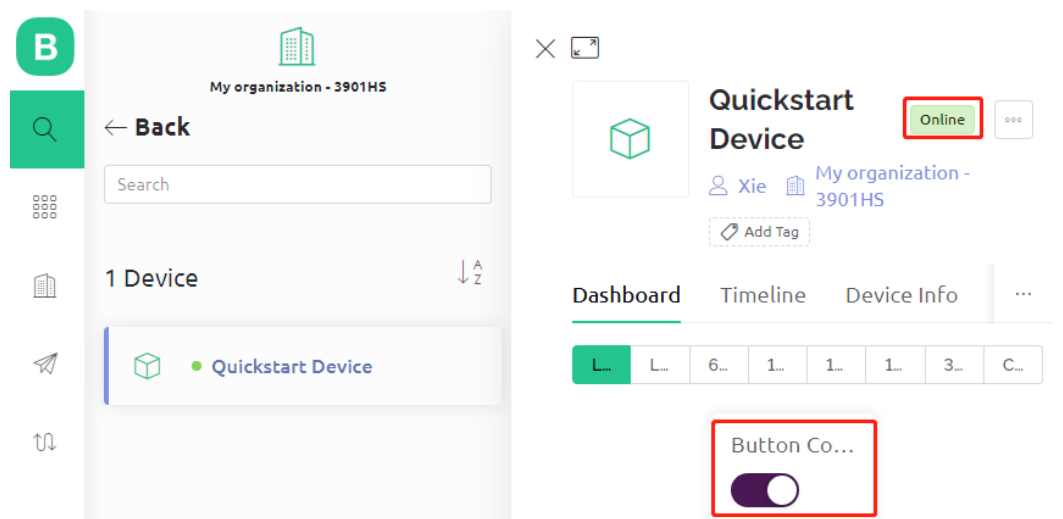


**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on your board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

5. Back at Blynk, you can see that the status has changed to online and you can now use the switch widget on blynk to control the LED connected to the R4 board.



6. If you want to use Blynk on mobile devices, please refer to [How to use Blynk on mobile device?](#).

### How it works?

The difference between the code in this project and the code in the previous chapter [1.4 Connecting the R4 board to Blynk](#) is the following lines.

```
const int ledPin=6;

BLYNK_WRITE(V0)
{
    int pinValue = param.asInt(); // assigning incoming value from pin V0 to a variable
    // You can also use:
    // String i = param.asStr();
    // double d = param.asDouble();
    digitalWrite(ledPin,pinValue);
}

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

Regarding the `pinMode` and `digitalWrite` of the `ledPin`, I'm sure you're already familiar with them, so I won't go over them again. What you need to focus on is the `BLYNK_WRITE(V0)` function.

What it will do is that when the value of Blynk's `V0` changes, Blynk.Cloud will tell your device "I am writing to **Virtual Pin V0**", and your device will be able to perform something once it gets this information.

We created the `V0` Datastream in the previous step and applied it to the Switch Widget. This means that every time we operate the Switch Widget, `BLYNK_WRITE(V0)` will be triggered.

We write two instructions in this function.

```
int pinValue = param.asInt();
```

Get the value of `V0` and assign it to the variable `pinValue`.

```
digitalWrite(ledPin,pinValue);
```

Write the value of `V0` obtained to the `ledPin`, so that the Switch widget on Blynk can control the LED.

## 6.3 3. Push Data to Blynk

This chapter will show you how to send data to Blynk.

We create a door and window detection device here. The circuit with the reed switch is placed next to the door and window, and the magnet is mounted on the edge of the door and window. When the door or window is closed, the Reed Switch will be turned on by the magnetic force and the corresponding pin value on the R4 board will change. Blynk.cloud will receive this value so that you can see if your house's doors and windows are closed even when you're away from home.

Now we will use an LED widget in Blynk to indicate if your windows and doors are closed (i.e. if the Reed Switch is on or off).

### Required Components

## SunFounder 3in1 Kit

---

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>ESP8266 Module</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Reed Switch</i>	-

### 1. Build the Circuit

---

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.

---

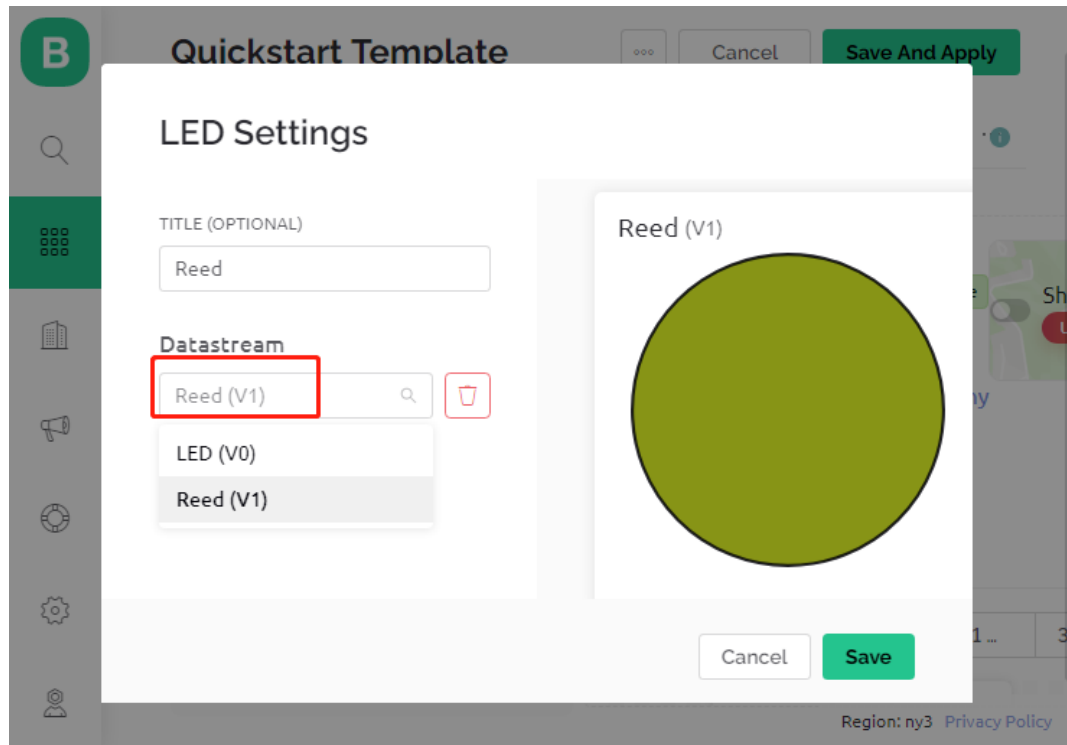
1. Create a **Datastream** of type **Virtual Pin** in the **Datastream** page to get the value of Reed Switch. Set the DATA TYPE to **Integer** and MIN and MAX to **0** and **1**.

1. Create a **Datastream** of type **Virtual Pin** in the **Datastream** page to get the value of Reed Switch. Set the DATA TYPE to **Integer** and MIN and MAX to **0** and **1**.

2. Drag and drop an **LED widget** on the **Web Dashboard** page, at a value of 1, it will light up (with color), otherwise it will be white.

3. In the settings page of the **LED widget**, select **Datastream** as **Reed(V1)**, and save it.





### 3. Run the Code

1. Open the `3.push_data_to_blynk.ino` file under the path of `3in1-kit\iot_project\3.push_data_to_blynk`, or copy this code into **Arduino IDE**.
2. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
3. After selecting the correct board and port, click the **Upload** button.
4. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.

```

COM63

#StandWithUkraine    https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"192.168.1.175"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).

☒ Autoscroll ☐ Show timestamp
Both NL & CR 115200 baud Clear output

```

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

- Now, Blynk will show the status of your doors and windows. If your doors and windows are closed, the LED widget will be green, otherwise, it will be gray.
- If you want to use Blynk on mobile devices, please refer to [How to use Blynk on mobile device?](#).

### How it works?

For this example, you should focus on the following lines. “Write data every second to Blynk Cloud’s V1 Datastream” is defined by these lines.

```

BlynkTimer timer;

void myTimerEvent()
{
    Blynk.virtualWrite(V1, pinValue);
}

void setup()
{

```

(continues on next page)

(continued from previous page)

```

    timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
    timer.run(); // Initiates BlynkTimer
}

```

Blynk library provides a built-in timer, first we create a timer object.

```
BlynkTimer timer;
```

Set the timer interval in `setup()`, here we set to execute the `myTimerEvent()` function every 1000ms

```
timer.setInterval(1000L, myTimerEvent);
```

Run `BlynkTimer` in `loop()`.

```
timer.run();
```

Edit the custom function `myTimerEvent()`, the code `Blynk.virtualWrite(V1, pinValue)` is used to write the data `pinValue` for `V1`.

```

void myTimerEvent()
{
    Blynk.virtualWrite(V1, pinValue);
}

```

## 6.4 4. Cloud Music Player

The goal of this project is to create a music player using Blynk. Music is played in the same way as in *5.7 Tone() or noTone()*, by writing the song in the program and playing it with a passive buzzer. however, in this example, we can click the switch to play/pause and slide the slider to change the playback progress.

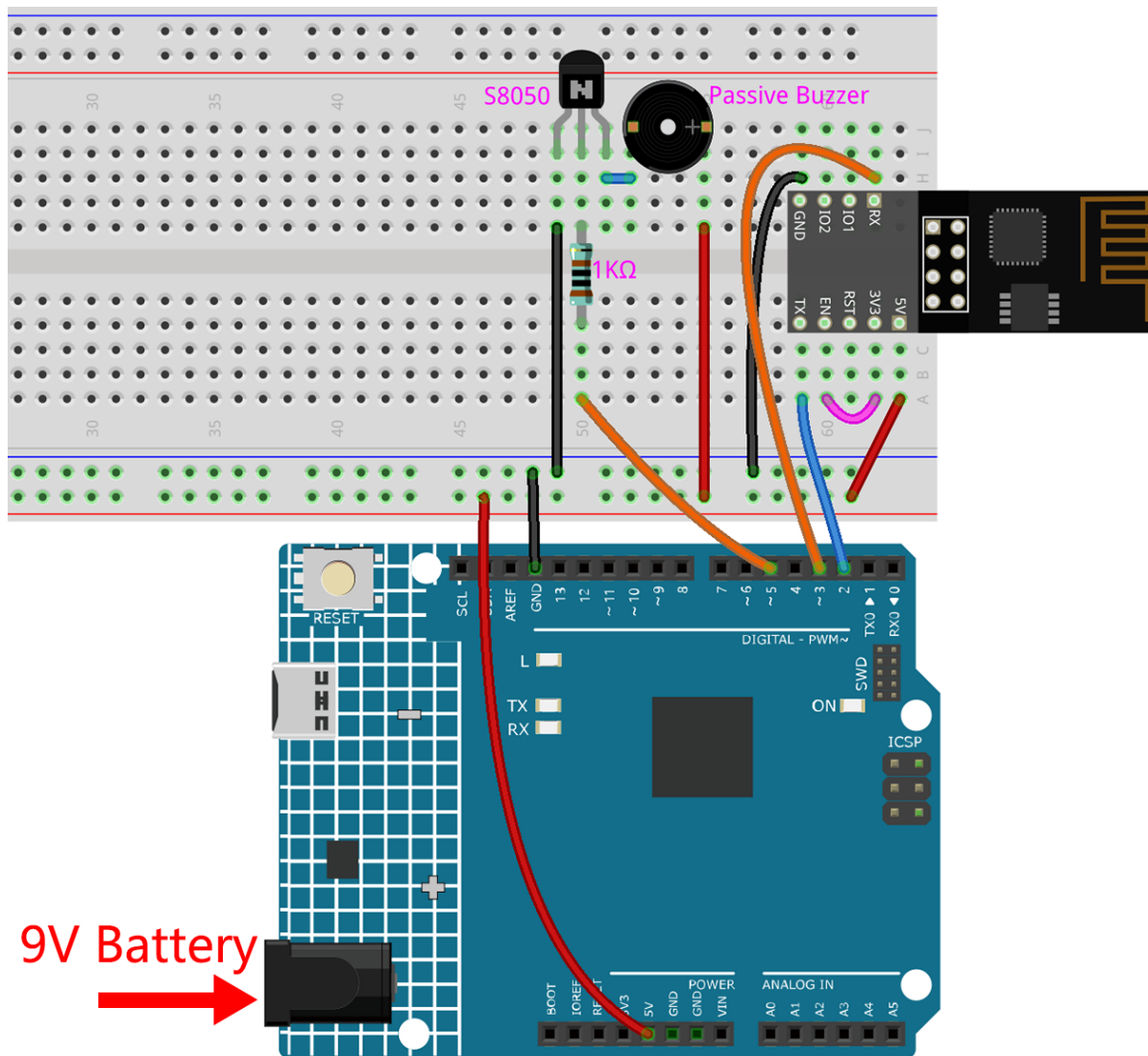
### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.



1. Create a **Datastream** of type **Virtual Pin** on the **Datastream** page as the value modified by the Slider widget added later or code. Set the DATA TYPE to **Integer** and MIN and MAX to **0** and **30**.

**Virtual Pin Datastream**

NAME:  ALIAS:

PIN:  DATA TYPE:

UNITS:

MIN:  MAX:  DEFAULT VALUE:

[+ ADVANCED SETTINGS](#)

Region: ny3 [Privacy Policy](#)

2. Also create another **Datastream** of type **Virtual Pin** to display the music name, and set the DATA TYPE to **String**.

**Virtual Pin Datastream**

NAME:  ALIAS:

PIN:  DATA TYPE:

DEFAULT VALUE:

Region: ny3 [Privacy Policy](#)

3. Go to the **Web Dashboard** page, drag a **Switch** widget and set **Datastream** to V0 (V0 is already set in 2. *Get Data from Blynk*); drag a **Label** widget and set it to V3; drag a **Slider** widget and set it to V2.

**Widget Box**  
3 of 30 widgets

**Chart**

**Map**

**Device name** Online  
[Device Owner](#) [Company Name](#)  
 Tag

**Dashboard**

**Last Hour** 6 Hours 1 Day 1 Week 1 Month

Button Co...

Slider

Song Name   
String

**Note:** Your virtual pins may be different from mine, yours will prevail, but you need to modify the corresponding pin number in the code.

### 3. Run the Code

1. Open the 4.cloud\_music\_player.ino file under the path of 3in1-kit\iot\_project\4.cloud\_music\_player.
2. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
3. After selecting the correct board and port, click the **Upload** button.
4. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.

```

COM63

#StandWithUkraine https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"1.1.1.175"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

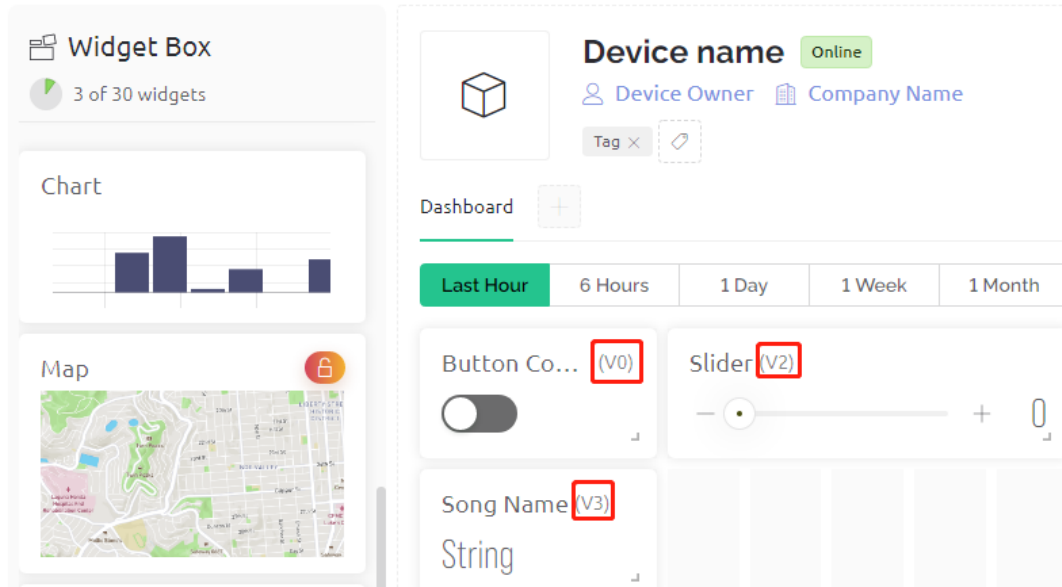
```

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

5. Now, you can use Blynk's Button Control widget to start/pause the music and the Slider to adjust the playback progress, and you'll also see the name of the music.



6. If you want to use Blynk on mobile devices, please refer to [How to use Blynk on mobile device?](#).

#### How it works?

The datastream **V0** is used to get the status of the Switch widget and assign it to the variable **musicPlayFlag**, which controls pausing and playing the music.

```
int musicPlayFlag=0;

BLYNK_WRITE(V0)
{
    musicPlayFlag = param.asInt(); // START/PAUSE MUSIC
}
```

The data stream **V2** is used to get the value of the slider widget and assign it to the variable **scrubBar** when the slider is moved.

```
int scrubBar=0;

BLYNK_WRITE(V2)
{
    scrubBar=param.asInt();
}
```

When the device is connected to the **Blynk Cloud**, write the music name for the **V3** datastream and then display it with the **Label** widget.

```
BLYNK_CONNECTED() {
    String songName = "Ode to Joy";
    Blynk.virtualWrite(V3, songName);
}
```

**Blynk Timer** will execute every second. Music is played if **musicPlayFlag** is not 0, i.e. the **Switch** widget is ON. As soon as two notes are played, the progress bar variable **scrubBar** is incremented by 2, and the value is then written to the **Blynk Cloud**, which synchronizes the value of the **Slider** widget.



```

void myTimerEvent()
{
    if(musicPlayFlag!=0)
    {
        tone(buzzerPin,melody[scrubBar],250);
        scrubBar=(scrubBar+1)%(sizeof(melody)/sizeof(int));
        delay(500);
        tone(buzzerPin,melody[scrubBar],250);
        scrubBar=(scrubBar+1)%(sizeof(melody)/sizeof(int));
        Serial.println(scrubBar);
        Blynk.virtualWrite(V2, scrubBar);
    }
}

```

## 6.5 5. Home Environment Monitoring

In this chapter, we will use Blynk to create a home environment monitor. You can measure the temperature, humidity, and light intensity of a room using the DHT11 and photoresistor. By sending these values to Blynk, you will be able to know the environment of your home via the internet.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

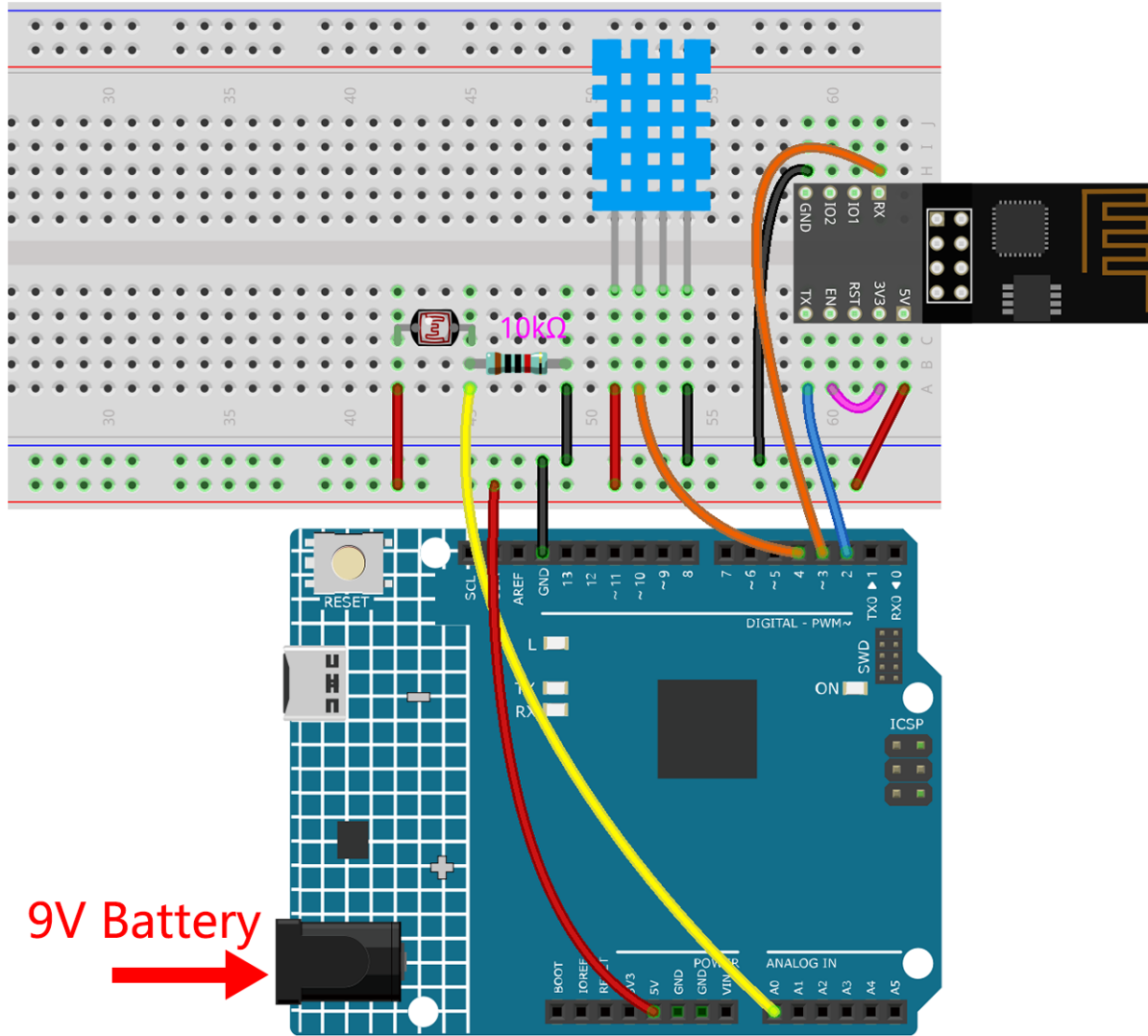
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>ESP8266 Module</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Photoresistor</i>	
<i>DHT11 Humiture Sensor</i>	-

### 1. Build the Cirduit

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.



## 2. Edit Dashboard

1. For recording humidity values, create a **DataStream** of type **Virtual Pin** on the **DataStream** page. Set the DATA TYPE to **Double** and MIN and MAX to **0** and **100**. Also set the units to **Percentage, %**.

**Virtual Pin Datastream**

NAME: Humidity ALIAS: Humidity

PIN: V4 DATA TYPE: Double

UNITS: Percentage, %

MIN: 0 MAX: 100 DECIMALS: ### DEFAULT VALUE: Default Value

[+ ADVANCED SETTINGS](#)

Cancel Save

Region: ny3 [Privacy Policy](#)

- Then create a **Datastream** of type **Virtual Pin** for recording the temperature. Set DATA TYPE to Double, MIN and MAX to -30 and 50, and units to **Celsius, °C**.

**Virtual Pin Datastream**

NAME: Temperature ALIAS: Temperature

PIN: V5 DATA TYPE: Double

UNITS: Celsius, °C

MIN: -30 MAX: 50 DECIMALS: ### DEFAULT VALUE: Default Value

[+ ADVANCED SETTINGS](#)

Cancel Save

Region: ny3 [Privacy Policy](#)

- Also create a **Datastream** of type **Virtual Pin** to record the light intensity. Use the default data type - **Integer**, with MIN and MAX set to 0 and 1024.

**Virtual Pin Datastream**

NAME:  ALIAS:

PIN:  DATA TYPE:

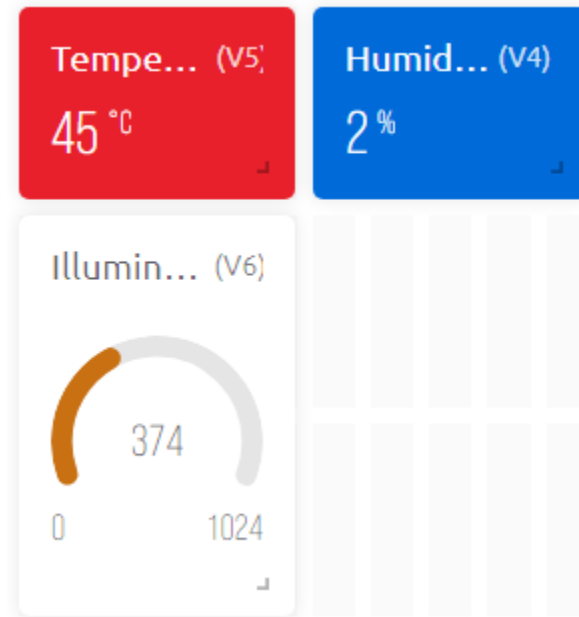
UNITS:

MIN:  MAX:  DEFAULT VALUE:

[+ ADVANCED SETTINGS](#)

Region: ny3 [Privacy Policy](#)

- Go to the **Web Dashboard** page, drag two **Label** widgets and set their data streams to **V4** and **V5** respectively, and drag a **Gauge** widget and set the data stream to **V6**. Also in the widget setting, you can enable **Change color based on value** and select the appropriate color to make the widget look better and more intuitive.



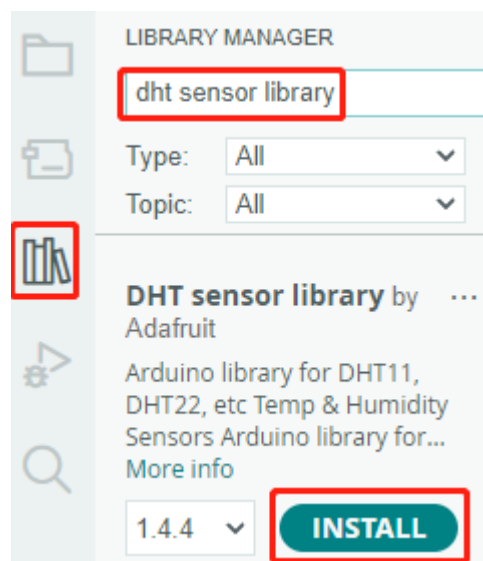
### 3. Run the Code

1. Open the `5.home_environment_monitoring.ino` file under the path of `3in1-kit\iot_project\5.home_environment_monitoring`, or copy this code into **Arduino IDE**.

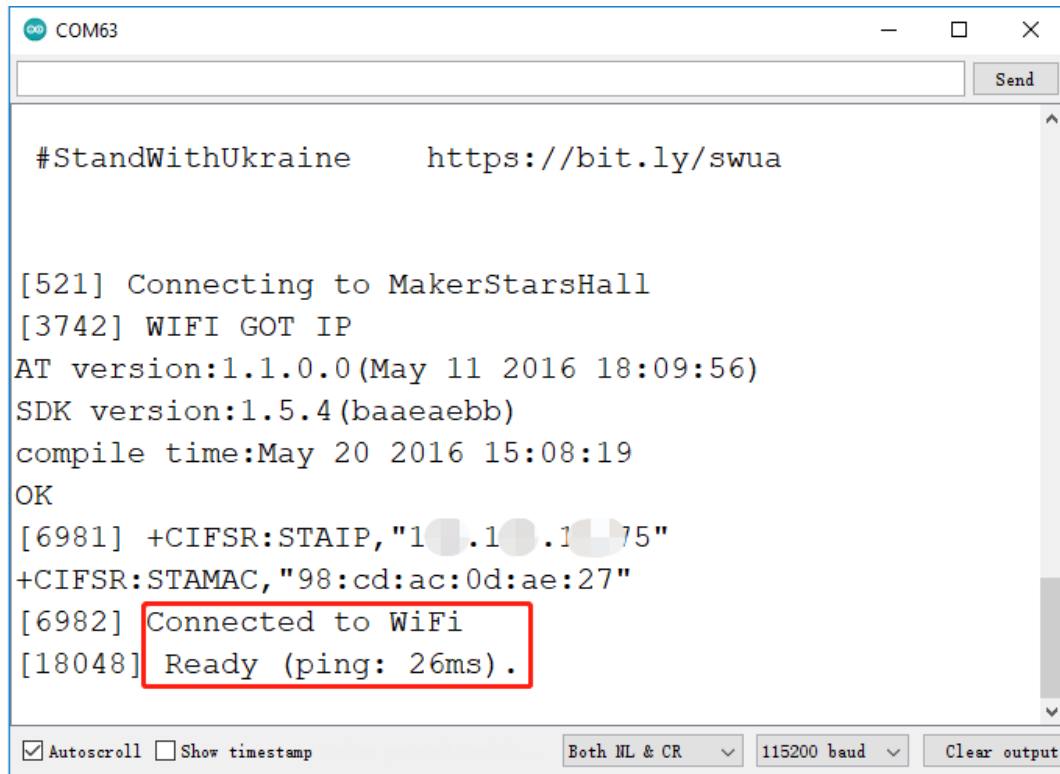
---

#### Note:

- The DHT sensor library is used here, you can install it from the **Library Manager**.



2. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
3. After selecting the correct board and port, click the **Upload** button.
4. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.



```
COM63

#StandWithUkraine  https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"1.1.1.175"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).
```

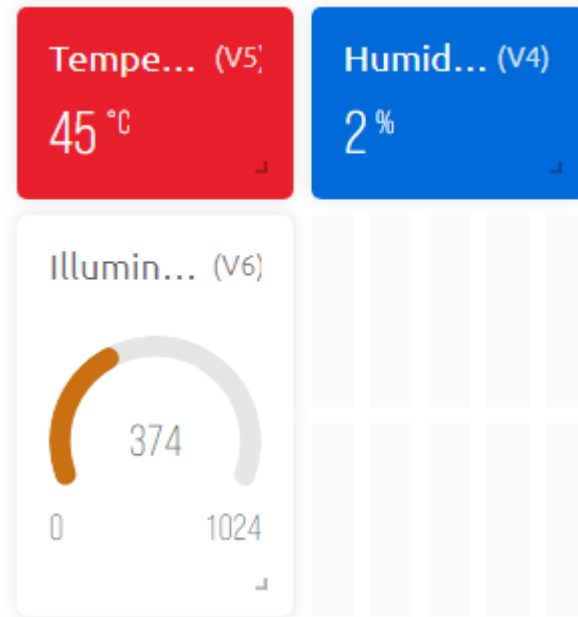
☒ Autoscroll ☐ Show timestamp    Both NL & CR    115200 baud    Clear output

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

5. Now, you will see the current ambient temperature, humidity and light intensity on Blynk.



6. If you want to use Blynk on mobile devices, please refer to *How to use Blynk on mobile device?*.



### How it works?

These two functions are used to get the temperature, humidity and light intensity of the room.

```
int readLight(){
    return analogRead(lightPin);
}

bool readDHT() {

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
```

(continues on next page)



(continued from previous page)

```

humidity = dht.readHumidity();
// Read temperature as Celsius (the default)
temperature = dht.readTemperature();

// Check if any reads failed and exit early (to try again).
if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return false;
}
return true;
}

```

With the Blynk Timer, the ambient temperature, humidity, and light intensity are obtained every second and sent to the data stream on the Blynk Cloud, from which the widgets display the data.

```

void myTimerEvent()
{
    bool chk = readDHT();
    int light = readLight();
    if(chk){
        Blynk.virtualWrite(V4,humidity);
        Blynk.virtualWrite(V5,temperature);
    }
    Blynk.virtualWrite(V6,light);
}

```

## 6.6 6. Plant Monitor

The purpose of this project is to create a smart watering system that detects the current temperature, humidity, intensity of light, and soil moisture and displays them on Blynk.

As soon as you turn on the Switch toggle in Blynk Cloud, the pump will start working and the plants will be hydrated.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

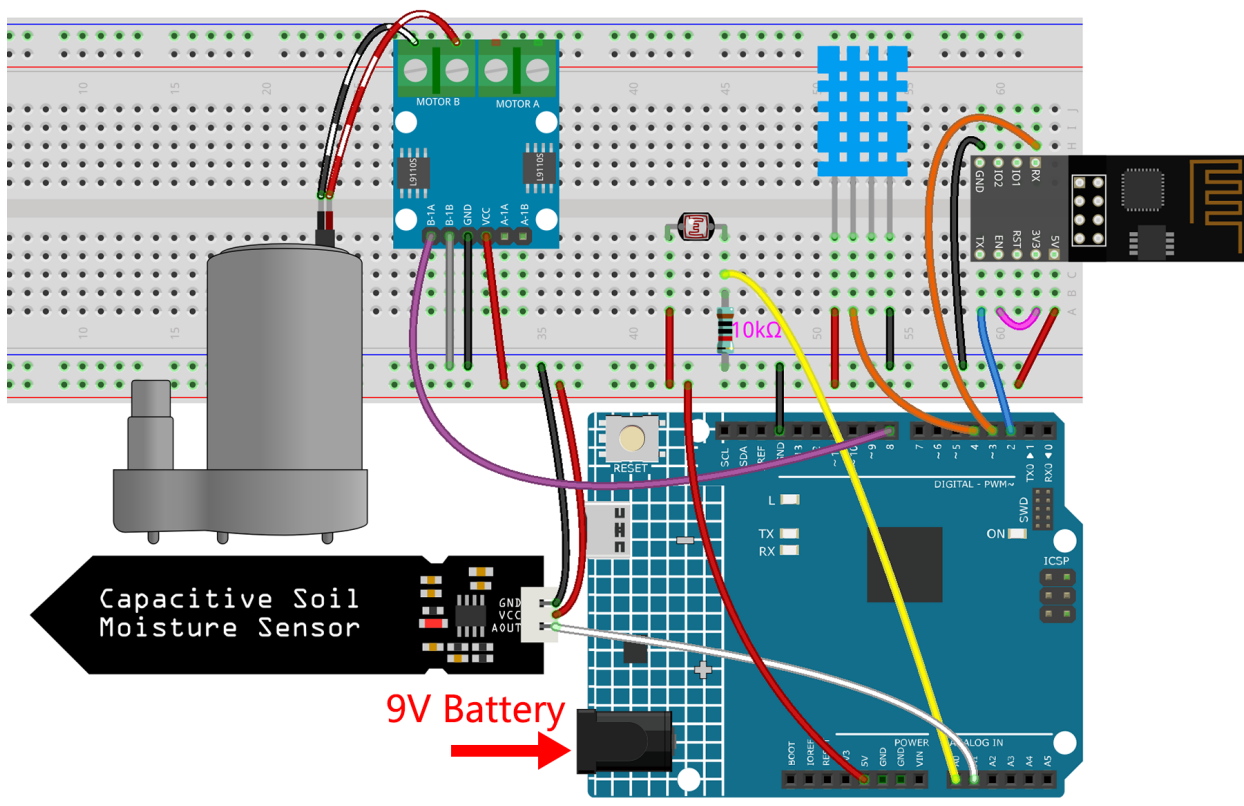
Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

You can also buy them separately from the links below.

COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>ESP8266 Module</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Photoresistor</i>	
<i>DHT11 Humiture Sensor</i>	-
<i>Soil Moisture Module</i>	
<i>L9110 Motor Driver Module</i>	-
<i>Centrifugal Pump</i>	-

## 1. Build the Cirduit

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.



## 2. Edit Dashboard

1. The data streams created in the previous projects need to be saved, and they will be used in this project as well.

- For recording soil moisture, create another **Datastream** of type **Virtual Pin** on the **Datastream** page. Set DATA TYPE to Integer and MIN and MAX to 0 and 1024.

**Virtual Pin Datastream**

NAME:  ALIAS:

PIN:  DATA TYPE:

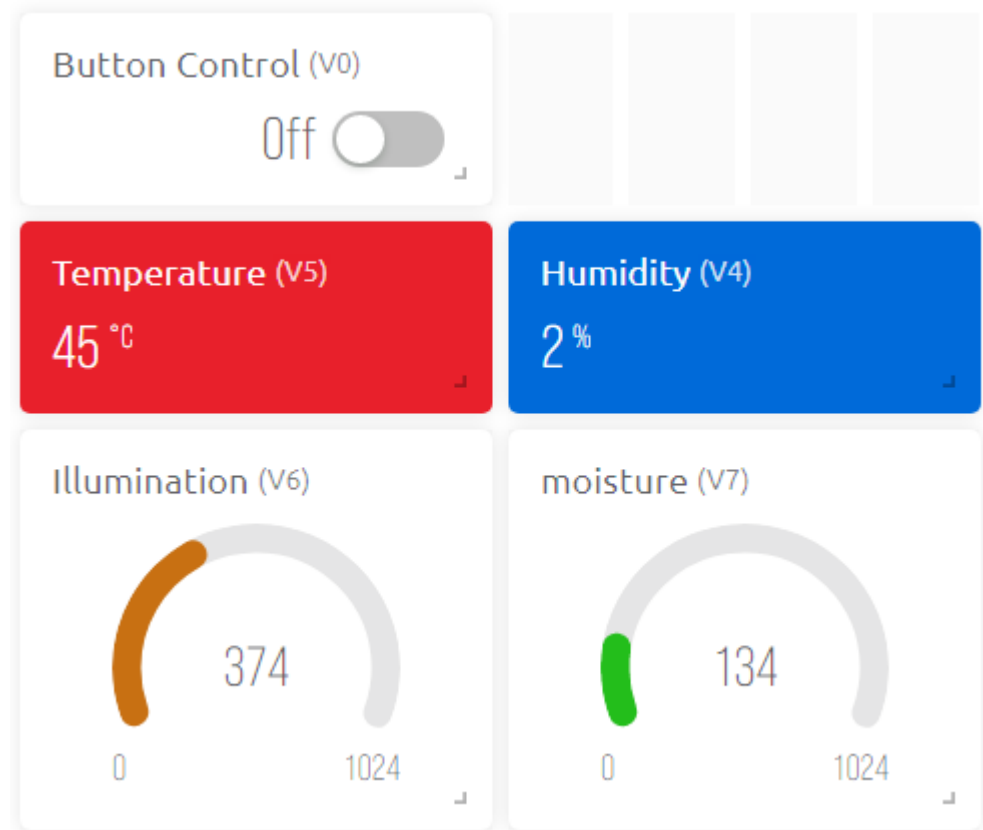
UNITS:

MIN:  MAX:  DEFAULT VALUE:

[+ ADVANCED SETTINGS](#)

Region: ny3 [Privacy Policy](#)

- Now go to the **Web Dashboard** page, drag 2 **Label** widgets and set their data streams to **V4** and **V5** respectively; drag 2 **Gauge** widgets and set their data streams to show **V6** and **V7** respectively; and finally drag a **Switch** widget and set its data stream to **V0**.

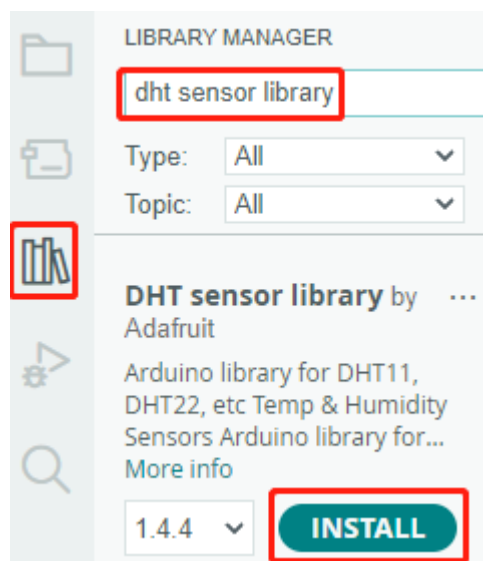


### 3. Run the Code

Open the 6.plant\_monitoring.ino file under the path of 3in1-kit\iot\_project\6.plant\_monitoring, or copy this code into **Arduino IDE**.

#### Note:

- The DHT sensor library is used here, you can install it from the **Library Manager**.



1. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
2. After selecting the correct board and port, click the **Upload** button.
3. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.

The screenshot shows the Serial Monitor window for COM63. The text displayed is as follows:

```
#StandWithUkraine      https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"192.168.1.75"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).
```

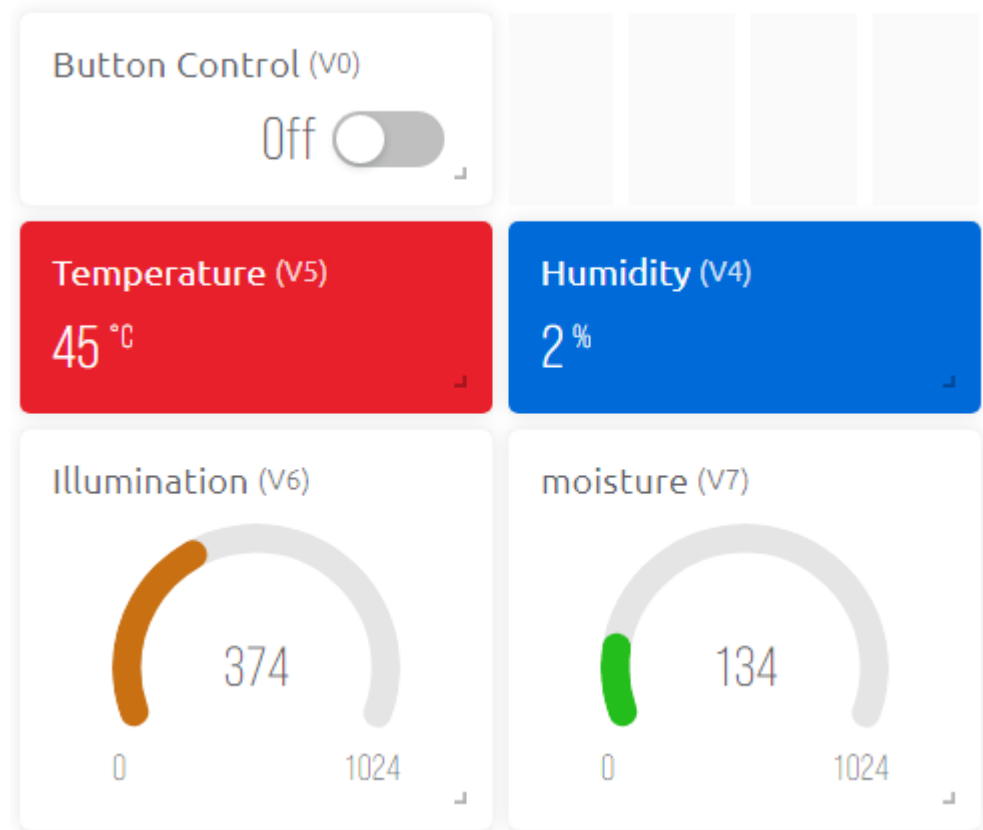
The lines "[6982] Connected to WiFi" and "[18048] Ready (ping: 26ms)." are highlighted with a red rectangle. At the bottom of the window, the settings are: ☒ Autoscroll, ☐ Show timestamp, Both NL & CR, 115200 baud, and a Clear output button.

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

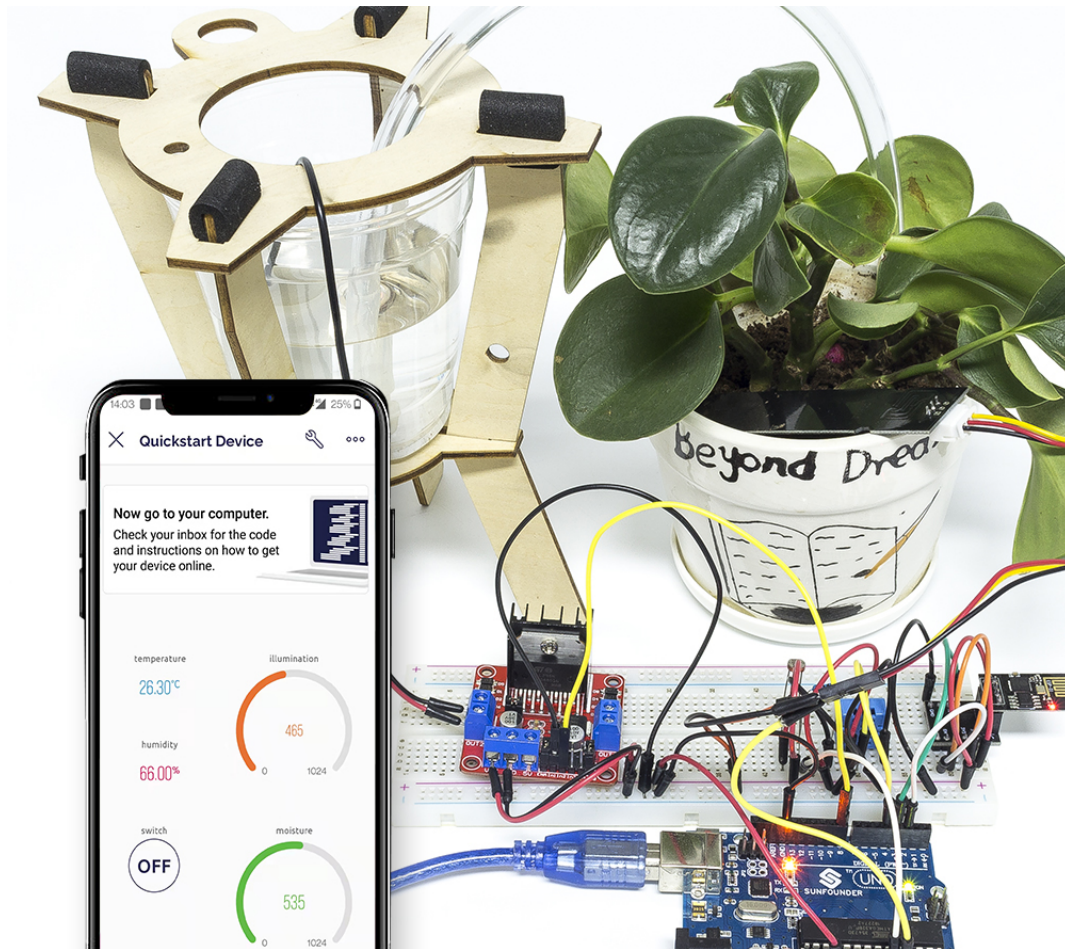
- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

4. Back to the Blynk, you will see the current temperature, humidity, light intensity, and soil moisture. You can let the pump water the plants if necessary by clicking the Button Control widget.



5. If you want to use Blynk on mobile devices, please refer to *How to use Blynk on mobile device?*.



### How it works?

This BLYNK\_WRITE causes Blynk's **Switch** widget to start the pump when it is ON and turn it off when it is OFF.

```
BLYNK_WRITE(V0)
{
  if(param.asInt()==1){
    digitalWrite(pumpA,HIGH);
  }else{
    digitalWrite(pumpA,LOW);
  }
}
```

These three functions are used to get the current environment temperature, humidity, light intensity and soil moisture.

```
int readMoisture(){
  return analogRead(moisturePin);
}

int readLight(){
  return analogRead(lightPin);
}

bool readDHT() {
```

(continues on next page)

(continued from previous page)

```

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

// Check if any reads failed and exit early (to try again).
if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return false;
}
return true;
}

```

With the Blynk Timer, the ambient temperature, humidity, light intensity and soil moisture are obtained every second and sent to the data stream on the **Blynk Cloud**, from which the widgets display the data.

```

void myTimerEvent()
{
    bool chk = readDHT();
    int light = readLight();
    int moisture = readMoisture();
    if(chk){
        Blynk.virtualWrite(V4,humidity);
        Blynk.virtualWrite(V5,temperature);
    }
    Blynk.virtualWrite(V6,light);
    Blynk.virtualWrite(V7,moisture);
}

```

## 6.7 7. Current Limiting Gate

Some situations, such as parking lots, require quantity management.

Here we create a smart gate: a servo is used as the gate, and an IR obstacle detector is placed in front of it; if an object (like a car) is detected, the gate will open and the number will be increased by 1. The count is displayed with a 7-segment display and is also uploaded to the Blynk Cloud for you to view remotely. Finally, Blynk has a Switch widget to enable or disable this smart gate system.

### Required Components

In this project, we need the following components.

It's definitely convenient to buy a whole kit, here's the link:

Name	ITEMS IN THIS KIT	LINK
3 in 1 Starter Kit	380+	

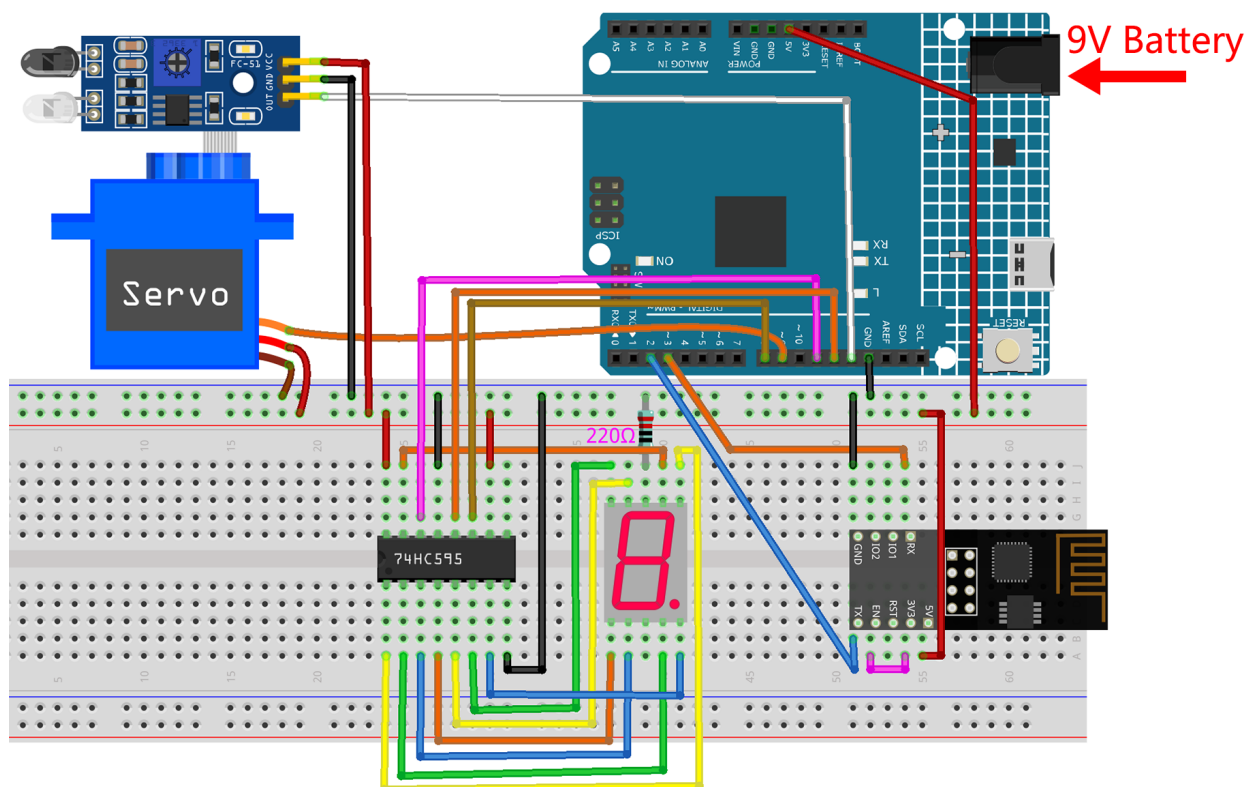
You can also buy them separately from the links below.



COMPONENT INTRODUCTION	PURCHASE LINK
<i>Arduino Uno R4 Minima</i>	-
<i>Breadboard</i>	
<i>ESP8266 Module</i>	
<i>Jumper Wires</i>	
<i>Resistor</i>	
<i>Servo</i>	
<i>Obstacle Avoidance Module</i>	
<i>7-segment Display</i>	
<i>74HC595</i>	

## 1. Build the Circuit

**Note:** The ESP8266 module requires a high current to provide a stable operating environment, so make sure the 9V battery is plugged in.



## 2. Edit Dashboard

1. To record the number, create a **Datastream** of type **Virtual Pin** on the **Datastream** page. Set DATA TYPE to

Integer and MIN and MAX to 0 and 10.

**Virtual Pin Datastream**

NAME:  ALIAS:

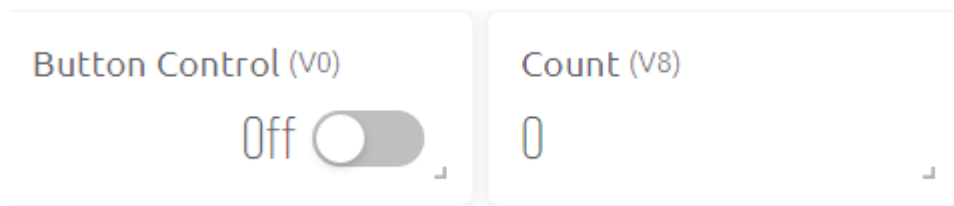
PIN:  DATA TYPE:

UNITS:

MIN:  MAX:  DEFAULT VALUE:

Region: ny3 [Privacy Policy](#)

- Now go to the **Wed Dashboard** page, drag a **Switch** widget to set its data stream to **V0** and a **Label** widget to set its data stream to **V8**.



### 3. Run the Code

- Open the `7.current_limiting_gate.ino` file under the path of `3in1-kit\iot_project\7.current_limiting_gate`, or copy this code into **Arduino IDE**.
- Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
- After selecting the correct board and port, click the **Upload** button.
- Open the Serial monitor (set baudrate to 115200) and wait for a prompt such as a successful connection to appear.

```

COM63

#StandWithUkraine https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"1.1.1.175"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).

☒ Autoscroll ☐ Show timestamp
Both NL & CR 115200 baud Clear output

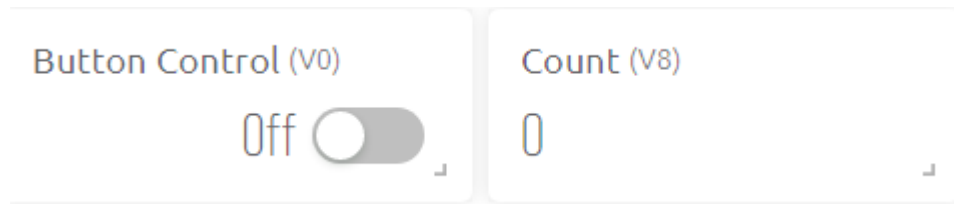
```

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

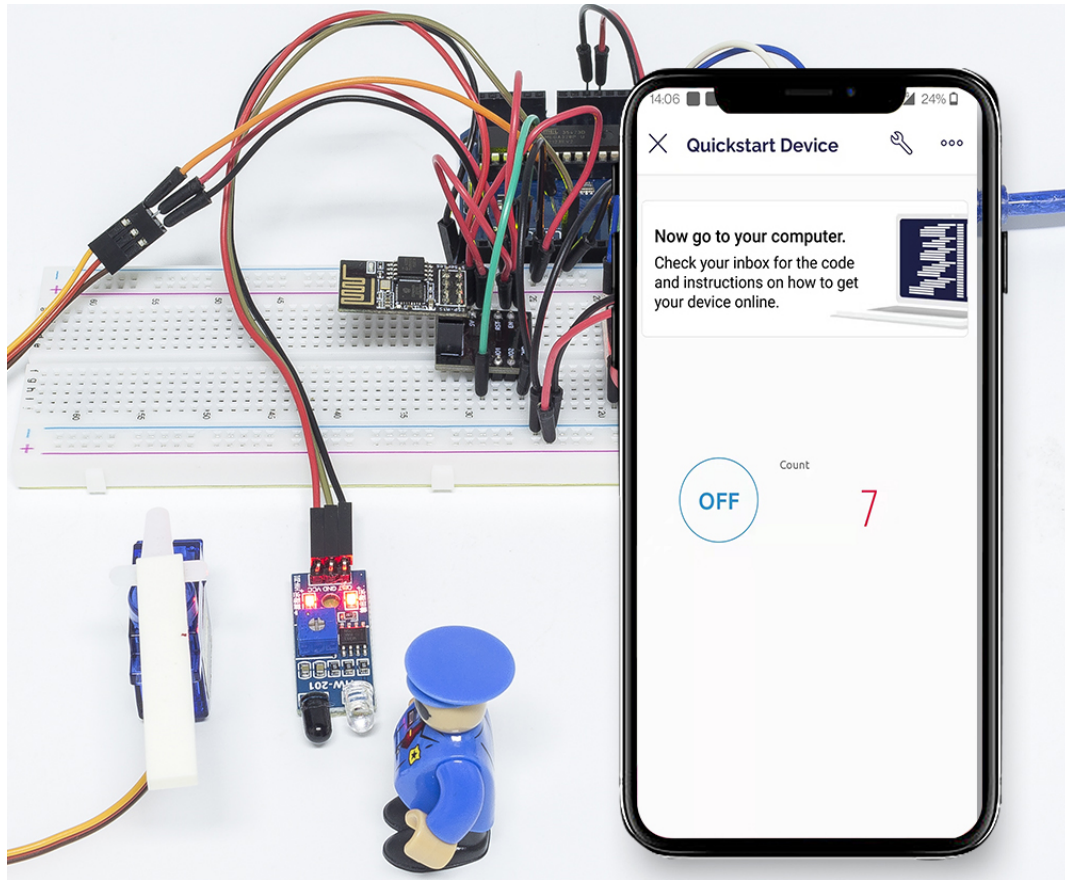
- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

5. Now click on the Button Control widget on Blynk to enable the smart door system. If the IR obstacle avoidance module detects an obstacle, the gate will open and the 7-segment display and the Count widget on Blynk will add 1.



6. If you want to use Blynk on mobile devices, please refer to *How to use Blynk on mobile device?*.



### How it works?

The function `BLYNK_WRITE(V0)` gets the status of the **Switch** widget and assigns it to the variable `doorFlag`, which will be used to determine if the smart gate system is enabled or not.

```
BLYNK_WRITE(V0)
{
    doorFlag = param.asInt(); // Enable Gate
}
```

In the Blynk Timer, `doorFlag` is judged every second and if it is enabled, the main function of the gate is executed.

```
void myTimerEvent()
{
    if (doorFlag)
    {
        channelEntrance();
    }
}
```

The main function of the gate is `channelEntrance()`. When an object approaches the gate (the sensor detects that there is an obstacle), the count is increased by 1. Write count to the datastream V8 of Blynk Cloud and 7-segment display on the circuit, and open the door. If the object goes from present to absent, which means the object has entered the door, close the door.

```
void channelEntrance()
```

(continues on next page)

(continued from previous page)

```

{
  int currentState = digitalRead(irPin); // 0:obstacle 1:no-obstacle
  if (currentState == 0 && lastState == 1) {
    count=(count+1)%10;
    Blynk.virtualWrite(V8, count);
    showNumber(count);
    operateGate(true);
  } else if ((currentState == 1 && lastState == 0)) {
    operateGate(false);
  }
  lastState = currentState;
}

```

The function `showNumber(int num)` is used to make the 7-segment display show the value.

```

void showNumber(int num)
{
  digitalWrite(STcp, LOW); //ground ST_CP and hold low for as long as you are
  ↳transmitting
  shiftOut(DS, SHcp, MSBFIRST, dataArray[num]);
  digitalWrite(STcp, HIGH); //pull the ST_CPST_CP to save the data
}

```

The function `operateGate(bool openGate)` slowly opens the door when the reference is True, and slowly closes the door when the reference is False.

```

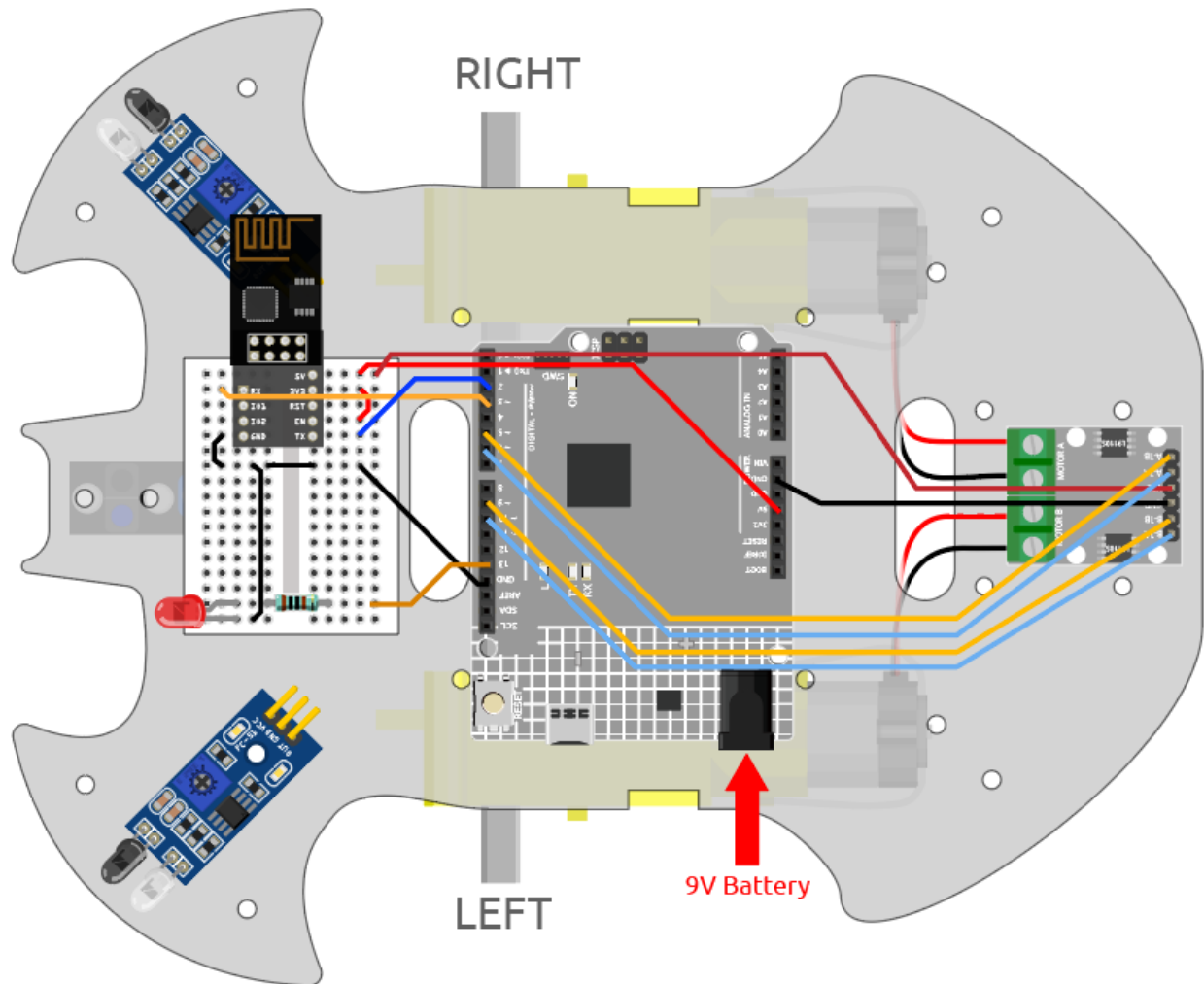
void operateGate(bool openGate) {
  if (openGate == true)
  {
    // open gate
    while (angle <= 90) {
      angle++;
      myservo.write(angle);
      delay(5);
    }
  } else {
    // close gate
    while (angle >= 0){
      angle--;
      myservo.write(angle);
      delay(5);
    }
  }
}
}

```

## 6.8 8. IoT Car

We used the Blynk APP on the cell phone to control the car for this project. But you need to refer to [Car Projects](#) to assemble the car and to get a basic understanding of it. In the era of 5G network popularity, this mode may become one of the main production methods in many industries, so let's experience this play in advance.

### 1. Build the Circuit



### 2. Edit Dashboard

Blynk on mobile cannot edit Datastream, so we still need to do these steps on the web side.

1. Create a **Datastream** of type **Virtual Pin** on the **Datastream** page, to record the X-axis value of the joystick. Set NAME to Xvalue, DATA TYPE to Integer, and MIN and MAX to -10 and 10.

**Virtual Pin Datastream**

NAME: Xvalue ALIAS: Xvalue

PIN: V9 DATA TYPE: Integer

UNITS: None

MIN: -10 MAX: 10 DEFAULT VALUE: 0

+ ADVANCED SETTINGS Cancel Create

2. Create a **Datastream** of type **Virtual Pin** to record the Y-axis value of the joystick. Set NAME to Yvalue, DATA TYPE to Integer, MIN and MAX to -10 and 10.

**Virtual Pin Datastream**

NAME: Yvalue ALIAS: Yvalue

PIN: V10 DATA TYPE: Integer

UNITS: None

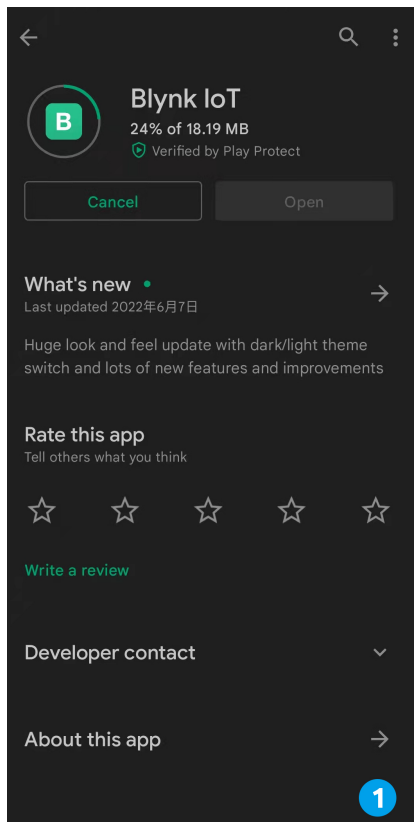
MIN: -10 MAX: 10 DEFAULT VALUE: 0

+ ADVANCED SETTINGS Cancel Create

Next you need to do the following on your phone.

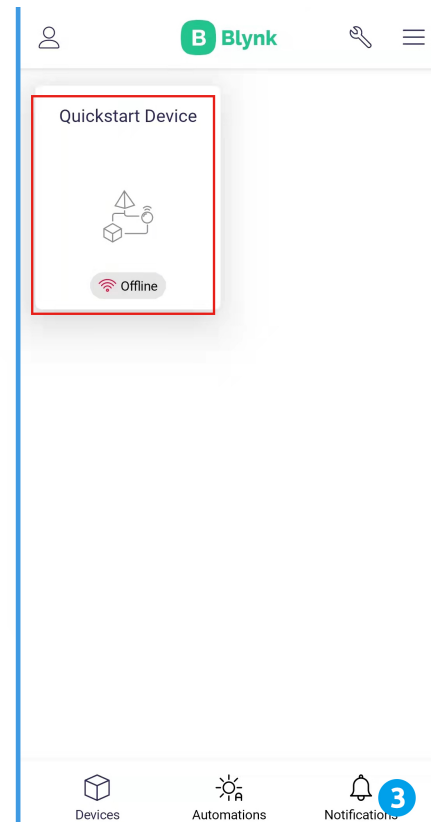
1. Search for “Blynk IoT” (not Blynk(legacy)) in GOOGLE Play or APP Store to download it.
2. After opening the APP, sign in, this account should be the same as the account used on the web client.
3. Then go to Dashboard (if you don’t have one, create one) and you will see that the Dashboard for mobile and web are independent of each other.



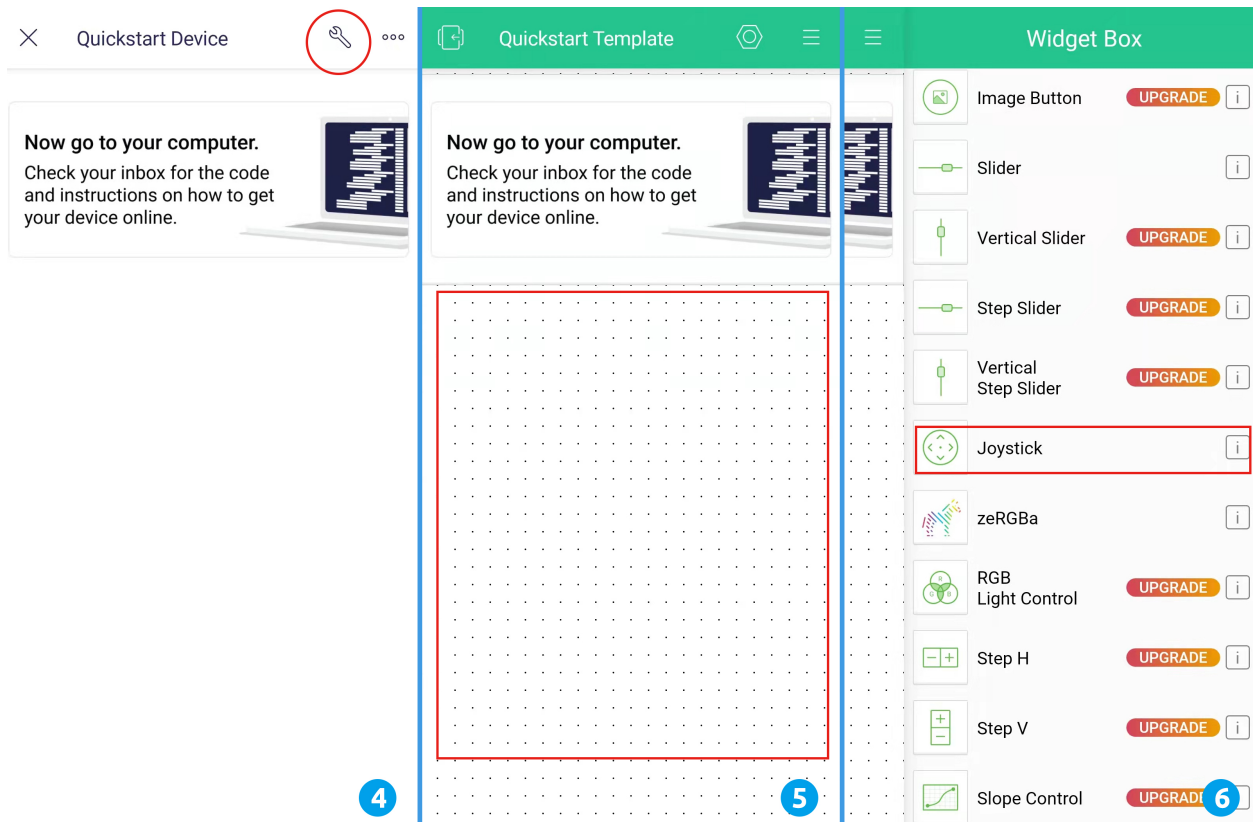


Sign Up

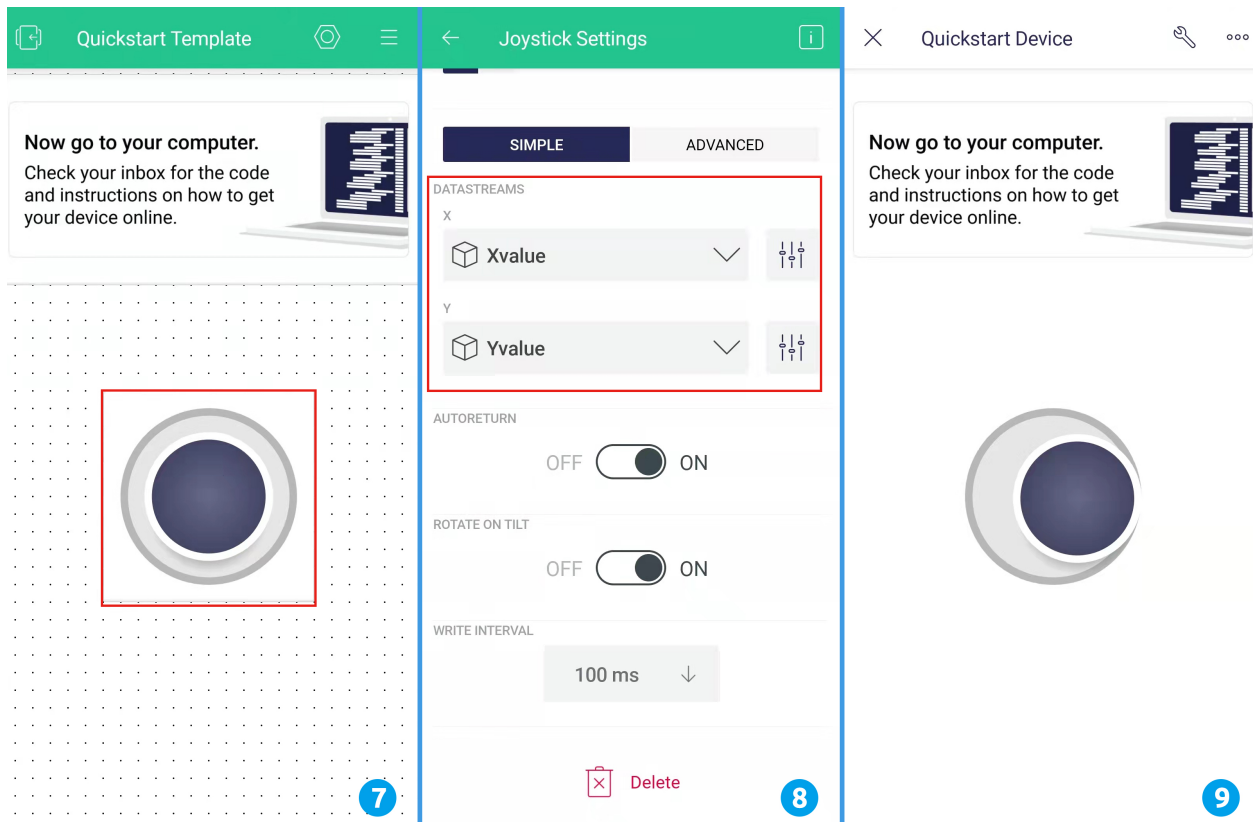
Log In



4. Click Edit Icon.
5. Click on the blank area.
6. Select a Joystick widget.

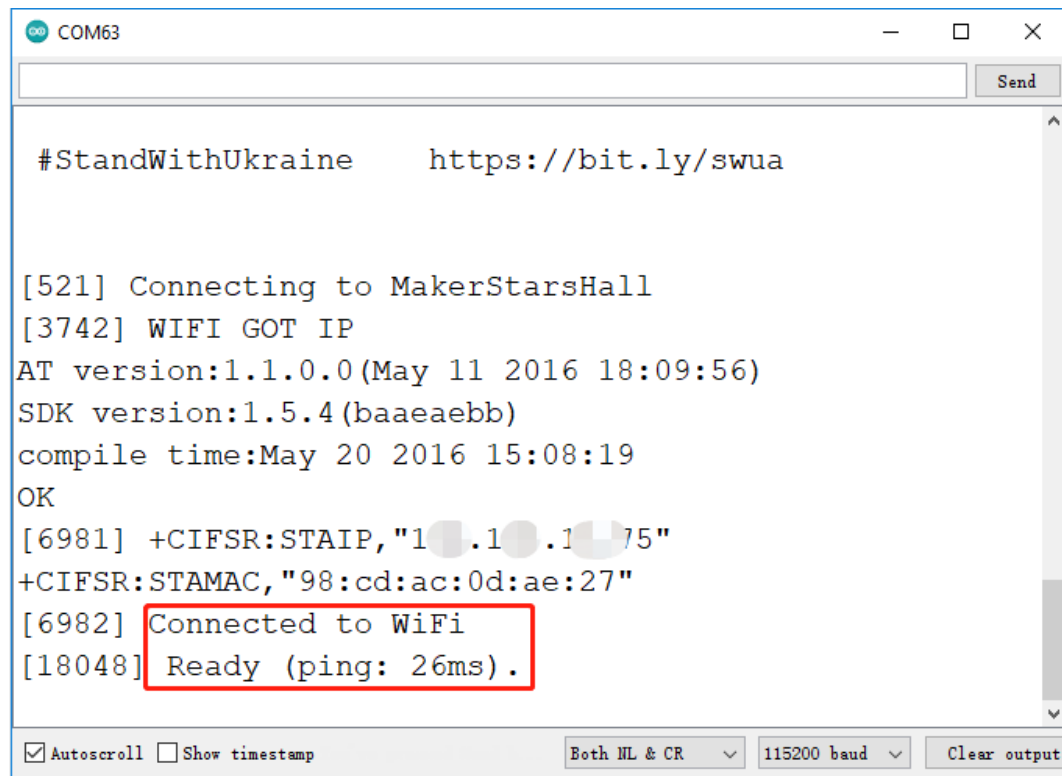


7. Now you will see a Joystick widget appear in the blank area, click on it.
8. Joystick Settings will appear, select the Xvalue and Yvalue you just set in the datastreams.
9. Go back to the Dashboard page and you can operate the Joystick when you want.



### 3. Run the Code

1. Open the `8.iot_car.ino` file under the path of `3in1-kit\iot_project\8.iot_car`, or copy this code into **Arduino IDE**.
2. Replace the Template ID, Device Name, and Auth Token with your own. You also need to enter the ssid and password of the WiFi you are using. For detailed tutorials, please refer to [1.4 Connecting the R4 board to Blynk](#).
3. After selecting the correct board and port, click the **Upload** button.
4. Open the Serial monitor(set baudrate to 115200) and wait for a prompt such as a successful connection to appear.



```
COM63

#StandWithUkraine  https://bit.ly/swua

[521] Connecting to MakerStarsHall
[3742] WIFI GOT IP
AT version:1.1.0.0(May 11 2016 18:09:56)
SDK version:1.5.4(baaeaebb)
compile time:May 20 2016 15:08:19
OK
[6981] +CIFSR:STAIP,"1.1.1.175"
+CIFSR:STAMAC,"98:cd:ac:0d:ae:27"
[6982] Connected to WiFi
[18048] Ready (ping: 26ms).

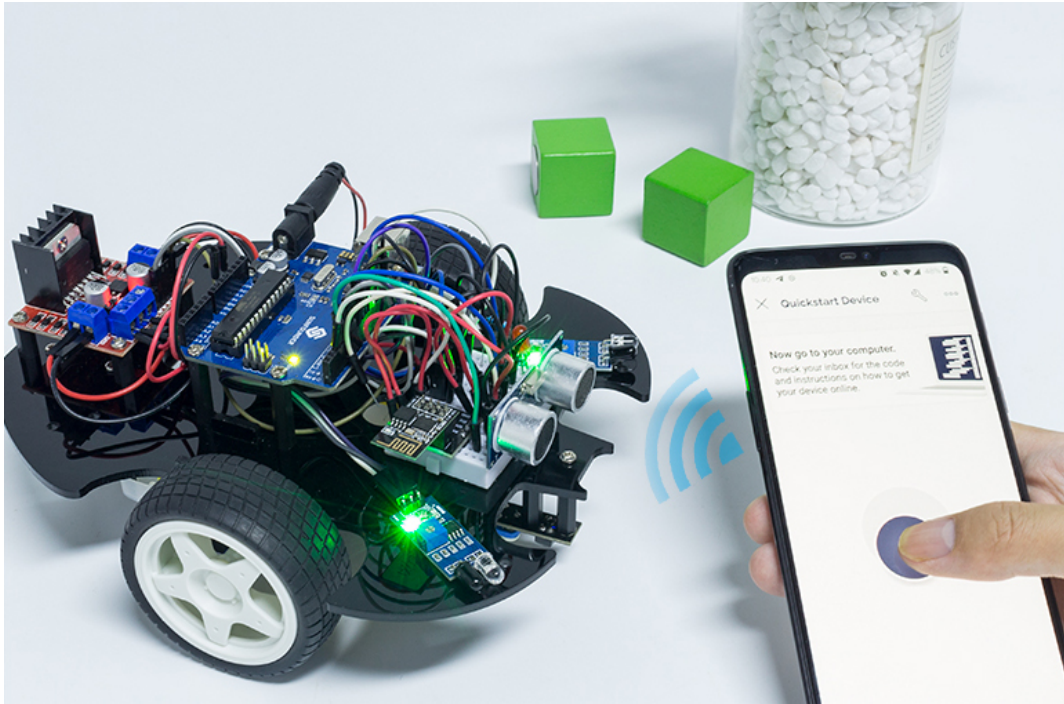
☒ Autoscroll ☐ Show timestamp
Both NL & CR 115200 baud Clear output
```

**Note:** If the message ESP is not responding appears when you connect, please follow these steps.

- Make sure the 9V battery is plugged in.
- Reset the ESP8266 module by connecting the pin RST to GND for 1 second, then unplug it.
- Press the reset button on the R4 board.

Sometimes, you may need to repeat the above operation 3-5 times, please be patient.

5. Now unplug the USB cable and power the cart with a 9V battery alone, then wait for the LED to light up, representing that the car is connected to Blynk.
6. Open Blynk on your phone and you can use the Joystick widget to control the movement of the car.



### How it works?

These functions are used to control the movement of the car.

```
void moveForward(int speed) {...}
void moveBackward(int speed) {...}
void turnRight(int speed) {...}
void turnLeft(int speed) {...}
void stopMove() {...}
```

The IoT section reads the values of the Joystick widget and assigns them to the variables Xvalue and Yvalue.

```
int Xvalue = 0;
int Yvalue = 0;

BLYNK_WRITE(V9)
{
    Xvalue = param.asInt();
}

BLYNK_WRITE(V10)
{
    Yvalue = param.asInt();
}
```

At loop(), make the car perform different actions based on Xvalue and Yvalue.

```
if (Yvalue >= 5) {
    moveForward(255);
} else if (Yvalue <= -5) {
    moveBackward(255);
} else if (Xvalue >= 5) {
```

(continues on next page)

(continued from previous page)

```
    turnRight(150);
} else if (Xvalue <= -5) {
    turnLeft(150);
} else {
    stopMove();
}
```

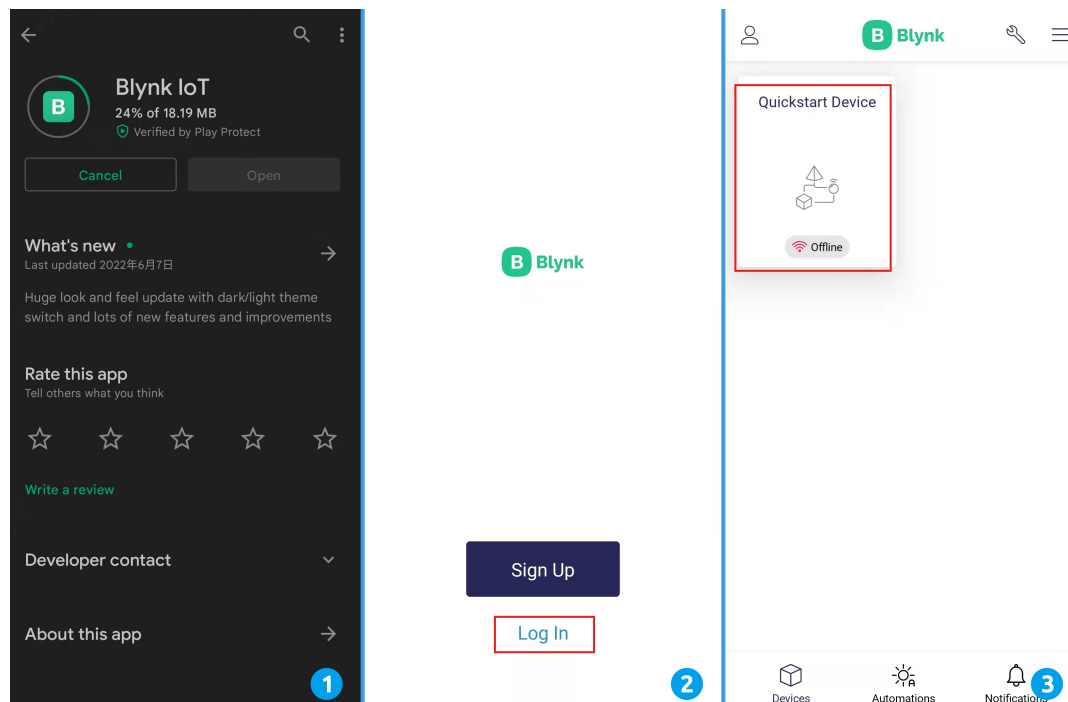
Also, add a network status determination to `loop()` to light up an LED if it is connected to Blynk Cloud.

```
if (!Blynk.connected()) {
    digitalWrite(ledPin, LOW);
    Serial.print("offline!");
    bool result = Blynk.connect();
    Serial.println(result);
} else {
    digitalWrite(ledPin, HIGH);
}
```

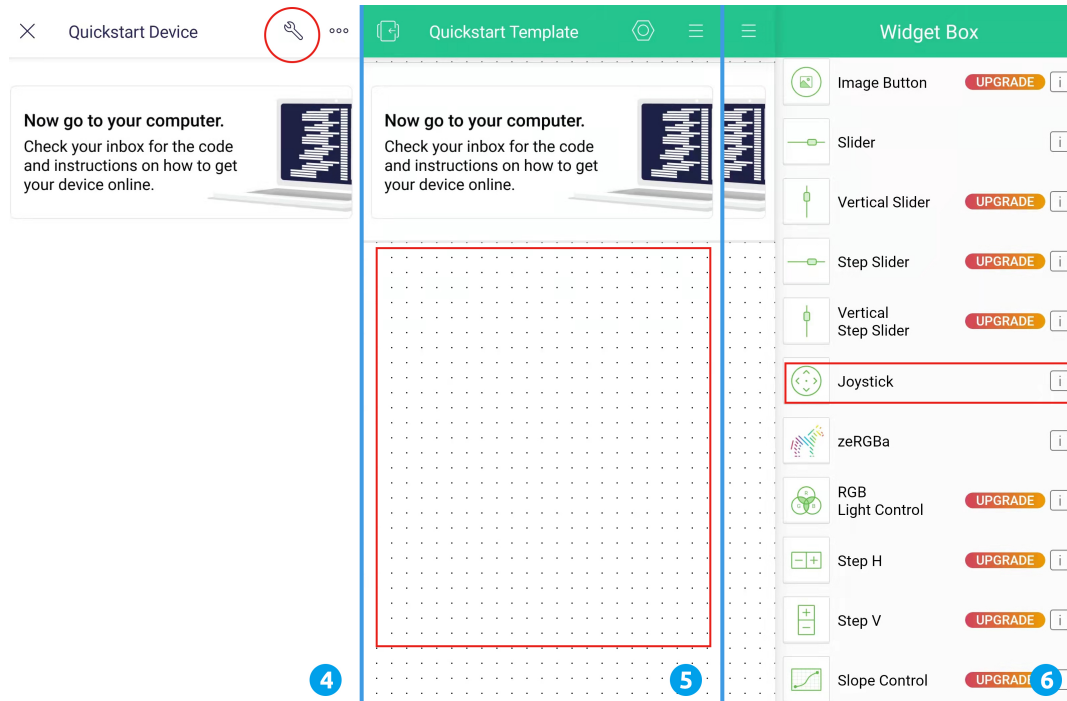
## 7.1 How to use Blynk on mobile device?

**Note:** As datastreams can only be created in Blynk on the web, you will need to reference different projects to create datastreams on the web, then follow the tutorial below to create widgets in Blynk on your mobile device.

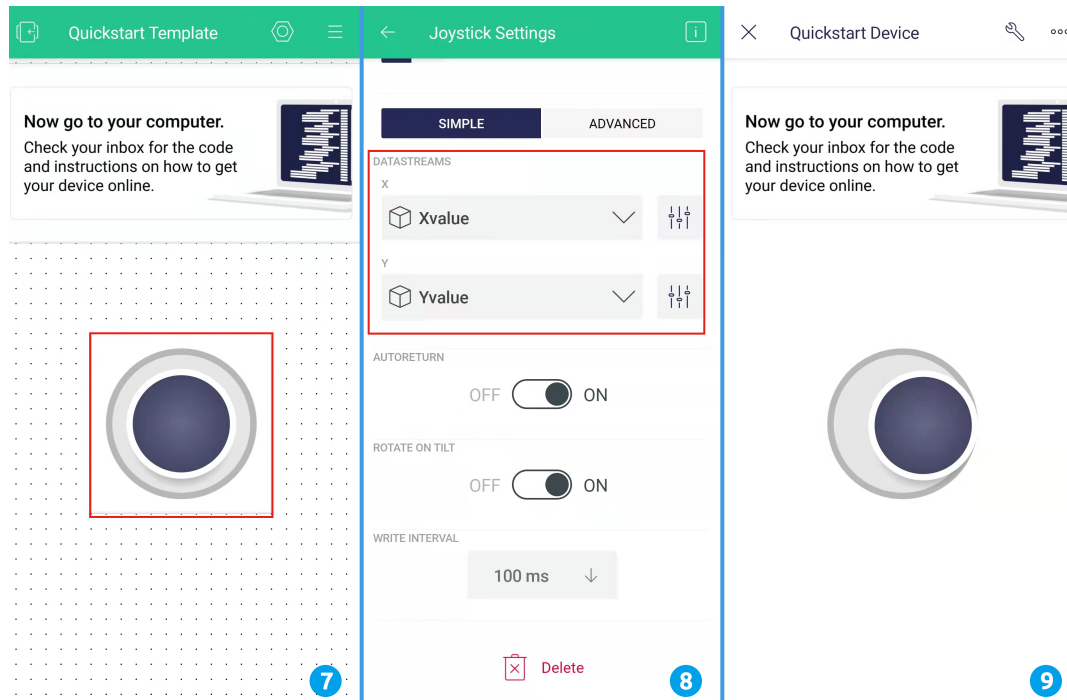
1. Open Google Play or APP Store on your mobile device and search for “Blynk IoT” (not Blynk(legacy)) to download.
2. After opening the APP, login in, this account should be the same as the account used on the web client.
3. Then go to **Dashboard** (if you don’t have one, create one) and you will see that the **Dashboard** for mobile and web are independent of each other.



4. Click **Edit** Icon.
5. Click on the blank area.
6. Choose the same widget as on the web page, such as select a **Joystick** widget.



7. Now you will see a **Joystick** widget appear in the blank area, click on it.
8. **Joystick** Settings will appear, select the **Xvalue** and **Yvalue** datastreams you just set in the web page. Note that each widget corresponds to a different datastream in each project.
9. Go back to the **Dashboard** page and you can operate the **Joystick** when you want.



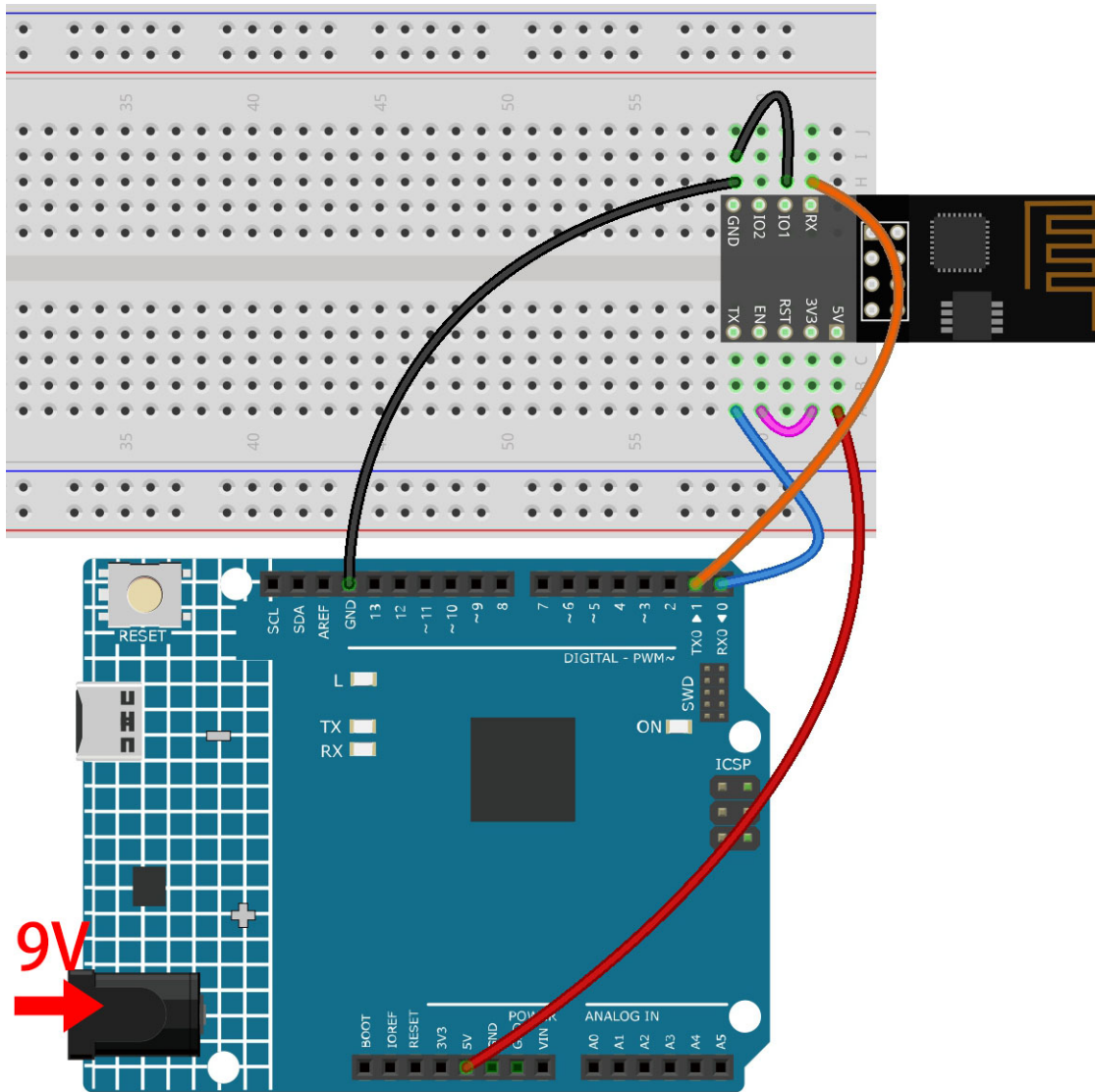


## 7.2 How to re-burn the firmware for ESP8266 module?

### 7.2.1 Re-brun the Firmware with R4

#### 1. Build the circuit

Connect ESP8266 and Arduino UNO R4 board.



#### 2. Upload the Following Code to R4

```
void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);
}

void loop() {
  if (Serial.available()) { // If anything comes in Serial (USB),
    Serial1.write(Serial.read()); // read it and send it out Serial1 (pins 0 & 1)
```

(continues on next page)

(continued from previous page)

```

    }
    if (Serial1.available()) { // If anything comes in Serial1 (pins 0 & 1)
        Serial.write(Serial1.read()); // read it and send it out Serial (USB)
    }
}

```

### 3. Burning the firmware

- Follow the steps below to burn the firmware if you are using **Windows**.
  1. Download firmware and burn-in tool.
    - ESP8266 Firmware
  2. After unzipping, you will see 4 files.
    - BAT\_AT\_V1.7.1.0\_1M.bin: The firmware to burn to the ESP8266 module.
    - esptool.exe: This is a command-line utility for Windows.
    - install\_r3.bat: This is the command package for Windows system, double click this file will run all the commands inside the file.
    - install\_r4.bat: Same as install\_r3.bat, but dedicated to UNO R4 board.
  3. Double click install\_r4.bat to start the firmware burning. If you see the following prompt, the firmware has been installed successfully.

```

C:\Windows\system32\cmd.exe

E:\Ba... d... in 1 Kit for A... \BAT_AT_V1.7.1.0_1M>esptool
.exe --chip esp8266 --before default_reset write_flash 0 "BAT_AT_V1.7.1.0_1M.bin
"
esptool.py v3.3
Found 2 serial ports
Serial port COM58
Connecting...
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 30:83:98:a4:69:21
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x000fffff...
Compressed 1048576 bytes to 327859...
Wrote 1048576 bytes (327859 compressed) at 0x00000000 in 29.7 seconds (effective
282.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

```

**Note:** If the burn-in fails, please check the following points.

- Reset the ESP8266 module by inserting the RST on the ESP8266 Adapter to GND and then unplugging it.
- Check if the wiring is correct.
- Whether the computer has recognized your board properly, and make sure the port is not occupied.

- Reopen the install.bat file.

- To burn the firmware, follow these steps if you are using a **Mac OS** system.

1. Use the following commands to install Esptool. Esptool is a Python-based, open-source, platform-independent utility to communicate with the ROM bootloader in Espressif chips.

```
python3 -m pip install --upgrade pip
python3 -m pip install esptool
```

2. If esptool is properly installed, it will output a message such as [usage: esptool] if you run `python3 -m esptool`.

3. Download firmware.

- ESP8266 Firmware

4. After unzipping, you will see 4 files.

- BAT\_AT\_V1.7.1.0\_1M.bin: The firmware to burn to the ESP8266 module.
- esptool.exe: This is a command-line utility for Windows.
- install\_r3.bat: This is the command package for Windows system.
- install\_r4.bat: Same as install\_r3.bat, but dedicated to UNO R4 board.

5. Open a terminal and use the `cd` command to go into the firmware folder you just downloaded, then run the following command to erase the existing firmware and re-burn the new firmware.

```
python3 -m esptool --chip esp8266 --before no_reset_no_sync erase_flash
python3 -m esptool --chip esp8266 --before no_reset_no_sync write_flash_
↪ 0 "BAT_AT_V1.7.1.0_1M.bin"
```

6. If you see the following prompt, the firmware has been installed successfully.



```
BAT_AT_V1.7.1.0_1M -- -bash -- 99x22
MacdeMacBook-Pro:BAT_AT_V1.7.1.0_1M mac$ python3 -m esptool --chip esp8266 --before default_reset w
rite_flash 0 "BAT_AT_V1.7.1.0_1M.bin"
esptool.py v4.3
[Found 3 serial ports
Serial port /dev/cu.usbmodem143401
Connecting...
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 30:83:98:a4:69:21
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x000fffff...
Compressed 1048576 bytes to 327859...
Wrote 1048576 bytes (327859 compressed) at 0x00000000 in 29.6 seconds (effective 283.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
MacdeMacBook-Pro:BAT_AT_V1.7.1.0_1M mac$
```

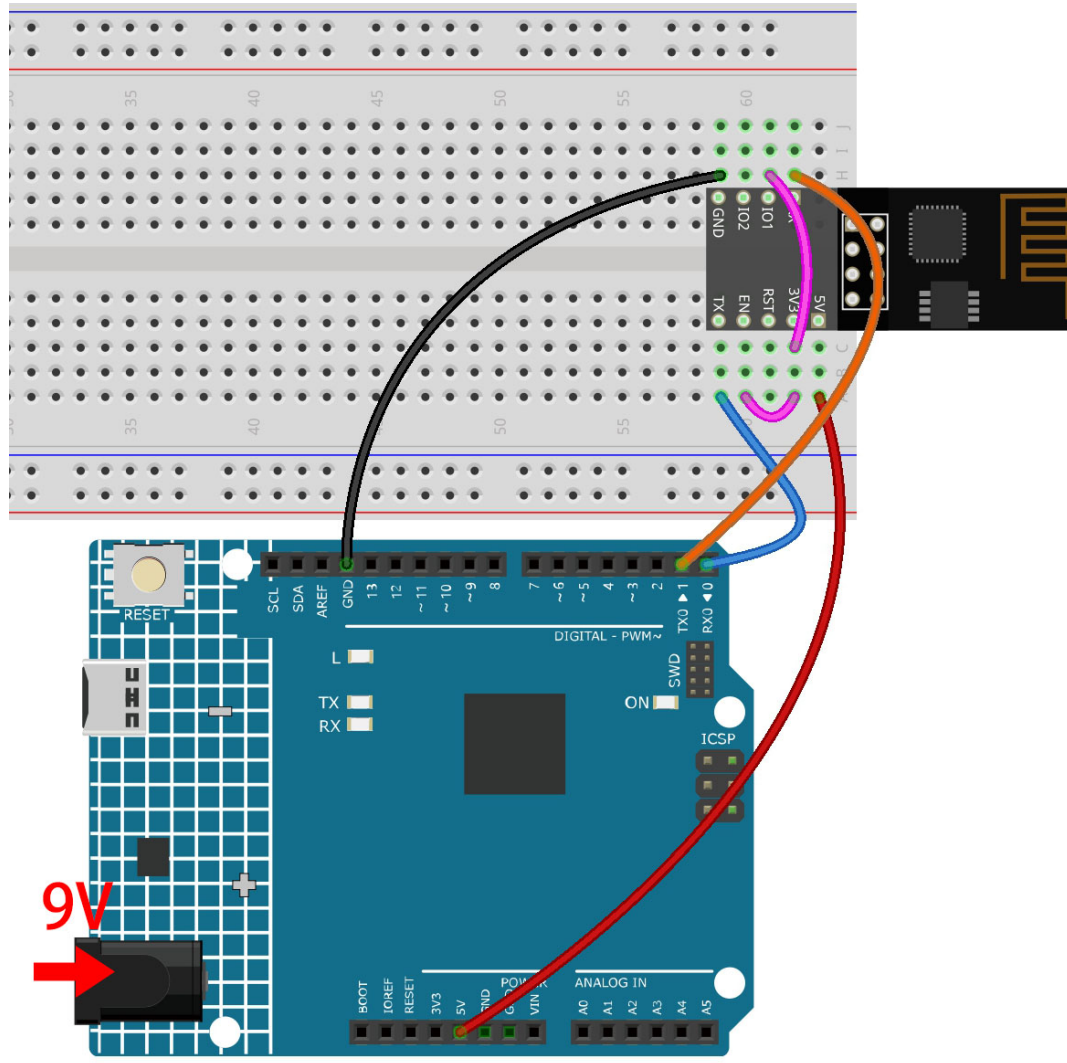
**Note:** If the burn-in fails, please check the following points.

- Reset the ESP8266 module by inserting the RST on the ESP8266 Adapter to GND and then unplugging it.
- Check if the wiring is correct.

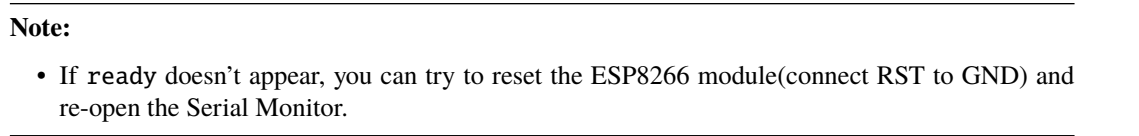
- Whether the computer has recognized your board properly, and make sure the port is not occupied.
- Reopen the install.bat file.

#### 4. Test

1. On the basis of the original wiring, connect IO1 to 3V3.



2. You will be able to see information about the ESP8266 module if you click the magnifying glass icon(Serial Monitor) in the upper right corner and set the baud rate to **115200**.



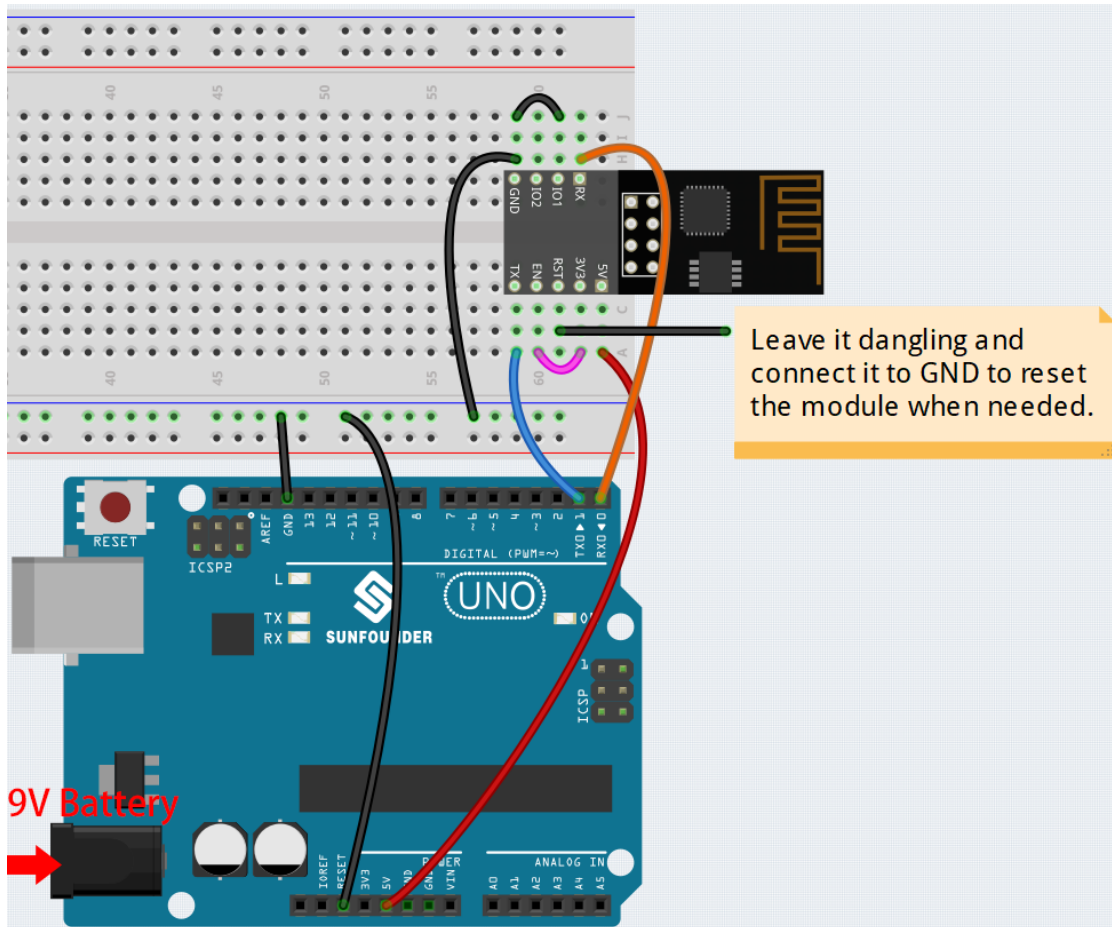
- 
- COM18
- AT<sup>2</sup>
- 14:56:06.776 -> load 0x40100000, len 27728, room 16
- 14:56:06.776 -> tail 0
- 14:56:06.776 -> chksum 0x2a
- 14:56:06.776 -> load 0x3ffe8000, len 2124, room 8
- 14:56:06.776 -> tail 4
- 14:56:06.776 -> chksum 0x07
- 14:56:06.776 -> load 0x3ffe8850, len 9276, room 4
- 14:56:06.776 -> tail 8
- 14:56:06.821 -> chksum 0xba
- 14:56:06.821 -> csum 0xba
- 14:56:07.004 -> \$\$\$r\$\$10\$<sup>3</sup>\$\$\$10\$<sup>3</sup>\$\$\$10\$<sup>3</sup>\$\$\$11'Orl\$\$\$s
- 14:56:07.096 -> ready
- 15:00:05.119 -> AT
- 15:00:05.119 ->
- 15:00:05.119 -> OK<sup>4</sup>
- ☒ Autoscroll ☒ Show timestamp <sup>1</sup>Both NL & CR 115200 baud Clear output

## 7.2. How to re-burn the firmware for ESP8266 module? 339

## 7.2.2 Re-brun the Firmware with R3

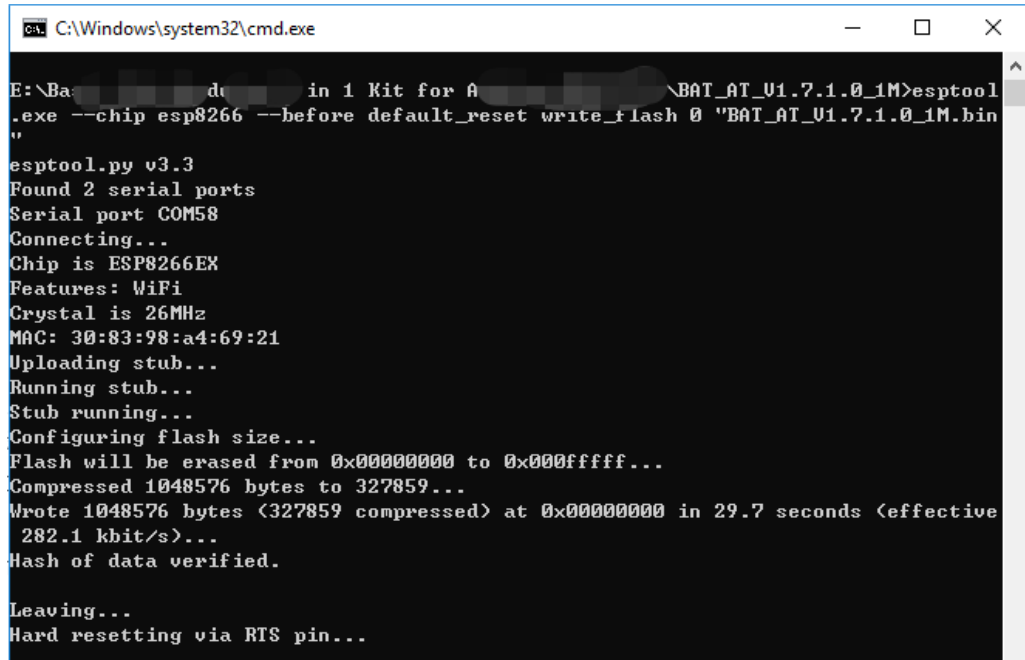
### 1. Build the circuit

Connect ESP8266 and SunFounder R3 board.



### 2. Burning the firmware

- Follow the steps below to burn the firmware if you are using **Windows**.
  1. Download firmware and burn-in tool.
    - ESP8266 Firmware
  2. After unzipping, you will see 4 files.
    - BAT\_AT\_V1.7.1.0\_1M.bin: The firmware to burn to the ESP8266 module.
    - esptool.exe: This is a command-line utility for Windows.
    - install\_r3.bat: This is the command package for Windows system, double click this file will run all the commands inside the file.
    - install\_r4.bat: Same as install\_r3.bat, but dedicated to UNO R4 board.
  3. Double click install\_r3.bat to start the firmware burning. If you see the following prompt, the firmware has been installed successfully.



```

C:\Windows\system32\cmd.exe

E:\Ba... in 1 Kit for A... \BAT_AT_V1.7.1.0_1M>esptool
.exe --chip esp8266 --before default_reset write_flash 0 "BAT_AT_V1.7.1.0_1M.bin"

esptool.py v3.3
Found 2 serial ports
Serial port COM58
Connecting...
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 30:83:98:a4:69:21
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x000fffff...
Compressed 1048576 bytes to 327859...
Wrote 1048576 bytes (327859 compressed) at 0x00000000 in 29.7 seconds (effective
282.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

```

**Note:** If the burn-in fails, please check the following points.

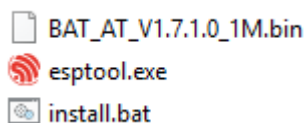
- Reset the ESP8266 module by inserting the RST on the ESP8266 Adapter to GND and then unplugging it.
- Check if the wiring is correct.
- Whether the computer has recognized your board properly, and make sure the port is not occupied.
- Reopen the install.bat file.

- To burn the firmware, follow these steps if you are using a **Mac OS** system.

1. Use the following commands to install Esptool. Esptool is a Python-based, open-source, platform-independent utility to communicate with the ROM bootloader in Espressif chips.

```
python3 -m pip install --upgrade pip
python3 -m pip install esptool
```

2. If esptool is properly installed, it will output a message such as [usage: esptool] if you run `python3 -m esptool`.
3. Download firmware.
  - ESP8266 Firmware
4. After unzipping, you will see 3 files.



BAT\_AT\_V1.7.1.0\_1M.bin  
esptool.exe  
install.bat

- BAT\_AT\_V1.7.1.0\_1M.bin: The firmware to burn to the ESP8266 module.



- esptool.exe: This is a command-line utility for Windows.
  - install\_r3.bat: This is the command package for Windows system.
  - install\_r4.bat: Same as install\_r3.bat, but dedicated to UNO R4 board.
5. Open a terminal and use the cd command to go into the firmware folder you just downloaded, then run the following command to erase the existing firmware and re-burn the new firmware.

```
python3 -m esptool --chip esp8266 --before default_reset erase_flash
python3 -m esptool --chip esp8266 --before default_reset write_flash 0
↪ "BAT_AT_V1.7.1.0_1M.bin"
```

6. If you see the following prompt, the firmware has been installed successfully.

```
MacdeMacBook-Pro:BAT_AT_V1.7.1.0_1M mac$ python3 -m esptool --chip esp8266 --before default_reset w
rite_flash 0 "BAT_AT_V1.7.1.0_1M.bin"
esptool.py v4.3
[Found 3 serial ports
Serial port /dev/cu.usbmodem143401
Connecting...
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 30:83:98:a4:69:21
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x000fffff...
Compressed 1048576 bytes to 327859...
Wrote 1048576 bytes (327859 compressed) at 0x00000000 in 29.6 seconds (effective 283.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
MacdeMacBook-Pro:BAT_AT_V1.7.1.0_1M mac$ █
```

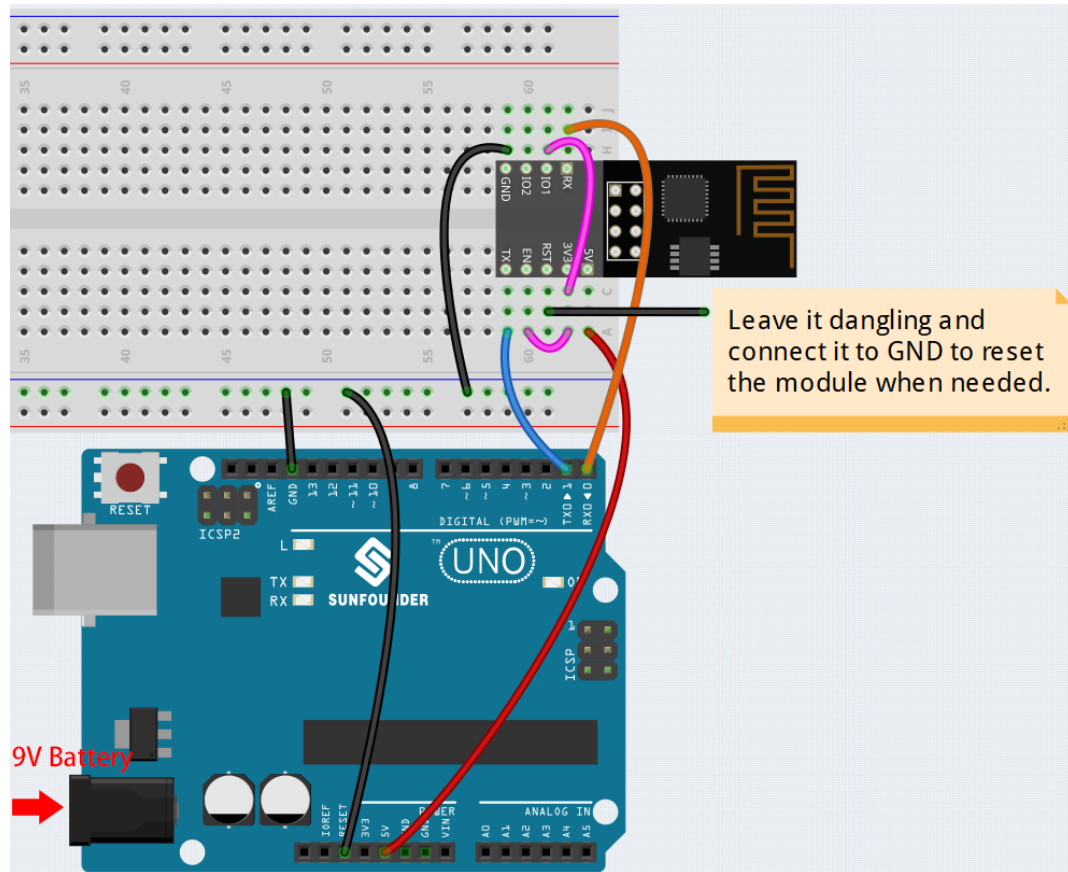
**Note:** If the burn-in fails, please check the following points.

- Reset the ESP8266 module by inserting the RST on the ESP8266 Adapter to GND and then unplugging it.
- Check if the wiring is correct.
- Whether the computer has recognized your board properly, and make sure the port is not occupied.
- Reopen the install.bat file.

### 3. Test

1. On the basis of the original wiring, connect IO1 to 3V3.





2. You will be able to see information about the ESP8266 module if you click the magnifying glass icon(Serial Monitor) in the upper right corner and set the baud rate to **115200**.

```

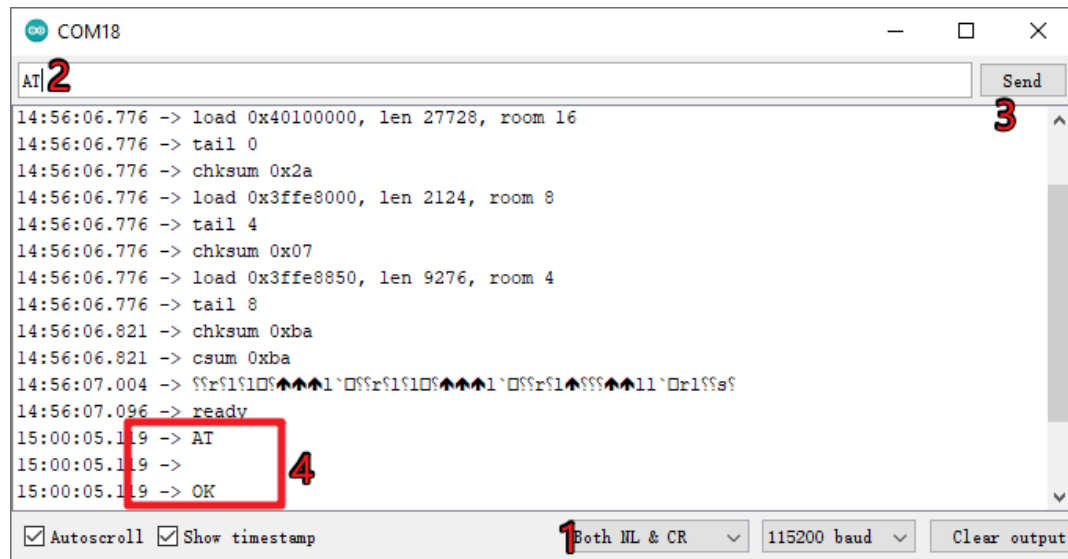
COM18
14:56:06.729 -> ets Jan 8 2013,rst cause:2, boot mode:(3,7)
14:56:06.776 ->
14:56:06.776 -> load 0x40100000, len 27728, room 16
14:56:06.776 -> tail 0
14:56:06.776 -> checksum 0x2a
14:56:06.776 -> load 0x3ffe8000, len 2124, room 8
14:56:06.776 -> tail 4
14:56:06.776 -> checksum 0x07
14:56:06.776 -> load 0x3ffe8850, len 9276, room 4
14:56:06.776 -> tail 8
14:56:06.821 -> checksum 0xba
14:56:06.821 -> csum 0xba
14:56:07.004 -> $$$r$1$10$1111`0$$r$1$10$1111`0$$r$1$111111`0r1$$$s$
14:56:07.096 -> ready
  
```

☒ Autoscroll ☒ Show timestamp No line ending 115200 baud Clear output

#### Note:

- If ready doesn't appear, you can try to reset the ESP8266 module(connect RST to GND) and re-open the Serial Monitor.

- Click on **NEWLINE DROPDOWN BOX**, select both NL & CR in the drop down option, enter AT, if it returns OK, it means ESP8266 has successfully established connection with R3 board.



Now you can continue to follow [1.1 Configuring the ESP8266](#) to set the working mode and baud rate of the ESP8266 module.

## THANK YOU

Thanks to the evaluators who evaluated our products, the veterans who provided suggestions for the tutorial, and the users who have been following and supporting us. Your valuable suggestions to us are our motivation to provide better products!

### Particular Thanks

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Now, could you spare a little time to fill out this questionnaire?

---

**Note:** After submitting the questionnaire, please go back to the top to view the results.

---



## **COPYRIGHT NOTICE**

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.