# SunFounder 3in1 Kit

**www.sunfounder.com**

**2024 年 02 月 19 日**

# 目次

SunFounder 3 in 1 スターターキットでをお選びいただき、ありがとうございます。

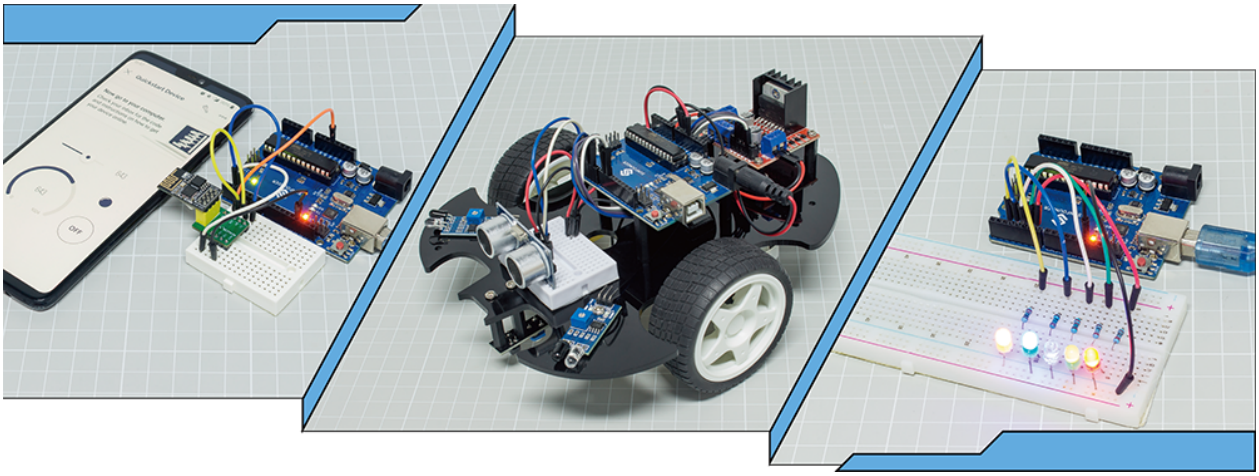
---

**注釈:** このドキュメントは以下の言語で利用可能です。

- 
- 
- 

ご希望の言語でドキュメントにアクセスするために、それぞれのリンクをクリックしてください。

---



オンラインで学習キットを購入したことはありますか？単なる PDF やブックレットが付属していて、プロジェクトの組み立て手順だけが書かれていましたか？

あるいは、自分でスマートカーを製作したいけど、オンラインで見つけたものは高価で複雑すぎると感じたことはありますか？

他の人が作った面白くて役立つ IoT プロジェクトを見て、どこから始めたらいいのかわからないことはありますか？

これらの問題は、私たちの 3 in 1 スターターキットで解決できます。

3 in 1 スターターキットには、Arduino を学ぶための完全なコースが含まれており、他の学習キットには提供されていないさまざまな興味深いプロジェクト、例えば、スマートカーのプロジェクトや IoT のプロジェクトも提供しています。キットのコースをステップバイステップで進めるだけで、コードをコピー&ペーストするのではなく、自分のコードを書いて Arduino プロジェクトを自由に実装できます。

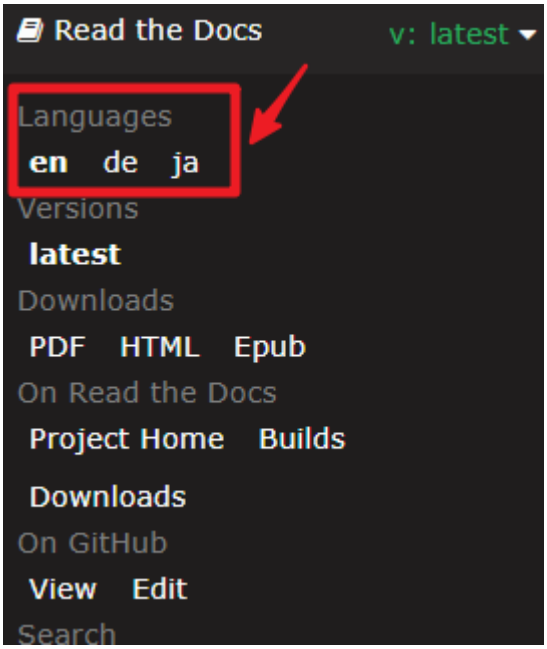
さらに、キットには 30 以上の Scratch プログラミングプロジェクトも提供しており、初心者はプログラミング経験がなくても自分の作品を作成できます！

さあ、ゼロからヒーローになるための Arduino プログラミングを始めましょう！

質問がある場合は、service@sunfounder.com までメールを送ってください。できるだけ早く返答いたします。

### 表示言語について

このドキュメントは、他の言語でも利用可能です。表示言語を切り替えるには、ページの左下にある **Read the Docs** アイコンをクリックしてください。




**Contents**

# 第 1 章

# キットのコンポーネントについて学ぶ

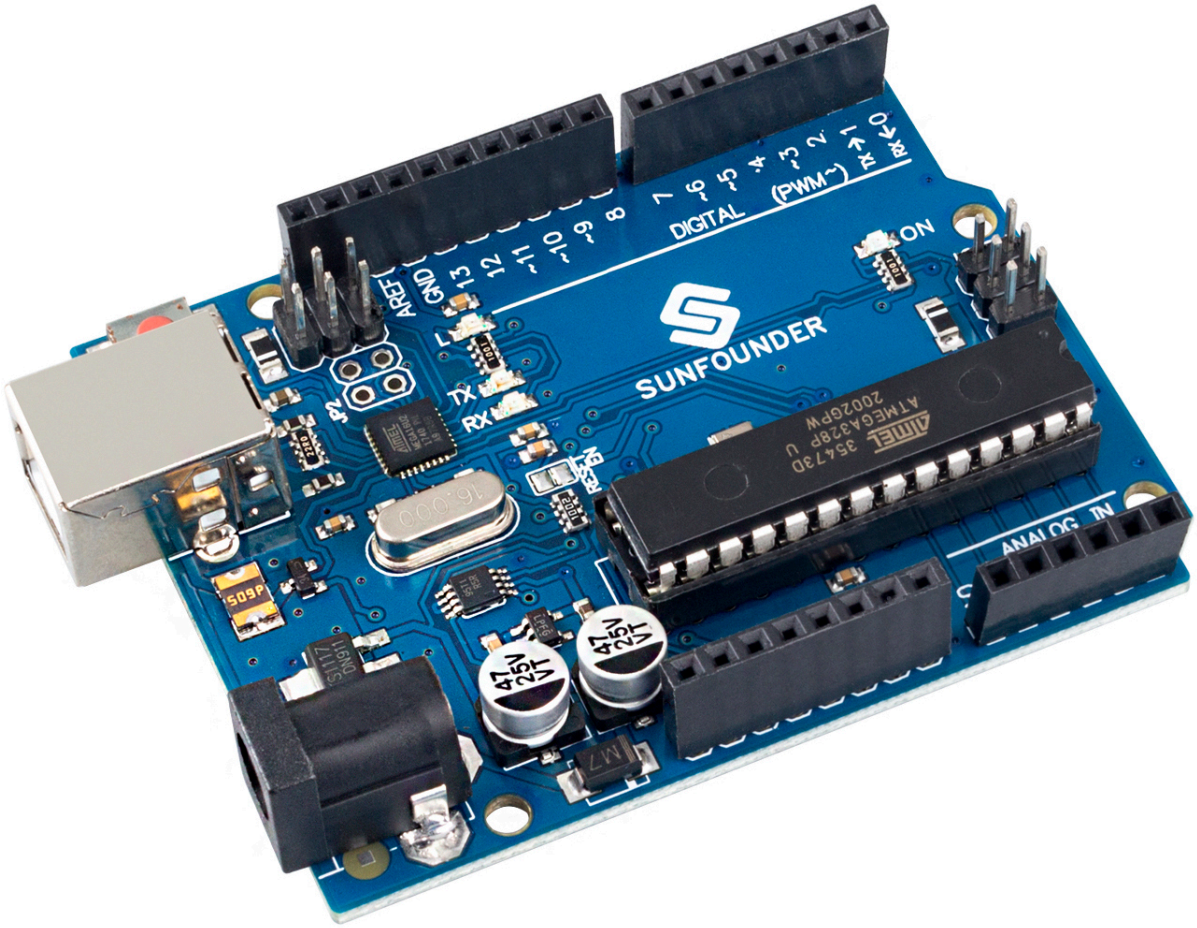
パッケージを開封した後、コンポーネントの数量が商品説明と一致しているか、およびすべてのコンポーネントが良好な状態であるかどうかを確認してください。

- 3 in 1 **キットコンポーネントリスト**

以下は、各コンポーネントの紹介です。これには、コンポーネントの動作原理と対応するプロジェクトが含まれています。

**Control Board**

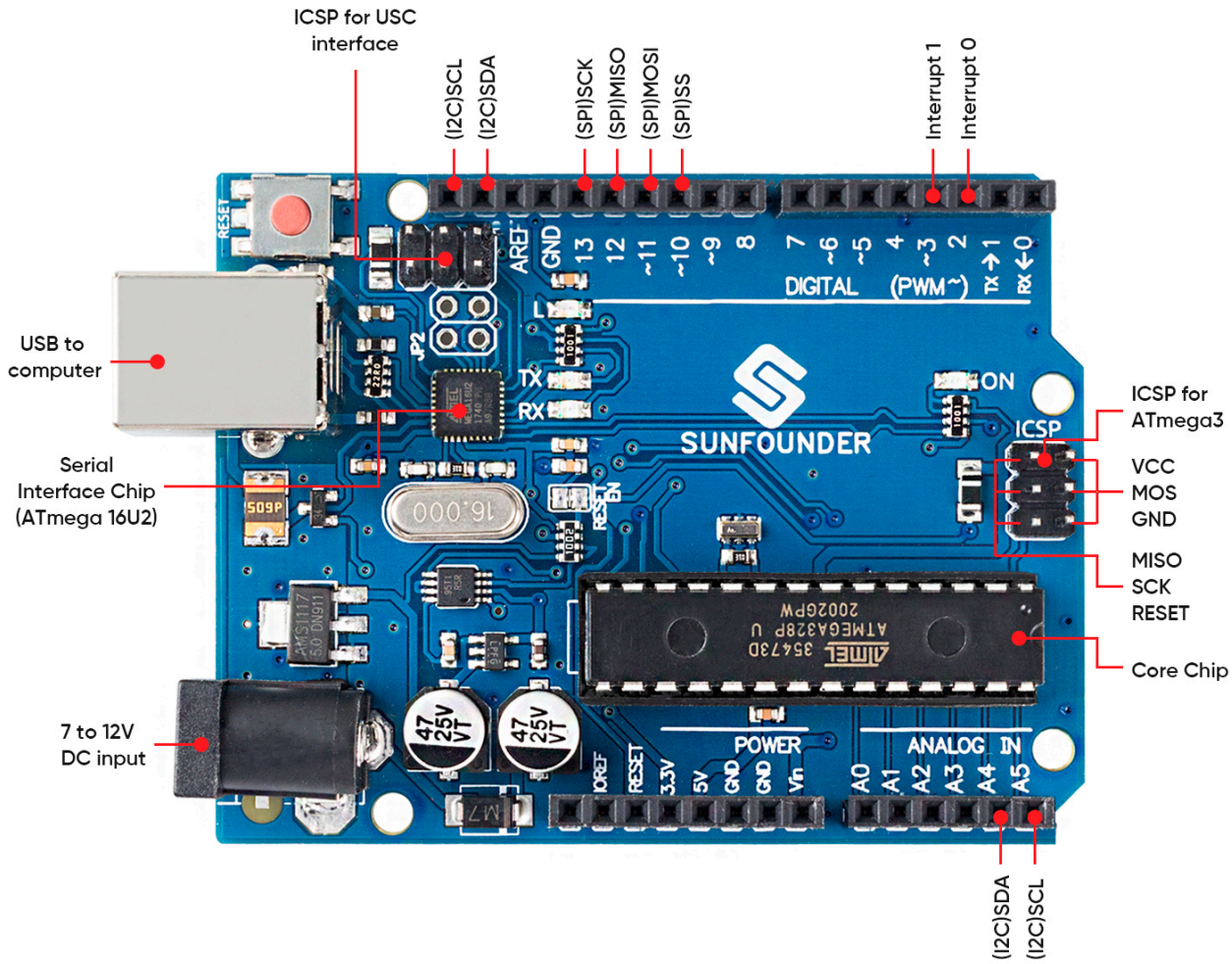
## 1.1 SunFounder R3 ボード




**注釈:** SunFounder R3 ボードは、 Arduino Uno とほとんど同じ機能を持つマザーボードです。両方のボードは互換性があります。

SunFounder R3 ボードは、ATmega328P (データシート) をベースとしたマイクロコントローラボードです。14 のデジタル入出力ピン（そのうち 6 つは PWM 出力として使用可能)、6 つのアナログ入力、16 MHz のセラミックレゾネーター (CSTCE16M0V53-R0)、USB 接続、電源ジャック、ICSP ヘッダ、リセットボタンを持っています。マイクロコントローラをサポートするために必要なものはすべて含まれているので、USB ケーブルでコンピュータに接続するか、AC から DC へのアダプタやバッテリーで電源を供給するだけで始めることができます。

**技術仕様**



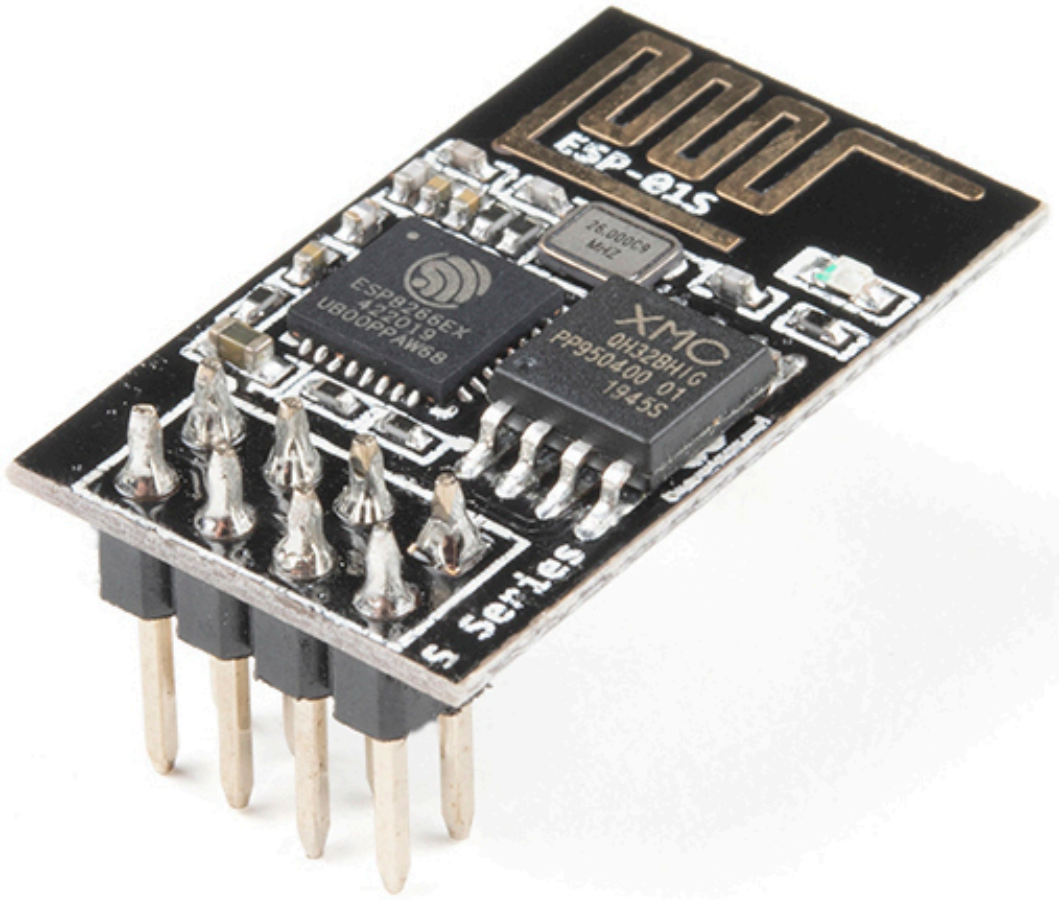

- マイクロコントローラ: ATmega328P
- 動作電圧: 5V
- 入力電圧 (推奨): 7-12V
- 入力電圧 (限界): 6-20V
- デジタル I/O ピン: 14 (0-13、うち 6 つは PWM 出力 (3, 5, 6, 9-11) 提供)
- PWM デジタル I/O ピン: 6 (3, 5, 6, 9-11)
- アナログ入力ピン: 6 (A0-A5)
- I/O ピン当たりの DC 電流: 20 mA
- 3.3V ピンの DC 電流: 50 mA
- フラッシュメモリ: 32 KB (ATmega328P) うち 0.5 KB はブートローダに使用
- SRAM: 2 KB (ATmega328P)

- EEPROM: 1 KB (ATmega328P)
- クロック速度: 16 MHz
- LED_BUILTIN: 13
- 長さ: 68.6 mm
- 幅: 53.4 mm
- 重量: 25 g
- I2C ポート: A4(SDA), A5(SCL)

**さらに**

- Arduino IDE
- Arduino プログラミング言語リファレンス
- *Arduino IDE 2.0* **のダウンロードとインストール**
- ATmega328P データシート

## 1.2 ESP8266 モジュール




ESP8266 は、低価格の Wi-Fi マイクロチップで、組み込みの TCP/IP ネットワーキングソフトウェアとマイクロコントローラー機能を持ち、中国上海の Espressif Systems によって製造されています。

このチップは、2014 年 8 月に Ai-Thinker というサードパーティ製造業者が製造した ESP-01 モジュールとともに、西洋のメーカーの注目を集めました。この小型のモジュールは、マイクロコントローラーが Wi-Fi ネットワークに接続し、Hayes 方式のコマンドを使用して簡単な TCP/IP 接続を行うことを可能にします。しかし、最初はこのチップとその受け入れるコマンドに関する英語のドキュメントはほとんどありませんでした。非常に低い価格と、モジュール上の外部コンポーネントが非常に少ないこと、そして量産されれば非常に安価になる可能性を示唆するこれらの事実が、多くのハッカーを引きつけてモジュール、チップ、その上のソフトウェアを探索し、中国語のドキュメントを翻訳することになりました。

ESP8266 のピンとその機能:

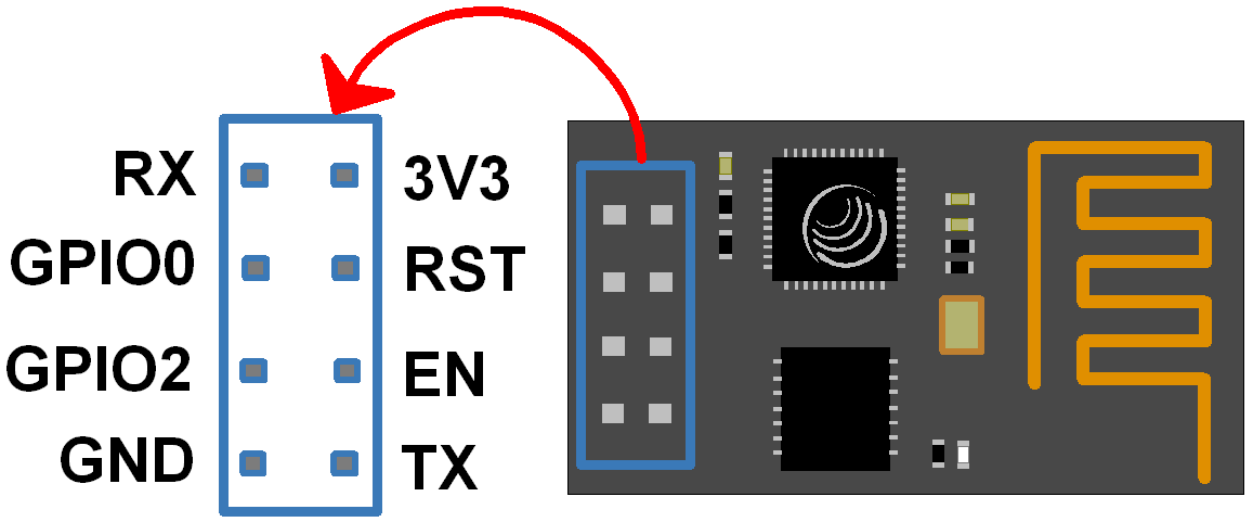
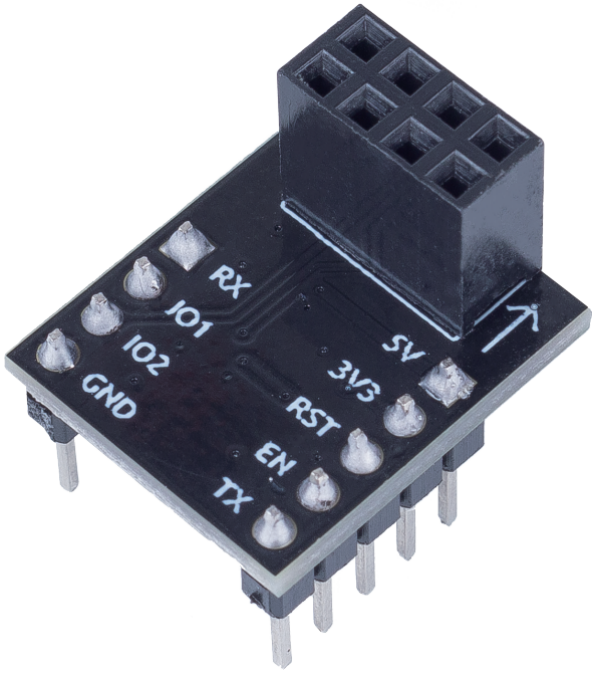



表 1　ESP8266-01 ピン

| ピン | 名称 | 説明 |
|---|---|---|
| 1 | TXD | UART_TXD、送信; 汎用入出力: GPIO1; 起動時のプルダウンは許可されていません。 |
| 2 | GND | GND |
| 3 | CU_PD | 高レベルで動作; 低レベルが供給されると電源オフ。 |
| 4 | GPIO2 | 電源投入時は高レベルでなければならず、ハードウェア的なプルダウンは許可されていません; デフォルトでプルアップ。 |
| 5 | RST | 外部リセット信号、低レベルが供給されるとリセット; 高レベル供給時に動作 (デフォルトは高レベル)。 |
| 6 | GPIO0 | Wi-Fi ステータスインジケータ; 動作モード選択: プルアップ: フラッシュブート, 動作モード; プルダウン: UART ダウンロード, ダウンロードモード。 |
| 7 | VCC | 電源供給 (3.3V) |
| 8 | RXD | UART_RXD、受信; 汎用入出力: GPIO3; |

- ESP8266 - Espressif
- ESP8266 AT 命令セット

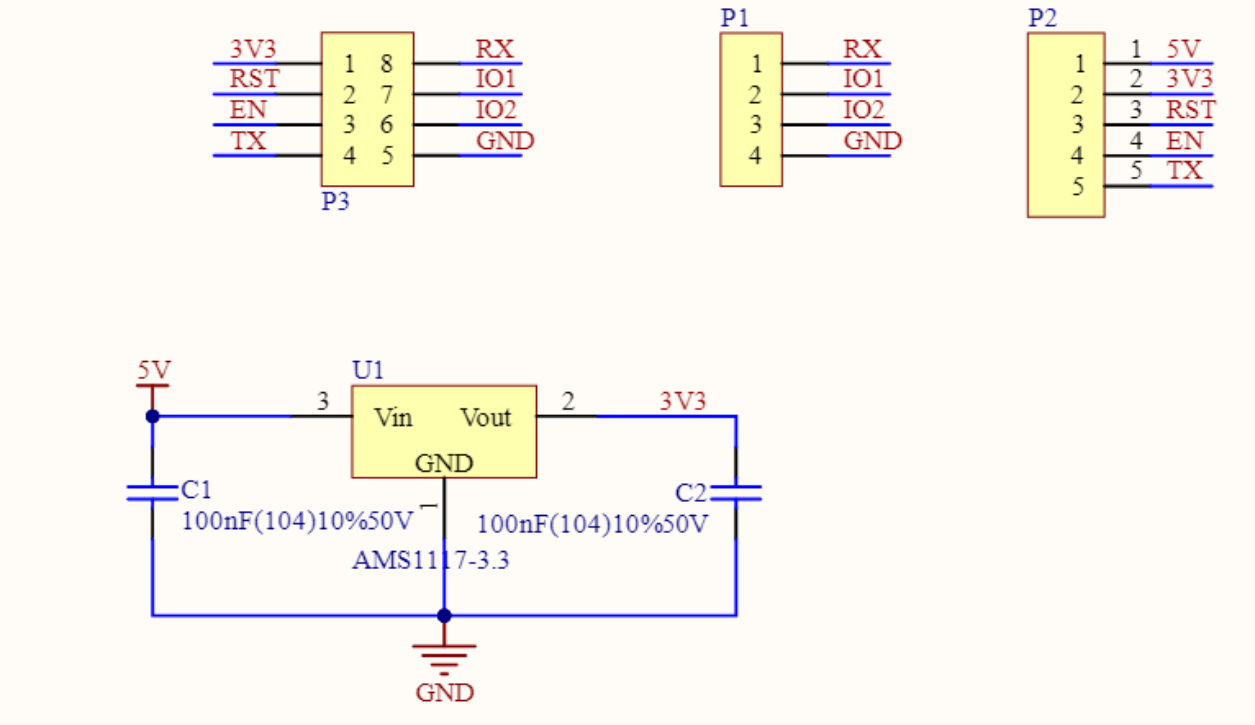
### 1.2.1 ESP8266 アダプタ




ESP8266 アダプタは、ESP8266 モジュールをブレッドボード上で使用できるようにする拡張ボードです。

これは ESP8266 のピン配置と完璧に一致しており、Arduino ボードからの電圧を受け取るための 5V ピンも追加されています。統合された AMS1117 チップは、電圧を 3.3V に落とした後で ESP8266 モジュールを駆動するために使用されます。

回路図は以下の通りです:

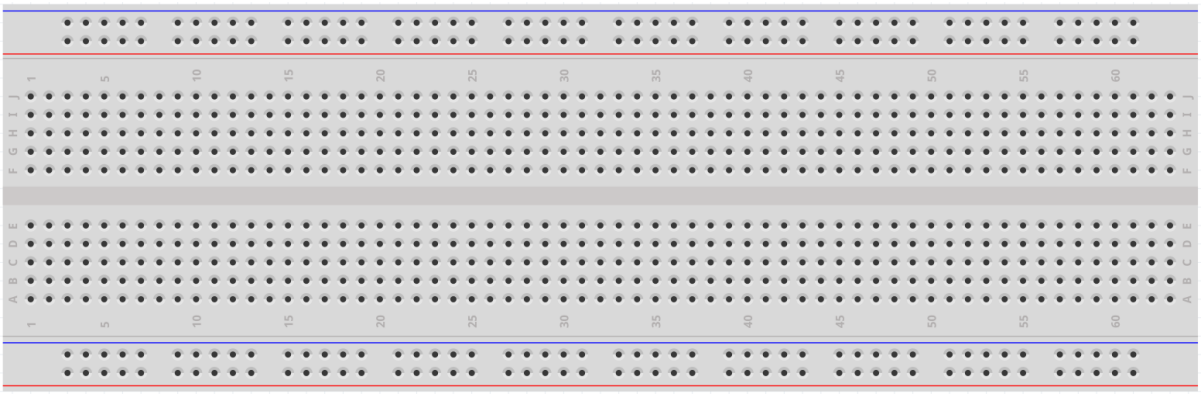



**例**

- *IoT* **プロジェクト** (IoT プロジェクト)

**Basic**

## 1.3 ブレッドボード

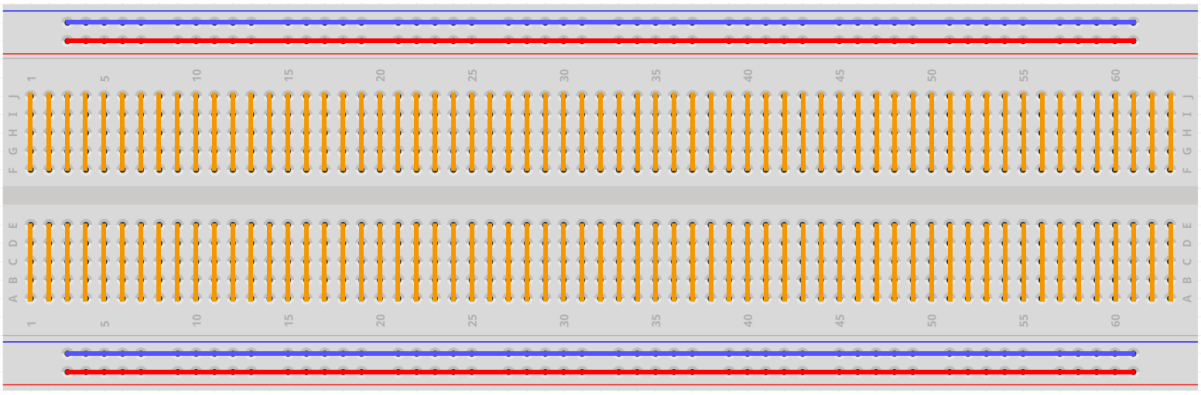


ブレッドボードは、電子機器のプロトタイピング用の基盤です。もともとこの言葉は、文字通りパンを切るための磨かれた木製の板を指していました [1]。1970 年代にはんだ不要のブレッドボード（別名：プラグボード、ターミナルアレイボード）が登場し、現在では「ブレッドボード」という言葉は一般的にこれを指して使われています。

ブレッドボードは、回路設計を完成させる前に、迅速に回路の構築やテストをするために使用されます。IC や抵

抗器、ジャンパーワイヤーなどの上記のコンポーネントを挿入するための多くの穴があります。ブレッドボードを使用すると、コンポーネントのプラグインや取り外しが容易になります。

写真はブレッドボードの内部構造を示しています。これらの穴は互いに独立しているように見えますが、内部では金属ストリップで互いに接続されています。



ブレッドボードについてもっと知りたい方は、以下を参照してください： ブレッドボードの使い方 - Science Buddies

## 1.4 抵抗器

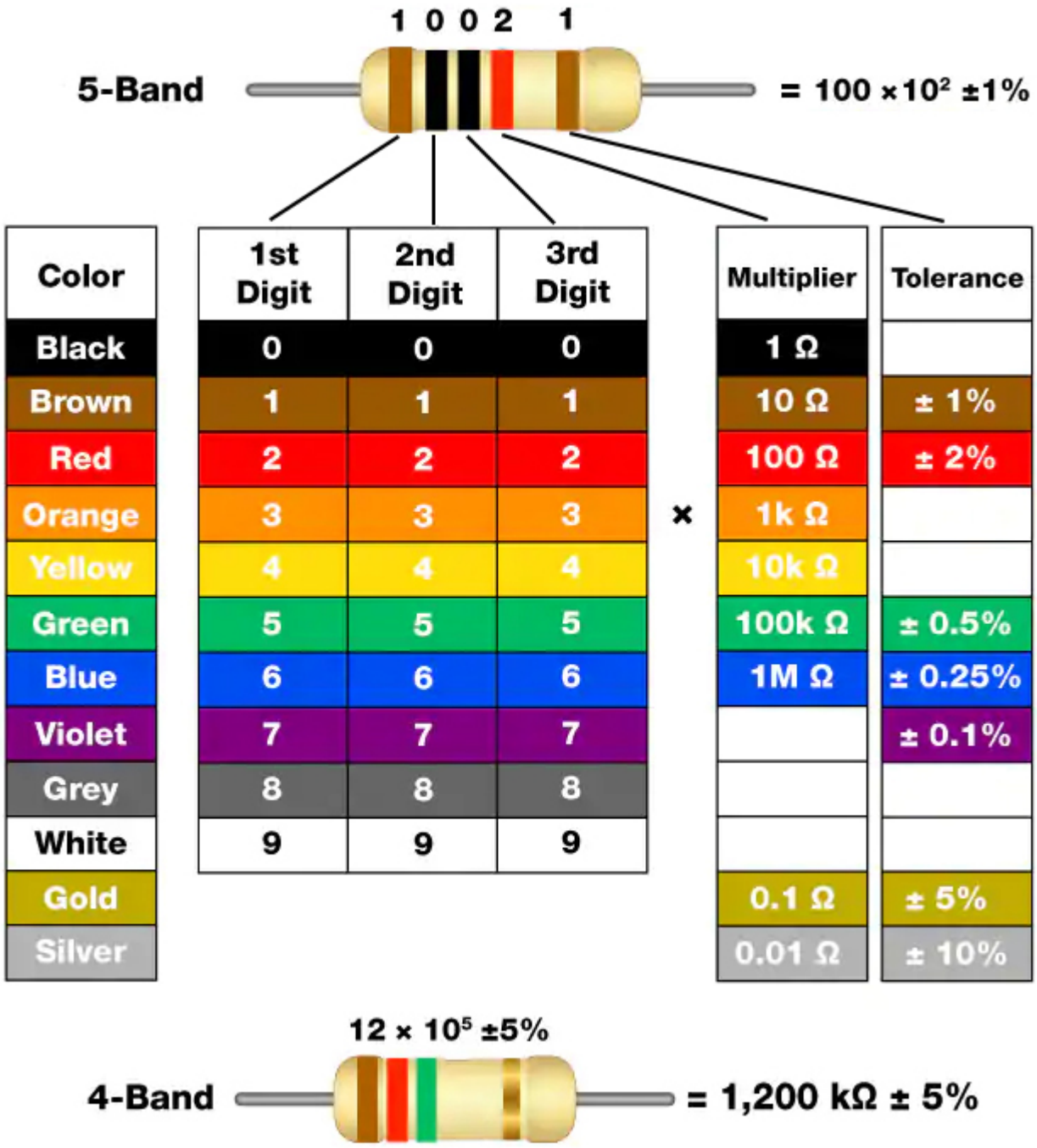
抵抗器は、分岐電流を制限することができる電子部品です。固定抵抗器は抵抗器値を変更することができない抵抗器であり、ポテンショメータや可変抵抗器の抵抗器値は調整可能です。

一般的に使用される抵抗器の回路記号です。通常、抵抗器値はこれに記載されています。したがって、回路内でこれらの記号を見た場合、それは抵抗器を示しています。



**Ω** は抵抗器の単位で、大きな単位には K Ω、M Ωなどがあります。これらの関係は次のように示されます：1 M Ω = 1000 K Ω、1 K Ω = 1000 Ω。通常、抵抗器の値はその上に記載されています。

抵抗器を使用する際には、まずその抵抗器値を知る必要があります。方法は 2 つあります：抵抗器のバンドを観察するか、マルチメータを使用して抵抗器値を測定する方法です。より便利で速いので、第一の方法を使用することをお勧めします。




このカードに示されているように、各色は数字を表します。

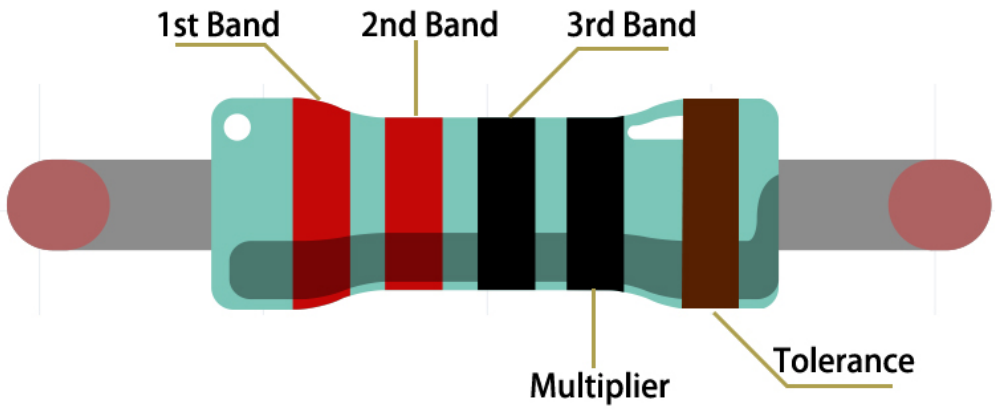
| 黒 | 茶 | 赤 | オレンジ | 黄 | 緑 | 青 | 紫 | 灰色 | 白 | 金 | 銀 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0.1 | 0.01 |

4 バンドおよび 5 バンドの抵抗器はよく使用されており、それぞれに 4 つおよび 5 つの彩色バンドがあります。

通常、抵抗器を取得すると、色を読み取るためのどちらの端を開始するかを判断するのが難しい場合があります。ヒントは、4 番目と 5 番目のバンドの間の間隔が比較的大きいことです。

したがって、抵抗器の一方の端で 2 つの彩色バンドの間の間隔を観察できます。他の任意のバンドの間隔よりも大きい場合、反対側から読むことができます。

以下に示す 5 バンドの抵抗器の抵抗器値の読み取り方法を見てみましょう。




この抵抗器の場合、抵抗器は左から右に読む必要があります。値は次のフォーマットである必要があります：1st Band 2nd Band 3rd Band x 10^Multiplier (Ω)、許容誤差は ±Tolerance% です。したがって、この抵抗器の抵抗器値は 2(赤) 2(赤) 0(黒) x 10^0(黒) Ω = 220 Ωで、許容誤差は ±1%(茶色) です。

Wiki で抵抗器についてもっと学ぶことができます： 抵抗器 - Wikipedia

## 1.5 コンデンサ


コンデンサは、与えられた電位差のもとでの電荷の蓄積量を指し、C として示され、国際単位はファラド (F) です。一般的に、電荷は電場内で力の下で移動します。導体の間に媒体が存在すると、電荷の移動は妨げられ、導体上に電荷が蓄積されます。

この蓄積された電荷の量は容量と呼ばれます。コンデンサは電子機器の中で最も広く使用される電子部品の一つであり、直流隔離、結合、バイパス、フィルタリング、チューニングループ、エネルギー変換、制御回路などの用途で幅広く利用されています。コンデンサは、電解コンデンサや固体コンデンサなどに分類されます。

材料特性に基づいて、コンデンサはアルミ電解コンデンサ、フィルムコンデンサ、タンタルコンデンサ、セラミックコンデンサ、スーパーコンデンサなどに分けられます。

このキットでは、セラミックコンデンサと電解コンデンサが使用されています。

- セラミックコンデンサ - Wikipedia
- 電解コンデンサ - Wikipedia

セラミックコンデンサには 103 や 104 のラベルがあり、これは容量値を表しており、103=10x10^3pF、104=10x10^4pF となります。

**単位変換**

1F=10^3mF=10^6uF=10^9nF=10^12pF

**例**

- *2.6* **ドアベル** (Scratch プロジェクト)

- *2.16* **ゲーム - りんごを食べる** (Scratch プロジェクト)
- *2.19* **ゲーム - 釣り** (Scratch プロジェクト)

## 1.6 ジャンパーワイヤー

二つの端子を接続するワイヤーをジャンパーワイヤーと呼びます。さまざまな種類のジャンパーワイヤーがありますが、ここではブレッドボードで使用されるものに焦点を当てます。特に、ブレッドボードの任意の位置からマイクロコントローラの入出力ピンに電気信号を伝送するために使用されます。

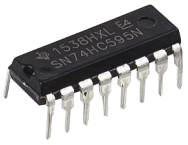
ジャンパーワイヤは、その"端子コネクタ"をブレッドボードに提供されるスロットに挿入することで取り付けられ、その表面の下には、エリアに応じて行または列のグループでスロットを接続する平行なプレートのセットがいくつかあります。 "端子コネクタ"は、特定のプロトタイプで接続する必要がある特定のスロットにはんだ付けせずにブレッドボードに挿入されます。

ジャンパーワイヤーには、メス-メス、オス-オス、およびオス-メスの 3 つのタイプがあります。オスからメスと呼ぶ理由は、一方の端に突出した先端と沈んだメスの端があるためです。オス-オスは両側がオスであり、メス-メスは両端がメスであることを意味します。


プロジェクトには複数のタイプが使用されることがあります。ジャンプワイヤの色は異なりますが、それがそれぞれの機能が異なることを意味するわけではありません。それは各回路間の接続をより簡単に識別するために設計されています。

**Chip**

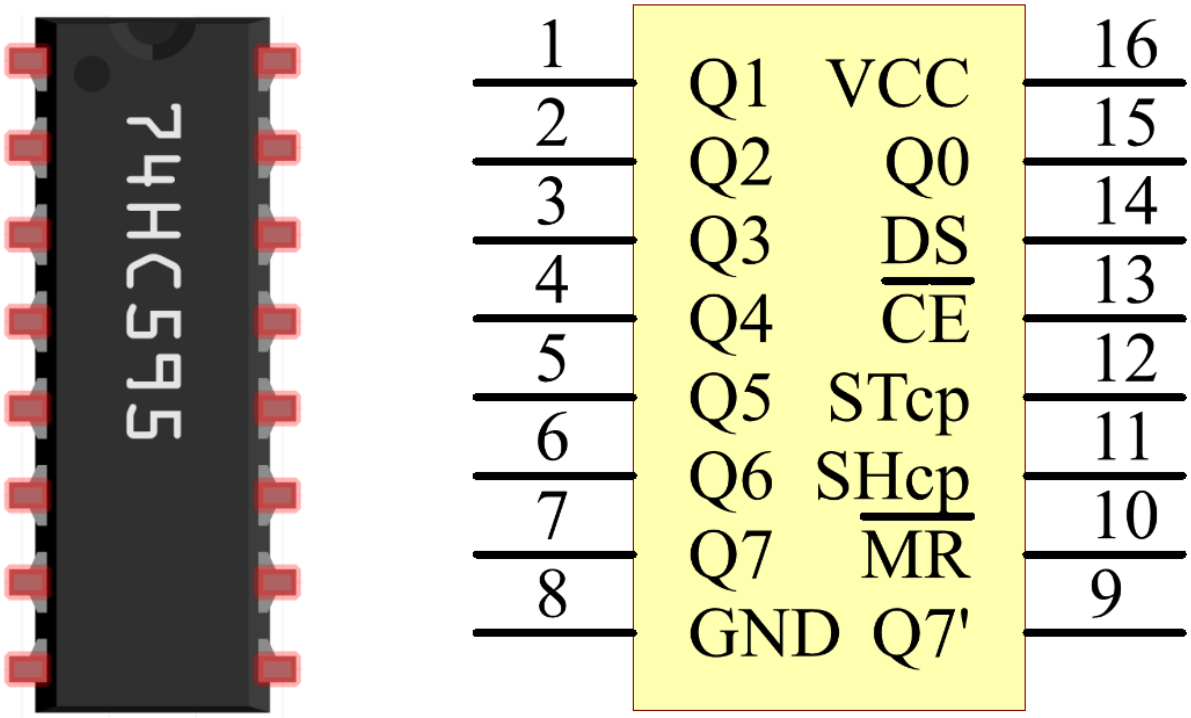
## 1.7 74HC595



74HC595 は 8 ビットのシフトレジスタと三状態の並列出力を持つ格納レジスタから成り立っています。シリアル入力を並列出力に変換することで、MCU の IO ポートを節約できます。MR (pin10) がハイレベル、OE (pin13) がローレベルの時、SHcp の立ち上がりエッジでデータが入力され、SHcp の立ち上がりエッジを通じてメモリレジスタに移動します。2 つのクロックが連結されている場合、シフトレジスタは常にメモリレジスタよりも 1 パルス

早いです。メモリレジスタには、シリアルシフト入力ピン (Ds)、シリアル出力ピン (Q)、非同期リセットボタン (ローレベル) があります。メモリレジスタは三状態の並列 8 ビットでバスを出力します。OE が有効 (ローレベル) のとき、メモリレジスタのデータがバスに出力されます。

- 74HC595 データシート




74HC595 のピンとその機能:

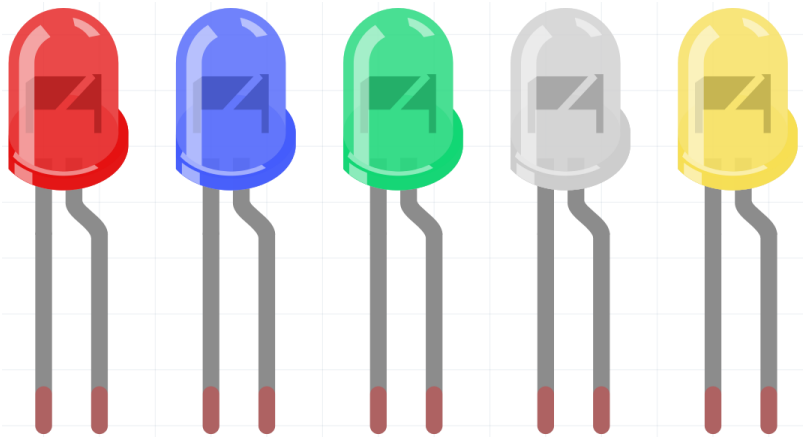
- **Q0-Q7**: 8 ビット並列データ出力ピン。8 つの LED や 7 セグメントディスプレイの 8 ピンを直接制御できる。
- **Q7 '** : シリーズ出力ピン。別の 74HC595 の DS に接続し、複数の 74HC595 を直列に接続する。
- **MR**: ローレベルでアクティブなリセットピン。
- **SHcp**: シフトレジスタのタイムシーケンス入力。立ち上がりエッジでは、シフトレジスタ内のデータが逐次的に 1 ビット移動する。
- **STcp**: ストレージレジスタのタイムシーケンス入力。立ち上がりエッジで、シフトレジスタのデータがメモリレジスタに移動する。
- **CE**: ローレベルでアクティブな出力有効ピン。
- **DS**: シリアルデータ入力ピン
- **VCC**: 正の供給電圧。
- **GND**: アース。

**例**

- *5.9 ShiftOut(LED)* (基本プロジェクト)
- *5.10 ShiftOut(7* **セグメント表示***)* (基本プロジェクト)
- *7.* **電流制限ゲート** (IoT プロジェクト)

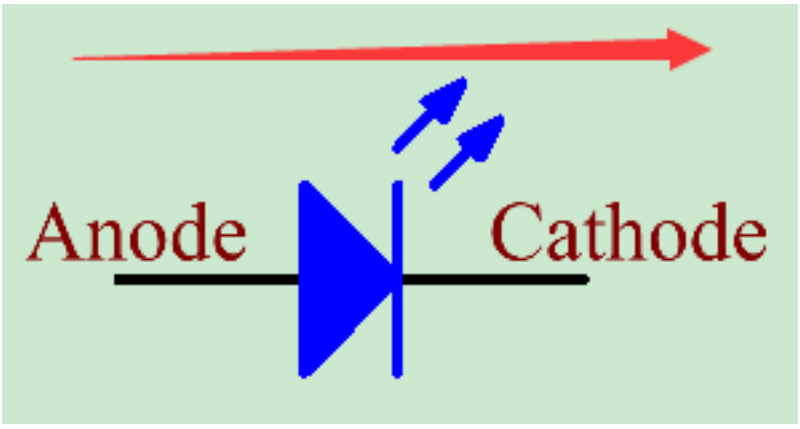
**Display**

## 1.8 LED



半導体発光ダイオードは、PN 接合を介して電気エネルギーを光エネルギーに変える部品の一種です。波長により、レーザダイオード、赤外発光ダイオード、そして通常 LED として知られる可視光発光ダイオードに分類されます。

ダイオードは一方向の導通性を持ちますので、回路記号の矢印が示す方向に電流が流れます。陽極に正の電源を供給し、カソードにマイナスを供給することで、LED は点灯します。




LED には 2 つのピンがあります。長い方が陽極で、短い方がカソードです。逆に接続しないように注意してください。LED には一定の順方向電圧降下がありますので、直接回路に接続することはできません。赤、黄、緑の LED の順方向電圧は 1.8V、白色は 2.6V です。ほとんどの LED は最大 20mA の電流に耐えることができますの

で、直列に電流制限抵抗を接続する必要があります。

抵抗値の計算式は次のとおりです：

R = (Vsupply – VD)/I

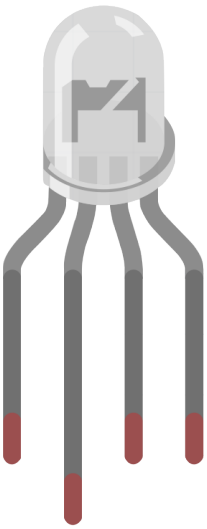
**R** は電流制限抵抗の抵抗値、 **Vsupply** は供給電圧、 **VD** は電圧降下、 **I** は LED の動作電流を示します。

詳しい LED の紹介はこちら： LED - Wikipedia。

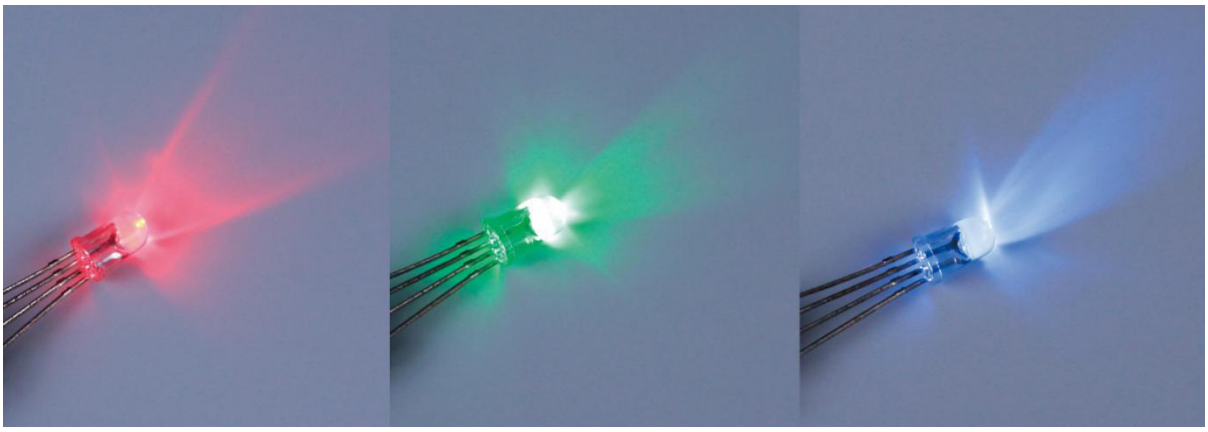
**例**

- *1.1* **ハロー、*LED*！** (基本プロジェクト)
- *2.1* **フェージング** (基本プロジェクト)
- *2. Blynk* **からデータを取得** (IoT プロジェクト)
- *2.2* **ブリージング** *LED* (Scratch プロジェクト)
- *2.1* **テーブルランプ** (Scratch プロジェクト)

## 1.9 RGB LED

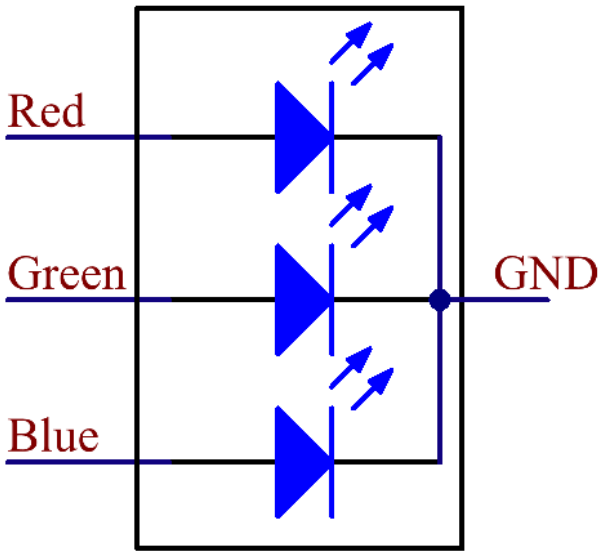


RGB LED は、さまざまな色の光を発するものです。赤、緑、青の 3 色の LED を透明または半透明のプラスチックケースに収めたものです。3 つのピンの入力電圧を変えて重ね合わせることで、統計によれば 1677 万 7216 色の異なる色を作り出すことができます。

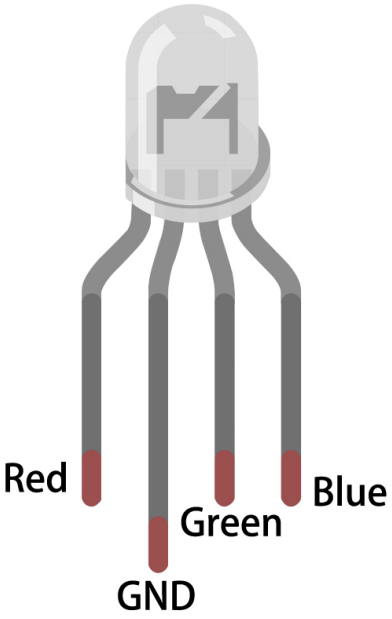


RGB LED には、共通アノード型と共通カソード型の 2 種類があります。このキットでは後者を使用しています。**共通カソード** 、または CC、は 3 つの LED のカソードを接続することを意味します。これを GND に接続し、3 つのピンを挿入すると、LED は対応する色で点滅します。

その回路記号は、以下の図のように示されています。



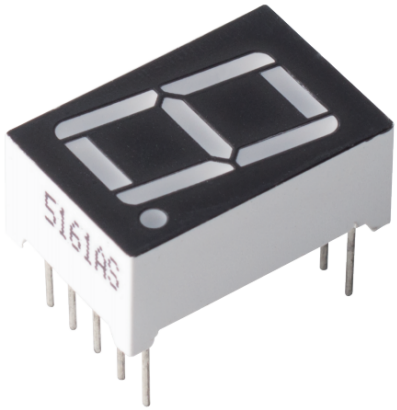

RGB LED には 4 つのピンがあります。最も長いものが GND で、残りの 3 つは赤、緑、青です。プラスチックケースに触れると、切り込みがあります。その切り込みに最も近いピンが最初のピンで、赤としてマークされ、次に GND、緑、青の順になります。




**例**

- *2.2* **カラフルな光** (基本プロジェクト)
- *5.2* **しきい値** (基本プロジェクト)
- *2.3* **カラフルボール** (Scratch プロジェクト)

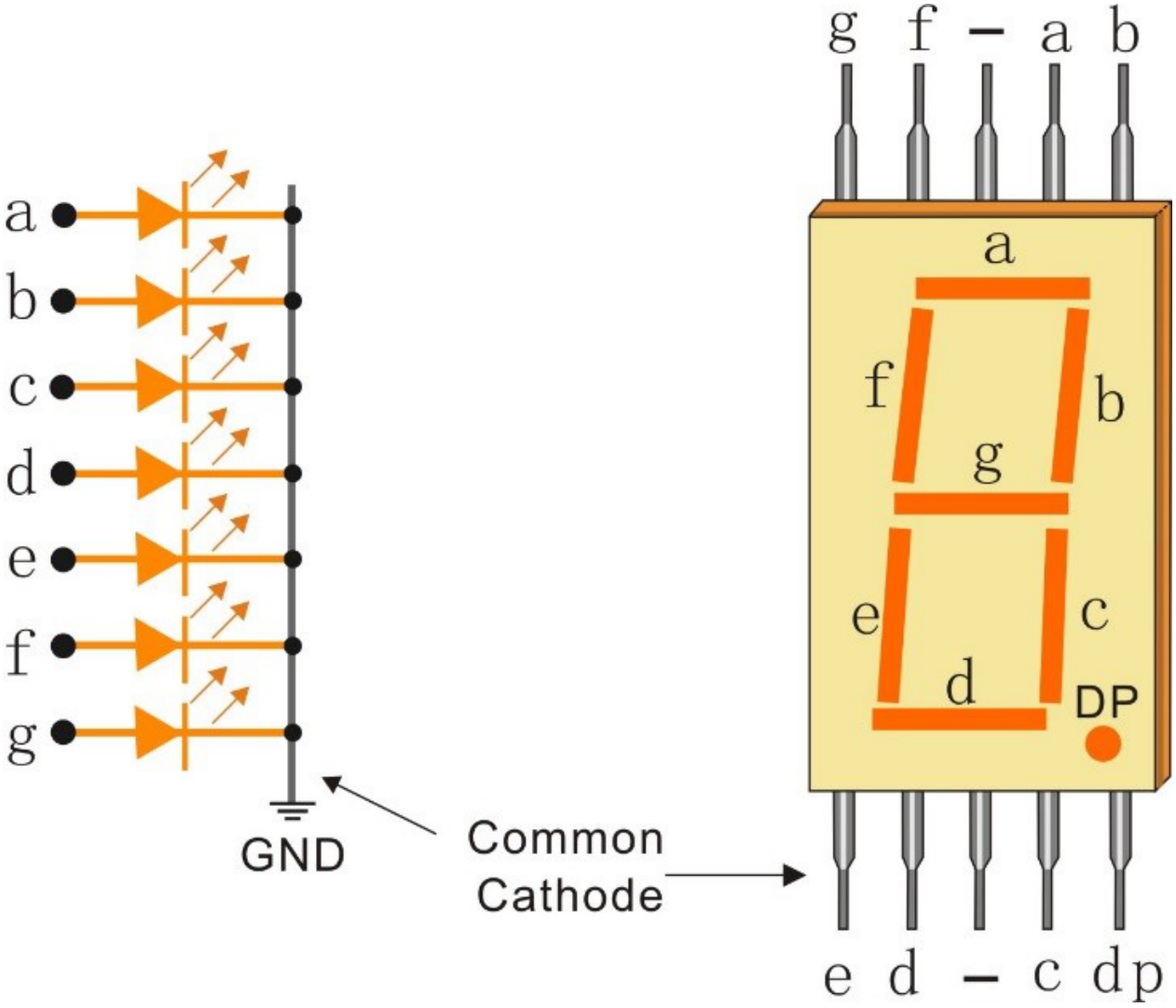
## 1.10 7 セグメント表示




7 セグメント表示は、8 の形をした 7 つの LED を搭載した部品です。各 LED はセグメントと呼ばれ、エネルギーを供給すると、表示する数字の一部を形成します。

ピンの接続には 2 つのタイプがあります：共通カソード (CC) と共通アノード (CA)。CC 表示は、7 つの LED のカソードがすべて接続されているのに対し、CA 表示は 7 つのセグメントのアノードがすべて接続されています。

このキットでは、共通カソードの 7 セグメント表示を使用しています。以下はその電子記号です。




表示内の LED のそれぞれは、位置を持つセグメントとして与えられ、接続ピンの 1 つが長方形のプラスチックパッケージから外に引き出されています。これらの LED ピンは、「a」から「g」までのラベルが付けられ、各個々の LED を表しています。他の LED のピンは一緒になっており、共通のピンを形成しています。したがって、LED のセグメントの適切なピンに順方向のバイアスをかけることで、一部のセグメントが明るくなり、他のセグメントが暗くなり、対応する文字が表示上に表示されます。

**表示コード**

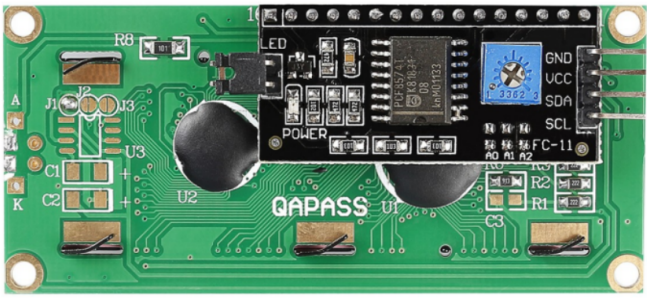
7 セグメント表示（共通カソード）が数字をどのように表示するかを理解するために、以下の表を描きました。Numbers は 7 セグメント表示に表示される数字 0-F を示し、(DP) GFEDCBA は対応する LED が 0 または 1 に設定されていることを示します。たとえば、00111111 は DP と G が 0 に設定されている一方、他のものは 1 に設定されていることを意味します。したがって、7 セグメント表示には数字 0 が表示され、HEX Code は 16 進数に対応します。

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

**例**

- *5.15 EEPROM* (基本プロジェクト)
- *7.* **電流制限ゲート** (IoT プロジェクト)

## 1.11 I2C LCD1602


- **GND**: グラウンド
- **VCC**: 電源供給、5V。
- **SDA**: シリアルデータライン。プルアップ抵抗を通して VCC に接続します。
- **SCL**: シリアルクロックライン。プルアップ抵抗を通して VCC に接続します。

LCD や他のディスプレイは人とマシンのインタラクションを豊かにしていますが、共通の弱点があります。それは、コントローラに接続すると、多くの IO ポートを占有し、コントローラの他の機能を制限します。

この問題を解決するために、I2C モジュール付きの LCD1602 が開発されました。この I2C モジュールは、内蔵の PCF8574 I2C チップを使って、I2C シリアルデータを LCD ディスプレイのための並列データに変換します。

- PCF8574 データシート

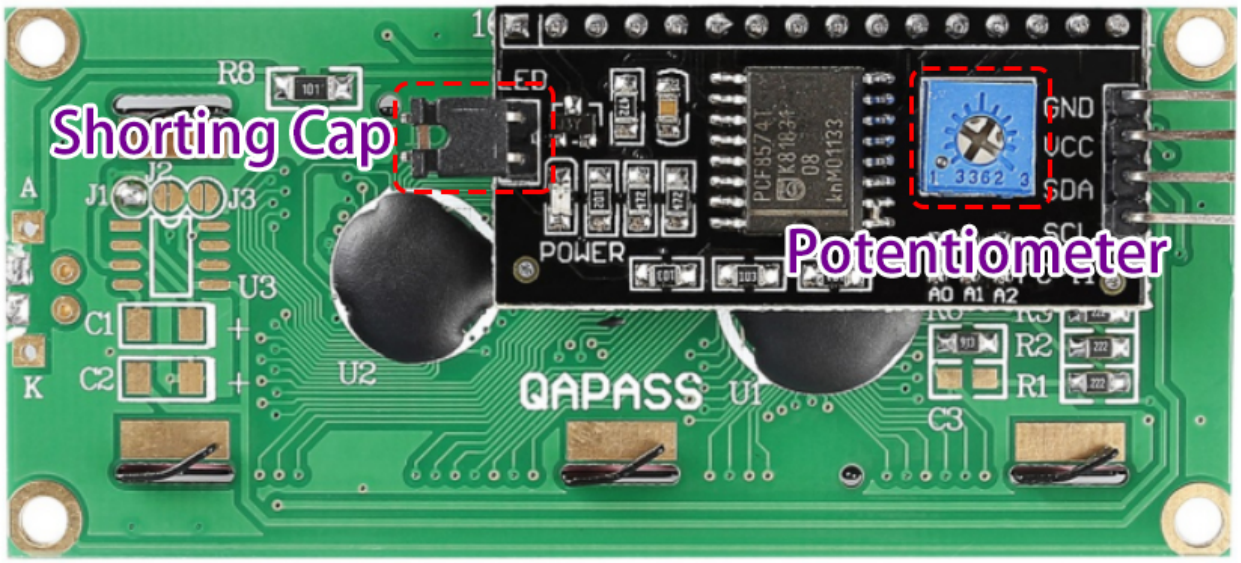
**I2C アドレス**

基本的なデフォルトアドレスは 0x27 で、稀に 0x3F の場合もあります。

デフォルトアドレスの 0x27 を例にとると、デバイスのアドレスは A0/A1/A2 パッドを短絡することで変更できます。デフォルト状態では、A0/A1/A2 は 1 で、パッドが短絡されると、A0/A1/A2 は 0 になります。


**バックライト/コントラスト**

ジャンパーキャップでバックライトを有効にできます。ジャンパーキャップを外すと、バックライトがオフになります。裏側の青いポテンショメータは、コントラスト（最も明るい白と最も暗い黒の間の明るさの比率）を調整するためのものです。



- **短絡キャップ**: このキャップでバックライトを有効にできます。キャップを外すと、バックライトがオフになります。
- **ポテンショメータ**: 表示テキストの明瞭度を調整するためのものです。時計回りで増加し、反時計回りで減少します。

**例**

- *5.11.1* **液晶ディスプレイ** (基本プロジェクト)
- *5.12* **シリアルリード** (基本プロジェクト)

**Sound**

## 1.12 ブザー

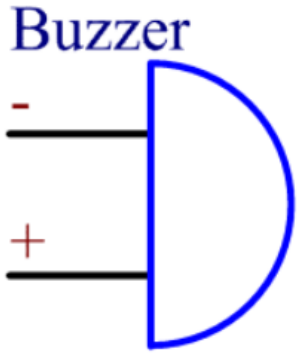
ブザーは DC 電源で駆動される統合型の電子ブザーとして、コンピュータ、プリンタ、コピー機、アラーム、電子おもちゃ、自動車電子機器、電話、タイマーなどの電子製品や音声デバイスに広く使用されています。

ブザーは、アクティブなものとパッシブなものに分類できます（下の写真を参照）。ブザーを上向きにして、緑の回路基板が付いているブザーはパッシブブザーで、黒いテープで囲まれているものはアクティブブザーです。

アクティブブザーとパッシブブザーの違い：

アクティブブザーには振動源が内蔵されているため、電力を供給すると音がします。しかし、パッシブブザーにはそのような源がないため、DC 信号を使用してもビープ音はしない。代わりに、2K から 5K の範囲の正方形の波形を使用して駆動する必要があります。アクティブブザーは、複数の内蔵振動回路のため、パッシブブザーよりも高価なことがよくあります。

以下はブザーの電気記号です。プラスとマイナスの極を持つ 2 つのピンがあります。表面に + があるものが陽極で、他方が陰極です。




ブザーのピンをチェックすると、長い方が陽極で、短い方が陰極です。接続する際に間違えないようにしてください。そうしないと、ブザーは音を出しません。

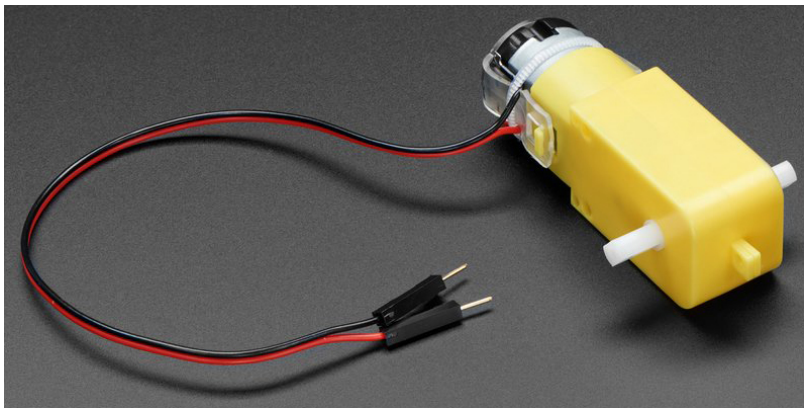
ブザー - Wikipedia

**例**

- *1.2* **ビープ** (基本プロジェクト)
- *5.7 Tone()* **または** *noTone()* (基本プロジェクト)
- *4.* **クラウド音楽プレイヤー** (IoT プロジェクト)

**Driver**

## 1.13 TT モーター



これはギア比 1:48 の TT DC ギアボックスモーターで、ブレッドボードに適合する 0.1"オスコネクタ付きの 2 x 200mm ワイヤーが付属しています。ブレッドボードや端子ブロックに直接挿入するのに最適です。

このモーターは 3〜6VDC で駆動できますが、もちろん、電圧が高いほど少し速くなります。

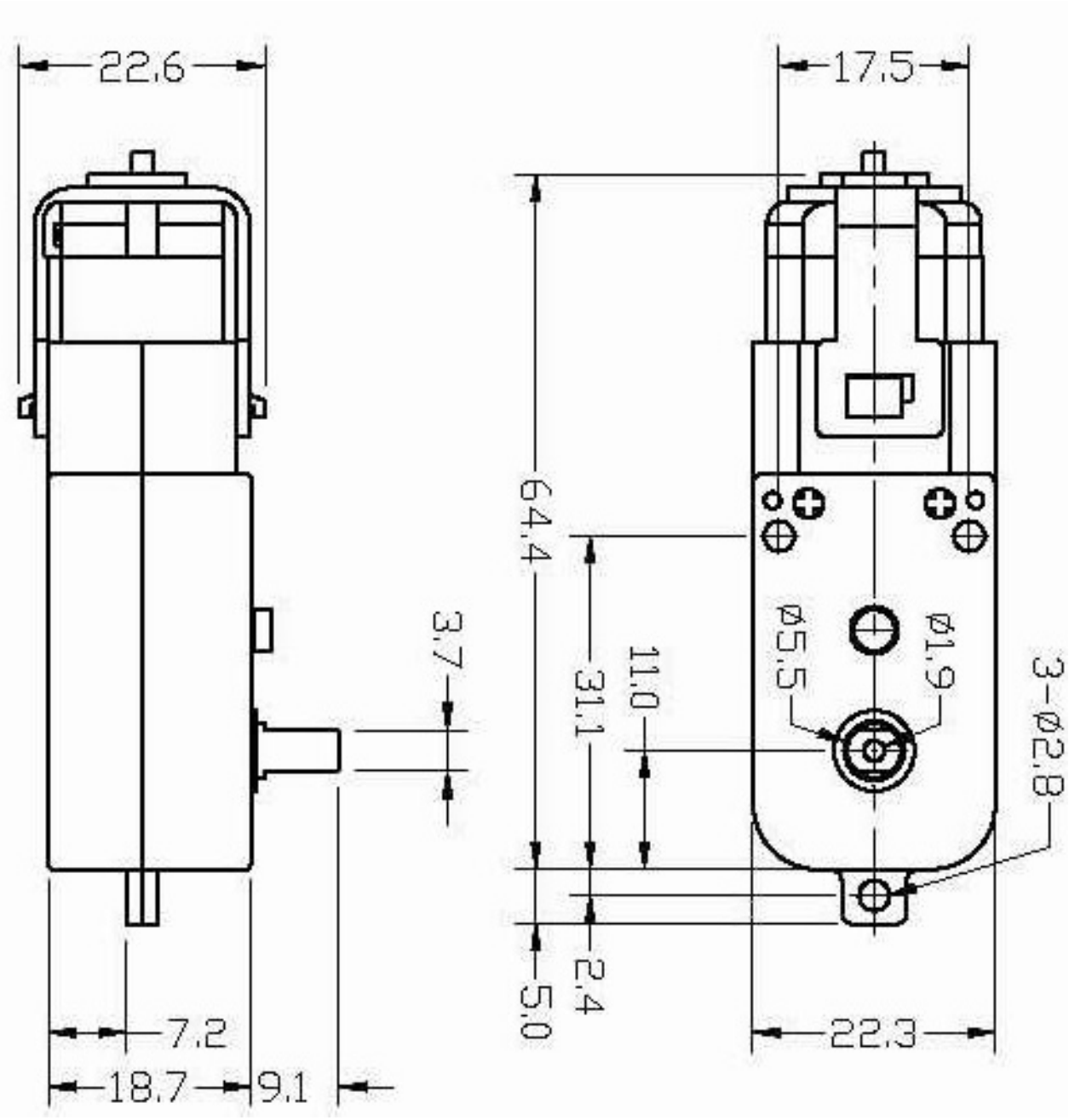
これは非常に基本的なモーターで、組み込みのエンコーダ、速度制御、位置フィードバックはありません。電圧が入力されると、モーターが回転します。モーターごとにバリエーションがあるため、正確な動きが必要な場合は、

別のフィードバックシステムが必要です。

### 技術詳細

- 定格電圧: 3~6V
- 連続無負荷電流: 150mA +/- 10%
- 最小動作速度 (3V): 90+/- 10% RPM
- 最小動作速度 (6V): 200+/- 10% RPM
- ストールトルク (3V): 0.4kg.cm
- ストールトルク (6V): 0.8kg.cm
- ギア比: 1:48
- 本体の寸法: 70 x 22 x 18mm
- ワイヤーの長さ: 200mm & 28 AWG
- 重さ: 30.6g

### 寸法図



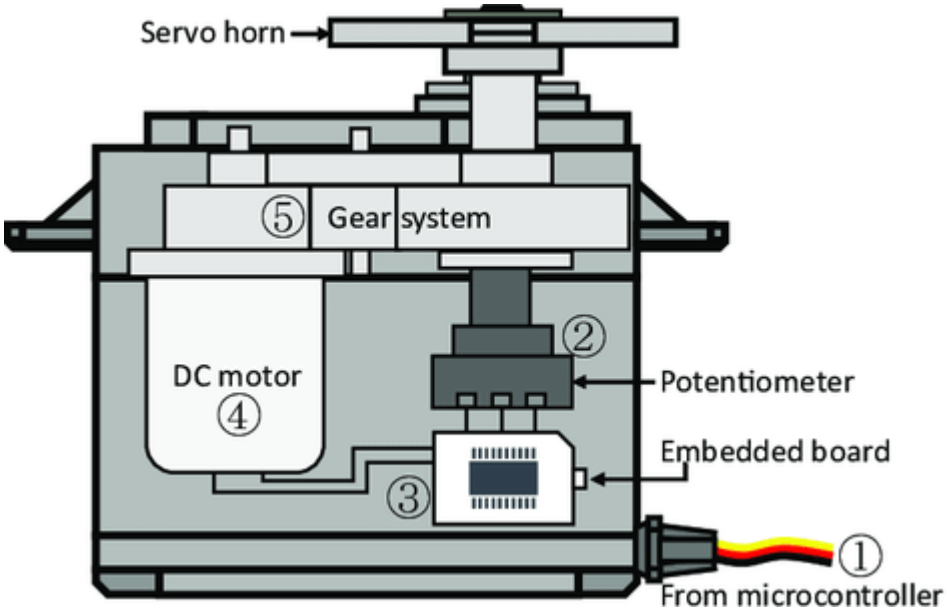

**例**

- *1.3* **車輪を回す** (基本プロジェクト)
- *1.* **移動** (車のプロジェクト)
- *3.* **スピードアップ** (車のプロジェクト)
- *8. IoT* **カー** (IoT プロジェクト)
- *3.1* **車をテストする** (Scratch プロジェクト)

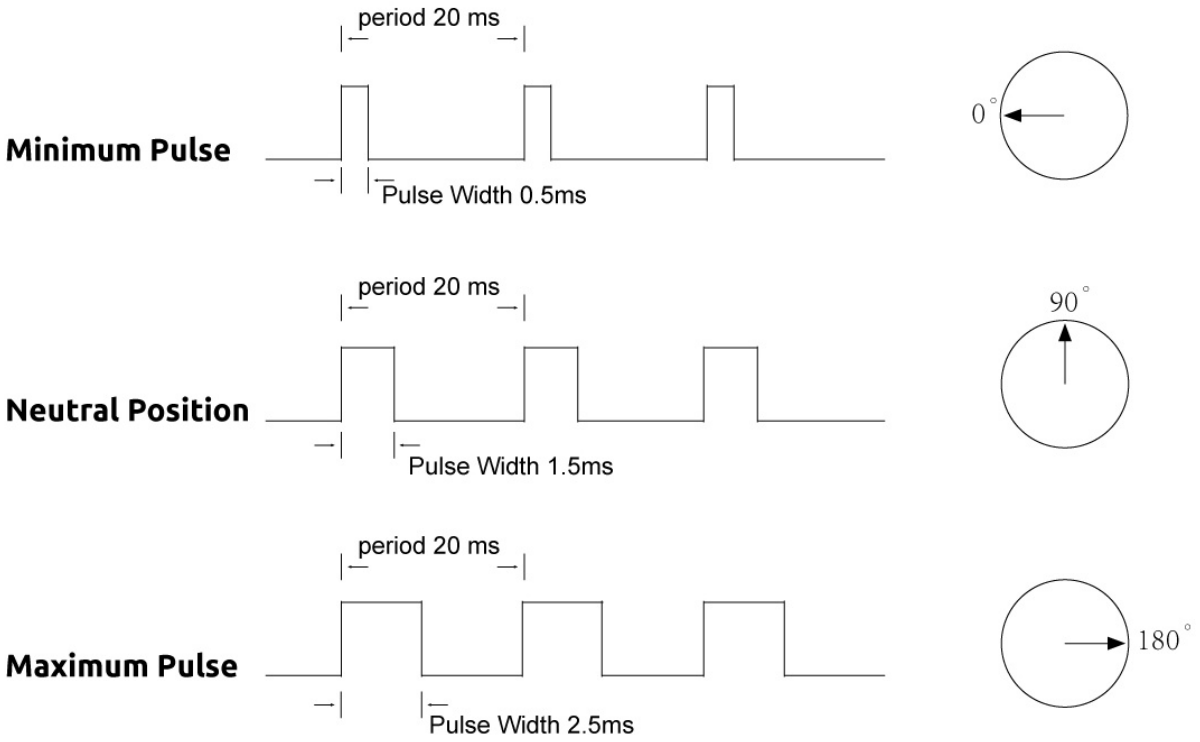
## 1.14 サーボ

サーボは通常、ケース、シャフト、ギアシステム、ポテンショメータ、DC モータ、組み込みボードなどの部品で構成されています。

動作原理は次のようになります：マイクロコントローラは PWM 信号をサーボに送信し、サーボ内の組み込みボードが信号ピンを介して信号を受信し、内部のモータを制御して回転します。その結果、モータはギアシステムを駆動し、減速後にシャフトを動かします。サーボのシャフトとポテンショメータは接続されています。シャフトが回転すると、ポテンショメータは電圧信号を組み込みボードに出力します。次に、ボードは現在の位置に基づいて回転の方向と速度を決定し、定義された正確な位置で正確に停止し、その位置を保持します。



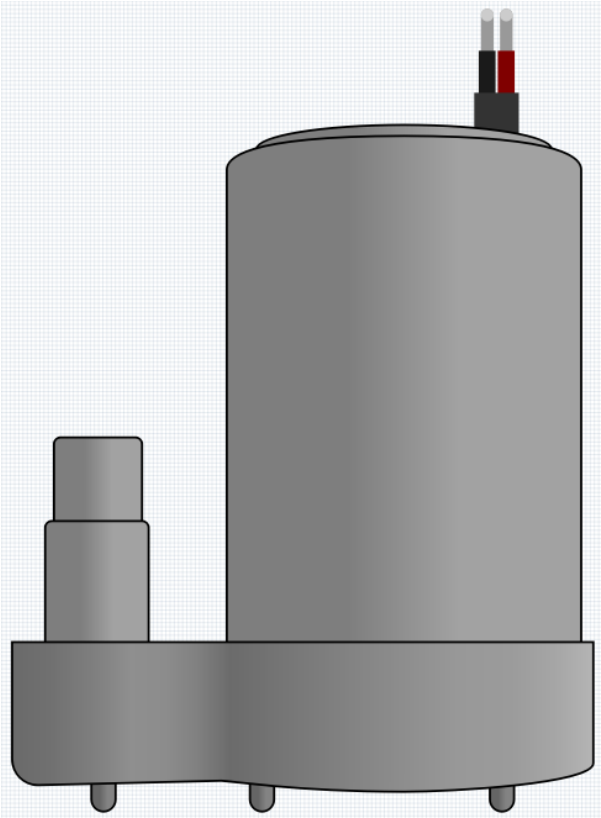

角度は、制御線に適用されるパルスの持続時間によって決まります。これはパルス幅変調と呼ばれます。サーボは、20ms ごとにパルスを受け取ることを期待しています。パルスの長さによってモータの回転角度が決まります。例えば、1.5ms のパルスはモータを 90 度の位置（中立位置）に回転させます。1.5ms より短いパルスがサーボに送信されると、サーボはある角度で回転し、出力シャフトを中立点から反時計回りの数度で保持します。パルスが 1.5ms よりも広い場合、逆のことが起こります。サーボによって指令を送るためのパルスの最小幅と最大幅はそれぞれのサーボによって異なります。一般に、最小のパルスは約 0.5ms、最大のパルスは 2.5ms となります。




**例**

- *5.5* **内蔵ライブラリの使用** (基本プロジェクト)
- *7.* **電流制限ゲート** (IoT プロジェクト)
- *2.10* **振り子** (Scratch プロジェクト)

## 1.15 遠心ポンプ



遠心ポンプは回転運動エネルギーを水力エネルギーに変換して液体を輸送します。この回転エネルギーは電動モーターから供給されます。液体は、回転する軸の近くやその周りを通ってポンプのインペラに入り、インペラによって加速され、放射状に外向きにディフューザやボルート室に流れ、そこから流れ出します。

遠心ポンプの一般的な用途には、水、下水、農業、石油、石油化学のポンプが含まれます。

- 遠心ポンプ - Wikipedia

**特徴**

- **電圧範囲**: DC 3 ~ 4.5V
- **動作電流**: 120 ~ 180mA
- **出力**: 0.36 ~ 0.91W
- **最大揚水高**: 0.35 ~ 0.55M
- **最大流量**: 80 ~ 100 L/H
- **連続動作時間**: 100 時間
- **防水等級**: IP68

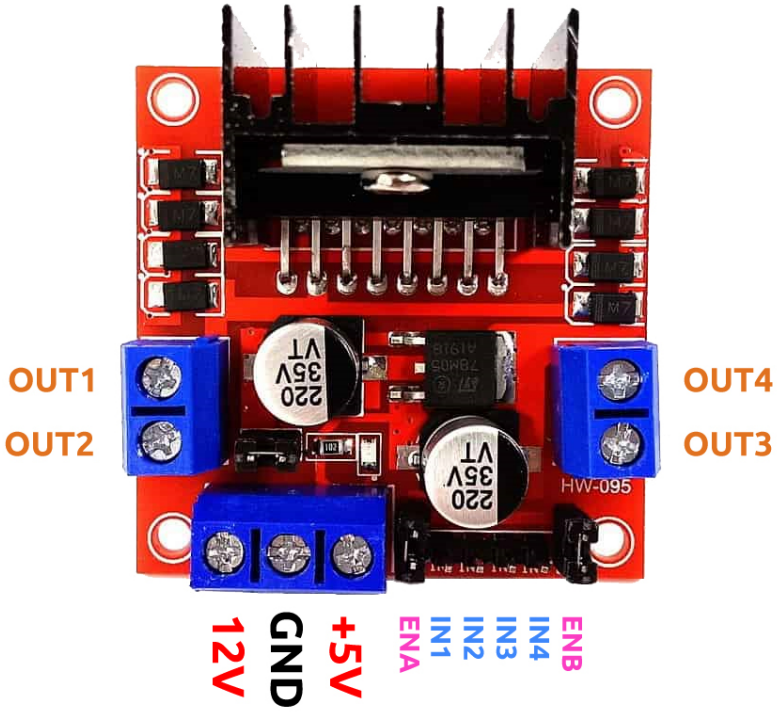
- **駆動方式**: DC、磁気駆動
- **材料**: エンジニアリングプラスチック
- **出口外径**: 7.8 mm
- **出口内径**: 6.5 mm
- このポンプは水中ポンプであり、そのように使用する必要があります。水中にして使用しない場合、過度に加熱してオーバーヒートのリスクがあります。

**例**

- *1.4* **ポンピング** (基本プロジェクト)
- *6.* **植物モニター** (IoT プロジェクト)

## 1.16 L298N モジュール

この L298N モータードライバーモジュールは、DC モーターおよびステッピングモーターを駆動するための高出力モータードライバーモジュールです。このモジュールは、L298 モータードライバ IC と 78M05 5V レギュレータから構成されています。L298N モジュールは、最大 4 つの DC モーター、または方向および速度制御を備えた 2 つの DC モーターを制御できます。




- **IN1 & IN2**: モーター A の入力ピン。モーター A の回転方向を制御するために使用されます
- **IN3 & IN4**: モーター B の入力ピン。モーター B の回転方向を制御するために使用されます

- **ENA**: モーター A の PWM 信号を有効にします。ここではジャンパキャップを使って 5V に接続されています。
- **ENB**: モーター B の PWM 信号を有効にします。ここではジャンパキャップを使って 5V に接続されています。
- **OUT1 & OUT2**: モーター A の出力ピン
- **OUT3 & OUT4**: モーター B の出力ピン
- **12V**: DC 電源からの 12V 入力
- **5V**: L298N IC 内部のスイッチングロジック回路の電源供給
- **GND**: グラウンドピン

**特徴**

- ドライバモデル: L298N 2A
- ドライバチップ: ダブル H ブリッジ L298N
- モーター供給電圧（最大）: 46V
- モーター供給電流（最大）: 2A
- ロジック電圧: 5V
- ドライバ電圧: 5-35V
- ドライバ電流:2A
- 論理電流:0-36mA
- 最大出力（W）: 25W
- 各モーターの電流センス
- より良い性能のためのヒートシンク
- 電源 LED インジケーター

**動作原理**

ドライバモジュールは 2 つのモーターを駆動することができます。ENA および ENB の有効端子は高レベルで効果的です。

ENA および IN1、IN2 間の動作関係は次のとおりです：

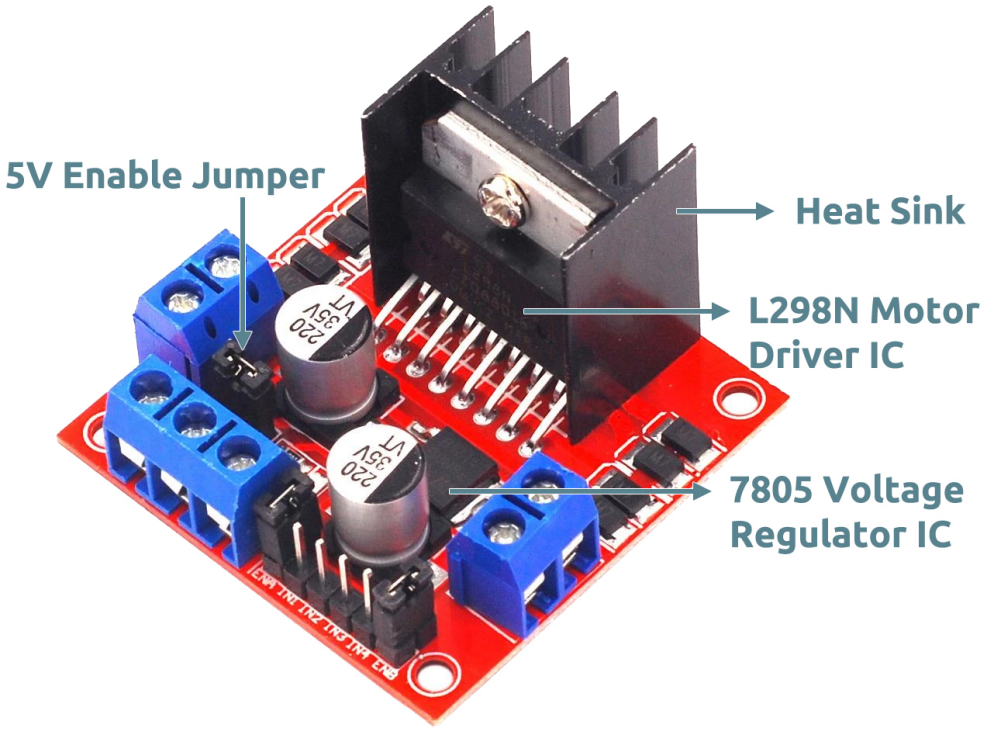
| ENA | IN1 | IN2 | **モーター A の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

ENB および IN3、IN4 間の動作関係は次のとおりです。

| ENB | IN3 | IN4 | **モーター B の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

**5V 有効キャップについて**

L298N モータードライバーモジュールは、L298 モータードライバ IC、78M05 電圧レギュレータ、抵抗、コンデンサ、電源 LED、5V ジャンパーを統合した回路から構成されています。




78M05 電圧レギュレータはジャンパが設置されている場合のみ有効になります。電源が 12V 以下の場合、内部回

路は電圧レギュレータで供給され、5V ピンはマイクロコントローラの電源として出力ピンとして使用できます。

電源が 12V を超える場合、ジャンパは設置しないでください。別の 5V を 5V 端子を通して供給して、内部回路の電源を供給する必要があります。

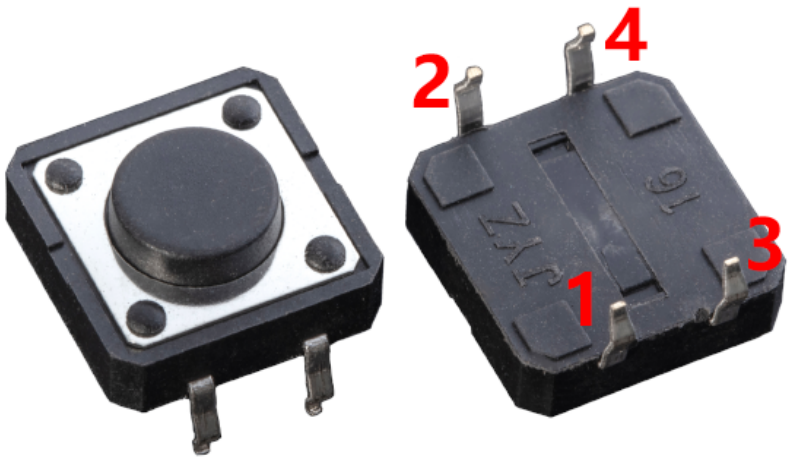
- L298N データシート

**例**

- *1.3* **車輪を回す** (基本プロジェクト)
- *1.* **移動** (カープロジェクト)
- *3.* **スピードアップ** (カープロジェクト)
- *8. IoT* **カー** (IoT プロジェクト)
- *3.1* **車をテストする** (Scratch プロジェクト)

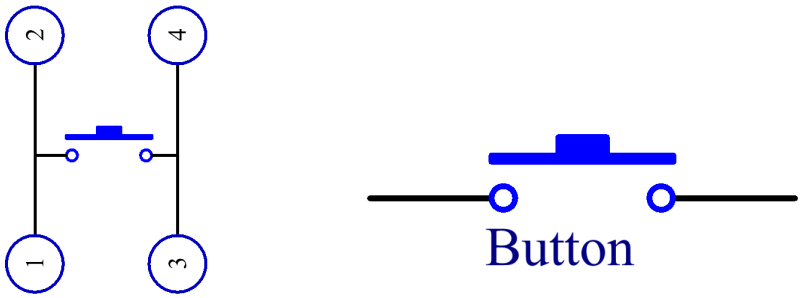
**Controller**

## 1.17 ボタン

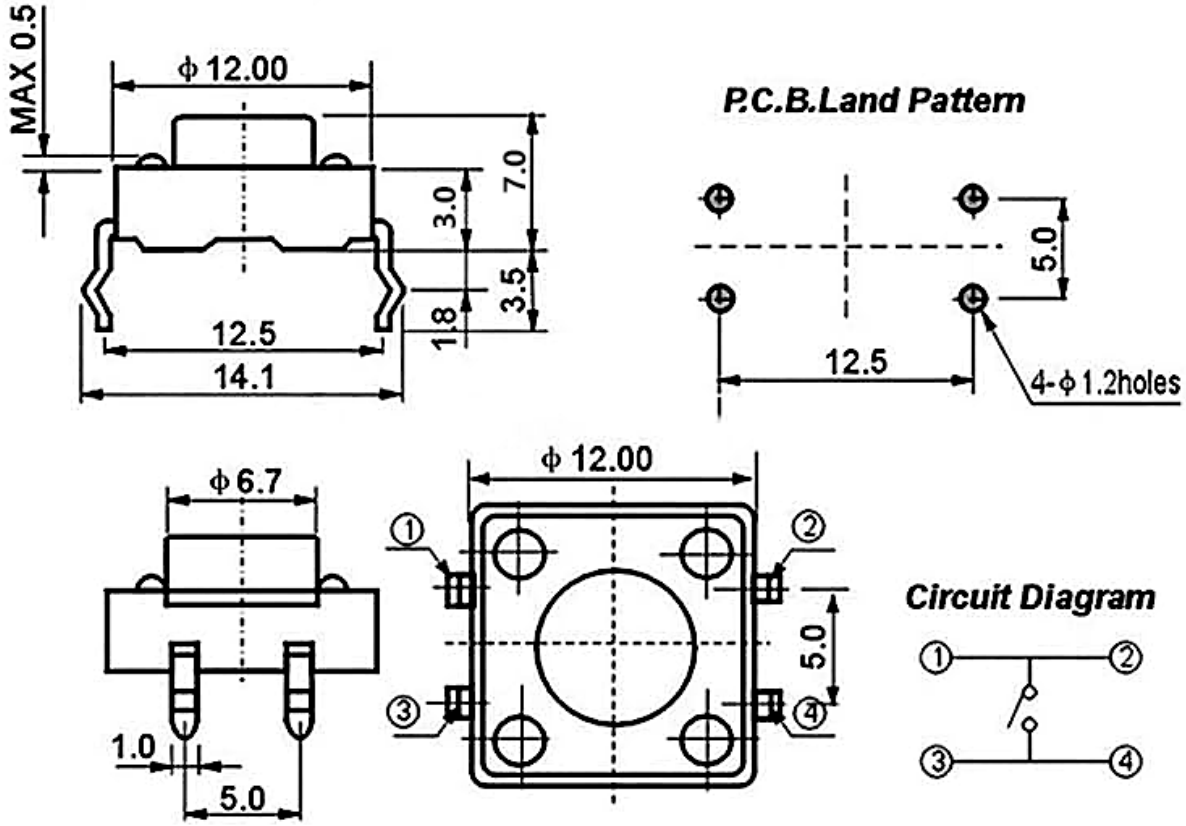



ボタンは、電子デバイスを制御するためによく使用される部品です。通常、回路を接続または切断するスイッチとして使用されます。ボタンはさまざまなサイズや形状で提供されますが、ここで使用されるものは、次の写真に示すような 6mm のミニボタンです。ピン 1 はピン 2 に、ピン 3 はピン 4 に接続されています。そのため、ピン 1 またはピン 2 のいずれかをピン 3 またはピン 4 に接続するだけです。

以下はボタンの内部構造です。右下のシンボルは、回路内でボタンを表すために通常使用されます。




ピン 1 がピン 2 に、ピン 3 がピン 4 に接続されているため、ボタンが押されると、4 つのピンが接続され、回路が閉じます。




**例**

- *3.1* **ボタンの値を読む** (基本プロジェクト)
- *2.6* **ドアベル** (Scratch プロジェクト)
- *2.16* **ゲーム - りんごを食べる** (Scratch プロジェクト)
- *2.19* **ゲーム - 釣り** (Scratch プロジェクト)

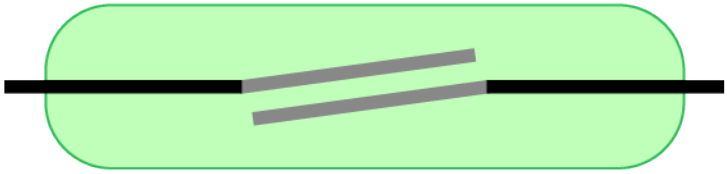
## 1.18 リードスイッチ

リードスイッチは、磁場の影響で動作する電気スイッチです。これは 1936 年に Bell Telephone Laboratories の Walter B. Ellwood によって発明され、1940 年 6 月 27 日にアメリカで特許番号 2264746 の下で特許を取得しました。

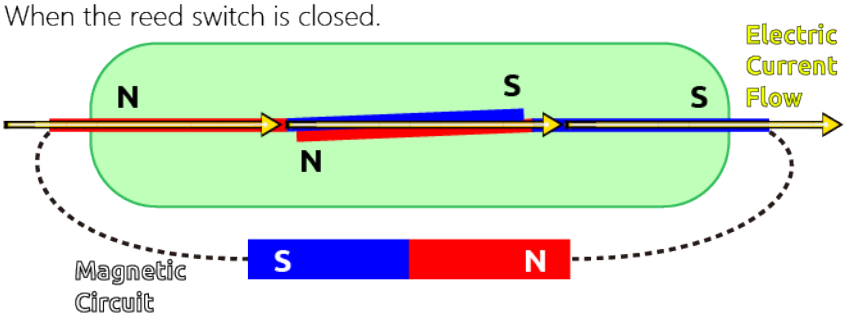
リードスイッチの動作原理は非常にシンプルです。鉄とニッケル（2 つの金属）でできている 2 つのリードが、先端で重なるようにガラス管の中に密封されています。これら 2 つのリードは、数マイクロンのわずかな隙間を保って重なっています。ガラス管は高純度の不活性ガス（例えば窒素）で満たされており、高電圧性能を向上させるために、真空を持つリードスイッチも作られています。

リードは磁束の伝導体として機能します。2 つのリードは動作していないときには接触していません。永久磁石や電磁コイルによって生成される磁場を通過すると、適用された磁場により 2 つのリードの端点近くに異なる極性が生じ、磁力がリード自体のバネ力を超えると、2 つのリードは回路を導通するために引き寄せられます。磁場が弱くなるか消失すると、リードは自身の弾性により放され、接触面は回路を開くために分離します。

When the reed switch is open.



When the reed switch is closed.

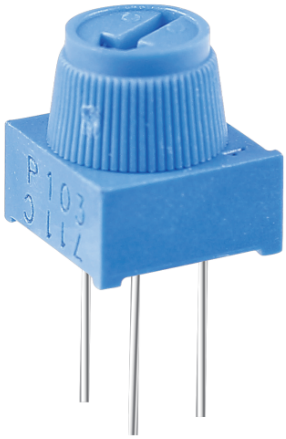



- リードスイッチ - Wikipedia

**例**

- *3.2* **磁気を感じる** (基本プロジェクト)
- *7.* **電流制限ゲート** (IoT プロジェクト)

## 1.19 ポテンショメータ

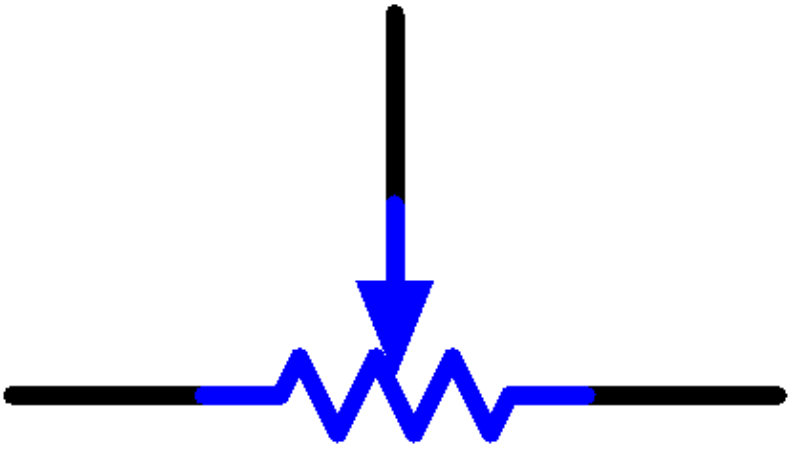



ポテンショメータは、3 つの端子を持つ抵抗コンポーネントでもあり、抵抗値は一定の変化に応じて調整することができます。

ポテンショメータにはさまざまな形状、サイズ、値がありますが、共通点として以下の特徴があります：

- 3 つの端子（または接続点）がある。
- 中央の端子と外側のどちらかの端子との間の抵抗を変化させるためのノブ、ネジ、またはスライダがある。
- ノブ、ネジ、またはスライダが移動すると、中央の端子と外側の端子の一方との間の抵抗が 0 Ωからポテンショの最大抵抗まで変化する。

こちらがポテンショメータの回路記号です。



ポテンショメータが回路内で果たす機能は以下の通りです：

1. 電圧分割器としての役割

    ポテンショメータは連続的に調整可能な抵抗器です。ポテンショメータの軸またはスライドハンドルを

調整すると、可動接点が抵抗体上でスライドします。この時点で、ポテンショメータにかかる電圧と、移動アームが回転した角度や移動した距離に応じて、電圧が出力されることができます。

2. リオスタットとしての役割

   ポテンショメータがリオスタットとして使用される場合、中央のピンと他の 2 つのピンのうちの 1 つを回路に接続します。このようにすると、移動接点の移動範囲内で、スムーズかつ連続的に変化する抵抗値を取得することができます。

3. 電流コントローラとしての役割

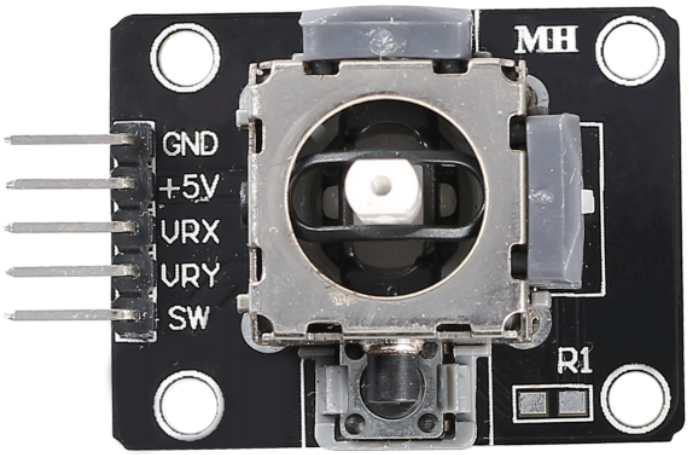
   ポテンショメータが電流コントローラとして機能する場合、スライド接触端子は出力端子の一方として接続する必要があります。

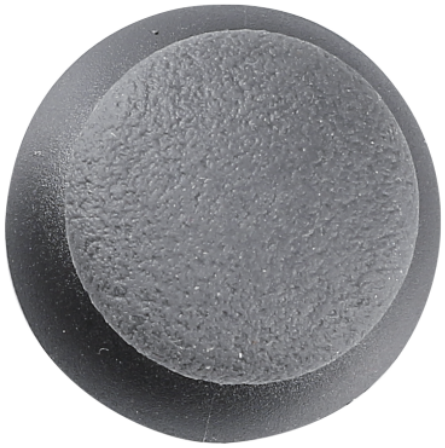
ポテンショメータに関する詳細は、以下を参照してください： ポテンショメータ - Wikipedia

**例**

- *4.1* **ノブを回す** (基本プロジェクト)
- *2.5* **移動するマウス** (Scratch プロジェクト)
- *2.18* **ゲーム - ブレイクアウトクローン** (Scratch プロジェクト)

## 1.20 ジョイスティックモジュール



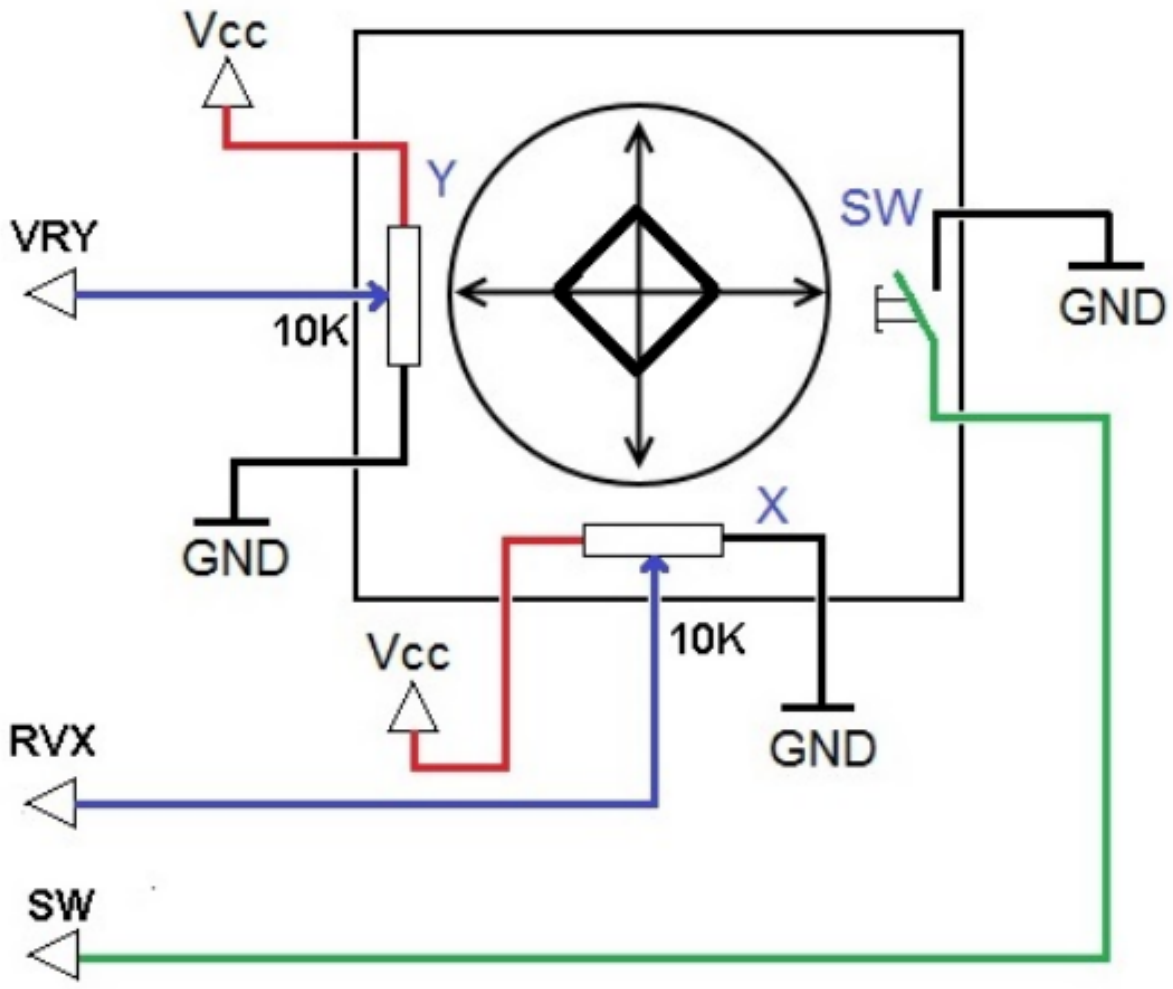



ジョイスティックの基本的な考え方は、スティックの動きをコンピュータが処理できる電子情報に変換することです。

コンピュータに完全な動きの範囲を伝えるために、ジョイスティックはスティックの位置を 2 つの軸、X 軸（左から右）と Y 軸（上から下）で測定する必要があります。基本的な幾何学のように、X-Y 座標はスティックの正確な位置を特定します。

スティックの位置を判断するために、ジョイスティック制御システムは各軸の位置を監視します。伝統的なアナログジョイスティックデザインはこれを二つのポテンショメータ、または可変抵抗器で行います。

ジョイスティックには、押下すると動作するデジタル入力も備えています。

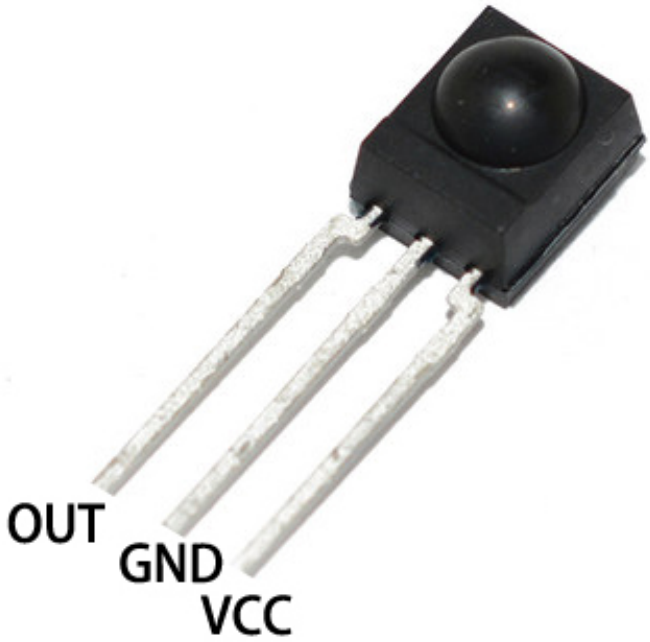



**例**

- *4.3* **ジョイスティックのトグル** (基本プロジェクト)
- *2.15 GAME -* **スタークロスド** (Scratch プロジェクト)
- *2.22 GAME -* **ドラゴン討伐** (Scratch プロジェクト)

## 1.21 IR レシーバー

**IR レシーバー**




- OUT: シグナル出力
- GND：GND
- VCC: 電源供給、3.3v~5V

赤外線受信器は、赤外線信号を受け取り、TTL レベルと互換性のある信号を独立して出力できる部品です。サイズは通常のプラスチックパッケージのトランジスタと同等で、あらゆる種類の赤外線リモコンや赤外線伝送に適しています。

赤外線、または IR、通信は人気があり、低コストで使いやすい無線通信技術です。赤外線は可視光線よりも僅かに長い波長を持っているため、人間の目には見えない - 無線通信に理想的です。赤外線通信の一般的な変調方式は 38KHz の変調です。

- 採用した HS0038B IR 受信センサー、高感度
- リモートコントロールに使用可能
- 電源供給: 5V
- インターフェース: デジタル
- 変調周波数: 38Khz
- ピン定義: (1) 出力 (2) Vcc (3) GND

- サイズ: 23.5mm x 21.5mm

**リモートコントロール**


これは、21 の機能ボタンと最大 8 メートルの伝送距離を持つミニサイズの薄型赤外線ワイヤレスリモコンで、子供部屋のさまざまなデバイスを操作するのに適しています。

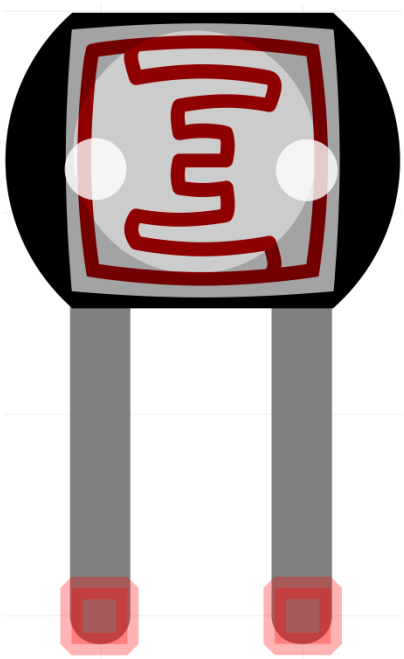
- サイズ: 85x39x6mm
- リモートコントロール範囲: 8-10m
- バッテリー: 3V ボタン型リチウムマンガン電池
- 赤外線キャリア周波数: 38KHz
- 表面粘着材: 0.125mm PET
- 有効寿命: 20,000 回以上

**例**

- *5.11.2 IR* **レシーバー** (基本プロジェクト)
- *9.* **リモートコントロール** (車プロジェクト)
- *10.* **ワンタッチスタート** (車プロジェクト)

**Sensor**

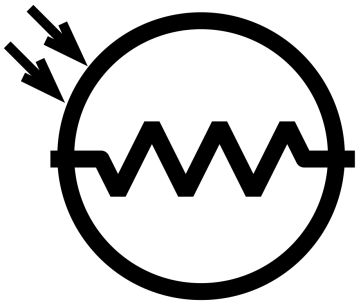
## 1.22 フォトレジスタ



フォトレジスタまたはフォトセルは、光に応じて変動する抵抗器です。フォトレジスタの抵抗は、入射光の強度が増加するにつれて減少します。言い換えれば、光電導性を示します。

フォトレジスタは、光感応検出回路や、抵抗半導体として動作する光活性化および暗活性化スイッチ回路に適用できます。暗闇では、フォトレジスタの抵抗は数メガオーム（M Ω）に達することができますが、明るいところでは、数百オームまで低くなることができます。

以下はフォトレジスタの電子記号です。



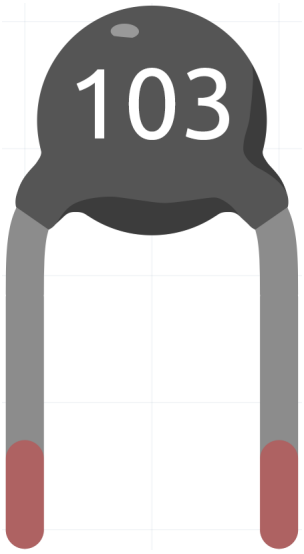
- フォトレジスタ - Wikipedia

**例**

- *4.2* **光を感じる** (基本プロジェクト)
- *5.* **住宅環境監視** (IoT プロジェクト)

- *6.* **植物モニター** (IoT プロジェクト)

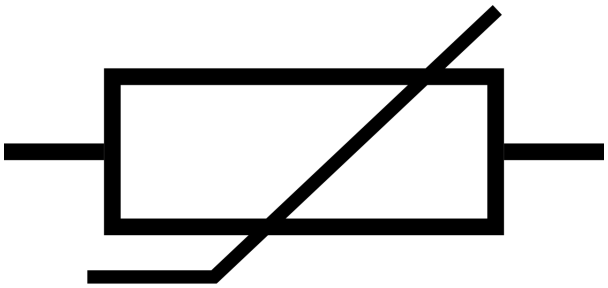
## 1.23 サーミスタ




サーミスタは、通常の抵抗器よりも温度に強く依存する抵抗値を持つタイプの抵抗器です。この言葉は、thermal（熱の）と resistor（抵抗器）の組み合わせです。サーミスタは、突入電流リミッタ、温度センサ（通常は負の温度係数または NTC 型）、自己リセット型の過電流保護器、および自己調整型加熱素子（通常は正の温度係数または PTC 型）として広く使用されています。

- サーミスタ - Wikipedia

こちらはサーミスタの電子記号です。



サーミスタには、基本的に 2 つの逆のタイプがあります：

- NTC サーミスタでは、温度が上昇すると抵抗が減少します。これは、熱激振により価電子帯から打ち上げられる導電電子の増加が原因です。NTC は通常、温度センサとして、または回路と直列に接続された突入電流リミッタとして使用されます。
- PTC サーミスタでは、温度が上昇すると抵抗が増加します。これは、不純物や欠陥の熱格子振動が特に増加するためです。PTC サーミスタは、回路と直列に取り付けられ、過電流条件に対する保護として、またはリ

セッタブルヒューズとして使用されることが一般的です。

このキットでは、NTC を使用しています。各サーミスタには標準抵抗があります。こちらは 25 度セルシウスで測定された 10k オームです。

以下は、抵抗と温度の関係です：

RT = RN * expB(1/TK – 1/TN)

- **RT** は、温度が TK のときの NTC サーミスタの抵抗です。
- **RN** は、評価温度 TN での NTC サーミスタの抵抗です。こちらの RN の数値は 10k です。
- **TK** はケルビン温度で、単位は K です。TK の数値は、273.15 + 度数セルシウスです。
- **TN** は評価ケルビン温度で、単位も K です。TN の数値は、273.15+25 です。
- **B(beta)** は、NTC サーミスタの材料定数で、熱感受性指数とも呼ばれ、数値は 3950 です。
- **exp** は指数の略で、基数 e は自然数であり、おおよそ 2.7 と等しいです。

この式 TK=1/(ln(RT/RN)/B+1/TN) を変換すると、ケルビン温度から 273.15 を引いた値が度数セルシウスと等しくなります。

この関係は経験的な式です。温度と抵抗が有効範囲内にある場合のみ正確です。

**例**

- *2.7* **低温警報** (Scratch プロジェクト)
- *6.3* **高温度警報** (基本プロジェクト)
- *4.5* **温度計** (基本プロジェクト)

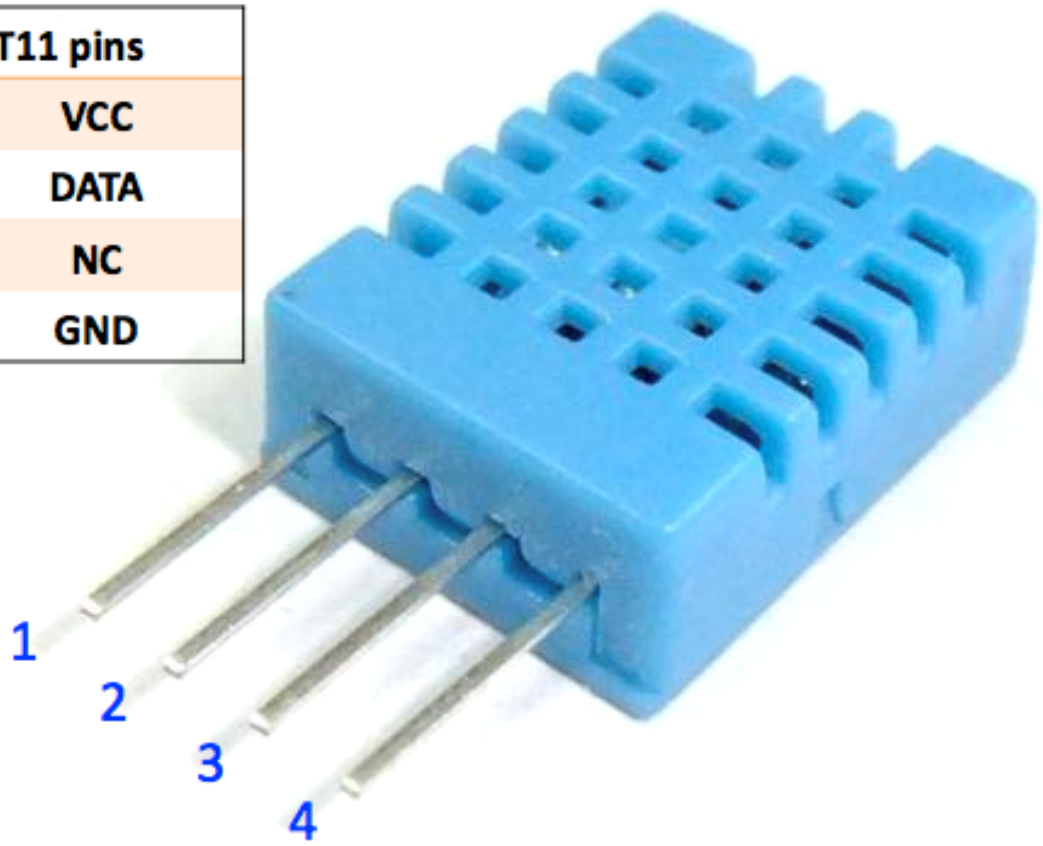
## 1.24 DHT11 湿度温度センサ

デジタル温度および湿度センサの DHT11 は、温度と湿度の校正されたデジタル信号出力を含む複合センサです。専用のデジタルモジュール収集技術と温度および湿度検知技術が適用されており、製品の高い信頼性と長期間の優れた安定性を確保しています。

このセンサは、湿度の抵抗式センサと NTC 温度計測デバイスを含み、高性能の 8 ビットマイクロコントローラに接続されています。

使用できるピンは 3 つだけです：VCC、GND、DATA。通信プロセスは、DATA ラインが DHT11 にスタート信号を送信することから始まります。DHT11 はこれらの信号を受け取り、応答信号を返します。その後、ホストは応答信号を受け取り、40 ビットの湿度データ（8 ビットの湿度整数 + 8 ビットの湿度小数 + 8 ビットの温度整数 + 8 ビットの温度小数 + 8 ビットのチェックサム）の受信を開始します。

| DHT11 pins | |
|---|---|
| 1 | VCC |
| 2 | DATA |
| 3 | NC |
| 4 | GND |




**特長**

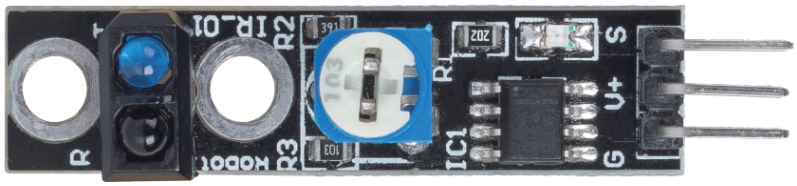
1. 湿度測定範囲: 20 - 90%RH
2. 温度測定範囲: 0 - 60 ℃
3. 温度および湿度を示すデジタル信号の出力
4. 動作電圧:DC 5V; PCB サイズ: 2.0 x 2.0 cm
5. 湿度測定の精度: ±5%RH
6. 温度測定の精度: ±2 ℃

- DHT11 データシート

**例**

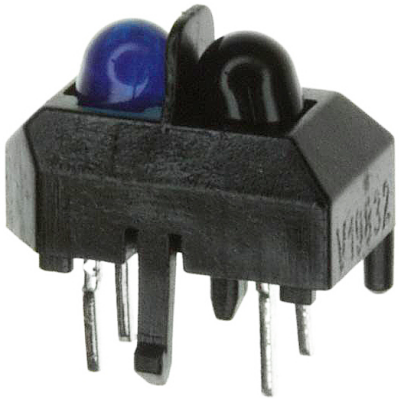
- *5.11.3* **温度・湿度** (基本プロジェクト)
- *5.* **住宅環境監視** (IoT プロジェクト)
- *6.* **植物モニター** (IoT プロジェクト)

## 1.25 ライン追跡モジュール




- S: 通常は低レベルで、黒いラインが検出された場合は高レベル。
- V+：電源、3.3v~5V
- G: グラウンド

これは、名前が示すように、白い背景上の黒いラインまたは黒い背景上の白いラインを追跡する 1 チャンネルのライン追跡モジュールです。



モジュールには TCRT5000 赤外線センサーが使用されており、赤外線 LED（青）と光感応三重組（黒）から構成されています。

- 青い赤外線 LED は、電源が入っていると、人の目には見えない赤外線を放射します。
- 黒いフォトトランジスタは、赤外線を受信するために使用され、内部の抵抗は、受信した赤外線の量に応じて抵抗が変化します。赤外線の量が多いほど、その抵抗は減少し、その逆も同様です。

モジュールには LM393 コンパレータがあり、フォトトランジスタの電圧を（ポテンショメータによって調整される）設定電圧と比較します。もし設定電圧よりも大きければ、出力は 1 です。そうでない場合、出力は 0 です。

したがって、赤外線放射管が黒い表面に光を当てると、黒は光を吸収するため、フォトセンシングトランジスタが受け取る赤外線は少なくなります。その結果、その抵抗は増加します（電圧も増加）。LM393 コンパレータを経て、高レベルが出力されます。

同様に、白い表面に光を当てると、反射光が増えてフォトセンシングトランジスタの抵抗が減少します（電圧が減少）。その結果、コンパレータは低レベルを出力し、指示 LED が点灯します。

- TCRT5000

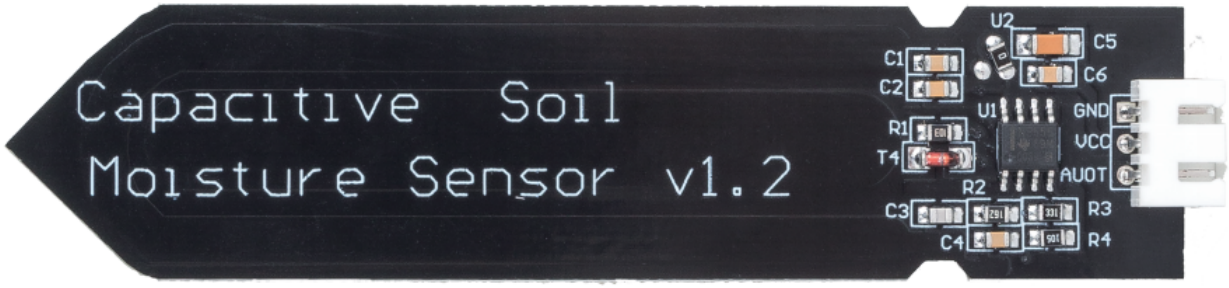
**特徴**

- 赤外線放射センサー TCRT5000 を使用
- 検出距離：1-8mm、焦点距離 2.5mm
- コンパレータの出力信号はクリーンで、波形が良好、駆動能力は 15mA を超える
- 感度調整のためのポテンショメータを使用
- 動作電圧：3.3V-5V
- デジタル出力：0（白）および 1（黒）
- 広電圧 LM393 コンパレータを使用。
- サイズ：42mmx10mm

**例**

- *3.4* **ラインを検出する** (基本プロジェクト)
- *4.* **ラインフォロー** (車のプロジェクト)
- *2.21 GAME -* **心を守れ** (スクラッチプロジェクト)

## 1.26 土壌湿度モジュール

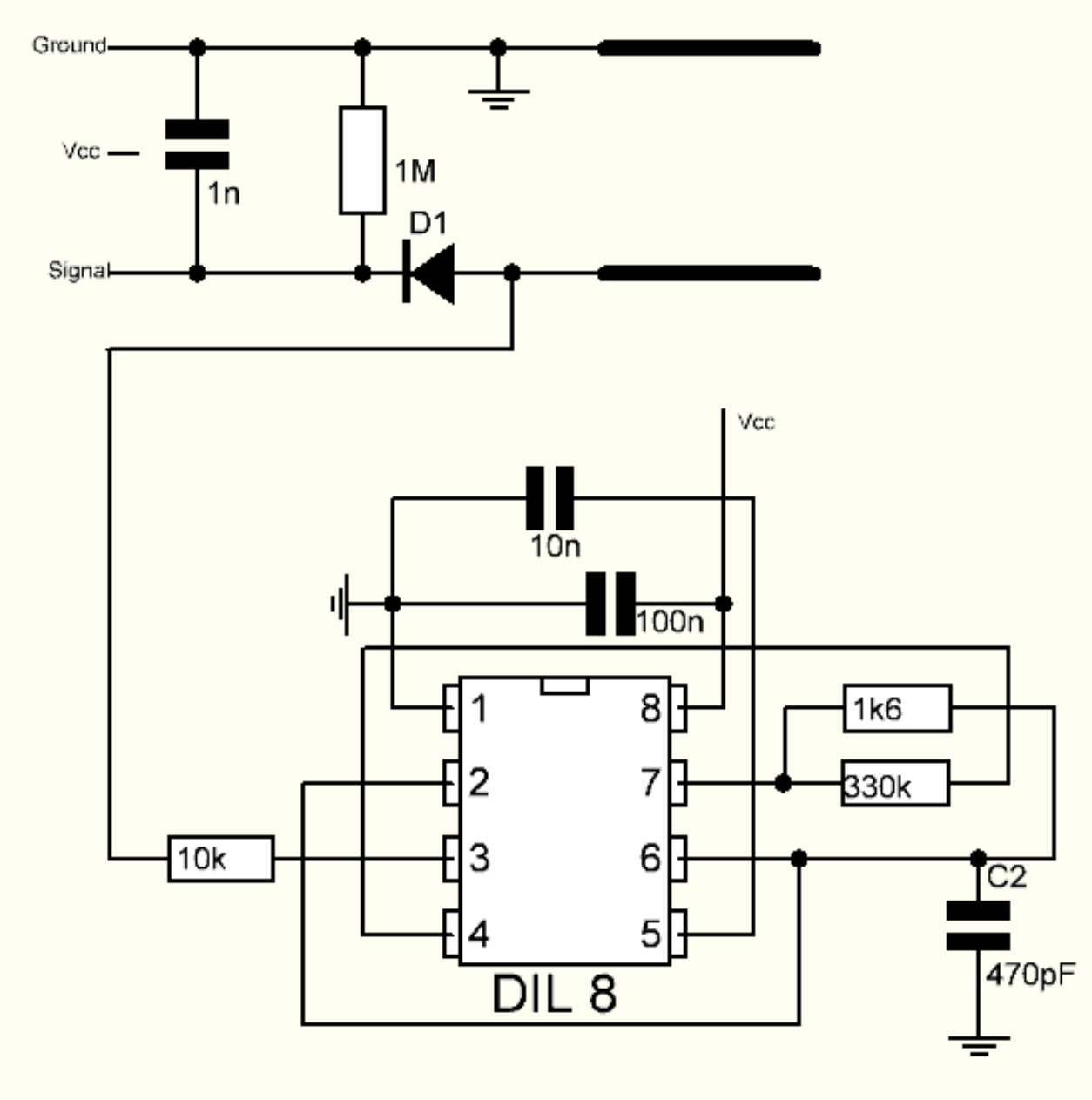



- GND: 接地
- VCC：電源供給、3.3v~5V
- AOUT: 土壌の湿度値を出力。土壌が湿っているほど、その値は小さくなります。

この容量性土壌湿度センサーは市場の多くの抵抗センサーと異なり、容量性誘導の原理を使用して土壌の湿度を検出しています。これにより、抵抗センサーが腐食に非常に敏感である問題を回避し、作動寿命を大幅に延ばしています。

腐食耐性の素材で作られており、優れた耐用年数があります。植物の周りの土壌に挿入し、リアルタイムで土壌の湿度データを監視します。このモジュールには、3.3～5.5 V の電圧範囲で動作できるようにオンボードの電圧調整器が含まれています。3.3 V および 5 V の電源を持つ低電圧のマイクロコントローラに適しています。

容量性土壌湿度センサーのハードウェア回路図は以下の通りです。




固定周波数の発振器があり、555 タイマー IC で構築されています。生成された正方形の波は、コンデンサのようなセンサーに供給されます。ただし、正方形の波信号には、純粋なオーム抵抗（ピン 3 の 10k 抵抗）の抵抗として、ある反応、または論争のために、電圧分配器を形成します。

土壌の湿度が高いほど、センサーの容量が高くなります。その結果、正方形の波は反応が少なく、信号線上の電圧が低下し、マイクロコントローラを介したアナログ入力の値が小さくなります。

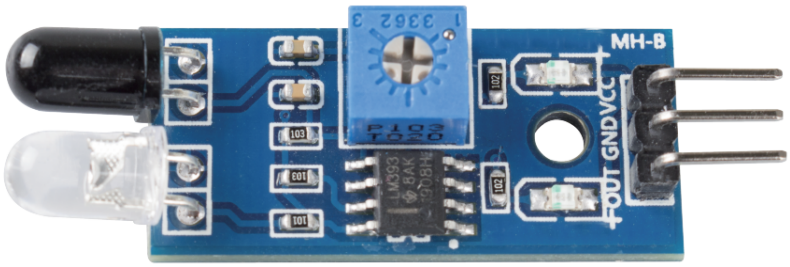
**仕様**

- 動作電圧: 3.3 ~ 5.5 VDC
- 出力電圧: 0 ~ 3.0VDC
- 動作電流: 5mA
- インターフェース: PH2.0-3P
- 寸法: 3.86 x 0.905 インチ (L x W)
- 重量: 15g

**例**

- *4.4* **土壌の湿度を測定する** (基本プロジェクト)
- *6.* **植物モニター** (IoT プロジェクト)

## 1.27 障害物回避モジュール

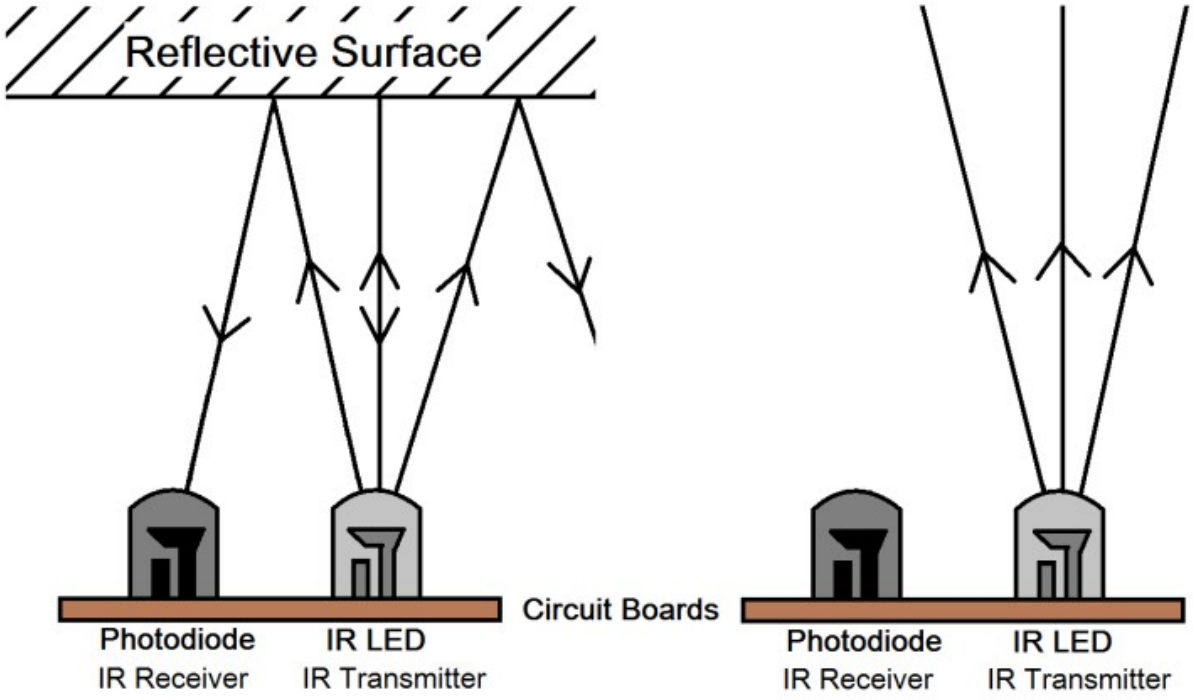



- **VCC**: 電源供給、3.3 ~ 5V DC。
- **GND**: 接地
- **OUT**: シグナルピン、通常は高レベルで、障害物が検出されると低レベルになります。

IR 障害物回避モジュールは、環境光への適応性が強く、赤外線の送受信チューブのペアを持っています。

送信チューブは赤外線周波数を放射し、検出方向に障害物がある場合、赤外線放射は受信チューブによって受信され、コンパレータ回路処理後、インジケータが点灯し、低レベル信号が出力されます。

検出距離はポテンショメータで調整可能であり、有効距離範囲は 2-30cm です。




**例**

- *3.3* **障害物を検出する** (基本プロジェクト)
- *5.* **障害物回避モジュールで遊ぶ** (カープロジェクト)
- *8.* **自動運転車** (カープロジェクト)
- *7.* **電流制限ゲート** (IoT プロジェクト)
- *2.13* **ゲーム - シューティング** (Scratch プロジェクト)
- *2.20* **ゲーム - 白いタイルをタップしないで** (Scratch プロジェクト)

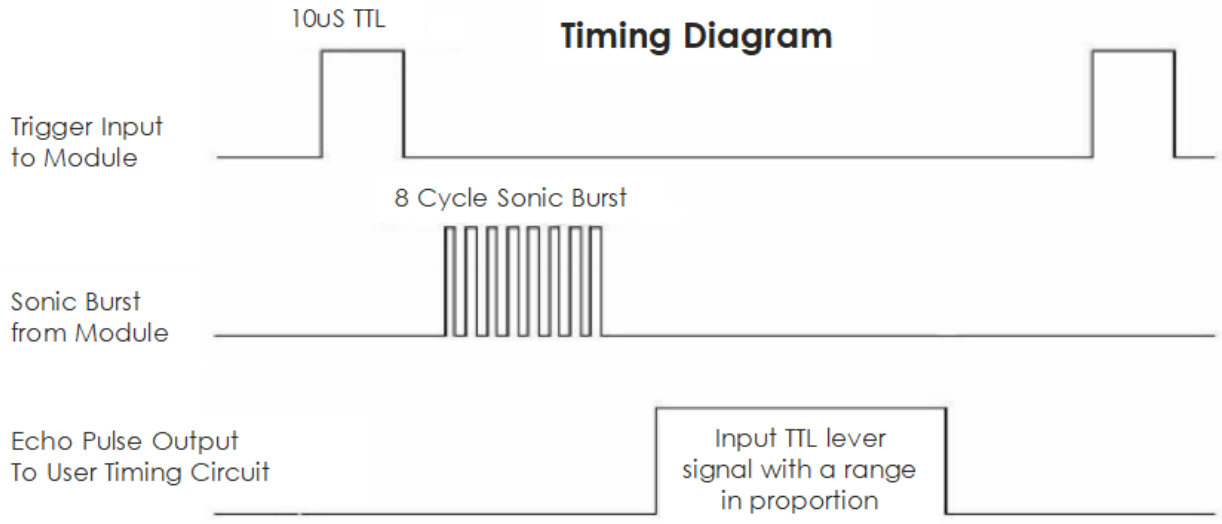
## 1.28 超音波モジュール


超音波測距モジュールは 2cm - 400cm の非接触測定機能を提供し、測定精度は 3mm に達することができます。5m 以内の信号が安定していることを保証でき、5m 後に信号が徐々に弱くなり、7m の位置で消失します。

モジュールには超音波送信機、受信機、および制御回路が含まれています。基本的な原理は以下のとおりです：

1. IO フリップフロップを使用して、少なくとも 10us の高レベル信号を処理します。
2. モジュールは自動的に八つの 40kHz を送信し、パルス信号の返信があるかどうかを検出します。
3. 信号が戻ってきた場合、高レベルを通過させ、高出力 IO の持続時間は、超音波の送信から返送までの時間です。ここで、テスト距離 = (高時間 x 音速 (340 m / s) / 2。

タイミングダイアグラムは以下のとおりです。




測距を開始するには、トリガ入力に短い 10us パルスを供給するだけで、次にモジュールは 40kHz での 8 サイクル

の超音波バーストを送出し、そのエコーを上げます。トリガ信号の送信とエコー信号の受信の間の時間間隔を通じて範囲を計算できます。

式：us / 58 = センチメートル または us / 148 =インチ；または：範囲 = 高レベル時間*速度（340M/S）/ 2；トリガ信号とエコー信号の信号衝突を防ぐために、60ms 以上の測定周期を使用することをおすすめします。

**例**

- *5.8* **ユーザー定義関数** (基本プロジェクト)
- *7.* **手を追いかける車** (カープロジェクト)
- *6.* **超音波モジュールで遊ぶ** (カープロジェクト)
- *2.17* **ゲーム - フラッピーパロット** (Scratch プロジェクト)

# 第 2 章

# Arduino を始めよう

Arduino について何も知らないなら、以下の言葉を覚えておいてください：電子工学、デザイン、プログラミング、そしてメイカー。これらの言葉が自分たちとはかけ離れていると感じるかもしれませんが、実際にはそうではありません。Arduino を通じて、私たちはプログラミングの世界に足を踏み入れ、メイカーとしての夢を実現することができます。このセッションでは次の内容を学びます：

- Arduino とは何か？
- Arduino は何をすることができるのか？
- Arduino のプロジェクトをどのように構築するのか？

## 2.1 Arduino とは？

まず、Arduino について簡単に紹介します。

Arduino は、便利で柔軟、そして使いやすいオープンソースの電子プロトタイピングプラットフォームであり、様々なモデルの Arduino ボードと Arduino IDE というソフトウェアを含んでいます。これは、迅速なプロトタイピングのためのエンジニアだけでなく、アーティスト、デザイナー、趣味人にも適しています。現代のメイカーには欠かせないツールとも言えます。

Arduino は非常に大きなシステムです。ソフトウェア、ハードウェア、そして共通の趣味を持つ、互いに知り合いではない多くの人々からなる非常に大きなオンラインコミュニティがあります。Arduino のコミュニティのメンバーは、自らの知恵を用い、手で制作し、次々と素晴らしい発明を共有しています。そして、あなたもその一部となることができます。

## 2.2 Arduino は何ができる？

Arduino が実際に何をすることができるのか疑問に思うかもしれません。簡単に言えば、Arduino があなたの問題をすべて解決します。

技術的に言えば、Arduino はプログラム可能なロジックコントローラーです。これは、リモートコントロールカー、ロボットアーム、バイオニックロボット、スマートホームなど、多くのエキサイティングでクリエイティブな電子クリエーションを作成するために使用できる開発ボードです。

Arduino ボードはシンプルで強力であり、学生、メイカー、さらにはプロのプログラマーにも適しています。

今日に至るまで、世界中の電子愛好者たちは、Arduino の開発ボードを基に、創造的な電子作品を続けて開発しています。

## 2.3 Arduino プロジェクトの構築方法

以下のステップに従って、ゼロから Arduino を使用する方法を学びましょう！

### 2.3.1 Arduino IDE 2.0 のダウンロードとインストール

Arduino IDE は、Arduino Integrated Development Environment としても知られ、Arduino プロジェクトを完了するために必要なすべてのソフトウェアサポートを提供します。これは Arduino 専用に設計されたプログラミングソフトウェアで、Arduino チームによって提供され、Arduino ボードにプログラムを書き込んだりアップロードしたりすることができます。

Arduino IDE 2.0 はオープンソースのプロジェクトです。堅牢な前身である Arduino IDE 1.x から大きく進化し、新しい UI、改善されたボード＆ライブラリマネージャ、デバッガ、オートコンプリート機能などが強化されています。

このチュートリアルでは、Windows、Mac、Linux のコンピュータに Arduino IDE 2.0 をダウンロードしてインストールする方法を示します。

#### 必要な環境

- Windows - Win 10 以上、64 ビット
- Linux - 64 ビット
- Mac OS X - バージョン 10.14 "Mojave" 以上、64 ビット

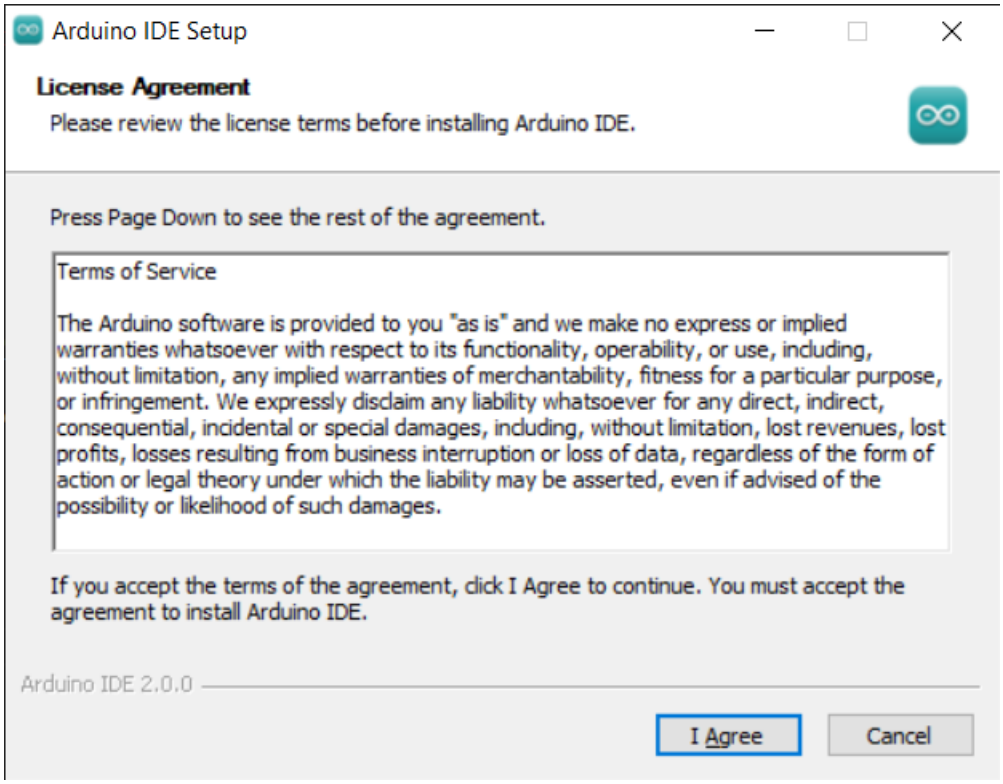
### Arduino IDE 2.0 のダウンロード

1. を訪問してください。
2. OS のバージョンに合わせて IDE をダウンロードします。

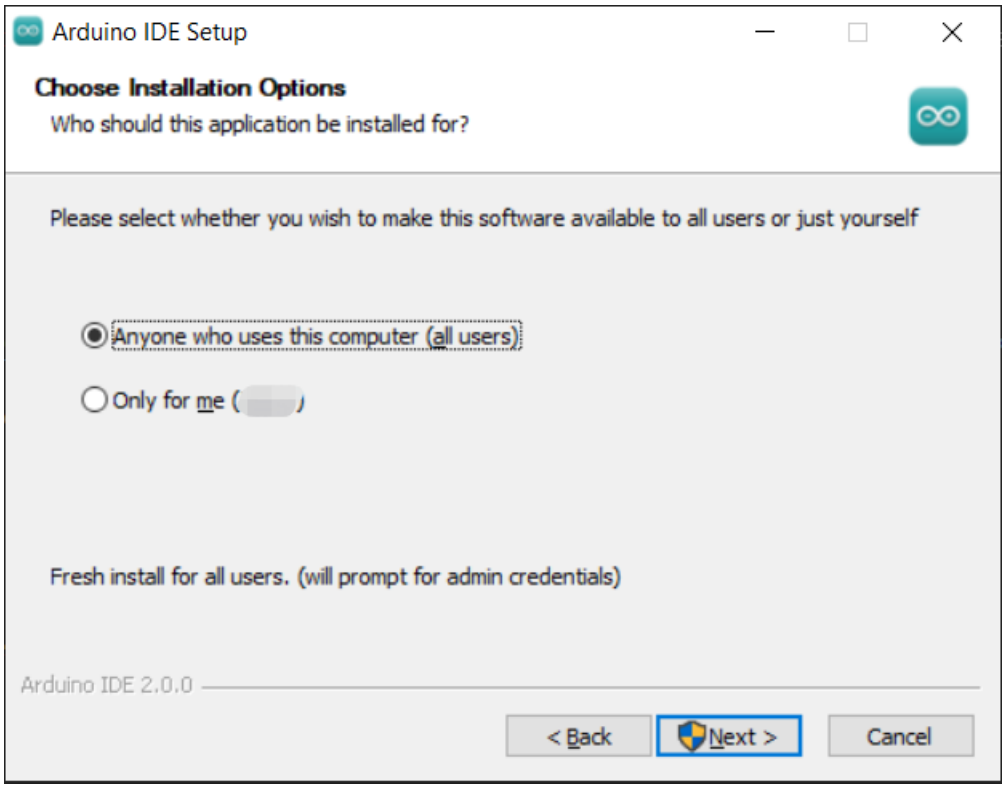

### インストール方法

### Windows

1. ダウンロードした `arduino-ide_xxxx.exe` ファイルをダブルクリックして実行します。
2. ライセンス契約を読み、同意します。

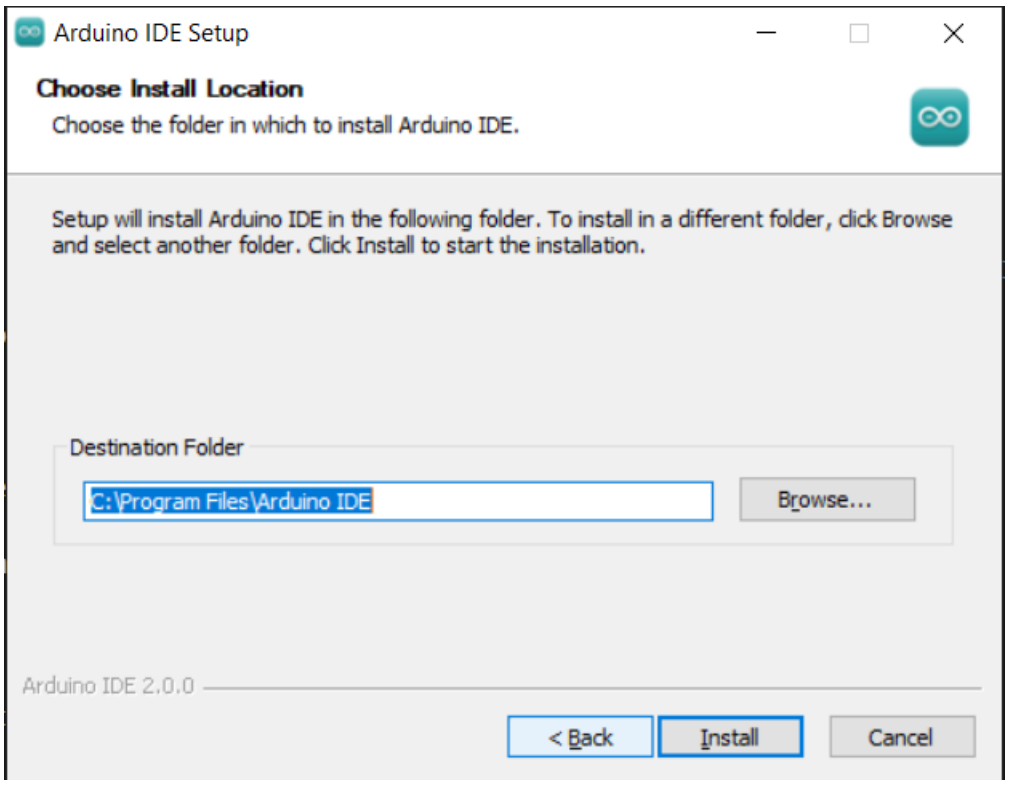



3. インストールオプションを選択します。

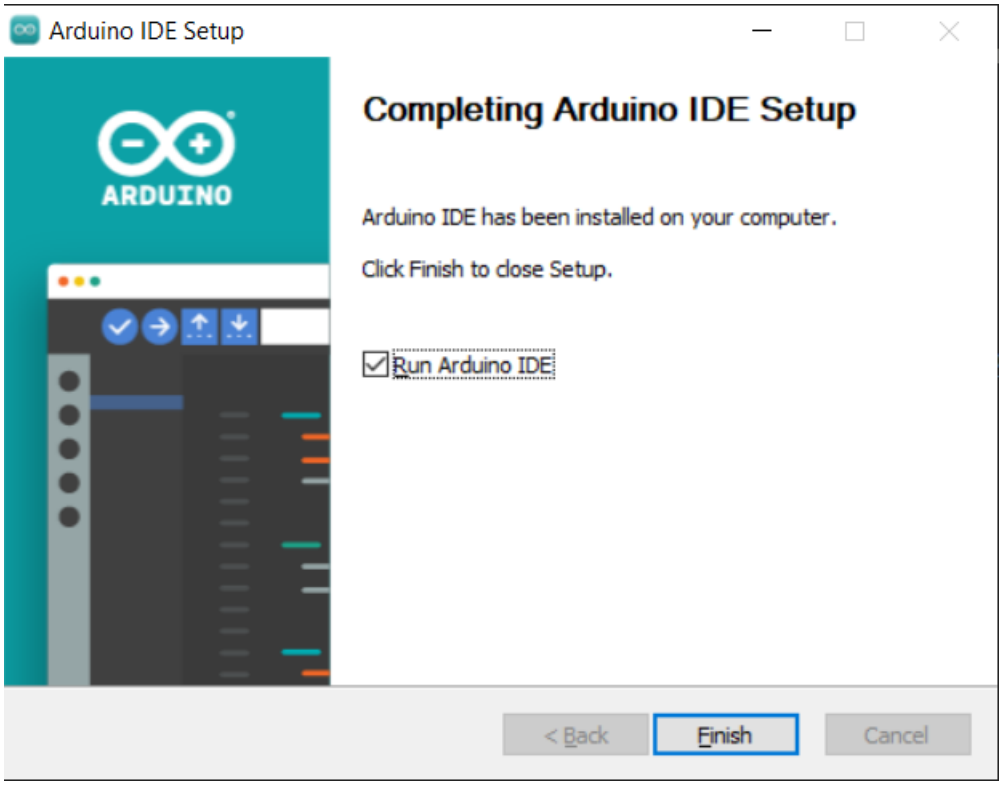



4. インストール先を選択します。システムドライブ以外のドライブにソフトウェアをインストールすることを

おすすめします。

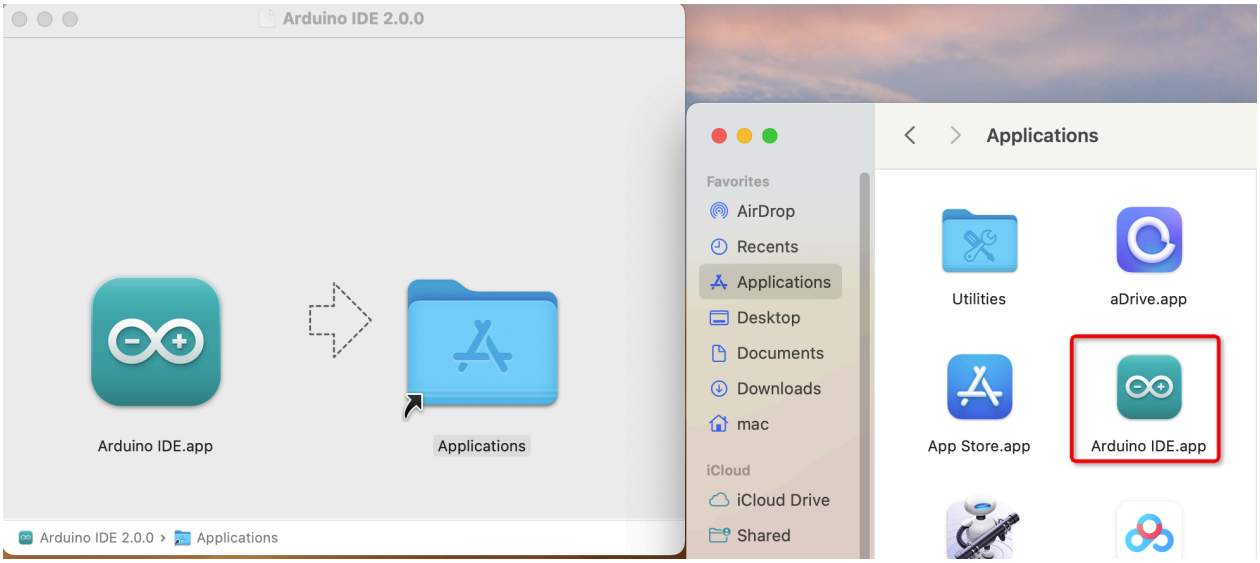



5. それで完了です。



### macOS

ダウンロードした `arduino_ide_xxxx.dmg` ファイルをダブルクリックし、指示に従って **Arduino IDE.app** を **Applications** フォルダにコピーします。数秒後に、Arduino IDE のインストールが成功したことが確認できます。

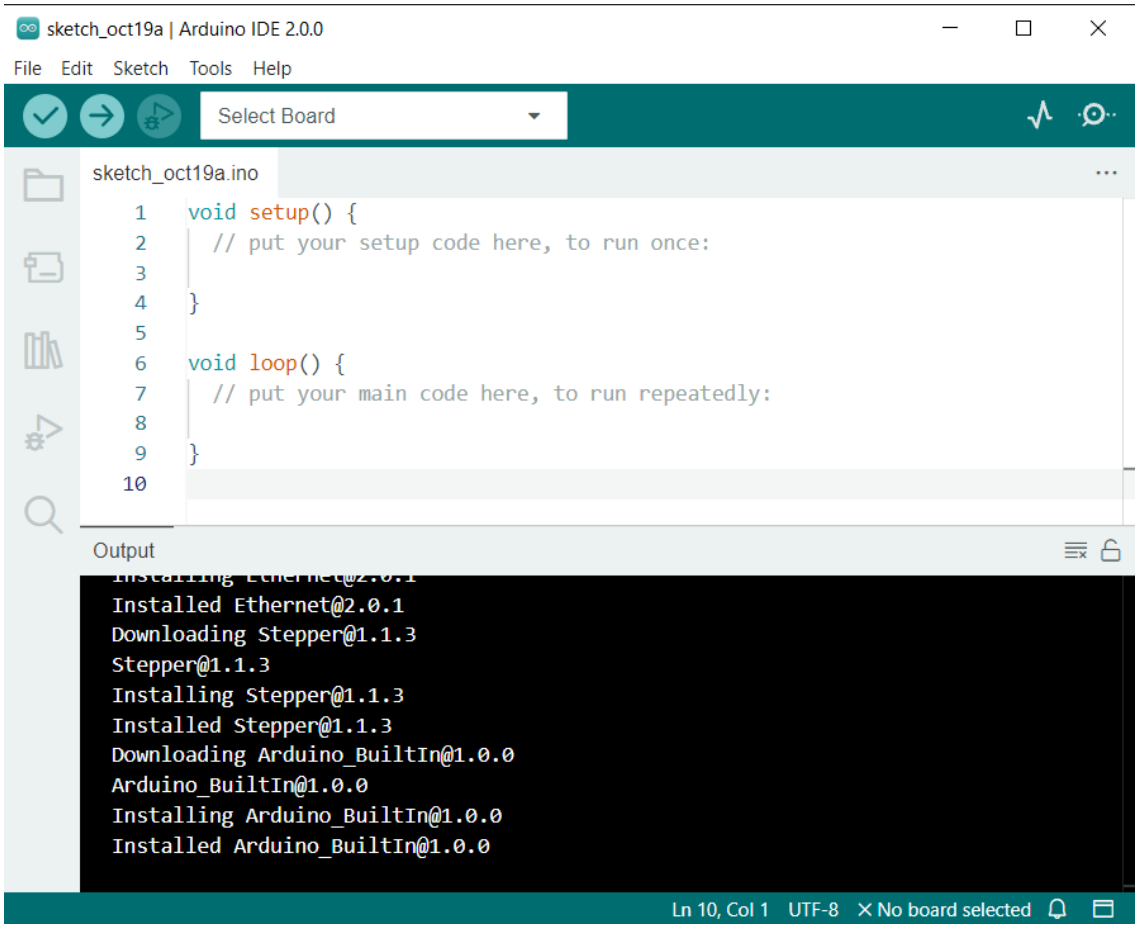



### Linux

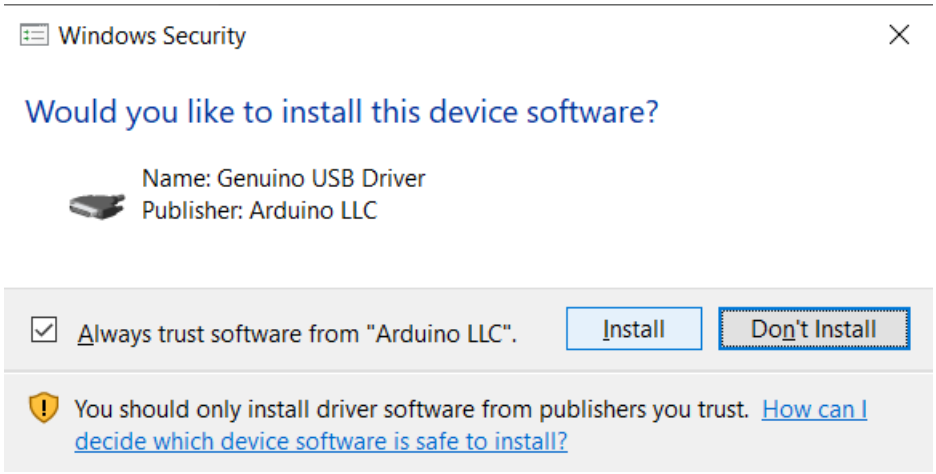
Linux システムで Arduino IDE 2.0 をインストールする方法についてのチュートリアルは、以下の URL を参照してください：https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux

### IDE の起動

1. Arduino IDE 2.0 を初めて開くと、Arduino AVR ボード、組み込みのライブラリ、およびその他の必要なファイルが自動的にインストールされます。




2. また、ファイアウォールやセキュリティセンターがいくつかのデバイスドライバをインストールするかどうかを尋ねるポップアップが表示されることがあります。すべてインストールしてください。

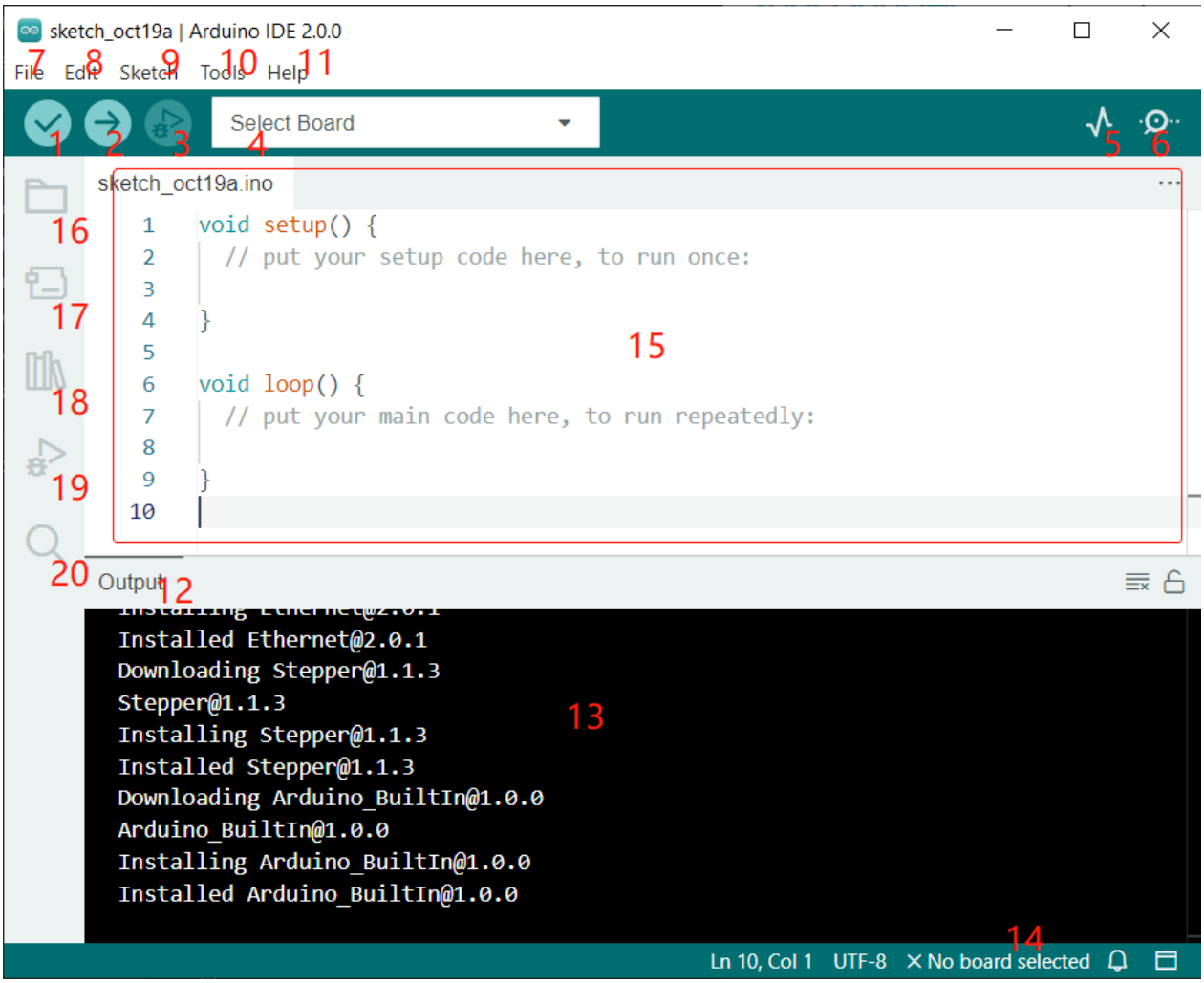



3. これで、Arduino IDE の準備が完了です！

**注釈:** ネットワークの問題やその他の理由で一部のインストールが正常に完了しなかった場合、Arduino IDE を再度開くと、残りのインストールが完了します。すべてのインストールが完了した後、確認また

はアップロードをクリックしない限り、出力ウィンドウは自動的に開きません。

### 2.3.2 Arduino IDE の紹介

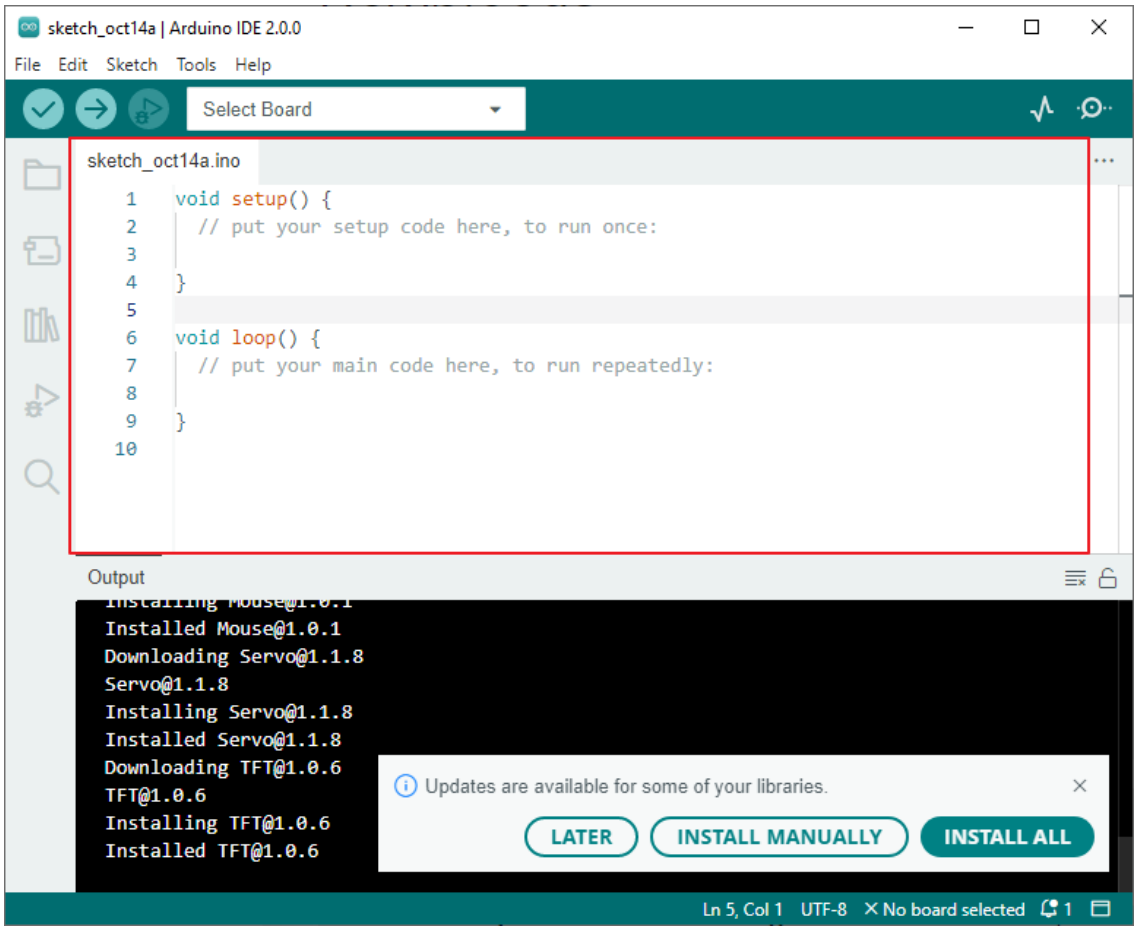



1. **検証（Verify）**：コードをコンパイルします。構文に問題がある場合、エラーが表示されます。
2. **アップロード（Upload）**：ボードにコードをアップロードします。ボタンをクリックすると、ボードの RX と TX の LED が高速で点滅し、アップロードが完了するまで点滅を続けます。
3. **デバッグ（Debug）**：行ごとのエラーチェックのため。
4. **ボード選択（Select Board）**：簡単にボードとポートを設定します。
5. **シリアルプロッタ（Serial Plotter）**：読取値の変化を確認します。
6. **シリアルモニタ（Serial Monitor）**：ボタンをクリックするとウィンドウが表示されます。コントロールボードから送信されたデータを受信します。デバッグに非常に便利です。

7. **ファイル（File）**：メニューをクリックするとドロップダウンリストが表示され、ファイルの作成、開く、保存、閉じる、パラメータの設定などが含まれます。
8. **編集（Edit）**：メニューをクリックします。ドロップダウンリストには、**切り取り（Cut）**、**コピー（Copy）**、**貼り付け（Paste）**、**検索（Find）** などの編集操作と、それに対応するショートカットが表示されます。
9. **スケッチ（Sketch）**：**検証（Verify）**、**アップロード（Upload）**、**ファイル追加（Add）** などの操作を含みます。重要な機能は **ライブラリを含める（Include Library）** - ライブラリを追加できます。
10. **ツール（Tool）**：いくつかのツールが含まれています。最も頻繁に使用されるのはボード（使用するボード）とポート（ボードの位置）です。コードをアップロードするたびに、これらを選択またはチェックする必要があります。
11. **ヘルプ（Help）**：初心者の場合、メニューのオプションを確認して、IDE の操作、紹介情報、トラブルシューティング、コードの説明などの必要なヘルプを取得できます。
12. **出力バー（Output Bar）**：ここで出力タブを切り替えます。
13. **出力ウィンドウ（Output Window）**：情報を表示します。
14. **ボードとポート（Board and Port）**：ここで、コードのアップロードのために選択されたボードとポートをプレビューできます。何か間違いがある場合は、**ツール（Tools）** -> **ボード（Board）** / **ポート（Port）** で再選択できます。
15. IDE の編集エリアです。ここでコードを記述できます。
16. **スケッチブック（Sketchbook）**：スケッチファイルを管理するため。
17. **ボードマネージャ（Board Manager）**：ボードドライバを管理するため。
18. **ライブラリマネージャ（Library Manager）**：ライブラリファイルを管理するため。
19. **デバッグ（Debug）**：コードのデバッグを支援します。
20. **検索（Search）**：スケッチからコードを検索します。

### 2.3.3 スケッチの作成、開く、保存方法は？

1. Arduino IDE を初めて開くか、新しいスケッチを作成すると、このようなページが表示されます。Arduino IDE が新しいファイルを作成してくれます。これを"スケッチ"と言います。

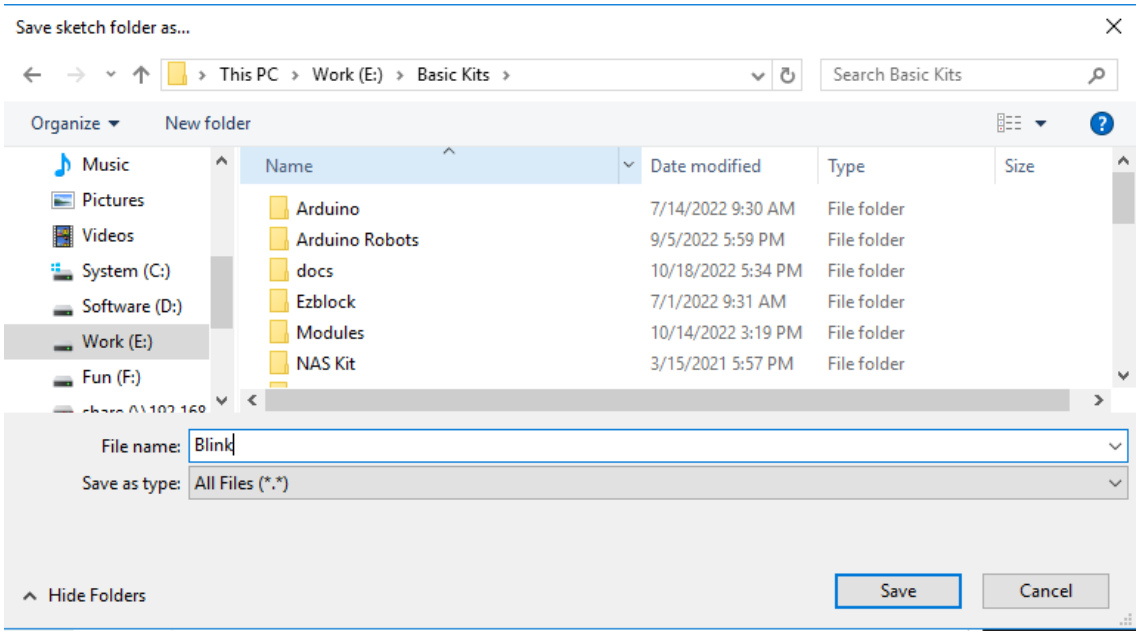



これらのスケッチファイルは一時的な名前が付けられており、ファイルの作成日から名前を判断することができます。 `sketch_oct14a.ino` は 10 月 14 日の最初のスケッチを意味し、 `.ino` はこのスケッチのファイル形式です。

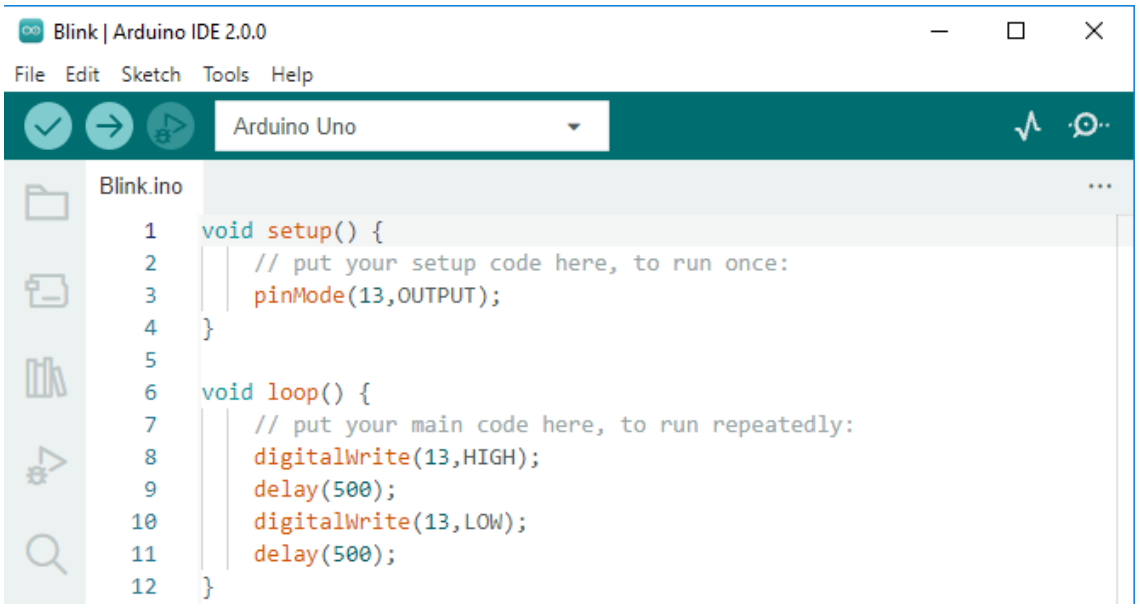
2. 新しいスケッチを作成してみましょう。以下のコードを Arduino IDE にコピーして、元のコードを置き換えます。


```
void setup() {
    // 初回のセットアップコードをここに記述:
    pinMode(13,OUTPUT);
}

void loop() {
    // 主要なコードをこちらに記述し、繰り返し実行:
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

3. Ctrl+S を押すか、 **ファイル（File）** -> **保存（Save）** をクリックします。デフォルトでスケッチは C:\Users\{your_user}\Documents\Arduino に保存されます。名前を変更したり、新しいパスを指定して保存することもできます。




4. 保存が成功すると、Arduino IDE の名前が更新されることがわかります。




次のセクションで、この作成したスケッチを Arduino ボードにアップロードする方法を学習してください。

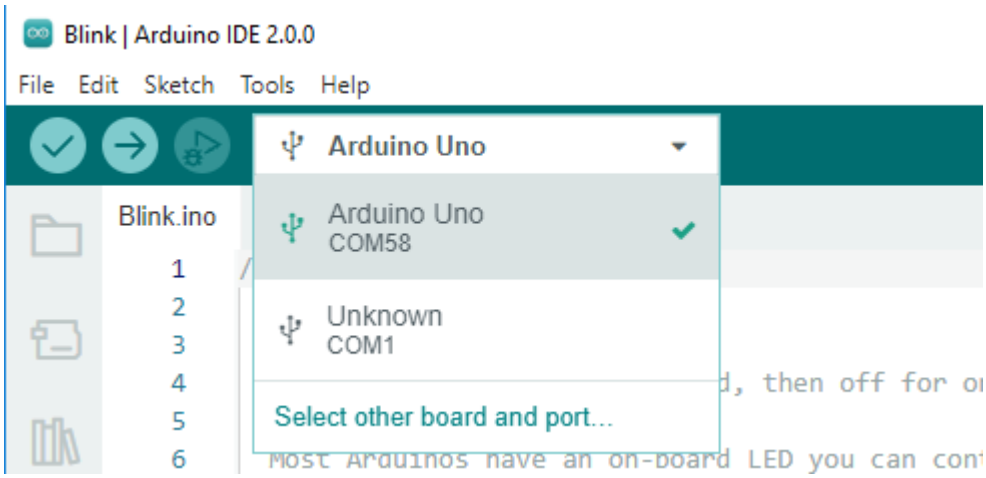
### 2.3.4 ボードにスケッチをアップロードする方法は？

このセクションでは、先ほど作成したスケッチを Arduino ボードにアップロードする方法、および考慮すべき点について学びます。

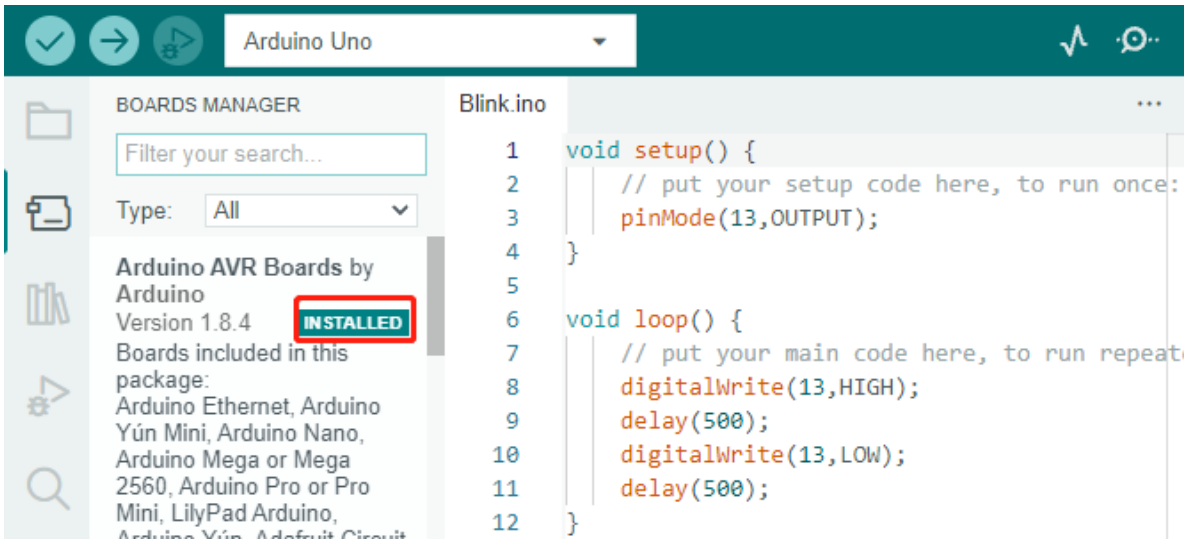
**1. ボードとポートを選択**

Arduino の開発ボードには通常、USB ケーブルが付属しています。これを使用してボードをコンピュータに接続します。

Arduino IDE で正しい **ボード（Board）** と **ポート（Port）** を選択します。通常、Arduino ボードはコンピュータに自動的に認識され、ポートが割り当てられるので、ここで選択できます。




もし、ボードが接続されているのに認識されない場合は、 **Boards Manager** の **Arduino AVR Boards** セクションに **INSTALLED** のロゴが表示されているか確認してください。表示されていない場合、少し下にスクロールして **INSTALL** をクリックしてください。

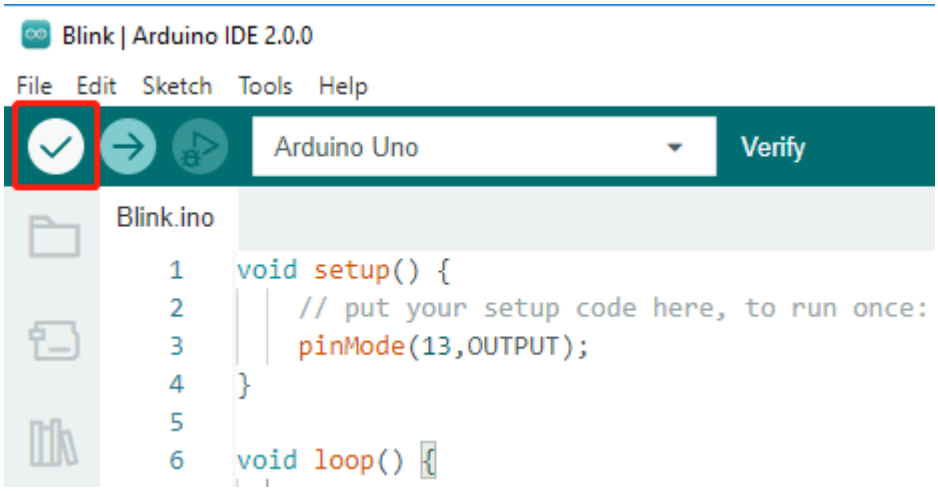



Arduino IDE を再度開いたり、Arduino ボードを再度接続すると、ほとんどの問題が解消されます。また、 **ツール**

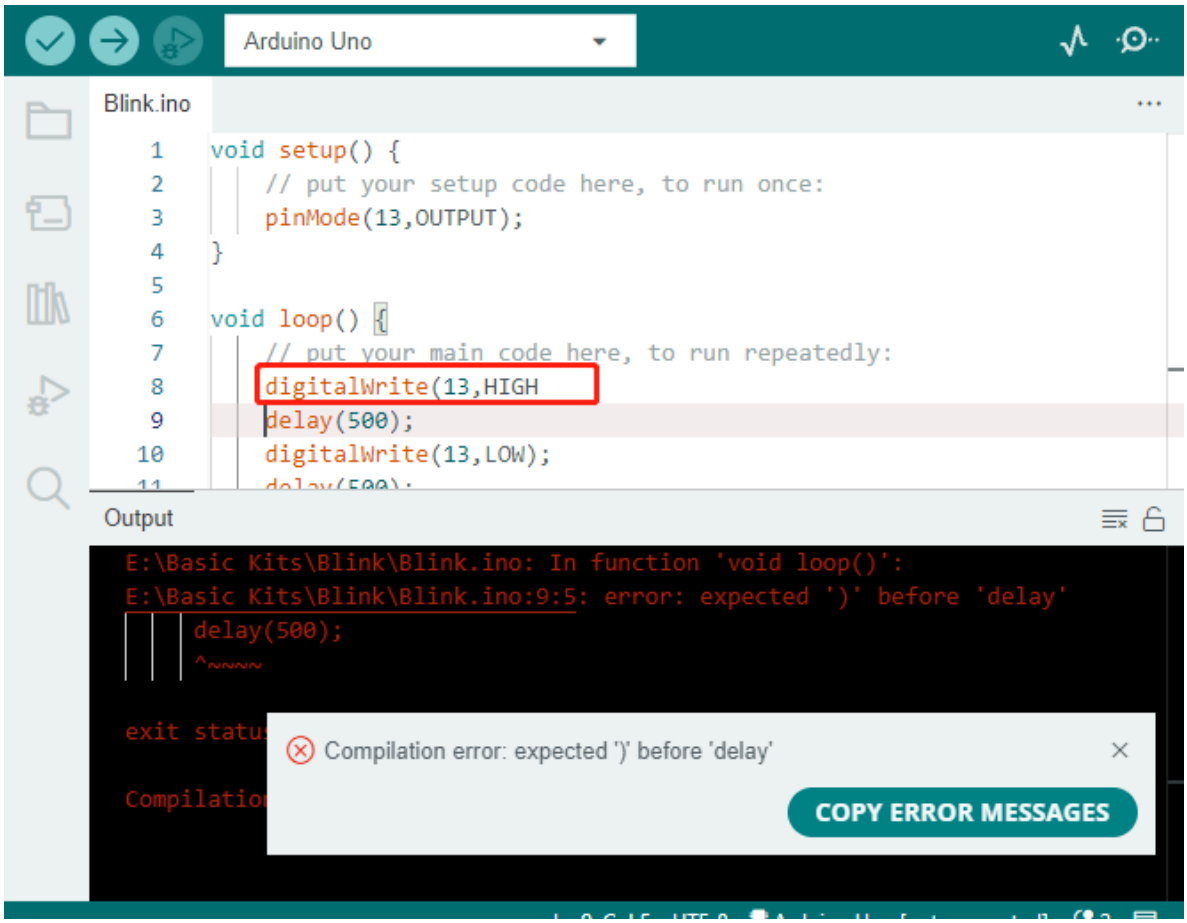
**（Tools） -> ボード（Board）** や **ポート（Port）** をクリックして選択することもできます。

**2. スケッチを検証**

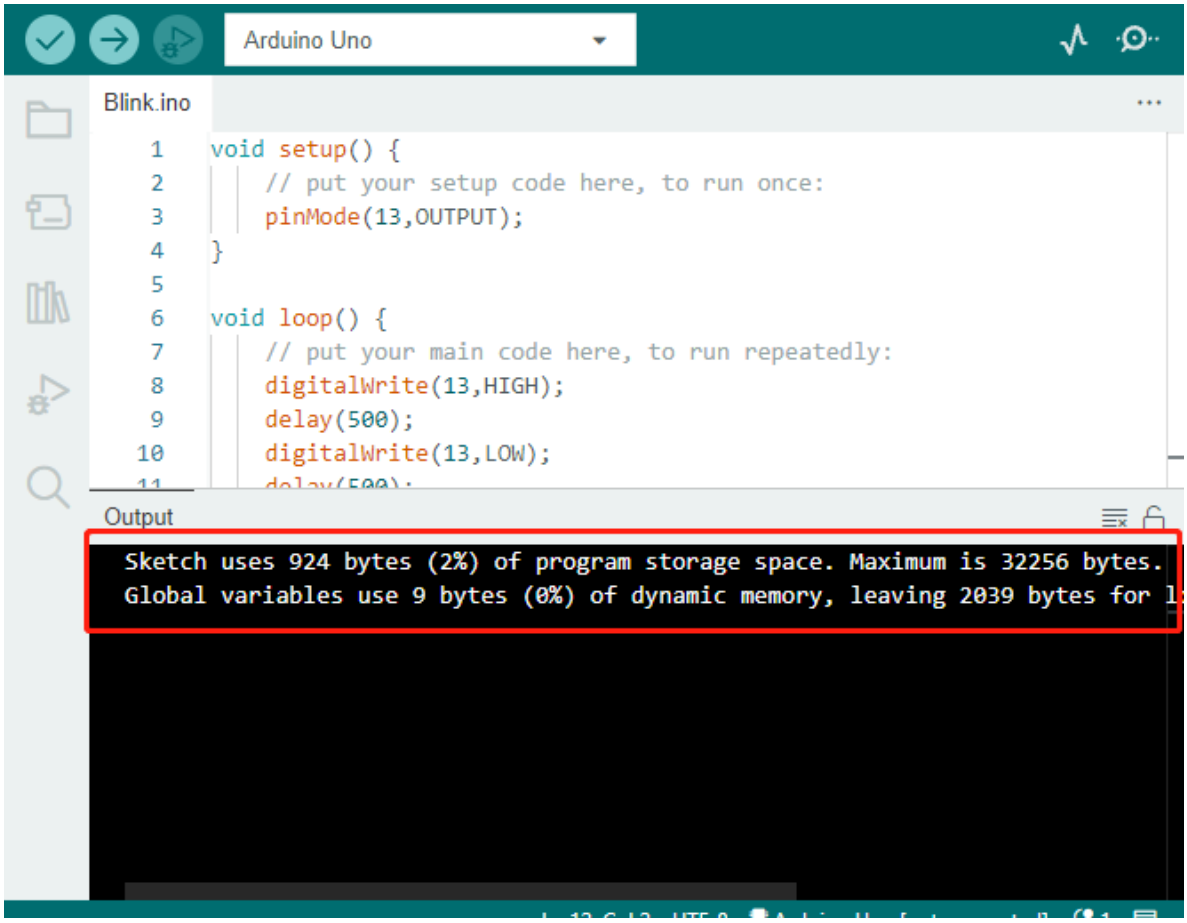
検証ボタンをクリックすると、スケッチがエラーがないかどうかコンパイルされます。




何らかの文字を削除したり、間違って数文字入力した場合など、ミスを見つけるのに役立ちます。メッセージバーから、どこでどのようなタイプのエラーが発生したかを確認できます。

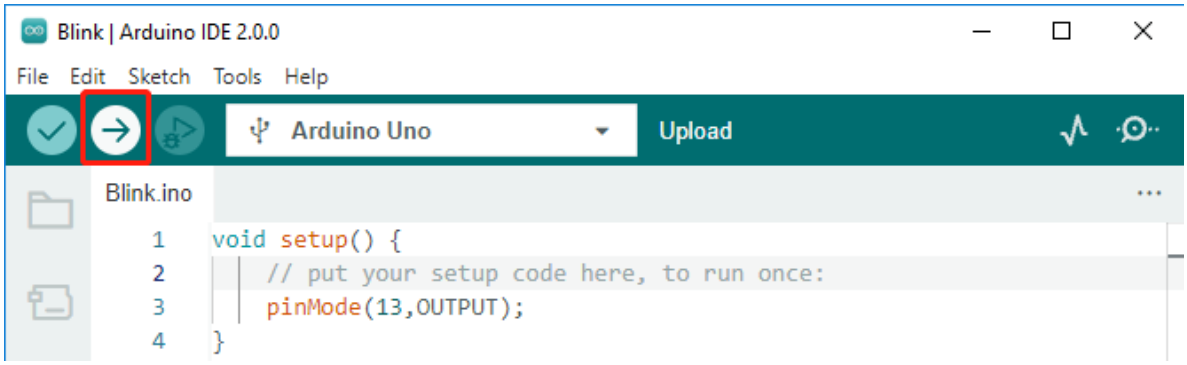


エラーがない場合、以下のようなメッセージが表示されます。

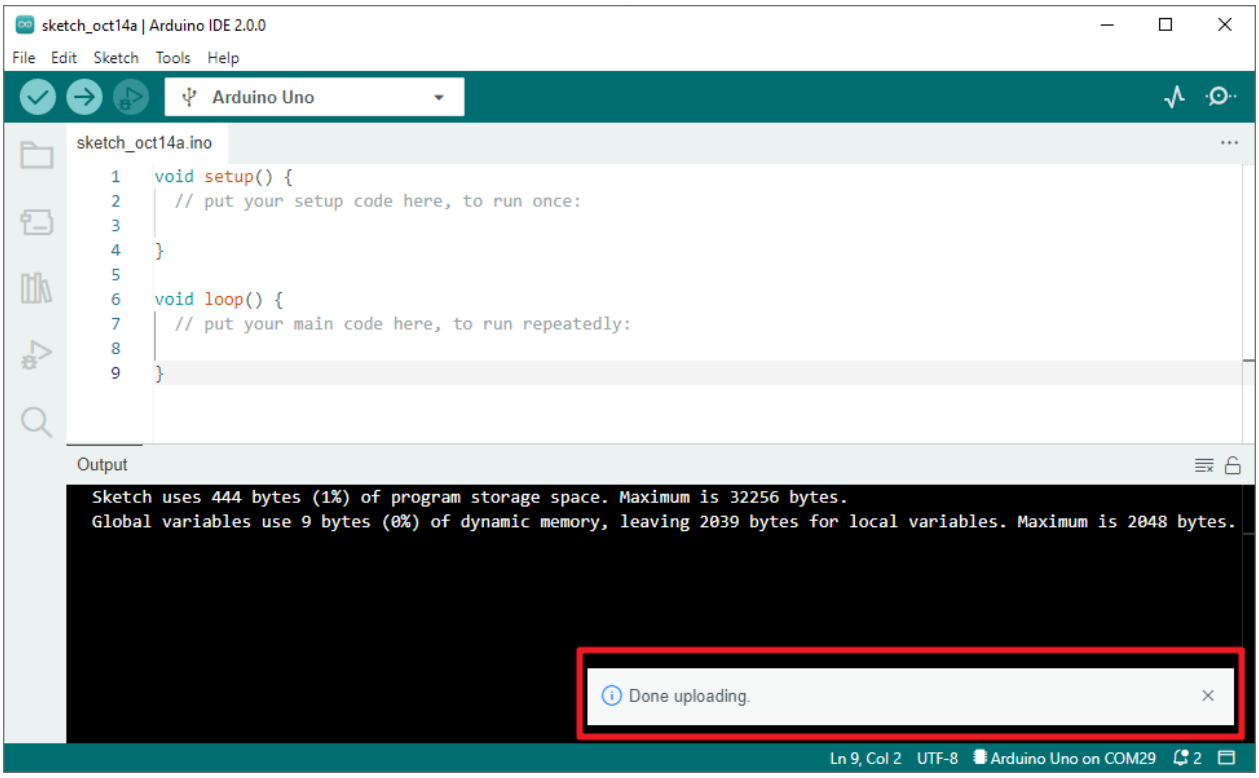



**3. スケッチをアップロード**

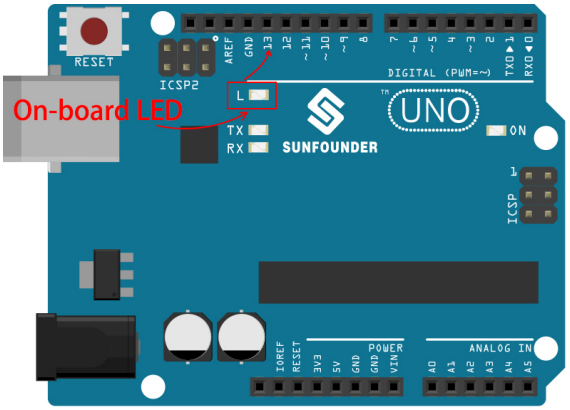
上記の手順を完了した後、 **アップロード（Upload）** ボタンをクリックして、このスケッチをボードにアップロードします。




成功すると、以下のプロンプトが表示されます。




同時に、ボード上の LED が点滅します。




スケッチがアップロードされた後、電源が適用されると Arduino ボードは自動的にスケッチを実行します。新しいスケッチをアップロードすることで、実行中のプログラムを上書きすることができます。

### 2.3.5 Arduino プログラムの構造

新しいスケッチファイルを見てみましょう。数行のコードが書かれていますが、実際には「空の」スケッチです。このスケッチを開発ボードにアップロードすると、何も起こりません。

```
void setup() {
// ここに初期設定のコードを記述し、一度だけ実行します:

}

void loop() {
// ここにメインのコードを記述し、繰り返し実行します:

}
```

`setup()` と `loop()` を削除して、スケッチを本当の `blank` ファイルにすると、検証に合格しないことがわかるだろう。これらは人間の骨格に相当し、欠かせないものです。

スケッチ作成時には、まず `setup()` が実行され、ボードに電源が供給されたりリセットされたりした後、その内部のコード（ `{}` 内部）が一度だけ実行されます。`loop()` はメインの機能を書くためのもので、 `setup()` 実行後に繰り返し実行されるコードを内部に記述します。

setup() と loop() の理解を深めるために、以下の四つのスケッチを使用します。それぞれの目的は Arduino のオンボード LED を点滅させることです。順番に各実験を実行して、具体的な効果を記録してください。

- スケッチ 1: オンボード LED を連続して点滅させる。

```
void setup() {
    // ここに初期設定のコードを記述し、一度だけ実行します:
    pinMode(13,OUTPUT);
}

void loop() {
    // ここにメインのコードを記述し、繰り返し実行します:
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

- スケッチ 2: オンボード LED を一度だけ点滅させる。

```
void setup() {
    // ここに初期設定のコードを記述し、一度だけ実行します:
    pinMode(13,OUTPUT);
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}

void loop() {
    // ここにメインのコードを記述し、繰り返し実行します:
}
```

- スケッチ 3: オンボード LED をゆっくり点滅させた後、早く点滅させる。

```
void setup() {
    // ここに初期設定のコードを記述し、一度だけ実行します:
    pinMode(13,OUTPUT);
    digitalWrite(13,HIGH);
    delay(1000);
    digitalWrite(13,LOW);
    delay(1000);
}

void loop() {
    // ここにメインのコードを記述し、繰り返し実行します:
    digitalWrite(13,HIGH);
    delay(200);
    digitalWrite(13,LOW);
    delay(200);
}
```

- スケッチ 4: エラーを報告する。

```
void setup() {
    // ここに初期設定のコードを記述し、一度だけ実行します:
    pinMode(13,OUTPUT);
}

digitalWrite(13,HIGH);
```

(次のページに続く)

(前のページからの続き)

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

void loop() {
    // ここにメインのコードを記述し、繰り返し実行します：
}
```

これらのスケッチを利用して、 setup-loop のいくつかの特徴をまとめることができます。

- ボードに電源が供給された後、 loop() は繰り返し実行されます。
- ボードに電源が供給された後、 setup() は一度だけ実行されます。
- ボードに電源が供給された後、まず setup() が実行され、次に loop() が実行されます。
- コードは setup() または loop() の {} の範囲内に書かれる必要があり、その枠組みの外に出るとエラーとなります。

---

**注釈:** digitalWrite(13,HIGH) のような命令はオンボード LED を制御するためのもので、後の章でその使用方法を詳しく説明します。

---

### 2.3.6 スケッチの作成ルール

あなたが友人にライトをつけてもらう場合、"ライトをつけてください"や"ライトオン、ブロ"と言うことができます。好きな口調で言うことができます。

しかし、Arduino ボードに何かをしてもらいたい場合は、Arduino のプログラム作成ルールに従ってコマンドを入力する必要があります。

この章では、Arduino 言語の基本的なルールを紹介し、自然言語をコードに変換する方法を理解するのに役立ちます。

もちろん、これは慣れるのに時間がかかるプロセスであり、初心者にとっては最もエラーが発生しやすい部分でもあるので、間違えることが多くても大丈夫です。何度も試してみてください。

### セミコロン ;

手紙を書くときのように、各文の最後に句点を書いて終わりとするように、Arduino 言語ではコマンドの終了をボードに伝えるために ; を使用する必要があります。

「ボード上の LED を点滅させる」というおなじみの例を取ってみましょう。正常なスケッチは次のようになります。

例:

```
void setup() {
    // ここに初期化コードを書く: 一度だけ実行されます
    pinMode(13,OUTPUT);
}

void loop() {
    // ここにメインのコードを書く: 繰り返し実行されます
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

次に、以下の 2 つのスケッチを見て、実行する前に Arduino に正しく認識されるかどうかを推測してみましょう。

スケッチ A:

```
void setup() {
    // ここに初期化コードを書く: 一度だけ実行されます
    pinMode(13,OUTPUT);
}

void loop() {
    // ここにメインのコードを書く: 繰り返し実行されます
    digitalWrite(13,HIGH)
    delay(500)
    digitalWrite(13,LOW)
    delay(500)
}
```

スケッチ B:

```
void setup() {
    // ここに初期化コードを書く：一度だけ実行されます
    pinMode(13,OUTPUT);
}

void loop() {
    // ここにメインのコードを書く：繰り返し実行されます
    digitalWrite(13,
HIGH);  delay
    (500
    );
    digitalWrite(13,

    LOW);
            delay(500)
    ;
}
```

結果として、 **Sketch A** はエラーを報告し、 **Sketch B** は実行されます。

- **Sketch A** のエラーは、 ；が欠落している点で、見た目は普通ですが、Arduino はこれを読むことができません。
- **Sketch B** は、見た目は人間には不親切ですが、実際には、Arduino プログラムではインデント、改行、ステートメントのスペースは存在しないので、Arduino のコンパイラにとっては、例のように見えます。

しかし、 **Sketch B** のようにコードを書かないでください。通常、コードを書き、閲覧するのは自然な人々ですので、自分自身を困らせないようにしてください。

#### 中括弧 {}

{} は Arduino プログラミング言語の主要なコンポーネントで、ペアで現れる必要があります。より良いプログラムの慣習は、左の中括弧を入力した直後に右の中括弧を入力して、構造体を挿入し、カーソルを中括弧の間に移動してステートメントを挿入することです。

### コメント //

コメントはコンパイラが無視するスケッチの部分です。通常、プログラムの動作方法を他者に伝えるために使用されます。

コードの行に隣接する 2 つのスラッシュを書くと、コンパイラはその行の最後まで何も無視します。

新しいスケッチを作成すると、2 つのコメントが含まれています。これらのコメントを削除しても、スケッチに影響はありません。

```
void setup() {
    // ここに初期化コードを書く：一度だけ実行されます
}

void loop() {
    // ここにメインのコードを書く：繰り返し実行されます
}
```

プログラミングでのコメントは非常に役立ちます。以下にいくつかの一般的な使用例を示します。

- 使用方法 A: このコードのセクションが何をするのか、自分自身や他者に伝える。

```
void setup() {
    pinMode(13,OUTPUT); // ピン 13 を出力モードに設定、ボード上の LED を制御します
}

void loop() {
    digitalWrite(13,HIGH); // ピン 13 をハイに設定して、ボード上の LED を活性化
    delay(500); // 500 ms のまま
    digitalWrite(13,LOW); // ボード上の LED をオフにする
    delay(500); // 500 ms のまま
}
```

- 使用 B: 一時的にいくつかの文を無効にする（削除せずに）そして、それらを使用する必要があるときにコメントを外すことで、それらを再度書き直す必要がない。これは、コードのデバッグを行い、プログラムのエラーの場所を特定するときに非常に便利です。

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

(次のページに続く)

(前のページからの続き)

```
}

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

**注釈:** コードを迅速にコメントまたはコメント解除するのに役立つショートカット Ctrl+/ を使用してください。

### コメント /**/

// と同様のコメントです。このコメントは複数の行にわたることができ、コンパイラが /* を読むと、 */ が現れるまで後続の内容を無視します。

例 1:

```
/* 点滅 */

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

void loop() {
    /*
    以下のコードはボード上の LED を点滅させます。
    delay() 内の数値を変更して、点滅の頻度を変更することができます。
    */
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

### #define

これは便利な C++ のツールです。

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

コンパイラは、それを読むときに自動的に identifier を token-string で置き換えます。これは、通常、定数定義に使用されます。

例として、define を使用してコードの可読性を向上させるスケッチを以下に示します。

```
#define ONBOARD_LED 13
#define DELAY_TIME 500

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

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

コンパイラにとって、実際には以下のように見えます。

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

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

identifier が置き換えられ、プログラム内に存在しないことがわかります。したがって、使用する際のいくつかの注意点があります。

1. token-string は手動でのみ変更でき、プログラムの算術で他の値に変換することはできません。

2. ; のようなシンボルの使用を避ける。例えば。

```
#define ONBOARD_LED 13;

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

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

コンパイラはそれを以下のように認識し、エラーとして報告されます。

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

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

**注釈:** #define の命名規則は、変数との混同を避けるために identifier を大文字にすることです。

### 2.3.7 変数

変数はプログラム内で最も強力で重要なツールの一つです。これにより、私たちのプログラム内でデータを保存し、呼び出すことができます。

以下のスケッチファイルは変数を使用しています。オンボード LED のピン番号を変数 ledPin に、数字 "500" を変数 delayTime に保存します。

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

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

(次のページに続く)

(前のページからの続き)

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

待って、これは #define の動作を複製しているのでしょうか？答えは NO です。

- #define の役割はテキストを単純かつ直接に置き換えることであり、プログラムの一部としてコンパイラに認識されることはありません。
- 一方、 variable はプログラム内に存在し、値を保存し呼び出すために使用されます。define ではできないことですが、プログラム内でその値を変更することもできます。

以下のスケッチファイルでは、変数に自己追加が行われ、各点滅ごとにオンボード LED の点滅が長くなります。

```
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; // 実行ごとに値を 200 増やします
}
```

### 変数の宣言

変数を宣言するとは、変数を作成することを意味します。

変数を宣言するには、データ型と変数名の 2 つが必要です。データ型は変数からスペースで区切り、変数の宣言は ; で終了する必要があります。

この変数を例に取り上げてみましょう。

```
int delayTime;
```

#### データ型

ここで int は整数型と呼ばれるデータ型であり、-32768 から 32766 までの整数を保存するために使用できます。また、小数を保存するためには使用できません。

変数は整数以外のさまざまなデータを保持することができます。Arduino 言語（実際には C++ です）は、以下に挙げる最も頻繁に使用されるものに対して組み込みのサポートを持っています。

- float: 小数点数を保存します、例：3.1415926。
- byte: 0 から 255 までの数値を保持します。
- boolean: True または False の 2 つの可能な値のみを保持しますが、メモリ内で 1 バイトを占有します。
- char: -127 から 127 までの数値を保持します。 char としてマークされると、コンパイラはそれを の文字に一致させようとします。
- string: 文字列、例： Halloween を保存できます。

#### 変数名

変数名は好きな名前に設定できます。例えば、 i、 apple、 Bruce、 R2D2、 Sectumsempra などですが、守るべき基本的なルールがいくつかあります。

1. それが何のために使用されるのかを説明します。ここでは、変数を delayTime と名付けたので、それが何をするのかを簡単に理解できます。変数名を barryAllen にするのも構いませんが、コードを見ている人に混乱をもたらします。
2. 一般的な命名法を使用します。CamelCase を使用することができ、私は delayTime の初めの T を使用して、変数が 2 つの単語で構成されていることが容易に分かるようにしました。また、UnderScoreCase を使用して変数を delay_time として書くこともできます。プログラムの実行に影響はありませんが、好みの命名法を使用すると、プログラマがコードを読むのが容易になります。
3. キーワードを使用しないでください。"int"をタイプするときと同じように、Arduino IDE はそれが特別な目的を持つ単語であることを思い出させるためにそれを色付けします。それは変数名として使用することができません。もしそれが色付けされていれば、変数の名前を変更します。
4. 特別な記号は許可されていません。例えば、スペース、#, $, /, +, % などです。英字（大文字・小文字を区別）、アンダースコア、数字の組み合わせは十分ですが、数字は変数名の最初の文字として使用することはできません。

#### 変数に値を割り当てる

変数を宣言したら、データを格納する時が来ます。代入演算子（すなわち =）を使用して変数に値を入れます。

変数を宣言すると同時にその値を割り当てることができます。

```
int delayTime = 500;
```

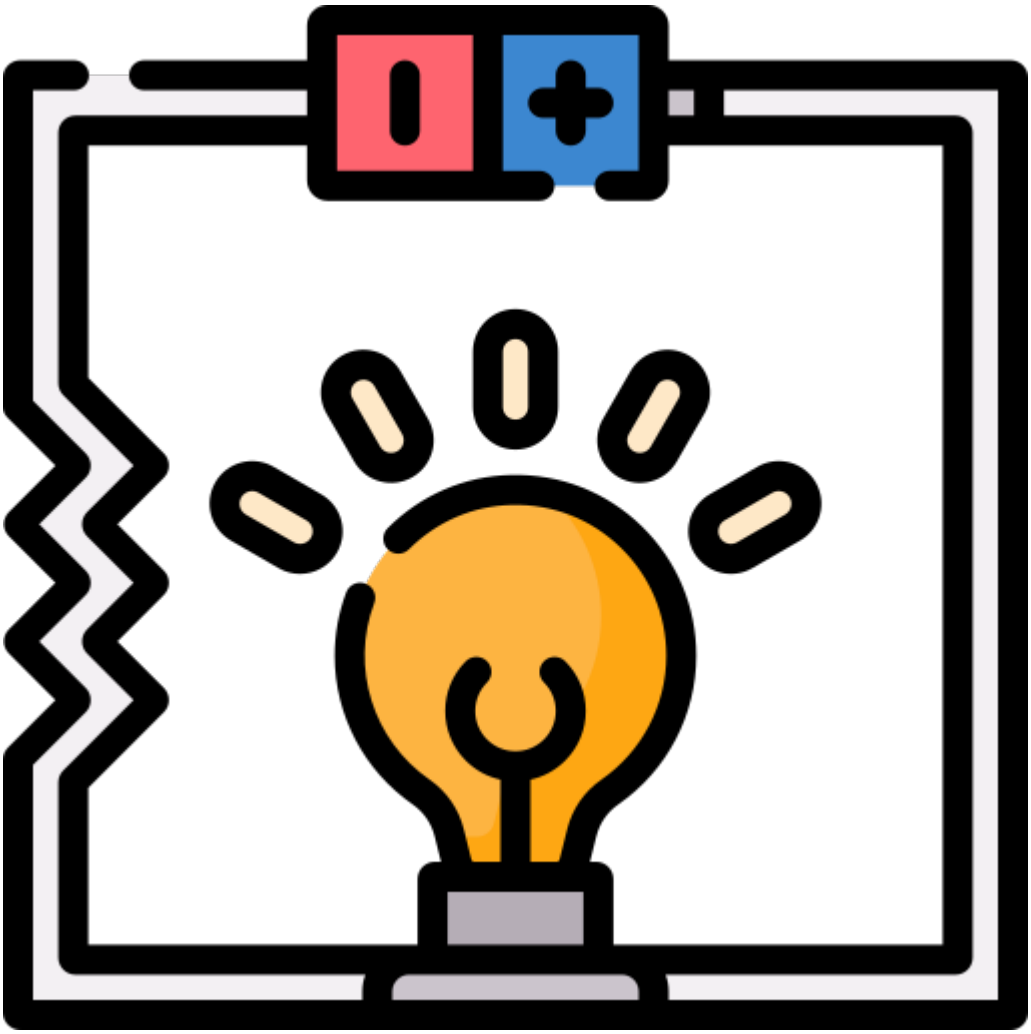
また、ある時点で新しい値を割り当てることも可能です。

```
int delayTime; // 値なし
delayTime = 500; // 値は 500
delayTime = delayTime +200; // 値は 700
```

### 2.3.8 回路の作成方法

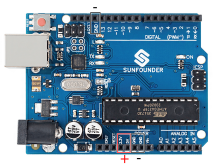
私たちが日常で使用している多くのものは、家の照明やこのコンピュータのように、電気で動いています。

電気を利用するためには、電気回路を組む必要があります。基本的に、回路は電気が流れる経路、または電子回路であり、特定の方法で接続された電気デバイスやコンポーネント（家電製品）から成り立っています。例としては、抵抗器、コンデンサ、電源、スイッチなどがあります。



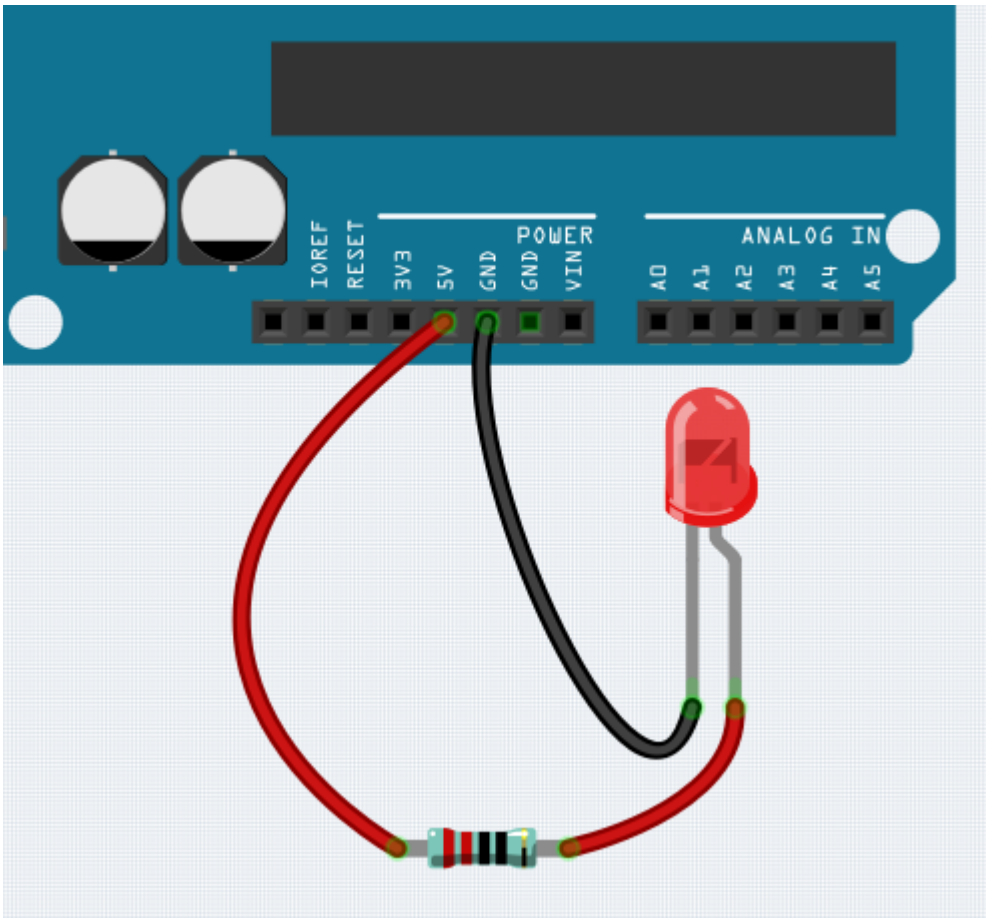
回路とは、電子が移動して電流を生む閉じた経路のことを指します。電流を流すためには、電源の正極と負極の間に導電経路が必要です。これを閉回路といい、これが切断されると開回路と呼ばれます。

Arduino ボードには、いくつかの電源出力ピン（正）とグラウンドピン（負）があります。これらのピンを電源の正極と負極として使用し、ボードに電源を接続することができます。



電気を使って、光や音、動きのある作品を作成することができます。LED の長いピンを正極に、短いピンを負極に接続することで LED を点灯させることができます。しかし、そのままでは LED がすぐに壊れてしまうため、回路内に 220 Ωの抵抗器を追加して保護する必要があります。

以下にその回路の形状を示します。




「この回路をどうやって組むの？」と疑問に思うかもしれません。ワイヤーを手で持って接続するのか、ピンとワイヤーをテープで固定するのか。

このような場面で、はんだ付けの不要なブレッドボードが非常に役立ちます。

### ブレッドボード、こんにちは！

ブレッドボードは、たくさんの小さな穴が開いている長方形のプラスチック板です。これらの穴を利用して、電子部品を簡単に挿入し、電子回路を組むことができます。ブレッドボードは電子部品を恒久的に固定しないので、何か問題が発生した場合でも、回路を簡単に修理してやり直すことができます。

---

**注釈:** ブレッドボードを使用するための特別なツールは必要ありません。しかし、多くの電子部品は非常に小さく、ピンセットを使用すると小さな部品をより簡単に取り扱うことができます。

---

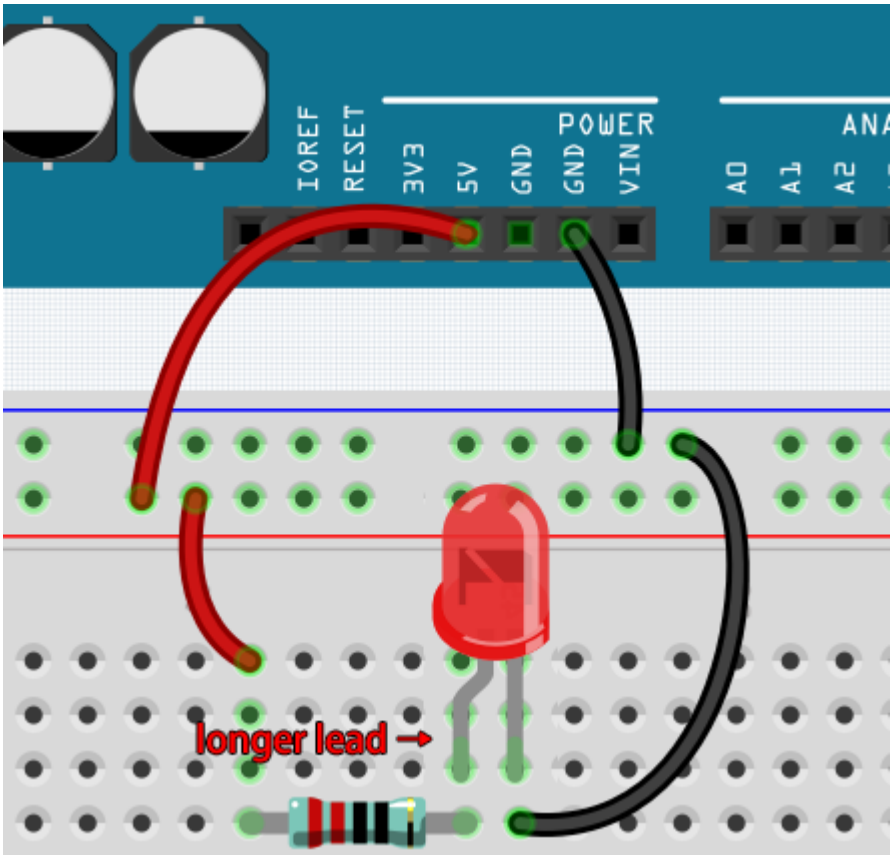
インターネット上にはブレッドボードに関する多くの情報があります。

- ブレッドボードの使用方法 - Science Buddies
- BREADBOARD って何？ - Makezine

ブレッドボードに関して知っておくべきいくつかの点を以下に示します。

1. 各半行グループ（例：行 1 の列 A-E や行 3 の列 F-J）は接続されています。したがって、A1 から電気信号が流れ込むと、B1、C1、D1、E1 から流れ出ることができますが、F1 や A2 からは流れ出すことはできません。
2. ほとんどの場合、ブレッドボードの両側は電源バスとして使用され、各列の穴（約 50 の穴）は互いに接続されています。一般的に、正の電源は赤いワイヤーの近くの穴に、負の電源は青いワイヤーの近くの穴に接続されます。
3. 回路内で、電流は負極に到達するまでの間に負荷を通過して正極から流れます。この場合、短絡が発生する可能性があります。

**電流の流れる方向に沿って、回路を組み立てましょう！**




1. この回路では、ボードの 5V ピンを使用して LED に電力を供給します。M2M ジャンパーワイヤーを使用して、それを赤い電源バスに接続します。

2. LED を保護するために、電流は 220 Ωの抵抗器を通過する必要があります。抵抗器の一方の端（どちらの端でもよい）を赤い電源バスに接続し、もう一方の端をブレッドボードのフリーローに接続します。

   ---

   **注釈:** 220 Ωの抵抗器のカラーリングは、赤、赤、黒、黒、茶色です。

   ---

3. LED を手に取ると、リードの一方が他方よりも長いことがわかります。長いリードを抵抗器と同じ行に接続し、短いリードを他の行に接続します。

   ---

   **注釈:** 長いリードはアノードと呼ばれ、回路の正面を表します。短いリードはカソードと呼ばれ、回路の負面を表します。

   アノードは抵抗器を介して GPIO ピンに接続する必要があり、カソードは GND ピンに接続する必要があります。

   ---

4. M2M ジャンパーワイヤーを使用して、LED の短いピンをブレッドボードの負の電源バスに接続します。

5. ジャンパーを使用して、ボードの GND ピンを負の電源バスに接続します。

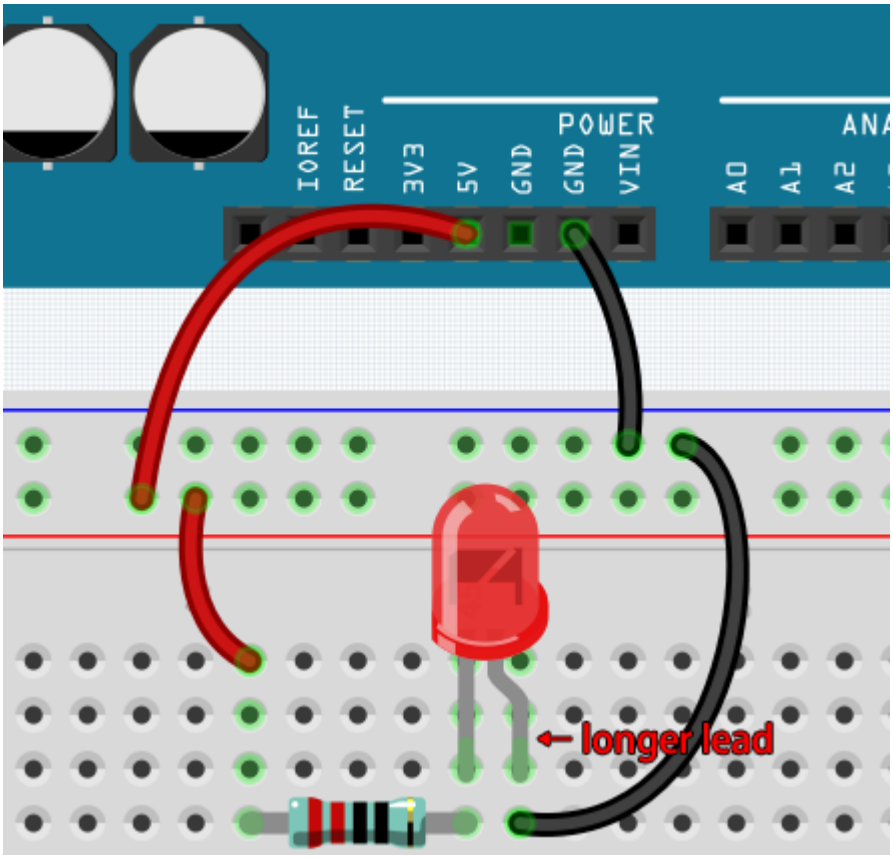
### 短絡に注意

短絡は、接続するべきでない二つの部品が「偶然」接続されたときに発生します。このキットには、長い金属のピンを持つ抵抗器、トランジスタ、コンデンサ、LED などが含まれており、これらがお互いにぶつかって短絡を引き起こす可能性があります。短絡が発生すると、いくつかの回路は正常に機能しなくなります。時折、短絡は電源とグラウンドバスの間で部品を恒久的に損傷させ、回路が非常に熱くなり、ブレッドボードのプラスチックが溶け、部品が焼けてしまうことがあります！

したがって、ブレッドボード上のすべての電子部品のピンが互いに接触していないことを常に確認してください。

### 回路の向き

回路には方向性があり、この方向性は特定の電子部品において非常に重要な役割を果たします。極性を持つデバイスがいくつかあり、これはその正と負の極に基づいて正しく接続する必要があることを意味します。方向を間違えて組み立てられた回路は正常に動作しません。




先ほど組み立てたこのシンプルな回路で LED を逆にすると、もう動作しなくなることがわかります。

対照的に、この回路の抵抗器のように、方向性を持たないデバイスもあります。そのため、それらを逆にしても

LED の正常な動作に影響はありません。

"+", "-", "GND", "VCC"などのラベルが付いている部品や異なる長さのピンを持つ部品は、特定の方法で回路に接続する必要があります。

### 回路の保護

電流は、完全な電気回路のある点を過ぎる電子の流れの速度です。基本的に、電流 = 流れです。アンペア（アンペア）は、電流を測定するための国際的な単位です。それは一定時間内に回路のある点を流れる電子の量（「電気的な充電」とも呼ばれる）を表します。

電流の流れの背後にある駆動力（電圧）は、電圧と呼ばれ、ボルト（V）で測定されます。

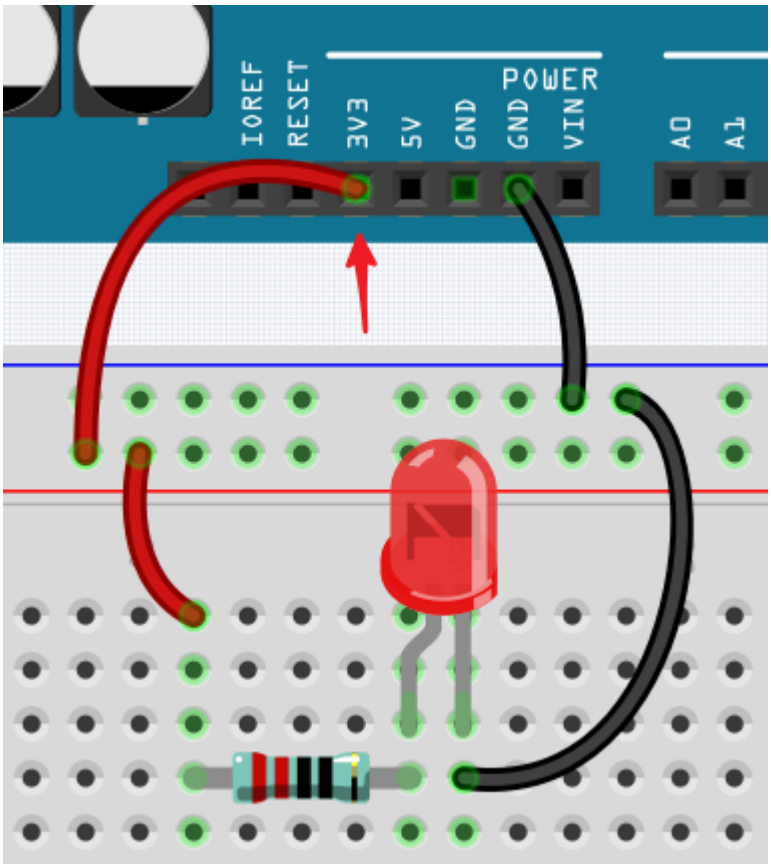
抵抗（R）は電流の流れを制限する材料の性質であり、オーム（Ω）で測定されます。

オームの法則によれば（温度が一定の場合)、電流、電圧、および抵抗は比例しています。回路の電流はその電圧に比例し、その抵抗に反比例します。

したがって、電流（I）= 電圧（V）/ 抵抗（R）です。

- オームの法則 - Wikipedia

オームの法則に関して、簡単な実験を行うことができます。



5V を 3.3V に接続するワイヤーを変更すると、LED の明るさが減少します。抵抗を 220 オームから 1000 オーム（色リング：茶、黒、黒、茶、茶）に変更すると、LED が以前よりも暗くなることに気付くでしょう。抵抗が大きいほど、LED は暗くなります。

---

**注釈:** 抵抗についての紹介や抵抗値の計算方法については、 **抵抗器** を参照してください。

---

ほとんどのパッケージ化されたモジュールは、適切な電圧（通常 3.3V または 5V）にのみアクセスする必要があります、例えば超音波モジュールのようなもの。

しかし、自作の回路では、電子デバイスの供給電圧と抵抗の使用に注意する必要があります。

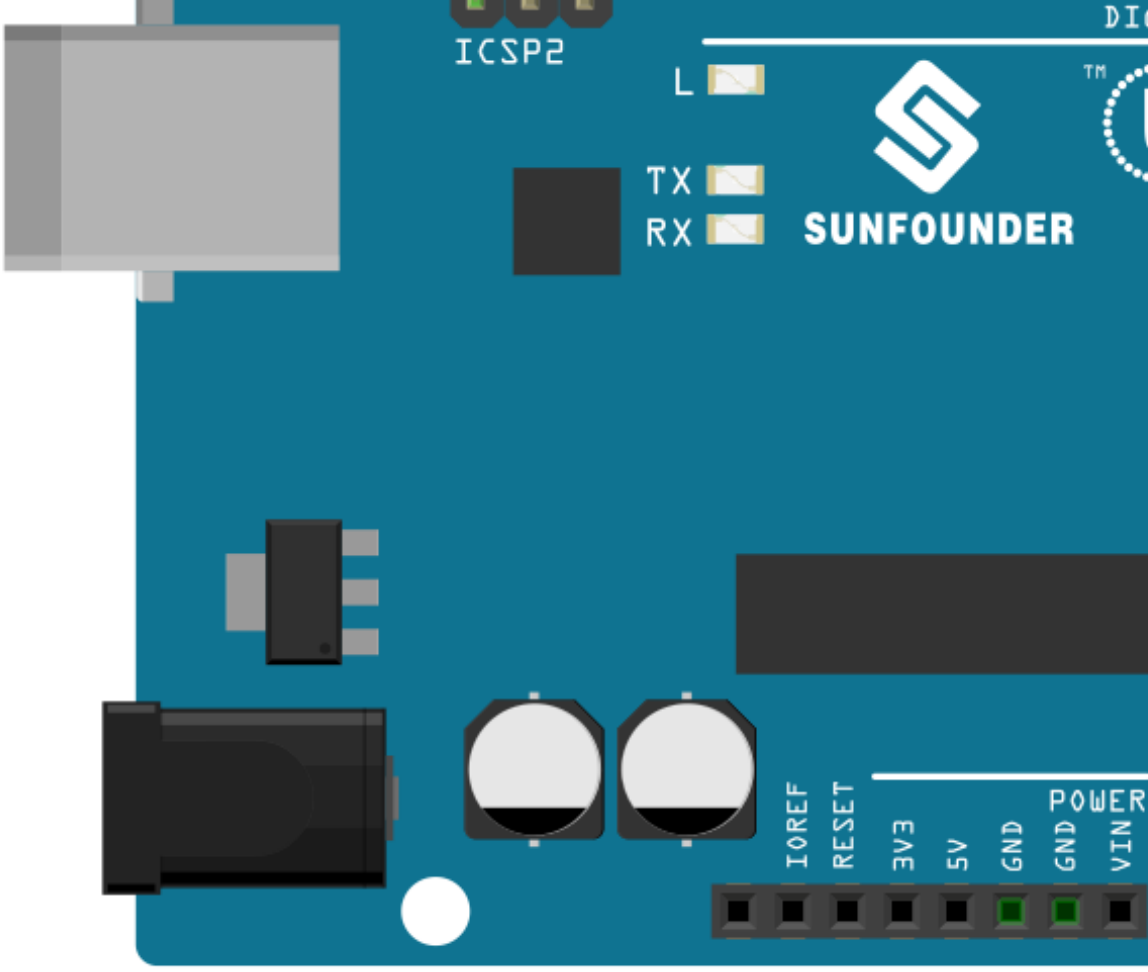
例として、LED は通常 20mA の電流を消費し、その電圧降下は約 1.8V です。オームの法則に従い、5V の電源を使用する場合、LED を焼き切らないためには、最低 160 オーム（(5-1.8)/20mA）の抵抗を接続する必要があります。

### Arduino での回路制御

Arduino のプログラミングと電子回路の基本的な理解ができたので、最も重要な問いに立ち向かう時がきました：Arduino で回路をどのように制御するか。

簡単に言えば、Arduino が回路を制御する方法は、ボード上のピンのレベルを変更することです。例えば、オンボード LED を制御するとき、ピン 13 に高いまたは低いレベルの信号を書き込みます。

さて、Arduino ボードをコード化して、ブレッドボード上の点滅する LED を制御してみましょう。LED がピン 9 に接続されているように回路を組み立てます。




次に、このスケッチを Arduino の開発ボードにアップロードします。

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

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

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

このスケッチは、オンボード LED の点滅を制御するために使用したものと非常に似ていますが、 `ledPin` の値が 9 に変更されている点が異なります。これは、今回、ピン 9 のレベルを制御しようとしているからです。

これで、ブレッドボード上の LED が点滅しているのが見えるでしょう。

# 第 3 章

# コードをダウンロード

以下のリンクから関連するコードをダウンロードしてください。

- Arduino 用 SunFounder 3 in 1 キット
- また、 Arduino 用 SunFounder 3 in 1 キット - GitHub でコードを確認してください。

# 第 4 章

# 基本的なプロジェクト

この章では、Arduino を使用して電子回路を制御する方法を学びます。

コンポーネントに応じて、Arduino の基本的な制御方法は 4 つのタイプに分けることができます。

- *1.* **デジタルライト**: ピンの出力電圧を高または低に設定し、ライトのオン/オフを制御できます。
- *2.* **アナログライト**: アナログ値（PWM 波）をピンに書き込み、ライトの明るさを調整できます。
- *3.* **デジタルリード**: デジタルピンのレベル信号を読み取り、スイッチの動作状態を読み取ることができます。
- *4.* **アナログ読み取り**: アナログピンの電圧を読み取り、ノブの動作状態を読み取ることができます。

また、追加のライブラリが必要なコンポーネントもあり、それらはセクション *5.11* **外部ライブラリのインストール** でまとめられています。

最後に、このキットには *6.* **面白いプロジェクト** も提供されており、多くのシンプルで役立つ操作が含まれています。このコードのセクションを試して、多くのシンプルなプロジェクトがどのように動作するかを理解してください。

## 4.1 1. デジタルライト

**デジタルライト** とは、デジタル信号をデジタルピンに出力または書き込むことです。デジタル信号には、0 または 1、0V または 5V の 2 つの状態しかないため、LED やブザーなどのコンポーネントをオン/オフにすることができます。

Arduino R3 ボードには、0 から 13 までの 14 個のデジタル I/0 ピンがあり、`pinMode()` と `digitalWrite()` 関数を使用して、これらのデジタルピンに高または低レベルを書き込むことができます。

- `pinMode(pin, mode)`: 特定のピンを `INPUT` または `OUTPUT` として設定します。ここでは `OUTPUT` として設定する必要があります。

**構文**

pinMode(pin, mode)

**パラメータ**

- pin: モードを設定する Arduino ピン番号。
- mode: INPUT, OUTPUT, または INPUT_PULLUP。

- digitalWrite(pin, value): デジタルピンに高レベル (5V) または低レベル (0V) を書き込み、コンポーネントの動作状態を変更します。pin が pinMode() で OUTPUT として設定されている場合、その電圧は次の値に設定されます：HIGH の場合は 5V（3.3V ボードでは 3.3V）、LOW の場合は 0V（グラウンド）。

**構文**

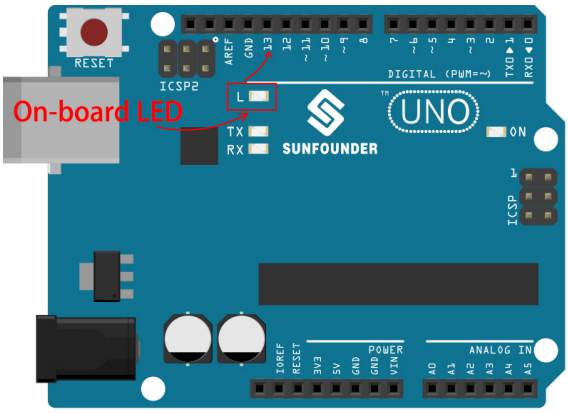
digitalWrite(pin, value)

**パラメータ**

- pin: Arduino のピン番号。
- value: HIGH または LOW。

**デジタルライトの例:**

```
const int pin = 13;

void setup() {
   pinMode(pin, OUTPUT);     // デジタルピンを出力として設定
}

void loop() {
   digitalWrite(pin, HIGH); // デジタルピンをオンに設定
   delay(1000);             // 1 秒待つ
   digitalWrite(pin, LOW);  // デジタルピンをオフに設定
   delay(1000);             // 1 秒待つ
}
```




**注意と警告**

- 0~13 のピンはすべてデジタルピンです。
- ピン 0 と 1 はコンピュータとの通信に使用されるので使用しないでください。これらのピンに何かを接続すると、通信が妨害され、ボードのアップロードが失敗することがあります。
- デジタルピンが使い果たされた場合、アナログピン（A0-A5）もデジタルピンとして使用できます。

**関連コンポーネント**

以下は関連するコンポーネントで、クリックすると使用方法を学ぶことができます。

### 4.1.1 1.1 ハロー、LED！

「Hello, world!」を表示することは、プログラミングを学ぶ最初のステップであるように、LED を駆動するプログラムを使用することは、物理的なプログラミングを学ぶ伝統的な導入です。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

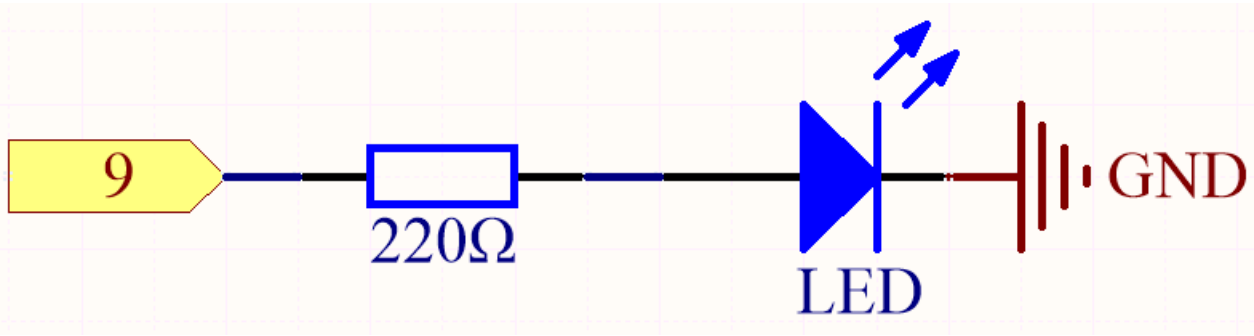
すべてのキットを購入するのは確かに便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

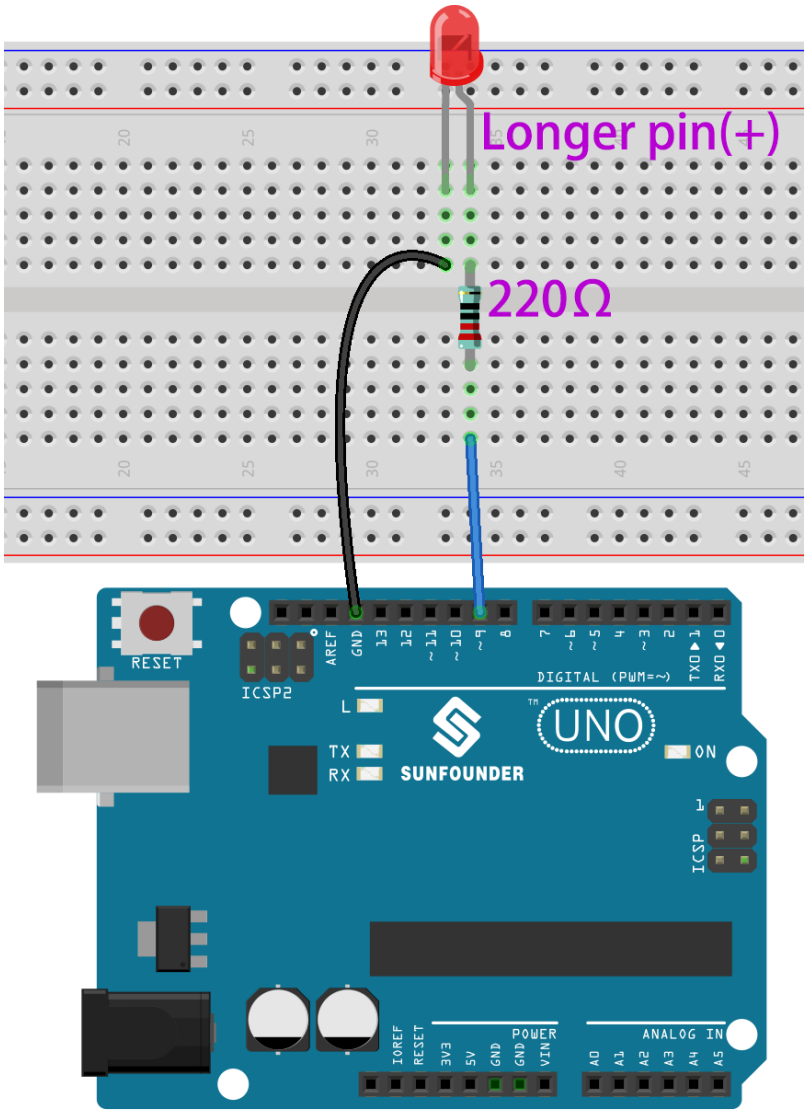
| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |

**回路図**




この回路の原理は単純で、図には電流の方向が示されています。ピン 9 が高レベル（5V）を出力すると、220ohm の電流制限抵抗を通過した後、LED が点灯します。ピン 9 が低レベル（0V）を出力すると、LED は消灯します。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\1.1.hello_led のパスの下にある 1.1.hello_led.ino ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードのアップロードが成功すると、LED が点滅するのが見えるでしょう。

**どのように動作するのか？**

ここでは、LED をデジタルピン 9 に接続しているため、プログラムの最初に ledpin という名前の int 変数を宣言し、9 という値を割り当てる必要があります。

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

次に、 setup() 関数内でピンを初期化し、ピンを OUTPUT モードに設定する必要があります。

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

loop() 内で、 digitalWrite() を使用して ledpin に 5V の高レベル信号を提供することで、LED のピン間に電圧差が生じ、LED が点灯します。

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

レベル信号が LOW に変わると、ledPin の信号が 0 V に戻って LED が消灯します。

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

オンとオフの間にインターバルが必要であり、変化を人々が見ることができるようにするため、 delay(1000) コードを使用してコントローラが 1000 ms 何もしないようにします。

```
delay(1000);
```

### 4.1.2 1.2 ビープ

アクティブブザーは LED を点灯させるのと同じくらい簡単に使える典型的なデジタル出力デバイスです！

キットには 2 種類のブザーが含まれています。アクティブブザーを使用する必要があります。裏返してみると、密封された背面（露出している PCB ではない）が必要なものです。


**必要な部品**

このプロジェクトでは、以下の部品が必要です。

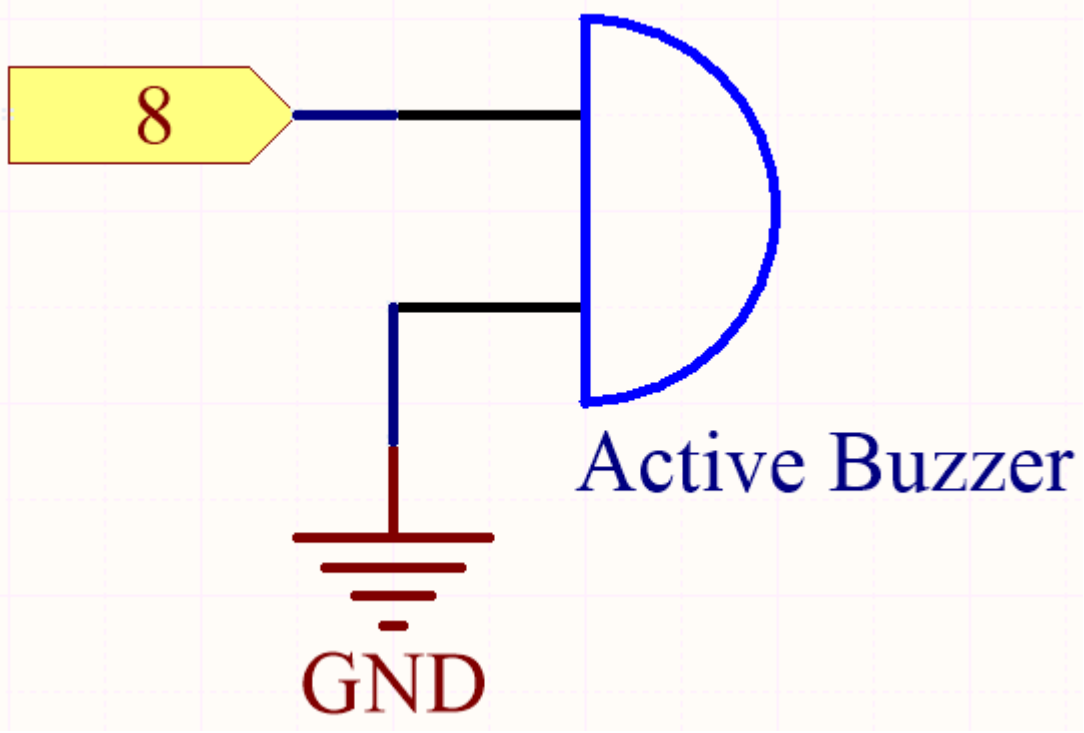
全体のキットを購入するのは非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

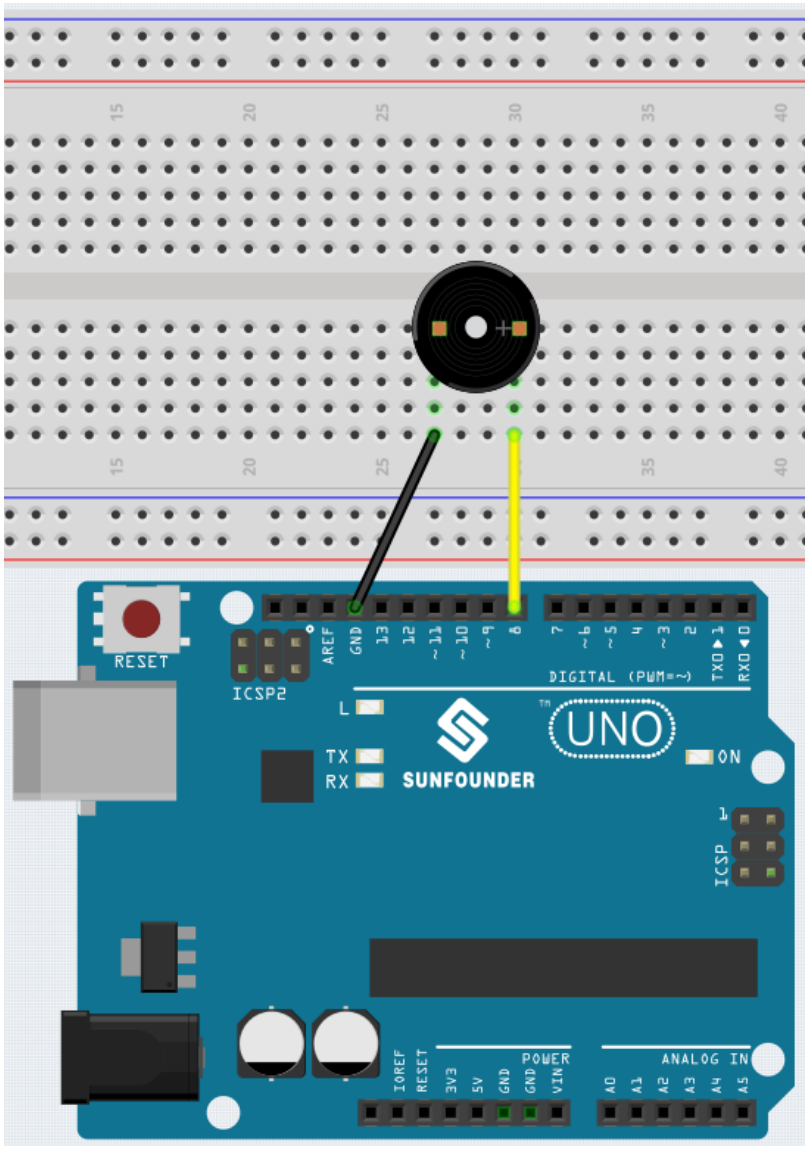
以下のリンクから別々に購入することもできます。

| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| **ブザー** | - |

**回路図**




配線図




**コード**

---

**注釈:**

- ファイル `1.2.beep.ino` を `3in1-kit\basic_project\1.2.beep` のパスで開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされると、毎秒ビープ音が聞こえます。

### 4.1.3 1.3 車輪を回す

モーターは典型的なデジタル出力デバイスであり、LED と同様の方法で使用されます。ただし、モーターは大電流で駆動する必要があり、大電流は R3 ボードなどのメイン制御ボードを損傷させる可能性があります。そのため、この場合には L298N モジュールを使用して、R3 ボードがモーターを安全に制御するのに役立ちます。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

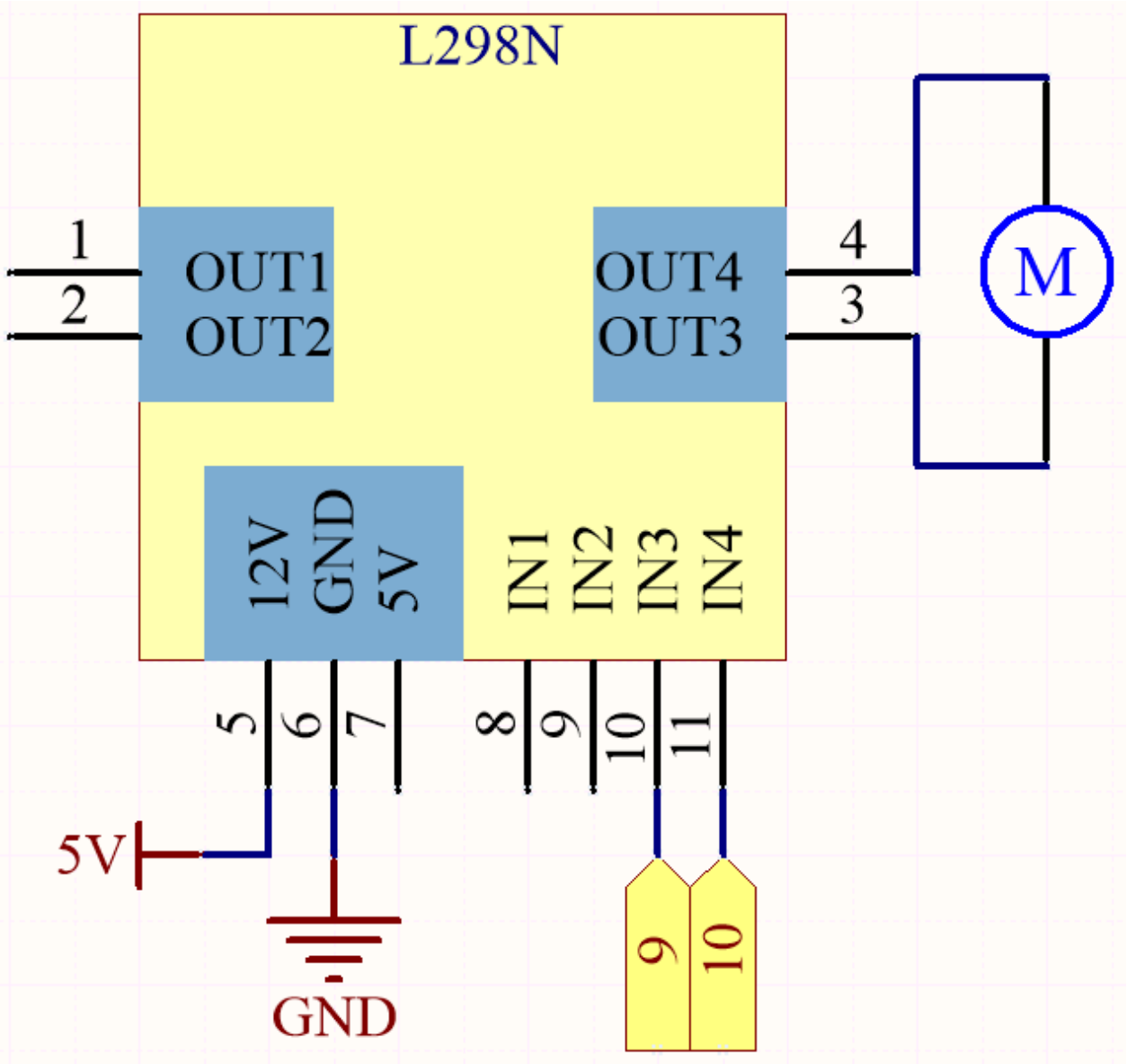
全体のキットを購入するのは非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから別々に購入することもできます。

| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| *TT* モーター | - |
| *L298N* モジュール | |

**回路図**

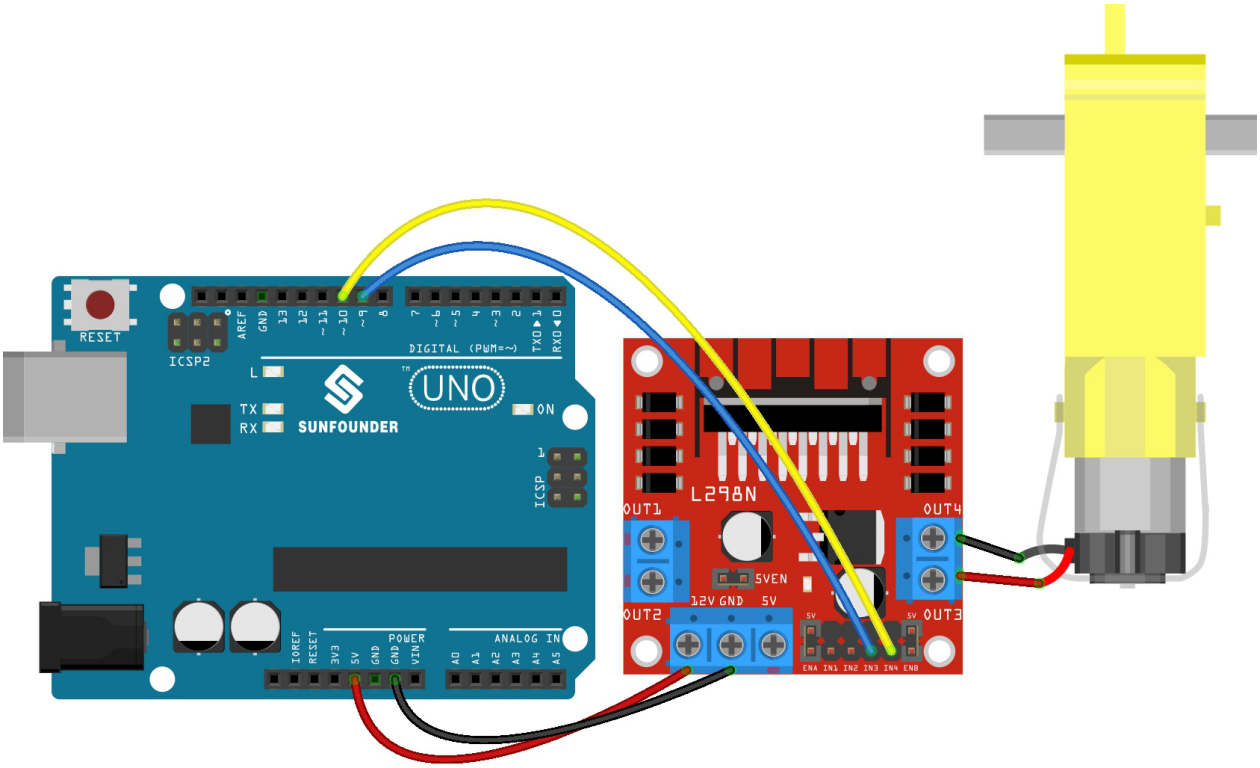



IN1～IN4 は L298N モジュールの入力であり、OUT1～OUT4 は出力です。

使い方としては、INx に高レベルを入力すると、OUTx が高レベルを出力します。INx に低レベルを入力すると、OUTx が低レベルを出力します。モーターの両端を OUT1 と OUT2 に接続し、IN1 と IN2 に逆のレベル信号を入力すると、モーターが回転します。OUT3 と OUT4 も同様の方法で使用できます。

**配線図**

| L298N | R3 ボード | モーター |
|---|---|---|
| 12V | 5V | |
| GND | GND | |
| IN3 | 9 | |
| IN4 | 10 | |
| OUT3 | | モーターのワイヤー |
| OUT4 | | モーターのワイヤー |




**コード**

---

**注釈:**

- ファイル `1.3.turn_the_wheel.ino` を `3in1-kit\basic_project\1.3.turn_the_wheel` のパスで開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

---

### 4.1.4 1.4 ポンピング

水ポンプもモーターの一つで、モーターや他の外部エネルギーの機械エネルギーを特別な構造を通じて液体を輸送するものです。

**必要な部品**

このプロジェクトには以下の部品が必要です。

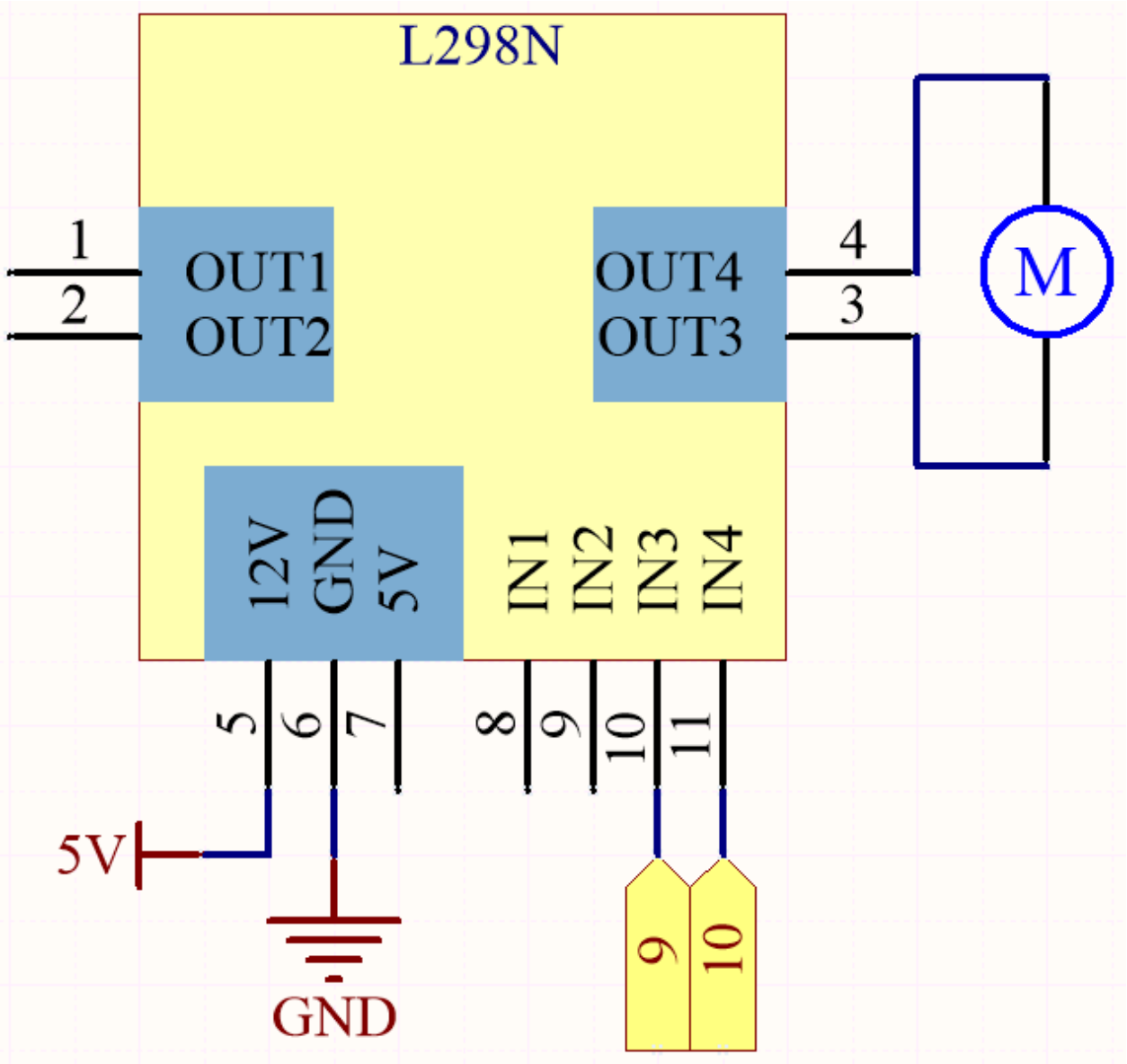
キット全体を購入すると非常に便利です、リンクは以下です:

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| *L298N* モジュール | |
| 遠心ポンプ | - |

**回路図**

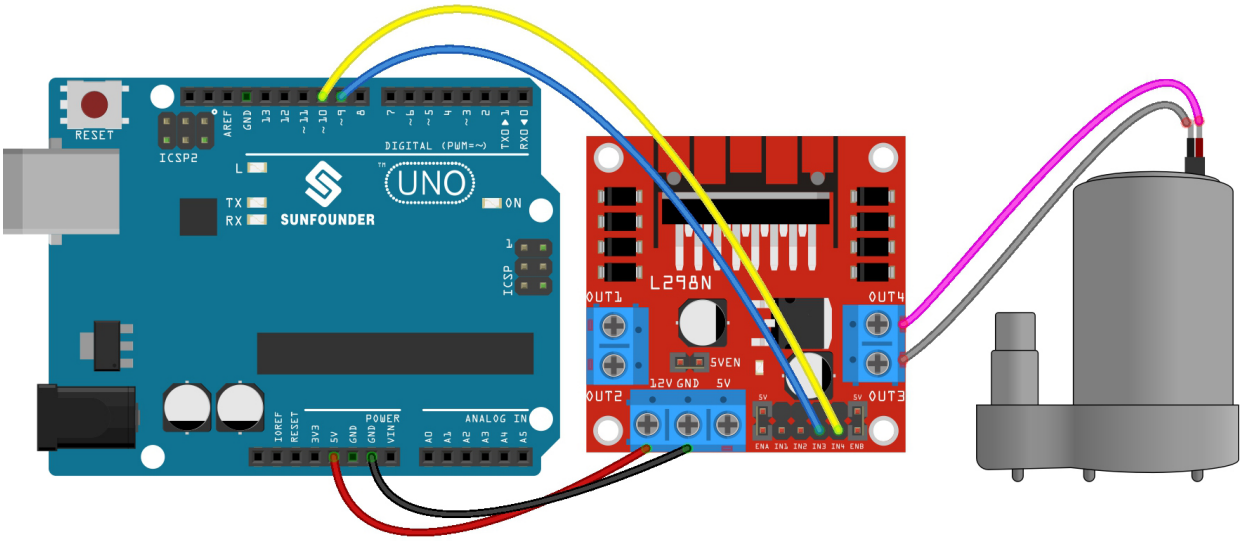



IN1～IN4 は L298N モジュールの入力であり、OUT1～OUT4 は出力です。

それらを使う簡単な方法は：INx に高レベルを入力すると、OUTx は高レベルを出力します。INx に低レベルを入力すると、OUTx は低レベルを出力します。モーターの両端を OUT1 と OUT2 に接続し、IN1 と IN2 に逆のレベルの信号を入力するとモーターが回転します。OUT3 と OUT4 も同じように使用できます。

**配線図**

| L298N | R3 ボード | モーター |
|---|---|---|
| 12V | 5V | |
| GND | GND | |
| IN3 | 9 | |
| IN4 | 10 | |
| OUT3 | | モーターのワイヤーの一方 |
| OUT4 | | モーターのワイヤーの一方 |




**コード**

---

**注釈:**

- ファイル `1.4.pumping.ino` を `3in1-kit\basic_project\1.4.pumping` のパスから開くことができます。
- また、このコードを **Arduino IDE** にコピーしてください。
- あるいは、 Arduino Web Editor を使ってコードをアップロードしてください。

---

ポンプにチューブを追加して、それをベースンに置きます。コードが正常にアップロードされた後、しばらくするとベースンの水が排水されるのを確認できます。この実験を行う際は、短絡を避けるため回路を水から離してください！

## 4.2 2. アナログライト

Arduino の 14 のデジタルピンのうち、6 つには PWM 出力機能も備わっています。したがって、これら 6 つのピンにデジタル信号を書き込むことに加えて、アナログ信号（PWM 波信号）も書き込むことができます。このようにして、LED をさまざまな明るさで点灯させたり、モーターをさまざまな速度で回転させることができます。

パルス幅変調、または PWM は、デジタル手段でアナログの結果を得るための技術です。文字通りの意味を理解するのは難しいかもしれませんので、LED の強度を制御する例を挙げて、より理解しやすく説明します。

高レベルと低レベルからなるデジタル信号をパルスと呼びます。これらのピンのパルス幅は、ON/OFF の速度を変えることで調整できます。簡単に言えば、LED を短い時間（例えば、ほとんどの人の視覚的な滞留時間である 20ms）でオン、オフ、オンとすると、LED が消えたことには気づかず、光の明るさがわずかに弱くなります。この期間中、LED がオンになっている時間が長いほど、LED の明るさは強くなります。つまり、一定の期間内で、パルスが広いほど、マイクロコントローラによって出力される「電気信号の強さ」が大きくなります。

PWM 波を書き込むための関数は以下の通りです。

- `analogWrite(pin, value)`

  ピンにアナログ値（PWM 波）を書き込みます。指定されたパルス信号を生成することで、異なる出力電圧（0-5V）をシミュレートできます。新しい read または write 文で呼び出されるまで、ピンはこの信号を保持します。

  **文法**

  analogWrite(pin, value)

  **パラメータ**

  - `pin`: 書き込む Arduino のピン。許容されるデータタイプ: int。
  - `value`: デューティサイクル：0（常にオフ）から 255（常にオン）の間。許容されるデータタイプ: int。

**アナログライトの例**

```
int pin = 9;      // PWM ピンに接続
void setup() {
   pinMode(pin, OUTPUT);  // ピンを出力として設定
}

void loop() {
   for (int i = 0 ;i<255 ; i++){
      analogWrite(pin, i); //analogWrite の値は 0 から 255 まで
      delay(30);
```

(次のページに続く)

(前のページからの続き)

```
    }
}
```

**注意と警告**

- R3 ボードをよく見ると、"~"の記号でマークされたピンにはアナログ出力機能があります。
- ピン 5 および 6 で生成される PWM 出力は、予想よりもデューティサイクルが高くなります。これは、これらの PWM 出力を生成するために使用される内部タイマーを共有する `millis()` および `delay()` 関数との相互作用のためです。これは主に低デューティサイクルの設定（例えば、0 - 10）で気づかれることが多く、ピン 5 および 6 の出力を完全にオフにしない 0 の値の結果として気づくことがあります。

**関連コンポーネント**

以下は関連するコンポーネントで、クリックするとそれらの使用方法を学ぶことができます。

### 4.2.1 2.1 フェージング

このプロジェクトは *1.1* **ハロー、*LED*！** に似ていますが、信号の種類が異なります。前者はデジタル信号（0&1）を出力して LED の点灯/消灯を制御するのに対し、このプロジェクトはアナログ信号を出力して LED の明るさを調節します。

**必要な部品**

このプロジェクトには、以下のコンポーネントが必要です。

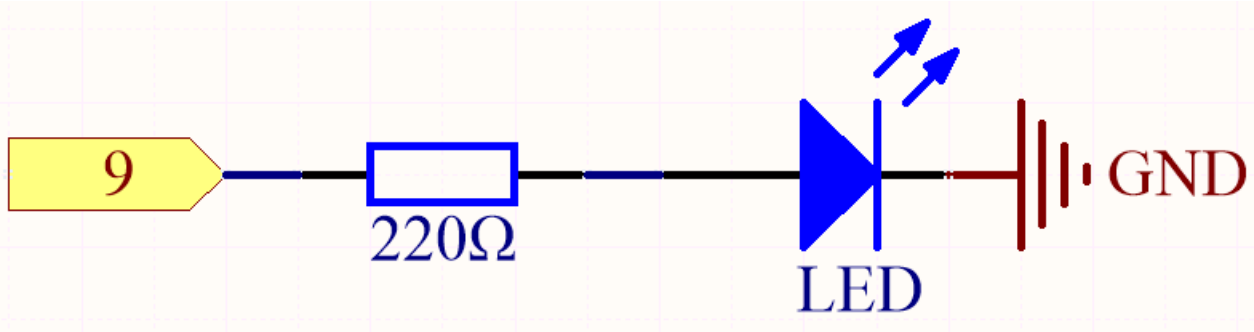
キット全体を購入すると非常に便利です。リンクは以下の通りです。

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

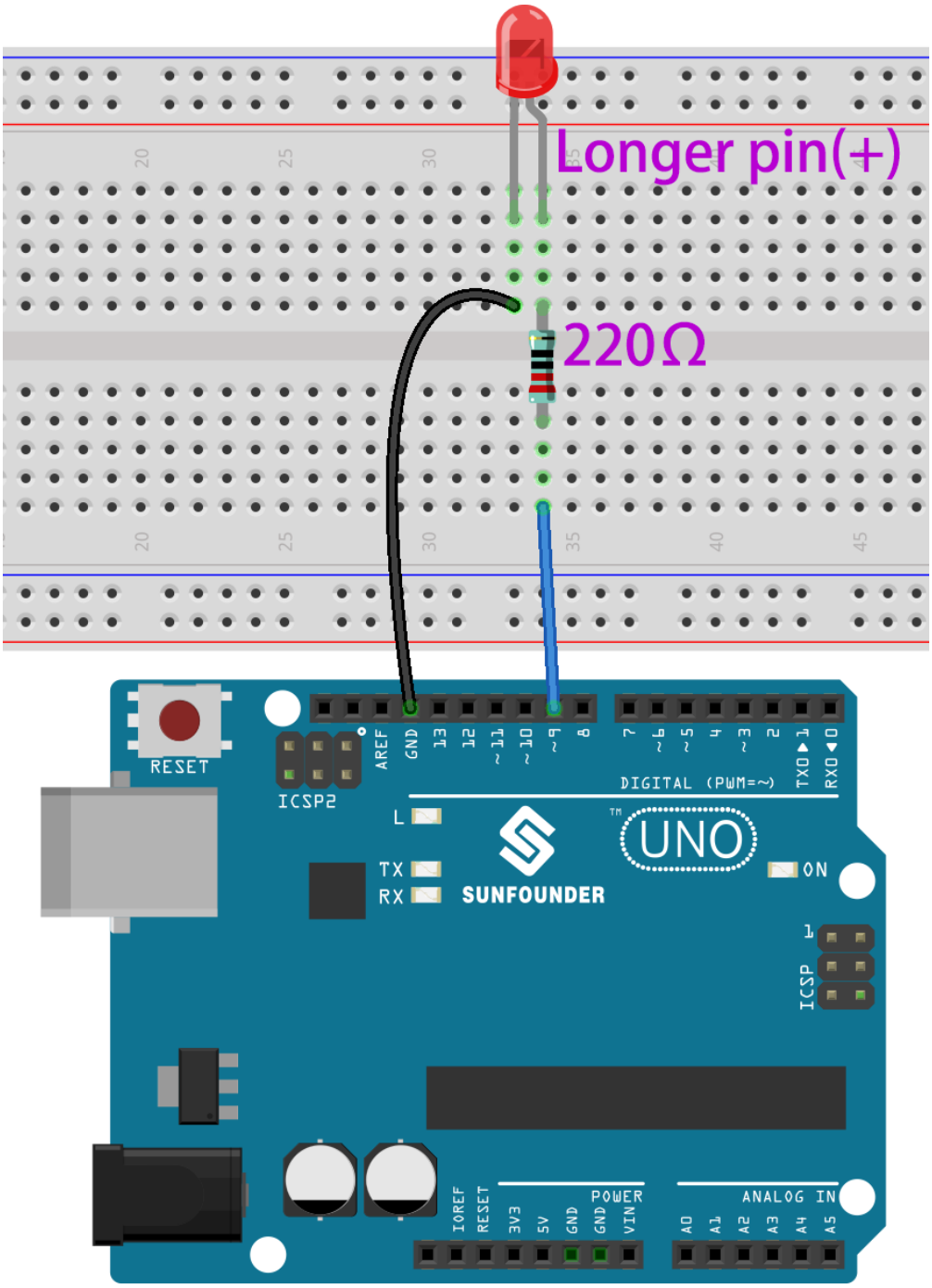
以下のリンクから、個別にも購入可能です。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |

**回路図**




**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\2.analogWrite\2.1.fading のパスの下の 2.1.fading.ino ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーしてください。

---

- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされた後、LED が息を吹きかえしているのが見えます。

### 4.2.2 2.2 カラフルな光

光は重ね合わせることができることは周知の通りです。例えば、青い光と緑の光を混ぜるとシアン色の光が得られ、赤い光と緑の光を混ぜると黄色の光が得られます。これは「加色混色の方法」と呼ばれます。

- 加色混色 - Wikipedia

この方法に基づいて、三原色を使用して、異なる比重に従って任意の色の可視光を混合することができます。例えば、赤色を多く、緑色を少なくすることでオレンジ色が得られます。

この章では、RGB LED を使用して、加色混色のミステリーを探求します！

RGB LED は、一つのランプキャップの下に赤、緑、青の LED を一つずつカプセル化したものと同等であり、これら三つの LED は共通のカソードピンを共有しています。各アノードピンに電気信号が供給されるので、対応する色の光が表示されます。各アノードの電気信号の強度を変更することで、さまざまな色を出力することができます。

**必要な部品**

このプロジェクトには、以下のコンポーネントが必要です。

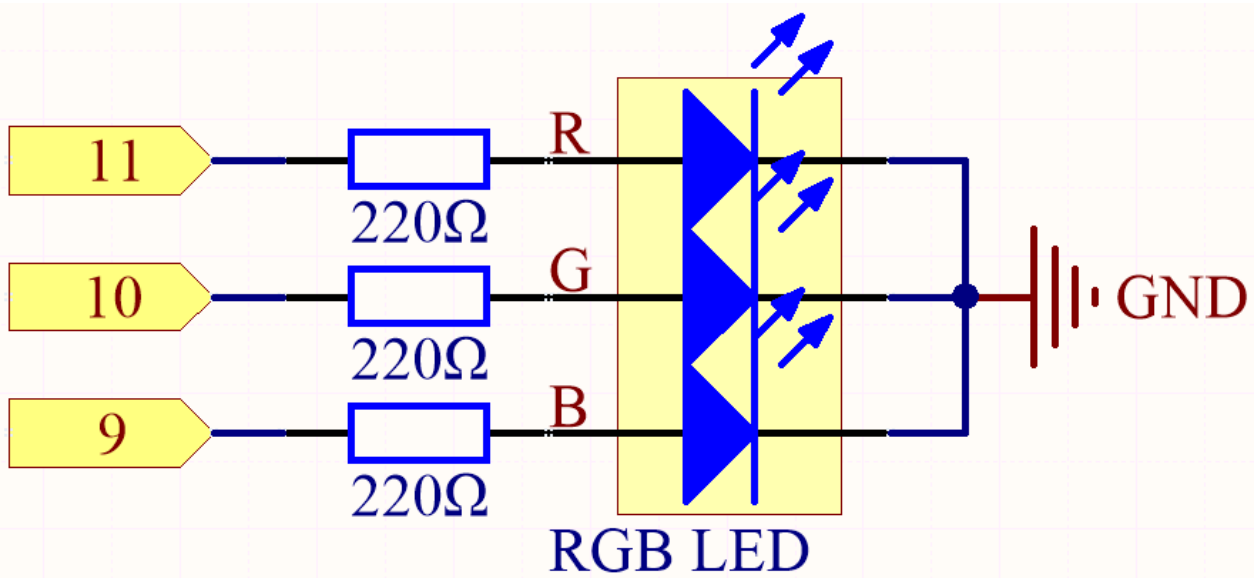
キット全体を購入することは非常に便利です。以下はリンクです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから、個別にも購入可能です。

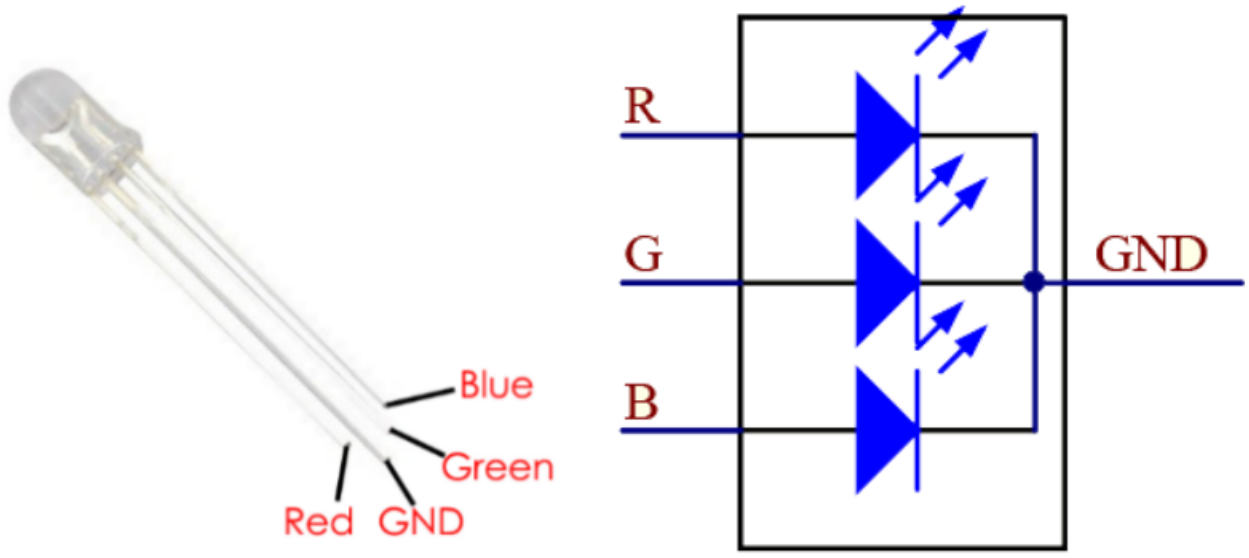
| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *RGB LED* | |

**回路図**

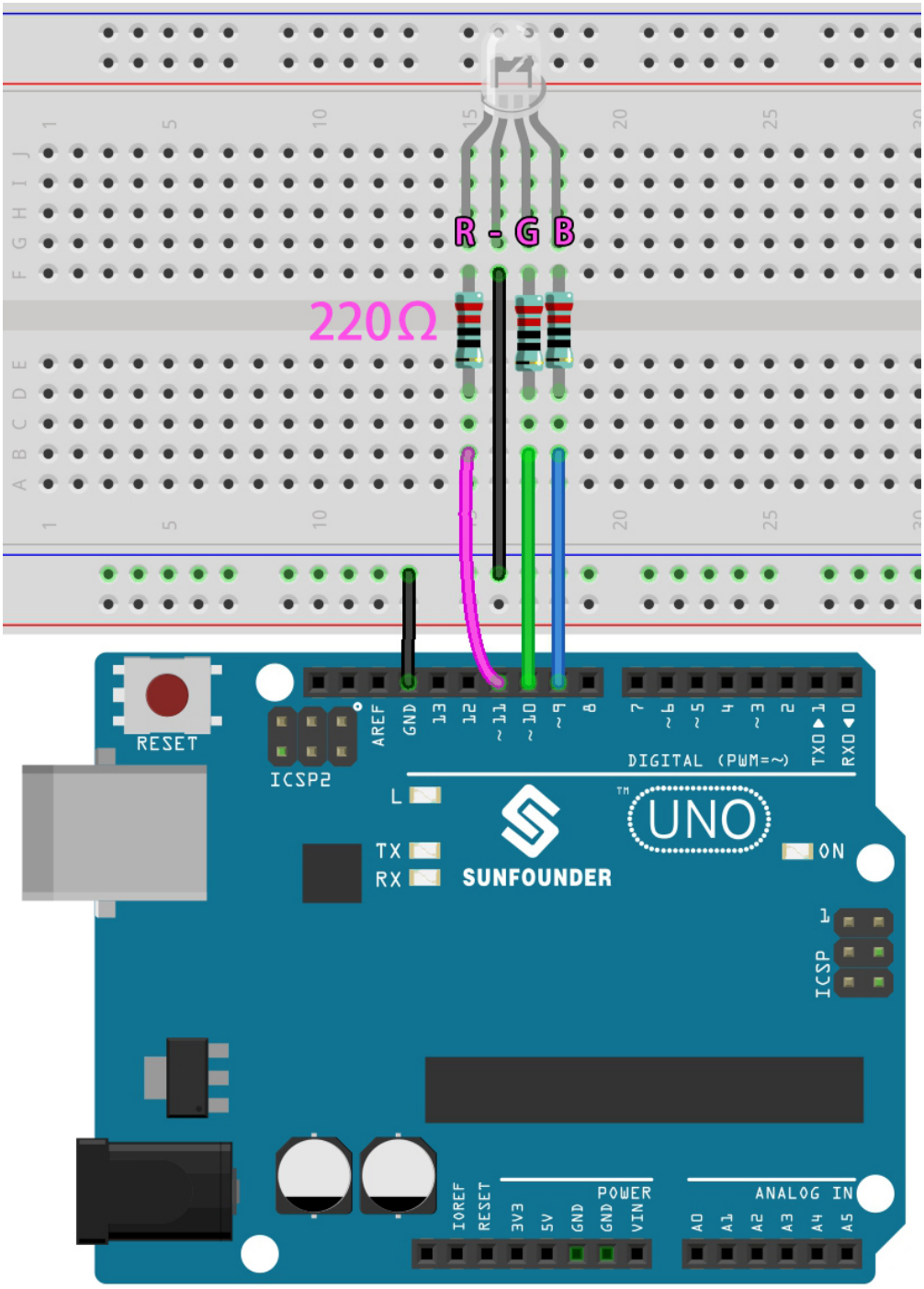



PWM ピンの 11、10、9 は、それぞれ RGB LED の赤、緑、青のピンを制御し、共通のカソードピンを GND に接続します。これにより、異なる PWM 値でこれらのピンの光を重ね合わせることで、RGB LED が特定の色を表示することができます。

**配線図**




RGB LED には 4 つのピンがあります：最も長いピンは共通のカソードピンで、通常は GND に接続されます。最も長いピンの隣の左側のピンは赤で、右側の 2 つのピンは緑と青です。




**コード**

ここでは、お気に入りの色を描画ソフトウェア（例：ペイント）で選択し、RGB LED で表示することができます。

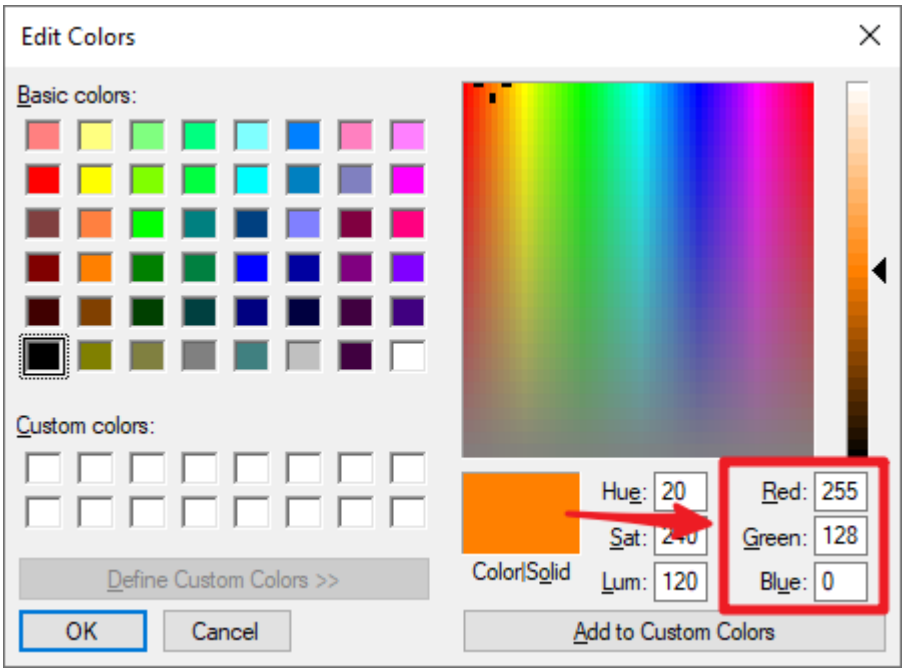
---

**注釈:**

- 3in1-kit\basic_project\2.analogWrite\2.2.colorful_light の パ ス の 下 の 2.2.

colorful_light.ino ファイルを開くことができます。

- または、このコードを **Arduino IDE** にコピーしてください。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。




color_set() に RGB 値を書き込むと、希望する色で RGB が点灯します。

**どのように動作するのか？**

この例では、RGB の三つのピンに値を割り当てるために使用される関数は、独立したサブ関数 color() にパッケージされています。

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

loop() 内で、RGB 値は color() 関数を呼び出して、RGB が異なる色を発することを実現する入力引数として動作します。

```
void loop()
{
    color(255, 0, 0); //  赤
```

(次のページに続く)

(前のページからの続き)

```
    delay(1000);
    color(0,255, 0); //  緑
    delay(1000);
    color(0, 0, 255); //  青
    delay(1000);
}
```

## 4.3 3. デジタルリード

センサーは実世界の情報をキャッチし、デジタルやアナログのピンを通じてメインボードに情報を伝達します。これにより、コンピュータは現状を知ることができます。

したがって、Arduino ボードは、ボタンや IR 障害回避モジュールのようなデジタルピンの値を読むことで、デジタルセンサーの動作状況を知ることができます。

以下は必要な関数です。

- `pinMode(pin, mode)`: 指定のピンを `INPUT` か `OUTPUT` として設定します。この場合は `INPUT` として設定する必要があります。

  **構文**

  pinMode(pin, mode)

  **パラメータ**

  - `pin`: モードを設定する Arduino のピン番号。
  - `mode`: INPUT、OUTPUT、または INPUT_PULLUP。

- `digitalRead(pin)`: 指定されたデジタルピンからの値（レベル状態）を読み取ります。

  **構文**

  digitalRead(pin)

  **パラメータ**

  - `pin`: 読み取りたい Arduino のピン番号

  **返り値**

  HIGH または LOW

**デジタルリードの例**

```
int ledPin = 13;  // LED はデジタルピン 13 に接続されています
int inPin = 7;    // プッシュボタンはデジタルピン 7 に接続されています
int val = 0;      // 読み取った値を保存する変数

void setup() {
   pinMode(ledPin, OUTPUT);  // デジタルピン 13 を出力として設定
   pinMode(inPin, INPUT);    // デジタルピン 7 を入力として設定
}

void loop() {
   val = digitalRead(inPin);   // 入力ピンを読み取り
   digitalWrite(ledPin, val);  // LED をボタンの値に設定
```

### 注意事項と警告

1. プルアップ & プルダウン。

   ピンがレベル信号を受け取っていない場合、 digitalRead() はランダムな、不確定な値を出力する可能性があります。入力ピンを既知の状態に指向させることで、プロジェクトの信頼性を高めることができます。デジタル入力ピンに並列してプルアップまたはプルダウン抵抗を接続することが通常必要です。

   プルアップ抵抗を直接接続するのではなく、コード内でピンモードを INPUT_PULLUP に設定することもできます。例： pinMode(pin,INPUT_PULLUP) 。この場合、ピンはソフトウェア経由で Atmega の内蔵プルアップ抵抗にアクセスし、プルアップ抵抗を接続するのと同じ効果が得られます。

2. Pin13 について。

   R3 ボード上のすべてのデジタルピン（1-13）は digitalRead() として使用できます。ただし、デジタルピン 13 は他のデジタルピンよりもデジタル入力として使用するのが難しいです。これは、LED と抵抗が接続されていて、ほとんどのボードにはんだ付けされているためです。内蔵の 20k プルアップ抵抗を有効にすると、オンボードの LED と系列抵抗が電圧レベルを低くするため、期待される 5V の代わりに約 1.7V になります。これは、常に LOW を返すことを意味します。もしピン 13 をデジタル入力として使用する必要がある場合は、 pinMode() を INPUT に設定し、外部のプルダウン抵抗を使用してください。

3. アナログピン。

   デジタルピンが不足している場合、アナログピン（A0-A5）もデジタルピンとして使用できます。pinMode(pin,mode) で INPUT に設定する必要があります。

### 関連コンポーネント

以下は関連コンポーネントです。クリックして使用方法を学ぶことができます。

### 4.3.1 3.0 シリアルモニタ

Arduino IDE には、コンピュータから Arduino ボードへのメッセージを送信（USB 経由）および Arduino からのメッセージを受信するためのシリアルモニタがあります。

このプロジェクトでは、Arduino ボードからデータを受信する方法を学びます。

---

**注釈:** Uno、Nano、Mini、Mega では、ピン 0 および 1 がコンピュータとの通信に使用されます。これらのピンに何かを接続すると、ボードへのアップロードが失敗する原因となる通信の妨害が発生する場合があります。

---

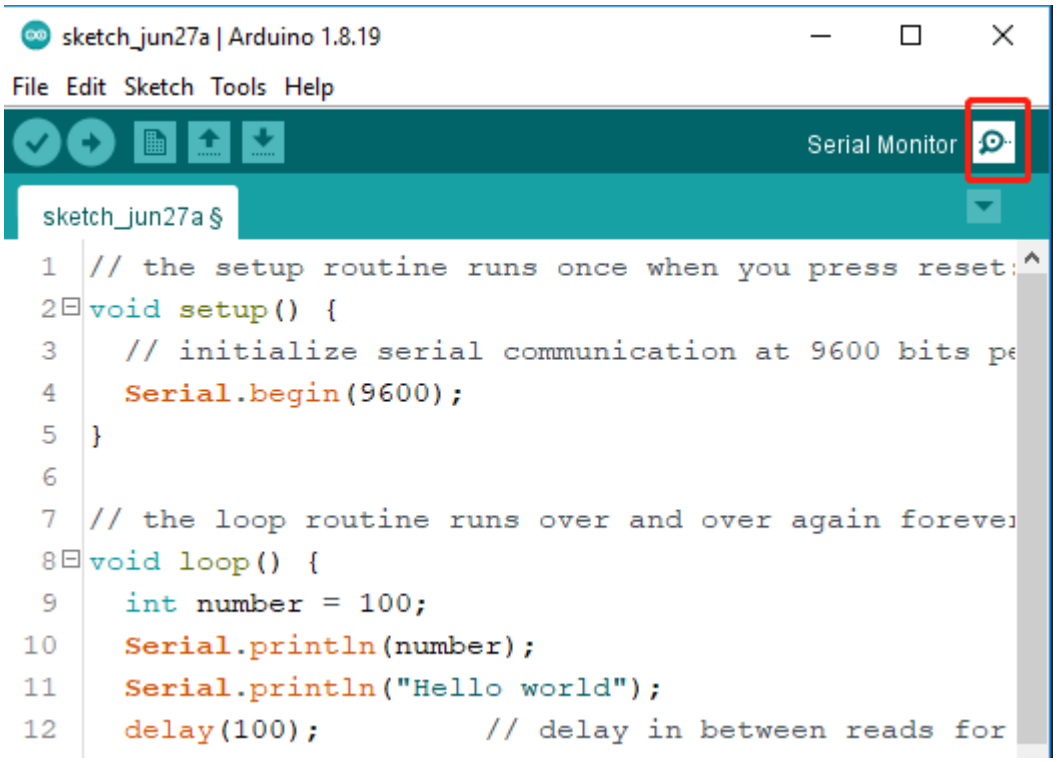
#### シリアルモニタの使用方法

1. Arduino IDE を開き、以下のコードを貼り付けます。

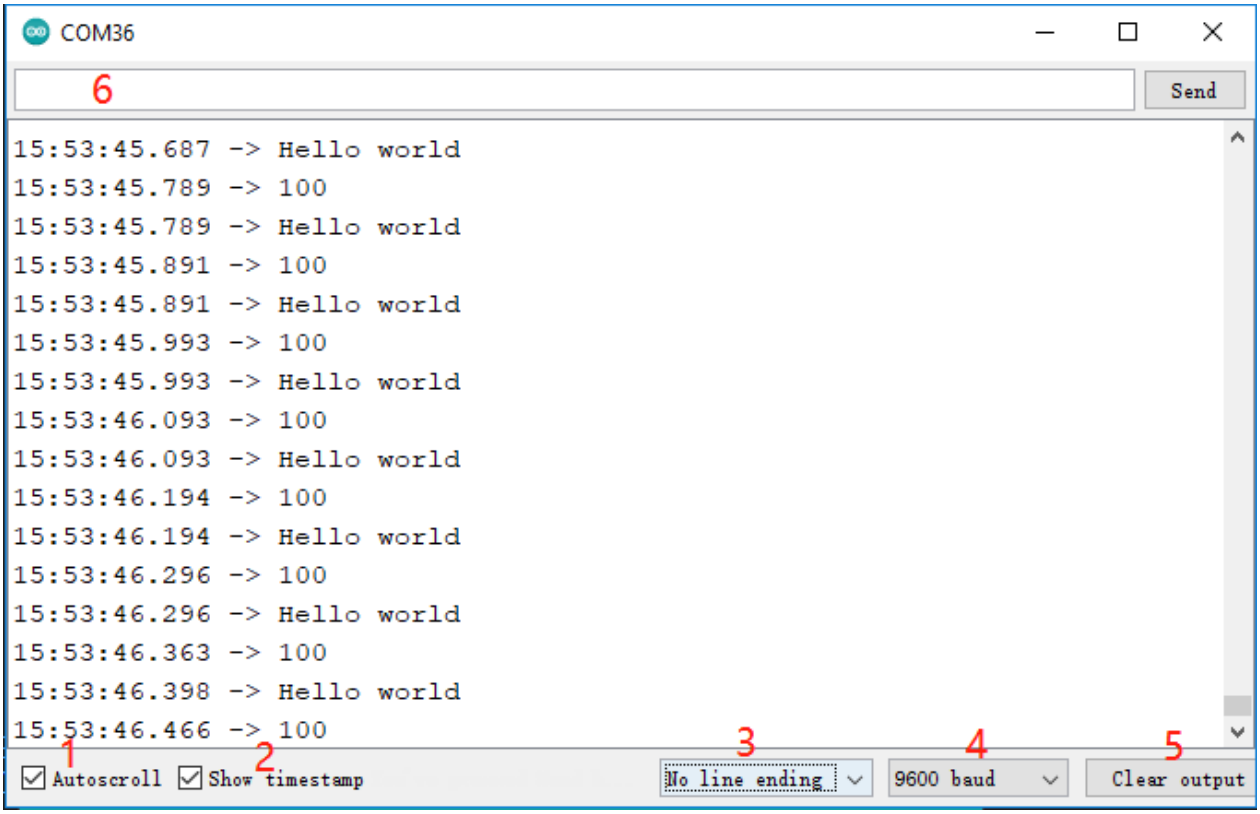
```
// setup ルーチンはリセットを押すと一度実行されます:
void setup() {
    // 9600 ビット毎秒でシリアル通信を初期化:
    Serial.begin(9600);
}

// loop ルーチンは永遠に繰り返し実行されます:
void loop() {
    int number = 100;
    Serial.println(number);
    Serial.println("Hello world");
    delay(100);          // 安定した読み取りのための遅延
}
```

   - Serial.begin(): シリアルデータ伝送のビット毎秒のデータレートを設定します。この場合は 9600 に設定されています。
   - Serial.println(): データをシリアルポートに ASCII テキストとして人間が読める形式で出力し、キャリッジリターン文字（ASCII 13 または 'r'）および改行文字（ASCII 10 または 'n'）に続きます。このコマンドは Serial.print() と同じ形式を取ります。

2. コードをアップロードするための正しいボードとポートを選択します。
3. ツールバーで、シリアルモニタを起動するための虫眼鏡アイコンをクリックします。




4. これがシリアルモニタです。




- **1**: 自動スクロールと非スクロールの選択オプション。

- **2**: シリアルモニタに表示されるデータの前にタイムスタンプを表示するオプション。
- **3**: 終了選択、Arduino に送信されるデータに追加される終了文字を選択します。選択肢には次のものがあります：
  - **No line Ending** は入力したものだけを送信します；
  - **Newline** は \n で、入力後に ASCII の新しい行コードを送信します；
  - **Carriage Return** は \r で、入力後に ASCII キャリッジリターン文字を送信します；
  - **Both NL & CR** は \r\n で、入力後にキャリッジリターンと新しい行の両方の文字を送信します。
- **4**: Arduino ボードと PC の間の通信速度を選択します。この値は `Serial.begin()` で設定した値と同じでなければなりません。
- **5**: 出力コンソール上の全テキストをクリアします。
- **6**: Arduino ボードへの文字送信のためのテキストボックスです。チュートリアルについては、 *5.12* **シリアルリード** を参照してください。

### 4.3.2 3.1 ボタンの値を読む

前のプロジェクトでは出力機能を使用しましたが、この章では入力機能を使用してボタンの値を読み取ります。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

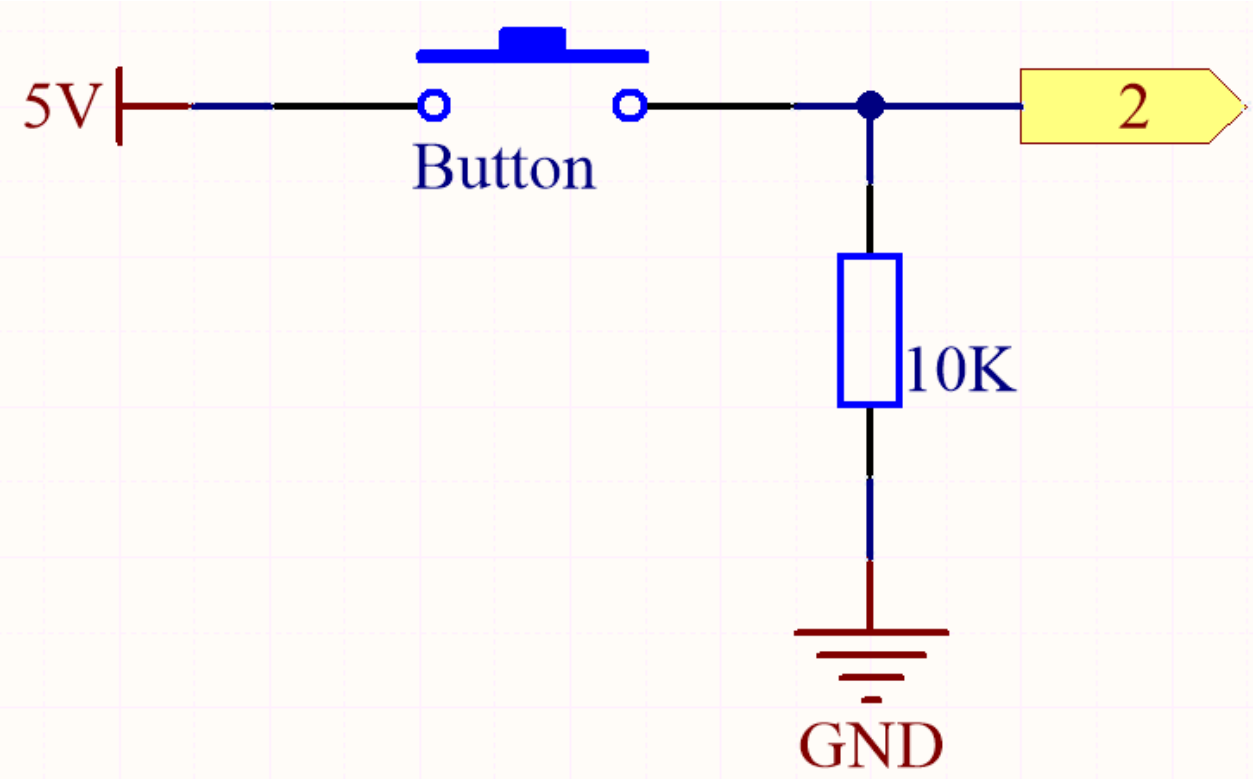
全体のキットを購入すると非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

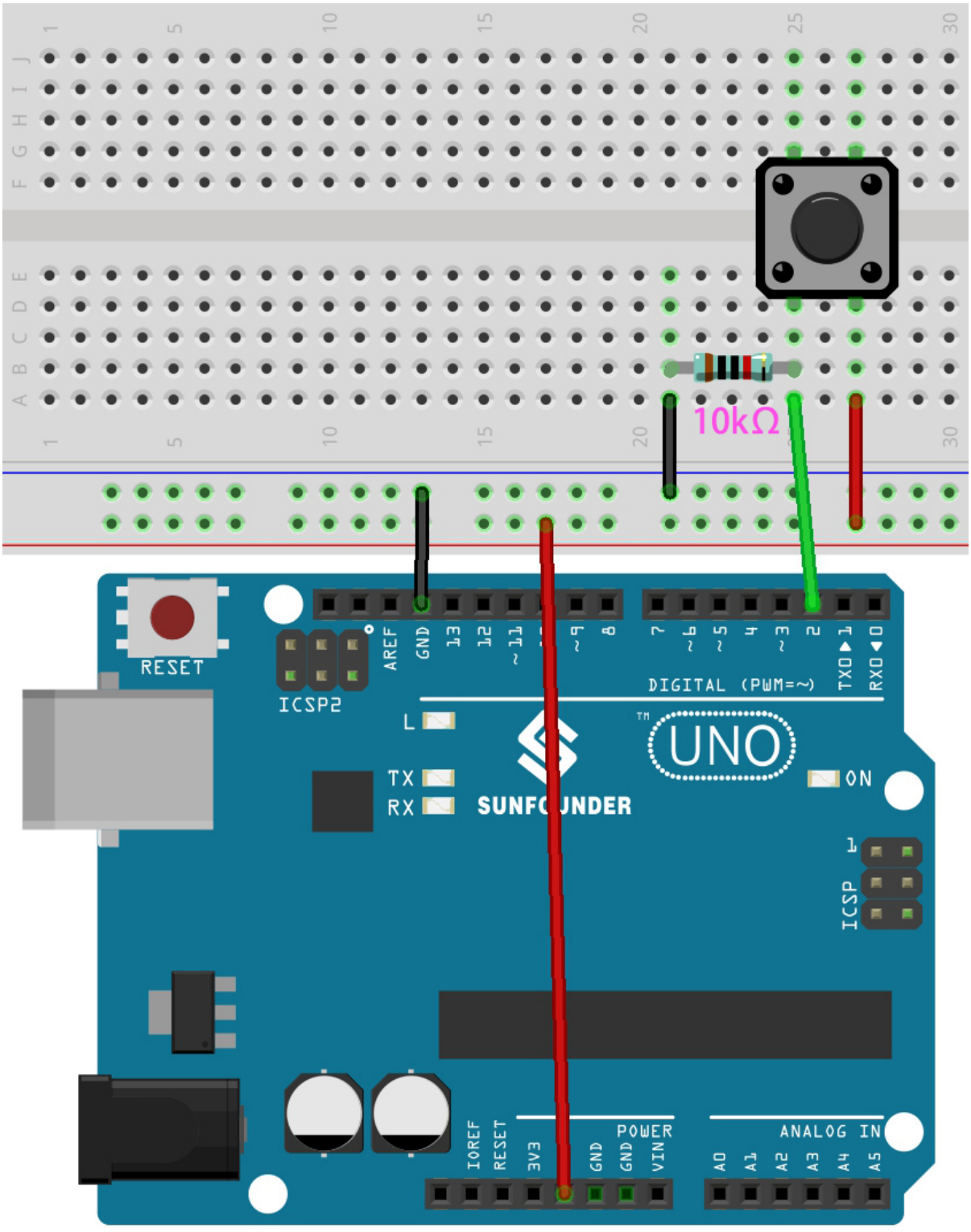
| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ボタン | |

**回路図**




ボタンのピンの一方は 5V に接続され、もう一方のピンはピン 2 に接続されているので、ボタンを押すと、ピン 2 はハイ状態になります。しかし、ボタンを押していないと、ピン 2 は未接続状態になり、ハイまたはローのどちらかになります。ボタンを押していないときに安定したローレベルを得るために、ピン 2 を 10K のプルダウン抵抗を介して GND に再接続する必要があります。

**配線図**




**コード**

**注釈:**

- 3in1-kit\basic_project\3.1.read_button_value のパスの下で 3.1.read_button_value.ino ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされた後、Arduino IDE の右上隅にある虫眼鏡アイコン（シリアルモニタ）をクリックします。


ボタンを押すと、シリアルモニタに"1"と表示されます。

### 4.3.3 3.2 磁気を感じる

最も一般的なリードスイッチは、スイッチが開いているときに小さな隙間で分離された、磁化可能な、柔軟な、金属製のリードのペアを含んでいます。

電磁石または永久磁石からの磁場は、リードがお互いを引き付ける原因となり、これにより電気回路が完成します。磁場が停止すると、リードのばねの力がそれらを分離させ、回路を開きます。

リードスイッチの一般的な使用例は、セキュリティアラームのためのドアや窓の開放を検出することです。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

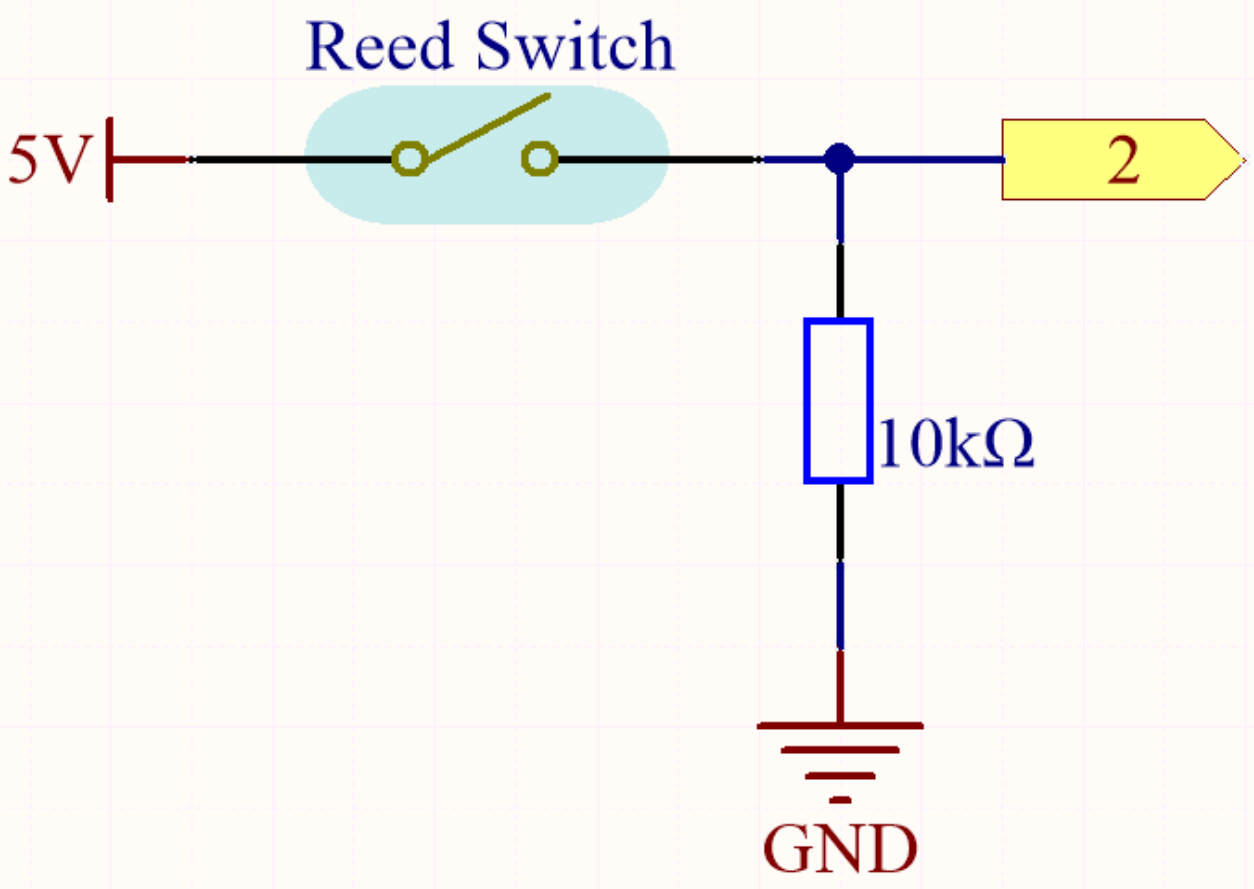
全体のキットを購入すると非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| リードスイッチ | - |

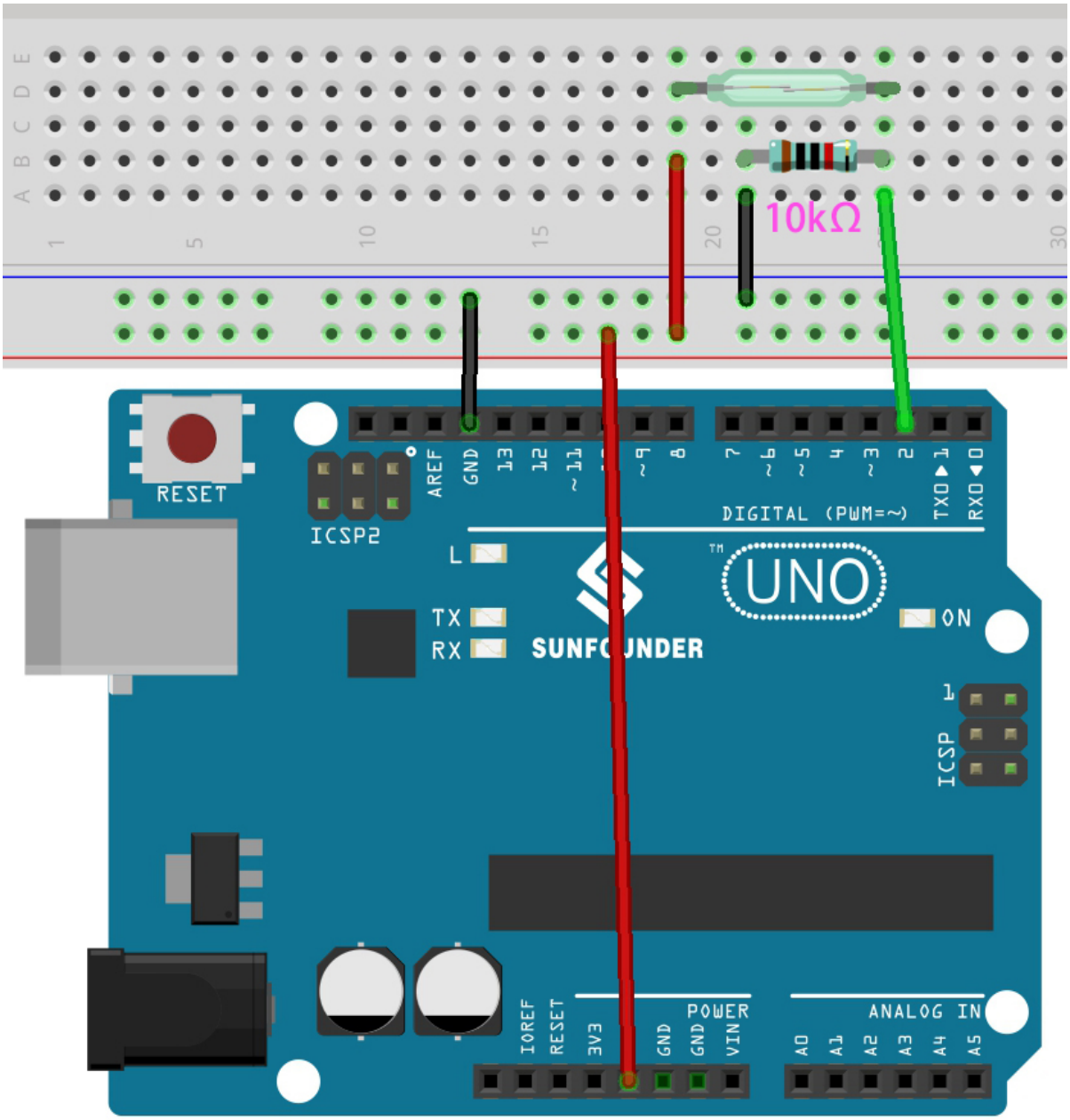
**回路図**




デフォルトでは、ピン 2 はローです。磁石がリードスイッチの近くにあると、ピン 2 はハイになります。

10K の抵抗の目的は、磁石が近くにないときにピン 2 を安定したローレベルに保つことです。

**配線図**




**コード**

---

**注釈:**

- `3in1-kit\basic_project\3.2.feel_the_magnetism` のパスの下で `3.2.feel_the_magnetism.ino` ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされた後、磁石がリードスイッチの近くにあると、シリアルモニタに 1 と表示されます。

### 4.3.4 3.3 障害物を検出する

このモジュールは、前方の障害物の存在を判断するために、車やロボットに一般的に取り付けられています。また、ハンドヘルドデバイスや水道蛇口など、幅広い用途で使用されています。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

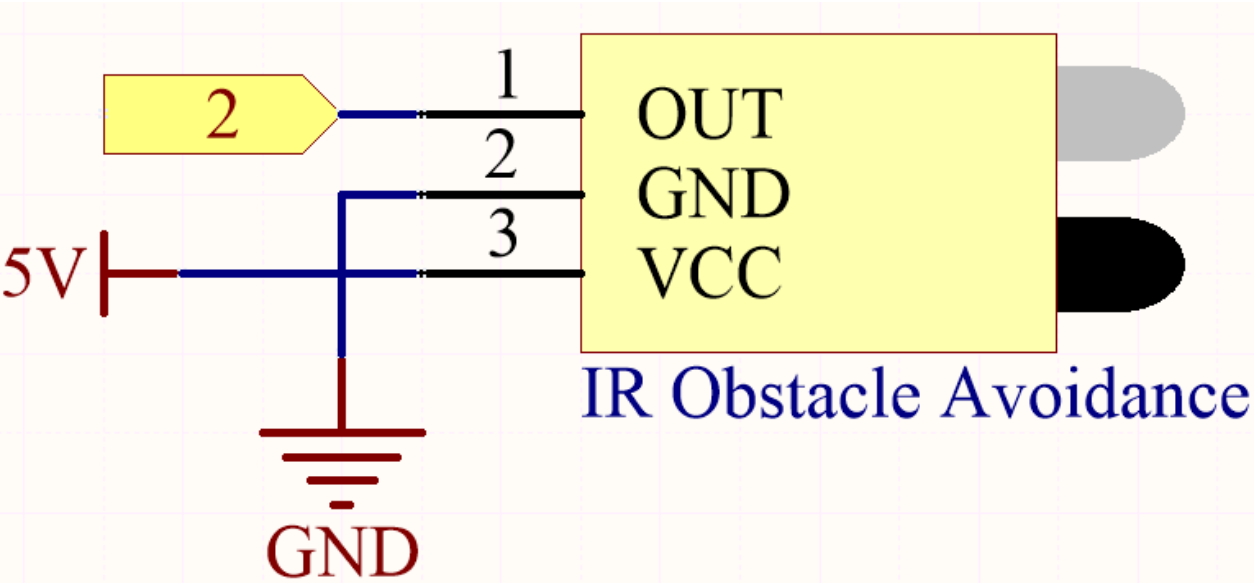
全体のキットを購入すると非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

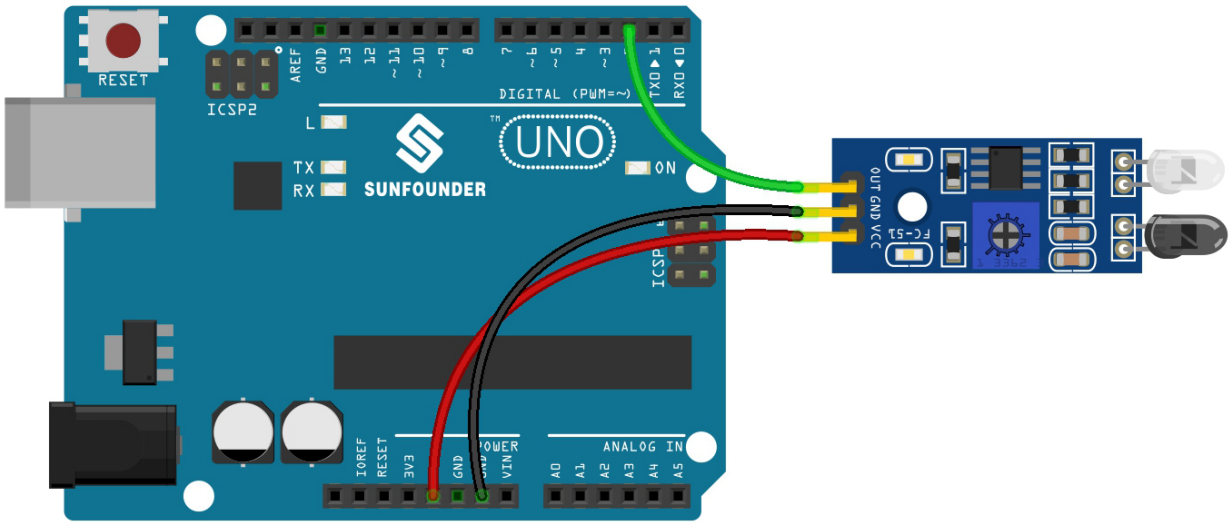
| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| 障害物回避モジュール | |

**回路図**




デジタルピン 2 は、IR 障害物回避モジュールの信号を読み取るために使用されます。IR センサーモジュールの VCC を 5V に、GND を GND に、OUT をデジタルピン 2 に接続します。

**配線図**




**コード**

---

**注釈:**

- `3in1-kit\basic_project\3.3.detect_the_obstacle` のパスの下で `3.3.detect_the_obstacle.ino` ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

IR 障害物回避モジュールが前方に何か障害物を検出すると、シリアルモニターに [0] が表示されます。それ以外の場合は、[1] が表示されます。

### 4.3.5 3.4 ラインを検出する

ライントラッキングモジュールは、地面に黒いエリア（電気テープで貼られた黒いラインなど）があるかどうかを検出するために使用されます。

モジュールの LED の一つが地面に適切な赤外線を放射し、黒い表面は光を吸収する能力が比較的強く、反射能力が弱いです。白い表面はその逆です。反射光を検出した場合、現在の地面が白いことを意味します。検出されない場合、黒いことを意味します。

それが動作原理です。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

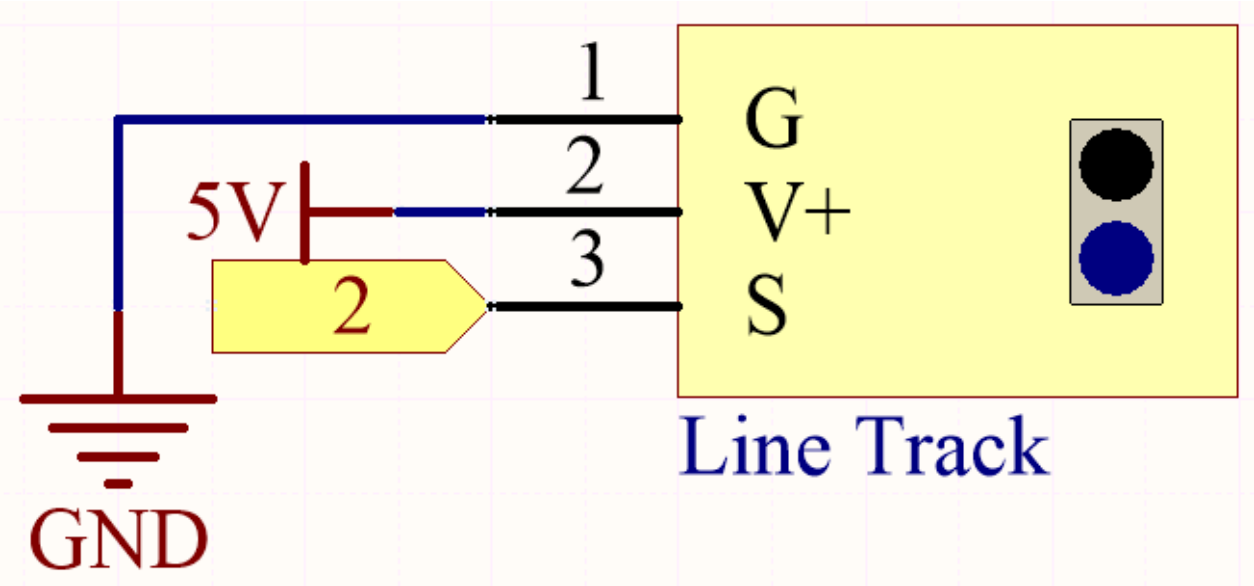
全体のキットを購入すると非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

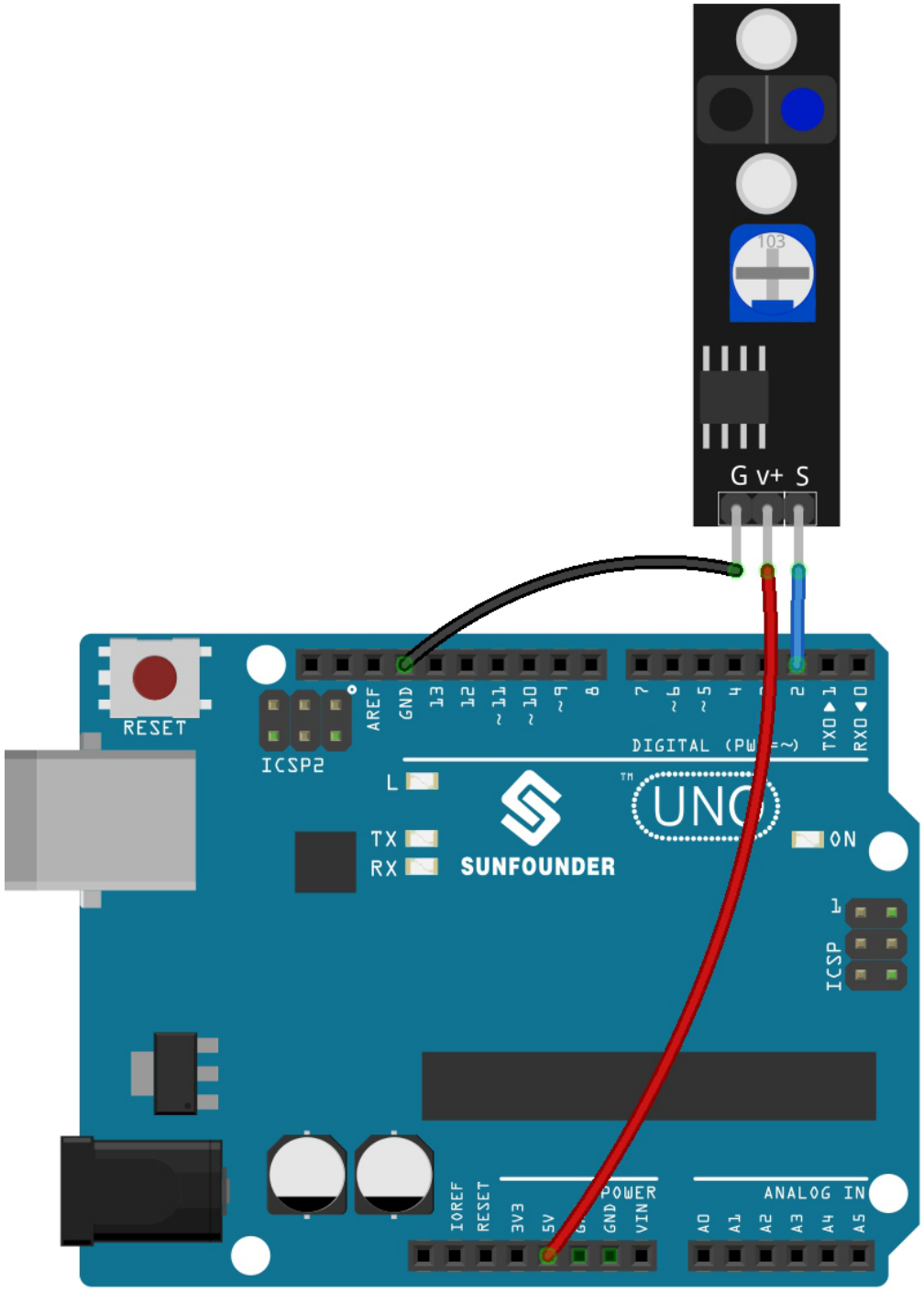
| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| ライン追跡モジュール | |

**回路図**




デジタルピン 2 は、ライントラックモジュールの信号を読み取るために使用されます。モジュールの VCC を 5V に接続し、GND を GND に、OUT をデジタルピン 2 に接続します。

**配線図**




**コード**

**注釈:**

- 3in1-kit\basic_project\3.4.detect_the_line のパスの下で 3.4.detect_the_line.ino ファイルを開くことができます。

- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

ライントラッキングモジュールが黒いラインを検出した場合、シリアルモニターに [1] が表示されます。それ以外の場合は、[0] が表示されます。

## 4.4 4. アナログ読み取り

Arduino はアナログピンを介して接続されたアナログセンサーの値を読み取ることができます。

R3 ボードには、多チャンネル、10 ビットのアナログ-デジタル変換器が含まれています。これは、入力電圧を 0 から動作電圧（5V または 3.3V）の間の整数値 0 から 1023 にマッピングすることを意味します。

アナログピンの値を読み取るには、 `analogRead(pin)` 関数が必要です。

- `analogRead(pin)`: 指定されたアナログピンの値を読み取ります。

  **構文**

  analogRead(pin)

  **パラメータ**

  - `pin`: 読み取りを行うアナログ入力ピンの名前（A0 から A5）。

  **戻り値**

  0-1023。データ型: int。

**アナログ読み取りの例**

```
int analogPin = A0; // アナログピン A0 に接続されたデバイス
                    // 外部はグラウンドと +5V につながる
int val = 0;  // 読み取り値を保存する変数

void setup() {
   Serial.begin(9600);           // シリアルのセットアップ
}

void loop() {
   val = analogRead(analogPin);  // 入力ピンを読み取る
   Serial.println(val);          // 値のデバッグ表示
}
```

**注意点と警告**

- アナログピンは A0-A5 です。
- アナログピンを呼び出す前に `pinMode()` を呼び出す必要はありませんが、ピンが以前 `OUTPUT` に設定されていた場合、`analogRead()` 関数は正しく動作しません。その場合、`pinMode()` を呼び出して `INPUT` に戻す必要があります。

**関連部品**

以下は関連する部品です。クリックして使用方法を学ぶことができます。

### 4.4.1 4.1 ノブを回す

ポテンショメータは 3 端子の抵抗成分であり、その抵抗値は一定の変化に応じて調整することができます。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

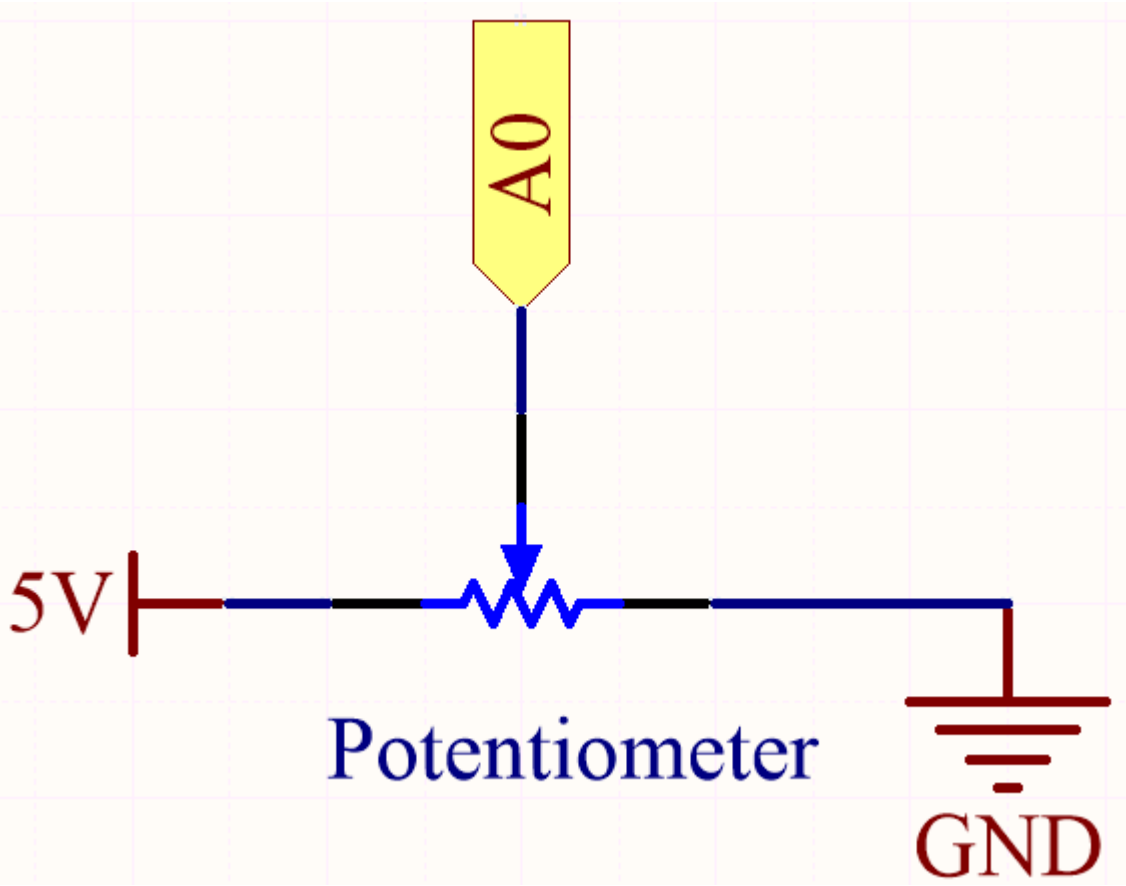
一式を購入するのは非常に便利です。購入リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

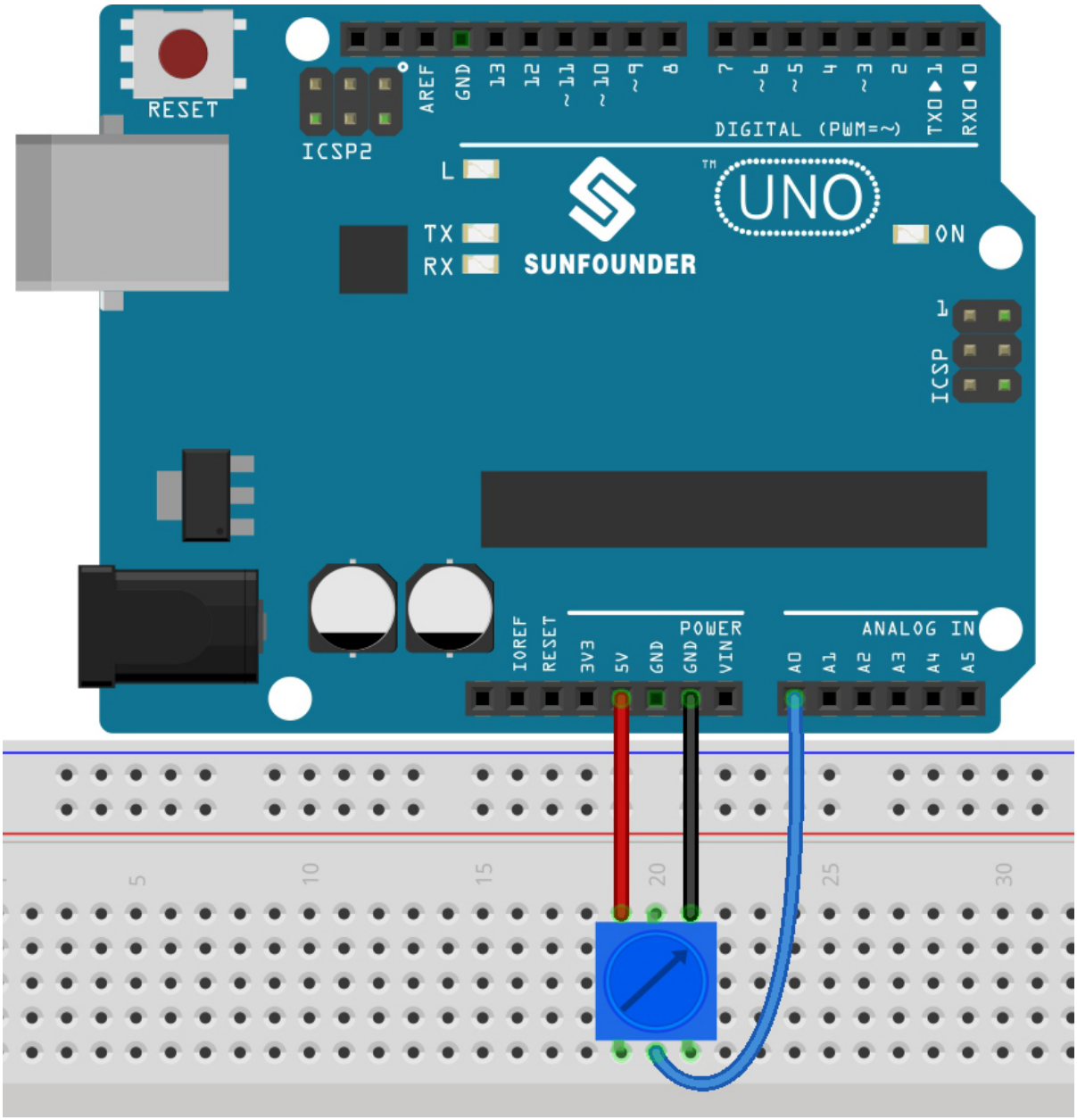
| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| **ポテンショメータ** | |

**回路図**




この例では、アナログピン（A0）を使用してポテンショメータの値を読み取ります。ポテンショメータの軸を回転させることで、これら 3 つのピンの間での抵抗の分配を変更し、中央のピンの電圧を変更することができます。中央と 5V に接続された外側のピンとの間の抵抗がゼロに近い（そして中央と他の外側のピンとの間の抵抗が 10k Ω に近い）場合、中央のピンの電圧は 5V に近くなります。逆の操作（中央と 5V に接続された外側のピンとの間の抵抗が 10k Ωに近い場合）は、中央のピンの電圧が 0V に近くなるようにします。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\4.1.turn_the_knob のパスの下にある 4.1.turn_the_knob.ino ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

ボードにコードをアップロードした後、シリアルモニターを開いてピンの読み取り値を確認することができます。ポテンショメータの軸を回転させると、シリアルモニターは「0」~「1023」の値を表示します。

### 4.4.2 4.2 光を感じる

フォトレジスタはアナログ入力の典型的なデバイスであり、ポテンショメータと非常に似た方法で使用されます。その抵抗値は光の強度に依存しており、照射される光が強ければ強いほど、その抵抗値は小さくなります。逆に、光が弱ければ抵抗値は増加します。

**必要な部品**

このプロジェクトには、以下のコンポーネントが必要です。

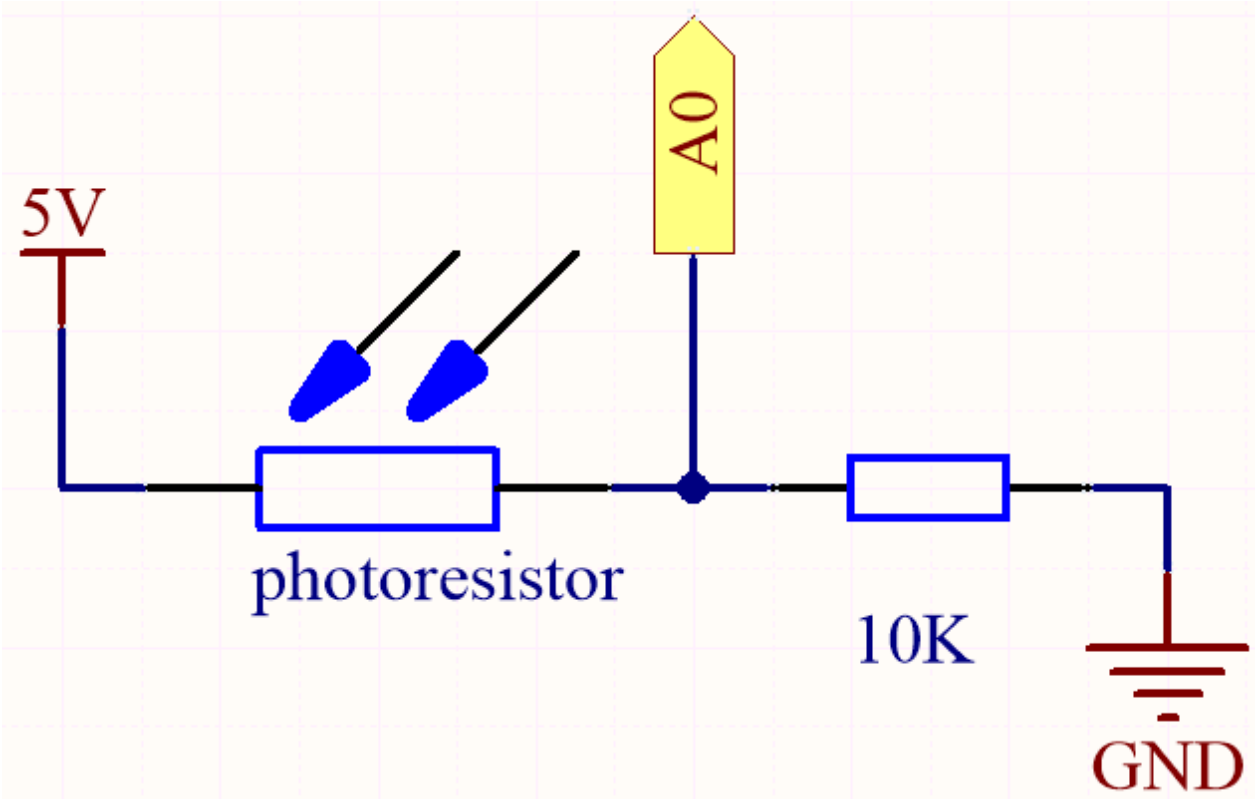
キット全体を購入すると非常に便利です。リンクは以下です：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| フォトレジスタ | |

**回路図**

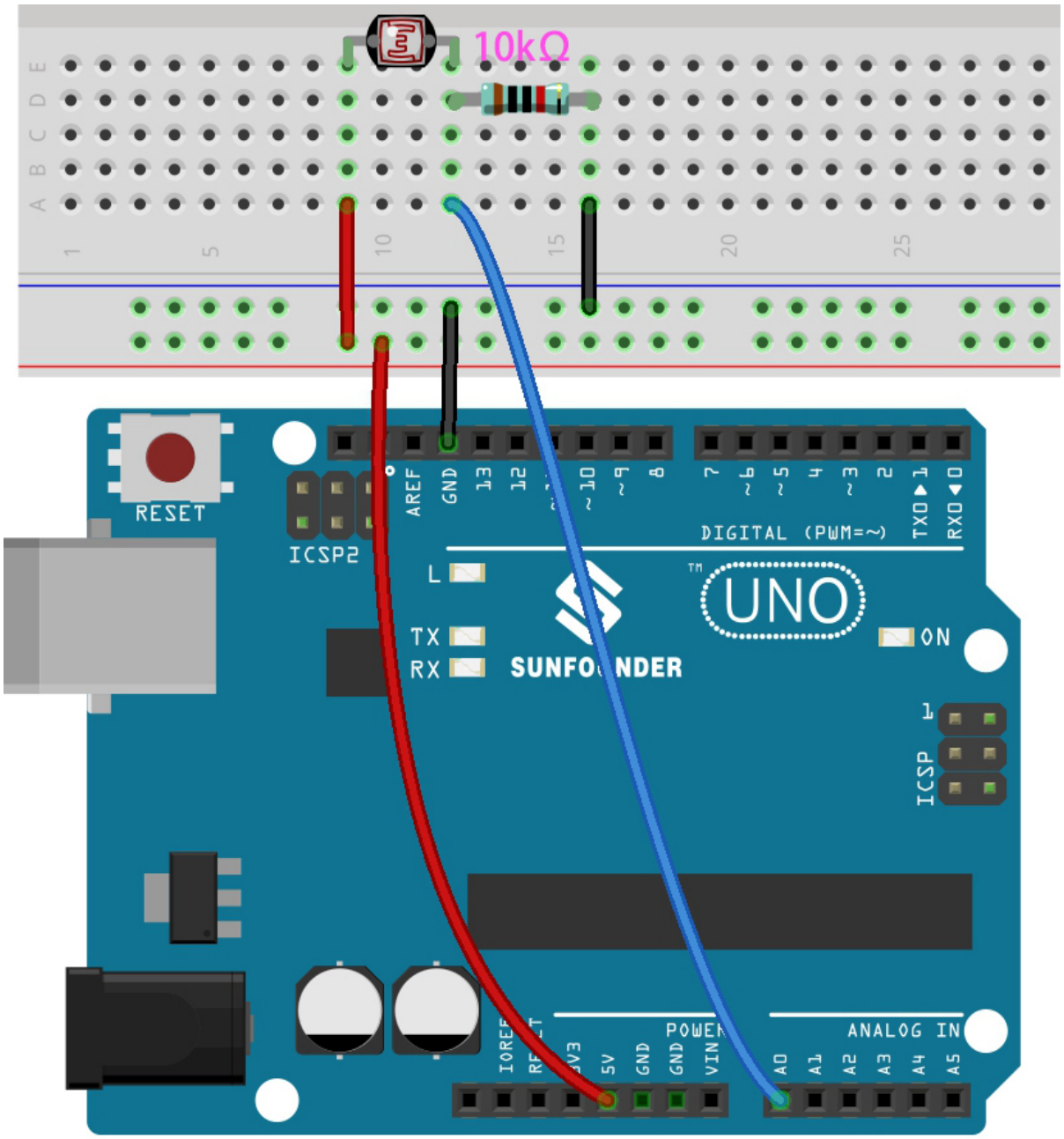



この回路では、10K の抵抗器とフォトレジスタが直列に接続されており、通過する電流は同じです。10K の抵抗器は保護として機能し、ピン A0 はフォトレジスタの電圧変換後の値を読み取ります。

光が増加すると、フォトレジスタの抵抗が減少し、その電圧が減少するため、ピン A0 からの値が増加します。光が十分強い場合、フォトレジスタの抵抗が 0 に近づき、ピン A0 の値が 1023 に近づくでしょう。この時、10K の抵抗器は保護役として機能し、5V と GND が短絡するのを防ぎます。

フォトレジスタを暗い場所に置くと、ピン A0 の値が減少します。十分に暗い場所では、フォトレジスタの抵抗は無限大となり、その電圧は 5V（10K の抵抗器は無視できる）に近づき、ピン A0 の値は 0 に近づくでしょう。

**配線図**




**コード**

**注釈:**

- 3in1-kit\basic_project\4.2.feel_the_light のパスの下にある 4.2.feel_the_light.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされると、シリアルモニタにフォトレジスタの値が表示されます。周囲の明るさが強いほど、シリアルモニタに表示される値が大きくなります。

### 4.4.3 4.3 ジョイスティックのトグル

ジョイスティックは、定期的にビデオゲームをプレイする人なら誰もが非常によく知っているはずです。通常、キャラクターの移動や画面の回転に使用されます。

私たちの動きはジョイスティックによって読み取られ、非常にシンプルな原理で動作します。これは、互いに直交する 2 つのポテンショメーターで構成されています。これらの 2 つのポテンショメーターは、ジョイスティックの垂直および水平の両方の方向でのアナログ値を測定し、直交座標系での値（x,y）を生成します。

このキットには、デジタル入力を備えたジョイスティックも含まれています。ジョイスティックが押されると、それがアクティブになります。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

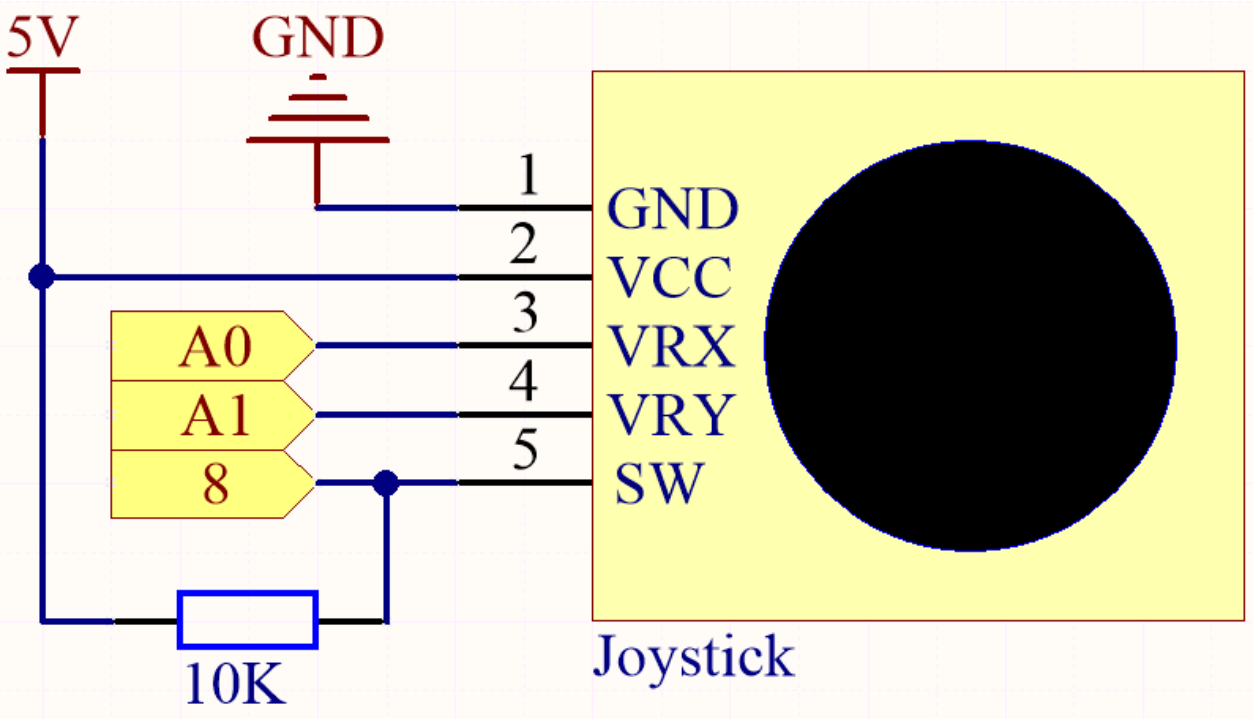
一式を購入することは確かに便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

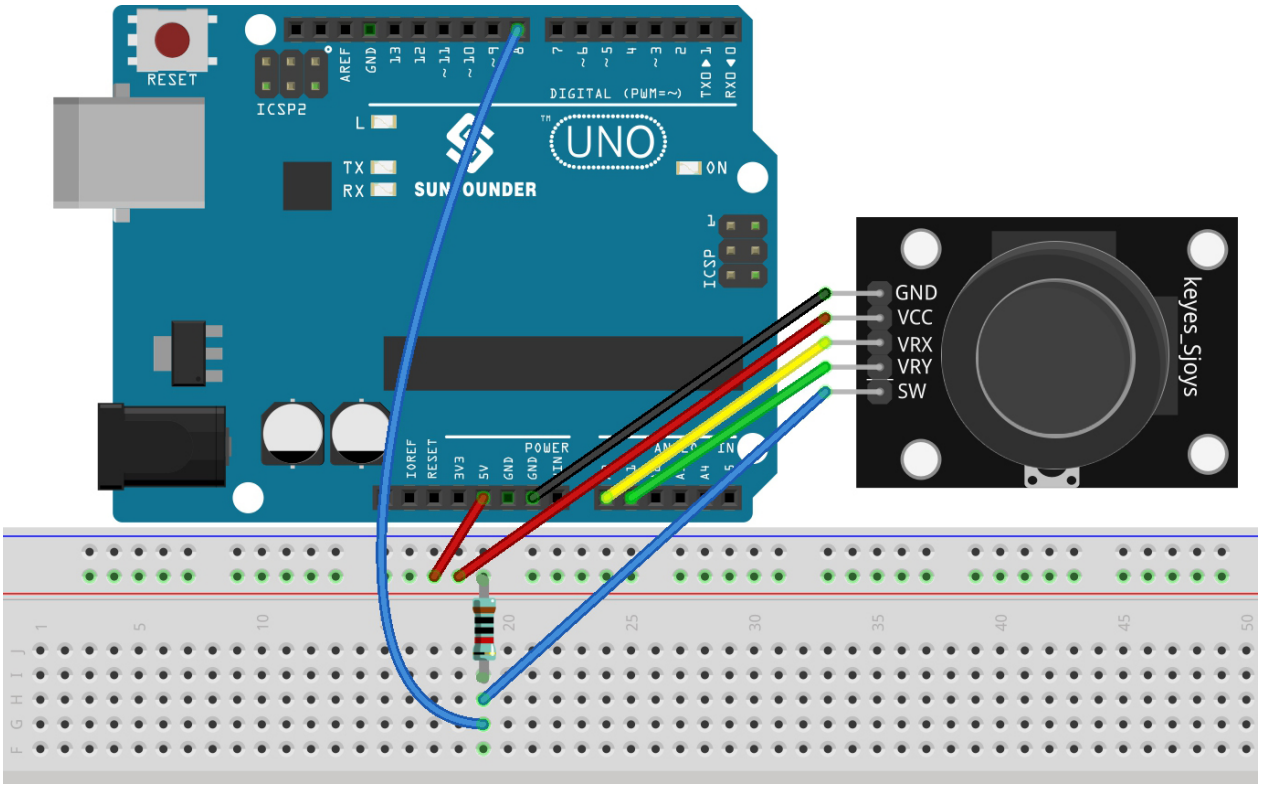
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ジョイスティックモジュール | - |

**回路図**




**注釈:** SW ピンは 10K のプルアップ抵抗器に接続されています。その理由は、ジョイスティックが押されていないときに SW ピン（Z 軸）で安定した高レベルを取得するためです。そうでなければ、SW は一時停止状態となり、出力値は 0/1 の間で変動するかもしれません。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\4.3.toggle_the_joystick のパスの下で 4.3.toggle_the_joystick.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードのアップロードが成功した後、シリアルモニタを開いてジョイスティックの x、y、z の値を確認してください。

- x 軸と y 軸の値は 0 から 1023 までのアナログ値です。
- Z 軸は 1 または 0 のステータスを持つデジタル値です（押されると 0 になります）。

### 4.4.4 4.4 土壌の湿度を測定する

農業の世界では、作物自体が土壌中の無機元素を直接取得することはできません。土壌の水は、これらの無機元素を溶解するための溶媒として作用します。

作物は根系を通じて土壌の湿度を吸収し、栄養を取得し、成長を促進します。

作物の成長と発展の過程で、土壌温度に対する要求も異なります。したがって、土壌湿度センサーが必要となります。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

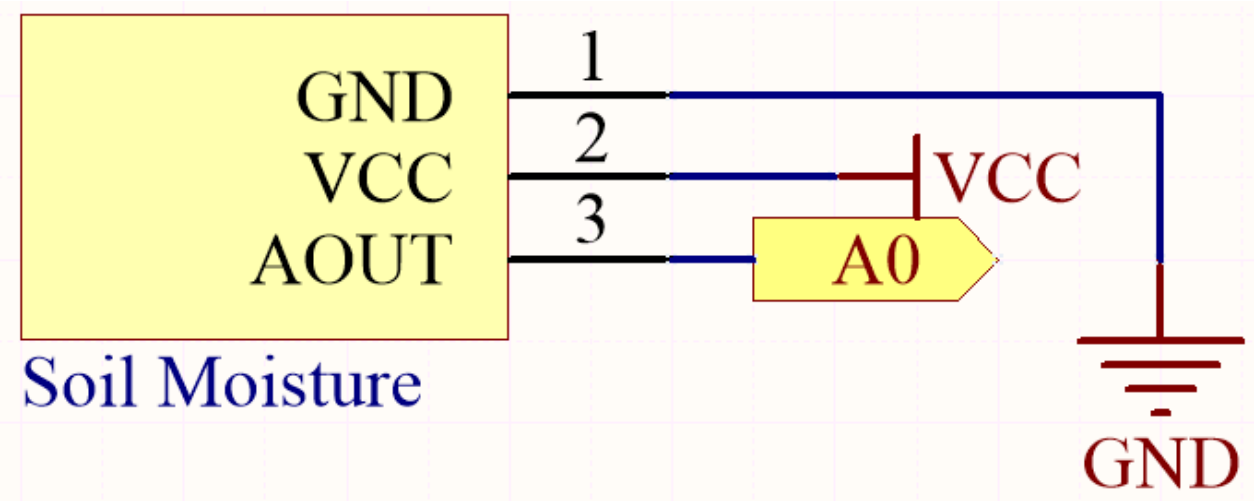
一式を購入することは確かに便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

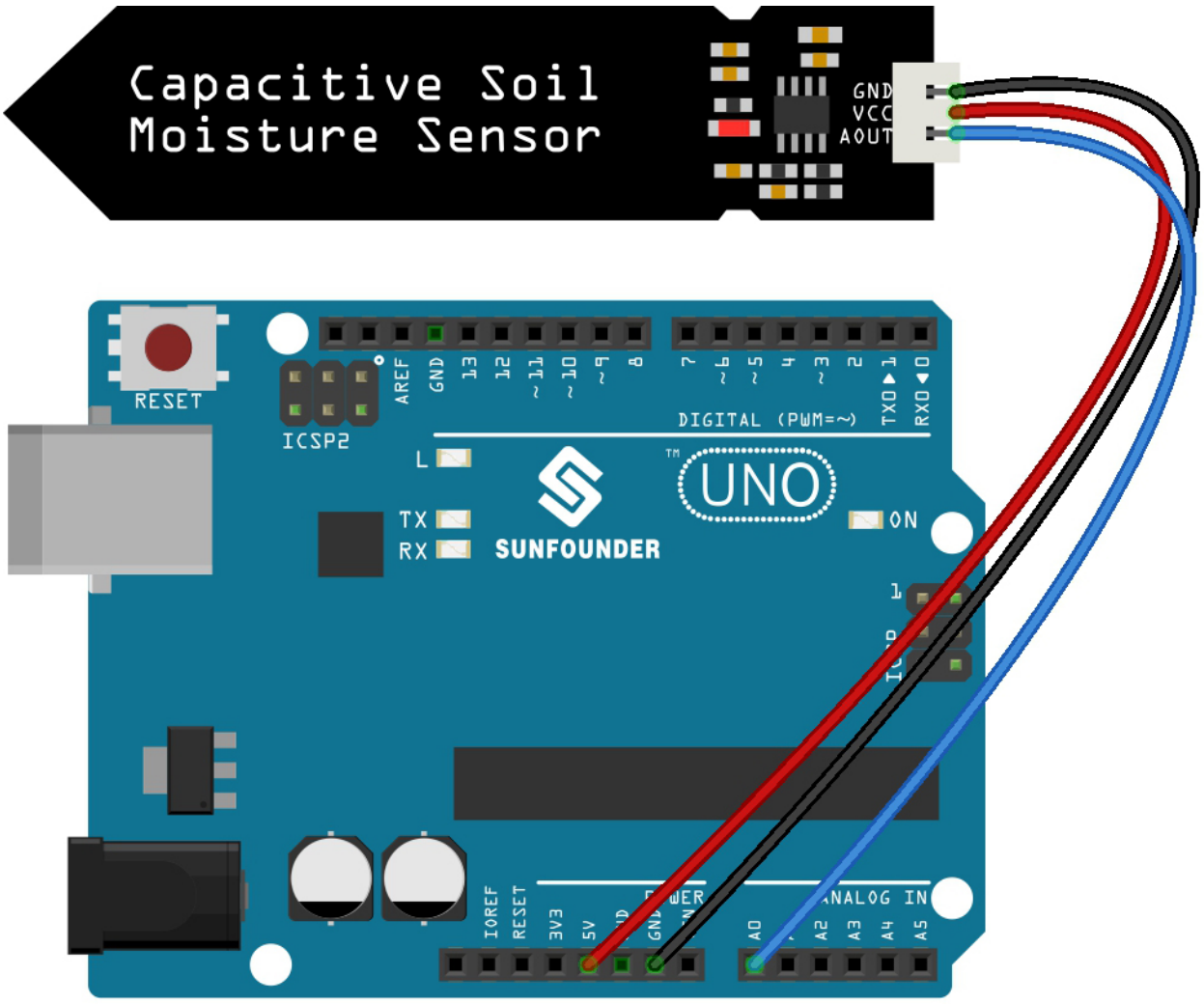
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| 土壌湿度モジュール | |

**回路図**


GND
VCC
AOUT
1
2
3
VCC
A0
Soil Moisture
GND

**配線図**


Capacitive Soil
Moisture Sensor
GND
VCC
AOUT
RESET
ICSP2
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
IOREF
RESET
3V3
5V
GND
ANALOG IN
A0
A1
A2
A3
A4
A5

**コード**

**注釈:**

- 3in1-kit\basic_project\4.4.measure_soil_moisture のパスの下で 4.4.measure_soil_moisture.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

コードのアップロードが成功したら、シリアルモニタは土壌の湿度値を表示します。

モジュールを土壌に挿入し、水をやると、土壌湿度センサの値は小さくなります。

### 4.4.5 4.5 温度計

温度計は、温度または温度勾配（物体の暖かさまたは寒さの度合い）を測定する装置です。温度計には 2 つの重要な要素があります：(1) 温度が変わると何らかの変化が生じる温度センサー（例：水銀温度計の球部や赤外線温度計の焼結センサー）および (2) この変化を数値に変換する手段（例：水銀温度計にマークされている可視スケールや赤外線モデルのデジタル表示)。温度計は、工業や技術でのプロセスの監視、気象学、医学、科学研究で幅広く使用されています。

サーミスタは、温度に強く依存する抵抗を持つ温度センサの一種で、2 つのタイプがあります：Negative Temperature Coefficient（NTC）および Positive Temperature Coefficient（PTC)、または NTC および PTC として知られています。 PTC サーミスタの抵抗は温度とともに増加し、NTC の状態は前者とは逆です。

この実験では、 **NTC サーミスタ** を使用して温度計を作成します。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

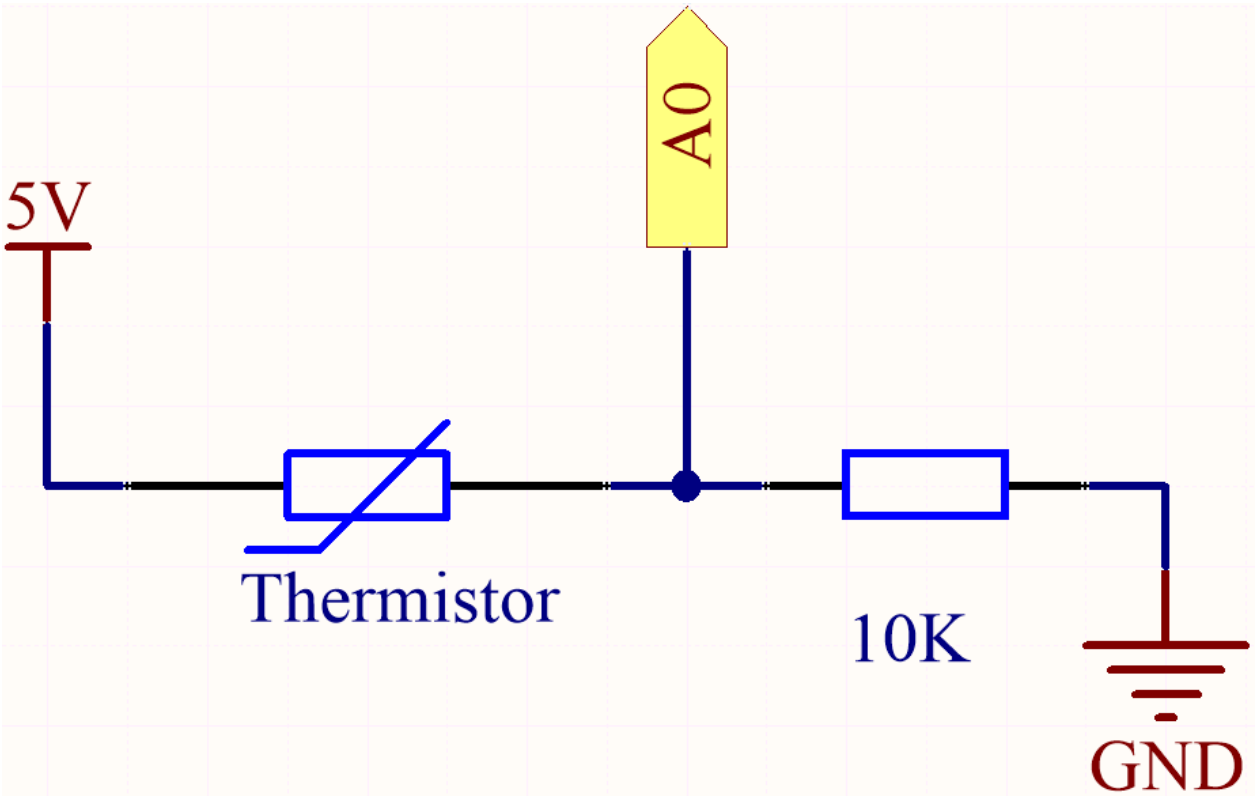
一式を購入することは非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| サーミスタ | |

**回路図**




各サーミスタには通常の抵抗があります。ここでは、それは 10k オームで、25 度セルシウスで測定されます。

温度が上昇すると、サーミスタの抵抗は減少します。その後、電圧データは A/D アダプタによってデジタル量に変換されます。

セルシウス度または華氏での温度は、プログラミングを介して出力されます。

抵抗と温度の間の関係は以下の通りです：

**RT =RN expB(1/TK – 1/TN)**

- **RT** は、温度が **TK** の場合の NTC サーミスタの抵抗です。
- **RN** は、定格温度 TN 下の NTC サーミスタの抵抗です。ここでは、RN の数値は 10k です。
- **TK** は、ケルビン温度で、単位は K です。ここでは、 **TK** の数値は `273.15` + **セルシウス度** です。
- **TN** は、定格ケルビン温度で、単位も K です。ここでは、TN の数値は `273.15+25` です。
- **B(beta)** は、NTC サーミスタの材料定数であり、熱感度指数とも呼ばれ、数値は `3950` です。
- **exp** は指数の略であり、基数 e は自然数で、おおよそ 2.7 に等しいです。

この式 `TK=1/(ln(RT/RN)/B+1/TN)` を変換して、ケルビン温度から 273.15 を引くとセルシウス度になります。

この関係は経験的な式です。温度と抵抗が有効範囲内の場合にのみ正確です。

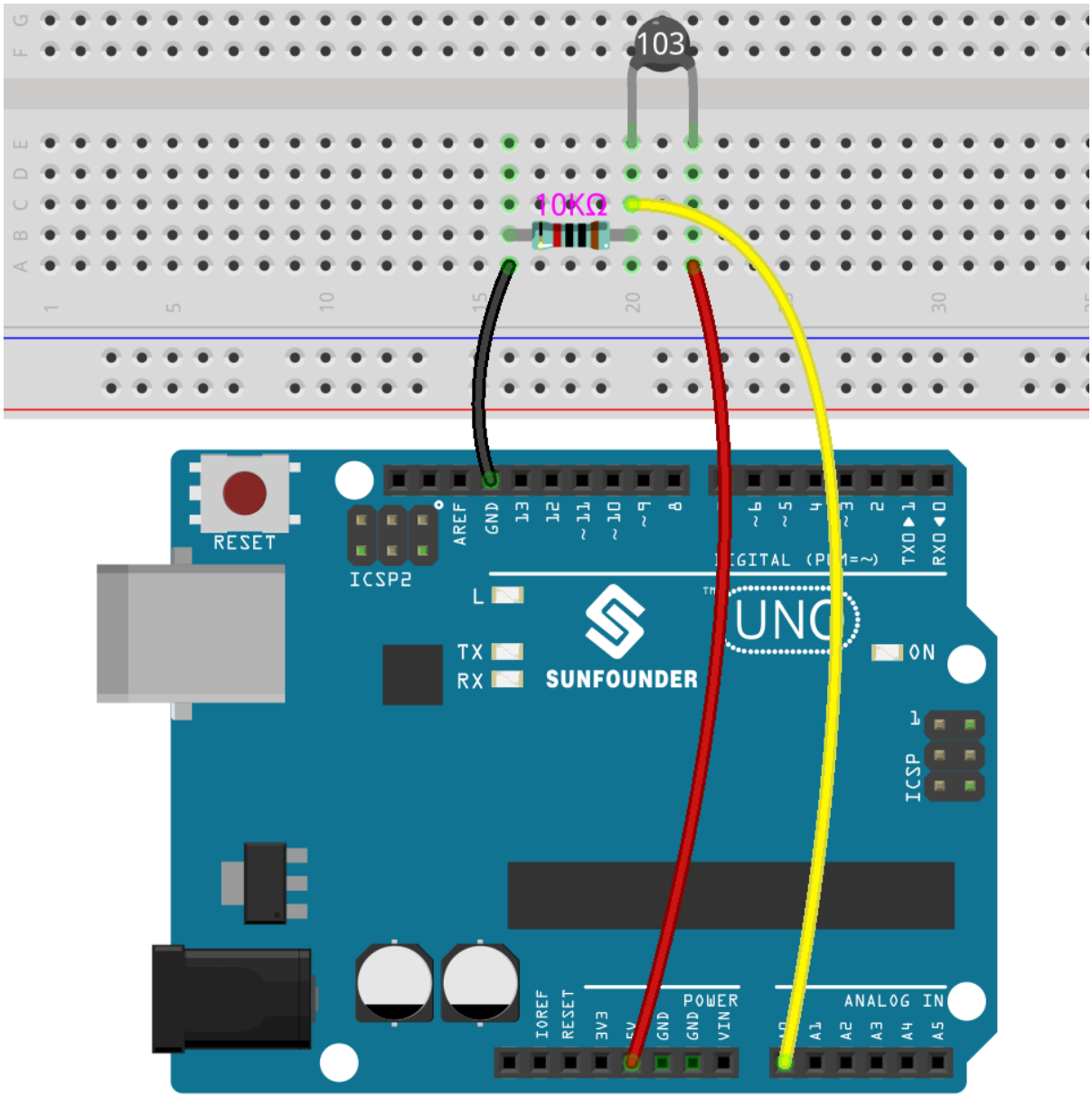
**配線図**

---

**注釈:**

- サーミスタは黒または緑で、103 とマークされています。

---




**コード**

---

**注釈:**

- `euler-kit/arduino/4.5_thermometer` のパスの下の `4.5_thermometer.ino` ファイルを開くことができます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

アップロードボタンをクリックする前に、Raspberry Pi Pico ボードと正しいポートを選択することを忘れないでく

ださい。

コードが正常にアップロードされると、シリアルモニタはセルシウス度と華氏温度を出力します。

## 4.5 5. さらなる文法

この章では、多くのプログラムが現実とどのように対話するのかの基本的なロジックを示す例をいくつか紹介します。これにより、Arduino のプログラミングに慣れる手助けとなります。創造的なアイデアが頭に浮かんだとき、プログラミングはもはやあなたにとっての課題ではなくなります。

### 4.5.1 5.1 If else

通常、私たちは条件判断を使って、最も基本的な現実のインタラクションプロジェクトを完成させます。ここでは、リードスイッチと LED を使用して、このロジックを示すドア検出システムを構築します。

ドアの片側に磁石を取り付け、ドアの反対側にリードスイッチ（回路付き）を取り付けます。ドアが閉まっていると、磁石がリードスイッチに近づき、それをオンにします。

**必要な部品**

このプロジェクトには、以下のコンポーネントが必要です。

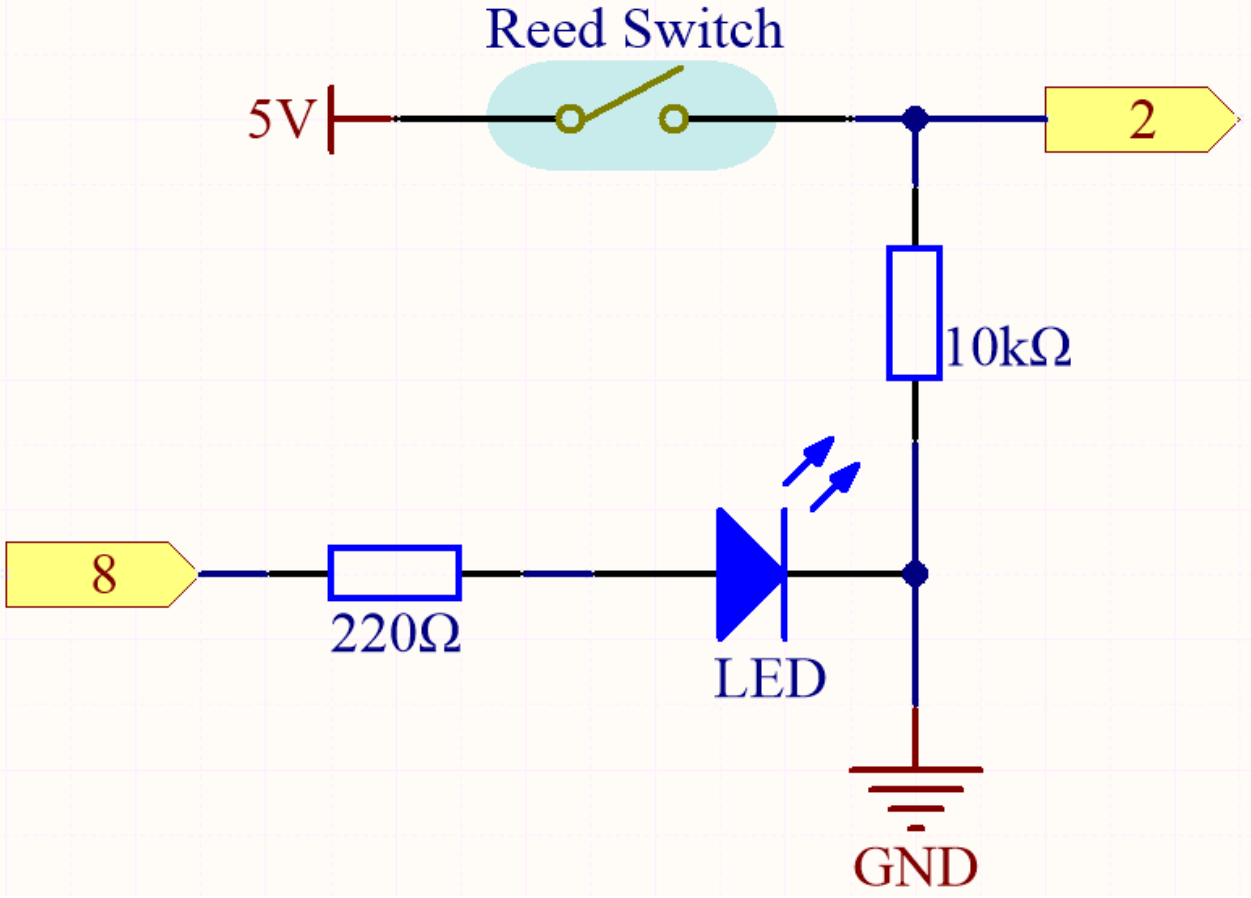
キット全体を購入するのは間違いなく便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

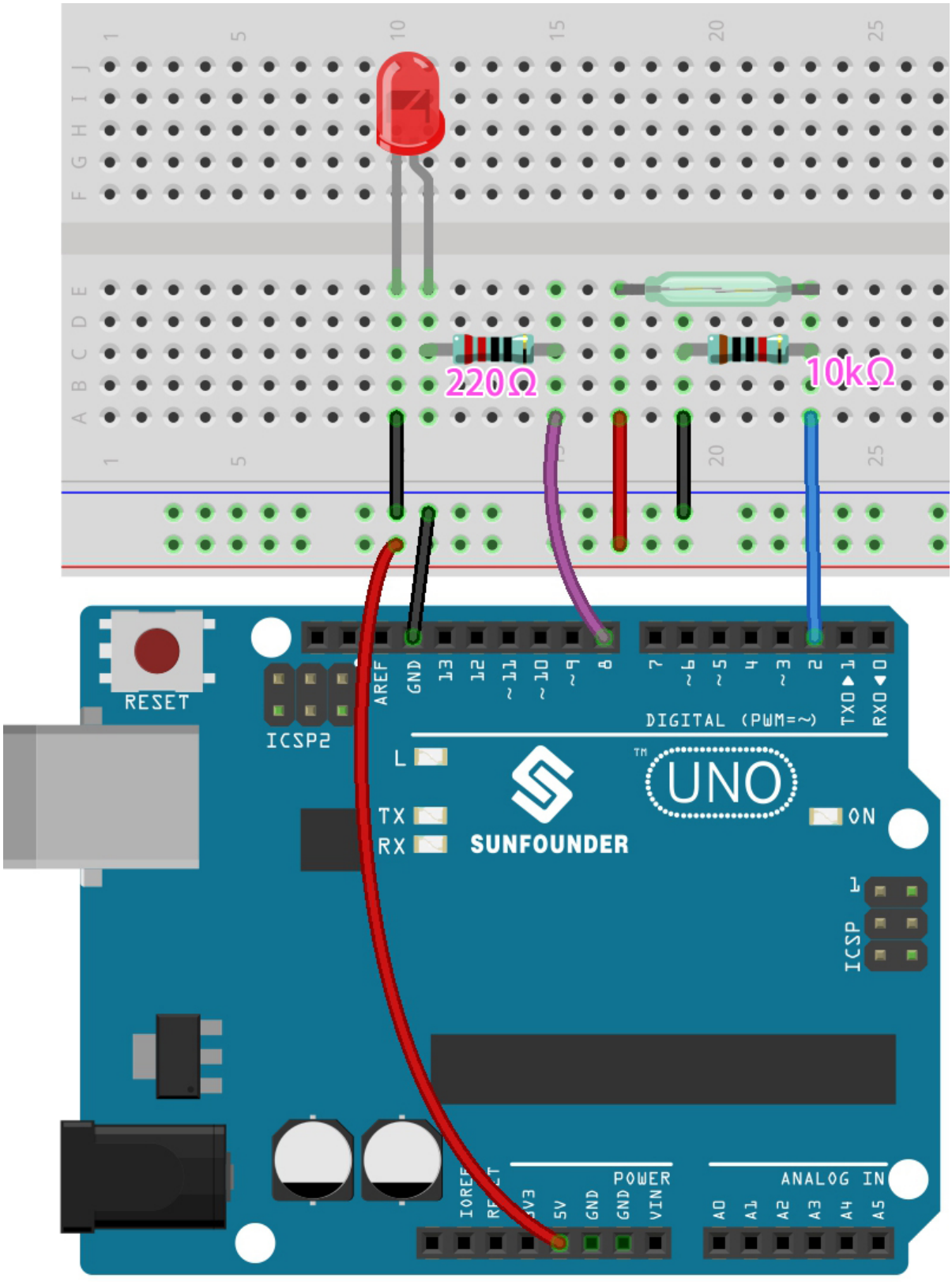
以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| リードスイッチ | - |

**回路図**




**配線図**


220Ω
10kΩ
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5

コード

**注釈:**

- `3in1-kit\basic_project\5.1.if_else` のパスの下で `5.1.if_else.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされた後、ドアを閉めないと、LED が点灯して、ドアを閉めるように促します。

ちなみに、ドアが閉まっているときに LED を点灯させる逆の効果が必要な場合、 `if` の条件を変更するだけです。

- if else

  `if else` は、複数のテストをグループ化できることで、基本的な if 文よりもコードの流れをより大きく制御することができます。

### 4.5.2 5.2 しきい値

多くのプロジェクトで、次のような要求に直面します。「xxx があるレベルに達したとき、次に...」

例えば、スマートホームでは、光の強度が 50Lux 未満の場合、照明を点けます；もう一つの例は、コンピュータのマザーボードで、CPU の動作温度が 65 ℃以上の場合、ファンを起動します。

これらの要求で、"しきい値"というキーワードが反映されます。

しきい値の数値を調整することで、回路の動作を個々のニーズに合わせて最適化できます。例えば、私が明るい生活環境を好む場合、スマートホームの自動照明のしきい値を 80Lux に上げることができます。もう一つの例は、私のスタジオの換気環境があまり良くなく、放熱の要求が高い場合、自動ファンの開始のしきい値を 50 ℃に調整できます。

ここでは、土壌湿度センサーと 2 つの LED を使用して、鉢のモニターを作ります。土壌が乾燥していると赤い LED が点灯し、土壌が十分に湿っていると緑の LED が点灯します。土壌の乾燥と湿潤を判断するためのしきい値を手動で調整する必要があります。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

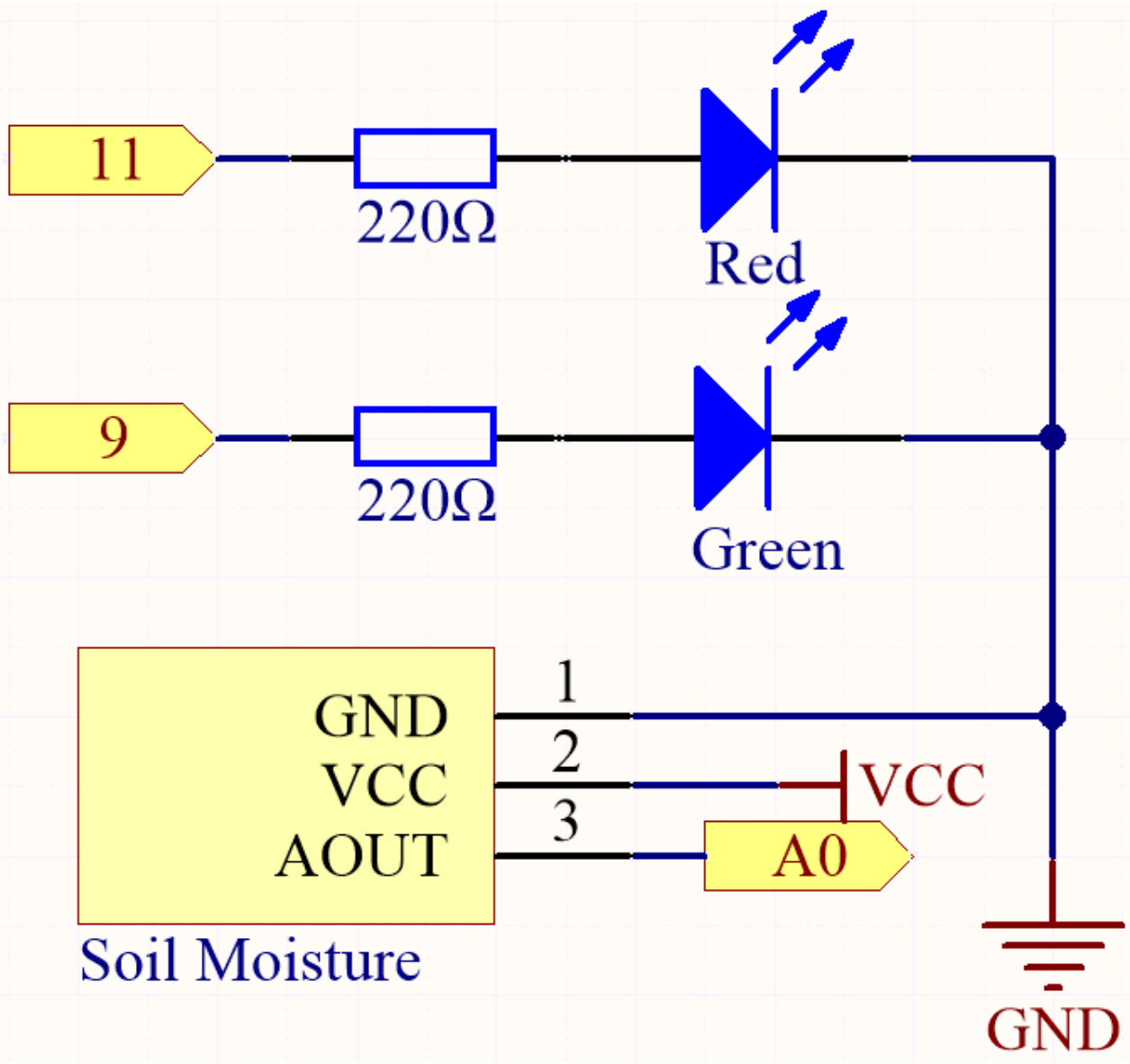
一式を購入するのは非常に便利です、リンクはこちらです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

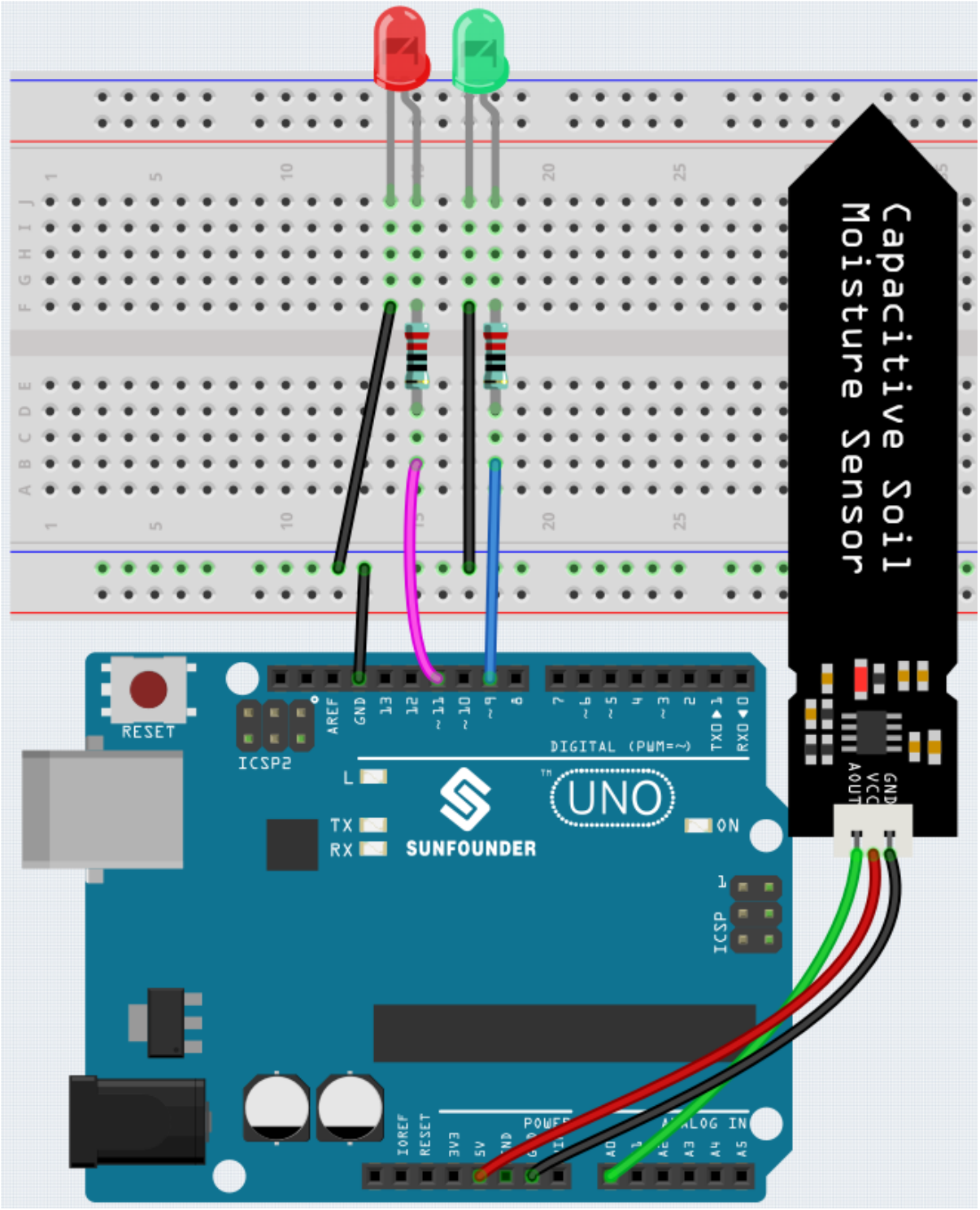
以下のリンクから、部品を個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| 土壌湿度モジュール | |

**回路図**


11
220Ω
Red
9
220Ω
Green
GND
VCC
AOUT
1
2
3
VCC
A0
Soil Moisture
GND

配線図




**コード**

**注釈:**

- 3in1-kit\basic_project\5.2.threshold のパス下の 5.2.threshold.ino ファイルを開きます。

- または、このコードを **Arduino IDE** にコピーします。

コードが正しくアップロードされた後、しきい値が正しく設定されている場合、土壌が乾燥していると赤い LED が点灯して水をやる必要があることを知らせます。水をやった後、緑の LED が点灯します。

**どのように動作するのか？**

```
...

void loop() {
    int sensorValue = analogRead(soilMoisture);
    Serial.println(sensorValue);
    if (sensorValue > threshold) {
        digitalWrite(redPin, HIGH); // 赤い LED を点灯
        digitalWrite(greenPin, LOW); // 緑をオフ
    } else {
        digitalWrite(greenPin, HIGH); // 緑の LED を点灯
        digitalWrite(redPin, LOW); // 赤をオフ
    }
}
...
```

まず、 threshold 値を設定し、その後、土壌湿度モジュールの値を読み取ります。湿度が高くなるとその値は減少します。現在読み取った値が設定した threshold よりも大きい場合、赤い LED を点灯させ、そうでない場合は緑の LED を点灯させます。

この threshold 値は実際の状況に応じて調整する必要があります。まず、コードをアップロードし、シリアルモニタを開いて値を確認します。濡れている状態と乾燥している状態の値を記録し、その中間の値を threshold 値として選択します。

### 4.5.3 5.3 ステート変更の検出

ボタンが他のデバイスを制御する際、ボタンが押されている間だけ動作するわけではなく、離されると動作が停止することもあります。ボタンが押されるたびに動作状態を切り替えることも可能です。

この効果を実現するためには、ボタンが押されるとオフとオンの間で動作状態を切り替える方法を知っておく必要があります。それは「ステート変更の検出」と言います。

このプロジェクトでは、ボタンを使用してモーターを制御します。

**必要な部品**

このプロジェクトで必要な部品は以下の通りです。

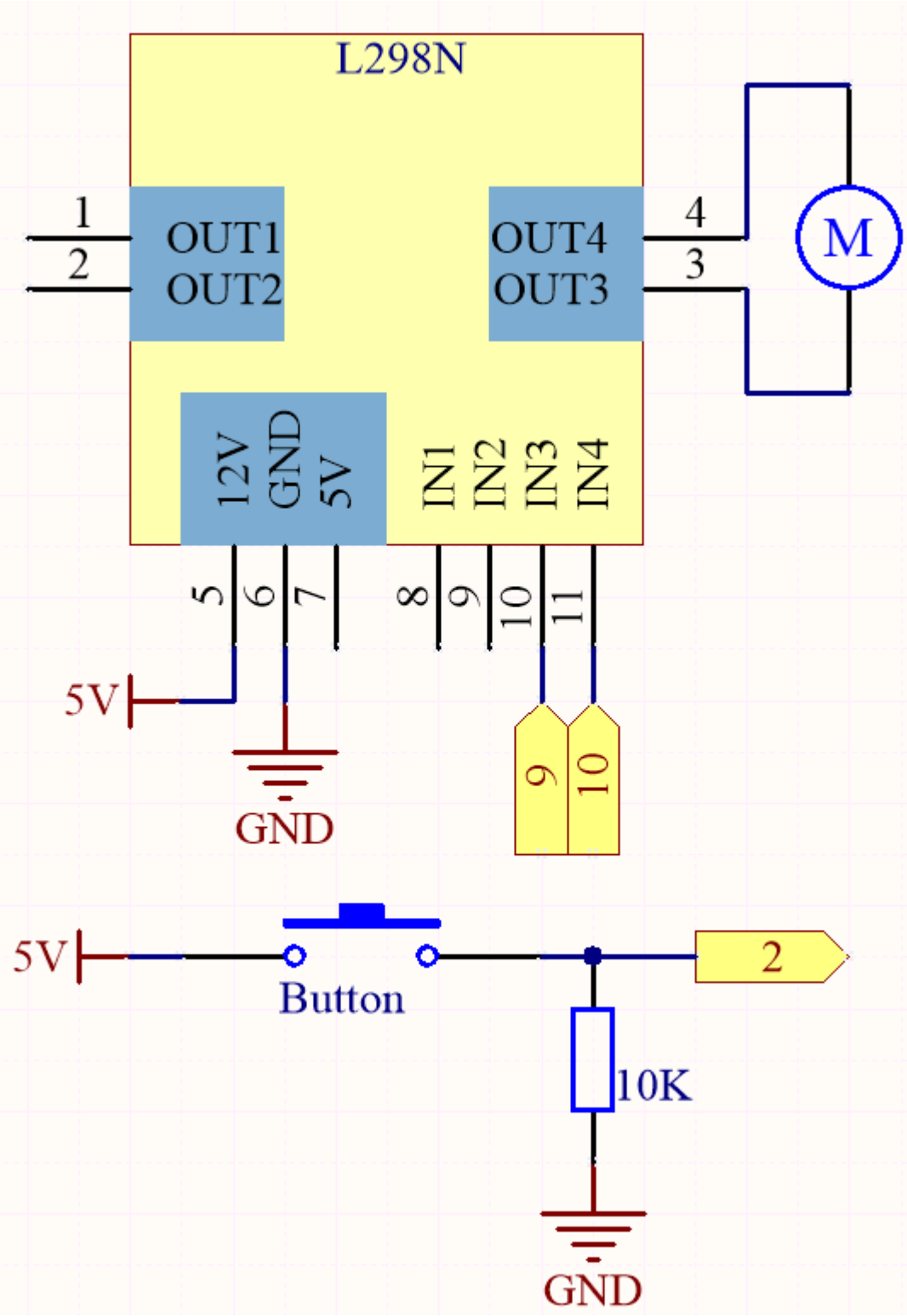
キット全体を購入するのは非常に便利です。リンクは以下の通りです：

| 名称 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

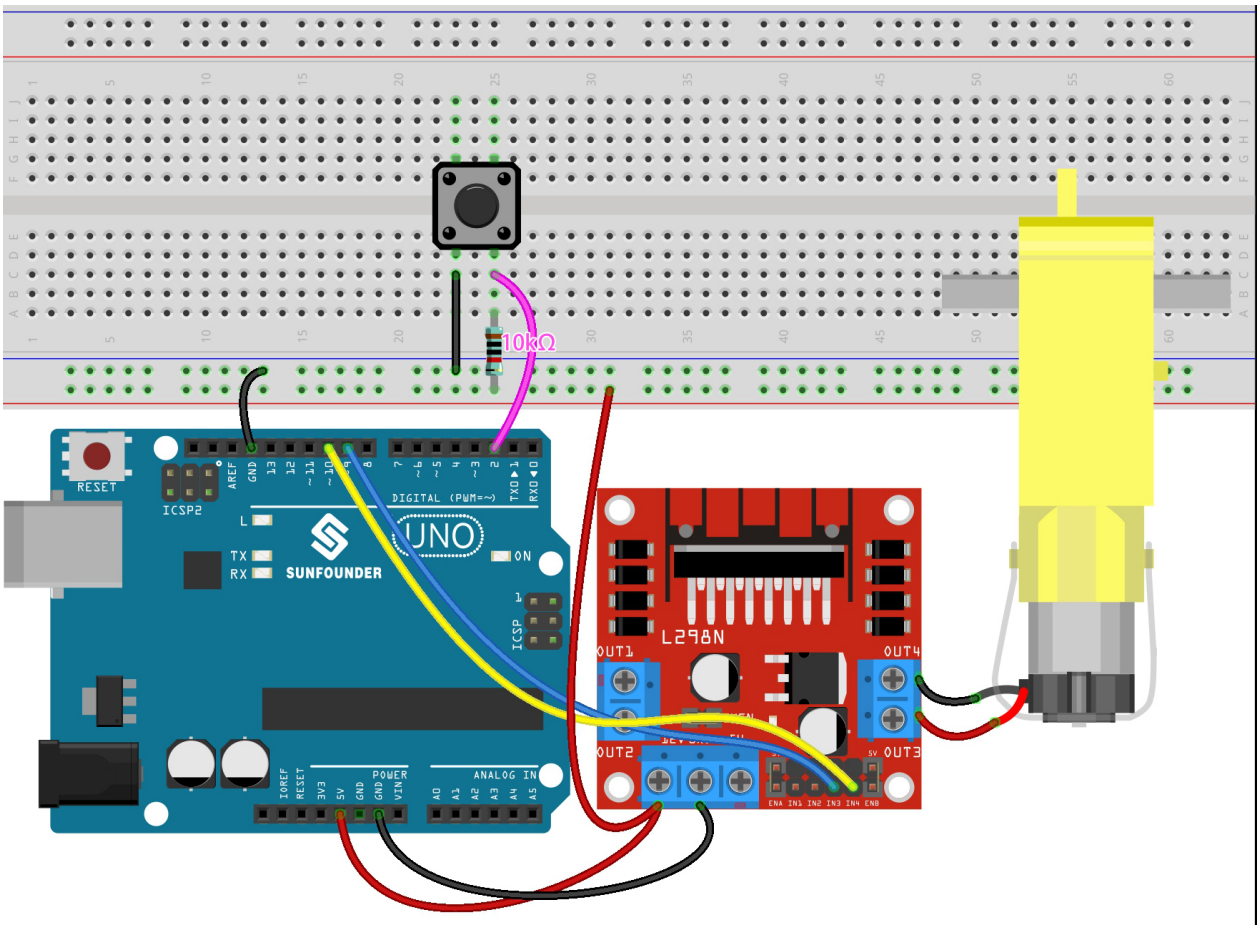
以下のリンクから個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ボタン | |
| *TT* モーター | - |
| *L298N* モジュール | |

**回路図**


L298N
1
2
OUT1
OUT2
OUT4
OUT3
4
3
M
12V
GND
5V
IN1
IN2
IN3
IN4
5
6
7
8
9
10
11
5V
GND
9
10
5V
Button
2
10K
GND

配線図




**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.3.state_change_detection のパスの下にある 5.3.state_change_detection.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされた後、ボタンを押すとモーターが回転し、再びボタンを押すとモーターが停止します。

**どのように動作するのか？**

1. モーターやボタンのピンの変数を作成し、ピンを定義します。

```
...
```

(次のページに続く)

(前のページからの続き)

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

- detectionState は、ボタンが押されるたびに値が変わるフラグです。例えば、今回は 0、次回は 1 と交互になります。
- buttonState と lastButtonState は、今回と前回のボタンの状態を記録するためのもので、ボタンが押されたか、放されたかを比較するために使用されます。

2. 各ピンを初期化し、シリアルモニタのボーレートを設定します。

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

3. ボタンの状態を最初に読み取り、ボタンが押されると、変数 detectionState が 0 から 1、または 1 から 0 に切り替えられます。 detectionState が 1 の場合、モーターが回転します。この回路の効果は、ボタンが押されるたびにモーターが回転し、次にボタンが押されるとモーターが停止する、というものです。

```
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(B_1A,HIGH);
        digitalWrite(B_1B,LOW);
```

(次のページに続く)

(前のページからの続き)

```
    }else{
        digitalWrite(B_1A,LOW);
        digitalWrite(B_1B,LOW);
    }
}
```

全体のワークフローは以下の通りです。

- ボタンの値を読み取ります。

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

- buttonState と lastButtonState が等しくない場合、ボタンの状態が変更されたことを意味します。次の判断を続け、この時点のボタンの状態を変数 lastButtonState に格納します。delay(50) はジッタを排除するために使用されます。

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

- ボタンが押されると、その値は HIGH になります。ここで、ボタンが押されると、変数 detectionState の値が変更されます。例えば、1 つの操作後に 0 から 1 になります。

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

- 変数 detectionState が 1 の場合、モーターを回転させ、それ以外の場合は停止します。

```
if(detectionState==1){
    digitalWrite(motorPinA,HIGH);
    digitalWrite(motorPinB,LOW);
}else{
    digitalWrite(motorPinA,LOW);
    digitalWrite(motorPinB,LOW);
}
```

### 4.5.4 5.4 インターバル

時折、同時に 2 つの作業を行う必要があります。たとえば、LED を点滅させながらボタンの押下を読み取りたい場合などです。この場合、 `delay()` を使用することはできません。なぜなら、Arduino は `delay()` の間、プログラムを一時停止します。 `delay()` を待っている間にボタンが押されると、プログラムはボタンの押下を検出しません。

例えるならば、電子レンジでピザを温めるときに、重要なメールを待っている状況と似ています。ピザを電子レンジに入れて 10 分間セットします。 `delay()` を使うことに例えると、10 分間カウントダウンするタイマーを見つめ続け、0 になるのを待つことになります。この時間内に重要なメールが届いても、気づくことはできません。

実際の生活では、ピザを焼き、メールをチェックし、何か他のことをし（あまり時間がかからない！）、時々電子レンジに戻ってタイマーがゼロになったかどうかを確認する。

このスケッチでは、 `delay()` を使用せずにブザーを鳴らす方法を示しています。ブザーをオンにし、時刻を記録します。その後、 `loop()` の中で所望のインターバル時間が経過したかどうかを確認します。経過していれば、ブザーを鳴らし、新しい時間を記録します。この方法で、スケッチの実行は一つの指示に遅延することなく、ブザーは継続して音を鳴らします。

この条件に基づいて、LED を制御するボタンのコードを追加することができます。これにより、ブザーが音楽を再生しても邪魔されることはありません。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

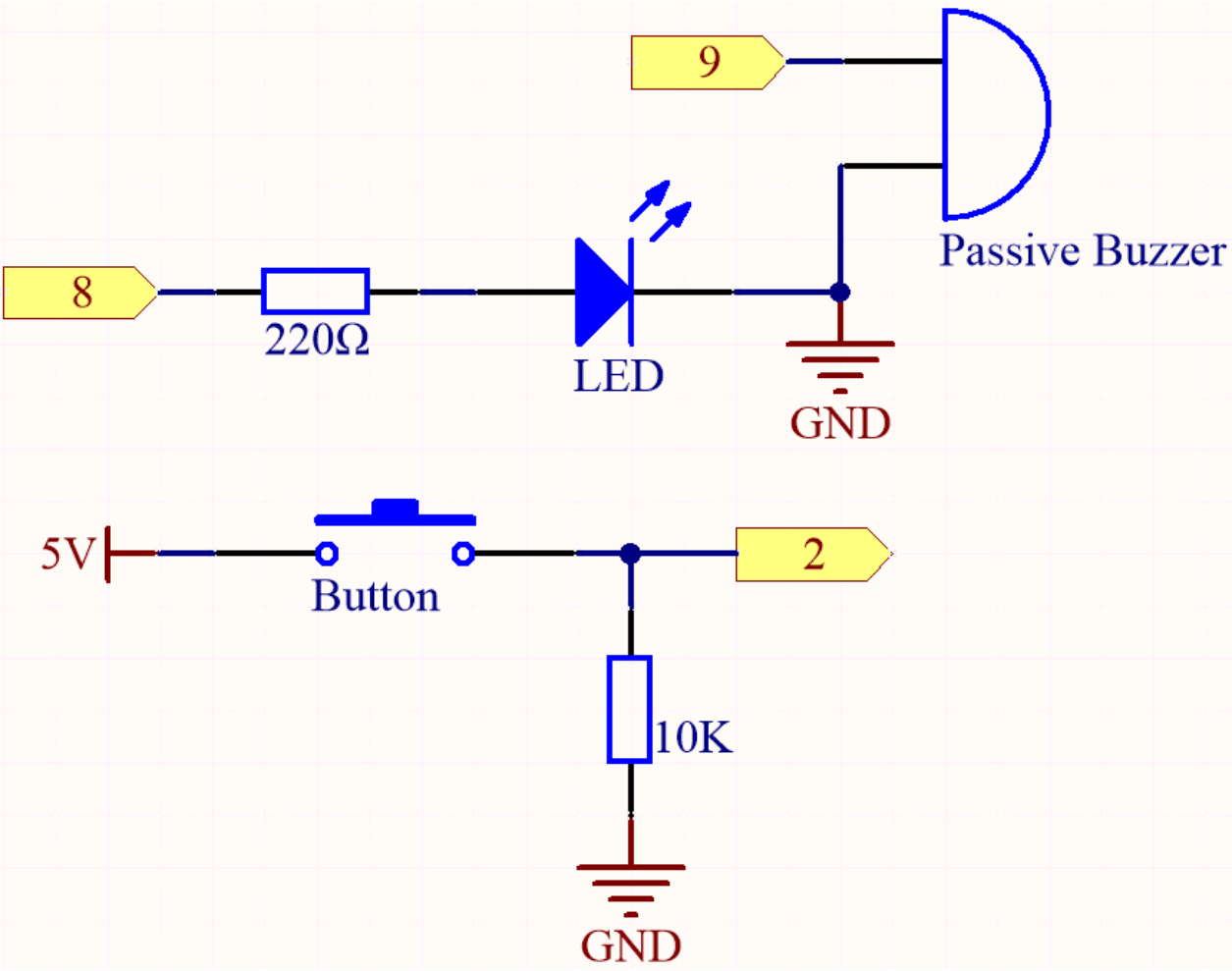
全てのキットを購入するのは確かに便利です、以下がリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

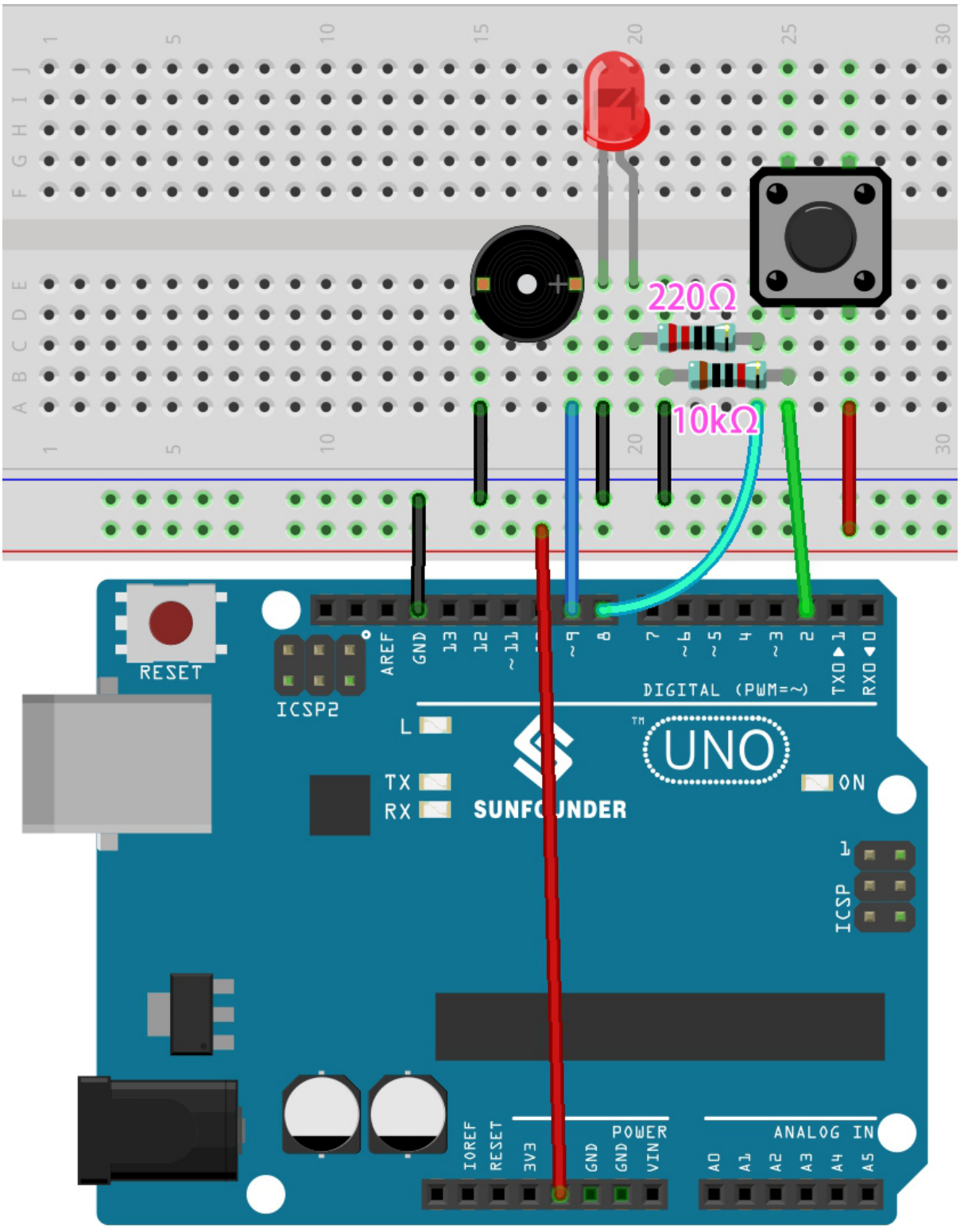
以下のリンクから別々に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| ボタン | |
| ブザー | |

**回路図**




配線図


220Ω
10kΩ
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5

コード

**注釈:**

- 3in1-kit\basic_project\5.4.interval のパスの下の 5.4.interval.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor でコードをアップロードします。

コードが正常にアップロードされると、ブザーは音楽を再生し、ボタンを押すたびに LED が点灯します。LED とブザーの動作は互いに干渉しません。

**どのように動作するのか？**

マイクロコントローラの前回の操作時間を保存する変数 previousMillis を初期化します。

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

どのノートが再生されるかをマークします。

```
int thisNote=0;
```

各ノートの間隔時間。

```
long interval = 1000;
```

loop() 内で、現在の時刻を保存するために currentMillis を宣言します。

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

現在の動作時間と最後の更新時間の間隔が 1000ms より大きい場合、特定の機能がトリガーされます。その間、次のトリガーが 1 秒後に発生するため、previousMillis を現在の時間に更新します。

```
if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;// 最後の音の最後の時間を保存
    //...
}
```

メロディのノートを順番に再生します。

```
tone(buzzerPin,melody[thisNote],100);
interval=1000/noteDurations[thisNote]; // 音を出す間隔
thisNote=(thisNote+1)%(sizeof(melody)/2); //メロディのノートを順番に
```

ボタンは LED を制御します。

```
// ボタンと LED の再生
digitalWrite(ledPin,digitalRead(buttonPin));
```

### 4.5.5 5.5 内蔵ライブラリの使用

Arduino IDE には、対応する .h ファイルを直接コードに追加することで、多数の内蔵ライブラリを使用することができます。

このプロジェクトでは Servo ライブラリを使用してサーボモータを駆動し、0 ° から 180 ° の間で回転させることができます。

**必要な部品**

このプロジェクトで必要な部品は以下の通りです。

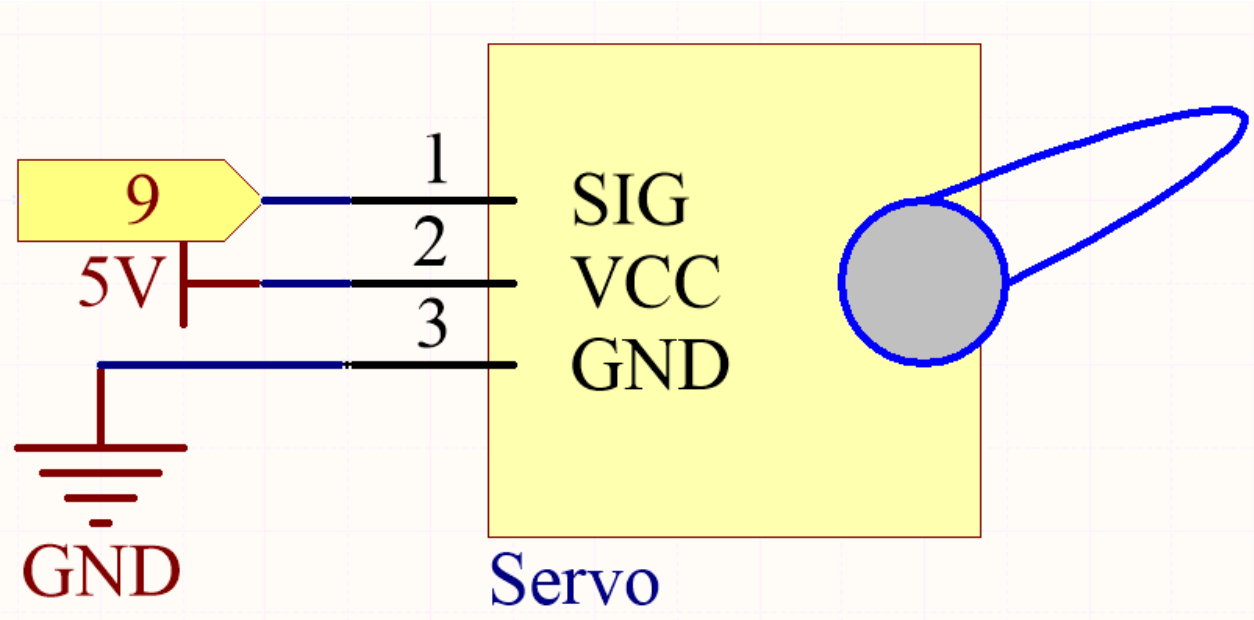
全てのキットを購入するのは確かに便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

また、以下のリンクから部品を個別に購入することもできます。

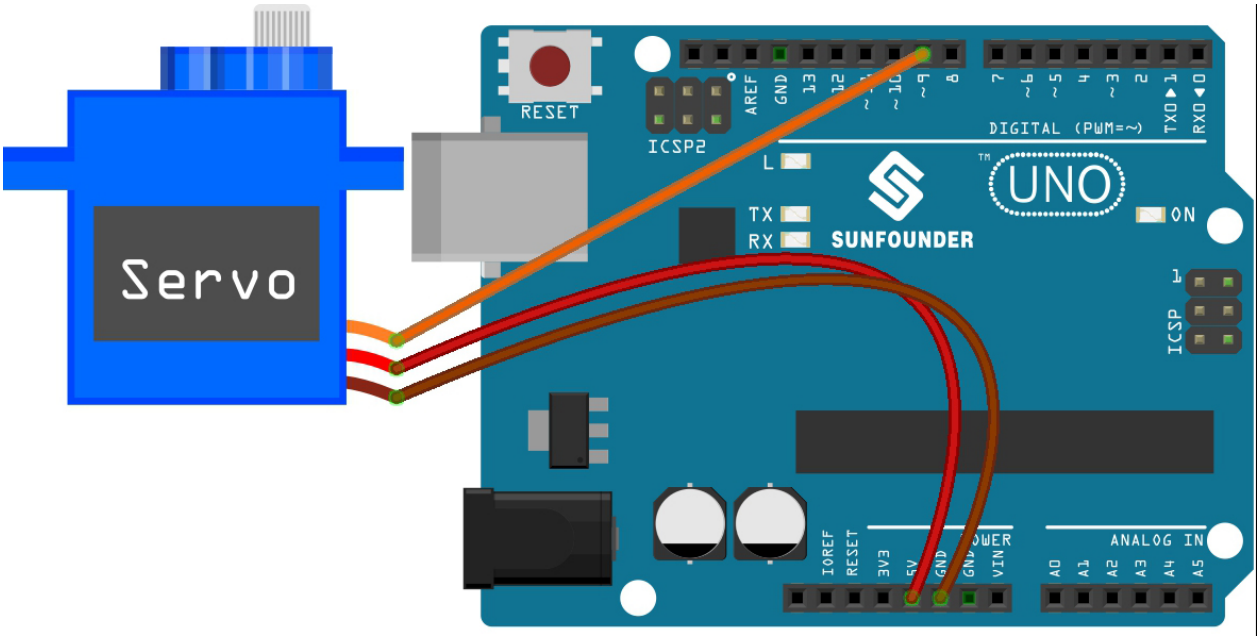
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| サーボ | |

**回路図**




このプロジェクトでは、PWM ピン 9 を使用してサーボを駆動します。サーボのオレンジワイヤーを PWM ピン 9 に、赤ワイヤーを 5V に、ブラウンワイヤーを GND に接続します。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.5.use_internal_library のパスの下の 5.5.use_internal_library.ino ファイルを開きます。

- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor でコードをアップロードします。

コードを R3 ボードにアップロードすると、サーボアームが 0 ° ～180 ° の範囲で回転するのを確認できます。

**どのように動作するのか？**

ライブラリ Servo.h を呼び出すことで、簡単にサーボを駆動することができます。

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

ライブラリの関数：

```
Servo
```

サーボを制御するための **Servo** オブジェクトを作成。

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

pinMode() を呼び出してピンをサーボドライバに変換し、失敗時に 0 を返す。

```
void detach();
```

サーボ駆動からピンを解放。

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

サーボの角度を度数で設定、0 から 180。

```
int read();
```

最後の write() で設定した値を返す。

```
bool attached();
```

サーボが現在接続されている場合は 1 を返す。

### 4.5.6 5.6 マッピング

よく観察してみると、多くの値がプログラミングの中で異なる範囲を持っていることに気がつくでしょう。例えば、アナログ入力の値範囲は (0~1023) です。アナログ出力の値範囲は (0~255) です。サーボの出力角度は (0~180) です。

これは、ポテンショメーターを使用して LED の明るさやサーボの角度を制御したい場合、マッピング操作を行う必要があることを意味します。

では、それをどのように実現するのか見てみましょう。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

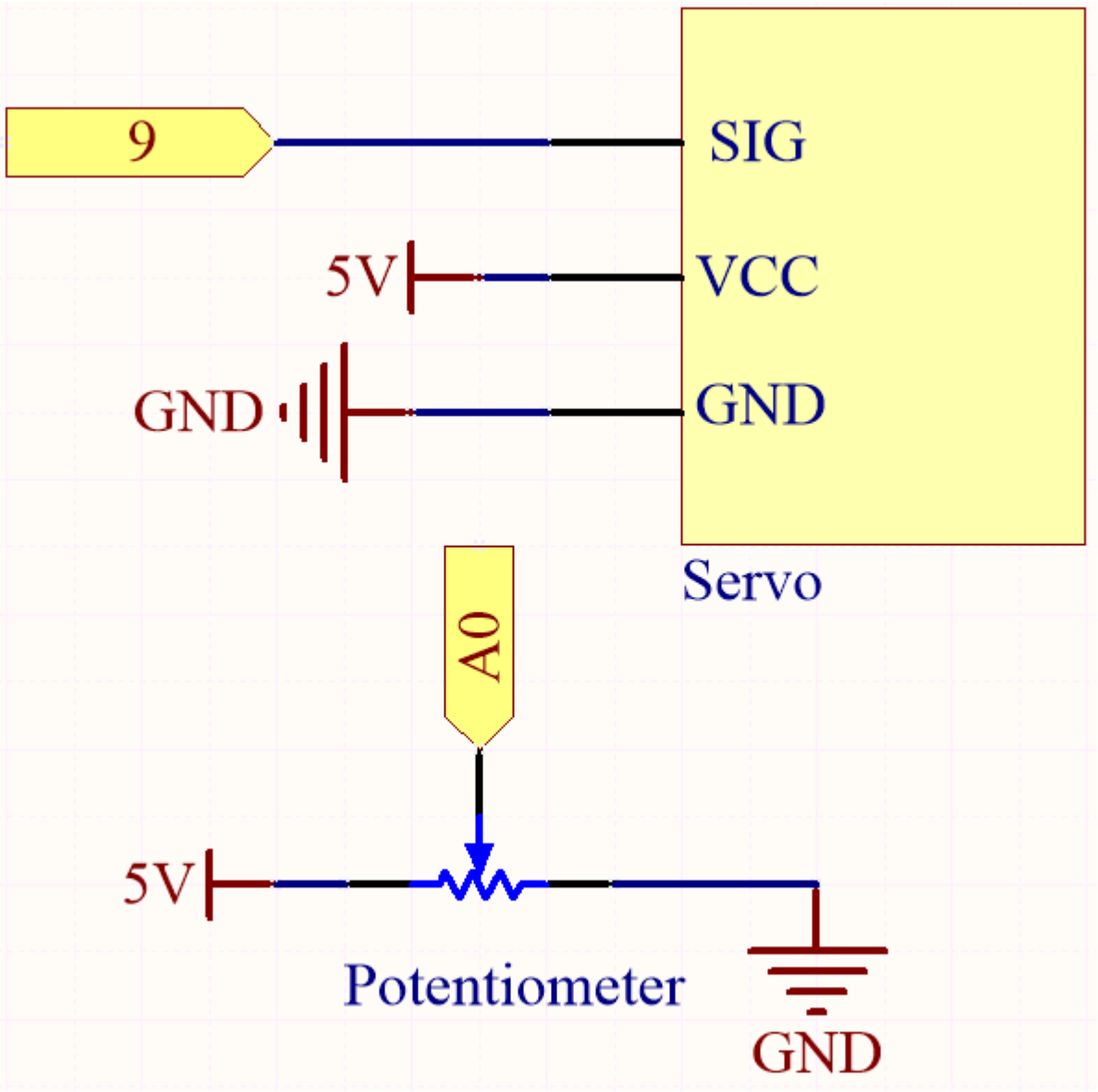
全体のキットを購入する方が便利です。以下がリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

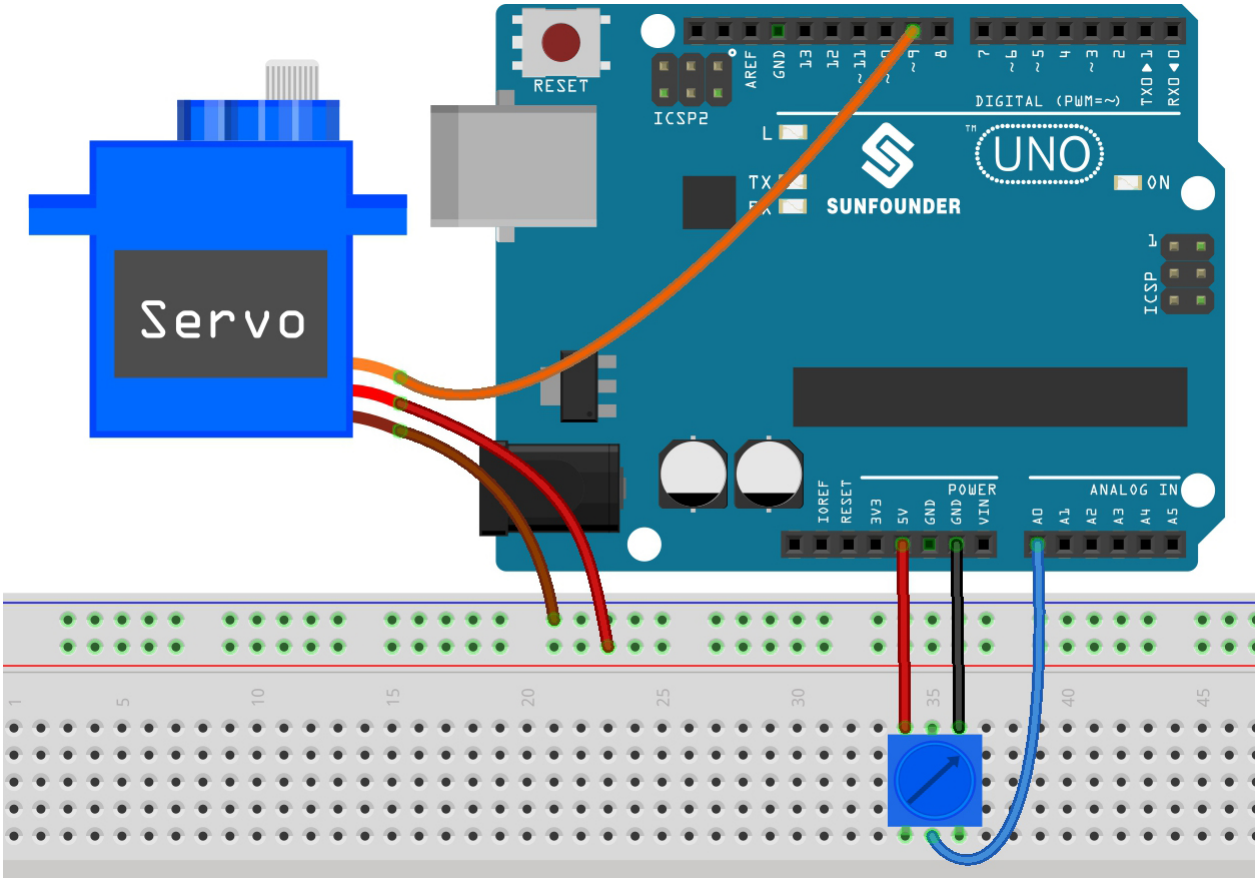
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| サーボ | |
| ポテンショメータ | |

**回路図**




配線図




**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.6.map のパスの下で 5.6.map.ino ファイルを開きます。
- またはこのコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor でコードをアップロードします。

---

コードが正常にアップロードされた後、ポテンショメータを前後に回転させると、サーボの出力軸も前後に回転します。

**どのように動作するのか？**

`map(value, fromLow, fromHigh, toLow, toHigh)`: ある範囲の数字を別の範囲にマッピングします。すなわち、fromLow の値は toLow に、fromHigh の値は toHigh にマッピングされます。

**構文**

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

**パラメータ**

- value: マッピングする数字。
- fromLow: 値の現在の範囲の下限。
- fromHigh: 値の現在の範囲の上限。
- toLow: 値の目標範囲の下限。
- toHigh: 値の目標範囲の上限。

ポテンショメーターで LED を制御する場合、マップを使用してタスクを完了することもできます。

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

**注意と警告**

- 両方の範囲の"下限"は"上限"より大きくまたは小さい場合があります。これは、 map() 関数が数の範囲を逆転させるために使用できることを意味します。

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

- マッピングは負の数にも適用されます。

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

- マッピングは整数を使用し、浮動小数点の小数部は破棄されます。

### 4.5.7 5.7 Tone() または noTone()

Tone() は、指定された周波数（および 50 %のデューティサイクル）の方形波をピンに生成するために使用されます。持続時間を指定することもできますが、そうしない場合、 noTone() が呼び出されるまで波は続きます。

このプロジェクトでは、この 2 つの関数を使用して、受動ブザーを振動させて音を出します。アクティブブザーと同様に、受動ブザーも電磁誘導の現象を利用して動作します。違いは、受動ブザーには振動源がないため、DC 信号を使用するとビープ音が鳴らないことです。ただし、このことにより、受動ブザーは自身の振動周波数を調整し、“ ド、レ、ミ、ファ、ソ、ラ、シ ” などの異なる音符を出すことができます。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

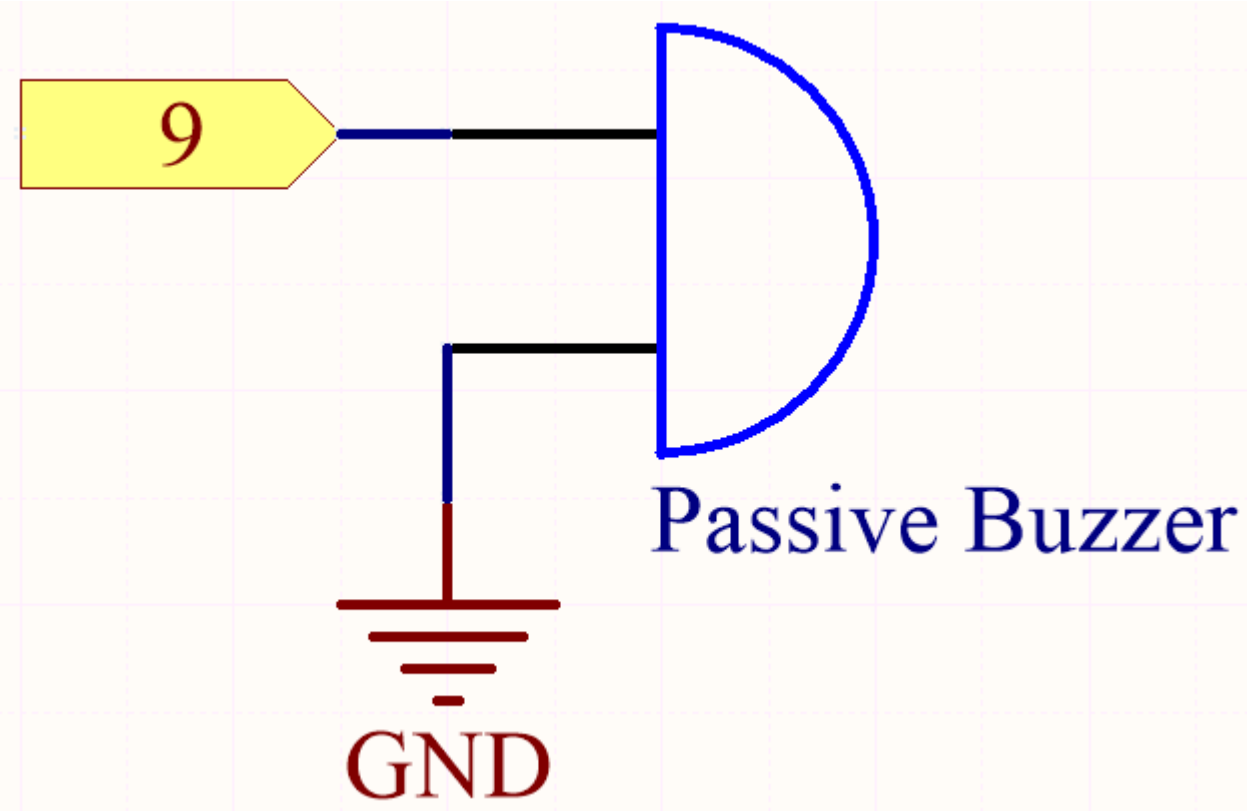
キット全体を購入すると非常に便利です、リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

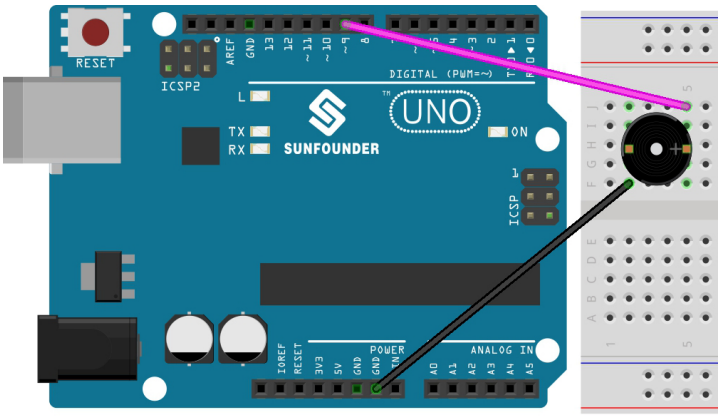
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| ブザー | |

**回路図**




ブザーのカソードを GND に、アノードをデジタルピン 9 に接続します。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.7.tone_notone のパスの下の 5.7.tone_notone.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードを R3 ボードにアップロードすると、7 つの音符を含むメロディーが聞こえます。

**どのように動作するのか？**

注意すべき 2 点があります：

1. tone() & noTone(): この関数は受動ブザーの音を直接制御するために使用され、そのプロトタイプは次のとおりです：

**文法**

void tone(int pin, unsigned int frequency)

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

**パラメータ**

- pin: トーンを生成する Arduino のピン。
- frequency: トーンの周波数（ヘルツ単位）。
- duration: トーンの持続時間（ミリ秒、オプション）

パッシブ・ブザーを振動させて音を出すように）指定された周波数の矩形波（デューティ・サイクルは 50%）をピンに発生させる。継続時間を指定することができ、指定しない場合は noTone() が呼び出されるまで波が継続する。このピンをピエゾブザーなどのスピーカーに接続し、音を鳴らすことができる。

一度に生成できるトーンは 1 つだけである。トーン が既に別のピンで再生されている場合、 tone() を呼び出しても効果はない。トーンが同じピンで再生されている場合、呼び出しはその周波数を設定する。

tone() 関数の使用は、ピン 3 および 11 の PWM 出力を妨害します。

31Hz より低いトーンを生成することはできません。

**文法**

void noTone(int pin)

**パラメータ**

pin: トーンを生成する Arduino のピン。

tone() によってトリガーされた方形波の生成を停止します。トーンが生成されていない場合、効果はありません。

これらの 2 つの関数を知ったら、コードの理解ができるでしょう―配列 melody[] と配列 noteDurations[] の設定は、後続の複数回の tone() 関数の呼び出しと、ループ内でのトーンと持続時間の変更が、音楽の再生効果をより良くするための準備です。

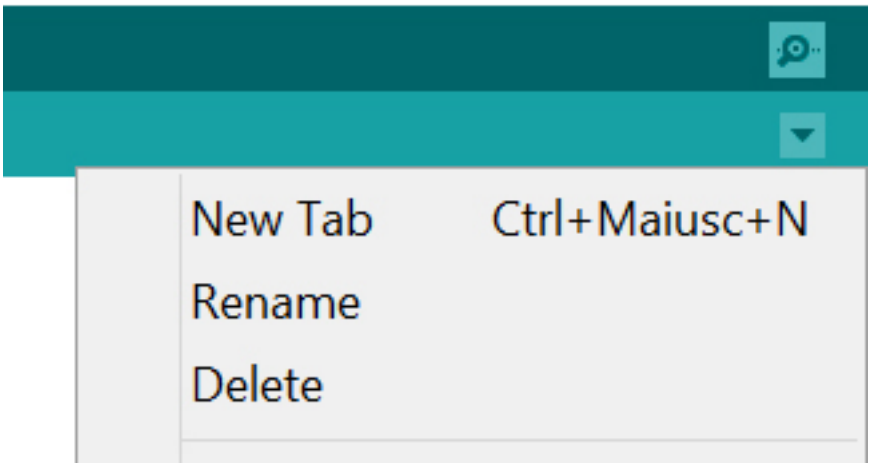
2. pitches.h: コードは追加のファイル、pitches.h を使用します。このファイルには、典型的な音符のピッチ値がすべて含まれています。たとえば、NOTE_C4 は中央の C です。NOTE_FS4 は F#です。この音符テーブルは、 tone() コマンドの基礎となっている Brett Hagman によって元々書かれていました。音楽の音符を作成したい場合、これが役立つでしょう。

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

**注釈:** このサンプルプログラムにはすでに pitches.h ファイルが含まれています。メインのコードと同じフォルダに置く場合、 pitches.h のインストールの後続の手順は省略できます。


コードファイルを開いた後、 pitches.h コードを開くことができない場合は、手動で 1 つ作成することができます。手順は以下の通りです：

pitches.h ファイルを作成するには、シリアルモニターアイコンのすぐ下のボタンをクリックして **New Tab** を選択するか、 **Ctrl+Shift+N** を使用します。




次に、以下のコードを貼り付けて、それを pitches.h として保存します：

```
/*****************
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
#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
```

(次のページに続く)

(前のページからの続き)

```
#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
```

(次のページに続く)

(前のページからの続き)

```
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#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 ユーザー定義関数

C 言語では、大きなプログラムを関数として知られる基本的な構築ブロックに分割することができます。関数には、{}で囲まれた一連のプログラム文が含まれます。関数は、C プログラムに再利用性とモジュール性を提供するために、複数回呼び出すことができます。言い換えれば、関数の集合がプログラムを構築すると言えます。関数は、他のプログラミング言語で手続きやサブルーチンとしても知られています。

関数の以下のような利点があります。

- 関数を使用することで、同じロジック/コードをプログラム内で何度も書く必要がなくなります。
- C の関数は、プログラム内の任意の場所から、任意の回数呼び出すことができます。

- 多数の関数に分割された大きな C プログラムは、追跡が容易です。
- 再利用性は、C 関数の主要な成果です。
- ただし、関数の呼び出しは、C プログラム内で常にオーバーヘッドとなります。

C プログラミングには 2 種類の関数があります：

- **ライブラリ関数**：C のヘッダーファイルで宣言されている関数。
- **ユーザー定義関数**：C プログラマーが作成した関数で、何度も使用することができます。これにより、大きなプログラムの複雑さが軽減され、コードが最適化されます。

このプロジェクトでは、超音波モジュールの値を読む関数を定義します。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

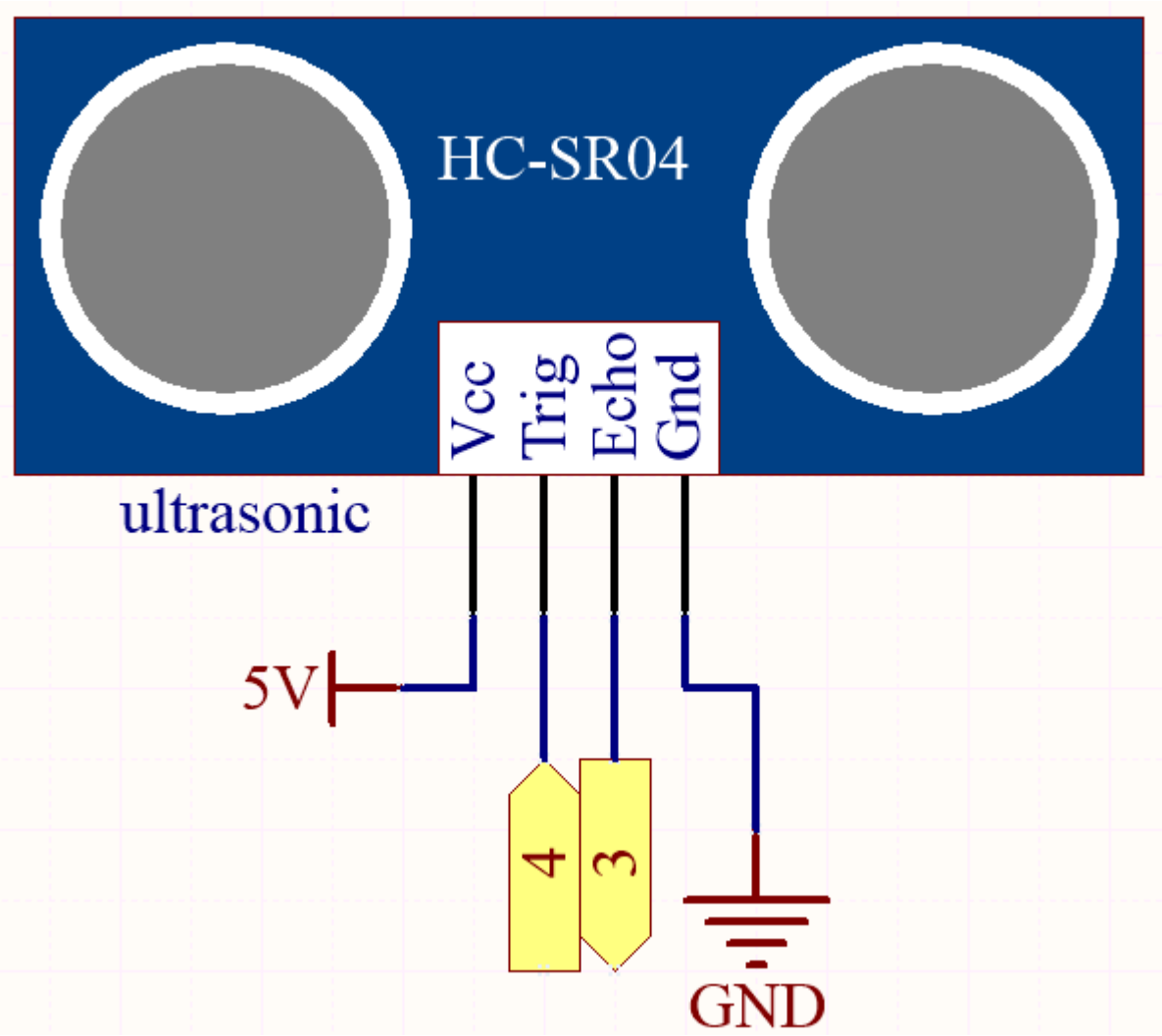
一式を購入するのが便利です。購入リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

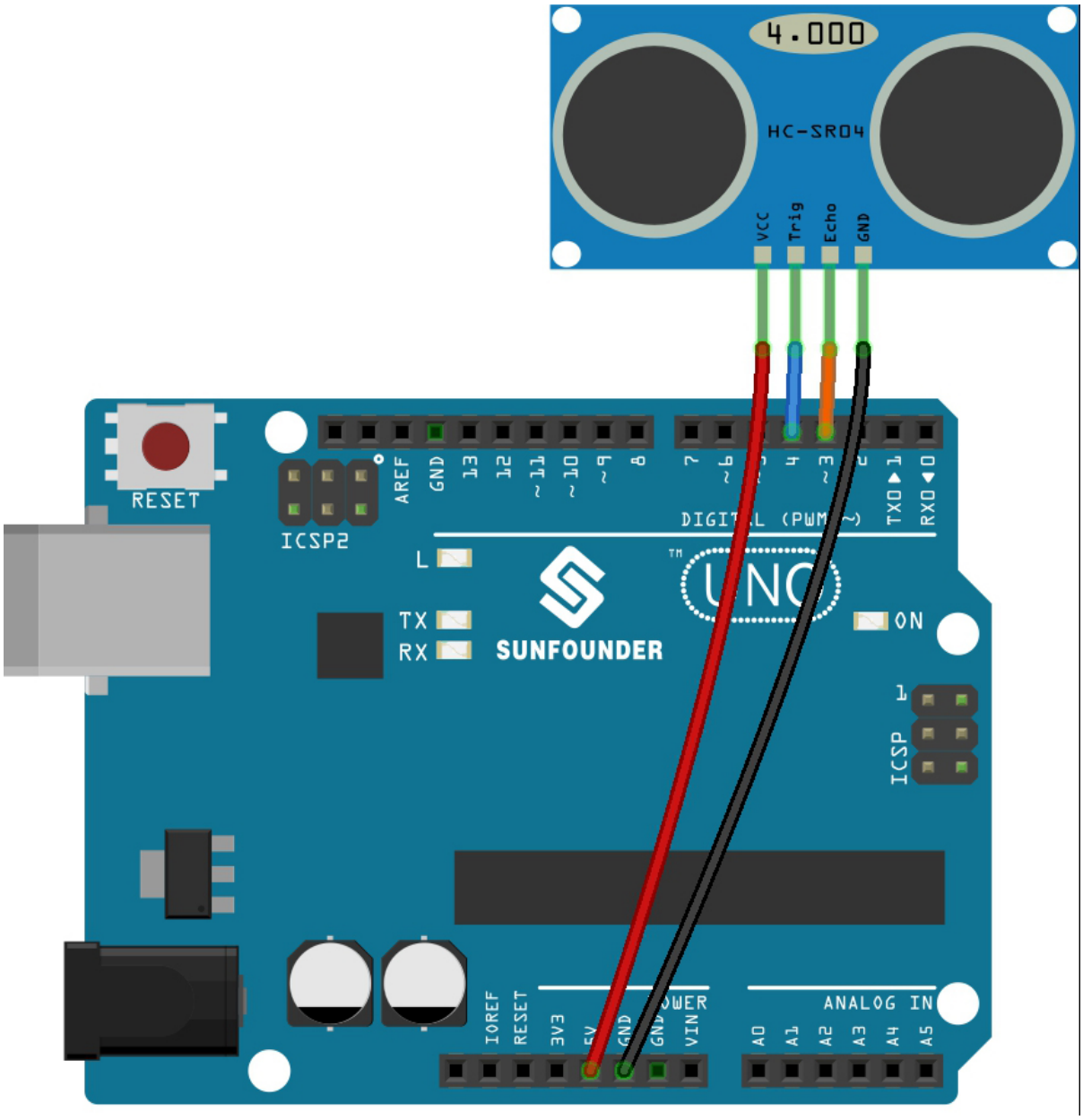
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ジャンパーワイヤー** | |
| **超音波モジュール** | |

**回路図**




配線図




**コード**

**注釈:**

- 3in1-kit\basic_project\5.8.user_function のパスの下で 5.8.user_function.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされると、シリアルモニタは超音波センサと前方の障害物との間の距離を表示します。

**どのように動作するのか？**

超音波センサの使用方法については、サブ関数を直接確認できます。

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

超音波モジュールの trigPin は、2us ごとに 10us の方形波信号を送信します。

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

echoPin は、範囲内に障害物が存在する場合に高レベルの信号を受信し、 pulseIn() 関数を使用して送信から受信までの時間を記録します。

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

音の速度は、340 m/s または 29 マイクロ秒/センチメートルです。

これにより、方形波が往復する距離が得られるので、障害物の距離を得るために 2 で割ります。

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

超音波センサは動作中にプログラムを一時停止するので、複雑なプロジェクトを書いているときに若干の遅延が発生することがあります。

### 4.5.9 5.9 ShiftOut(LED)

shiftOut() は 74HC595 が 8 つのデジタル信号を出力するために使用されます。バイナリ数の最後のビットを Q0 に、最初のビットの出力を Q7 に行います。つまり、バイナリ数 " 00000001 " を書き込むと、Q0 が高レベルを出力し、Q1〜Q7 が低レベルを出力します。

このプロジェクトでは、74HC595 の使用方法を学びます。74HC595 は 8 ビットのシフトレジスタと三状態の並列出力を持つストレージレジスタで構成されています。シリアル入力を並列出力に変換するため、MCU の IO ポートを節約できます。

具体的には、74hc595 は 8 ビットのバイナリ数を書き込むことでデジタル信号出力のための 8 つのピンを置き換えることができます。

- バイナリ数 - Wikipedia

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

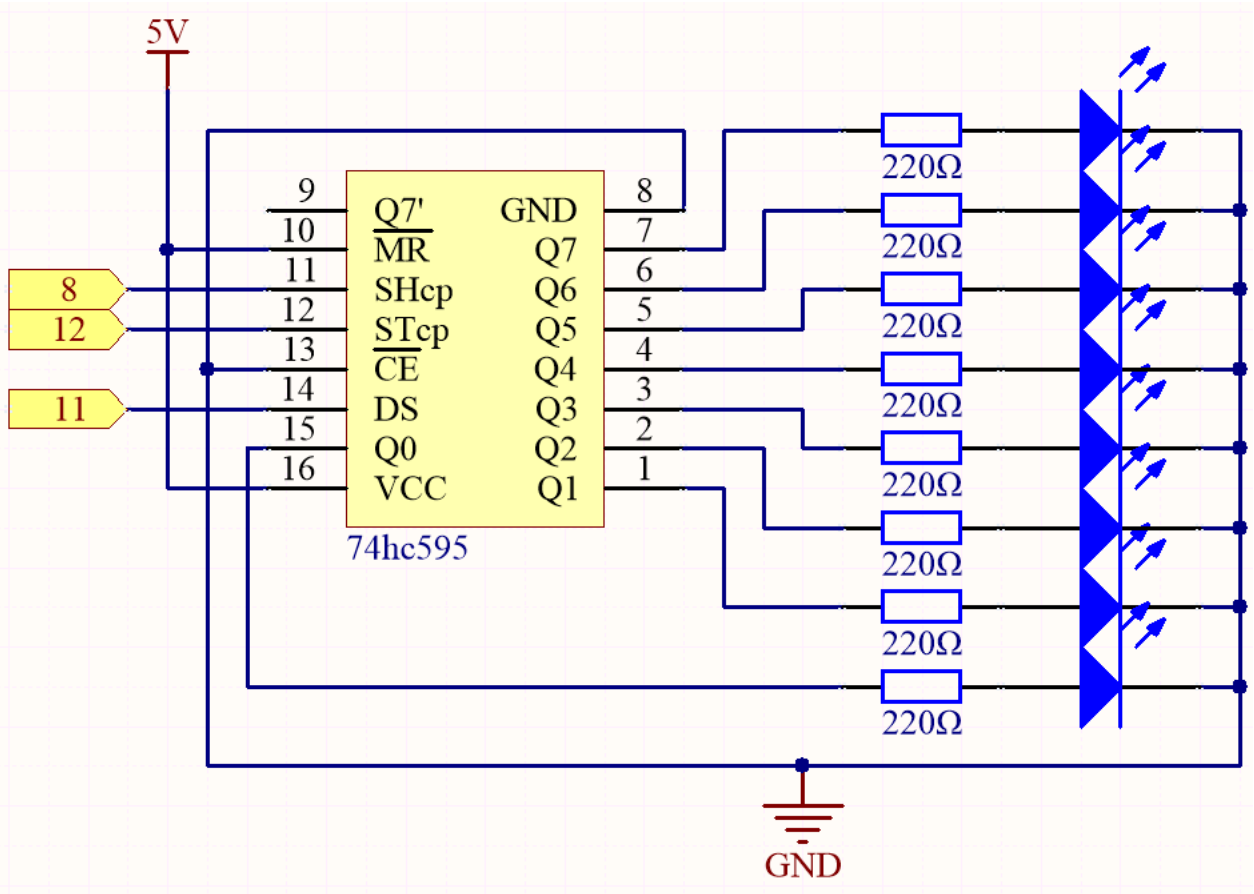
一式をまとめて購入することは非常に便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

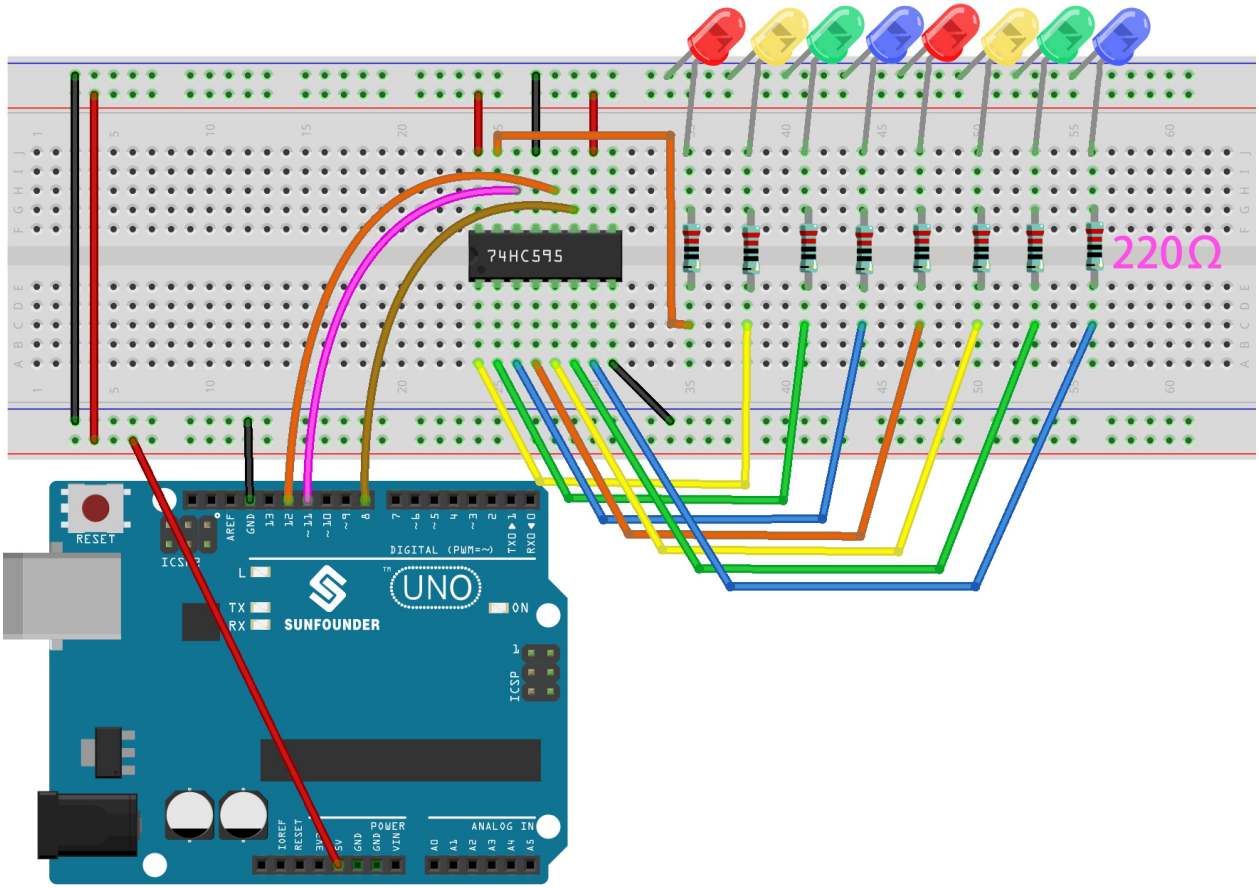
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| **抵抗器** | |
| *LED* | |
| *74HC595* | |

**回路図**




- MR (pin10) が高レベルで OE (pin13) が低レベルのとき、データは SHcp の立ち上がりエッジで入力され、SHcp の立ち上がりエッジを通ってメモリレジスタに移動します。
- 2 つのクロックが連結されている場合、シフトレジスタは常にメモリレジスタよりも 1 パルス早くなります。
- メモリレジスタにはシリアルシフト入力ピン (Ds)、シリアル出力ピン (Q)、非同期リセットボタン（低レベル）があります。
- メモリレジスタは、3 状態での 8 ビット並列のバスを出力します。
- OE が有効（低レベル）のとき、メモリレジスタのデータはバス (Q0〜Q7) に出力されます。

**配線図**




**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.9.shiftout_led のパスの下の 5.9.shiftout_led.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

R3 ボードにコードをアップロードすると、LED が次々に点灯するのが見られます。

**どのように動作するのか？**

配列を宣言し、74HC595 によって制御される 8 つの LED の動作状態を変更するために使用されるいくつかの 8 ビットバイナリ数を保存します。

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

最初に STcp を低レベルに設定し、次に高レベルに設定します。これにより、STcp の立ち上がりエッジのパルスが生成されます。

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

shiftOut() は、一度に 1 ビットのデータをシフトアウトするために使用され、datArray[num] のデータのバイトを DS ピンを使用してシフトレジスタにシフトします。 **MSBFIRST** は高ビットからの移動を意味します。

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

digitalWrite(STcp,HIGH) が実行された後、 STcp は立ち上がりエッジになります。この時点で、シフトレジスタのデータがメモリレジスタに移動されます。

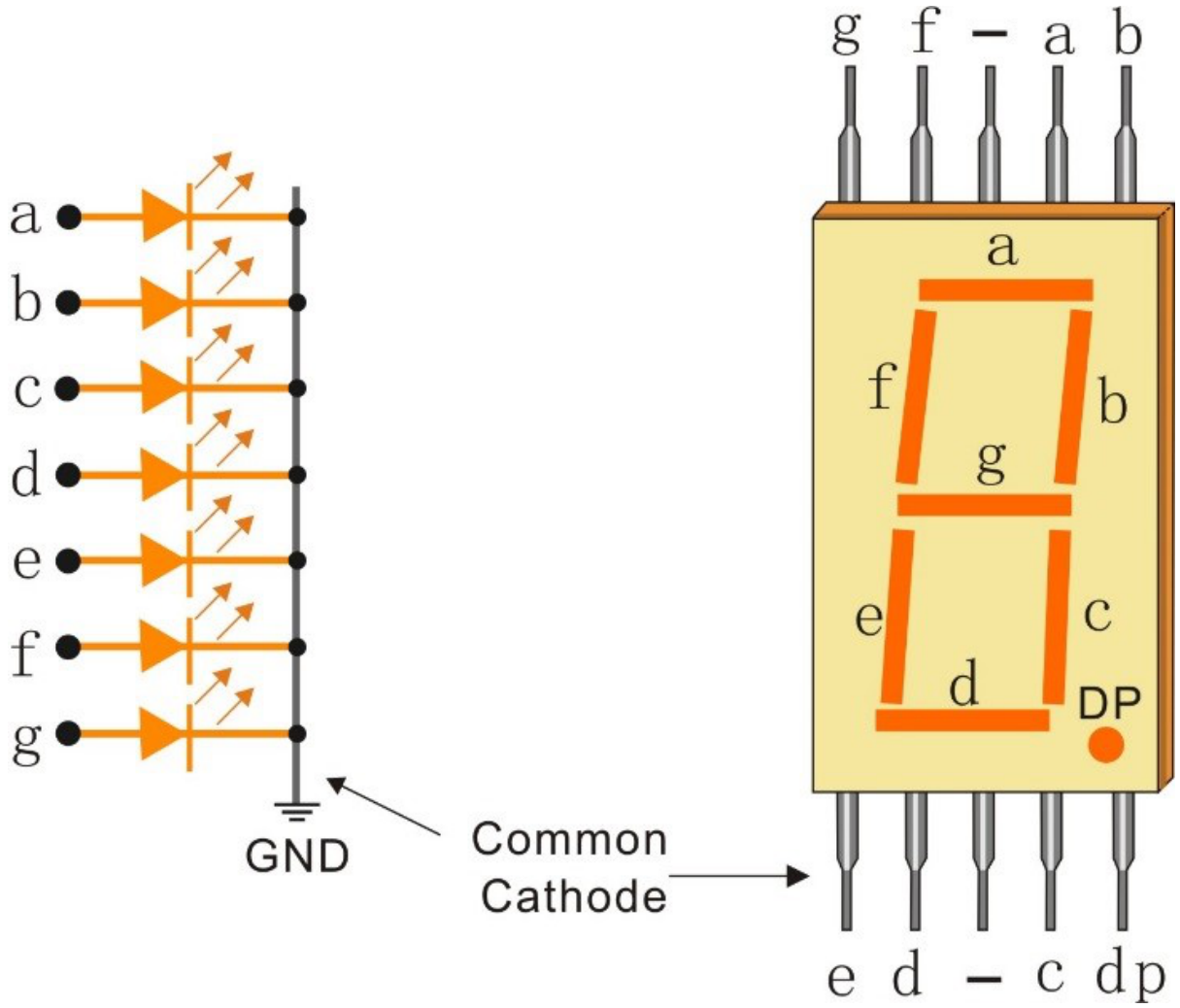
```
digitalWrite(STcp,HIGH);
```

8 回の後に、1 バイトのデータがメモリレジスタに転送されます。次に、メモリレジスタのデータがバス (Q0-Q7) に出力されます。例えば、 B00000001 をシフトアウトすると、Q0 によって制御される LED が点灯し、Q1～Q7 によって制御される LED が消灯します。

### 4.5.10 5.10 ShiftOut(7 セグメント表示)

以前は、 shiftout() 関数を使用して 8 つの LED を点灯させましたが、ここではそれを使用して 7 セグメントディスプレイに 0-9 を表示します。

7 セグメントディスプレイは、基本的に 8 つの LED で構成されたデバイスであり、7 つのストリップ形状の LED が「8」の形を作り、小さな点 LED が小数点としてあります。これらの LED は a, b, c, d, e, f, g, dp としてマークされています。それぞれが独自のアノードピンを持ち、カソードを共有しています。ピンの位置は下の図に示されています。




**必要な部品**

このプロジェクトでは、以下の部品が必要です。

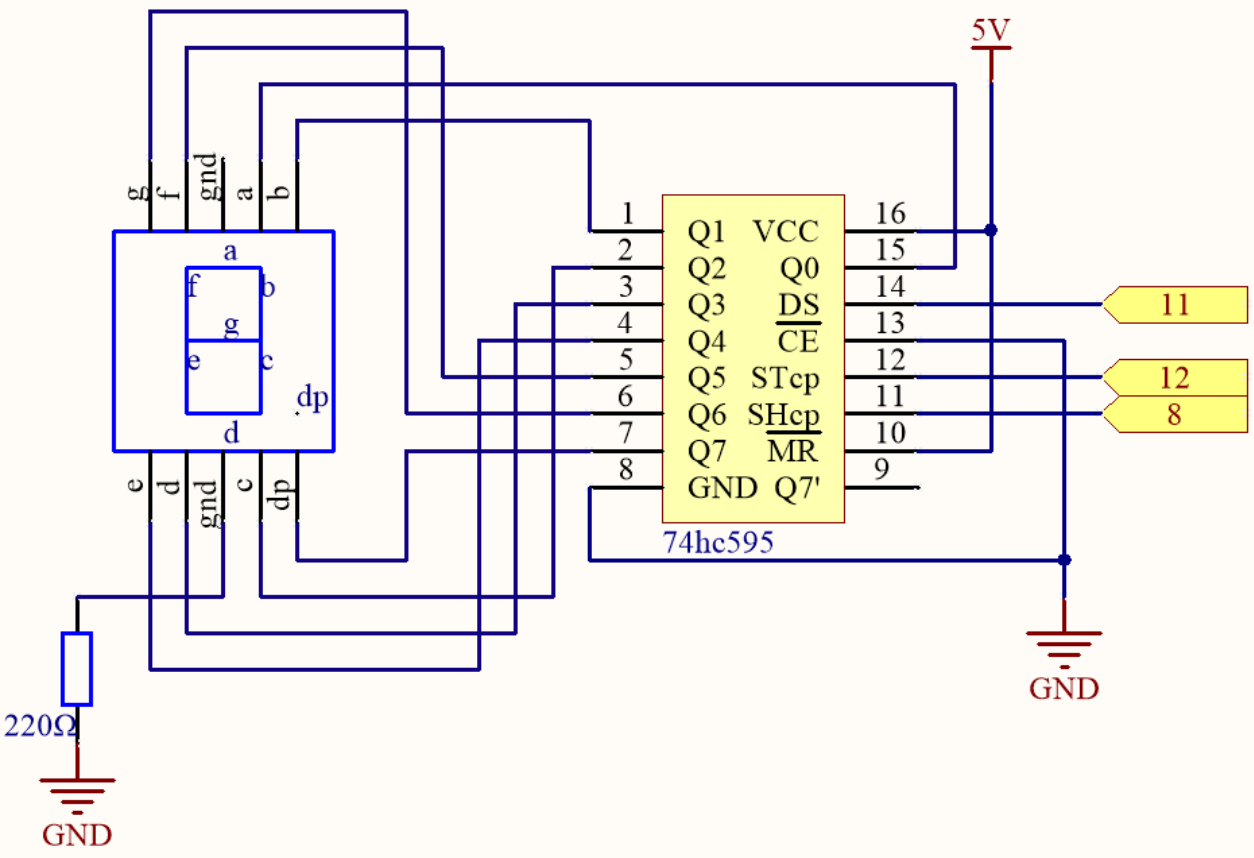
キット全体を購入するのは確かに便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *7* セグメント表示 | |
| *74HC595* | |

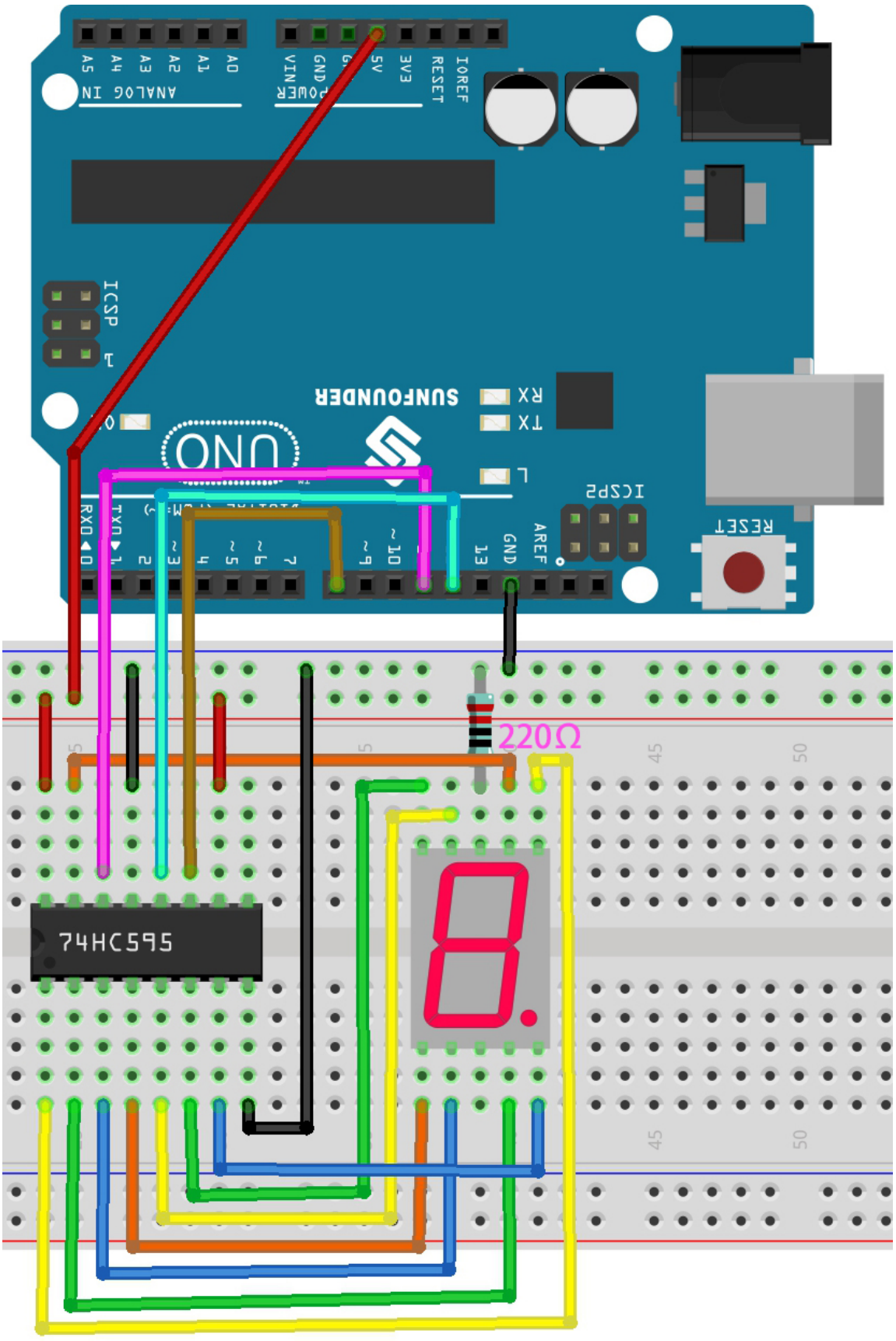
**回路図**




**配線図**

表 1　配線

| 74HC595 | LED セグメントディスプレイ |
|---|---|
| Q0 | a |
| Q1 | b |
| Q2 | c |
| Q3 | d |
| Q4 | e |
| Q5 | f |
| Q6 | g |
| Q7 | dp |


220Ω
74HC595
UNO
SUNFOUNDER
ANALOG IN
POWER
ICSP
RESET

**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.10.shiftout_segment のパスの下にある 5.10.shiftout_segment.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされた後、LED セグメントディスプレイが 0〜9 を順番に表示するのを見ることができます。

**どのように動作するのか？**

shiftOut() は 74HC595 に 8 つのデジタル信号を出力させます。それは、二進数の最後のビットを Q0 に、最初のビットの出力を Q7 にします。つまり、二進数"00000001"を書くと、Q0 はハイレベルを出力し、Q1〜Q7 はローレベルを出力します。

7 セグメントディスプレイで数字の「2」を表示すると仮定すると、a, b, d, e, g にハイレベルを書き、c, f, dp にローレベルを書く必要があります。つまり、二進数 "01011011"を書く必要があります。可読性のために、16 進表記として"0x5b"を使用します。


- 16 進数
- BinaryHex コンバータ

同様に、7 セグメントディスプレイで他の数字も同じ方法で表示させることができます。以下の表は、これらの数字に対応するコードを示しています。

表 2　グリフコード

| 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 |

これらのコードを `shiftOut()` に書き込むと、LED セグメントディスプレイが対応する数字を表示します。

### 4.5.11 5.11 外部ライブラリのインストール

ライブラリは、Arduino IDE の機能を拡張するための既存のコードや関数の集合体です。ライブラリは、様々な機能のための使用可能なコードを提供し、複雑な機能のコーディングで時間や労力を節約することができます。

ライブラリをインストールする主な方法は 2 つあります。

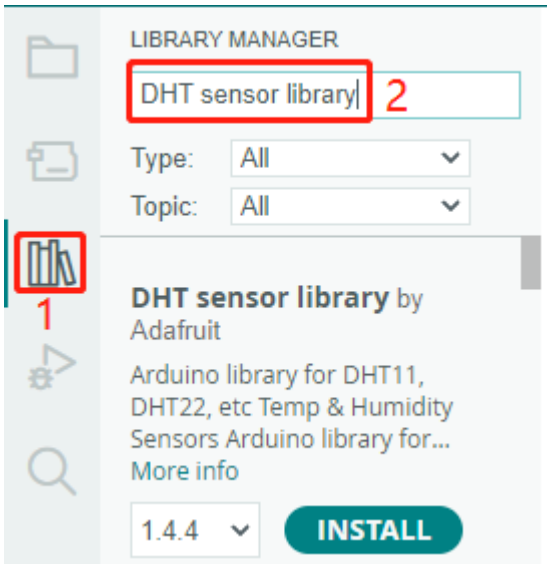
Arduino の **Library Manager** を通じて、多くのライブラリが直接利用可能です。次の手順で **Library Manager** にアクセスできます：

1. **Library Manager** で、名前で所望のライブラリを検索したり、異なるカテゴリを参照することができます。

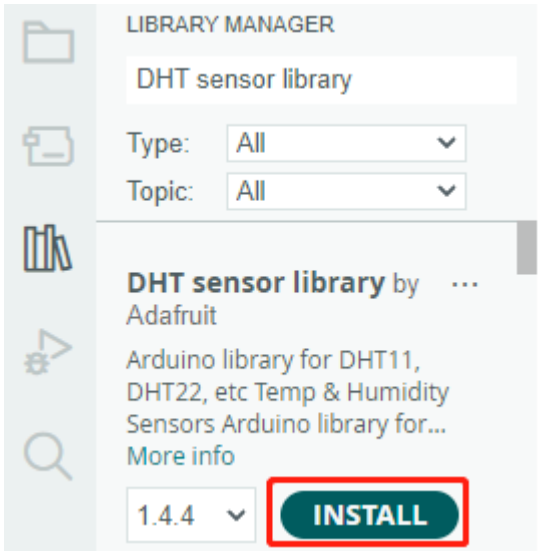
---

**注釈:** ライブラリのインストールが必要なプロジェクトでは、どのライブラリをインストールするかを示すプロンプトが表示されます。指示に従って、"ここで `DHT sensor library` が使用されています。 **Library Manager** からインストールできます"のように、推奨されるライブラリをインストールしてください。

---




2. インストールしたいライブラリを見つけたら、それをクリックして **Install** ボタンをクリックします。




3. Arduino IDE は自動的にライブラリをダウンロードしてインストールします。

**関連コンポーネント**

以下は関連するコンポーネントです。クリックして使用方法を学ぶことができます。

### 5.11.1 液晶ディスプレイ

I2C LCD1602 は、LCD1602 と I2C モジュールで構成されています。LCD1602 は文字、数字などを表示するために使用できますが、多くのピンをメインコントロールに使用する必要があります。I2C モジュールを設定すると、この LCD1602 を駆動するのに 2 つの I/0 ピンのみが必要になります。

この I2C CDL1602 がどのように動作するかを見てみましょう。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

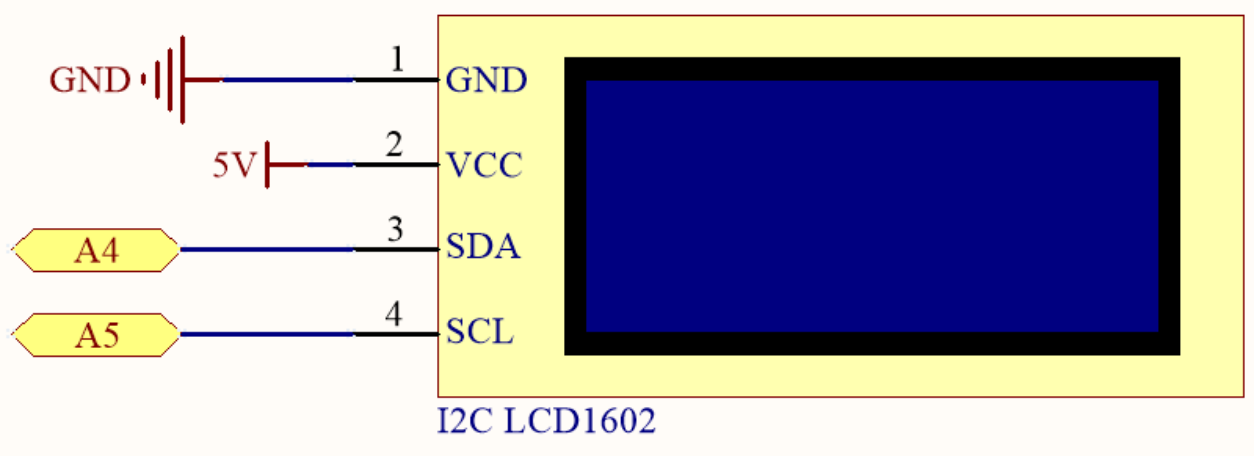
キット全体を購入するのは非常に便利です。リンクは以下のとおりです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

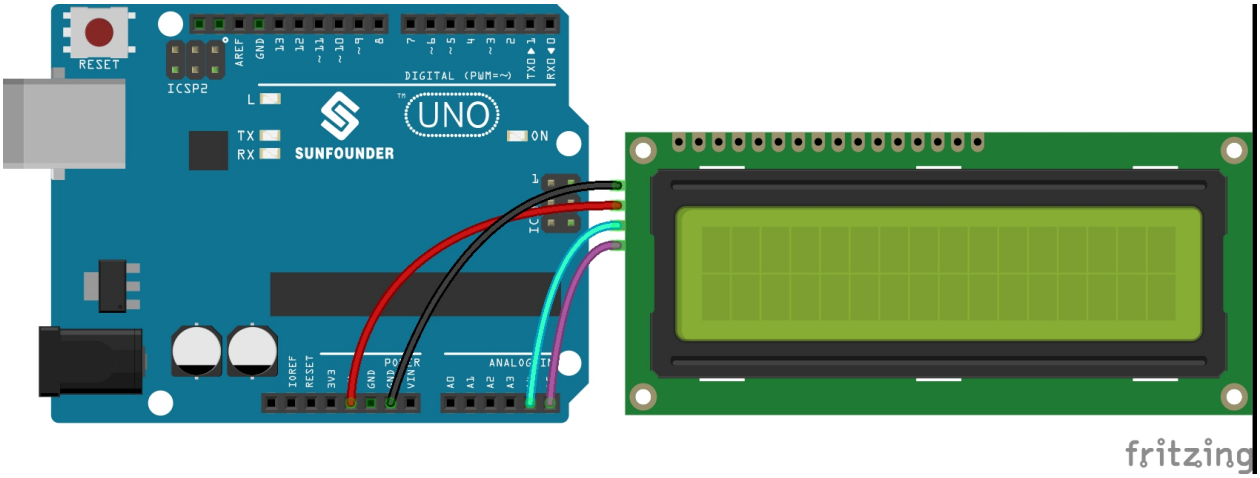
以下のリンクからそれぞれ別々に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ジャンパーワイヤー** | |
| *I2C LCD1602* | |

**回路図**




**配線図**

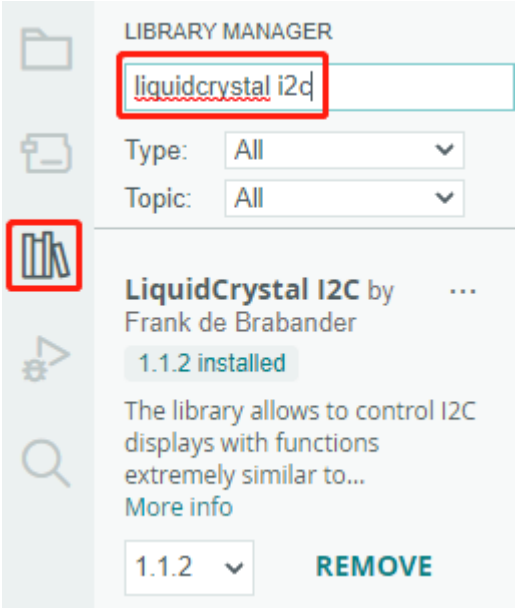



**注釈:** R3 ボードの SDA と SCL は、ピン A4 と A5 です。

### コード

**注釈:**

- `3in1-kit\basic_project\5.11.liquid_crystal_display` のパスの下にある `5.11.liquid_crystal_display.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- ここでは `LiquidCrystal I2C` ライブラリが使用されています。 **Library Manager** からインストールできます。



コードが正常にアップロードされると、I2C LCD1602 に"SunFounder"、"Hello World"が表示されます。

**注釈:** コードと配線が正しいのに LCD が内容を表示しない場合、背面のポテンショメータを回してみてください。

**どのように動作するのか？**

LiquidCrystal_I2C.h ライブラリを呼び出すことで、LCD を簡単に駆動することができます。

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

ライブラリの機能：

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

Arduino ボードに接続された特定の LCD を表す LiquidCrystal_I2C クラスの新しいインスタンスを作成します。

- lcd_AddR: LCD のアドレスはデフォルトで 0x27。
- lcd_cols: LCD1602 には 16 の列があります。
- lcd_rows: LCD1602 には 2 行あります。

```
void init()
```

lcd を初期化します。

```
void backlight()
```

(オプションの) バックライトをオンにします。

```
void nobacklight()
```

(オプションの) バックライトをオフにします。

```
void display()
```

LCD ディスプレイをオンにします。

```
void nodisplay()
```

LCD ディスプレイをすぐにオフにします。

```
void clear()
```

表示をクリアし、カーソルの位置をゼロに設定します。

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

カーソルの位置を col,row に設定します。

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

テキストを LCD に表示します。

- data: 表示するデータ（char、byte、int、long、または文字列）。
- BASE（**オプション**）: 数字を印刷する基数：BIN(2 進数)、DEC(10 進数)、OCT(8 進数)、HEX(16 進数)。

### 5.11.2 IR レシーバー

このプロジェクトでは、IR レシーバーの使用方法を学びます。

赤外線受信機は、赤外線信号を受信し、独立して赤外線を受信して TTL レベルと互換性のある信号を出力する部品です。通常のプラスチックパッケージのトランジスタとサイズが似ており、各種の赤外線リモコンや赤外線伝送に適しています。

#### 必要な部品

このプロジェクトでは、以下の部品が必要です。

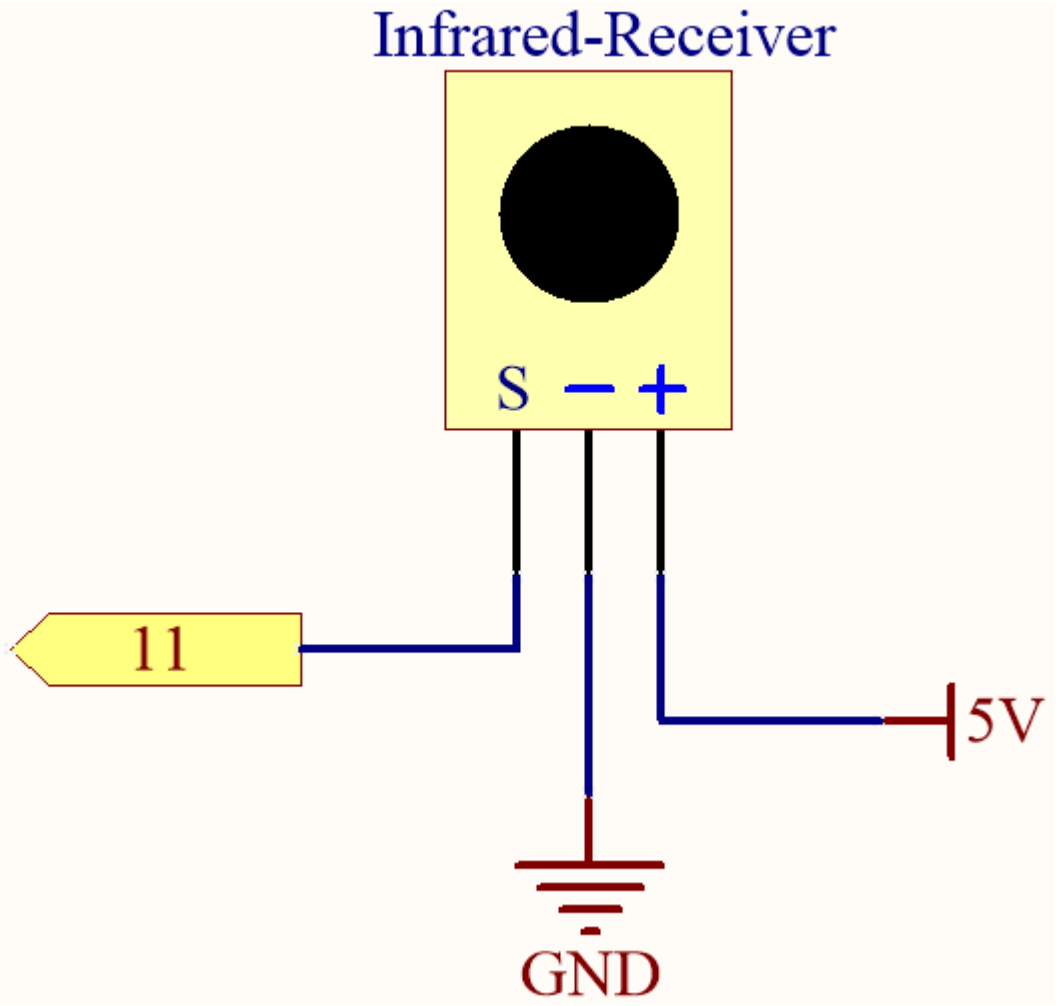
全体のキットを購入すると非常に便利です、リンクは以下です：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクからそれぞれ購入することもできます。

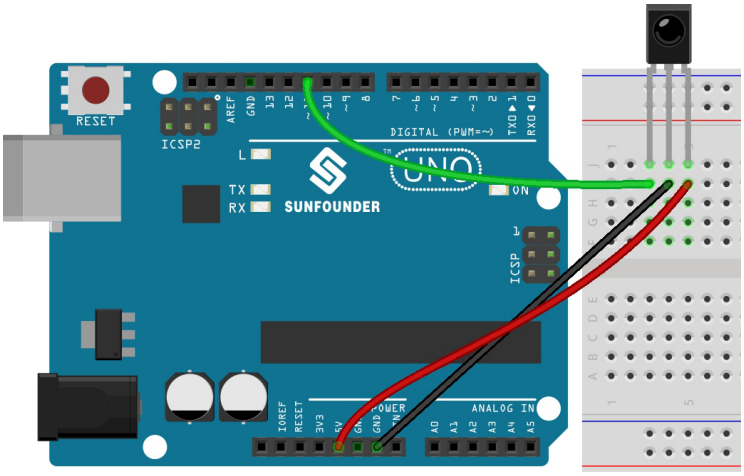
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| *IR* レシーバー | - |

**回路図**




**配線図**

この例では、IR レシーバーの左のピンをピン 11 に、中央のピンを GND に、右のピンを 5V に接続します。

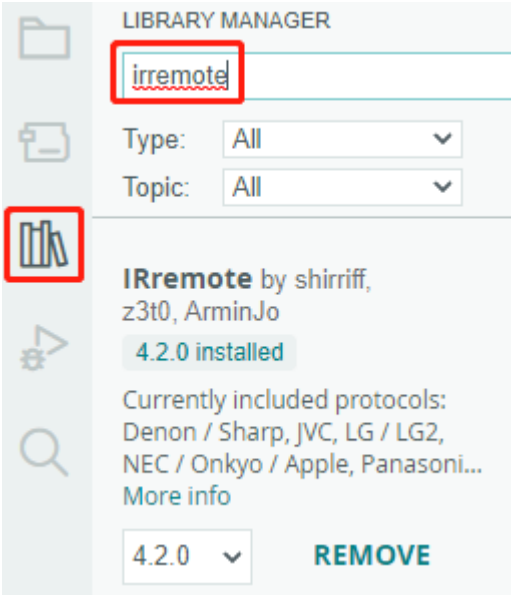



**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.11.ir_receiver のパスの下にある 5.11.ir_receiver.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- ここでは IRremote ライブラリが使用されています。 **Library Manager** からインストールできます。




---

コードを R3 ボードにアップロードすると、IR リモートコントローラの押されたボタンの現在の値がシリアルモニタに表示されます。

**どのように動作するのか？**

このコードは、 IRremote ライブラリを使用して赤外線（IR）リモコンとともに動作するように設計されています。詳細は以下のとおりです：

1. ライブラリのインクルード： IRremote ライブラリを含めます。これは、IR リモコンと連携するための関数を提供します。

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

2. IR センサーの信号ピンが接続されている Arduino のピンを定義し。

```
const int IR_RECEIVE_PIN = 11;  // IR センサーのピン番号を定義する。
```

3. ボーレート 9600 でシリアル通信を初期化します。指定されたピン (IR_RECEIVE_PIN) で IR レシーバを初期化し、LED フィードバックを有効にします (該当する場合)。

```
void setup() {
    Serial.begin(9600);                                    // ボーレート 9600 でシリアル通信を開始する。
    IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);  // IR レシーバーの起動
}
```

4. ループは、入力された IR リモコンの信号を継続的に処理します。

```
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
    }
}
```

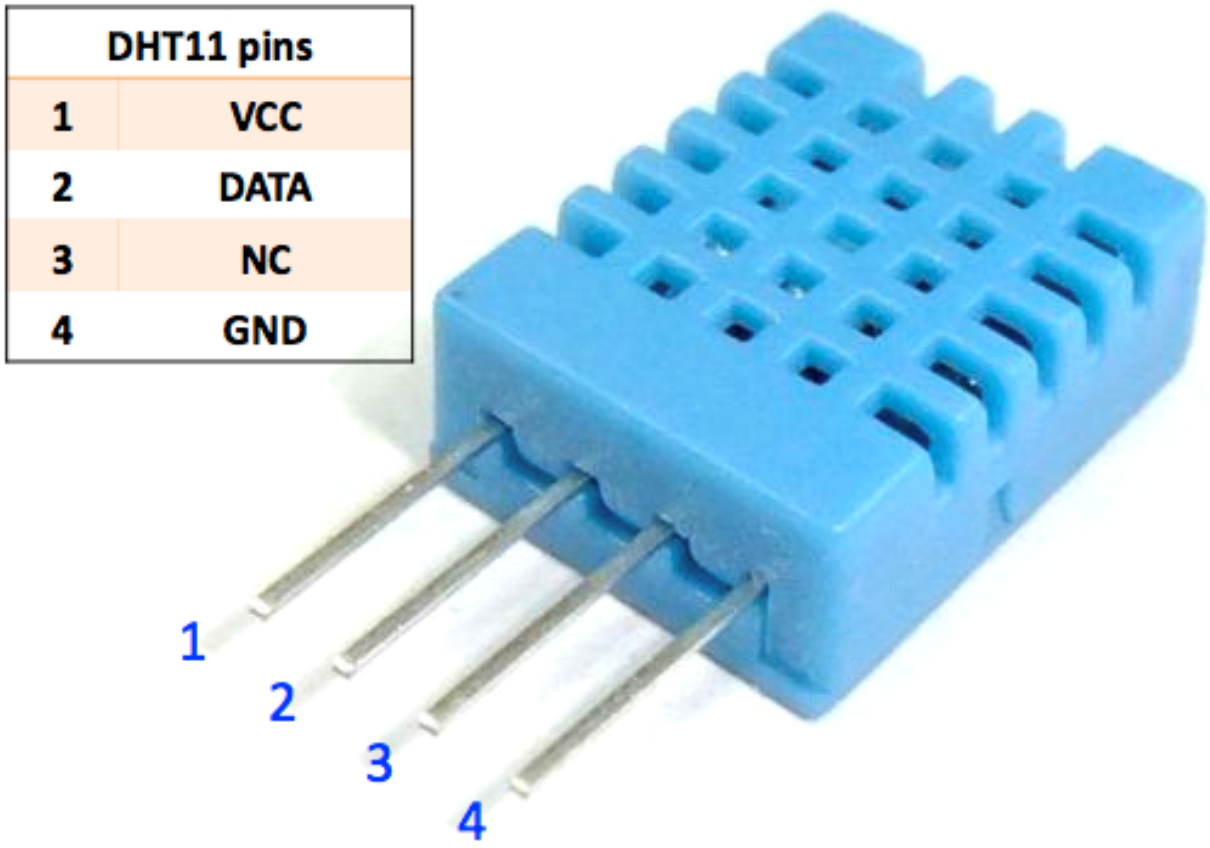
- IR 信号が受信され、正常にデコードされたかどうかを確認します。
- IR コマンドをデコードし、カスタムの decodeKeyValue() 関数を使用して decodedValue に保存します。
- デコードされた値がエラーでないかを確認します。
- デコードされた IR 値をシリアルモニタに印刷します。
- 次の信号の IR 信号受信を再開します。

### 5.11.3 温度・湿度

湿度と温度は物理的な数量自体から実際の人々の生活まで密接に関連しています。人間の環境の温度と湿度は、人体の体温調節機能と熱伝達効果に直接影響します。さらに、思考活動と精神状態に影響を与え、私たちの学習や仕事の効率に影響を与えます。

温度は国際単位系における 7 つの基本的な物理量の 1 つで、物体の熱さや寒さの度合いを測定するために使用されます。摂氏は、世界でよく使われる温度の尺度の一つで、"℃"の記号で表されます。

湿度は、空気中に存在する水蒸気の濃度です。日常生活では空気の相対湿度が一般的に使用され、%RH で表されます。相対湿度は温度と密接に関連しています。一定の体積の密封されたガスの場合、温度が高いほど相対湿度が低く、温度が低いほど相対湿度が高くなります。




このキットには、デジタル温度・湿度センサーである dht11 が含まれています。このセンサーは、周囲の空気を測定するための容量性湿度センサーとサーミスターを使用し、データピンにデジタル信号を出力します。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

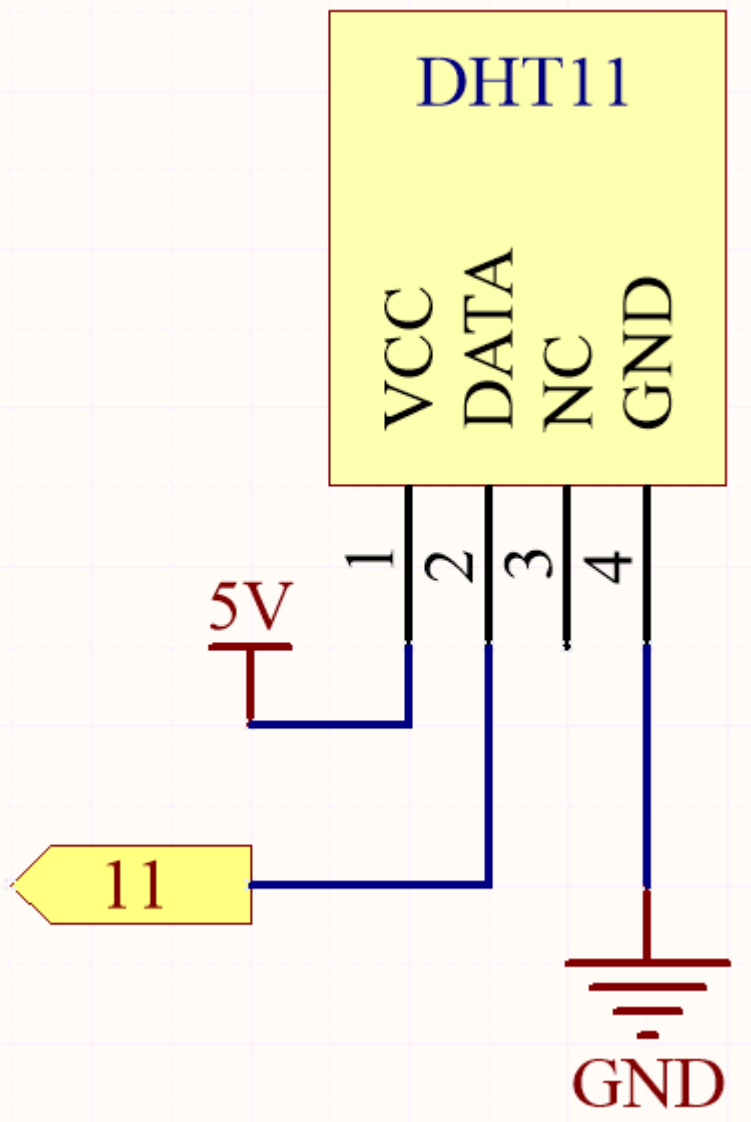
キット全体を購入するのは非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

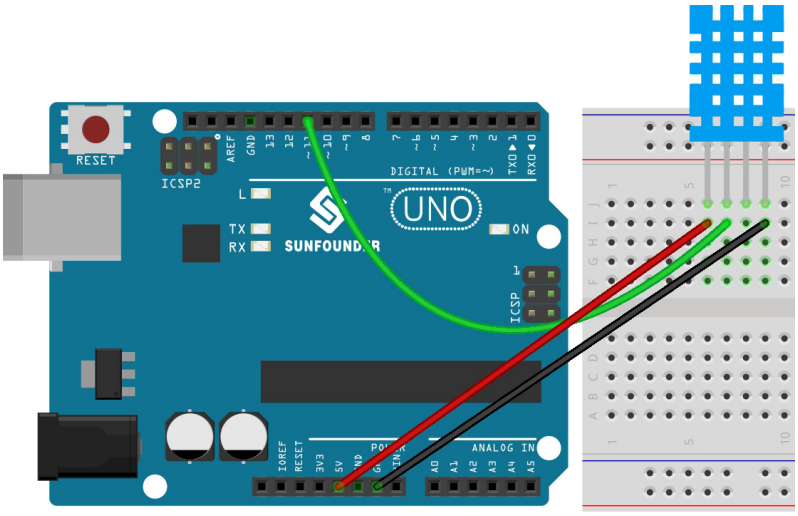
以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| *DHT11* 湿度温度センサ | - |

**回路図**


DHT11
VCC
DATA
NC
GND
1
2
3
4
5V
11
GND

**配線図**

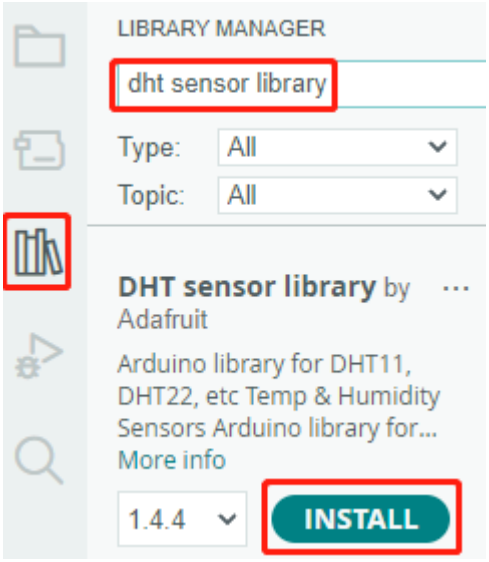

RESET
ICSP2
DIGITAL (PWM=~)
UNO
ON
TX
RX
SUNFOUNDER
ICSP
ANALOG IN

**コード**

---

**注釈:**

- 3in1-kit\basic_project\5.11.temperature_humidity の パ ス の 下 で 5.11.temperature_humidity.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- ここでは DHT sensor library が使用されています。 **Library Manager** からインストールできます。




---

コードが正常にアップロードされると、シリアルモニタに継続的に温度と湿度が出力されます。プログラムが安定して動作すると、これらの 2 つの値はますます正確になります。

**どのように動作するのか？**

1. DHT.h ライブラリをインクルードします。これは DHT センサーと対話するための関数を提供します。次に、DHT センサーのピンとタイプを設定します。

```
#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. 115200 のボーレートでシリアル通信を初期化し、DHT センサーを初期化します。

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

3. loop() 関数で、DHT11 センサから温度と湿度の値を読み取り、シリアルモニタに出力します。

```
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");
}
```

- dht.readHumidity() 関数は、DHT センサから湿度の値を読むために呼び出されます。
- dht.readTemperature() 関数は、DHT センサから温度の値を読むために呼び出されます。
- isnan() 関数は、読み取りが有効かどうかを確認するために使用されます。湿度または温度の値が NaN（数値でない）の場合、センサーからの読み取りが失敗したことを示し、エラーメッセージが出力されます。

### 4.5.12 5.12 シリアルリード

`Serial.print()` 関数を使用するとき、これに気づいたかもしれません。印刷があるなら、読取りはありますか？シリアルモニターのテキストボックスは何に使われるのでしょうか？そう、推測通り、シリアルモニタのテキストボックスに情報を入力することで、プログラムや回路を制御する方法があります。

このプロジェクトでは、I2C LCD1602 を使用して、シリアルモニタに入力されたテキストを表示し、 `Serial.read()` の使用法を体験します。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

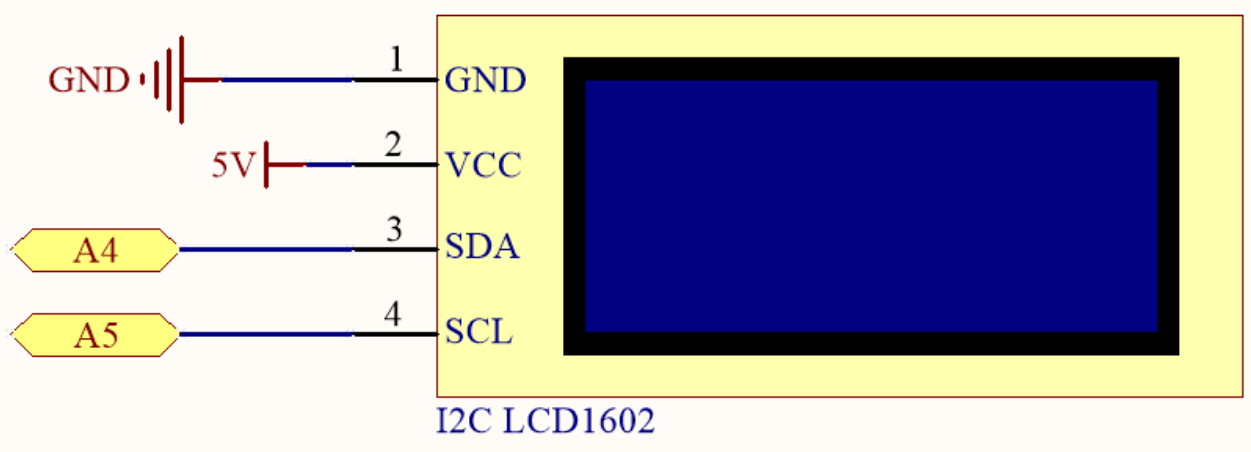
キットをまとめて購入するのはとても便利です、リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

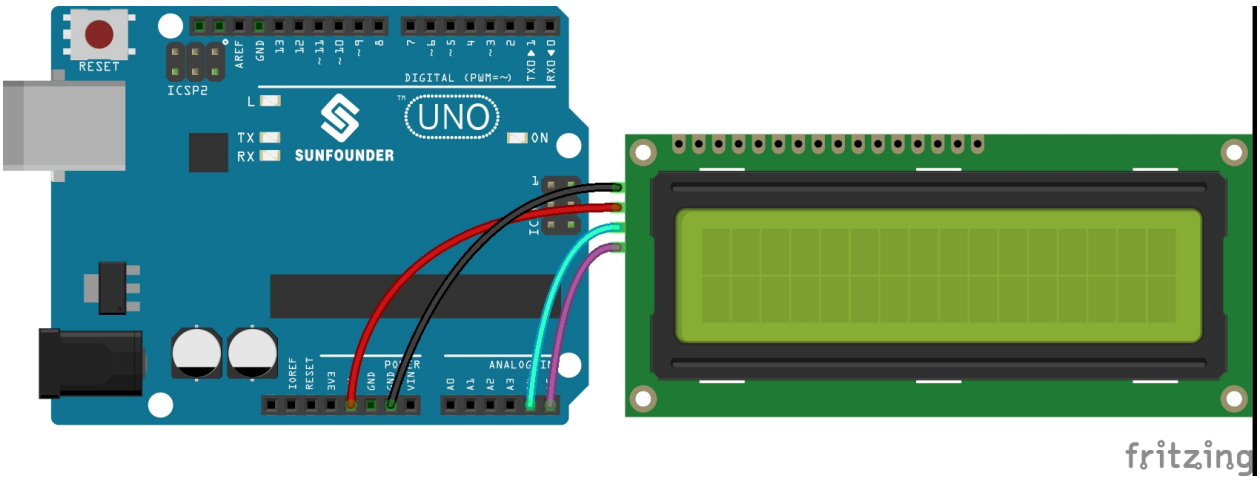
下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ジャンパーワイヤー** | |
| *I2C LCD1602* | |

**回路図**



**配線図**

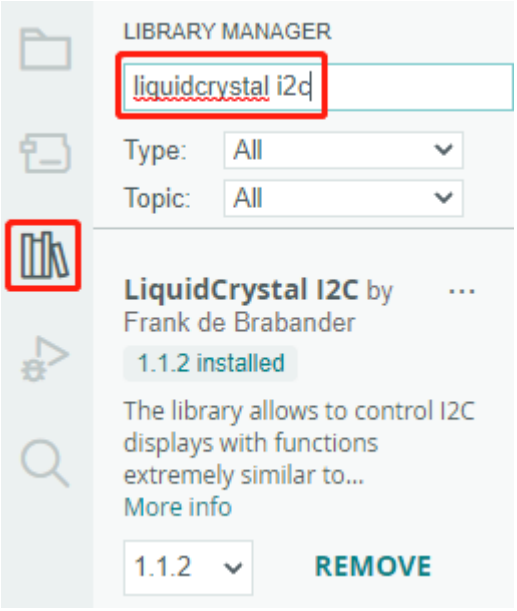



**コード**

---

**注釈:**

- `3in1-kit\basic_project\5.12.serial_read` のパスの下にある `5.12.serial_read.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- ここでは `LiquidCrystal I2C` ライブラリが使用されています。 **Library Manager** からインストールできます。




---

コードが正常にアップロードされたら、シリアルモニターのテキストボックスにテキストを入力することができ、

LCD に情報が表示されます。

**どのように動作するのか？**

```
void loop()
{
// シリアルポートから文字が届いたとき...
    if (Serial.available()) {
        // メッセージがすべて届くのを少し待つ
        delay(100);
        // 画面をクリア
        lcd.clear();
        // 利用可能なすべての文字を読み取る
        while (Serial.available() > 0) {
            // LCD に各文字を表示
            lcd.write(Serial.read());
        }
    }
}
```

- Serial.available() は、テキストボックスから何かを入力したときに、入ってくるストリームの文字の数を取得できます。入力には 2 つの終端記号があるため、 A を入力すると 3 文字、 AB を入力すると 4 文字が得られます。
- Serial.read() は、入ってくるストリームから最初の文字を取得します。例えば、 AB を入力した場合、Serial.read() を一度だけ呼び出すと、文字 A が得られます。2 回目の呼び出しで B が得られます。3 回目と 4 回目の呼び出しで、2 つの終了記号が得られます。入力ストリームに利用可能な文字がない状態でこの関数を呼び出すとエラーになります。

要するに、上記の二つを組み合わせ、 while ループを使用して、入力されたすべての文字を毎回読み取ることが一般的です。

```
while (Serial.available() > 0) {
    Serial.print(Serial.read());
}
```

ちなみに、入ってくるストリームから文字を取得するために Serial.read() を使用しないと、入ってくるストリームの文字が重なり合ってしまいます。例えば、 A の後に AB を入力すると、入ってくるストリームは 7 文字を蓄積します。

### 4.5.13 5.13 割り込み

`delay()` をセンサーを使用するプロジェクトで使用すると、これらのセンサーをトリガーするときに、プログラムが何の効果も持たない可能性があります。これは、delay 文がプログラムの一時停止を引き起こし、プログラムがセンサーからメイン制御ボードに送信される信号を取得できないためです。

この場合、割り込みを使用できます。割り込みを使用すると、プログラムがパルスを逃さないようになります。

この章では、アクティブブザーとボタンを使用して、割り込みの使用過程を体験します。

`loop()` 関数では、 `delay(1000)` を使用して秒をカウントします。ボタンを使ってブザーを制御する部分を ISR に入れることで、delay の影響を受けずにタスクをスムーズに完了させることができます。

---

**注釈:** ISR は、他の関数にはない一部のユニークな制限を持つ特別な関数です。 ISR にはパラメータを持つことができず、何も返すべきではありません。一般的に、ISR はできるだけ短く、高速であるべきです。複数の ISR を使用するスケッチの場合、一度に 1 つだけ実行でき、他の割り込みは、それらが持っている優先順位に基づいて、現在のものが終了した後に実行されます。

---

#### 必要な部品

このプロジェクトでは、以下のコンポーネントが必要です。

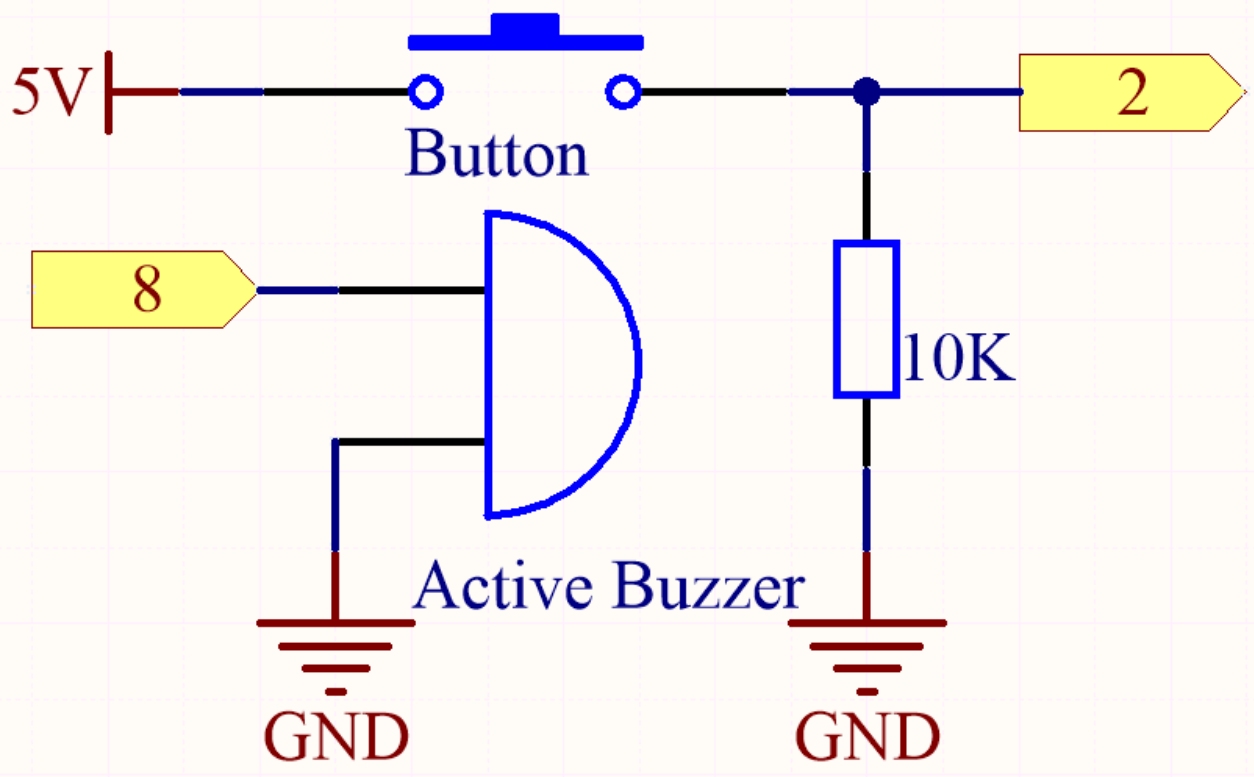
キット全体を購入すると非常に便利です、リンクはこちら:

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

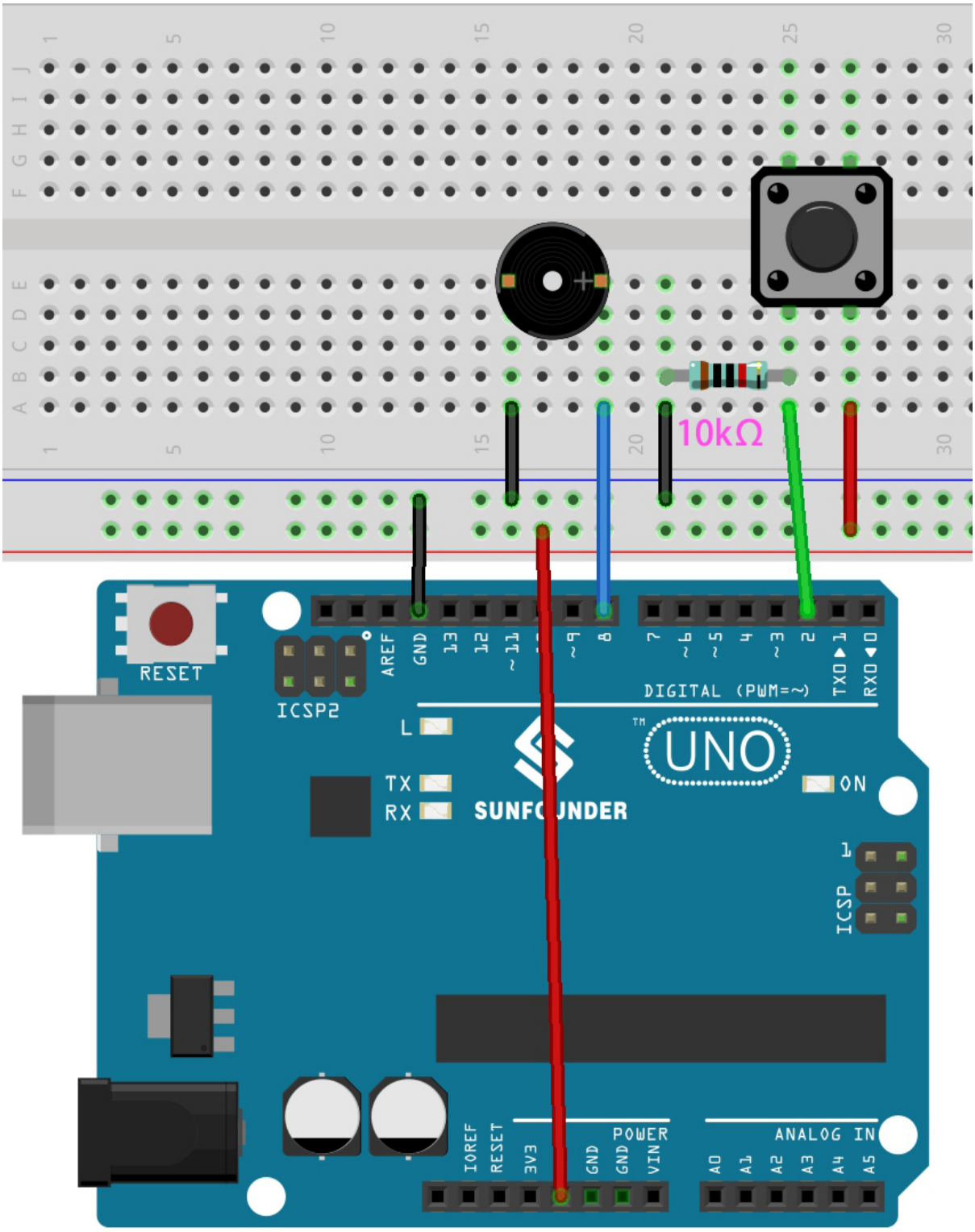
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ボタン | |
| ブザー | - |

**回路図**




**配線図**


10kΩ
RESET
ICSP2
AREF
GND
13
12
~11
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
GND
GND
VIN
A0
A1
A2
A3
A4
A5

**コード**

**注釈:**

- 3in1-kit\basic_project\5.13.interrupt のパスの下で 5.13.interrupt.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされると、シリアルモニタをオンにして、毎秒自動でインクリメントされる数字が表示されます。ボタンを押すと、ブザーが鳴ります。ボタンで制御されるブザー機能とタイミング機能は互いに競合しません。

**どのように動作するのか？**

- attachInterrupt(digitalPinToInterrupt(pin), ISR, mode): 割り込みを追加します。

  **文法**

  attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)

  **パラメータ**

  - pin: Arduino のピン番号。実際のデジタルピンを特定の割り込み番号に変換するために、digitalPinToInterrupt(pin) を使用する必要があります。例えば、ピン 3 に接続する場合、第一パラメータとしてその digitalPinToInterrupt(3) を使用します。
  - ISR: 割り込みが発生したときに呼び出す ISR。この関数はパラメータを取らず、何も返さない必要があります。この関数は、割り込みサービスルーチンとしても参照されることがあります。
  - mode: 割り込みがトリガされるタイミングを定義します。有効な値として 4 つの定数が事前に定義されています：
    - LOW は、ピンが低い場合に割り込みをトリガします。
    - CHANGE は、ピンの値が変わるたびに割り込みをトリガします。
    - RISING は、ピンが低から高になったときにトリガします。
    - FALLING は、ピンが高から低になったときにトリガします。

**注釈:** 異なるメイン制御ボードは割り込みピンを異なる方法で使用することができます。R3 ボードでは、ピン 2 とピン 3 だけが割り込みを使用することができます。

### 4.5.14 5.14 キャリブレーション

アナログ入力コンポーネント、例えばフォトレジスタや土壌湿度センサなどを使用する際、その読み取り範囲が 0 から 1023 ではなく、例えば 0 から 800 や 600 から 1000 のような範囲であることがわかるかもしれません。これは、これらのデバイスの限界に通常の使用で到達することができないためです。

このような場合、センサーの入力をキャリブレーションする技術が利用できます。起動時に、制御ボードはセンサーの読み取りを 5 秒間行い、最も高いおよび最も低い読み取りを記録します。この 5 秒間の読み取りは、サイクル中に取得される読み取りの最小および最大の予想値を示します。

このプロジェクトでは、上記のキャリブレーション技法を用いて、フォトレジスタとパッシブブザーを利用し、テレミン のようなゲームを実装します。

---

**注釈:** テレミン は物理的な接触を必要としない電子楽器で、プレイヤーの手の位置を感知して異なる音を生成します。

---

#### 必要な部品

このプロジェクトに必要な部品は以下のとおりです。

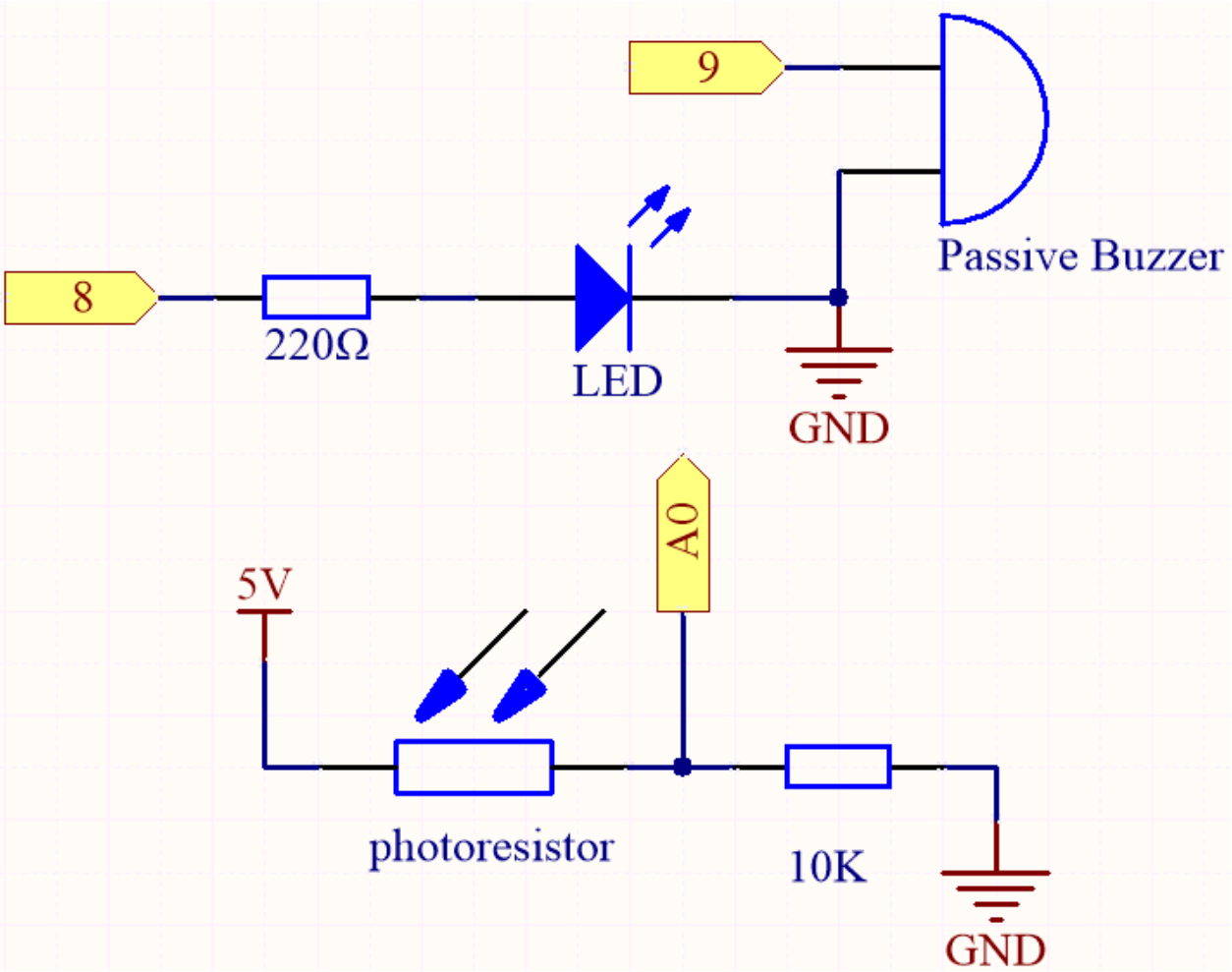
一式を購入すると非常に便利です。こちらがリンクです：

| 名称 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

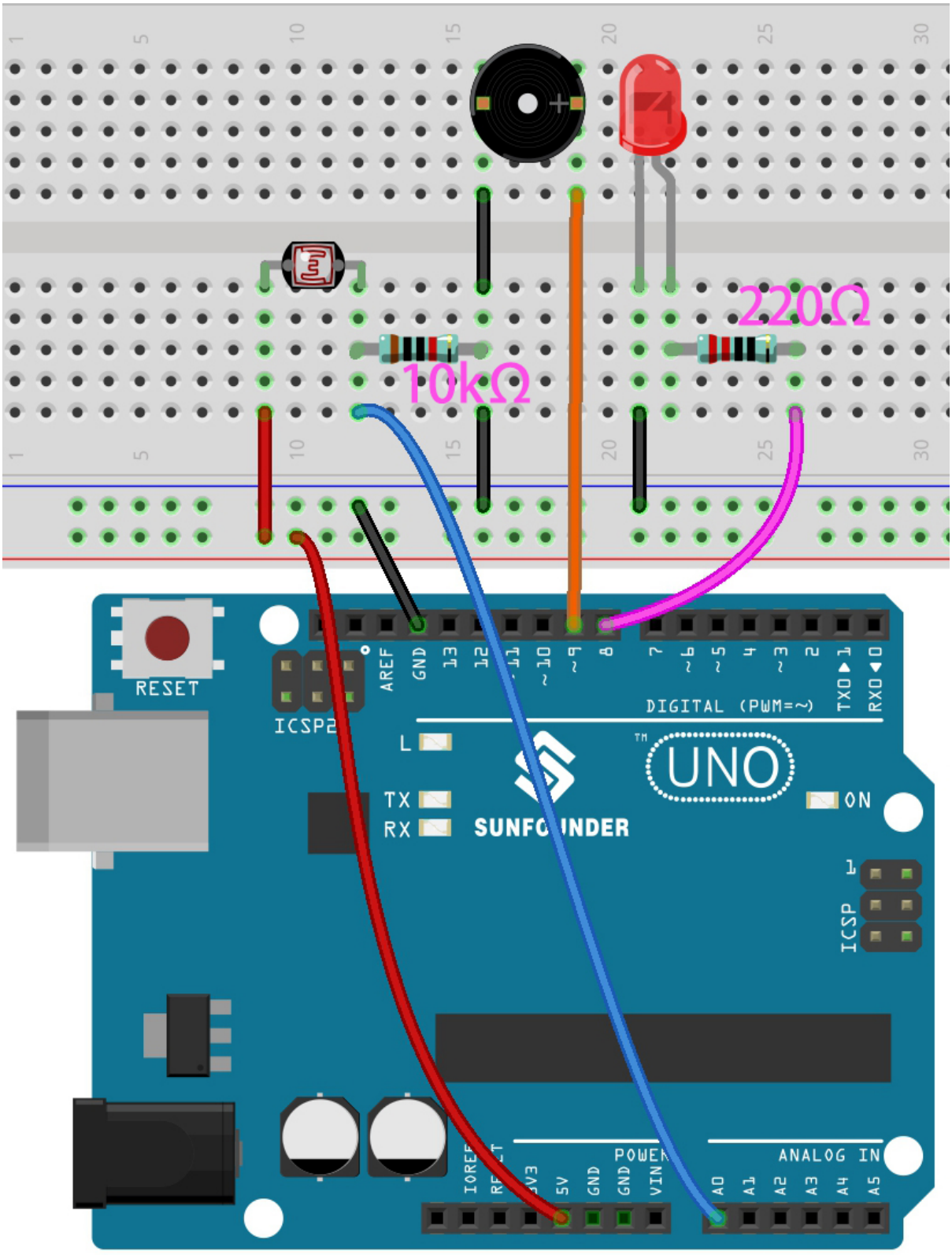
また、以下のリンクから各部品を個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ブザー | |
| *LED* | |
| フォトレジスタ | |

**回路図**




配線図


220Ω
10kΩ
RESET
ICSP2
DIGITAL (PWM=~)
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN

コード

**注釈:**

- 5.14.calibration.ino ファイルを 3in1-kit\basic_project\5.14.calibration のパスで開いてください。
- または、このコードを **Arduino IDE** にコピーしてください。
- あるいは、 Arduino Web Editor を利用してコードをアップロードしてください。

コードが正常にアップロードされた後、LED が点灯し、フォトレジスタの検出範囲をキャリブレーションするための 5 秒間が与えられます。これは、使用するたびに異なる光の環境下にある可能性があるためです（例：正午と夕暮れ時の光の強度は異なる）。

この時点で、フォトレジスタの上で手を上下に振る必要があり、手の動きの範囲がこの楽器の演奏範囲にキャリブレーションされます。

5 秒後、LED が消灯し、フォトレジスタ上で手を振って演奏できます。

**それはどのように動作しますか？**

1. すべてのコンポーネントの初期値とピンを設定します。

```
const int buzzerPin = 9;
const int ledPin = 8;
const int photocellPin = A0;  //フォトレジスタは A2 に接続

int lightLow = 1023;
int lightHigh = 0;

int sensorValue = 0;         // センサーから読み取った値
int pitch = 0;             // センサーの値を LED 'バー'に変換

unsigned long previousMillis = 0;
const long interval = 5000;
```

2. setup() でのキャリブレーションプロセスを設定します。

```
void setup()
{
    pinMode(buzzerPin, OUTPUT);// すべての LED ピンを出力に設定
    pinMode(ledPin, OUTPUT); //LED ピンを出力させる
```

(次のページに続く)

(前のページからの続き)

```
    /* フォトレジスタの最大値 & 最小値のキャリブレーション */
    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);
}
```

動作の流れは以下のとおりです。

- 5000ms の間隔で `millis()` を使用してタイミングを測定。

```
previousMillis = millis();
...
while (millis() - previousMillis <= interval) {
...
}
```

- この 5 秒間、フォトレジスタの周りで手を振ると、検出された光の最大値と最小値が記録され、それぞれ `lightHigh` と `lightLow` に割り当てられます。

```
sensorValue = analogRead(photocellPin);
if (sensorValue > lightHigh) {
    lightHigh = sensorValue;
}
if (sensorValue < lightLow) {
    lightLow = sensorValue;
}
```

3. これでテレミンの演奏を開始することができます。フォトレジスタの値を `sensorValue` に読み込み、小さな範囲から大きな範囲にマッピングして、ブザーの周波数として使用します。

```
void loop()
```

(次のページに続く)

(前のページからの続き)

```
{
/* play*/
sensorValue = analogRead(photocellPin); //A0 の値を読み取る
pitch = map(sensorValue, lightLow, lightHigh, 50, 6000);  // ブザーの周波数にマップ
する
if (pitch > 50) {
    tone(buzzerPin, pitch, 20);
}
delay(10);
}
```

### 4.5.15 5.15 EEPROM

EEPROM はメモリであり、メインコントロールボードをオフにしてもそのデータは消去されません。これを使用してデータを記録し、次回電源を入れたときにそれを読み取ることができます。

例として、毎日のロープスキップの回数を追跡するスポーツカウンターを作成することができます。

また、あるプログラムでデータを書き込み、別のプログラムでそれを読み取ることもできます。例えば、車のプロジェクトで作業しているとき、2 つのモーターの速度が一致しない場合、モーター速度の補正値を記録するキャリブレーションプログラムを書くことができます。

#### 必要な部品

このプロジェクトでは、以下の部品が必要です。

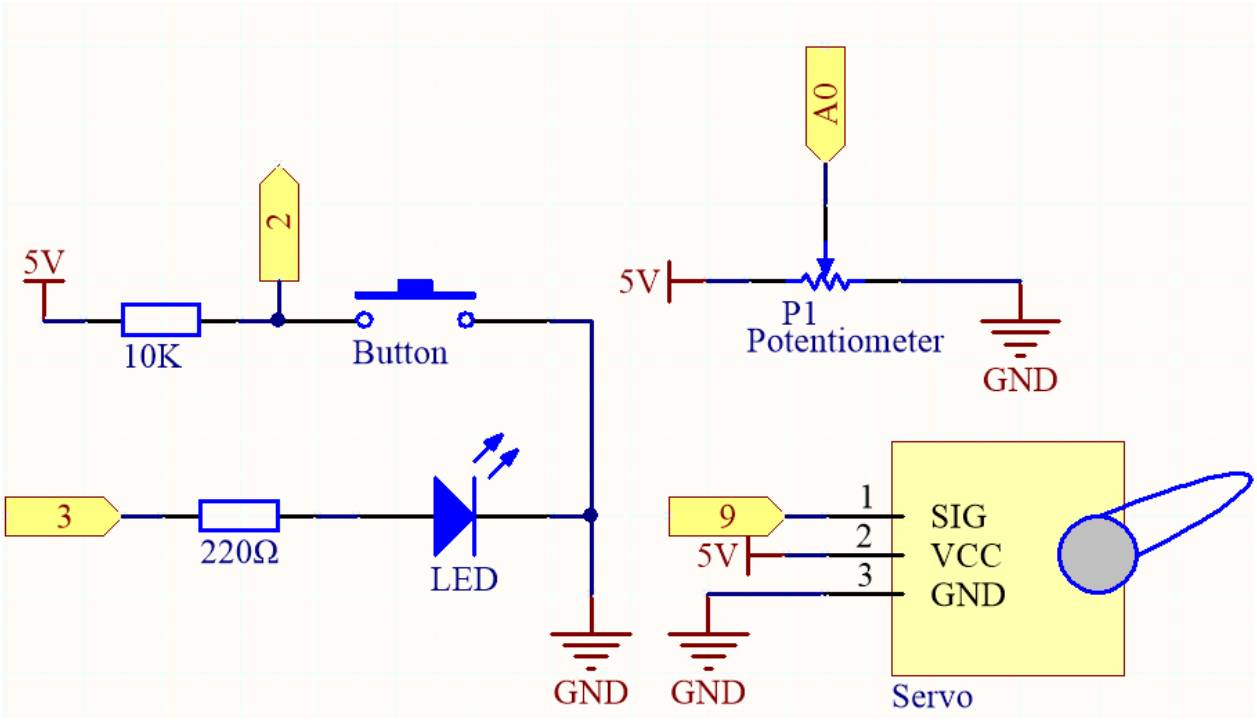
全てを含むキットを購入すると非常に便利です、こちらのリンクから購入できます：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

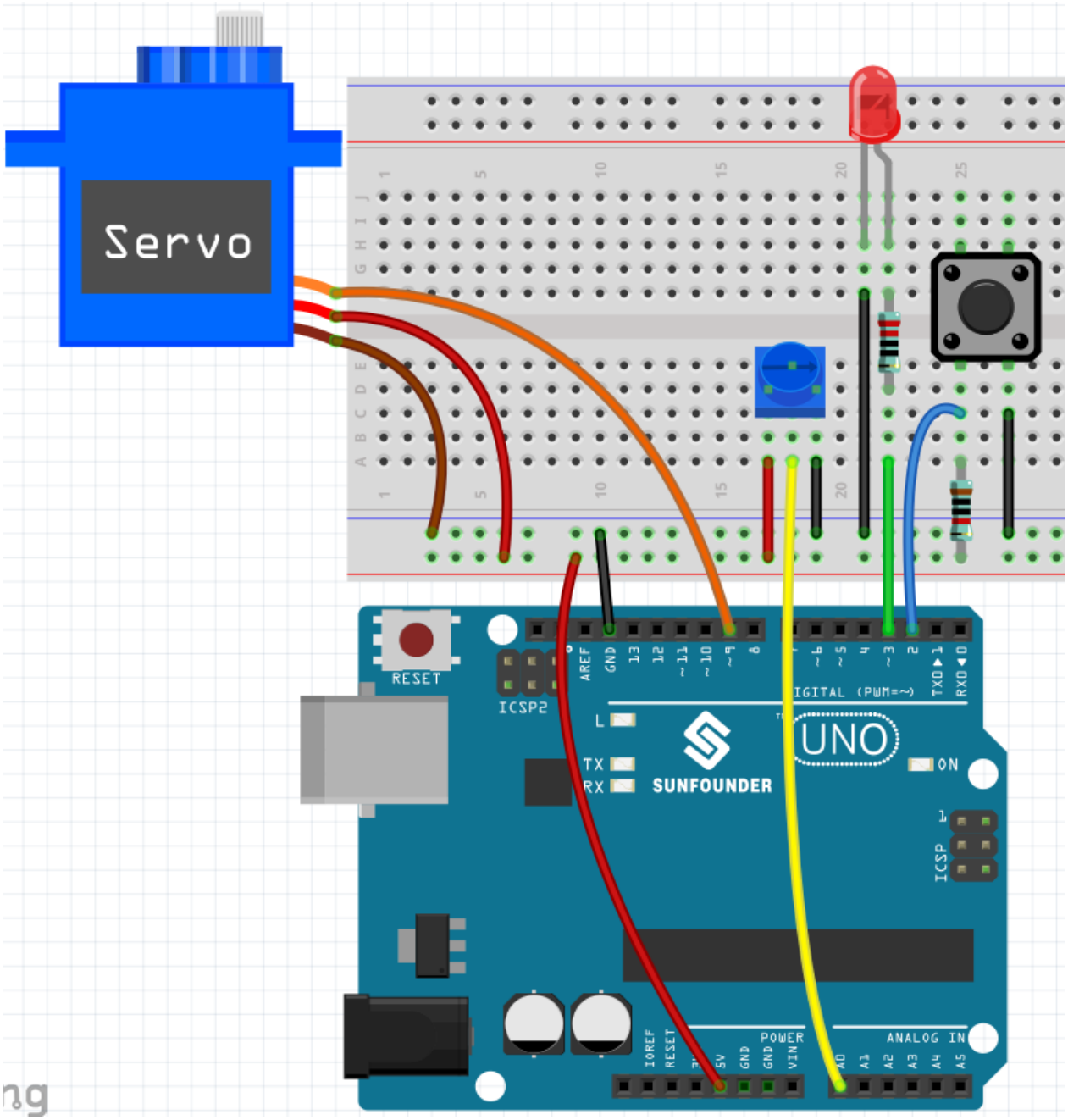
下記のリンクから別々に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| サーボ | |
| ボタン | |
| ポテンショメータ | |

**回路図**




**配線図**




**コード**

**注釈:**

- 3in1-kit\basic_project\5.15.eeproom のパスの下で 5.15.eeproom.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

この回路を使用するには、ボタンを押して記録を開始し、ポテンショメータを通じて所望の情報を入力するだけです。現在、ボードはあなたの操作を無限に繰り返します（そして、各反復ごとに LED が点滅します）。新しい操作を記録するためにボタンを再度押すまでです。resolution と recordTime の値を変更することで、記録される時間を変更することもできます。

**どのように動作しますか？**

1. EEPROM.h ライブラリをインポートし、EEPROM メモリを初期化します。

```
...
#include <EEPROM.h>//記録した値を保存するために使用


...
float resolution = 1000;//EEPROM.length() より小さくなければならない
float recordTime = 5; //遅延時間
bool recording = false;
...
```

/EEPROM.length() **より小さくなければならない** に注意してください。 setup() で、ボードの EEPROM のメモリが SunFounder R3 ボードの場合は 1024 と表示されます。別のボードを使用している場合は、変数 resolution の値を変更できます。

2. ボードの EEPROM メモリを印刷します。

```
void setup() {
    ...
    Serial.begin(9600);
    //Serial.println(EEPROM.length());
}
```

ボードの EEPROM メモリのサイズを知るには、Serial.println(EEPROM.read(i)) の行のコメントを解除してください。これにより、シリアルモニタに EEPROM のサイズが表示され、変数 resolution の値をそれに応じて変更できます。

3. ボタンが押されたことを検出すると、すぐに記録が開始され、ポテンショメータを使用して必要な情報が入力されます。その後、ボードはあなたのアクションを無限に繰り返します（各反復で LED が点滅します）。新しいアクションを記録するために再度ボタンを押すまでです。

```
void loop() {
    if (recording == true) {//記録
        for (int i = 1; i <= resolution; i++) {
            digitalWrite(ledPin, HIGH); //状態 LED の点灯
            int val = map(analogRead(A0), 0, 1023, 0, 180);
```

(次のページに続く)

(前のページからの続き)

```
            EEPROM.write(i, val);
            //Serial.println(EEPROM.read(i));
            myServo.write(val);
            delay(recordTime);
        }
        digitalWrite(ledPin, LOW); //状態 LED をオフ
        delay(1000);//人のための時間を与える
        recording = false;
    }
    else {
        for (int i = 1; i <= resolution; i++) {//再生
            if (digitalRead(buttonPin) == 0) {//再生を停止し、新しい値を記録
                recording = true;
                break;
            }
            int readval = EEPROM.read(i);
            myServo.write(readval);
            //Serial.println(readval);
            delay(recordTime);
        }
        digitalWrite(ledPin, HIGH); //新しい繰り返しを示す
        delay(100);
        digitalWrite(ledPin, LOW);
    }
}
```

- ボタンが押されたときに変数 recording を true にします。
- 変数 recording が true の場合、メモリ範囲にアクションの記録を開始します。
- ポテンショメータの値を読み取り、0-180 にマッピングして EEPROM に保存し、サーボの回転を制御します。
- LED は、記録の開始時に点灯し、終了時に消灯します。
- LED の速い点滅で記録されたアクションを繰り返し、新しい繰り返しのリマインダとして機能します。

4. EEPROM ライブラリについて。

   以下は、その関数の一部です。

   - write(address,value): EEPROM にバイトを書き込みます。

  - address: 書き込む場所、0 から始まる (int)
  - value: 書き込む値、0 から 255 まで (byte)
  - EEPROM への書き込みは 3.3ms で完了します。EEPROM メモリは、100,000 回の書き込み/消去サイクルの寿命が指定されているため、頻繁に書き込む場合は注意が必要です。

- Read(address): EEPROM からバイトを読み取ります。書き込まれていない場所の値は 255 です。
- update(address,value): EEPROM にバイトを書き込みます。値は、同じアドレスにすでに保存されているものと異なる場合のみ書き込まれます。
  - EEPROM への書き込みは 3.3ms で完了します。EEPROM メモリは、100,000 回の書き込み/消去サイクルの寿命が指定されているため、書き込むデータが頻繁に変更されない場合は、write() の代わりにこの関数を使用してサイクルを節約できます。
- EEPROM.put(address, data): 任意のデータタイプやオブジェクトを EEPROM に書き込みます。
  - address: 読み取る場所、0 から始まる (int)。
  - data: 読み取るデータ、プリミティブタイプ（例：float）やカスタム構造体。
  - この関数は EEPROM.update() を使用して書き込みを実行するため、値が変更されていない場合は再書き込みしません。
- EEPROM.get(address, data): EEPROM から任意のデータタイプやオブジェクトを読み取ります。
  - address: 読み取る場所、0 から始まる (int)。
  - data: 読み取るデータ、プリミティブタイプ（例：float）やカスタム構造体。

## 4.6 6. 面白いプロジェクト

この章では、多くのプログラムが現実とどのように対話するのかの基本的なロジックを示す例をいくつか紹介します。これにより、Arduino のプログラミングに慣れる手助けとなります。創造的なアイデアが頭に浮かんだとき、プログラミングはもはやあなたにとっての課題ではなくなります。

### 4.6.1 6.1 光によって制御される流れる LED

光抵抗またはフォトセルは、光によって変化する可変抵抗です。光抵抗の抵抗は、入射光の強度が増加すると減少します。言い換えれば、これは光電導を示しています。光抵抗は、光感応検出回路や、光と暗闇に反応するスイッチ回路に適用できます。

光抵抗の抵抗は、入射光の強度によって変化します。光の強度が高くなると、抵抗は減少し、光の強度が低くなると、抵抗は増加します。この実験では、光の強度を示すために 8 つの LED を使用します。光の強度が十分に高いと、すべての LED が点灯します。光がない場合、すべての LED は消えます。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

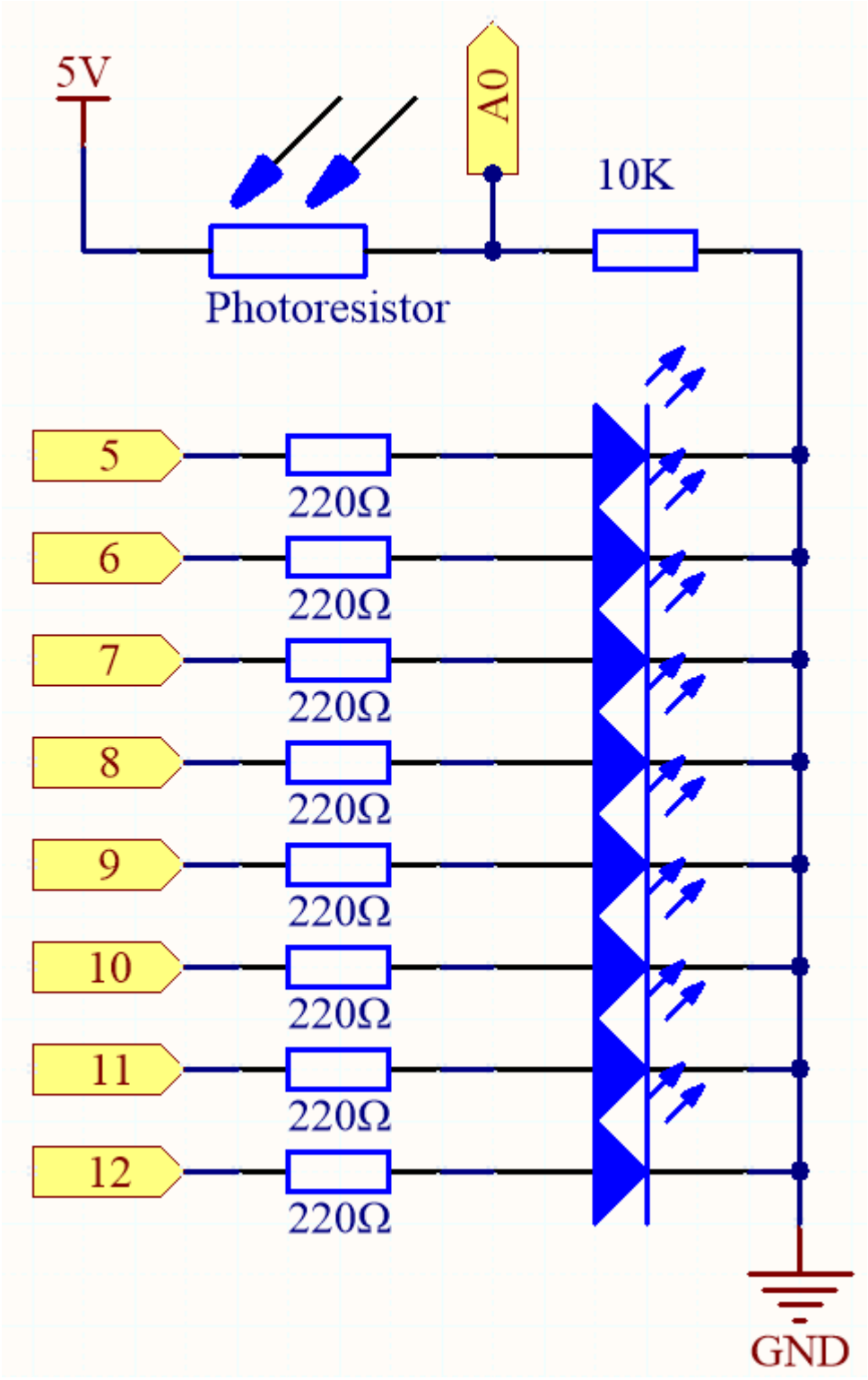
キット全体を購入するのは確かに便利です、リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

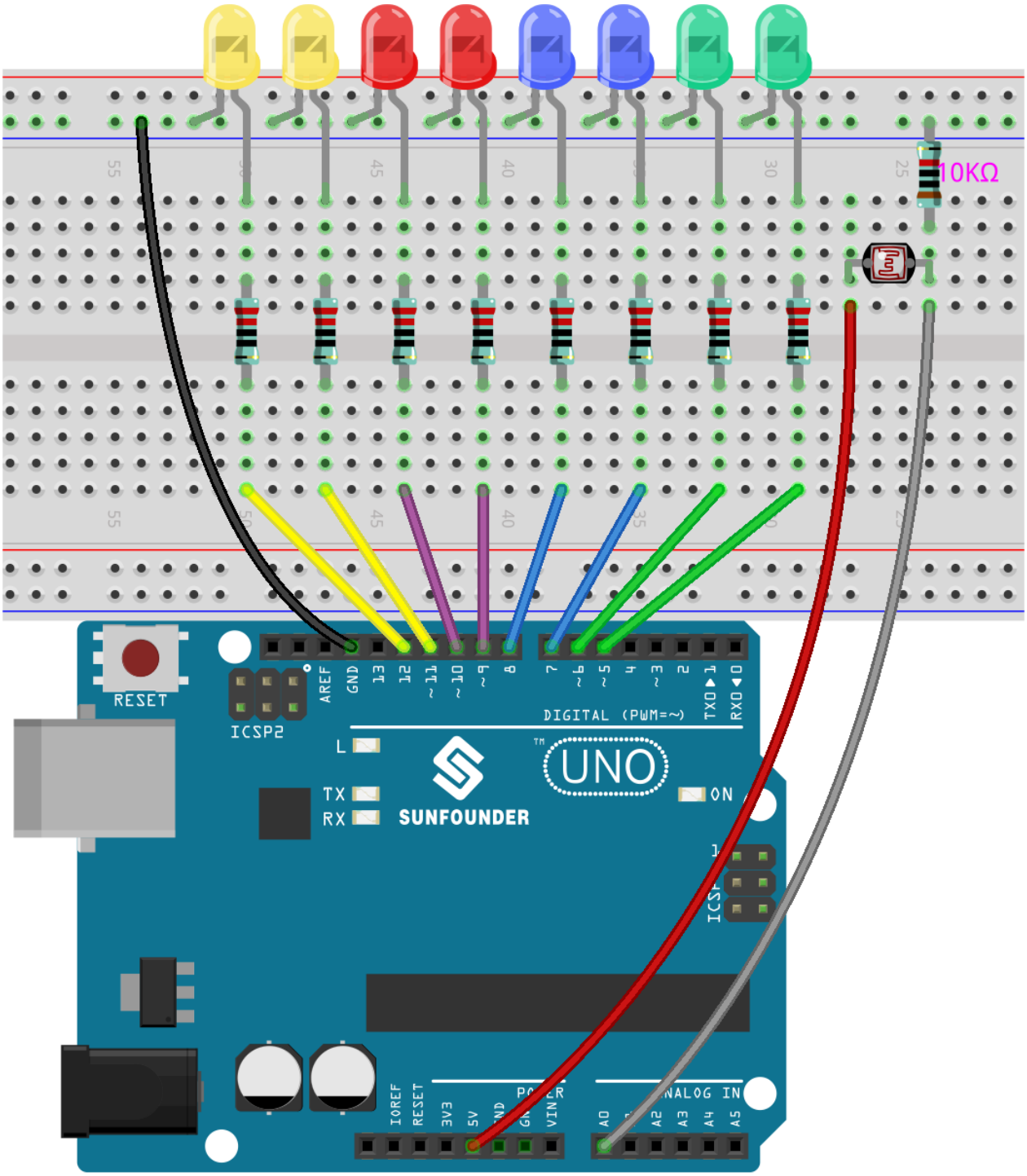
以下のリンクから、それぞれの部品も購入することができます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| **抵抗器** | |
| *LED* | |
| **フォトレジスタ** | |

**回路図**


5V
A0
10K
Photoresistor
5
6
7
8
9
10
11
12
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
GND

配線図




**コード**

---

**注釈:**

- 3in1-kit\basic_project\6.1.light_control_led のパスの下で 6.1.light_control_led.ino ファイルを開きます。

---

- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

光抵抗に光を当てると、いくつかの LED が点灯します。もっと光を当てると、さらに多くの LED が点灯します。暗い環境に置くと、すべての LED が消えます。

**どのように動作するのか？**

```
void loop()
{
    sensorValue = analogRead(photocellPin); // A0 の値を読む
    ledLevel = map(sensorValue, 300, 1023, 0, NbrLEDs); // LED の数にマップする
    for (int led = 0; led < NbrLEDs; led++)
    {
        if (led < ledLevel ) // led が ledLevel より小さい場合、以下のコードを実行します
        {
            digitalWrite(ledPins[led], HIGH); // レベルより低いピンをオンにする
        }
        else
        {
            digitalWrite(ledPins[led],LOW); // レベルより高いピンをオフにする
        }
    }
}
```

map() 関数を使用することで、光抵抗の値を 8 つの LED にマップすることができます。例えば、sensorValue が 560 の場合、ledLevel は 4 となり、この時点で ledPins[0] から ledPins[4] までが点灯し、ledPins[5] から ledPins[7] までが消灯することになります。

### 4.6.2 6.2 電子サイコロ

このプロジェクトでは、ボタン、7 セグメント、および 74hc595 を使用して電子サイコロを作成します。ボタンが押されるたびに、1 から 6 までのランダムな数字が生成され、7 セグメントディスプレイに表示されます。

**必要な部品**

このプロジェクトには以下の部品が必要です。

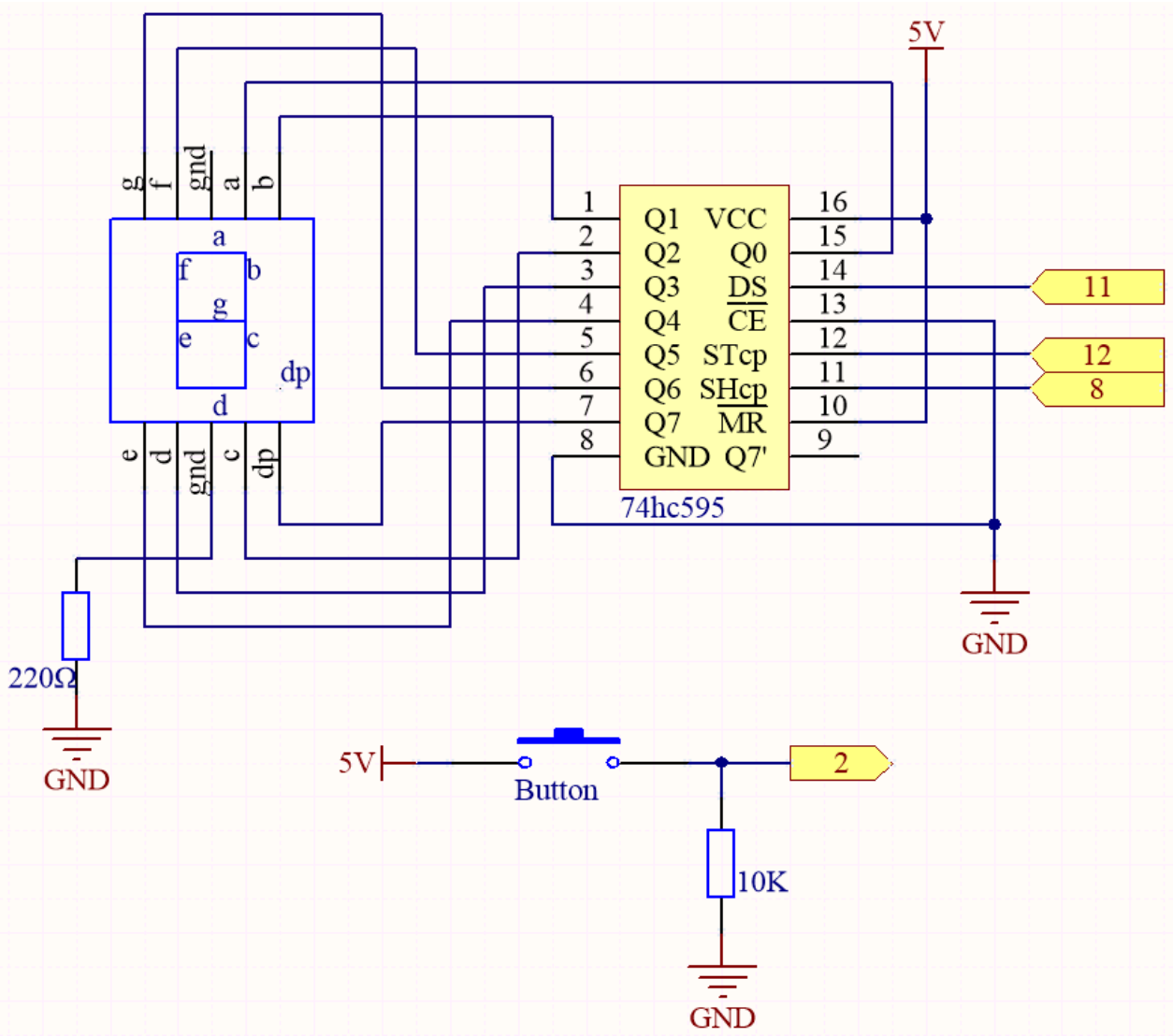
一式を購入するのは確かに便利です、リンクはこちら：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

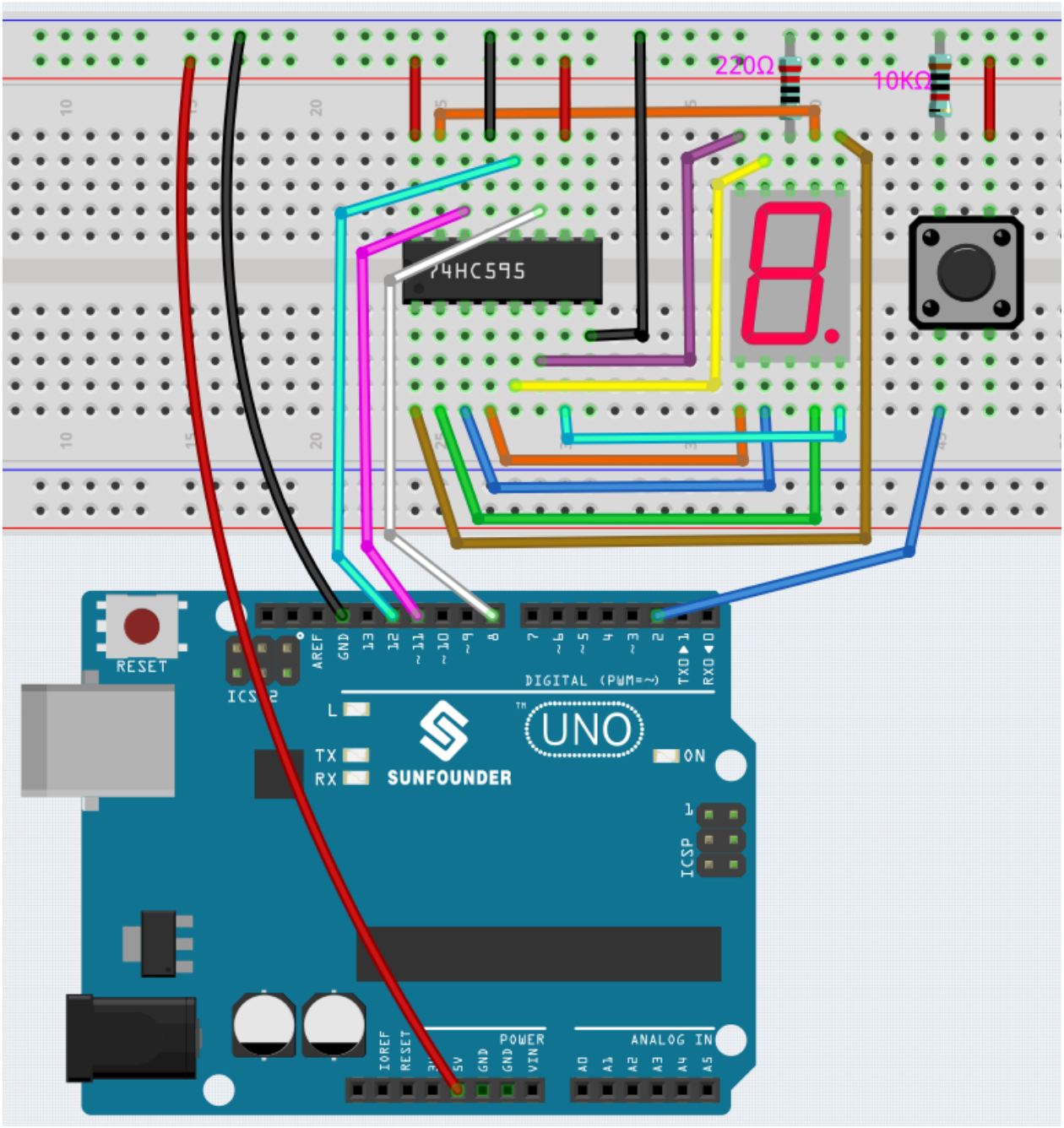
以下のリンクから部品を個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ボタン | |
| *74HC595* | |
| *7* セグメント表示 | |

**回路図**


5V
g
f
gnd
a
b
a
f
b
g
e
c
dp
d
e
d
gnd
c
dp
1
2
3
4
5
6
7
8
Q1
Q2
Q3
Q4
Q5
Q6
Q7
GND
VCC
Q0
DS
CE
STcp
SHcp
MR
Q7'
16
15
14
13
12
11
10
9
74hc595
11
12
8
GND
220Ω
GND
5V
Button
2
10K
GND

配線図




**コード**

---

**注釈:**

- `3in1-kit\basic_project\6.2.electronic_dice` のパスの下にある `6.2.electronic_dice.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。

- または、 Arduino Web Editor を通じてコードをアップロードします。

コードが正常にアップロードされると、7 セグメントディスプレイは 0-7 を高速でスクロールして表示し、ボタンを押すとランダムな数字が表示されてスクロールが停止します。ボタンを再度押すと、スクロール表示が再開されます。

**どのように動作するのか？**

このプロジェクトは *5.10 ShiftOut(7* **セグメント表示***)* を基にしており、7 セグメントディスプレイ上のスクロール表示を開始/一時停止するボタンが付いています。

1. 各ピンを初期化し、ボタンの値を読み取ります。

```
void setup ()
{

    ...
    attachInterrupt(digitalPinToInterrupt(buttonPin), rollDice, FALLING);
}
```

   - ここで割り込みは、ボタンの状態を読むために使用されます。 buttonPin のデフォルト値は低く、ボタンが押されると低から高に変わります。
   - rollDice は割り込みがトリガされたときに呼び出される関数を表しており、変数 state の値をトグルするために使用されます。
   - FALLING は、buttonPin が低から高に移行するときに割り込みがトリガされることを意味します。

2. 変数 state が 0 の場合、関数 showNumber() が呼び出され、7 セグメントディスプレイが 1 から 7 までのランダムな数字を表示します。

```
void loop()
{
    if (state == 0) {
        showNumber((int)random(1, 7));
        delay(50);
    }
}
```

3. rollDice() 関数について。

```
void rollDice() {
    state = !state;
```

(次のページに続く)

(前のページからの続き)

```
}
```

この関数が呼び出されると、state の値をトグルします。例えば、前回が 1 で、今回が 0 です。

4. showNumber() 関数について。

```
void showNumber(int num) {
    digitalWrite(STcp, LOW); //ST_CP を接地し、送信している間だけ Low を保持する
    shiftOut(DS, SHcp, MSBFIRST, datArray[num]);
    //ラッチ・ピンをハイ・レベルに戻し、チップにそれを知らせる
    //もはや情報をリッスンする必要はない
    digitalWrite(STcp, HIGH); //ST_CPST_CP をプルしてデータを保存する。
}
```

これはプロジェクト *5.10 ShiftOut(7 セグメント表示)* の loop() 内のコードを関数 showNumber() に移したものです。

### 4.6.3 6.3 高温度警報

次に、サーミスター、プッシュボタン、ポテンショメータ、そして LCD を使用して、高温度警報装置を作成します。LCD1602 は、サーミスターによって検出された温度と、ポテンショメータを使用して調整できる高温度閾値を表示します。閾値は同時に EEPROM に保存されるため、現在の温度が閾値を超えると、ブザーが鳴ります。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

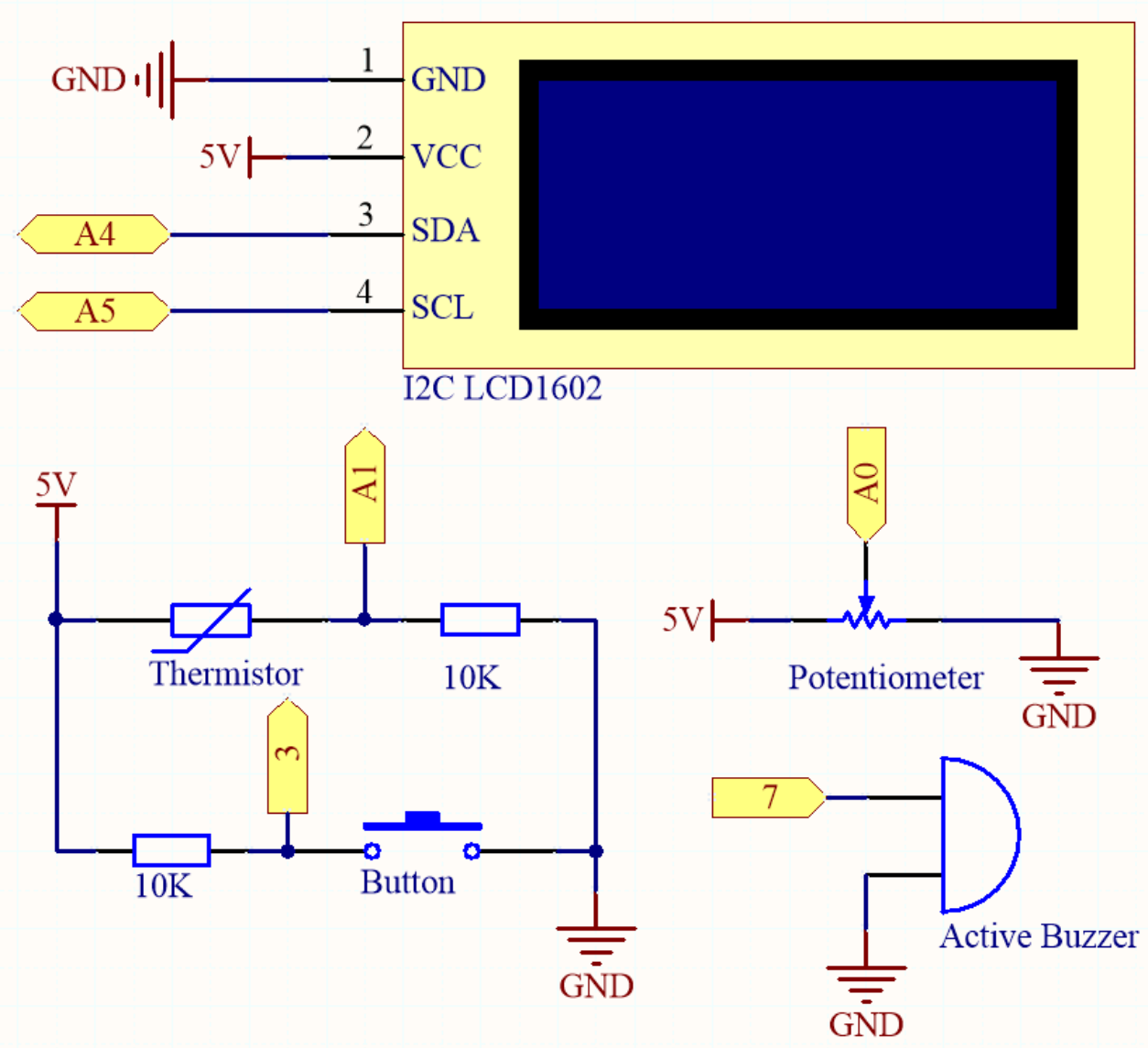
キット全体を購入することは確かに便利です、リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

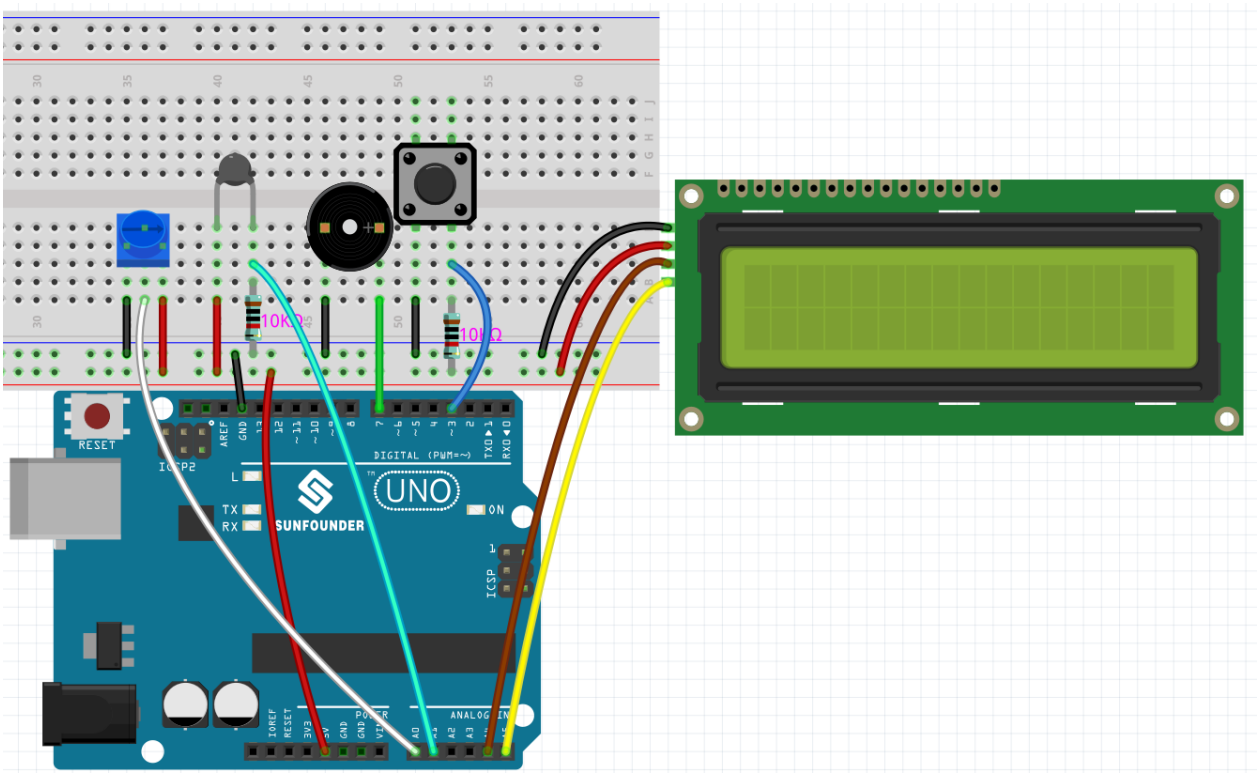
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ブザー | - |
| ボタン | |
| *I2C LCD1602* | |
| サーミスタ | |
| ポテンショメータ | |

**回路図**


GND
1
GND
5V
2
VCC
A4
3
SDA
A5
4
SCL
I2C LCD1602
5V
A1
Thermistor
10K
3
10K
Button
GND
A0
5V
Potentiometer
GND
7
Active Buzzer
GND

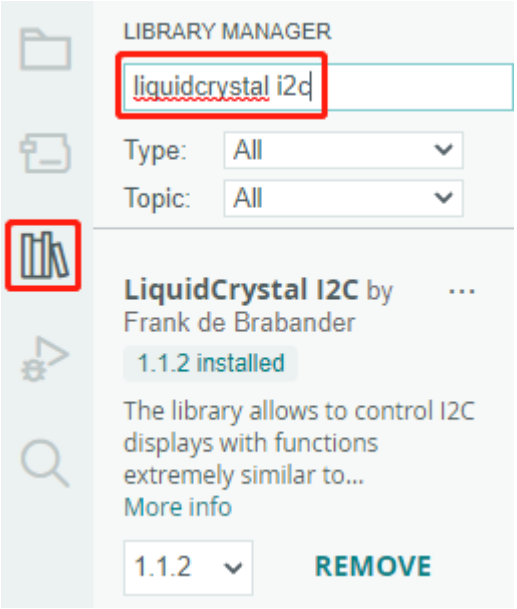
配線図




**コード**

**注釈:**

- 3in1-kit\basic_project\6.3.high_tem_alarm のパスの下にある 6.3.high_tem_alarm.ino ファイルを直接開くことができます。
- または、このコードを Arduino IDE 1/2 にコピーします。
- ここでは LiquidCrystal I2C ライブラリが使用されています。 **Library Manager** からインストールできます。




コードのアップロードが成功すると、LCD1602 はサーミスターによって検出された温度と高温度の閾値を表示します。閾値はポテンショメータで調整することができます。閾値は同時に EEPROM に保存されるため、現在の温度が閾値を超えると、ブザーが鳴ります。

**注釈:** コードと配線が正しい場合でも、LCD がコンテンツを表示しない場合は、背面のポテンショメータを回してください。

**どのように動作しますか？**

1. ボタン、ブザー、I2C LCD1602 を初期化し、EEPROM の値を読み取ります。ボタンのステータスもここで読み取るために割り込みを使用しています。

```
void setup()
{
    pinMode(buzzerPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    lcd.init();
    lcd.backlight();
    upperTem = EEPROM.read(0);
    delay(1000);
    attachInterrupt(digitalPinToInterrupt(buttonPin), buttonState, FALLING);
}
```

   - この割り込みはボタンの状態を読み取るために使用されます。ボタンが押されると、 buttonPin

はローよりハイに変わります。

- 割り込みがトリガされると、buttonState 関数が呼び出され、変数 state の値を切り替えます。
- FALLING は、buttonPin がローからハイに移行するときに割り込みが発生することを意味します。

2. 高温閾値を設定するには、メインプログラム内で state が 1 のとき（ボタンを押すと 0 と 1 の間で切り替えられる）に upperTemSetting() 関数が呼び出され、それ以外の場合は monitoringTemp() が現在の温度と設定された閾値を表示するために呼び出されます。

```
void loop()
{
    if (state == 1)
    {
        upperTemSetting();
    }
    else {
        monitoringTemp();
    }
}
```

3. upperTemSetting() 関数について。

```
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();
```

(次のページに続く)

(前のページからの続き)

```
            return;
        }
    }
}
```

- この関数を使用して閾値を設定することができます。この関数に入ると、LCD1602 に現在の閾値が表示され、ポテンショメータを使用してこの閾値を変更することができます。この閾値は EEPROM に保存され、ボタンが再び押されると終了します。

4. monitoringTemp() 関数について。

```
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   ");
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);
}
}
```

- この関数を使用すると、温度を表示し、アラームを設定することができます。
- サーミスタの値は読み取られ、その後、式によって摂氏温度に変換され、LCD1602 に表示されます。
- 設定した閾値も LCD に表示されます。
- 現在の温度が閾値を超えると、ブザーがアラームを鳴らします。

### 4.6.4 6.4 駐車補助

科学技術の発展に伴い、多くのハイテク製品が車に取り付けられています。その中で、バックアップアシストシステムはその一つです。このプロジェクトでは、超音波モジュール、LCD、LED、およびブザーを使用して、シンプルな超音波駐車補助システムを作成します。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

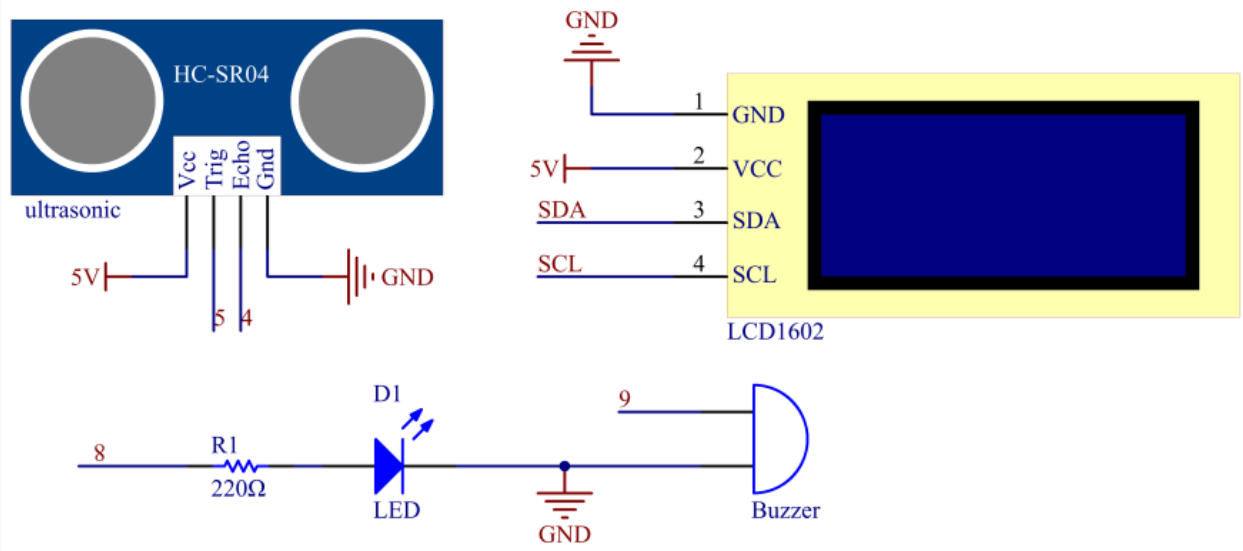
一式を購入するのが便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

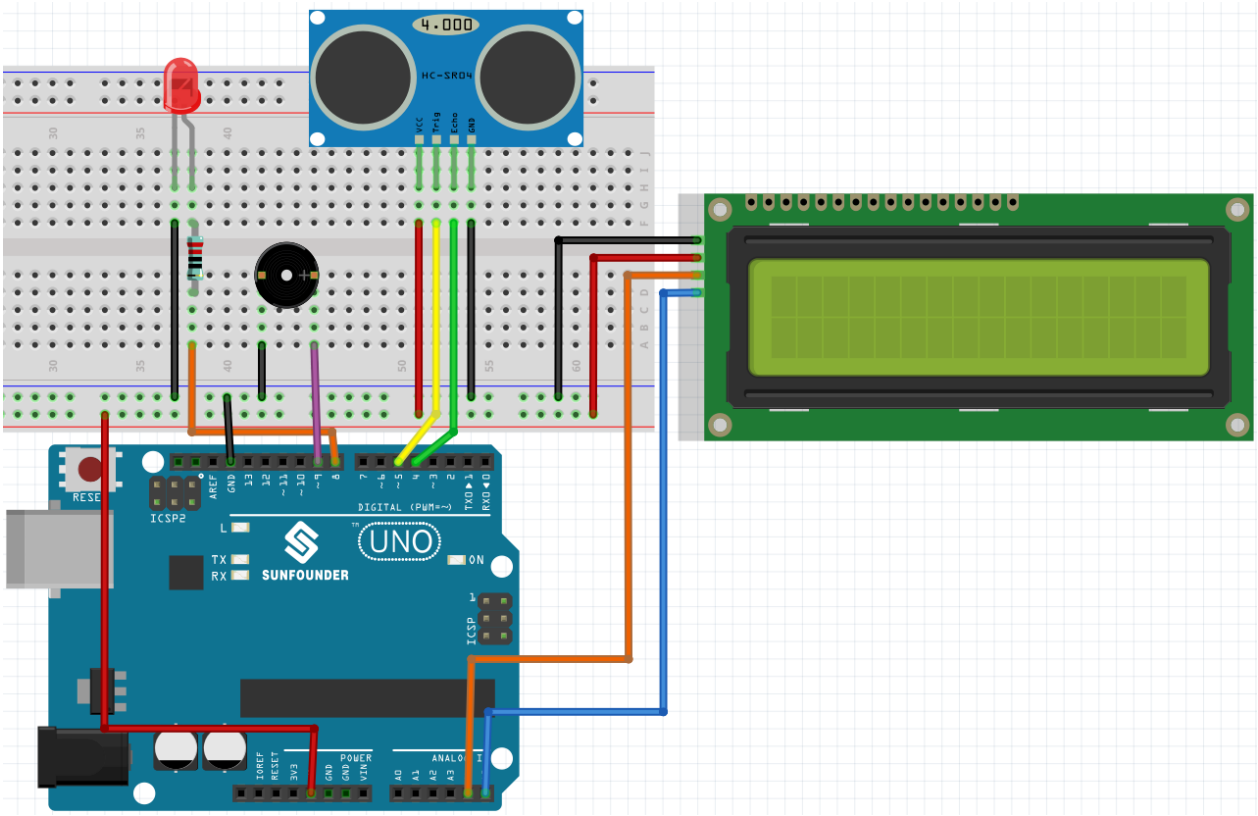
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| ブザー | |
| *I2C LCD1602* | |
| 超音波モジュール | |

**回路図**



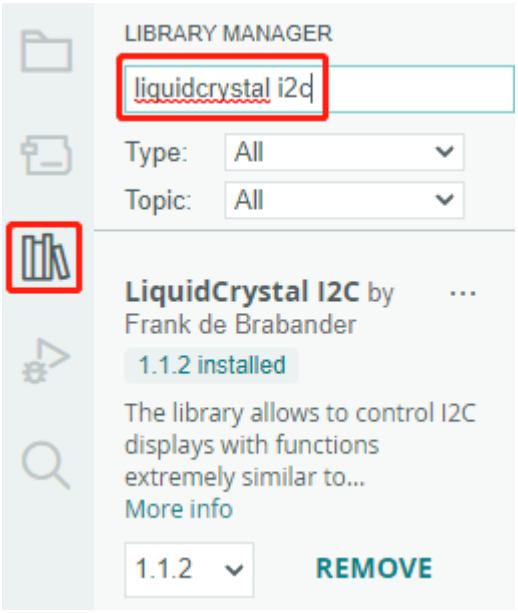

**配線図**




**コード**

**注釈:**

- `3in1-kit\basic_project\6.4_reversingAid` のパスの下の `6.4_reversingAid.ino` ファイルを直接開くことができます。
- または、このコードを Arduino IDE 1/2 にコピーします。
- ここでは `LiquidCrystal I2C` ライブラリが使用されています。 **Library Manager** からインストールできます。




コードのアップロードが成功すると、現在検出された距離が LCD に表示されます。そして、ブザーは異なる距離に応じて音の周波数を変えます。

**注釈:** コードと配線が正しいのに、LCD に内容が表示されない場合は、裏側のポテンショメーターを回してみてください。

**どのように動作するのか？**

このコードを使用すると、オブジェクト間の距離を測定し、LCD ディスプレイとブザーを通じてフィードバックを提供するシンプルな距離測定装置を作成できます。

`loop()` 関数はプログラムの主要なロジックを含み、連続して実行されます。 `loop()` 関数について詳しく見てみましょう。

1. 距離を読み取り、パラメータを更新するループ

   `loop` 内で、コードはまず超音波モジュールで測定された距離を読み取り、その距離に基づいてインターバルパラメータを更新します。

```
// 距離を更新
distance = readDistance();

// 距離に基づいてインターバルを更新
if (distance <= 10) {
    intervals = 300;
```

(次のページに続く)

(前のページからの続き)

```
} else if (distance <= 20) {
    intervals = 500;
} else if (distance <= 50) {
    intervals = 1000;
} else {
    intervals = 2000;
}
```

2. ビープ音を鳴らすタイミングを確認

   コードは現在の時間と前回のビープ音の時間の差を計算し、その差がインターバル時間以上であれば、ブザーを鳴らして前回のビープ音の時間を更新します。

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

3. LCD ディスプレイを更新

   コードは LCD ディスプレイをクリアし、最初の行に"Dis:"と現在の距離（センチメートル）を表示します。

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

delay(100);
```

### 4.6.5 6.5 反応時間ゲーム

私たちの体は、オーディオ RT、ビジュアル RT、タッチ RT など、多くの反応時間を持っています。

反応時間は、運転中の通常よりも遅い反応時間が重大な結果を招くことがあるなど、私たちの日常生活に多くの影響を及ぼします。

このプロジェクトでは、3 つのボタンと 2 つの LED を使用して、私たちの視覚的な反応時間を測定します。

Arduino のシリアルモニターには「waiting...」というメッセージが表示されます。Ready ボタンを押すと、2 つの LED のうちの 1 つがランダムな時間間隔後にランダムに点灯する必要があります。テスターができるだけ早く対応するボタンを押すことが重要です。Arduino は、LED が点灯したときと、人が対応するボタンを押したときとの時間差を記録し、Arduino のシリアルモニターに測定された応答時間を表示します。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

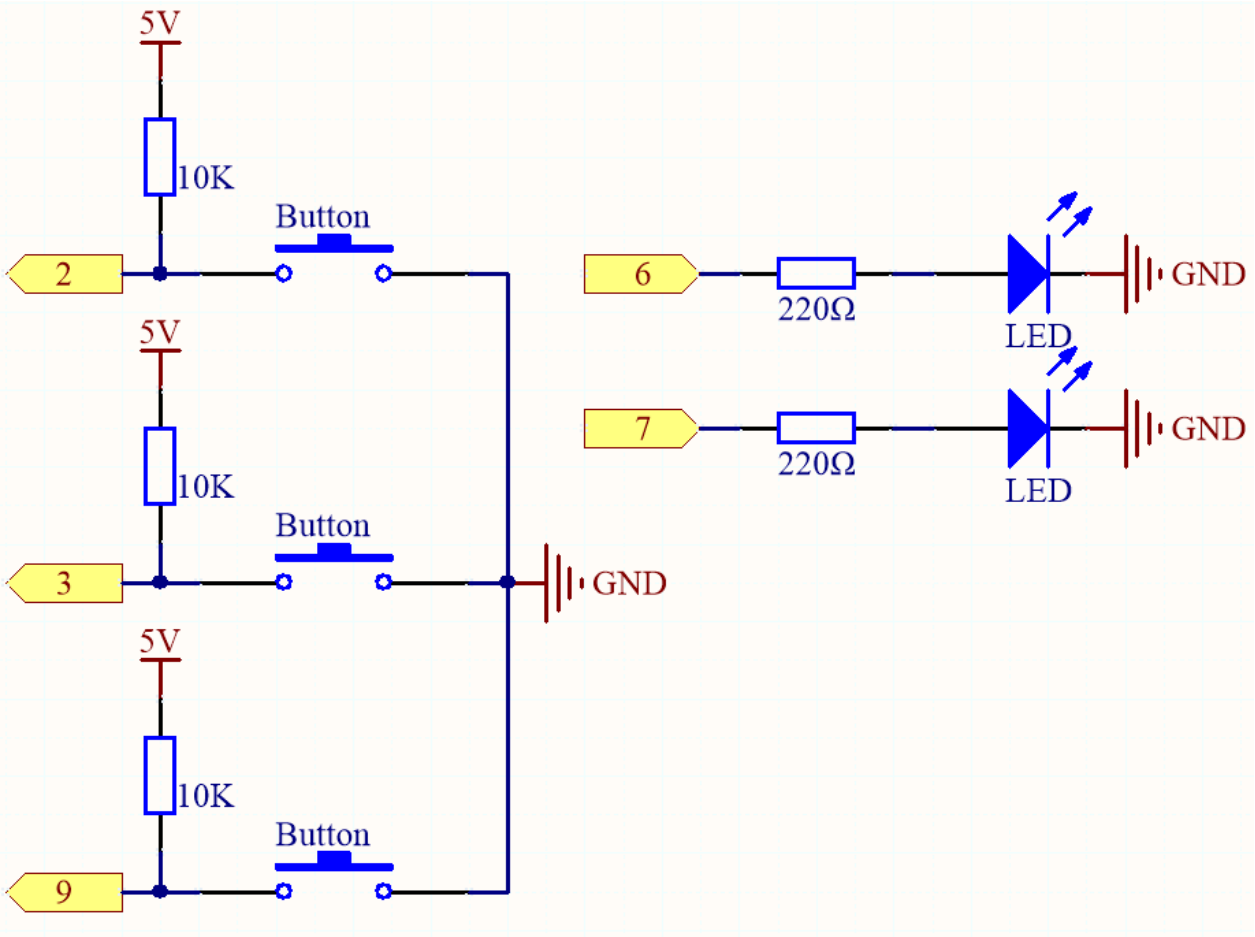
一式をまとめて購入すると確実に便利です。リンクはこちらです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

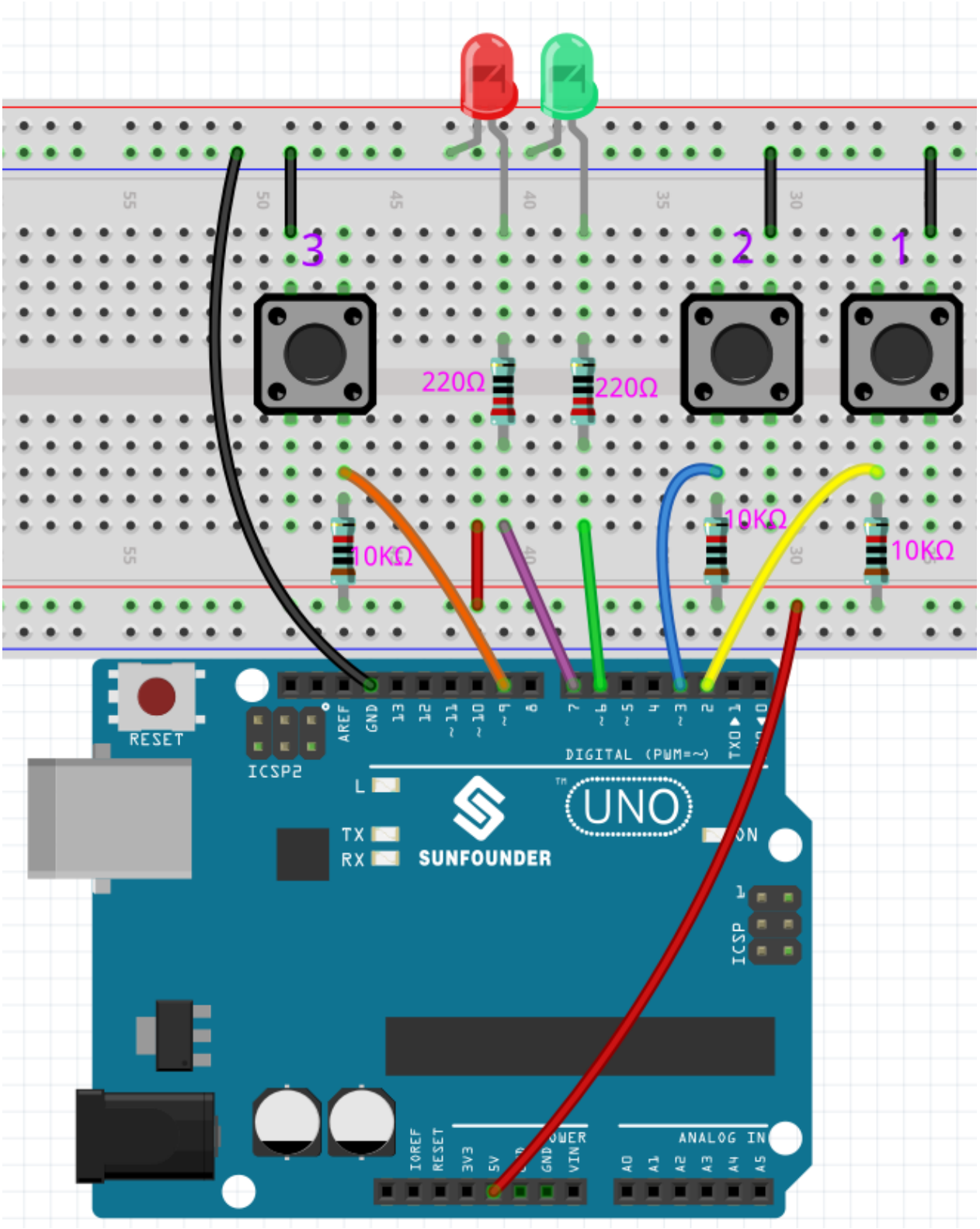
以下のリンクからそれぞれ購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |
| ボタン | |

**回路図**


5V
10K
Button
2
5V
10K
Button
3
GND
5V
10K
Button
9
6
220Ω
LED
GND
7
220Ω
LED
GND

配線図


3
2
1
220Ω
220Ω
10KΩ
10KΩ
10KΩ
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

コード

**注釈:**

- ファイル 6.5_reaction_time.ino を 3in1-kit\basic_project\6.5_reversingAid のパスで直接開くことができます。
- また、このコードを Arduino IDE 1/2 にコピーすることもできます。
- LiquidCrystal_I2C ライブラリを追加していることを確認してください。詳しいチュートリアルは *5.11* **外部ライブラリのインストール** を参照してください。

**どのように動作するのか？**

1. ボタンと LED を初期化し、ボタンの状態を読むためにここで 2 つの割り込みを使用します。

```
void setup()
{
    ...
    attachInterrupt(digitalPinToInterrupt(buttonPin1), pressed1, FALLING);
    attachInterrupt(digitalPinToInterrupt(buttonPin2), pressed2, FALLING);
    ...
}
```

2. rstBtn ボタンが押されると、ゲームが再開します。2 から 5ms のランダムな時間で、LED の 1 つを点灯させます。

```
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) {
```

(次のページに続く)

(前のページからの続き)

```
            digitalWrite(ledPin2, HIGH);
        }
    }
    delay(200);
}
```

- フラグが-1 で rstBtn ボタンが押されたとき、 random() 関数を使って 2-5 秒のランダムな時間を生成する。
- この時間は、LED の点灯制御に使われる。
- また、2 つの LED の点灯は randomTime % 2 によって 0 と 1 でランダムに生成されます。 flag が 0 の場合、LED1 が点灯します。 1 の場合、LED2 が点灯します。

3. pressed1() 関数について

```
void pressed1() {
    if (flag == -1) {
        return;
    }
    if (flag == 0) {
        int currentTime = millis();
        Serial.print("Correct! Your reaction time is: ");
        Serial.print(currentTime - timer);
        Serial.println(" ms");
    } else if (flag == 1) {
        Serial.println("Wrong Click!");
    }
    flag = -1;
}
```

これはボタン 1 が押されたときに起動される関数である。ボタン 1 が押されたとき、このときフラグが 0 であれば応答時間が出力され、そうでなければ押下エラーが促される。

4. pressed2() 関数について

```
void pressed2() {
    if (flag == -1) {
        return;
    }
    if (flag == 1) {
```

(次のページに続く)

(前のページからの続き)

```
        int currentTime = millis();
        Serial.print("Correct! Your reaction time is: ");
        Serial.print(currentTime - timer);
        Serial.println(" ms");
    } else if (flag == 0) {
        Serial.println("Wrong Click!");
    }
    flag = -1;
}
```

これはボタン 2 が押されたときに起動される関数である。ボタン 2 が押されたとき、このときフラグが 1 であれば応答時間が出力され、そうでなければ押下エラーが促される。

### 4.6.6 6.6 数字を当てるゲーム

数字を当てるゲームは、楽しいパーティーゲームで、友人と交代で数字（0~99）を入力します。数字を入力するごとに範囲が狭まり、正解すると、そのプレイヤーは負けとなり、罰を受けます。例えば、選ばれたラッキーナンバーが見えない 51 で、プレイヤー 1 が 50 を入力すると、数字の範囲は 50~99 になります。プレイヤー 2 が 70 を入力すると、数字の範囲は 50~70 になります。プレイヤー 3 が 51 を入力すると、彼または彼女が負け者となります。ここでは、IR リモートコントローラーで数字を入力し、LCD で結果を表示します。

#### 必要な部品

このプロジェクトには、以下の部品が必要です。

一式をまとめて購入することが便利です。リンクはこちらです：

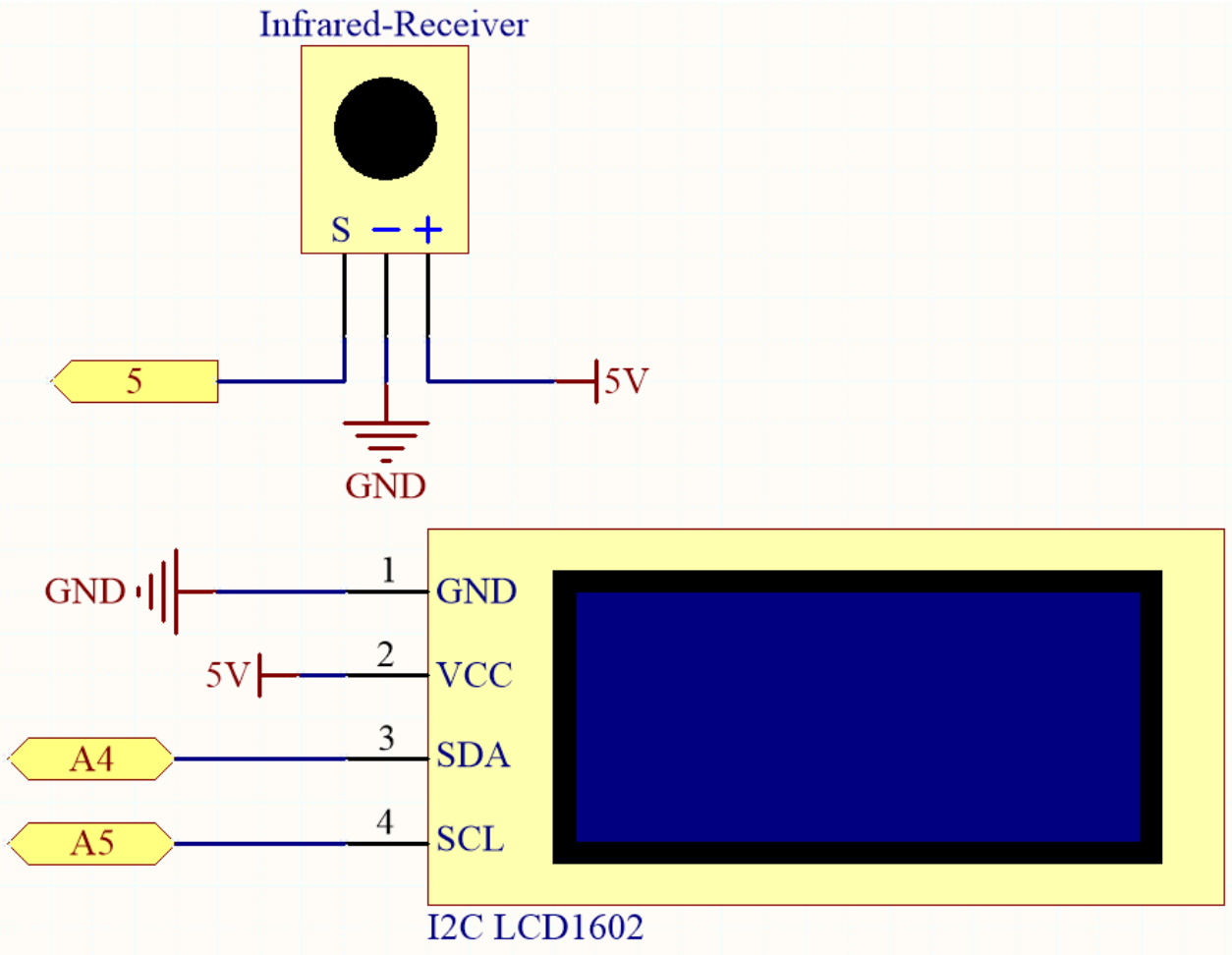
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクからそれぞれ購入することもできます。

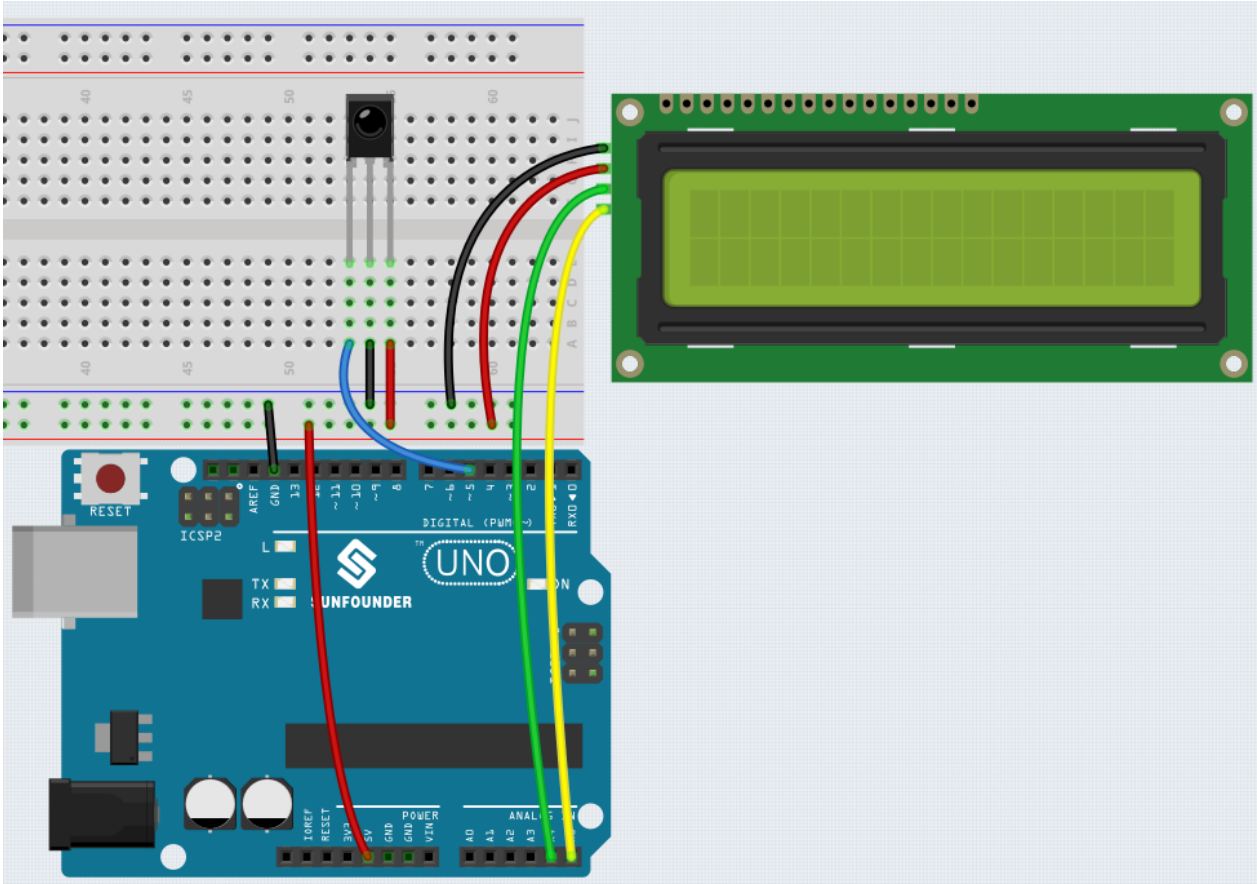
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| *I2C LCD1602* | |
| *IR* レシーバー | - |

**回路図**

この例では、LCD1602 と赤外線受信モジュールの配線は以下の通りです。




**配線図**




**コード**

---

**注釈:**

- ファイル `6.6.guess_number.ino` は、 `3in1-kit\basic_project\6.6.guess_number` のパスから直接開くことができます。
- また、このコードを Arduino IDE 1/2 にコピーしても良いです。
- ここでは、 `LiquidCrystal I2C` と `IRremote` **ライブラリ** を使用しています。 **Library Manager** からインストールすることができます。

---

コードが正しくアップロードされると、LCD1602 にウェルカムメッセージが表示されます。画面上の範囲のヒントに従って数字を押すと、表示がどんどん狭くなり、ラッキーナンバーを当てるまで続けます。

---

**注釈:** コードと配線が正しければ、しかし、LCD が表示されない場合、背面の可変抵抗を回してコントラストを調整することができます。

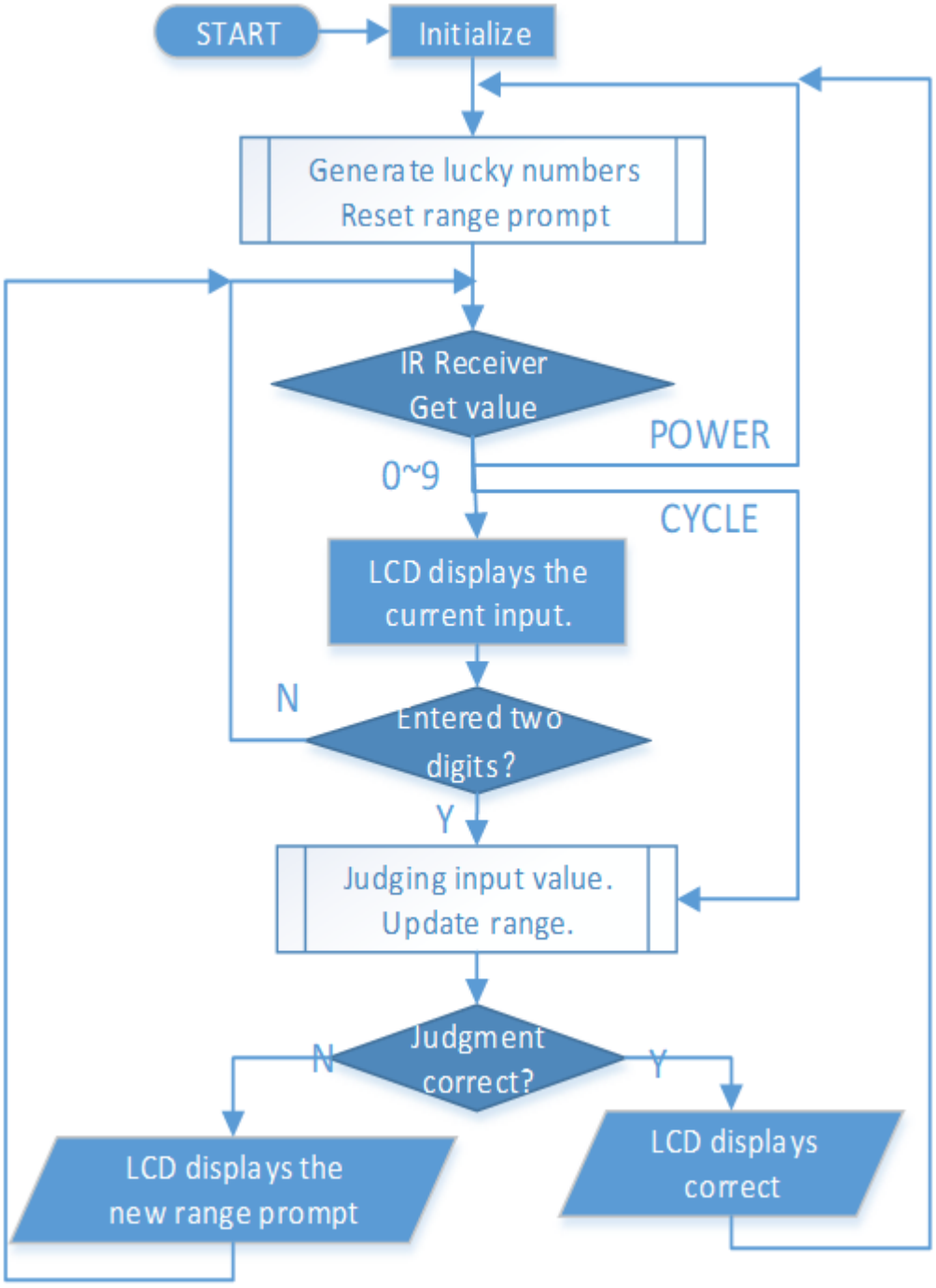
---

**どのように動作するのか？**

数字を当てるゲームを面白く、リアルにするために、以下の機能を実現する必要があります：

1. ゲームを開始やリセットすると、ラッキーナンバーが表示され、数字の範囲ヒントが 0〜99 にリセットされます。
2. LCD は入力中の数字と数字の範囲ヒントを表示します。
3. 2 桁を入力すると、自動的に結果の判断が現れます。
4. 1 桁の数字を入力した場合、結果の判断を開始するために CYCLE キー（コントローラの中央のキー）を押すことができます。
5. 答えが当てられなかった場合、新しい数字の範囲ヒントが表示されます（ラッキーナンバーが 51 で、50 を入力すると、数字の範囲ヒントが 50~99 に変わります)。
6. ラッキーナンバーが当てられた後、ゲームは自動的にリセットされ、プレイヤーは新しいラウンドをプレイできます。
7. POWER ボタン（左上のボタン）を直接押すことでゲームをリセットすることができます。

結論として、プロジェクトのワークフローは、フローチャートに示されています。


START
Initialize
Generate lucky numbers
Reset range prompt
IR Receiver
Get value
POWER
0~9
CYCLE
LCD displays the
current input.
N
Entered two
digits?
Y
Judging input value.
Update range.
Judgment
correct?
N
Y
LCD displays the
new range prompt
LCD displays
correct

# 第 5 章

# カープロジェクト

たくさんの異なるスマートロボットカーを見たことがあると思いますが、基本的な機能は似ています。基本の動き、障害物回避、ライン追従、フォロー、リモコンでの操作などです。

ここでは、最もシンプルな構造でスマートロボットカーを組み立てますが、上記のすべての機能も実現可能です。さらに、携帯電話で操作することもできます。チュートリアルについては、 *8. IoT* **カー** を参照してください。

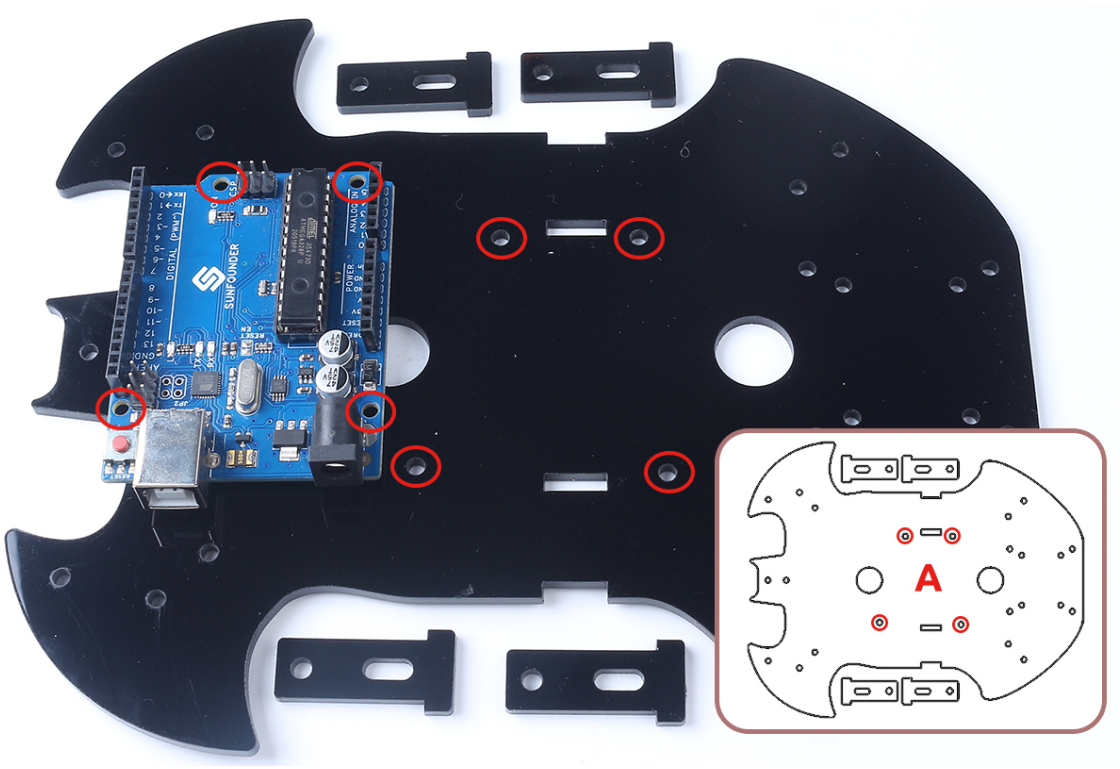
**組み立て手順**

## 5.1 車の組み立て

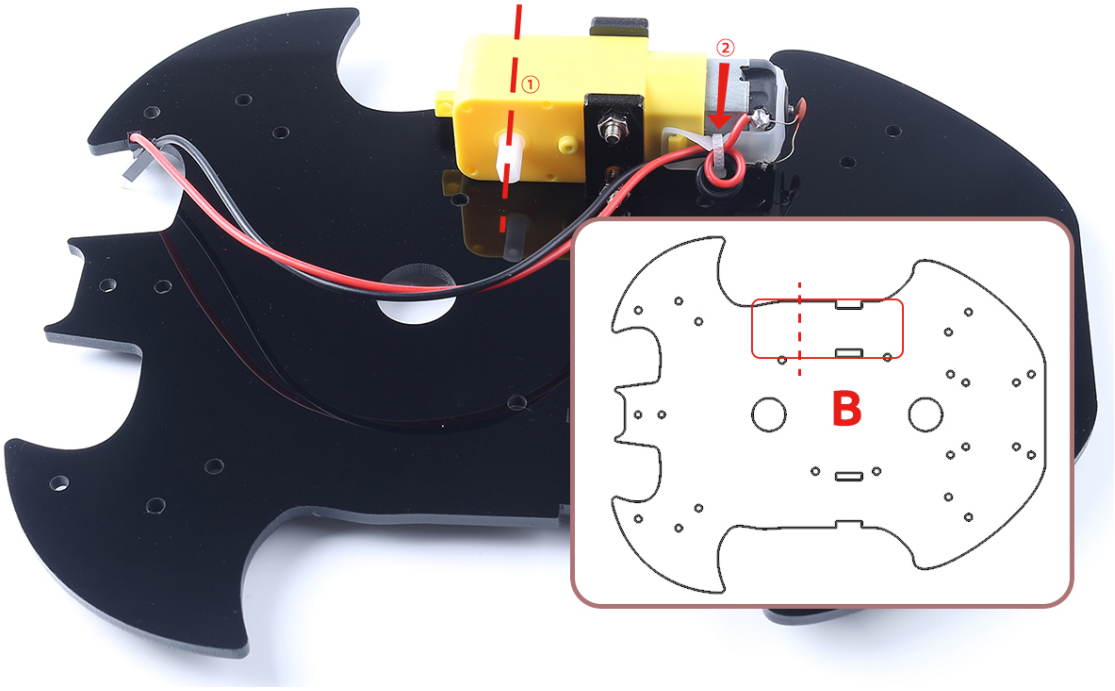
以下の手順に従って、車の組み立てを完了させてください。

1. アクリルの保護フィルムを取り外してください。

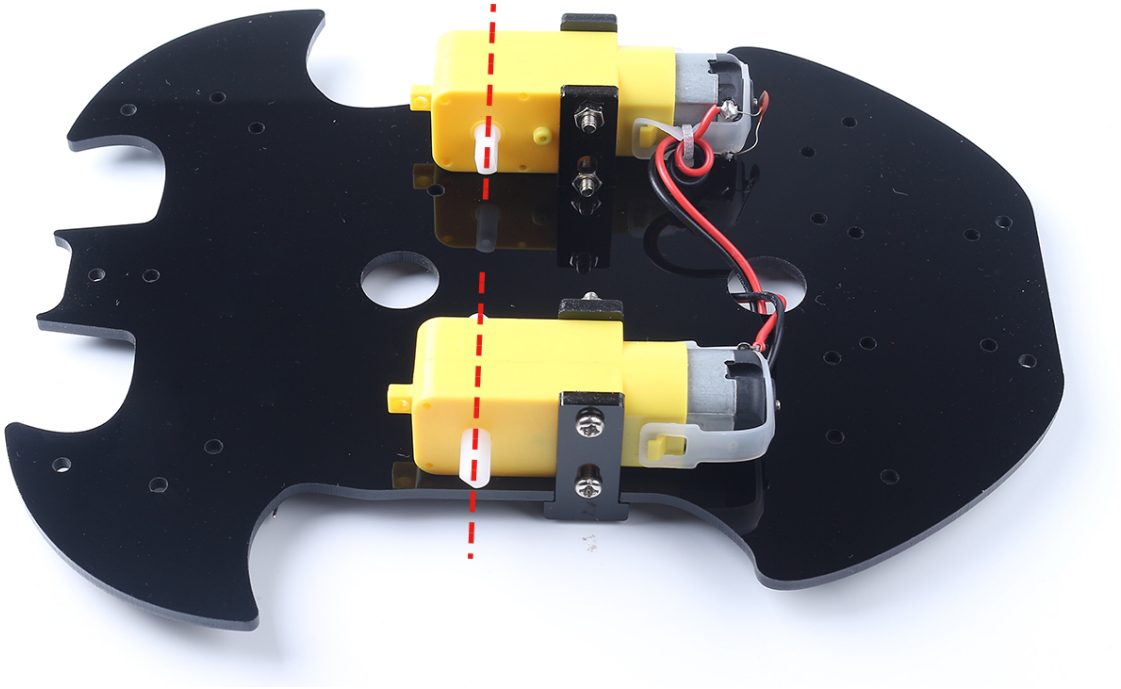
2. 画像に示されているように、ボードをテーブルに置いてください。R3 ボードと同じ穴がある側を A と呼び、裏側を B と呼ぶことにします。これにより、組み立て中のミスを避けるのに役立ちます。



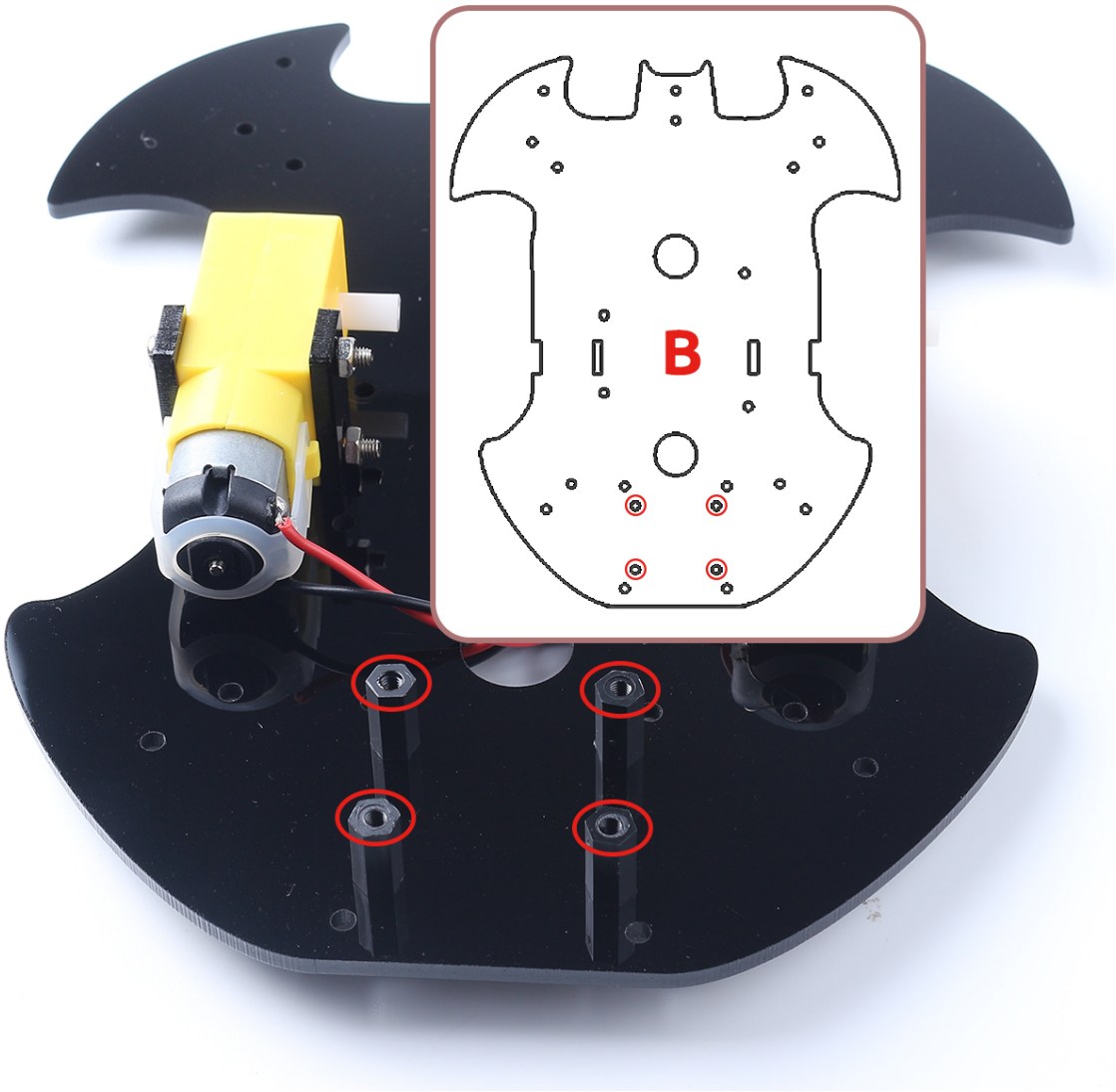

3. B 側に向け、M3x30mm のネジと M3 のナットを使用して TT モーターを取り付けます。注意点は 2 つ: 1 - 出力軸はコウモリの形をした側を向いていること、2 - モーターケーブルは内側を向いていること。



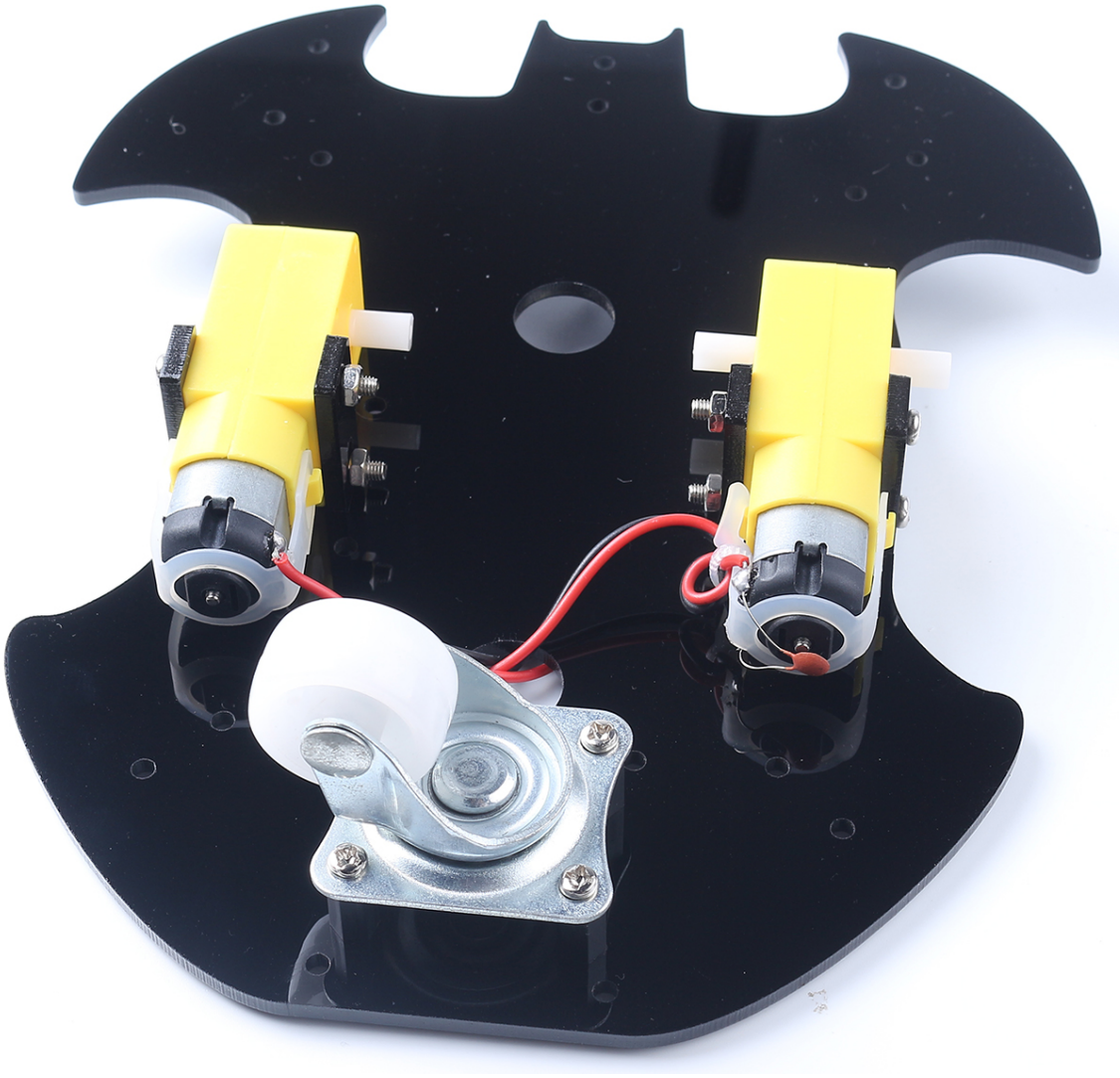
4. 別の TT モーターを取り付けます。出力軸とケーブルの方向にも同様の注意が必要です。



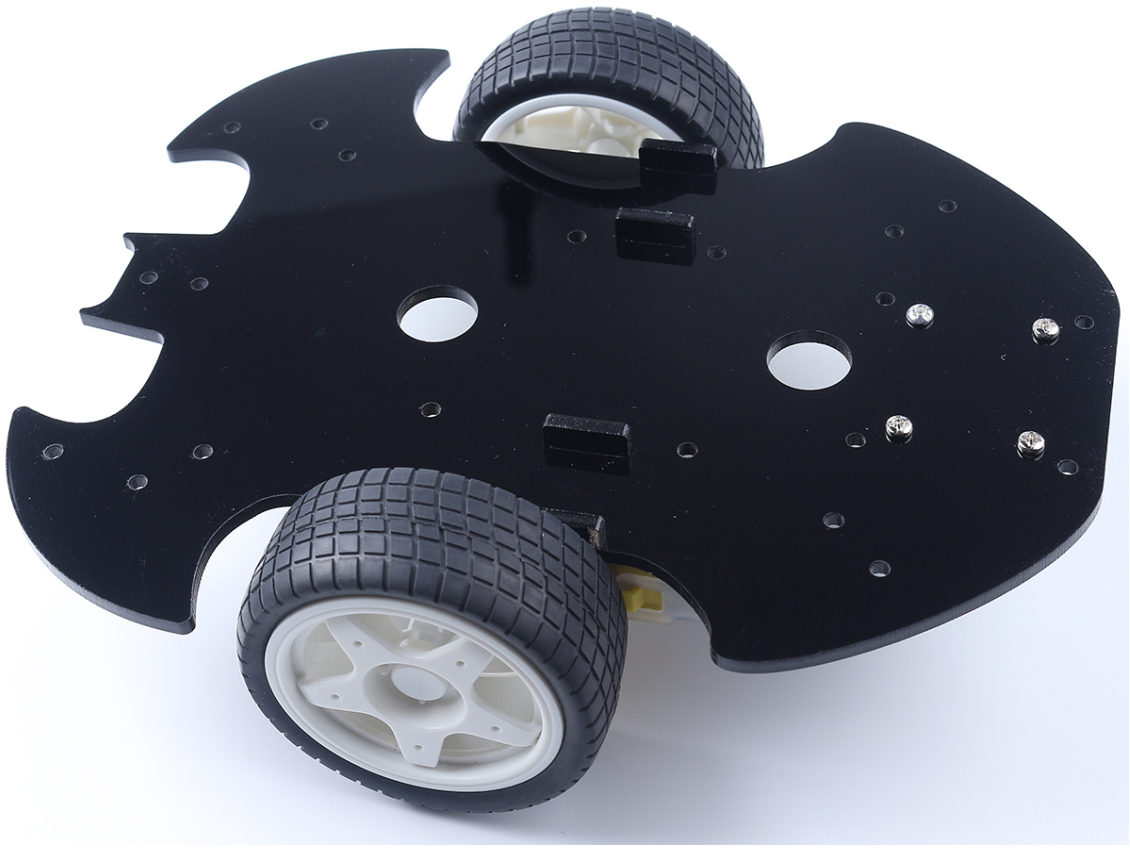
5. M3x6mm のネジを使用して、下に示されている位置に M3x12mm のスペーサーを取り付けます。




6. M3x6mm のネジを使用して、万能車輪を取り付けてください。

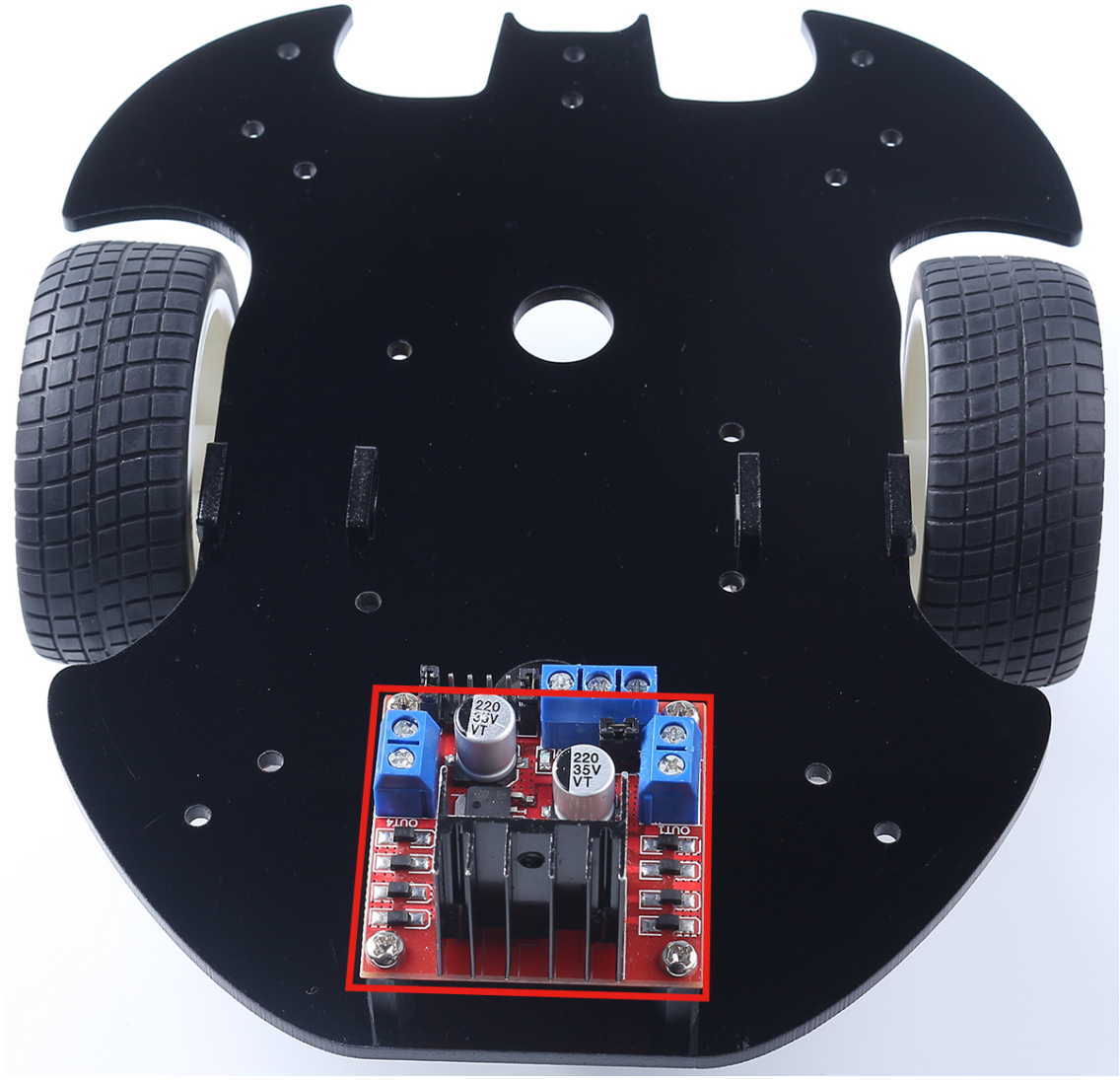


7. 2 つの車輪を取り付けると、車の基本構造が完成します。

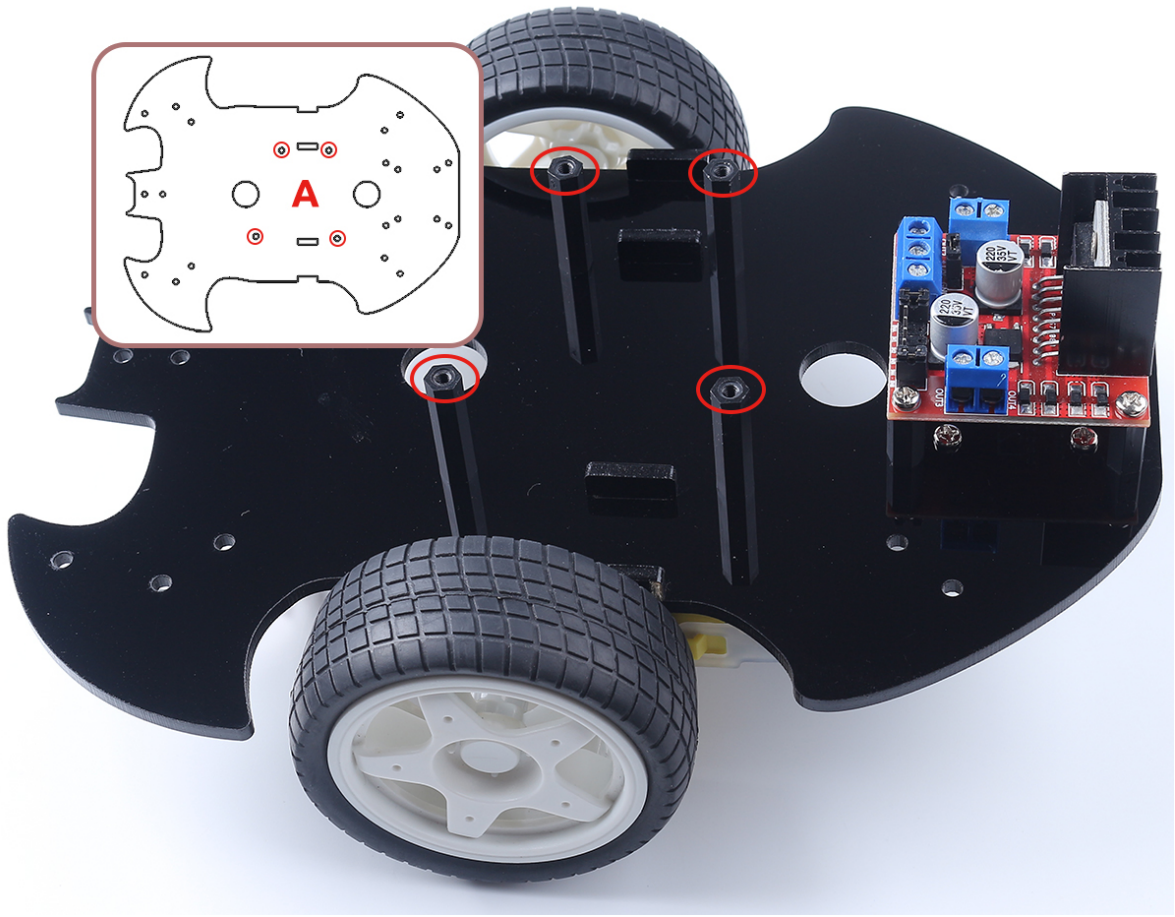


8. M3x6mm のネジで車の後部に M3x12mm のブラケットを取り付けます。

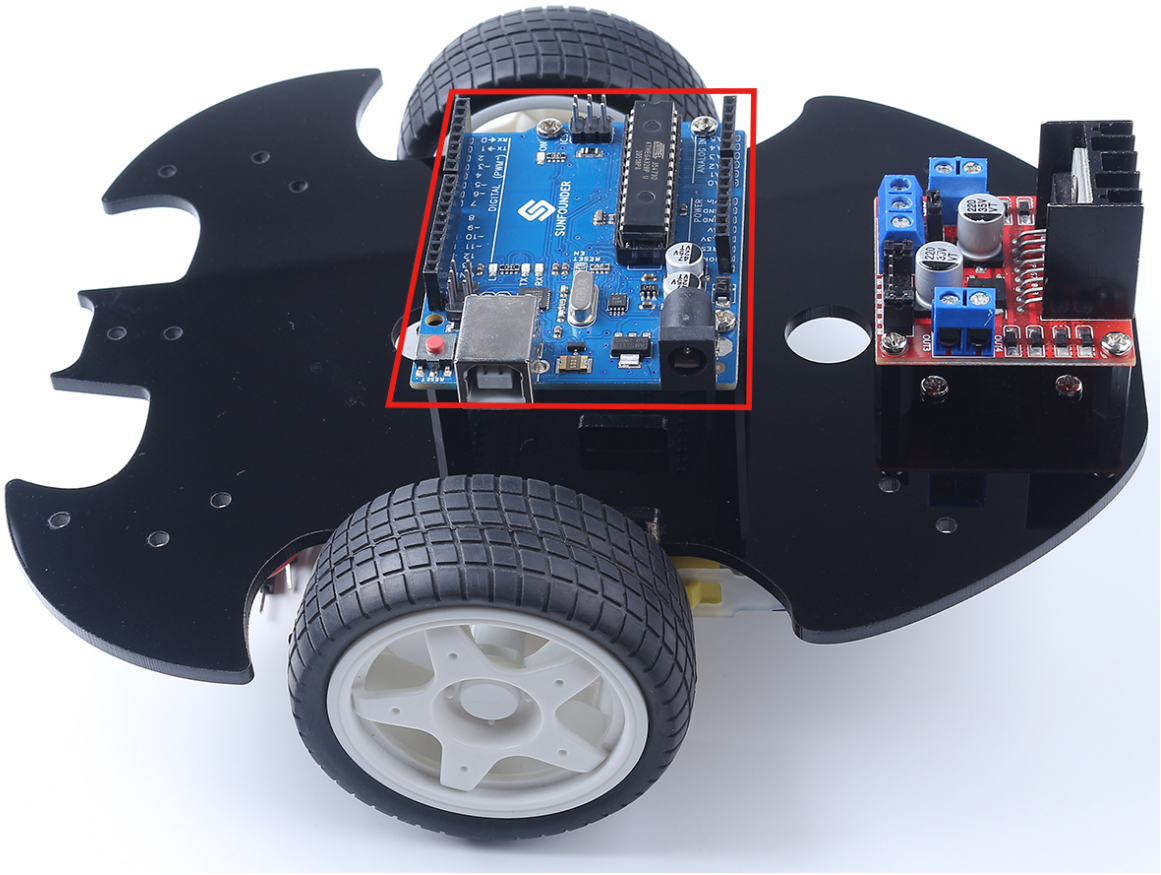

9. M3x6mm のネジで L298N モジュールを取り付けてください。



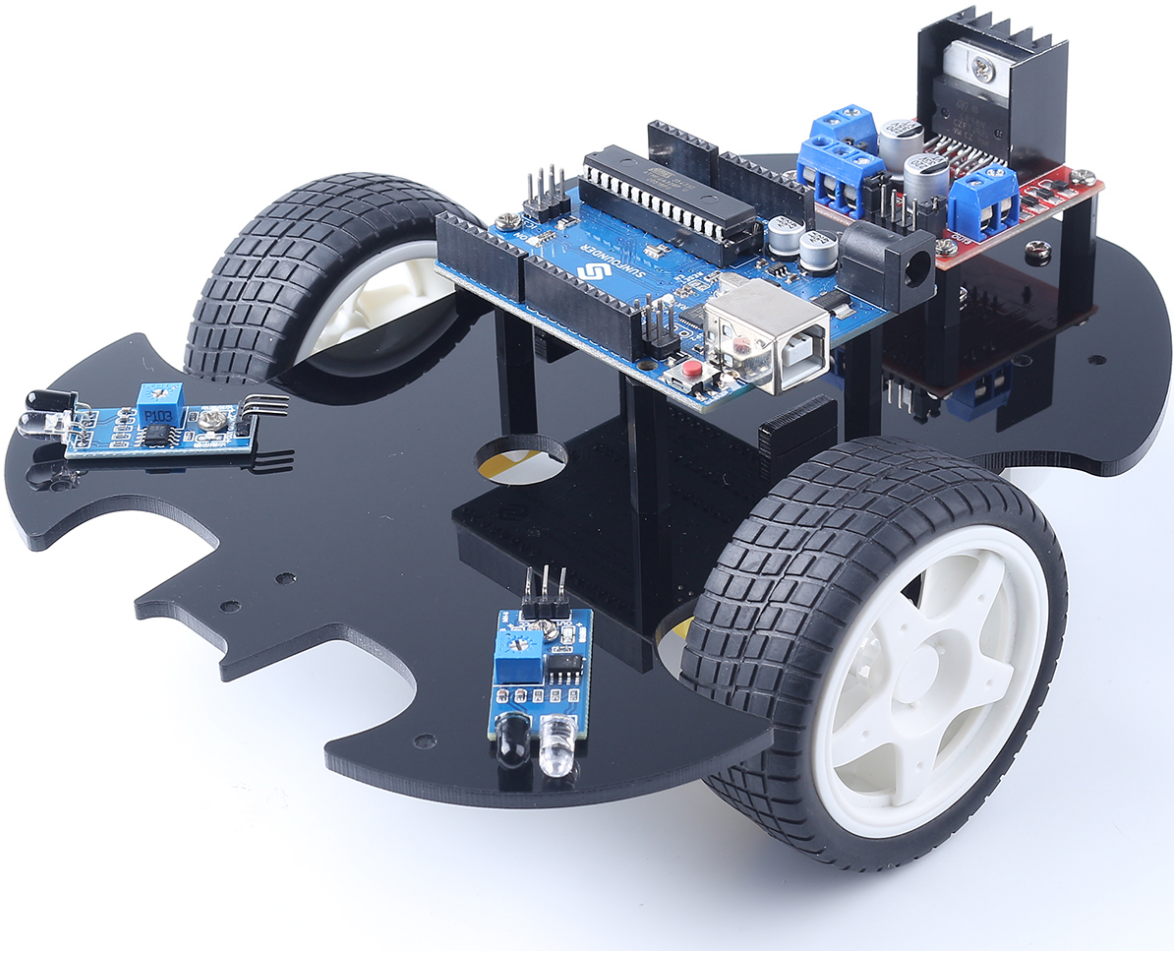

10. 以下に示されている位置に M3x6mm のネジで M3x24mm のスペーサーを取り付けます。



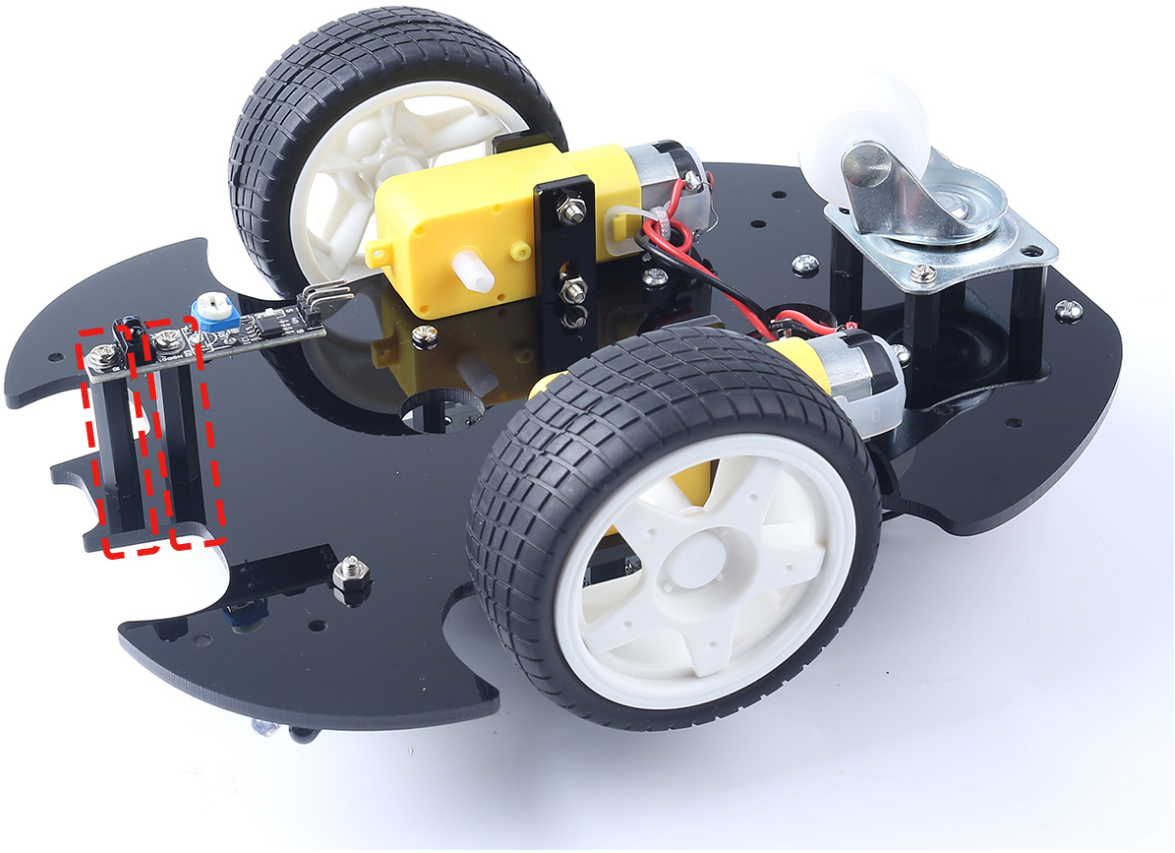

11. M3x6mm のネジで R3 ボードを取り付けてください。



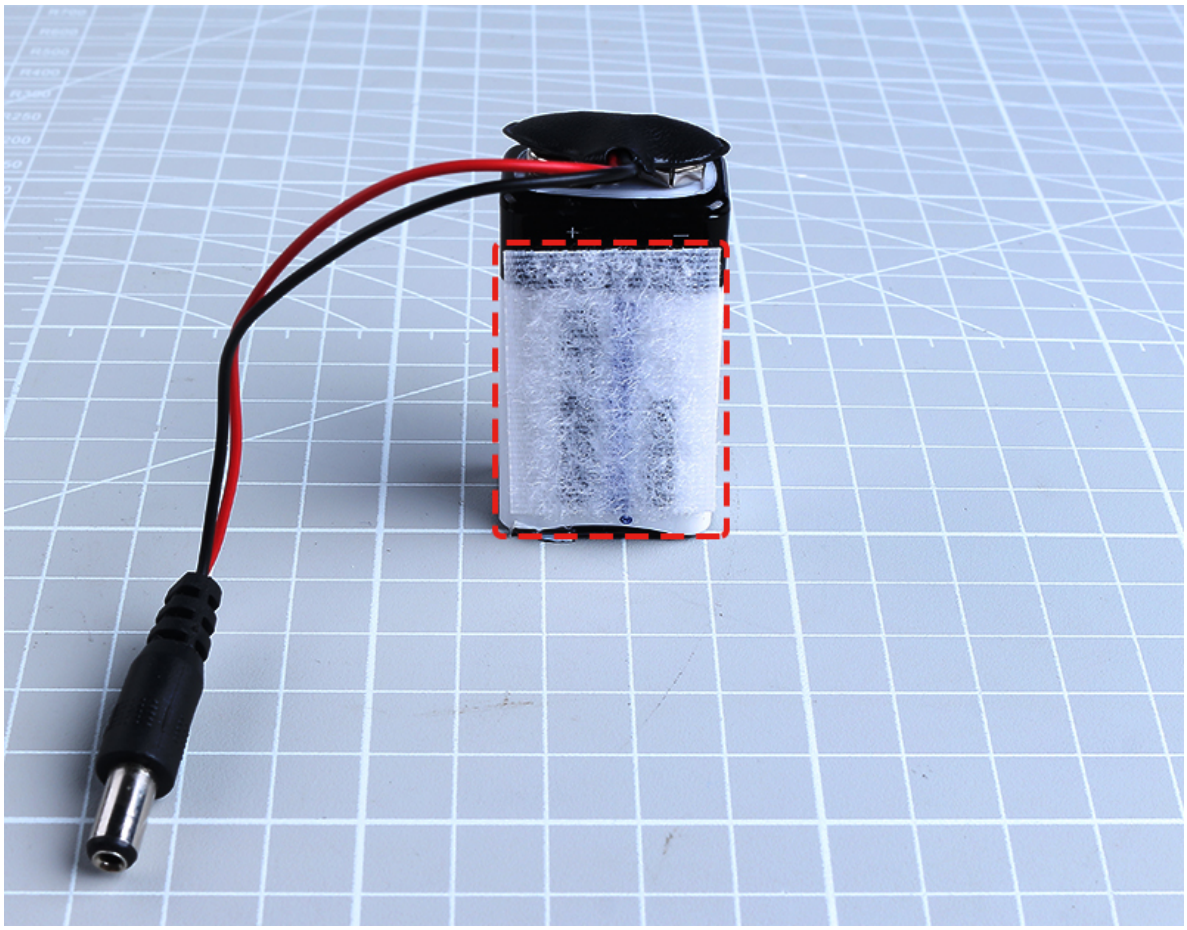

12. M3x10mm のネジと M3 のナットを使用して、2 つの IR 障害物モジュールを組み立ててください。



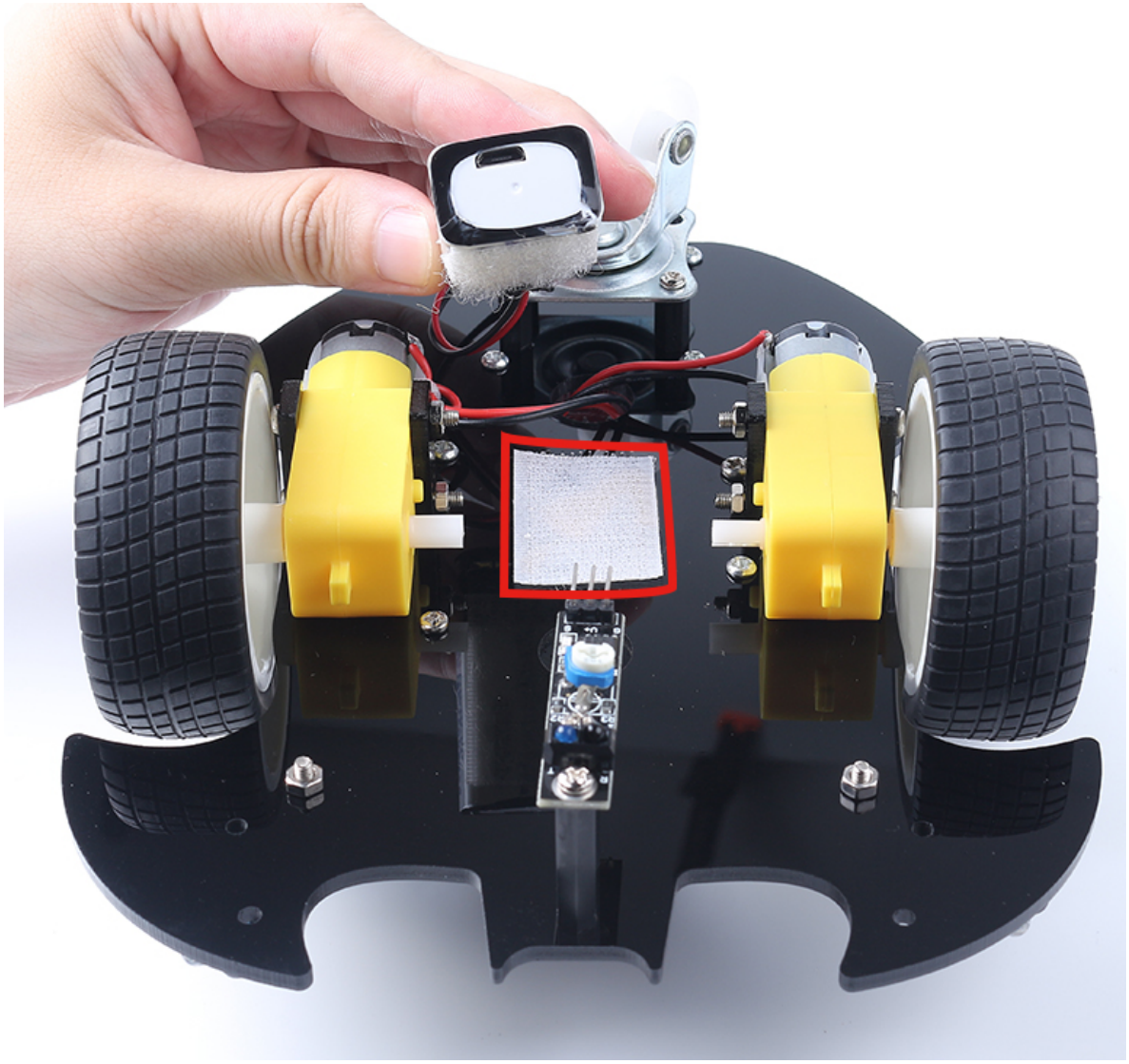

13. B 側に向けて、4 つの M3x6mm のネジと 2 つの M3x24mm のスペーサーを使用して、ライントラックモジュールを取り付けます。



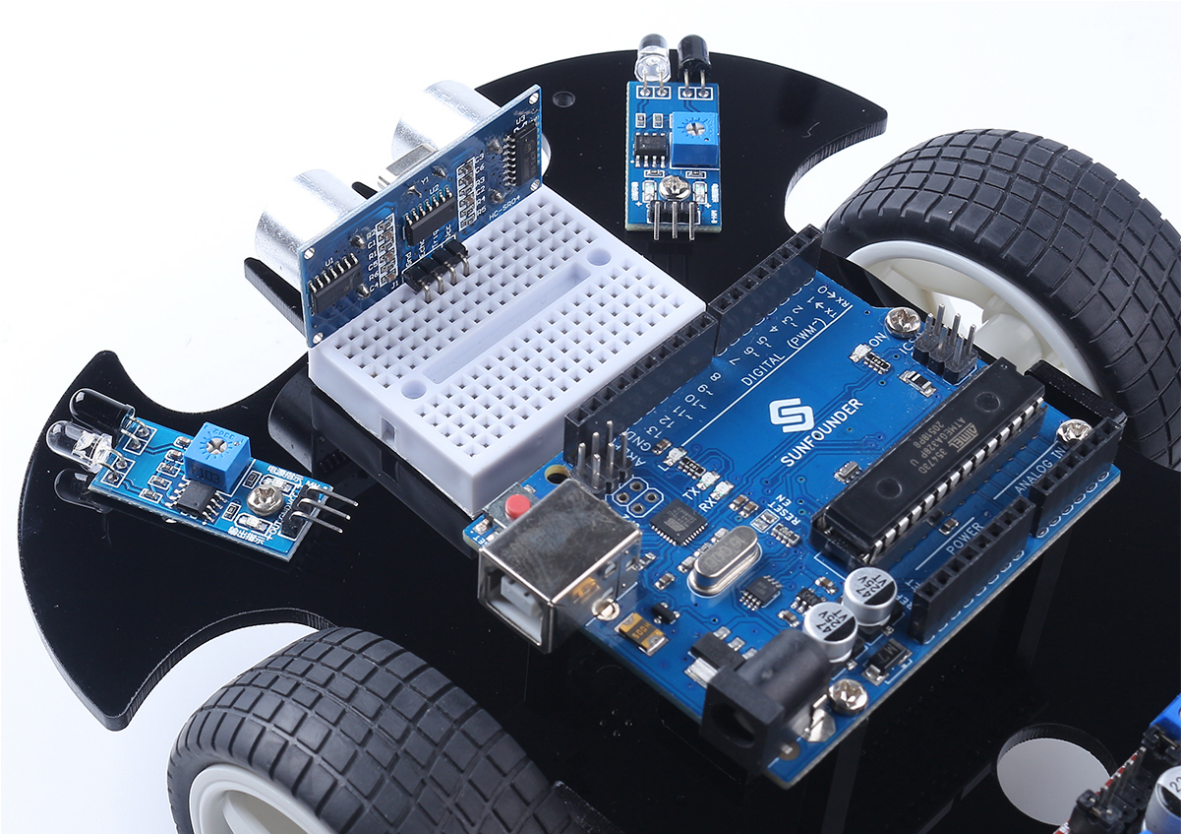
14. 9V のバッテリーにベルクロを貼り、バッテリークリップを取り付けてください。



15. バッテリーを固定するために、車にベルクロのもう一方の部分を貼り付けてください。



16. A 側に戻して、ブレッドボードを車の前部に取り付けます。その後、プロジェクトに必要に応じて、ブレッドボードに異なるコンポーネント（例：超音波モジュール）を追加できます。




17. 車を動かすためには、配線とコードの記述も必要です。これに関する詳細は、後続のセクションで記述されます。

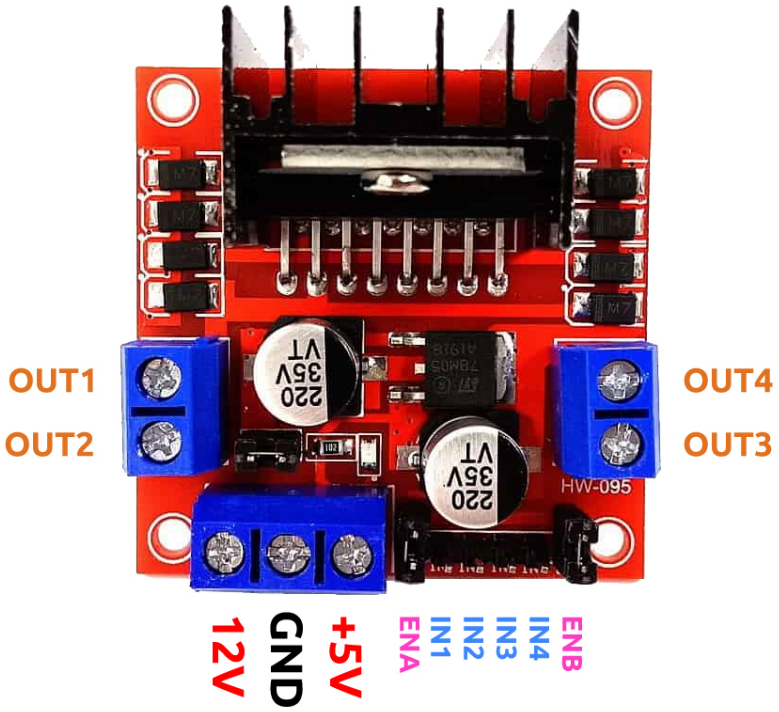
**プロジェクト**

以下は、Arduino IDE で C 言語でプログラムされたカープロジェクトの一部です。Arduino に特に慣れていない場合は、 *Arduino* **を始めよう** を参照してください。

次のプロジェクトは、プログラムの難易度の順に記述されています。順番にこれらの資料を読むことをおすすめします。

Scratch で車のプログラムをしたい場合は、 *Scratch* **で遊ぼう** を参照してください。

## 5.2 1. 移動




プログラミングを始める前に、L298N の動作原理を確認しましょう。

IN1～IN4 は L298N モジュールの入力、OUT1～OUT4 は出力です。

これらを使用する簡単な方法は：IN1 に高レベルを入力すると、OUT1 は高レベルを出力します。IN1 に低レベルを入力すると、OUT1 は低レベルを出力します。モータの両端を OUT1 と OUT2 に接続し、IN1 と IN2 に逆のレベル信号を入力すると、モータが回転します。OUT3 と OUT4 も同じ方法で使用できます。

ENA と IN1,IN2 の間の動作関係は以下の通りです。

| ENA | IN1 | IN2 | **右モータ (A) の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

ENB と IN3,IN4 の間の動作関係は以下の通りです。

| ENB | IN3 | IN4 | **左モータ (B) の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

### 必要な部品

このプロジェクトでは、以下の部品が必要です。

全体のキットを購入するのは確かに便利です、リンクはこちら：

| **名前** | **このキットのアイテム** | **リンク** |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクからそれぞれの部品を個別に購入することもできます。

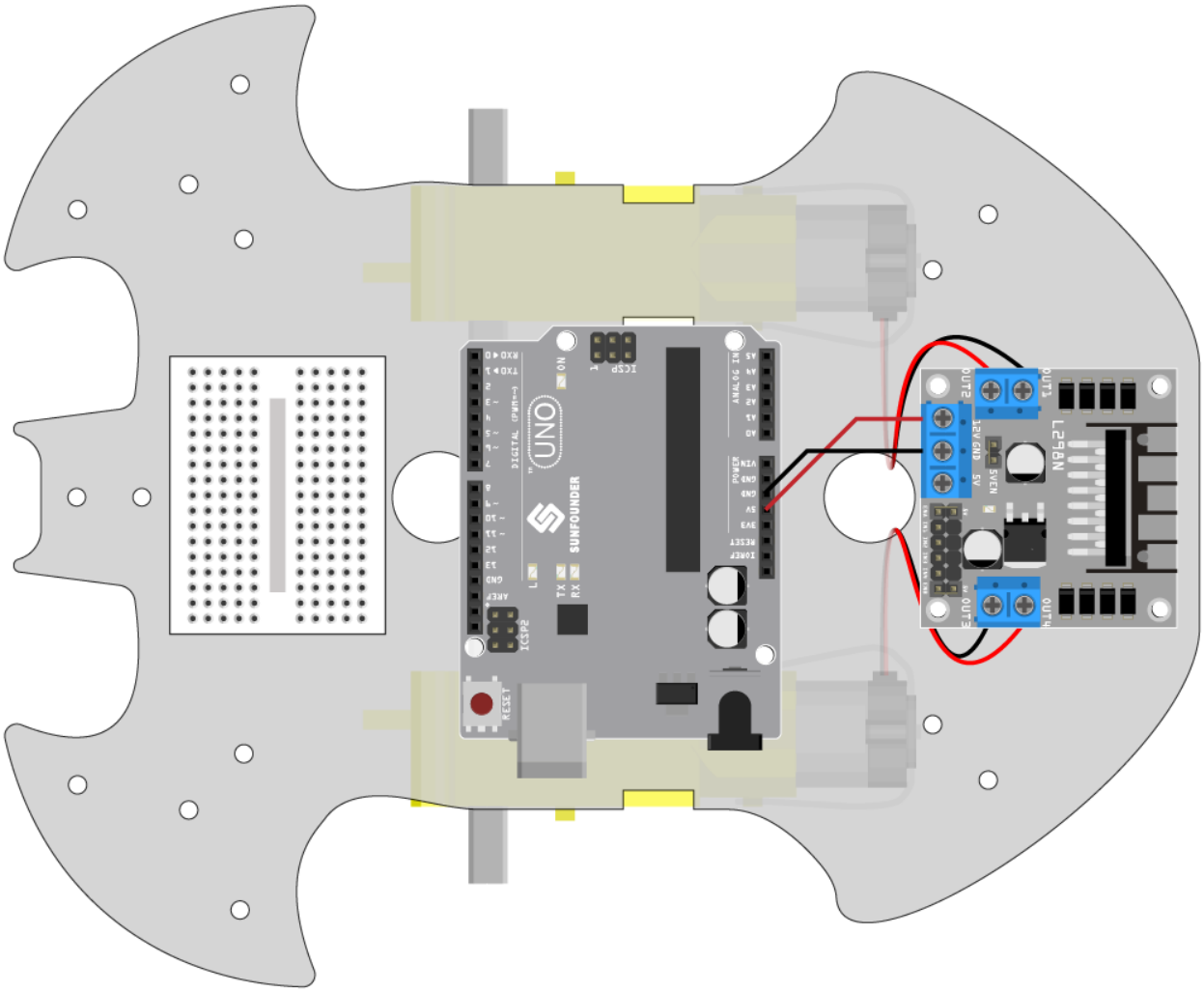
| **コンポーネントの紹介** | **購入リンク** |
|---|---|
| *L298N* **モジュール** | |

### 前進

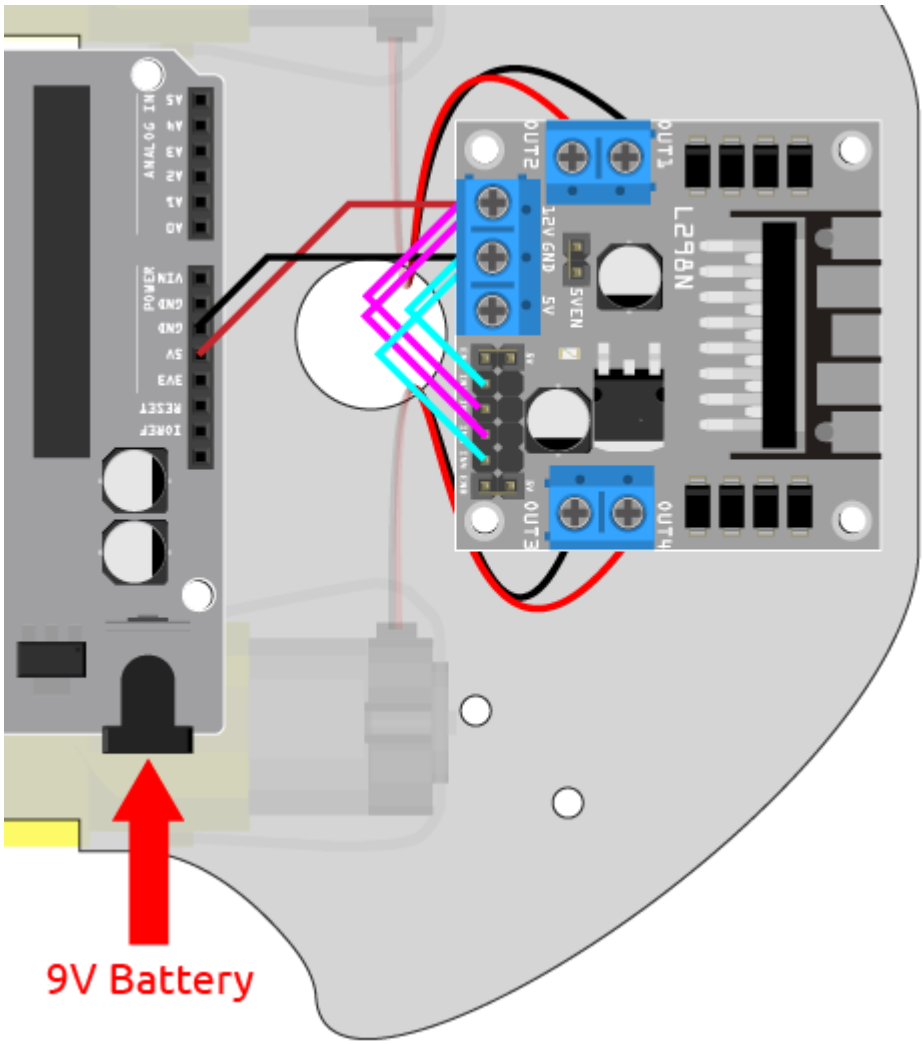
L298N モジュールの入力を直接 12V と GND に接続することで、車を動かしてみましょう。

1. R3 ボード、L298N モジュール、2 つのモータを接続します。

| L298N | R3 **ボード** | **モーター** |
|---|---|---|
| 12V | 5V | |
| GND | GND | |
| OUT1 | | 右モータの黒線 |
| OUT2 | | 右モータの赤線 |
| OUT3 | | 左モータの黒線 |
| OUT4 | | 左モータの赤線 |




2. IN2 と IN3 を 12V に、IN1 と IN4 を GND に接続すると、車が前進するのがわかります。

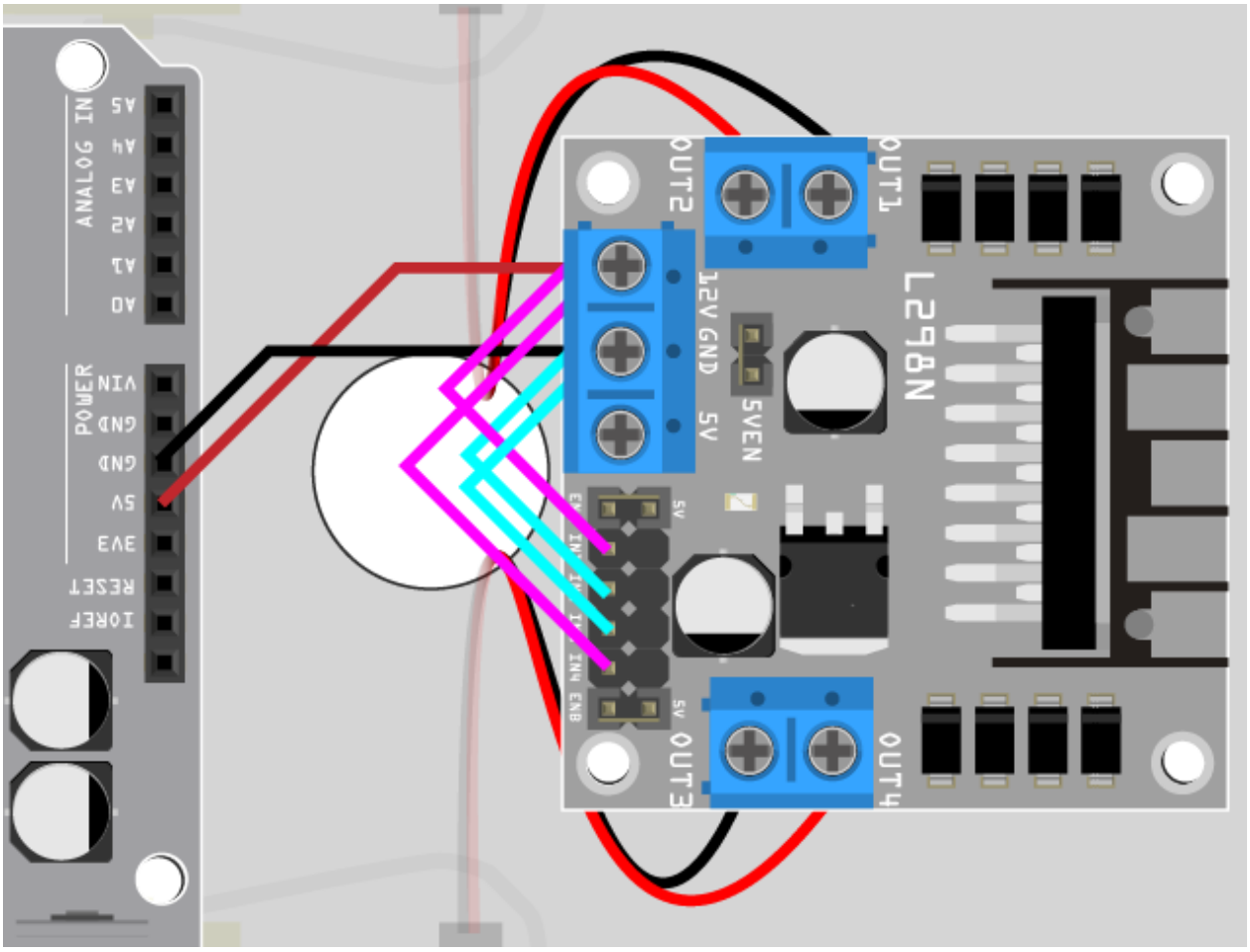



両方のモータが前進しない場合、次の状況が発生した場合は、2 つのモータの配線を再調整する必要があります。

- 両方のモータが同時に後退する場合（左モータが時計回り、右モータが反時計回り）、左と右のモータの配線を同時に交換します。OUT1 と OUT2、OUT3 と OUT4 を交換します。
- 左モータが後退する場合（時計回り）、左モータの OUT3 と OUT4 の配線を交換します。
- 右モータが後退する場合（反時計回り）、右モータの OUT1 と OUT1 の配線を交換します。

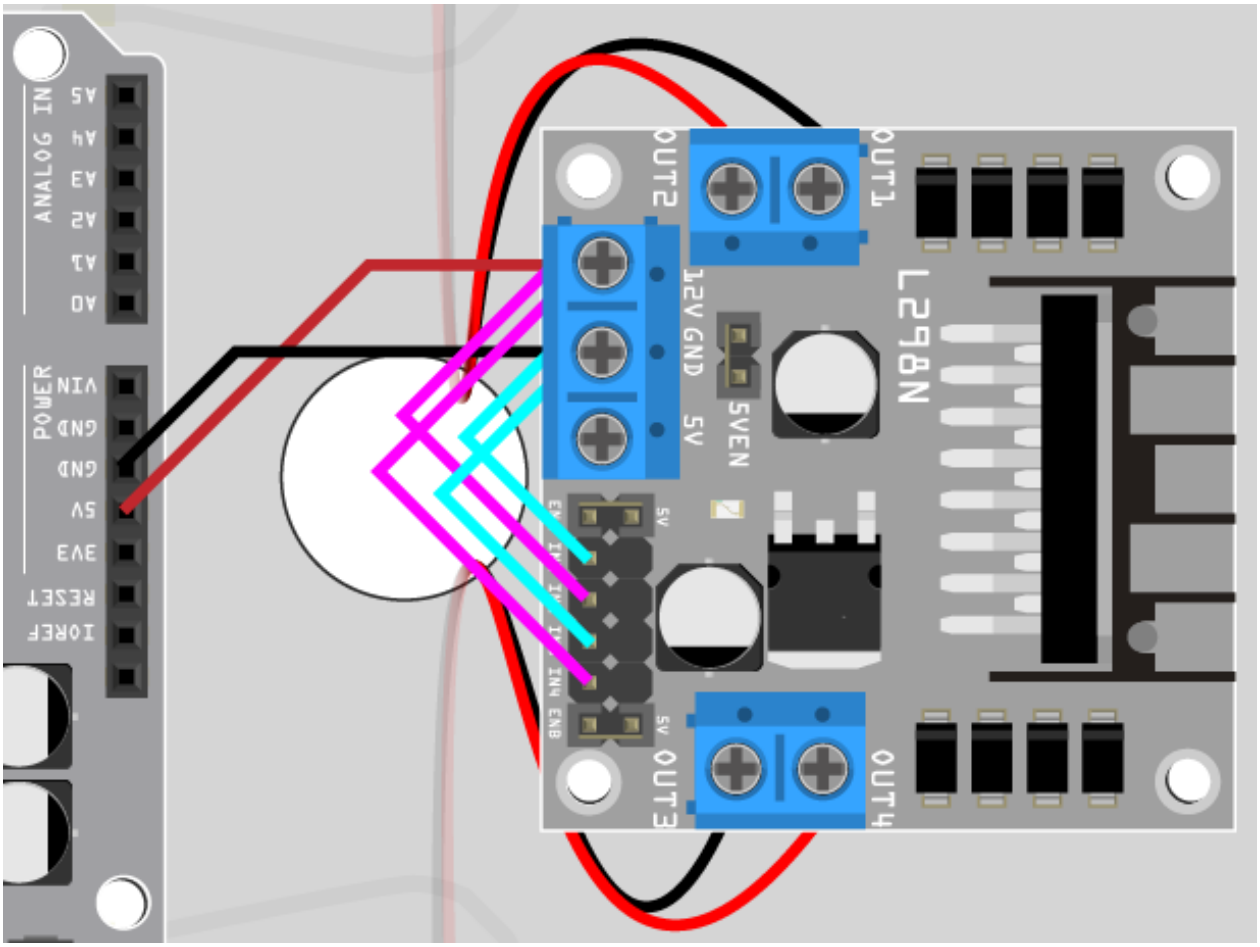
**後退**

IN2 と IN3 を GND に、IN1 と IN4 を 12V に接続すると、車が後退するのがわかります。



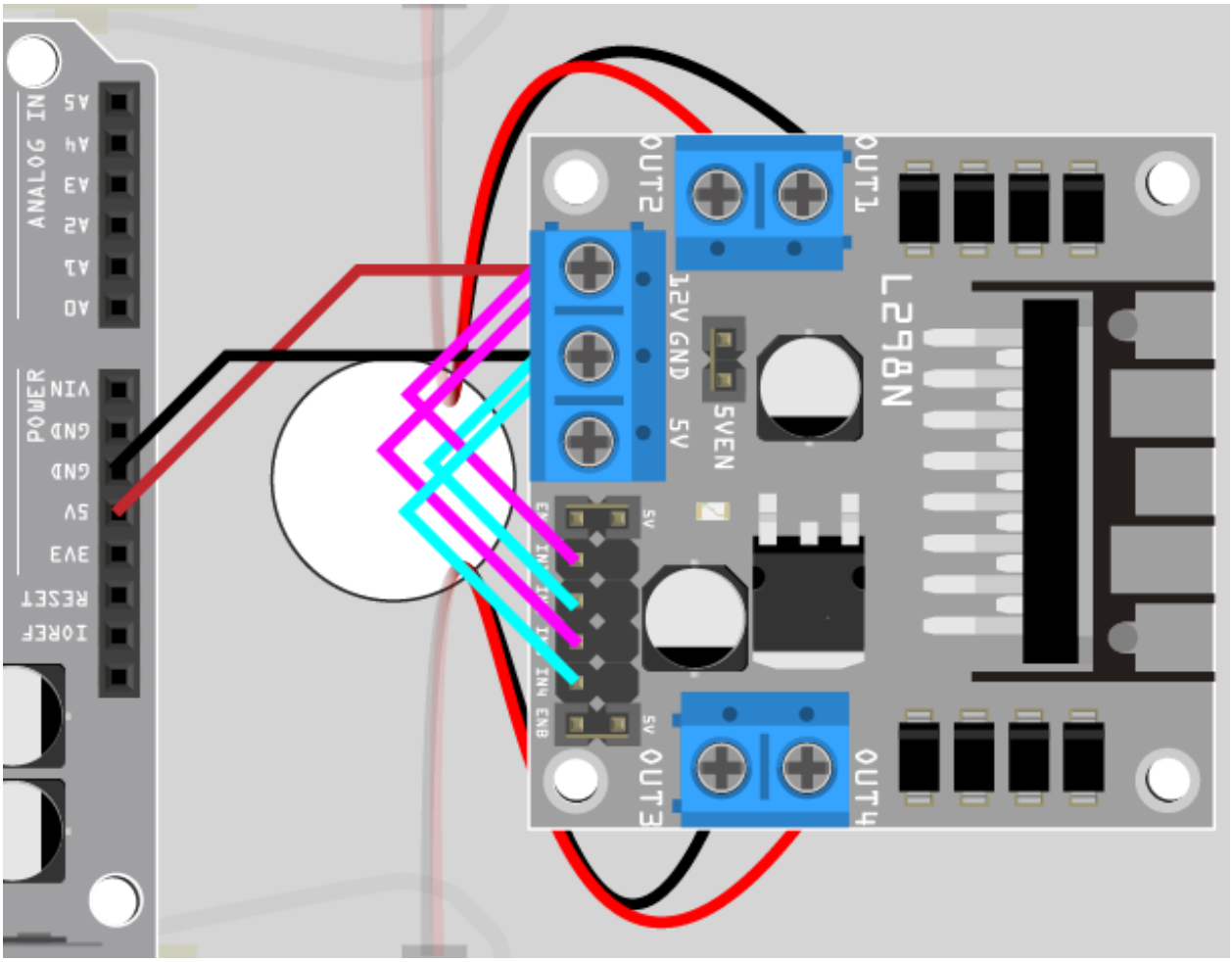

### 左折

車を左に曲げたい場合、すなわち、両方のモータを反対の方向に回転させる場合、IN1 を 12V に、IN2 を GND に、IN3 を GND に、IN4 を 12V に接続します。




**右折**

車を右に曲げたい場合、すなわち、両方のモータを反対の方向に回転させる場合、IN1 を GND に、IN2 を 12V に、IN3 を 12V に、IN4 を GND に接続します。




**停止**

モーターを停止するには、同じ側の入力を同時に 12V または GND に接続します。例えば、IN1 と IN2 を同時に 12V または 5V に接続し、IN3 と IN4 も同様に接続します。

これはもちろん理論的なもので、後にコードで制御する際に必要となる。ここで、車への電源供給を停止することができます削除します。

## 5.3 2. コードでの移動

前のプロジェクトでは、L298N の入力の異なるレベルの信号を使用してモーターの操作を制御しました。

プログラムを通じてレベル信号を変更することで、車の動きを柔軟に制御することができます。ここでは、L298N の IN1～IN4 ピンを R3 ボードの 5、6、9、10 ピンに順に接続します。

**必要な部品**

このプロジェクトには以下の部品が必要です。

キット全体を購入すると非常に便利です。リンクは以下のとおりです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクからそれぞれ個別に購入することもできます。

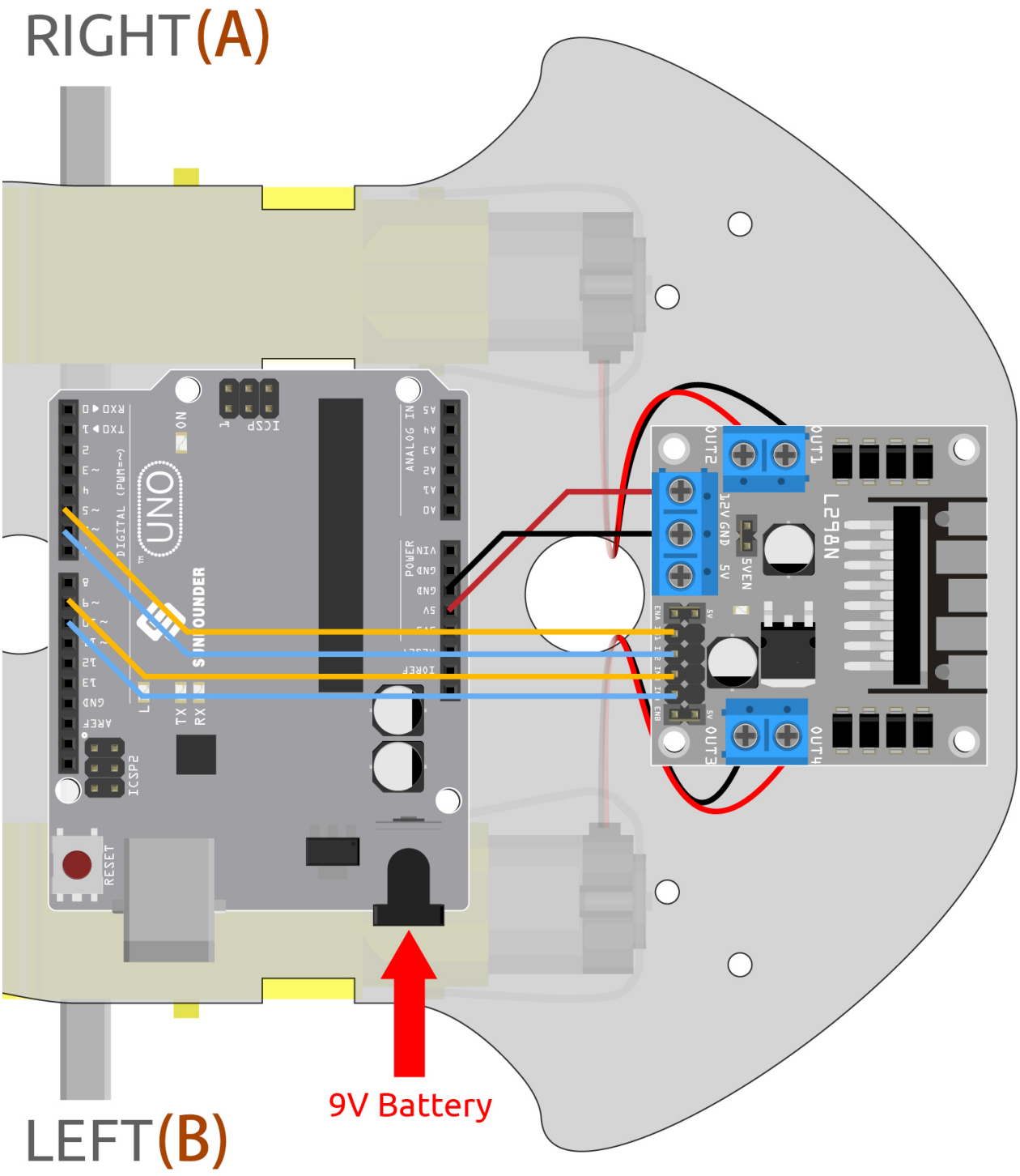
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |

**配線図**

L298N モータードライバモジュールは、DC モーターとステッピングモーターを駆動するための高出力モータードライバモジュールです。L298N モジュールは、最大で 4 台の DC モーター、または方向と速度の制御が可能な 2 台の DC モーターを制御することができます。

以下の図に従って、L298N モジュールと R3 ボードの間にワイヤーを接続してください。

| L298N | R3 ボード | モーター |
|---|---|---|
| IN1 | 5 | |
| IN2 | 6 | |
| IN3 | 9 | |
| IN4 | 10 | |
| OUT1 | | 右モーターの黒ワイヤー |
| OUT2 | | 右モーターの赤ワイヤー |
| OUT3 | | 左モーターの黒ワイヤー |
| OUT4 | | 左モーターの赤ワイヤー |




**コード**

**注釈:**

- 3in1-kit\car_project\2.move のパスの下の 2.move.ino ファイルを開きます。

- または、このコードを **Arduino IDE** にコピーしてください。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードをアップロードすると、車はそれぞれ 2 秒間で前進、後退、左右に移動します。

**どのように動作するのか？**

このプロジェクトは基本的に前回のものと同じで、IN1 から IN4 までの異なるレベルを与えることで車を前進、後退、左右に動かし、停止させるものです。

1. IN1~IN4 のピン配線を初期化する。

```
const int in1 = 5;
const int in2 = 6;
const int in3 = 9;
const int in4 = 10;

void setup() {
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
}
```

2. 左右のモーターの回転を制御するために IN1~IN4 を異なるハイまたはローレベルに設定し、それらを個別の関数にカプセル化する。

```
void moveForward() {
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
}

void moveBackward() {
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
}
...
```

3. loop() でこれらの関数を呼び出す。

```
void loop() {
    moveForward();
    delay(2000);
    stopMove();
    delay(500);

    moveBackward();
    delay(2000);
    stopMove();
    delay(500);
...
```

- digitalWrite(pin, value)

  - pin: Arduino のピン番号。
  - value: HIGH または LOW。

  デジタルピンに HIGH または LOW の値を書き込む。ピンが pinMode() で OUTPUT として設定されていれば、その電圧は対応する値に設定されます：HIGH の場合は 5V（または 3.3V ボードで 3.3V）、LOW の場合は 0V（グラウンド）。

- pinMode(pin, mode)

  - pin: モードを設定する Arduino のピン番号。
  - mode: INPUT、OUTPUT、または INPUT_PULLUP。

  指定されたピンを入力または出力として動作するように設定します。

- delay(ms)

  - ms: 一時停止するミリ秒数。許可されるデータタイプ: unsigned long。

  パラメータとして指定された時間（ミリ秒）の間、プログラムを一時停止します。（1 秒には 1000 ミリ秒があります。）

## 5.4 3. スピードアップ

デジタル信号（HIGH/LOW）に加えて、L298N の入力は PWM 信号も受け取ることができ、これにより出力の速度を制御することができます。

つまり、 `AnalogWrite()` を使用して車の移動速度を制御することができます。

このプロジェクトでは、車が前進する速度を徐々に変えるようにしています。最初は加速し、その後は減速します。

**配線図**

このプロジェクトの配線は、 *2.* **コードでの移動** と同じです。

**コード**

---

**注釈:**

- `3in1-kit\car_project\3.speed_up` のパスの下で `3.speed_up.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

プログラムが実行されると、車は徐々に加速し、その後徐々に減速します。

**どのように動作するのか？**

このプロジェクトの目的は、IN1~IN4 に異なる PWM の値を書き込むことで、車の前進速度を制御することです。

1. `for()` 文を使用して、0 から 255 までの値を 5 のステップで `speed` に与え、車の前進速度の変化を確認します。

```
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. `moveForward()` 関数について。

2. **コードでの移動** では IN1~IN4 に直接高/低のレベルを与えていましたが、ここでは高いレベルを与える必要がある場所に speed パラメータを渡します。

```
void moveForward(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, speed);
    analogWrite(in3, speed);
    analogWrite(in4, 0);
}
```

- for

for 文は、波括弧で囲まれた文のブロックを繰り返し実行するために使用されます。増分カウンタは通常、ループを増分および終了するために使用されます。

```
for (initialization; condition; increment) {
// statement(s);
}
```

- initialization: 最初に一度だけ実行されます。
- condition: ループを通るたびに、condition がテストされます。true であれば、文のブロックと増分が実行され、その後で再び condition がテストされます。condition が false になると、ループは終了します。
- increment: condition が true のとき、ループを通じて実行されます。

## 5.5 4. ラインフォロー

この車にはライントラックモジュールが装備されており、黒いラインに沿って車を走らせることができます。

ラインフォローモジュールが黒いラインを検出すると、右のモーターが回転し、左のモーターは回転しないため、車は左前方に一歩移動します。車が移動すると、ラインモジュールはラインから外れ、次に左のモーターが回転し、右のモーターは回転しないため、車は右に一歩移動してラインに戻ります。上記の 2 つのステップを繰り返すことで、車は黒いラインに沿って移動します。

プロジェクトを開始する前に、黒いラインテープでカーブマップを作成する必要があります。推奨されるラインの幅は 0.8-1.5cm で、曲がり角の角度は 90 度未満であってはなりません。

**必要な部品**

このプロジェクトには以下の部品が必要です。

全セットを購入するのは確かに便利です、リンクはこちら：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

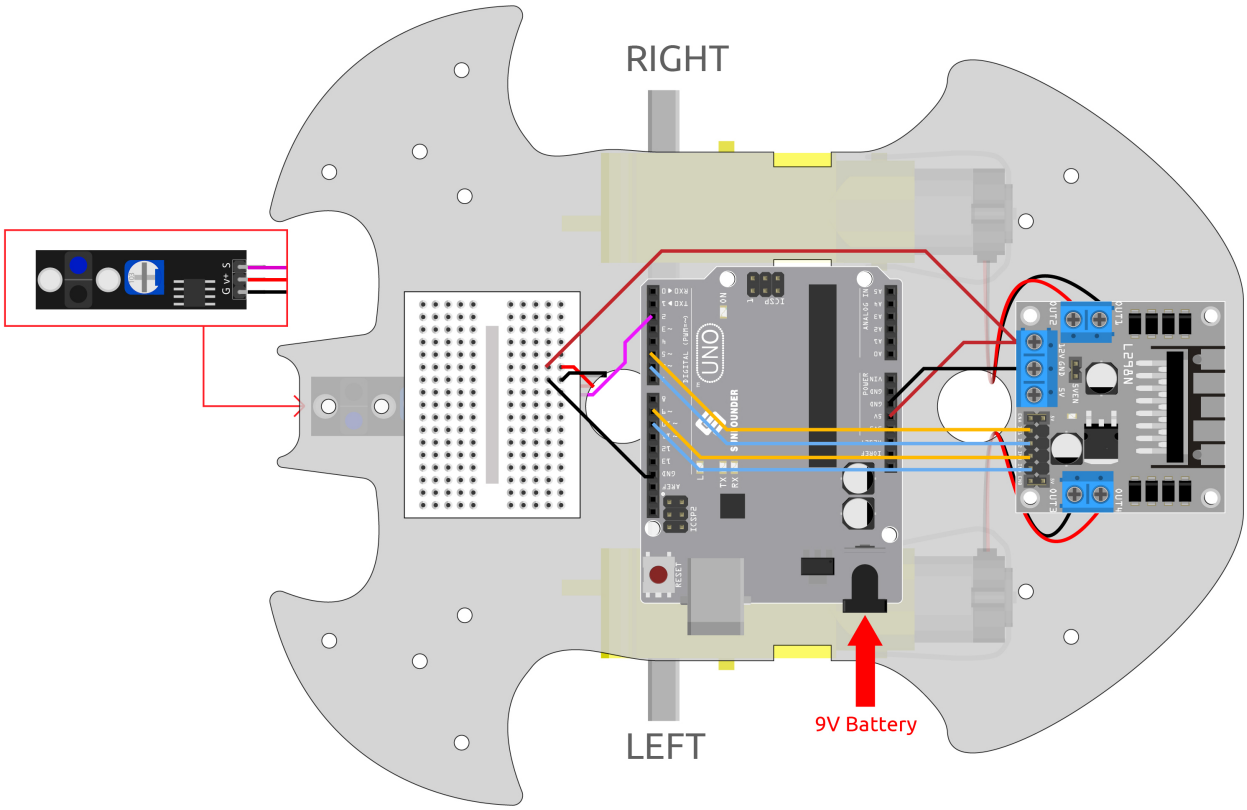
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| ライン追跡モジュール | |

### 配線図

これはデジタルライントラッキングモジュールで、黒いラインが検出されると 1 を出力し、白いラインが検出されると 0 の値を出力します。さらに、モジュール上のポテンショメータを通じて感知距離を調整することができます。

以下の図に従って回路を組み立ててください。

| ライントラッキングモジュール | R3 ボード |
| --- | --- |
| S | 2 |
| V+ | 5V |
| G | GND |




### モジュールの調整

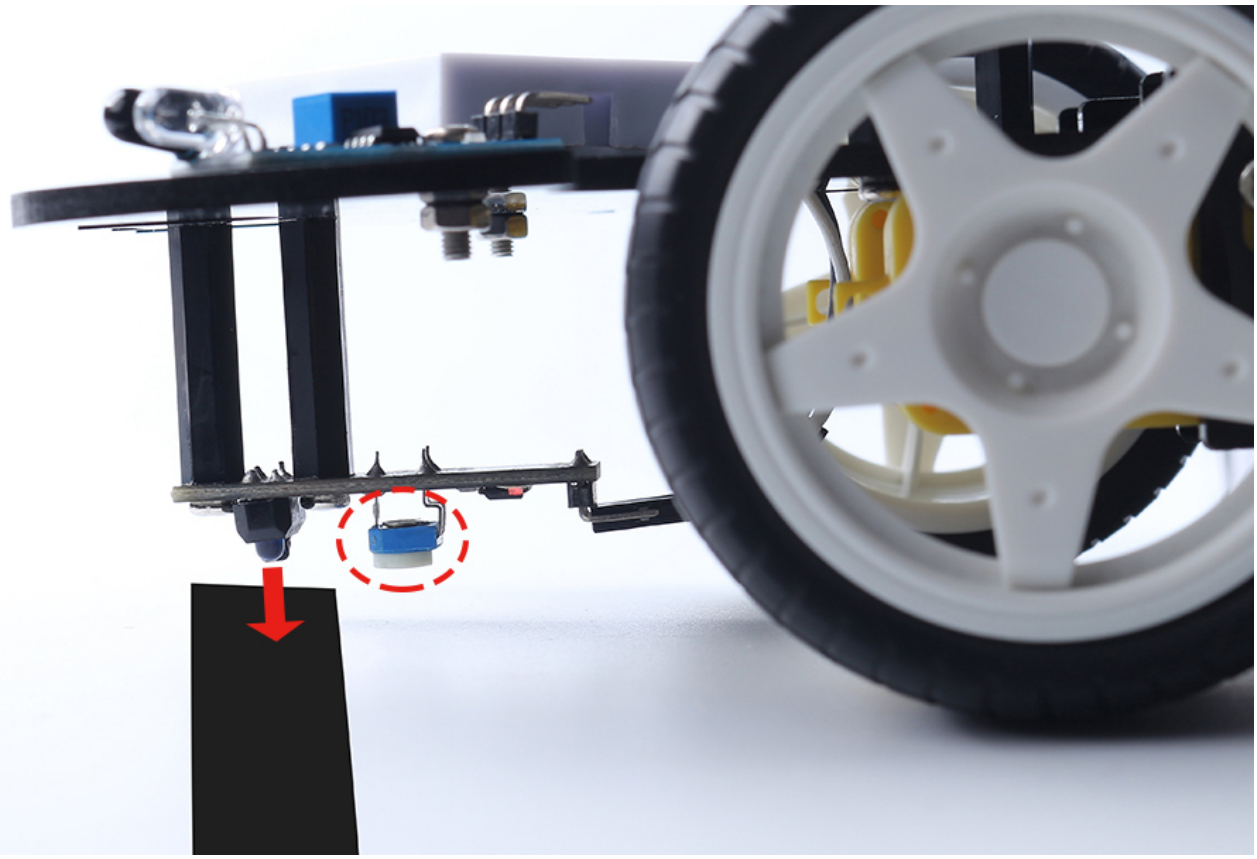
プロジェクトを開始する前に、モジュールの感度を調整する必要があります。

上記の図に従って配線を行い、R3 ボードに電源を供給します（USB ケーブルを直接接続するか、9V のバッテリーボタンケーブルを使用）。コードをアップロードしないでください。

テーブルに黒い電気テープを貼り、その上にカートを置きます。

モジュールの信号 LED を観察して、白いテーブル上で点灯し、黒いテープ上で消灯することを確認します。

もしそうでない場合、モジュール上のポテンショメータを調整して、上記の効果を得るようにします。



**コード**

---

**注釈:**

- 3in1-kit\car_project\4.follow_the_line のパスの下で 4.follow_the_line.ino ファイルを開きます。
- あるいは、このコードを **Arduino IDE** にコピーします。
- また、 Arduino Web Editor を通じてコードをアップロードすることもできます。

---

R3 ボードにコードをアップロードした後、車の下のライントラッキングモジュールを黒いラインに合わせると、車がラインを追従するのを見ることができます。

**どのように動作するのか？**

このコードでは、ライントラックモジュールの値に応じて、2 つのモーターを微調整して左右に回転させることで、車が黒いラインを追従するのを見ることができます。

1. ライントラッキングモジュールのピン定義を追加し、 INPUT として設定します。ここではシリアルモニタも初期化し、ボーレートを 9600bps に設定します。

```
...
const int lineTrack = 2;
Serial.begin(9600);
void setup() {
    ...
    pinMode(lineTrack, INPUT);
}
```

2. ライントラッキングモジュールの値を読み取ります。もし値が 1 ならば、車を左に前進させます。それ以外の場合は、右に前進させます。また、右上の虫眼鏡アイコンをクリックしてシリアルモニタを開くことで、USB ケーブルを抜く前に黒と白のライン上でのライントラッキングモジュールの値の変化を見ることができます。

```
void loop() {

    int speed = 150;

    int lineColor = digitalRead(lineTrack); // 0: 白 1: 黒
    Serial.println(lineColor);
    if (lineColor) {
        moveLeft(speed);
    } else {
        moveRight(speed);
    }
}
```

3. moveLeft() および moveRight() 関数について。

   プロジェクト *2.* **コードでの移動** の左右のターン機能とは異なり、ここでは小さな左右のターンだけが必要です。したがって、IN2 または IN3 の値を毎回調整するだけでよいです。例えば、左前に移動する場合（ moveLeft() ）、IN2 のスピードを設定して、他のすべてを 0 に設定するだけで、右のモーターは時計回りに回転し、左のモーターは動かないようになります。

```
void moveLeft(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, speed);
    analogWrite(in3, 0);
    analogWrite(in4, 0);
}
```

(次のページに続く)

(前のページからの続き)

```
void moveRight(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, 0);
    analogWrite(in3, speed);
    analogWrite(in4, 0);
}
```

- Serial

  Arduino ボードとコンピュータや他のデバイスとの通信に使用されます。

  - `Serial.begin()`: シリアルデータ伝送のビットレート（ボーレート）を設定します。
  - `Serial.println()`: データをシリアルポートに人間が読める ASCII テキストとして印刷し、次にキャリッジリターン文字（ASCII 13、または 'r'）および改行文字（ASCII 10、または 'n'）が続きます。

- if else

  `if else` ステートメントは、基本的な if ステートメントよりもコードの流れをより制御することができます。複数のテストをグループ化できます。

## 5.6 5. 障害物回避モジュールで遊ぶ

車の前部には 2 つの赤外線障害物回避モジュールが取り付けられており、近くの障害物を検出するのに使用できます。

このプロジェクトでは、車は自由に前進することができ、障害物に遭遇した場合、それを回避して他の方向に移動を続けることができます。

**必要な部品**

このプロジェクトには、以下のコンポーネントが必要です。

全てのキットをまとめて購入するのは非常に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから別々に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 障害物回避モジュール | |

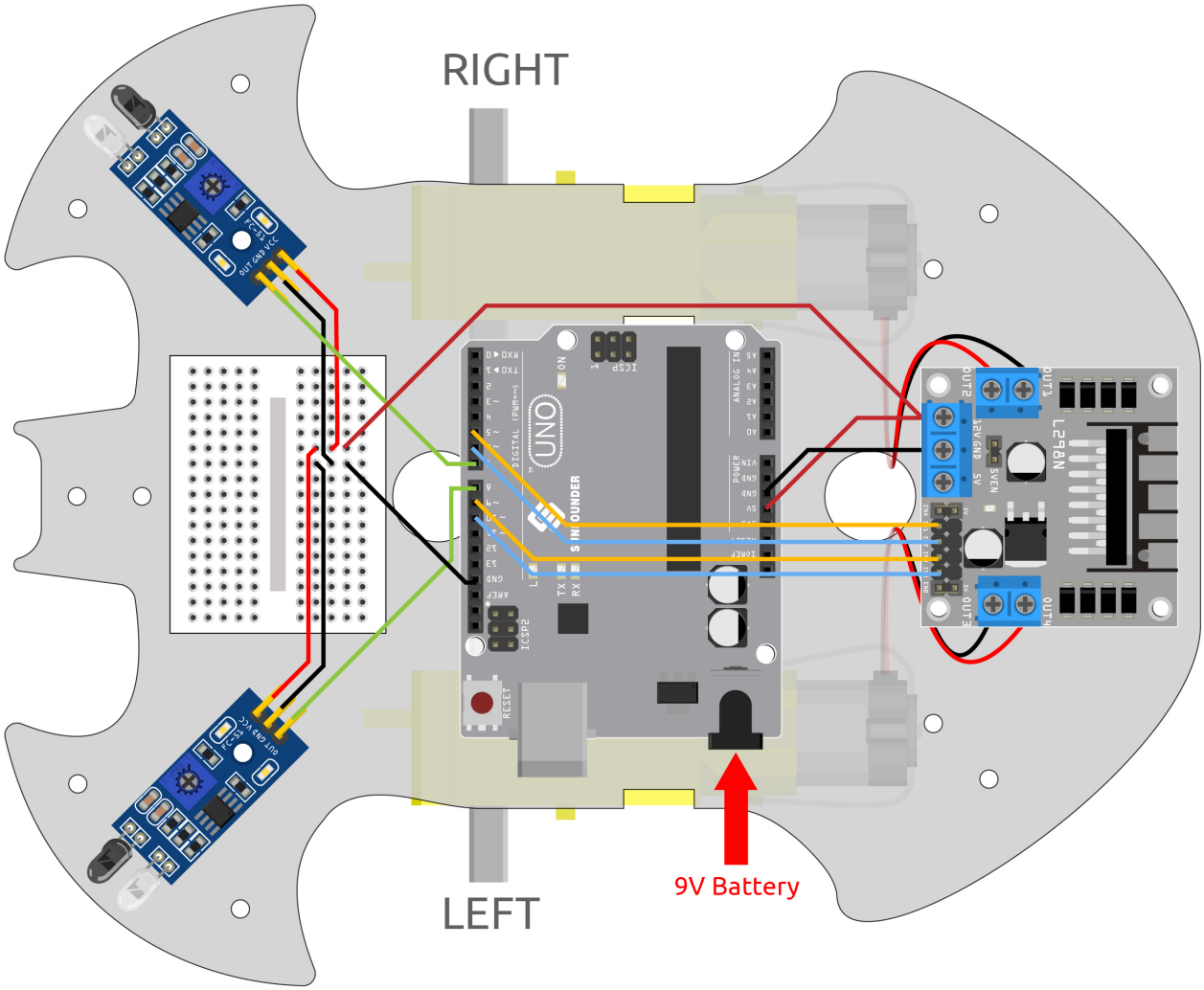
### 配線図

障害物回避モジュールは、距離を調整可能な赤外線近接センサーであり、障害物を検出した場合には通常の出力は高く、低くなります。

以下の図に従って回路を作成してください。

| 左 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 8 |
| GND | GND |
| VCC | 5V |

| 右 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 7 |
| GND | GND |
| VCC | 5V |



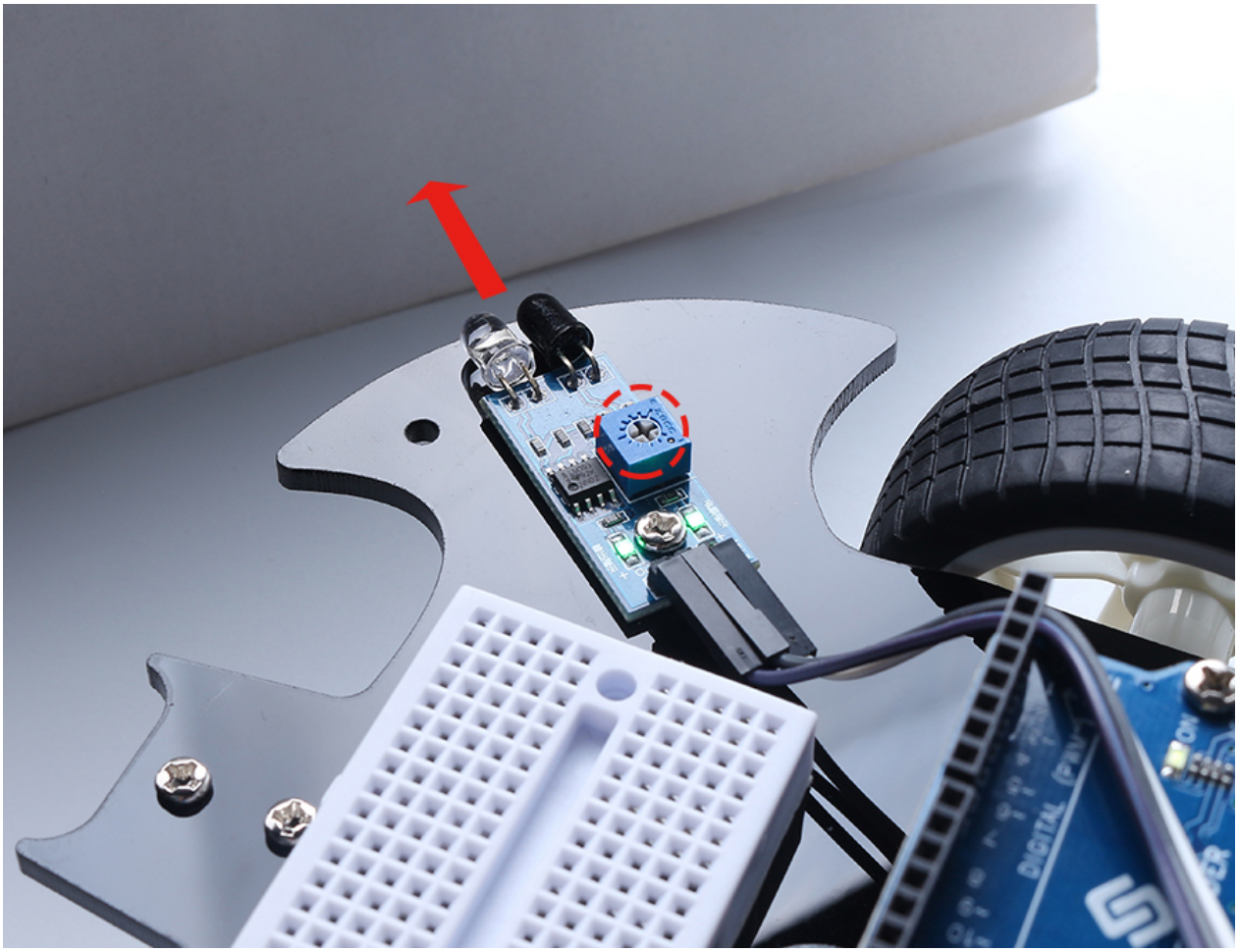

**モジュールの調整**

プロジェクトを開始する前に、モジュールの検出距離を調整する必要があります。

上記の図に従って配線し、R3 ボードに電源を供給してください（USB ケーブルを直接接続するか、9V の電池ケーブルを取り付けることによって）。コードをアップロードせずに、IR 障害物回避の前に約 5cm のノートや他の平らな物を置きます。

次に、モジュールの信号インジケータがちょうど点灯するまで、モジュール上のポテンショメータをドライバーで回して、最大検出距離を 5cm に調整します。

もう一つの赤外線モジュールも同じ方法で調整します。



**コード**

---

**注釈:**

- 3in1-kit\car_project\5.obstacle_avoidance_module のパスの下で 5.obstacle_avoidance_module.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされると、車は前進します。左の赤外線モジュールが障害物を検出すると、車は左に後退します。右の赤外線モジュールが障害物を検出すると、車は右に後退します。両方の側面で障害物を検出すると、車はまっすぐ後退します。

**どのように動作するか？**

このプロジェクトは、左右の赤外線障害物回避モジュールの値に基づいて、車に適切な動作をさせるものです。

1. 2 つの障害物回避モジュールのピン定義を追加します。ここでは、 INPUT に設定されています。

```
...
const int rightIR = 7;
const int leftIR = 8;

void setup() {
...

//IR 障害物
    pinMode(leftIR, INPUT);
    pinMode(rightIR, INPUT);
}
```

2. 左右の赤外線モジュールの値を読み取り、車に対応する動作をさせます。

```
void loop() {

    int left = digitalRead(leftIR);   // 0: 障害物あり  1: 空
    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);
    }
}
```

- 左の IR モジュールが 0（障害物検出）で、右の IR モジュールが 1 の場合、車は左に後退します。
- 右の IR モジュールが 0（障害物検出）の場合、車は右に後退します。
- 2 つの IR モジュールが同時に障害物を検出すると、車は後退します。
- それ以外の場合、車は前進し続けます。

3. backLeft() 関数について。

   右のモーターが反時計回りに回転し、左のモーターが回転していない場合、車は左に後退します。

```
void backLeft(int speed) {
    analogWrite(in1, speed);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, 0);
}
```

4. backLeft() 関数について。

   左のモーターが時計回りに回転し、右のモーターが回転していない場合、車は右に後退します。

```
void backRight(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, speed);
}
```

- &&: 両方のオペランドが真の場合にのみ、論理 AND は真を返します。
- !: オペランドが偽の場合、論理 NOT は真を返します。

## 5.7 6. 超音波モジュールで遊ぶ

*5.* **障害物回避モジュールで遊ぶ** プロジェクトでは、2 つの赤外線障害物回避モジュールを障害物回避のために使用していますが、IR 障害物回避モジュールの検出距離は短いため、車が障害物を避けるのが遅くなる可能性があります。

このプロジェクトでは、超音波モジュールを使って遠距離の検出を行い、車がもっと遠くの障害物を感知して判断できるようにします。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

全体のキットを購入すると非常に便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

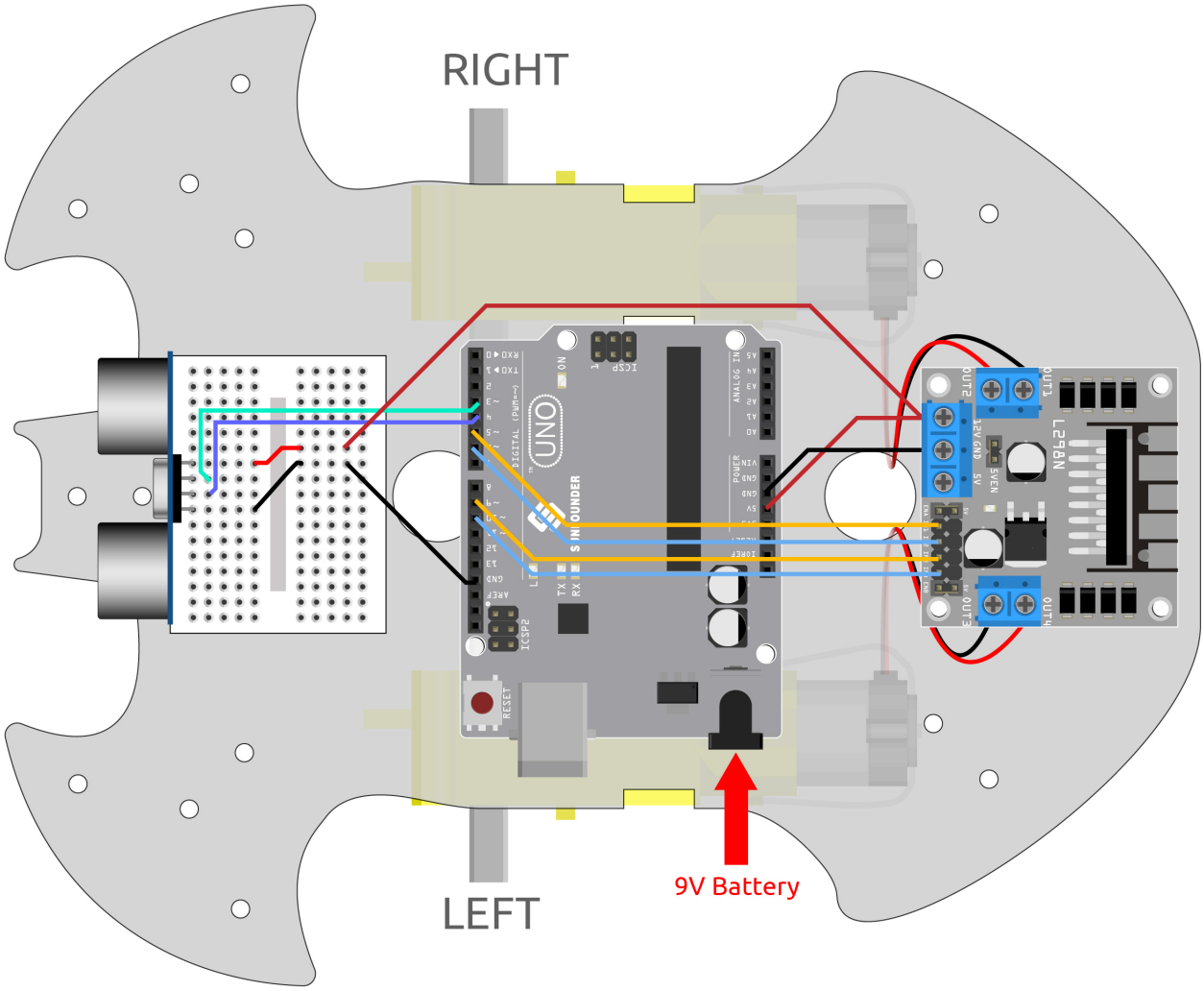
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 超音波モジュール | |

### 配線図

超音波センサモジュールは、超音波を使って物体までの距離を測定する機器です。2 つのプローブがあります。1 つは超音波を送信し、もう 1 つは超音波を受信して送受信の時間を距離に変換し、装置と障害物との間の距離を検出します。

以下の図に従って回路を組み立ててください。

| 超音波モジュール | R3 ボード |
| --- | --- |
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |




**コード**

---

**注釈:**

- 3in1-kit\car_project\6.ultrasonic_module のパスの下で 6.ultrasonic_module.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされた後、車を壁に向けてください。距離が遠すぎる場合は前進し、近すぎる場合は後退し、安全な距離であれば停止します。

**どのように動作するのか？**

このプロジェクトは、超音波モジュールから読み取った距離に基づいて、車に適切な動作をさせるものです。

1. 超音波モジュールのピン定義を追加します。 trigPin は超音波を送信するために使用され、 OUTPUT に設定されます。 echoPin は超音波を受信するために INPUT に設定されます。

```
...
const int trigPin = 3;
const int echoPin = 4;

void setup() {
...

//超音波
    pinMode(echoPin, INPUT);
    pinMode(trigPin, OUTPUT);
}
```

2. まず、超音波モジュールから取得した距離の値を読み取ります。距離が 25 より大きい場合、車を前進させます。距離が 2~10cm の間であれば、車を後退させます。それ以外の場合（10~25 の間）、停止します。

```
void loop() {
    float distance = readSensorData();
    if (distance > 25) {
        moveForward(200);
    }
    else if (distance < 10 && distance > 2) {
        moveBackward(200);
    } else {
        stopMove();
    }
}
```

3. readSensorData() 関数について。

   超音波モジュールの送信機は、2us ごとに 10us の正方形の波形信号を送信し、範囲内に障害物がある場合、受信機は高レベル信号を受信します。 pulseIn() 関数を使用して、送信から受信までの時間を記録し、音速 340m/s で除算し、さらに 2 で除算すると、このモジュールと障害物との距離が cm 単位で得られます。

```
float readSensorData() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
```

(次のページに続く)

(前のページからの続き)

```
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    float distance = pulseIn(echoPin, HIGH) / 58.00; //(340m/s*1us)/2 に相当。
    return distance;
}
```

- pulseIn(pin, value)
  - pin: パルスを読みたい Arduino のピンの番号。許可されているデータ型: int。
  - value: 読みたいパルスの種類：HIGH または LOW。許可されているデータ型: int。

  ピン上でのパルス（HIGH または LOW）を読み取ります。例えば、value が HIGH の場合、pulseIn() は、ピンが LOW から HIGH になるのを待ち、タイミングを開始し、次にピンが LOW になるのを待ち、タイミングを停止します。

## 5.8 7. 手を追いかける車

この車をペットと考えてください。あなたが手を振ると、その車はあなたのもとへと駆けてきます。

### 必要な部品

このプロジェクトでは、以下の部品が必要です。

キットを購入するのが便利です。以下がリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

下記のリンクから部品を個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 超音波モジュール | |
| 障害物回避モジュール | |

### 配線図

超音波モジュールと 2 つの IR 障害物回避モジュールを同時に接続します。

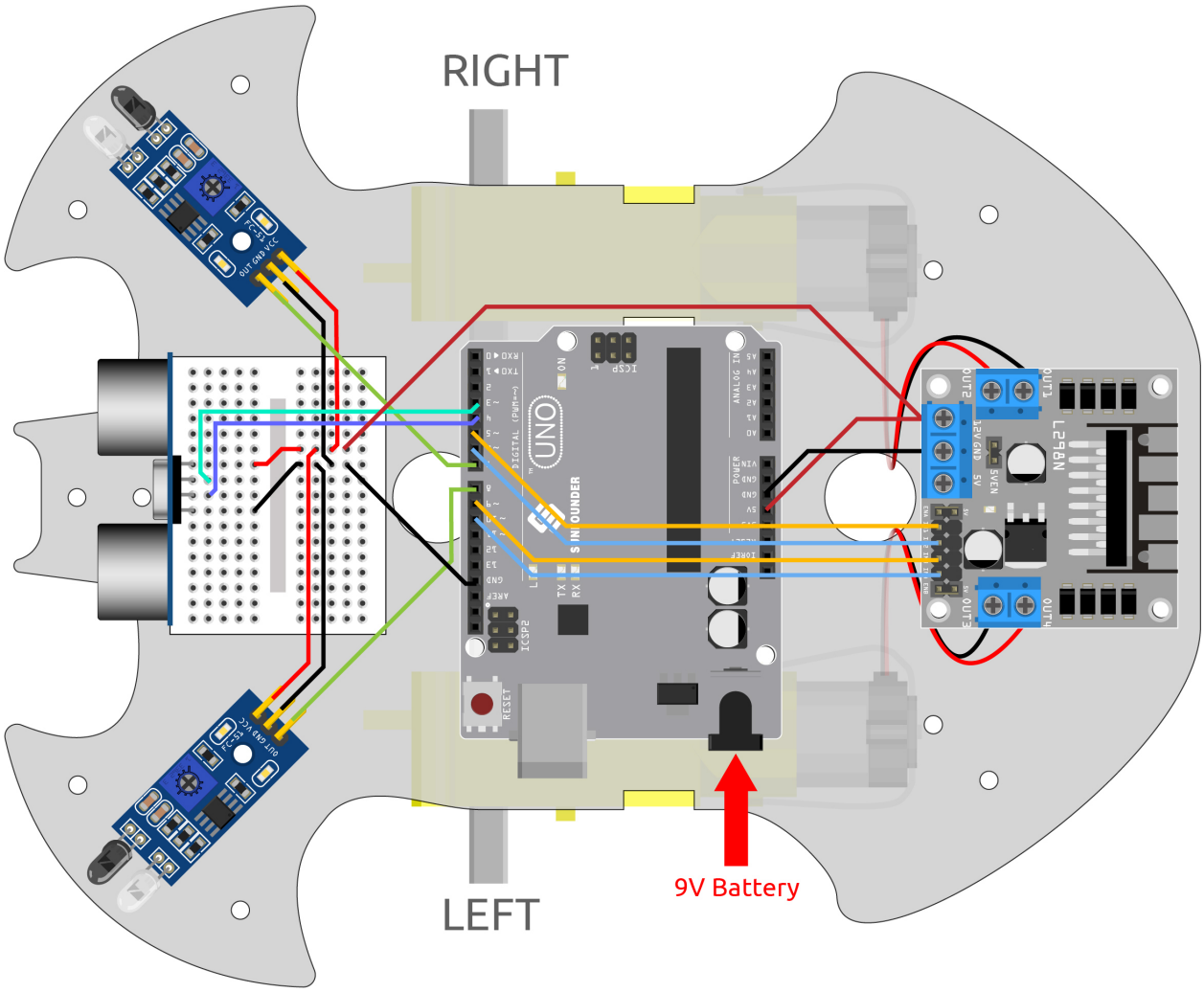
R3 ボードに超音波を次のように接続します。

| 超音波モジュール | R3 ボード |
| --- | --- |
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |

2 つの IR 障害物回避モジュールの R3 ボードへの配線は次の通りです。

| 左 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 8 |
| GND | GND |
| VCC | 5V |

| 右 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 7 |
| GND | GND |
| VCC | 5V |




**コード**

---

**注釈:**

- 3in1-kit\car_project\7.follow_your_hand のパスの下で 7.follow_your_hand.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーペーストします。
- あるいは、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされたら、車を地面に置きます。車の前に手を 5～10cm 程度近づけると、車は手の方向に進むように追従します。IR 障害物モジュールの両側に手を近づけると、対応する方向にも曲がります。

**どのように動作するのか？**

このプロジェクトは以前の二つのプロジェクト、 *6.* **超音波モジュールで遊ぶ** と *5.* **障害物回避モジュールで遊ぶ**

の組み合わせですが、実現される効果は異なります。以前の 2 つのプロジェクトは障害物を後ろで検出していましたが、ここでは手が前方または回転方向に追従するのを検出します。このプロジェクトのワークフローは次のとおりです。

- 超音波モジュールによって検出された距離と両方の赤外線モジュールの値を読み取ります。
- 距離が 5〜10cm の場合、車を手に合わせて動かします。
- 左の IR モジュールが手を検出した場合、左に曲がります。
- 右の IR モジュールが手を検出した場合、右に曲がります。
- 赤外線モジュールも超音波モジュールも手を検出しない場合、車を停止させます。

```
void loop() {

    float distance = readSensorData();

    int left = digitalRead(leftIR);   // 0: 遮断 1: 空
    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. 自動運転車

このプロジェクトは、*6.* **超音波モジュールで遊ぶ** と *5.* **障害物回避モジュールで遊ぶ** の二つのプロジェクトを組み合わせたものです。2 つの赤外線障害物回避モジュールは、短距離やエッジの検出を行い、超音波モジュールは長距離検出を行って、自由な運転過程で車が障害物にぶつからないことを確認します。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

全体のキットを購入するのが確かに便利です、リンクはこちらです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから、部品を個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 超音波モジュール | |
| 障害物回避モジュール | |

### 配線図

超音波モジュールと 2 つの IR 障害物回避モジュールを同時に接続します。

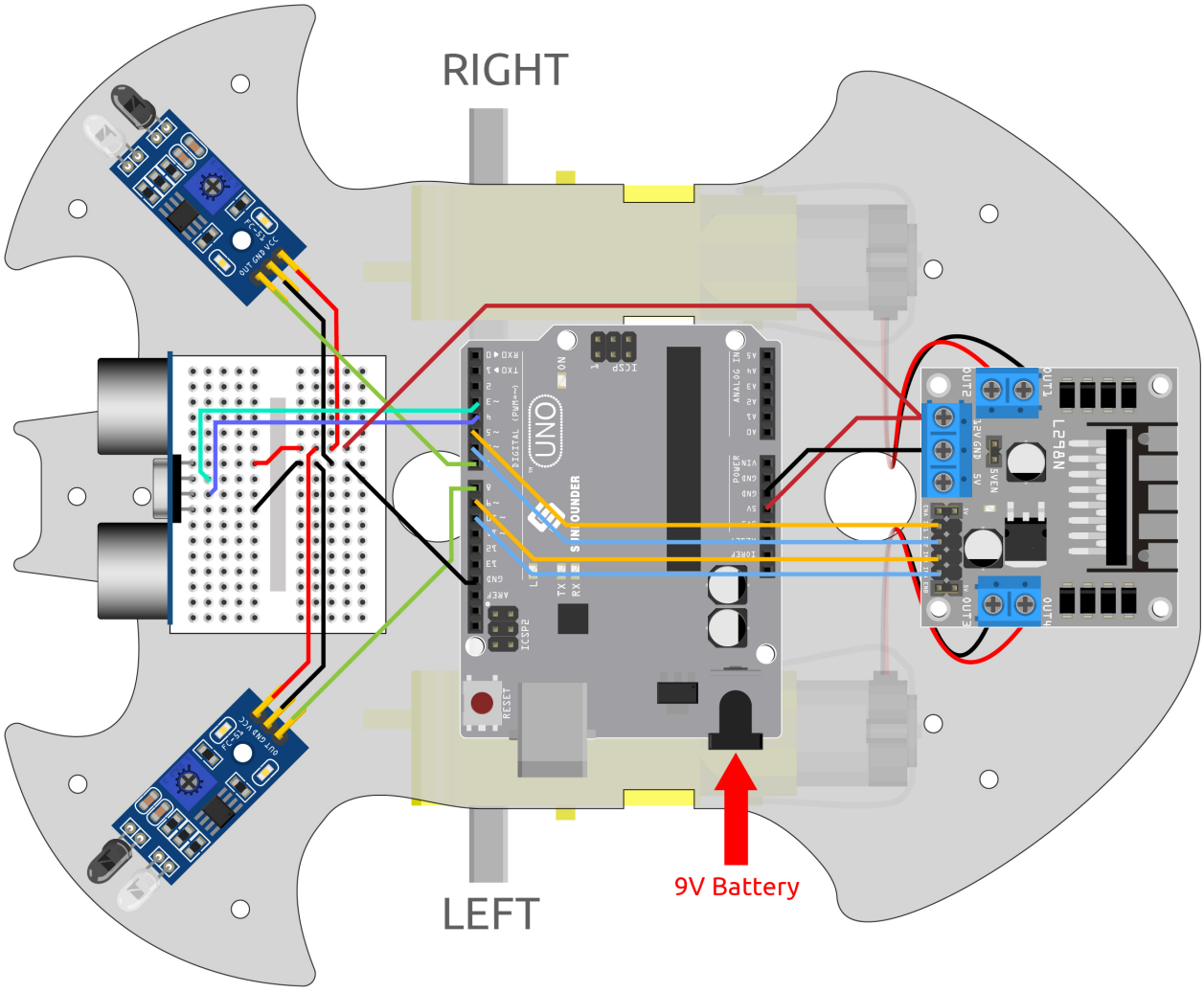
超音波を R3 ボードに以下のように配線します。

| 超音波モジュール | R3 ボード |
| --- | --- |
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |

2 つの IR 障害物回避モジュールの R3 ボードへの配線は以下の通りです。

| 左 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 8 |
| GND | GND |
| VCC | 5V |

| 右 IR モジュール | R3 ボード |
|---|---|
| OUT | 7 |
| GND | GND |
| VCC | 5V |




**コード**

---

**注釈:**

- `3in1-kit\car_project\8.self_driving_car` のパスの下の `8.self_driving_car.ino` ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- または、 Arduino Web Editor を通じてコードをアップロードします。

---

コードが正常にアップロードされると、車は自由に走り始めます。両側の IR 障害物モジュールが障害物を検出すると、緊急回避のために逆方向に移動します。もし車の正面 2~10cm 以内に障害物がある場合、左にバックアップして方向を調整し、その後前進します。

**どのように動作するのか？**

このプロジェクトのワークフローは以下の通りです。

- 左右の IR 障害物回避モジュールの値を優先して読み取ります。
- 左の IR モジュールが 0（障害物検出）、右の IR モジュールが 1 の場合、車を左にバックアップさせます。
- 右の IR モジュールが 0（障害物検出）の場合、車を右にバックアップさせます。
- 2 つの IR モジュールが同時に障害物を検出する場合、車はバックアップします。
- それ以外の場合は、超音波モジュールによって検出された距離を読み取ります。
- 距離が 50cm 以上の場合、車を前進させます。
- 距離が 2-10cm の場合、転回する前に車を後退させます。
- 距離が 10-50cm の場合、車を低速で前進させます。

```
void loop() {

    int left = digitalRead(leftIR);   // 0: 遮断 1: 空
    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) { // 安全
            moveForward(200);
        } else if (distance < 10 && distance > 2) { // 注目
            moveBackward(200);
            delay(1000);
            backLeft(150);
            delay(500);
```

(次のページに続く)

(前のページからの続き)

```
        } else {
            moveForward(150);
        }
    }
}
```

## 5.10 9. リモートコントロール

このキットには、IR レシーバーが含まれているので、IR リモートコントロールを使用して車の動きをコントロールすることができます。

### 必要な部品

このプロジェクトには、以下のコンポーネントが必要です。

キット全体を購入するのが便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクからそれぞれのアイテムを購入することもできます。

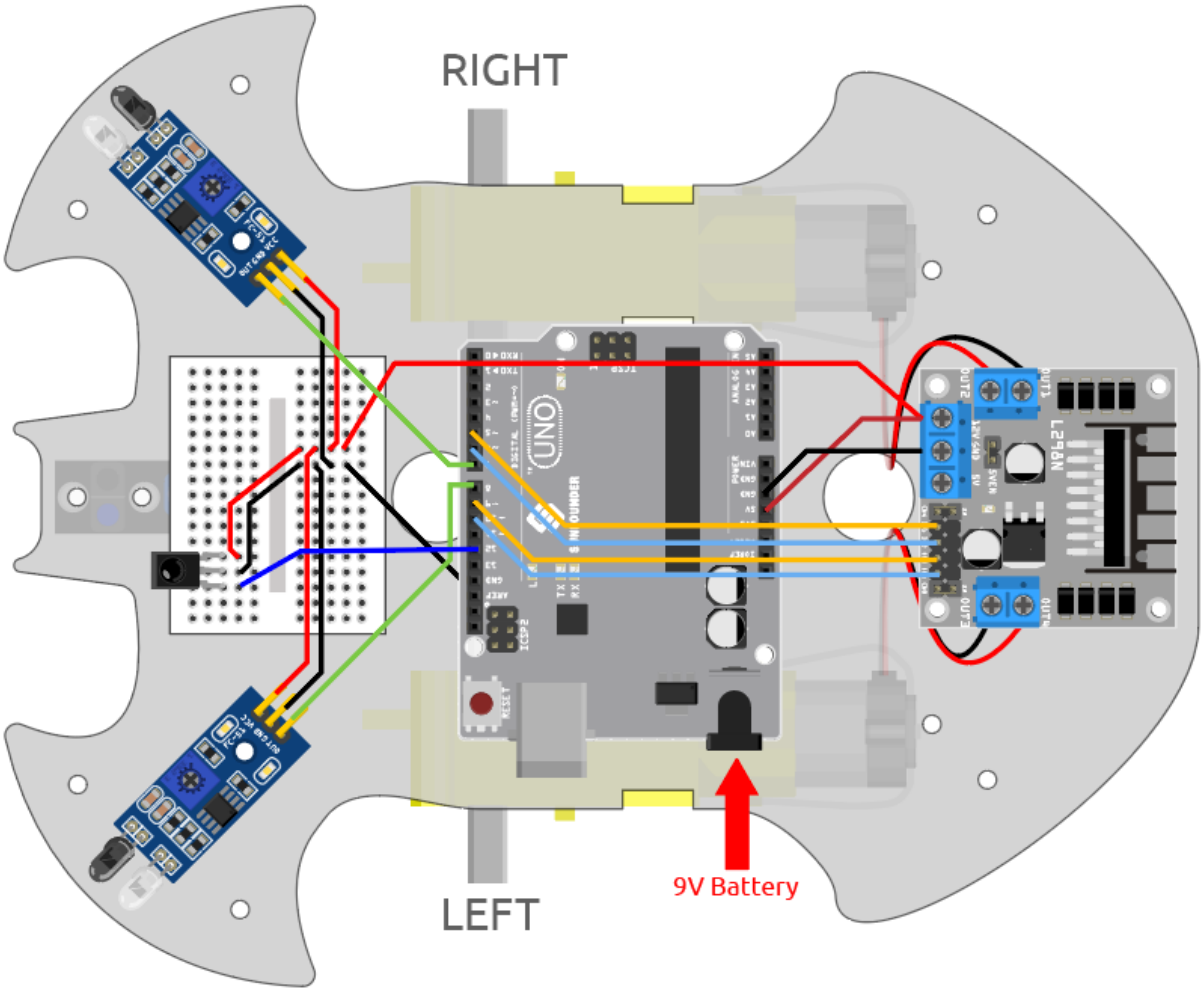
| コンポーネント紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| *LED* | |
| *IR* レシーバー | - |

### 配線図

以下の図に従って回路を組み立ててください。

| IR レシーバー | R3 ボード |
|---|---|
| OUT | 12 |
| GND | GND |
| VCC | 5V |

| LED | R3 ボード |
|---|---|
| アノード (長いピン) | 13 |
| カソード | GND |



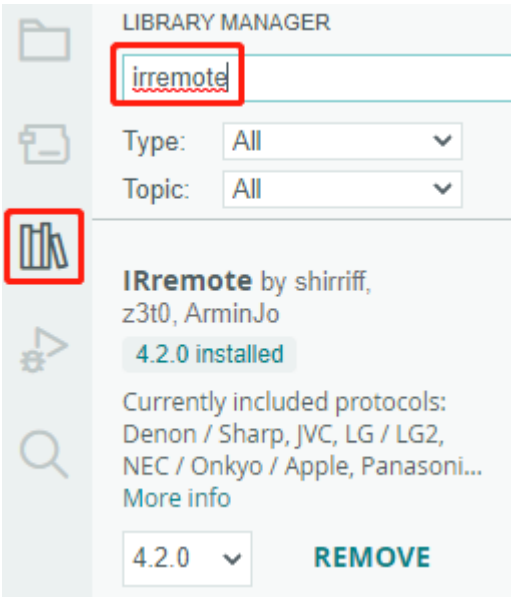

**コード**

**注釈:**

- 3in1-kit\car_project\9.remote_control のパスにある 9.remote_control.ino ファイルを開きま

す。

- または、このコードを **Arduino IDE** にコピーします。
- ここでは `IRremote` ライブラリが使用されています。 **Library Manager** からインストールできます。




---

コードのアップロードが成功したら、リモートコントロールのボタンを押すと、LED が一回点滅して信号が受信されたことを示し、ボタンの指示に従って車が動きます。以下のキーを押して車を操作することができます。

- **+**：加速
- **-**：減速
- **1**：左前方へ進む
- **2**：前進
- **3**：右前方へ進む
- **4**：左折
- **6**：右折
- **7**：左後方へ後退
- **8**：後退
- **9**：右後方へ後退

**どのように動作するのか？**

このプロジェクトの効果は、IR リモートコントロールのキー値を読み取り、車を動かすことです。さらに、ピン 13 の LED が点滅して、赤外線信号の受信が成功したことを示します。

1. IRremote ライブラリをインポートします。 **Library Manager** からインストールできます。

```
#include <IRremote.h>

const int IR_RECEIVE_PIN = 12;  // IR センサのピン番号を定義する
```

2. ボーレート 9600 でシリアル通信を初期化します。指定されたピン (IR_RECEIVE_PIN) で IR レシーバを初期化し、LED フィードバックを有効にします (該当する場合)。

```
...

void setup() {

    ...
    //IR リモート
    IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);  // IR レシーバを開始する
    Serial.println("REMOTE CONTROL START");

}
```

3. リモートコントロールのキーを押すと、赤外線受信機はどのキーが押されたかを知り、車は対応するキー値に従って動きます。

```
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);
```

(次のページに続く)

(前のページからの続き)

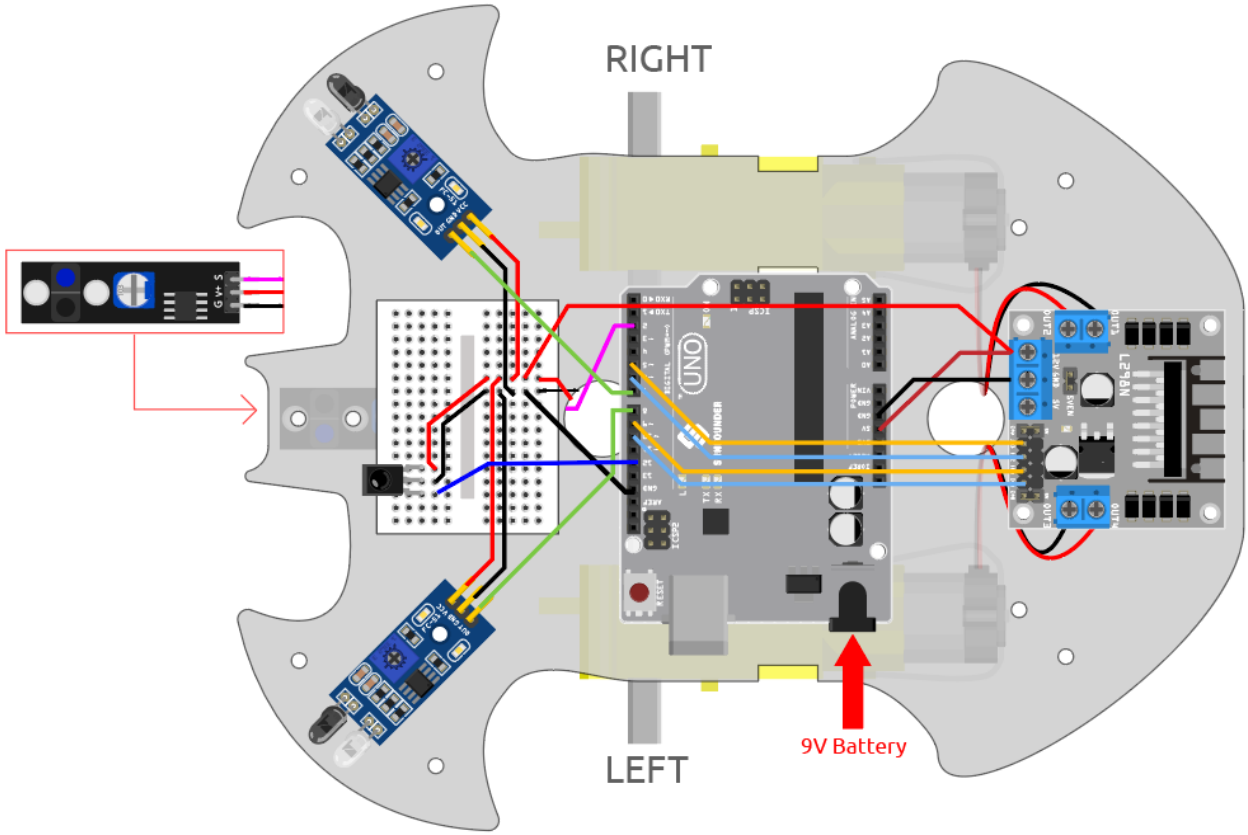
```
                delay(1000);
            ...
            }
            IrReceiver.resume();  // 次の値の受信を有効にする
    }
}
```

- IR 信号が受信され、正常にデコードされたかどうかを確認する。
- IR コマンドをデコードし、カスタム `decodeKeyValue()` 関数を使用して `key` に保存する。
- デコードされた値がエラーでないかを確認します。
- デコードされた IR 値をシリアルモニタに表示する。
- 次の信号の IR 信号受信を再開する。

## 5.11 10. ワンタッチスタート

このプロジェクトでは、これまでのプロジェクト（ライン追従、追跡、障害物回避、自動運転など）を統合しました。リモコンのボタンで切り替えることができるので、車をスタートしてすべての機能を一度に体験できます。

**配線図**

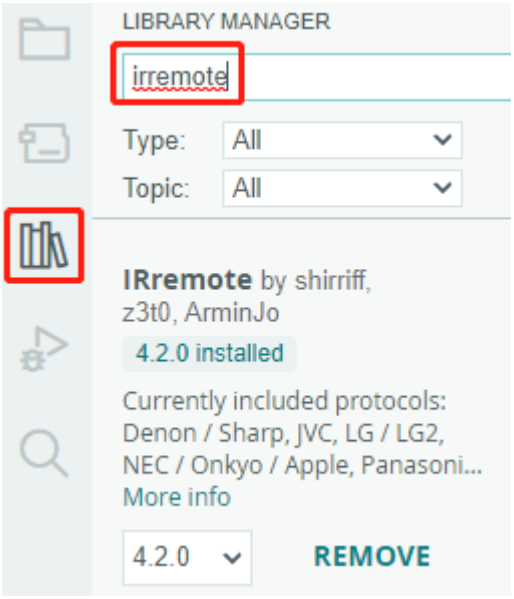



**コード**

---

**注釈:**

- 3in1-kit\car_project\10.one_touch_start のパスの下の 10.one_touch_start.ino ファイルを開きます。
- または、このコードを **Arduino IDE** にコピーします。
- ここでは IRremote ライブラリが使用されています。 **ライブラリマネージャー** からインストールできます。




コードが正常にアップロードされた後、IR レシーバーがリモートコントロールからの信号を受信するたびに、LED が 3 回急速に点滅します。カートを操作するための以下のキーを押すことができます。

- **+**: 加速
- **-**: 減速
- **1**: 左前方に移動
- **2**: 前進
- **3**: 右に移動
- **4**: 左を向く
- **6**: 右を向く
- **7**: 左に後退
- **8**: 後退
- **9**: 右に後退
- **CYCLE**: ラインを追従
- **U/SD**: 自動運転
- **|**: 超音波モジュールを使用した障害物回避
- **|**: IR 障害物モジュールを使用した障害物回避
- **EQ**: 手を追跡

- **0**: 停止

## 5.12 11. 速度のキャリブレーション

車を前進させる際、車がまっすぐ進まないことがあるかもしれません。これは、工場での 2 つのモーターの速度が同じでないためです。しかし、2 つのモーターにオフセットを書き込むことで、それらの回転速度を一致させることができます。

このプロジェクトでは、オフセットを EEPROM に保存する方法を学びます。この目的は、キャリブレーション後に、すべてのプロジェクトが EEPROM から直接オフセット値を取得できるようにするためです。これにより、車はスムーズに直線で進むことができます。

**配線図**

このプロジェクトの配線は *2.* **コードでの移動** と同じです。

**遊び方は？**

1. `3in1-kit\car_project\11.speed_calibration` のパス下の `11.speed_calibration.ino` ファイルを開く。または、このコードを **Arduino IDE** にコピーします。
2. コードが正常にアップロードされたら、車を 9V のバッテリーに接続し、地面に置いて前進させ、どちら側にオフセットされているかを確認します。

- 車が左前方に移動する場合、右モーターの速度が速すぎることを意味し、減速する必要があります。

```
EEPROM.write(1, 100) // 1 は右モータを意味し、100 は速度が 100% であることを意味します。
実際の状況に応じて 90、95 などに設定することができます。
```

- 車が右に移動する場合、左モーターの速度が速すぎることを意味し、減速する必要があります。

```
EEPROM.write(0, 100) // 0 は左モーターを意味し、100 は速度が 100% であることを意味します。
実際の状況に応じて 90、95 などに設定することができます。
```

3. コードを変更した後、R3 ボードにコードをアップロードして効果を確認します。車がほぼまっすぐになるまで上記の手順を繰り返します。
4. このオフセットは EEPROM に記録されます。他のプロジェクトで使用する際にこのオフセットを読み込むだけでよく、 *5.* **障害物回避モジュールで遊ぶ** が一例として取られます。

```
#include <EEPROM.h>

float leftOffset = 1.0;
float rightOffset = 1.0;
```

(次のページに続く)

(前のページからの続き)

```
const int in1 = 5;
const int in2 = 6;
const int in3 = 9;
const int in4 = 10;

const int rightIR = 7;
const int leftIR = 8;

void setup() {
    Serial.begin(9600);

    //motor
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, 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 {
```

(次のページに続く)

(前のページからの続き)

```
        moveForward(speed);
    }
}

void moveForward(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, int(speed * leftOffset));
    analogWrite(in3, int(speed * rightOffset));
    analogWrite(in4, 0);
}

void moveBackward(int speed) {
    analogWrite(in1, speed);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, speed);
}

void backLeft(int speed) {
    analogWrite(in1, speed);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, 0);
}

void backRight(int speed) {
    analogWrite(in1, 0);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, speed);
}
```

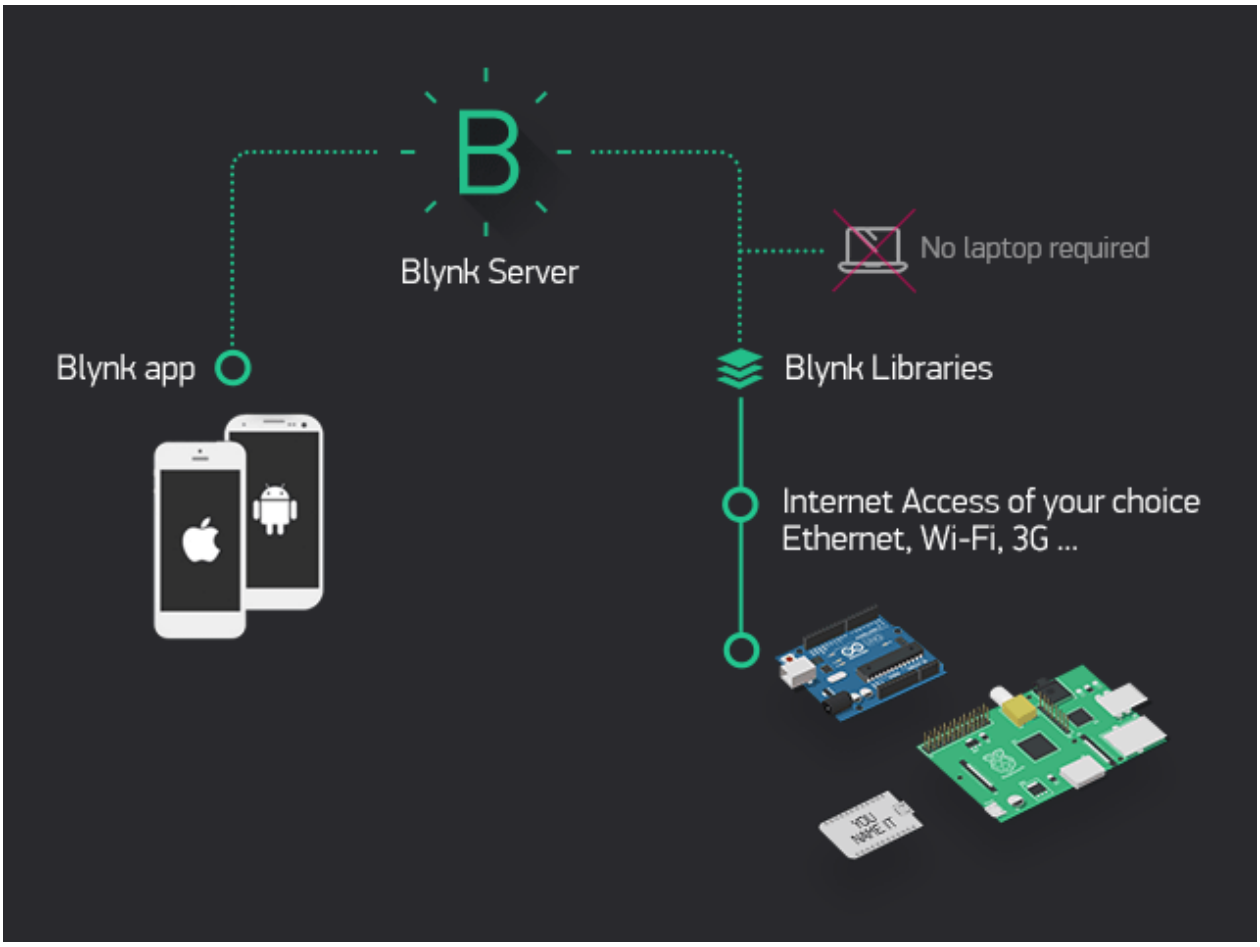
# 第 6 章

# IoT プロジェクト

このキットには、Arduino が IoT 実験のためにインターネットに接続することを可能にする ESP8266 Wifi モジュールが含まれています。

ここでは、ESP8266 Wifi モジュールを使用して Arduino を BLYNK プラットフォームに接続する方法をガイドします。これにより、興味深い IoT プロジェクトを実現できます。また、携帯電話の Blynk APP を使用してスマートカーを操作することもできます。

Blynk は、個人の IoT プロジェクトから数百万の商用接続製品まで、任意の規模での接続された電子デバイスのプロトタイプ制作、デプロイ、遠隔管理に必要なソフトウェアのフルセットです。Blynk を使用すると、誰でもハードウェアをクラウドに接続し、コードなしで iOS、Android、およびウェブアプリケーションを構築して、デバイスからのリアルタイムおよび履歴データを分析し、世界中のどこからでも遠隔操作することができます。また、重要な通知を受け取ることができ、さらに多くのことができます…




## 6.1 1. Blynk でのスタート

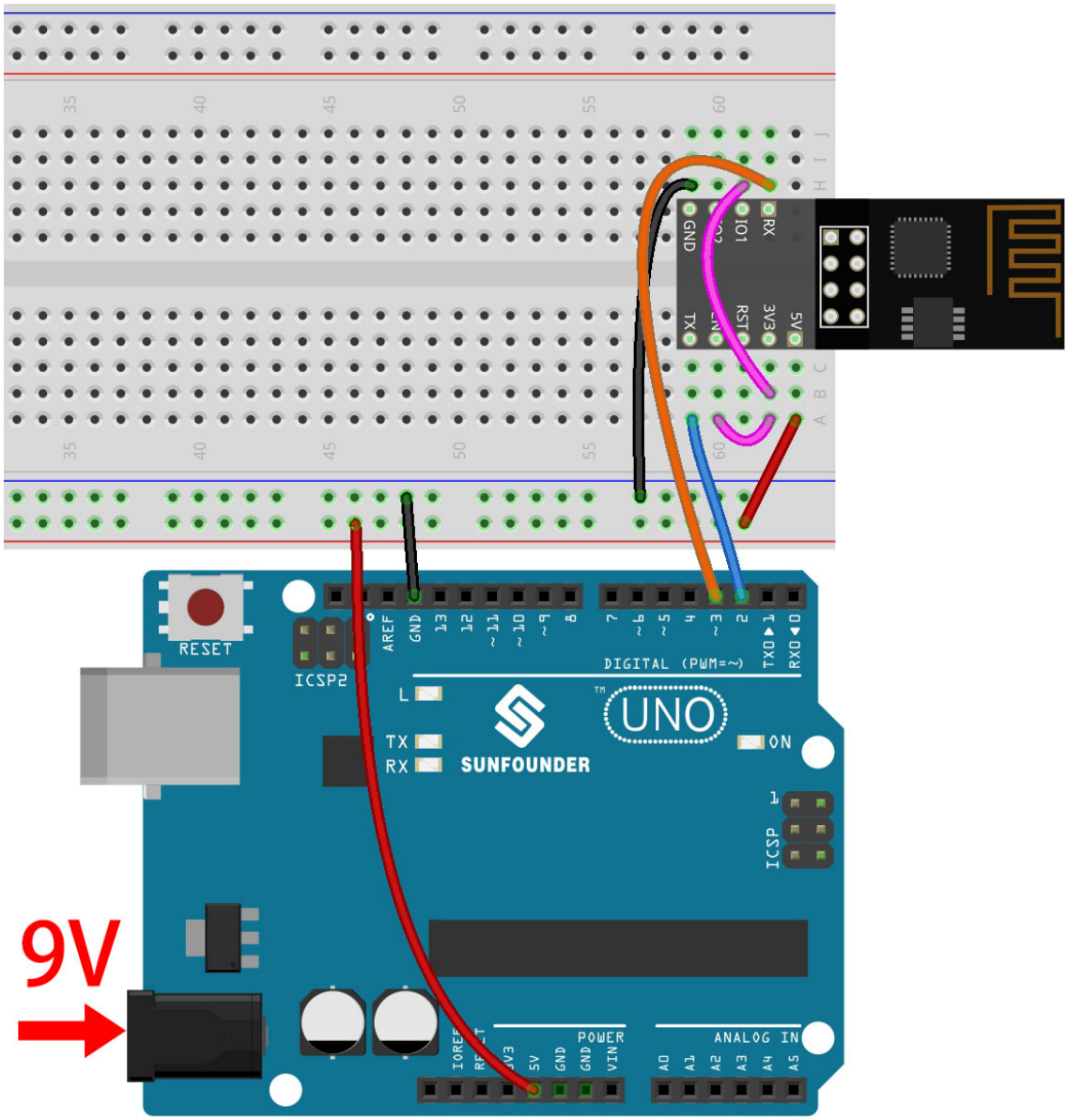
R3 ボードを Blynk と通信させるには、Blynk を初めて使用する際にいくつかの設定が必要です。

以下の手順に従ってください。そして、章を飛ばさず順番通りに行う必要があります。

### 6.1.1 1.1 ESP8266 の設定

このキットに同梱されている ESP8266 モジュールはすでに AT ファームウェアで書き込み済みですが、以下の手順に従って設定を変更する必要があります。

1. 回路を組み立てる。




2. 3in1-kit\iot_project\1.set_software_serial のパスの下にある 1.set_software_serial.ino ファイルを開くか、このコードを **Arduino IDE** にコピーします。

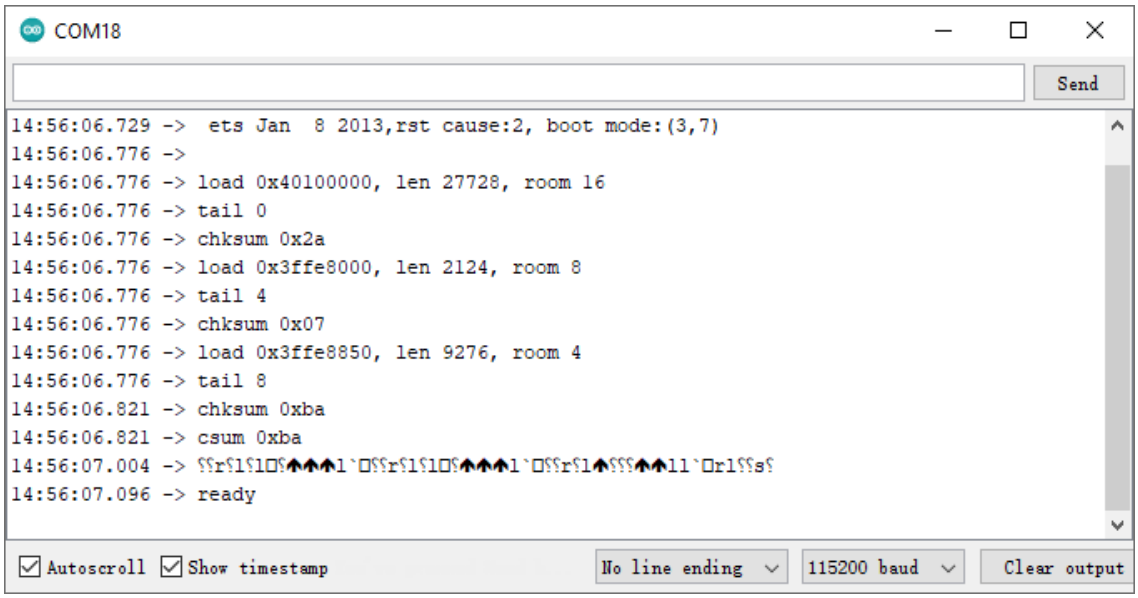
```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(2, 3); //Rx,Tx

void setup() {
    // セットアップコードをここに書いて、一度だけ実行する：
    Serial.begin(115200);
    espSerial.begin(115200);
}
```

(次のページに続く)

(前のページからの続き)

```
void loop() {
    if (espSerial.available()) {
        Serial.write(espSerial.read());
    }
    if (Serial.available()) {
        espSerial.write(Serial.read());
    }
}
```

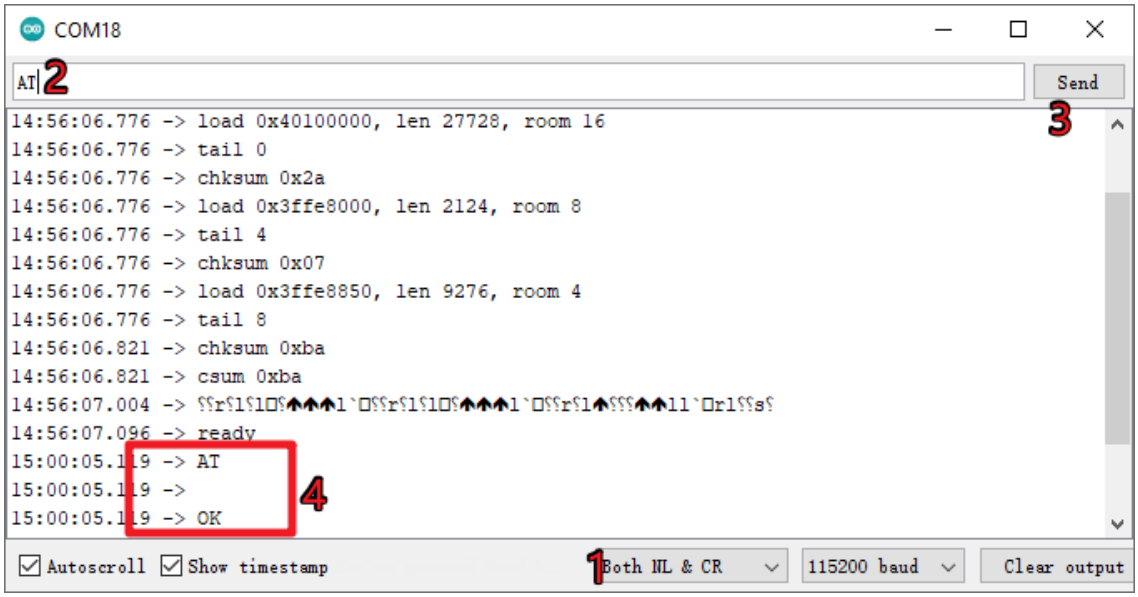
3. 右上の虫眼鏡アイコン（シリアルモニタ）をクリックし、ボーレートを **115200** に設定します。(私のようにいくつかの情報が表示されるか、表示されない場合もありますが、問題ありません。次のステップに進んでください。)



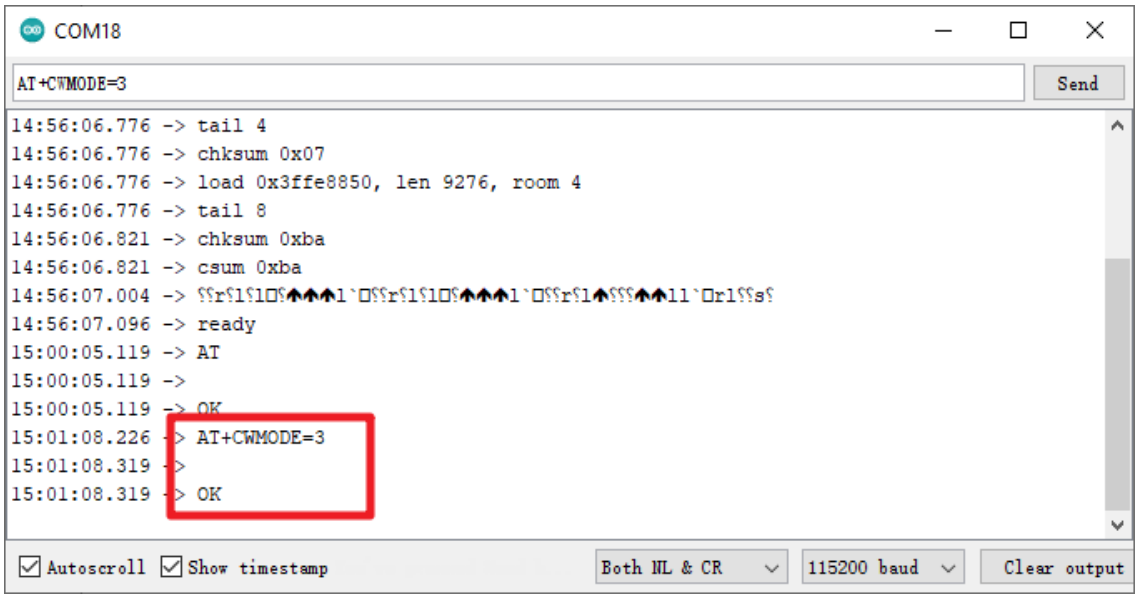

**警告:**

- ready が表示されない場合は、ESP8266 モジュールをリセットして（RST を GND に接続）シリアルモニタを再度開いてみてください。
- さらに、結果が OK の場合、ファームウェアを再書き込みする必要があるかもしれません。詳しくは *ESP8266* **モジュールのファームウェアを再書き込みする方法は?** を参照してください。それでも解決しない場合は、シリアルモニタのスクリーンショットを sevice@sunfounder.com に送信してください。可能な限り早く問題を解決するお手伝いをします。

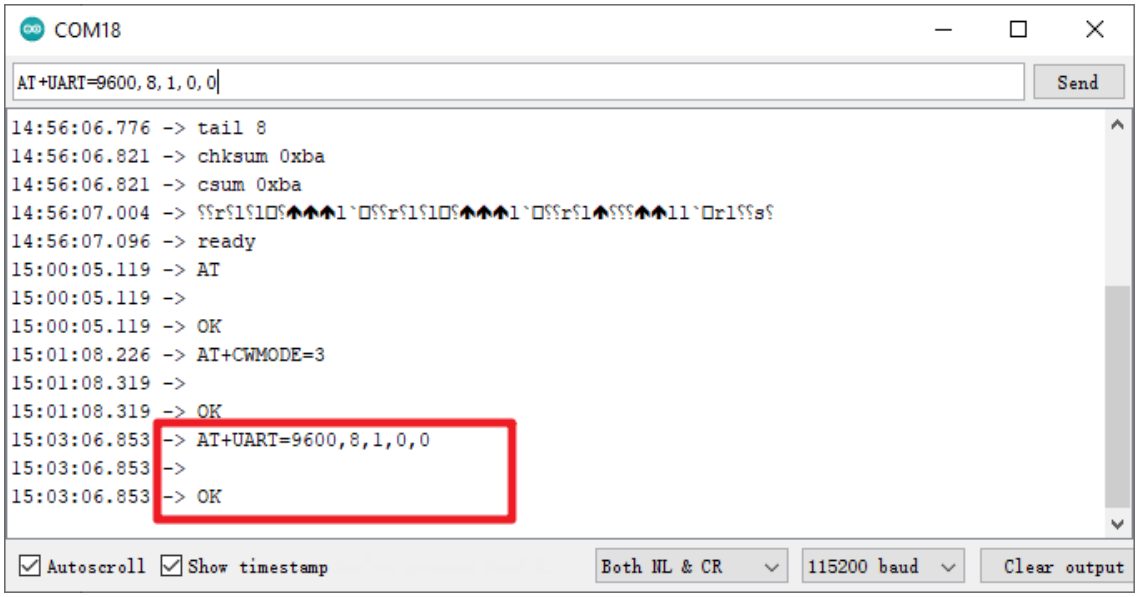
4. **NEWLINE DROPDOWN BOX** をクリックし、ドロップダウンオプションで both NL & CR を選択し、AT を入力します。OK が返ってくれば、ESP8266 が R3 ボードと正常に接続されていることを意味します。




5. AT+CWMODE=3 を入力すると、管理モードが **Station and AP** 共存に変更されます。

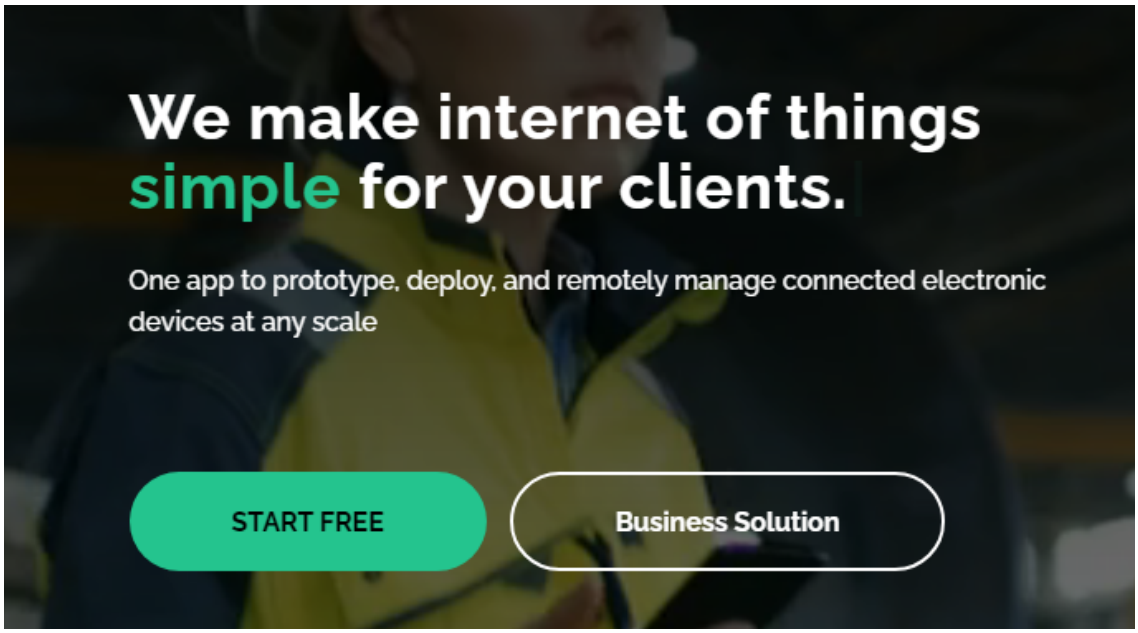



6. 後でソフトウェアシリアルを使用するために、ESP8266 のボーレートを 9600 に変更するため AT+UART=9600,8,1,0,0 を入力する必要があります。

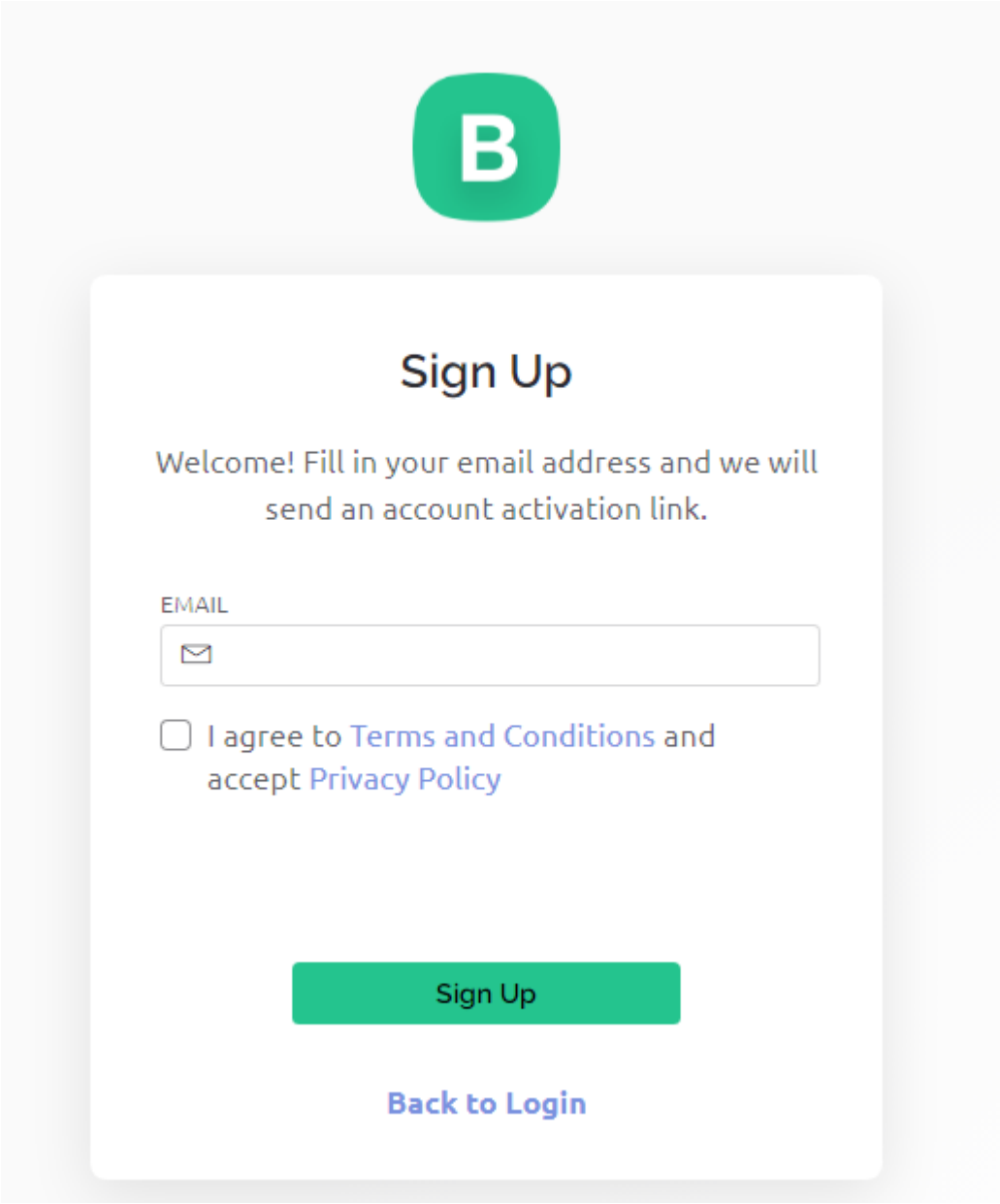



### 6.1.2 1.2 Blynk の設定

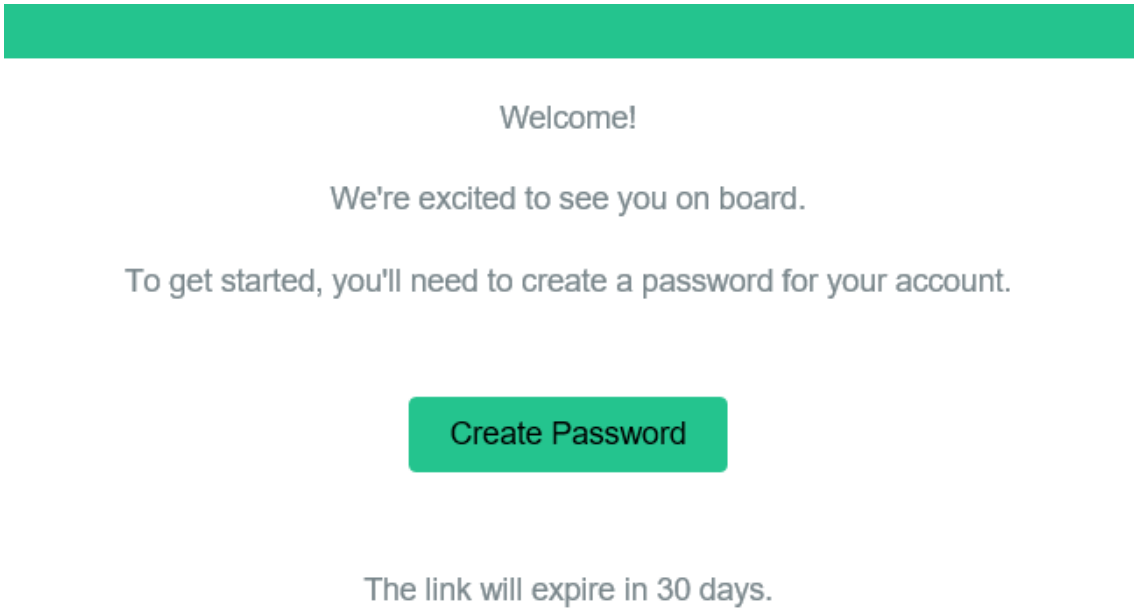
1. BLYNK にアクセスして、 **START FREE** をクリックします。



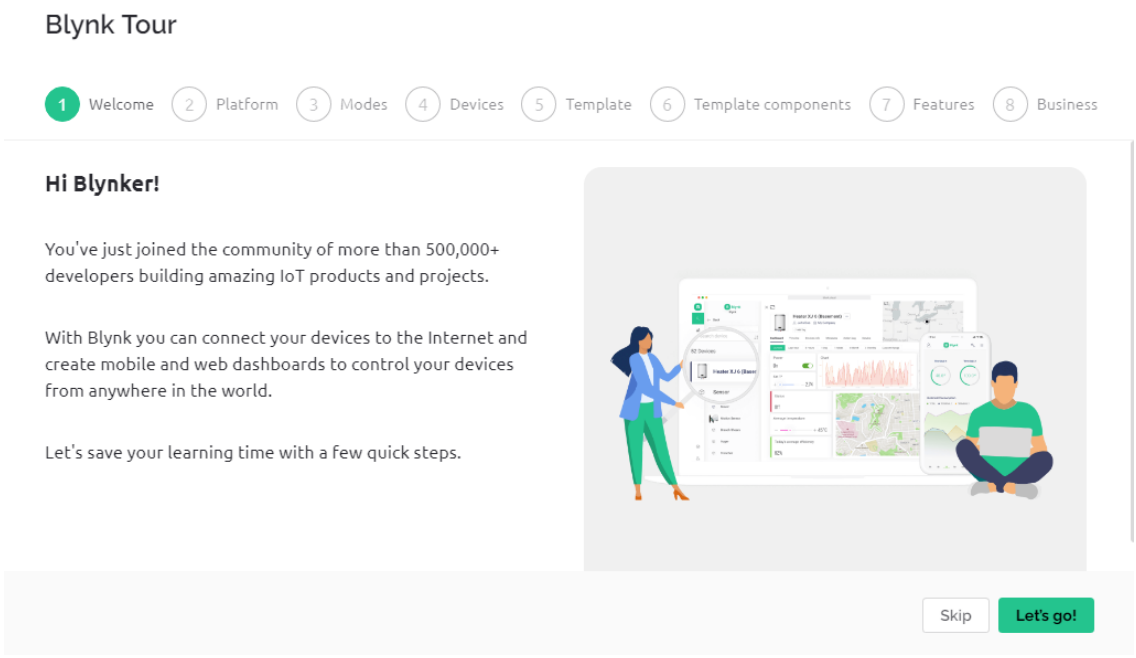

2. あなたのメールアドレスを入力してアカウントを登録します。



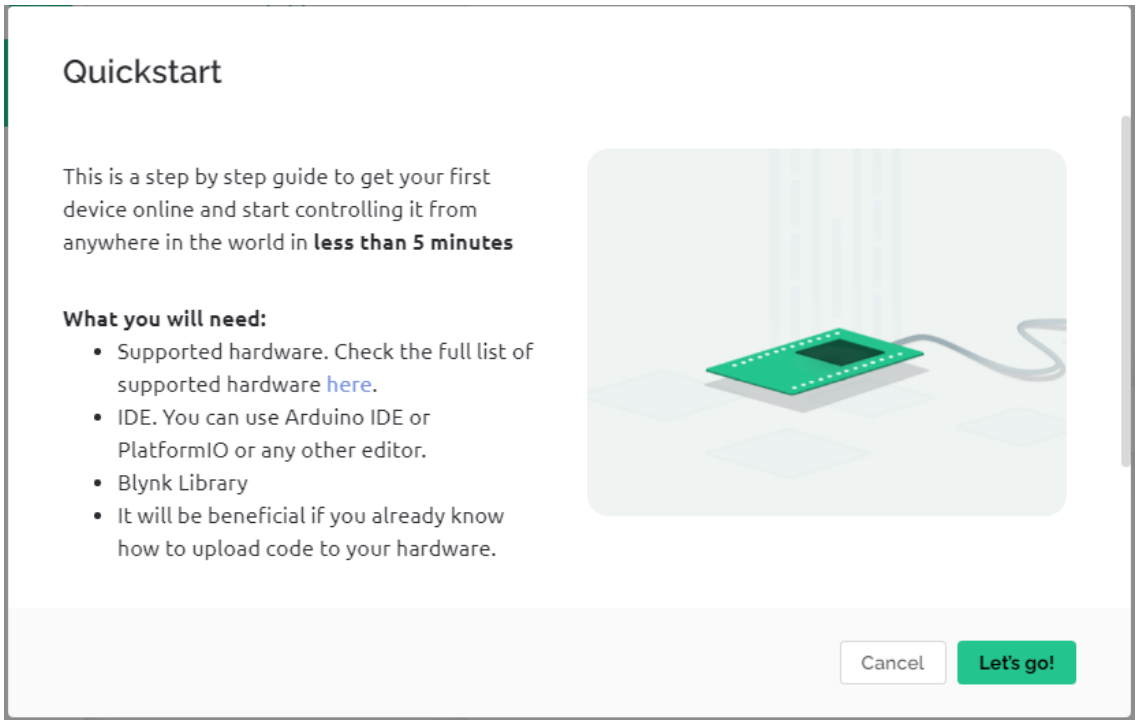

3. 登録したメールアドレスにアクセスして、アカウント登録を完了します。



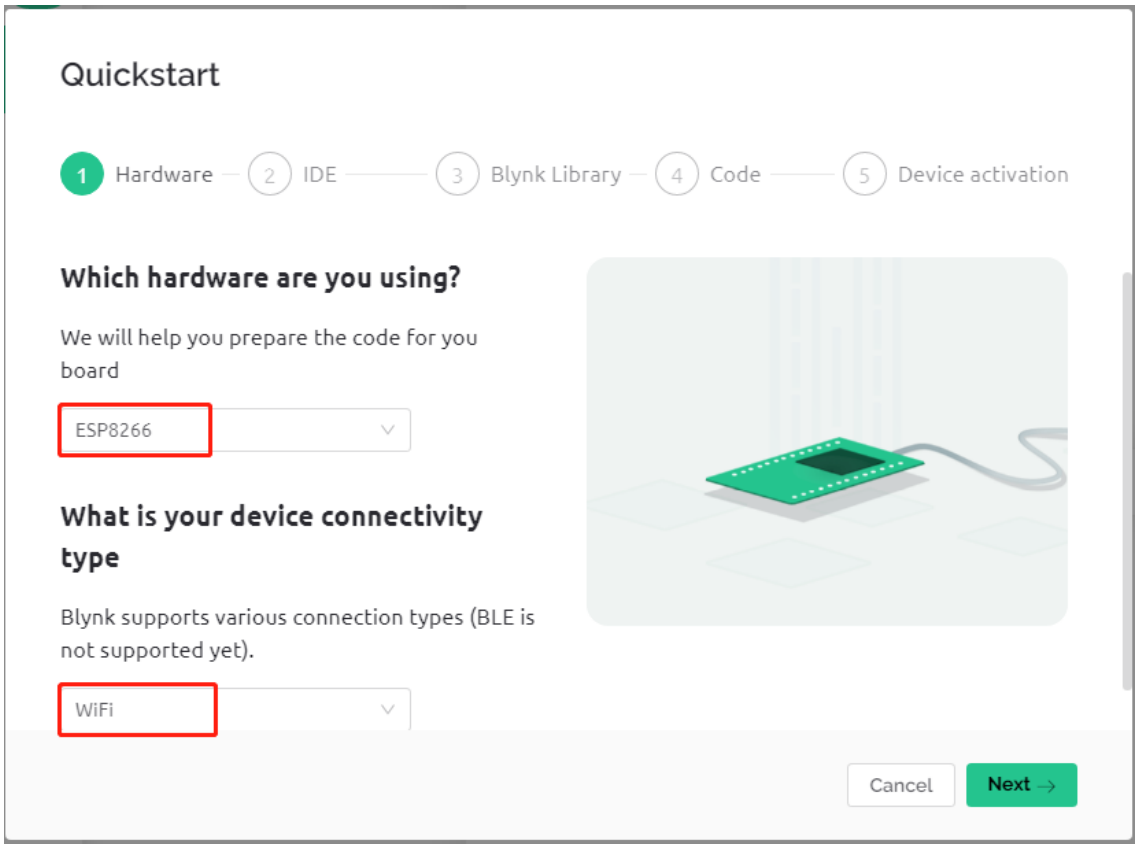

4. その後、 **Blynk Tour** が表示されるので、Blynk に関する基本情報を読むことができます。



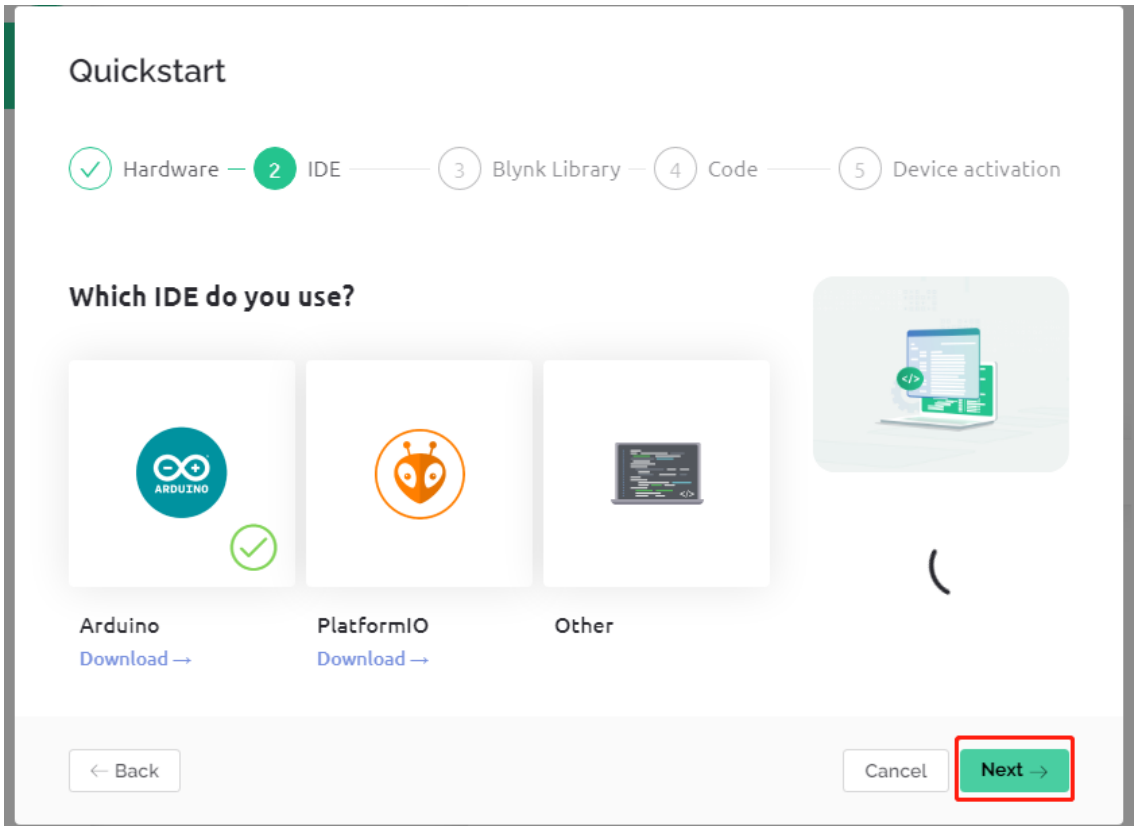

5. 次に、この **Quick Start** でテンプレートとデバイスを作成する必要があります。 **Let's go** をクリックします。



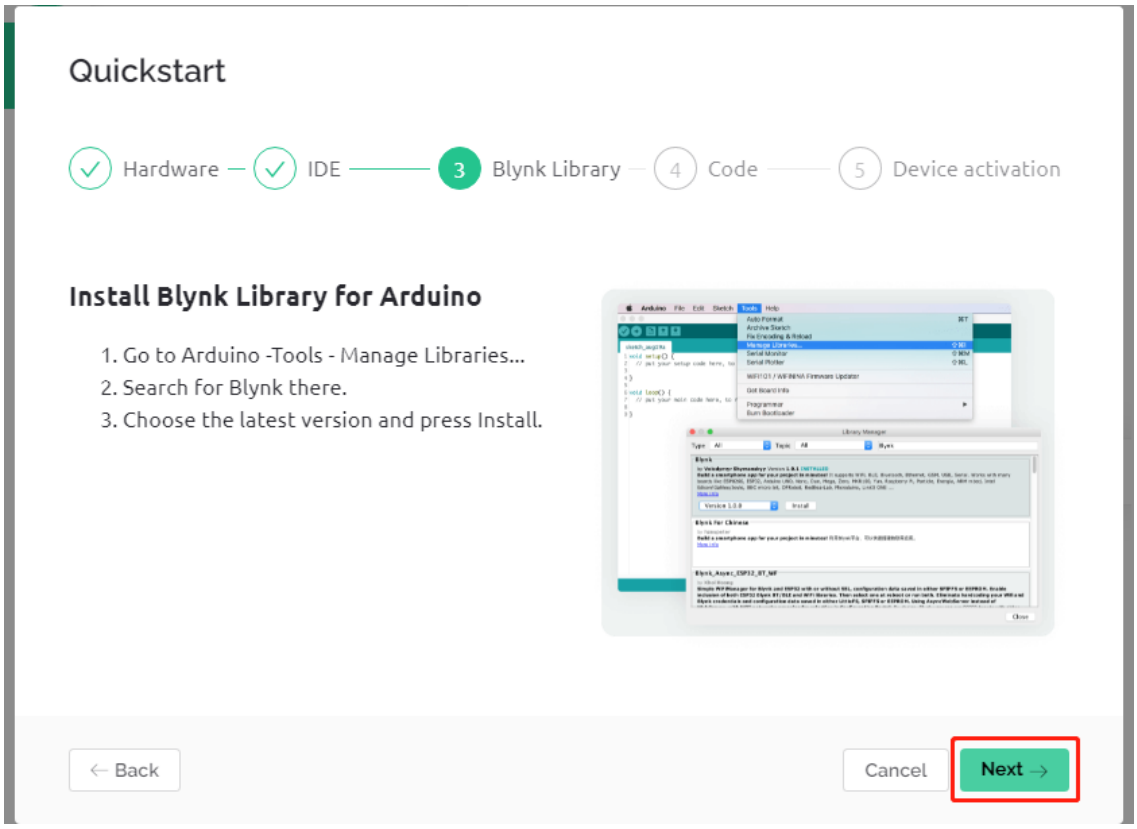

6. ハードウェアと接続タイプを選択します。



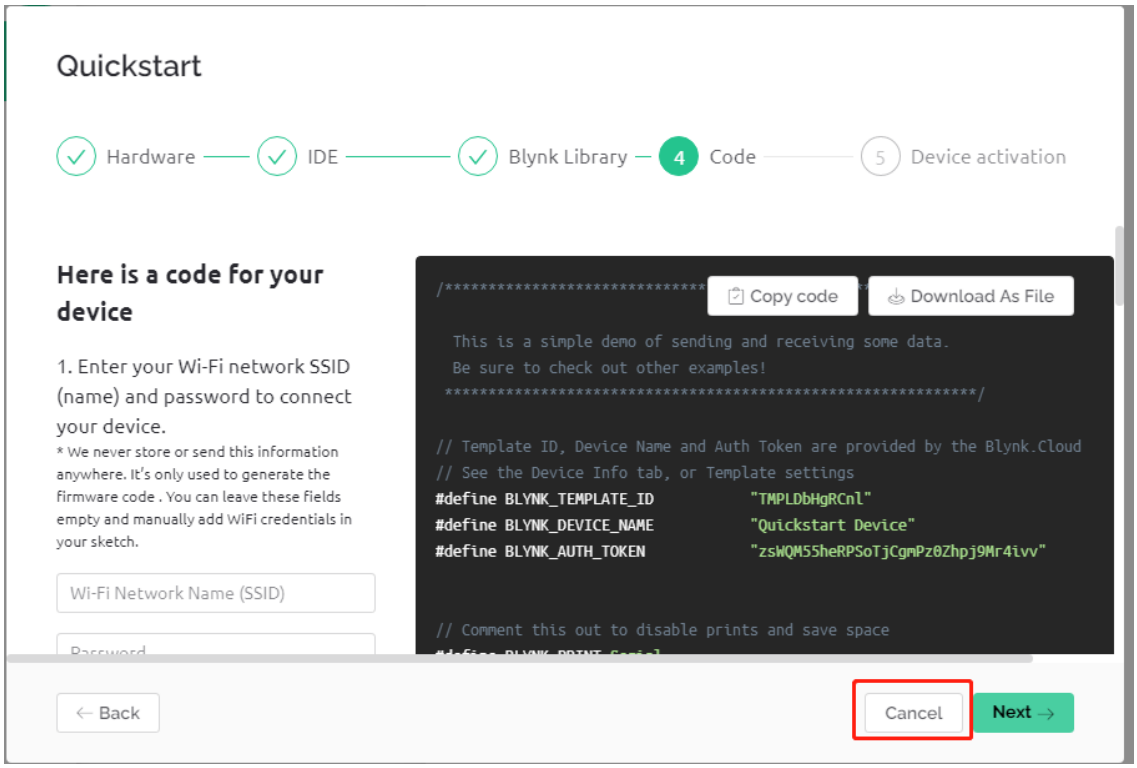

7. どの IDE を準備する必要があるかを知らされます。 **Arduino IDE** を推奨します。



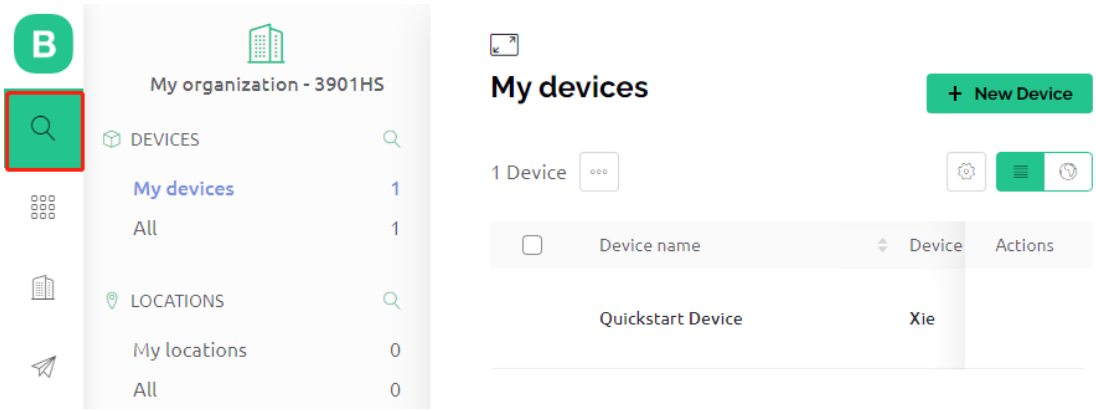

8. 必要なライブラリが表示されますが、ここで推奨されているライブラリには問題があります。手動で他のライブラリを追加する必要があります（後述）。ここでは **Next** をクリックして、新しいテンプレートとデバイスが作成されます。



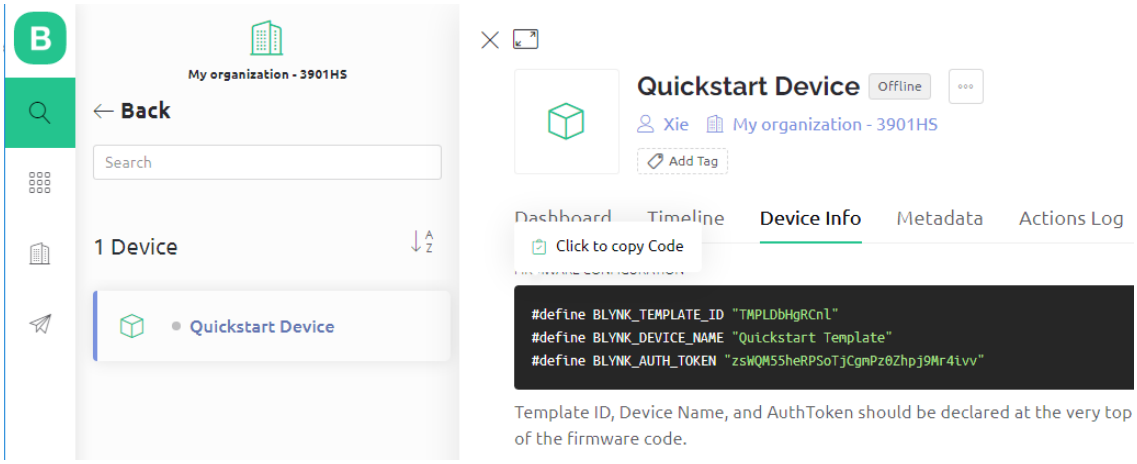

9. 次のステップは、関連するコードをアップロードしてボードを Blynk に接続することですが、先ほど提供されたライブラリに問題があるため、再度他のライブラリを追加する必要があります。したがって、 **Quick Start** を停止するために、ここで **Cancel** をクリックします。




10. **Search** ボタンをクリックすると、作成した新しいデバイスが表示されます。



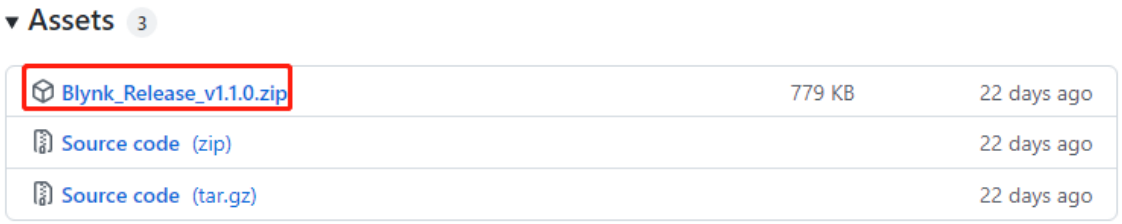

11. この **Quickstart Device** にアクセスすると、 **Device info** ページに TEMPLATE_ID 、 DEVICE_NAME 、 AUTH_TOKEN が表示されます。これらの情報は後でコピーする必要があります。



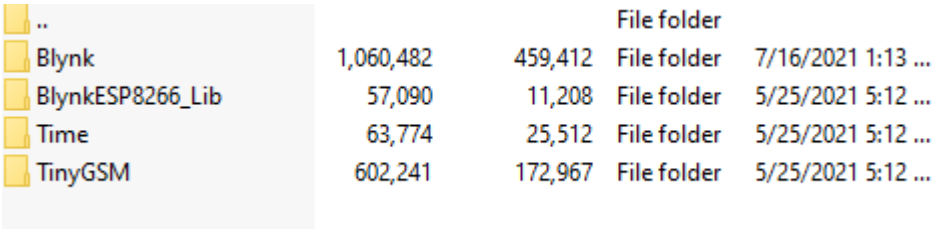

### 6.1.3 1.3 必要なライブラリの追加

Arduino IDE で Blynk を使用するための適切なライブラリを追加する必要があります。

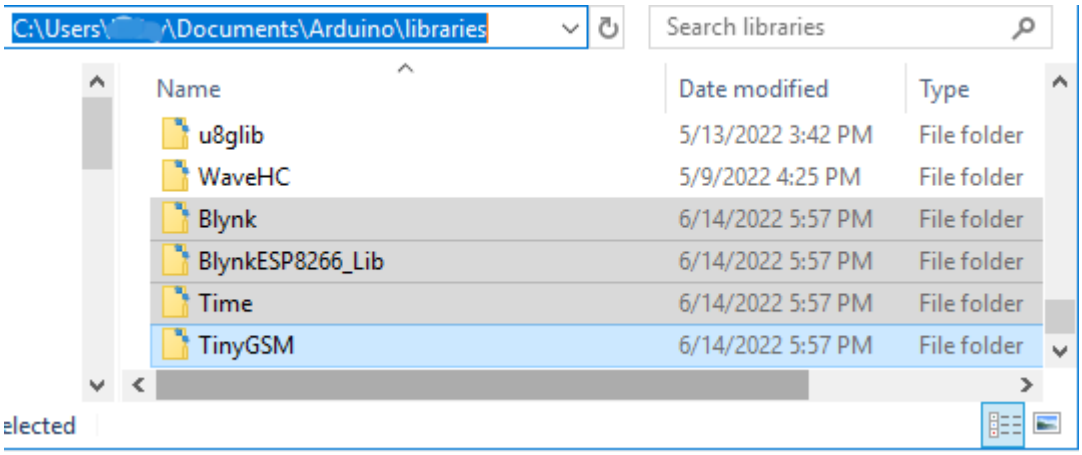
1. こちら をクリックして、ページの最下部にスクロールして最初の `.zip` ファイルをダウンロードします。



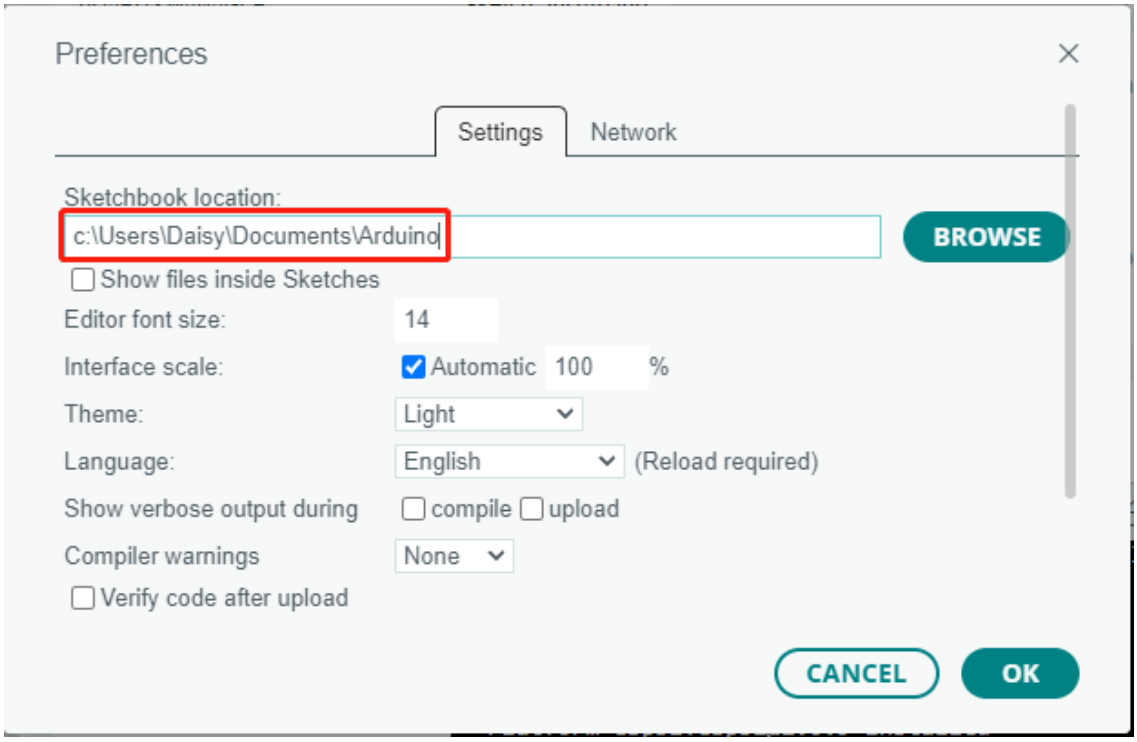

2. このファイルを解凍すると、以下のフォルダが表示されます。




3. これらのフォルダをすべてコピーし、Arduino IDE のデフォルトのライブラリディレクトリ、通常は `C:\Users\xxx\Documents\Arduino\libraries` にペーストします。




4. ライブラリディレクトリが異なる場合、 **File -> Preferences** にアクセスして確認することができます。



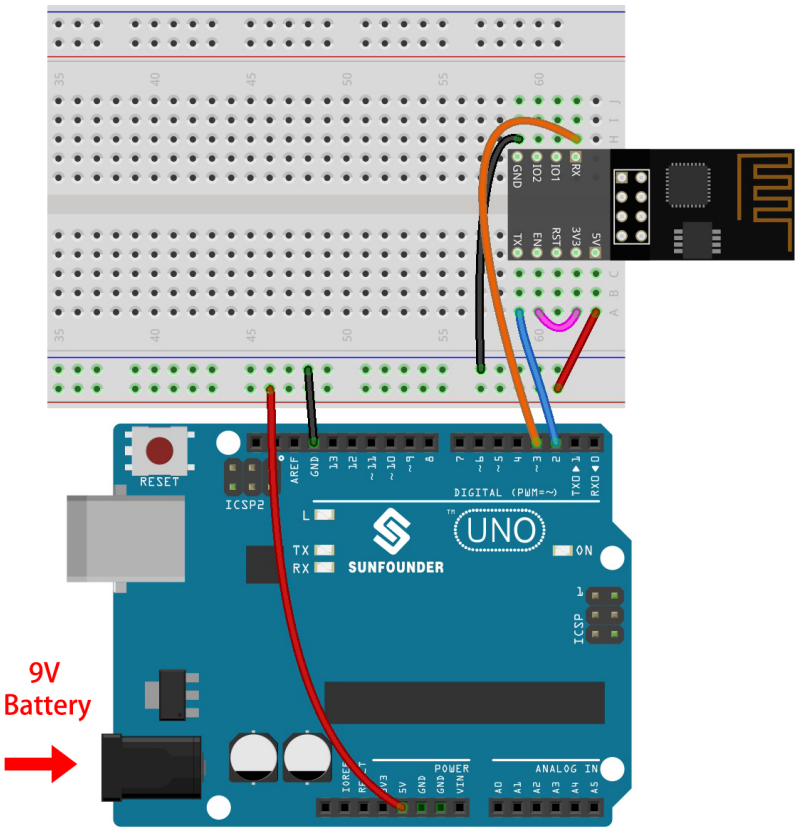

### 6.1.4 1.4 R3 ボードを Blynk に接続

1. ESP8266 モジュールと R3 ボードを再接続します。ここではソフトウェアシリアルが使用されているため、TX と RX は R3 ボードのピン 2 と 3 にそれぞれ接続されます。

---

**注釈:** ESP8266 モジュールは、安定した動作環境を提供するために高い電流を必要とします。9V のバッテリーが接続されていることを確認してください。

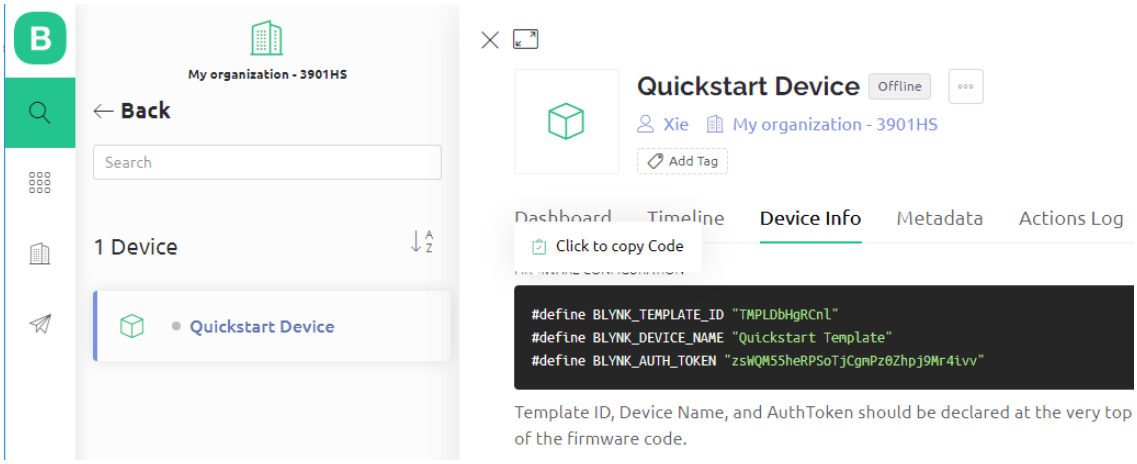
---




2. 3in1-kit\iot_project\1.connect のパスの下の 1.connect.ino ファイルを開きます。または、このコードを **Arduino IDE** にコピーします。

3. **Device info** ページからコピーできる以下の 3 行のコードを置き換えます。これらの 3 行のコードにより、R3 ボードがあなたの blynk アカウントを見つけることができます。

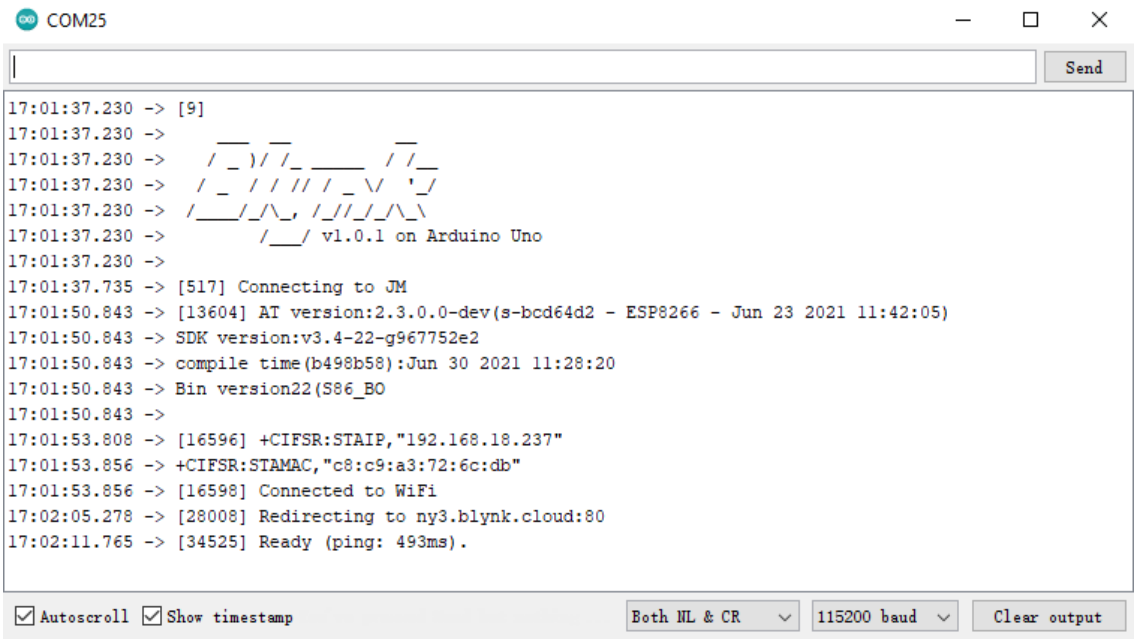
```
#define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
#define BLYNK_DEVICE_NAME "Device"
#define BLYNK_AUTH_TOKEN "YourAuthToken"
```




4. 使用している WiFi の ssid と password を入力します。

```
char ssid[] = "ssid";
char pass[] = "password";
```

5. コードを R3 ボードにアップロードします。次に、シリアルモニタを開き、ボーレートを 115200 に設定します。R3 ボードが Blynk と正常に通信すると、シリアルモニタに ready 文字が表示されます。



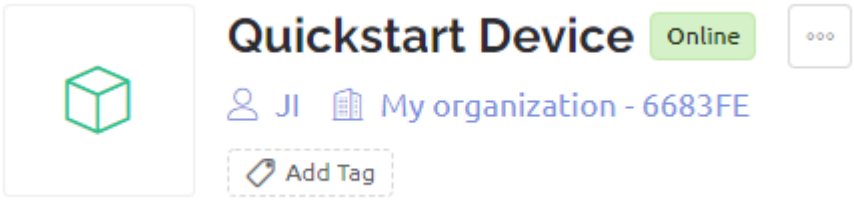

**注釈:** 接続するときに ESP is not responding というメッセージが表示された場合、以下の手順に従ってください。

- 9V のバッテリーが接続されていることを確認してください。
- RST ピンを 1 秒間 GND に接続して ESP8266 モジュールをリセットし、それを抜きます。

- R3 ボードのリセットボタンを押します。

これらの操作を 3〜5 回繰り返す必要がある場合があります。お手数をおかけしますが、お待ちください。

6. Blynk のステータスが **offline** から **online** に変わります。




## 6.2 2. Blynk からデータを取得

この章では、Blynk を使って回路を制御する方法を学びます。インターネットを介して LED を点灯させましょう！

**必要な部品**

このプロジェクトには、以下の部品が必要です。

一式を購入することは非常に便利です。以下がリンクです：

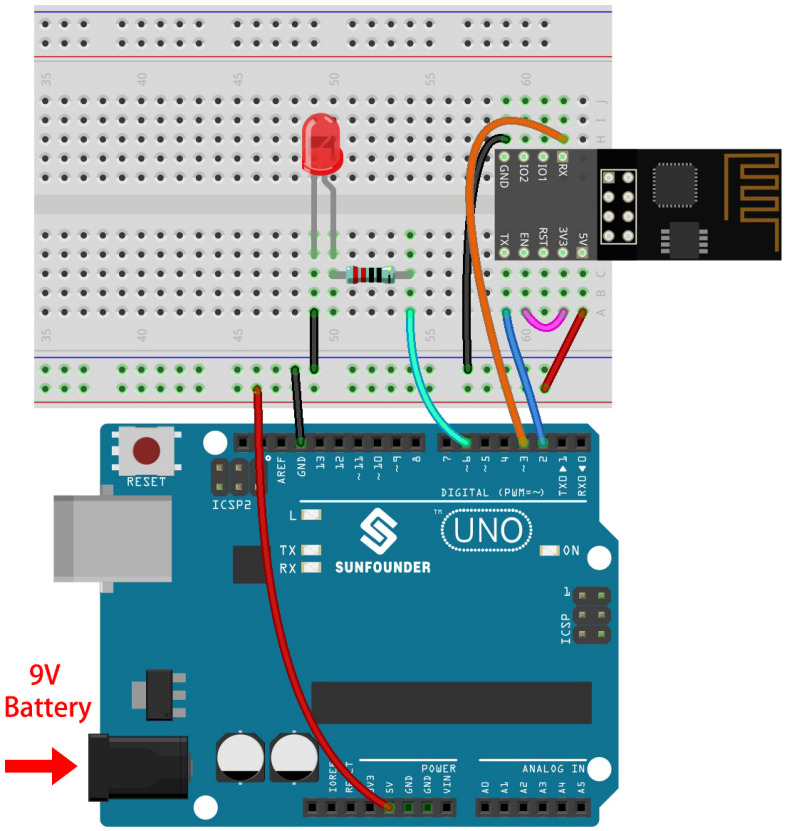
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| *ESP8266* モジュール | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |

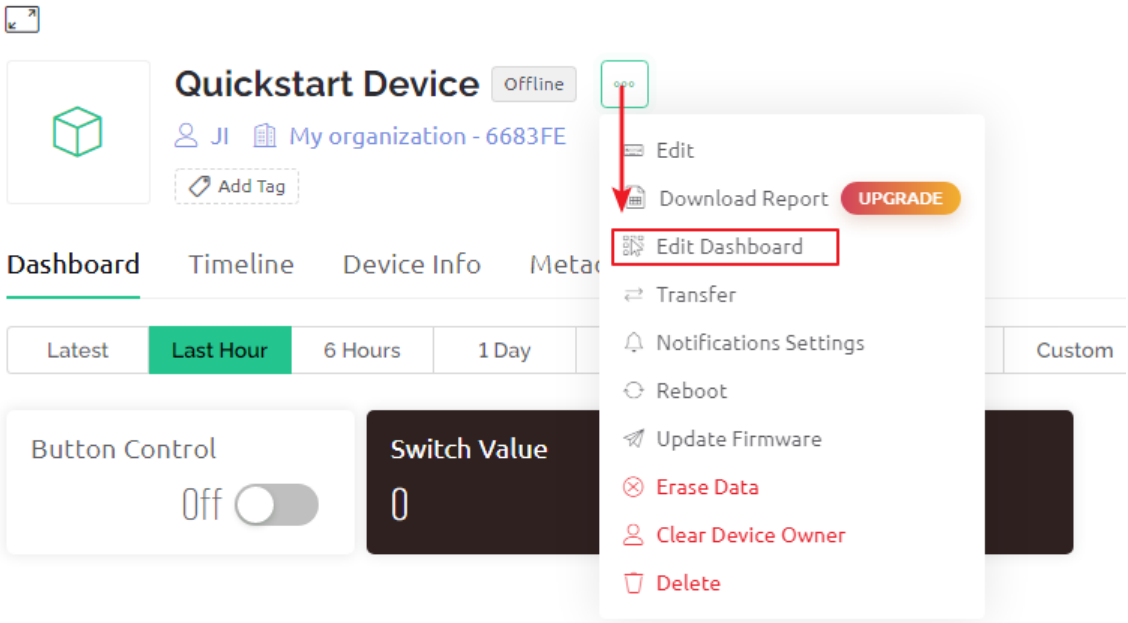
### 1. 回路を組む

**注釈:** ESP8266 モジュールは、安定した動作環境を提供するために高い電流が必要です。9V のバッテリーが接続されていることを確認してください。

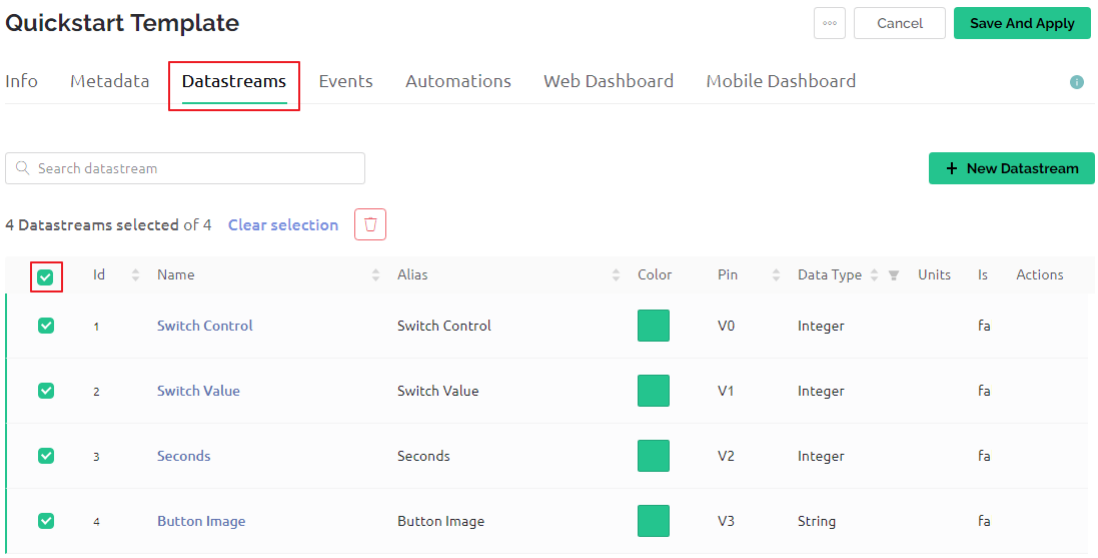


### 2. ダッシュボードを編集

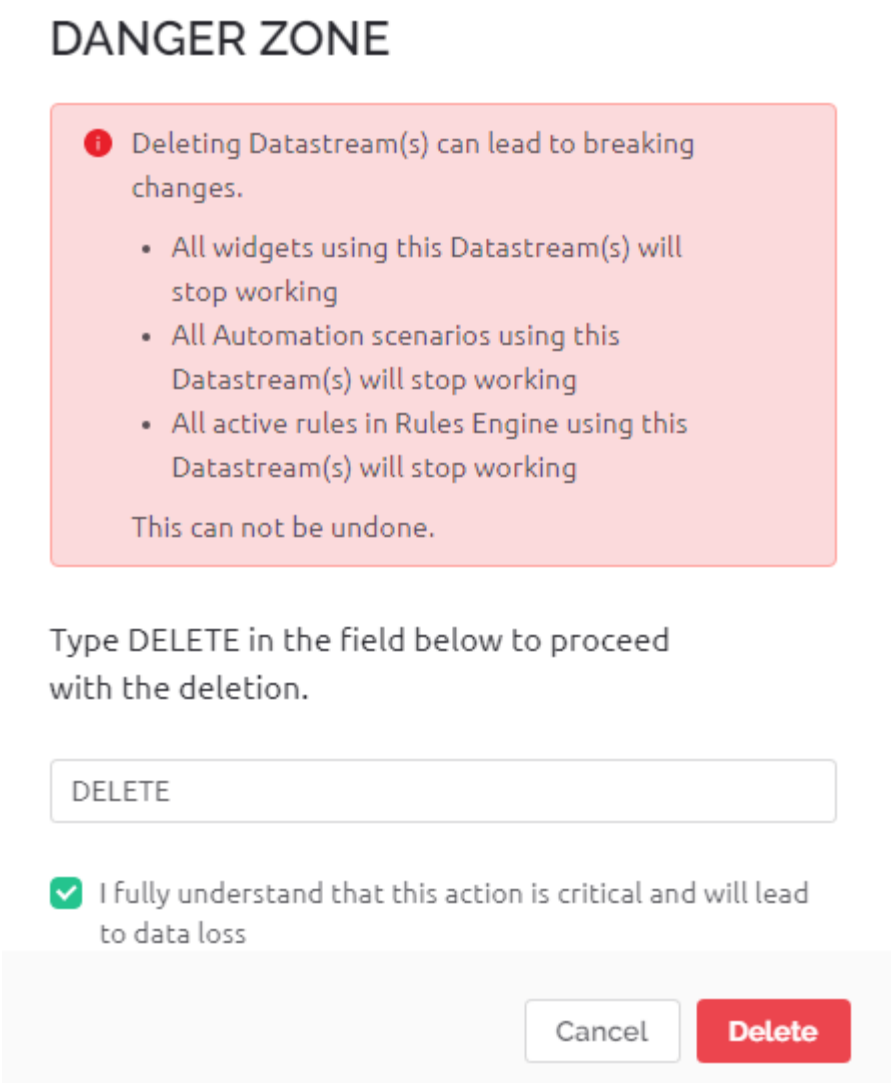
1. 以前に作成した **Quickstart Device** にアクセスし、右上のメニューアイコンをクリックして **edit dashboard** を選択します。



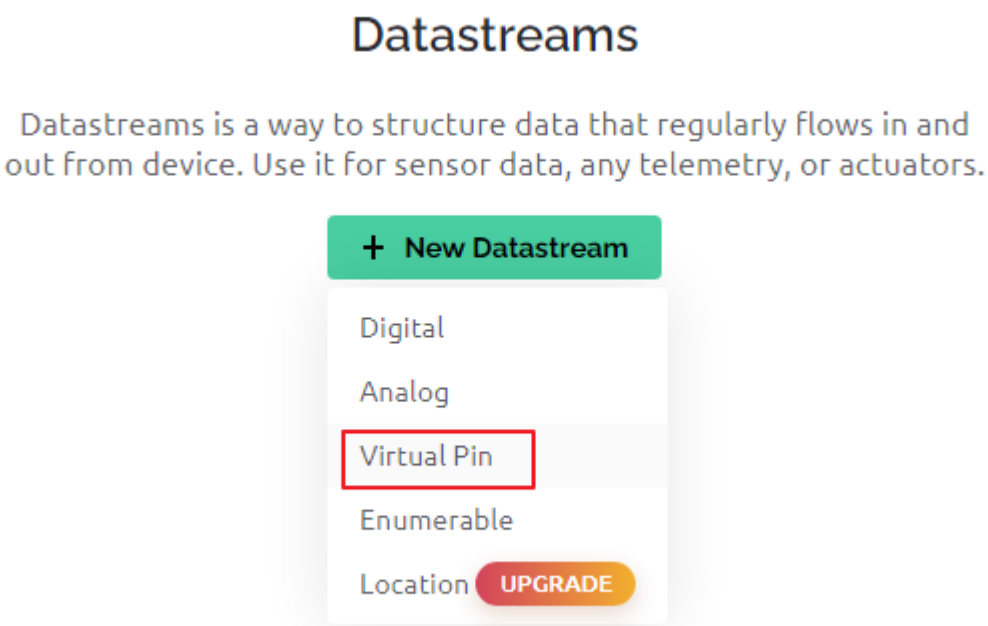

2. Datastreams は、Blynk のウィジェットと R3 ボードのコードが相互に認識するための機能です。完全な設定プロセスを体験するために、Datastreams ページからすべての Datastreams を削除してください。



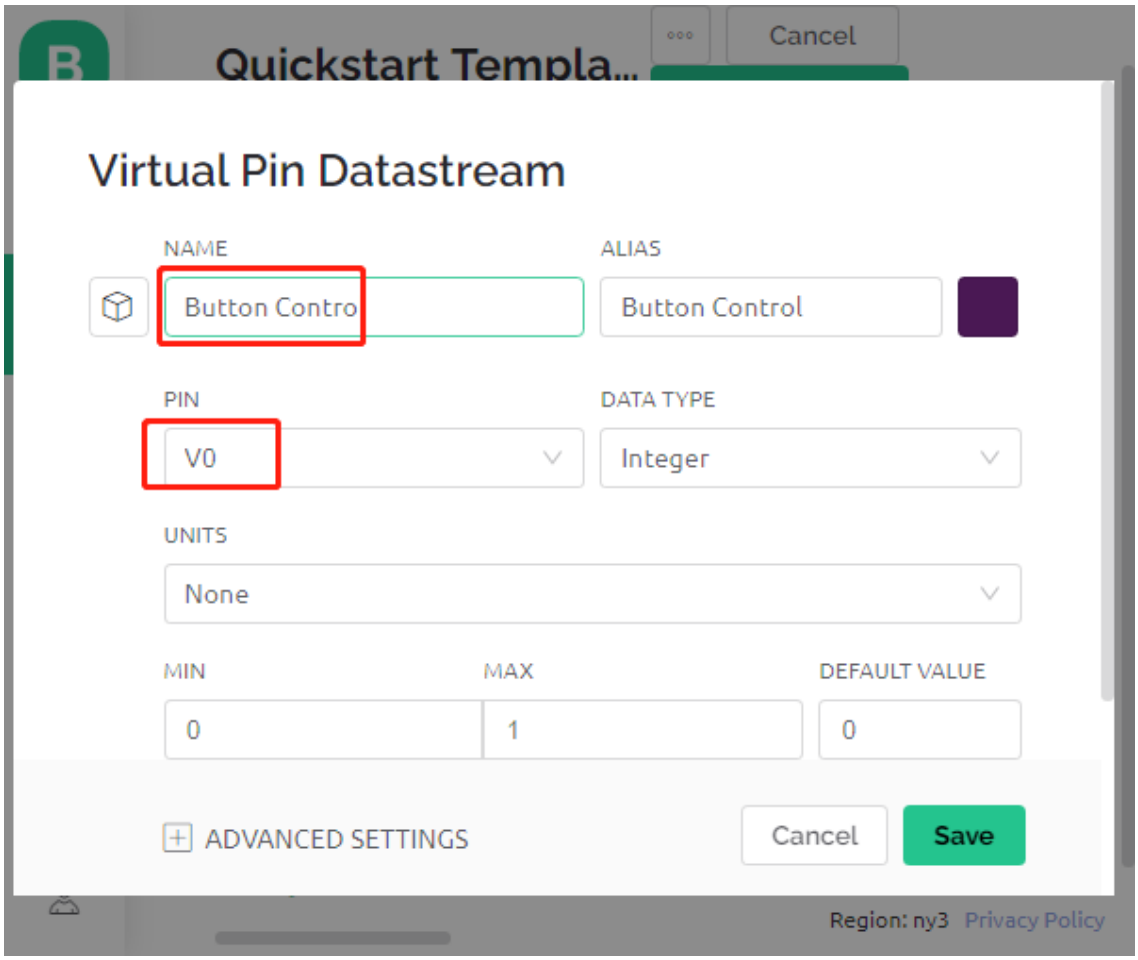

3. Datastreams を削除する前に、警告を注意深く読んで正しいことを確認してください。



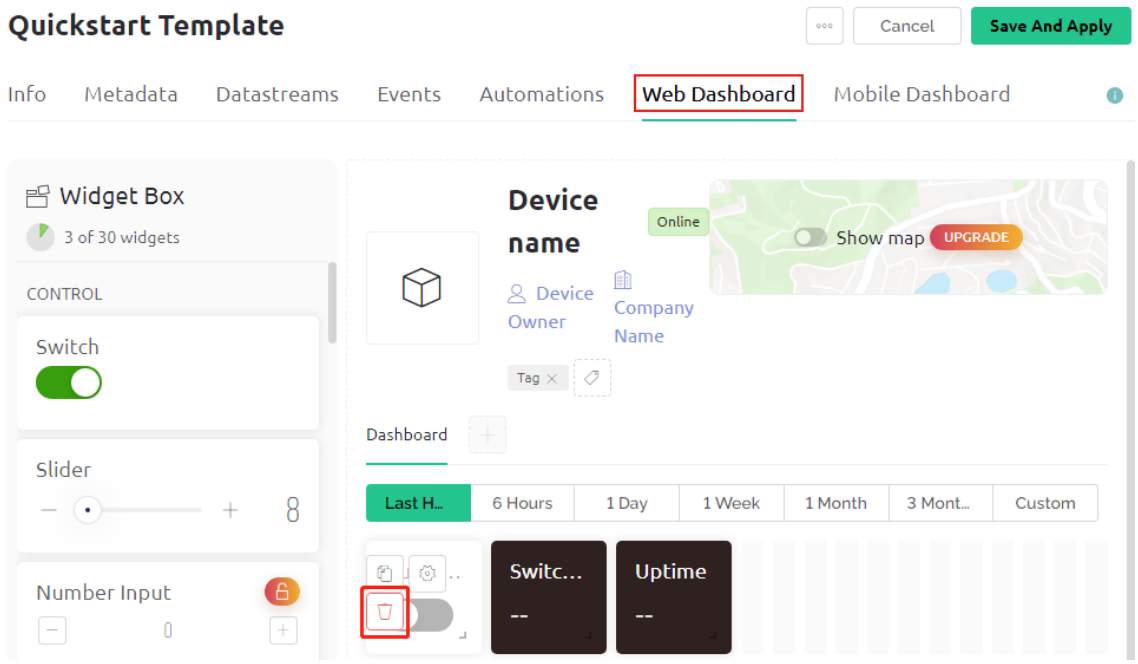

4. Blynk のスイッチを使用して LED を制御するための **Virtual Pin** タイプの Datastream を作成します。



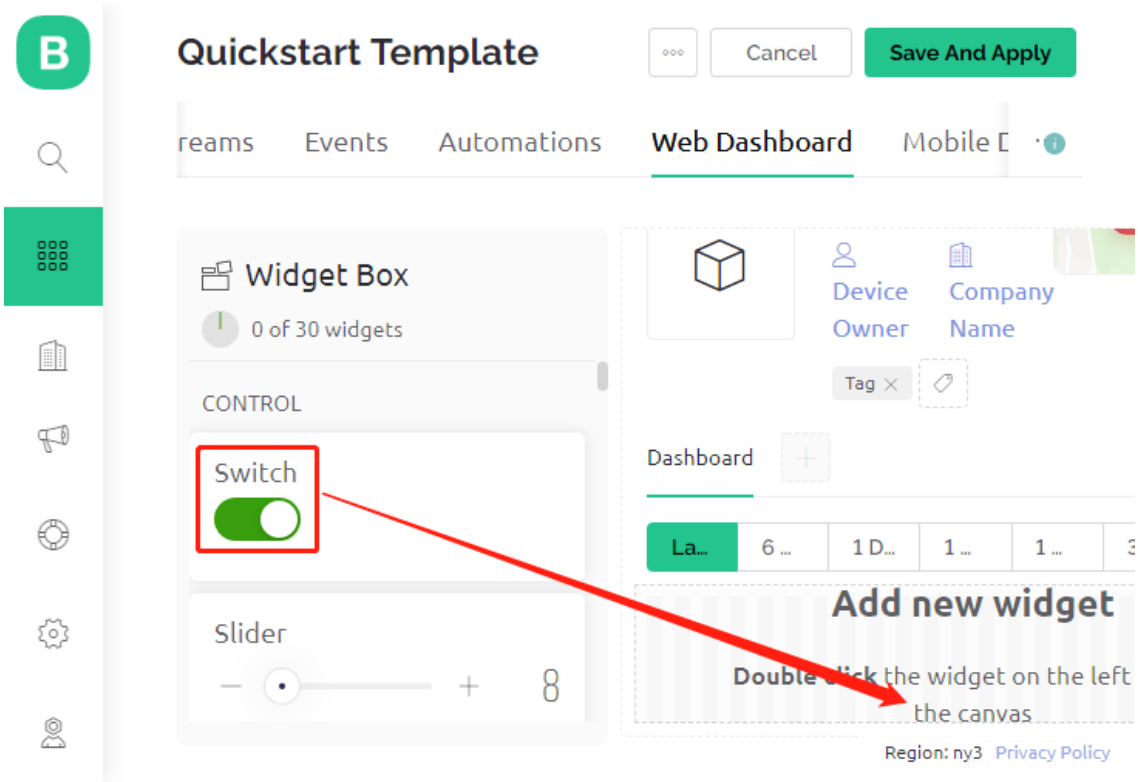

5. **Virtual Pin** を設定します。ボタンと LED は ON と OFF のみが必要なので、DATA TYPE を `Integer` に設定し、MIN と MAX を `0` と `1` に設定します。



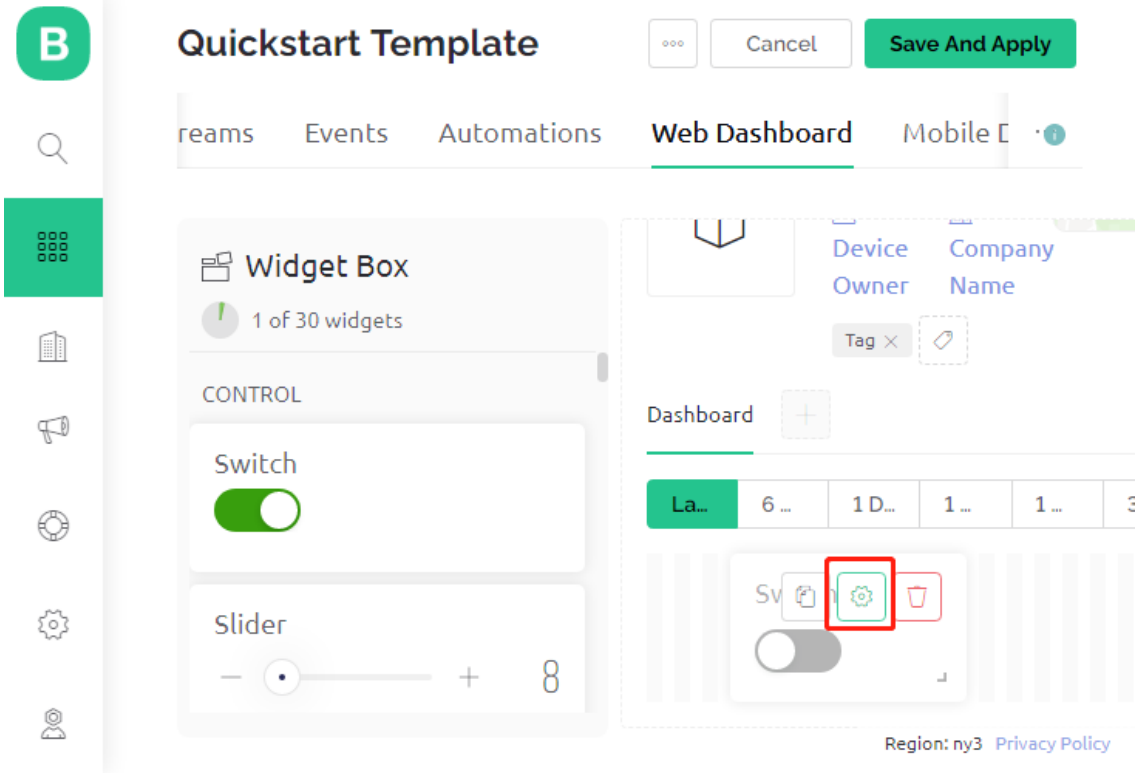

6. **Web Dashboard** ページに移動し、既存のウィジェットを削除します。



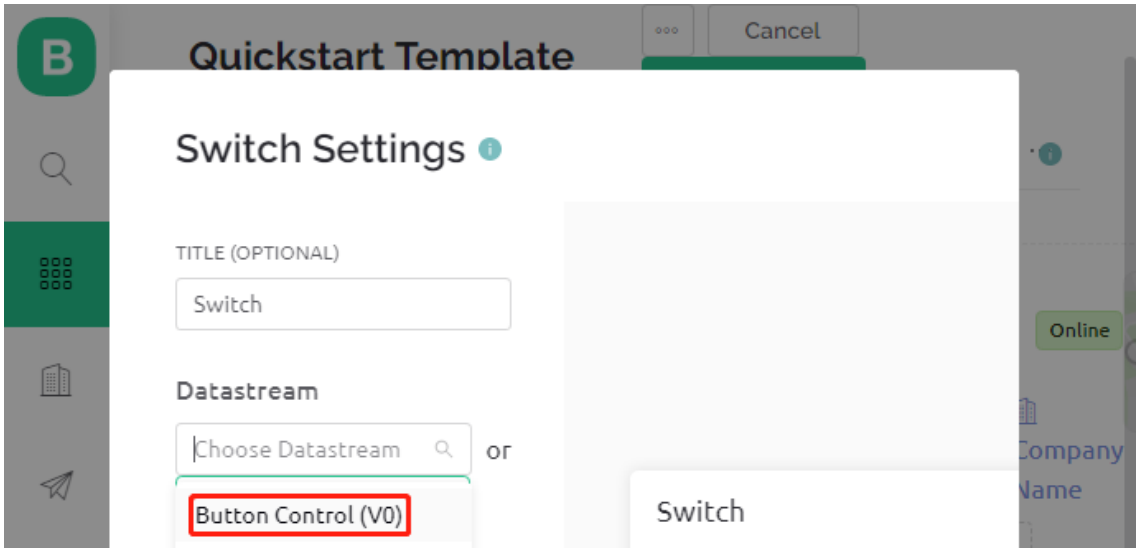
7. 左の **Widget Box** から **switch** ウィジェットをドラッグアンドドロップします。



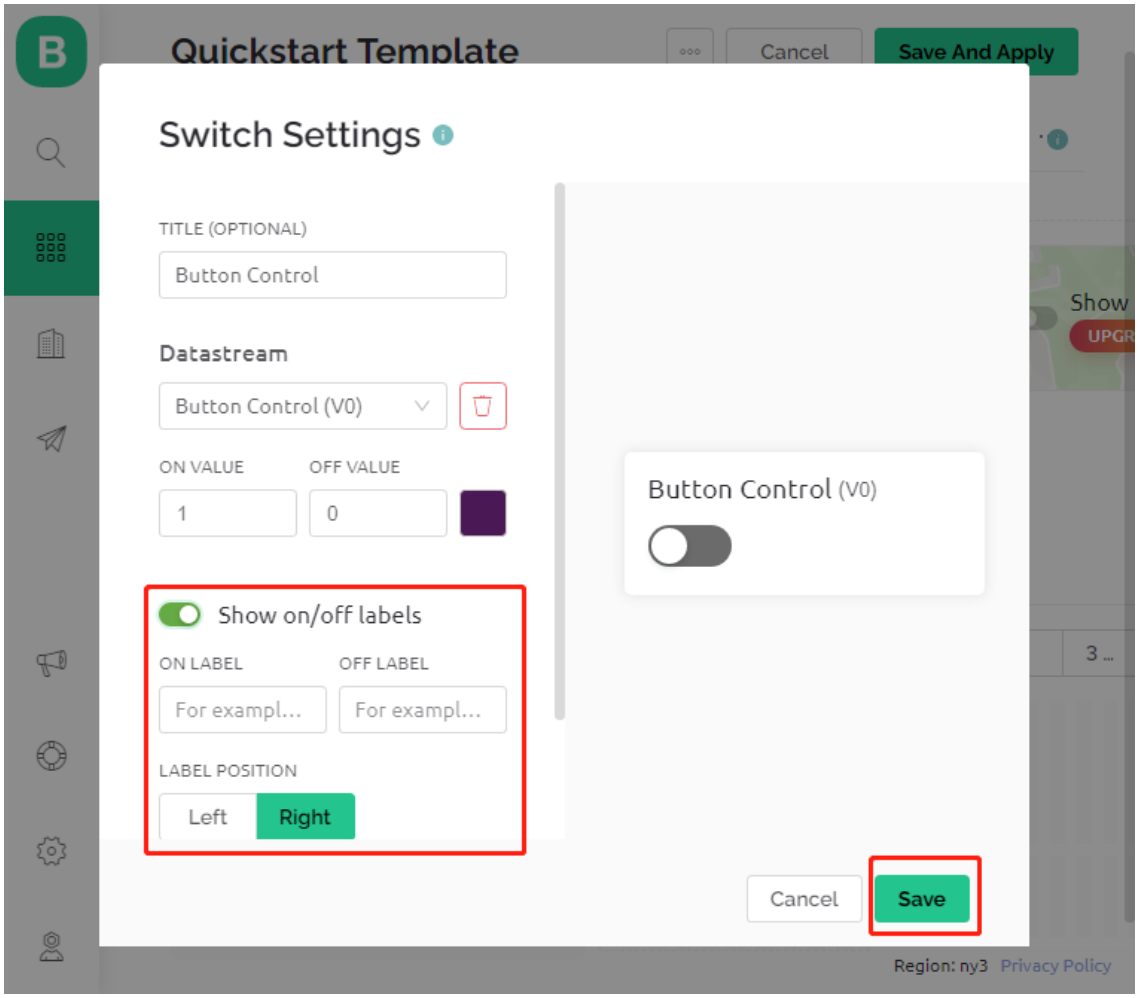

8. これで設定します。



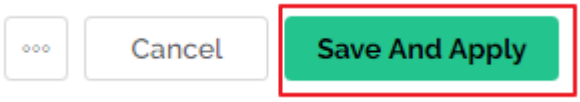
9. 以前に設定した **Datastream** を選択します。




10. Datastream を選択すると、いくつかのカスタム設定が表示されます。それらを設定した後、保存をクリックします。



11. 最後に、 **Save And Apply** をクリックします。

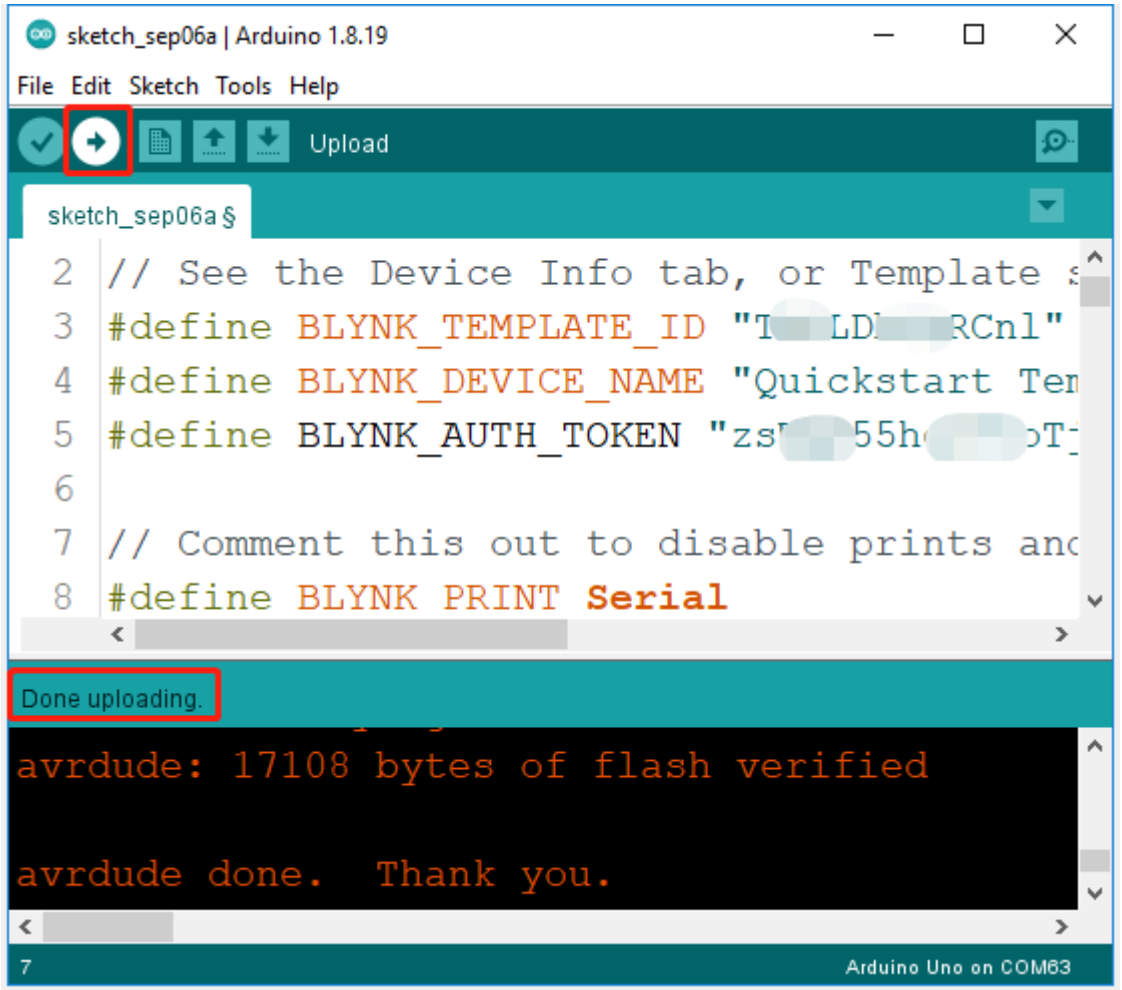



### 3. コードの実行

1. `3in1-kit\iot_project\2.get_data_from_blynk` のパスの下の `2.get_data_from_blynk.ino` ファイルを開くか、このコードを **Arduino IDE** にコピーします。

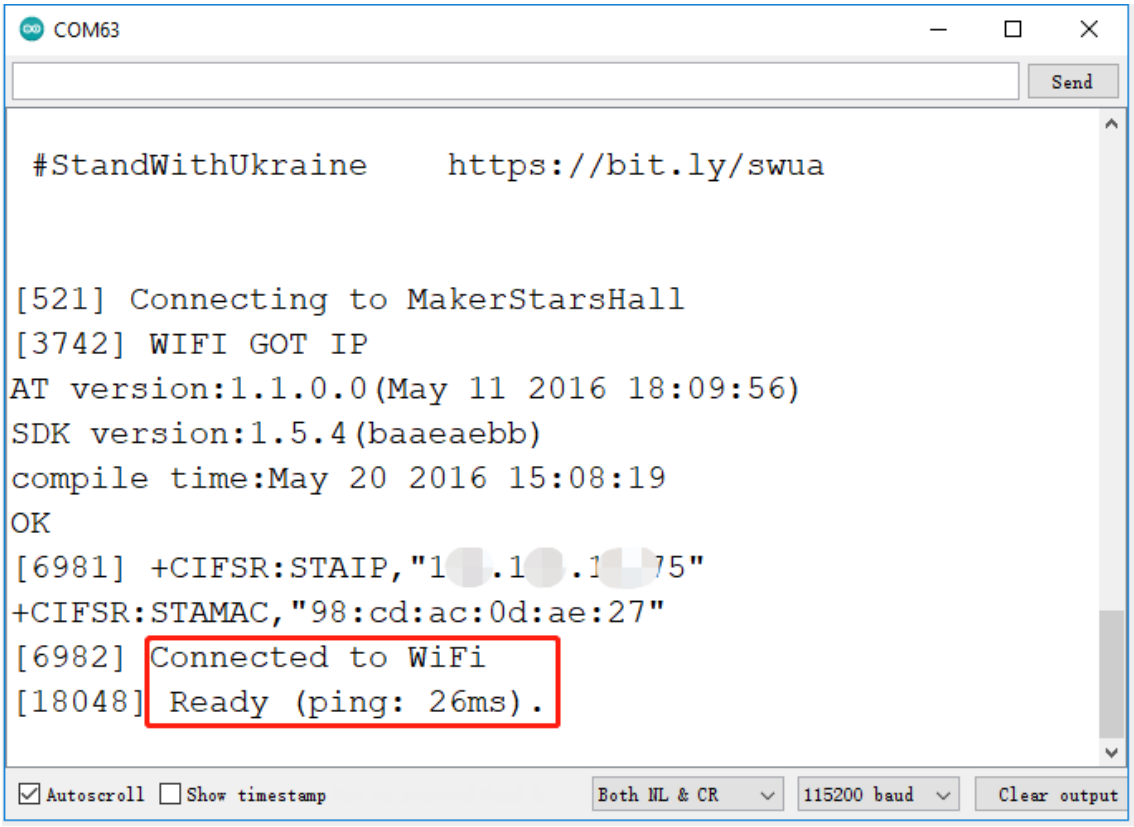
2. `Template ID`、`Device Name`、`Auth Token` をあなた自身のものに置き換えます。使用中の WiFi の `ssid` と `password` も入力する必要があります。詳しいチュートリアルは、 *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 正しいボードとポートを選択したら、 **Upoad** ボタンをクリックします。




4. シリアルモニター（ボーレートを 115200 に設定）を開き、成功した接続のようなプロンプトが表示される

のを待ちます。




---

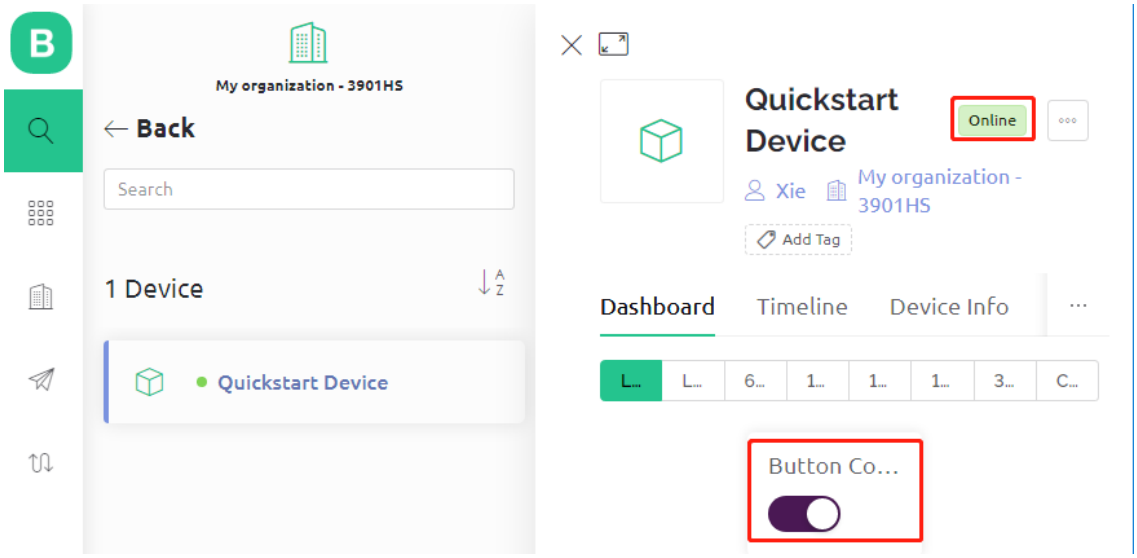
**注釈:** 接続時に `ESP is not responding` のメッセージが表示される場合、次の手順に従ってください。

- 9V のバッテリーが接続されていることを確認します。
- RST ピンを 1 秒間 GND に接続して ESP8266 モジュールをリセットし、その後、それを取り外します。
- R3 ボードのリセットボタンを押します。

ときどき、上記の操作を 3〜5 回繰り返す必要があるかもしれませんので、お待ちください。

---

5. Blynk に戻ると、ステータスがオンラインに変わっているのがわかります。これで、blynk のスイッチウィジェットを使用して R3 ボードに接続されている LED を制御することができます。




6. モバイルデバイスで Blynk を使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。

**どのように動作するのか？**

このプロジェクトのコードと前章の *1.4 R3* **ボードを** *Blynk* **に接続** のコードとの違いは、以下の行になります。

```
const int ledPin=6;

BLYNK_WRITE(V0)
{
    int pinValue = param.asInt(); // V0 からの入力値を変数に割り当てる
    // 以下も使用可能:
    // String i = param.asStr();
    // double d = param.asDouble();
    digitalWrite(ledPin,pinValue);
}

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

ledPin の `pinMode` と `digitalWrite` については、すでにお馴染みだと思いますので、再度説明しません。注目すべきは、`BLYNK_WRITE(V0)` 関数です。

この関数が行うのは、Blynk の `V0` の値が変更されると、Blynk.Cloud があなたのデバイスに「**Virtual Pin** V0 に書き込みをしています」と通知し、この情報を受け取ったデバイスが何かの動作をすることができることです。

前のステップで V0 Datastream を作成し、スイッチウィジェットに適用しました。これは、スイッチウィジェットを操作するたびに、 `BLYNK_WRITE(V0)` がトリガされることを意味します。

この関数には 2 つの命令を書き込んでいます。

```
int pinValue = param.asInt();
```

V0 の値を取得し、変数 `pinValue` に割り当てます。

```
digitalWrite(ledPin,pinValue);
```

取得した V0 の値を ledPin に書き込むことで、Blynk のスイッチウィジェットで LED を制御することができます。

## 6.3 3. Blynk へデータをプッシュ

この章では、Blynk にデータを送信する方法を紹介します。

ここでは、ドアと窓の検出デバイスを作成します。リードスイッチの回路はドアや窓の隣に配置され、磁石はドアや窓の端に取り付けられます。ドアや窓が閉じている場合、磁力でリードスイッチがオンになり、R3 ボード上の対応するピンの値が変わります。Blynk.cloud はこの値を受け取るので、家を離れていても家のドアや窓が閉まっているかどうかを確認できます。

今、Blynk の LED ウィジェットを使用して、窓やドアが閉まっているか（すなわち、リードスイッチがオンかオフか）を表示します。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

一式を購入するのが確実に便利です。以下にリンクを示します：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

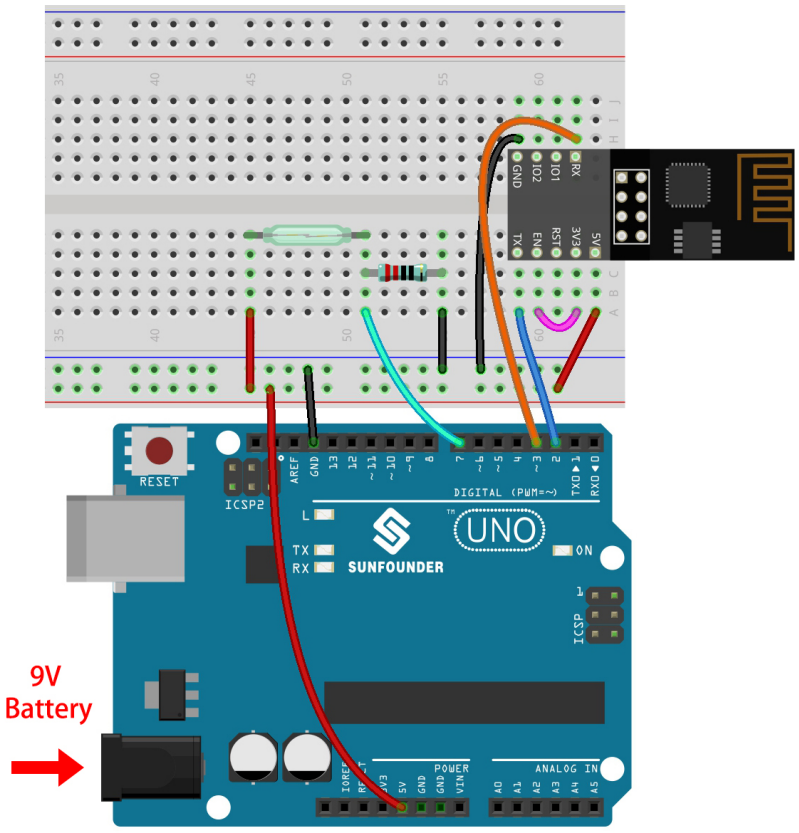
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| *ESP8266* モジュール | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| リードスイッチ | - |

**1. 回路を組む**

---

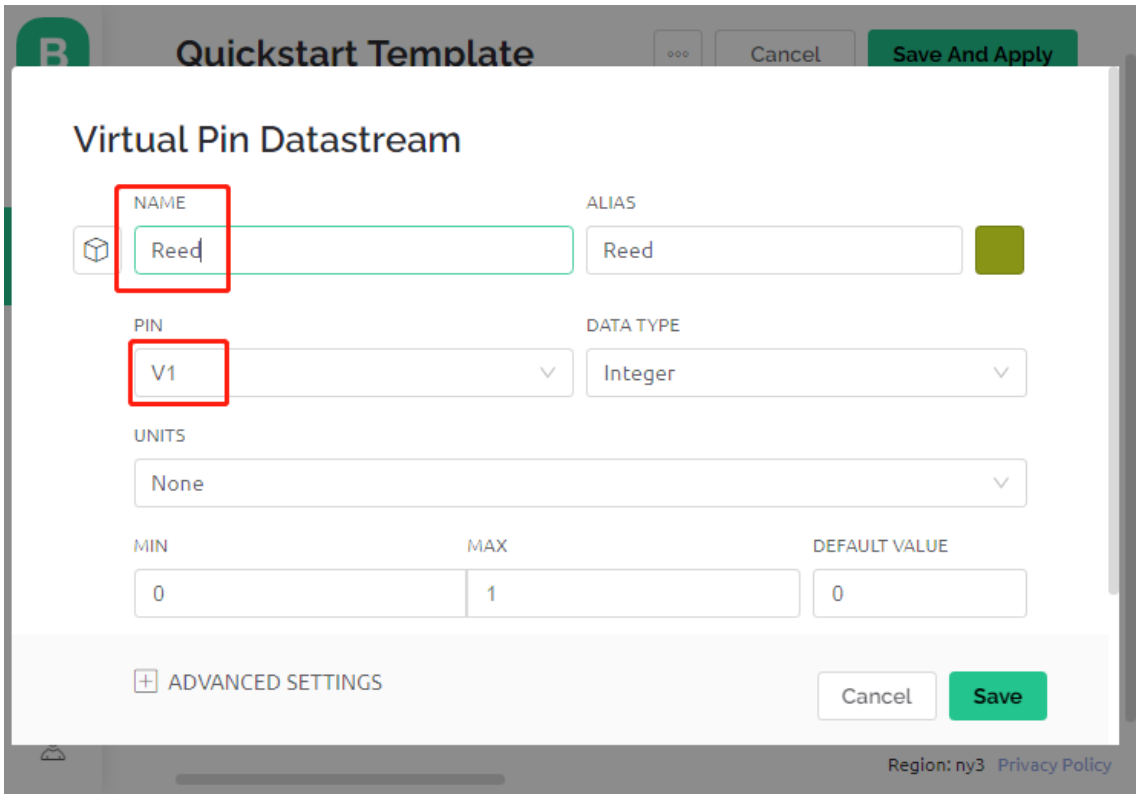
**注釈:** ESP8266 モジュールは安定した動作環境を提供するために高い電流が必要ですので、9V の電池が接続されていることを確認してください。

---

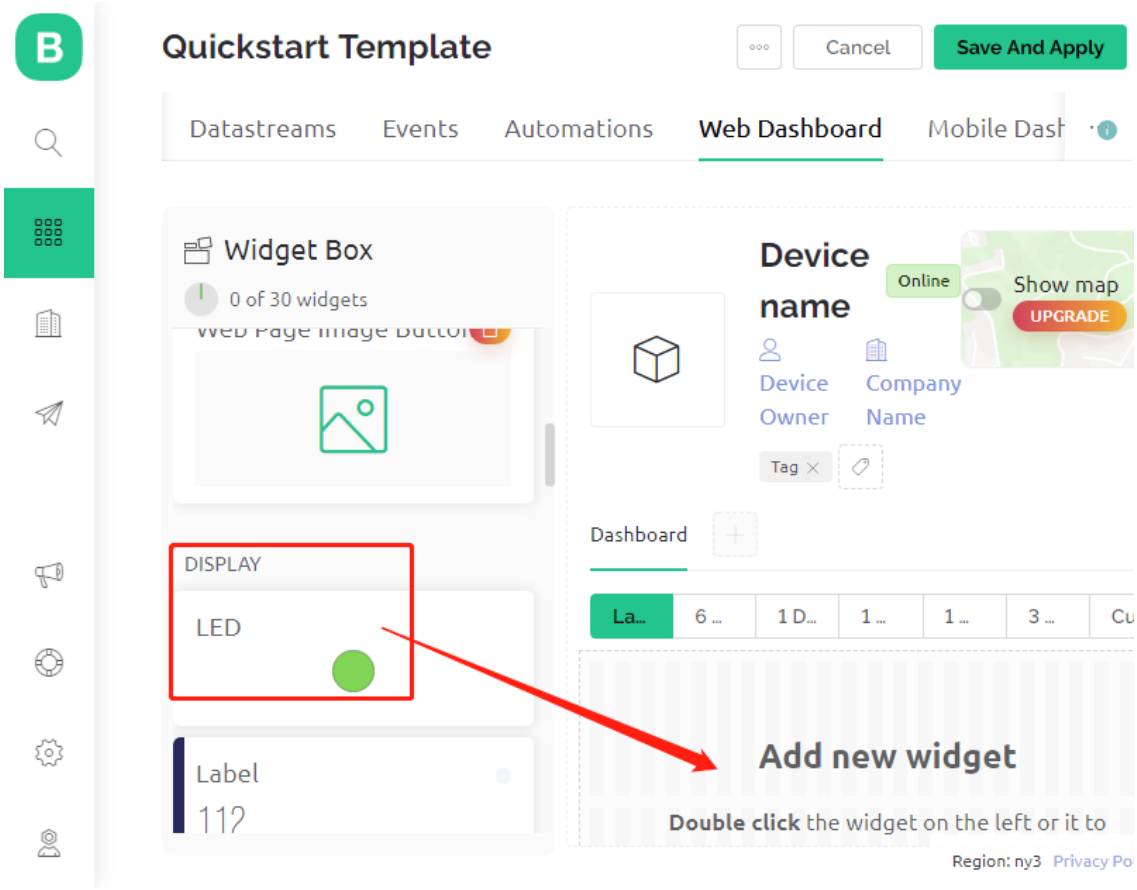


### 2. ダッシュボードを編集

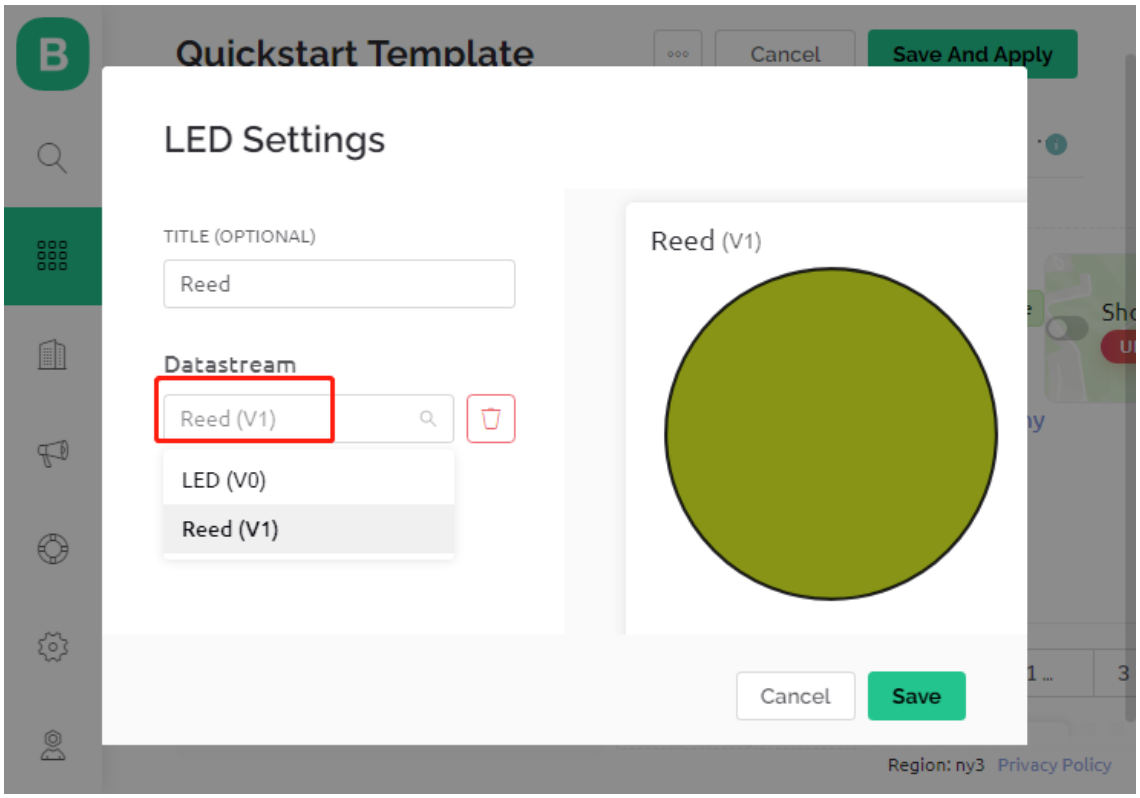
1. **Datastream** ページで、リードスイッチの値を取得するための **Virtual Pin** タイプの **Datastream** を作成します。データタイプは **Integer** 、MIN と MAX を **0** および **1** に設定します。




2. **Wed Dashboard** ページに **LED widget** をドラッグアンドドロップします。値が 1 の場合、それは光る（色付き）、それ以外の場合、それは白になります。




3. **LED widget** の設定ページで、 **Datastream** として **Reed(V1)** を選択し、保存します。



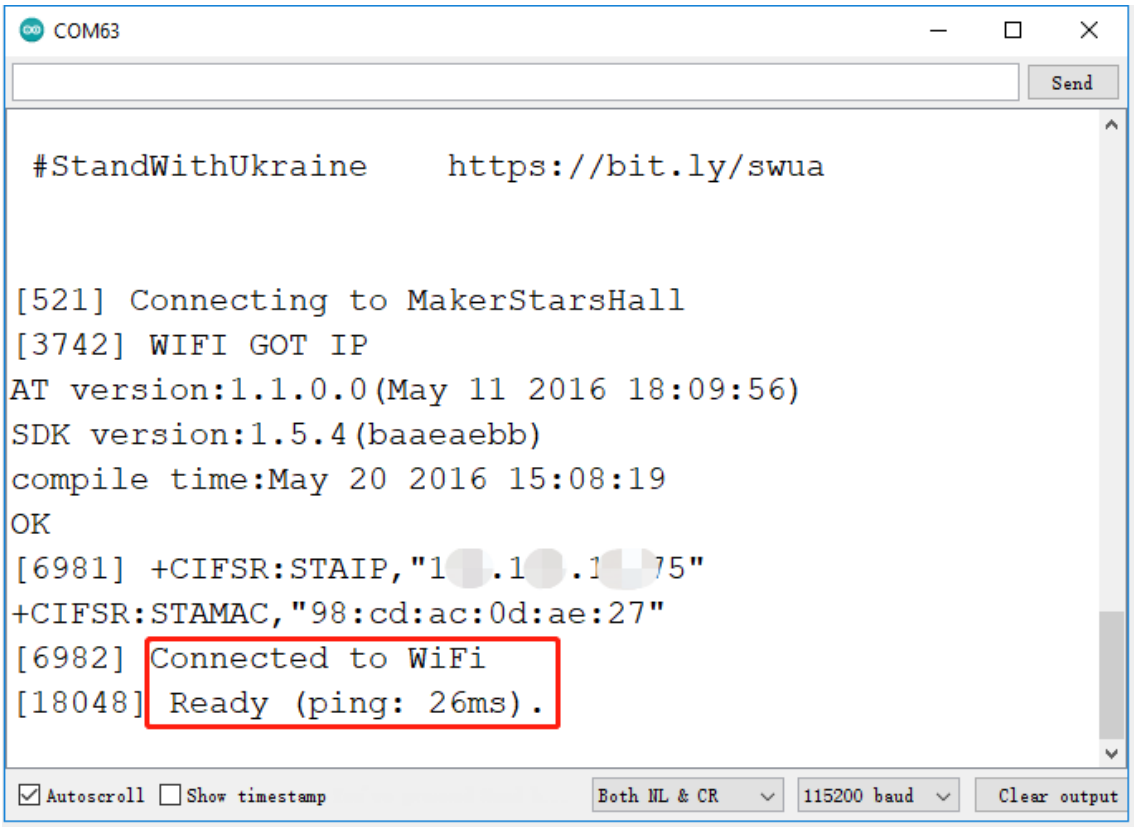

### 3. コードの実行

1. `3in1-kit\iot_project\3.push_data_to_blynk` のパスの下にある `3.push_data_to_blynk.ino` ファイルを開く、またはこのコードを **Arduino IDE** にコピーします。

2. あなた自身の `Template ID`、`Device Name`、および `Auth Token` で置き換えます。また、使用している WiFi の `ssid` と `password` を入力する必要があります。詳しいチュートリアルは、 *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 正しいボードとポートを選択した後、 **Upoad** ボタンをクリックします。

4. シリアルモニタを開き（ボーレートを 115200 に設定)、成功した接続などのプロンプトが表示されるのを待ちます。




---

**注釈:** 接続時に `ESP is not responding` というメッセージが表示される場合、以下の手順に従ってください。

- 9V のバッテリーが接続されていることを確認してください。
- RST ピンを GND に 1 秒間接続して ESP8266 モジュールをリセットし、その後、それを抜きます。
- R3 ボード上のリセットボタンを押します。

ときどき、上記の操作を 3〜5 回繰り返す必要がある場合がありますので、お待ちください。

---

5. これで、Blynk はあなたのドアや窓の状態を表示します。ドアや窓が閉まっている場合、LED ウィジェットは緑色になり、それ以外の場合は灰色になります。
6. Blynk をモバイルデバイスで使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。

**どのように動作するのか？**

この例では、以下の行に注目する必要があります。「Blynk Cloud の V1 Datastream に毎秒データを書き込む」は、これらの行で定義されています。

```
BlynkTimer timer;

void myTimerEvent()
{
    Blynk.virtualWrite(V1, pinValue);
}

void setup()
{
    timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
    timer.run(); // BlynkTimer を起動
}
```

Blynk ライブラリは組み込みのタイマーを提供しています、まずタイマーオブジェクトを作成します。

```
BlynkTimer timer;
```

setup() 内でタイマーの間隔を設定します。ここでは、1000ms ごとに myTimerEvent() 関数を実行するように設定しています。

```
timer.setInterval(1000L, myTimerEvent);
```

loop() で BlynkTimer を実行します。

```
timer.run();
```

カスタム関数 myTimerEvent() を編集します。コード Blynk.virtualWrite(V1, pinValue) は、V1 のデータ pinValue を書き込むために使用されます。

```
void myTimerEvent()
{
    Blynk.virtualWrite(V1, pinValue);
}
```

## 6.4 4. クラウド音楽プレイヤー

このプロジェクトの目標は Blynk を使った音楽プレイヤーを作ることです。音楽の再生は *5.7 Tone()* **または** *noTone()* と同じように、プログラムに曲を書いてパッシブ・ブザーで再生する。しかし、この例では、スイッチをクリックすると再生/一時停止ができ、スライダーをスライドさせると再生の進行状況を変えることができます。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

一式をまとめて購入するのは便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから、部品を個別に購入することもできます。

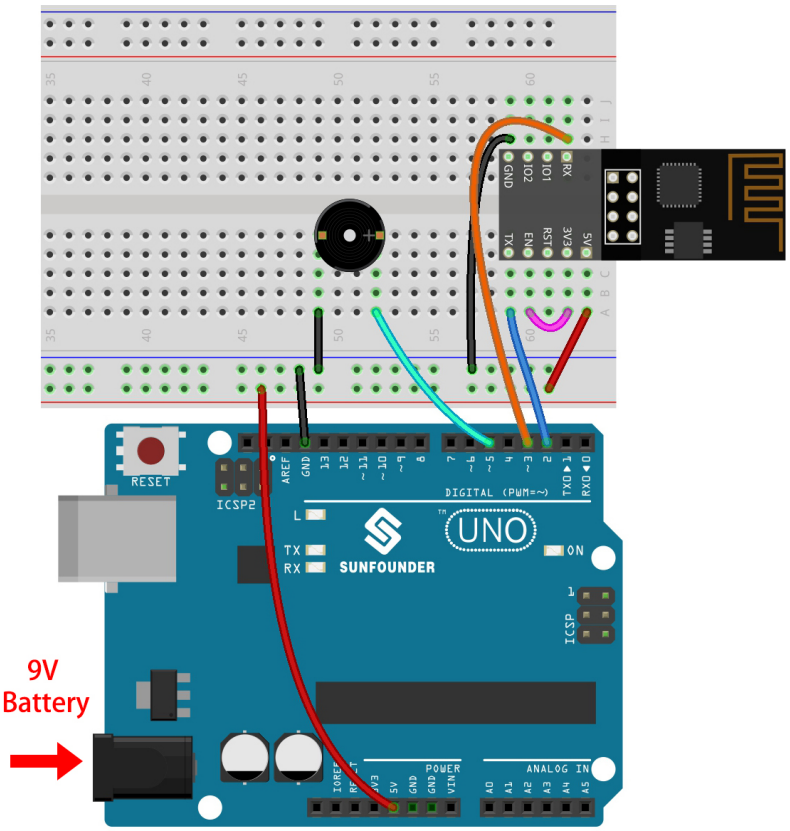
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| *ESP8266* **モジュール** | |
| **ジャンパーワイヤー** | |
| **ブザー** | |

**1. 回路を組む**

---

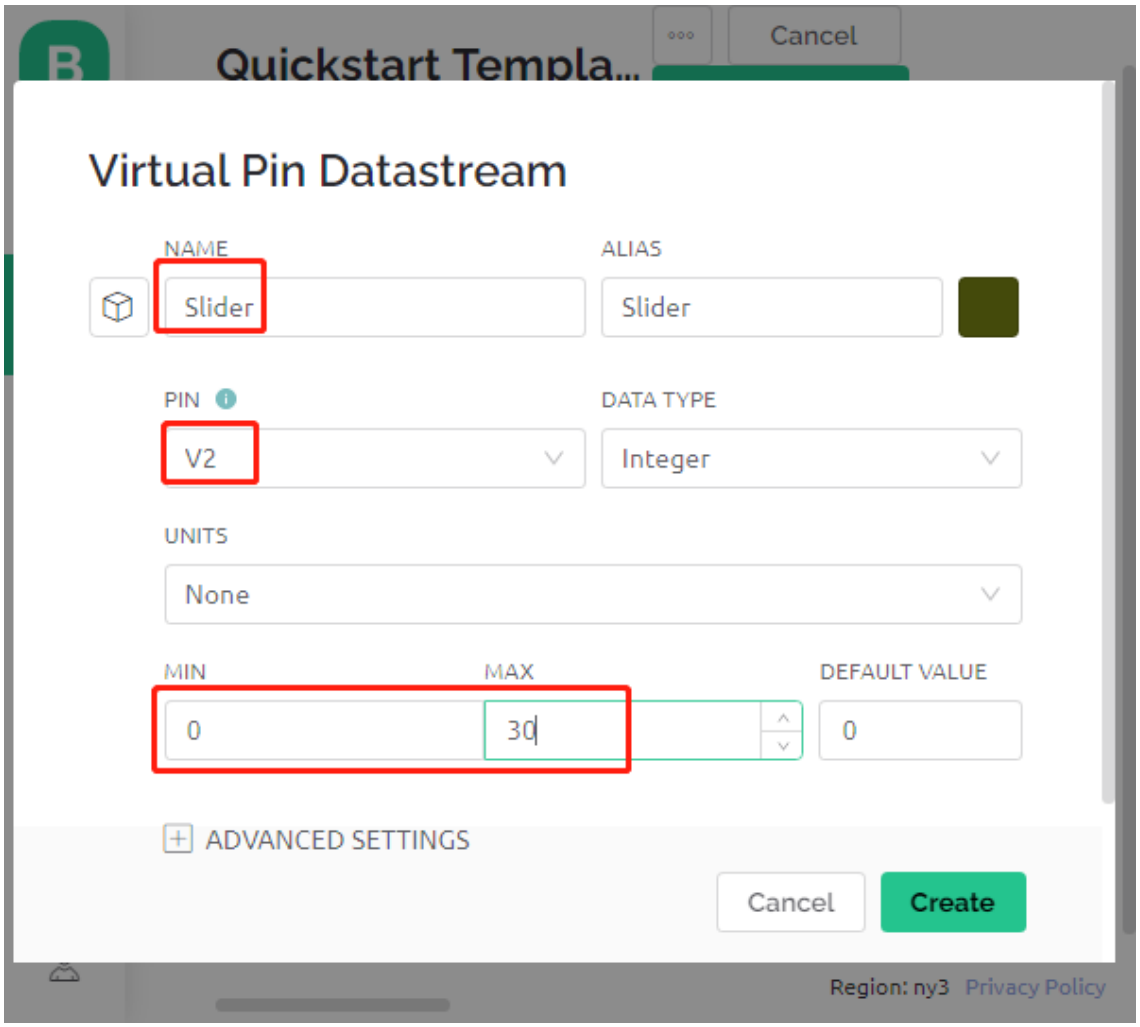
**注釈:** ESP8266 モジュールは、安定した動作環境を確保するために高電流が必要です。9V のバッテリーが接続されていることを確認してください。

---

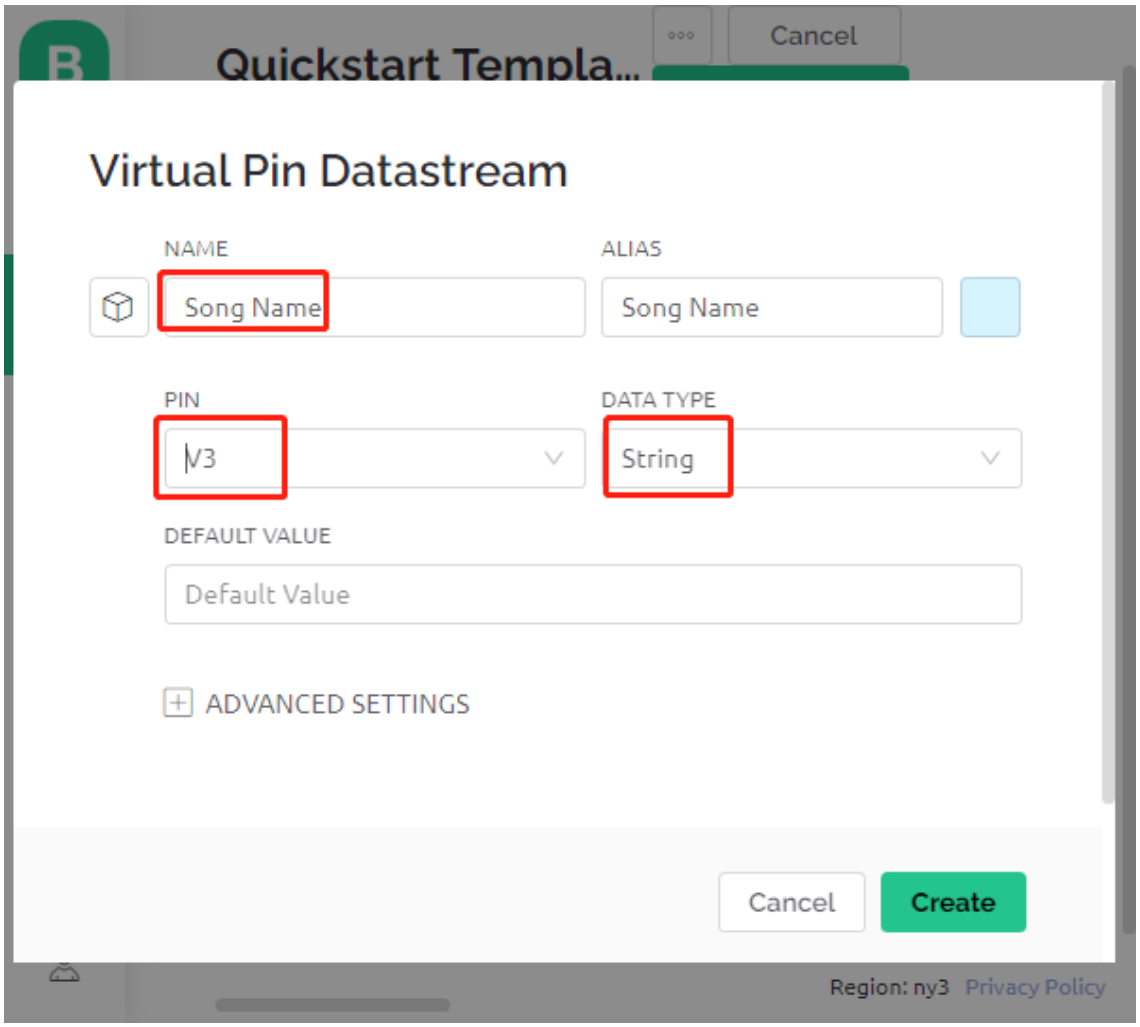



### 2. ダッシュボードを編集

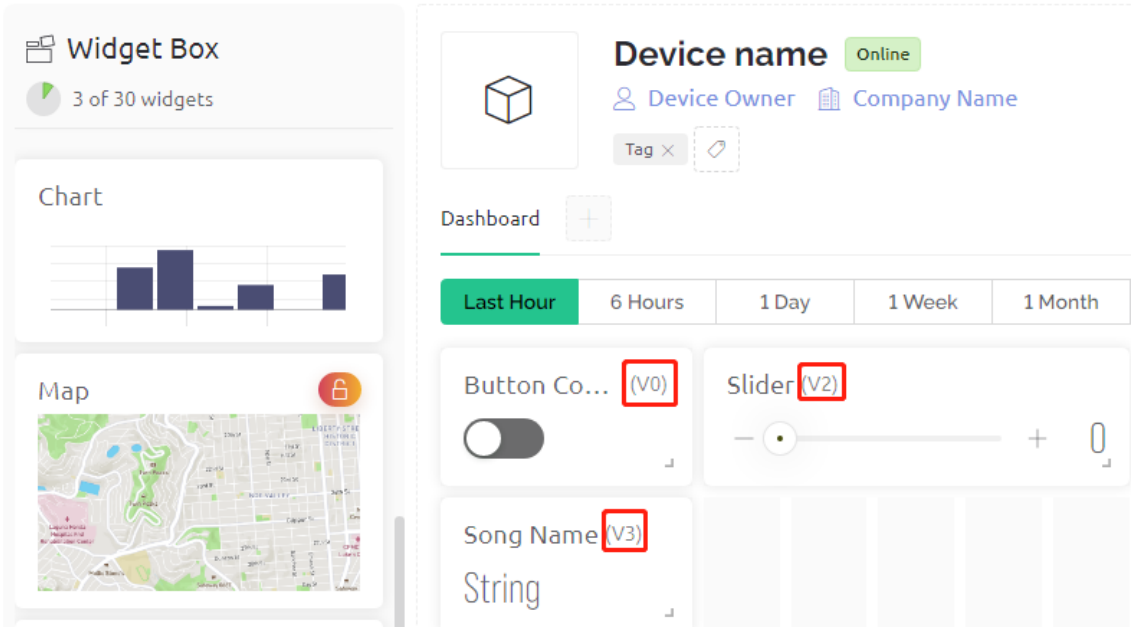
1. **Datastream** ページで、後で追加されるスライダーウィジェットまたはコードで変更する値として **Virtual Pin** タイプの **Datastream** を作成します。データタイプは **Integer** にし、MIN と MAX を **0** と **30** に設定します。




2. 音楽の名前を表示するための **Virtual Pin** タイプの **Datastream** も作成します。データタイプは `String` に設定してください。




3. **Wed Dashboard** ページに移動して、 **Switch** ウィジェットをドラッグし、 **Datastream** を V0 に設定します ( *2. Blynk* **からデータを取得** で既に設定済み); **Label** ウィジェットをドラッグして V3 に設定; **Slider** ウィジェットをドラッグして V2 に設定します。




**注釈:** あなたの仮想ピンの番号は私のものと異なるかもしれません。あなたの設定が優先されますので、コード内の対応するピン番号を修正してください。

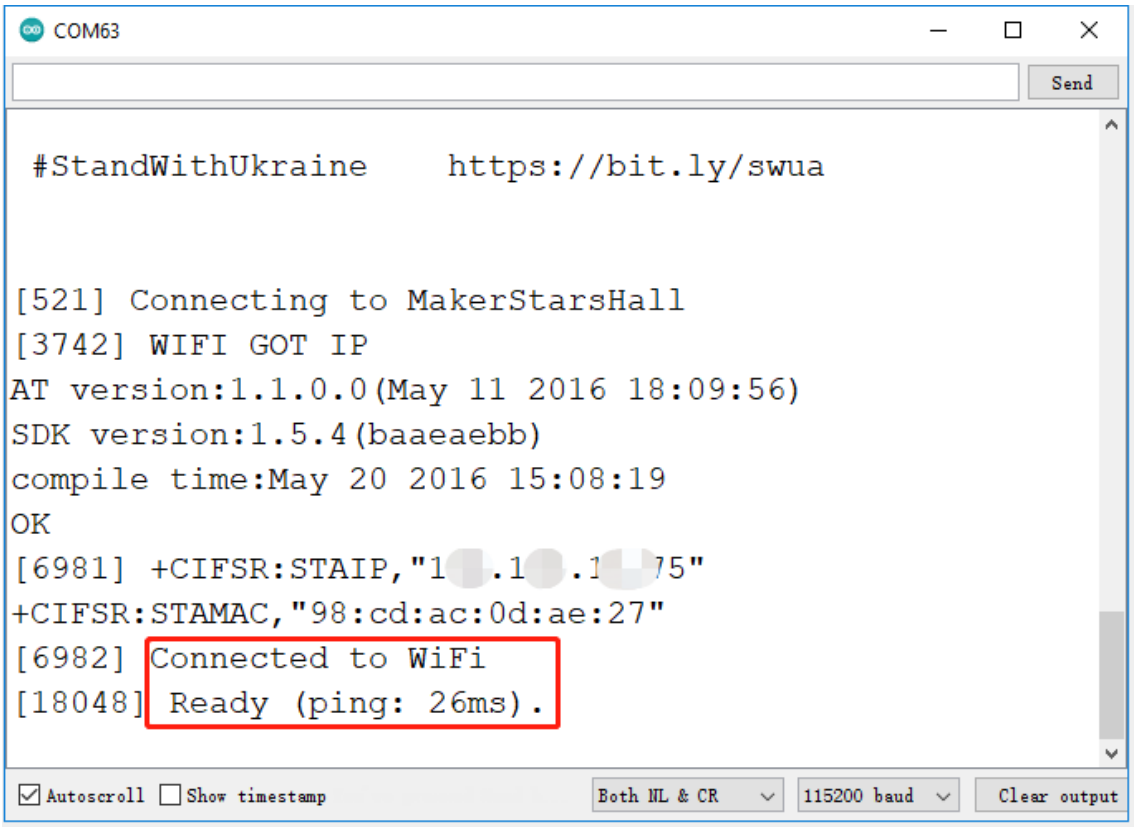
### 3. コードの実行

1. パス `3in1-kit\iot_project\4.cloud_music_player` の下にある `4.cloud_music_player.ino` ファイルを開きます。

2. `Template ID`、`Device Name`、および `Auth Token` を自分のものに置き換えます。使用している WiFi の `ssid` と `password` も入力する必要があります。詳しいチュートリアルについては、 *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 正しいボードとポートを選択した後、 **Upoad** ボタンをクリックします。

4. シリアルモニタを開き（ボーレートを 115200 に設定)、成功した接続のようなプロンプトが表示されるのを待ちます。




---

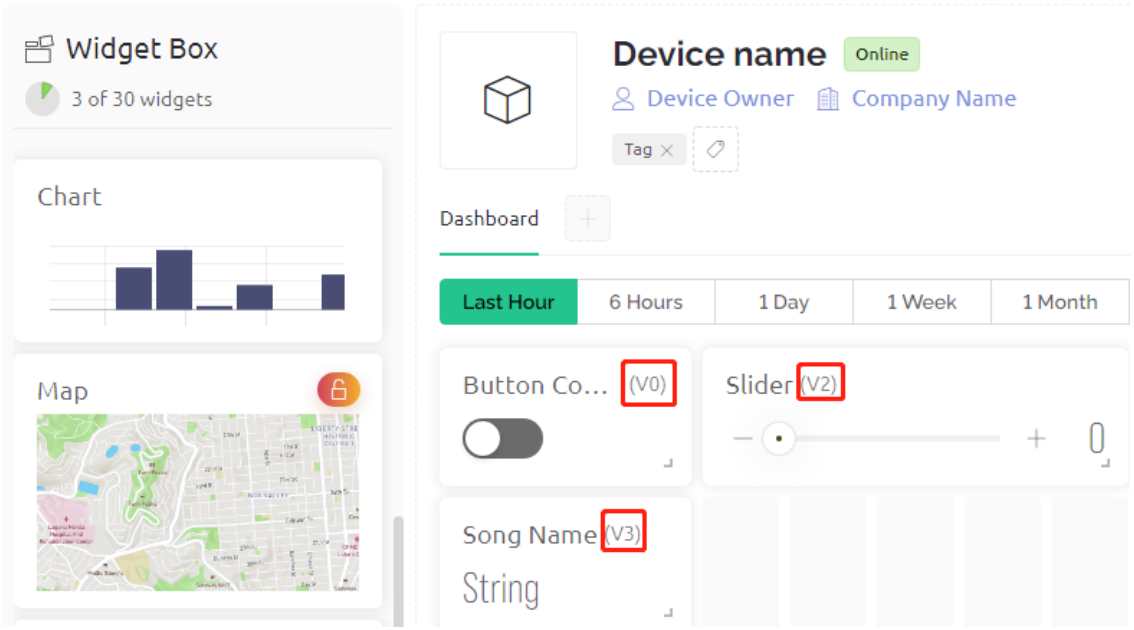
**注釈:** 接続時に `ESP is not responding` メッセージが表示される場合は、以下の手順に従ってください。

- 9V のバッテリーが接続されていることを確認してください。
- RST ピンを 1 秒間 GND に接続して ESP8266 モジュールをリセットし、その後プラグを抜きます。
- R3 ボードのリセットボタンを押します。

ときどき、上記の操作を 3〜5 回繰り返す必要がある場合があります。お待ちください。

---

5. これで、Blynk のボタンコントロールウィジェットを使用して音楽の再生/一時停止を切り替えたり、スライダーで再生の進行状況を調整したりできます。また、音楽の名前も表示されます。




6. Blynk をモバイルデバイスで使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。

**どのように動作するのか？**

データストリーム **V0** は、Switch ウィジェットのステータスを取得し、変数 **musicPlayFlag** に割り当てるために使用されます。これは音楽の再生と一時停止を制御します。

```
int musicPlayFlag=0;

BLYNK_WRITE(V0)
{
    musicPlayFlag = param.asInt(); // 音楽の開始/一時停止
}
```

データストリーム **V2** は、スライダーウィジェットの値を取得し、スライダーが移動したときに変数 **scrubBar** に割り当てるために使用されます。

```
int scrubBar=0;

BLYNK_WRITE(V2)
{
    scrubBar=param.asInt();
}
```

デバイスが **Blynk Cloud** に接続されているとき、 **V3** データストリームの音楽の名前を書き込み、それを **Label** ウィジェットで表示します。

```
BLYNK_CONNECTED() {
    String songName = "Ode to Joy";
    Blynk.virtualWrite(V3, songName);
}
```

**Blynk Timer** は毎秒実行されます。 **musicPlayFlag** が 0 でない場合、つまり **Switch** ウィジェットが ON の場合、音楽が再生されます。2 つのノートが再生されると、進行バー変数 scrubBar が 2 増加し、その値は次に **Blynk Cloud** に書き込まれ、 **Slider** ウィジェットの値が同期されます。

```
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. 住宅環境監視

この章では、Blynk を使用して住宅の環境モニターを作成します。DHT11 とフォトレジスタを使用して、部屋の温度、湿度、および光の強度を測定できます。これらの値を Blynk に送信することで、インターネット経由で自宅の環境を知ることができます。

**必要な部品**

このプロジェクトには、以下の部品が必要です。

全体のキットを購入すると非常に便利です。以下のリンクを参照してください：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

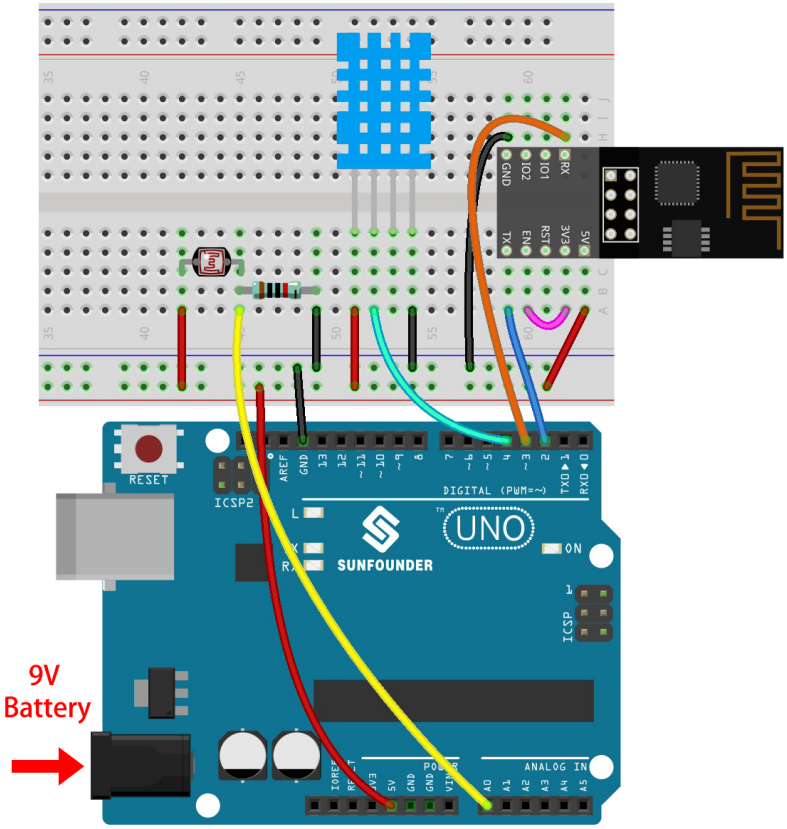
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| *ESP8266* モジュール | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| フォトレジスタ | |
| *DHT11* 湿度温度センサ | - |

**1. 回路を組む**

---

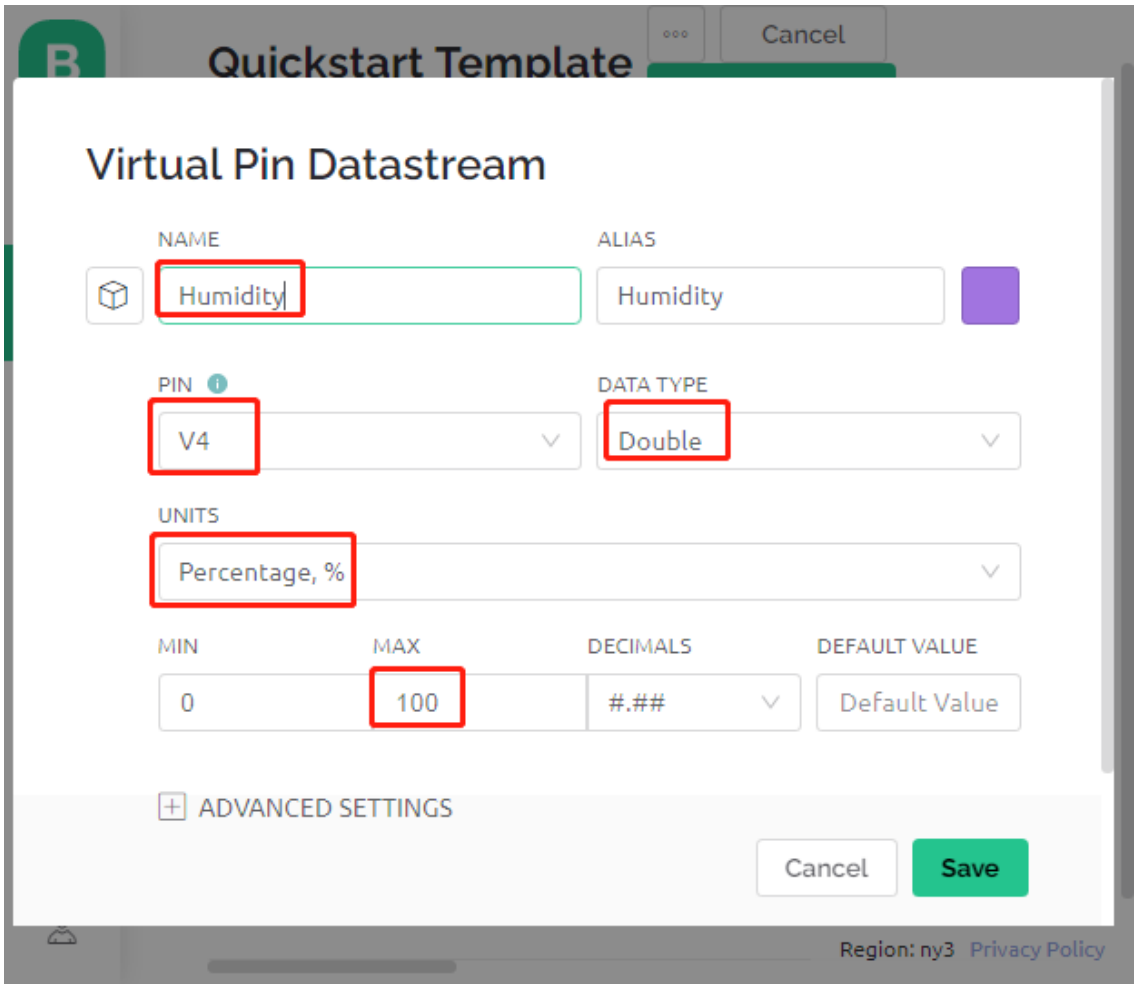
**注釈:** ESP8266 モジュールは安定した動作環境を提供するために高い電流が必要ですので、9V のバッテリーが接続されていることを確認してください。

---

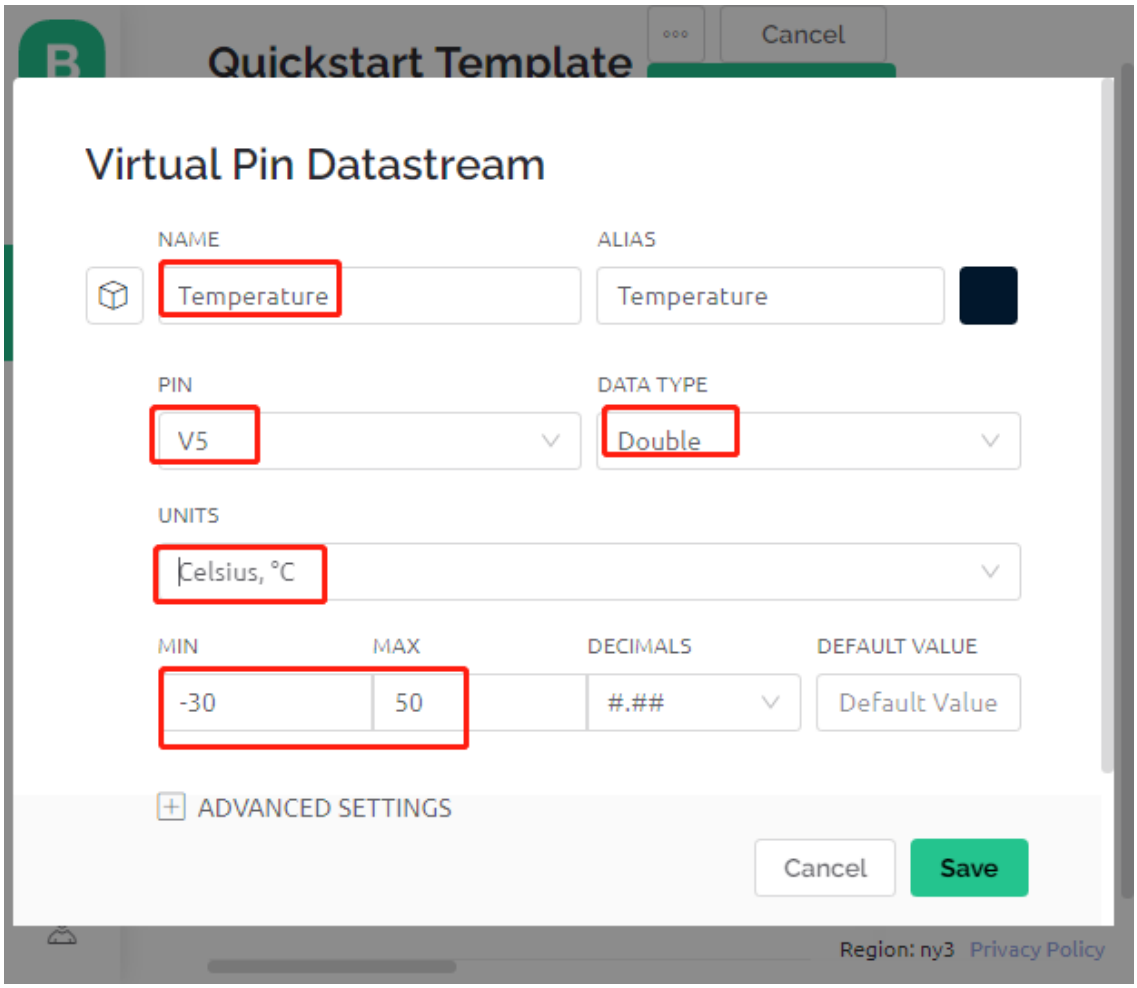



### 2. ダッシュボードを編集

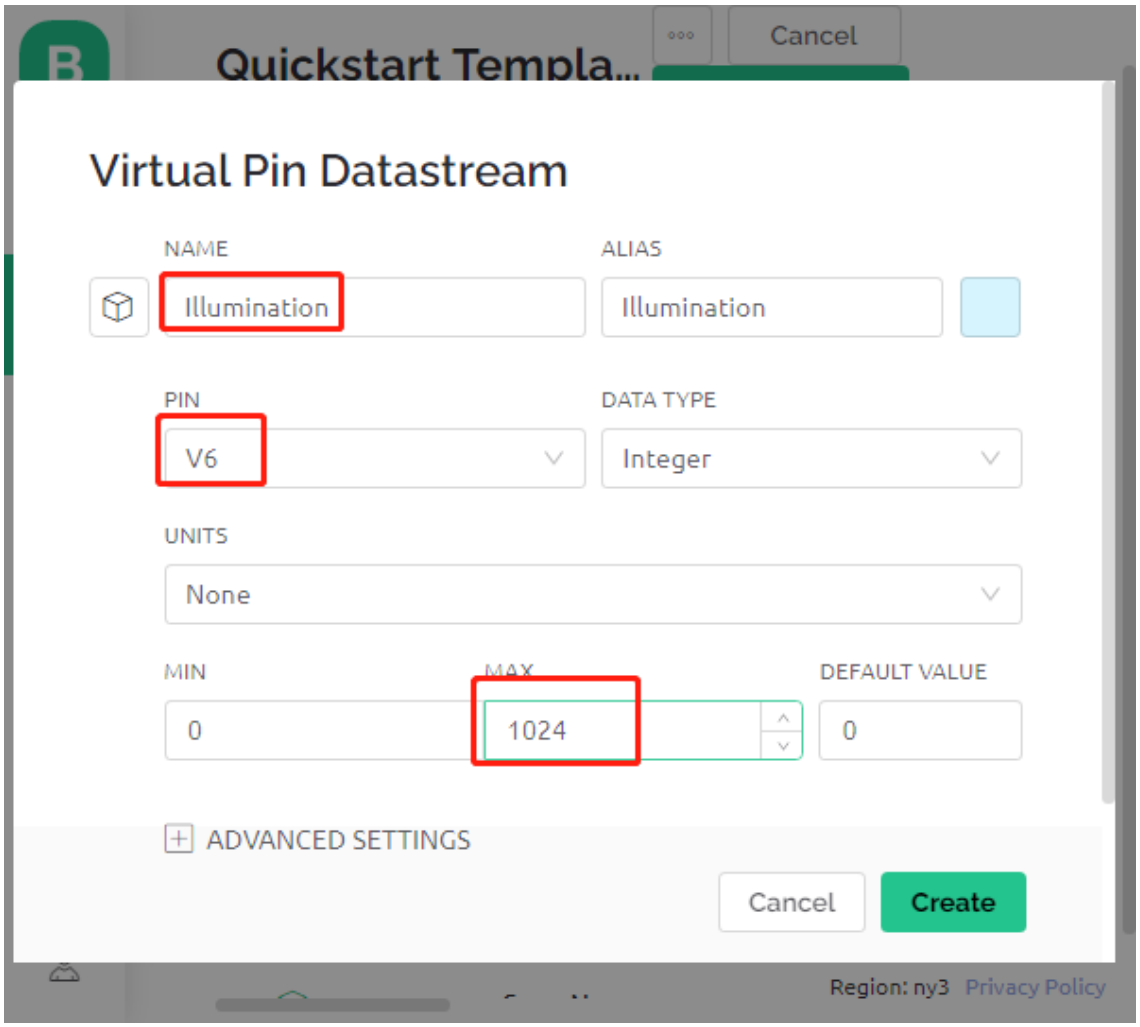
1. 湿度の値を記録するために、**Datastream** ページで **Virtual Pin** タイプの **Datastream** を作成します。DATA TYPE を **Double** に設定し、MIN と MAX を **0** と **100** に設定します。また、単位を **Percentage, %** に設定します。



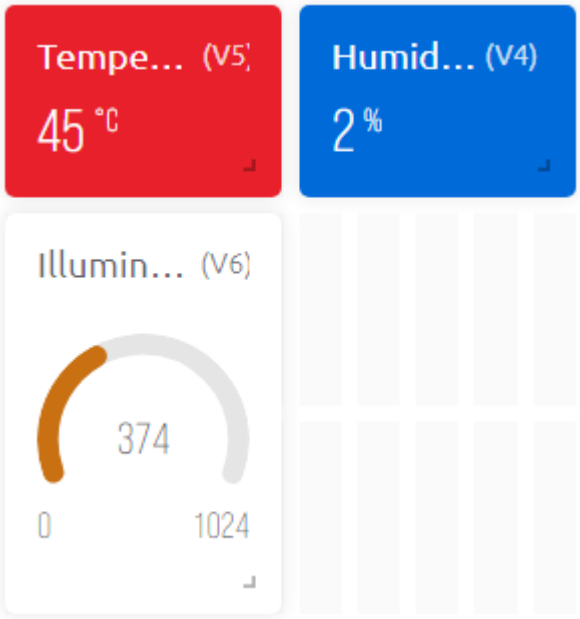

2. 次に、温度を記録するための **Virtual Pin** タイプの **Datastream** を作成します。DATA TYPE を `Double` に設定し、MIN と MAX を `-30` と `50` に設定し、単位を **Celsius, ° C** にします。




3. また、光の強度を記録するための **Virtual Pin** タイプの **Datastream** を作成します。デフォルトのデータタイプである **Integer** を使用し、MIN と MAX を `0` と `1024` に設定します。




4. **Wed Dashboard** ページに移動し、2 つの **Label** ウィジェットをドラッグして、それぞれのデータストリームを **V4** および **V5** に設定し、 **Gauge** ウィジェットをドラッグしてデータストリームを **V6** に設定します。ウィジェットの設定でも、 **値に基づいて色を変更** を有効にし、適切な色を選択してウィジェットをより見やすく、直感的にすることができます。



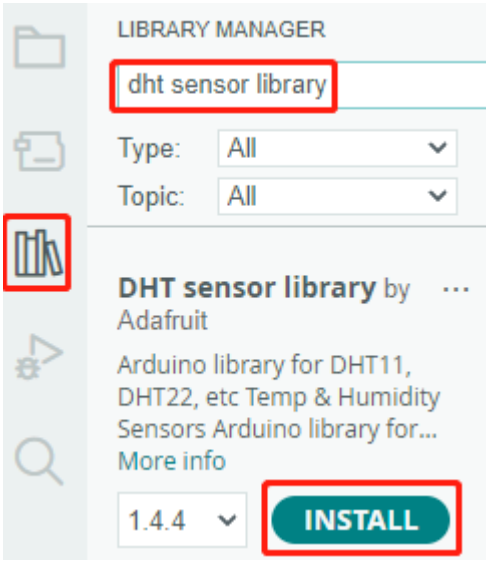

### 3. コードの実行

1. 3in1-kit\iot_project\5.home_environment_monitoring のパスの下にある 5. home_environment_monitoring.ino ファイルを開くか、このコードを **Arduino IDE** にコピーします。

---

**注釈:**

- ここでは DHT sensor library を使用しています。 **Library Manager** からインストールできます。

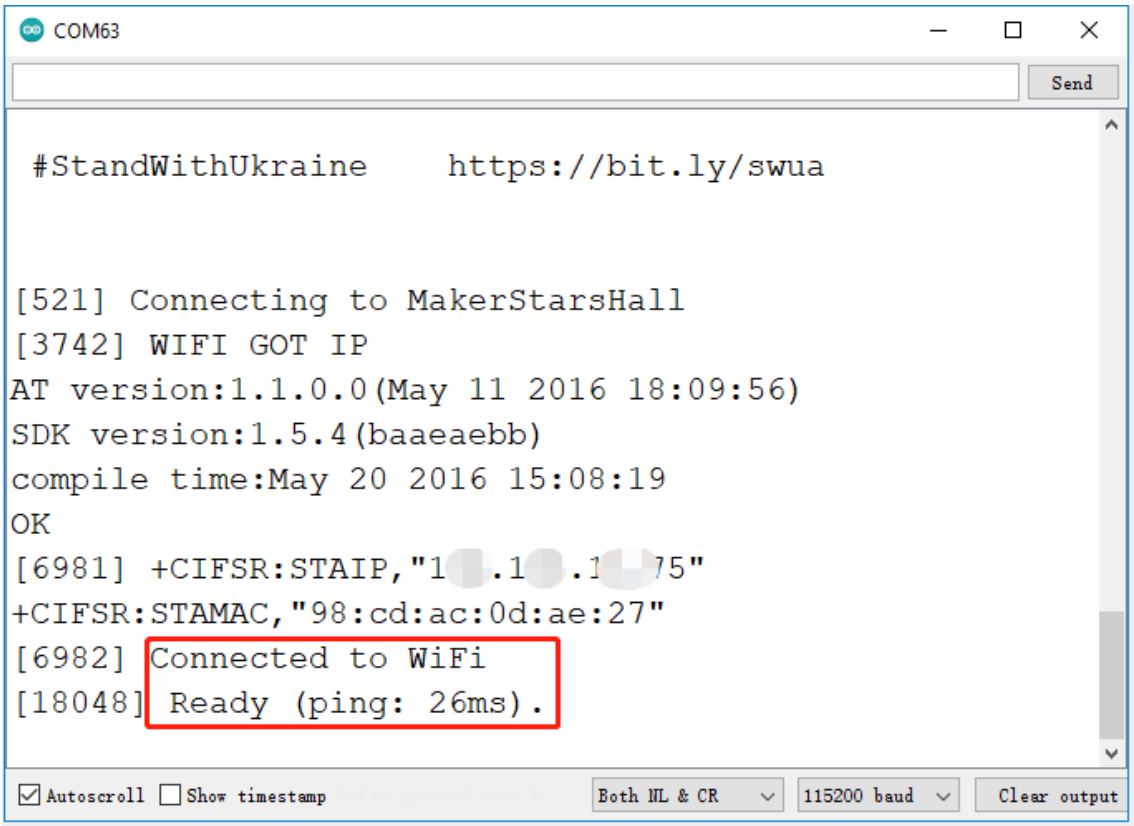



---

2. `Template ID`、 `Device Name`、そして `Auth Token` を自分のものに置き換えてください。また、使用している WiFi の `ssid` と `password` も入力する必要があります。詳しいチュートリアルは *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 適切なボードとポートを選択した後、 **Upoad** ボタンをクリックします。

4. シリアルモニター（ボーレートを 115200 に設定）を開き、成功した接続のようなプロンプトが表示されるのを待ちます。




---

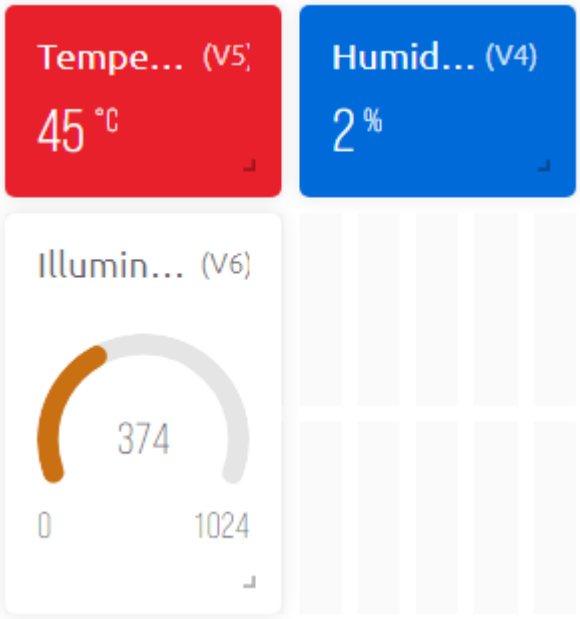
**注釈:** 接続時に `ESP is not responding` というメッセージが表示された場合は、以下の手順に従ってください。

- 9V のバッテリーが接続されていることを確認してください。
- RST ピンを 1 秒間 GND に接続して ESP8266 モジュールをリセットし、その後、プラグを抜いてください。
- R3 ボードのリセットボタンを押します。

こうした操作を 3〜5 回繰り返すことが必要な場合もありますので、お待ちください。

---

5. 今、Blynk 上に現在の室温、湿度、光の強度が表示されるようになります。




6. Blynk をモバイルデバイスで使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。


**どのように動作するのか？**

以下の二つの関数は、部屋の温度、湿度、および光の強度を取得するために使用されます。

```
int readLight(){
    return analogRead(lightPin);
}
```

(次のページに続く)

(前のページからの続き)

```
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)
    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;
}
```

Blynk の Timer を使用して、毎秒、室内の温度、湿度、および光の強度が取得され、Blynk Cloud のデータストリームに送信されます。このデータはウィジェットによって表示されます。

```
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. 植物モニター

本プロジェクトの目的は、現在の温度、湿度、光の強度、土壌の湿度を検出し、それらを Blynk に表示するスマートな水やりシステムを作成することです。

Blynk Cloud のスイッチトグルをオンにすると、ポンプが動作し始め、植物に水が供給されます。

**必要な部品**

このプロジェクトでは、以下の部品が必要です。

一式を購入するのが便利です。リンクは以下のとおりです:

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

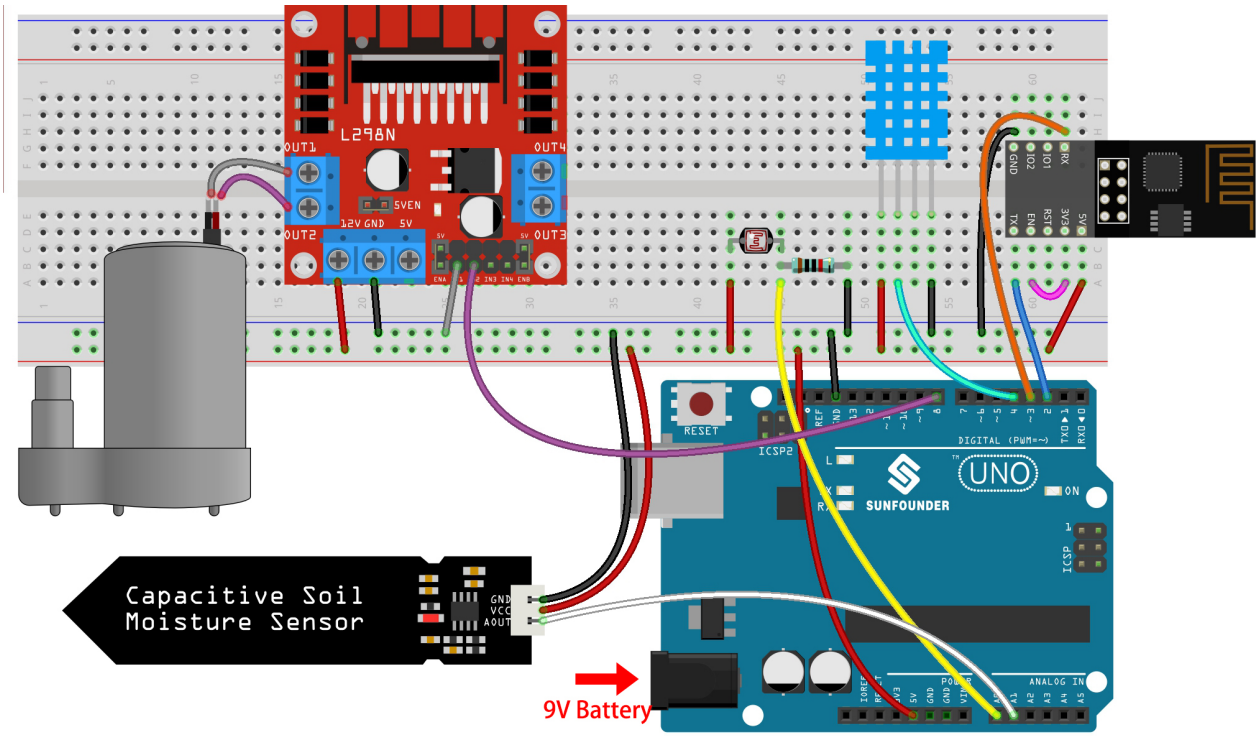
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| *ESP8266* モジュール | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| フォトレジスタ | |
| *DHT11* 湿度温度センサ | - |
| 土壌湿度モジュール | |
| *L298N* モジュール | |
| 遠心ポンプ | - |

**1. 回路を組む**

---

**注釈:** ESP8266 モジュールは、安定した動作環境を提供するために高電流が必要ですので、9V のバッテリーが接続されていることを確認してください。

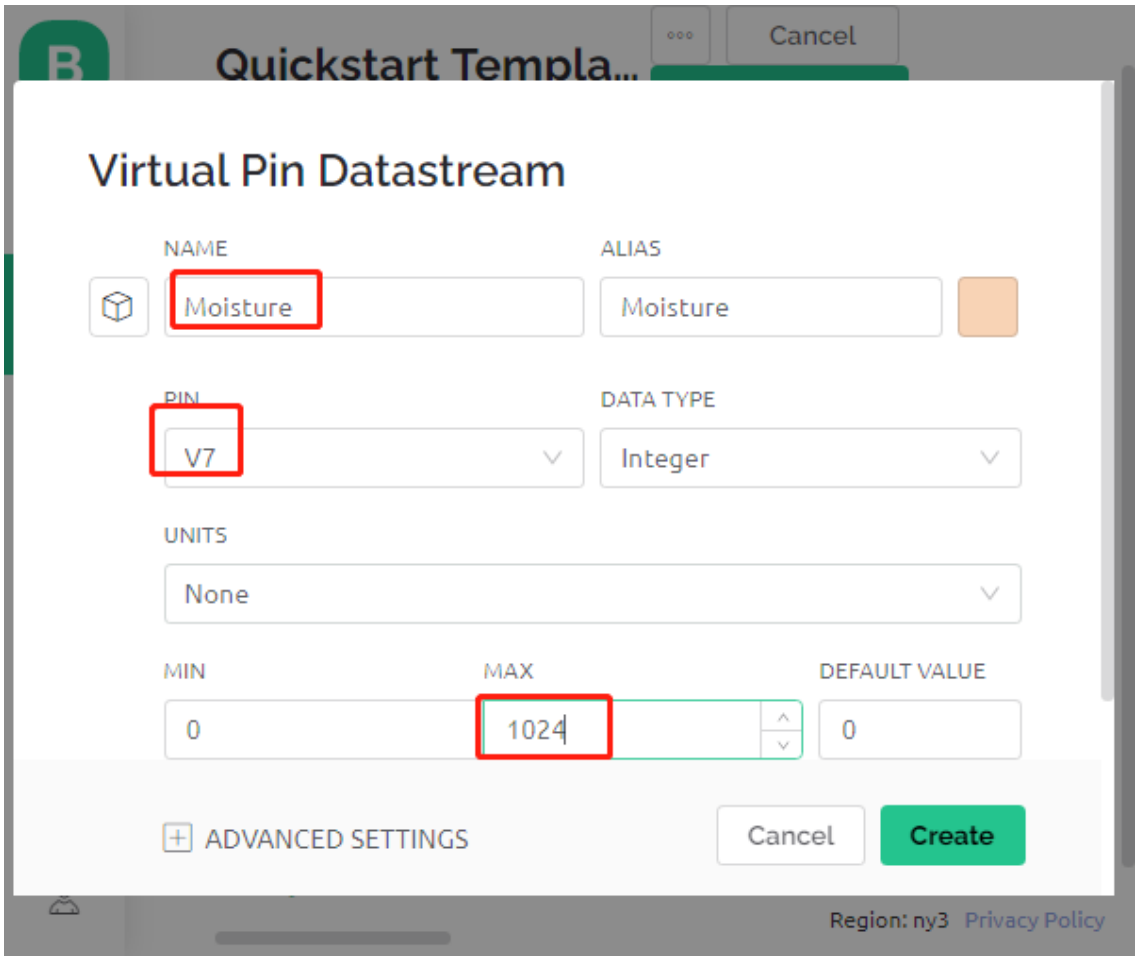
---



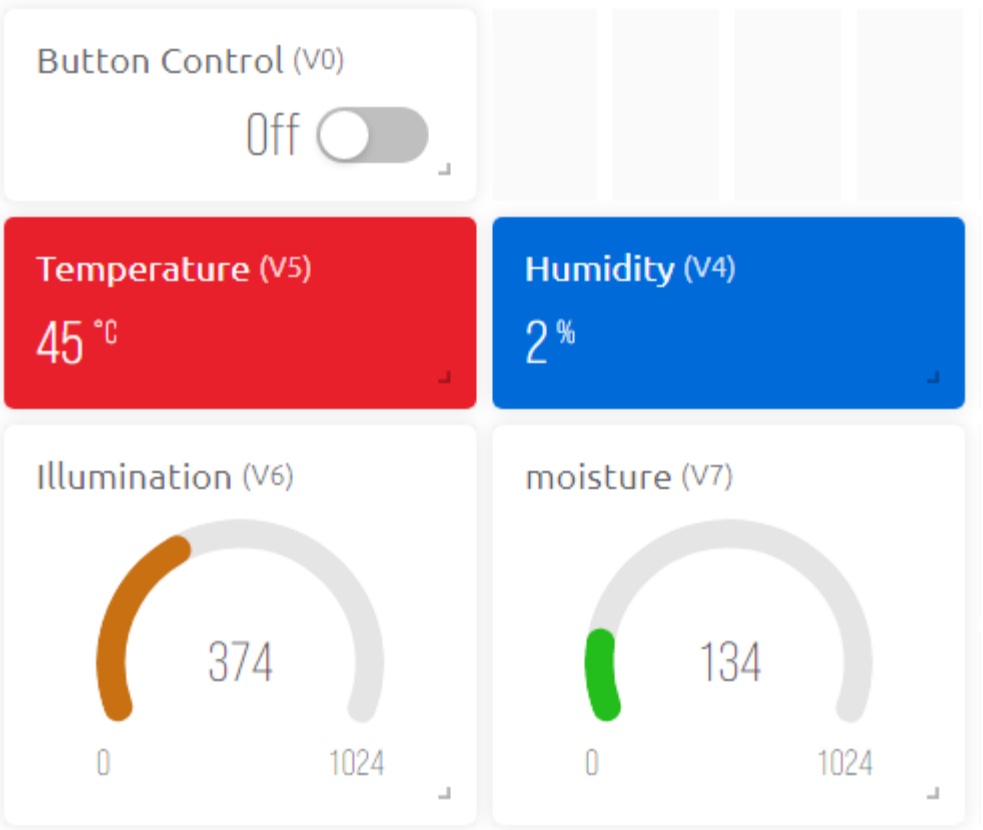

### 2. ダッシュボードを編集

1. 以前のプロジェクトで作成したデータストリームは保存しておく必要があり、このプロジェクトでも使用されます。

2. 土壌の湿度を記録するために、 **Datastream** ページで **Virtual Pin** タイプの別の **Datastream** を作成します。DATA TYPE を `Integer` に設定し、MIN と MAX を `0` および `1024` に設定します。




3. さらに、 **Web Dashboard** ページに移動し、2 つの **Label** ウィジェットをドラッグして、それぞれのデータストリームを **V4** および **V5** に設定します。次に、2 つの **Gauge** ウィジェットをドラッグして、それぞれのデータストリームを **V6** および **V7** に設定します。最後に、 **Switch** ウィジェットをドラッグして、そのデータストリームを **V0** に設定します。



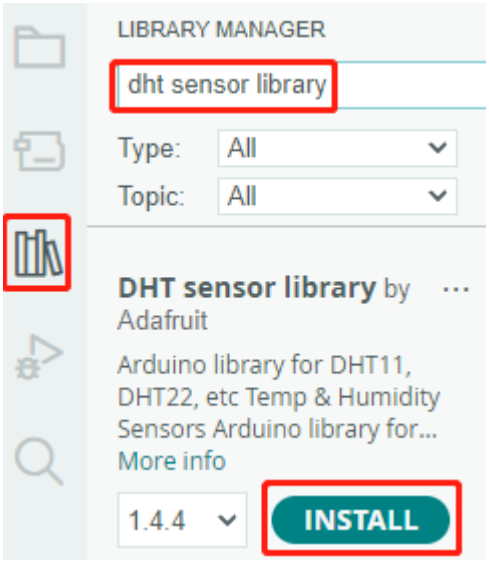

### 3. コードの実行

`3in1-kit\iot_project\6.plant_monitoring` のパスの下で `6.plant_monitoring.ino` ファイルを開くか、このコードを **Arduino IDE** にコピーします。

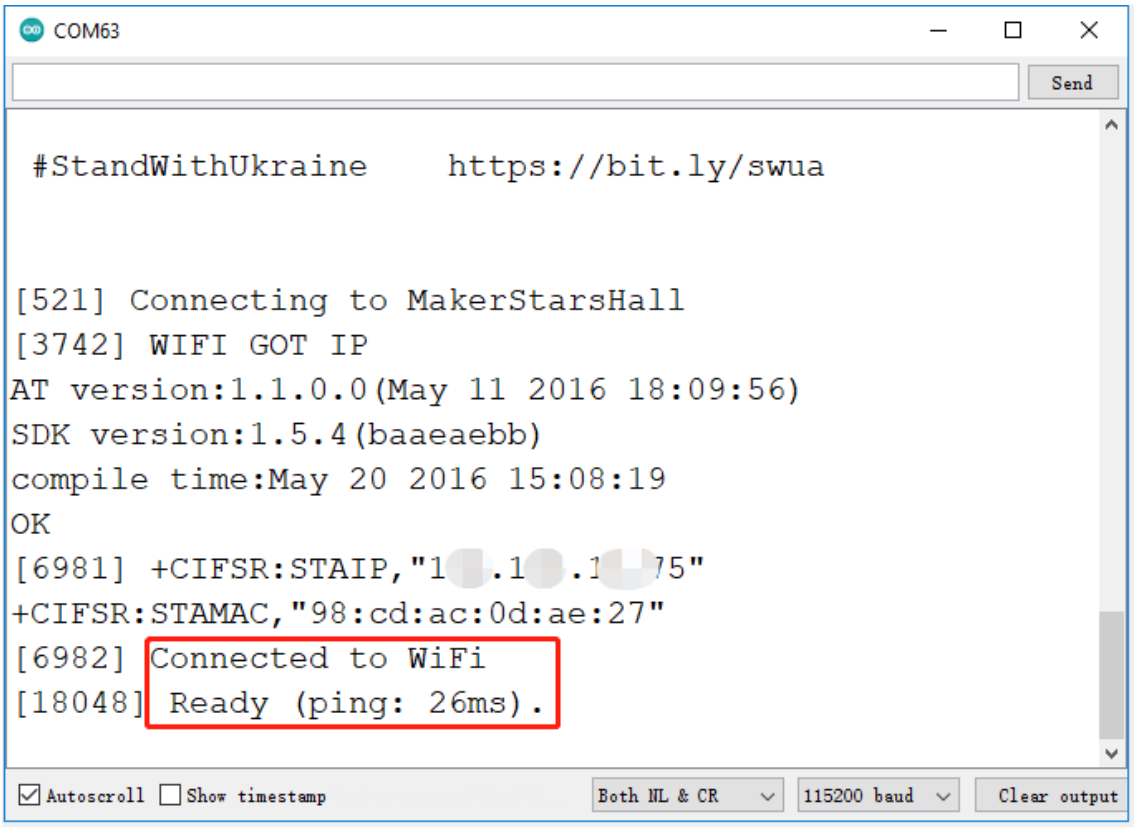
---

**注釈:**

- ここでは `DHT sensor library` が使用されています。 **Library Manager** からインストールできます。




1. `Template ID`、`Device Name`、`Auth Token` を自分のものに置き換えます。また、使用している WiFi の `ssid` と `password` を入力する必要があります。詳細なチュートリアルは *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。
2. 正しいボードとポートを選択した後、 **Upoad** ボタンをクリックします。
3. シリアルモニター (ボーレートを 115200 に設定) を開き、接続成功などのプロンプトが表示されるのを待ちます。




---

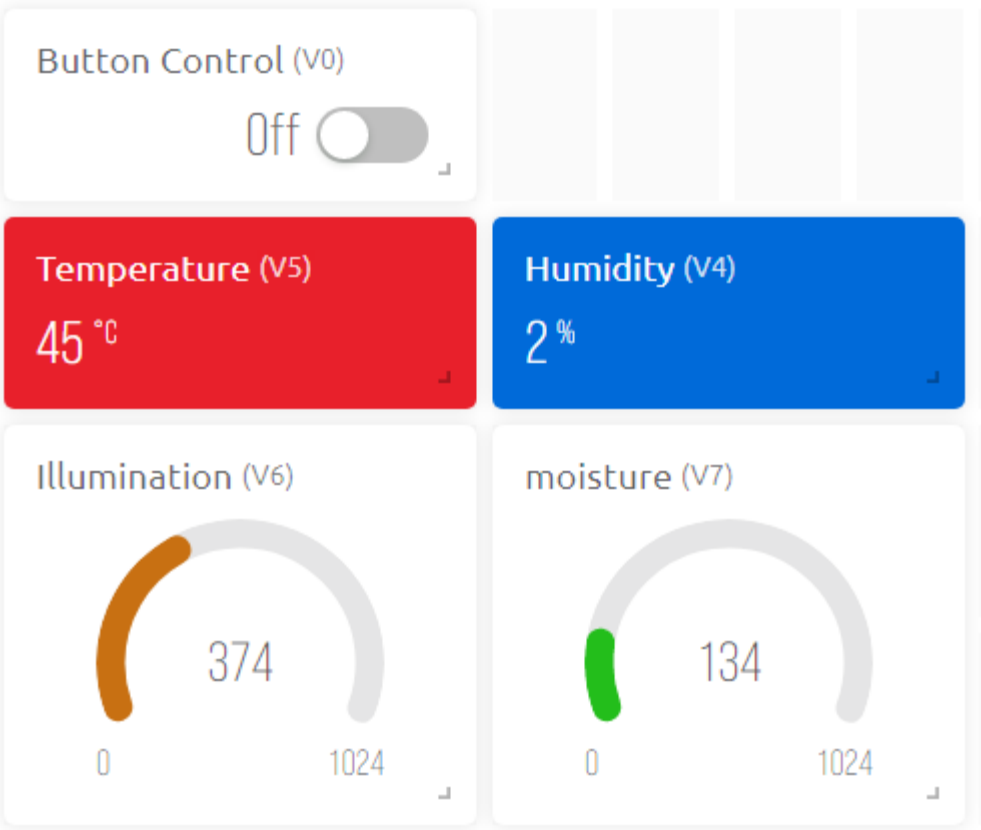
**注釈:** 接続時に `ESP is not responding` というメッセージが表示される場合、次の手順に従ってください。

- 9V のバッテリーが接続されていることを確認します。
- ピン RST を 1 秒間 GND に接続して ESP8266 モジュールをリセットし、その後プラグを抜きます。
- R3 ボードのリセットボタンを押します。

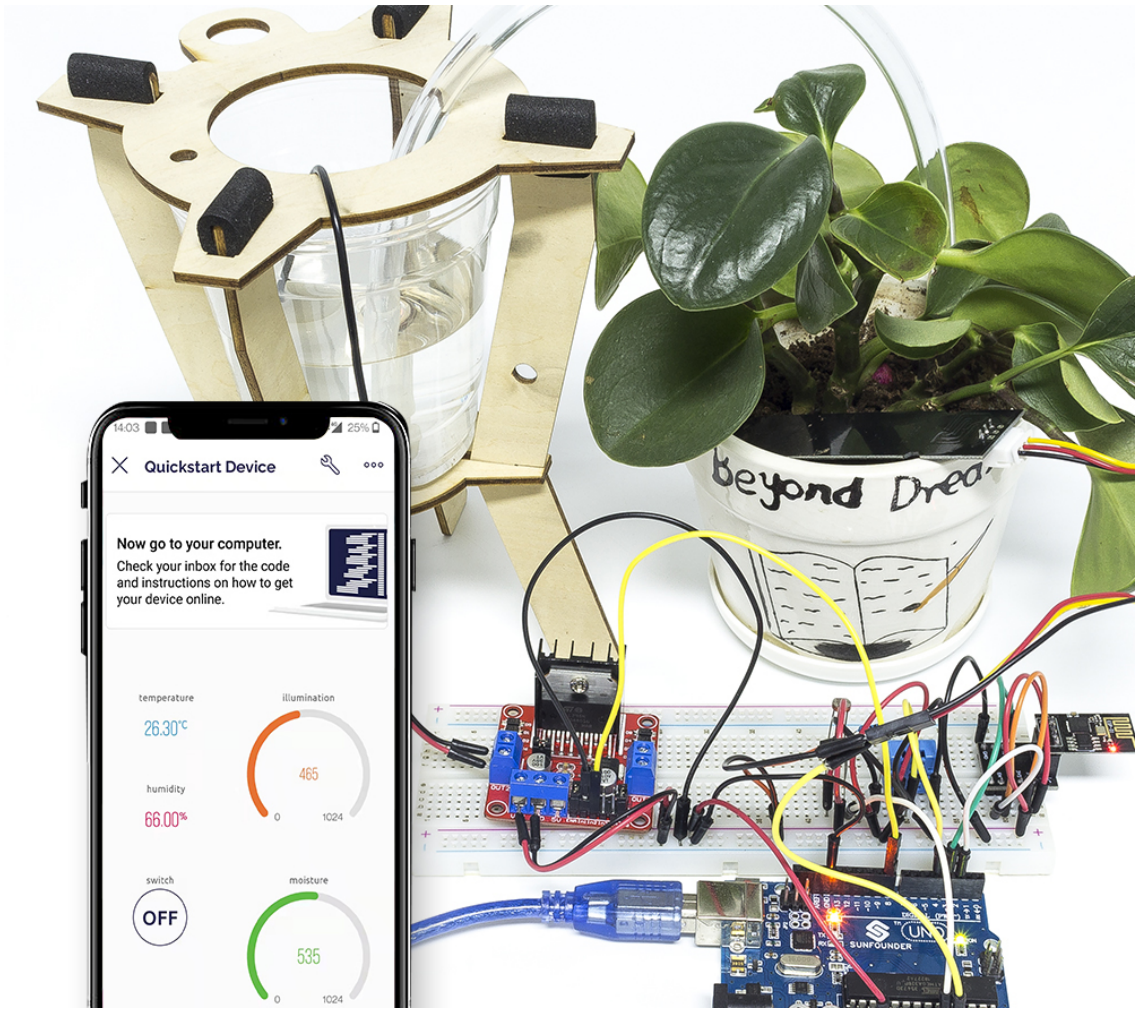
ときどき、上記の操作を 3〜5 回繰り返す必要がありますので、お待ちください。

---

4. Blynk に戻ると、現在の温度、湿度、光の強度、土壌の湿度が表示されます。必要に応じて、ボタン制御ウィジェットをクリックして植物に水をやることができます。




5. Blynk をモバイルデバイスで使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。




**どのように動作するのか？**

この BLYNK_WRITE は、Blynk の **Switch** ウィジェットが ON のときにポンプを起動し、OFF のときにポンプをオフにします。

```
BLYNK_WRITE(V0)
{
    if(param.asInt()==1){
        digitalWrite(pumpA,HIGH);
    }else{
        digitalWrite(pumpA,LOW);
    }
}
```

以下の三つの関数は、現在の環境温度、湿度、光の強度、土壌の湿度を取得するために使用されます。

```
int readMoisture(){
    return analogRead(moisturePin);
}

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)
    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;
}
```

Blynk の Timer を使用して、周囲の温度、湿度、光の強度、土壌の湿度が毎秒取得され、 **Blynk Cloud** 上のデータストリームに送信され、ウィジェットがデータを表示します。

```
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. 電流制限ゲート

一部の状況、例えば駐車場などでは、数量の管理が必要となります。

ここではスマートゲートを作成します。サーボをゲートとして使用し、その前に IR 障害物検出器を配置します。オブジェクト（例えば車）が検出されると、ゲートが開き、数字が 1 増加します。そのカウントは 7 セグメントディスプレイで表示され、Blynk クラウドにもアップロードされるので、遠隔での閲覧も可能です。最後に、Blynk にはこのスマートゲートシステムを有効/無効にするためのスイッチウィジェットがあります。

**必要な部品**

このプロジェクトでは、以下のコンポーネントが必要です。

全てのキットを一括で購入するのが便利です。リンクはこちら：

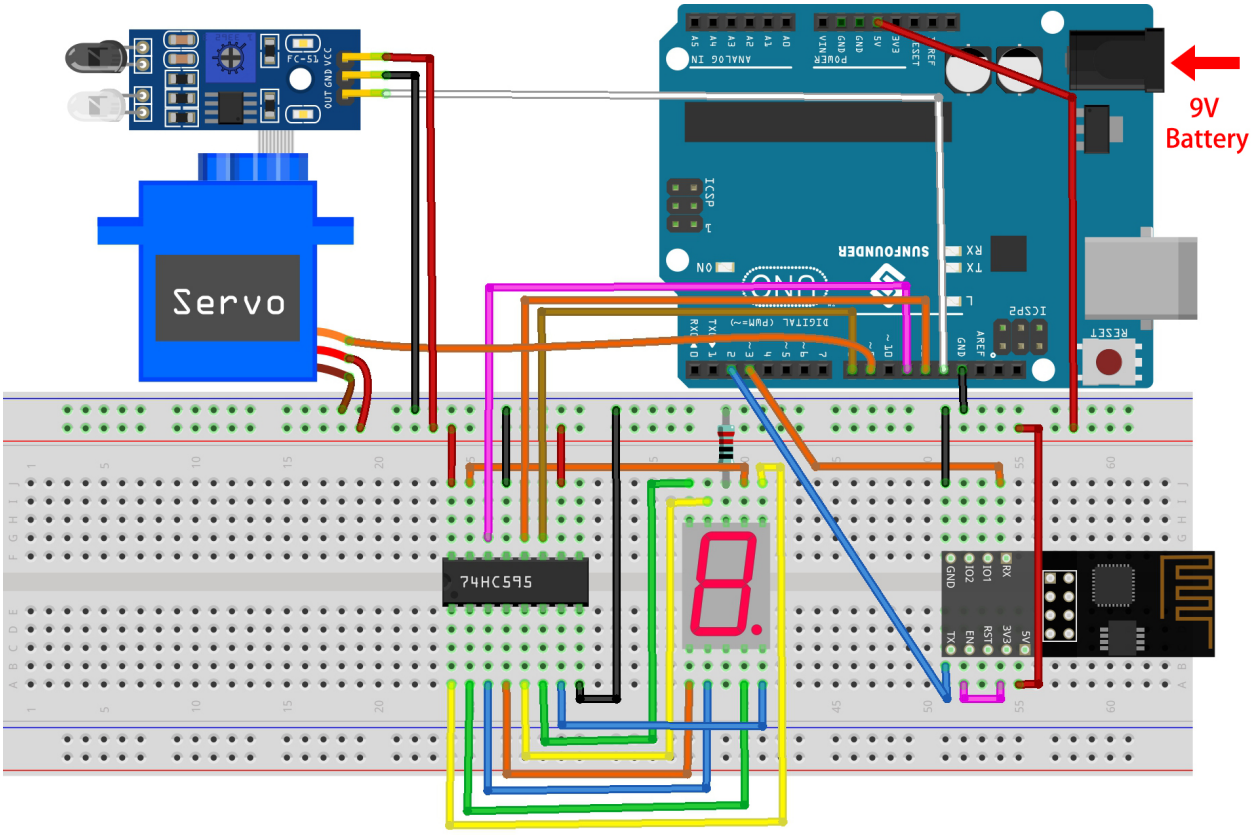
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| *ESP8266* モジュール | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| サーボ | |
| 障害物回避モジュール | |
| *7* セグメント表示 | |
| *74HC595* | |

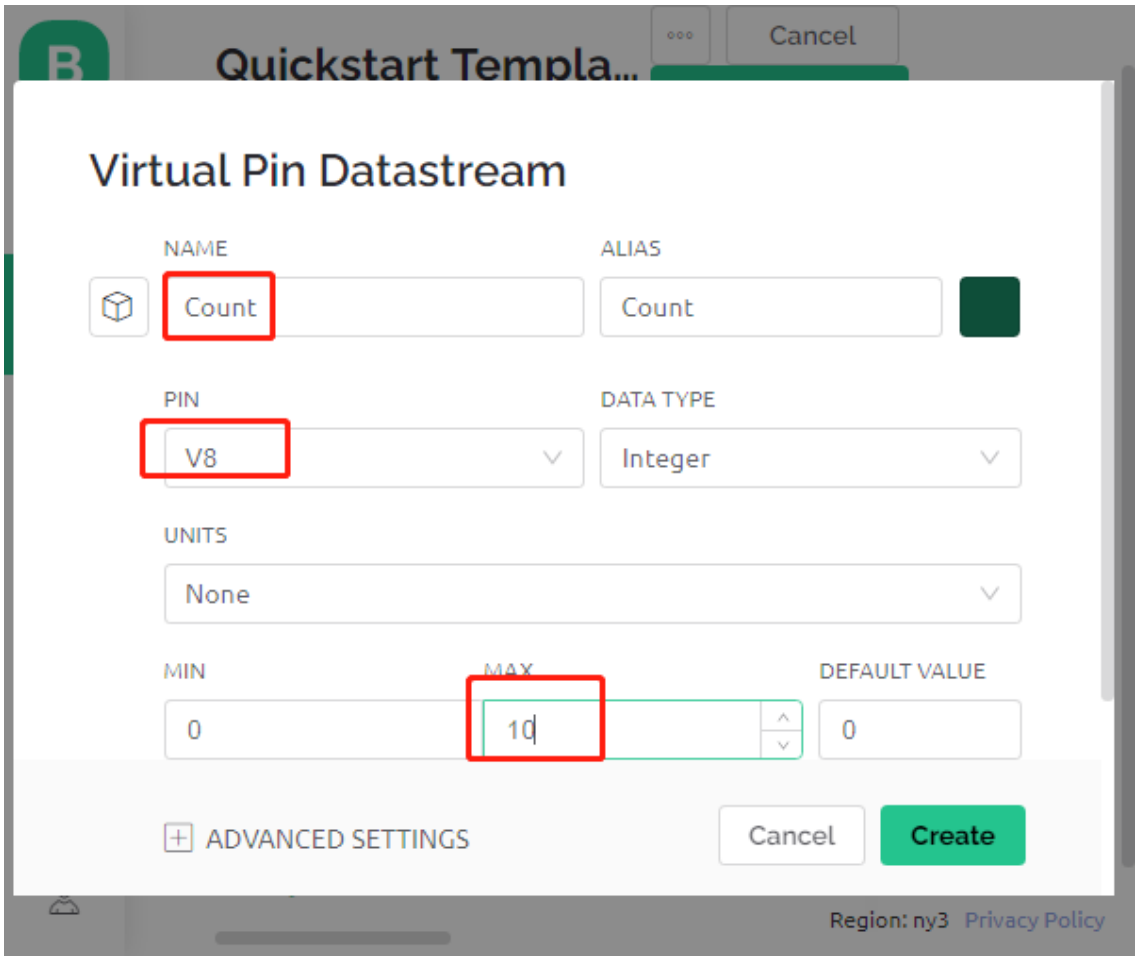
**1. 回路を組む**

**注釈:** ESP8266 モジュールは、安定した動作環境を提供するために高電流を必要としますので、9V のバッテリーが接続されていることを確認してください。

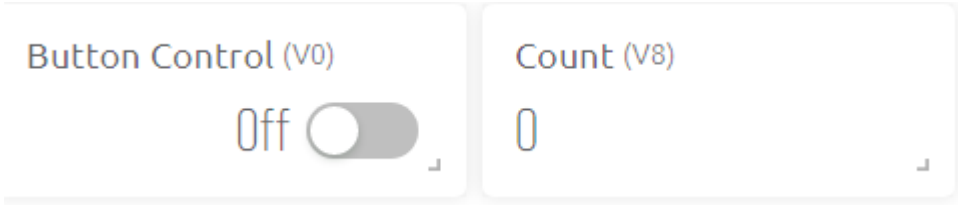



### 2. ダッシュボードを編集

1. 数を記録するために、**Datastream** ページで **Virtual Pin** タイプの **Datastream** を作成します。DATA TYPE を `Integer` に設定し、MIN と MAX を `0` と `10` に設定します。




2. 次に、 **Wed Dashboard** ページに移動し、 **Switch** ウィジェットをドラッグして、そのデータストリームを **V0** に設定し、 **Label** ウィジェットをドラッグして、そのデータストリームを **V8** に設定します。



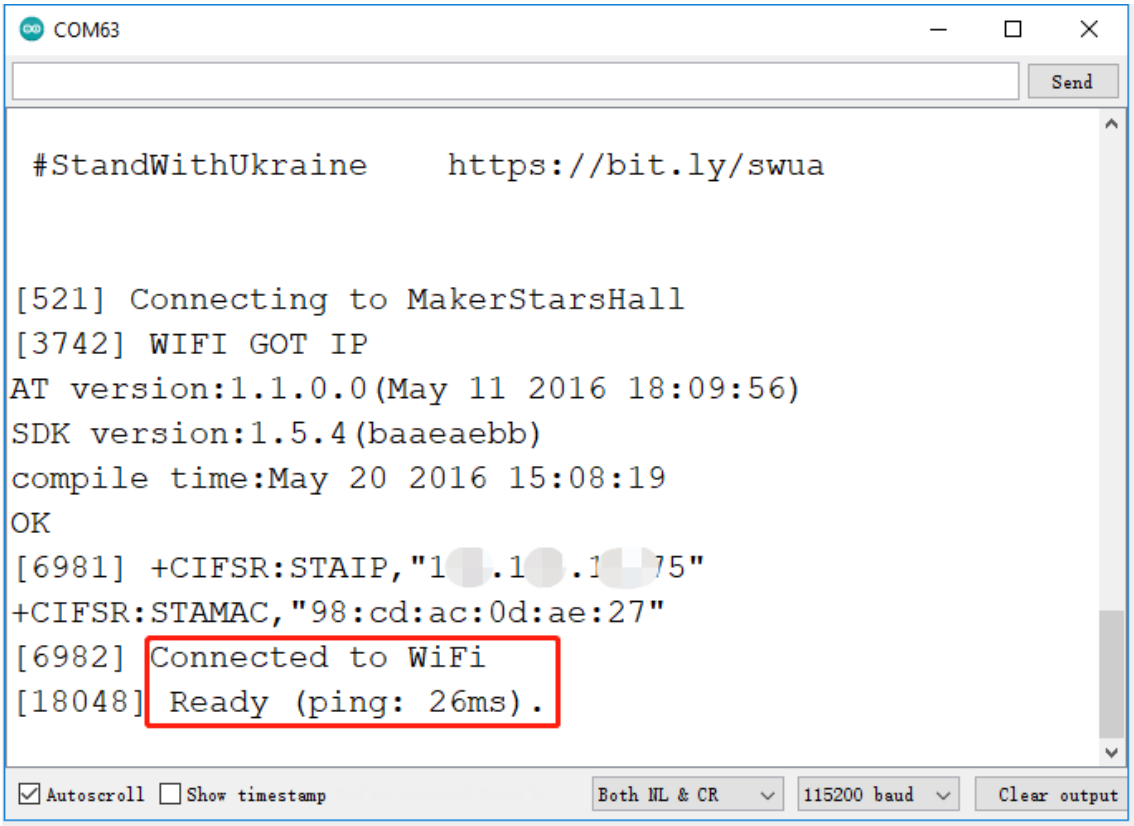

### 3. コードの実行

1. `3in1-kit\iot_project\7.current_limiting_gate` のパスの下の `7.current_limiting_gate.ino` ファイルを開くか、このコードを **Arduino IDE** にコピーします。

2. `Template ID` 、 `Device Name` 、および `Auth Token` を自分のものに置き換えます。また、使用している WiFi の `ssid` と `password` も入力する必要があります。詳細なチュートリアルは *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 正しいボードとポートを選択した後、 **Upoad** ボタンをクリックします。

4. シリアルモニターを開き（ボーレートを 115200 に設定）、成功した接続などのプロンプトが表示されるのを待ちます。




---

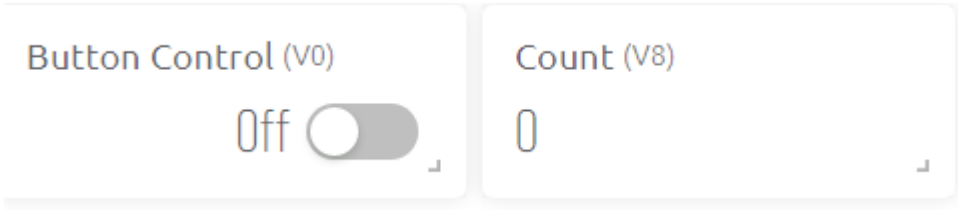
**注釈:** 接続時に `ESP is not responding` というメッセージが表示された場合、次の手順に従ってください。

- 9V のバッテリーが接続されていることを確認します。
- RST ピンを 1 秒間 GND に接続して ESP8266 モジュールをリセットし、その後、それを取り外します。
- R3 ボードのリセットボタンを押します。

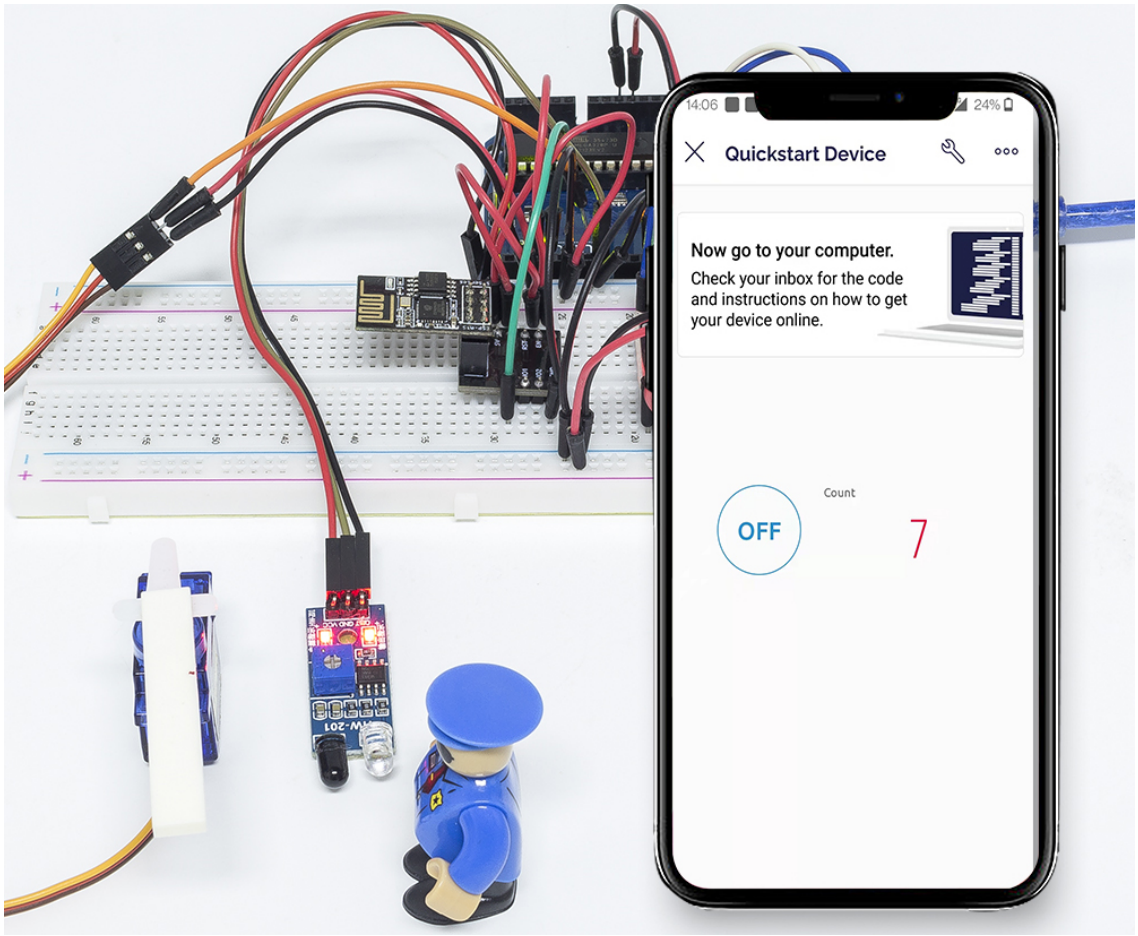
これらの操作を 3-5 回繰り返すことが必要な場合もありますので、お待ちください。

---

5. Blynk の Button Control ウィジェットをクリックして、スマートドアシステムを有効にします。IR 障害物回避モジュールが障害物を検出すると、ゲートが開き、Blynk の 7 セグメント表示および Count ウィジェットが 1 加算されます。




6. モバイルデバイスで Blynk を使用したい場合は、 **モバイルデバイスで *Blynk* を使用する方法は？** を参照してください。




**どのように動作するのか？**

`BLYNK_WRITE(V0)` 関数は **Switch** ウィジェットのステータスを取得し、それを変数 `doorFlag` に割り当てます。これにより、スマートゲートシステムが有効かどうかを判断するために使用されます。

```
BLYNK_WRITE(V0)
{
    doorFlag = param.asInt(); // ゲートを有効にする
}
```

Blynk タイマーでは、 `doorFlag` が毎秒判定され、有効になっている場合、ゲートのメイン関数が実行されます。

```
void myTimerEvent()
{
    if (doorFlag)
    {
        channelEntrance();
    }
}
```

ゲートのメイン関数は channelEntrance() です。オブジェクトがゲートに近づくと (センサーが障害物があることを検出すると)、 count が 1 増加します。Blynk Cloud の V8 データストリームと回路の 7 セグメント表示に count を書き込み、ドアを開けます。オブジェクトが存在から不在に移行する場合、つまりオブジェクトがドアに入った場合、ドアを閉じます。

```
void channelEntrance()
{
    int currentState = digitalRead(irPin); // 0: 障害物 1: 障害物なし
    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;
}
```

showNumber(int num) 関数は、7 セグメント表示に値を表示するために使用されます。

```
void showNumber(int num)
{
    digitalWrite(STcp, LOW); //ST_CP を接地して、伝送中は常に低く保つ
    shiftOut(DS, SHcp, MSBFIRST, datArray[num]);
    digitalWrite(STcp, HIGH); //データを保存するために ST_CPST_CP を引き上げる
}
```

operateGate(bool openGate) 関数は、参照が True の場合、ドアをゆっくり開け、参照が False の場合、ドアをゆっくり閉じます。

```
void operateGate(bool openGate) {
```

(次のページに続く)

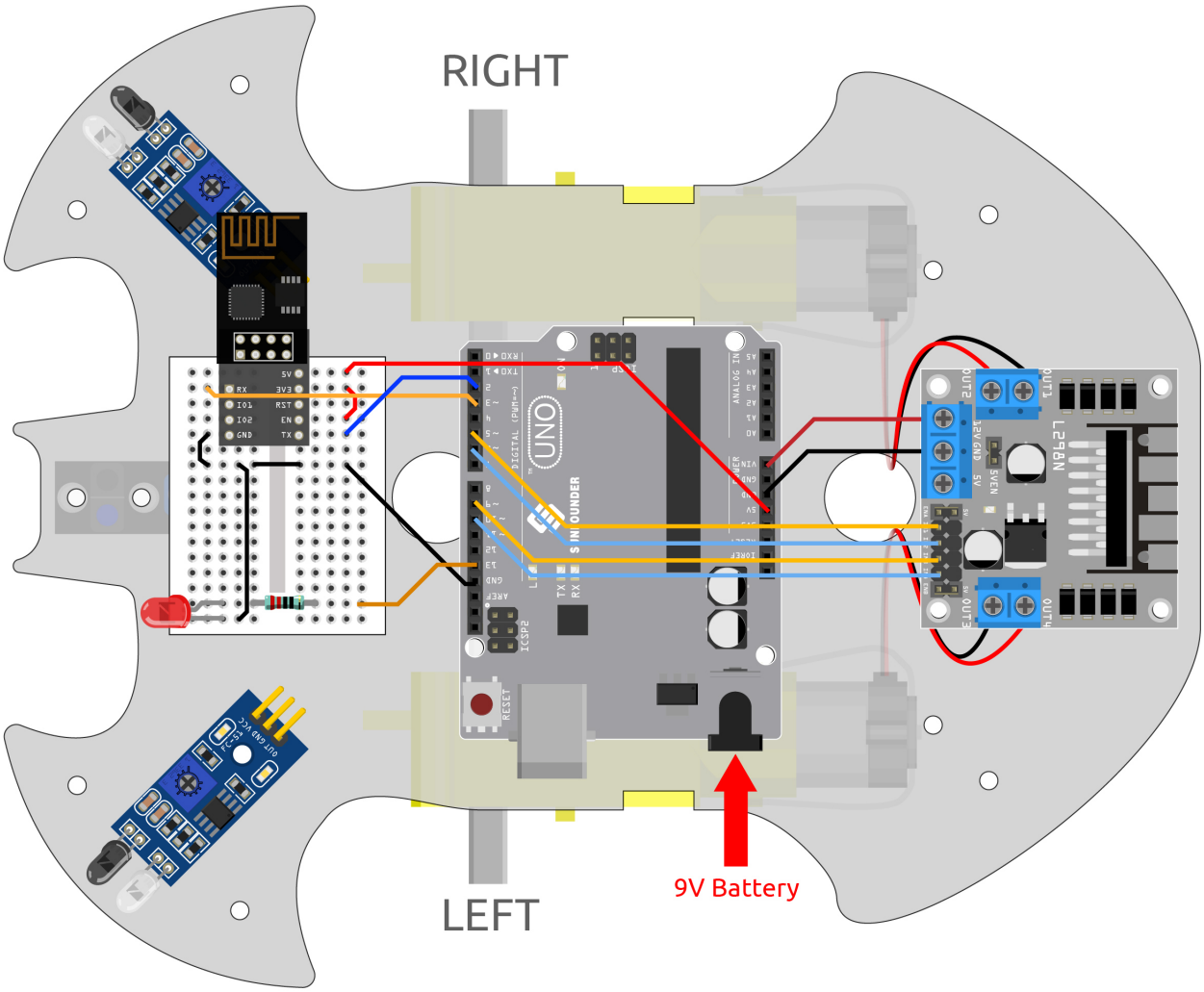
(前のページからの続き)

```
    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 カー

このプロジェクトでは、スマホの Blynk アプリを使用して車を制御しました。しかし、車を組み立て、基本的な理解を得るためには **カープロジェクト** を参照する必要があります。5G ネットワークが普及する時代に、このモードは多くの産業での主要な生産方法の一つになるかもしれません。先取りしてこの遊びを体験しましょう。

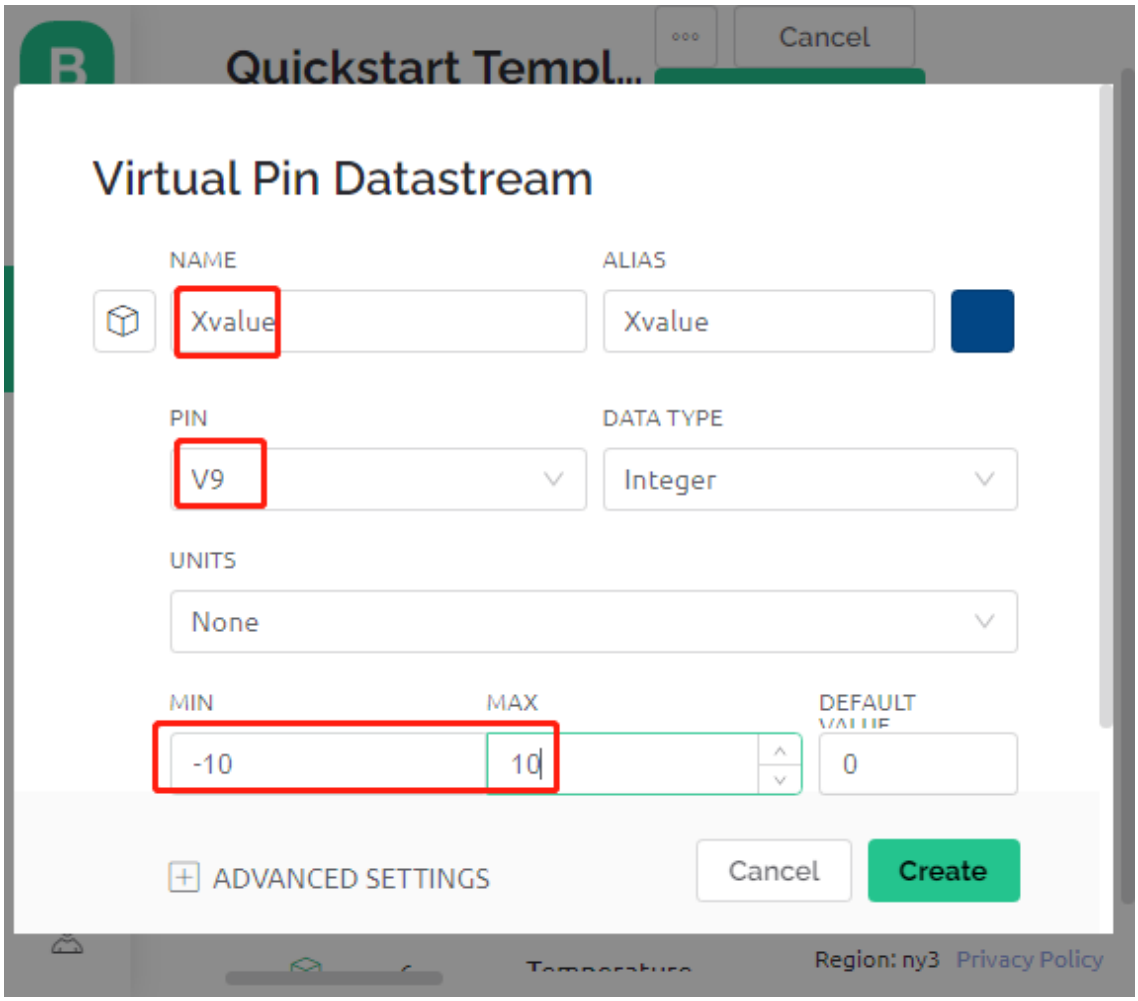
**1. 回路を組む**



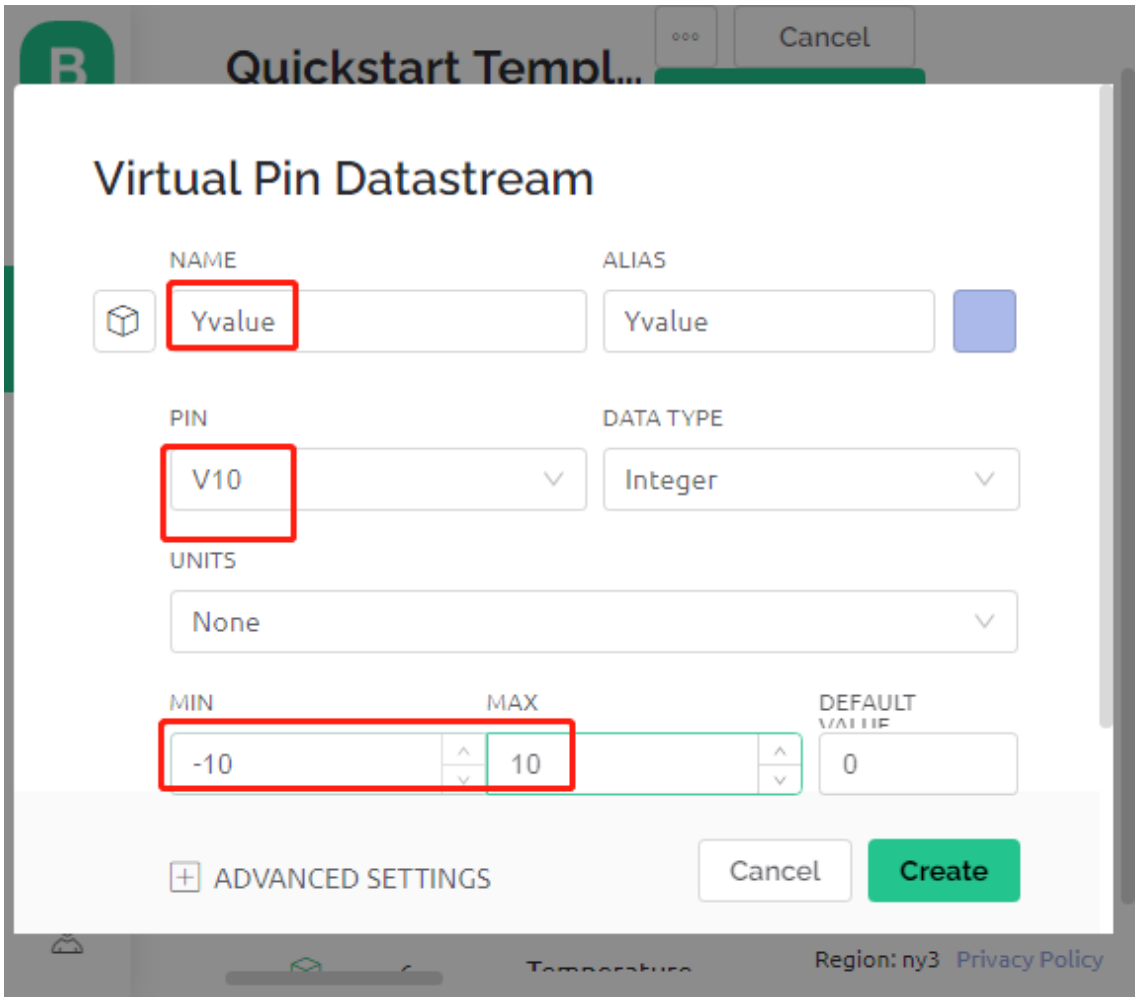

### 2. ダッシュボードを編集

携帯の Blynk ではデータストリームを編集できないため、これらの手順は Web 側で行う必要があります。

1. **Datastream** ページで、ジョイスティックの X 軸の値を記録するための **Virtual Pin** タイプの **Datastream** を作成します。名前は `Xvalue` 、データタイプは `Integer` 、最小値と最大値は `-10` と `10` に設定します。

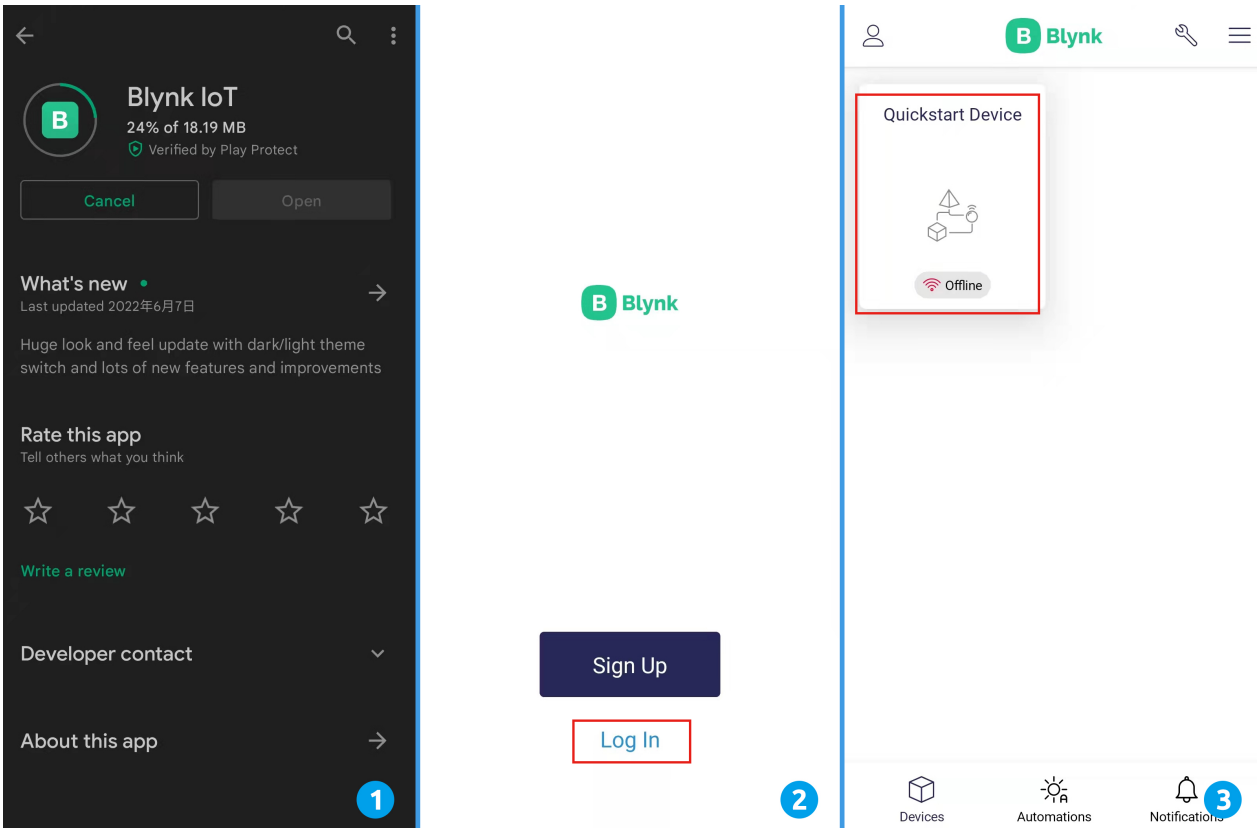



2. ジョイスティックの Y 軸値を記録するために、 **Virtual Pin** 型の **Datastream** を作成します。NAME を `Yvalue` 、DATA TYPE を `Integer` 、MIN と MAX を `-10` と `10` に設定する。




次に、携帯電話で以下の操作を行う必要があります。

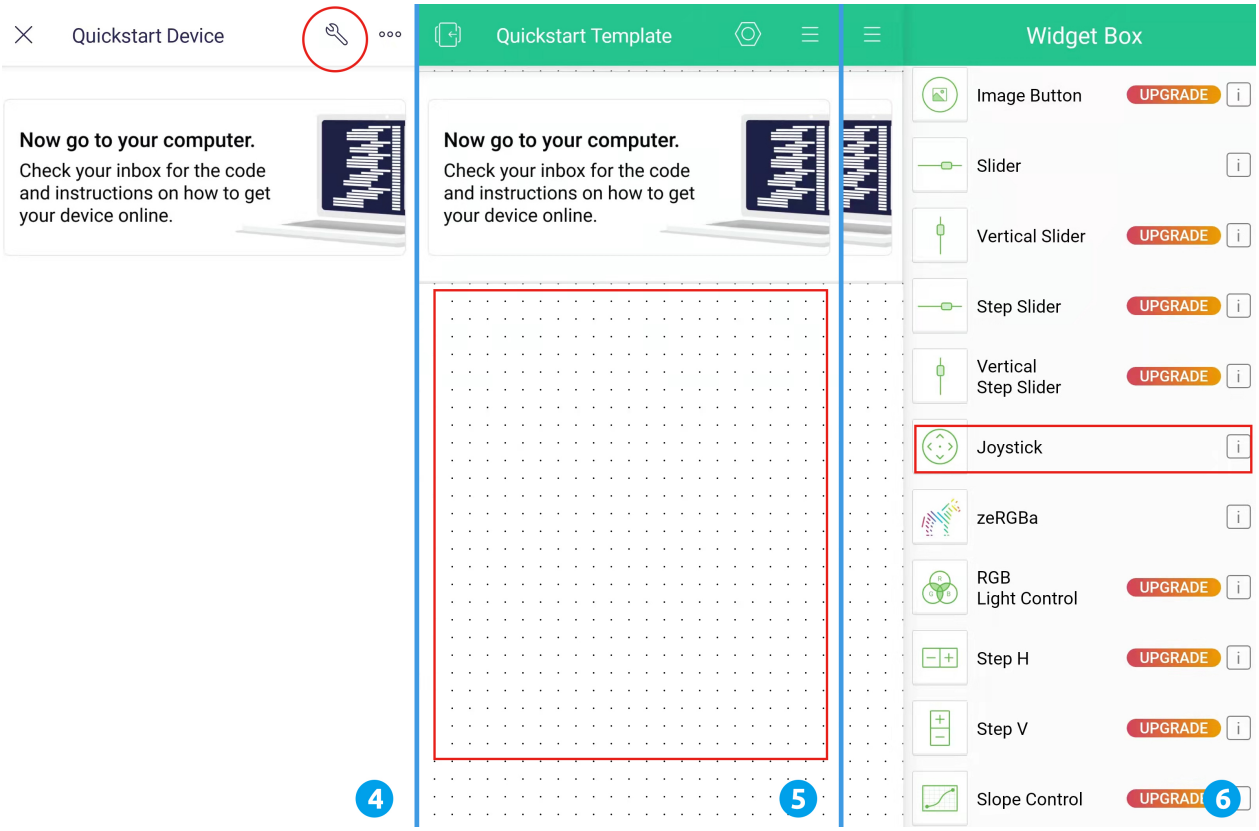
1. GOOGLE Play または APP Store で "Blynk IoT" (Blynk(legacy) ではない) を検索してダウンロードします。
2. アプリを開いた後、ログインします。このアカウントは、Web クライアントで使用されたものと同じである必要があります。
3. 次に、ダッシュボードに移動します (持っていない場合は作成します)。ここで、モバイル用と Web 用のダッシュボードが互いに独立していることがわかります。



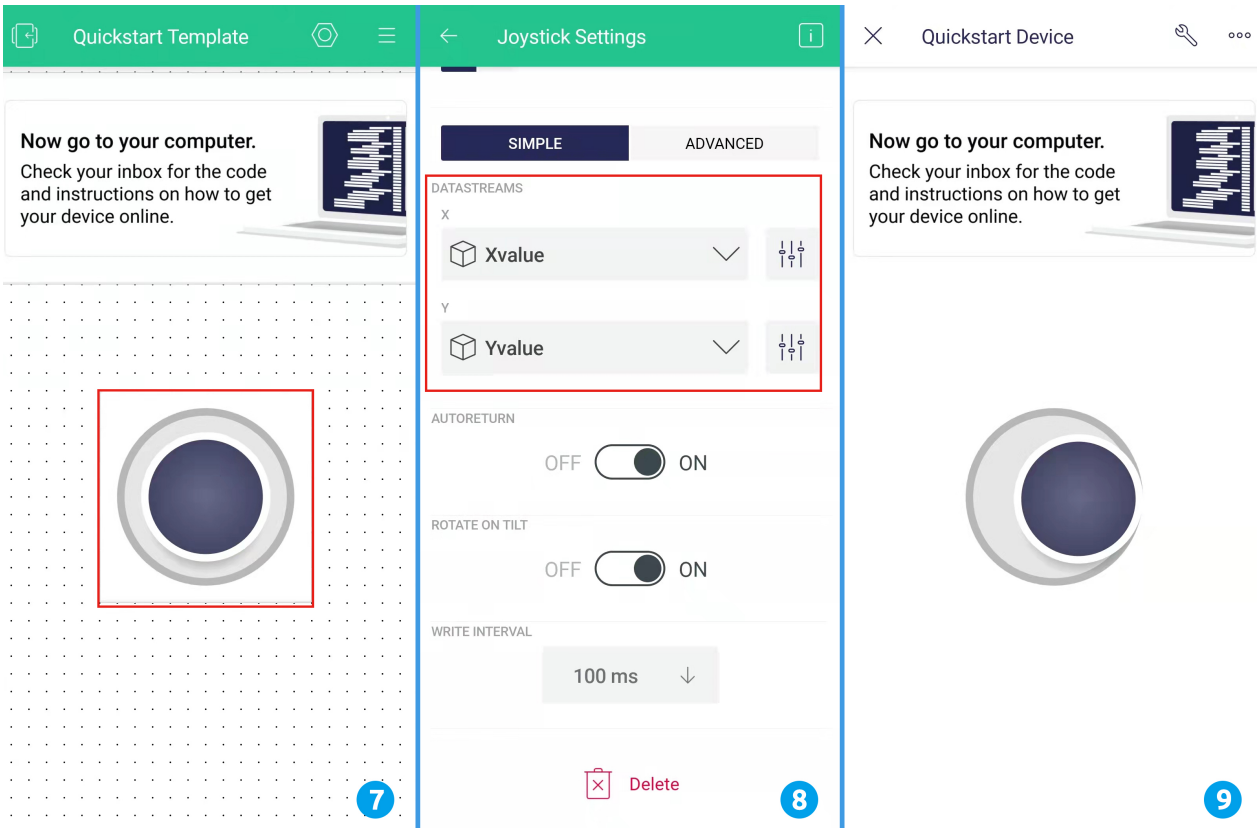

4. 編集アイコンをクリックします。

5. 空白のエリアをクリックします。

6. ジョイスティックウィジェットを選択します。




7. 空白のエリアにジョイスティックウィジェットが表示されますので、それをクリックします。
8. ジョイスティックの設定が表示されますので、データストリームで先ほど設定した Xvalue と Yvalue を選択します。
9. ダッシュボードページに戻り、ジョイスティックを操作することができます。



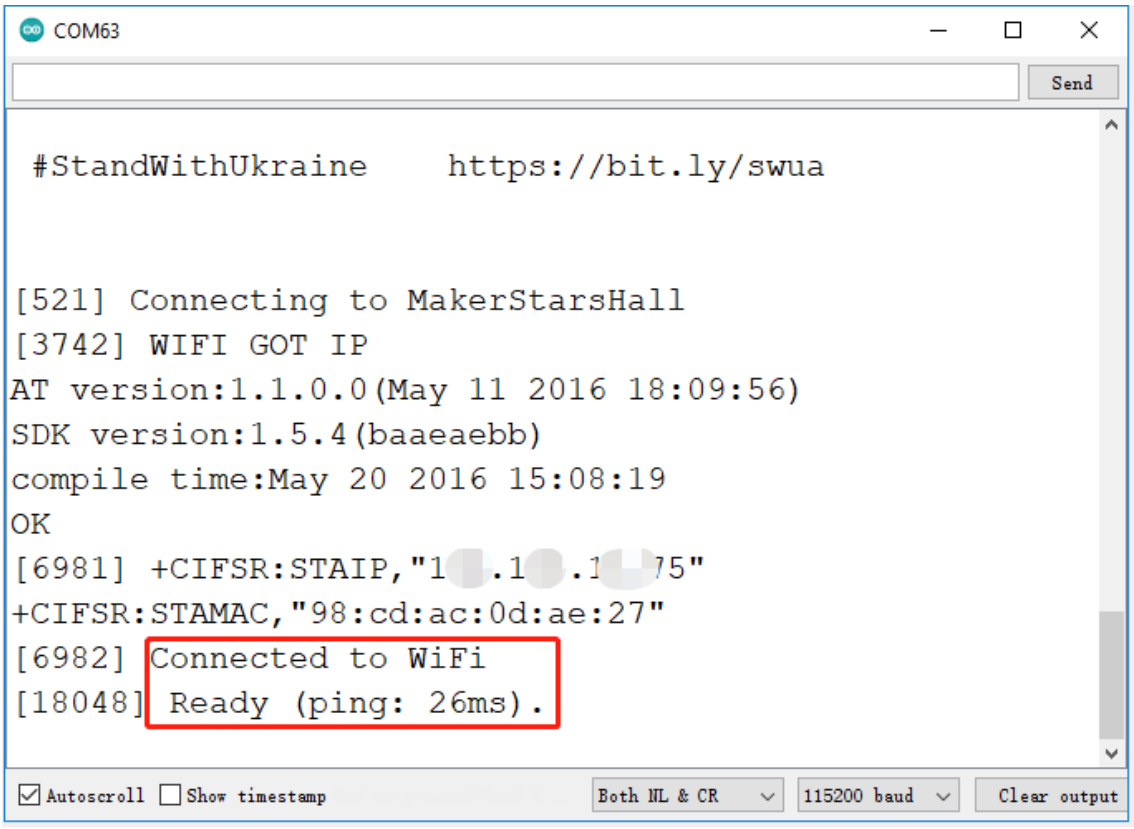

### 3. コードの実行

1. パス `3in1-kit\iot_project\8.iot_car` の下にある `8.iot_car.ino` ファイルを開くか、このコードを **Arduino IDE** にコピーします。

2. `Template ID`、 `Device Name`、および `Auth Token` を自分のものに置き換えます。また、使用している WiFi の `ssid` および `password` を入力する必要があります。詳しいチュートリアルは、 *1.4 R3* **ボードを** *Blynk* **に接続** を参照してください。

3. 正しいボードとポートを選択した後、 **Upoad** ボタンをクリックします。

4. シリアルモニターを開き (ボーレートを 115200 に設定)、接続成功のようなプロンプトが表示されるのを待ちます。




---

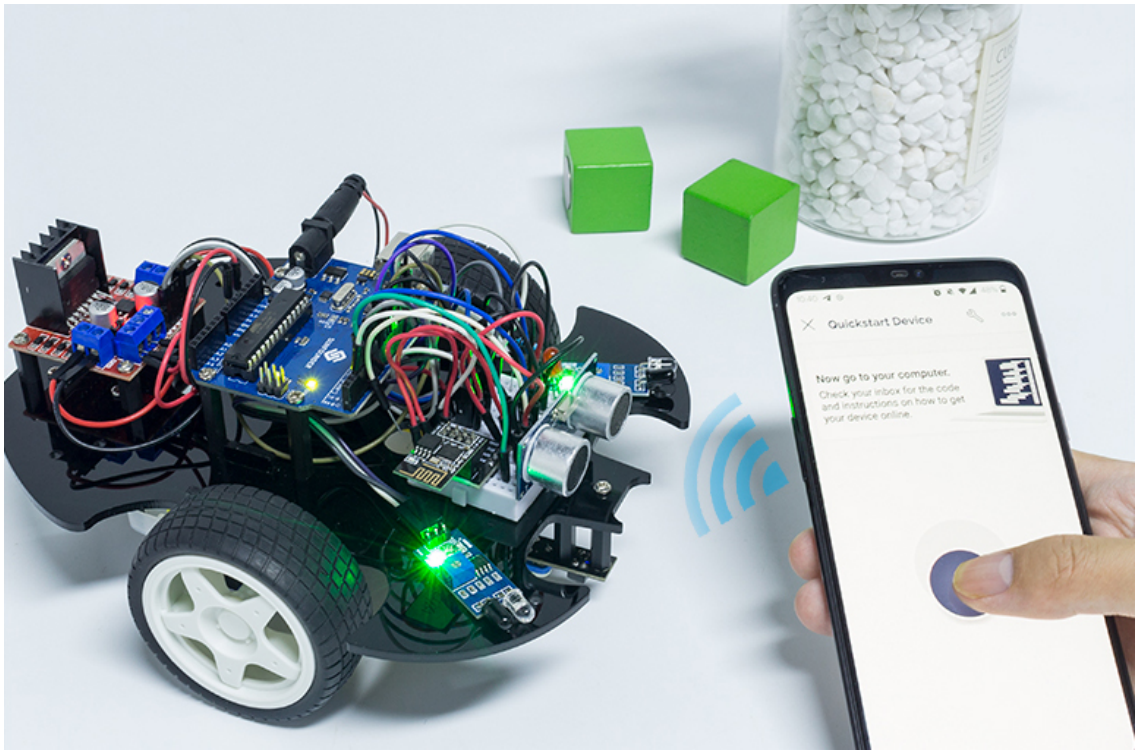
**注釈:** 接続時に `ESP is not responding` というメッセージが表示された場合は、以下の手順に従ってください。

- 9V のバッテリーが接続されていることを確認します。
- ESP8266 モジュールの RST ピンを 1 秒間 GND に接続してリセットし、その後、取り外します。
- R3 ボードのリセットボタンを押します。

これらの操作を 3-5 回繰り返す必要があることもありますので、お待ちください。

---

5. USB ケーブルを抜いて、9V のバッテリーだけでカートに電力を供給します。LED が点灯すると、車が Blynk に接続されていることを示しています。
6. 携帯電話の Blynk を開き、ジョイスティックウィジェットを使用して車の動きを制御します。




**どのように動作するか?**

これらの関数は、車の動きを制御するために使用されます。

```
void moveForward(int speed) {...}
void moveBackward(int speed) {...}
void turnRight(int speed) {...}
void turnLeft(int speed) {...}
void stopMove() {...}
```

IoT セクションでは、ジョイスティックウィジェットの値を読み取り、 Xvalue と Yvalue の変数に代入します。

```
int Xvalue = 0;
int Yvalue = 0;

BLYNK_WRITE(V9)
{
    Xvalue = param.asInt();
}

BLYNK_WRITE(V10)
{
    Yvalue = param.asInt();
```

(次のページに続く)

(前のページからの続き)

```
}
```

loop() の中で、 Xvalue と Yvalue に基づいて車が異なるアクションを実行するようにします。

```
if (Yvalue >= 5) {
    moveForward(255);
} else if (Yvalue <= -5) {
    moveBackward(255);
} else if (Xvalue >= 5) {
    turnRight(150);
} else if (Xvalue <= -5) {
    turnLeft(150);
} else {
    stopMove();
}
```

また、 loop() に Blynk Cloud に接続されている場合に LED を点灯するネットワークステータスの判定を追加します。

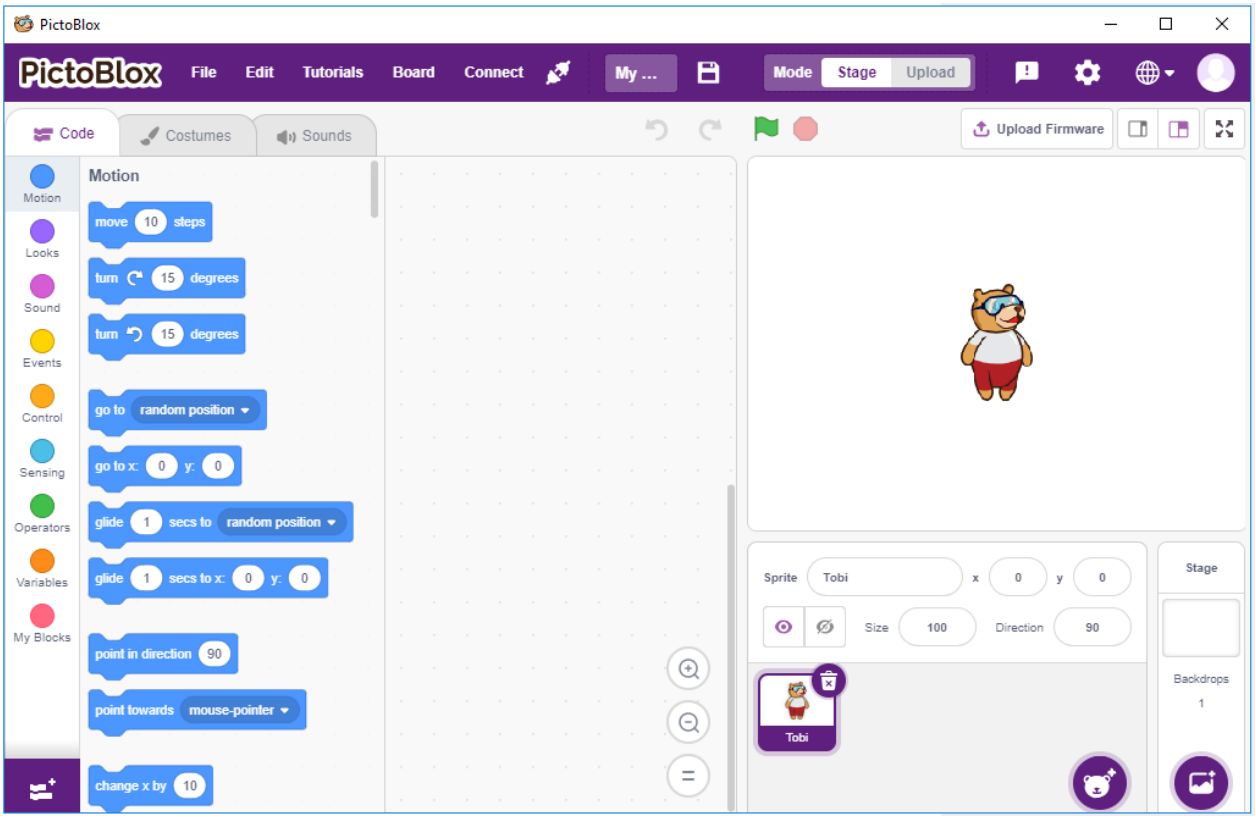
```
if (!Blynk.connected()) {
    digitalWrite(ledPin, LOW);
    Serial.print("offline!");
    bool result = Blynk.connect();
    Serial.println(result);
} else {
    digitalWrite(ledPin, HIGH);
}
```

# 第 7 章

# Scratch で遊ぼう

Arduino IDE でのプログラミングだけでなく、グラフィカルなプログラミングも利用できます。

ここでは Scratch を使用したプログラミングを推奨していますが、現在の公式 Scratch は Raspberry Pi とのみ互換性があります。そこで、STEMPedia という企業と提携し、Arduino ボード（Uno, Mega2560, Nano）用の Scratch 3 ベースのグラフィカルプログラミングソフトウェア「PictoBlox」を開発しました。




これは Scratch 3 の基本機能を保持しながら、Arduino Uno、Mega、Nano、ESP32、Microbit、および STEMPedia のホームメイドのメインボードなどの制御ボードを追加しています。これにより、外部センサーやロボットを使用してステージ上のスプライトを制御することができ、強力なハードウェア対話機能が備わっています。

さらに、AI や機械学習の機能もあります。プログラミングの基礎があまりなくても、これらの人気の高いハイテク技術を学び、使用することができます。

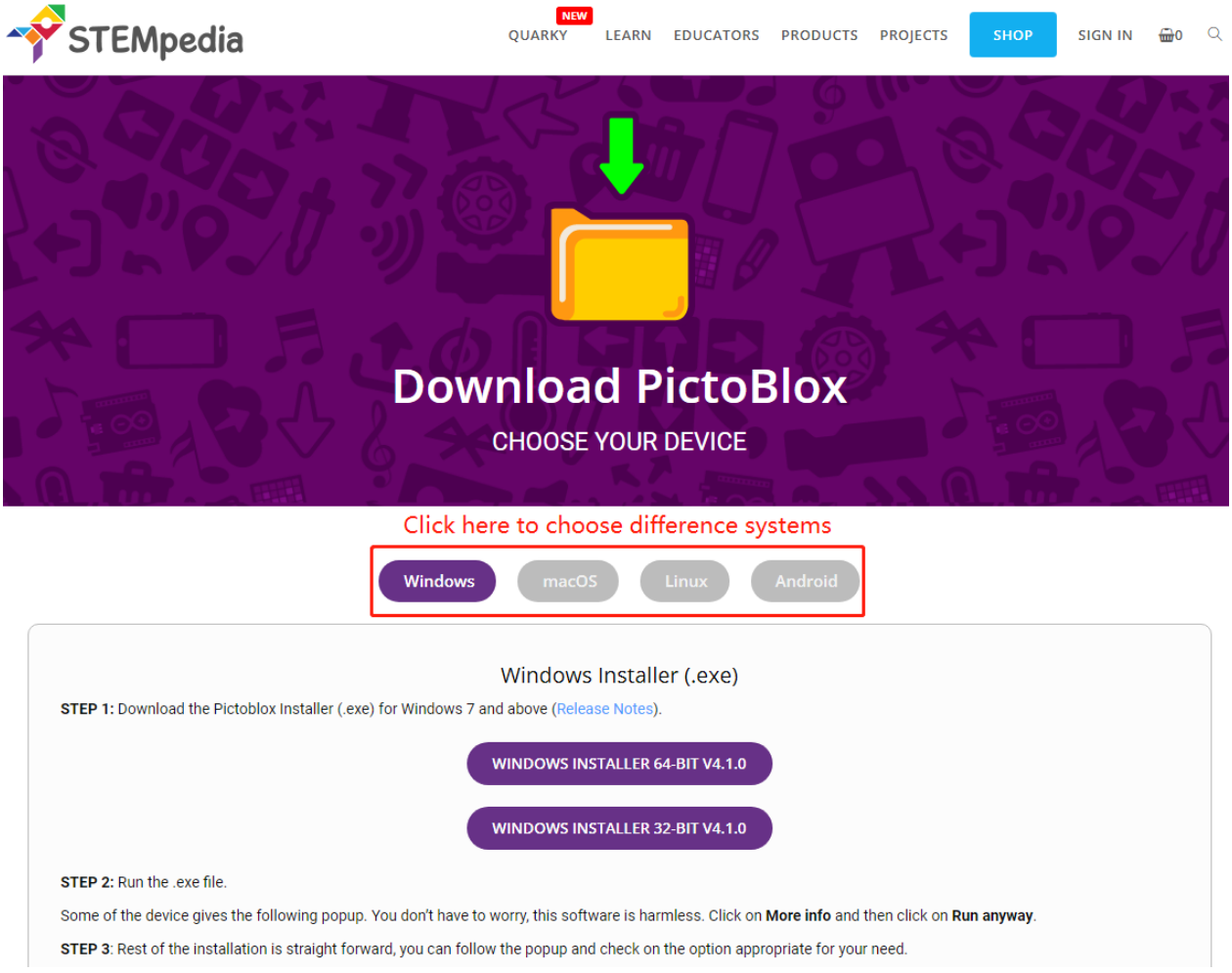
Scratch のコーディングブロックをドラッグアンドドロップするだけで、クールなゲームやアニメーション、インタラクティブなプロジェクト、そしてロボットを自分の思い通りに制御することができます！

さあ、この発見の旅を始めましょう！

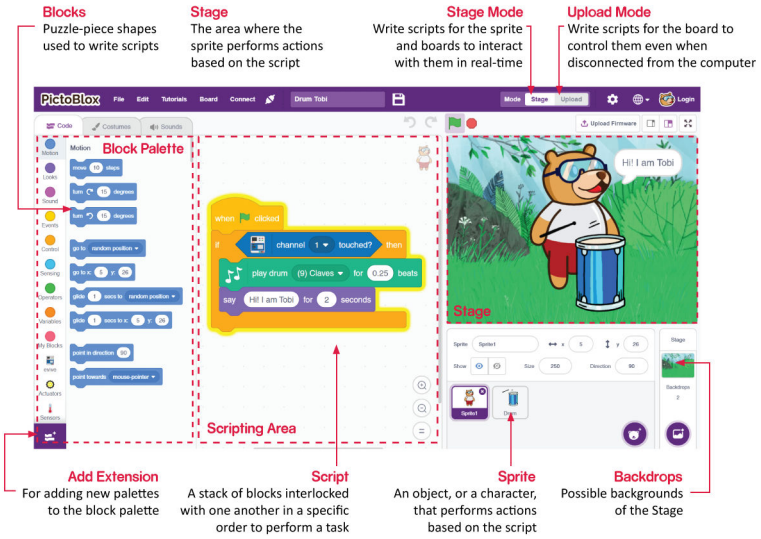
**1. はじめに**

## 7.1 1.1 PictoBlox のインストール

このリンクをクリックしてください：https://thestempedia.com/product/pictoblox/download-pictoblox/。適切なオペレーティングシステム（Windows、macOS、Linux）を選択し、インストール手順に従ってください。



## 7.2 1.2 インターフェースの紹介




### スプライト

スプライトは、プロジェクトでさまざまなアクションを実行するオブジェクトやキャラクターです。それは与えられたコマンドを理解し、従います。各スプライトには、カスタマイズもできる特定のコスチュームと音があります。

### ステージ

ステージは、あなたのプログラムに従ってバックドロップでスプライトがアクションを実行する領域です。

### バックドロップ

バックドロップはステージを装飾するために使用されます。PictoBlox からバックドロップを選択する、自分で描く、またはコンピュータから画像をアップロードすることができます。

### スクリプトエリア

スクリプトは PictoBlox/Scratch の言葉でのプログラムやコードです。特定のタスクや一連のタスクを実行するために特定の順序で配置された「ブロック」のセットです。複数のスクリプトを書くことができ、すべて同時に実行できます。スクリプトは画面の中央のスクリプトエリアでのみ書くことができます。

### ブロック

ブロックは、スクリプトエリアで単純に積み重ねることでプログラムを書くために使用されるパズルのようなものです。ブロックを使用してコードを書くことは、プログラミングを容易にし、エラーの確率を減少させることができます。

### ブロックパレット

ブロックパレットは左側のエリアに位置しており、モーション、サウンド、コントロールなどの機能によって名前が付けられています。各パレットには異なるブロックがあり、たとえば、モーションパレットのブロックはスプラ

イトの動きを制御し、コントロールパレットのブロックは特定の条件に基づいてスクリプトの動作を制御します。

**Add Extension** ボタンからロードできる他の種類のブロックパレットもあります。

**モード**

Scratch とは異なり、PictoBlox には 2 つのモードがあります：

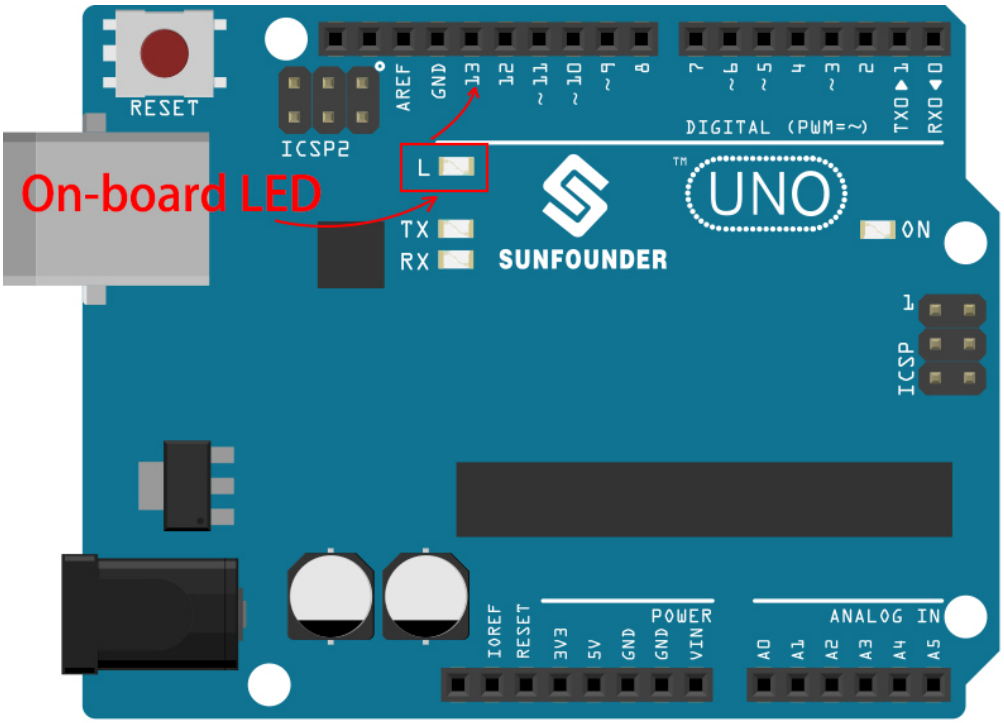
- **ステージモード**: このモードでは、スプライトとボードのスクリプトを書き、リアルタイムでスプライトと対話できます。Pictoblox とのボードの接続を切断すると、もう対話することはできません。
- **アップロードモード**: このモードでは、スクリプトを書き、ボードにアップロードして、コンピュータに接続されていないときでも使用できるようになります。たとえば、移動するロボットのためのスクリプトをアップロードする必要があります。

詳細については、https://thestempedia.com/tutorials/getting-started-pictoblox を参照してください。

## 7.3 1.3 PictoBlox のクイックガイド

PictoBlox の 2 つのモードでの使い方を学びましょう。

また、R3 ボードの Pin 13 に接続されたオンボード LED がありますので、2 つの異なるモードでこの LED を点滅させる方法を学びます。

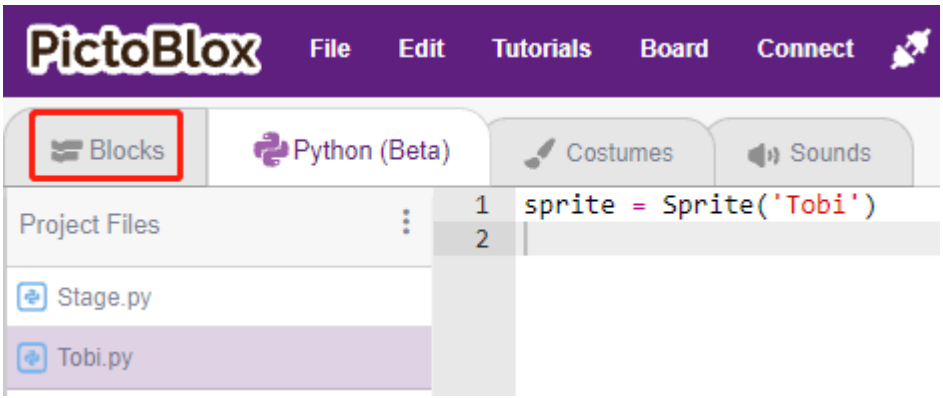


### 7.3.1 ステージモード

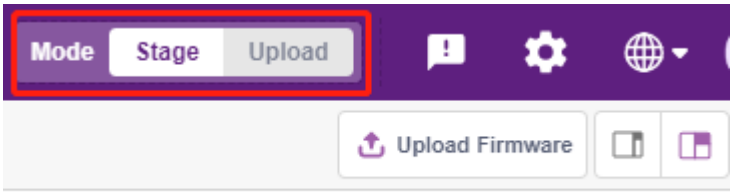
**1. Arduino ボードを接続**

USB ケーブルで Arduino ボードをコンピュータに接続します。通常、コンピュータは自動的にボードを認識し、最終的に COM ポートを割り当てます。

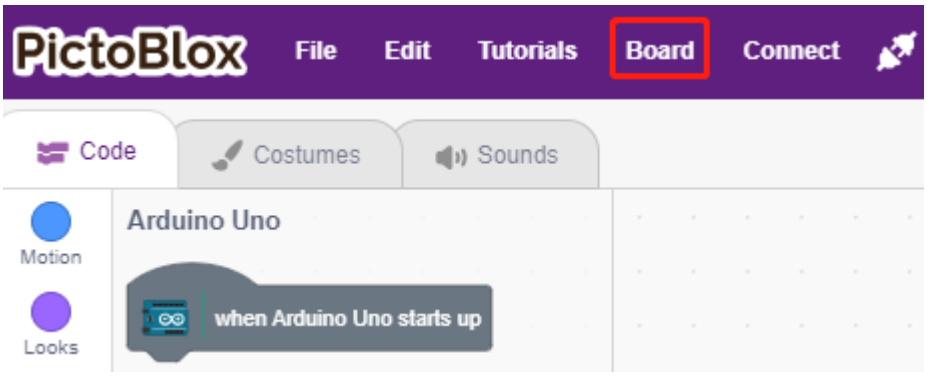
PictoBlox を開くと、Python のプログラミングインターフェースがデフォルトで開きます。ブロックインターフェースに切り替える必要があります。



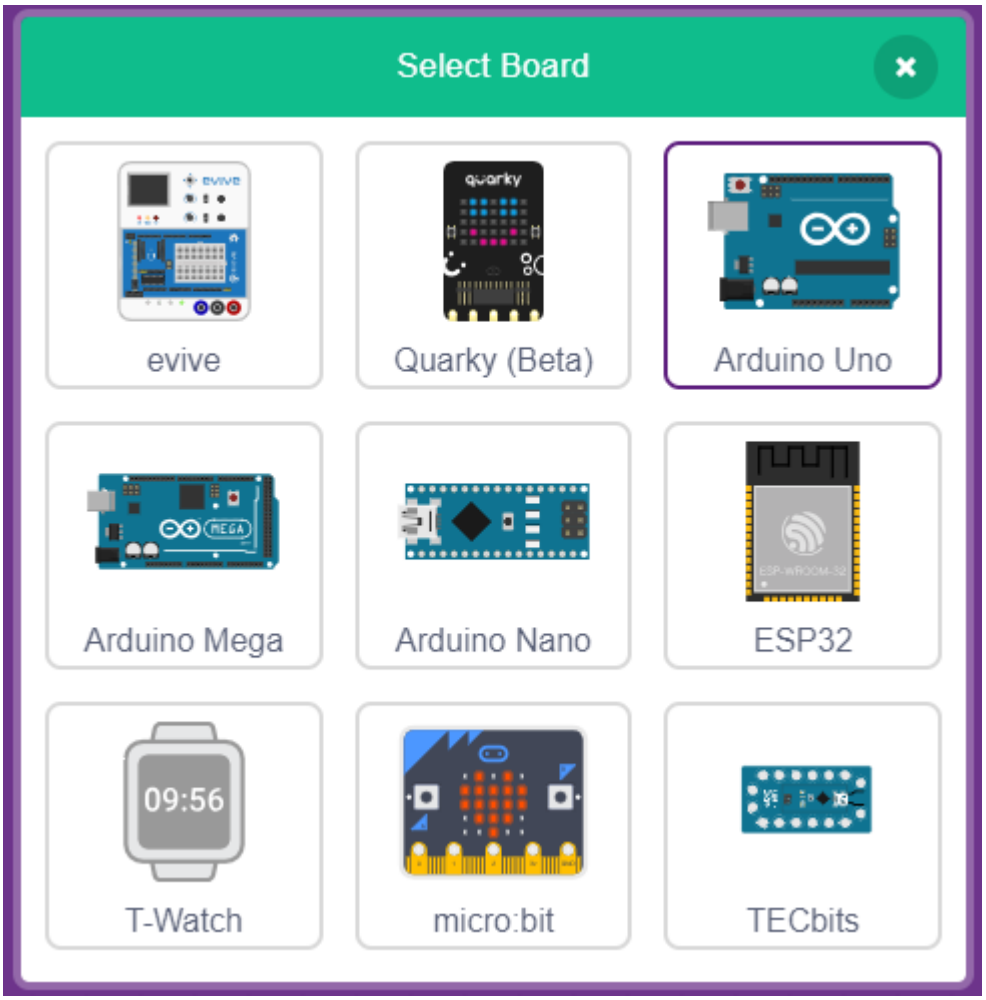

次に、モード切替のための右上隅が表示されます。デフォルトはステージモードで、Tobi がステージに立っています。



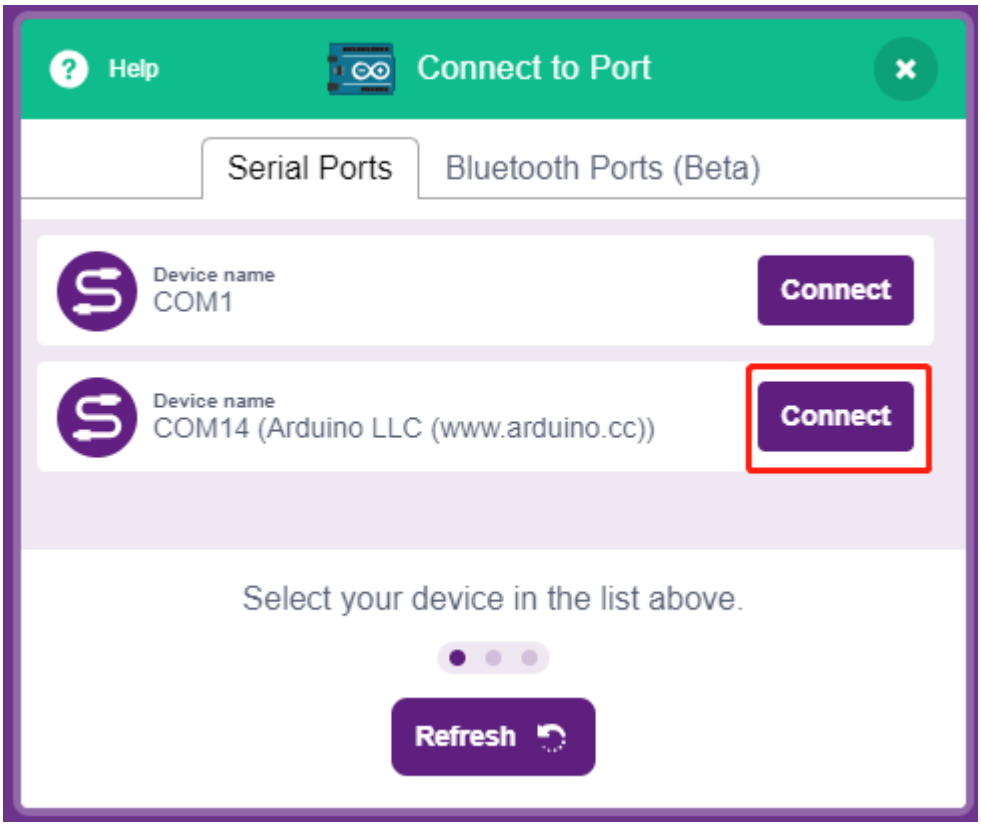

上部のナビゲーションバーで **Board** をクリックしてボードを選択します。



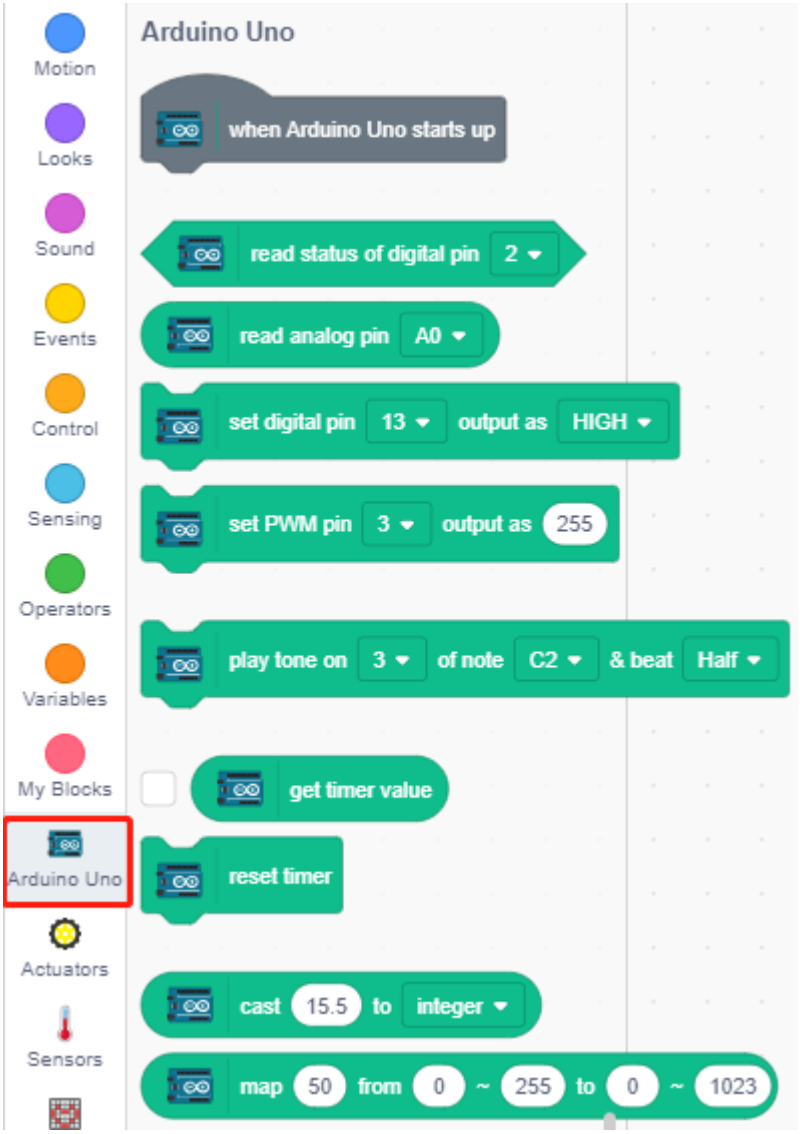

例として、 **Arduino Uno** を選びます。




次に、接続するポートを選択するための接続ウィンドウが表示されます。接続が完了するとホームページに戻ります。使用中に接続が切れた場合は、 **Connect** をクリックして再接続することもできます。




同時に、 **Block Palette** に Arduino Uno 関連のパレット、例えば Arduino Uno や Actuators などが表示されます。




### 2. ファームウェアのアップロード

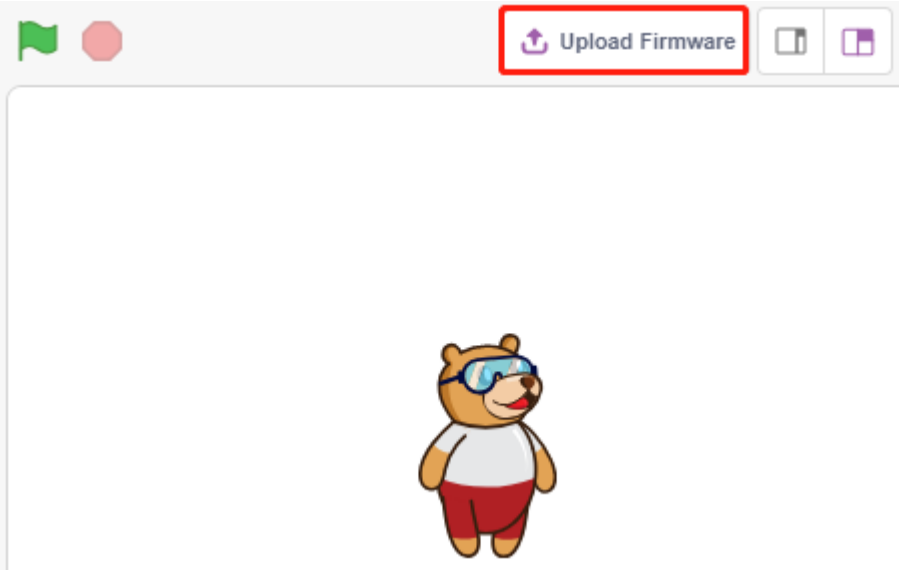
ステージモードで作業を行うためには、ボードにファームウェアをアップロードする必要があります。これによりボードとコンピュータ間のリアルタイム通信が可能となります。ファームウェアのアップロードは一度だけのプロセスです。Upload Firmware ボタンをクリックして行います。

少し待つと、アップロード成功のメッセージが表示されます。

---

**注釈:** この Arduino ボードを PictoBlox で初めて使用する場合、またはこの Arduino が以前 Arduino IDE でアップロードされていた場合は、使用する前に **Upload Firmware** をタップする必要があります。

---

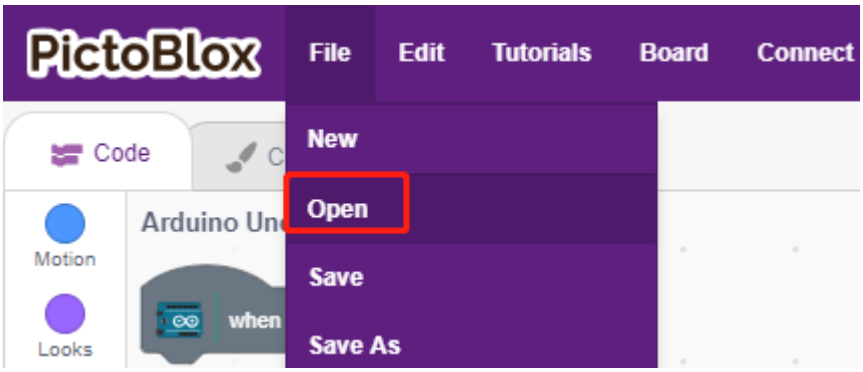



### 3. プログラミング

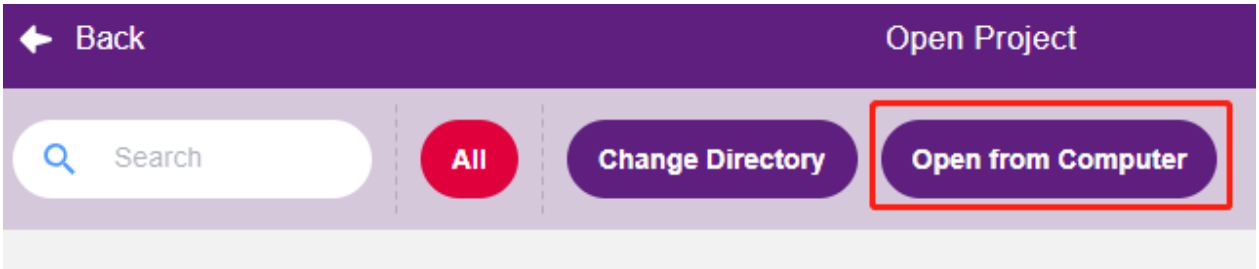
- スクリプトを直接開くと実行

もちろん、スクリプトを直接開いて実行することもできますが、まず github からダウンロードしてください。

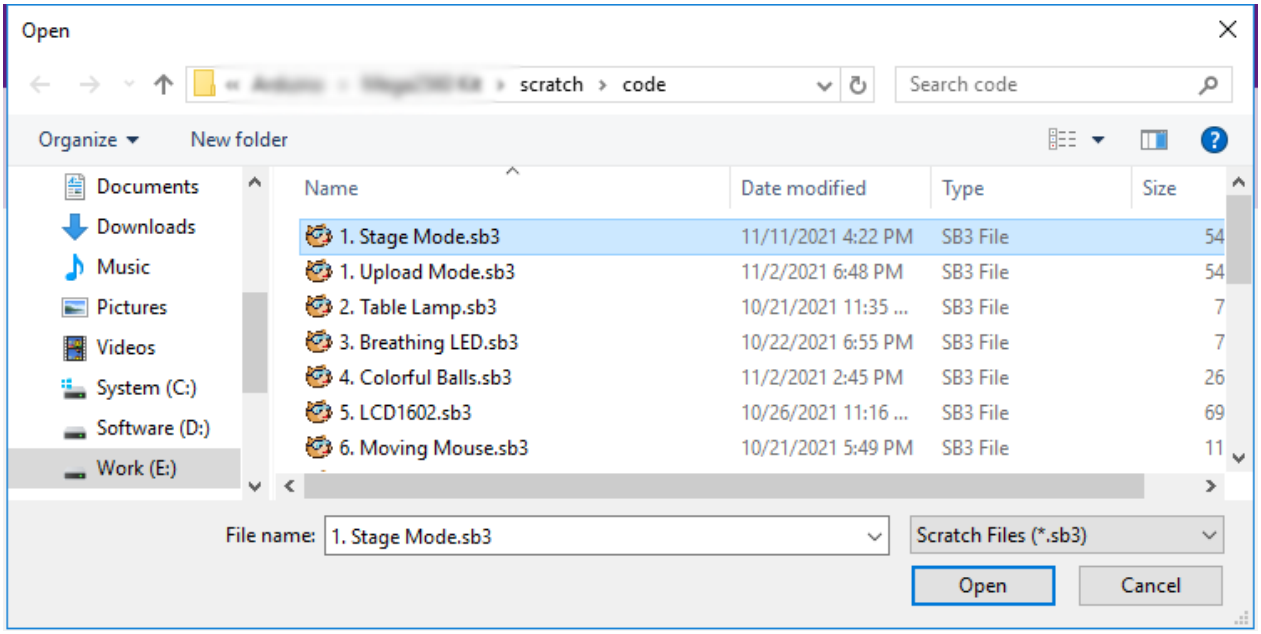
右上隅の **File** をクリックし、 **Open** を選択します。



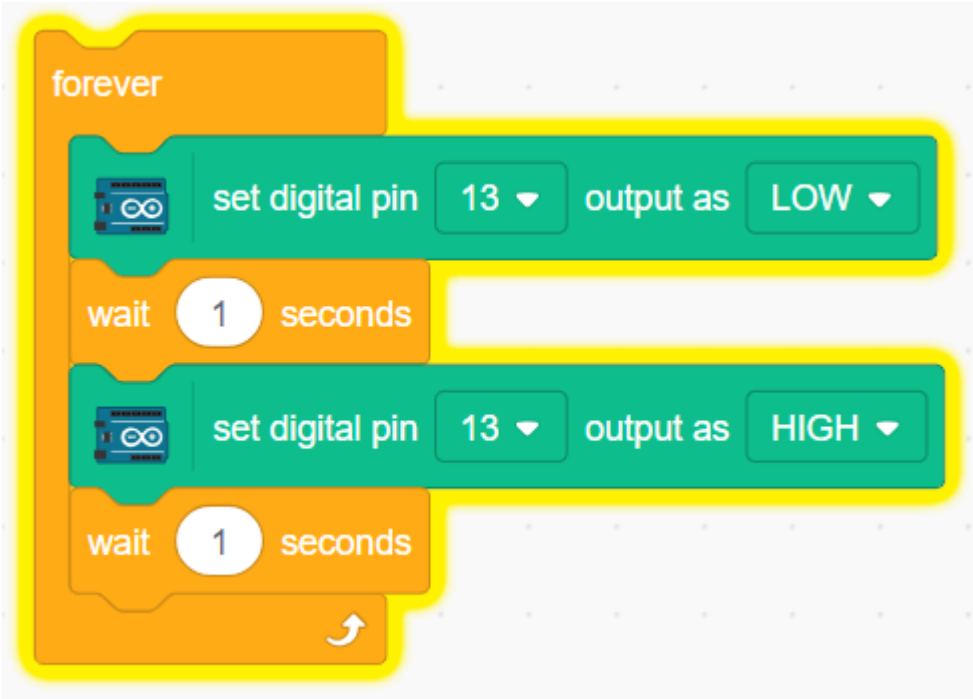

**Open from Computer** を選択します。




`3in1-kit\scratch_project\code` のパスに移動して、 **1. Stage Mode.sb3** を開きます。必要なコードを github からダウンロードしたことを確認してください。



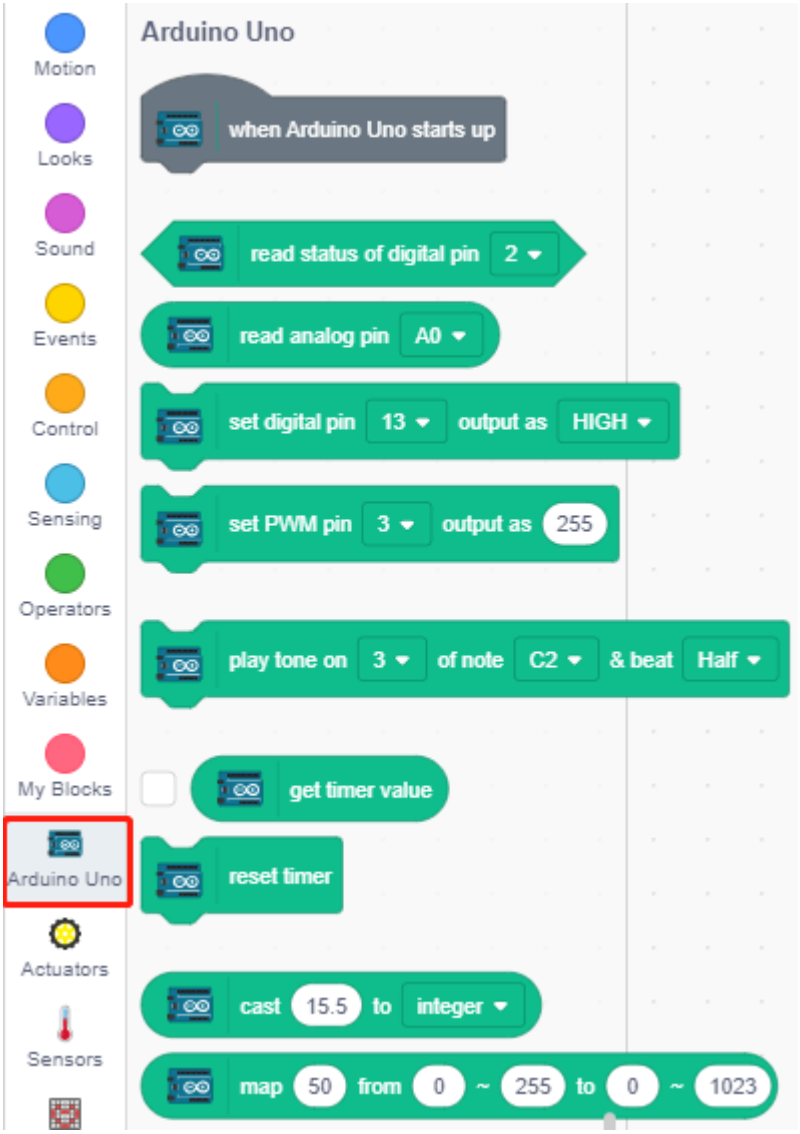

スクリプトを直接クリックして実行します。いくつかのプロジェクトは緑の旗をクリックするか、スプライトをクリックします。




- ステップバイステップのプログラム

これらの手順に従って、スクリプトをステップバイステップで書くこともできます。

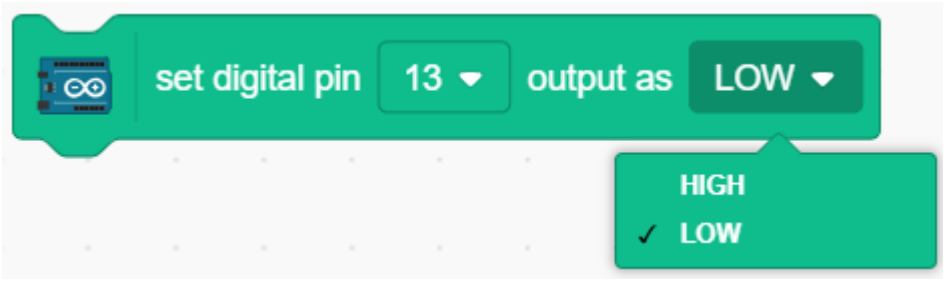
**Arduino Uno** のパレットをクリックします。




Arduino ボード上の LED は、デジタルピン 13 で制御されます（2 つの状態、HIGH または LOW のみ）。そのため、[set digital pin out as] ブロックをスクリプトエリアにドラッグします。

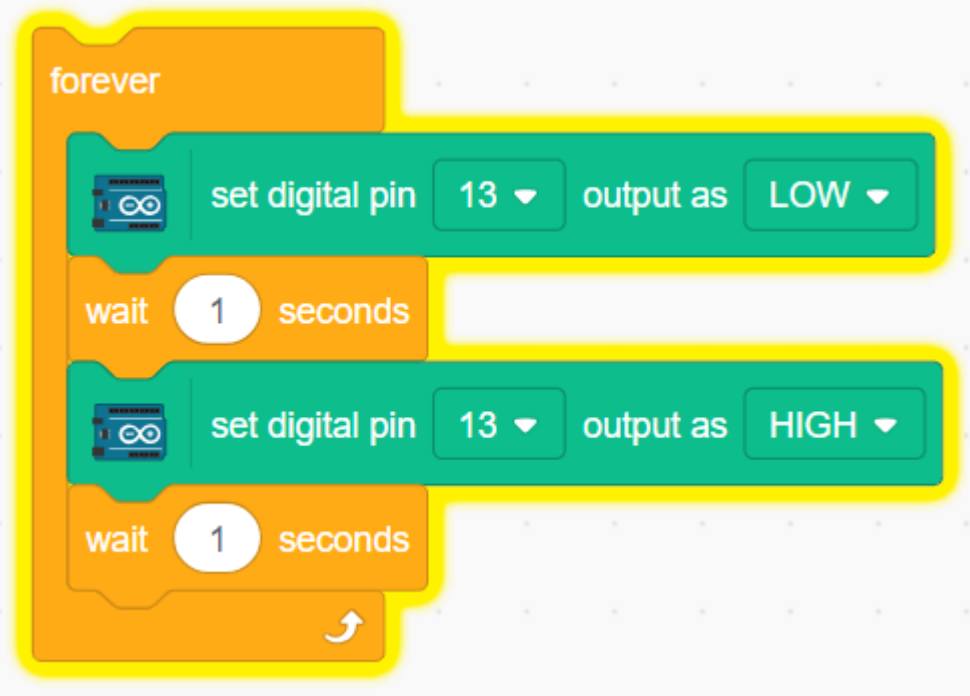
LED のデフォルトの状態は点灯しているため、ピン 13 を LOW に設定して、このブロックをクリックすると LED が消灯するのを確認できます。

- [set digital pin out as]: デジタルピン (2〜13) を (HIGH/LOW) レベルに設定する。



連続して LED を点滅させる効果を確認するには、 **Control** パレット内の [Wait 1 seconds] ブロックと [forever] ブロックを使用する必要があります。これらのブロックを書いた後にクリックすると、黄色のハロが表示されていることを意味します。

- [Wait 1 seconds]: **Control** パレットから、2 つのブロックの間の時間間隔を設定するために使用されます。
- [forever]: **Control** パレットから、手動で一時停止しない限りスクリプトを継続的に実行するために使用されます。

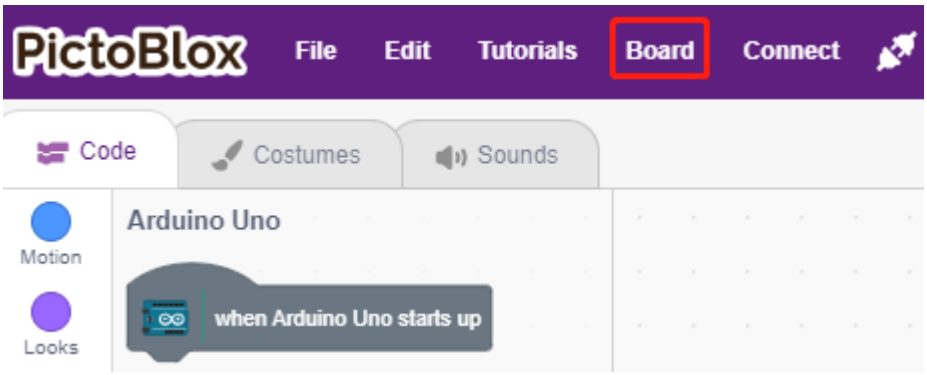



### 7.3.2 アップロードモード

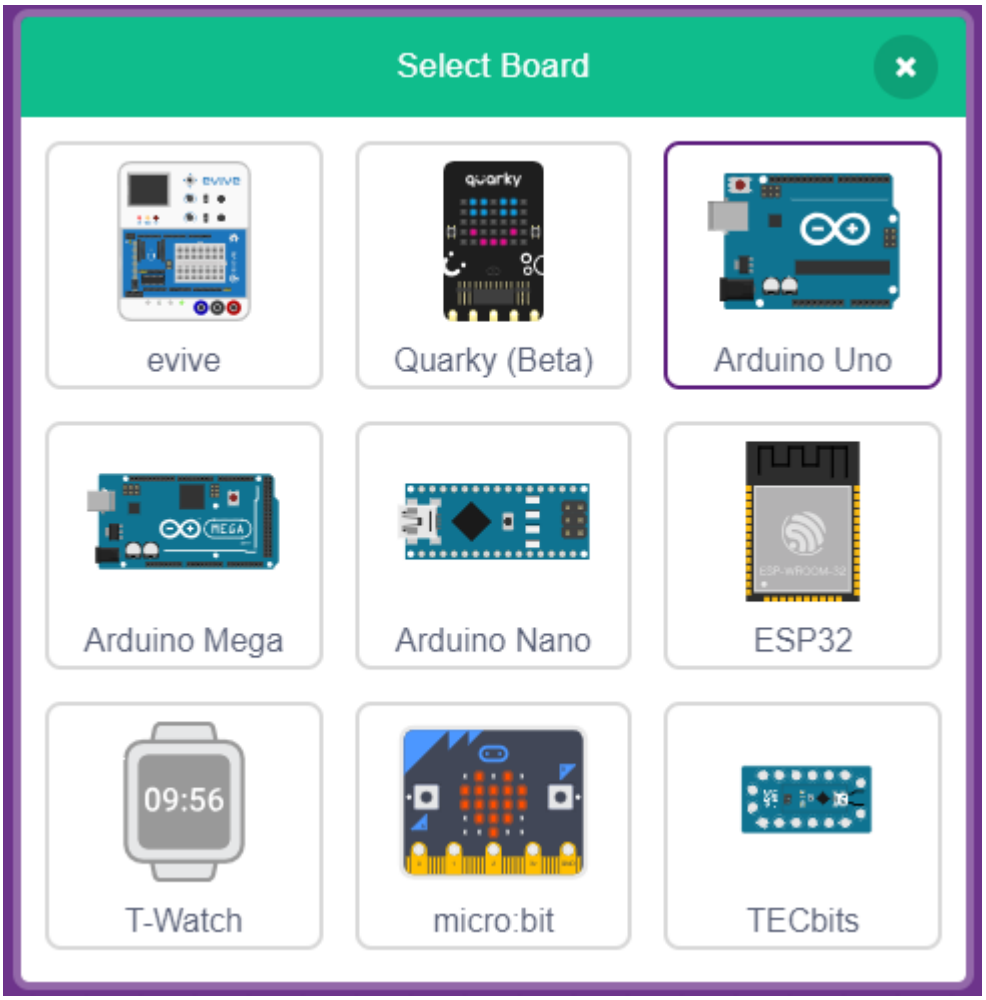
#### 1. Arduino ボードに接続する

Arduino ボードを USB ケーブルでコンピュータに接続します。通常、コンピュータはボードを自動的に認識し、最終的に COM ポートを割り当てます。

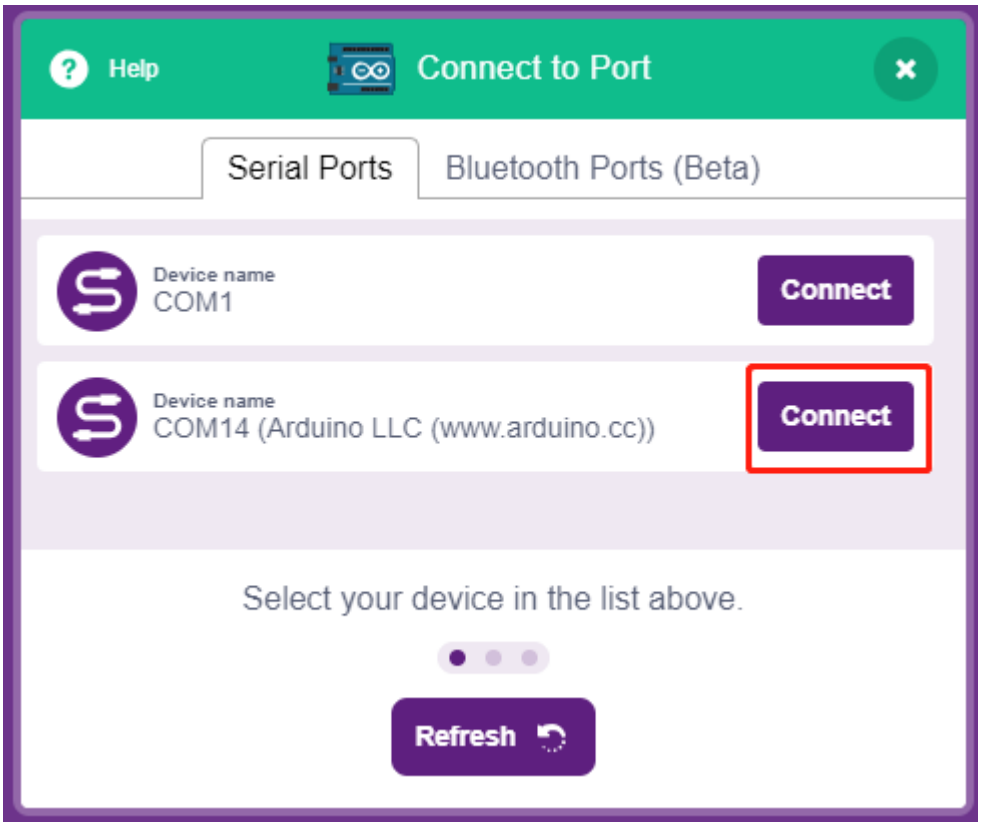
PictoBlox を開き、右上のナビゲーションバーで **Board** をクリックしてボードを選択します。




例として、 **Arduino Uno** を選択します。

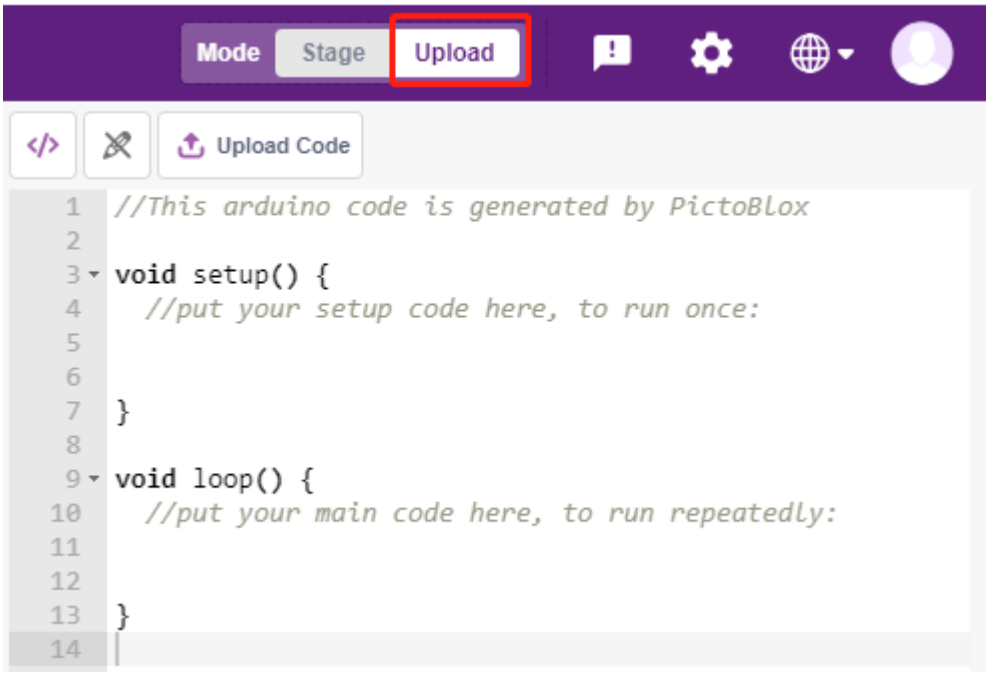



接続ウィンドウが表示され、接続するポートを選択します。接続が完了するとホームページに戻ります。使用中に接続が切れた場合は、 **Connect** をクリックして再接続できます。




同時に、 **Block Palette** に Arduino Uno 関連のパレット、例えば Arduino Uno やアクチュエータなどが表示されます。

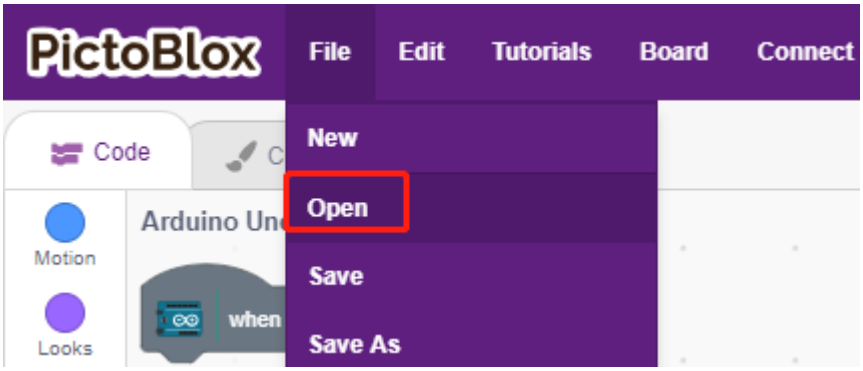

アップロードモードを選択すると、ステージは元の Arduino コードエリアに切り替わります。



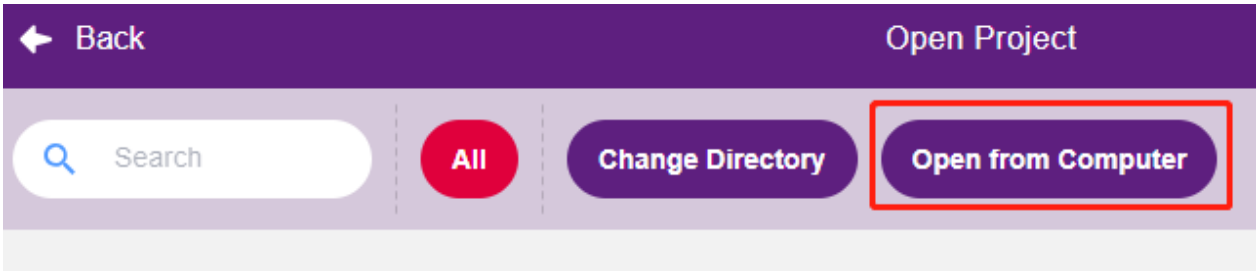

**2. プログラミング**

- スクリプトを直接開く・実行する

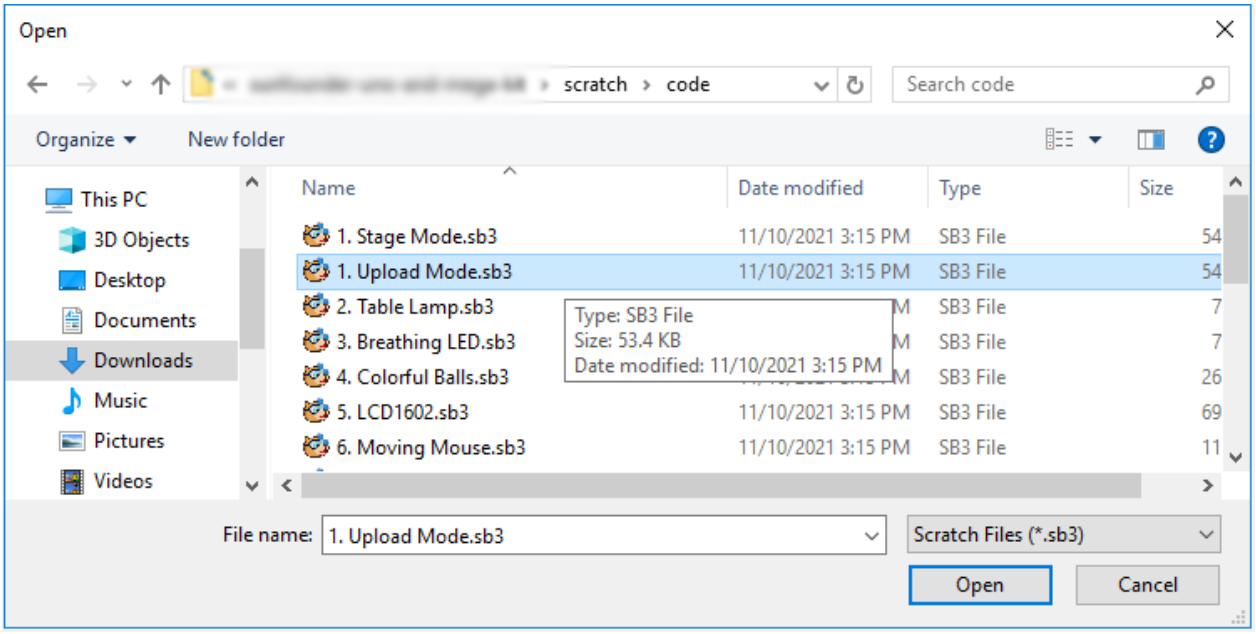
右上のコーナーの **File** をクリックします。



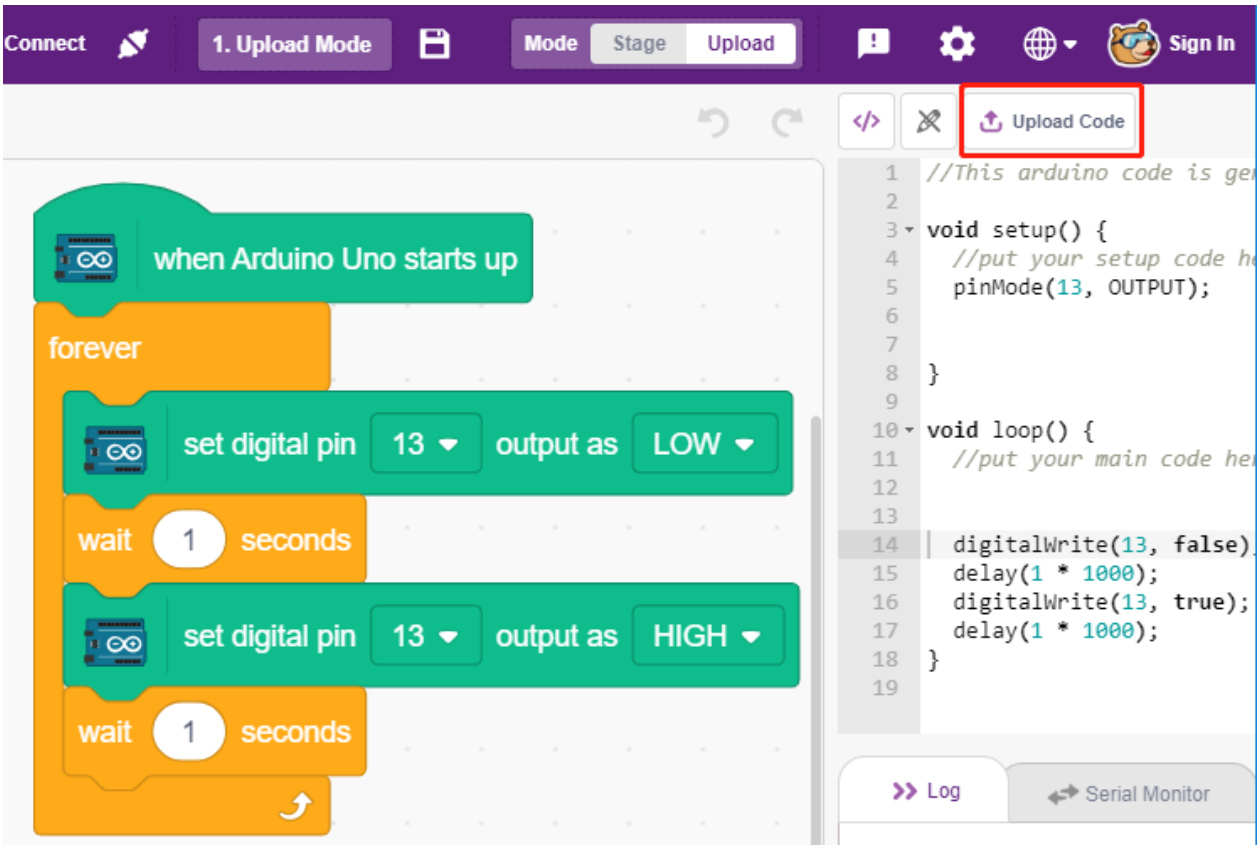

**Open from Computer** を選択します。




`3in1-kit\scratch_project\code` のパスに移動して、**1. Upload Mode.sb3** を開きます。必要なコードは github からダウンロードしたことを確認してください。




最後に、 **Upload Code** ボタンをクリックします。

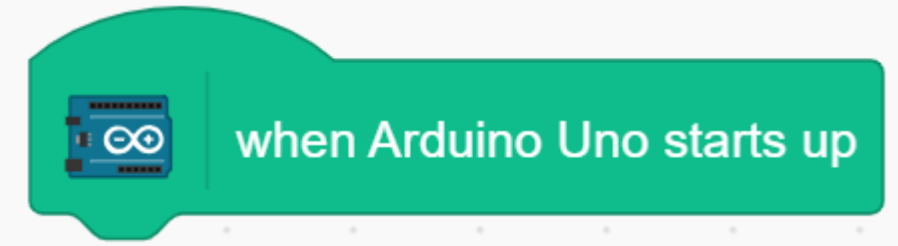



- ステップバイステップでプログラムする

これらの手順に従って、スクリプトをステップバイステップで記述することもできます。

**Arduino Uno** パレットをクリックします。

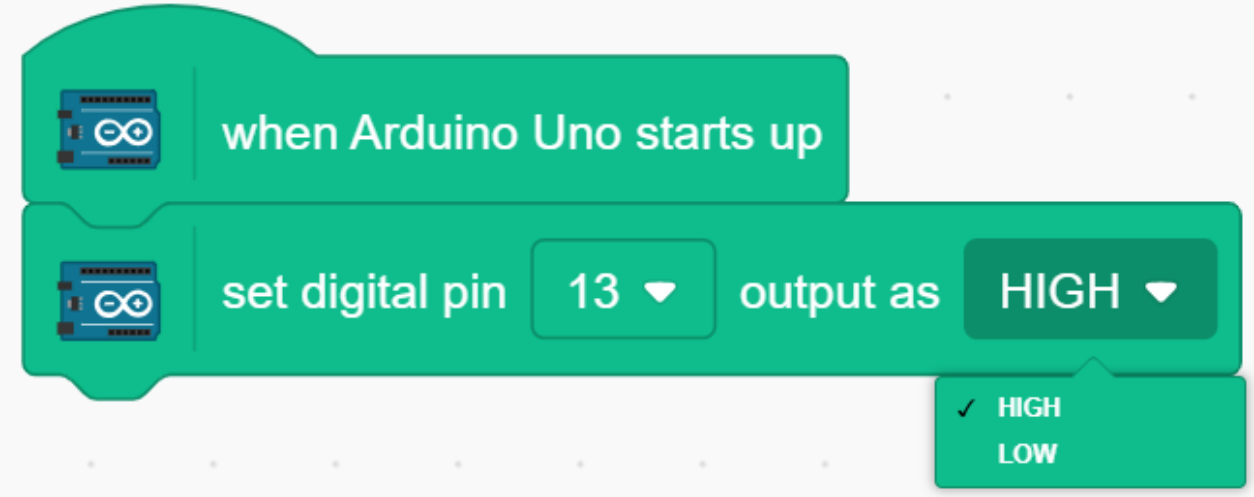

スクリプトエリアに [when Arduino Uno starts up] をドラッグします。これはすべてのスクリプトに必要です。




Arduino ボードの LED はデジタル pin13（2 つの状態のみ HIGH または LOW）によって制御されるため、[set digital pin out as] ブロックをスクリプトエリアにドラッグします。

LED のデフォルトの状態は点灯しているので、pin 13 を LOW に設定し、このブロックをクリックすると LED が消えます。

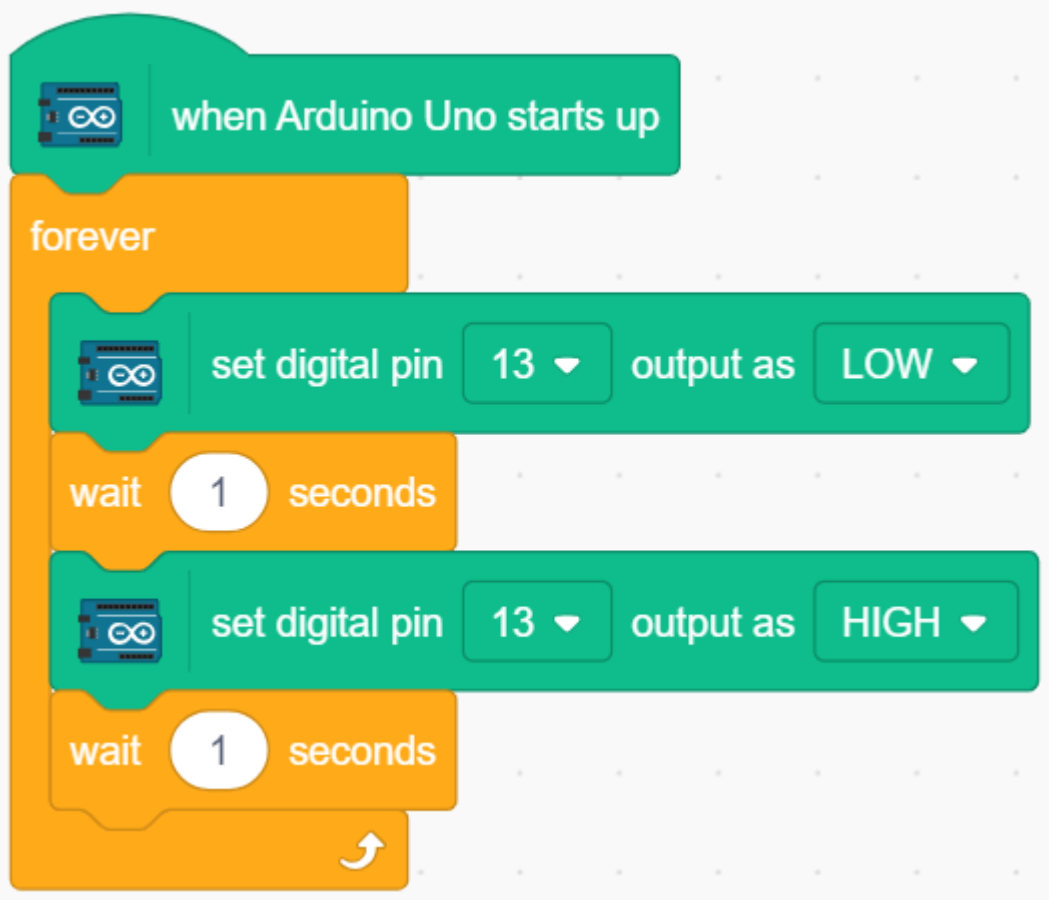
- [set digital pin out as]: デジタルピン（2〜13）を (HIGH/LOW) レベルに設定します。



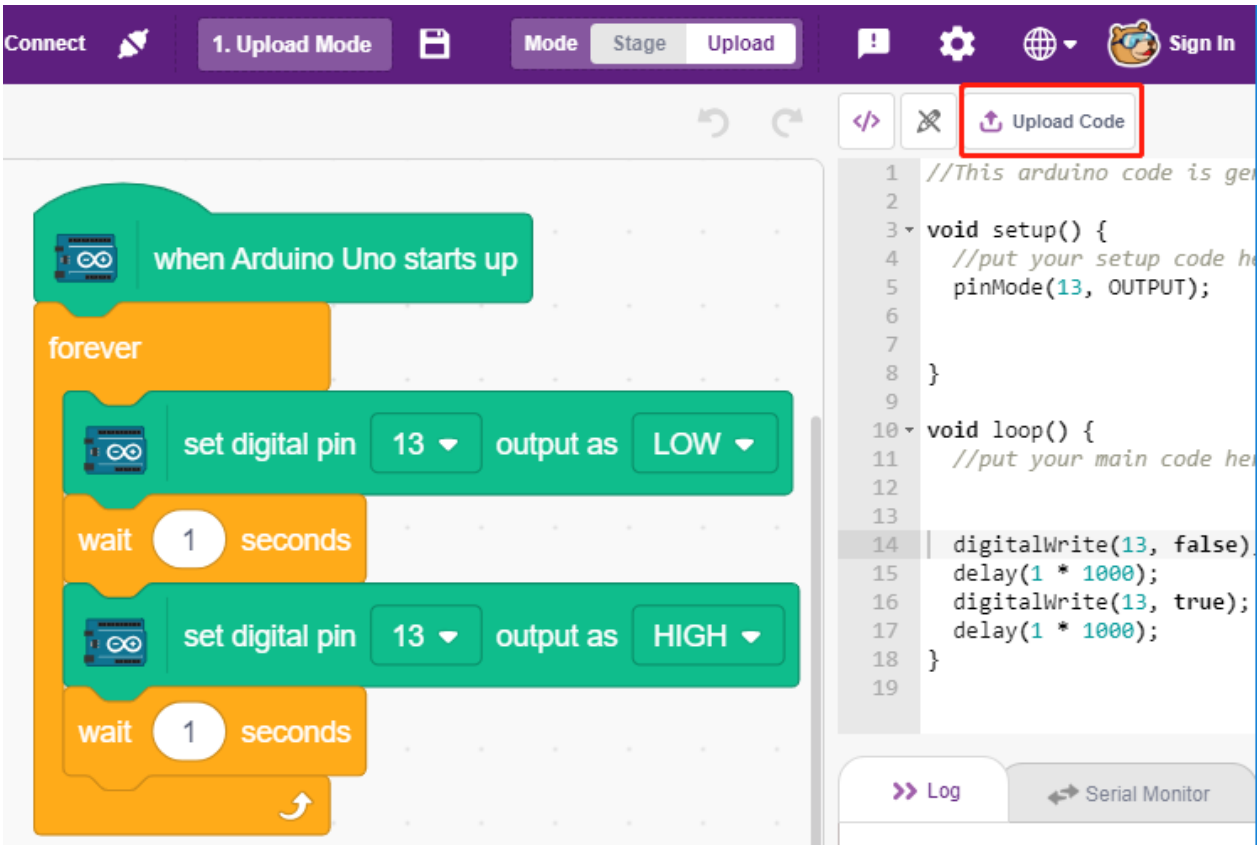

この時点で、右側に Arduino コードが表示されます。このコードを編集したい場合は、編集モードをオンにできます。


連続して点滅する LED の効果を見るために、 **Control** パレットの [Wait 1 seconds] と [forever] ブロックを使用する必要があります。これらのブロックを書き込んだ後、クリックすると黄色いハローが表示されていることを確認してください。これは、それが実行中であることを意味します。

- [Wait 1 seconds]: **Control** パレットから、2 つのブロック間の時間間隔を設定するために使用します。
- [forever]: **Control** パレットから、スクリプトを手動で一時停止するまで実行し続けることができます。




最後に、 **Upload Code** ボタンをクリックします。




**2. プロジェクト**

以下のプロジェクトはプログラミングの難易度の順に書かれています。これらの本を順番に読むことをおすすめします。

各プロジェクトでは、回路の組み立て方やプログラムの手順をステップバイステップで非常に詳細に教えています。最終結果を達成するための方法も含まれています。

もちろん、スクリプトを直接開いて実行することもできますが、関連する資料を github からダウンロードしたことを確認する必要があります。

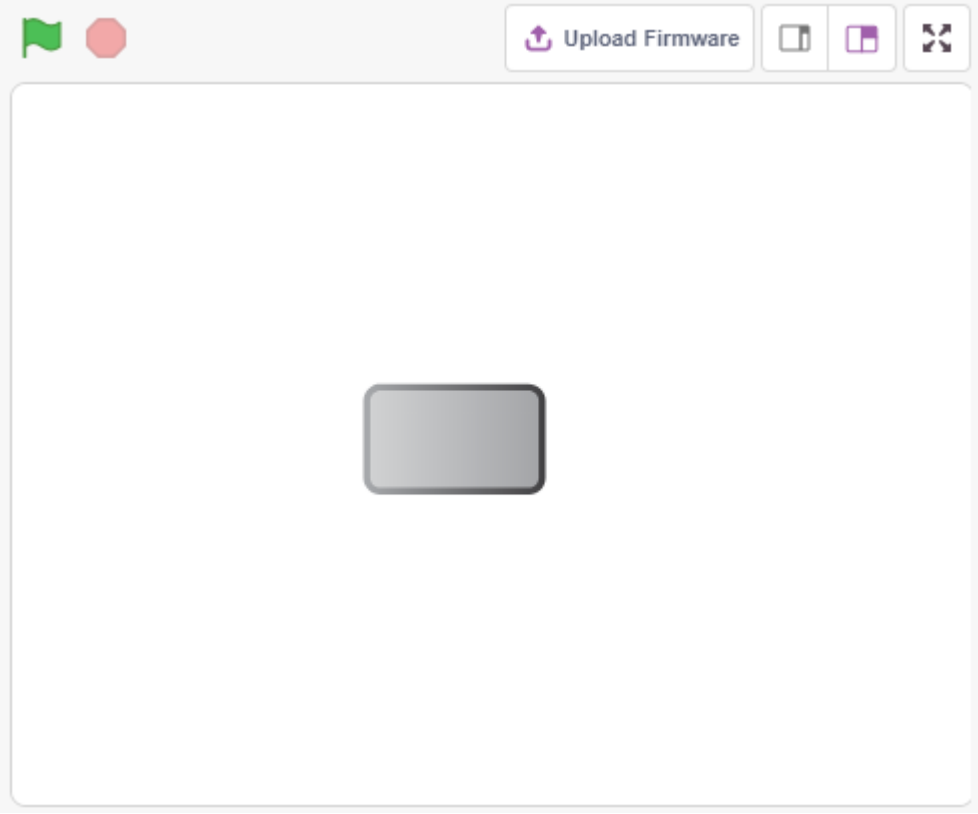
ダウンロードが完了したら、それを解凍します。 **ステージモード** を参照して、個々のスクリプトを直接実行します。

ただし、 *2.9* **温湿度の読取り** は **アップロードモード** を使用しています。

## 7.4 2.1 テーブルランプ

ここでは、ブレッドボードに LED を接続し、スプライトがこの LED の点滅を制御するようにします。

ステージ上のボタンスプライトがクリックされると、LED は 5 回点滅し、その後停止します。




### 7.4.1 学べること

- ブレッドボード、LED、抵抗
- ブレッドボード上の回路の作成
- スプライトの削除と選択
- コスチュームの切り替え
- 回数制限のある繰り返しループの設定

### 7.4.2 必要な部品

このプロジェクトでは、以下のコンポーネントが必要です。

全体のキットを購入すると便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

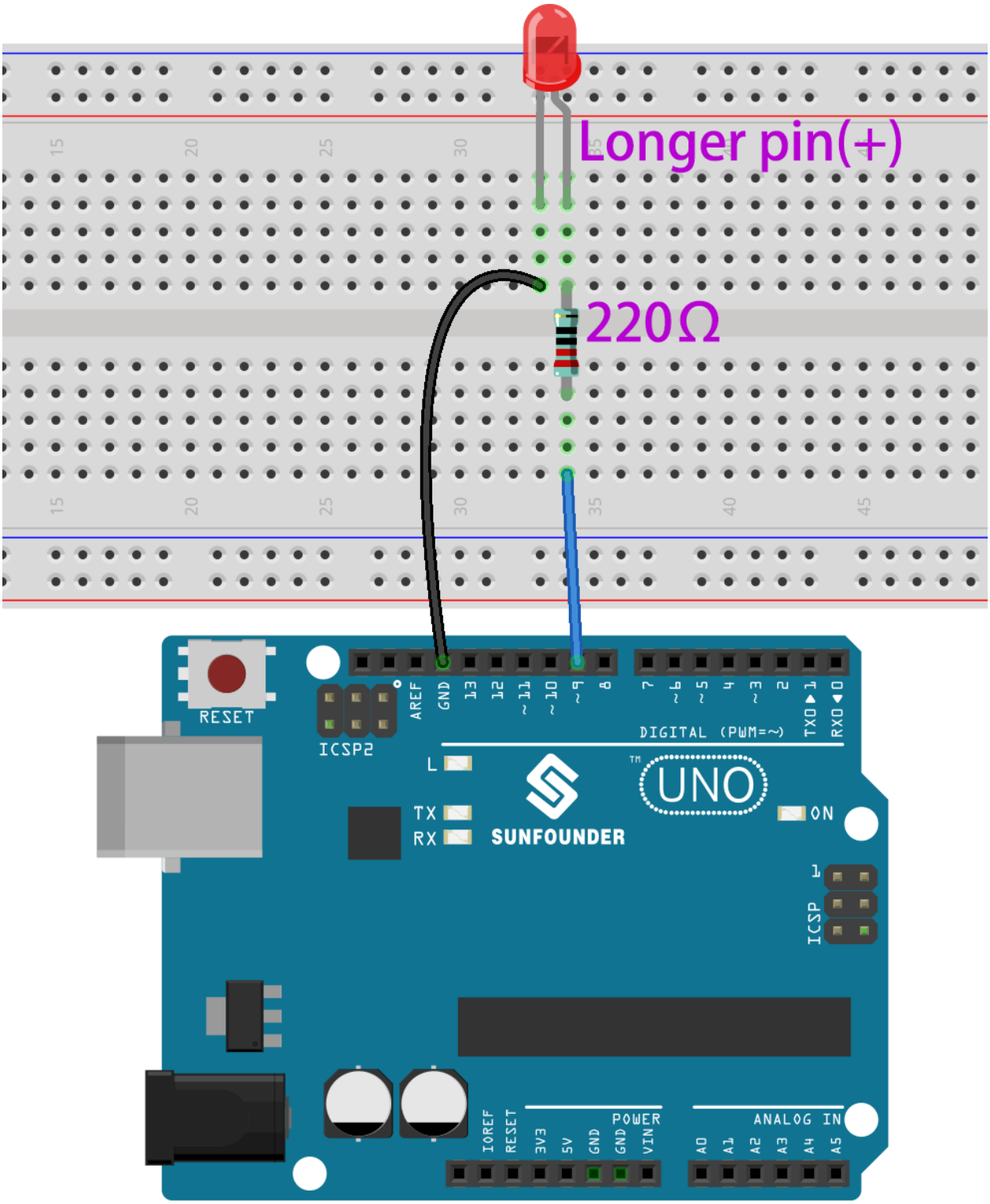
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| **抵抗器** | |
| *LED* | |

### 7.4.3 回路の作成

以下の図に従ってブレッドボード上に回路を作ります。

LED のアノード（長いピン）は 220 Ωの抵抗を通じてピン 9 に接続され、LED のカソードは GND に接続されているので、ピン 9 に高レベルを与えることで LED を点灯させることができます。

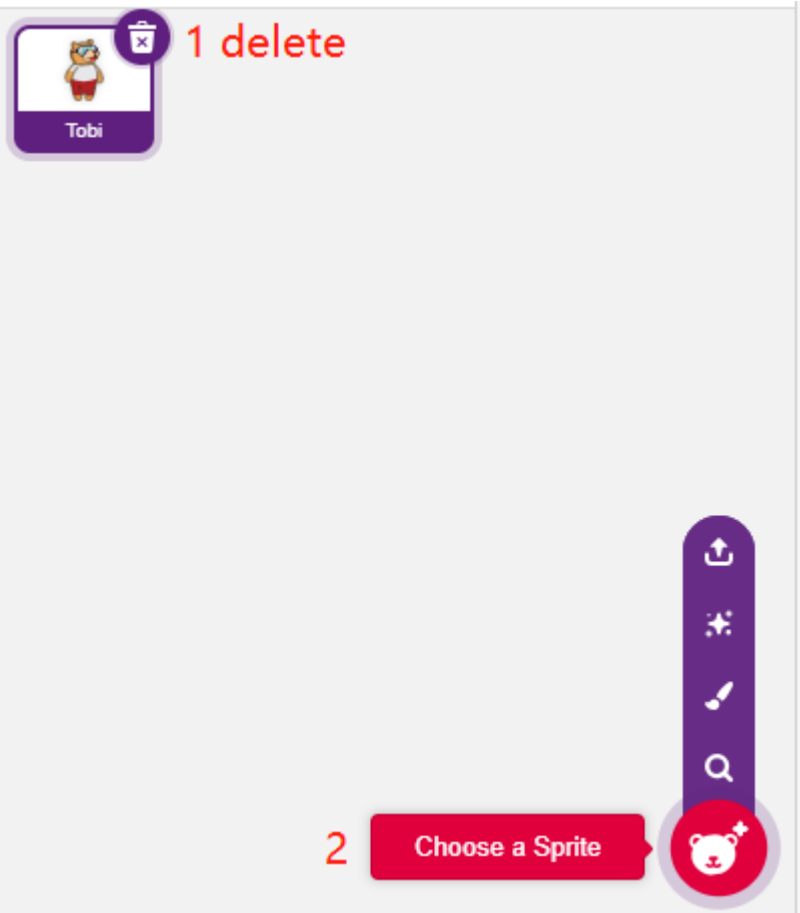

Longer pin(+)
220Ω
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

### 7.4.4 プログラミング

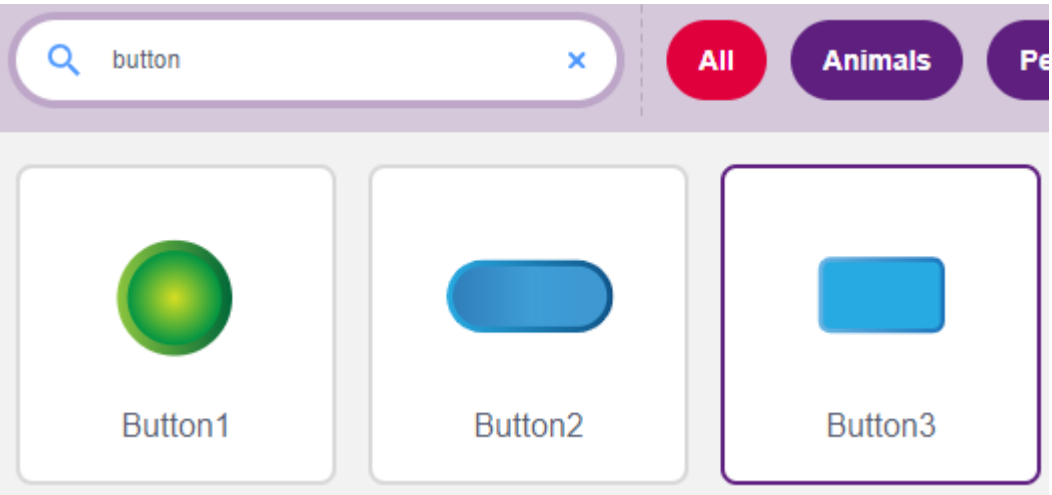
全体のプログラミングは 3 部分に分かれています。第 1 部は目的のスプライトを選択すること、第 2 部はスプライトのコスチュームを切り替えてクリック可能に見せること、第 3 部は LED を点滅させることです。

**1. Button3 スプライトの選択**

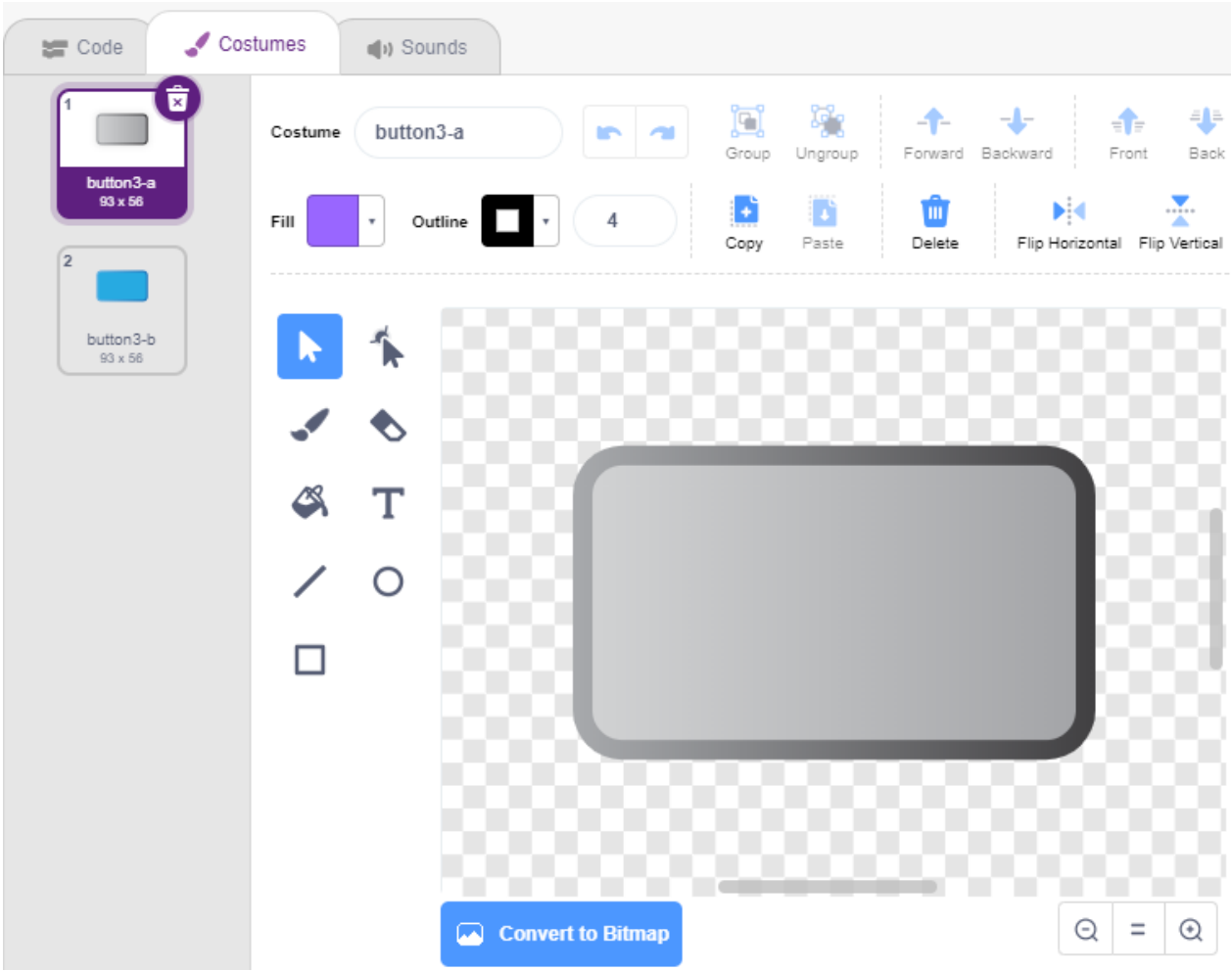
右上角の Delete ボタンを使用して既存の Tobi スプライトを削除し、再度スプライトを選択します。




ここでは、 **Button3** スプライトを選択します。

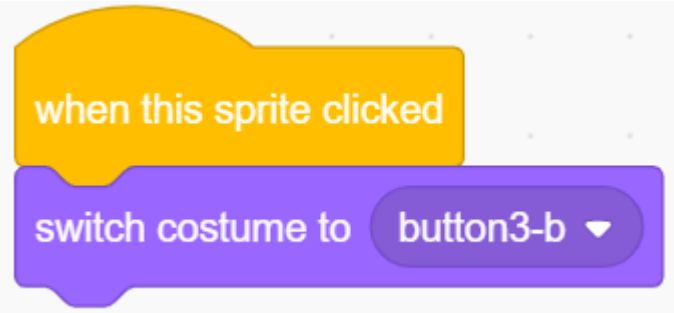



右上の Costumes をクリックすると、Button3 スプライトには 2 つのコスチュームがあることがわかります。**button3-a** をリリース状態、 **button3-b** を押下状態に設定します。




**2. コスチュームの切り替え**

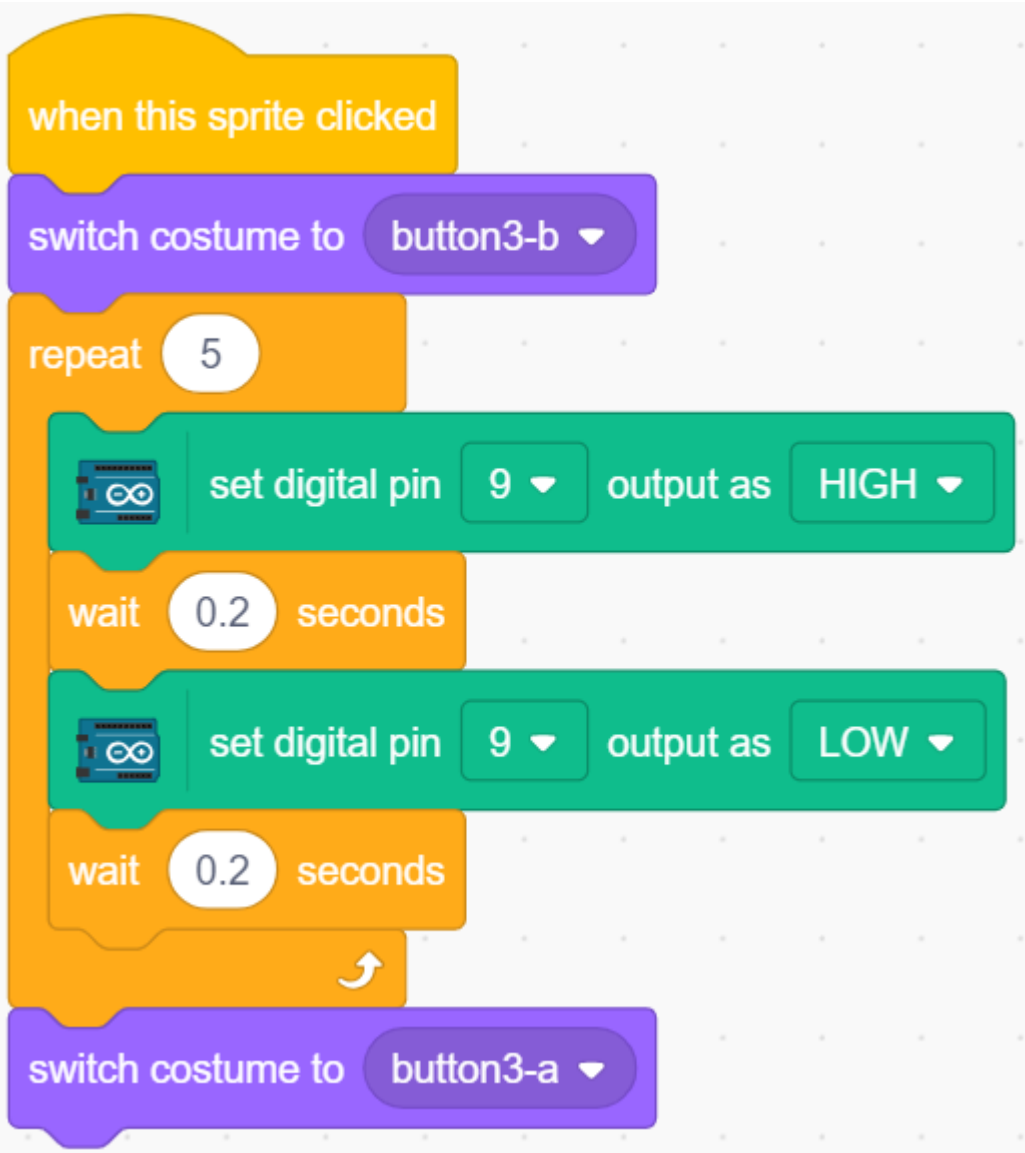
スプライトがクリックされたとき（ **Events** パレット）、 **button3-b** のコスチュームに切り替えます（ **looks** パレット）。




**3. LED を 5 回点滅させる**

[Repeat] ブロックを使用して LED を 5 回点滅させる（High-> LOW サイクル）、ピン 13 をピン 9 に変更し、最後にコスチュームを **button3-a** に戻します。

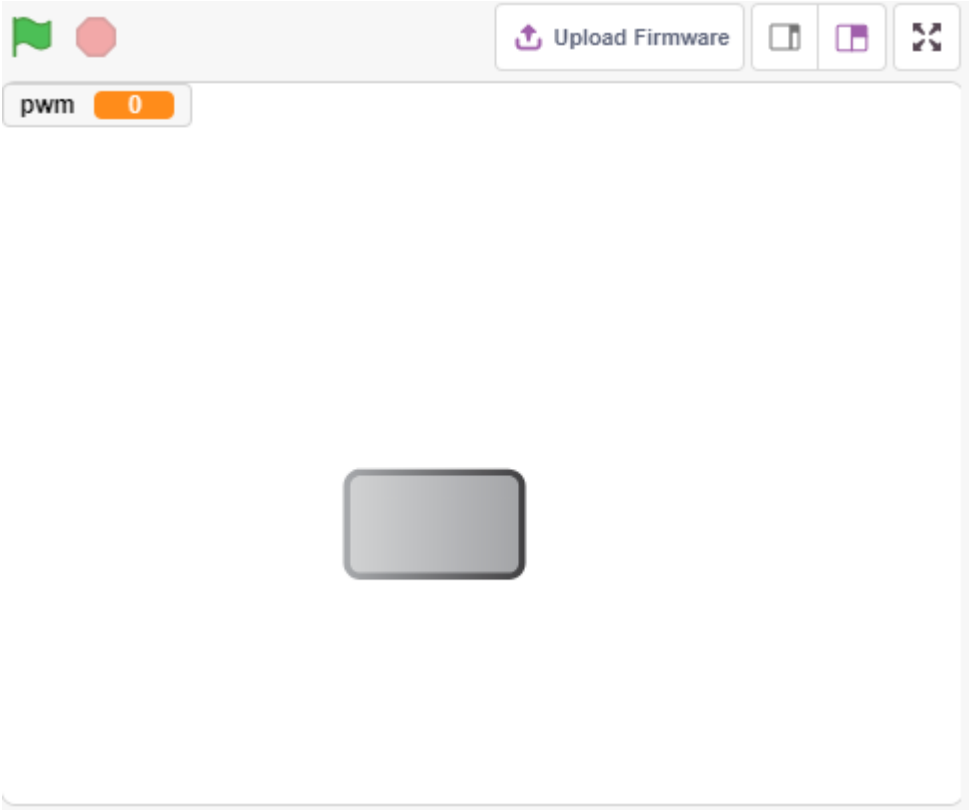
- [Repeat 10]: 回数制限のある繰り返しループ。自分で繰り返し回数を設定できます。 **Control** パレットから選択します。




## 7.5 2.2 ブリージング LED

LED の明るさを制御するための別の方法を使います。前のプロジェクトとは異なり、ここでは LED の明るさを徐々に減少させ、消失するようにします。

ステージ上のスプライトをクリックすると、LED の明るさが徐々に増し、瞬時に消灯します。




### 7.5.1 学べる内容

- PWM ピンの出力値の設定
- 変数の作成
- スプライトの明るさの変更

### 7.5.2 必要な部品

このプロジェクトでは、以下のコンポーネントが必要です。

一式をまとめて購入すると便利です。購入リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

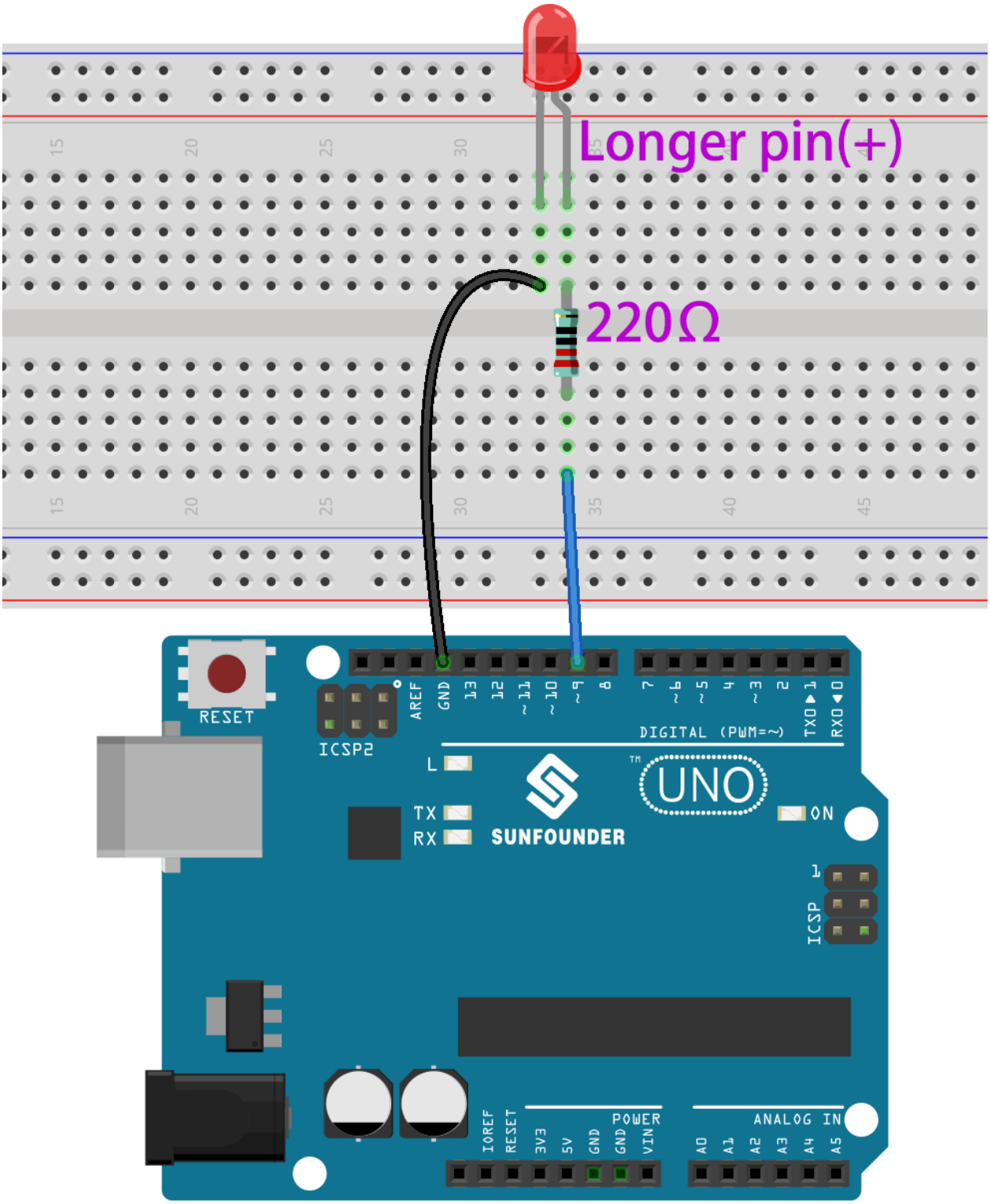
以下のリンクから、それぞれの部品を個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *LED* | |

### 7.5.3 回路の作成

このプロジェクトは前のプロジェクト *2.1* テーブルランプ と同じ回路を使用しますが、LED を点灯または消灯させるために HIGH/LOW を使用する代わりに、 PWM - Wikipedia シグナルを使用して LED を徐々に点灯または消灯させます。

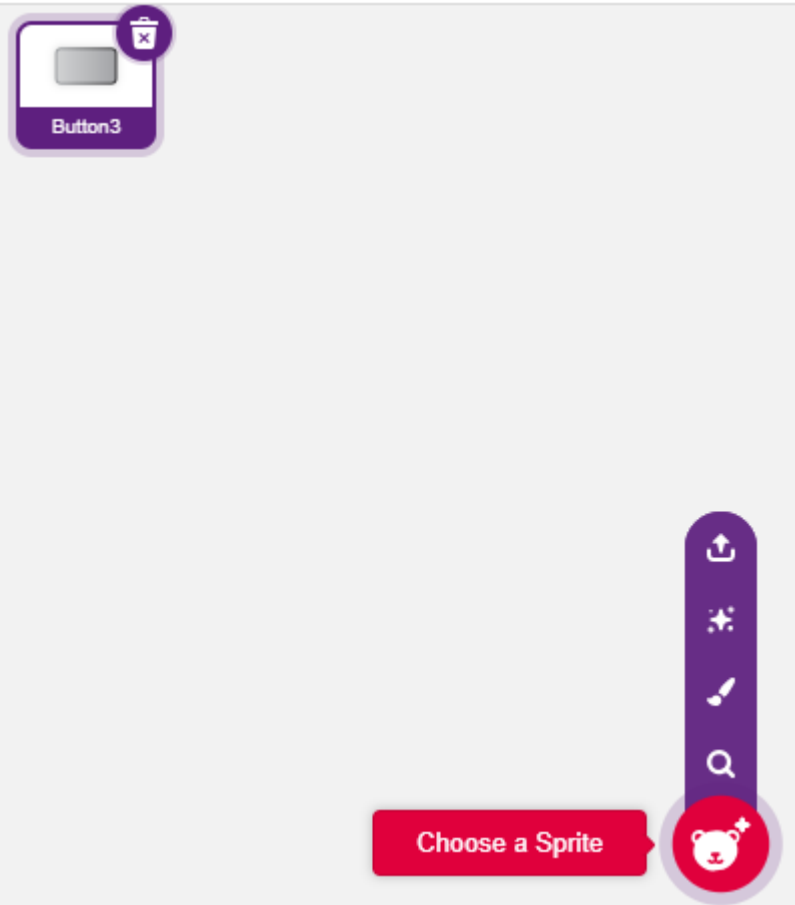
PWM 信号の範囲は 0-255 で、Arduino Uno ボードでは 3, 5, 6, 9, 10, 11 が PWM 信号を出力できます。Mega2560 では、2 - 13, 44 - 46 が PWM 信号を出力できます。


Longer pin(+)
220Ω
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

### 7.5.4 プログラミング

**1. スプライトを選択**

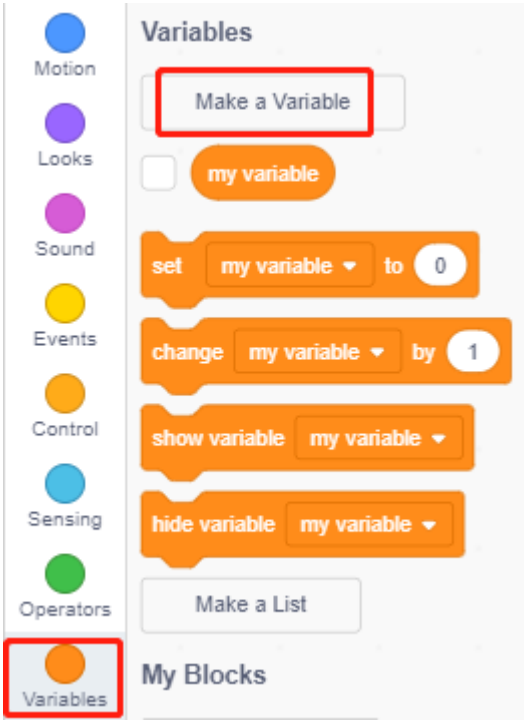
デフォルトのスプライトを削除し、スプライト領域の右下隅にある **Choose a Sprite** ボタンをクリックし、検索ボックスに **button3** と入力して追加します。



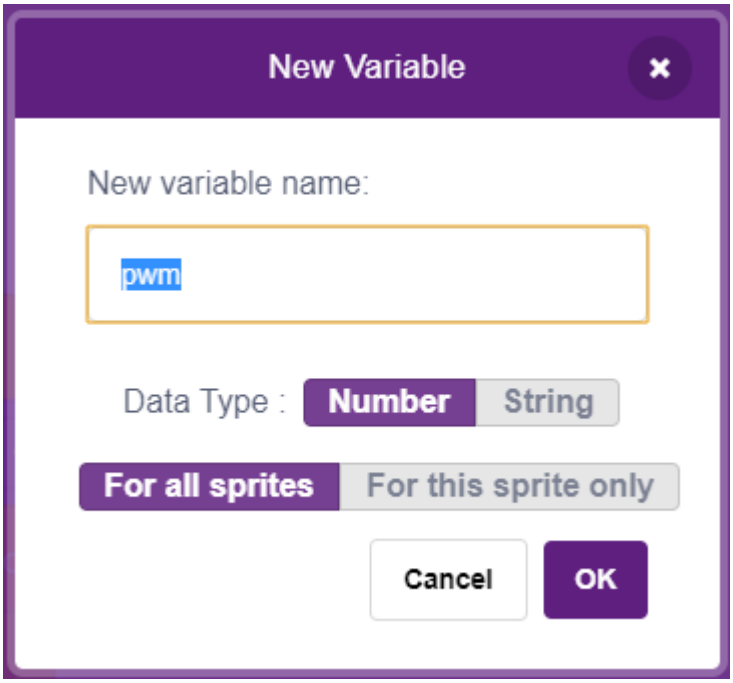

**2. 変数の作成**。

pwm の値の変化を保存するための変数 **pwm** を作成します。

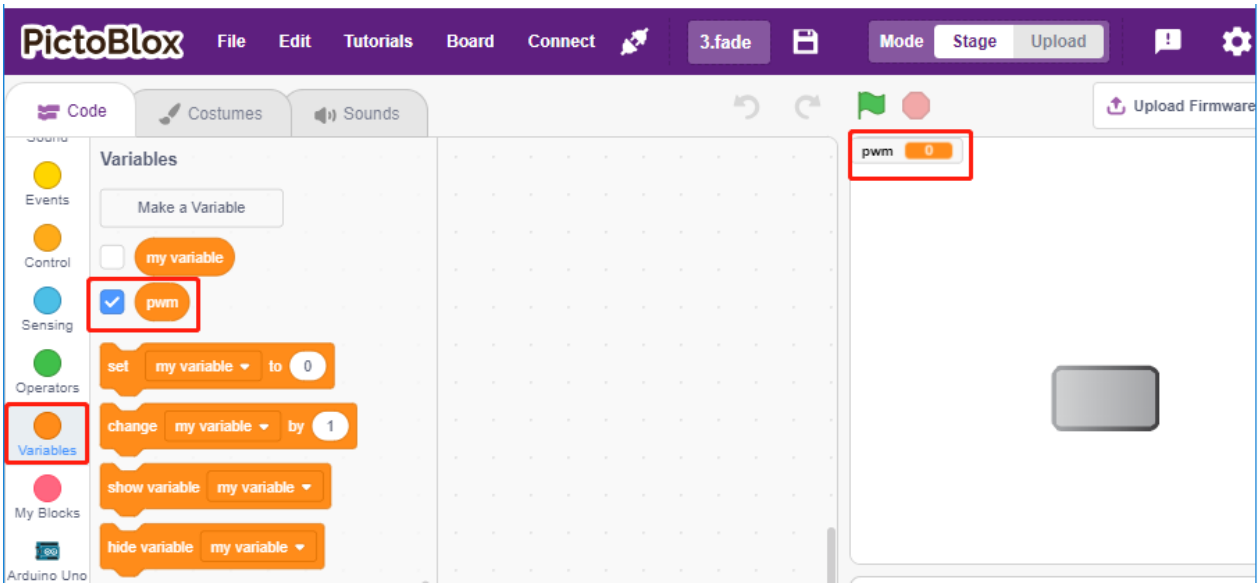
**Variables** パレットをクリックして **Make a Variable** を選択します。




変数の名前を入力します。任意の名前で構いませんが、その機能を説明することを推奨します。データタイプは数値で、すべてのスプライト用です。



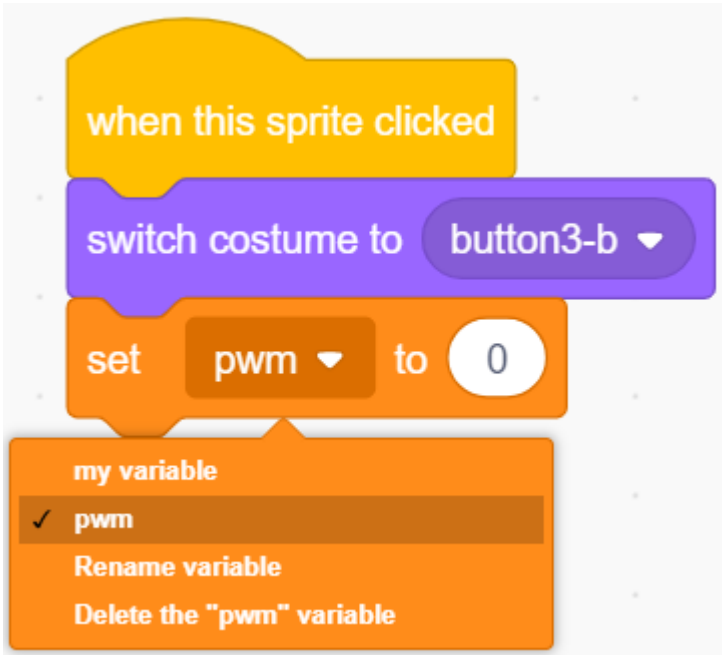

作成したら、 **Variables** パレット内とチェック状態で **pwm** が表示されます。これは、この変数がステージ上に表示されることを意味します。チェックを外して、ステージ上に pwm がまだ存在するかどうか確認してみてください。




### 3. 初期状態の設定

**button3** スプライトがクリックされたとき、コスチュームを **button-b**（クリック状態）に切り替え、変数 **pwm** の初期値を 0 に設定します。

- [set pwm to 0]: **Variables** パレットから、変数の値を設定するために使用されます。



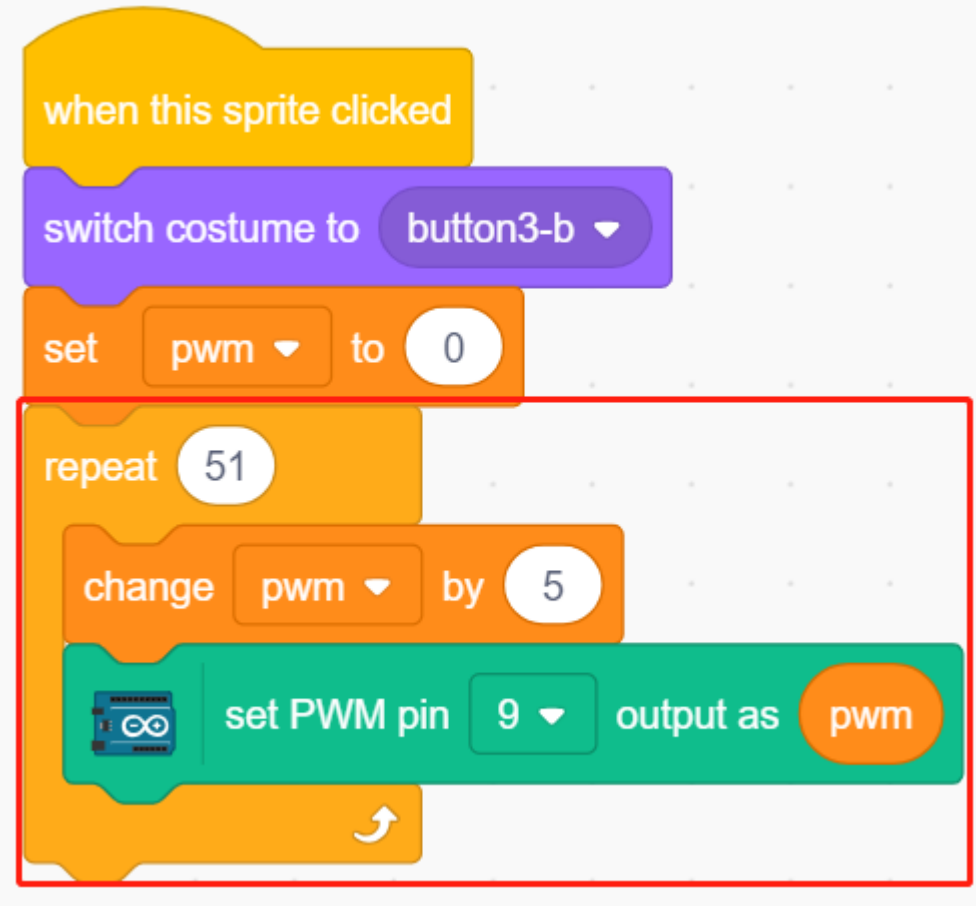

### 4. LED を次第に明るくする

pwm の範囲は 255 なので、[repeat] ブロックを使用して、変数 **pwm** を 5 ずつ 255 に累積し、[set PWM pin] ブロックに入れると、LED が徐々に点灯するのを見ることができます。

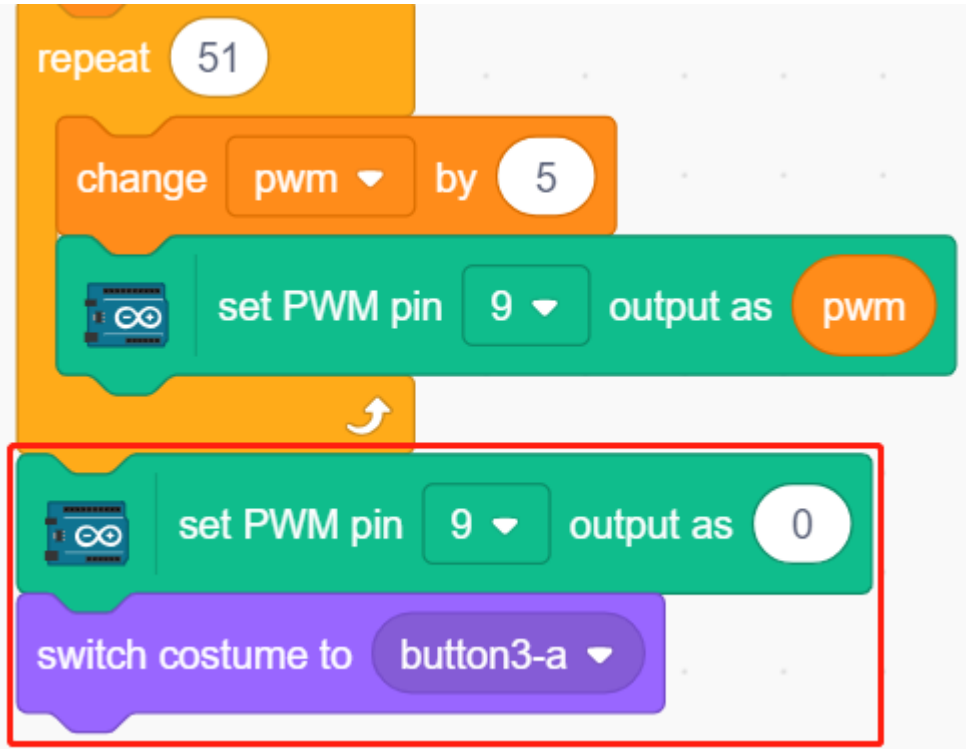
- [change pwm by 5]: **Variables** パレットから、変数を特定の数だけ変更するために使用されます。正または

負の数をとることができ、正は毎回増加、負は毎回減少を意味します。例えば、ここでは変数 pwm を毎回 5 ずつ増加させています。

- [set PWM pin]: **Arduino Uno** パレットから、pwm ピンの出力値を設定するために使用されます。



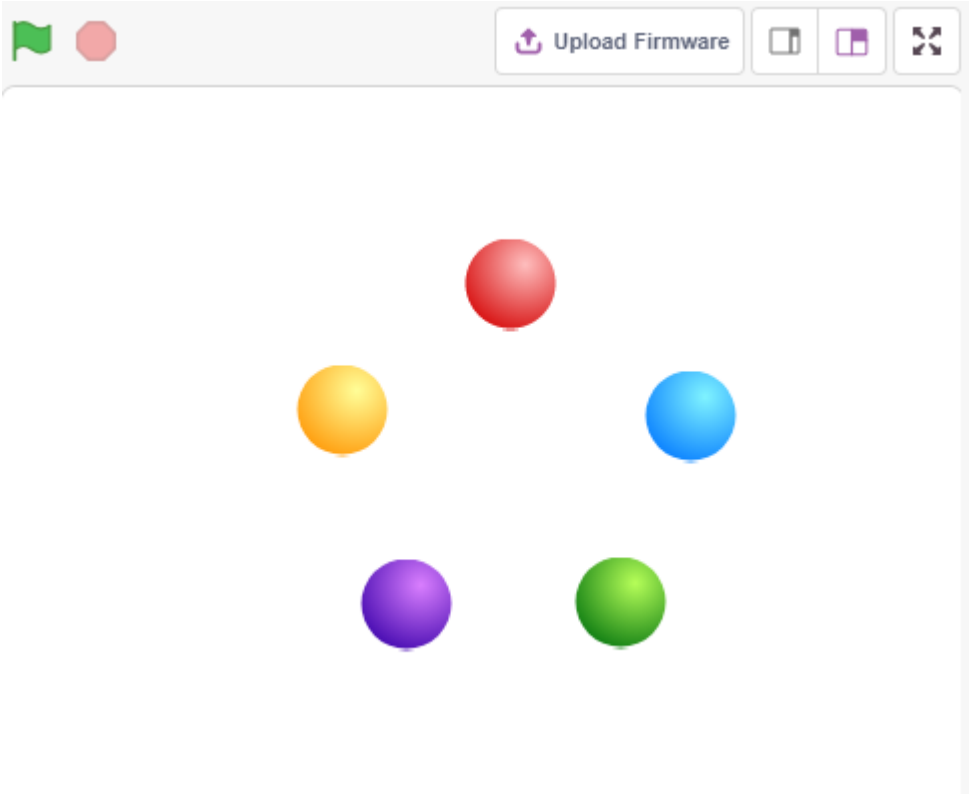

最後に、button3 のコスチュームを **button-a** に切り替え、PWM ピンの値を 0 にして、LED が徐々に点灯した後、再び消灯するようにします。




## 7.6 2.3 カラフルボール

このプロジェクトでは、RGB LED を使ってさまざまな色を表示させます。

ステージエリアの異なる色のボールをクリックすると、RGB LED が異なる色で点灯します。




### 7.6.1 学べること

- RGB LED の原理
- スプライトの複製と異なるコスチュームの選択
- 三原色の重ね合わせ

### 7.6.2 必要な部品

このプロジェクトには以下の部品が必要です。

全体のキットを購入するのが確かに便利です。リンクはこちら：

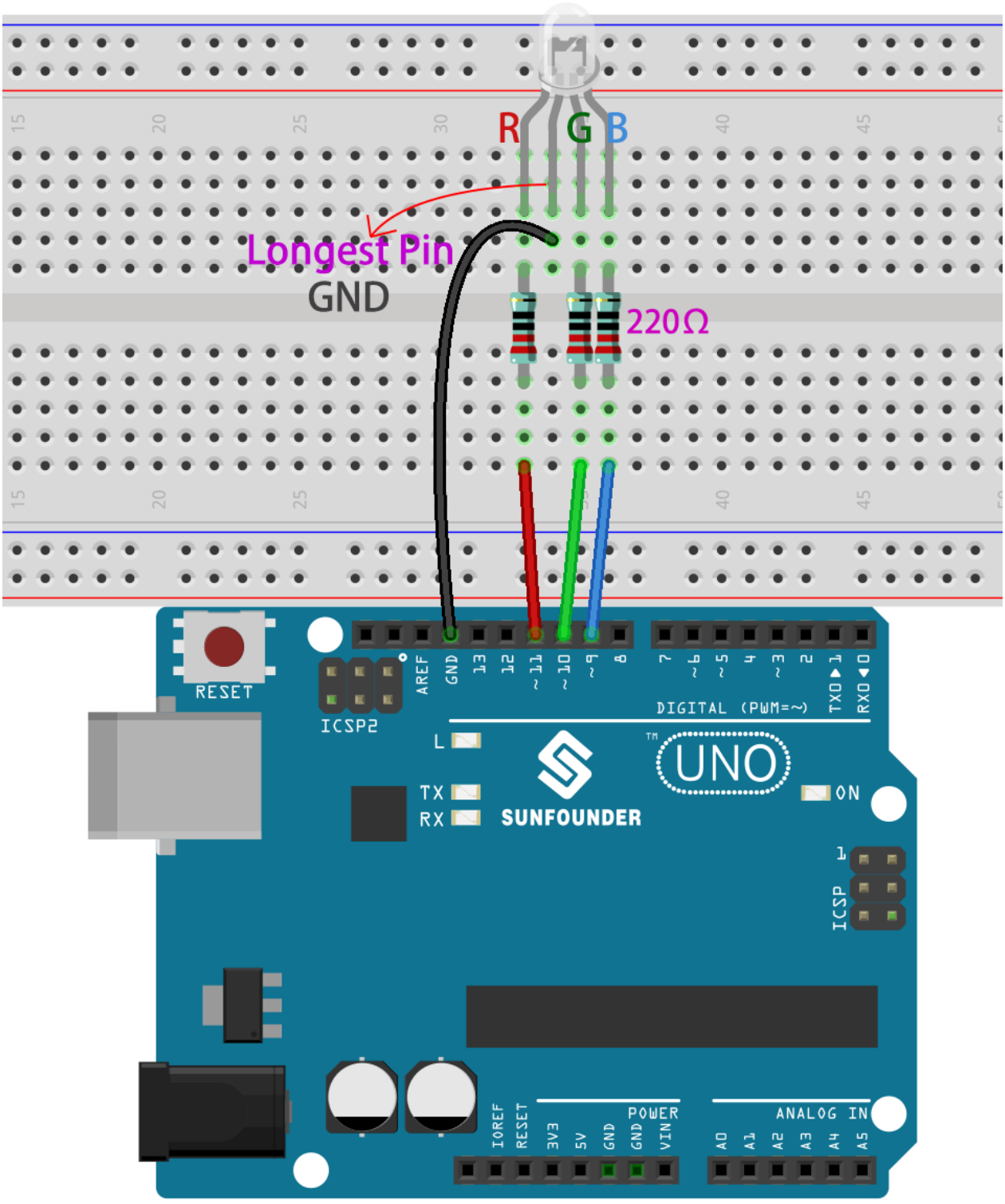
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| *RGB LED* | |

### 7.6.3 回路の作成

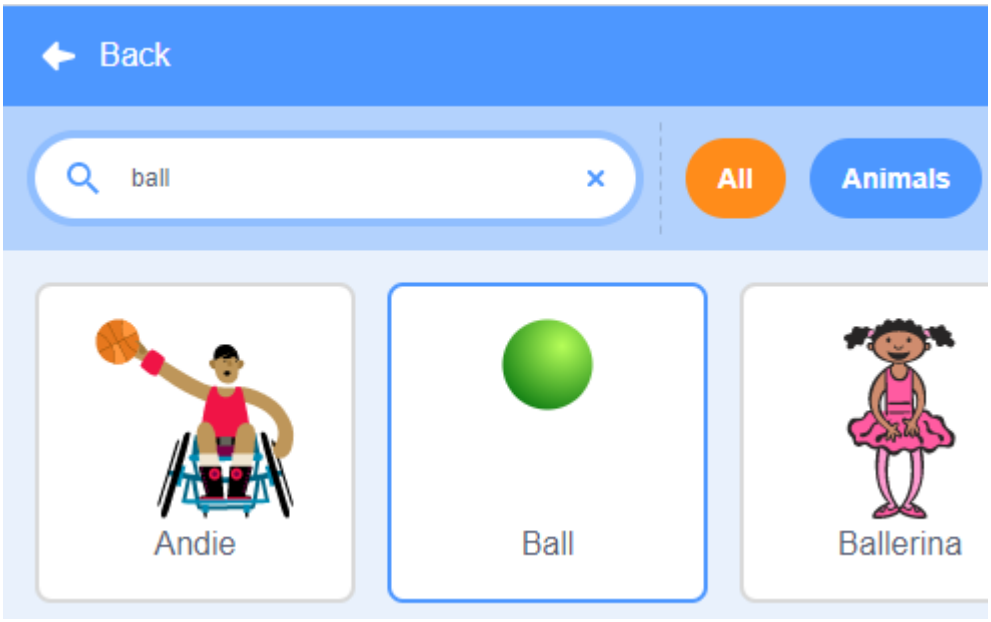
RGB LED は、赤、緑、青の 3 つの LED を透明または半透明のプラスチックシェルにパッケージングします。3 つのピンの入力電圧を変えることでさまざまな色を表示でき、それらを重ね合わせることで、統計によれば 16,777,216 色の異なる色を作り出すことができます。


R
G
B
Longest Pin
GND
220Ω
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

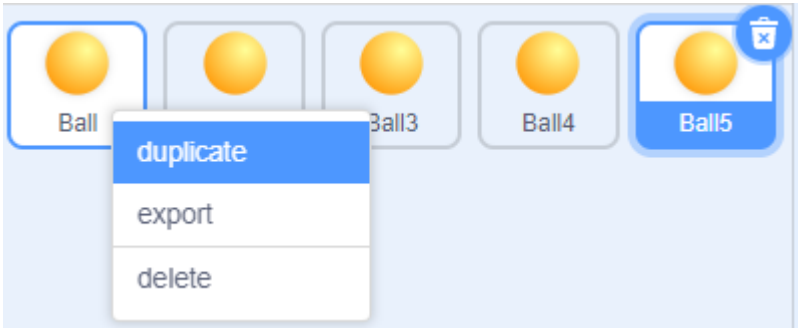
### 7.6.4 プログラミング

#### 1. スプライトの選択

デフォルトのスプライトを削除し、 **Ball** スプライトを選択します。




それを 5 回複製します。

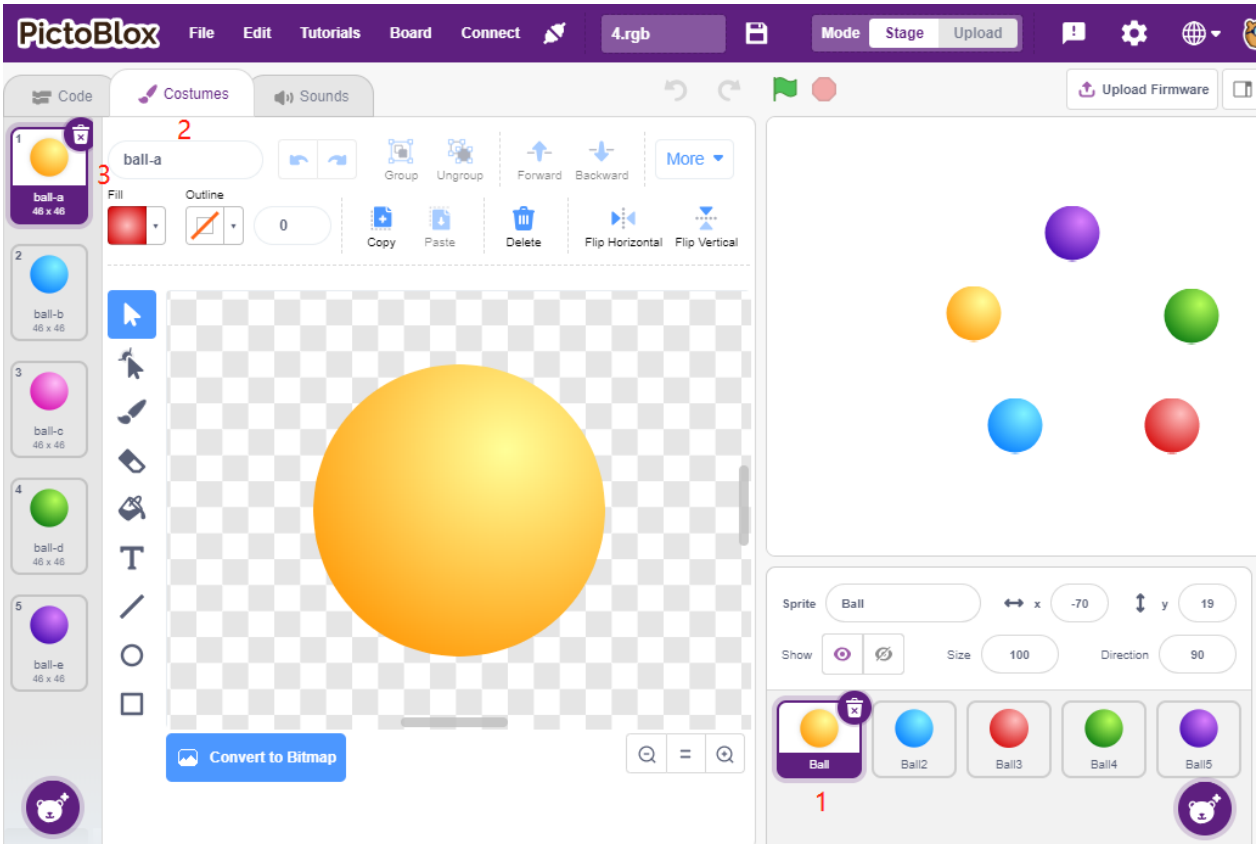



これらの 5 つの **Ball** スプライトに異なるコスチュームを選び、それぞれの位置に移動させます。

---

**注釈:** **Ball3** のスプライトのコスチュームの色は、手動で赤に変更する必要があります。

---

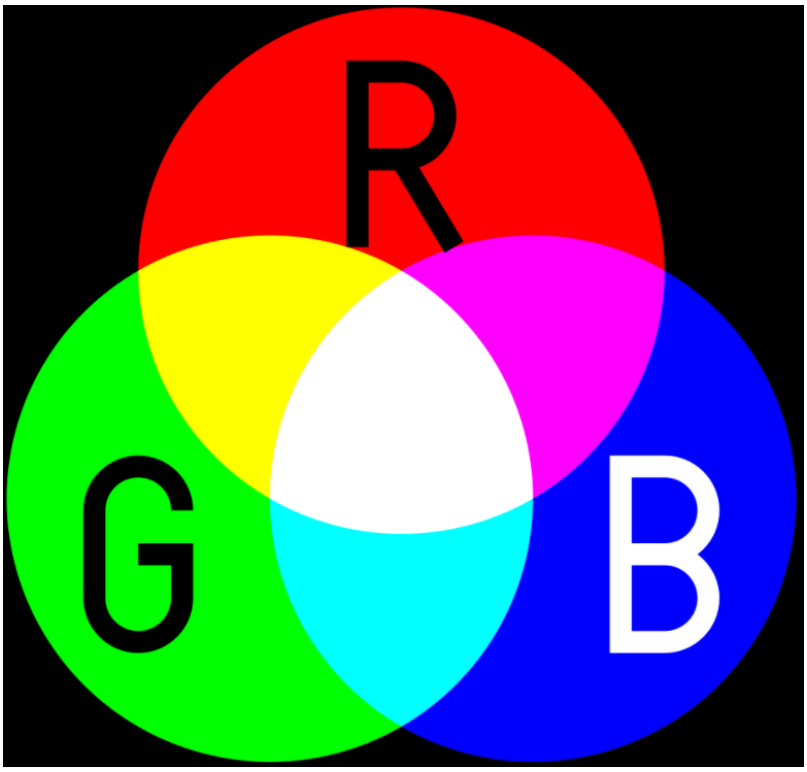



**2. RGB LED を適切な色で点灯させる**

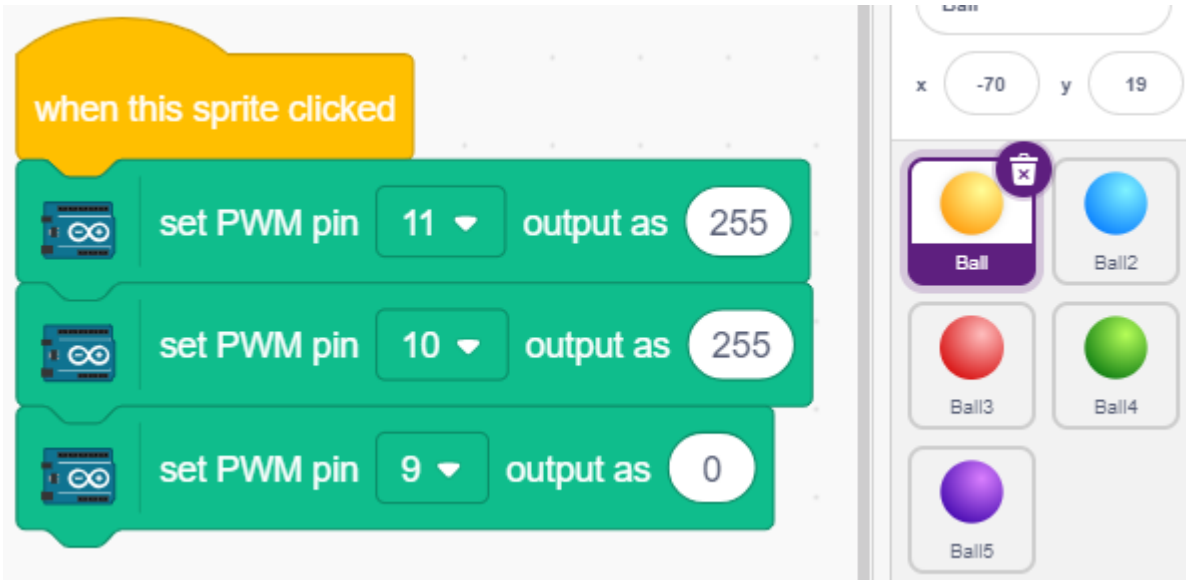
コードを理解する前に、 RGB 色モデル を理解する必要があります。

RGB 色モデルは、赤、緑、青の光をさまざまな方法で加えて、幅広い色を再現する加色モデルです。

加色混合：赤と緑を加えると黄色、緑と青を加えるとシアン、青と赤を加えるとマゼンタ、3 つの原色をすべて加えると白になります。



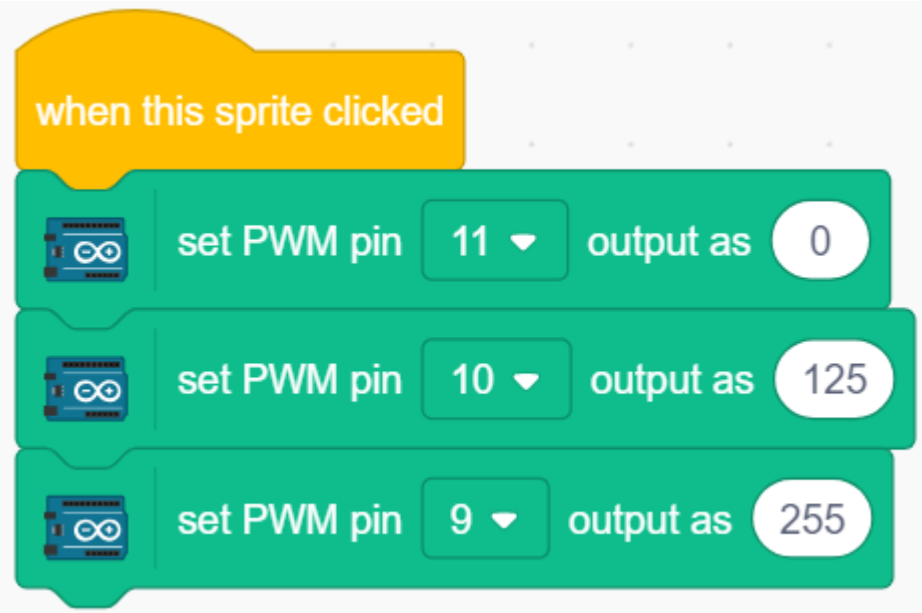

RGB LED を黄色に点灯させるコードは以下の通りです。



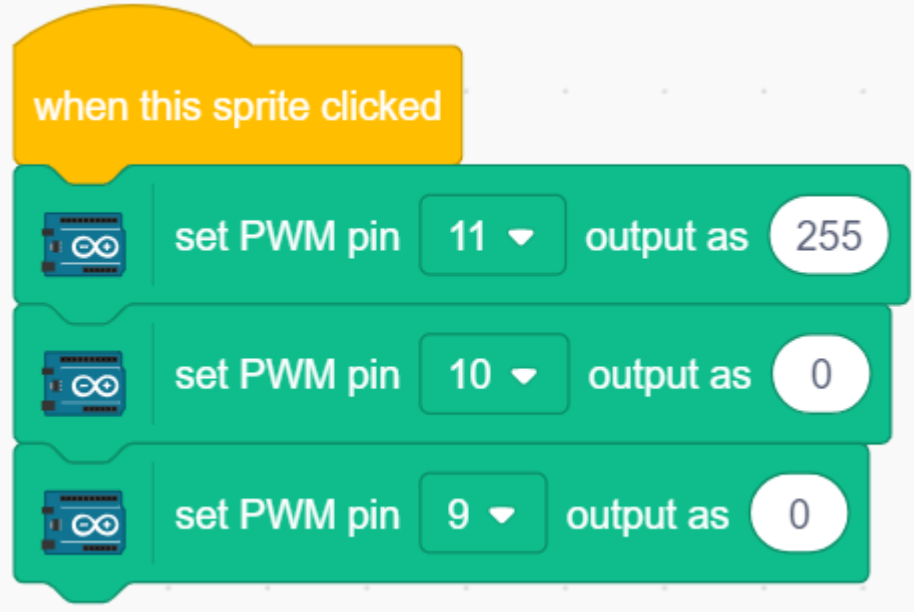

ボールスプライト（黄色のボール）がクリックされたとき、ピン 11 をハイ（赤い LED 点灯）、ピン 10 をハイ（緑の LED 点灯）、ピン 9 をロー（青い LED 消灯）に設定して、RGB LED が黄色に点灯するようにします。

他のスプライトにも、対応する色で RGB LED を点灯させるコードを書くことができます。

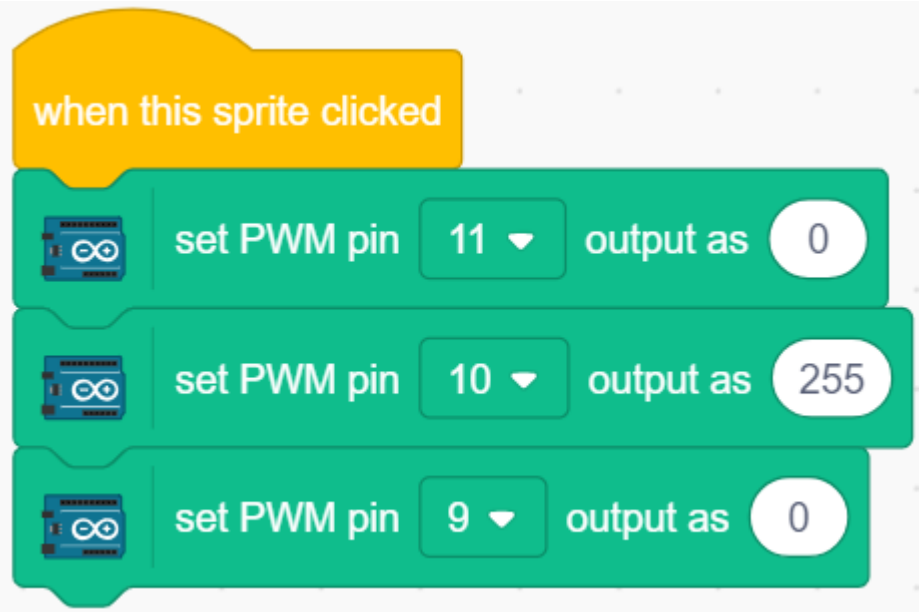
**3. Ball2 スプライト（ライトブルー）**



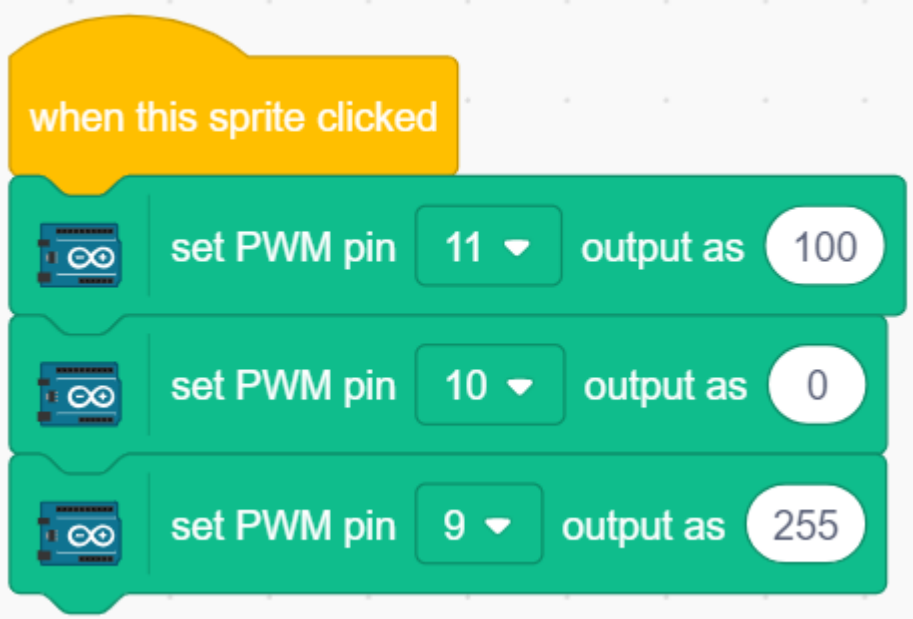

4. Ball3 スプライト（赤）




5. Ball4 スプライト（緑）



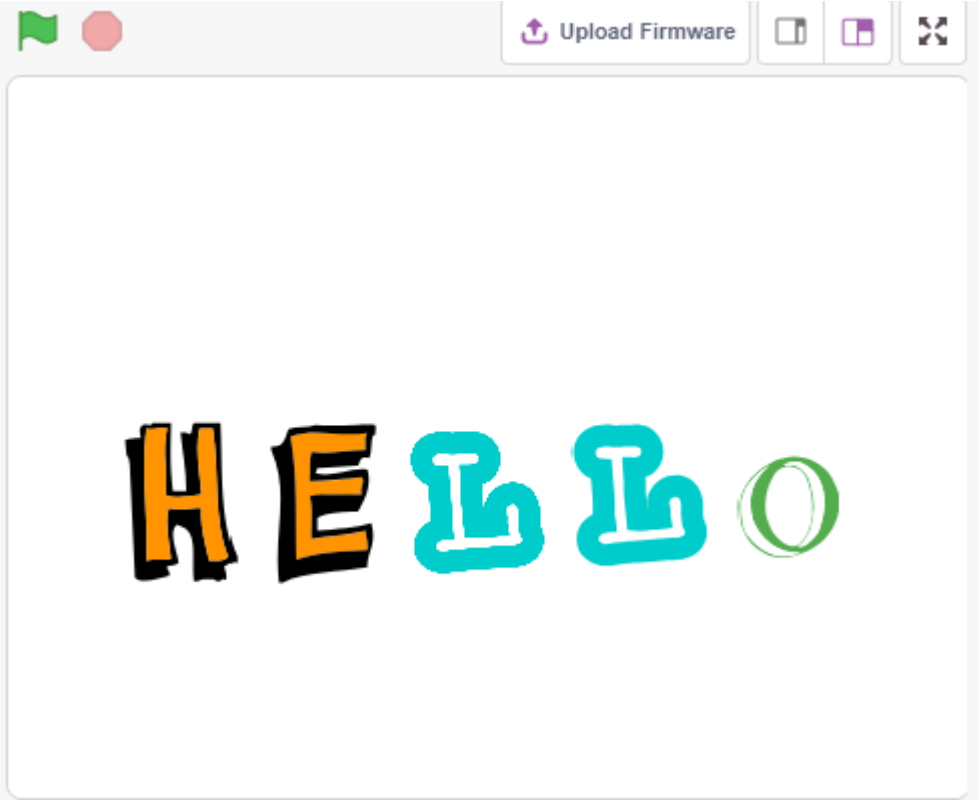

**6. Ball5 スプライト（紫）**




## 7.7 2.4 LCD1602

LCD1602 は 2x16 文字を表示することができます。今回、ステージ上のキャラクタースプライトと対応する文字を表示させます。

ステージ上で「Hello」と一つずつクリックすると、異なるアニメーション効果が現れ、同時に LCD1602 に文字が表示されます。




### 7.7.1 学べること

- LCD1602 の使用方法
- 複数の異なるスプライトの選択
- スプライトのサイズ、回転角度、色の変更、表示・非表示の設定

### 7.7.2 必要な部品

このプロジェクトには以下のコンポーネントが必要です。

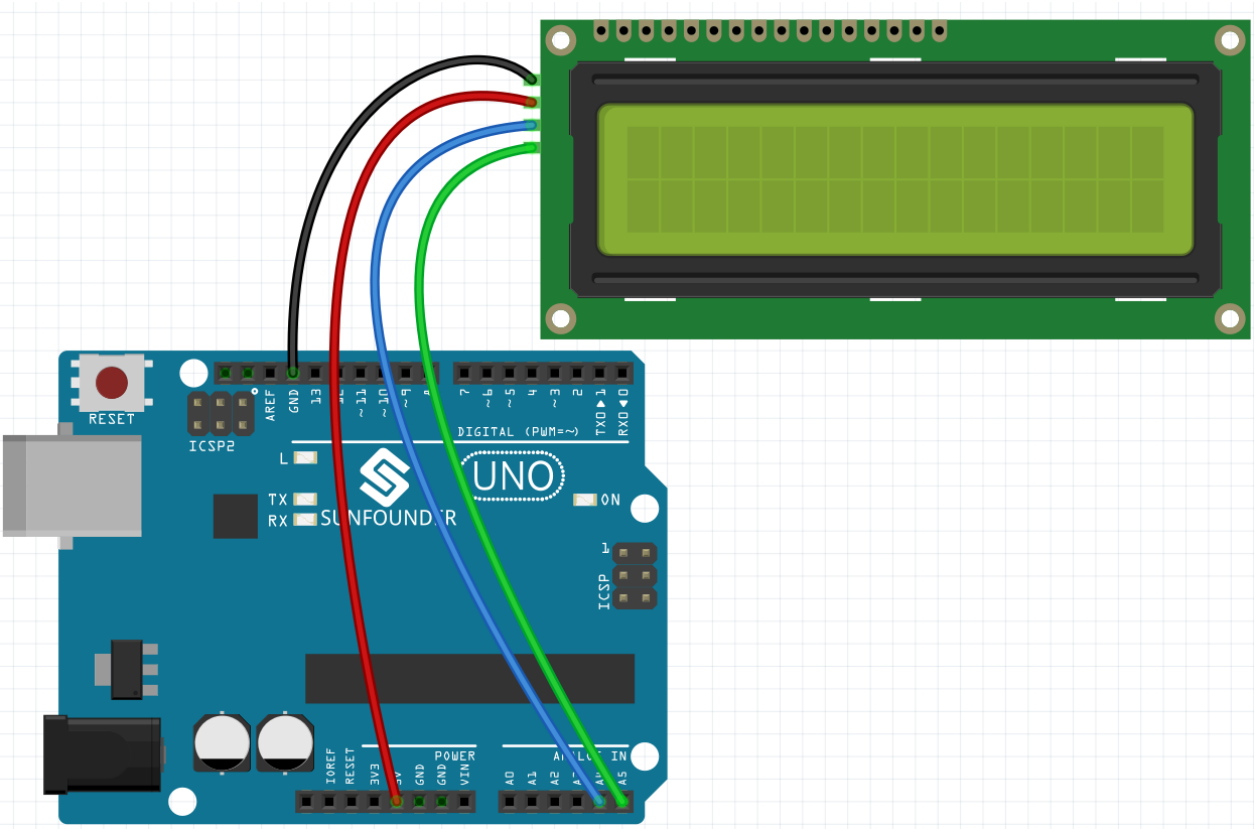
一式を購入すると非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| *I2C LCD1602* | |

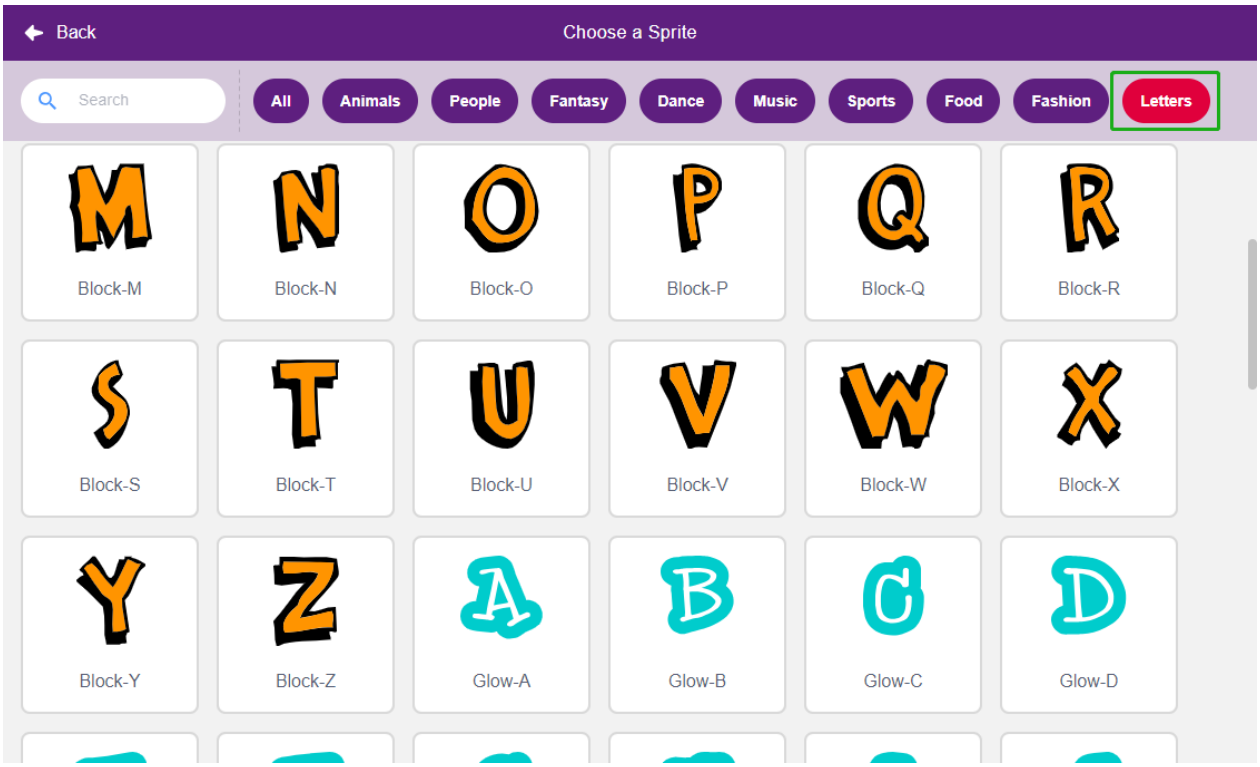
### 7.7.3 回路の作成



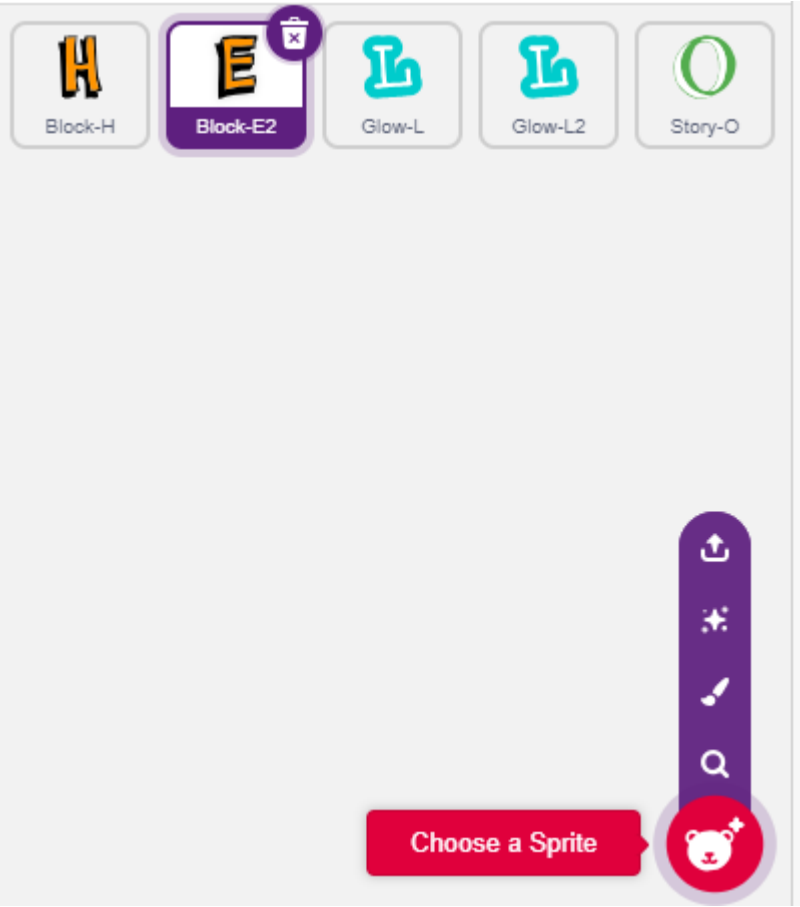

### 7.7.4 プログラミング

#### 1. スプライトを選択

デフォルトのスプライトを削除し、 **Choose a Sprite** をクリックしてから **letters** をクリックし、希望するスプライトを選択します。




例として「Hello」を選択しました。以下に示します。



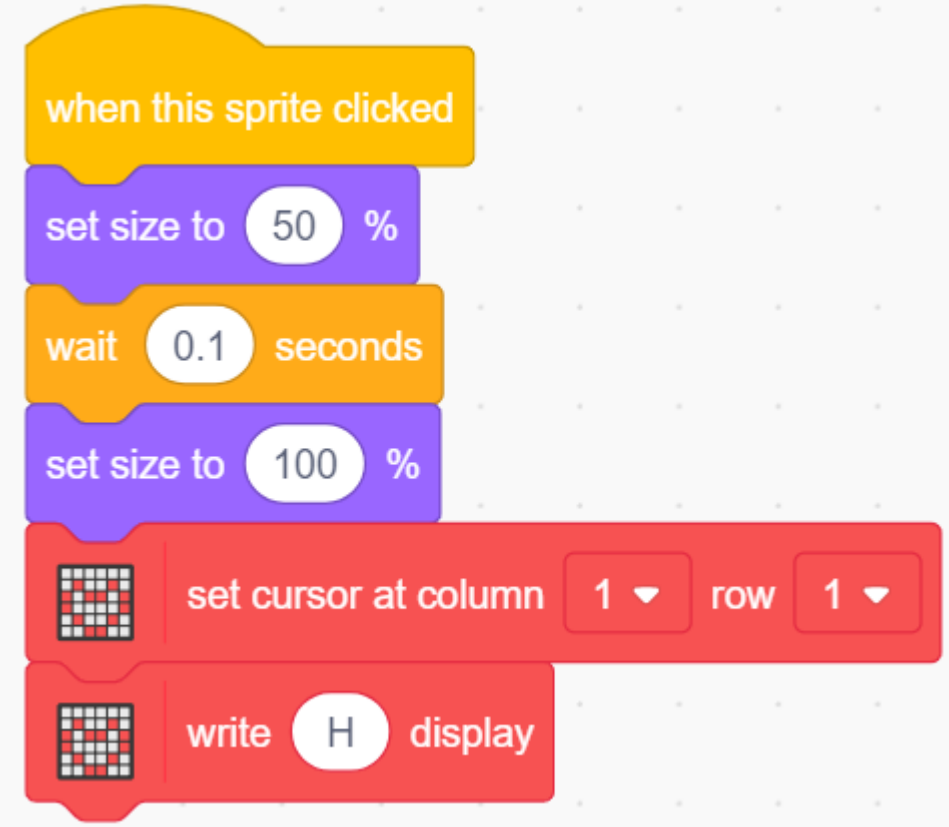

これらのスプライトに異なる効果を設定し、クリックすると同時に LCD1602 に表示します。

**2. H は拡大と縮小**

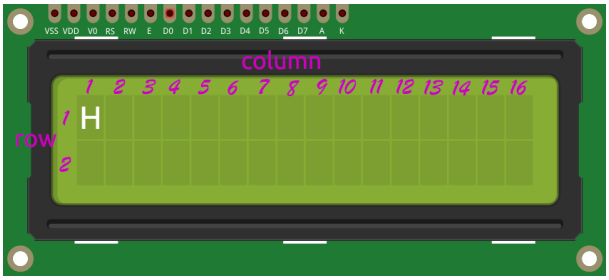
**H** スプライトをクリックし、スクリプトを記述します。

スプライト **H** がクリックされると、そのサイズを 50 %にし、元に戻す。同時に LCD1602 の第 1 行第 1 列に H を表示する。

- スプライトのサイズを設定します： **Looks** パレットからスプライトのサイズを 0% から 100% の間で設定します。
- カーソルをコルマン行に設定]： Set cursor at columan row]: **Display Modules** パレットから、LCD1602 の特定の行にカーソルを設定し、文字の表示を開始します。
- 表示を書き込む]：**Display Modules** パレットから、LCD1602 に文字や文字列を表示するために使用します。




LCD1602 の行と列の分布は、図に示されています。

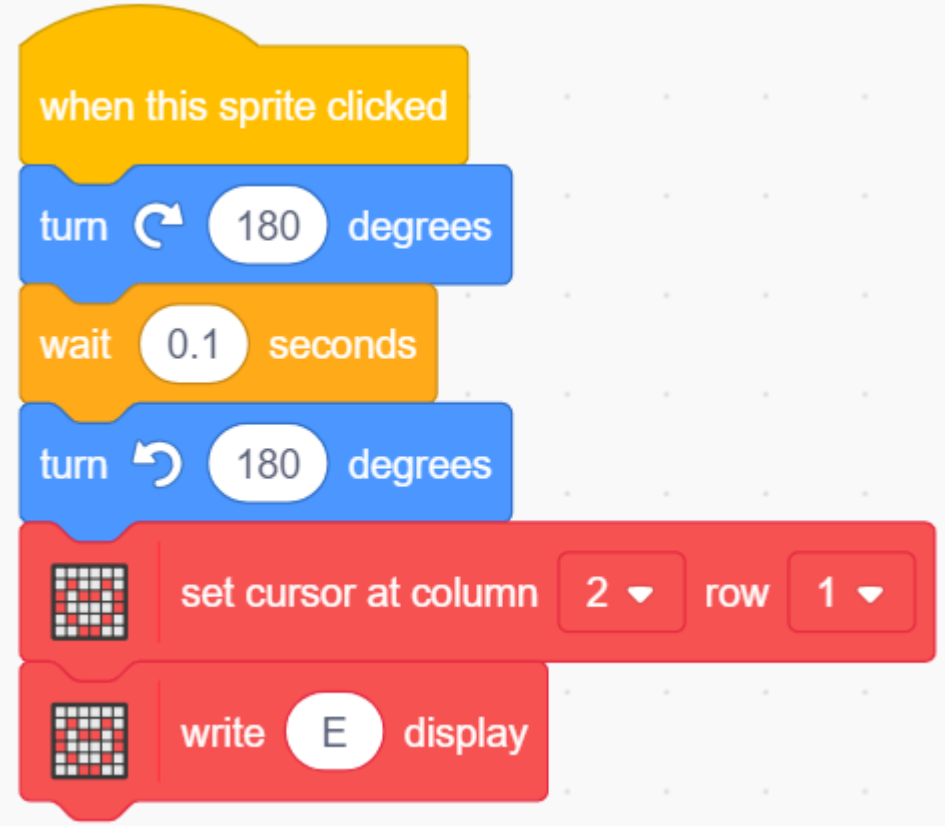



**3. E は左右にフリップ**

**E** スプライトをクリックし、スクリプトを記述します。

スプライト **E** がクリックされたら、180 度時計回りに回転し、180 度反時計回りに回転して左右にフリップさせる。LCD1602 の第 1 行第 2 列に H を表示する。

- 回転度]： **Motions** パレットから、スプライトを時計回りまたは反時計回りに回転させるために使用します。

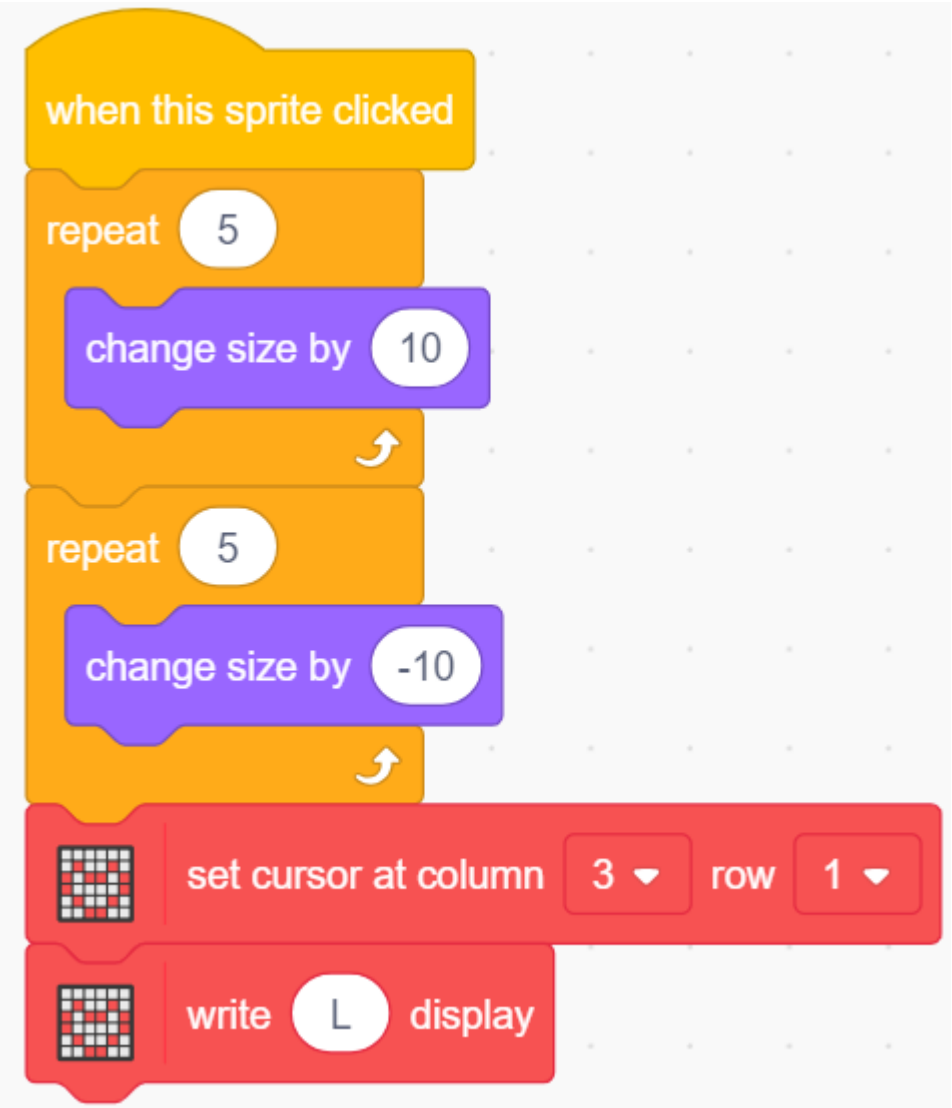



**4. L は徐々に縮小して拡大**

**first L** スプライトをクリックし、スクリプトを記述します。

スプライト **L** がクリックされると、[repeat] ブロックを使用してサイズを 50 %増加させ（5 回、各 10 回）、同じ方法で元のサイズに縮小します。LCD1602 の第 1 行第 3 列に L を表示する。

- サイズ変更 モーションパレットから、スプライトのサイズを変更するために使用します。

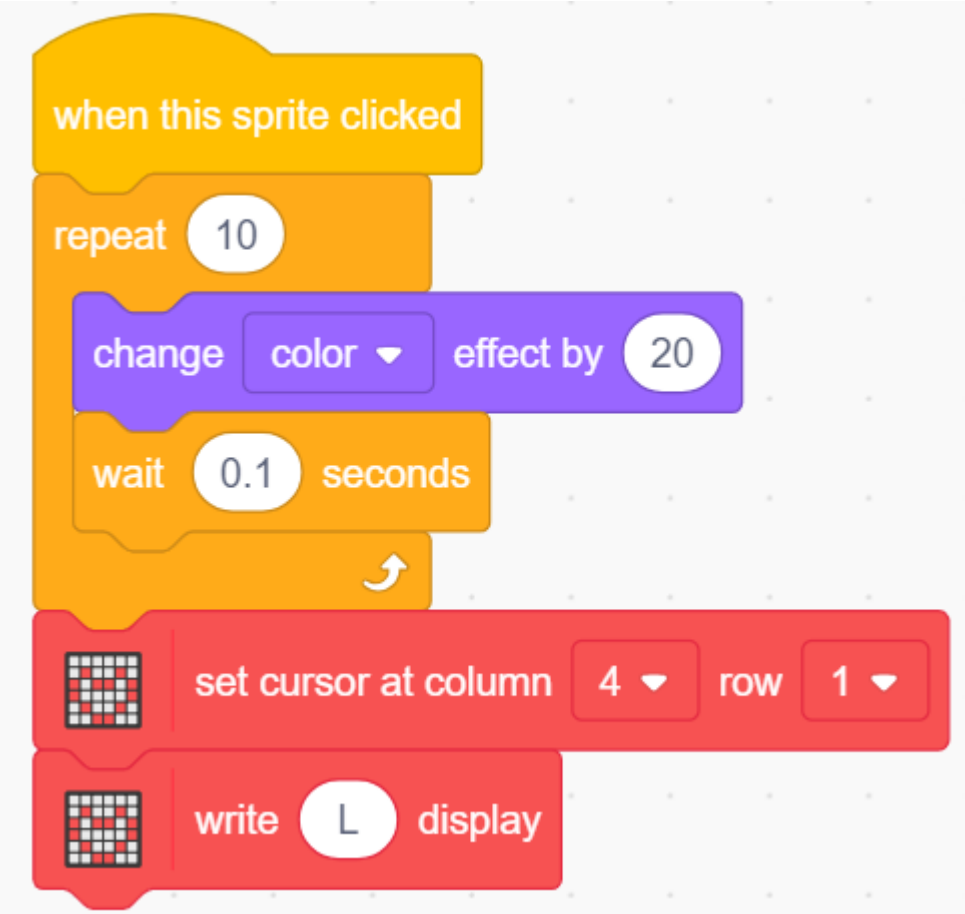



#### 5. 2 番目の L は色を変える

**second L** スプライトをクリックして、スクリプトを記述します。

スプライト **L** がクリックされたら、[repeat] ブロックを使用して、20 の増分で 10 回色を変更し、元の色に戻します。LCD1602 の第 1 行第 4 列に L を表示する。

- [change color effect by]: によるカラーエフェクトの変更： 色効果を変更するために使用します。1 つのコスチュームは色効果を使用して 200 の異なる色調を取ることができます。

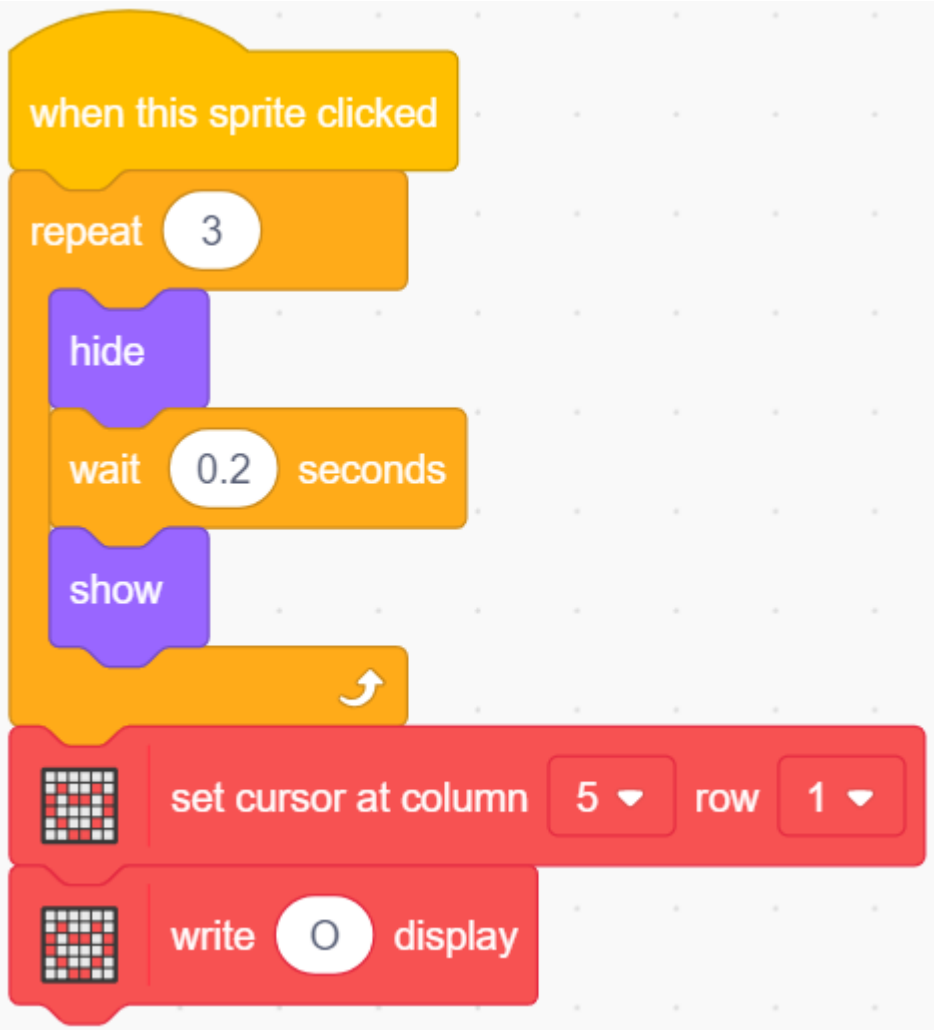



**6. O は非表示と表示**

**O** スプライトをクリックし、スクリプトを記述します。

**O** スプライトがクリックされると、非表示と表示のプロセスを 3 回繰り返し、LCD1602 の第 1 行第 5 列に O を表示する。

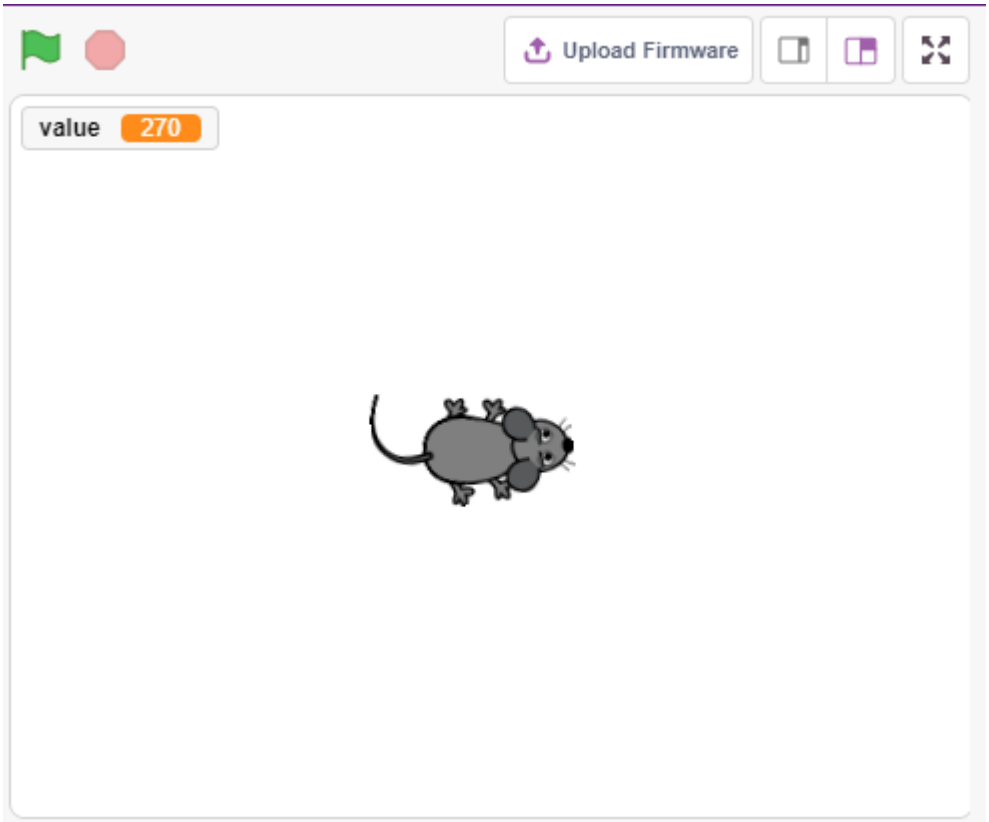
- [Hide] と [Show]：スプライトを隠したり見せたりします。




## 7.8 2.5 移動するマウス

今日は、ポテンショメーターで制御されるマウスのおもちゃを作成します。

緑の旗がクリックされたとき、ステージ上のマウスが前進し、ポテンショメーターを回すと、マウスが移動の方向を変えます。




### 7.8.1 学べること

- ポテンショメーターの原理
- アナログピンと範囲の読み取り
- ある範囲を別の範囲にマッピングする
- スプライトの移動と方向変更

### 7.8.2 必要な部品

このプロジェクトでは、以下の部品が必要です。

全てのキットを購入することは確かに便利です、リンクはこちらです：

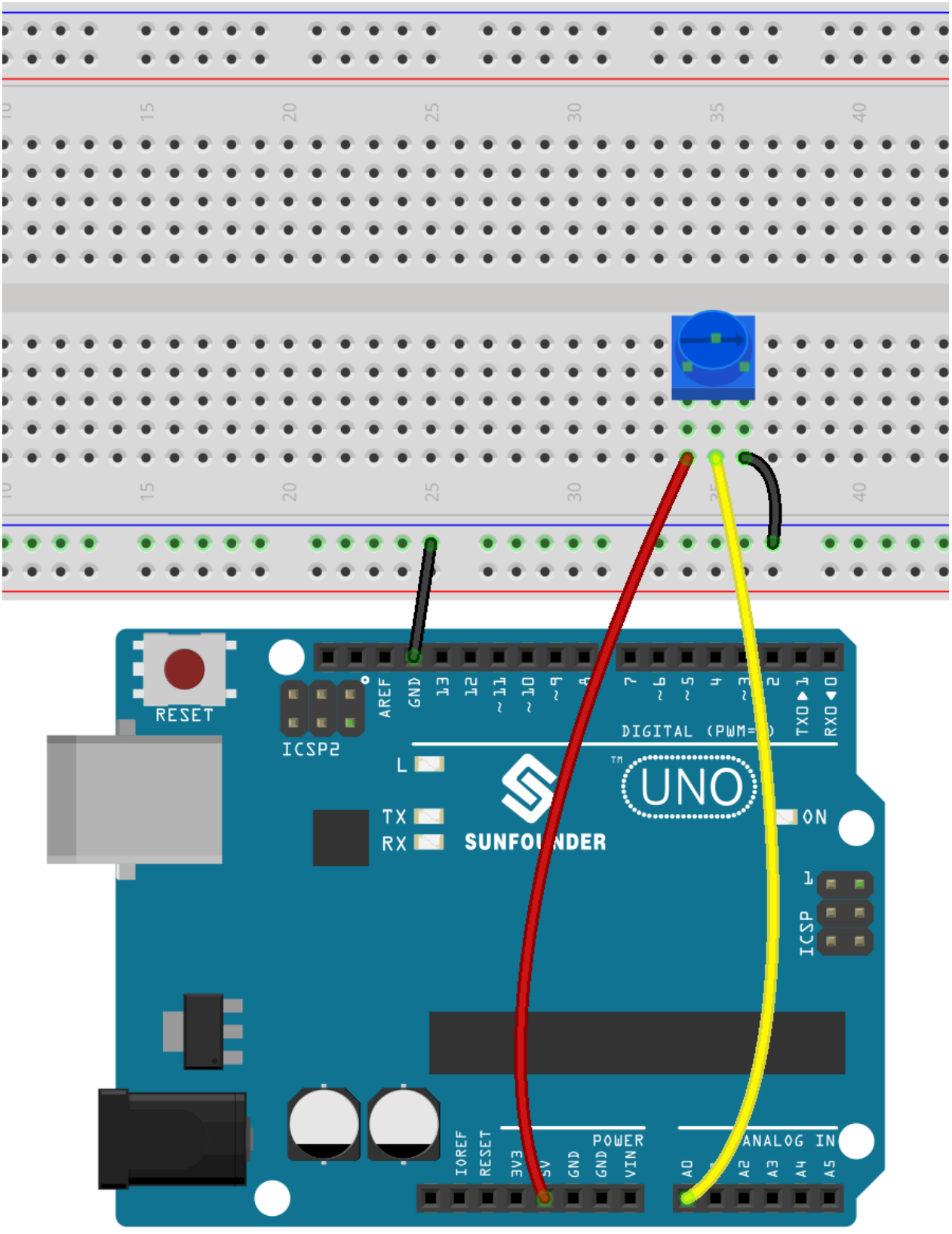
| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| ポテンショメータ | |

### 7.8.3 回路の作成

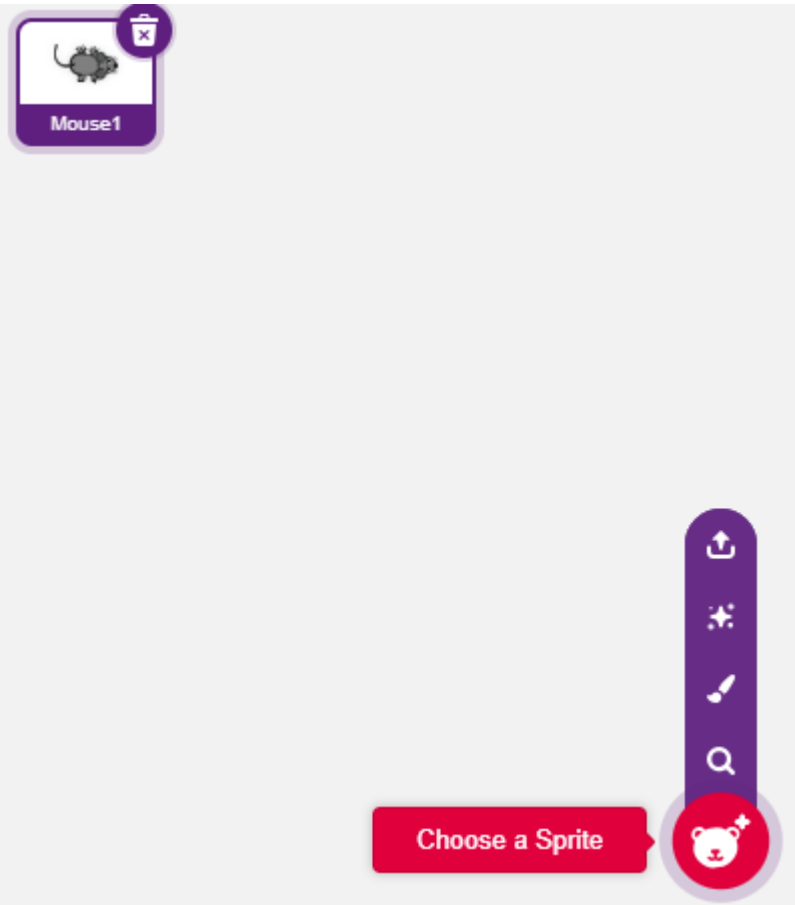
ポテンショメーターは 3 端子の抵抗素子で、2 つの側面ピンは 5V と GND に接続され、中央のピンは A0 に接続されます。Arduino ボードの ADC コンバーターによる変換後、値の範囲は 0-1023 です。


RESET
ICSP2
L
TX
RX
ON
ICSP
AREF
GND
DIGITAL (PWM=~)
TXO▲1
RXO◀0
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A2
A3
A4
A5

### 7.8.4 プログラミング

#### 1. スプライトを選択する

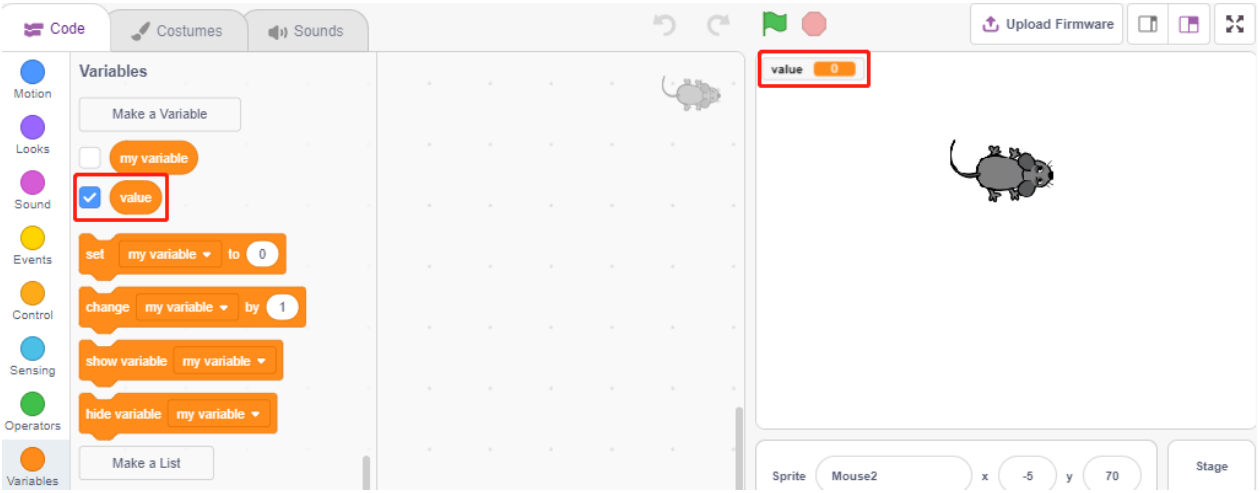
デフォルトのスプライトを削除し、スプライトエリアの右下にある **Choose a Sprite** ボタンをクリックします。検索ボックスに **mouse** を入力し、それをクリックして追加します。




#### 2. 変数の作成

ポテンショメータの読み取り値を保存するための変数 **value** を作成します。

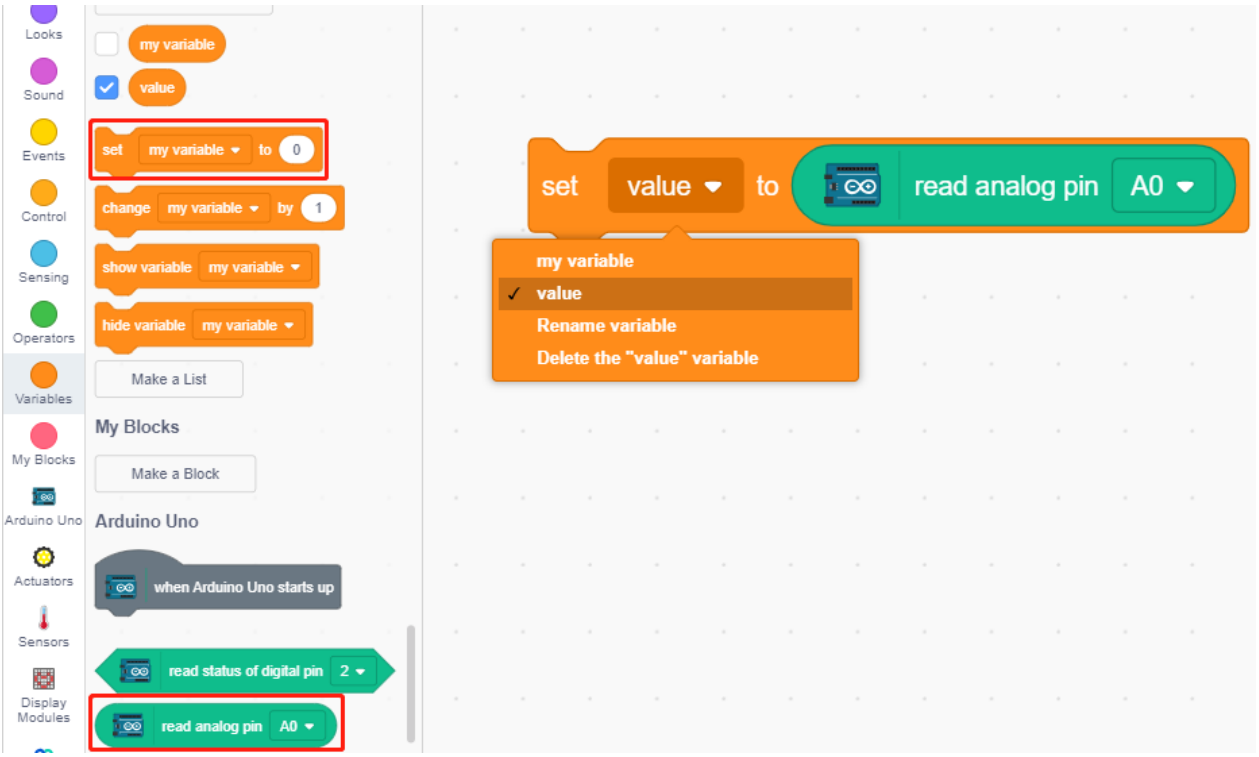
作成すると、 **value** が **Variables** パレット内に表示され、チェックされた状態で表示されます。これは、この変数がステージに表示されることを意味します。




### 3. A0 の値を読み取る

A0 の読み取り値を変数 **value** に保存します。

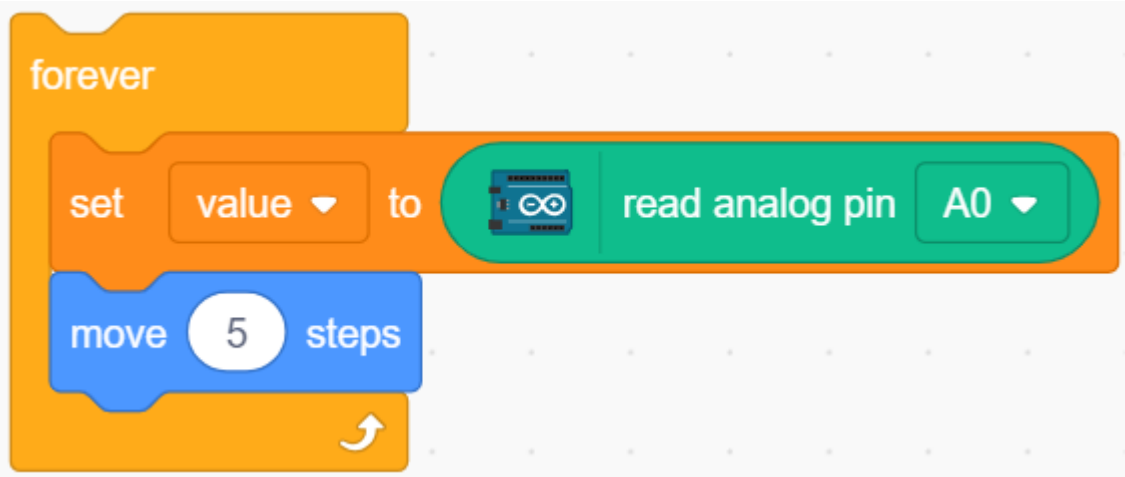
- [set my variable to 0]: 変数の値を設定します。
- [read analog pin A0]: A0~A5 の値を 0-1023 の範囲で読み取ります。




全体を読むためには、[forever] ブロックを使用する必要があります。このスクリプトをクリックして実行し、ポテンショメーターを両方向に回転させると、値の範囲が 0-1023 であることがわかります。


#### 4. スプライトを移動する

[move steps] ブロックを使用してスプライトを移動します。スクリプトを実行すると、スプライトが中央から右に移動するのが見えます。

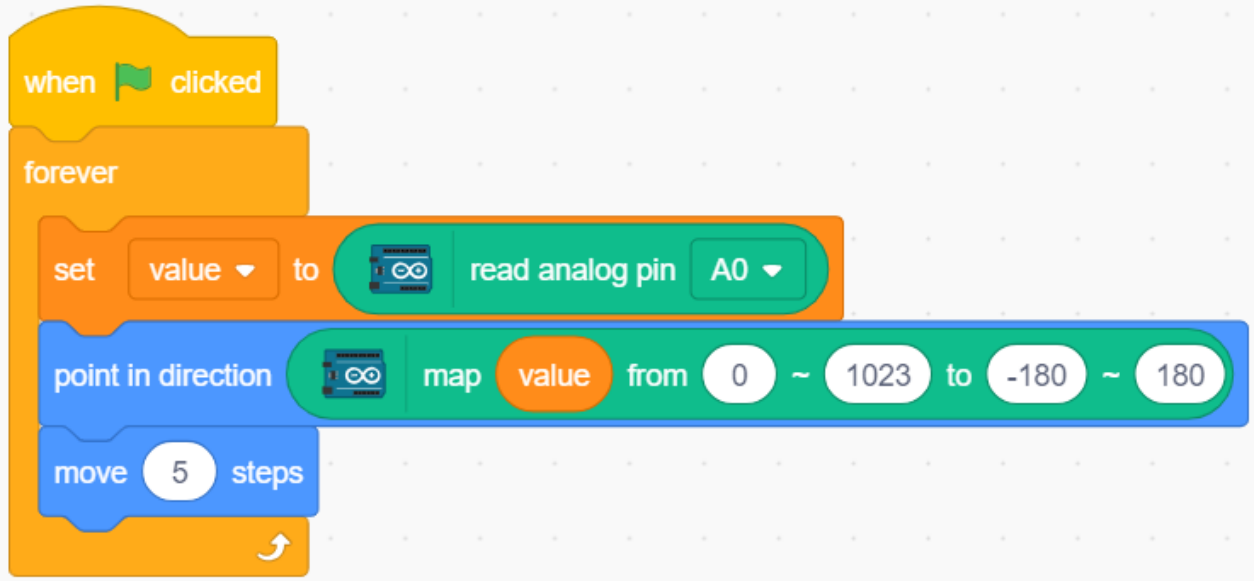



#### 5. スプライトの方向を変える

A0 の値によってスプライトの移動方向を変えます。A0 の値の範囲は 0-1023 ですが、スプライトの回転方向は-180~180 ですので、[map] ブロックを使用する必要があります。

また、スクリプトの開始時に [when green flag clicked] を追加します。

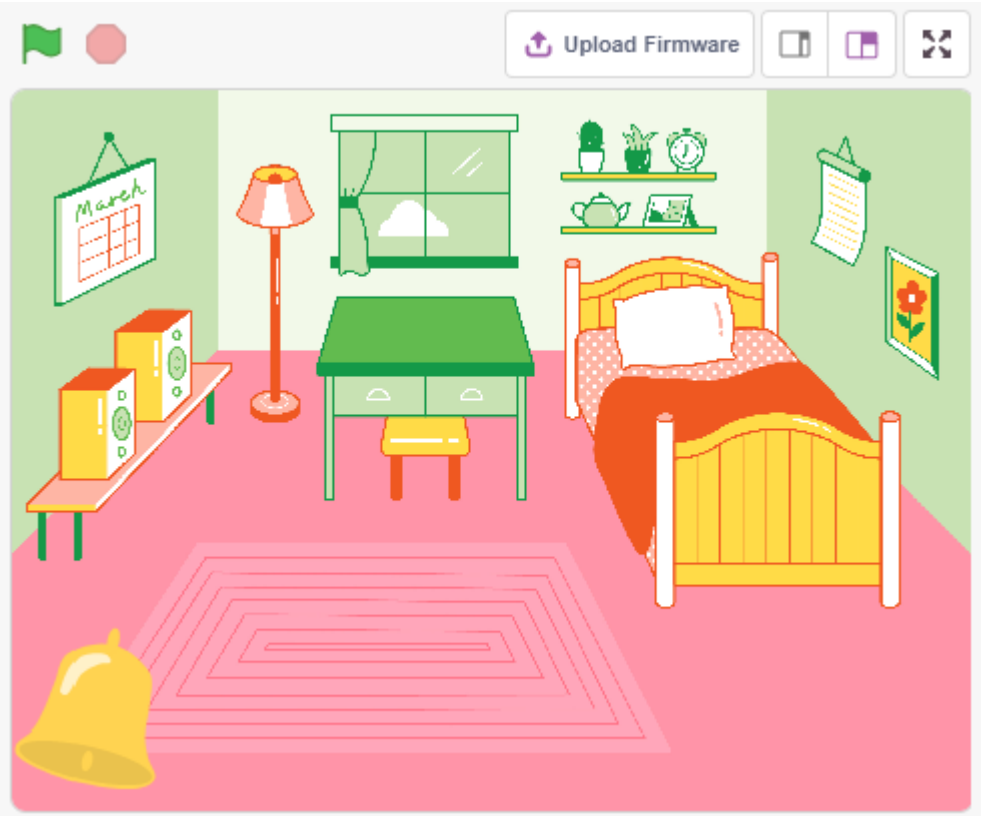
- [point in direction]: スプライトの操舵角を設定します。 **Motion** パレットから。
- [map from to]: ある範囲を別の範囲にマッピングします。




## 7.9 2.6 ドアベル

このプロジェクトでは、ステージ上のボタンとベルを使ってドアベルを作成します。

緑のフラグがクリックされた後、ボタンを押すとステージ上のベルが音を鳴らします。



### 7.9.1 学べること

- ボタンの動作方法
- デジタルピンと範囲の読み取り
- 条件ループの作成
- 背景の追加
- 音の再生

### 7.9.2 必要な部品

このプロジェクトには以下の部品が必要です。

全体のキットを購入するのが便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

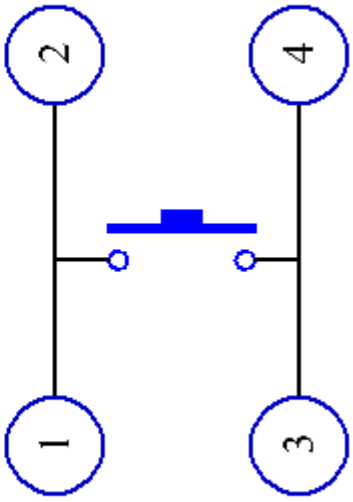
以下のリンクから個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| ボタン | |
| コンデンサ | |

### 7.9.3 回路の作成

ボタンは 4 ピンのデバイスであり、ピン 1 がピン 2 に、ピン 3 がピン 4 に接続されています。ボタンが押されると、4 つのピンが接続され、回路が閉じます。

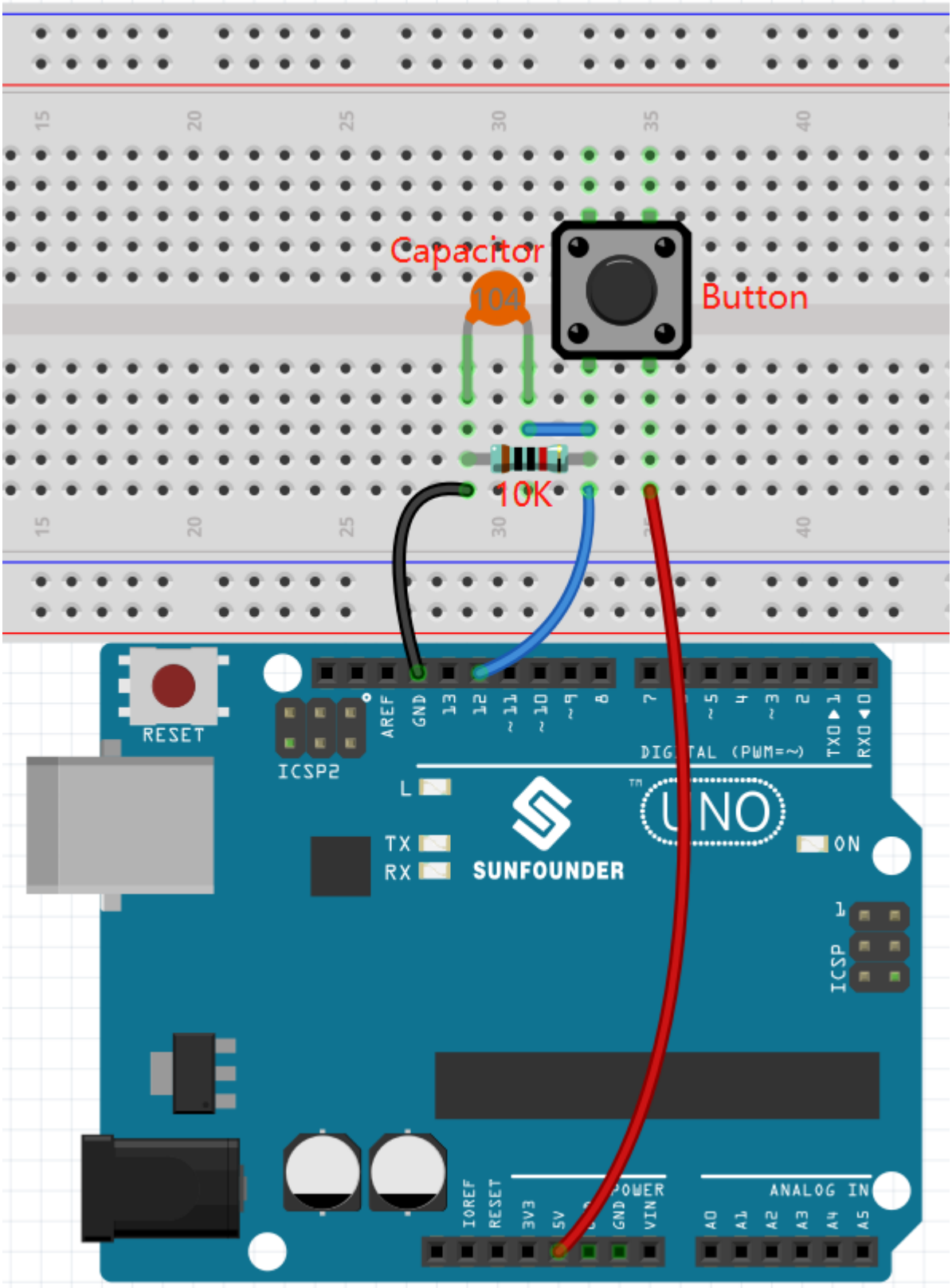

Button




Internal Structure

以下の図に従って回路を組み立てます。

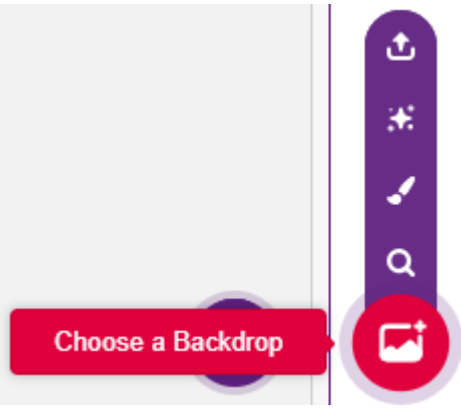
- ボタンの左側のピンの一つを、プルダウン抵抗と 0.1uF（104）キャパシター（ボタン動作時のジッタを除去し、ボタンが動作するときの安定したレベルを出力するため）に接続されているピン 12 に接続します。
- 抵抗とキャパシターのもう一方の端子を GND に、ボタンの右側のピンの一つを 5V に接続します。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
5
4
~ 3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

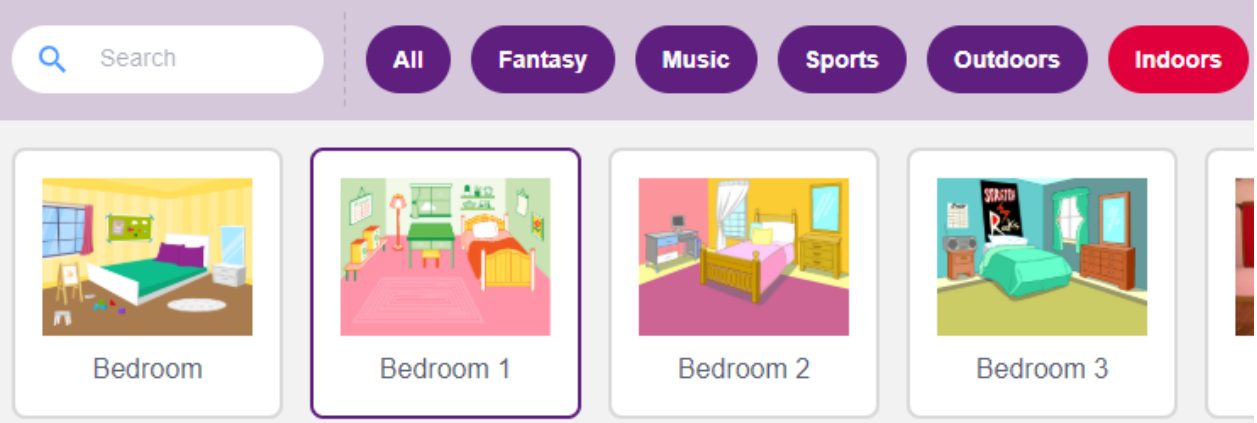
### 7.9.4 プログラミング

#### 1. 背景を追加する

右下の **Choose a Backdrop** ボタンをクリックします。

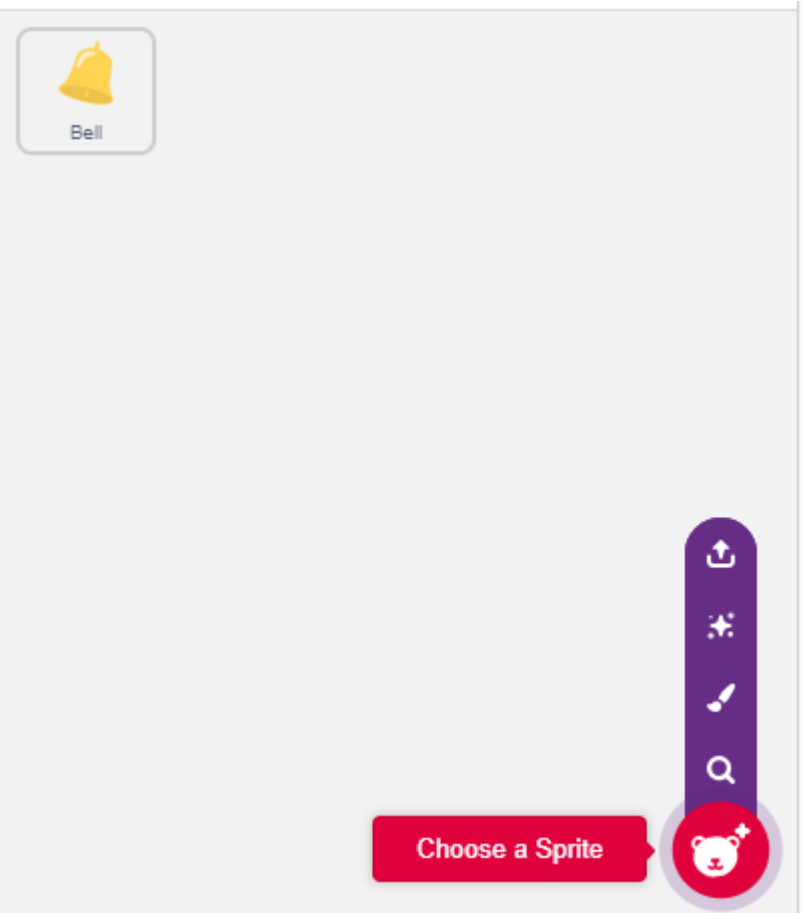



**Bedroom 1** を選択します。

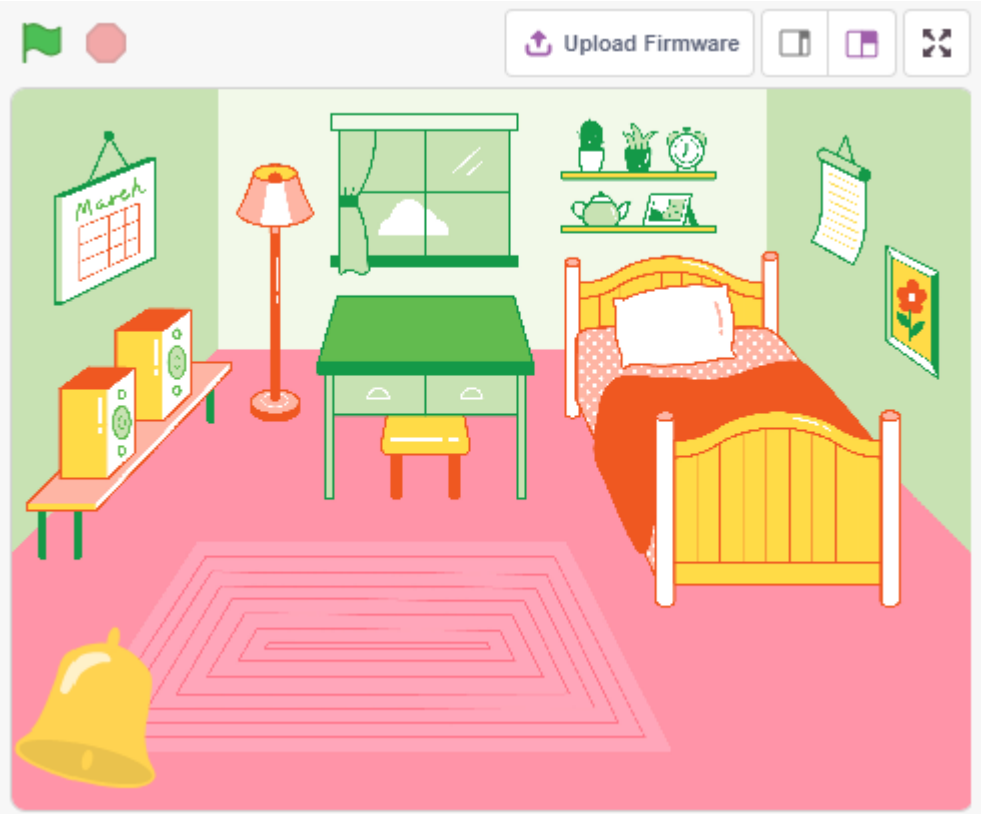



#### 2. スプライトを選択する

デフォルトのスプライトを削除し、スプライトエリアの右下の **Choose a Sprite** ボタンをクリックして、検索ボックスに **bell** を入力し、それを追加します。



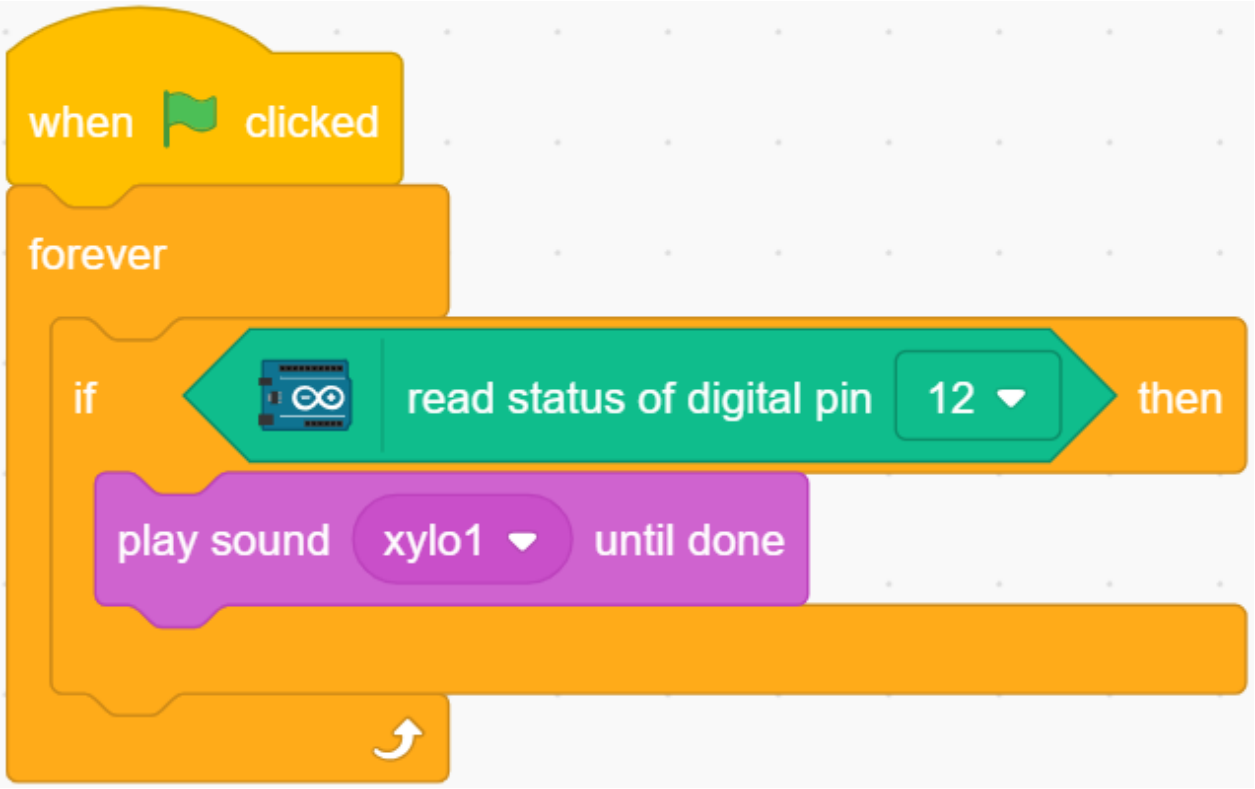

ステージ上の **bell** スプライトを選択し、正しい位置に移動します。




### 3. ボタンを押すとベルが音を鳴らす

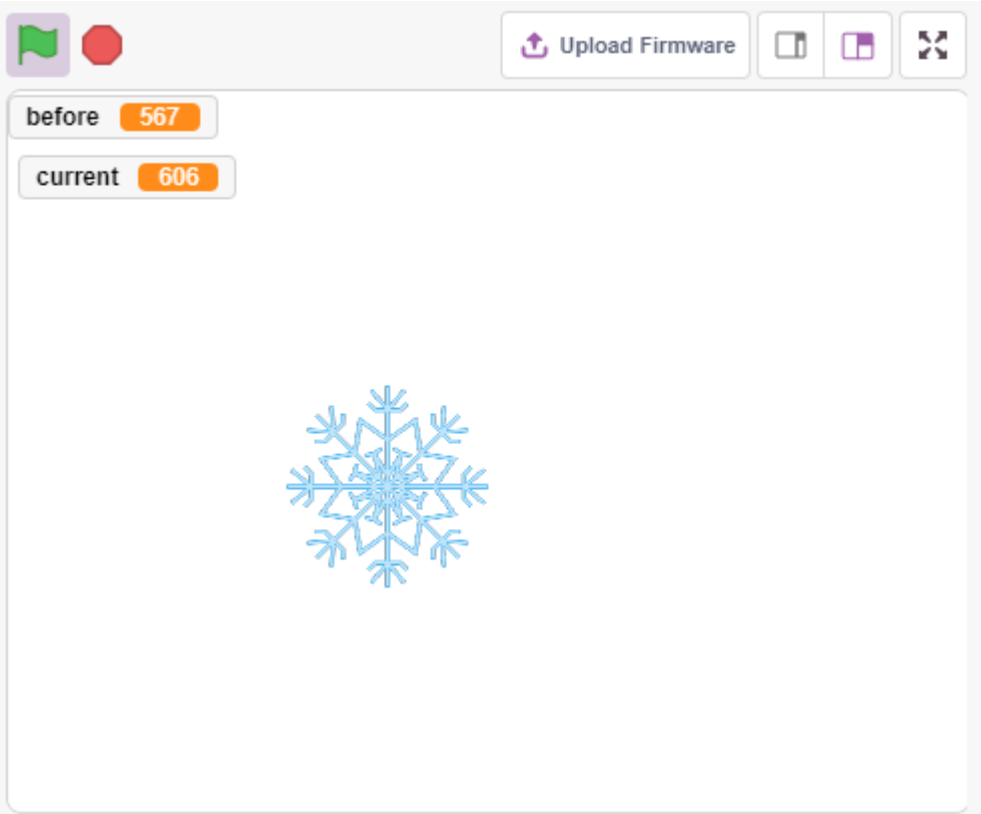
[if then] を使用して条件文を作成します。ピン 12 の読み取り値が 1（キーが押されている）の場合、 **xylo1** の音が再生されます。

- [read status of digital pin]: このブロックは **Arduino Uno** パレットから来ており、デジタルピンの値を読むために使用され、結果は 0 または 1 です。
- [if then]: このブロックは制御ブロックであり、 **Control** パレットから来ています。そのブーリアン条件が真である場合、その中に保持されているブロックが実行され、その後関与するスクリプトが続行されます。条件が偽の場合、ブロック内のスクリプトは無視されます。条件は一度だけチェックされます。ブロック内のスクリプトが実行されている間に条件が偽に変わっても、完了するまで実行し続けます。
- [play sound until done]: Sound パレットから、特定の音を再生するために使用されます。




## 7.10 2.7 低温警報

このプロジェクトでは、低温警報システムを作成します。温度がしきい値を下回ると、ステージに **Snowflake** スプライトが表示されます。




### 7.10.1 学べること

- サーミスタの動作原理
- 多変量および減算操作

### 7.10.2 必要な部品

このプロジェクトには、以下のコンポーネントが必要です。

キット全体を購入することは確かに便利です。こちらのリンクを参照してください：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| サーミスタ | |

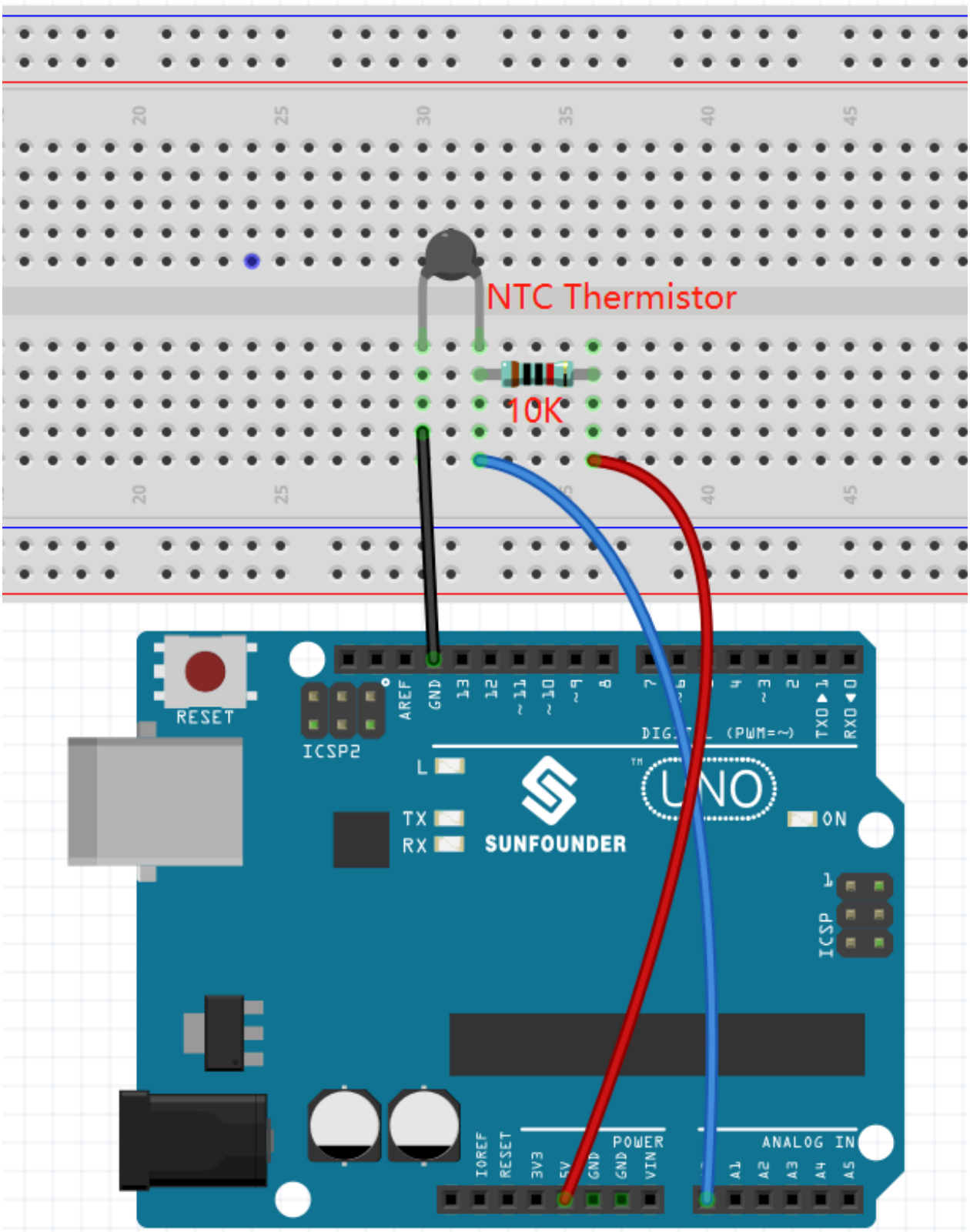
### 7.10.3 回路の作成

サーミスタは、標準的な抵抗よりも、温度に強く依存する抵抗のタイプで、PTC（温度が上がると抵抗が上がる）と NTC（温度が上がると抵抗が下がる）の 2 種類の抵抗があります。

以下の図に従って回路を組み立ててください。

サーミスタの一端は GND に接続され、もう一端は A0 に接続され、10K の抵抗が 5V に直列に接続されています。

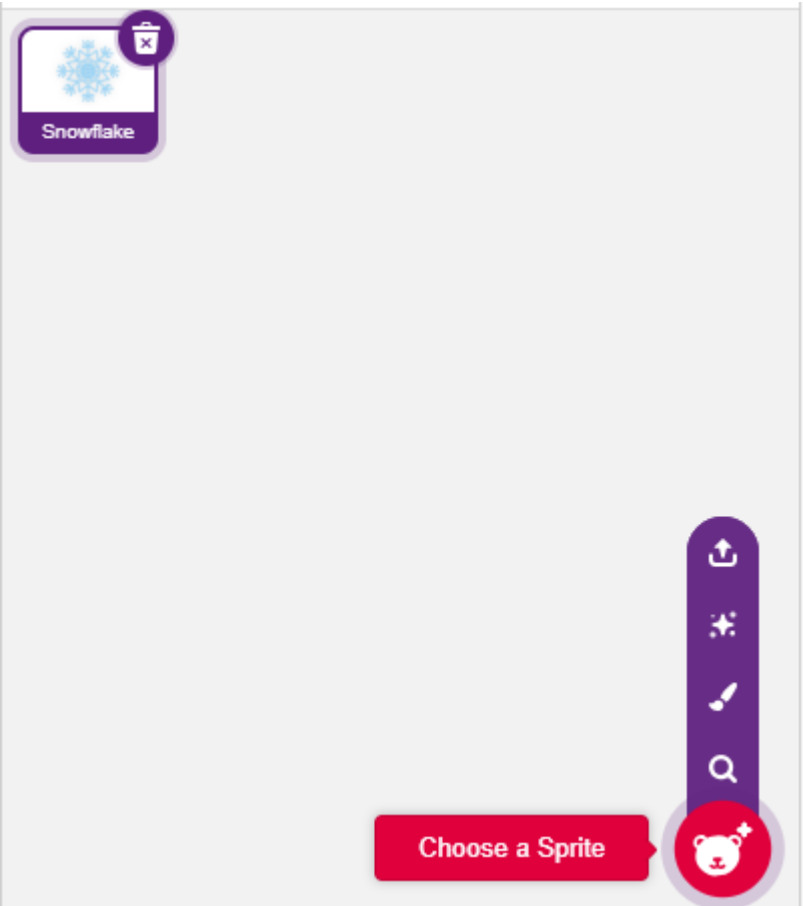
ここでは NTC サーミスタが使用されているため、温度が上昇するとサーミスタの抵抗が減少し、A0 の電圧分割が減少し、A0 から取得される値が減少し、逆に増加します。


NTC Thermistor
10K
RESET
ICSP2
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
ANALOG IN
AREF
GND
IOREF
RESET
3V3
5V
VIN
A1
A2
A3
A4
A5
TXD
RXD

### 7.10.4 プログラミング

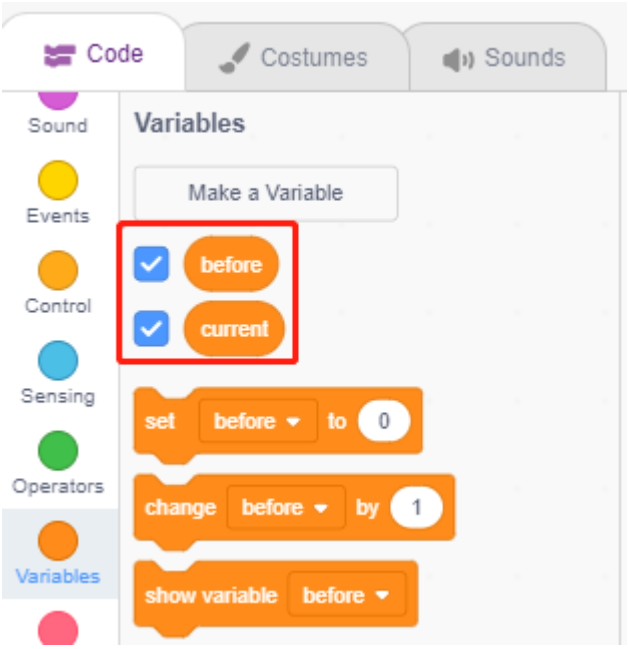
#### 1. スプライトを選択する

デフォルトのスプライトを削除し、スプライトエリアの右下の **Choose a Sprite** ボタンをクリックして、検索ボックスに **Snowflake** と入力し、クリックして追加します。




#### 2. 変数を 2 つ作成する

A0 の値を異なるケースで保存するための 2 つの変数、 **before** と **current** を作成します。

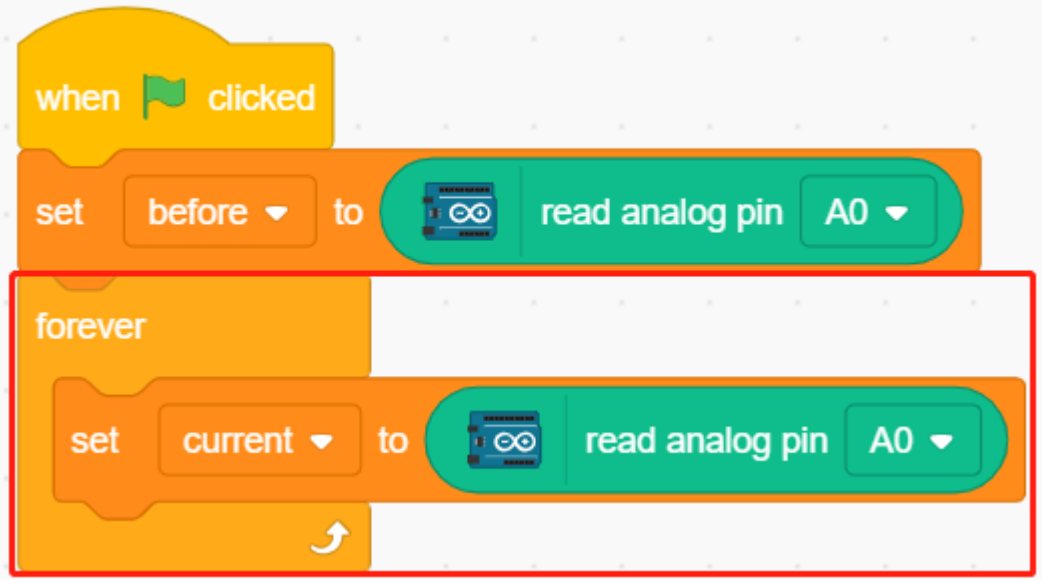



**3. A0 の値を読む**

緑の旗がクリックされたとき、A0 の値が読み取られ、変数 **before** に保存されます。


**4. A0 の値を再び読む**

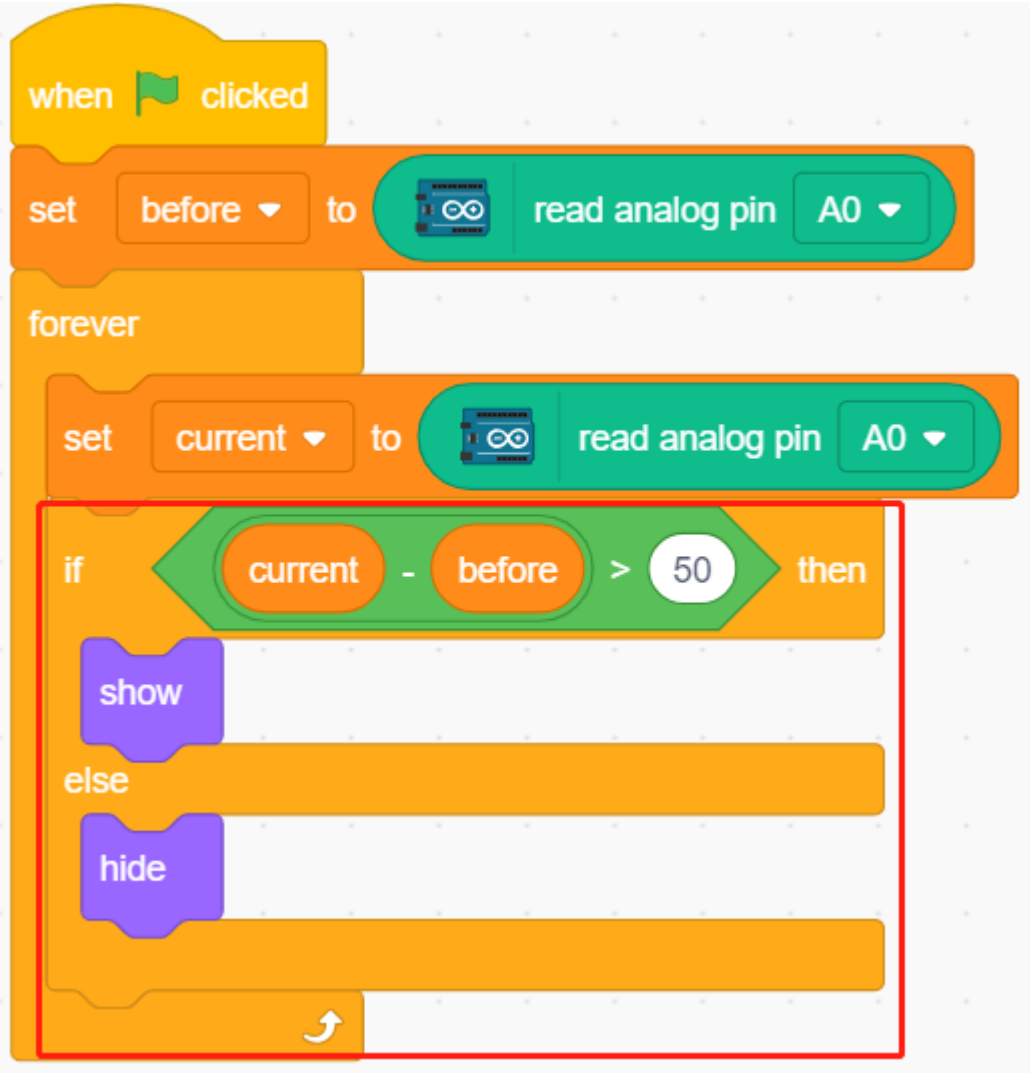
[forever] の中で、A0 の値を再び読み取り、変数 **current** に保存します。




**5. 温度変化を判断する**

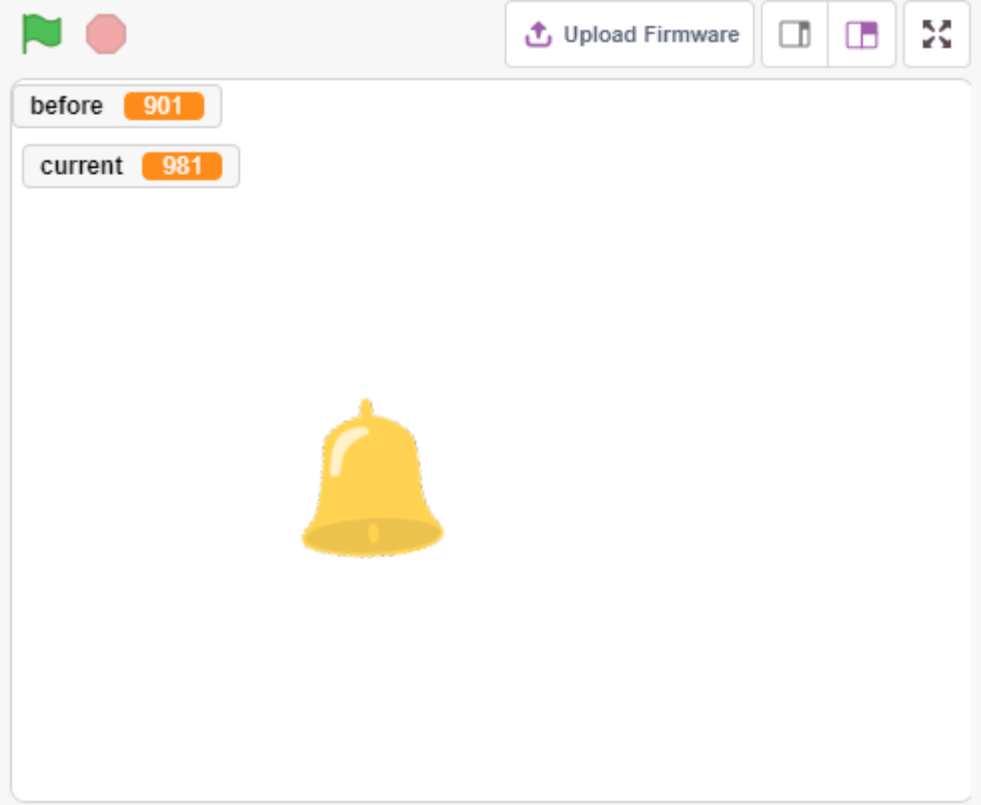
[if else] ブロックを使用して、A0 の現在の値が before よりも 50 大きいかどうかを判断します。これは温度の低下を表します。この時点で **Snowflake** スプライトを表示させ、それ以外の場合は非表示にします。

- [-] & [>]: **Operators** パレットからの減算および比較演算子。




## 7.11 2.8 光アラーム時計

私たちの日常生活にはさまざまなアラーム時計があります。今回は、光で制御されるアラーム時計を作成してみましょう。朝になり、明るさが増すと、この光アラーム時計はあなたに起きる時間であることを知らせます。




### 7.11.1 学べること

- フォトレジスタの動作原理
- 音の再生の停止およびスクリプトの実行停止

### 7.11.2 必要な部品

このプロジェクトには、以下のコンポーネントが必要です。

一式をまとめて購入するのはとても便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| フォトレジスタ | |

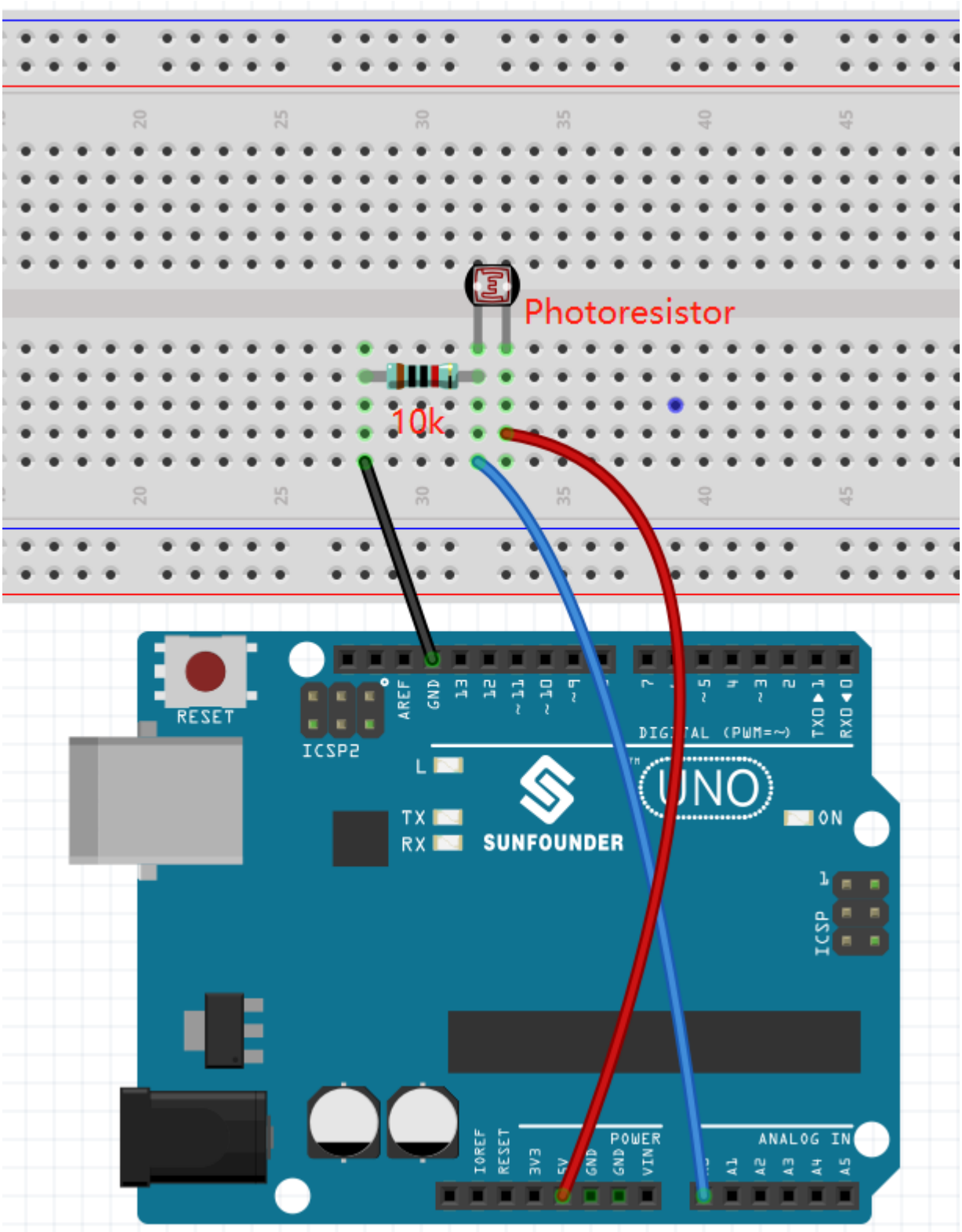
### 7.11.3 回路の作成

フォトレジスタまたはフォトセルは、光によって変わる可変抵抗器です。フォトレジスタの抵抗は、入射光の強度が増加すると減少します。

以下の図に従って回路を組み立てます。

フォトレジスタの一方の端を 5V に、他方の端を A0 に接続し、この端に 10K の抵抗を GND とシリーズに接続します。

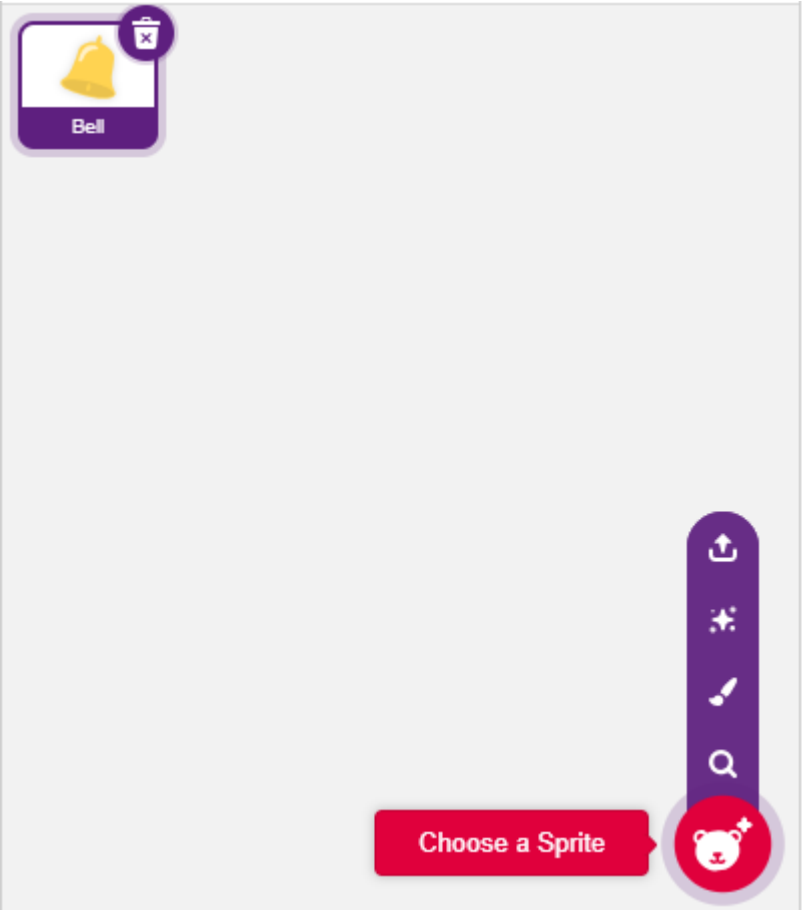
したがって、光の強度が増加すると、フォトレジスタの抵抗が減少し、10K の抵抗の電圧分割が増加し、A0 から得られる値が大きくなります。


Photoresistor
10k
RESET
ICSP2
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
POWER
ANALOG IN

### 7.11.4 プログラミング

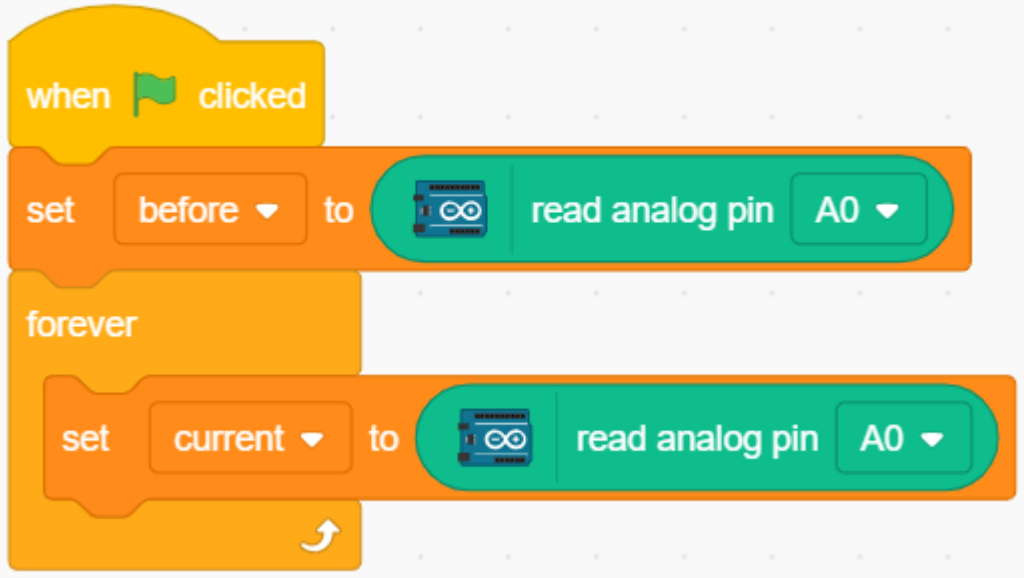
#### 1. スプライトを選択

デフォルトのスプライトを削除し、スプライト領域の右下の **Choose a Sprite** ボタンをクリックし、検索ボックスに **bell** と入力して、それを追加します。



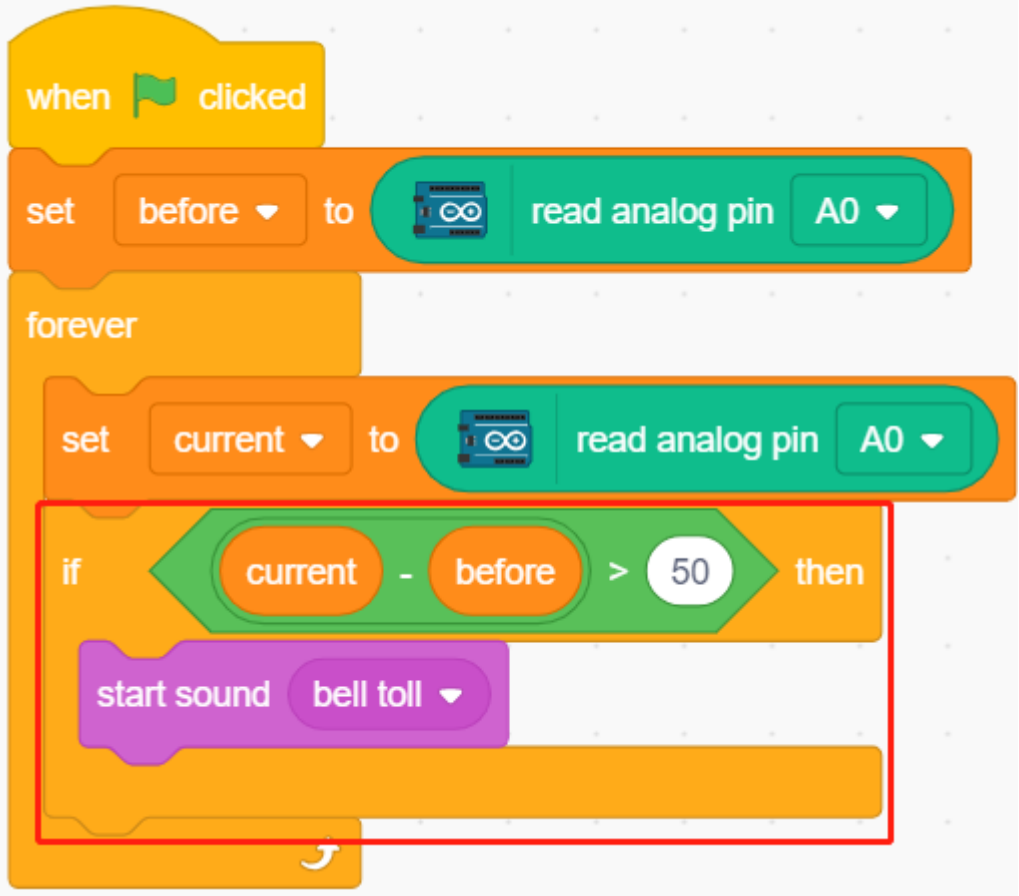

#### 2. A0 の値を読む

**before** と **current** の 2 つの変数を作成します。緑のフラグがクリックされると、A0 の値を読み取り、参照値として変数 **before** に格納します。[forever] の中で、A0 の値を再度読み取り、変数 **current** に格納します。



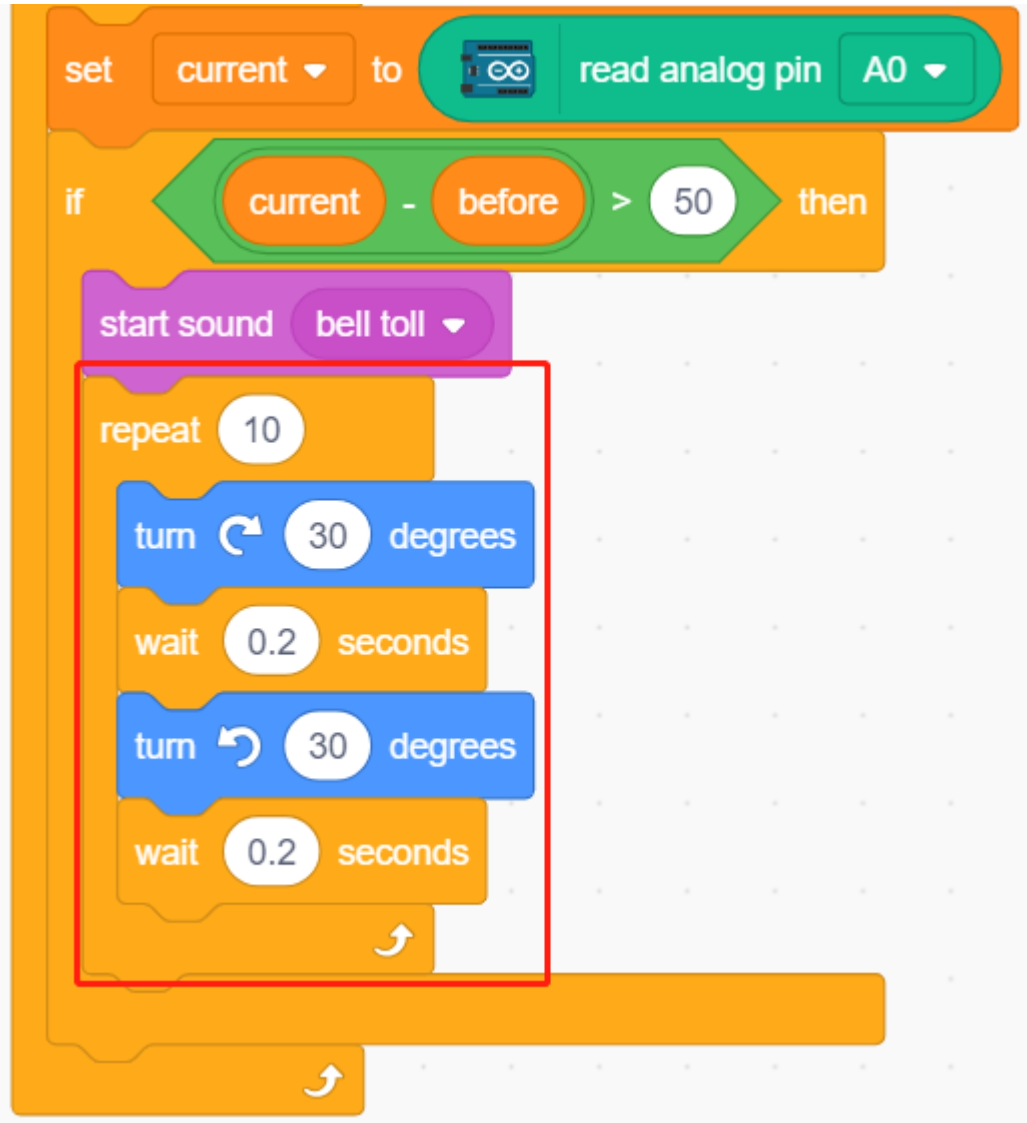

**3. 音を鳴らす**

現在の A0 の値が前の値よりも 50 大きい場合、これは現在の光の強度がしきい値よりも大きいことを示し、その場合にはスプライトに音を鳴らさせます。



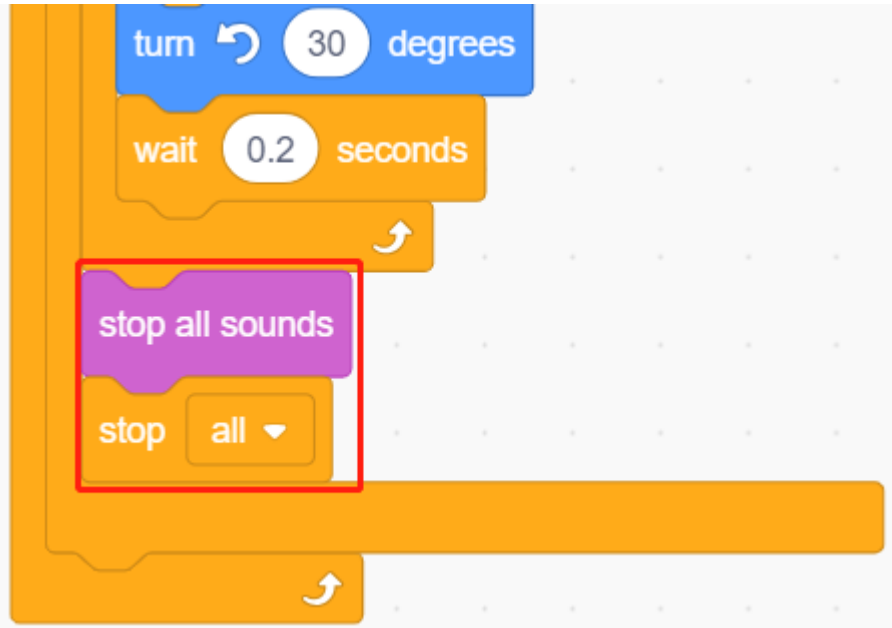
#### 4. スプライトを回転させる

[turn block] を使用して、 **bell** スプライトを左右に回転させ、アラーム効果を実現します。



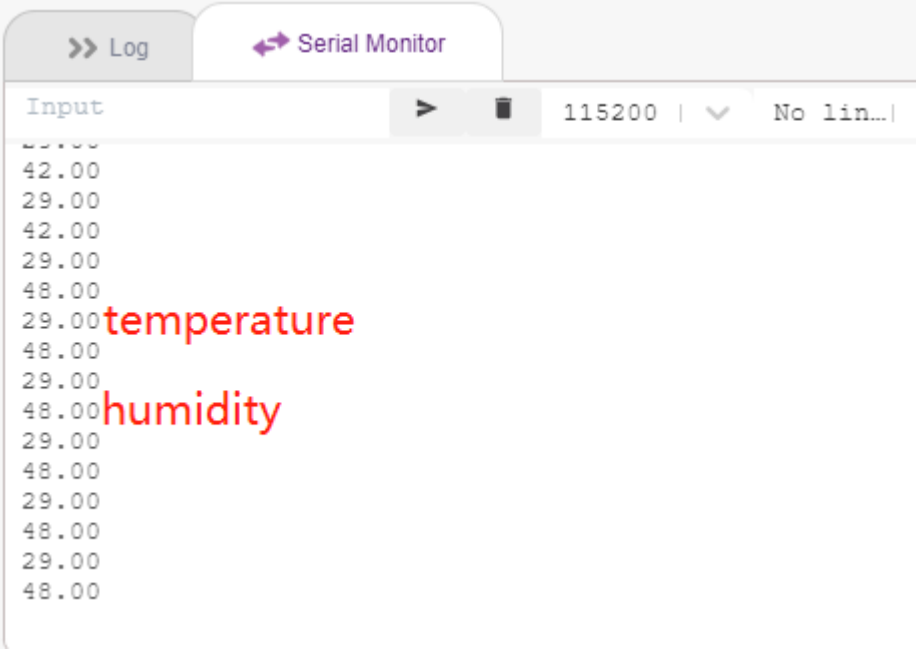

#### 5. すべてを停止

一定の時間アラームが鳴った後、アラームを停止します。




## 7.12 2.9 温湿度の読取り

前のプロジェクトではステージモードを使用していましたが、アップロードモードでしか使用できない機能もあります。例えば、シリアル通信の機能です。このプロジェクトでは、 **アップロードモード** のシリアルモニターを使用して、DHT11 の温湿度を表示します。



### 7.12.1 学べること

- DHT11 モジュールからの温湿度の取得
- **アップロードモード** のためのシリアルモニタ
- 拡張の追加

### 7.12.2 必要な部品

このプロジェクトには以下のコンポーネントが必要です。

一式を購入することは非常に便利です。リンクはこちらです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

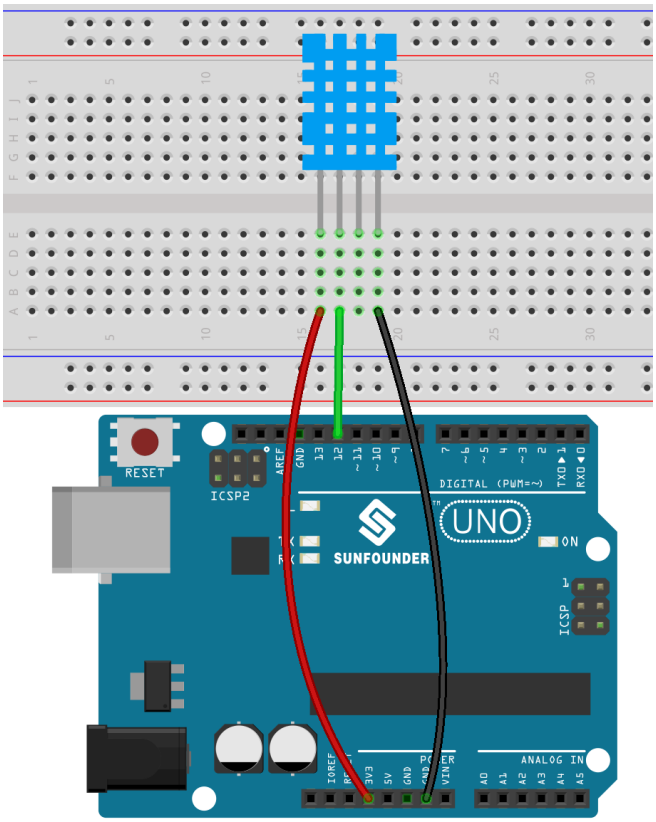
以下のリンクから別々に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| **ブレッドボード** | |
| **ジャンパーワイヤー** | |
| *DHT11* **湿度温度センサ** | - |

### 7.12.3 回路の作成

DHT11 のデジタル温湿度センサーは、温度と湿度の較正されたデジタル信号出力を含む複合センサーです。

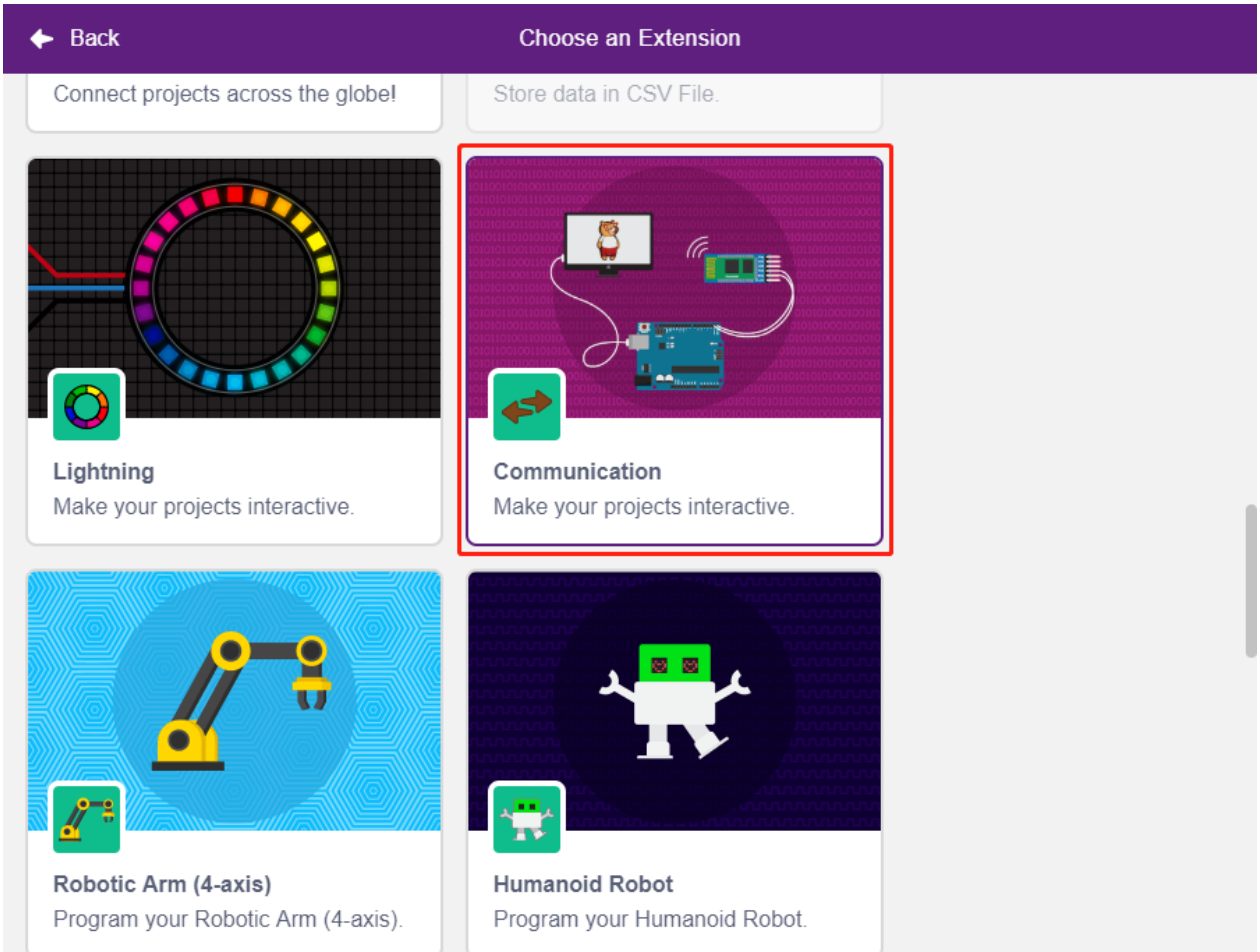
次の図に従って回路を組み立ててください。




### 7.12.4 プログラミング

#### 1. 拡張の追加

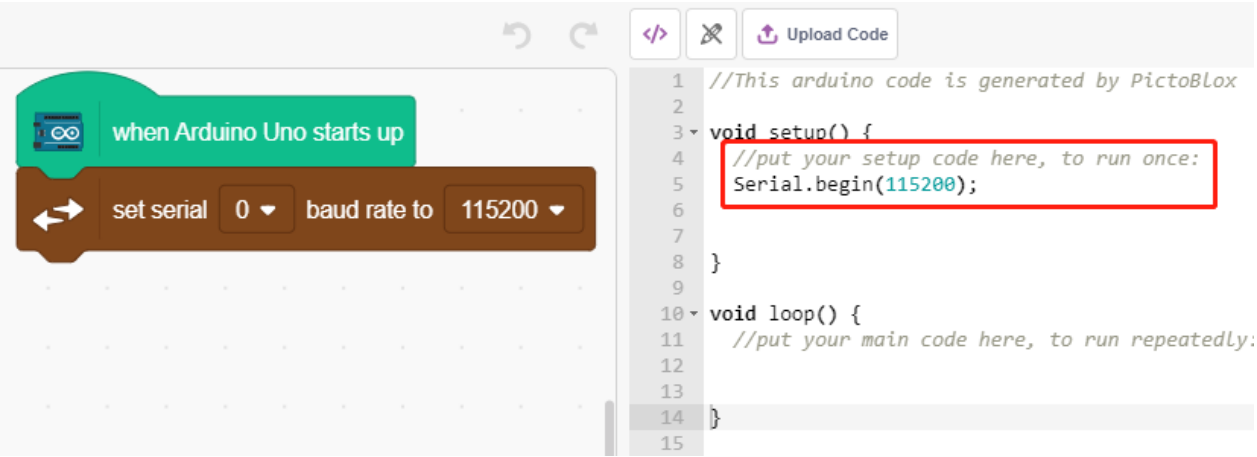
**Upload** モードに切り替え、左下の **Add Extension** ボタンをクリックし、 **Communication** を選択して追加します。それがパレットエリアの最後に表示されます。




**2. Arduino Uno とシリアルモニターの初期化**

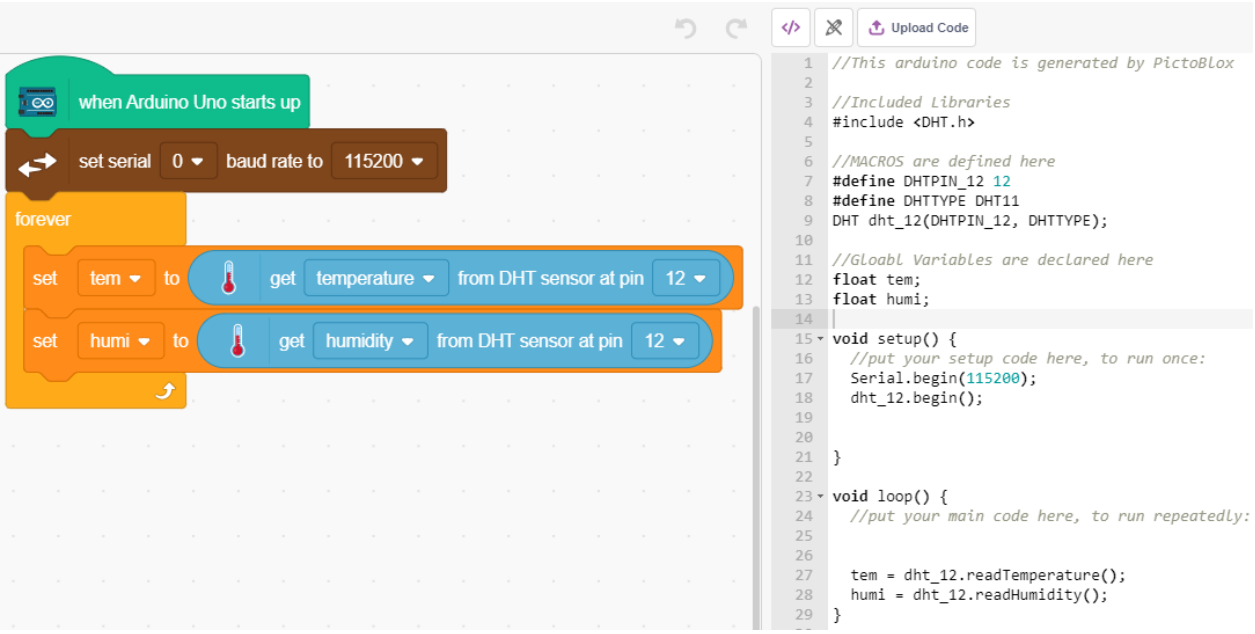
**Upload** モードで Arduino Uno を起動し、シリアルポートのボーレートを設定します。

- [Arduino 起動時]: **Upload** モードで Arduino Uno を起動します。
- [シリアルボーレートを設定]: **Communications** パレットから、シリアルポート 0 のボーレートを設定します。デフォルトは 115200 です。Mega2560 を使用している場合は、シリアルポート 0~3 でボーレートを設定することができます。




### 3. 温度と湿度の読取り

温度と湿度をそれぞれ保存するための 2 つの変数 **tem** と **humi** を作成します。ブロックをドラッグ&ドロップすると、コードが右側に表示されます。

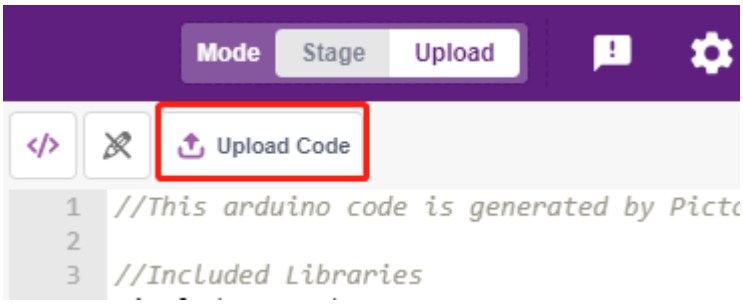



### 4. シリアルモニターに表示

読み取った温度と湿度をシリアルモニターに書き込みます。転送が速すぎて PictoBlox がジャムを起こすのを避けるため、[wait seconds] ブロックを使用して、次回の印刷のための一定の時間間隔を追加します。

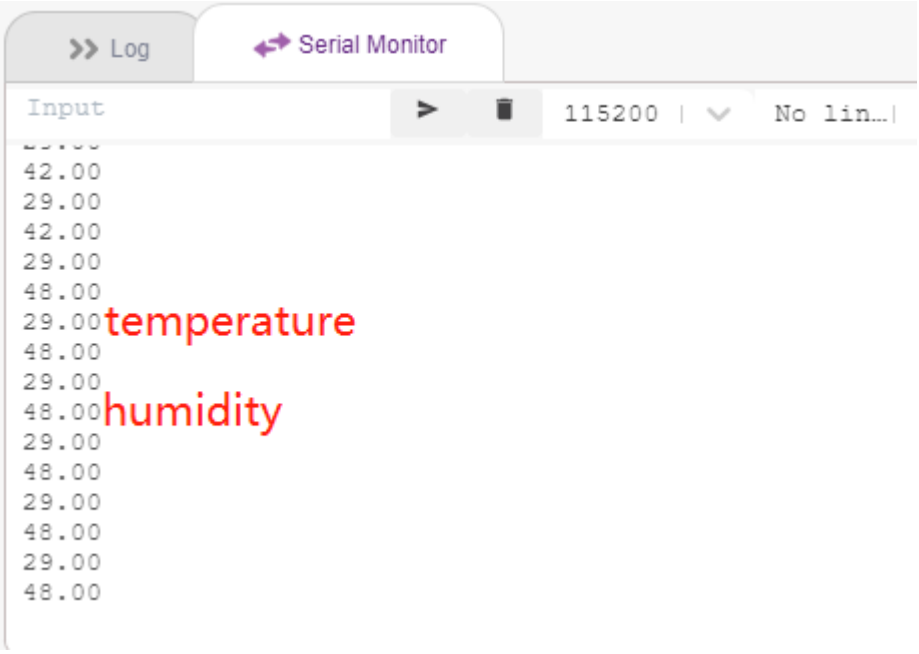

**5. コードのアップロード**

**Stage** モードとは異なり、 **Upload** モードのコードは、効果を見るために **Upload Code** ボタンを使用して Arduino ボードにアップロードする必要があります。このようにして、USB ケーブルを抜いてもプログラムが実行され続けます。



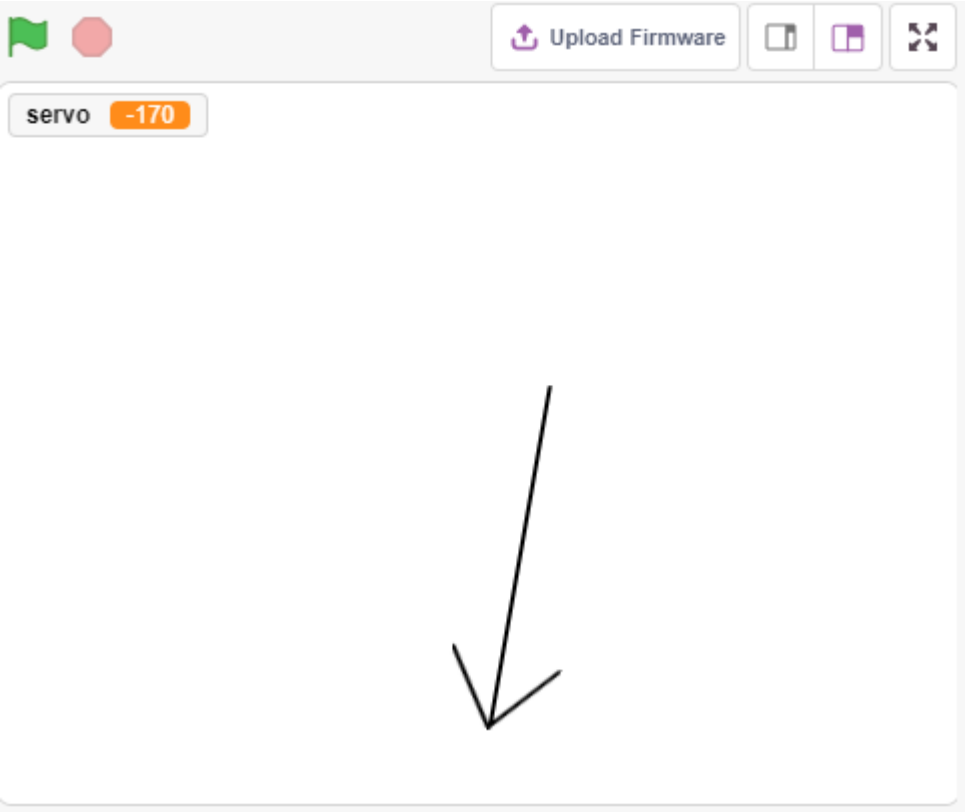

**6. シリアルモニターの起動**

今、 **Serial Monitor** を開き、温度と湿度を確認します。




## 7.13 2.10 振り子

このプロジェクトでは、矢印の振り子を作りながら、サーボが回転に従います。



### 7.13.1 学べること

- サーボの動作と角度の範囲
- スプライトを描き、中心点を尾に配置する。

### 7.13.2 必要な部品

このプロジェクトには、以下の部品が必要です。

キット全体を購入するのは確かに便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

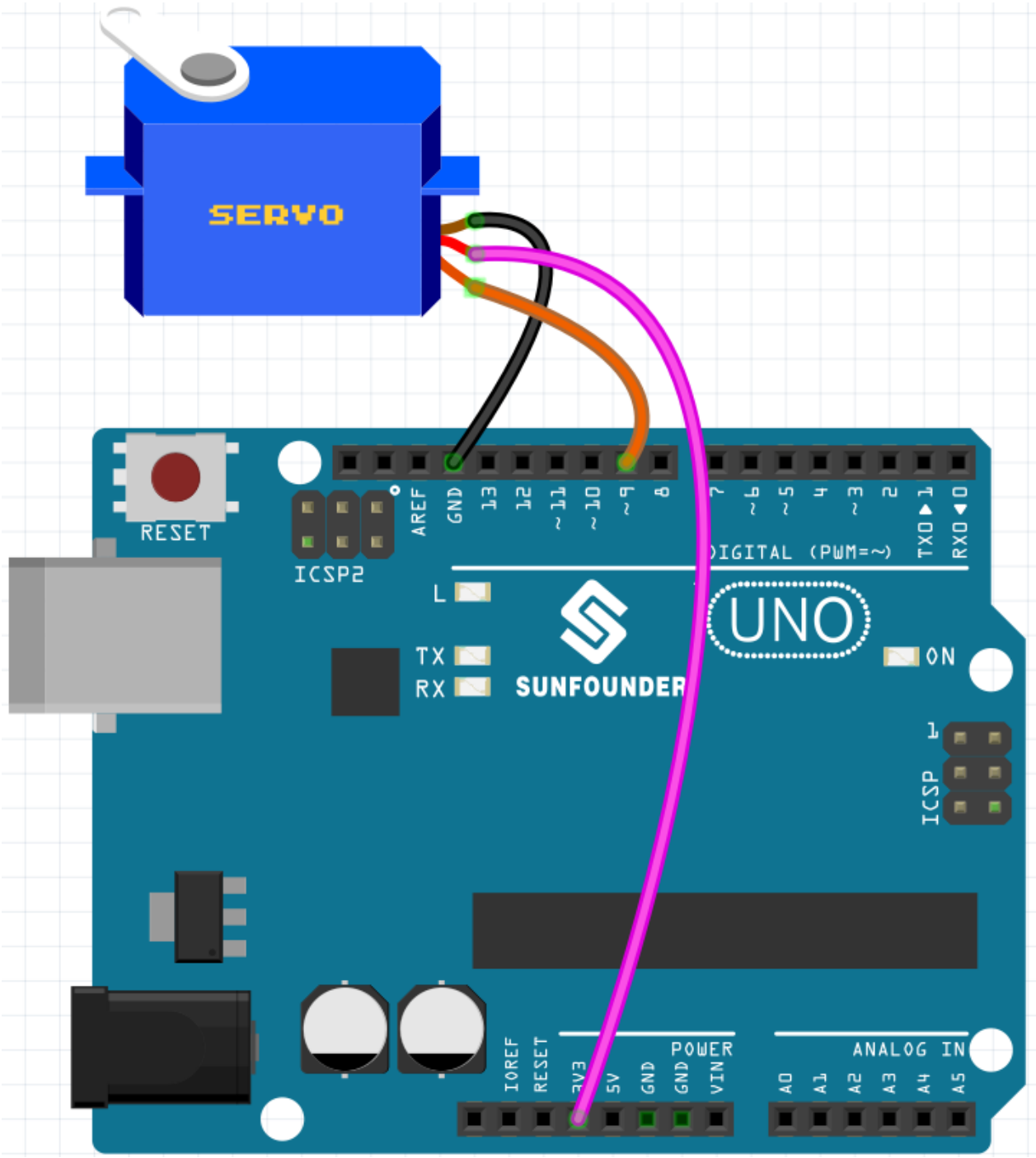
| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* **ボード** | |
| **ジャンパーワイヤー** | |
| **サーボ** | |

### 7.13.3 回路の作成

サーボは、180 度しか回転できない歯車式のモーターです。回路板からの電気パルスで制御されます。これらのパルスは、サーボにどの位置に移動するべきかを指示します。

サーボには 3 本のワイヤーがあります。茶色のワイヤーは GND、赤は VCC（3.3V に接続）、オレンジは信号ワイヤーです。角度の範囲は 0-180 度です。

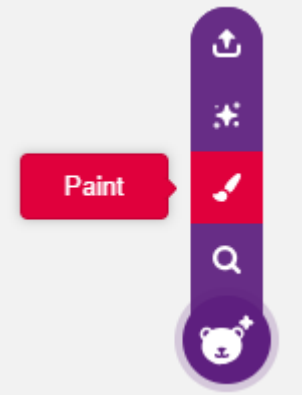
以下の図に従って回路を組み立ててください。


SERVO
RESET
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
DIGITAL (PWM=~)
TXD
RXD
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

### 7.13.4 プログラミング

#### 1. スプライトの描画

デフォルトのスプライトを削除し、スプライトボタンを選択して **Paint** をクリックすると、空のスプライト **Sprite1** が表示されます。



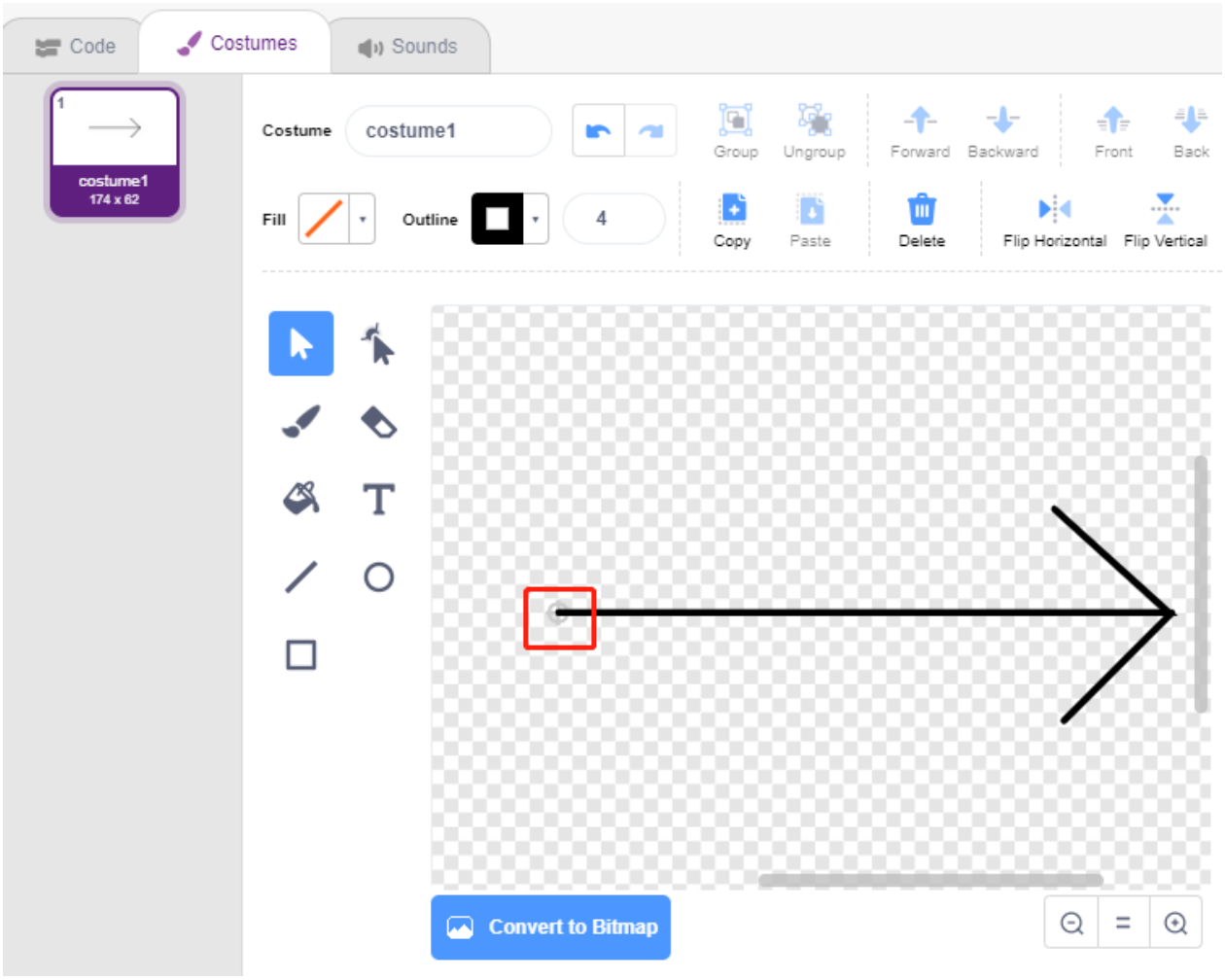

開かれた **Costumes** ページで、 **Line tool** を使用して矢印を描きます。

---

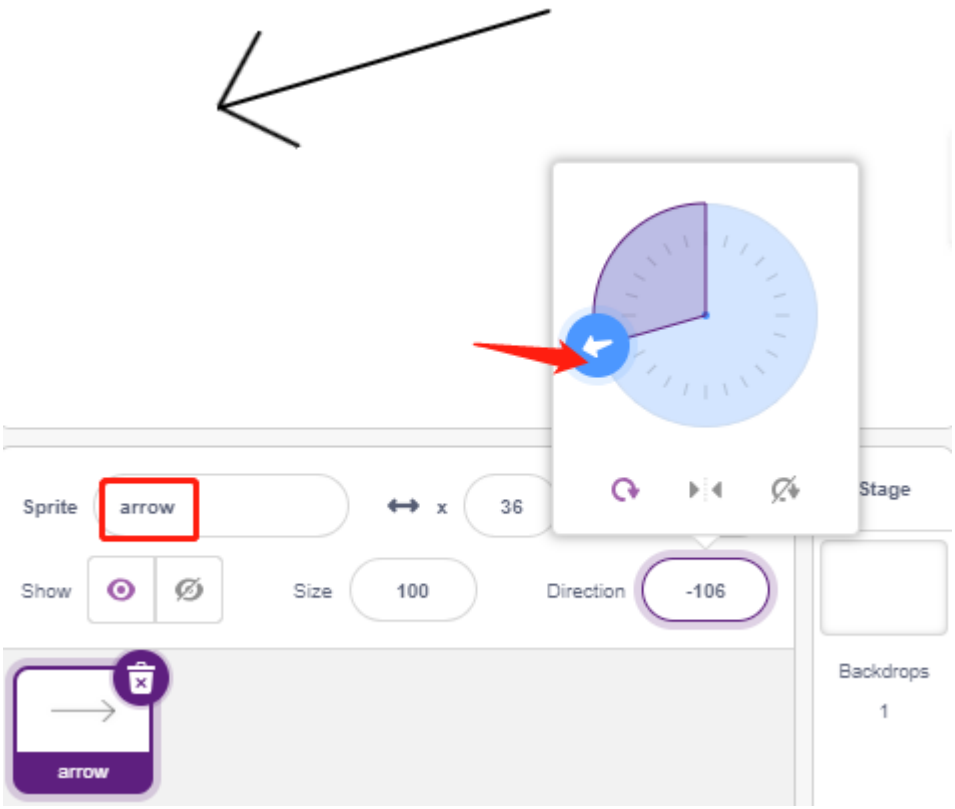
**注釈:**

- キャンバスの中心から外向きに矢印を描き始めることで、矢印が中心点を原点として円を描くように回転します。
- Shift キーを押しながら、ラインの角度を直線または 45 度にします。

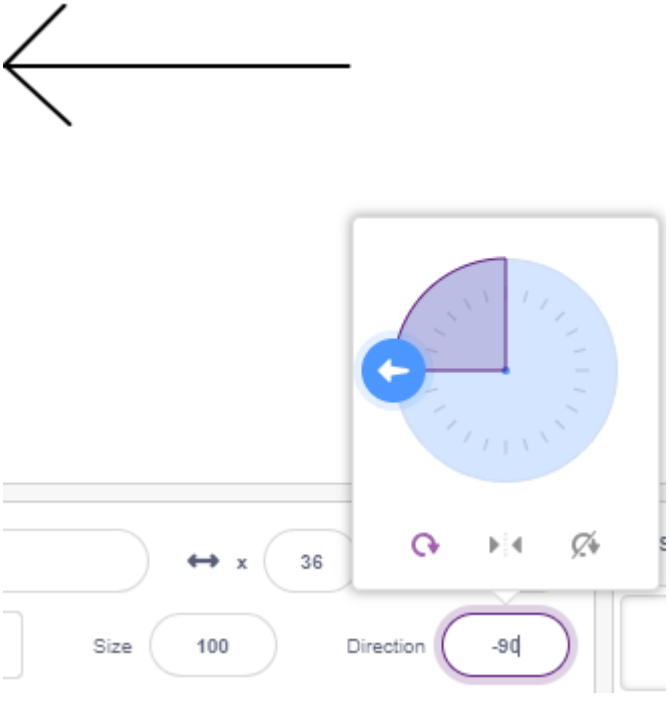
---

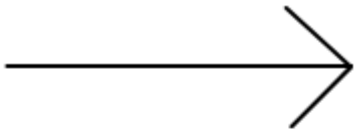


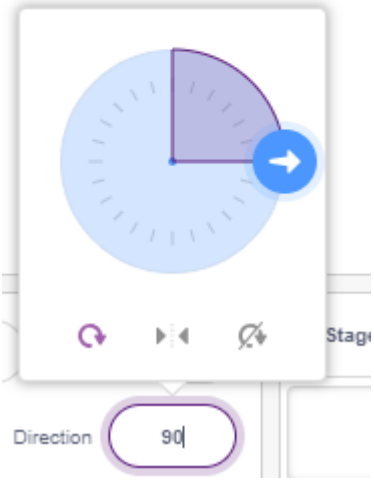

描画後、 **arrow** スプライトがステージに表示されます。それを **arrow** と名付けます。その後、 **Direction** の後の数字をクリックすると、円形のダイヤルが表示されます。この矢印をドラッグして、ステージ上の **arrow** スプライトが尾を原点として回転するかどうかを確認します。




**arrow** スプライトを左から右に振るためには、角度の範囲は-90 から-180、180 から 90 です。

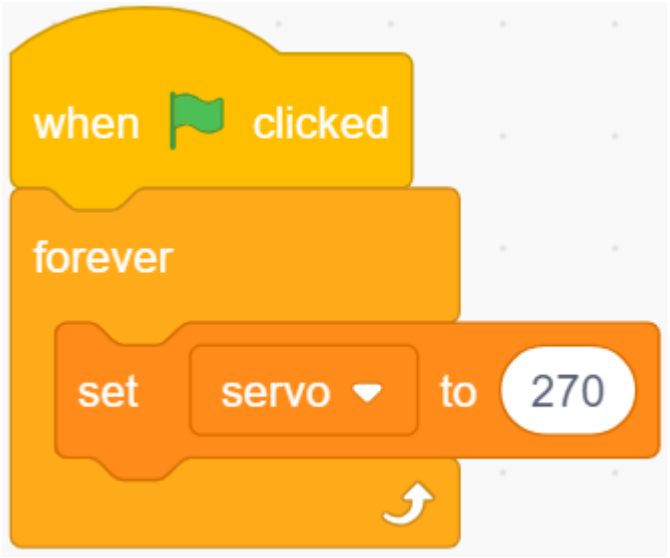







**2. 変数の作成**

**servo** という名前の変数を作成します。これは角度の値を格納し、初期値を 270 に設定します。



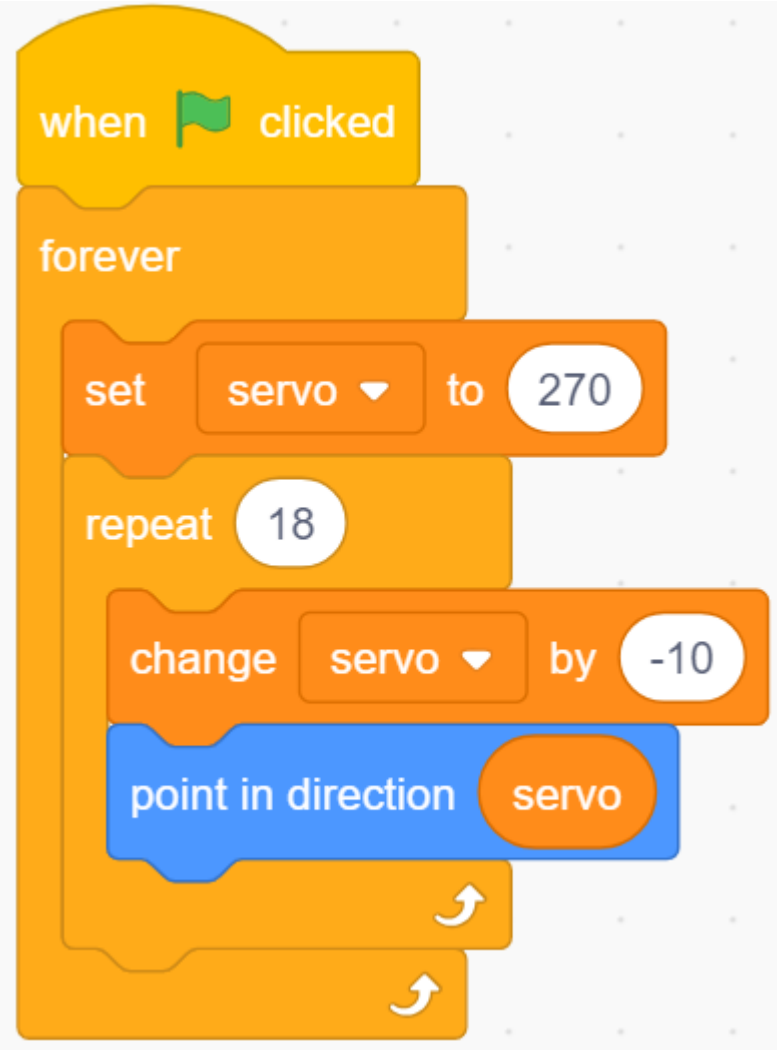

**3. 左から右に振る**

**arrow** スプライトが左の-90 度の位置から右の 90 度の位置まで振られるようにします。

[repeat] ブロックを使って、毎回変数に-10 を加えると、18 回で 90 度になります。それから [point in block] を使って、矢印スプライトがこれらの角度に向かうようにします。

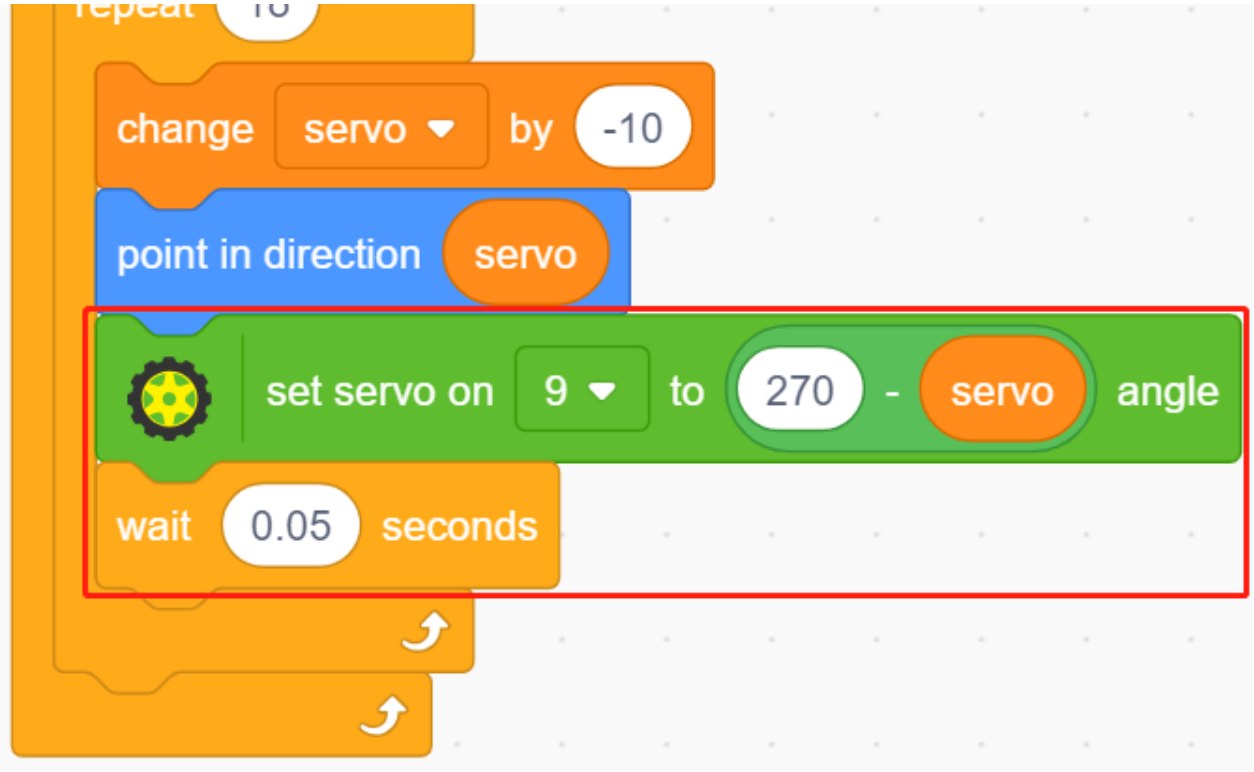
スプライトの回転角度は-180 ~ 180 であるため、この範囲外の角度は以下の条件で変換されます。

- 角度 > 180 の場合、角度 -360。




#### 4. サーボを回転させる

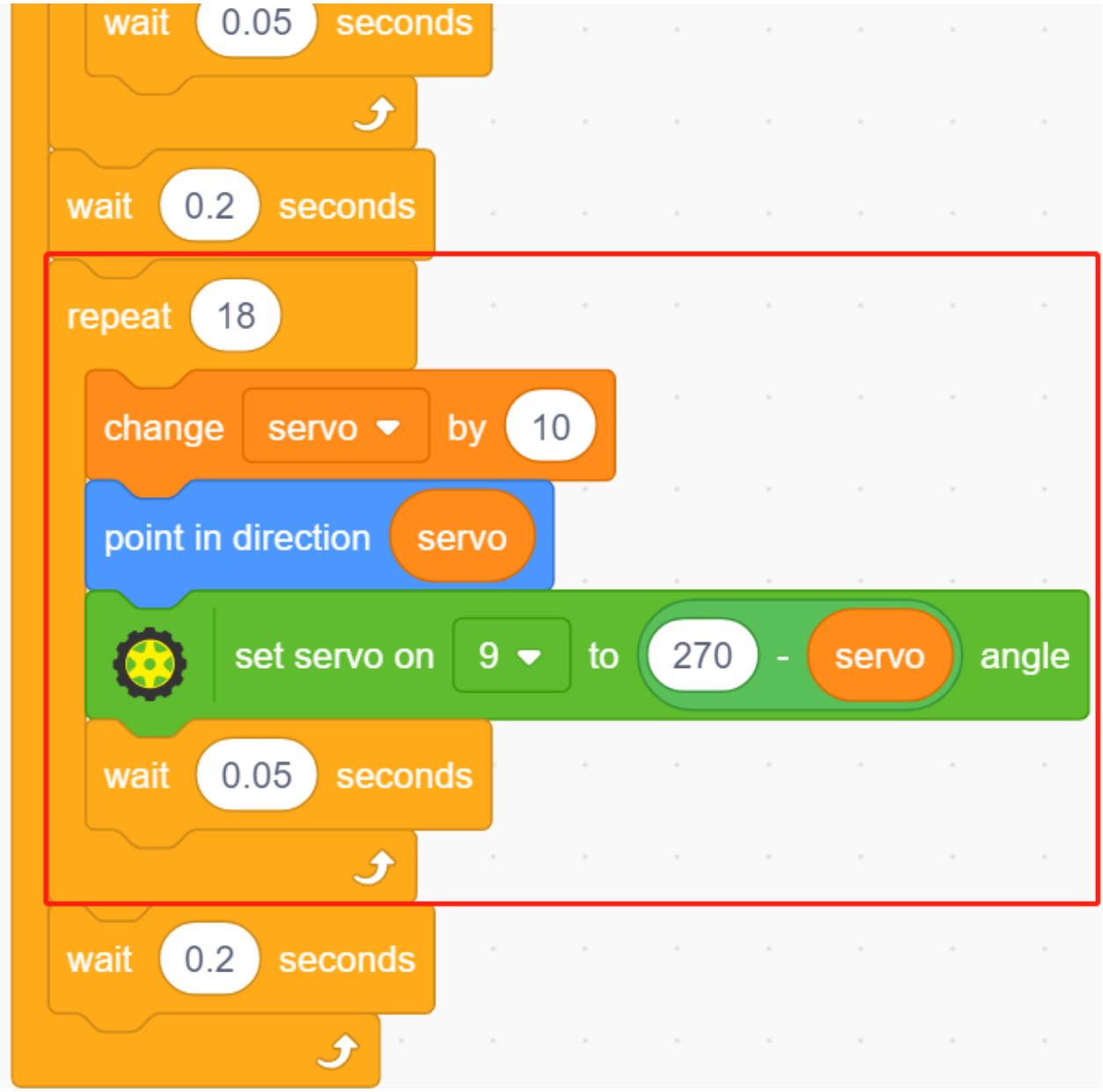
緑の旗をクリックすると、矢印がすぐに右に回転してから左に戻るのがわかります。そのため、ここで [wait seconds] ブロックを使用して、回転を遅くします。また、[set servo on to angle] ブロックを使用して、Arduino ボードに接続されたサーボを特定の角度に回転させます。




**5. 右から左への振り**

同じ方法で、サーボと **arrow** スプライトを右から左にゆっくりと回転させます。

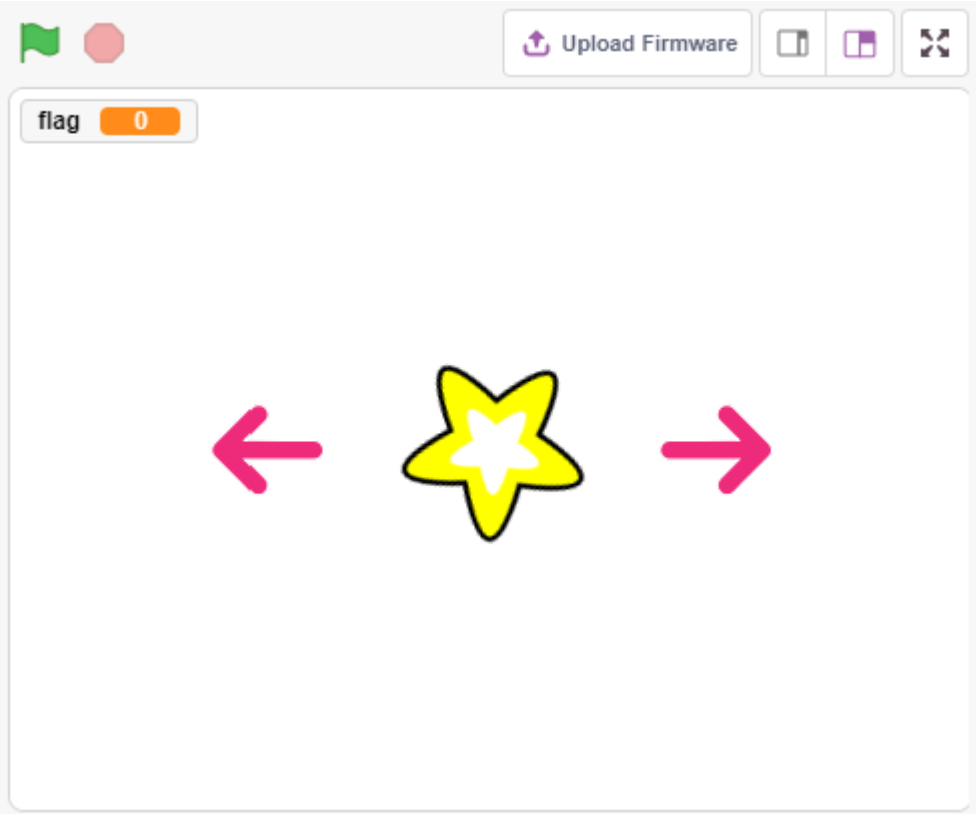
- 角度 > 180 の場合、角度 -360。




## 7.14 2.11 回転する扇風機

このプロジェクトでは、回転する星のスプライトと扇風機を作ります。

ステージ上の左右の矢印スプライトをクリックすると、モーターと星のスプライトの時計回りと反時計回りの回転を制御でき、星のスプライトをクリックすると回転が停止します。




### 7.14.1 学べること

- モーターの動作原理
- ブロードキャストの機能
- スプライト内の他のスクリプトを停止するブロック

### 7.14.2 必要な部品

このプロジェクトには、以下の部品が必要です。

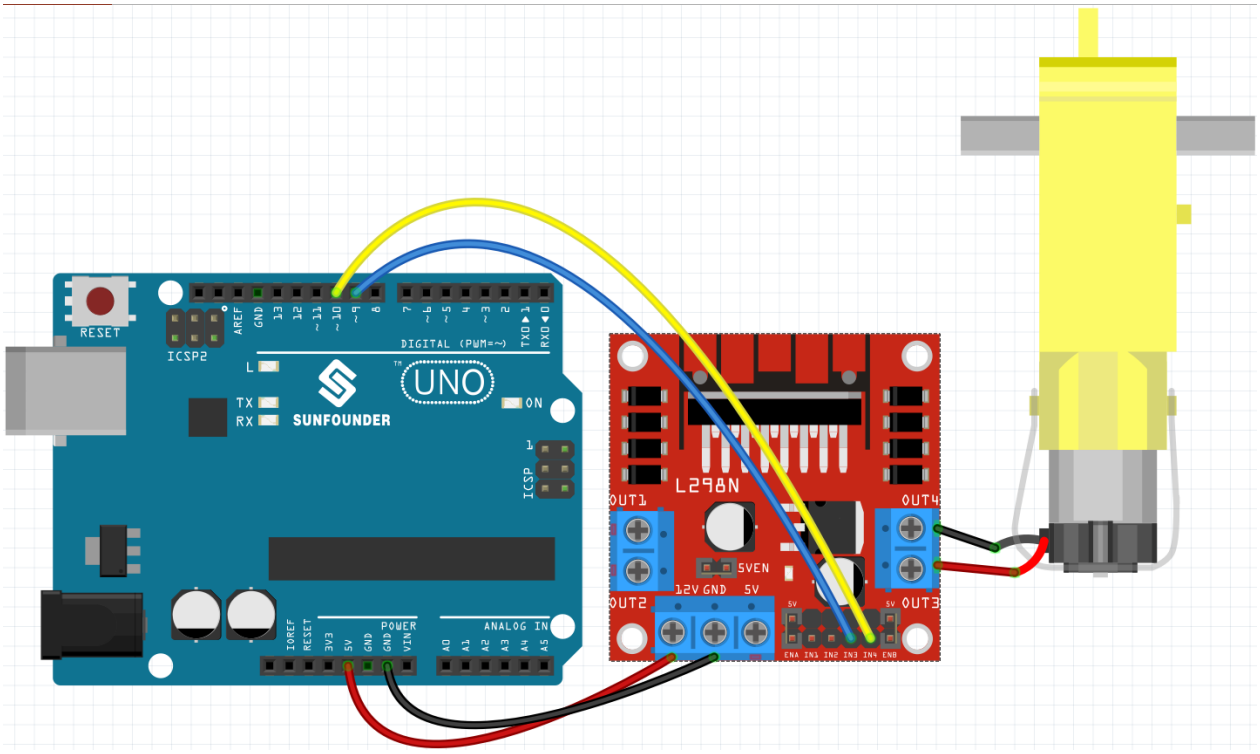
一式を購入するのは非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| *TT* モーター | - |
| *L298N* モジュール | |

### 7.14.3 回路の作成

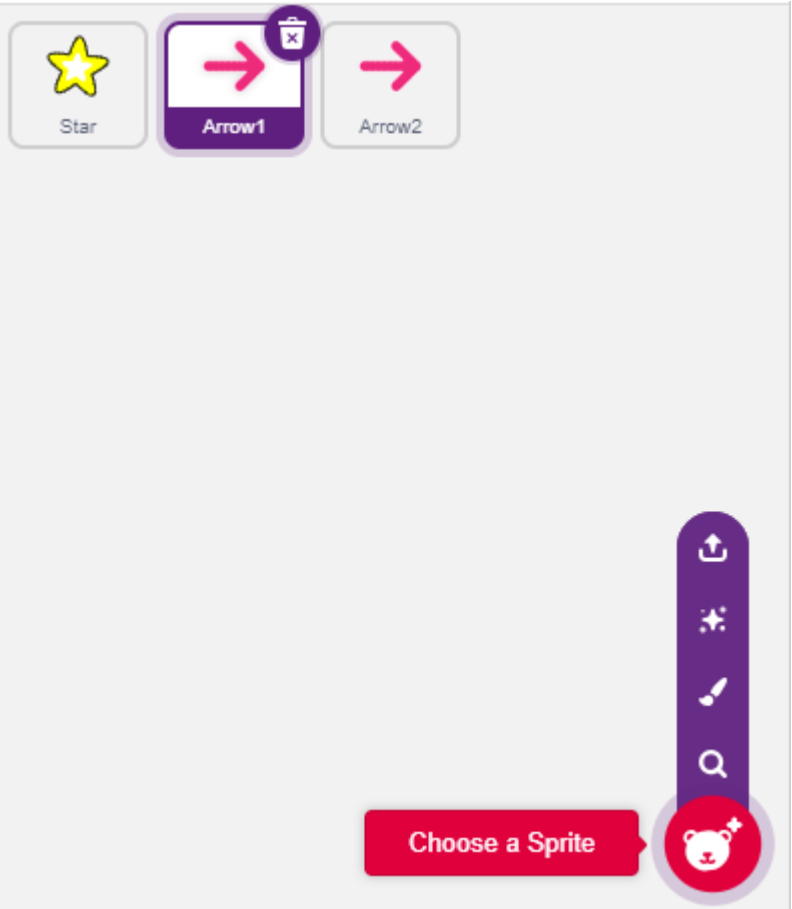



### 7.14.4 プログラミング

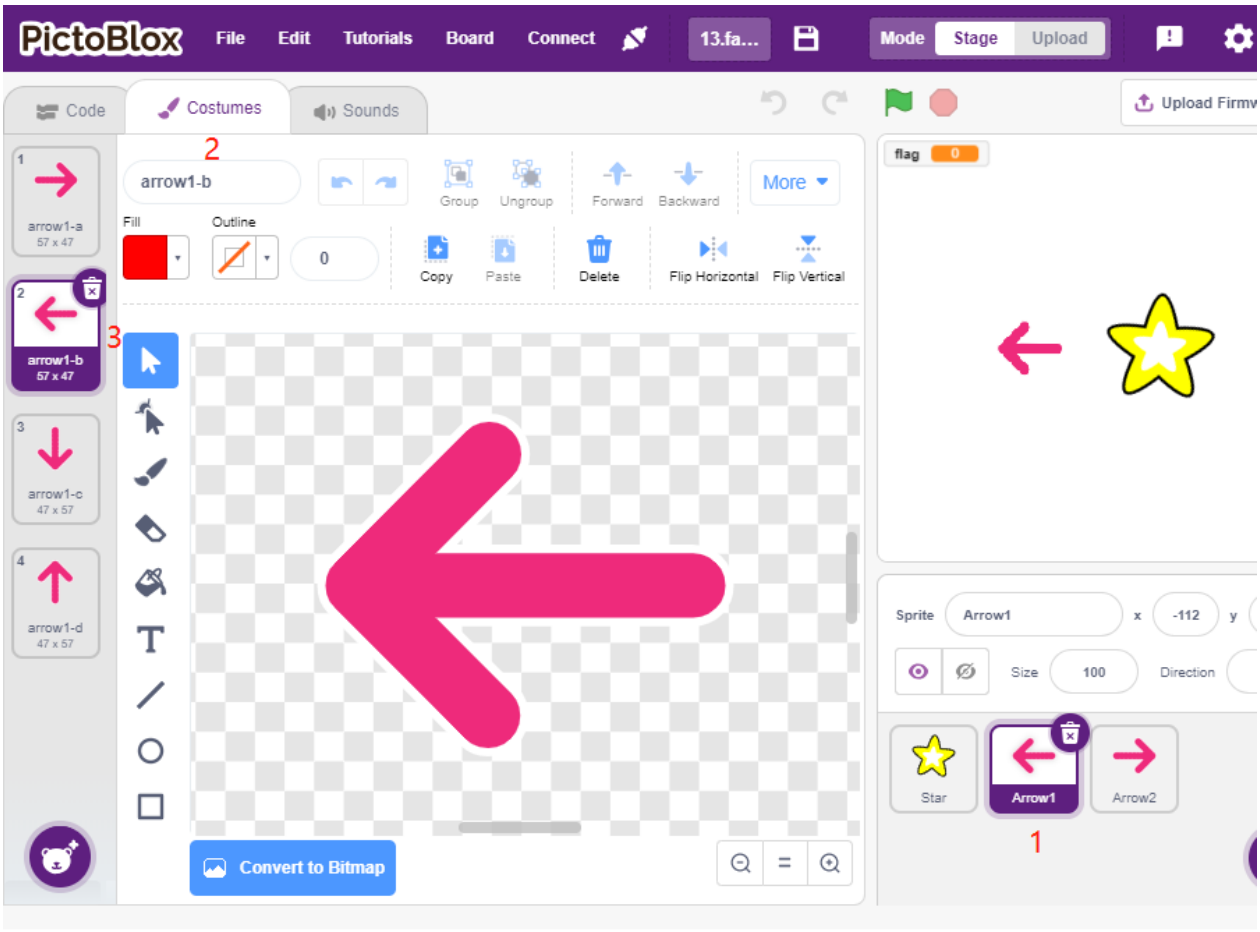
目指す効果は、2 つの矢印スプライトを使用してモーターと星のスプライトの時計回りと反時計回りの回転をそれぞれ制御し、星のスプライトをクリックするとモーターの回転を停止させることです。

#### 1. スプライトの追加

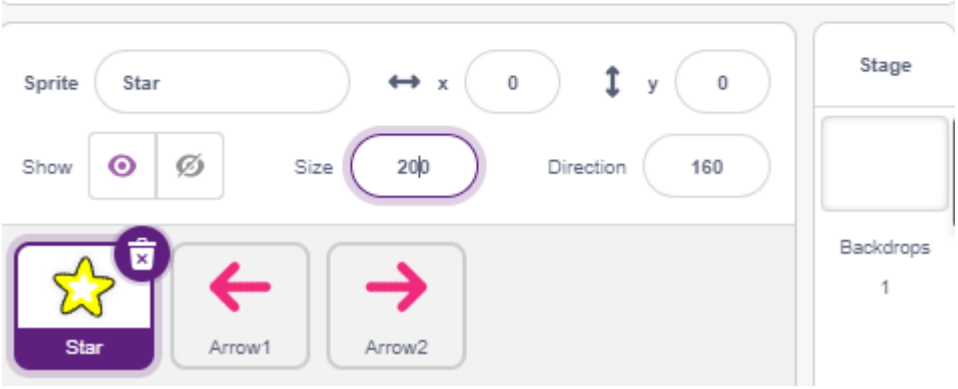
デフォルトのスプライトを削除し、 **Star** スプライトと **Arrow1** スプライトを選択し、 **Arrow1** を 1 つコピーします。




**Costumes** オプションで、 **Arrow1** スプライトを異なる方向のコスチュームに変更します。




スプライトのサイズと位置を適切に調整します。

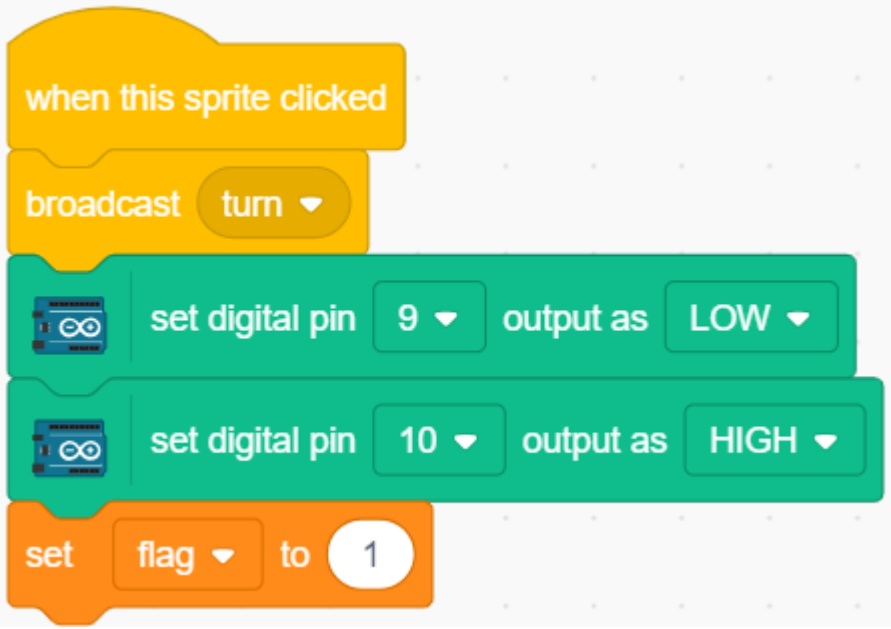



**2. 左の矢印スプライト**

このスプライトをクリックすると、メッセージ - turn をブロードキャストし、デジタルピン 9 をローにし、ピン 10 をハイに設定し、変数 **flag** を 1 に設定します。左の矢印スプライトをクリックすると、モーターが反時計回りに回転します。もし時計回りに回転する場合は、ピン 9 とピン 10 の位置を交換してください。

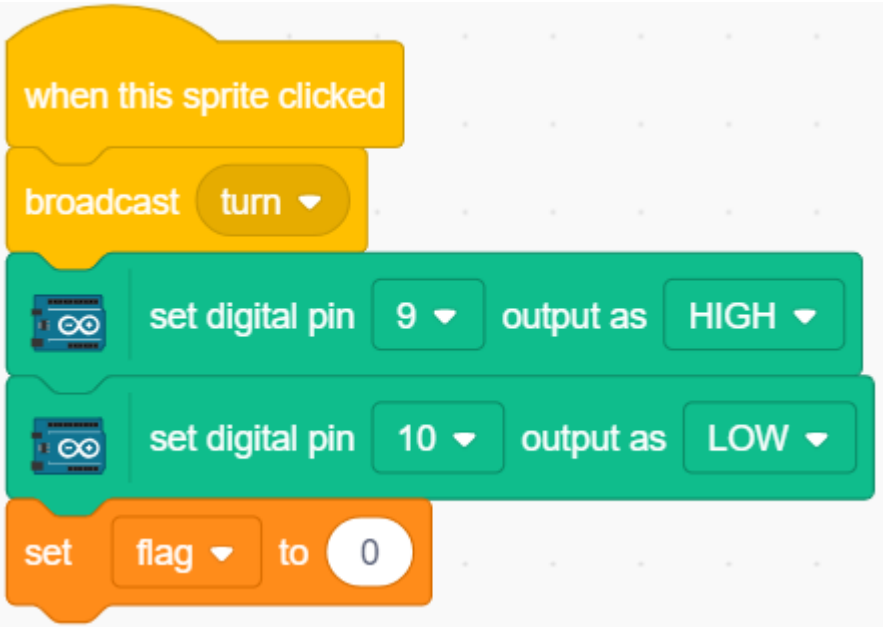
注意点が 2 つあります。

- [broadcast]: **Events** パレットから、他のスプライトにメッセージをブロードキャストするために使用します。他のスプライトがこのメッセージを受け取ると、特定のイベントを実行します。例えば、ここでは **turn** で、 **star** スプライトがこのメッセージを受け取ると、回転スクリプトを実行します。
- 変数 flag: 星のスプライトの回転方向は flag の値によって決まります。ですので、 **flag** 変数を作成する際には、すべてのスプライトに適用する必要があります。




### 3. 右の矢印スプライト

このスプライトをクリックすると、メッセージ turn をブロードキャストし、デジタルピン 9 をハイにし、ピン 10 をローにしてモーターを時計回りに回転させ、 **flag** 変数を 0 に設定します。

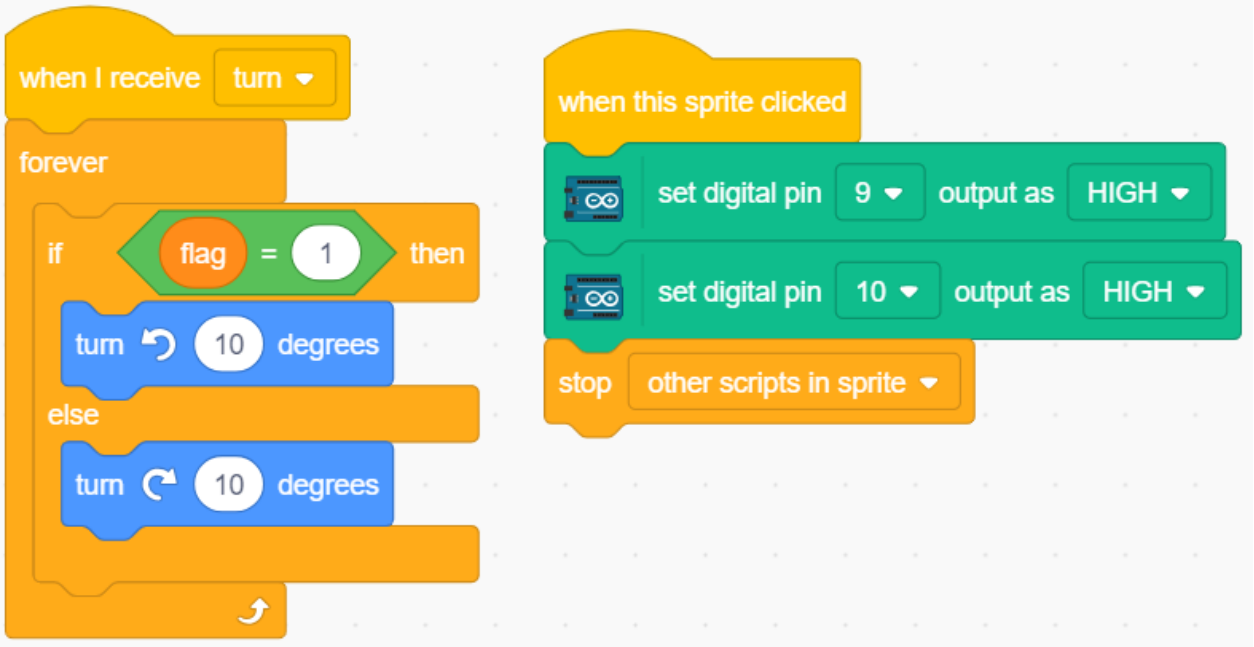



### 4. 星のスプライト

ここには 2 つのイベントが含まれています。

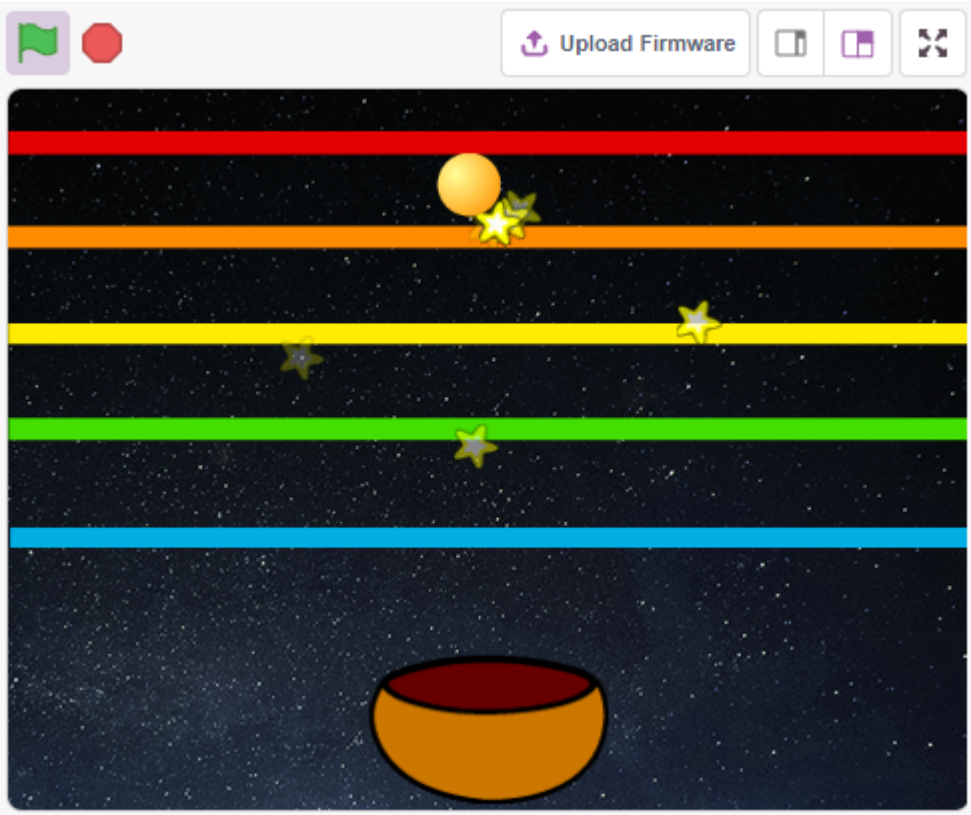
- **star** スプライトがブロードキャストされたメッセージ turn を受け取ると、flag の値を判断します。もし flag が 1 なら、左に 10 度回転し、それ以外の場合は逆になります。[FOREVER] にあるので、常に回転し続けます。

- このスプライトをクリックすると、モーターの両方のピンをハイにして回転を停止させ、このスプライト内の他のスクリプトを停止させます。




## 7.15 2.12 光感知ボール

このプロジェクトでは、フォトレジスタを使用して、ステージ上のボールを上方に飛ばします。フォトレジスタの上に手を置いて、受け取る光の強度をコントロールします。手をフォトレジスタに近づけるほど、その値は小さくなり、ステージ上のボールはより高く飛びます。そうでなければ、ボールは落ちます。ボールが糸に触れると、きれいな音ときらきらとした星の光が放たれます。




### 7.15.1 学べること

- スプライトを色で塗りつぶす
- スプライト間の接触

### 7.15.2 必要な部品

このプロジェクトで必要な部品は以下の通りです。

キット全体を購入すると、確かに便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| フォトレジスタ | |

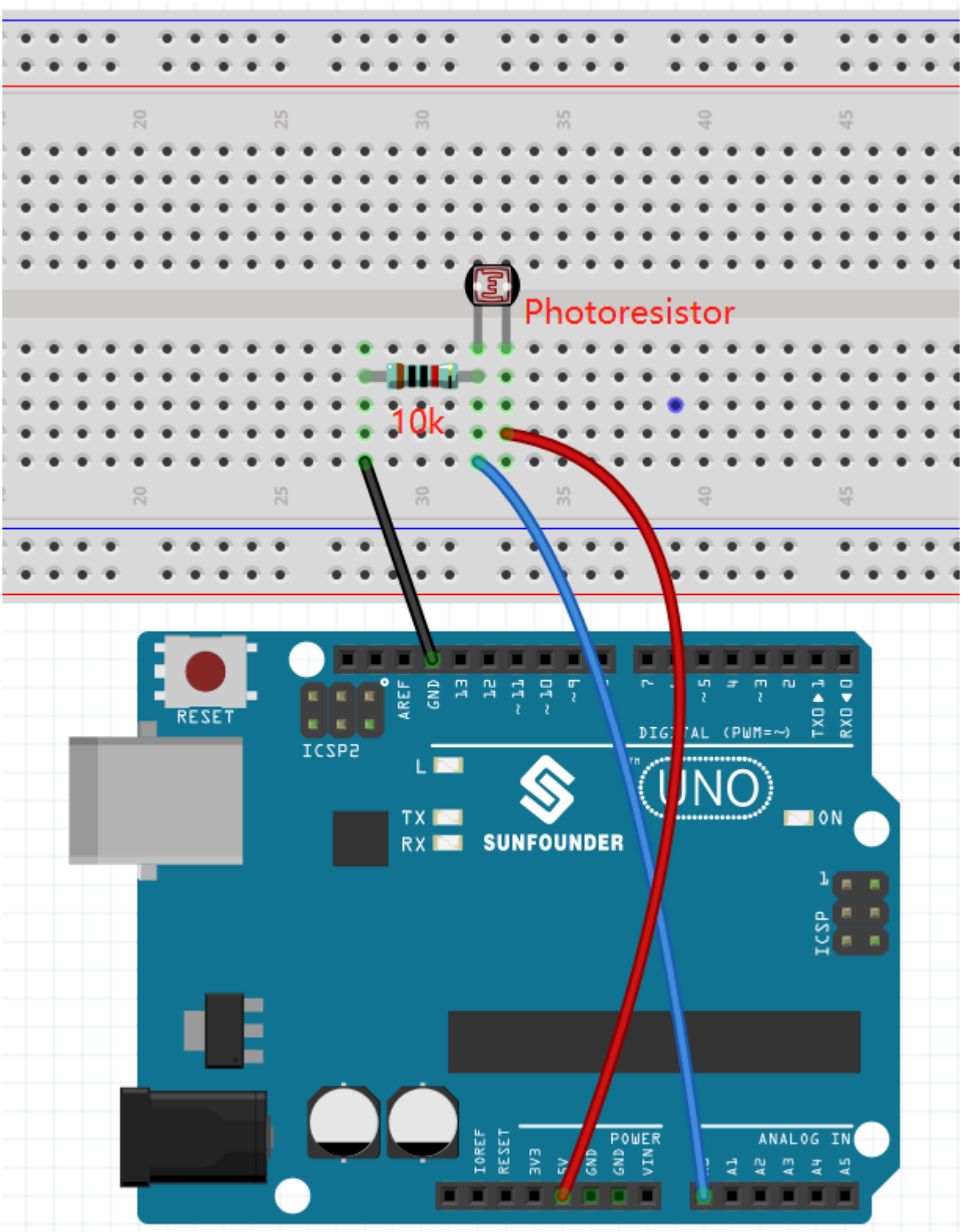
### 7.15.3 回路の作成

フォトレジスタまたはフォトセルは、光に制御される可変抵抗器です。フォトレジスタの抵抗は、入射光の強度が増えると減少します。

以下の図に従って回路を組み立てます。

フォトレジスタの一方の端子を 5V に、他方の端子を A0 に接続し、この端子で 10K の抵抗を GND と直列に接続します。

したがって、光の強度が増えると、フォトレジスタの抵抗が減少し、10K 抵抗の電圧分割が増加し、A0 で取得される値が大きくなります。

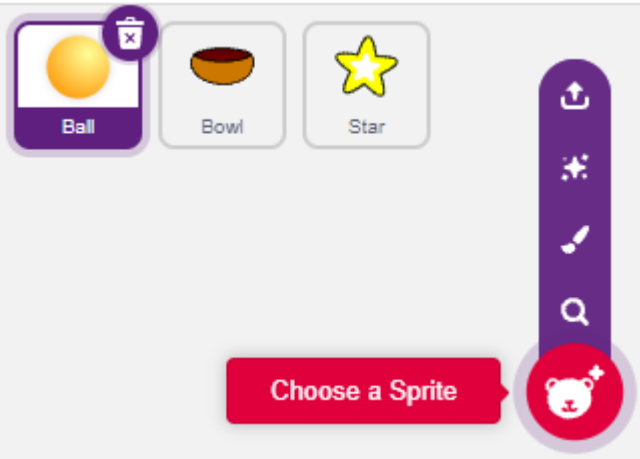

Photoresistor
10k
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
7
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A1
A2
A3
A4
A5
ANALOG IN

### 7.15.4 プログラミング

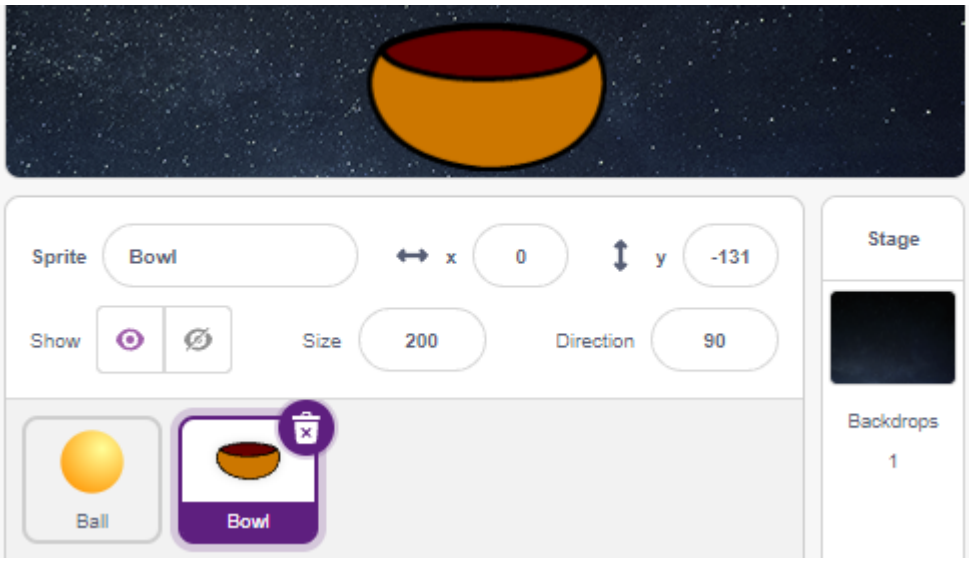
目指す効果は、手をフォトレジスタに近づけるほど、ステージ上のボールのスプライトが上に上がり続け、そうでなければボウルのスプライト上に落ちることです。上を歩くか、下に落ちる際に、ラインのスプライトに触れると、楽音を発し、全方向に星のスプライトを放出します。

**1. スプライトと背景の選択**

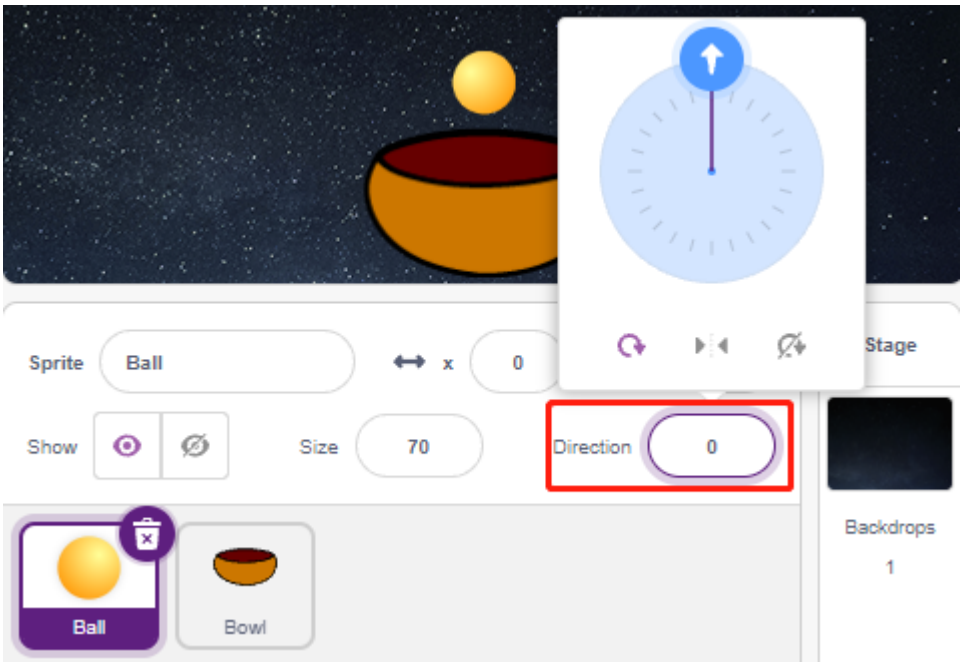
デフォルトのスプライトを削除し、 **Ball** 、 **Bowl** 、および **Star** スプライトを選択します。



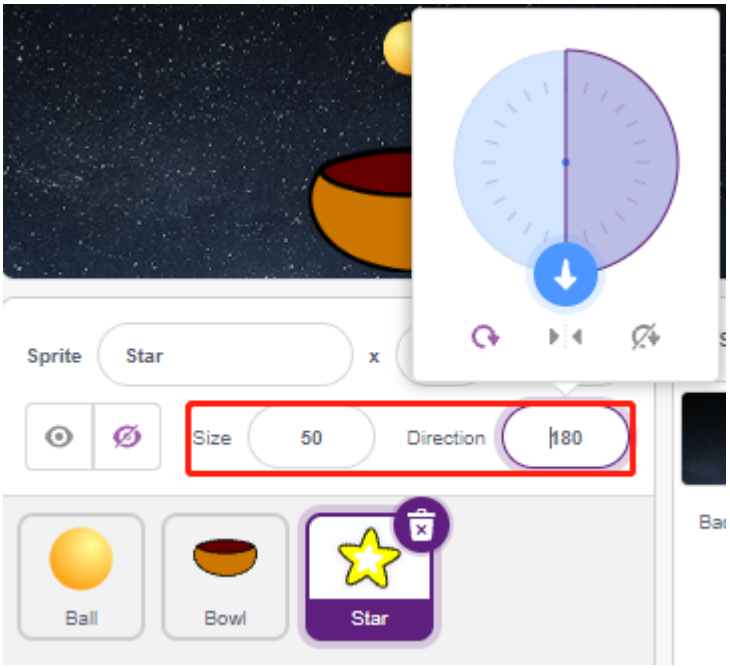

**Bowl** スプライトをステージの下中央に移動し、そのサイズを拡大します。



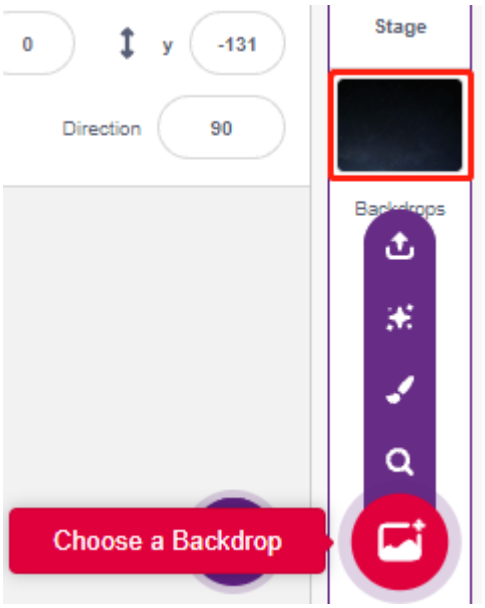

上に移動する必要があるので、 **Ball** スプライトの方向を 0 に設定します。




**Star** スプライトのサイズと方向を 180 に設定します。これにより、下に落ちるようになりますが、別の角度に変更することもできます。

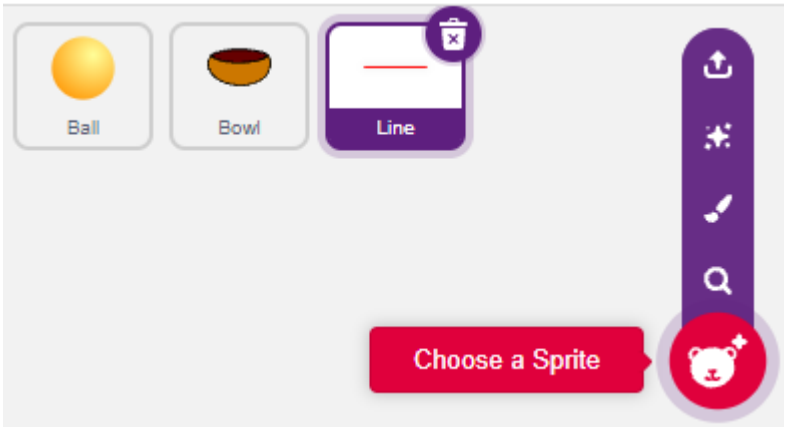



**Stars** の背景を追加します。

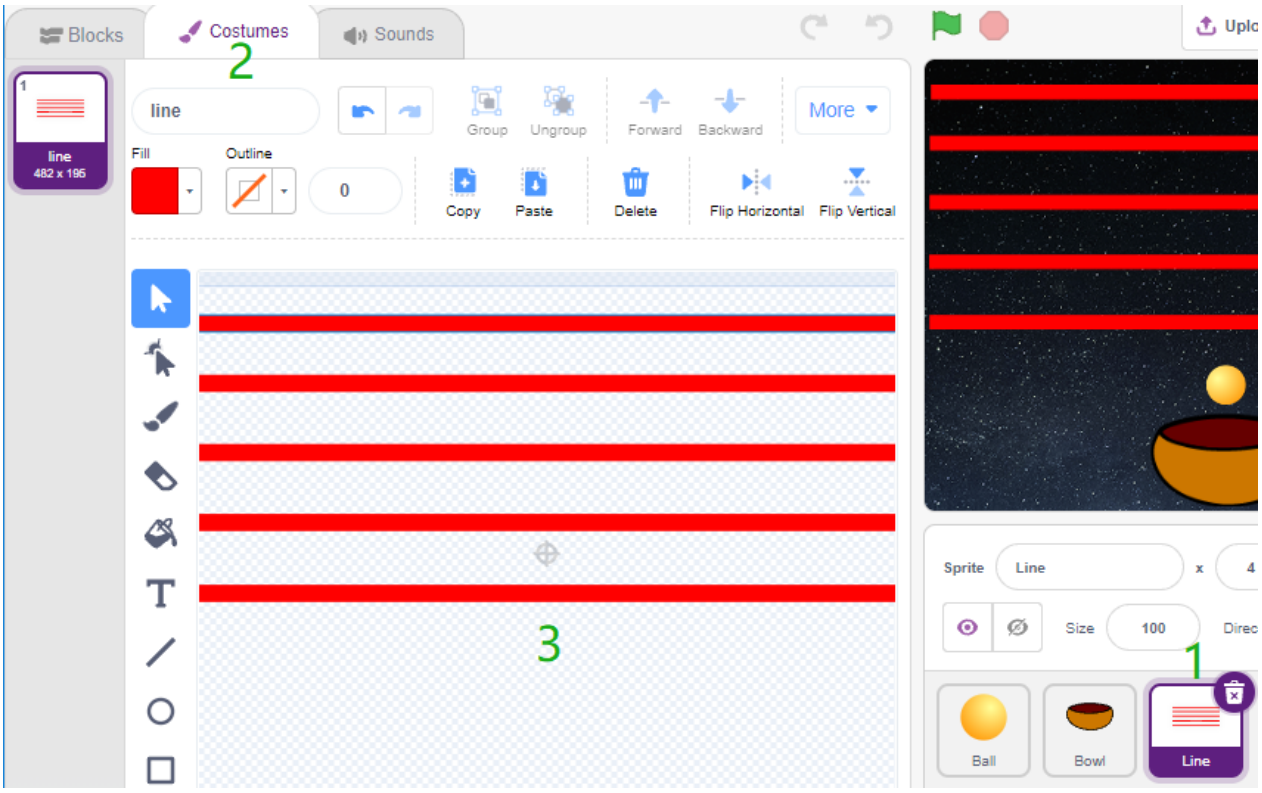



**2. Line スプライトの描画**

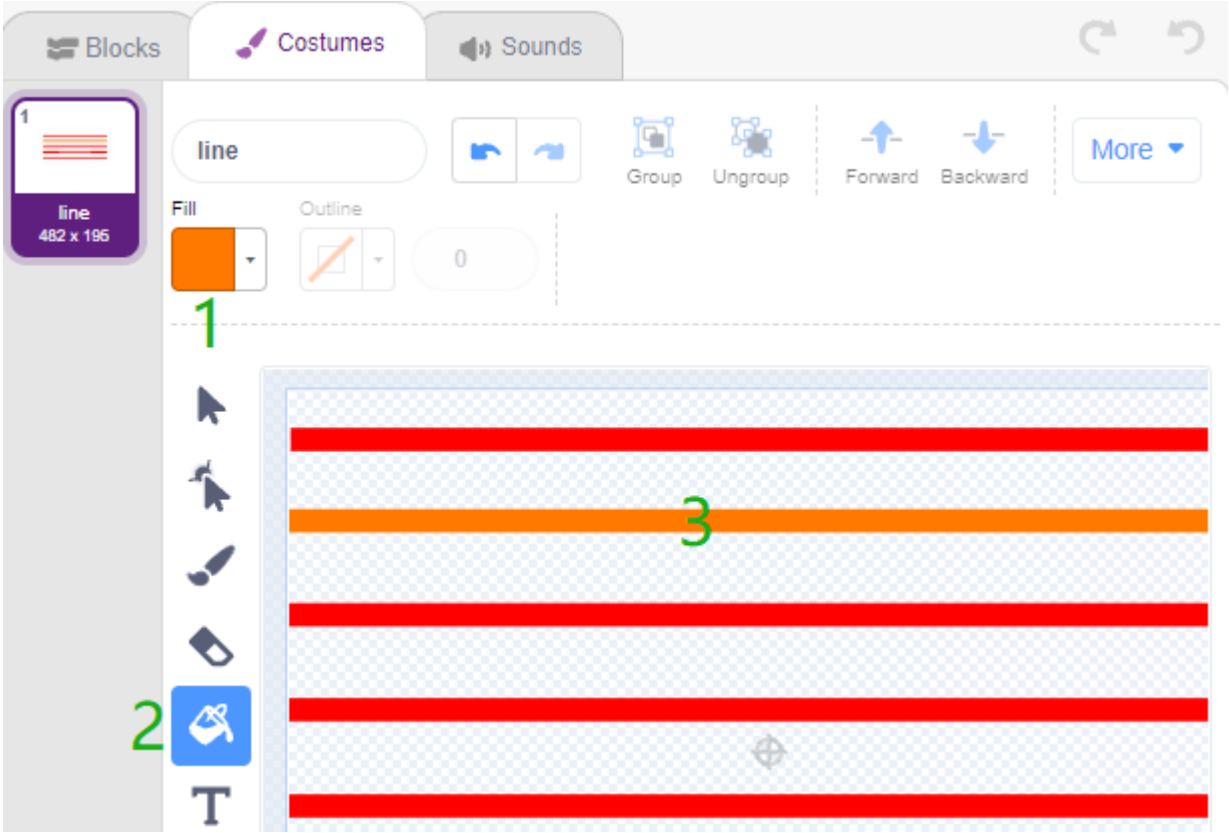
Line スプライトを追加します。



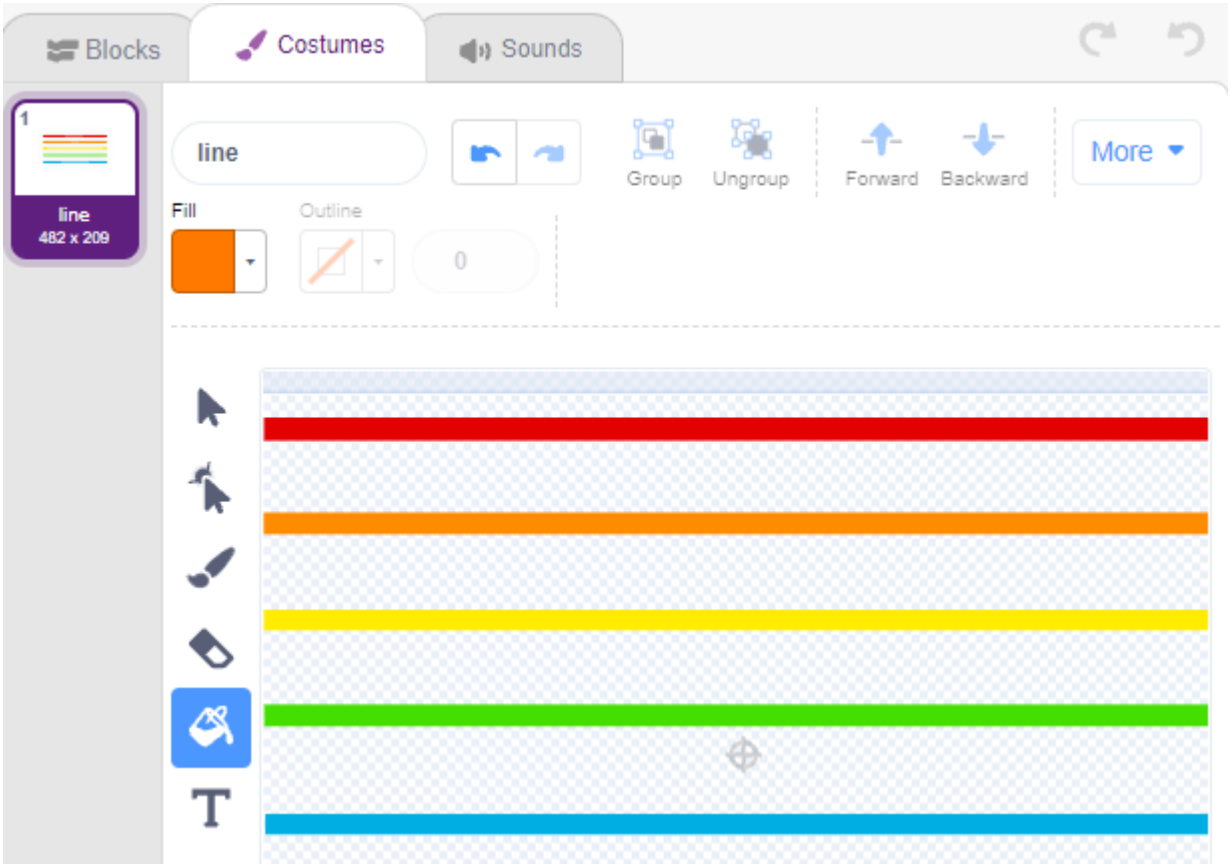

**Line** スプライトの **Costumes** ページに移動し、キャンバス上の赤いラインの幅を若干狭め、それを 5 回コピーしてラインを整列させます。




ラインに異なる色を塗りつぶします。好きな色を選択し、 **Fill** ツールをクリックして、ラインの上でマウスを動かして色を塗りつぶします。



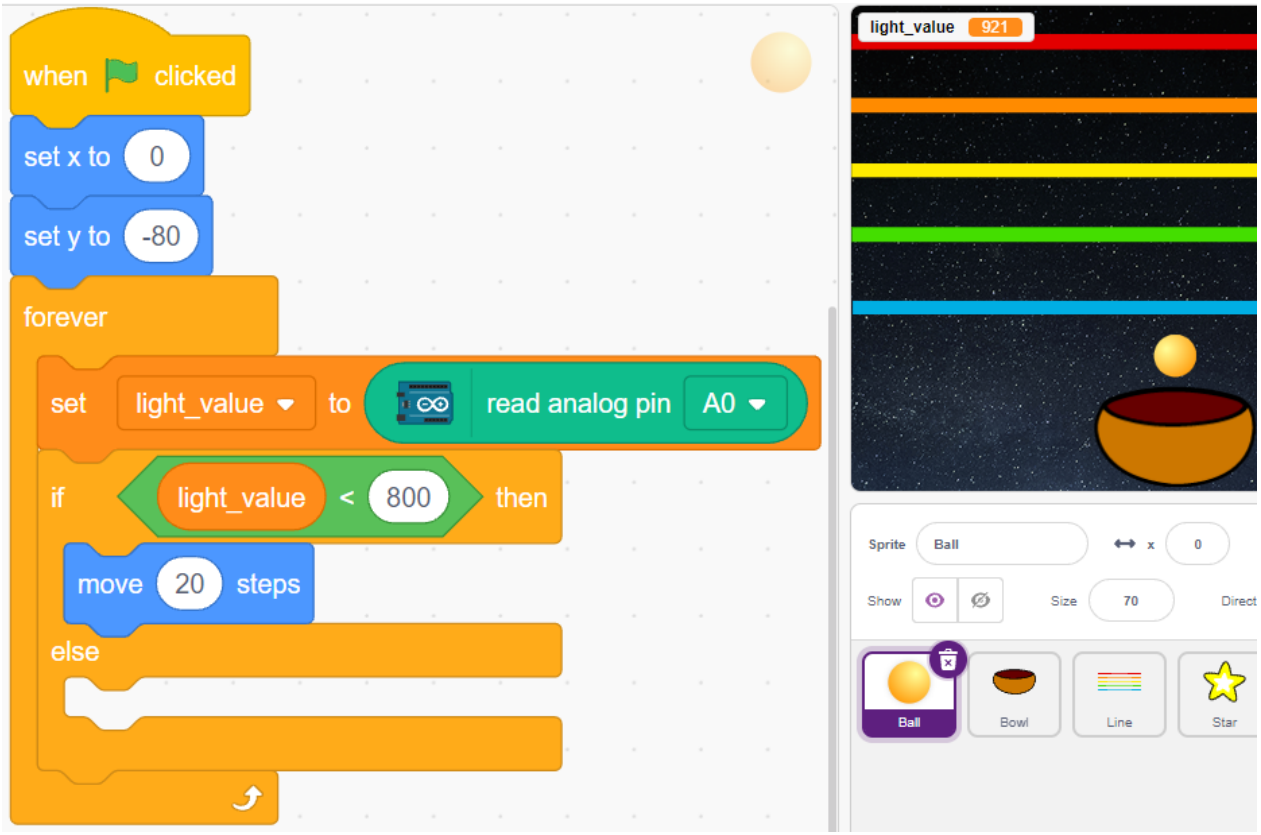

同じ方法で、他のラインの色を変更します。



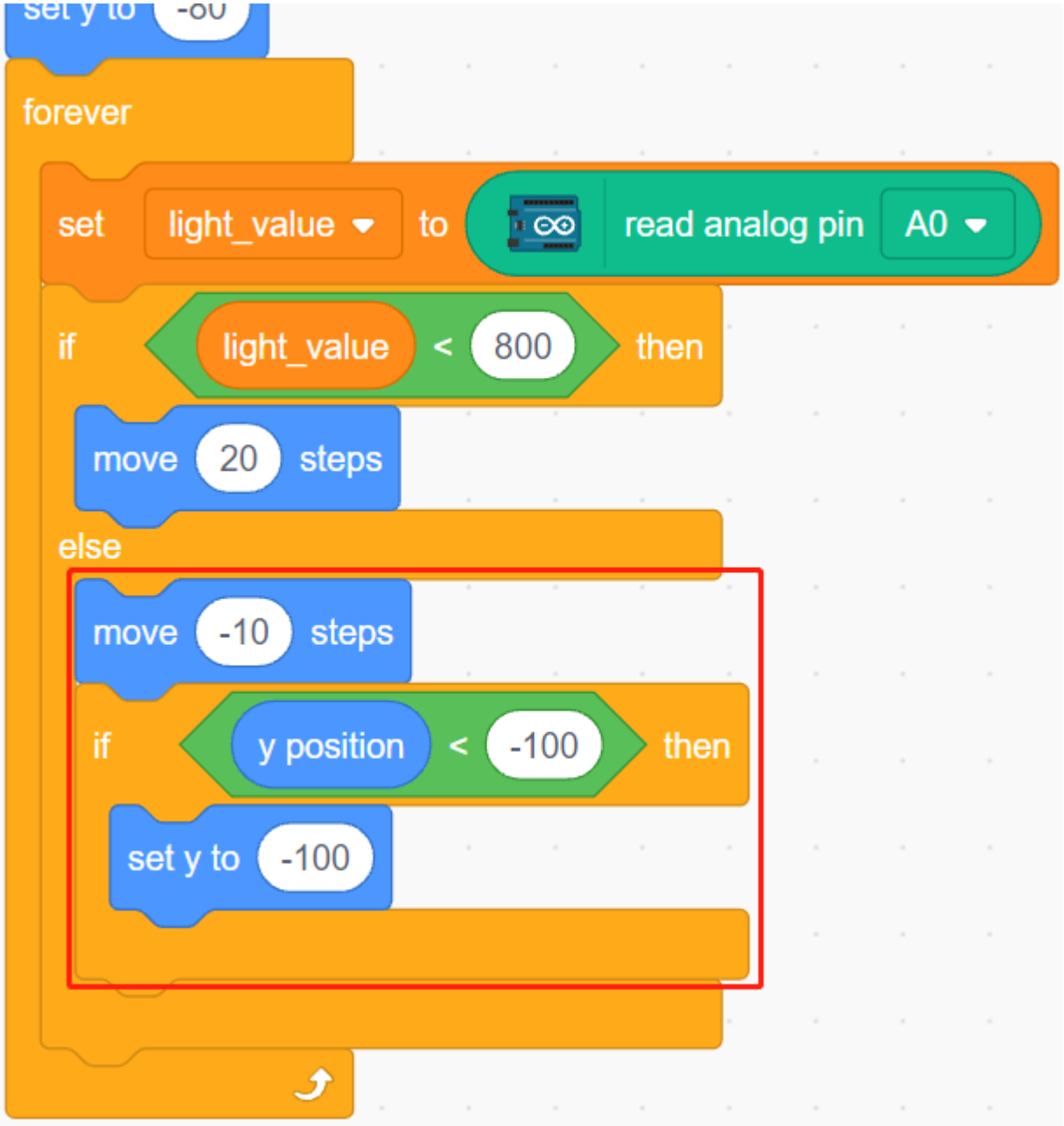

### 3. Ball スプライトのスクリプト

**Ball** スプライトの初期位置を設定し、光の値が 800 未満の場合（現在の環境に応じて別の値にすることができます）、Ball を上に動かします。

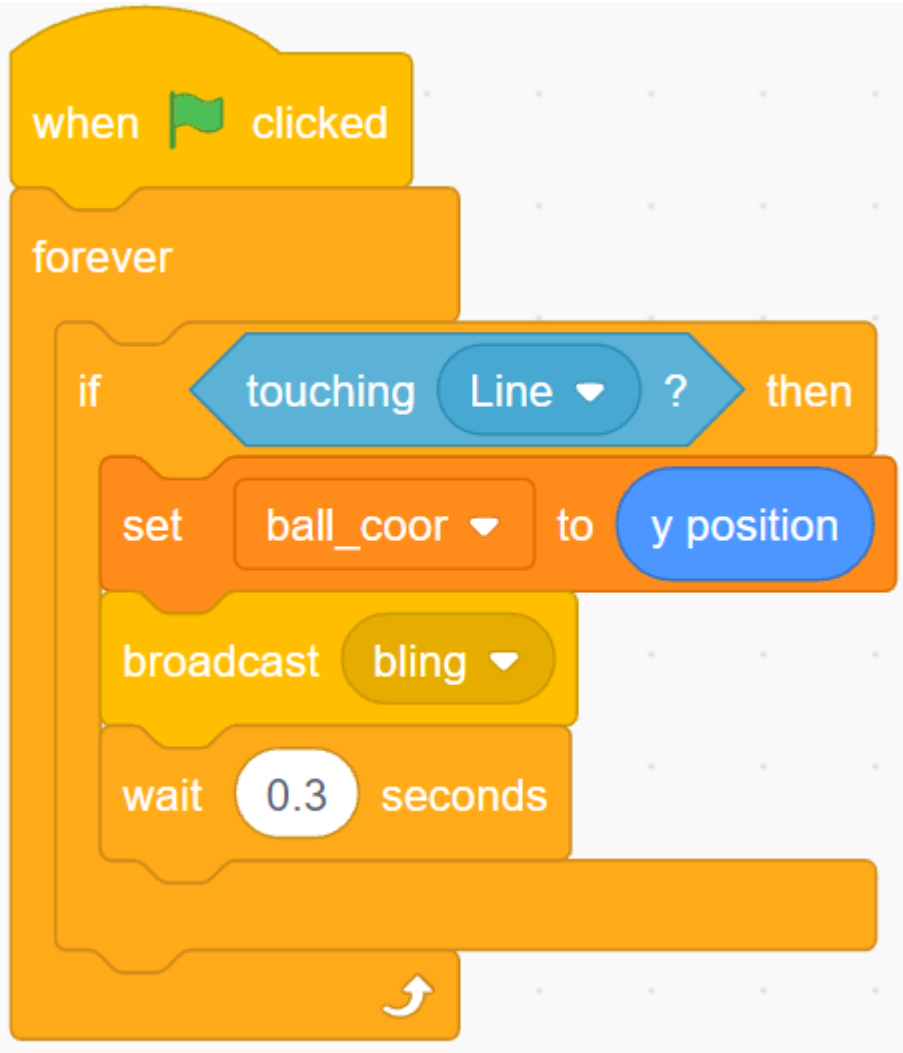
light_value の変数をステージ上で表示させて、光の強度の変化を随時観察することができます。




そうでなければ、 **Ball** スプライトは落ち、その Y 座標を-100 の最小値に制限します。これを修正して、 **Bowl** スプライト上に落ちているように見せることができます。

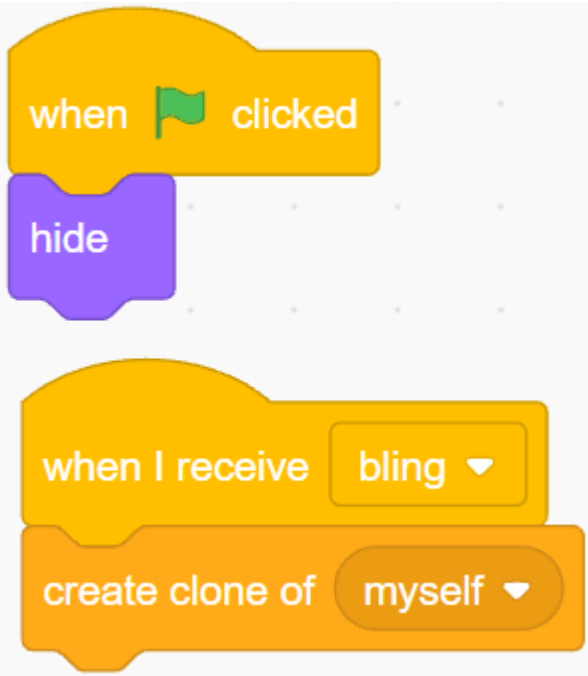



**Line** スプライトが当たったとき、現在の Y 座標を **ball_coor** 変数に保存し、 **Bling** メッセージがブロードキャストされます。

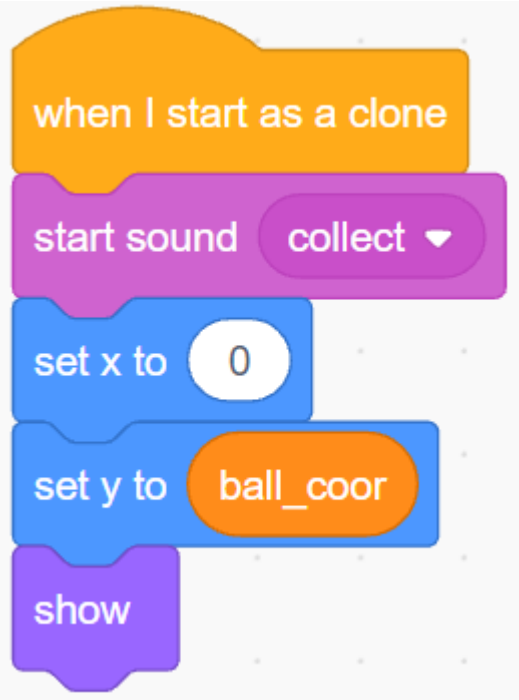



**4. Star スプライトのスクリプト**

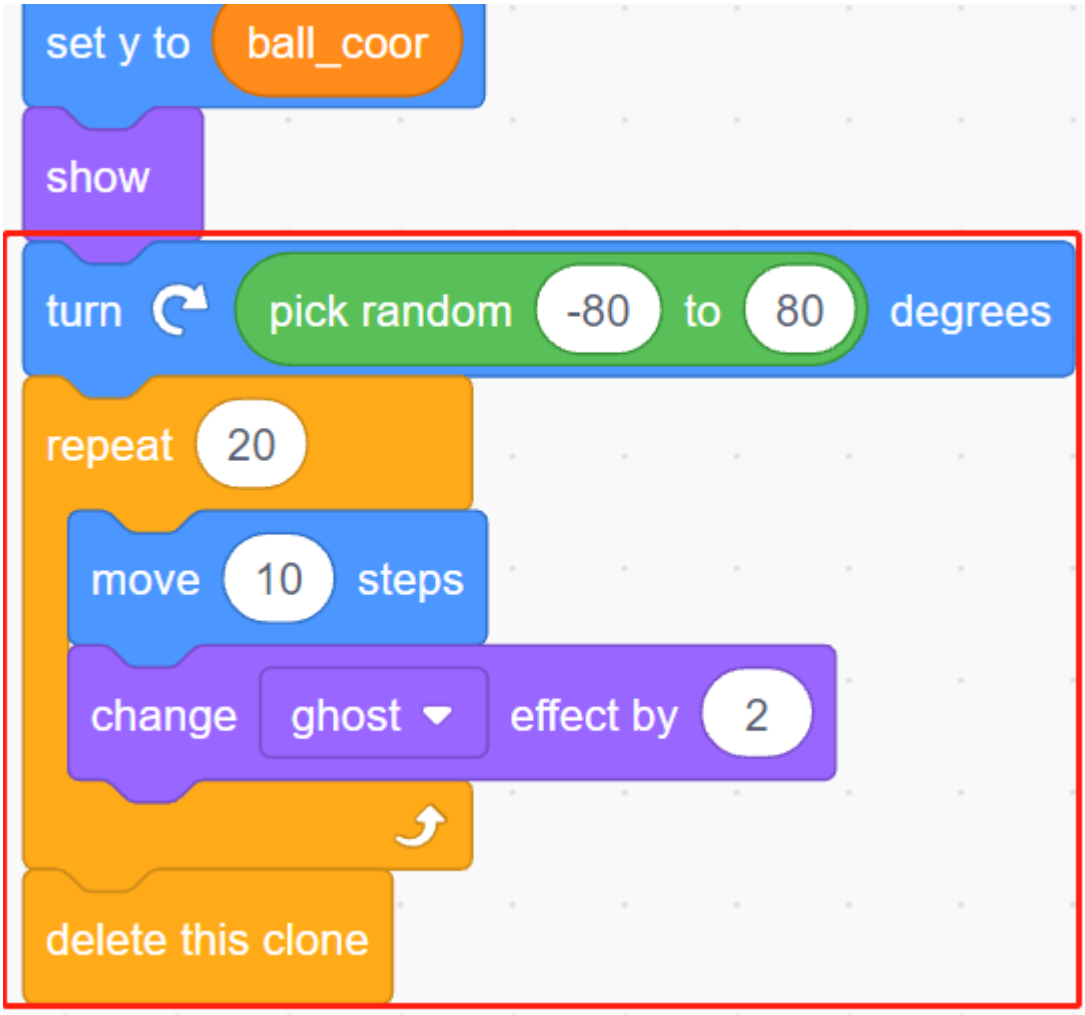
スクリプトが開始されたとき、まず **Star** スプライトを隠します。 **Bling** メッセージを受信すると、 **Star** スプライトをクローンします。




**Star** スプライトがクローンとして表示されると、音声効果を再生し、その座標を **Ball** スプライトと同期させます。




**Star** スプライトの表示効果を作成し、必要に応じて調整します。

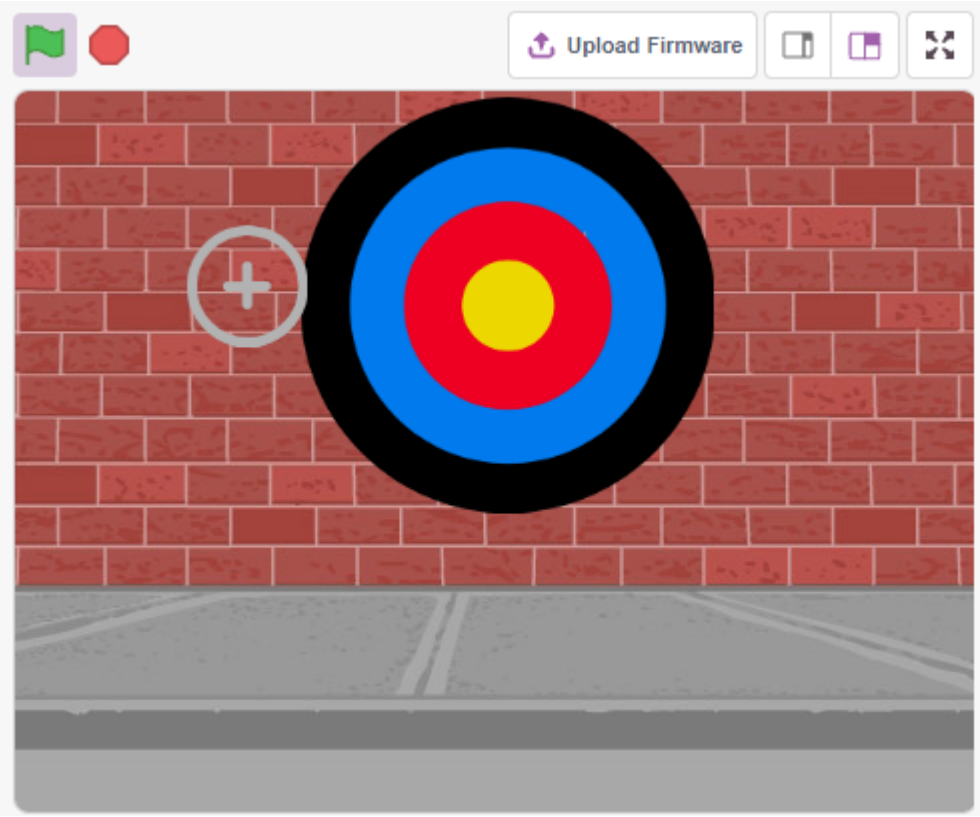



## 7.16 2.13 ゲーム - シューティング

テレビでシューティングゲームを見たことはありますか？ターゲットの中心に近づくほど、得点が高くなります。

今日も Scratch でシューティングゲームを作ります。このゲームでは、クロスヘアでできるだけ中心に近づけてシュートして、高得点を狙います。

緑のフラグをクリックして開始します。障害物回避モジュールを使用して弾を発射します。




### 7.16.1 学べること

- 障害物回避モジュールの仕組みと角度の範囲
- 様々なスプライトの描画
- タッチカラー

### 7.16.2 必要な部品

このプロジェクトには以下のコンポーネントが必要です。

一式を購入するのが便利です、リンクはこちら:

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

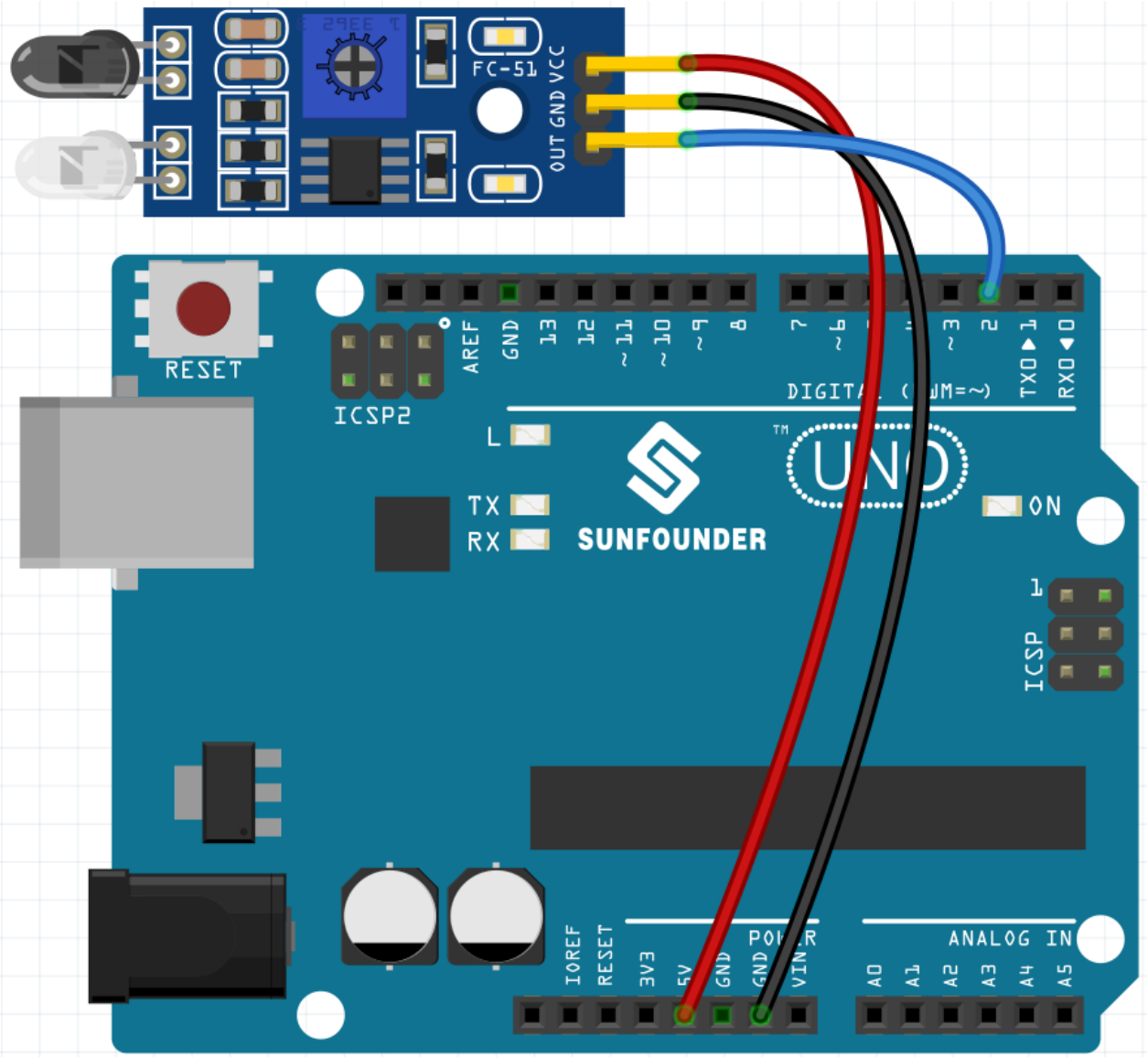
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| 障害物回避モジュール | |

### 7.16.3 回路の作成

障害物回避モジュールは、距離が調整可能な赤外線近接センサーで、通常はハイ出力で、障害物が検出されるとローになります。

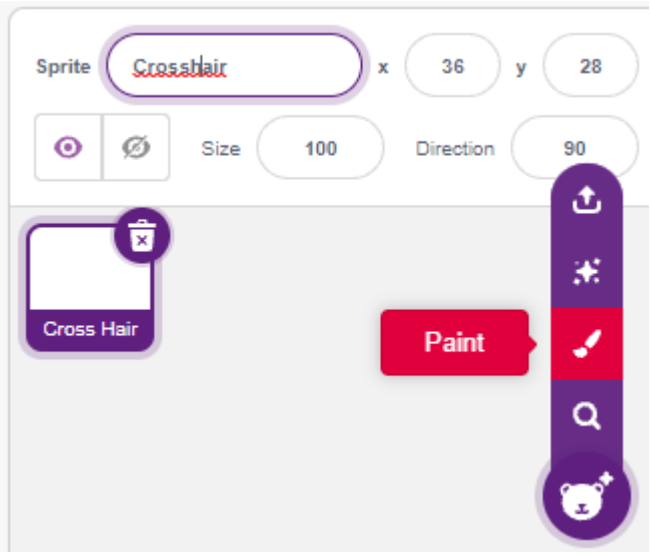
下の図に従って回路を組み立ててください。



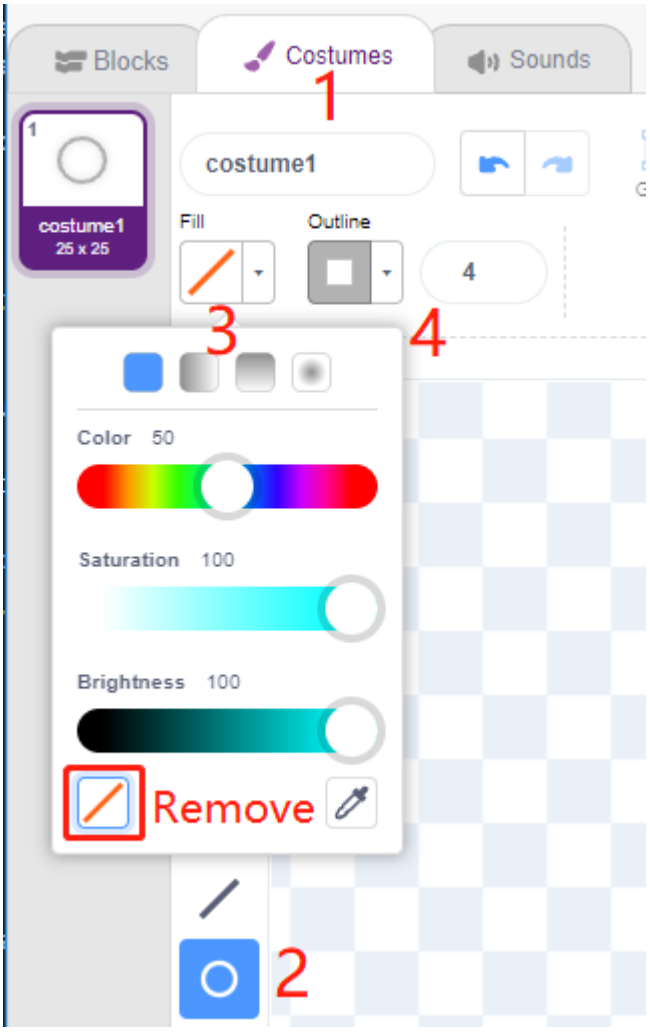

### 7.16.4 プログラミング

#### 1. クロスヘアスプライトの描画

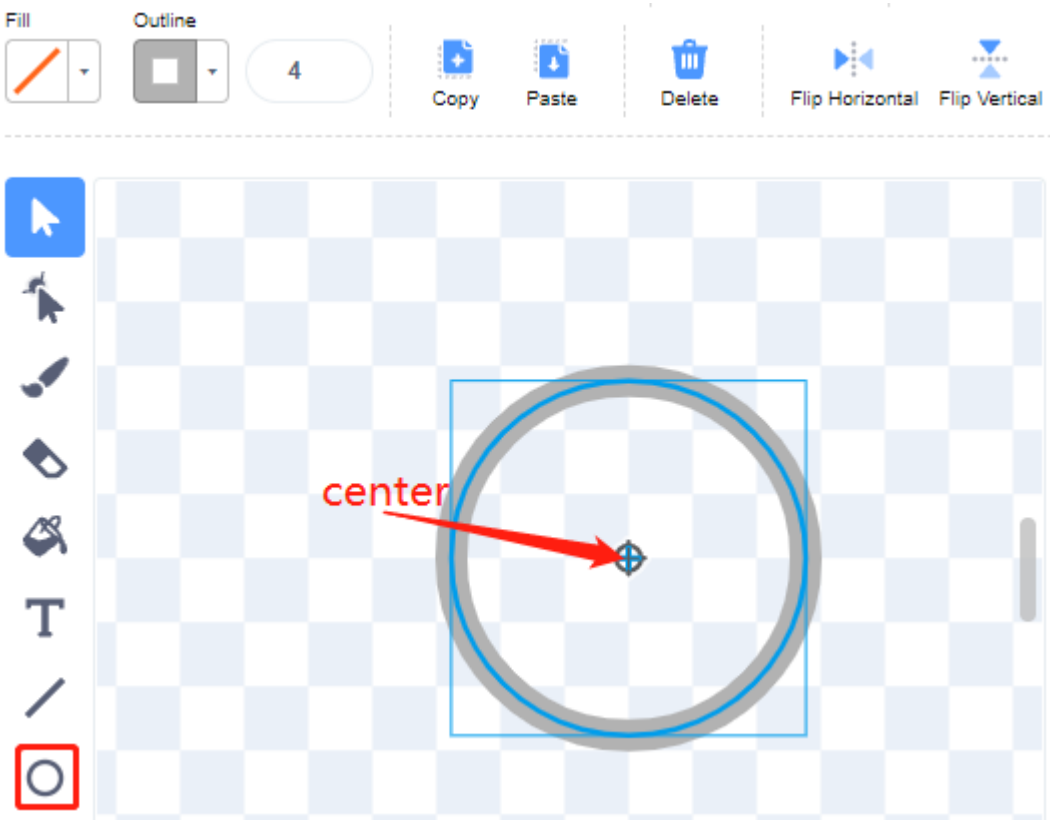
デフォルトのスプライトを削除し、 **Sprite** ボタンを選択し、 **Paint** をクリックすると、空白のスプライト **Sprite1** が現れますので、名前を **Crosshair** とします。



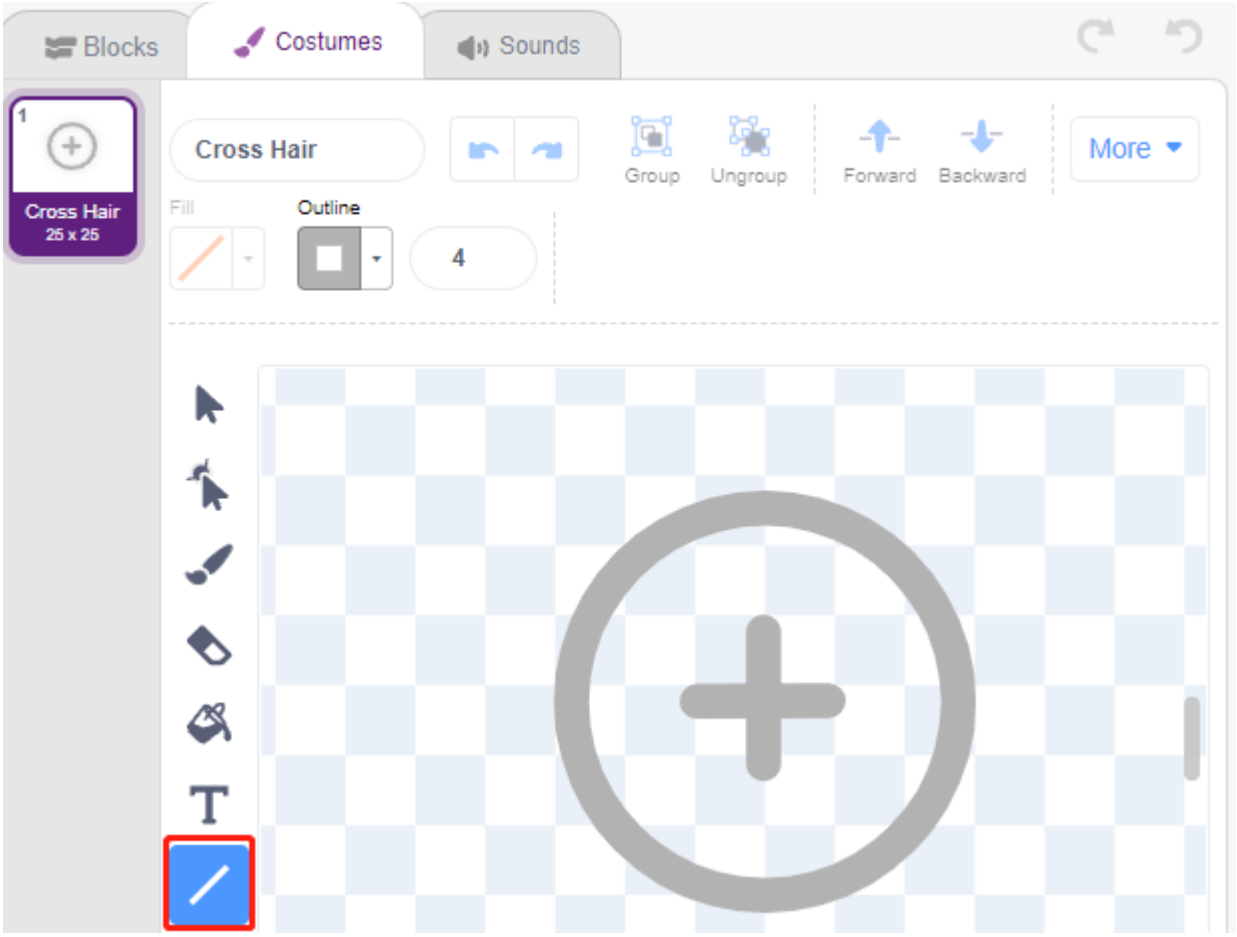

**Crosshair** スプライトの **Costumes** ページに移動します。 **Circle** ツールをクリックし、塗りつぶし色を削除し、アウトラインの色と幅を設定します。



**Circle** ツールで円を描きます。描画が終わったら、 **Select** ツールをクリックして、原点がキャンバスの中心に合わせるように円を移動します。

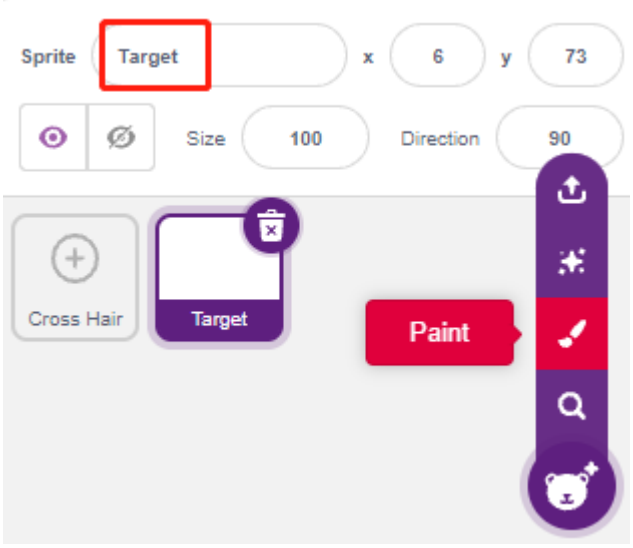



**Line** ツールを使用して、円の中に十字を描きます。



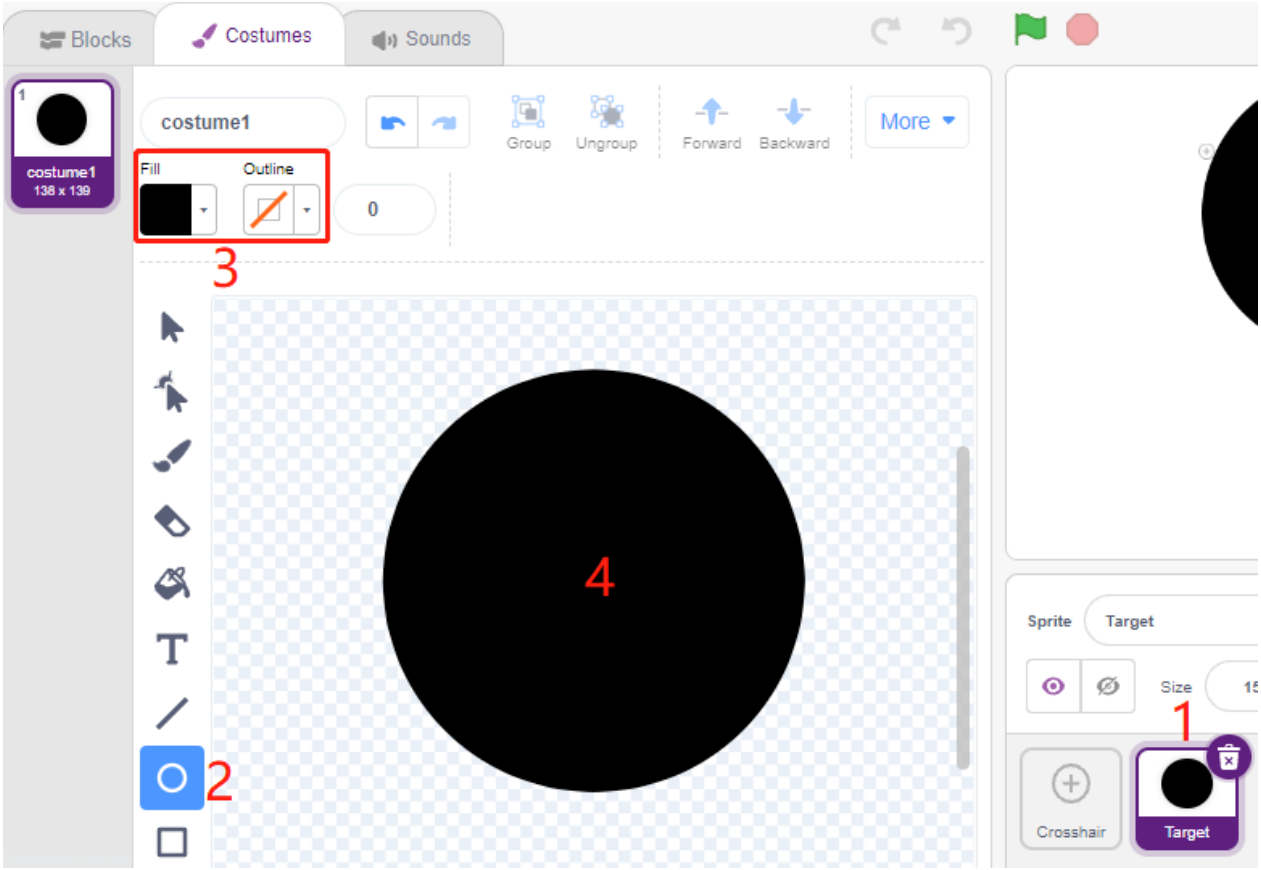

**ターゲットスプライトの描画**

新しいスプライトとして **Target** スプライトを作成します。

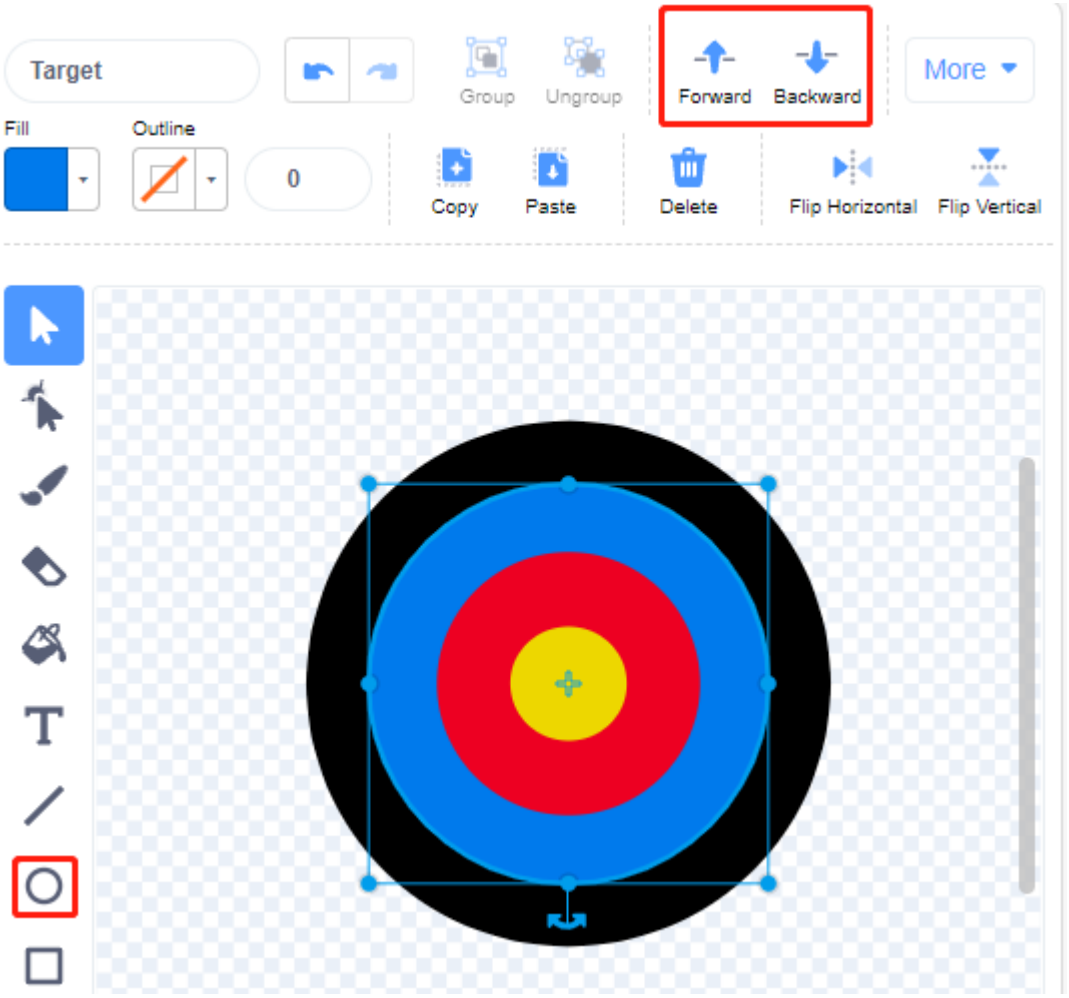



**Target** スプライトのコスチュームページに移動し、 **Circle** ツールをクリックして、塗りつぶし色を選択し、アウ

トラインを削除して大きな円を描画します。

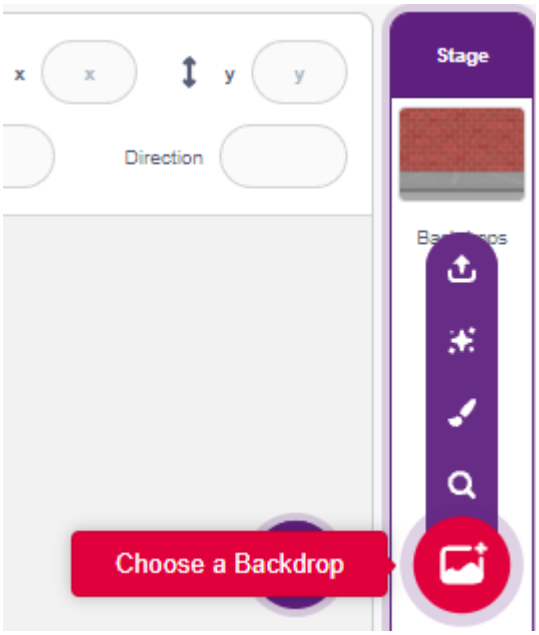



同じ方法で、異なる色の円を追加で描画します。 **Forward** や **Backbard** ツールを使用して、重なる円の位置を変更できます。すべての円の原点とキャンバスの中心が合うように、ツールで円を移動することも忘れないでください。



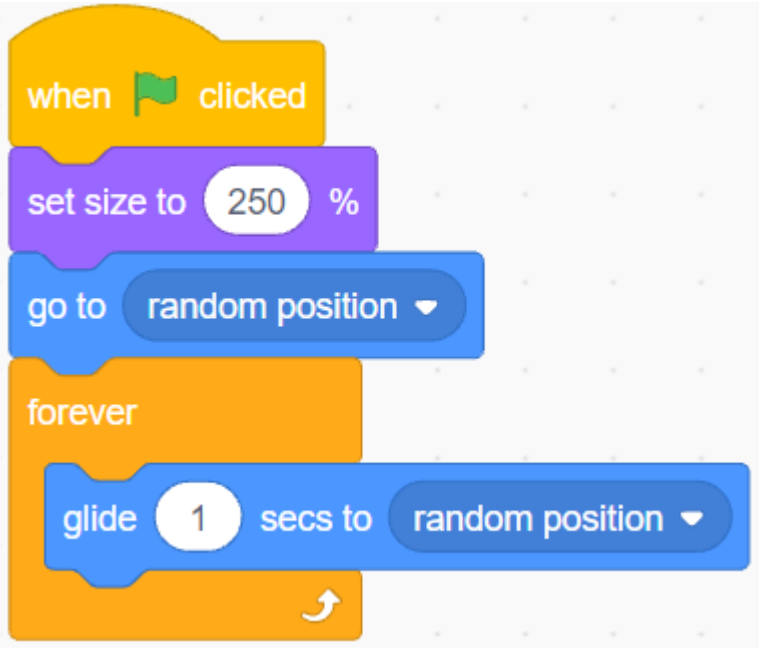

**3. 背景の追加**

色が多すぎず、 **Target** スプライトの色と一致しない背景を追加します。ここでは **Wall1** の背景を選択しました。

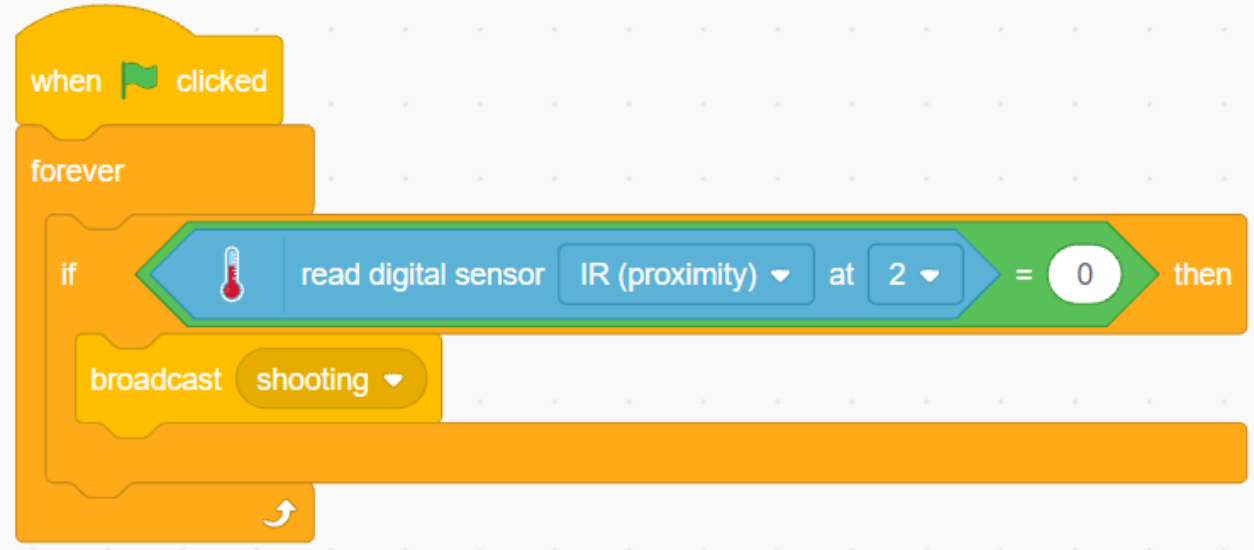



#### 4. クロスヘアスプライトのスクリプト

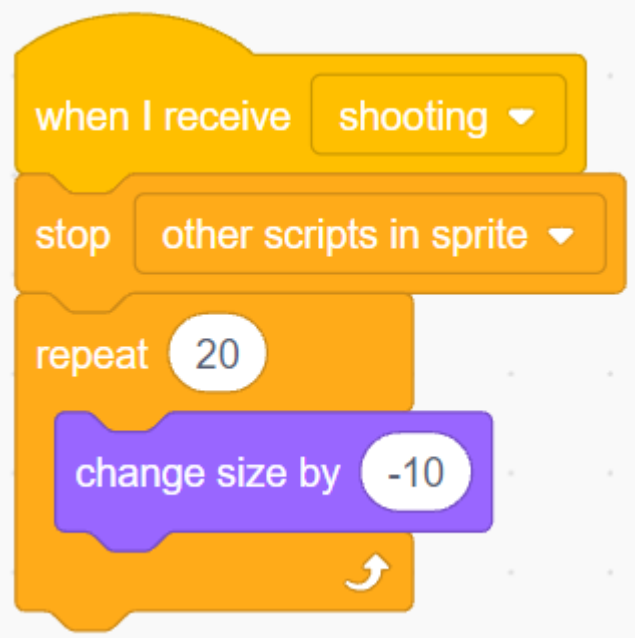
**Crosshair** スプライトの位置とサイズをランダムに設定し、ランダムに動かします。



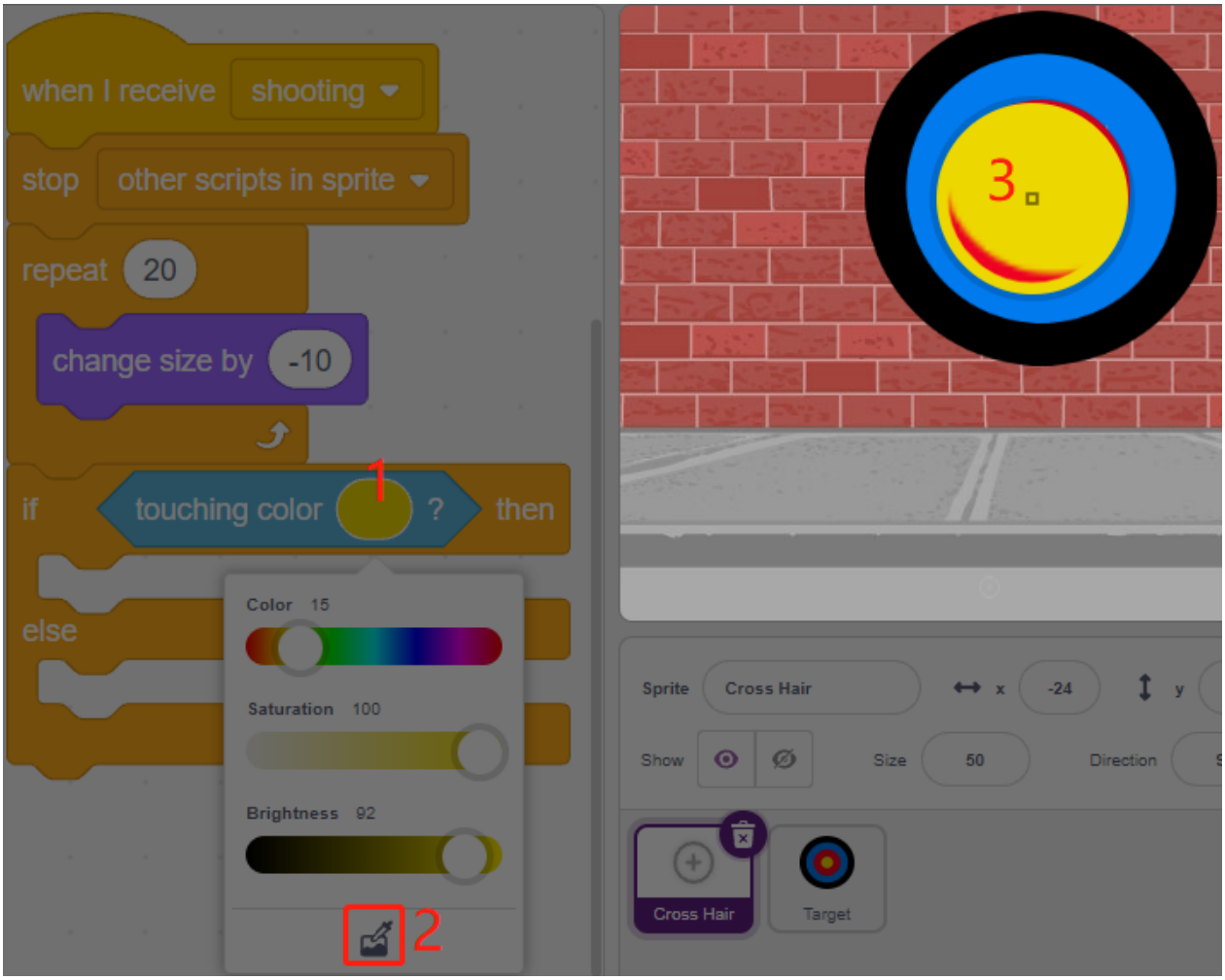

障害物回避モジュールの前に手を置くと、低レベルを送信信号として出力します。



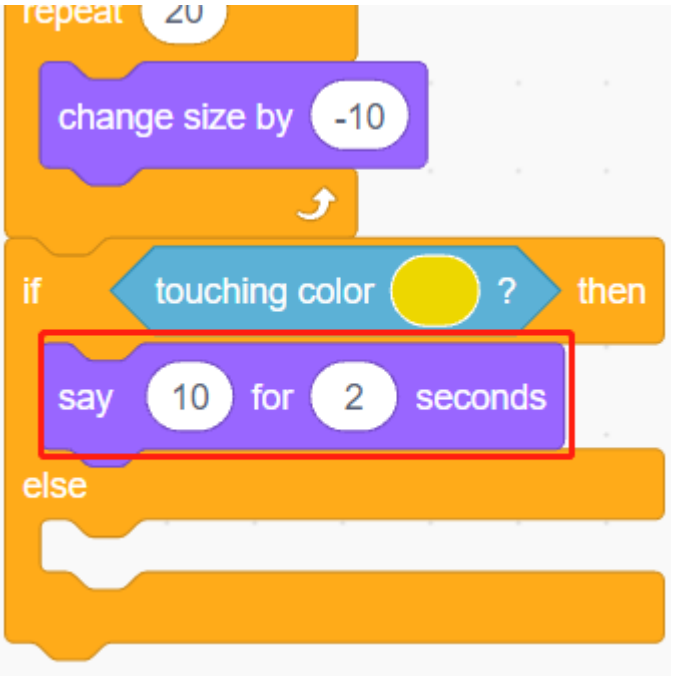

**shooting** メッセージを受け取ると、スプライトは動きを停止し、ゆっくりと縮小します。これは、弾が発射される効果をシミュレートしています。



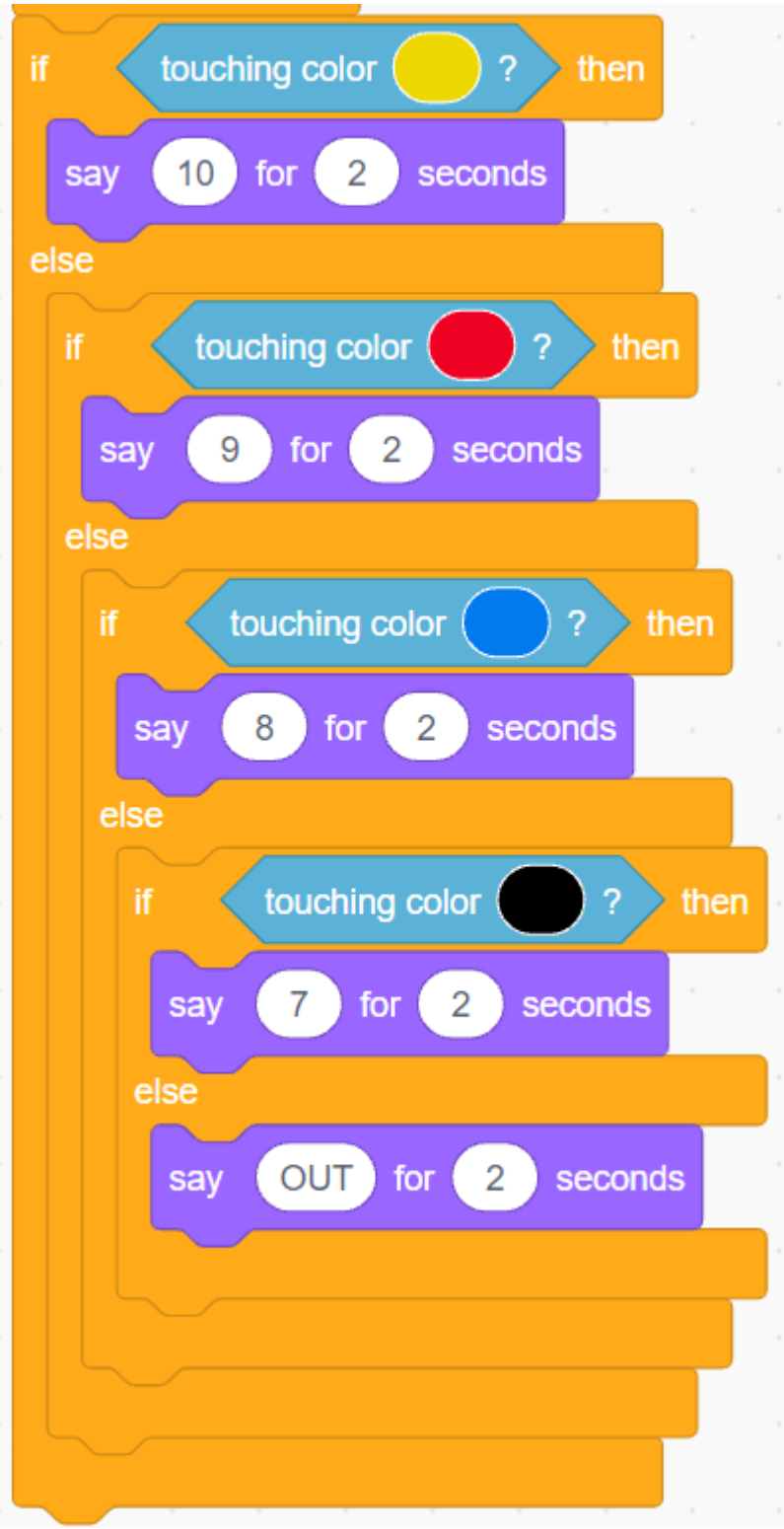

[Touch color ()] ブロックを使用して、ショットの位置を判断します。




ショットが黄色い円の中にある場合、10 が報告されます。



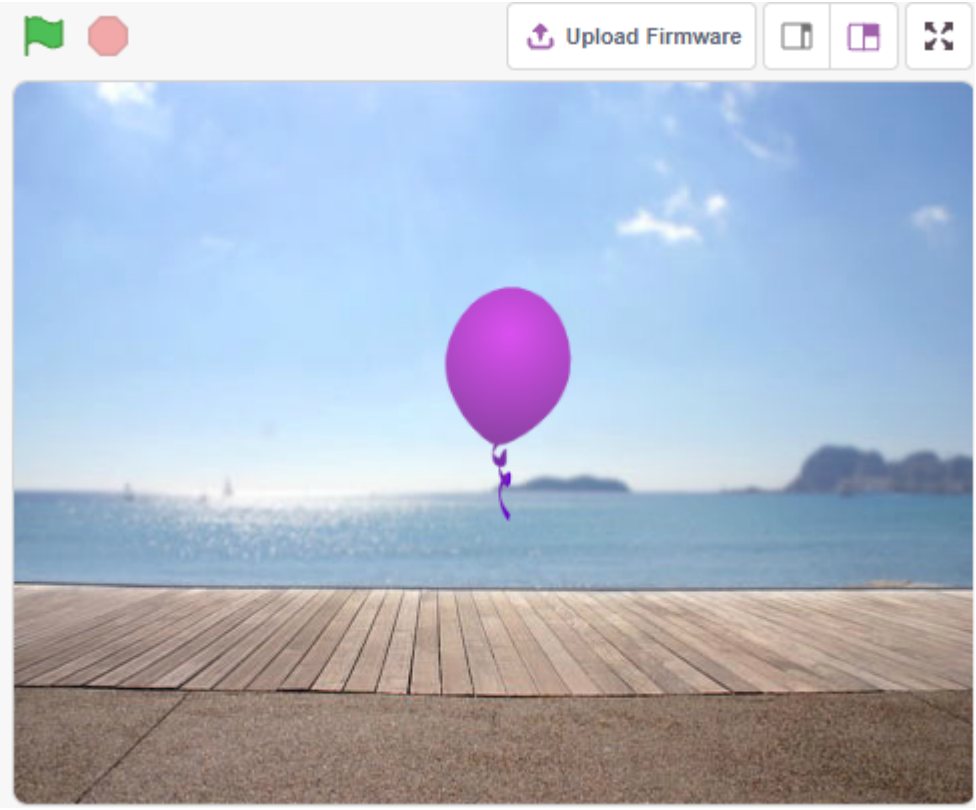

同じ方法を使用して、弾のショット位置を判断します。それが **Target** スプライトに設定されていない場合、円の外にあることを意味します。


if touching color ? then
say 10 for 2 seconds
else
if touching color ? then
say 9 for 2 seconds
else
if touching color ? then
say 8 for 2 seconds
else
if touching color ? then
say 7 for 2 seconds
else
say OUT for 2 seconds

## 7.17 2.14 ゲーム - 風船を膨らます

ここでは、風船を膨らませるゲームをします。

緑のフラグをクリックすると、風船がどんどん大きくなります。風船が大きすぎると爆発します。逆に、小さすぎると落ちてしまいます。いつボタンを押して上に飛ばすかの判断が求められます。




### 7.17.1 学べること

- スプライトのコスチュームをペイントする

### 7.17.2 必要な部品

このプロジェクトに必要なコンポーネントは以下の通りです。

一式を購入すると便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

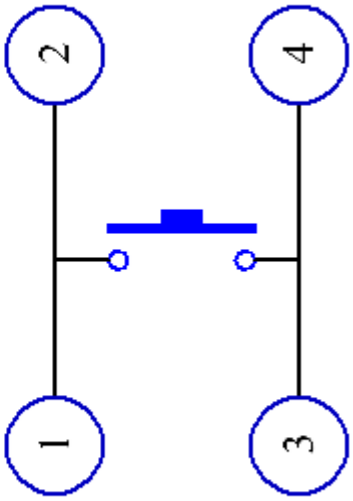
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| コンデンサ | |
| ボタン | |

### 7.17.3 回路の作成

ボタンは 4 ピンのデバイスです。ピン 1 はピン 2 に、ピン 3 はピン 4 に接続されています。ボタンが押されると、4 つのピンが接続され、回路が閉じます。

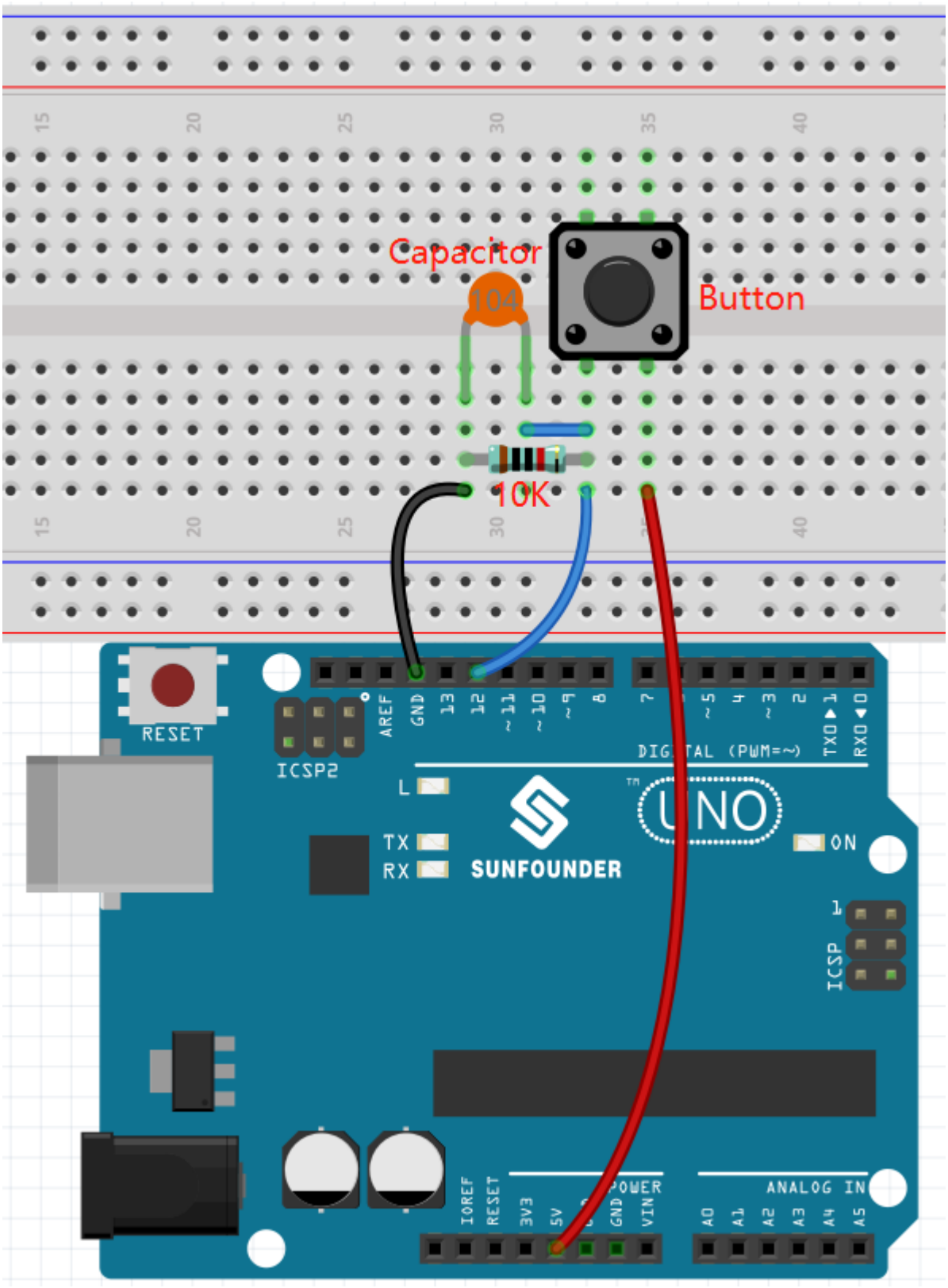

Button




Internal Structure

以下の図に従って回路を組み立ててください。

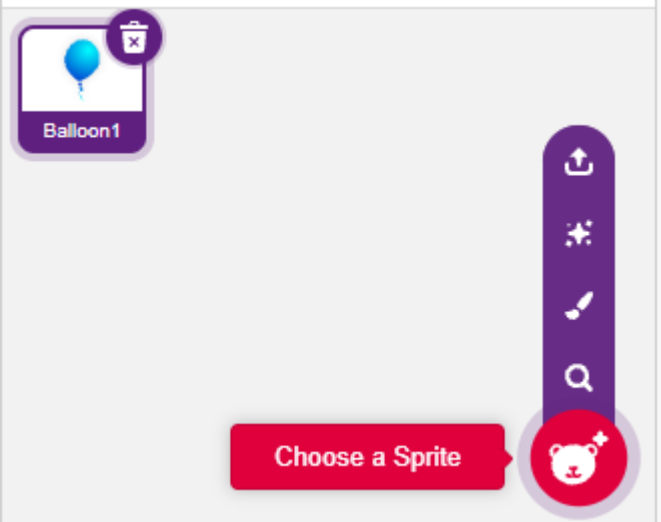
- ボタンの左側のピンの 1 つを、プルダウン抵抗および 0.1uF（104）キャパシタ（ボタンの動作時にジッタを除去し、安定したレベルを出力するため）に接続されているピン 12 に接続します。
- 抵抗とキャパシタの他端を GND に、ボタンの右側のピンの 1 つを 5V に接続します。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

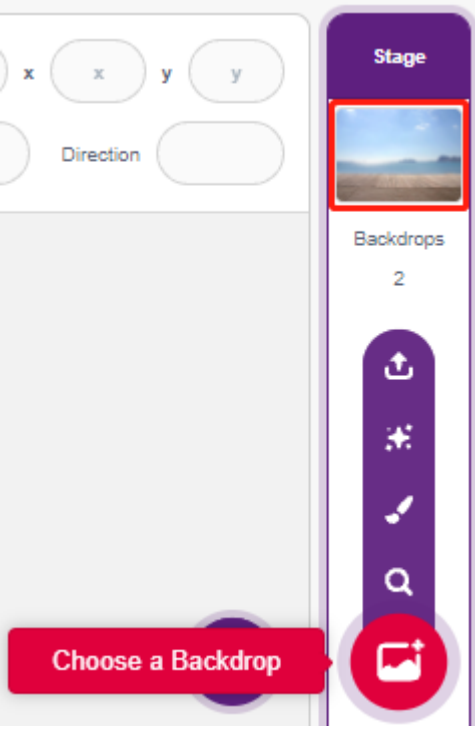
### 7.17.4 プログラミング

#### 1. スプライトと背景の追加

デフォルトのスプライトを削除し、スプライトエリアの右下の **Choose a Sprite** ボタンをクリックして、 **Balloon1** スプライトを選択します。



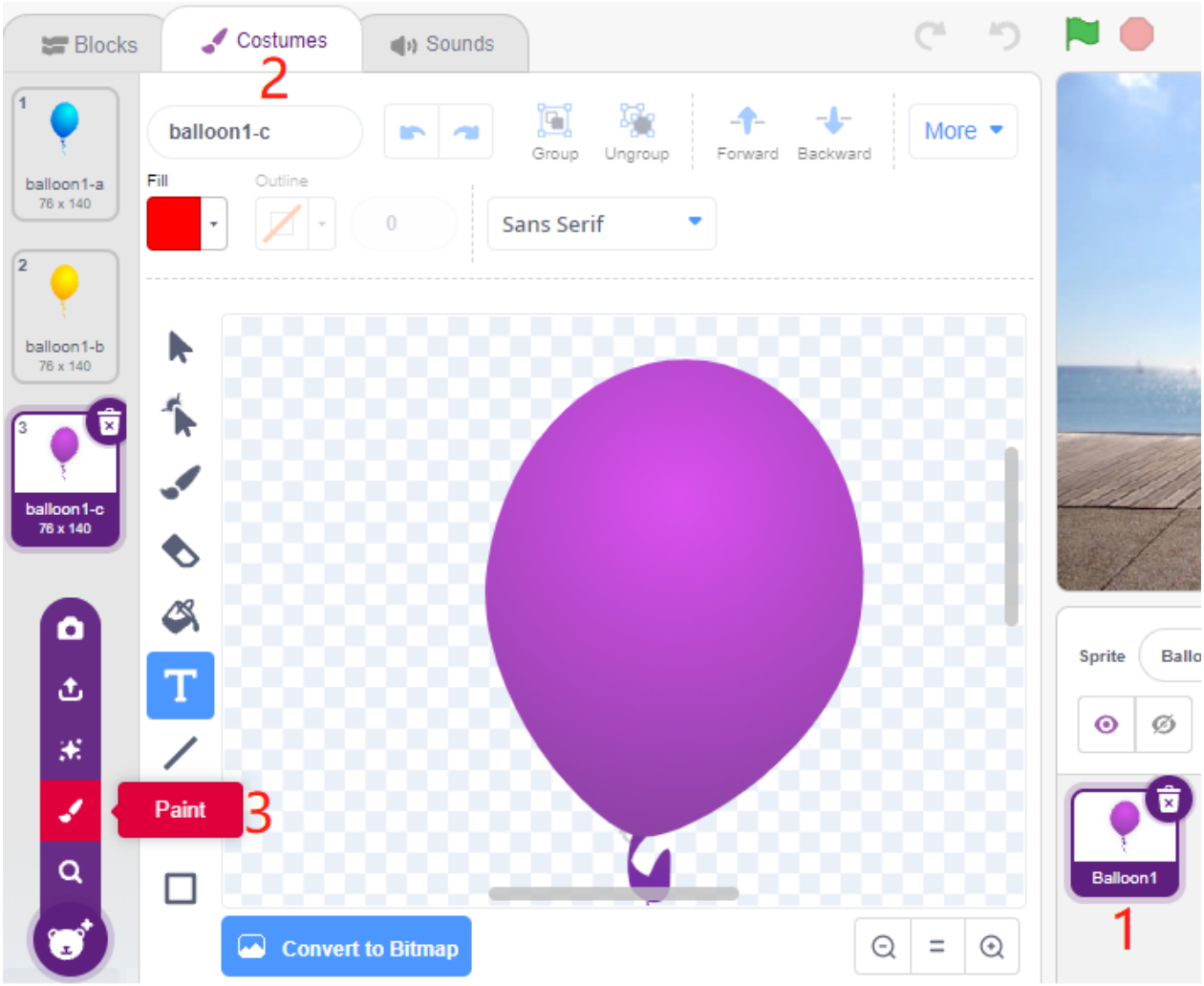

**Choose a backdrop** ボタンを使って **Boardwalk** の背景を追加します。または、好きな背景を選んでください。



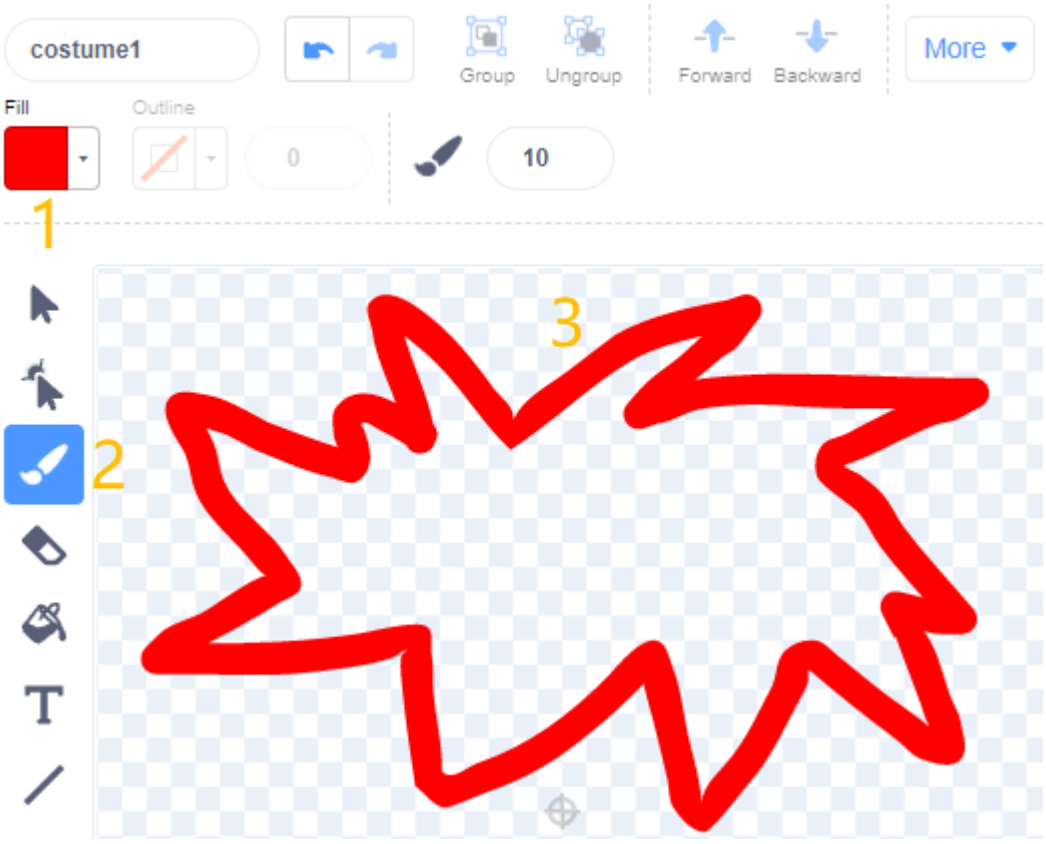

#### 2. Balloon1 スプライトのコスチュームを描画

風船スプライトの爆発エフェクトのコスチュームを描画しましょう。

**Balloon1** スプライトの **Costumes** ページに移動し、左下の **Choose a Costume** ボタンをクリックし、 **Paint** を選択して、空の **Costumes** 画面を開きます。

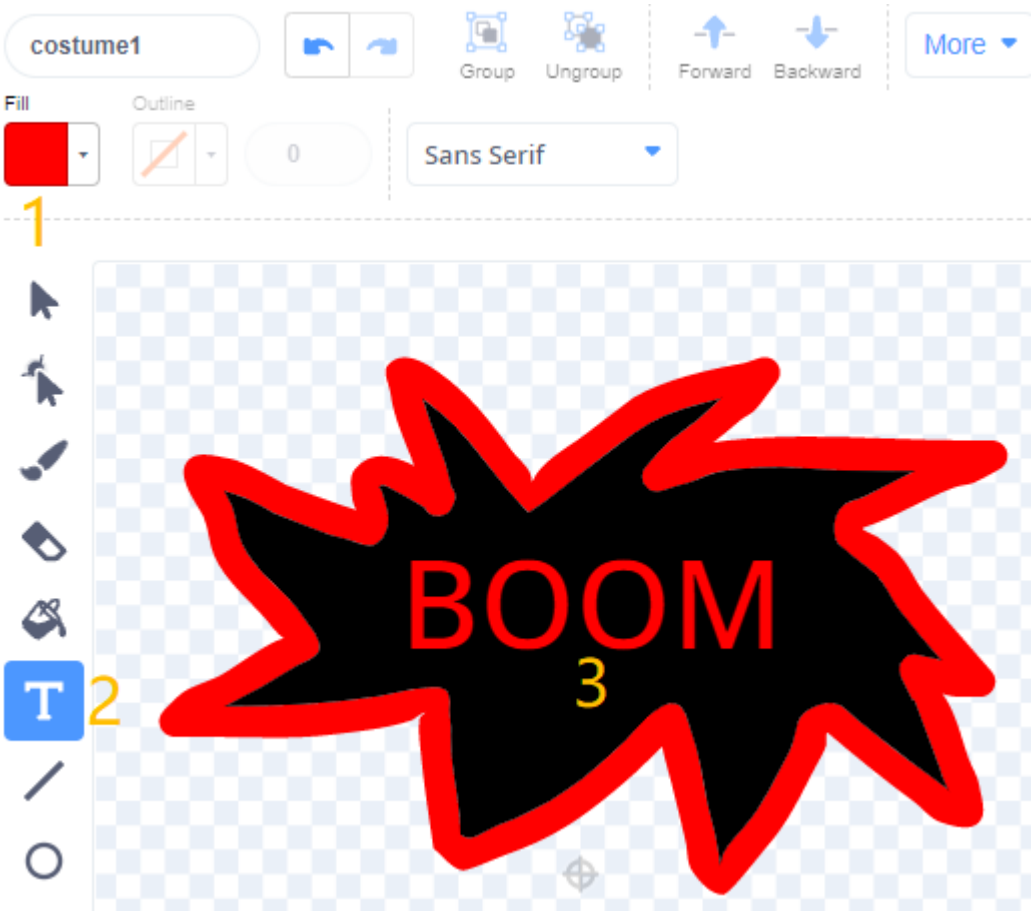



色を選んで **Brush** ツールを使って模様を描きます。



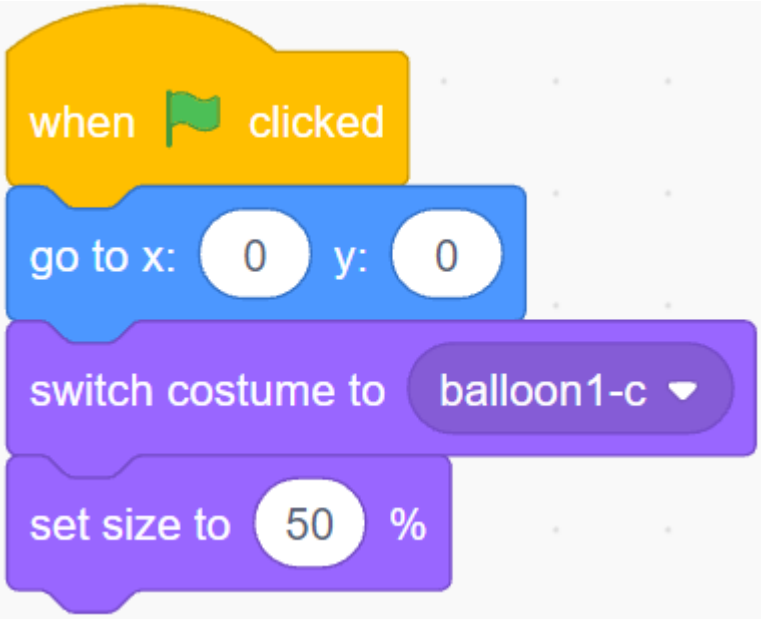

再度色を選択し、塗りつぶしツールをクリックし、マウスを模様の内部に移動して色を塗りつぶします。


最後に、BOOM というテキストを書いて、爆発エフェクトのコスチュームが完成します。

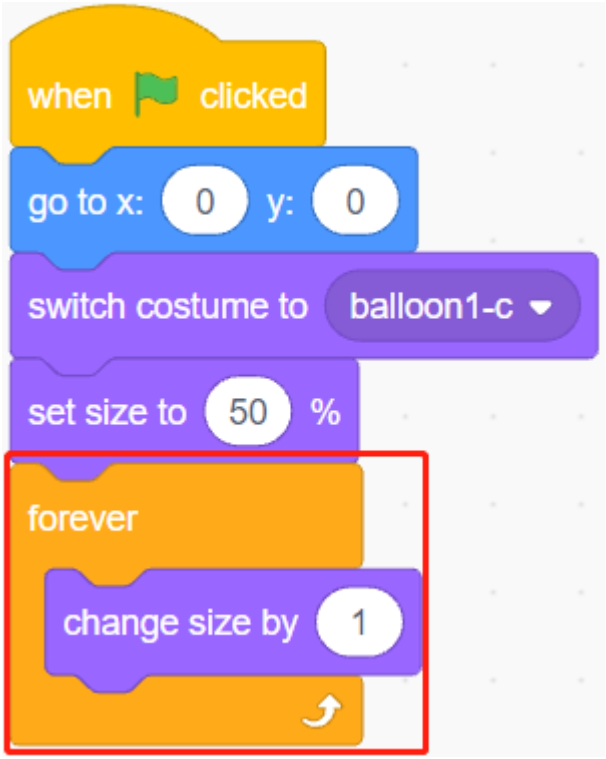



**3. Balloon スプライトのスクリプティング**

**Balloon1** スプライトの初期位置とサイズを設定します。

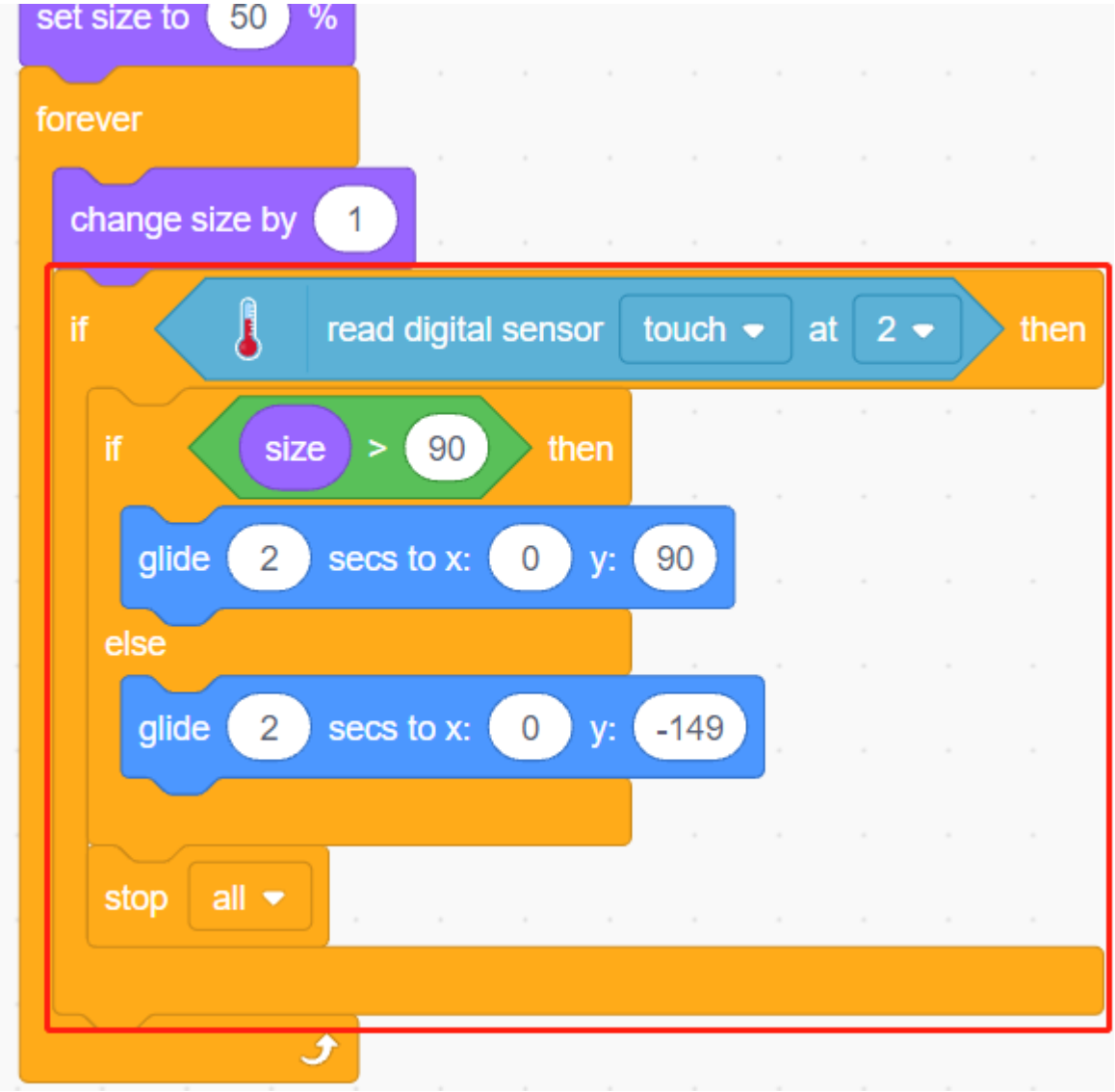


次に、**Balloon** スプライトをゆっくりと大きくします。

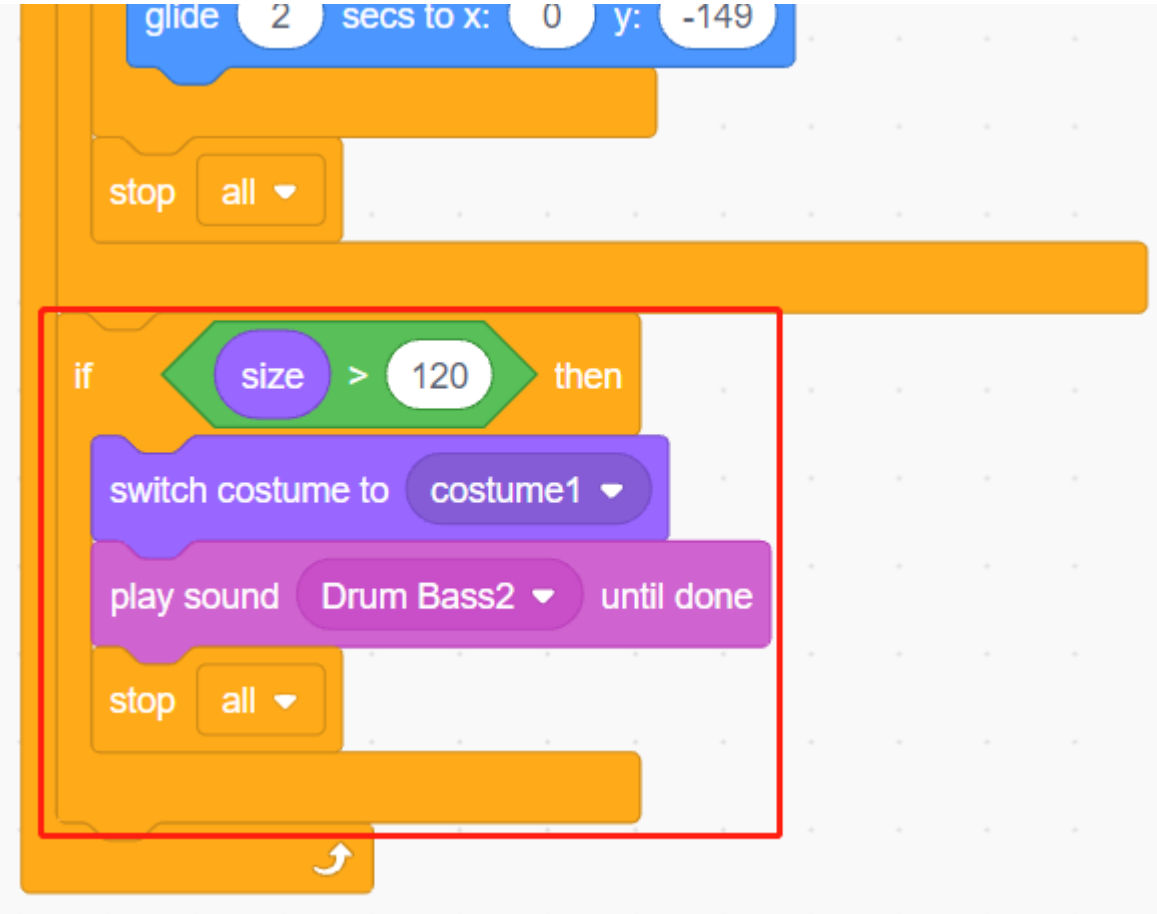



ボタンが押されたとき（値が 1 の場合）、 **Balloon1** スプライトのサイズの増大が停止します。

- サイズが 90 未満の場合、落ちる（y 座標が減少）。
- サイズが 90 より大きく、120 より小さい場合、空に飛ぶ（y 座標が増加）。




ボタンが押されていない場合、風船はゆっくりと大きくなり、サイズが 120 より大きくなると、爆発します（爆発エフェクトのコスチュームに切り替えます）。

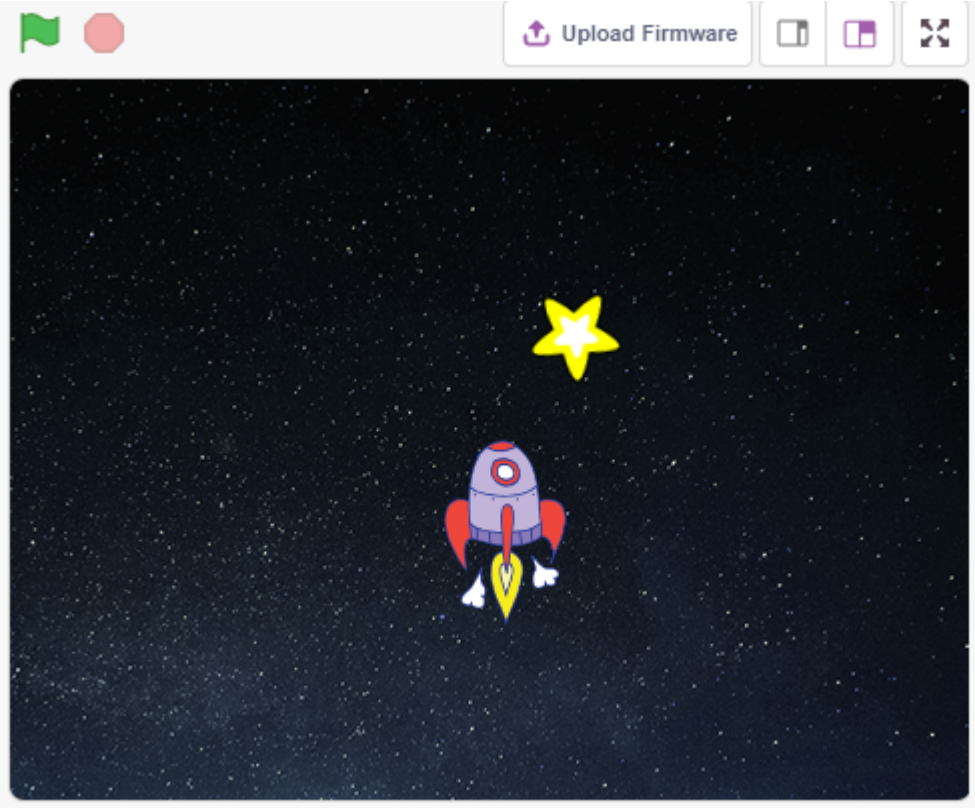



## 7.18 2.15 GAME - スタークロスド

次のプロジェクトでは、PictoBlox で楽しいミニゲームをプレイします。

このゲームではジョイスティックモジュールを使用して「スタークロスド」というゲームをプレイします。

スクリプトが実行されると、星がランダムにステージ上に表示されます。ロケットを操作して星を避けるためにジョイスティックを使ってください。星に触れるとゲームオーバーとなります。




### 7.18.1 学べること

- ジョイスティックモジュールの仕組み
- スプライトの x、y 座標の設定

### 7.18.2 必要な部品

このプロジェクトで必要なコンポーネントは以下の通りです。

キットで一式を購入すると便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから部品を個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| ジョイスティックモジュール | - |

### 7.18.3 回路の作成

ジョイスティックは、基盤上でピボットとして動くスティックで構成された入力デバイスです。これは主にビデオゲームやロボットの制御に使われます。

完全な動きをコンピューターに伝えるため、ジョイスティックはスティックの位置を二つの軸、X 軸（左から右）と Y 軸（上から下）で測定する必要があります。

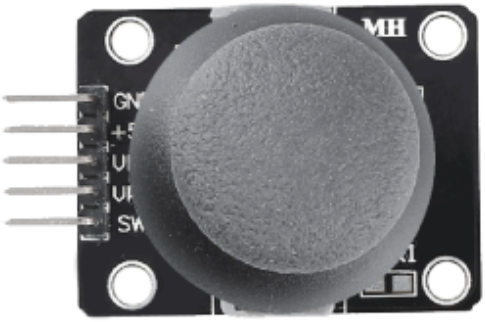
ジョイスティックの動きの座標は次の図で示されています。

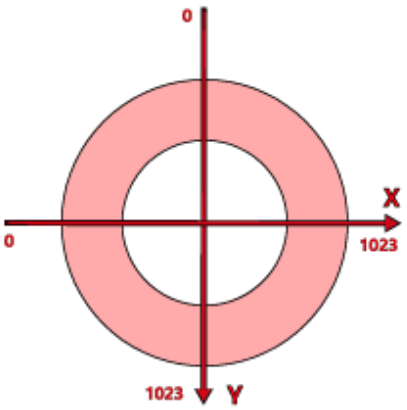
---

**注釈:**

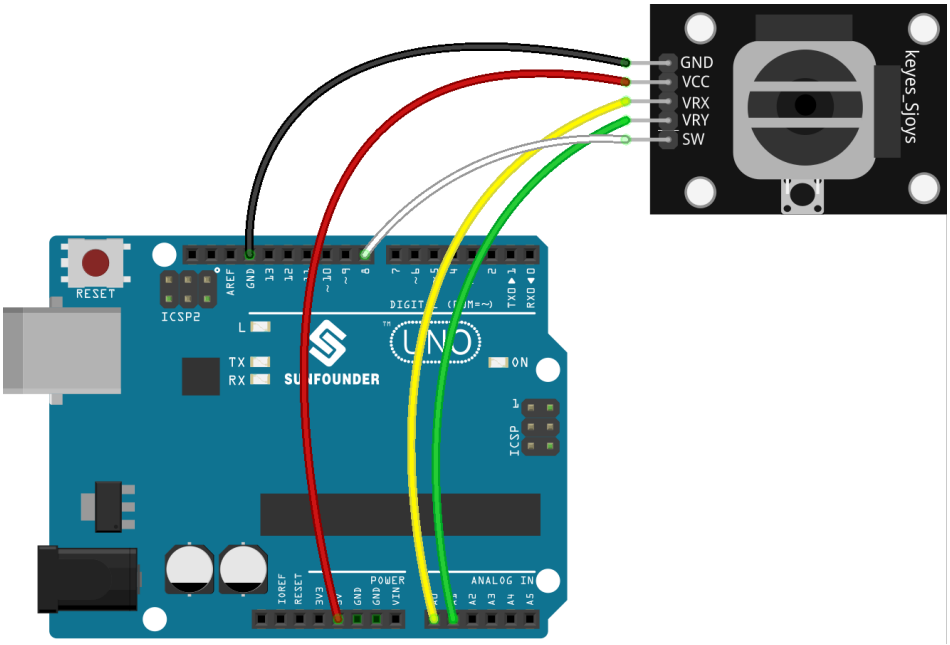
- x 座標は左から右、範囲は 0-1023。
- y 座標は上から下、範囲は 0-1023。

---




次に、以下の図に従って回路を組み立ててください。

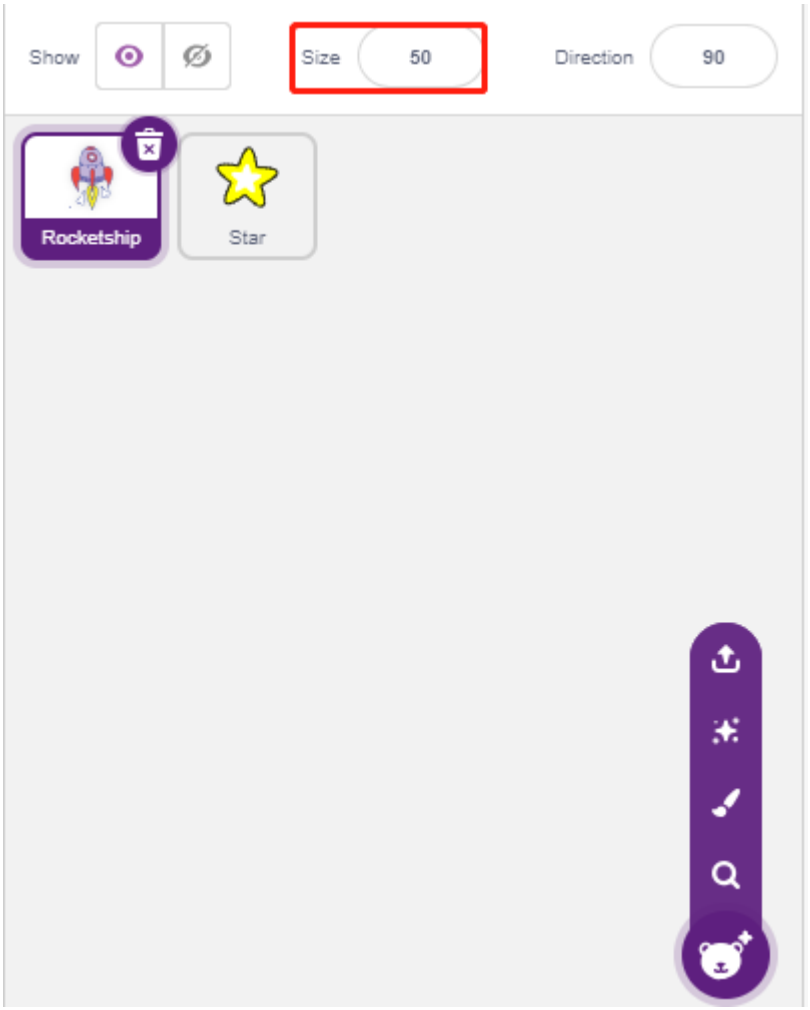



### 7.18.4 プログラミング

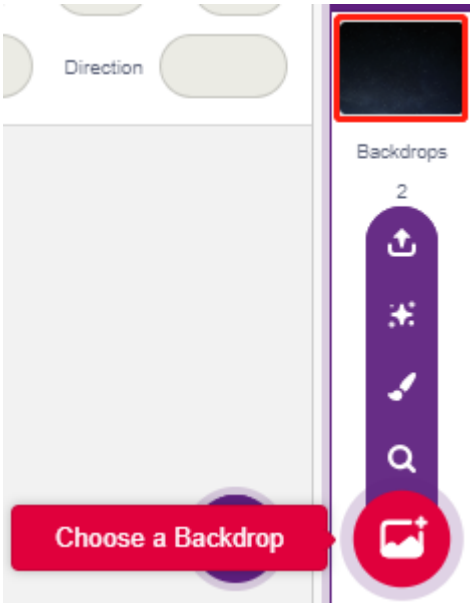
このスクリプトの全体の目的は、緑のフラグがクリックされると、 **Stars** スプライトがステージ上で曲線を描いて動き、ジョイスティックを使用して **Rocketship** を動かし、 **Star** スプライトに触れないようにすることです。

#### 1. スプライトと背景の追加

デフォルトのスプライトを削除し、 **Choose a Sprite** ボタンを使用して **Rocketship** スプライトと **Star** スプライトを追加します。 **Rocket** スプライトのサイズは 50% に設定してください。




次に、 **Choose a Backdrop** で **Stars** の背景を追加してください。

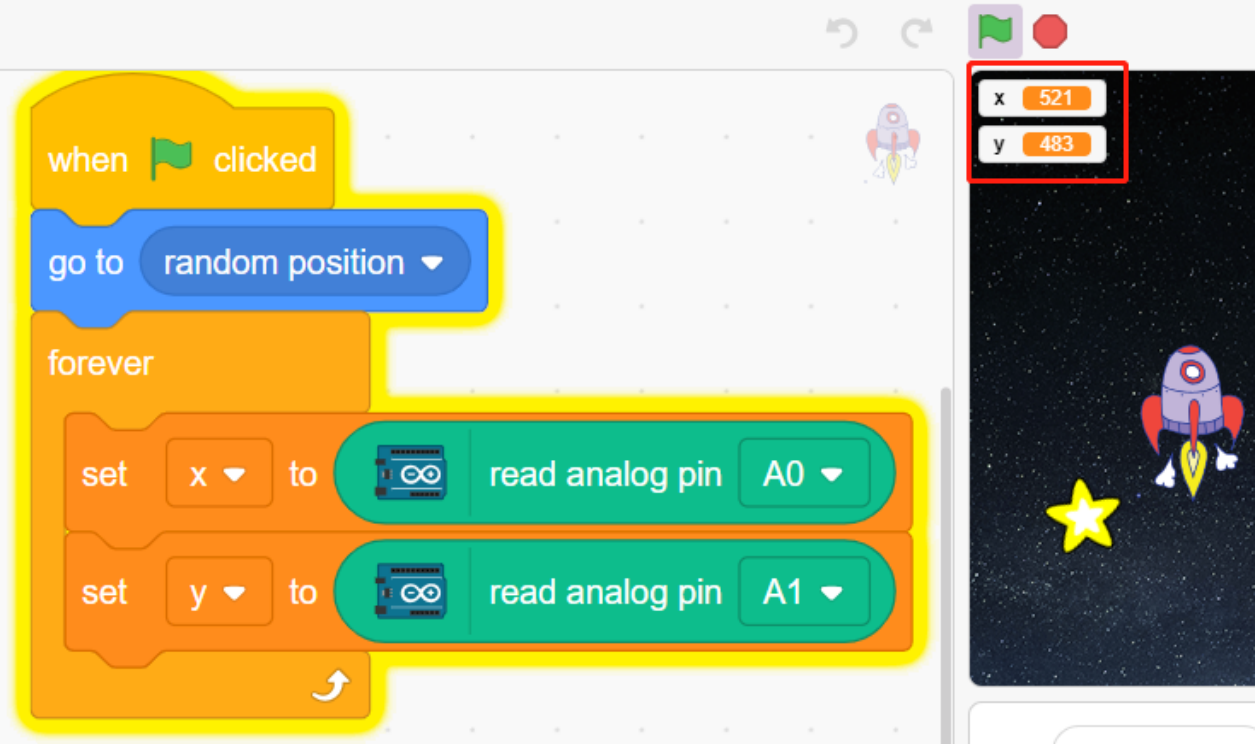


### 2. Rocketship のスクリプティング

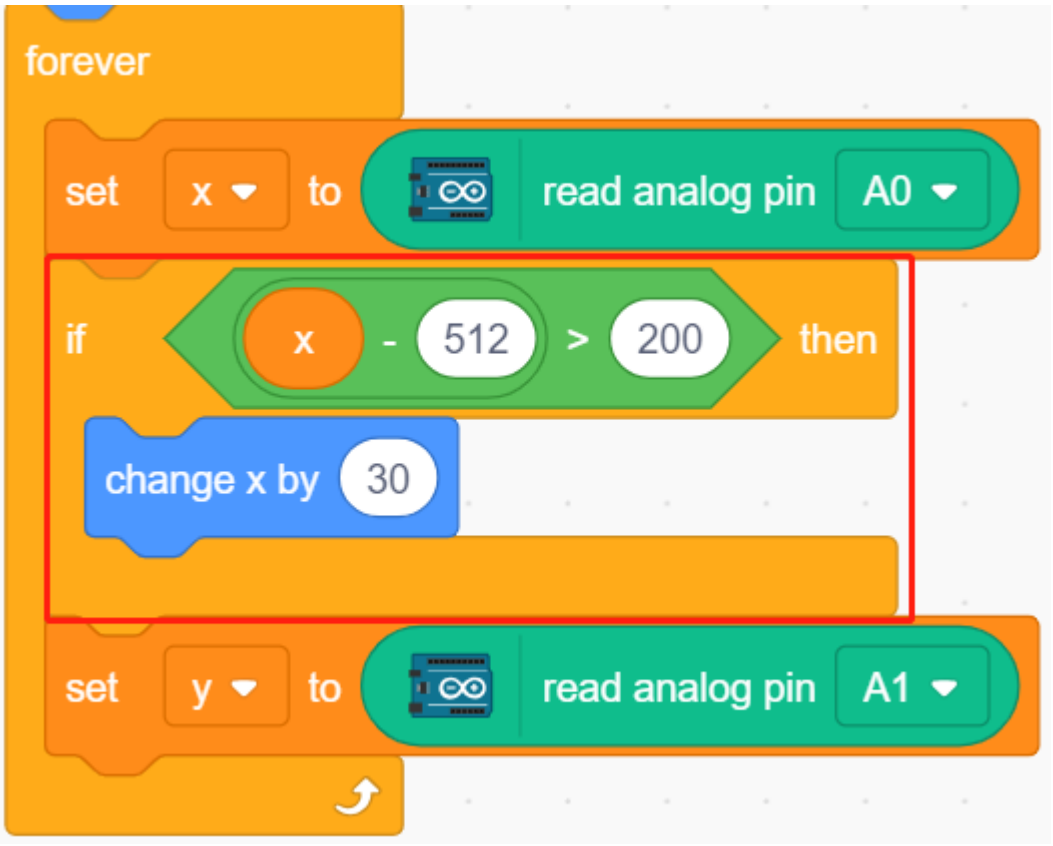
**Rocketship** スプライトは、ランダムな位置に表示され、次にジョイスティックで上、下、左、右に動かす効果を実現します。

作業の流れは以下の通りです。

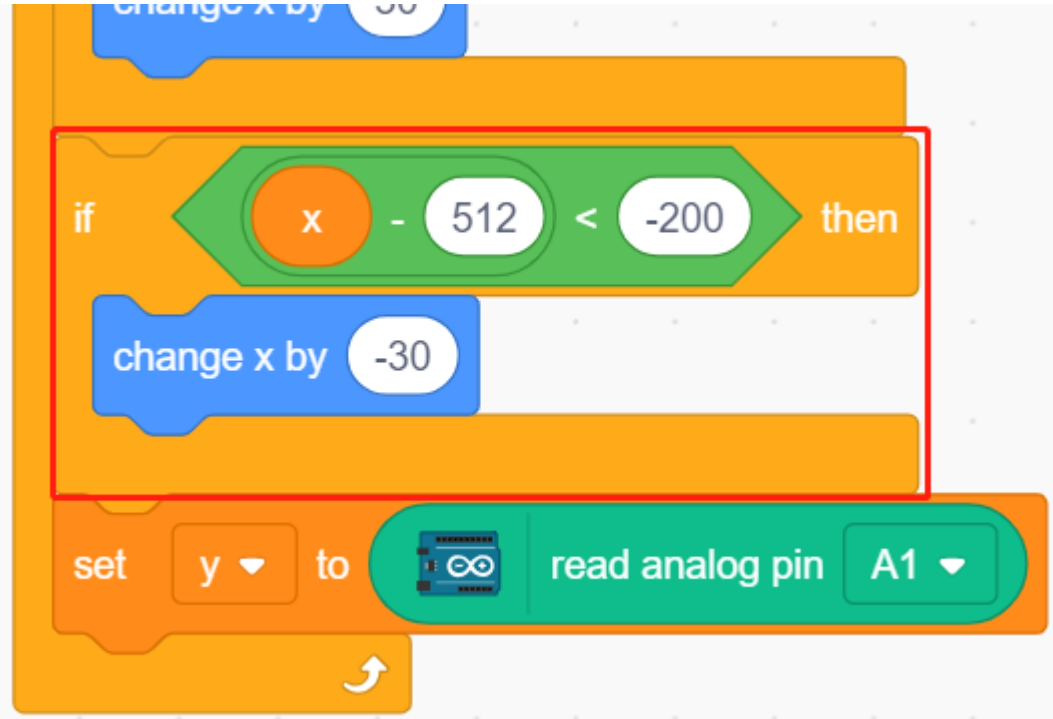
- 緑のフラグがクリックされると、スプライトはランダムな位置に移動し、2 つの変数 **x** と **y** を作成します。これらは、A0（ジョイスティックの VRX）と A1（ジョイスティックの VRY）から読み取られる値をそれぞれ格納します。スクリプトを実行し、ジョイスティックを上下、左右に切り替えて、x と y の値の範囲を確認することができます。



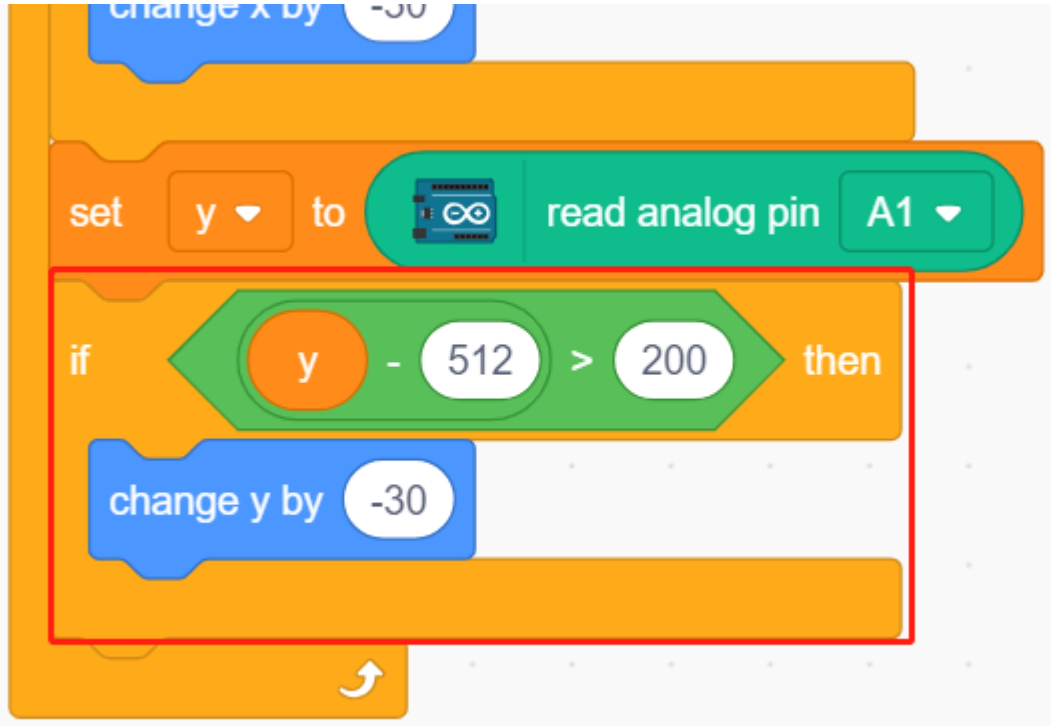

- A0 の値は 0-1023 の範囲内にあり（中央は約 512）、`x-512>200` を使用してジョイスティックが右に切り替えられているかどうかを判断し、そうであれば、スプライトの x 座標を +30 にします（スプライトを右に移動）。




- ジョイスティックが左に切り替えられている場合 (`x-512<-200`)、スプライトの x 座標は-30 にします（スプライトを左に移動)。



- ジョイスティックの y 座標は上（0）から下（1023）へ、そしてスプライトの y 座標は下から上へです。そのため、ジョイスティックを上に動かし、スプライトを上に動かすためには、スクリプト内の y 座標は-30 でなければなりません。

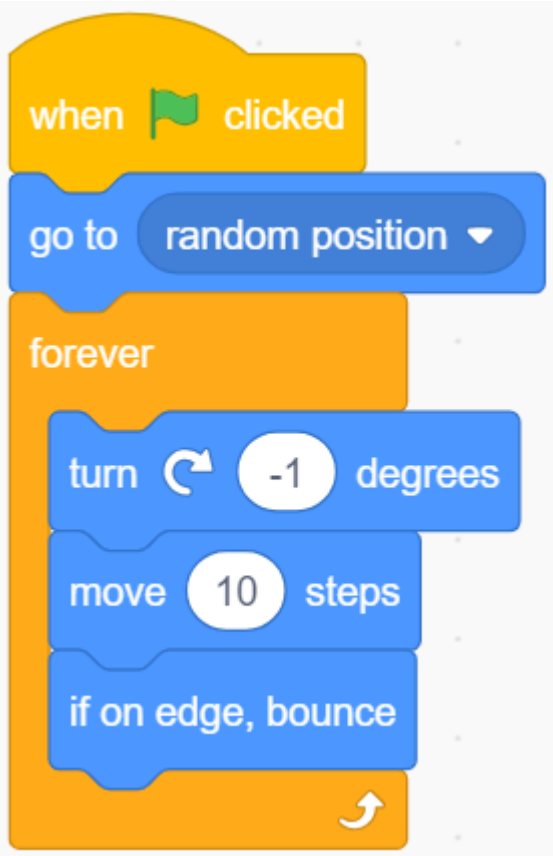



- ジョイスティックが下にフリックされると、スプライトの y 座標は +30 になります。

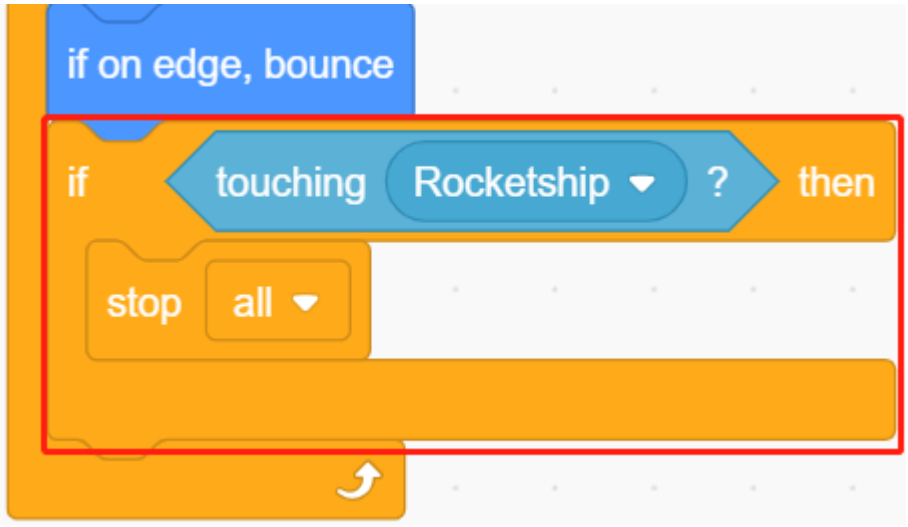

### 3. Star のスクリプティング

**Star** スプライトの目的は、ランダムな位置に表示され、 **Rocketship** に当たった場合は、スクリプトが停止してゲームが終了することです。

- 緑のフラグがクリックされ、スプライトがランダムな位置に移動すると、[turn degrees] ブロックは **Star** スプライトを少し角度を変えて前進させ、曲線で動いているように見えるようにします。もしエッジに当たったら、バウンスします。



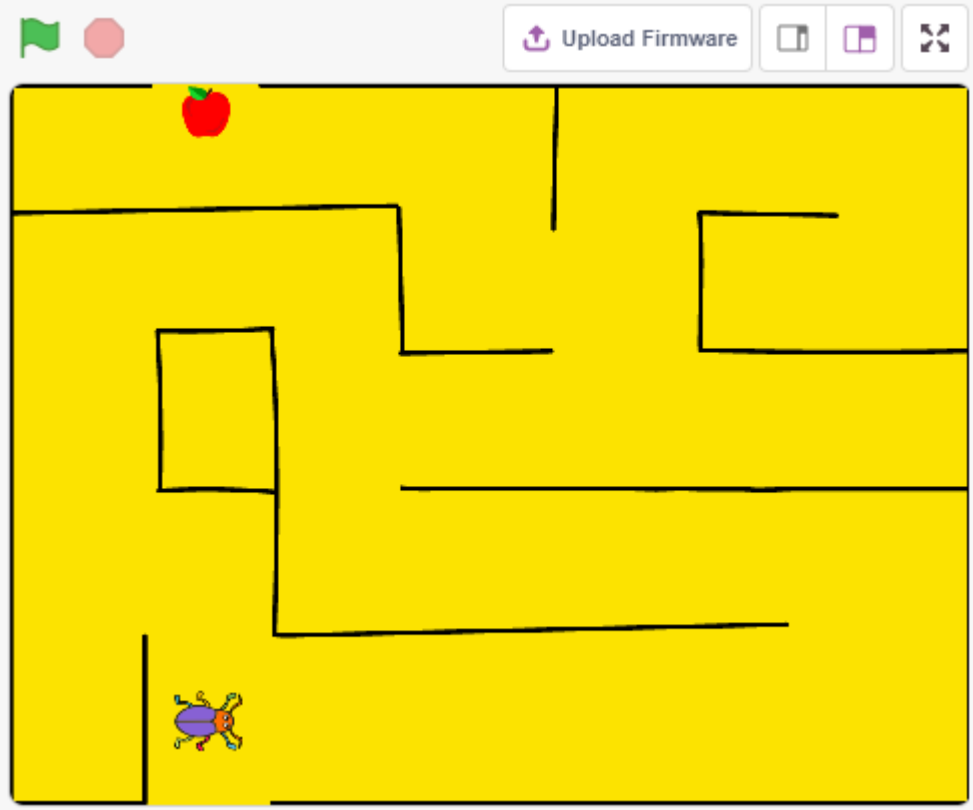

- スプライトが動いている間に **Rocketship** スプライトに触れた場合、スクリプトの実行を停止します。



## 7.19 2.16 ゲーム - りんごを食べる

このプロジェクトでは、ボタンを使用してビートルを操作し、りんごを食べるゲームを楽しみます。

緑のフラグをクリックした後、ボタンを押すとビートルは回転します。ボタンを再度押すとビートルはその角度で直進します。マップ上の黒い線に触れずに、適切な角度でビートルを操作し、りんごを食べるまで進めてください。もし黒い線に触れてしまったら、ゲームオーバーとなります。




### 7.19.1 必要な部品

このプロジェクトで必要な部品は以下の通りです。

全ての部品が含まれたキットを購入するのは非常に便利です。リンクはこちら：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

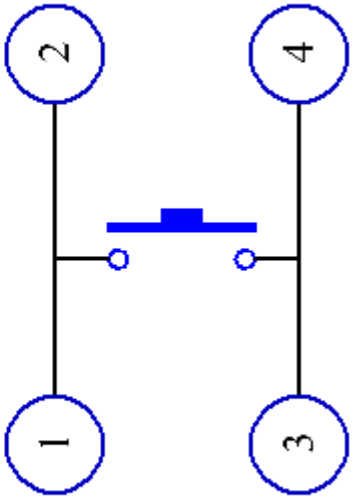
以下のリンクから、部品を個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| コンデンサ | |
| ボタン | |

### 7.19.2 回路の作成

ボタンは 4 ピンのデバイスです。ピン 1 はピン 2 に、ピン 3 はピン 4 に接続されています。ボタンが押されると、4 つのピンが接続され、回路が完成します。


Button



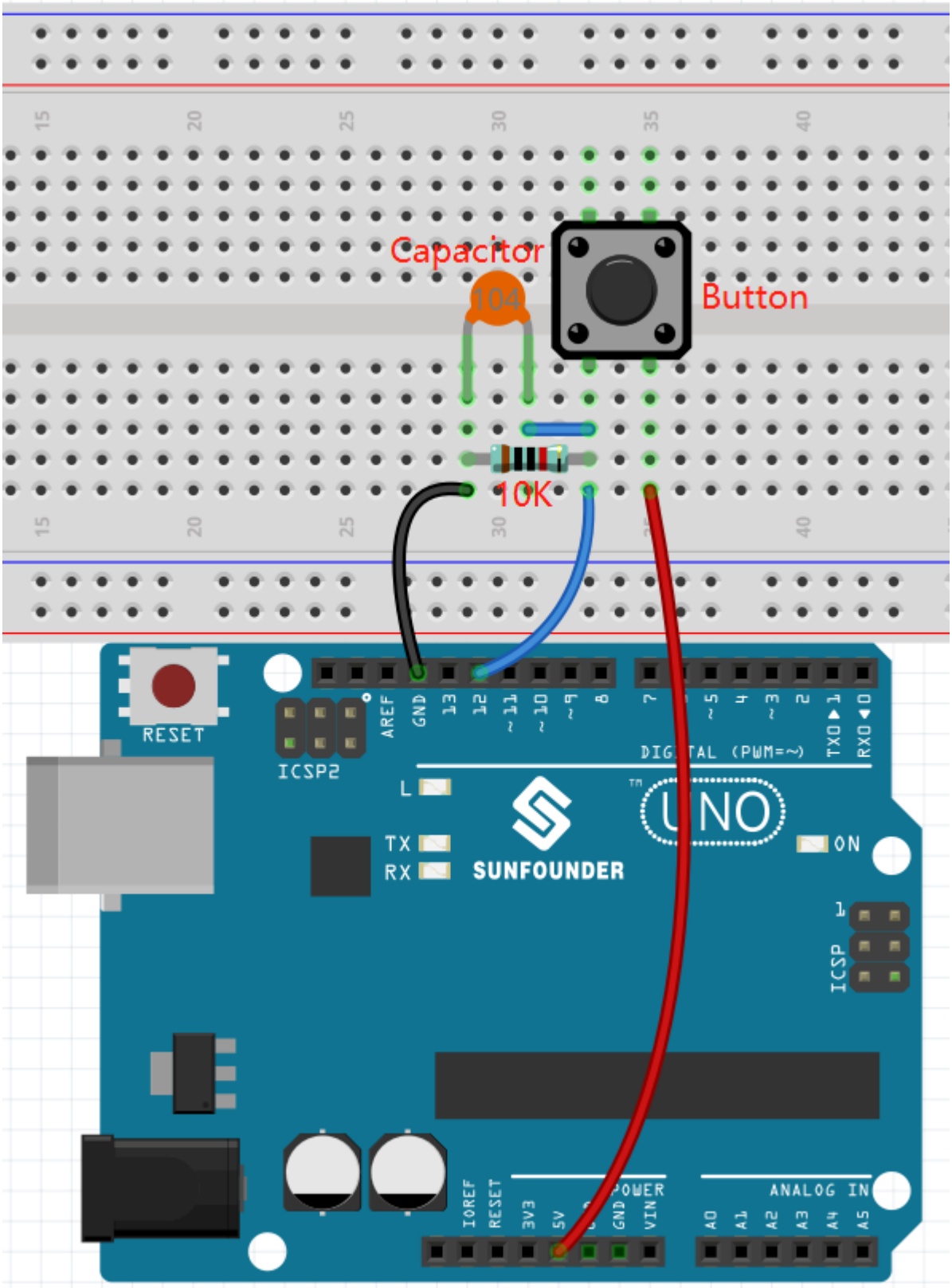

Internal Structure

下記の図に従い、回路を組み立ててください。

- ボタンの左側のピンの 1 つを、プルダウン抵抗と 0.1uF (104) のキャパシターに接続されているピン 12 に

接続します（ボタン動作時のジッタを除去し、安定した出力を得るため）。

- 抵抗とキャパシタのもう一方の端子を GND に接続し、ボタンの右側のピンの 1 つを 5V に接続します。


Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

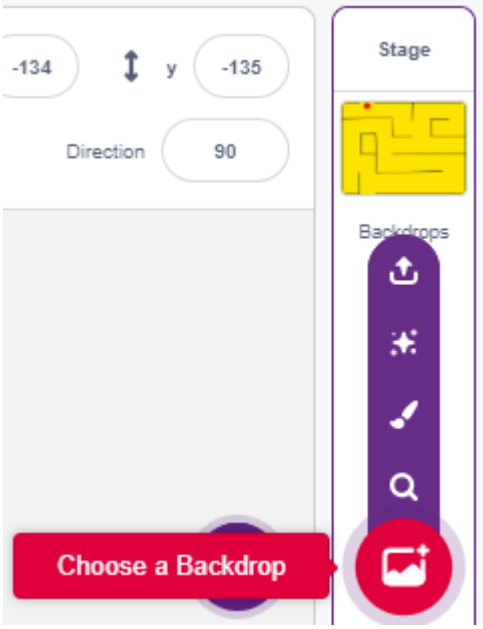
### 7.19.3 プログラミング

私たちが達成したい効果は、ボタンを使って **Beetle** スプライトの方向を制御し、 **Maze** バックドロップ上の黒い線に触れずに前進してりんごを食べ、食べるとバックドロップが切り替わることです。

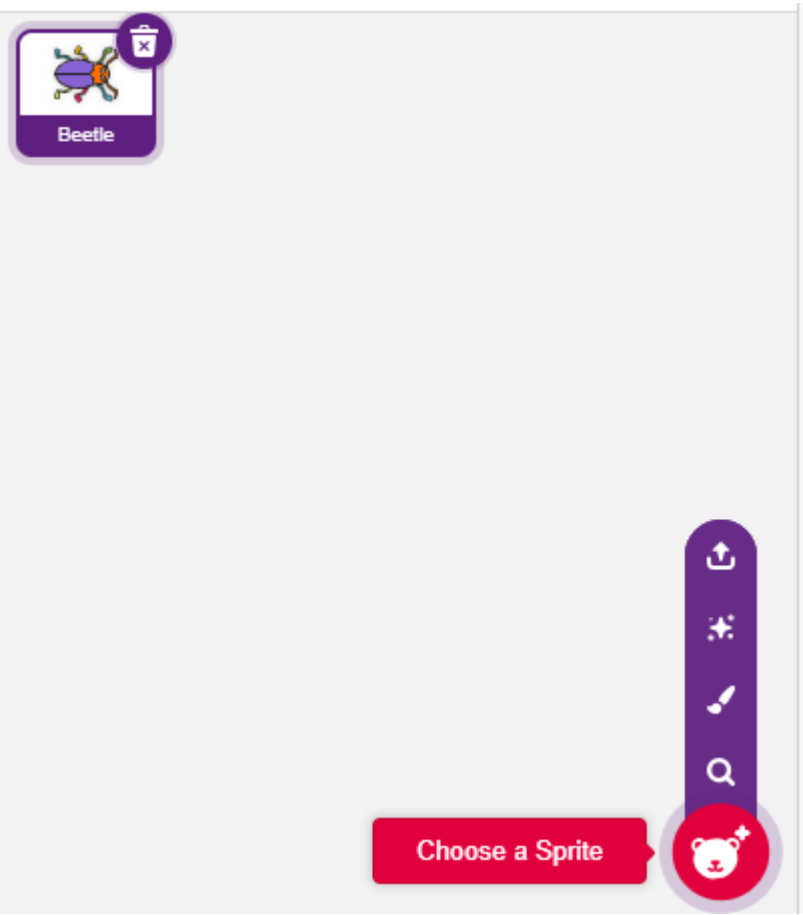
まず関連するバックドロップとスプライトを追加します。

**1. バックドロップとスプライトの追加**

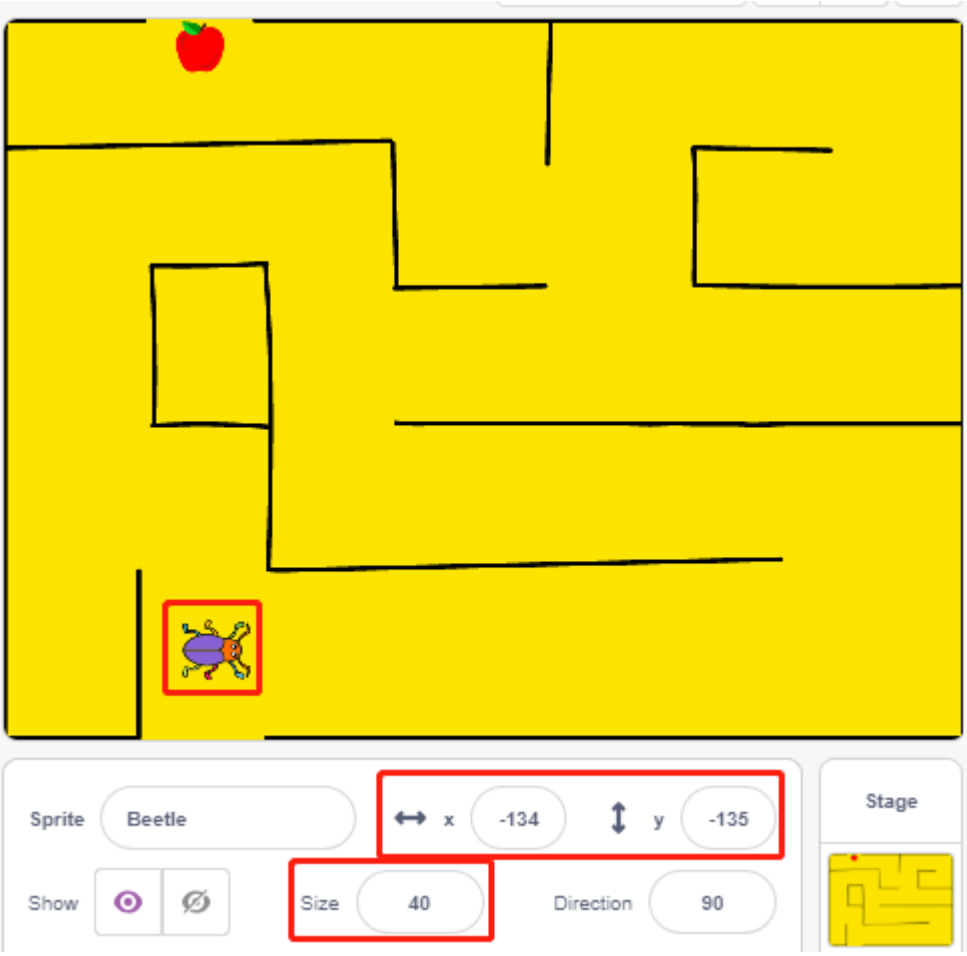
**Choose a backdrop** ボタンを使って **Maze** バックドロップを追加します。




デフォルトのスプライトを削除し、 **Beetle** スプライトを選択します。



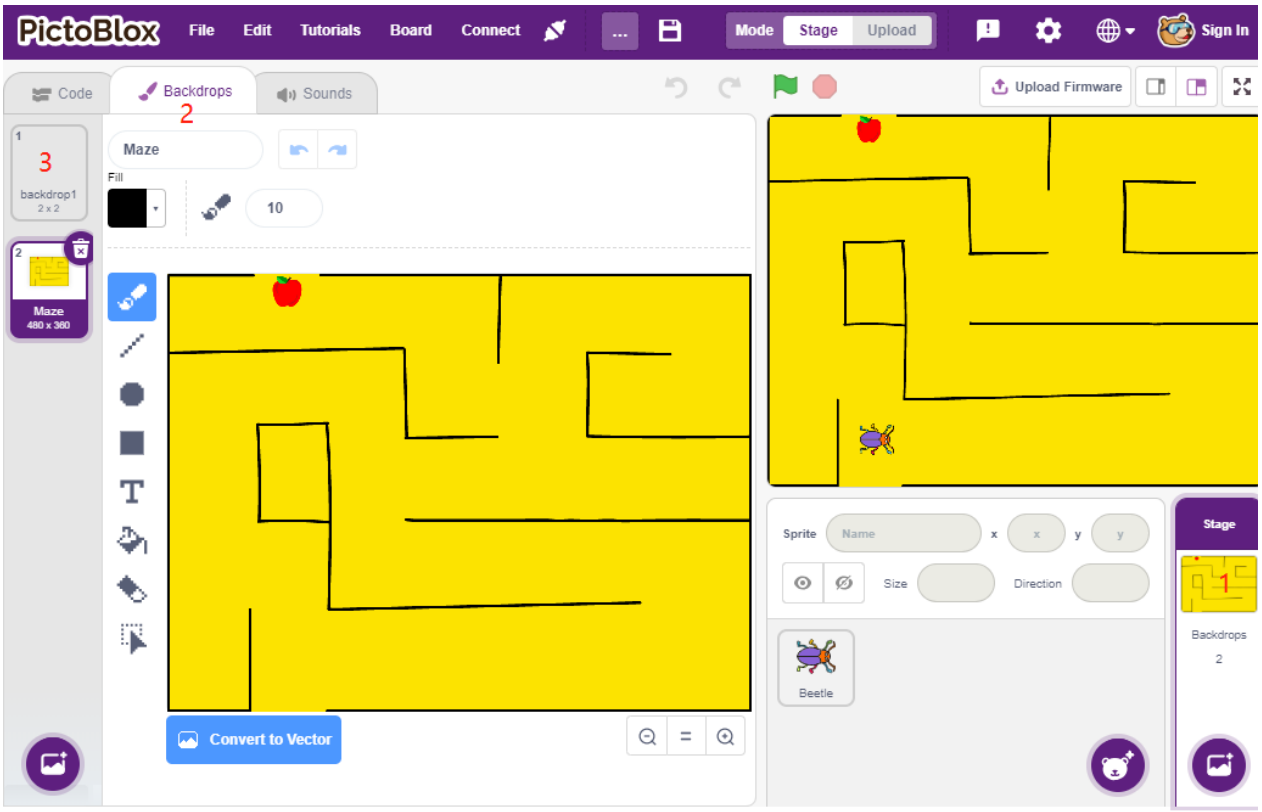

**Beetle** スプライトを **Maze** バックドロップの入口に配置し、この時点での x,y 座標の値を覚えておき、スプライトのサイズを 40 %に調整します。




**2. バックドロップの描画**

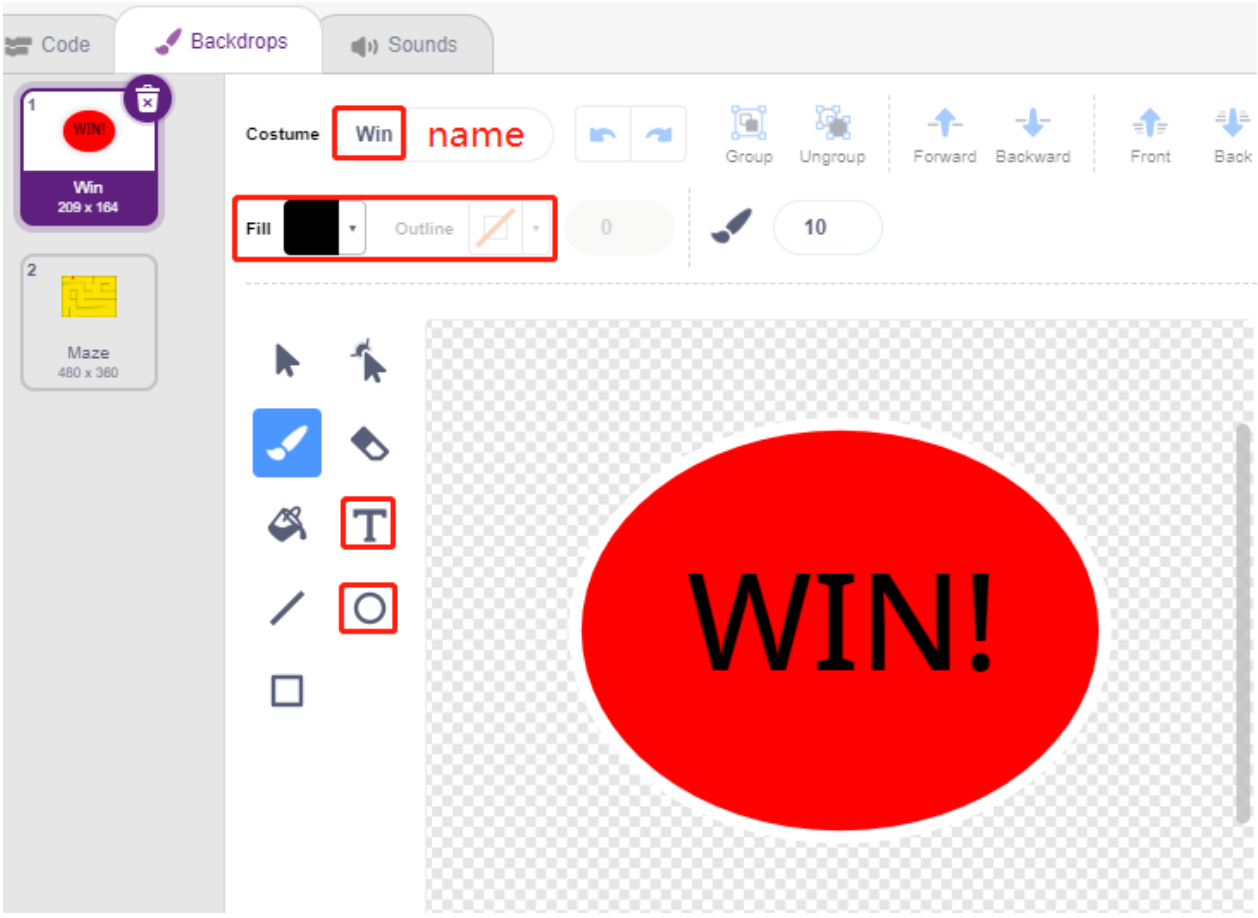
次に、WIN!キャラクターが表示されるバックドロップを簡単に描画します。

まずバックドロップのサムネイルをクリックして **Backdrops** ページに移動し、blank backdrop1 をクリックします。



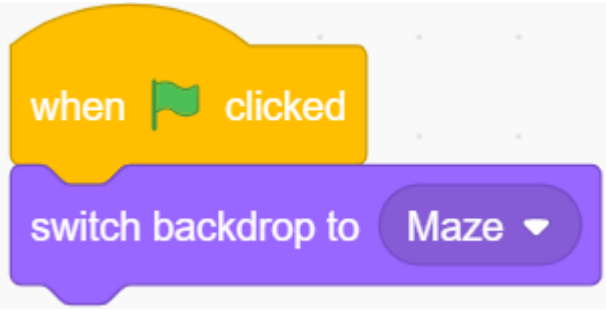

描画を開始します。下の画像を参考に描くことができますし、勝利の表情として独自のバックドロップを描くこともできます。

- **Circle** ツールを使用して、色を赤に設定し、アウトラインを持たない楕円を描きます。
- 次に **Text** ツールを使って、"WIN!"という文字を書き、文字の色を黒に設定し、文字のサイズと位置を調整します。
- バックドロップの名前を **Win** にします。




### 3. バックドロップのスクリプト作成

ゲームが開始するたびに、バックドロップを **Maze** に切り替える必要があります。



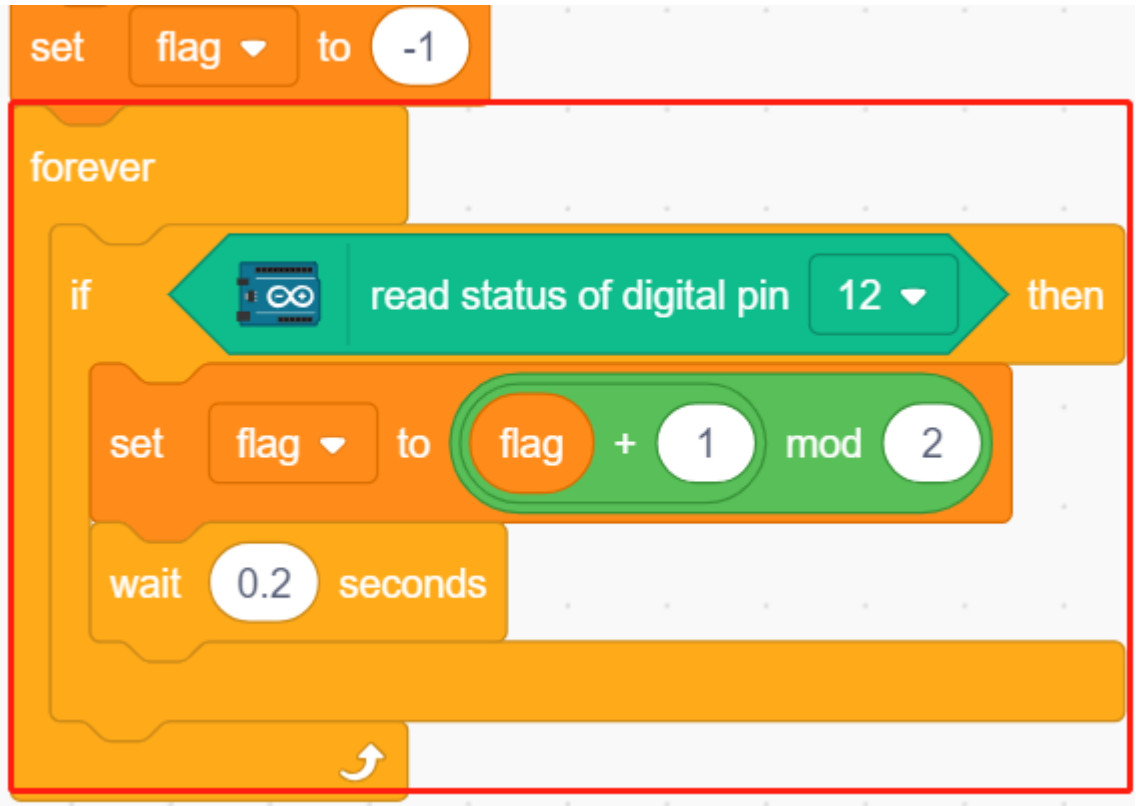

### 4. Beetle スプライトのスクリプトの記述

ボタンの制御の下で前進と方向転換ができるように、スプライト **Beetle** のスクリプトを書きます。ワークフローは以下の通りです。

- 緑のフラグがクリックされたとき、 **Beetle** の角度を 90 に設定し、位置を (-134, -134) に設定します。また、自分で配置した位置の座標値に置き換えることもできます。変数 **flag** を作成し、初期値を-1 に設定します。

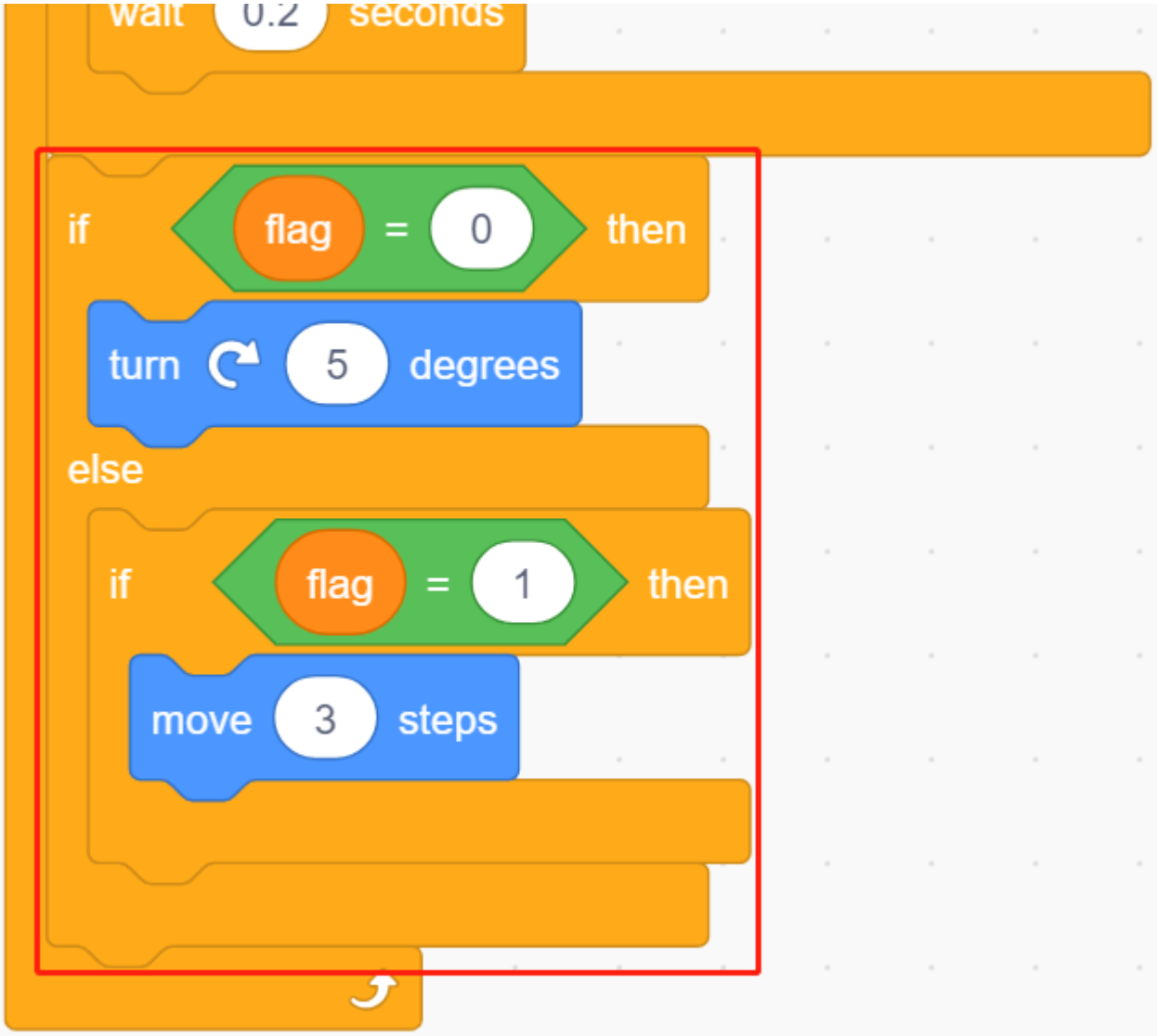

次に、[forever] ブロック内で、4 つの [if] ブロックを使用して様々な可能性のあるシナリオを判断します。

- キーが 1（pressed）の場合、[mod] ブロックを使用して変数 **flag** の値を 0 と 1 の間で切り替えます（このプレスのための 0、次のプレスのための 1 を交互にする）。




- flag=0（このキーの押下）の場合、 **Beetle** スプライトは時計回りに回転します。次に flag が 1（再びキーが押された）と等しいかどうかを判断し、 **Beetle** スプライトが前進するか、時計回りに回り続けるかを決定

します。

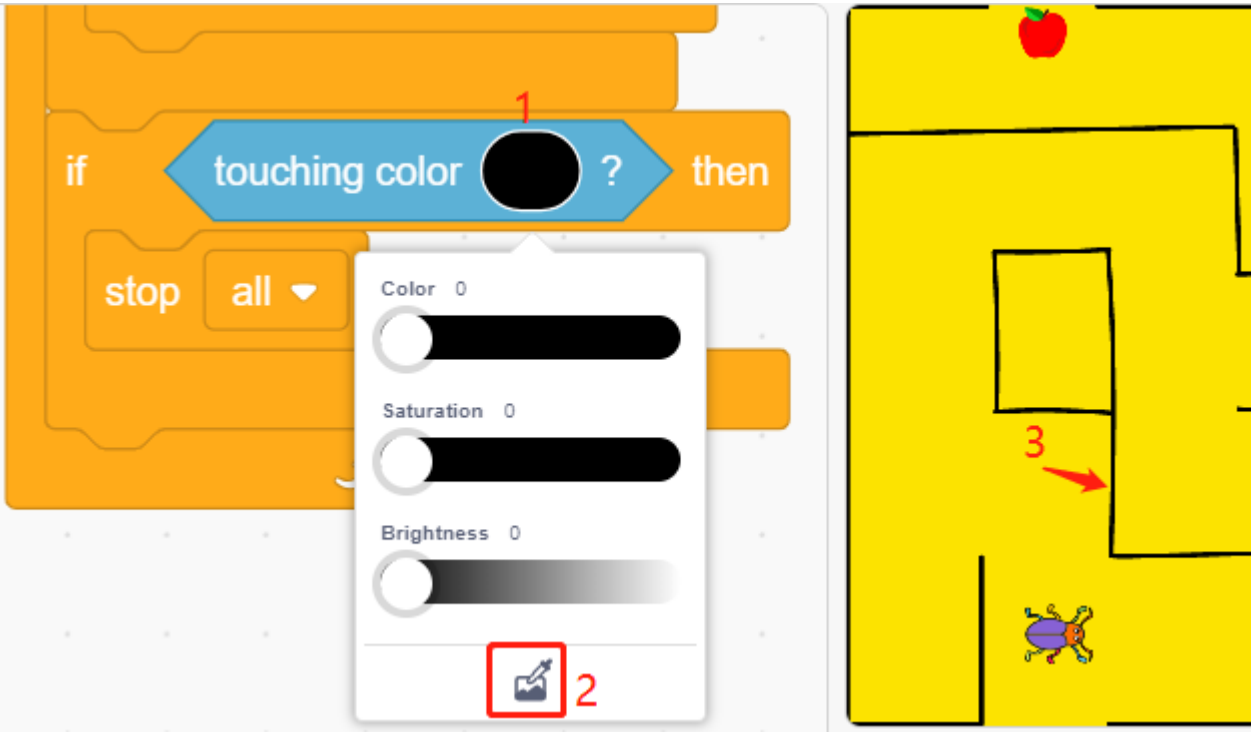



- Beetle スプライトが黒（**Maze** バックドロップ上の黒い線）に触れると、ゲームは終了し、スクリプトは実行を停止します。

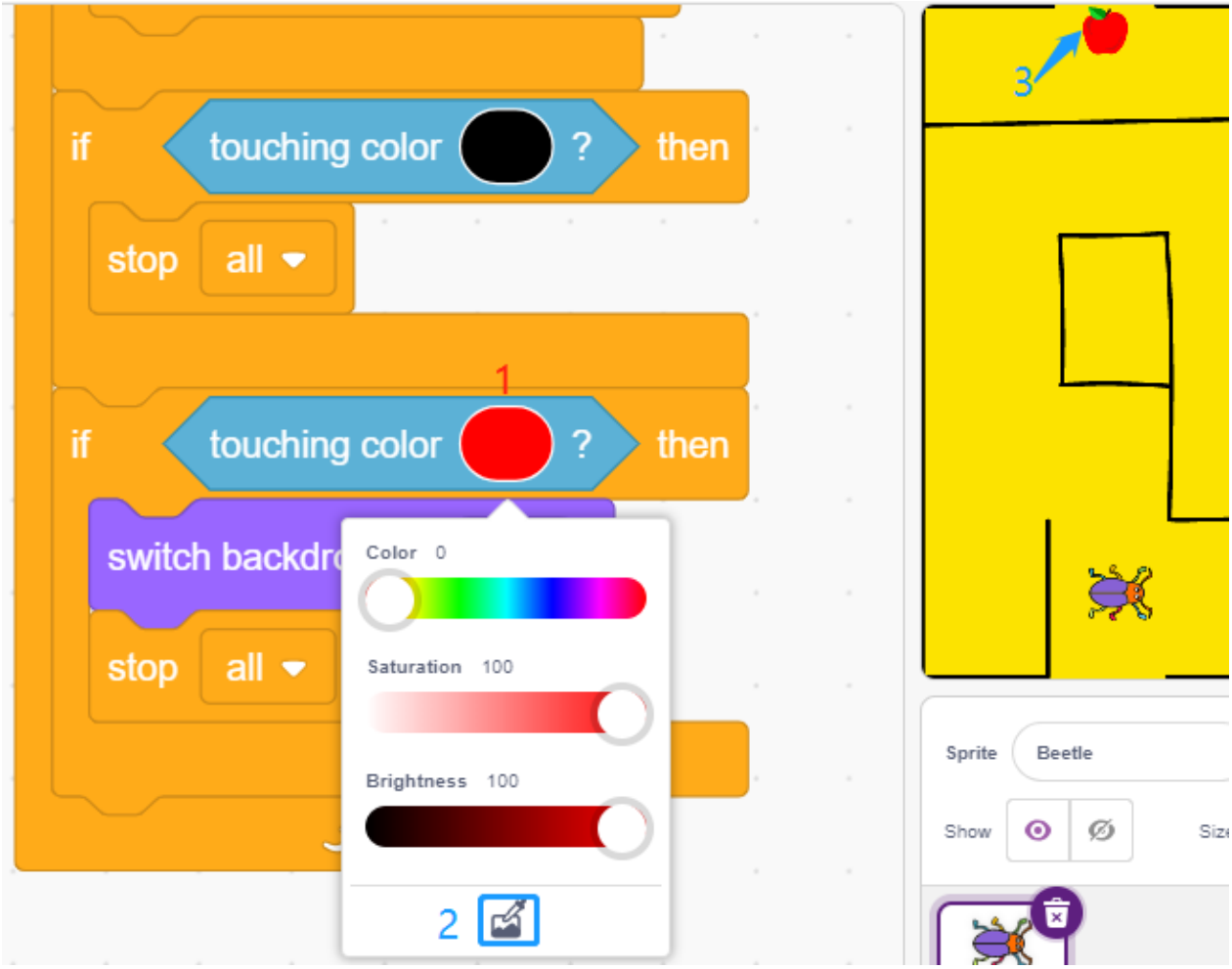
---

**注釈:** [Touch color] ブロック内の色領域をクリックし、アイドロッパーツールを選択してステージ上の黒い線の色を取得する必要があります。任意で黒を選択すると、この [Touch color] ブロックは動作しません。

---



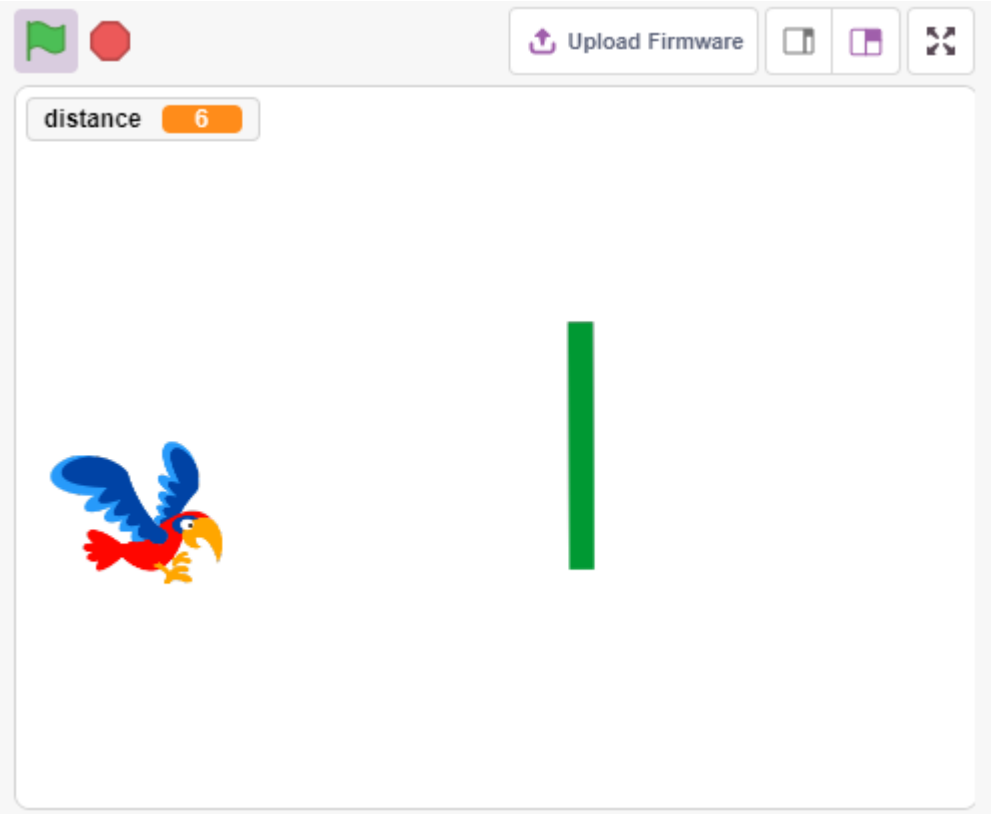

- Beetle が赤に触れると（アップルの赤い色を取得するためにストロツールも使用します)、バックドロップは **Win** に切り替わり、ゲームは成功し、スクリプトの実行が停止します。




## 7.20 2.17 ゲーム - フラッピーパロット

このゲームでは、超音波モジュールを利用して、フラッピーパロットというゲームをプレイします。

スクリプトが実行されると、緑の竹がランダムな高さで右から左へとゆっくりと移動します。超音波モジュールの上に手を置き、超音波モジュールと手の距離が 10cm 未満の場合、パロットは上昇します。それ以外の場合、下降します。緑の竹（パドル）にぶつからないように、手と超音波モジュールとの距離を調節する必要があります。もし接触した場合、ゲームオーバーです。




### 7.20.1 必要な部品

このプロジェクトには以下の部品が必要です。

全てのキットをまとめて購入するのが便利です。以下にリンクを記載します。

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

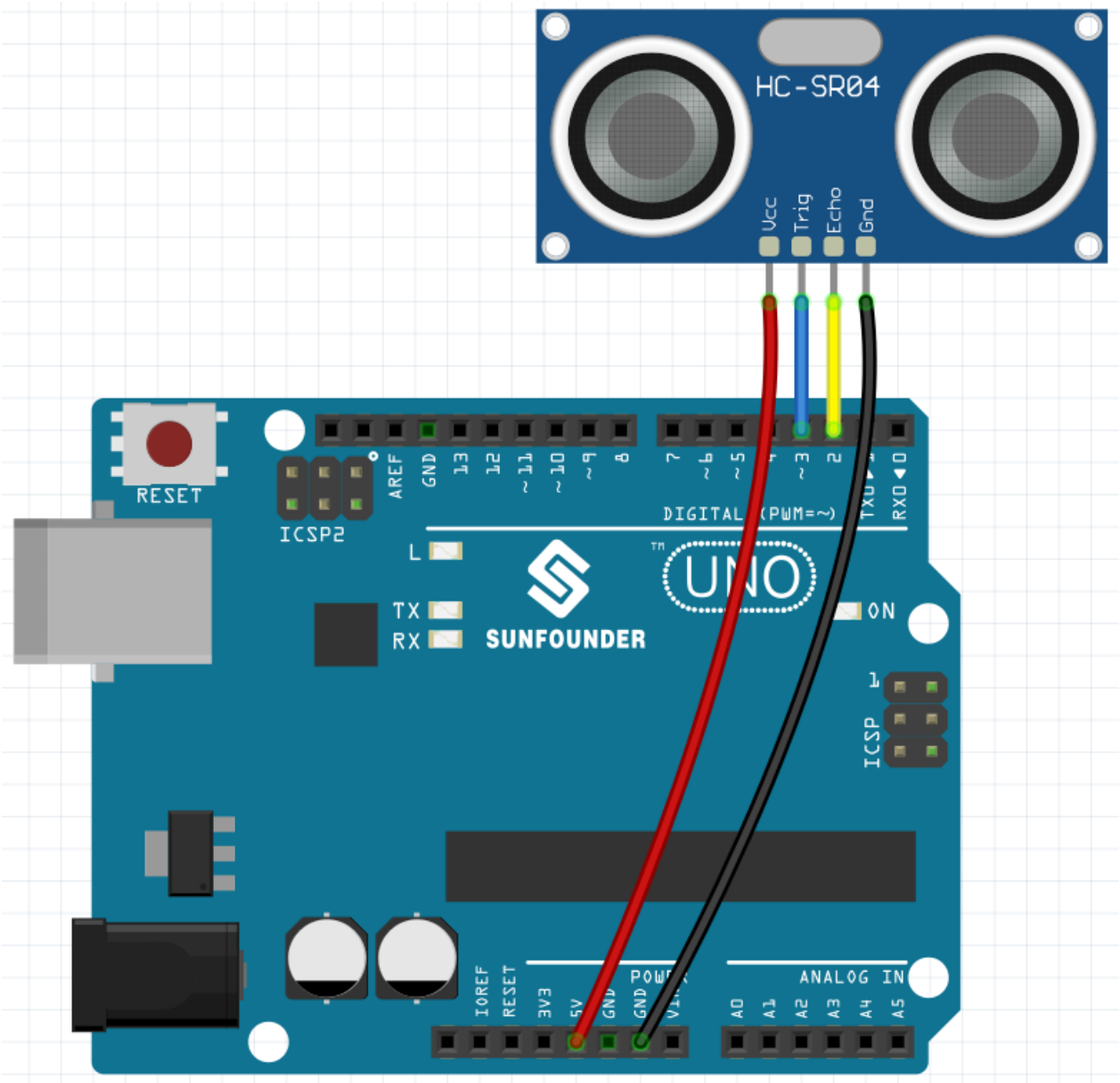
以下のリンクから、個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| 超音波モジュール | |

### 7.20.2 回路の作成

超音波センサーモジュールは、超音波を使って対象物までの距離を測定する機器である。2 つのプローブがある。1 つは超音波を送信するもので、もう 1 つは超音波を受信し、送受信時間を距離に変換することで、障害物との距離を検出する。

以下の図に従って回路を組み立ててください。

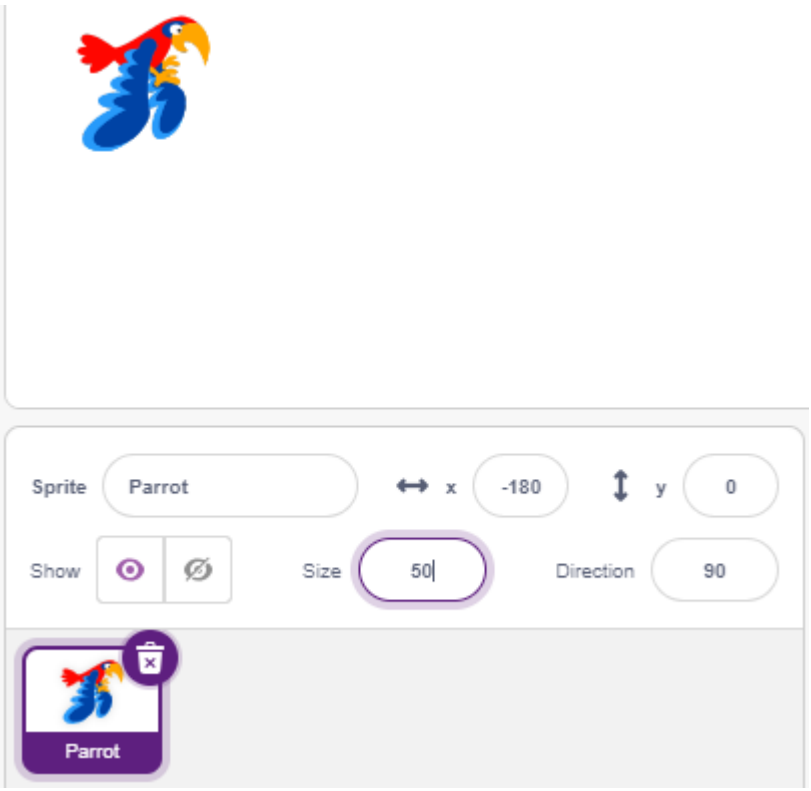


### 7.20.3 プログラミング

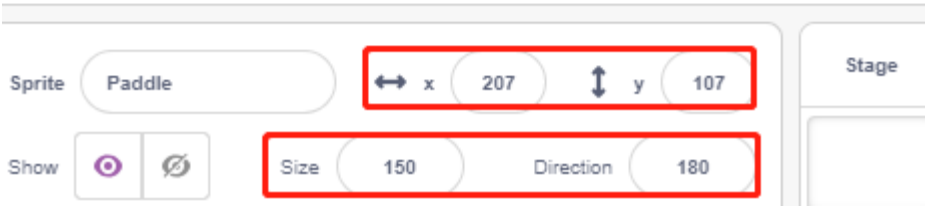
実現したい効果は、超音波モジュールを使って、スプライト **Parrot** の飛行高さをコントロールし、同時に **Paddle** スプライトを避けることです。

**1. スプライトの追加**

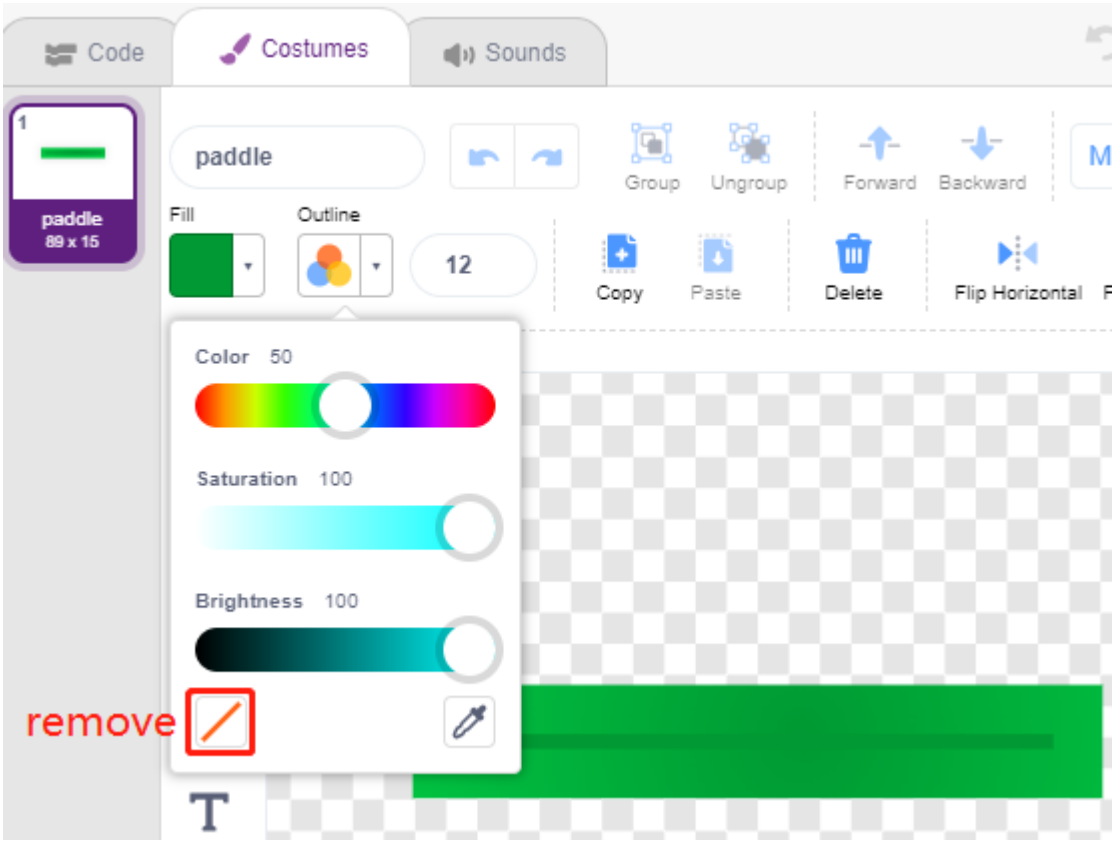
デフォルトのスプライトを削除し、 **Choose a Sprite** ボタンで **Parrot** スプライトを追加します。サイズを 50% に設定し、位置を左中央に移動します。




次に、 **Paddle** スプライトを追加し、サイズを 150% に設定、角度を 180 に設定し、初期位置を右上隅に移動します。



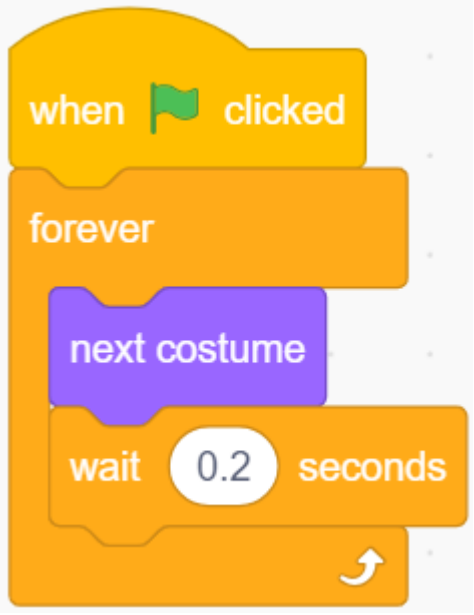

**Paddle** スプライトの **Costumes** ページに移動して、アウトラインを削除します。



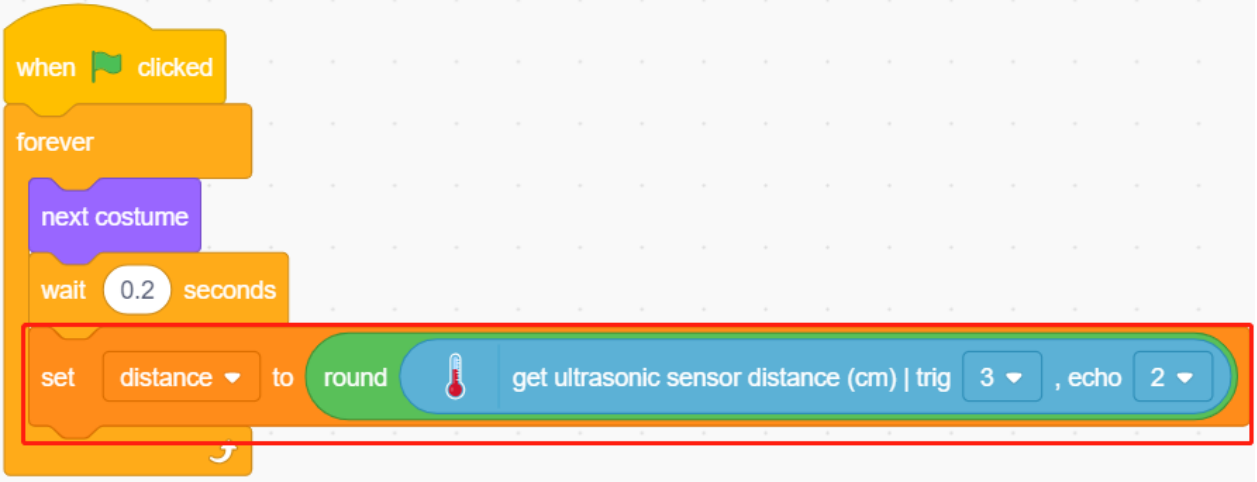

**2. Parrot スプライトのスクリプト作成**

次に、飛行中の **Parrot** スプライトのスクリプトを記述します。飛行高さは超音波モジュールの検出距離によって決まります。

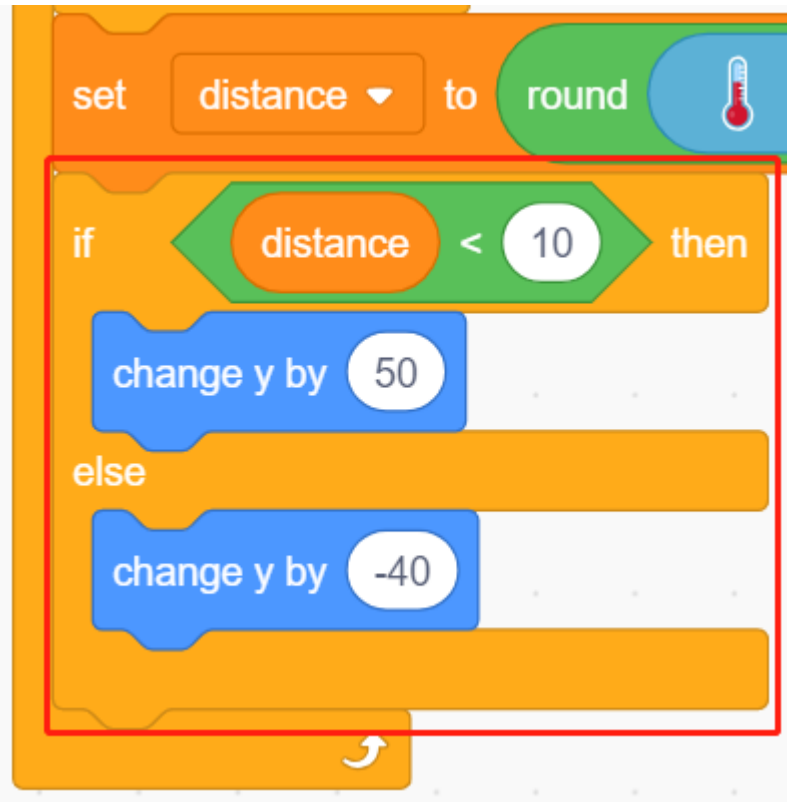
- 緑のフラグがクリックされたとき、0.2 秒ごとにコスチュームを切り替えて常に飛んでいるように見せます。



- 超音波モジュールの値を読み取り、[round] ブロックを使って四捨五入した後、変数 **distance** に格納します。




- 超音波の検出距離が 10cm 未満の場合、y 座標を 50 増やし、 **Parrot** スプライトは上昇します。それ以外の場合、y 座標の値は 40 減少し、 **Parrot** は下降します。

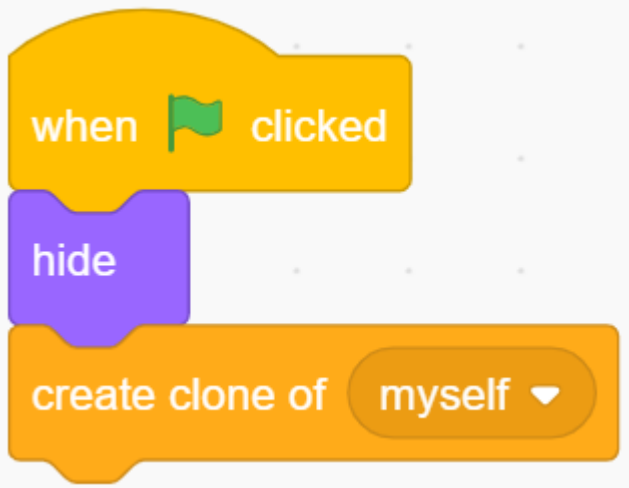



- **Parrot** スプライトが **Paddle** スプライトに触れると、ゲームは終了し、スクリプトの実行が停止します。


**3. Paddle スプライトのスクリプト作成**

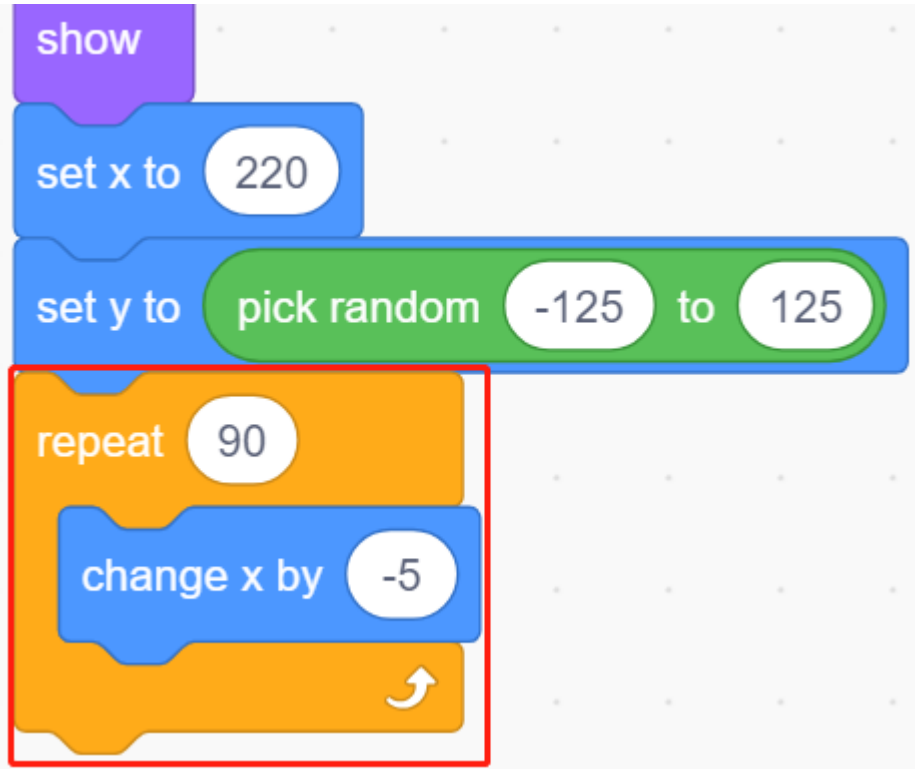
次に、 **Paddle** スプライトのスクリプトを書きます。これはステージ上でランダムに表示する必要があります。

- 緑のフラグがクリックされたとき、スプライト **Paddle** を非表示にし、同時に自分自身のクローンを作成します。[create clone of] ブロックはコントロールブロックおよびスタックブロックです。引数のスプライトのクローンを作成します。

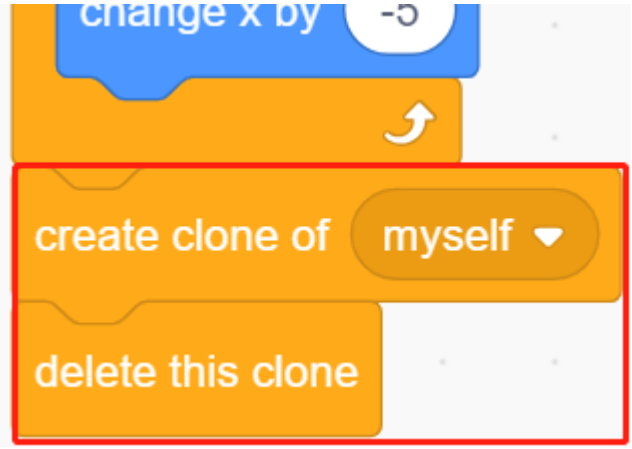



- **Paddle** がクローンとして表示されるとき、x 座標は 220（一番右）で、y 座標はランダムに (-125 から 125)（高さランダム）になります。


- [repeat] ブロックを使用して、x 座標の値をゆっくりと減少させることで、 **Paddle** スプライトのクローンが右から左にゆっくりと移動するのを見ることができます。



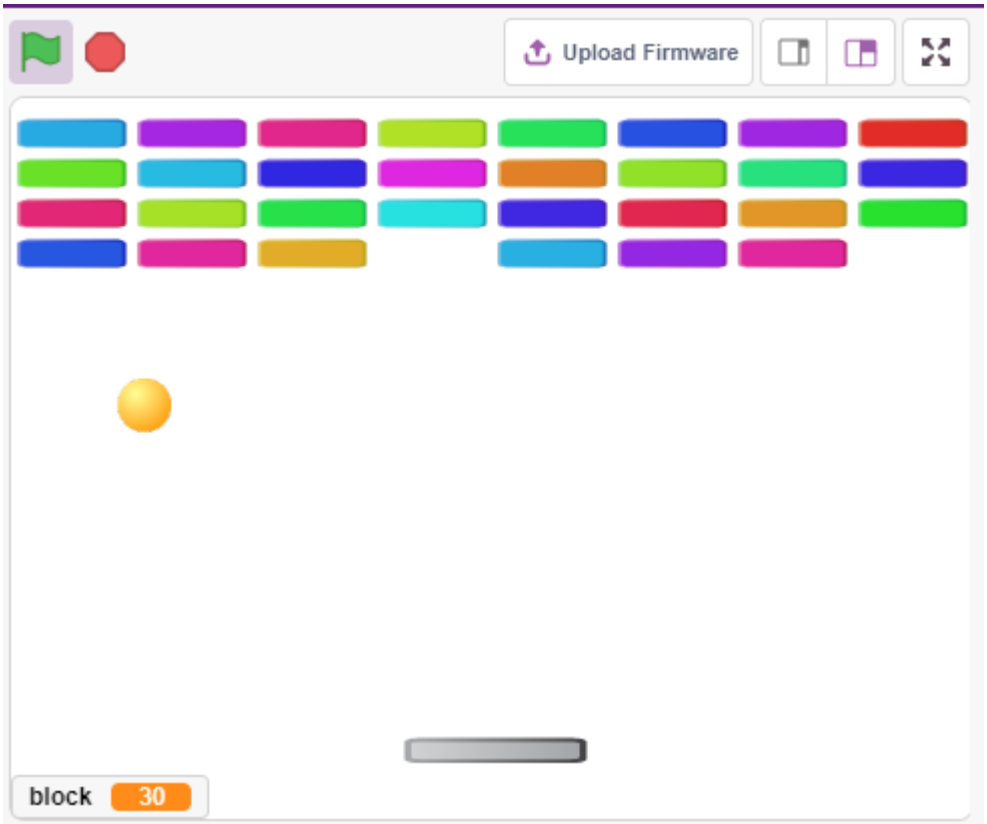

- 新しい **Paddle** スプライトのクローンを再度作成し、前のクローンを削除します。




## 7.21 2.18 ゲーム - ブレイクアウトクローン

このゲームでは、ポテンショメータを利用してブレイクアウトクローンゲームをプレイします。

緑のフラグをクリックした後、ポテンショメータでステージ上のパドルを操作してボールをキャッチします。ボールを上に移動させ、ブロックをヒットさせると、すべてのブロックが消え、ゲームに勝利します。ボールをキャッチしないと、ゲームに敗北します。



### 7.21.1 必要な部品

このプロジェクトで必要となる部品は以下のとおりです。

一括でキットを購入することがおすすめです。以下のリンクを参照してください：

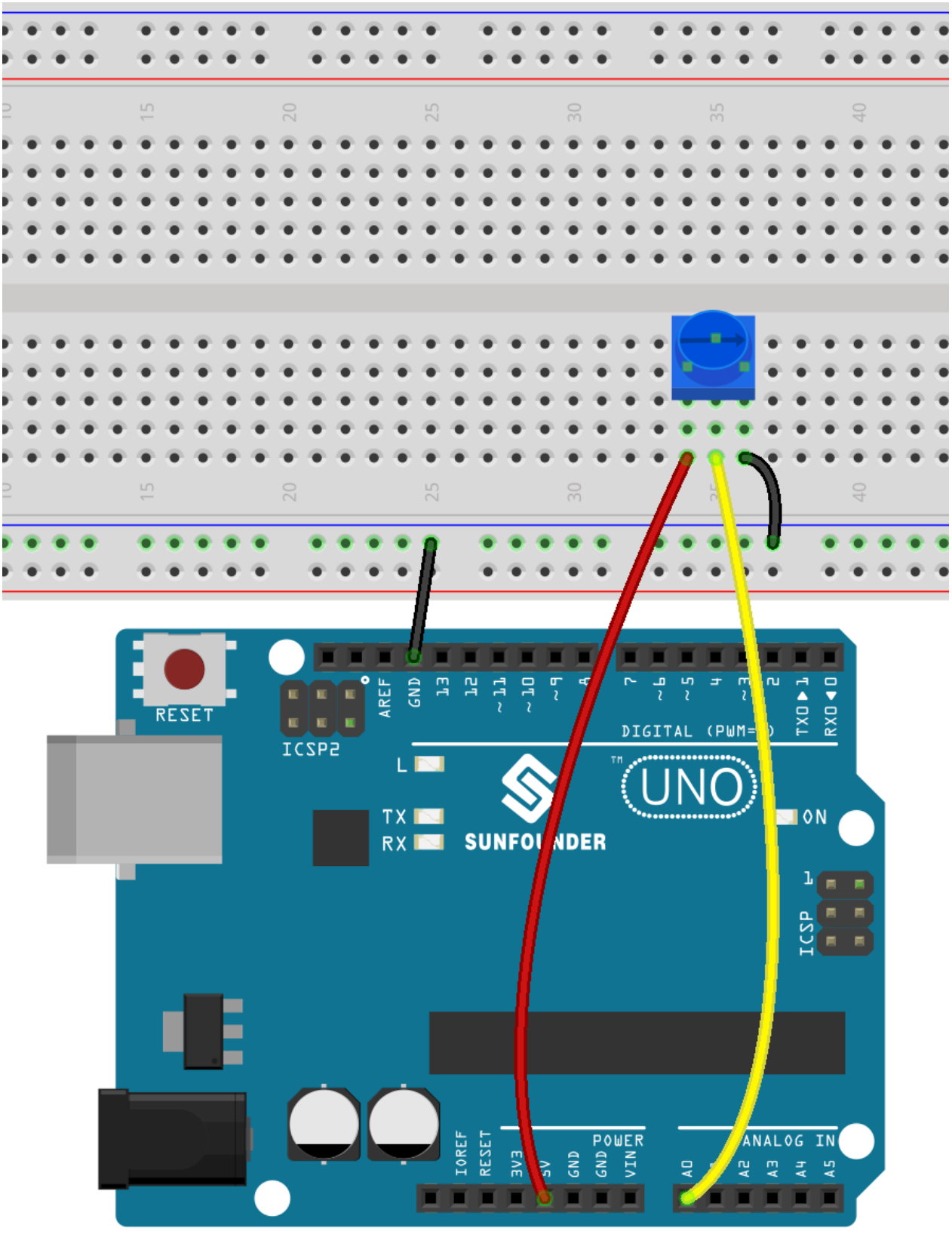
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから部品を個別に購入することも可能です。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| ポテンショメータ | |

### 7.21.2 回路の作成

ポテンショメータは 3 端子の抵抗素子です。2 つのサイドピンは 5V と GND に接続し、中央のピンは A0 に接続します。Arduino ボードの ADC コンバータを使用して変換した後、値の範囲は 0-1023 となります。


RESET
ICSP2
AREF
GND
13
12
DIGITAL (PWM=~)
TXO▲1
RXO◀0
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A2
A3
A4
A5

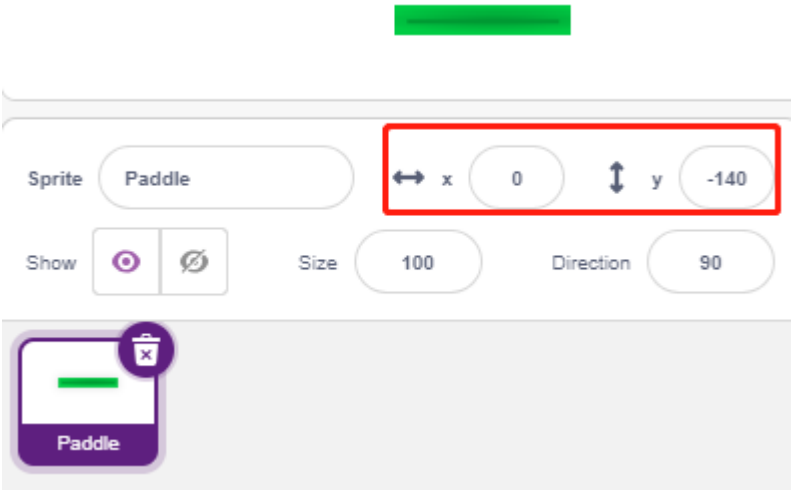
### 7.21.3 プログラミング

ステージには 3 つのスプライトが存在します。

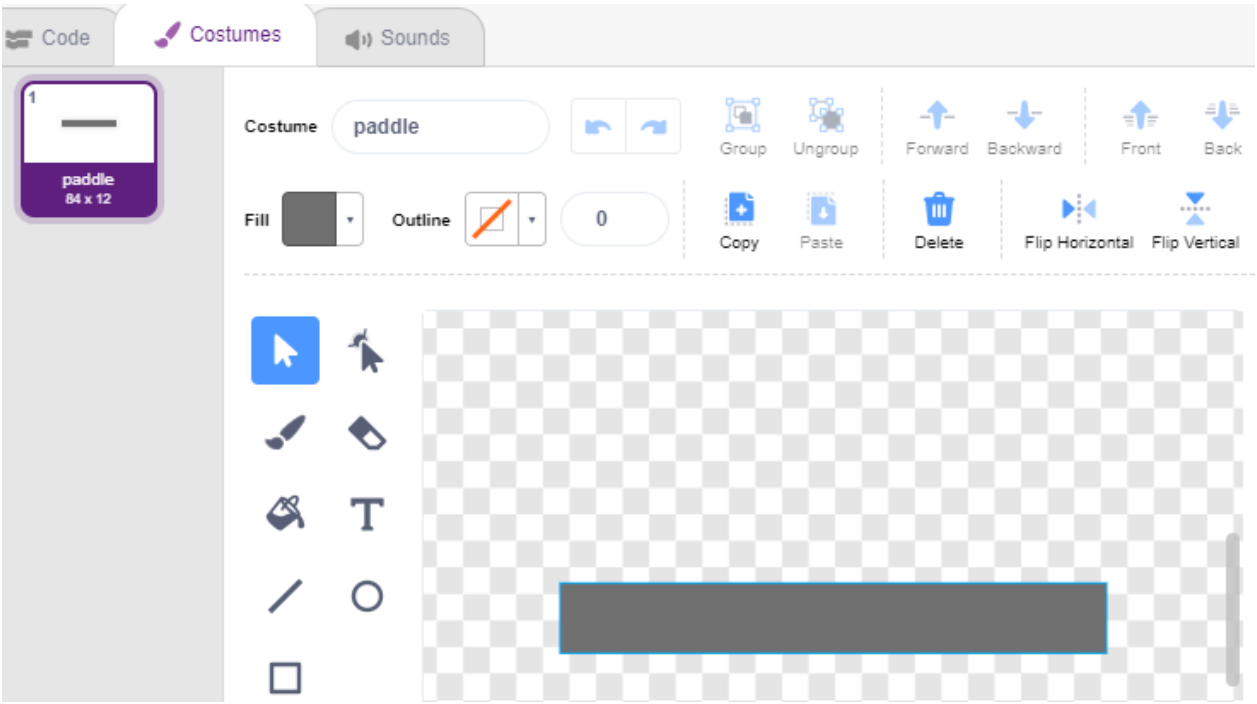
**1. パドルスプライト**

**Paddle** の目的は、ステージの底の中央に初期位置があり、ポテンショメータで左右に動かすことです。

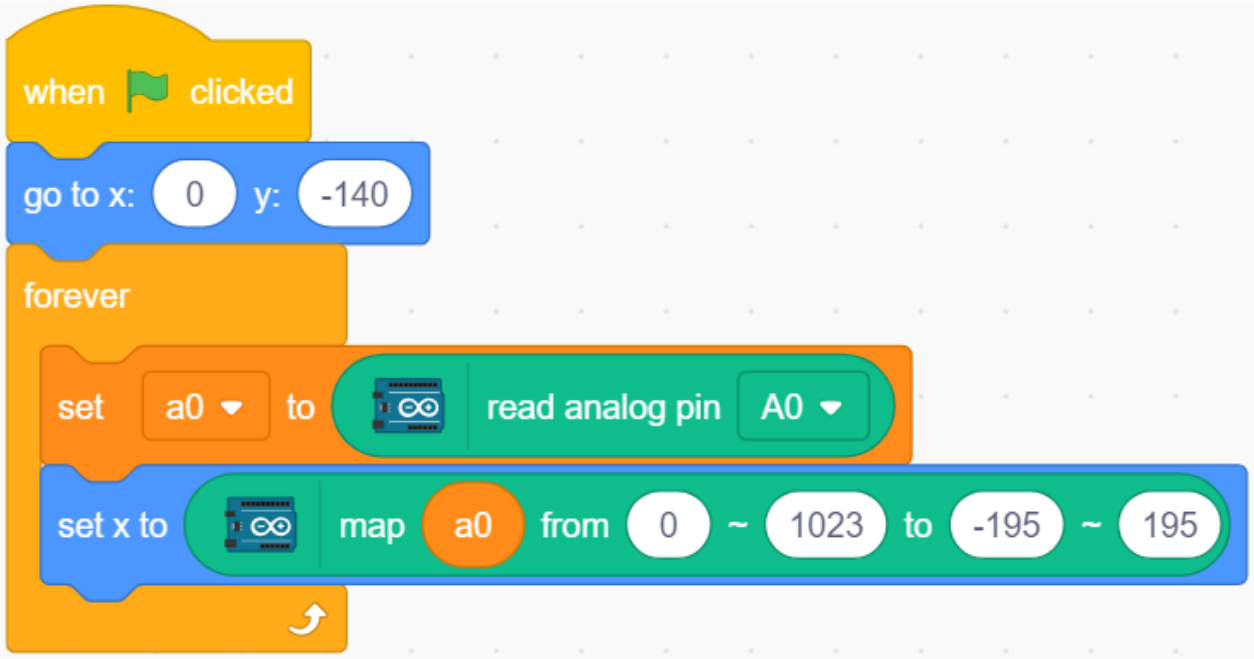
- デフォルトのスプライトを削除し、 **Choose a Sprite** ボタンで **Paddle** スプライトを追加して、x と y の位置を (0, -140) に設定します。




- **Costumes** ページに移動して、アウトラインを削除し、色を濃い灰色に変更します。



- 緑のフラグがクリックされたとき、 **Paddle** スプライトの初期位置を (0, -140) に設定し、A0 の値（ポテンショメータ）を変数 **a0** に読み取ります。 **Paddle** スプライトは x 座標 -195~195 で左右に移動するため、[map] ブロックを使用して、変数 **a0** の範囲 0~1023 を-195~195 にマッピングします。

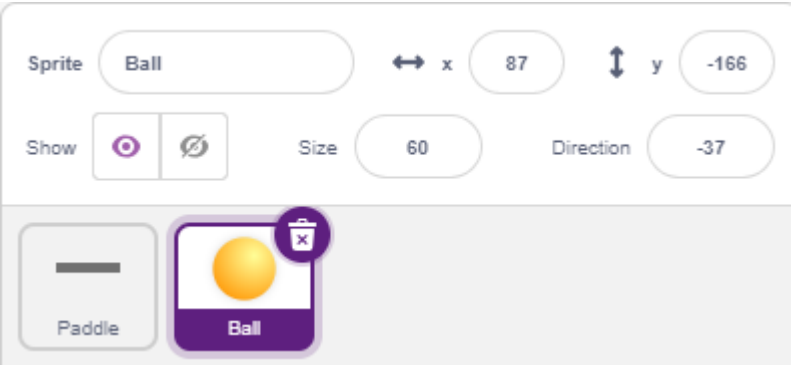



- ポテンショメータを回して、ステージ上で **Paddle** が左右に動くかを確認します。

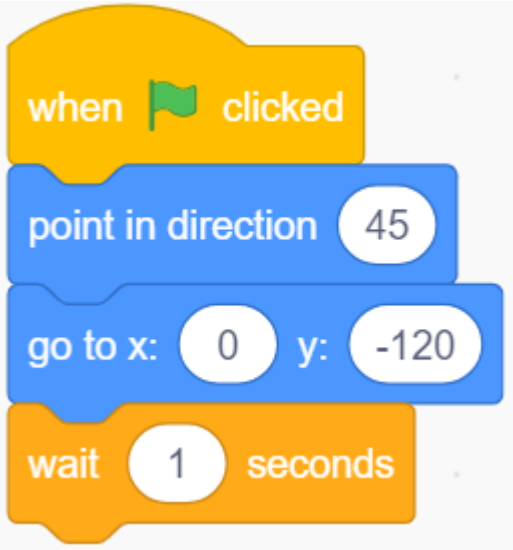
### 2. ボールスプライト

ボールスプライトの動作は、ステージを移動し、端に触れると跳ね返ります。ステージの上のブロックに触れると下に跳ね返り、落下中にパドルスプライトに触れると上に跳ね返ります。そうでない場合は、スクリプトの実行を停止し、ゲームが終了します。

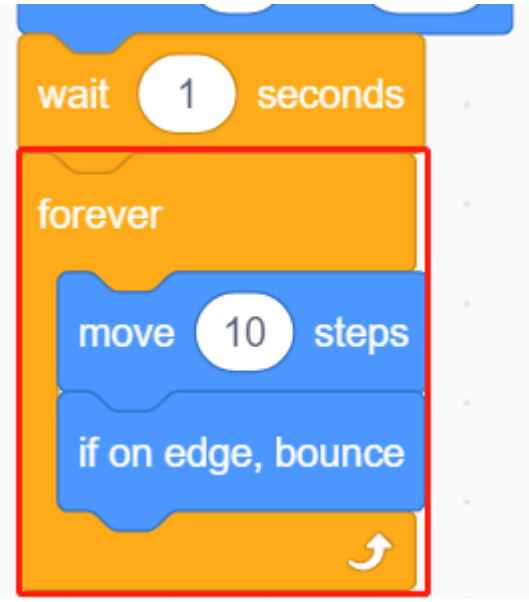
- **Ball** スプライトを追加します。



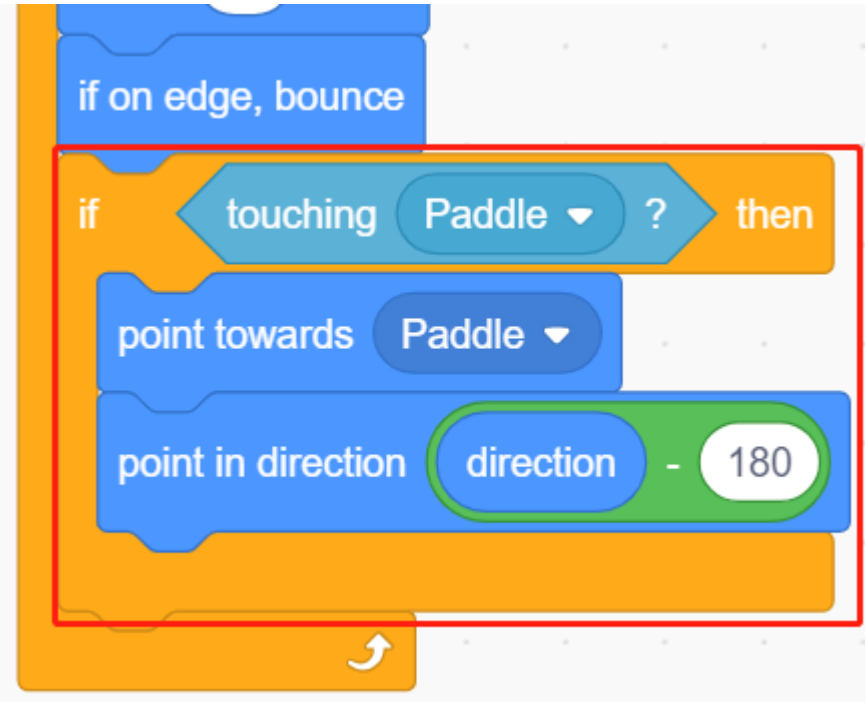

- 緑のフラグをクリックすると、 **Ball** スプライトの角度を 45 ° に設定し、初期位置を (0, -120) に設定します。

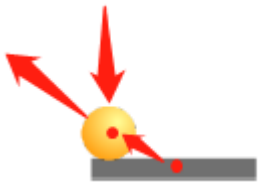



- さて、 **Ball** スプライトをステージの周りに移動させ、端に触れると跳ね返るようにします。緑のフラグをクリックして効果を確認してください。




- **Ball** スプライトが **Paddle** スプライトに触れた場合、反射を行います。これを簡単に実行する方法は、角度を直接反転させることですが、それを行うとボールの軌道が完全に固定されてしまい、非常に退屈になります。したがって、2 つのスプライトの中心を使用して計算し、バッフルの中心の反対方向にボールを跳ね返すようにします。



- **Ball** スプライトがステージの端に落ちると、スクリプトの実行が停止し、ゲームが終了します。


**3. Block1 スプライト**

**Block1** スプライトは、ステージの上で自身の 4x8 のクローンをランダムな色で表示し、 **Ball** スプライトに触れられるとクローンを削除する効果があります。

**Block1** スプライトは **PictoBlox** ライブラリには含まれていないので、自分で描くか、既存のスプライトを修正する必要があります。ここでは **Button3** スプライトを使って修正します。

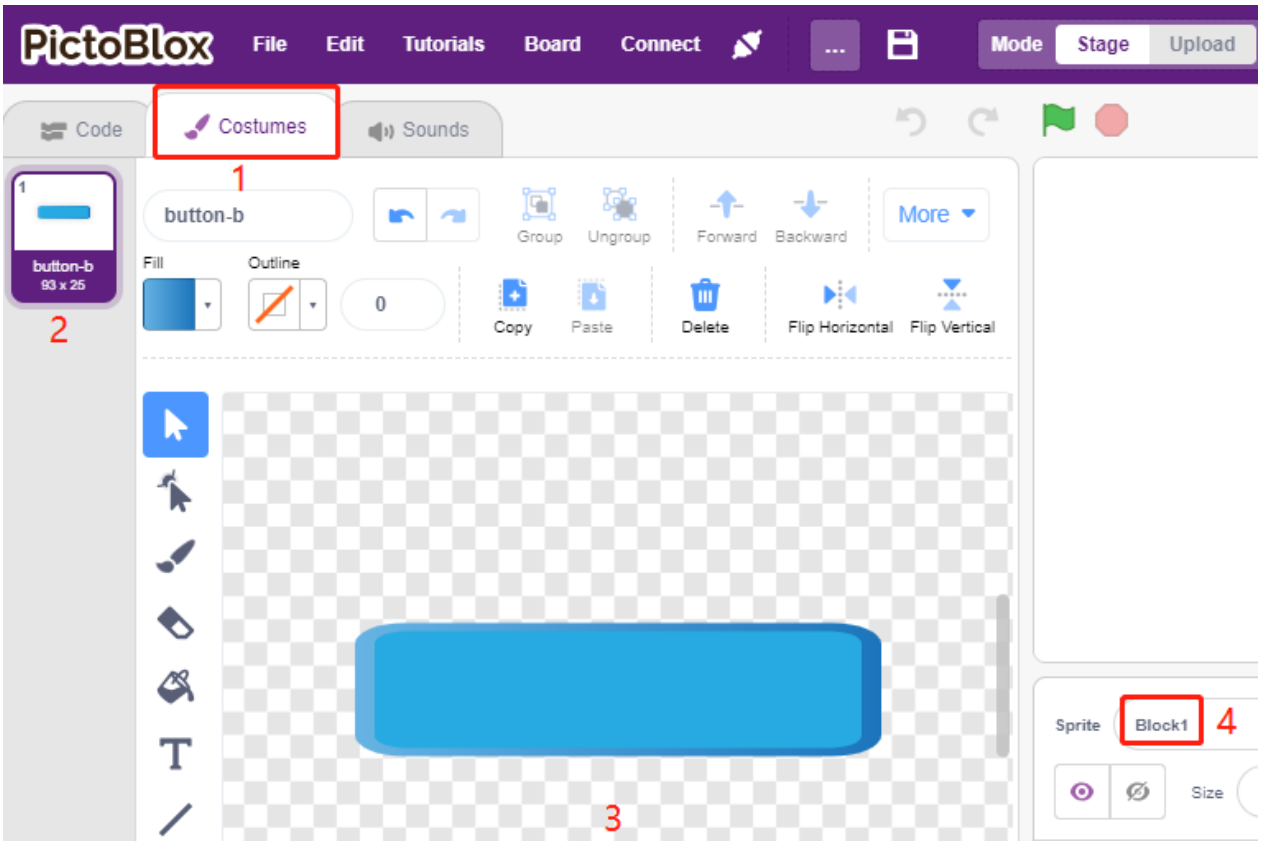
- **Button3** スプライトを追加した後、 **Costumes** ページに移動します。まず **button-a** を削除し、 **button-b** の幅と高さを縮小し、スプライトの名前を **Block1** に変更します。

---

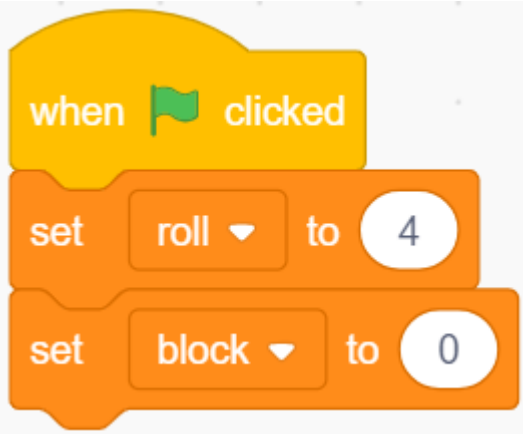
**注釈:**

- **Block1** の幅については、画面上で 8 つ並べることができるかどうかをシミュレートして確認してください。できない場合は、幅を適切に縮小してください。
- **Block1** スプライトを縮小する過程で、中心点をスプライトの中央に保持する必要があります。

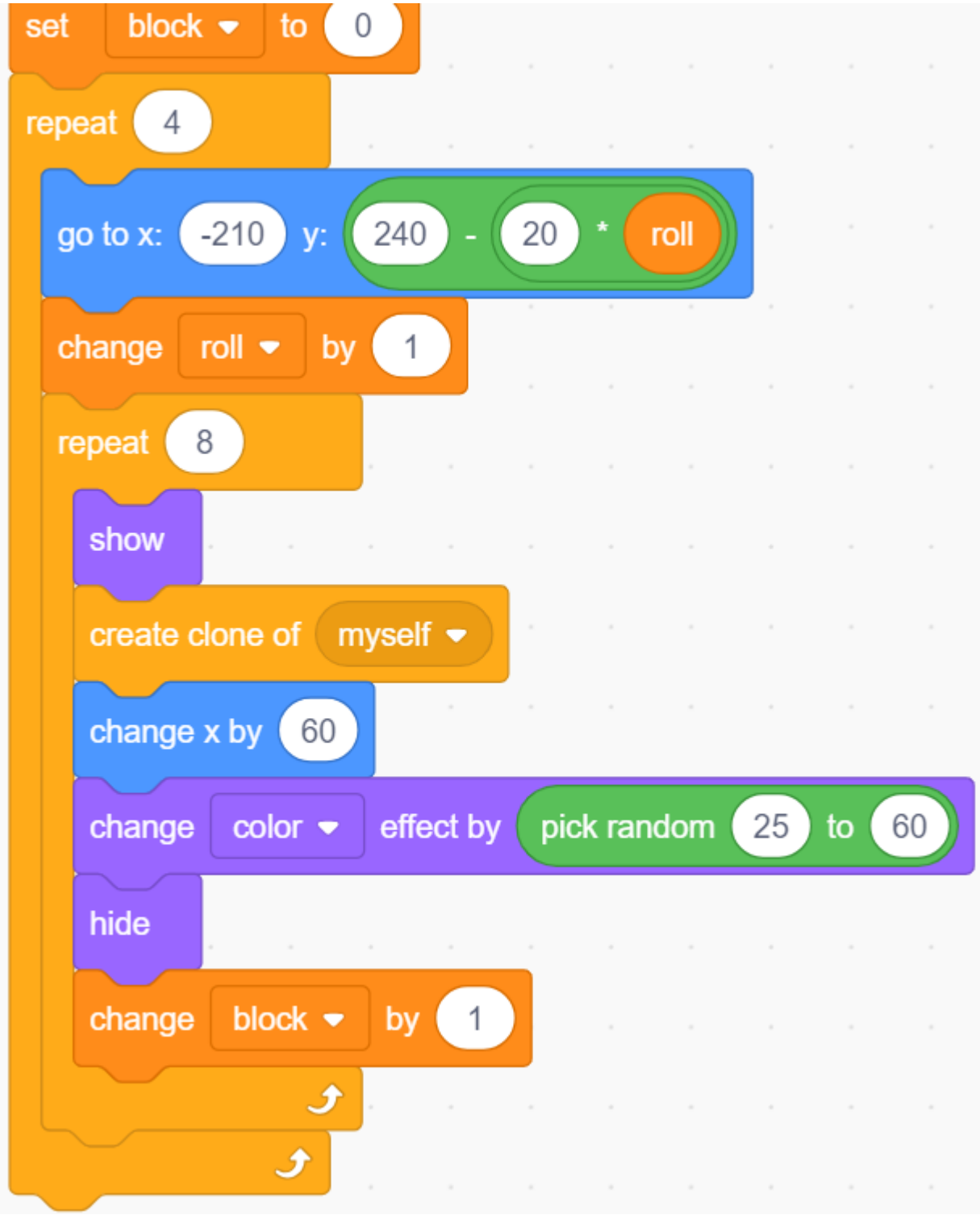
---



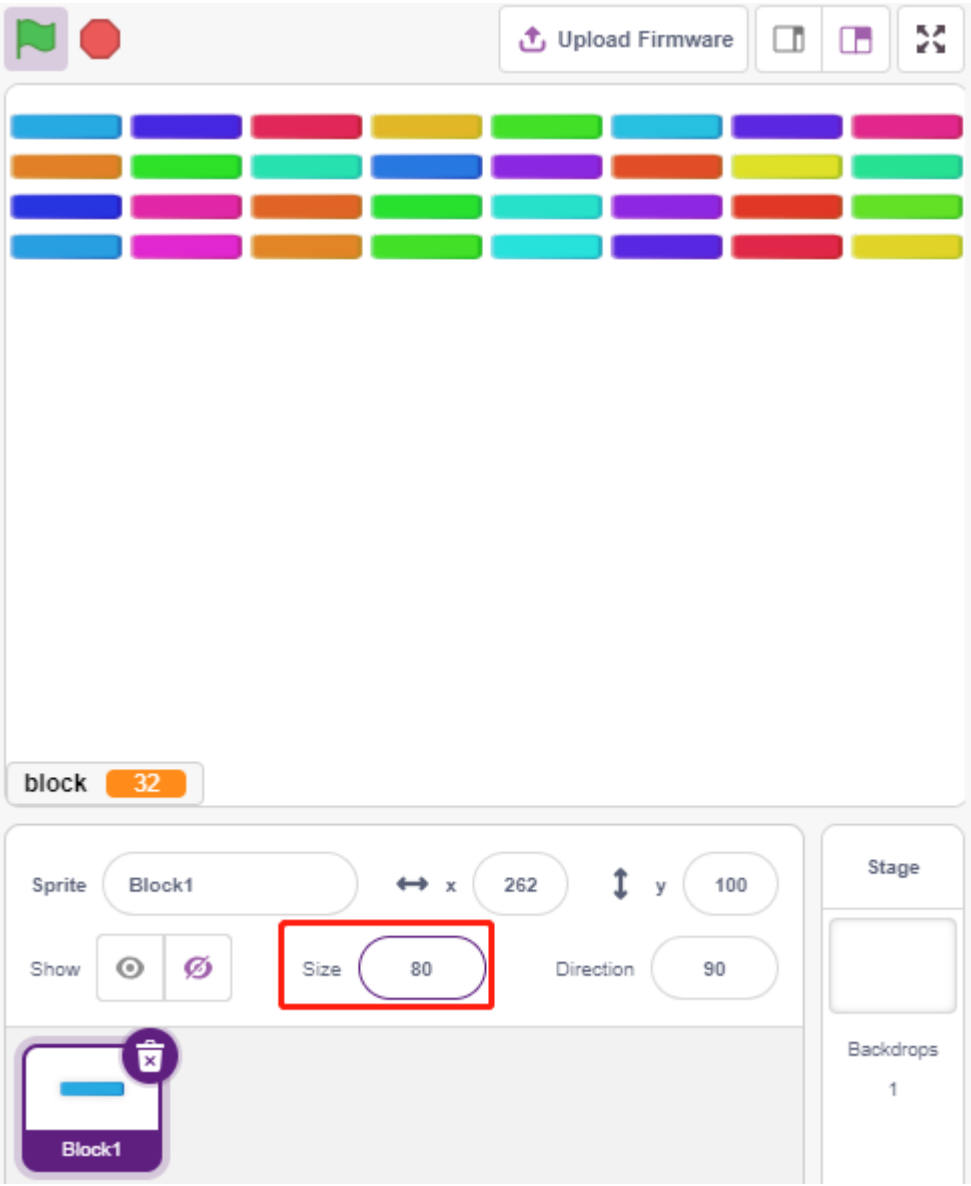

- まず 2 つの変数を作成します。 **block** はブロックの数を、 **roll** は行数を保存します。




- **Block1** スプライトのクローンを作成し、左から右、上から下に 1 つずつ、合計 4x8 でランダムな色で表示する必要があります。

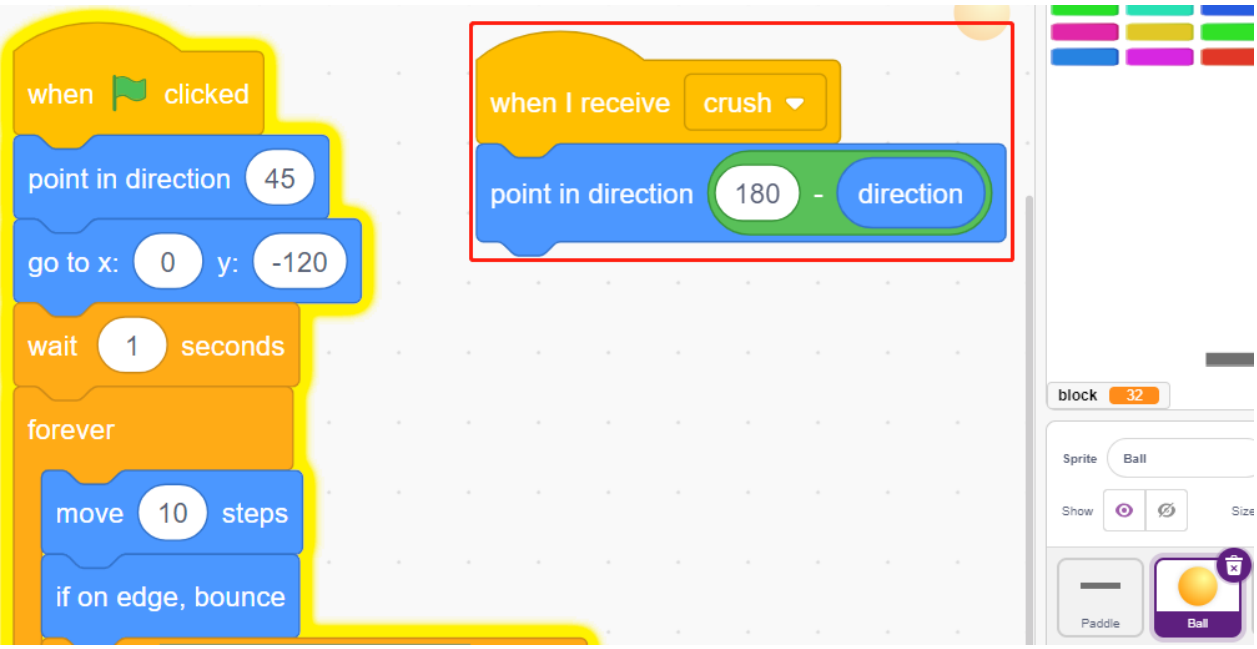



- スクリプトを書き終えたら、緑のフラグをクリックしてステージ上の表示を確認します。もし、表示がコンパクトすぎるか小さすぎる場合は、サイズを変更できます。




- トリガーイベントを書きます。クローン化された **Block1** スプライトが **Ball** スプライトに触れると、クローンを削除し、 **crush** というメッセージをブロードキャストします。

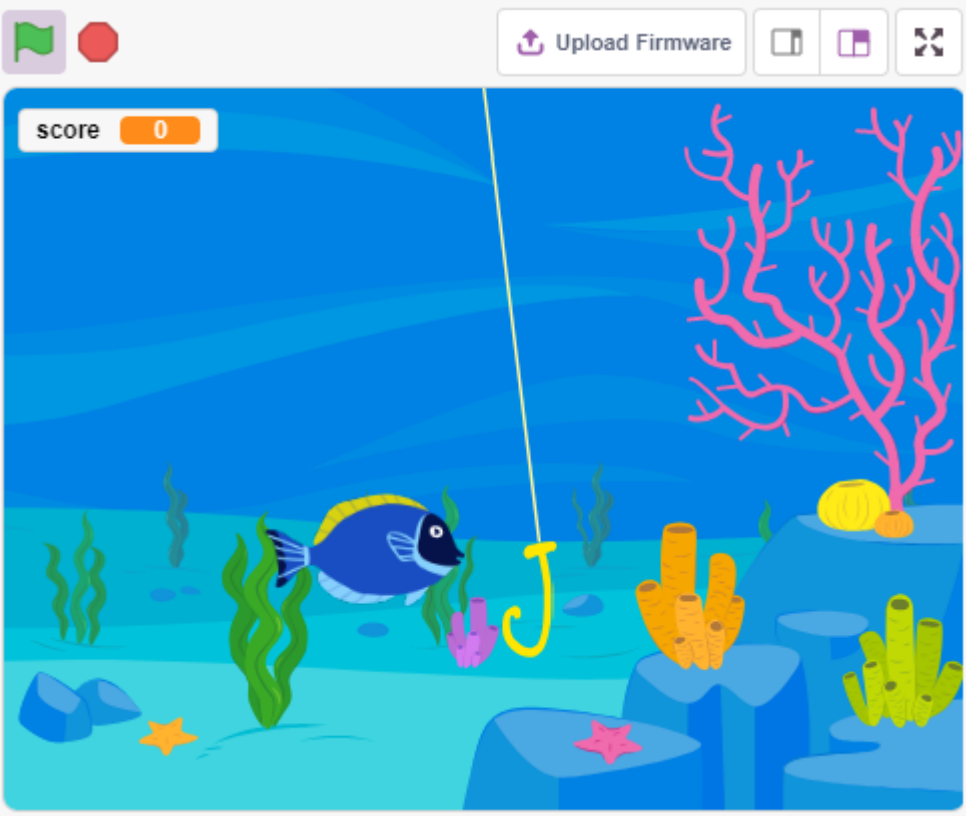

- **Ball** スプライトに戻って、 **crush** が受信された場合（ **Ball** スプライトが **Block1** スプライトのクローンに触れた場合）、 **Ball** は逆の方向にポップします。



## 7.22 2.19 ゲーム - 釣り

このゲームでは、ボタンを使って釣りを楽しむことができます。

スクリプトを実行すると、魚がステージの左右を泳ぐので、魚が釣り針に近づいたとき（長押し推奨）にボタンを押して魚を釣ります。釣れた魚の数は自動的に記録されます。




### 7.22.1 必要な部品

このプロジェクトには、以下の部品が必要です。

全ての部品を含むキットを購入すると非常に便利です。リンクはこちら：

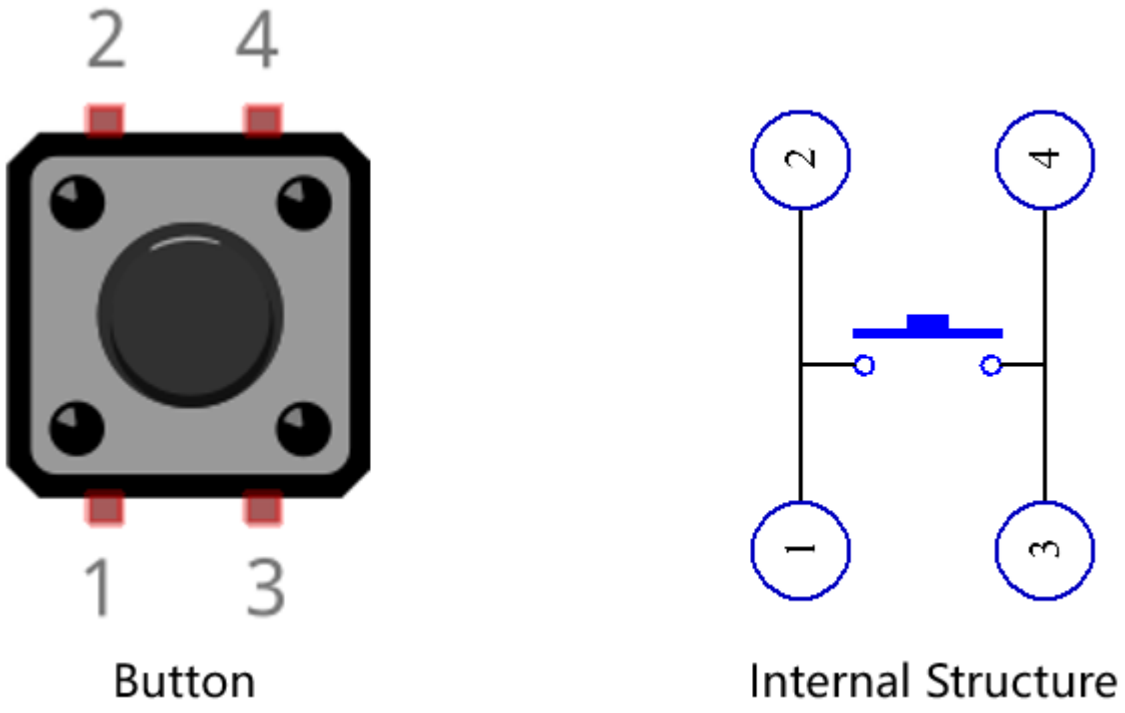
| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから、個別にも購入できます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 抵抗器 | |
| コンデンサ | |
| ボタン | |

### 7.22.2 回路の作成

ボタンは 4 ピンのデバイスで、ピン 1 はピン 2 に、ピン 3 はピン 4 に接続されています。ボタンを押すと、4 つのピンが接続されて回路が閉じます。

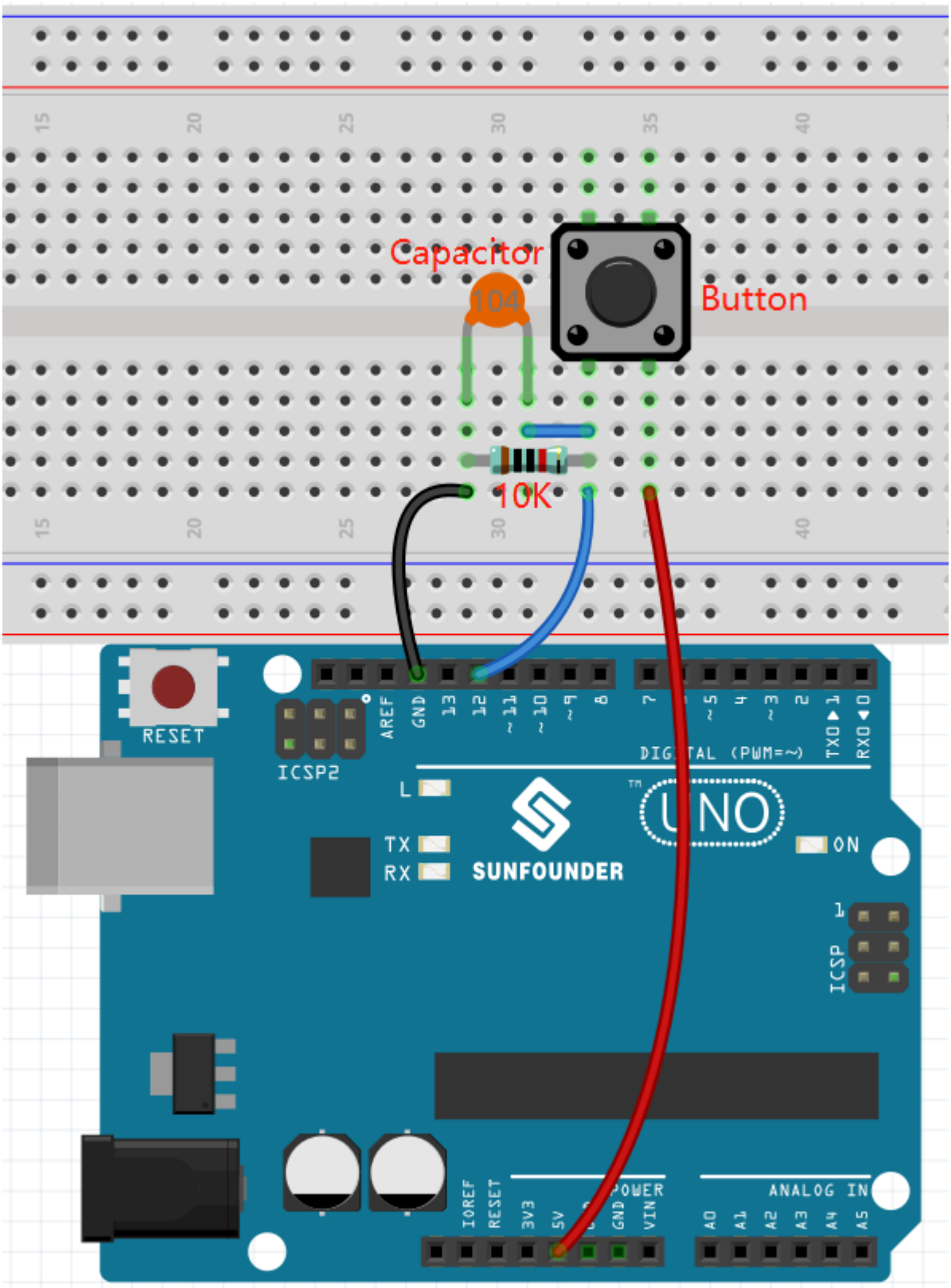



以下の図に従って回路を組み立てます。

- ボタンの左側のピンの一つを、プルダウン抵抗と 0.1uF (104) のキャパシタ（ボタンが動作する際のジッタ

を除去し、安定したレベルを出力するため）に接続されているピン 12 に接続します。

- 抵抗とキャパシタのもう一方の端を GND に、ボタンの右側のピンの一つを 5V に接続します。

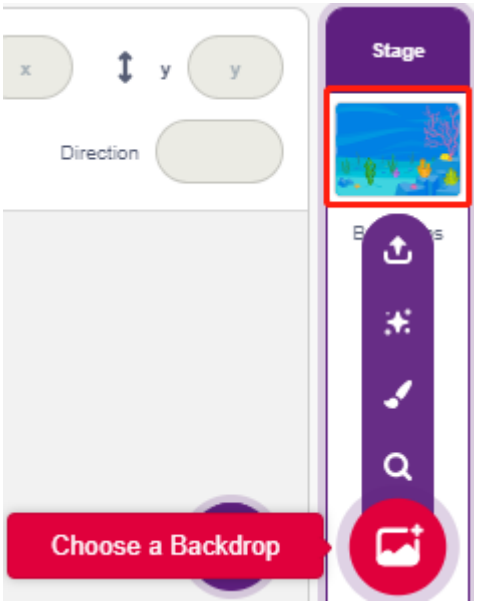

Capacitor
104
Button
10K
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
SUNFOUNDER
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

### 7.22.3 プログラミング

まず、**Underwater** の背景を選択し、**Fish** スプライトを追加してステージ上で左右に泳がせます。次に、**Fishhook** スプライトを描き、ボタンで制御して釣りを開始します。 **Fish** スプライトがフック状態（赤くなる）で **Fishhook** スプライトに触れると、フックされます。

**1. 背景の追加**

**Choose a Backdrop** ボタンを使って、 **Underwater** の背景を追加します。

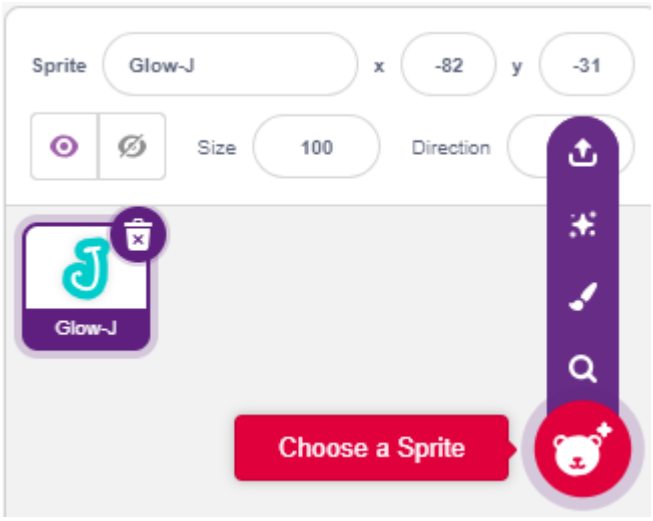



**2. 釣り針スプライト**

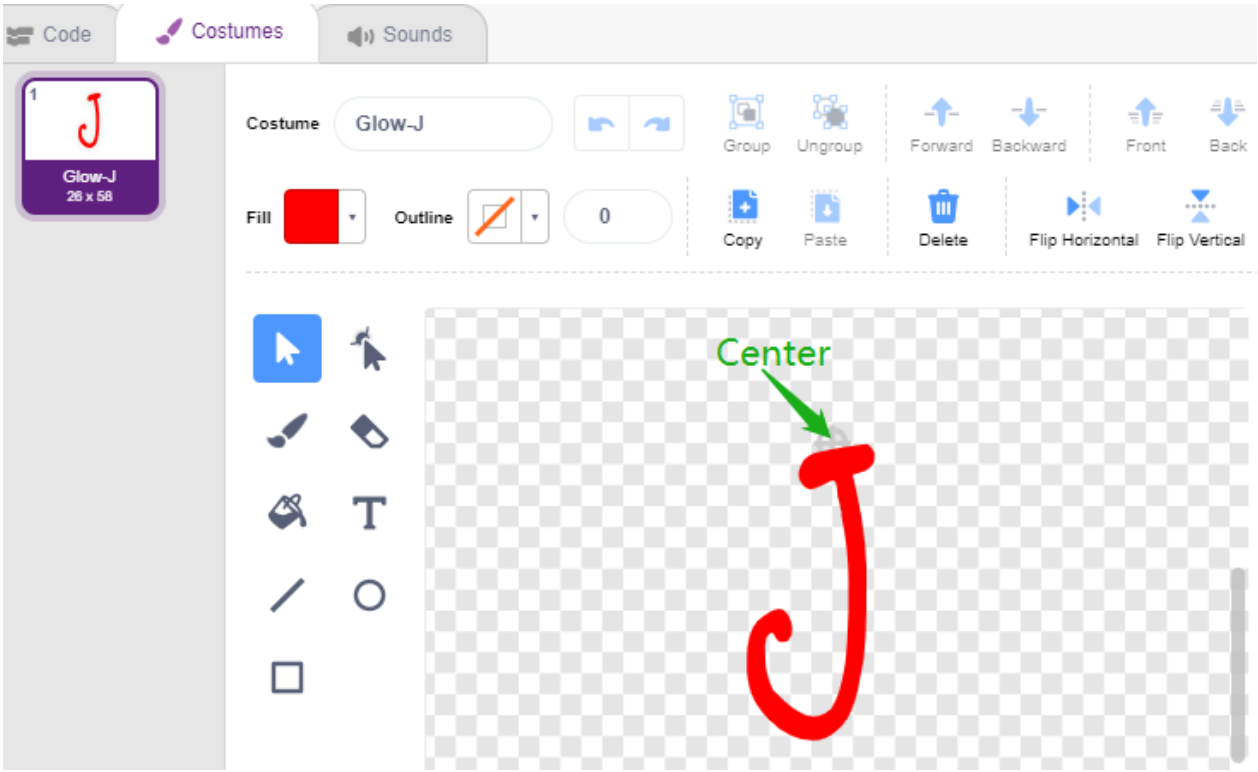
**Fishhook** スプライトは通常、黄色の状態で水中に留まっています。ボタンが押されると、釣り状態（赤）になり、ステージの上に移動します。

Pictoblox には **Fishhook** スプライトがありませんので、 **Glow-J** スプライトを修正して釣り針のように見せることができます。

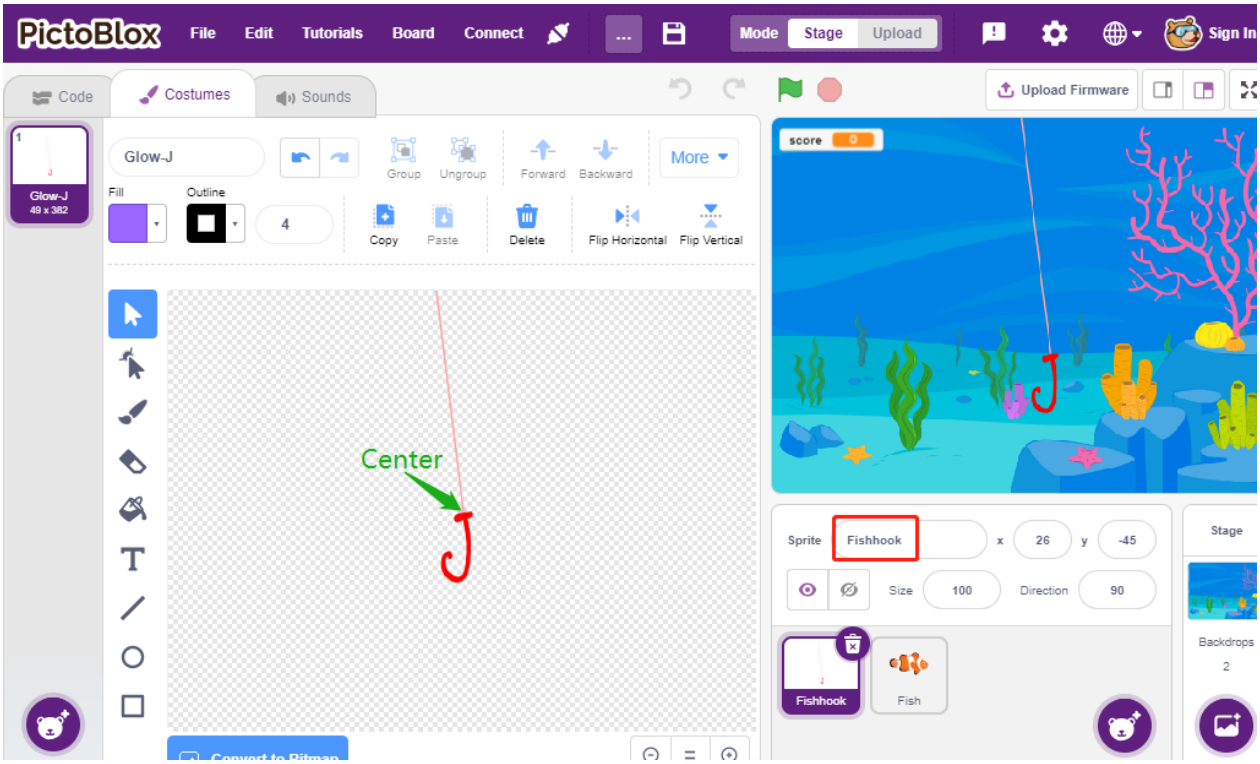
- **Choose a Sprite** で **Glow-J** スプライトを追加します。



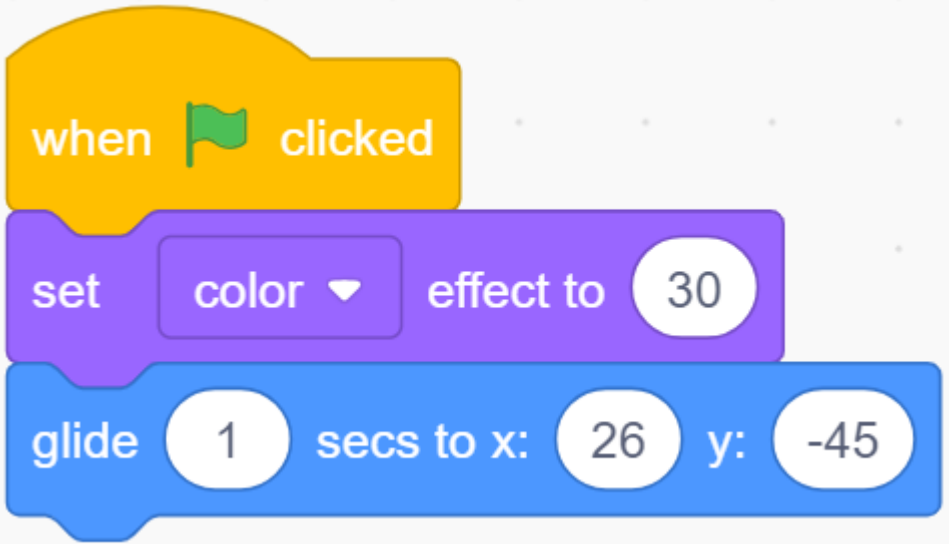

- 次に、 **Glow-J** スプライトの **Costumes** ページに移動し、画面の Cyan の塗りつぶしを選択して削除します。その後、J の色を赤に変更し、その幅も縮小します。最も重要な点は、その上部を中心点ちょうどに配置することです。



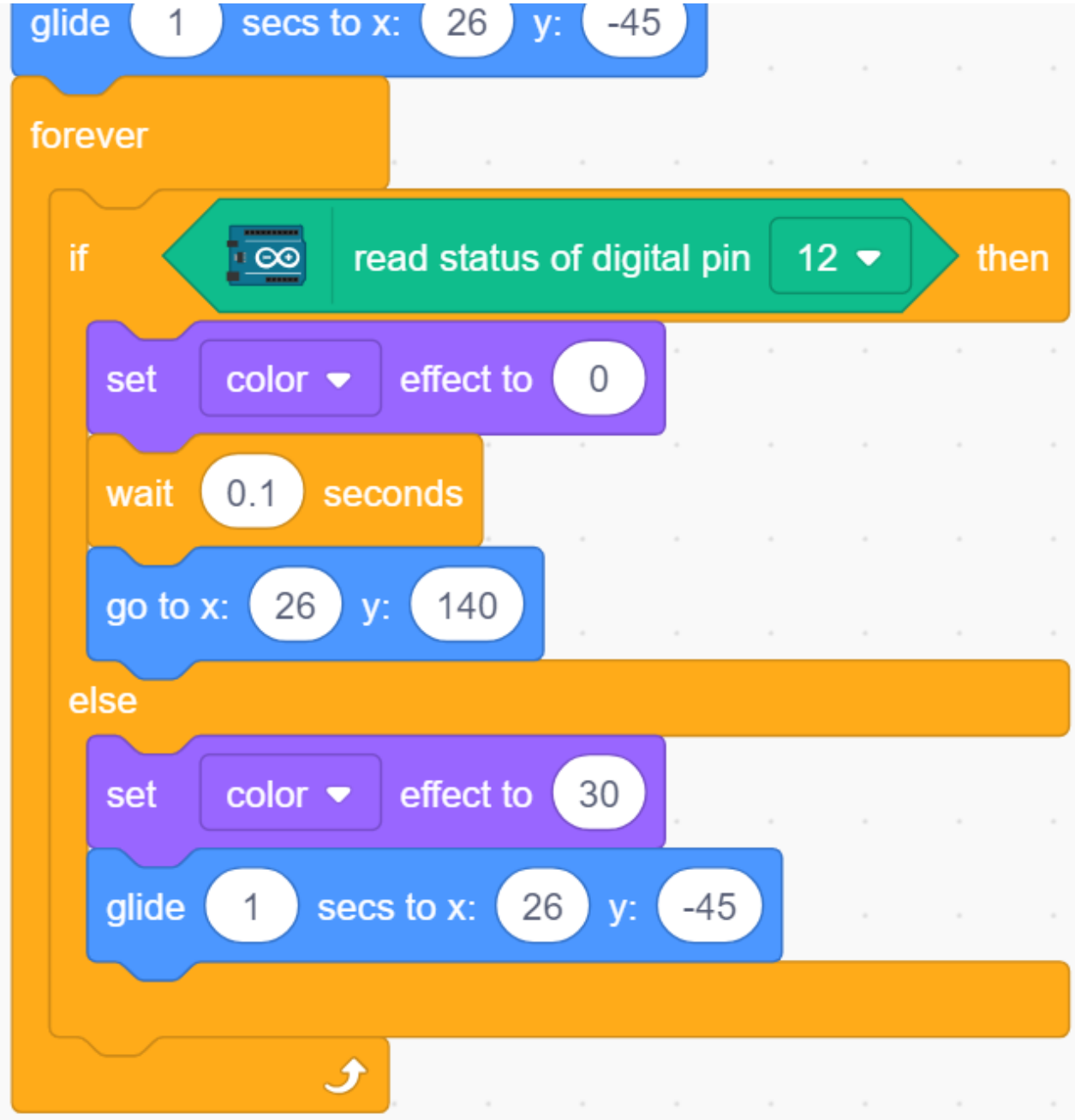

- **Line tool** を使用して、中心点から上方向に可能な限り長く線を引きます（ステージ外の線）。スプライトが描かれたら、スプライトの名前を **Fishhook** に変更し、適切な位置に移動します。




- 緑のフラグがクリックされたとき、スプライトの色効果を 30（黄色）に設定し、その初期位置を設定します。



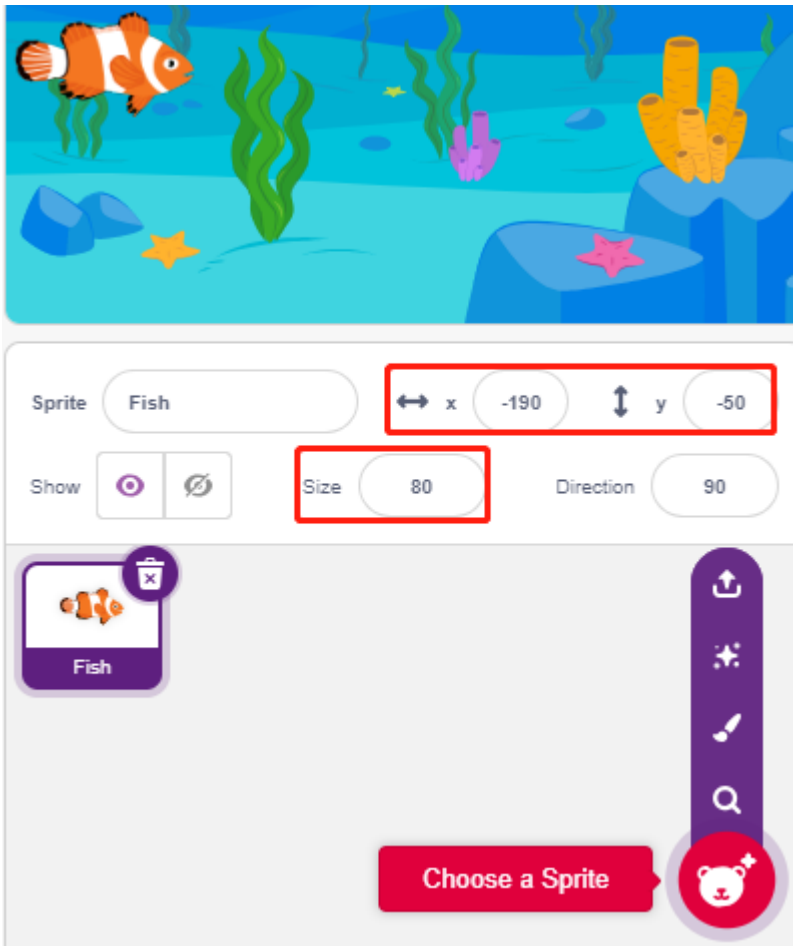

- ボタンが押された場合、色効果を 0（赤、釣り開始状態）に設定し、0.1 秒待った後、 **Fishhook** スプライトをステージの上部に移動します。ボタンを離すと、 **Fishhook** は初期位置に戻ります。



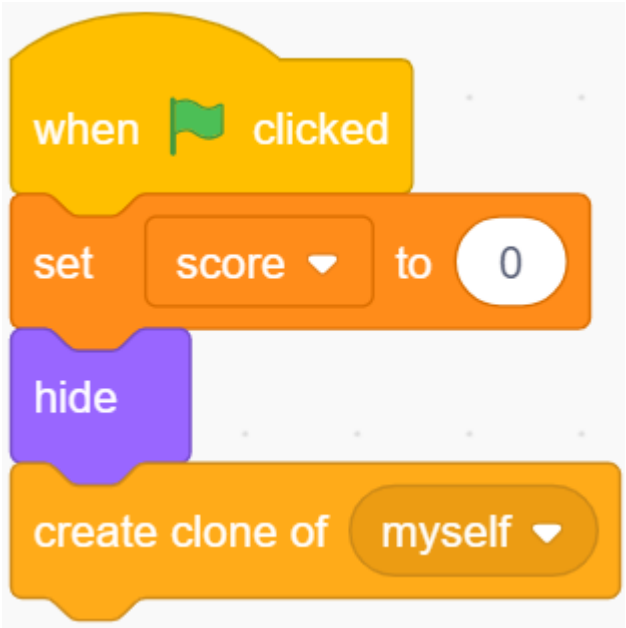

### 3. 魚スプライト

**Fish** スプライトが達成すべき効果は、ステージの左右に移動し、釣りの状態である **Fishhook** スプライトに遭遇したとき、それを縮小して特定の位置に移動させてから消失させ、新しい **fish** スプライトを再度クローンすることです。

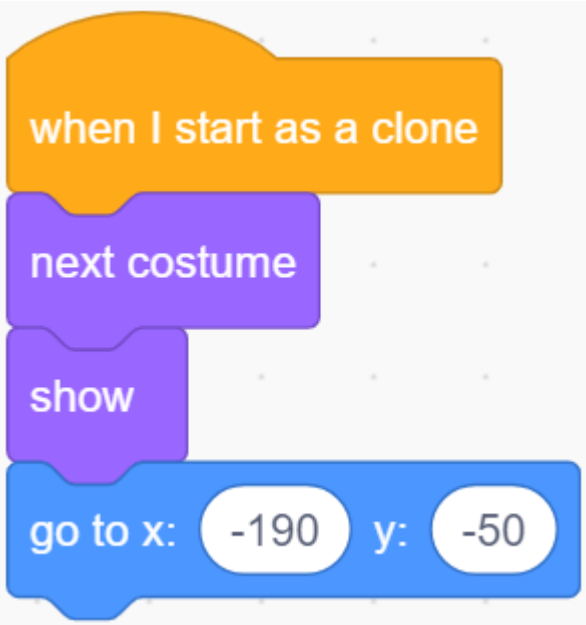
- 今度は **fish** スプライトを追加し、そのサイズと位置を調整します。




- 釣れた魚の数を保存する変数 **score** を作成し、このスプライトを隠し、それをクローンします。



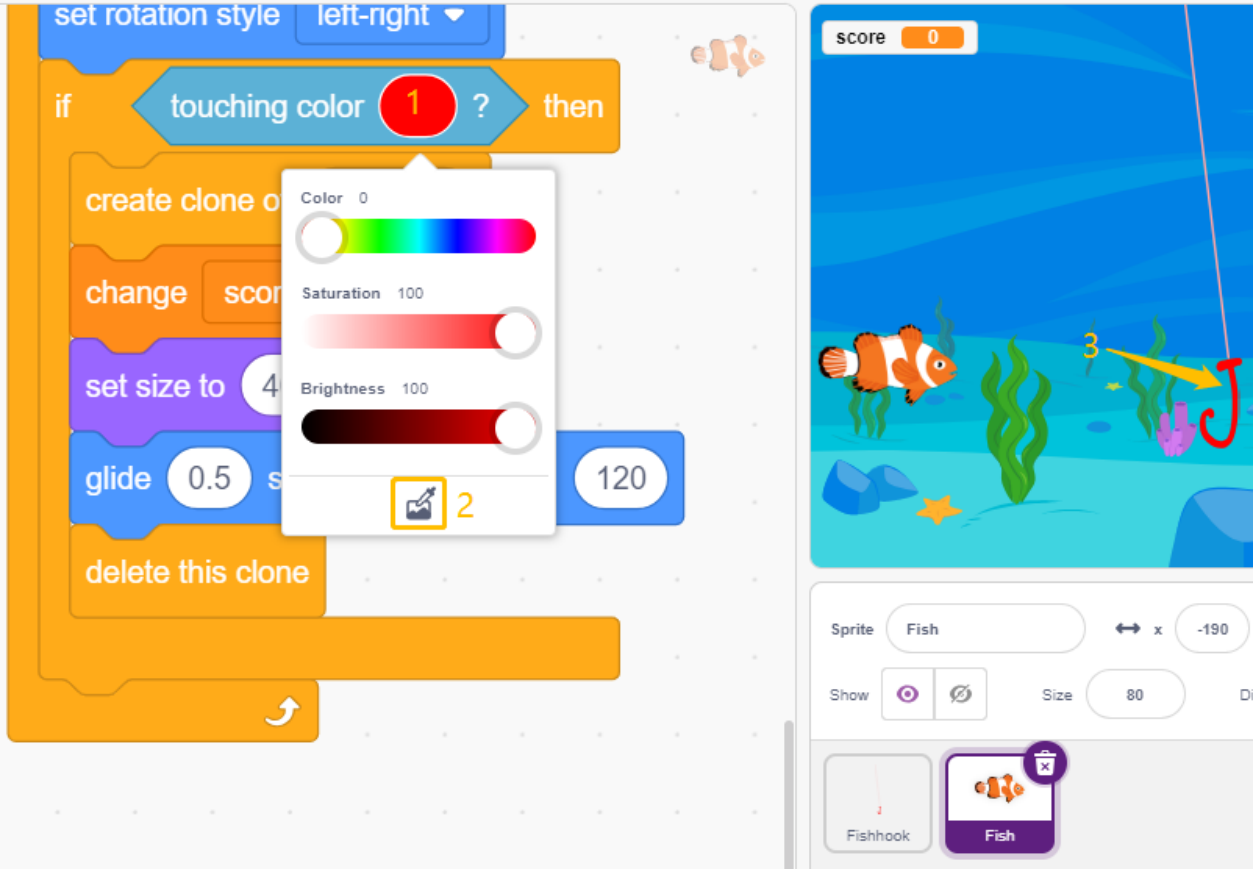

- **fish** スプライトのクローンを表示し、そのコスチュームを切り替え、最後に初期位置を設定します。




- **fish** スプライトのクローンを左右に移動させ、エッジに触れると反射させます。


- **Fishhook** スプライトを通過しても、 sss**fish** スプライト（クローンのもの）は反応しません。釣りの状態（赤くなる）で **Fishhook** スプライトに触れると、それが捕まり、その時点でスコア（変数 score）+1、そしてスコアアニメーションも表示されます（サイズを 40 %縮小し、スコアボードの位置に素早く移動して消失）。同時に、新しい魚が作成され（新しい魚スプライトのクローン）、ゲームが続行されます。

**注釈:** [Touch color] ブロックの色エリアをクリックして、ステージの **Fishhook** スプライトの赤色を取得するためのアイドロッパーツールを選択する必要があります。適当に色を選択すると、この [Touch color] ブロックは機能しません。




## 7.23 2.20 ゲーム - 白いタイルをタップしないで

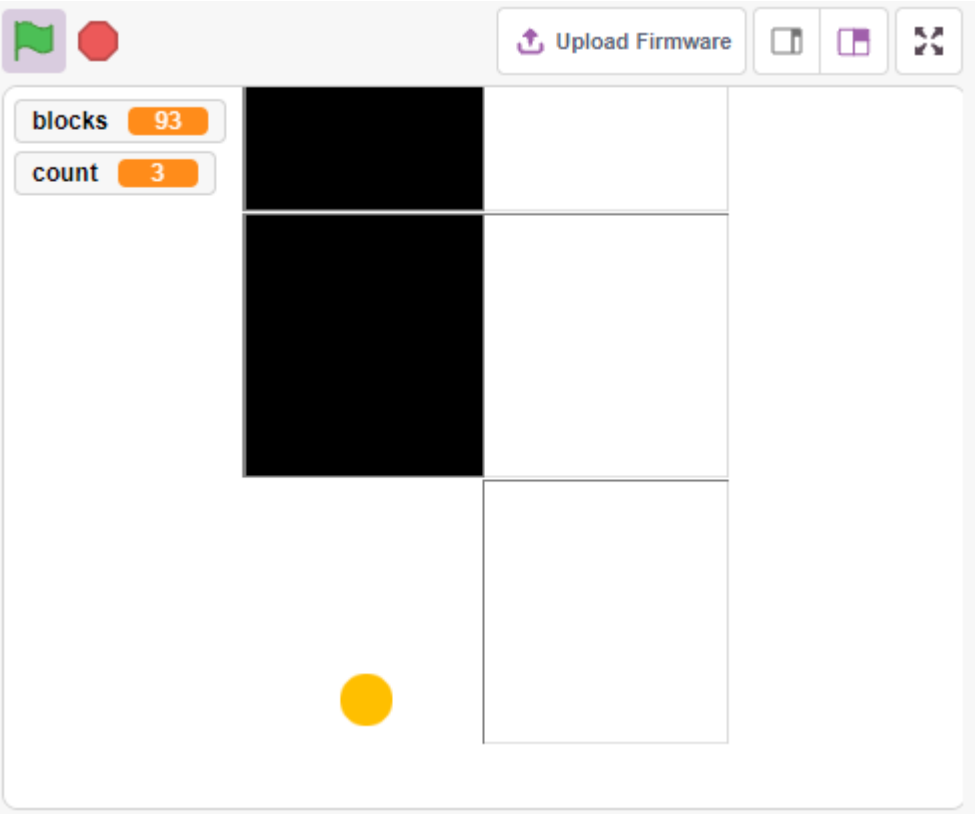
多くの方がスマートフォンでこのゲームをプレイしたことがあると思います。このゲームは、ランダムに出現する黒いタイルをタップしてポイントを追加し、スピードはどんどん速くなります。白いブロックをタップしたり、黒いブロックを見逃したりするとゲームオーバーです。

今回は、PictoBlox を使用してこれを再現します。

ブレッドボード上に 2 つの IR 障害物回避モジュールを縦に挿入します。手を IR モジュールの上に置くと、ステージ上に点滅するドットが表示され、タップが行われたことを示します。

黒いブロックにタップすると、スコアは 1 増え、白いブロックに触れると、スコアは 1 減ります。

ステージ上の黒いブロックの位置に応じて、左の IR モジュールの上か、右の IR モジュールの上に手を置くかを決める必要があります。




### 7.23.1 必要な部品

このプロジェクトには、以下のコンポーネントが必要です。

全てのキットを購入するのは確かに便利です。リンクはこちら:

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

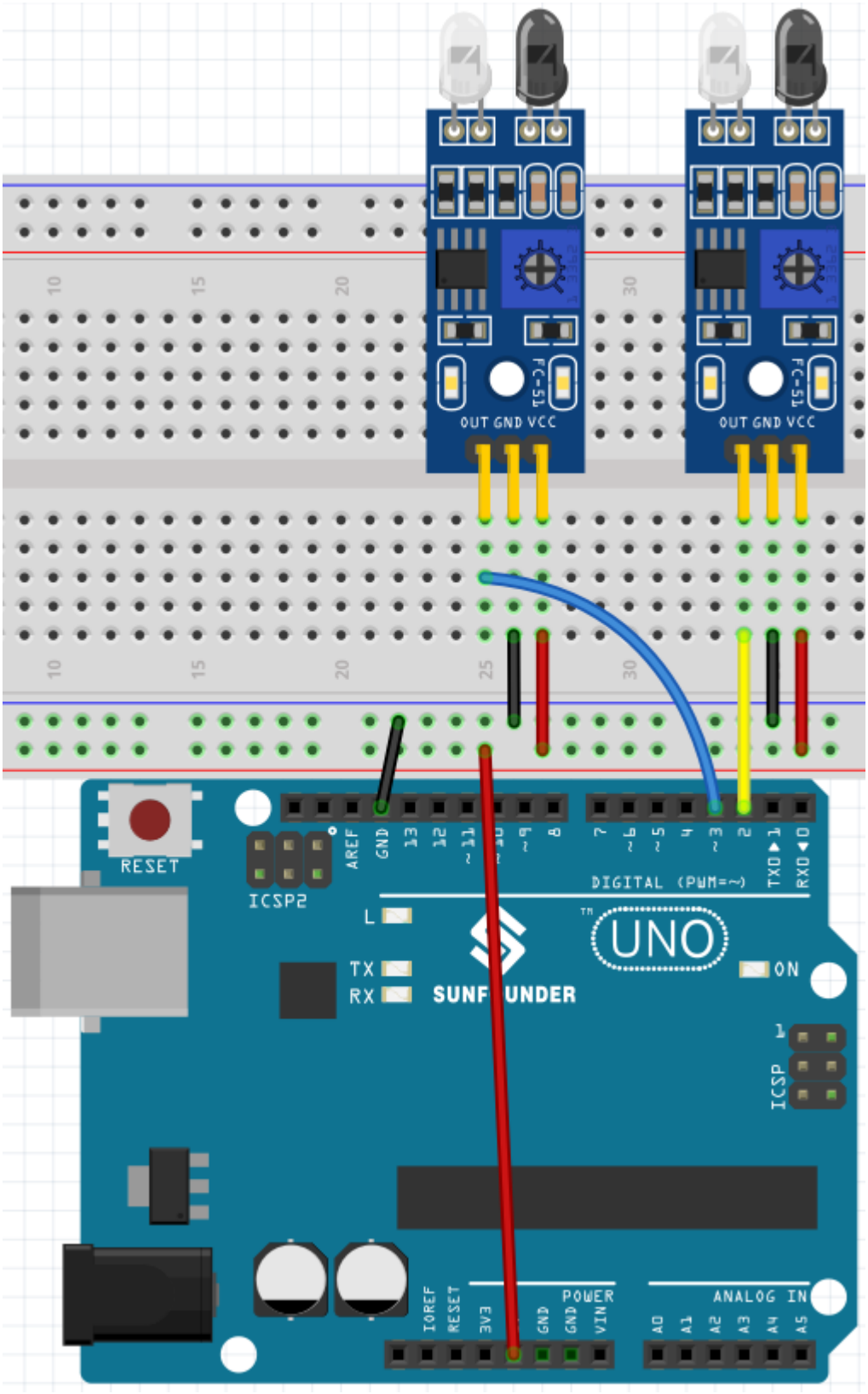
以下のリンクから個別に購入することもできます。

| コンポーネント紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| ブレッドボード | |
| ジャンパーワイヤー | |
| 障害物回避モジュール | |

### 7.23.2 回路の作成

障害物回避モジュールは、出力が通常は高く、障害物が検出されると低くなる距離調整可能な赤外線近接センサーです。

以下の図に従って回路を組み立てます。


OUT GND VCC
FC-51
OUT GND VCC
FC-51
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5

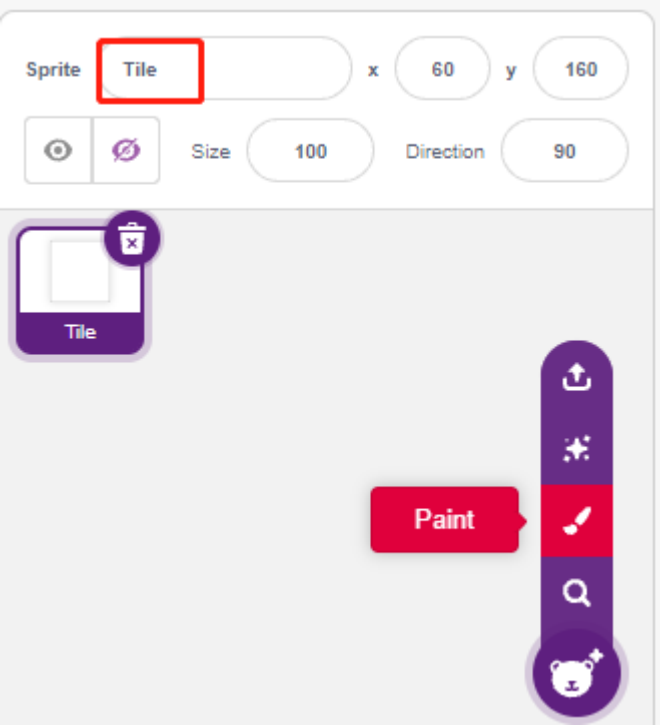
### 7.23.3 プログラミング

ここでは 3 つのスプライト、 **Tile** 、 **Left IR** 、 **Right IR** が必要です。

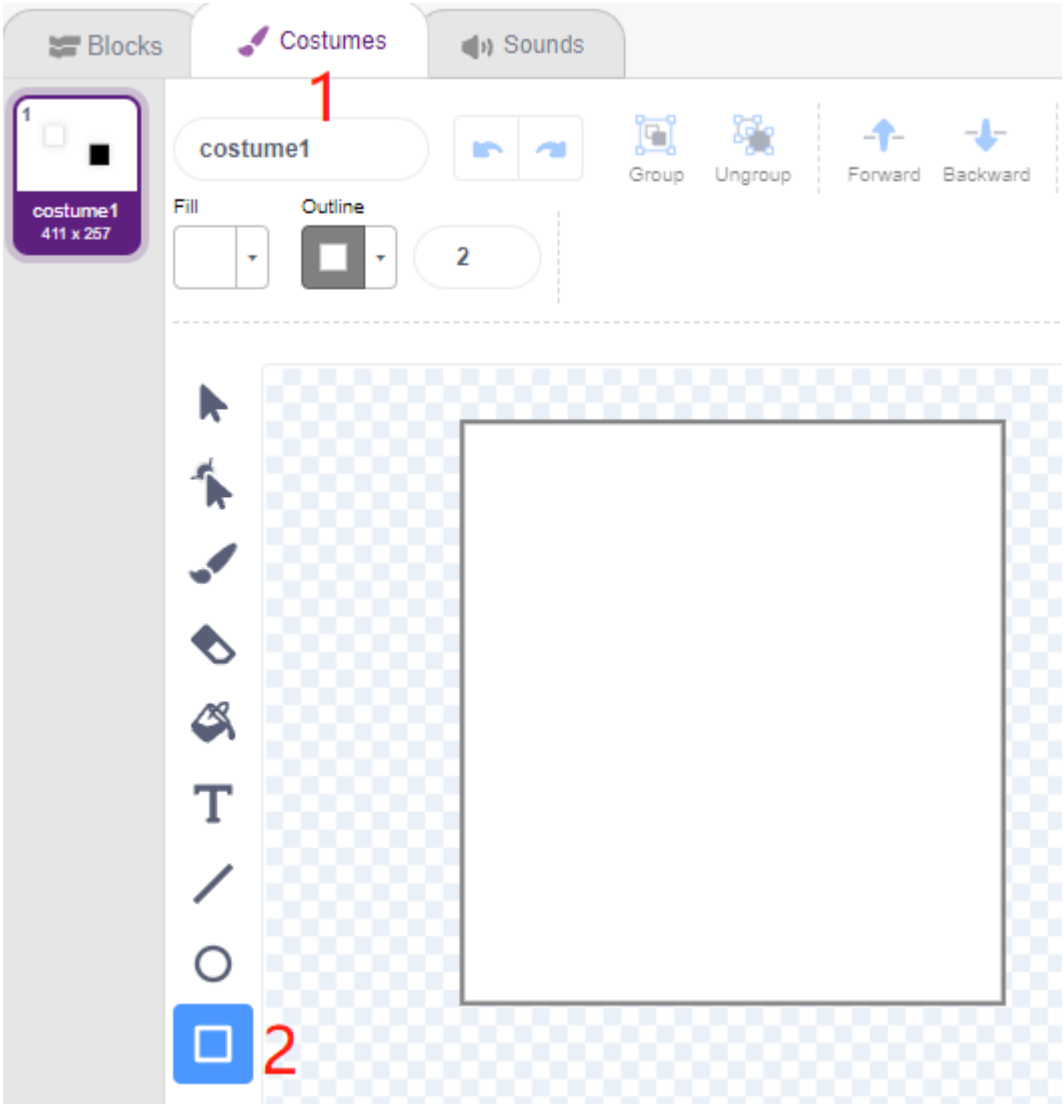
- **Tile** スプライト: 黒と白のタイルが交互に下方向に移動する効果を実現するために使用されます。スマートフォンのこのゲームは通常 4 列ですが、ここでは 2 列だけを行います。
- **Left IR** スプライト: クリック効果を実現するために使用されます。左の IR モジュールがあなたの手を感知すると、 **Left IR** スプライトにメッセージ - **left** を送信し、それを起動します。ステージ上の黒いタイルに触れると、スコアは 1 増加し、それ以外の場合はスコアは 1 減少します。
- **Right IR** スプライト: 機能は **Left IR** とほぼ同じですが、 **Right** の情報を受け取ります。

**1. Tile スプライトを描く**。

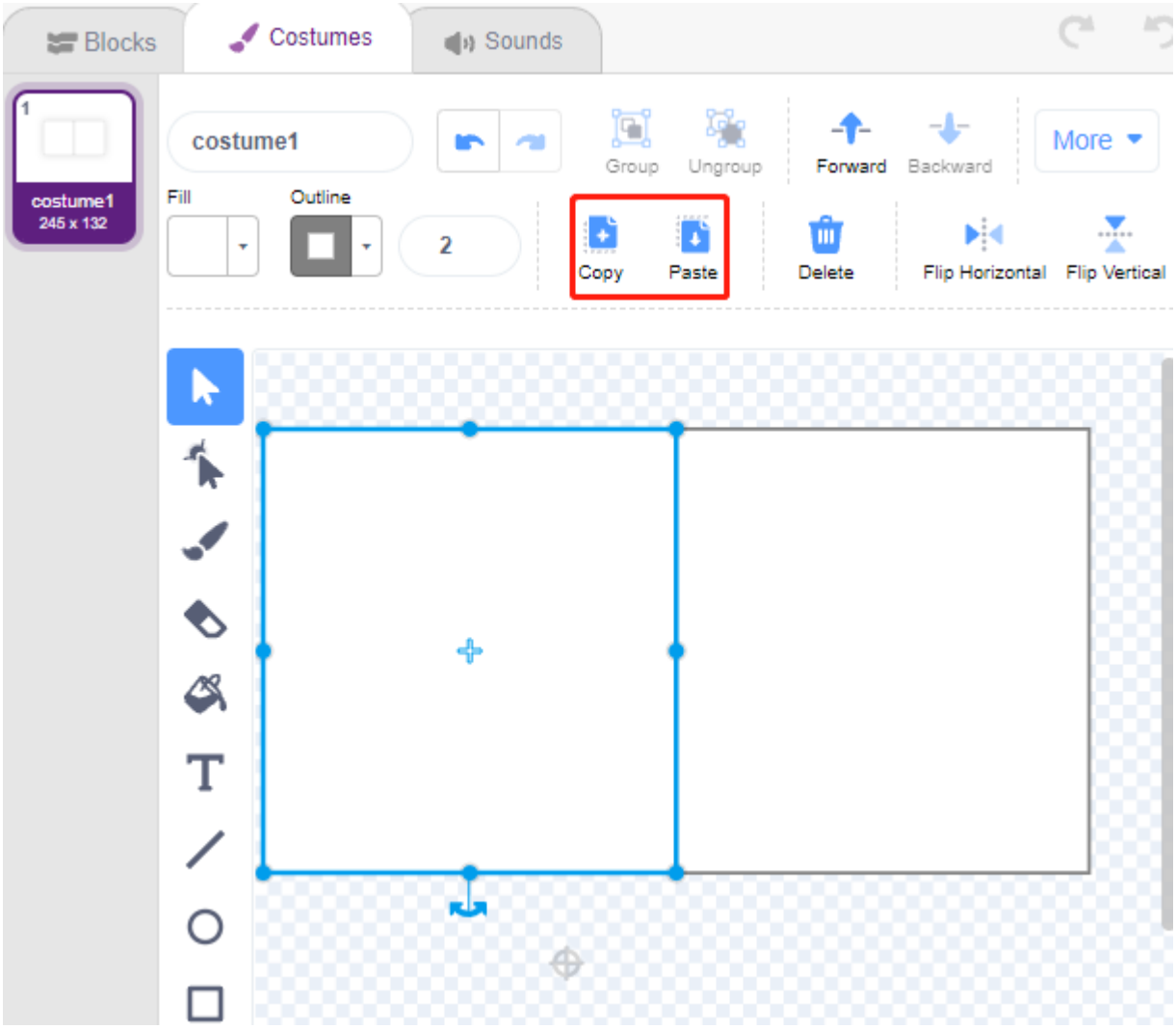
デフォルトのスプライトを削除し、 **Add Sprite** アイコンにマウスオーバーして **Paint** を選択し、空のスプライトが表示され、それを **Tile** と名付けます。



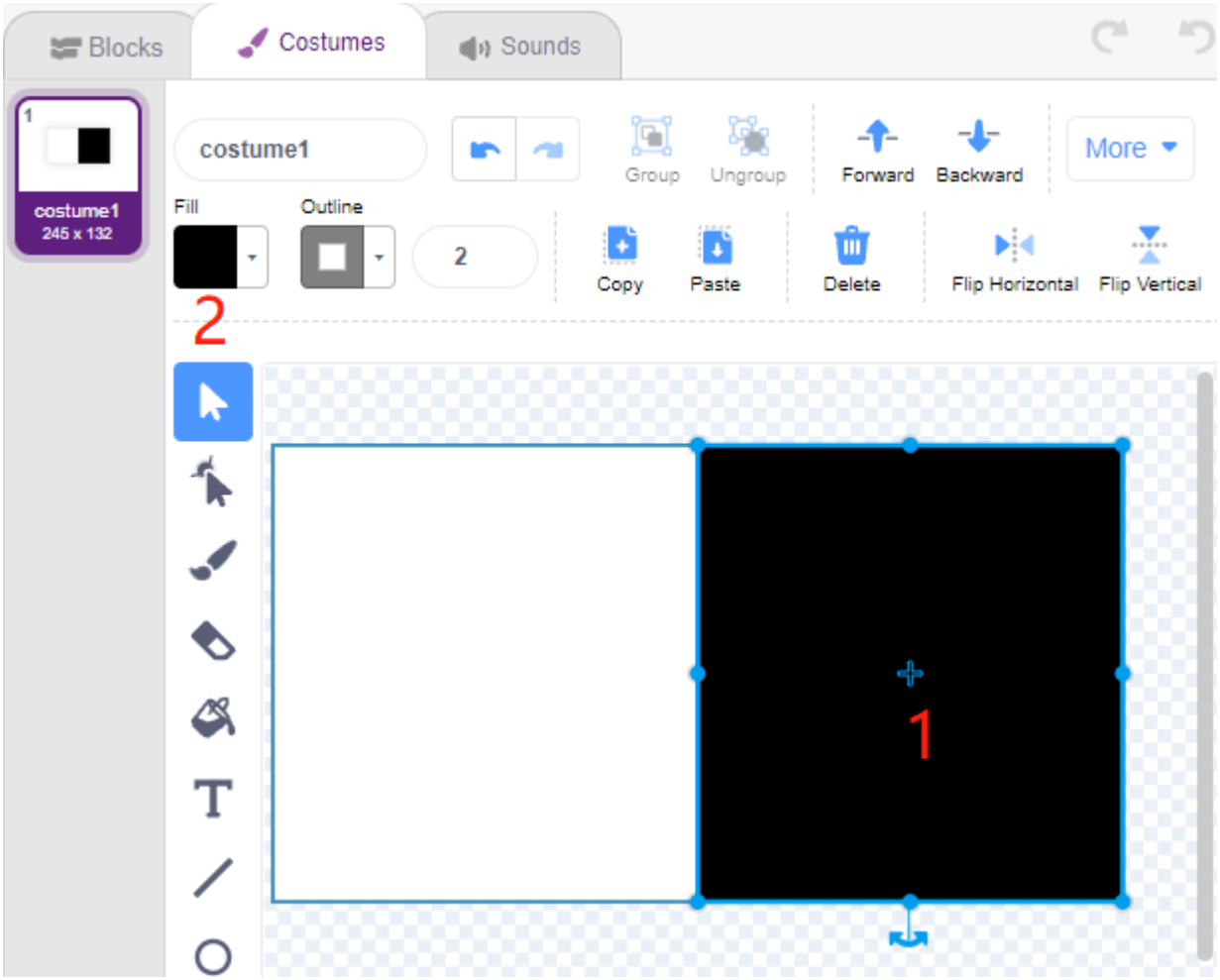

**Costumes** ページに移動し、 **Rectangle** ツールを使用して長方形を描きます。



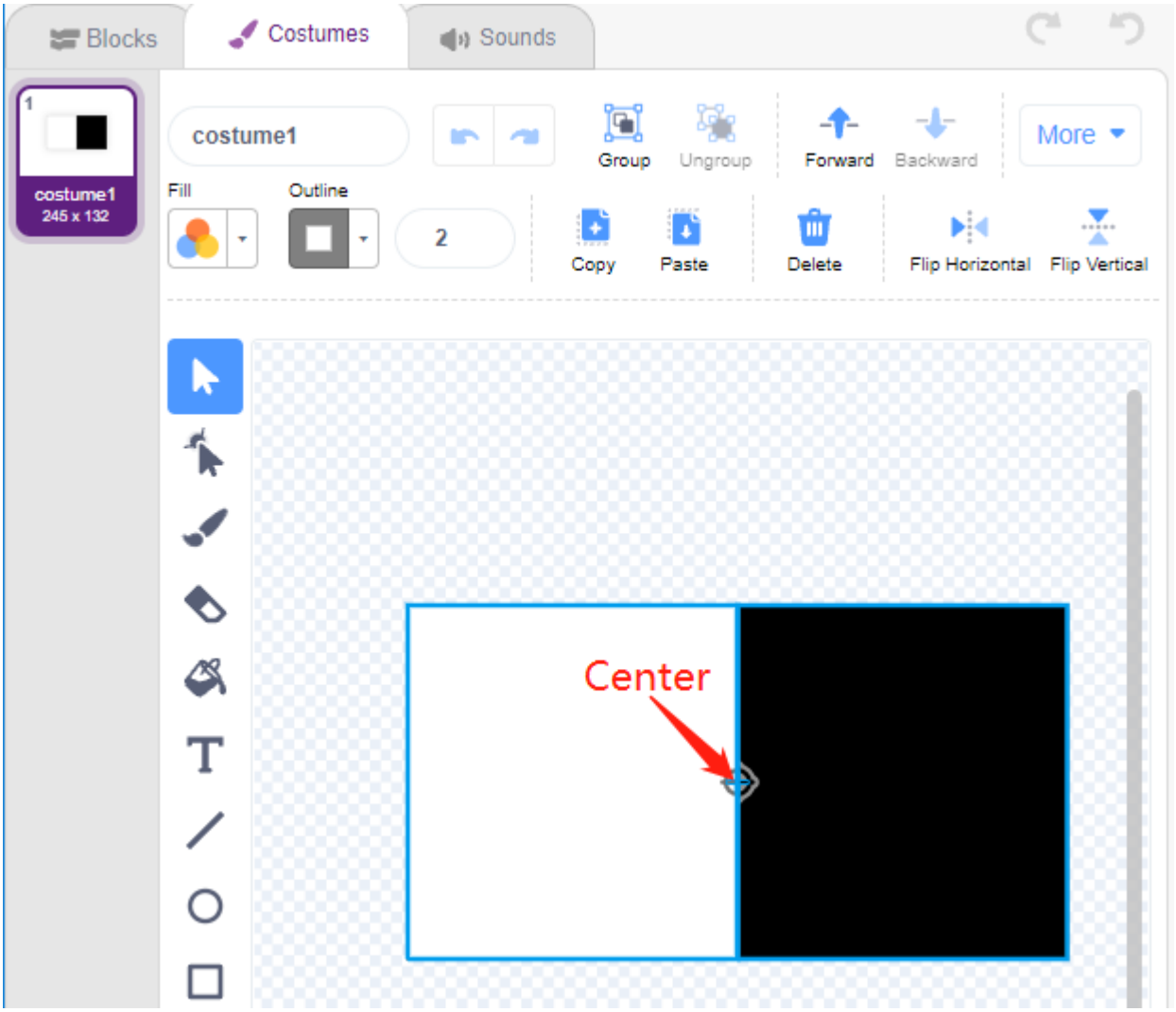

長方形を選択し、 **Copy -> Paste** をクリックして同じ長方形を作成し、2 つの長方形をフラッシュポジションに移動します。



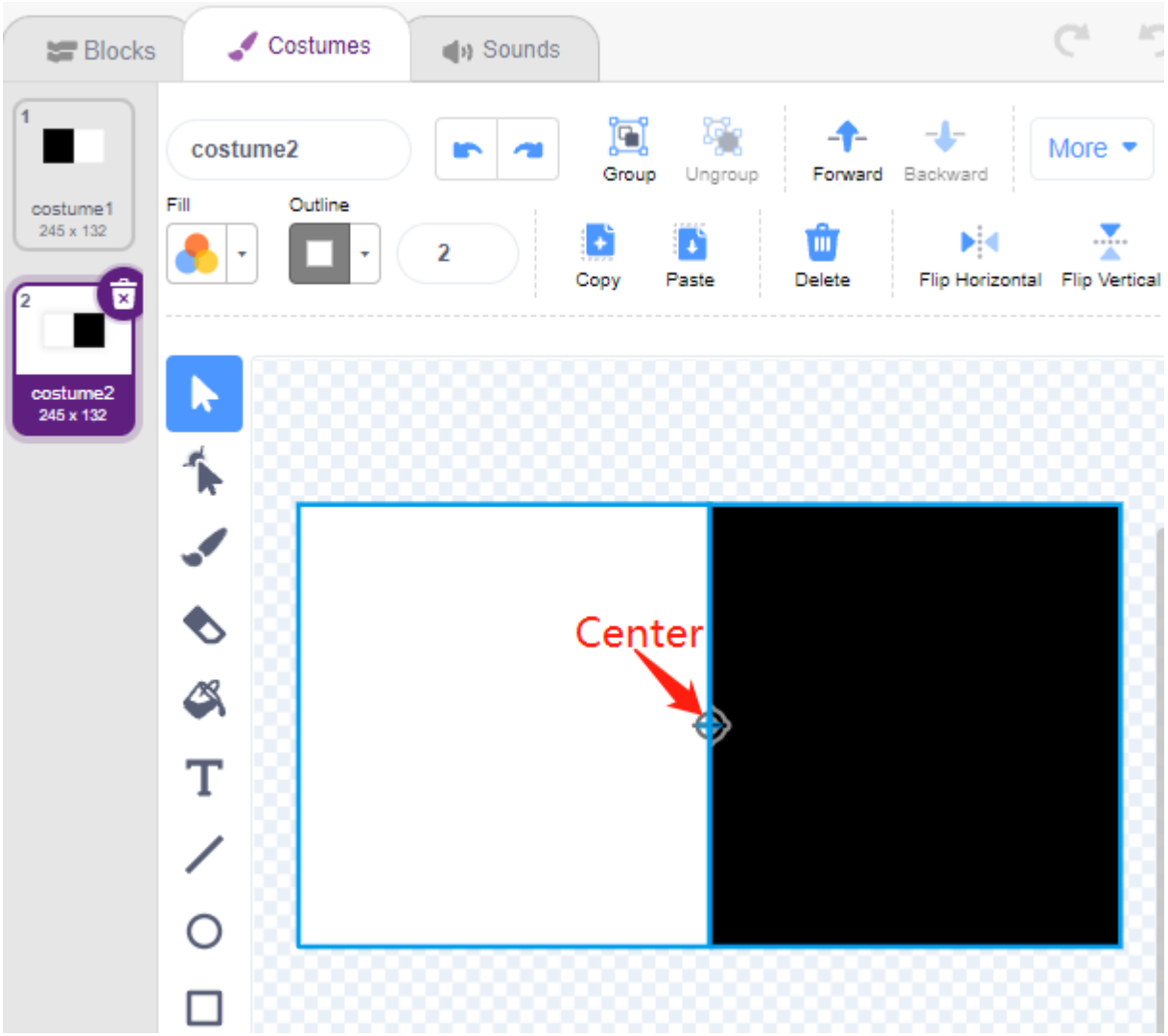

長方形の 1 つを選択し、塗りつぶしの色を黒に選択します。




2 つの長方形を選択して、キャンバスの中心点が一致するように移動します。

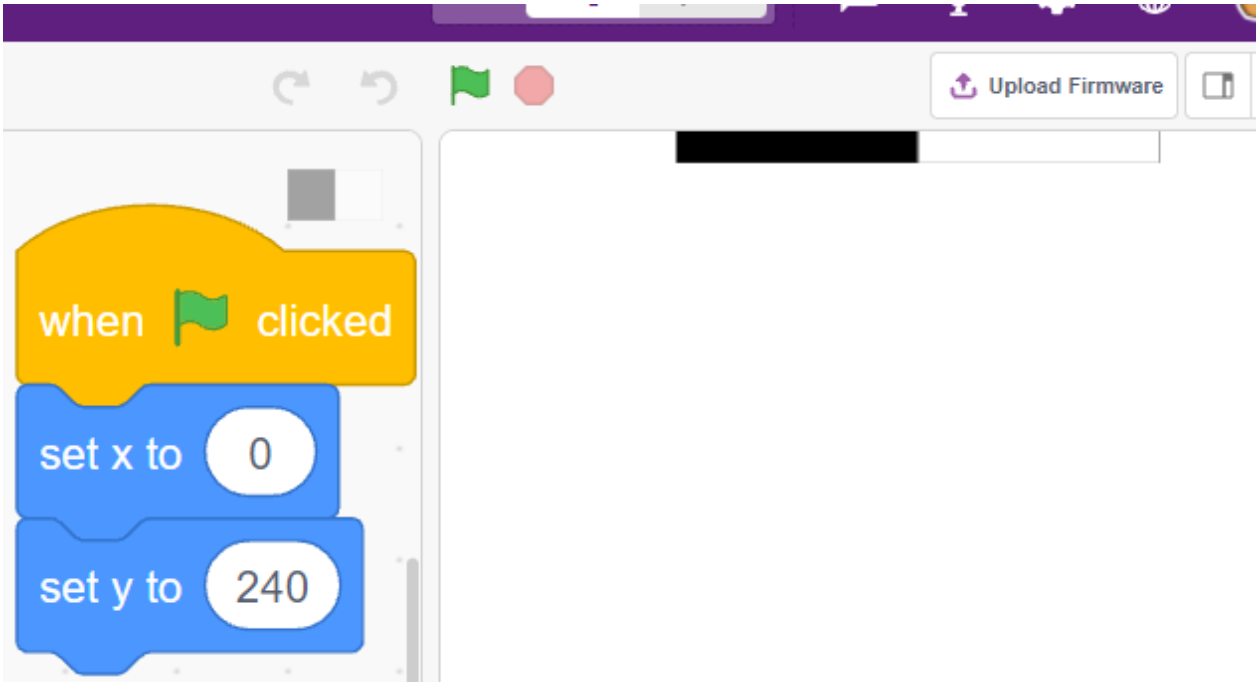



costume1 を複製し、2 つの長方形の塗りつぶしの色を交互に変更します。例えば、costume1 の塗りつぶしの色は左が白で右が黒、costume2 の塗りつぶしの色は左が黒で右が白です。



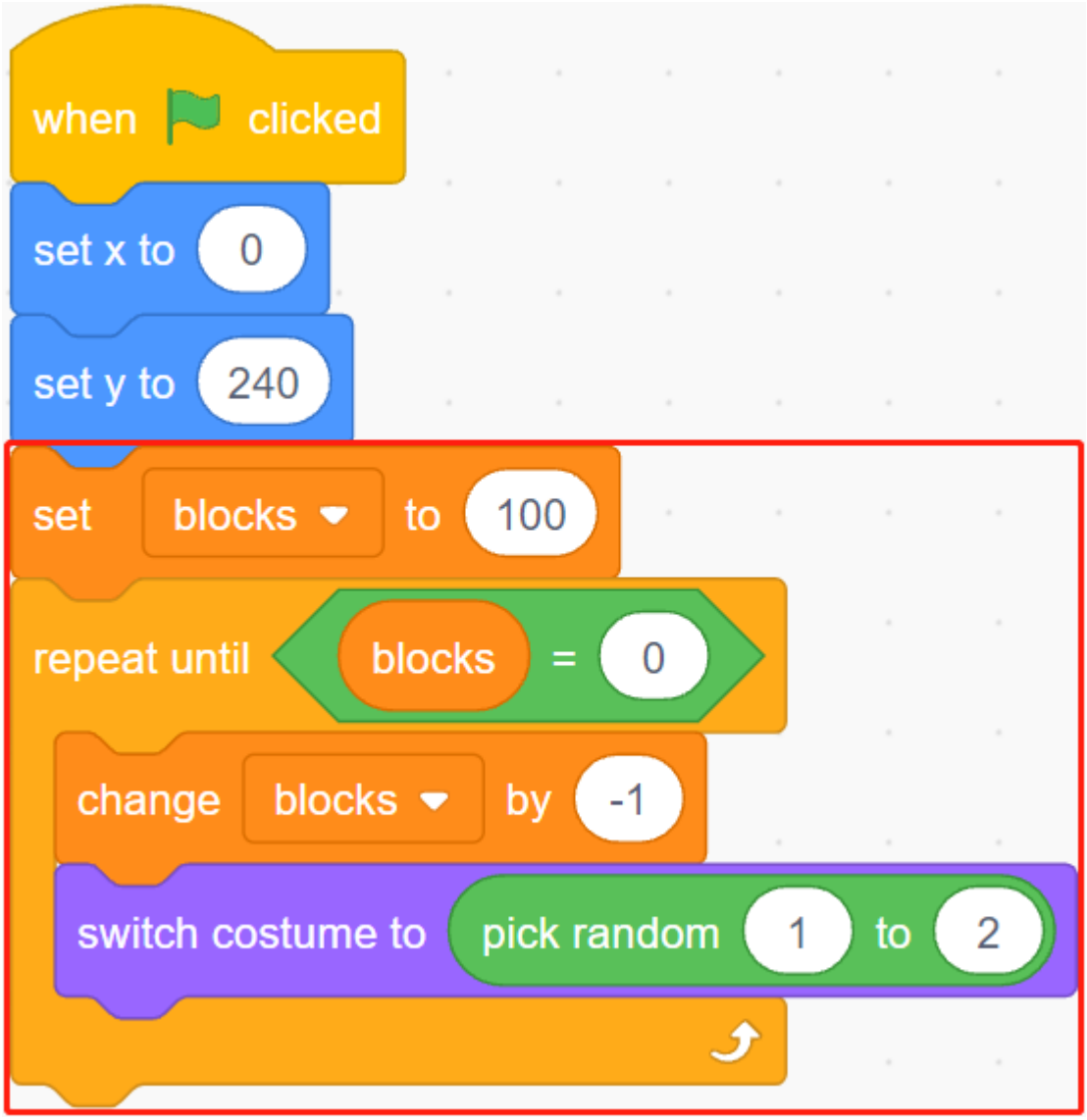

**2. Tile スプライトのスクリプト作成**

**Blocks** ページに戻り、 **Tile** スプライトの初期位置をステージの上部に設定します。

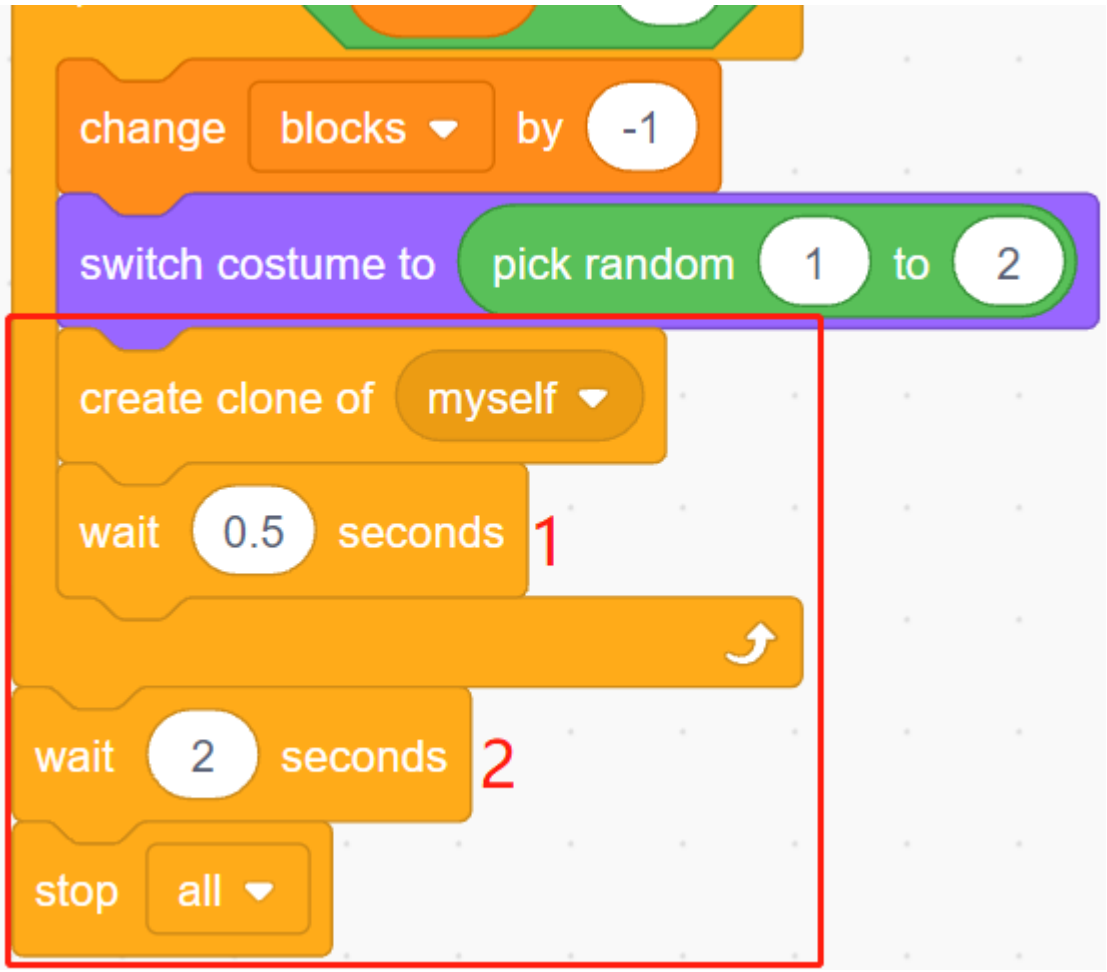



変数 - **blocks** を作成し、**Tile** スプライトが何回現れるかを決定する初期値を設定します。[repeat until] ブロックを使用して、 **blocks** 変数を徐々に減少させ、 **blocks** が 0 になるまで続けます。この間、スプライト **Tile** のコスチュームをランダムに切り替えます。

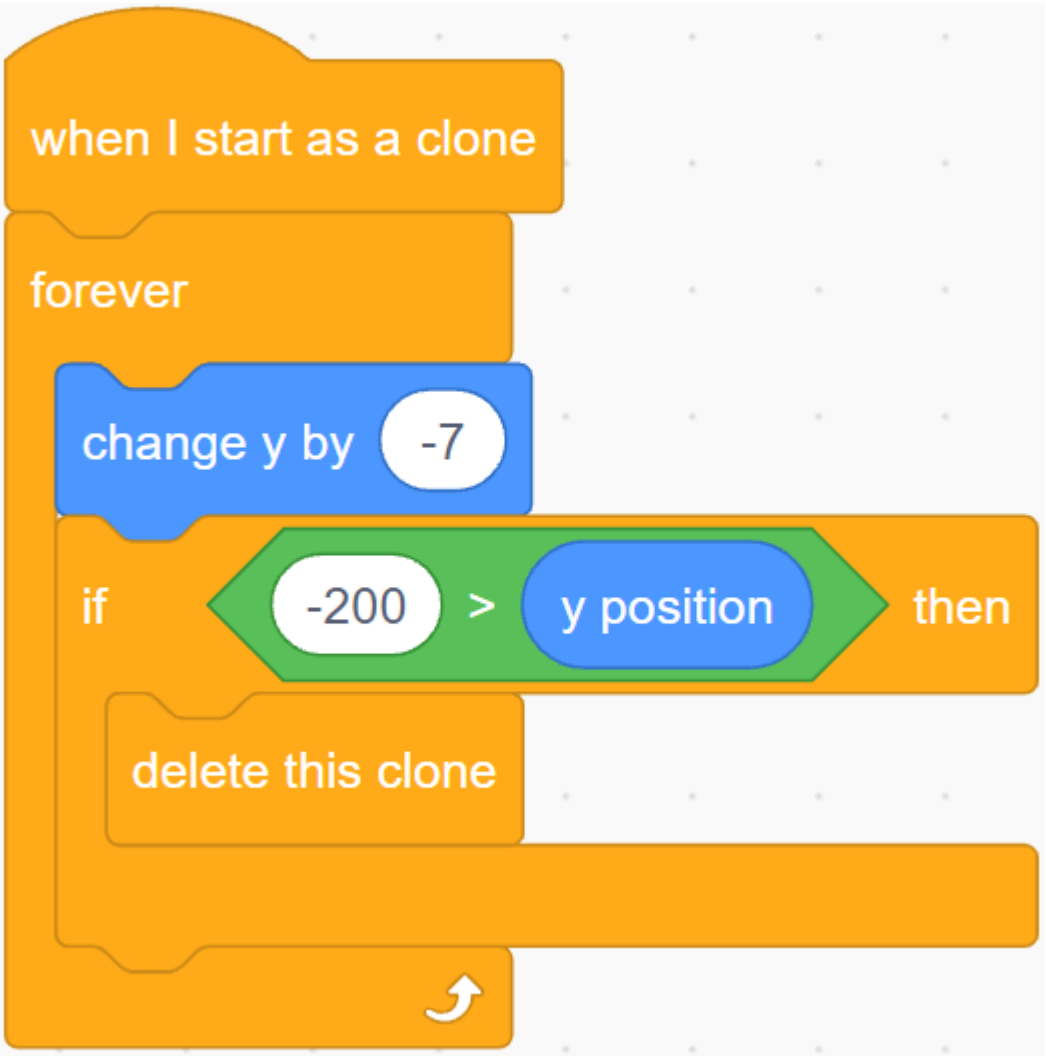
緑の旗をクリックすると、ステージ上で **Tile** スプライトがコスチュームを素早く切り替えるのを見ることができます。



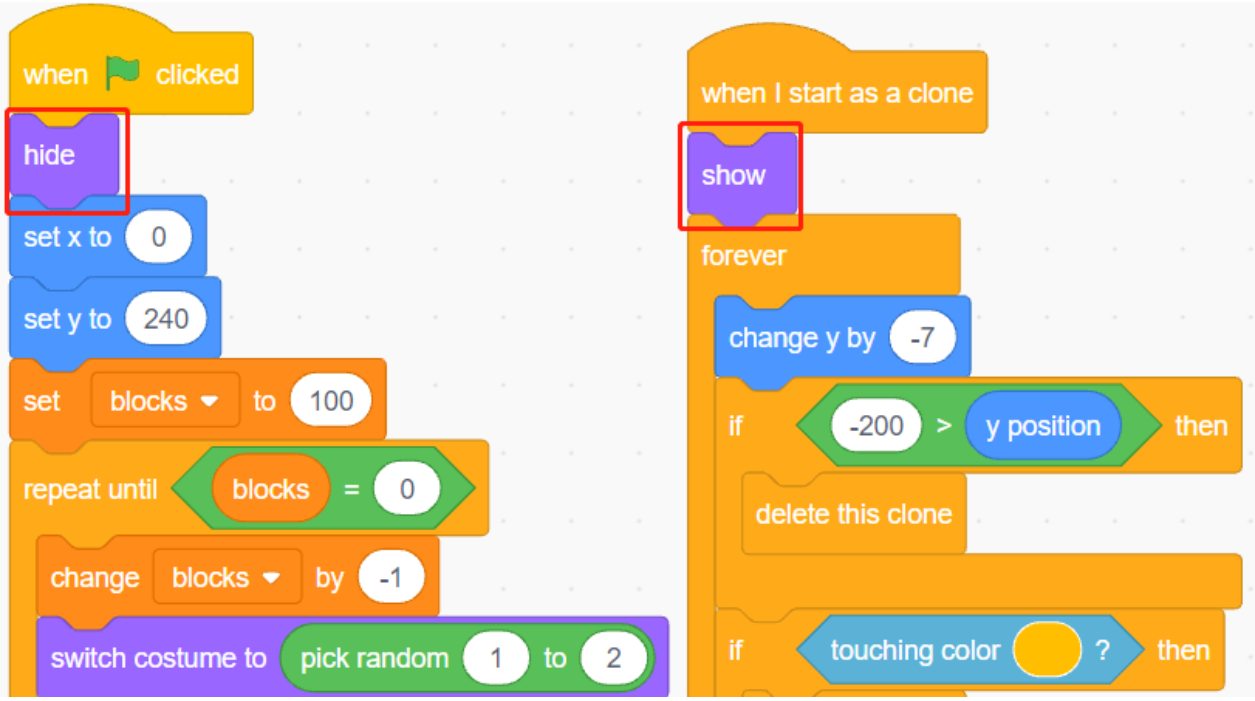

**blocks** 変数が減少している間、 **Tile** スプライトのクローンを作成し、blocks が 0 になったらスクリプトの実行を停止します。ここでは 2 つの [wait () seconds] ブロックが使用されています。1 つ目は **Tile** のクローン間の間隔を制限するため、2 つ目は変数 blocks がすぐにプログラムを停止せずに 0 に減少させるため、最後のタイルスプライトに十分な移動時間を与えるためです。




**Tile** スプライトのクローンがゆっくりと下に移動し、ステージの底に達したら削除するようにスクリプトを書きます。y 座標の変化は、落下速度に影響を与えます。値が大きいほど、落下速度は速くなります。



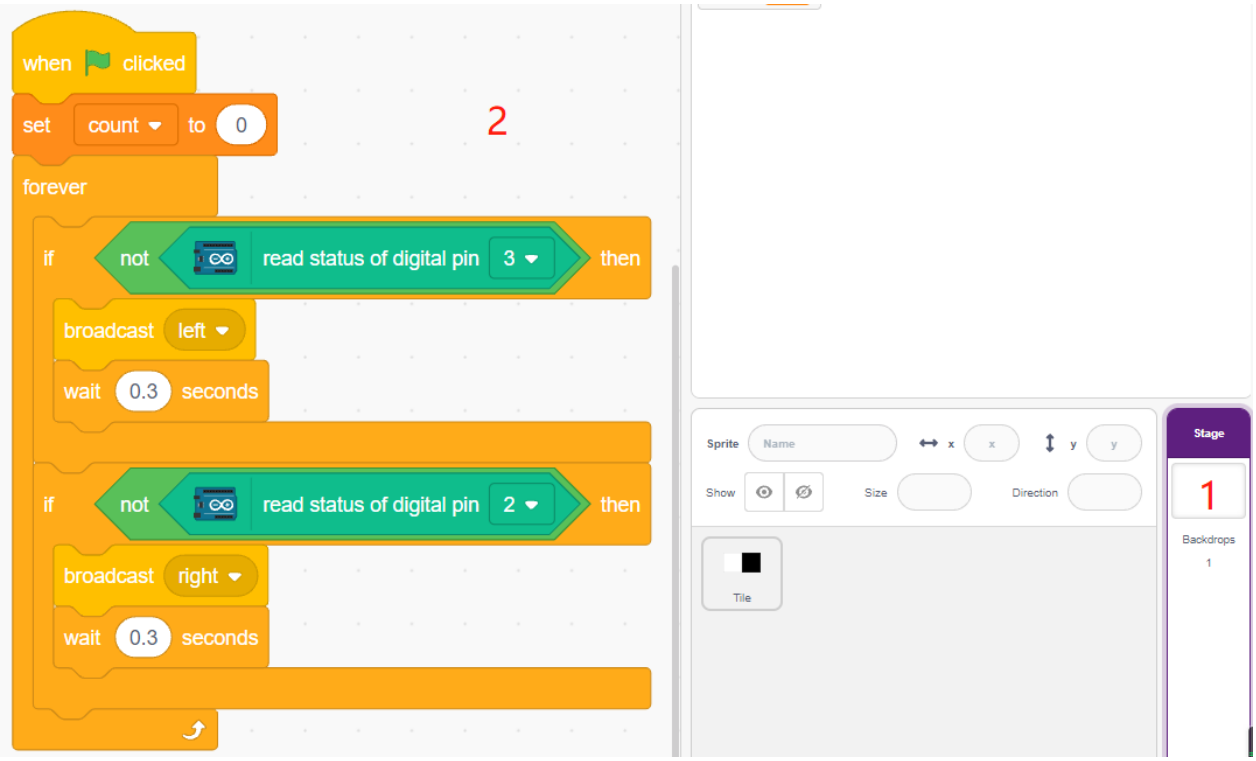

本体を非表示にし、クローンを表示します。




### 3. 2 つの IR モジュールの値を読む

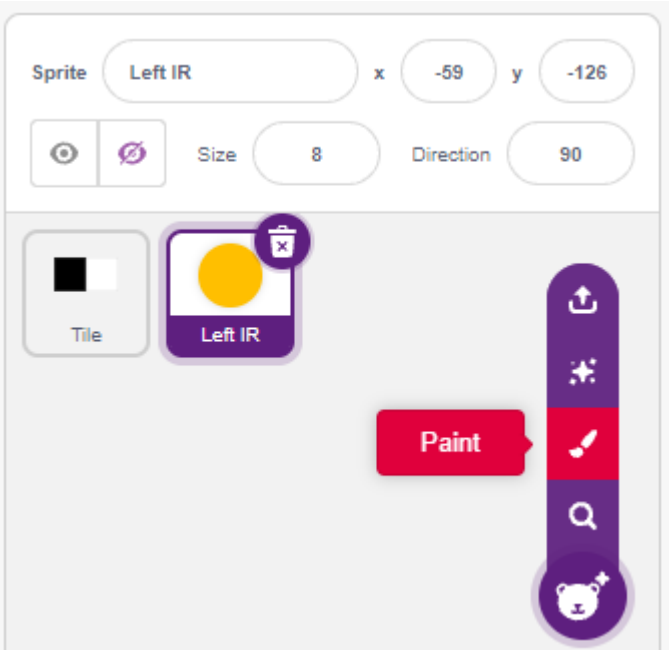
バックドロップで、2 つの IR モジュールの値を読み取り、対応するアクションを行います。

- 左の IR 障害物回避モジュールが手を感知した場合、メッセージ - **left** をブロードキャストします。
- 左の IR 回避モジュールが手を感知した場合、メッセージ - **right** をブロードキャストします。

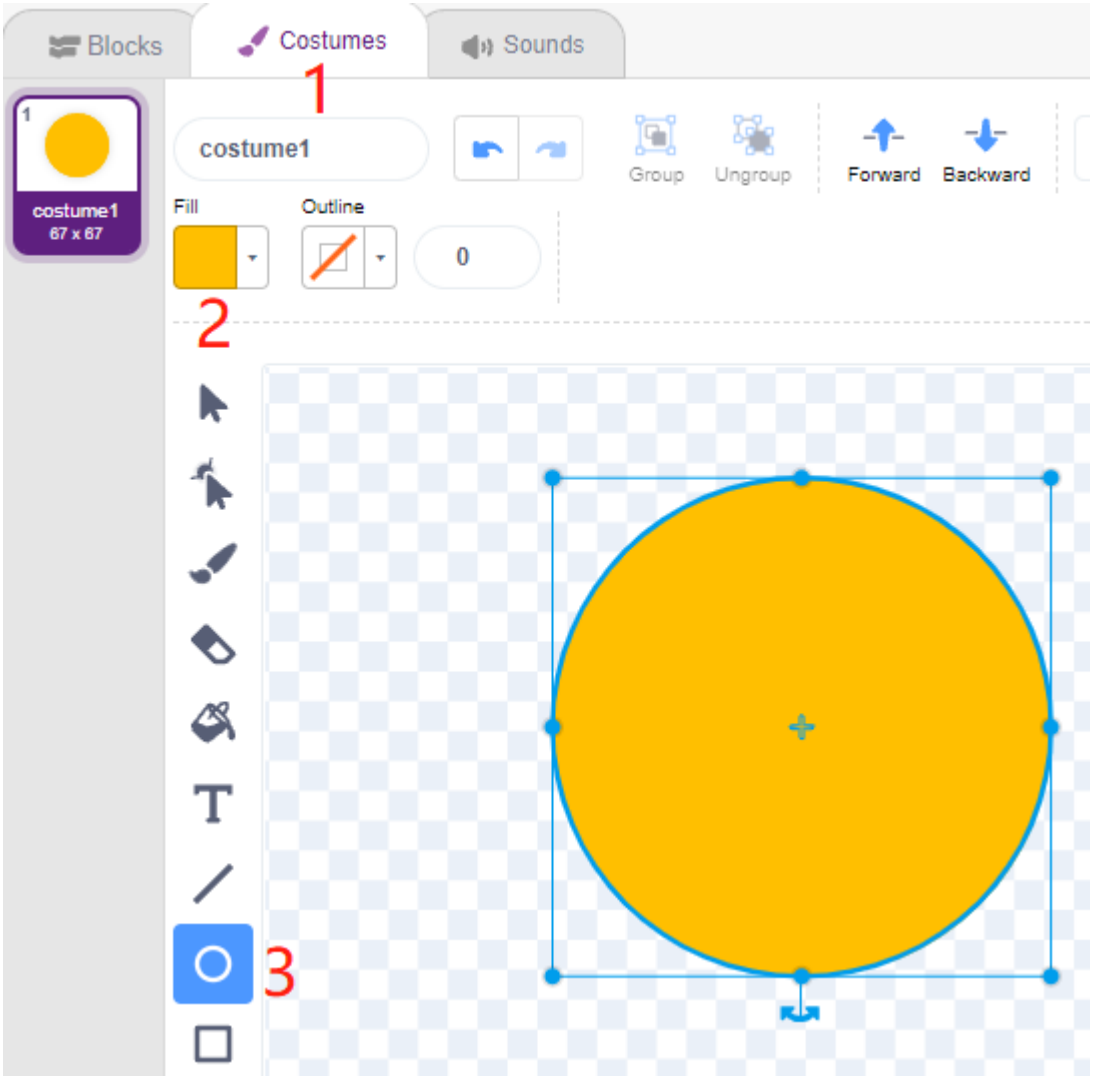


### 4. Left IR スプライト

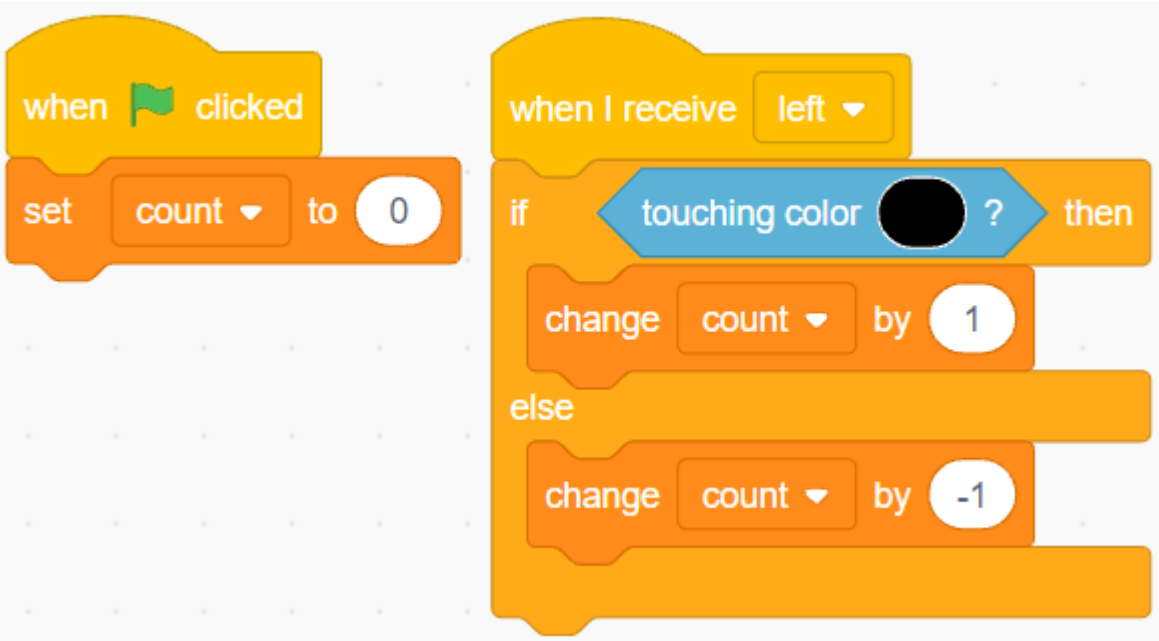
再度、**Add sprite** アイコンの上にマウスを置き、**Paint** を選択して **Left IR** という新しいスプライトを作成します。



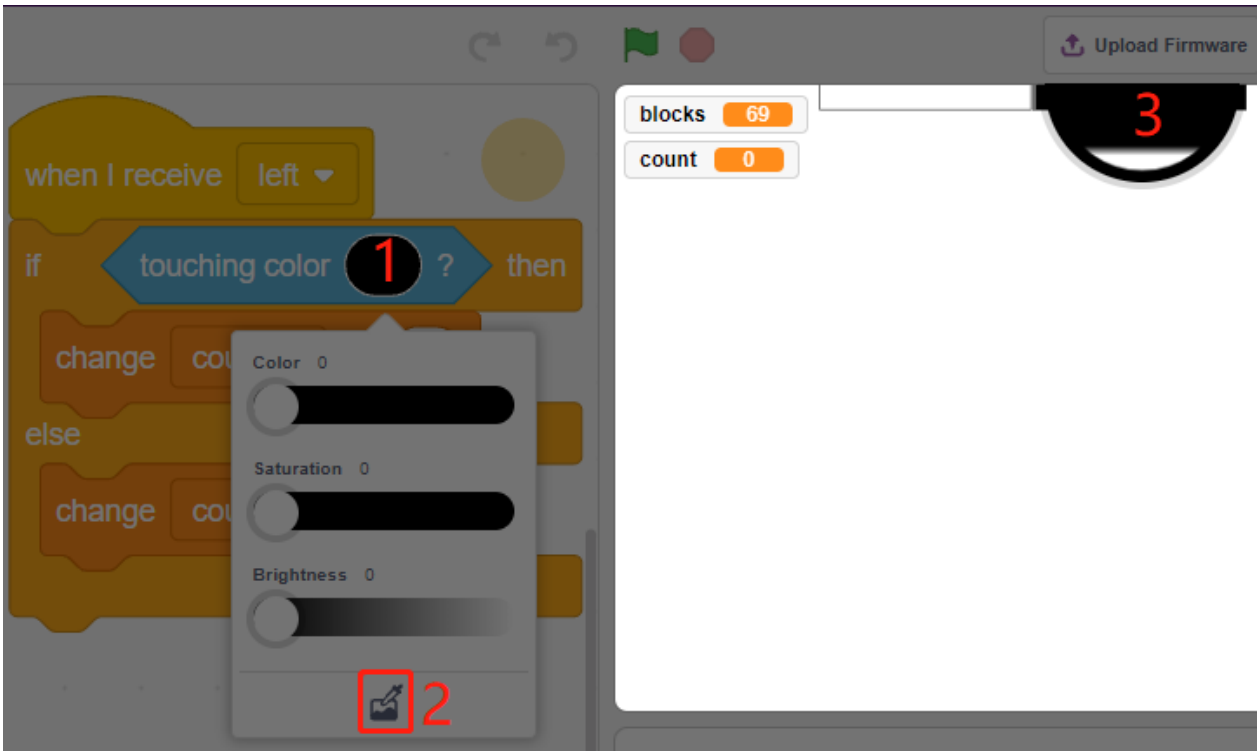

**Left IR** スプライトの **Costumes** ページに移動し、塗りつぶし色（黒と白を除く任意の色）を選択し、円を描きます。



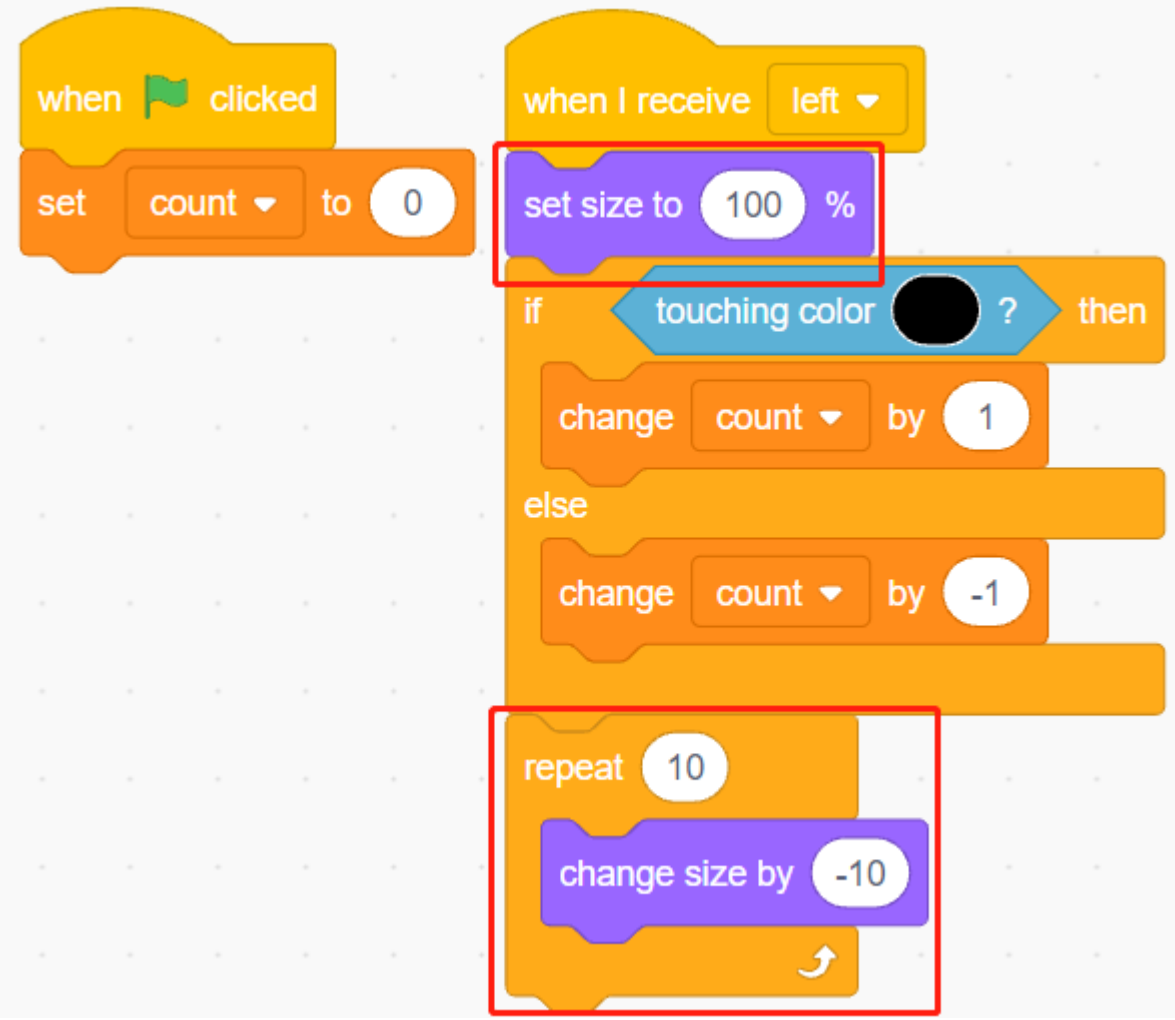

次に、 **Left IR** スプライトのスクリプトを開始します。メッセージ - **left** が受信されたとき（左の IR 受信モジュールが障害物を検出した場合）、 **Tile** スプライトの黒ブロックがタッチされているかどうかを判断し、タッチされていれば、変数 **count** を 1 増やし、そうでなければ 1 減らします。



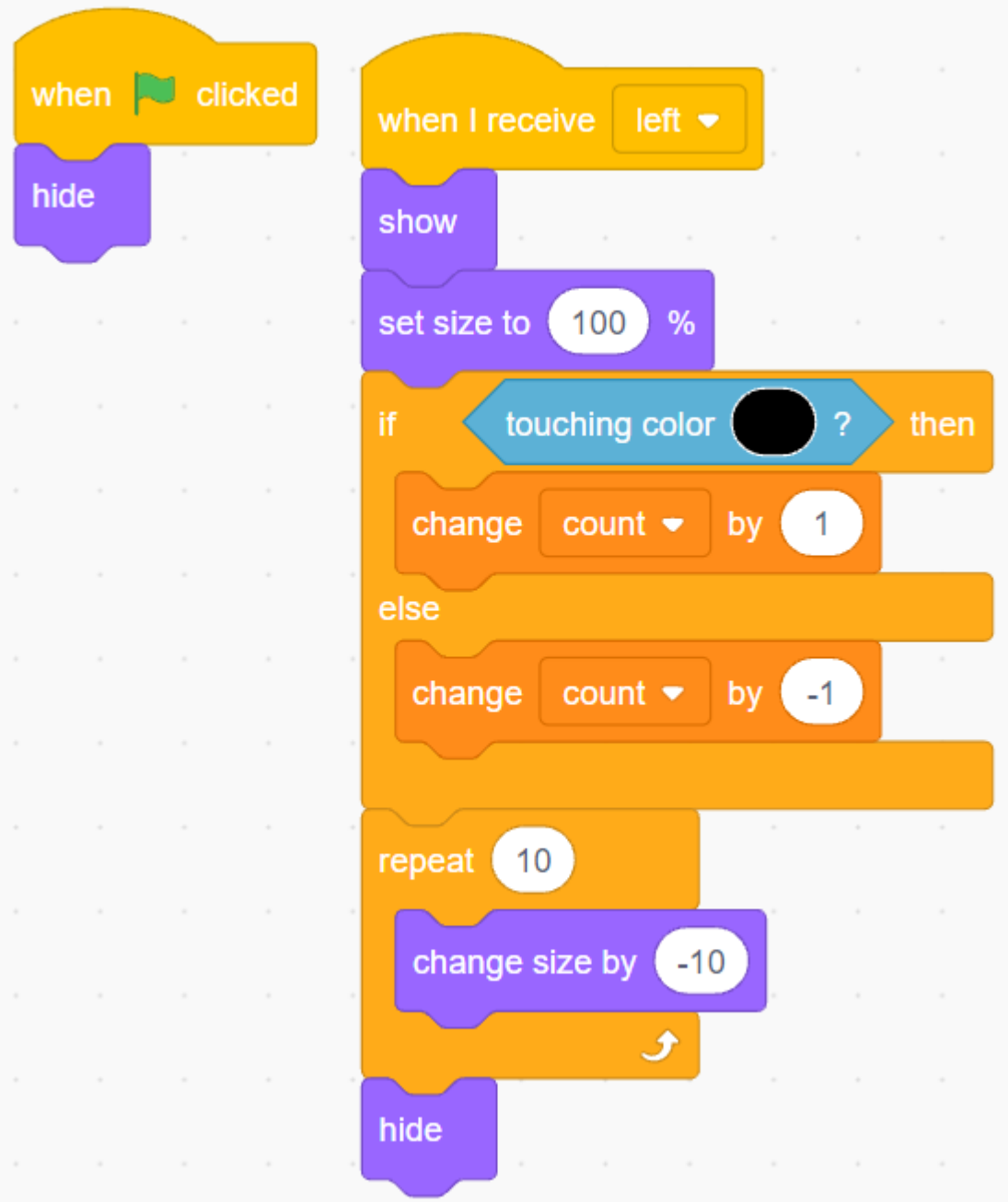

**注釈:** **Tile** スプライトをステージに表示させ、 **Tile** スプライトの黒ブロックの色を吸収する必要があります。




**Left IR** のセンシング効果（ズームイン/アウト）を実装しましょう。

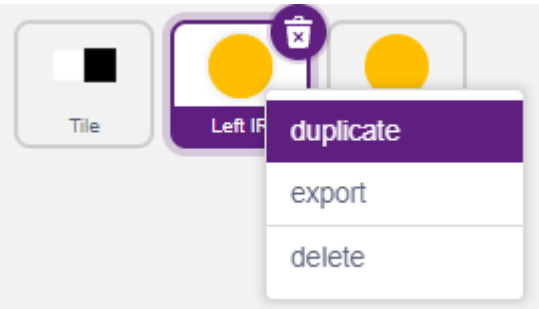



緑の旗がクリックされたときに **Left IR** スプライトを非表示にし、メッセージ - **left** が受信されたときに表示し、最終的に再び非表示にします。

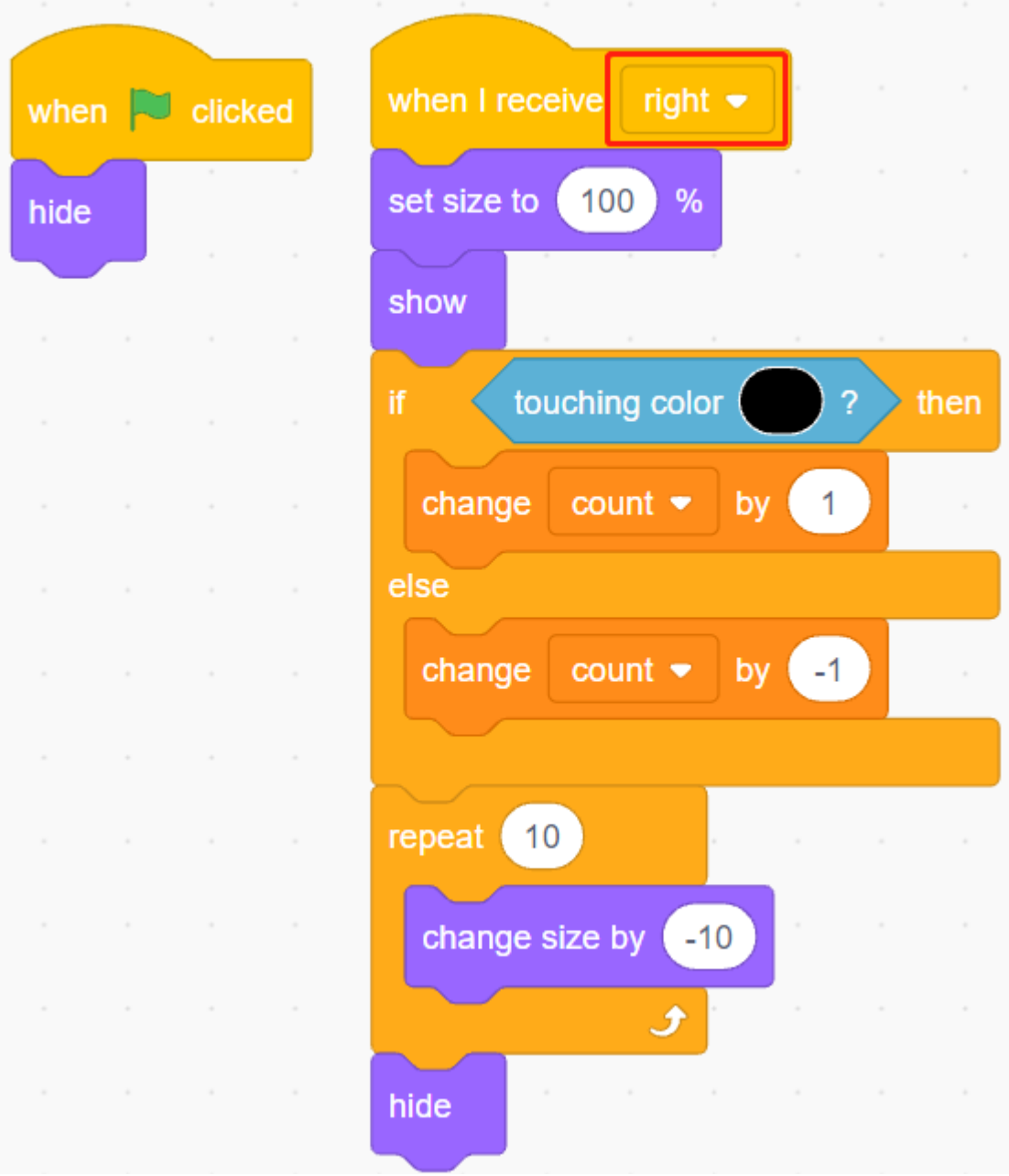



**5. Right IR スプライト**

**Left IR** スプライトをコピーして、 **Right IR** に名前を変更します。




次に、受信メッセージを - **right** に変更します。



すべてのスクリプト作成が完了したら、緑の旗をクリックしてスクリプトを実行できます。

## 7.24 2.21 GAME - 心を守れ

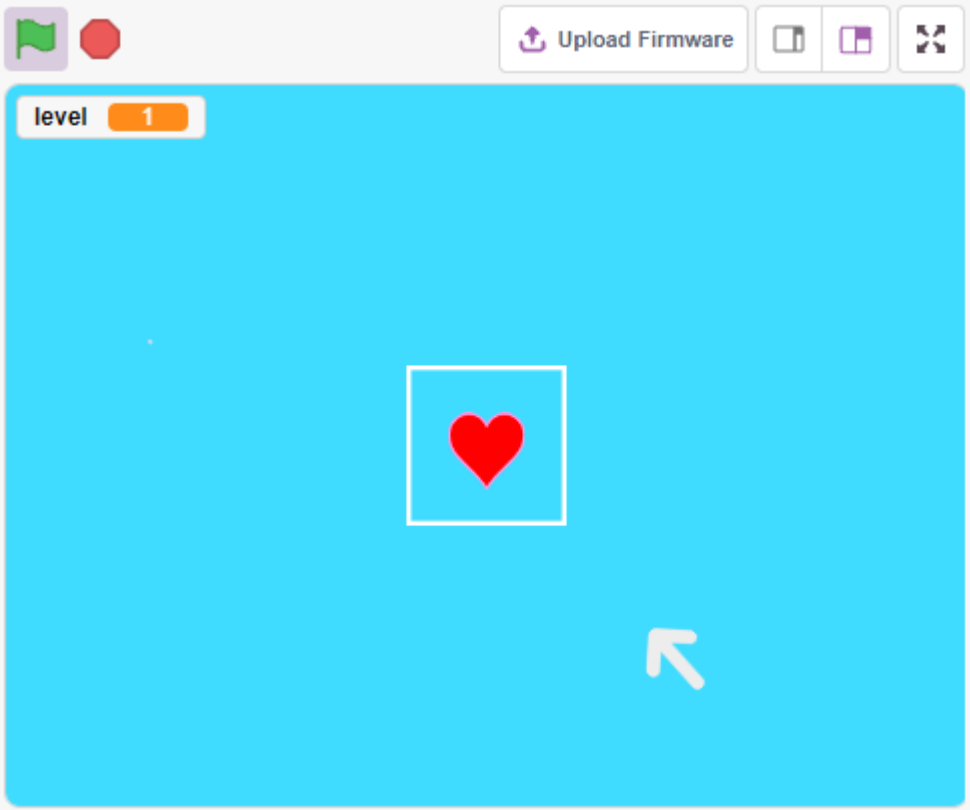
このプロジェクトでは、反応速度を試すゲームを作成しましょう。

舞台には矩形の箱に守られた心があり、舞台の任意の位置からこの心に向かって矢が飛んできます。矢の色はランダムに黒と白の間で変わり、矢はますます速く飛びます。

矩形の箱の色と矢の色が同じ場合、矢は外部にブロックされ、レベルが 1 追加されます。両方の色が同じでない場合、矢は心を撃ち抜き、ゲームは終了します。

ここでは、矩形のボックスの色は Line Tracking モジュールによって制御されます。モジュールが黒い表面（反射する表面）に置かれたとき、矩形のボックスの色は黒であり、それ以外の場合は白です。

従って、矢の色に応じて、Line Tracking モジュールを白い表面または黒い表面に置くかどうかを決定する必要があります。



### 7.24.1 必要な部品

このプロジェクトには、以下のコンポーネントが必要です。

キット全体を購入すると確かに便利です。リンクはこちらです：

| 名前 | このキットのアイテム | リンク |
| --- | --- | --- |
| 3 in 1 Starter Kit | 380+ | |

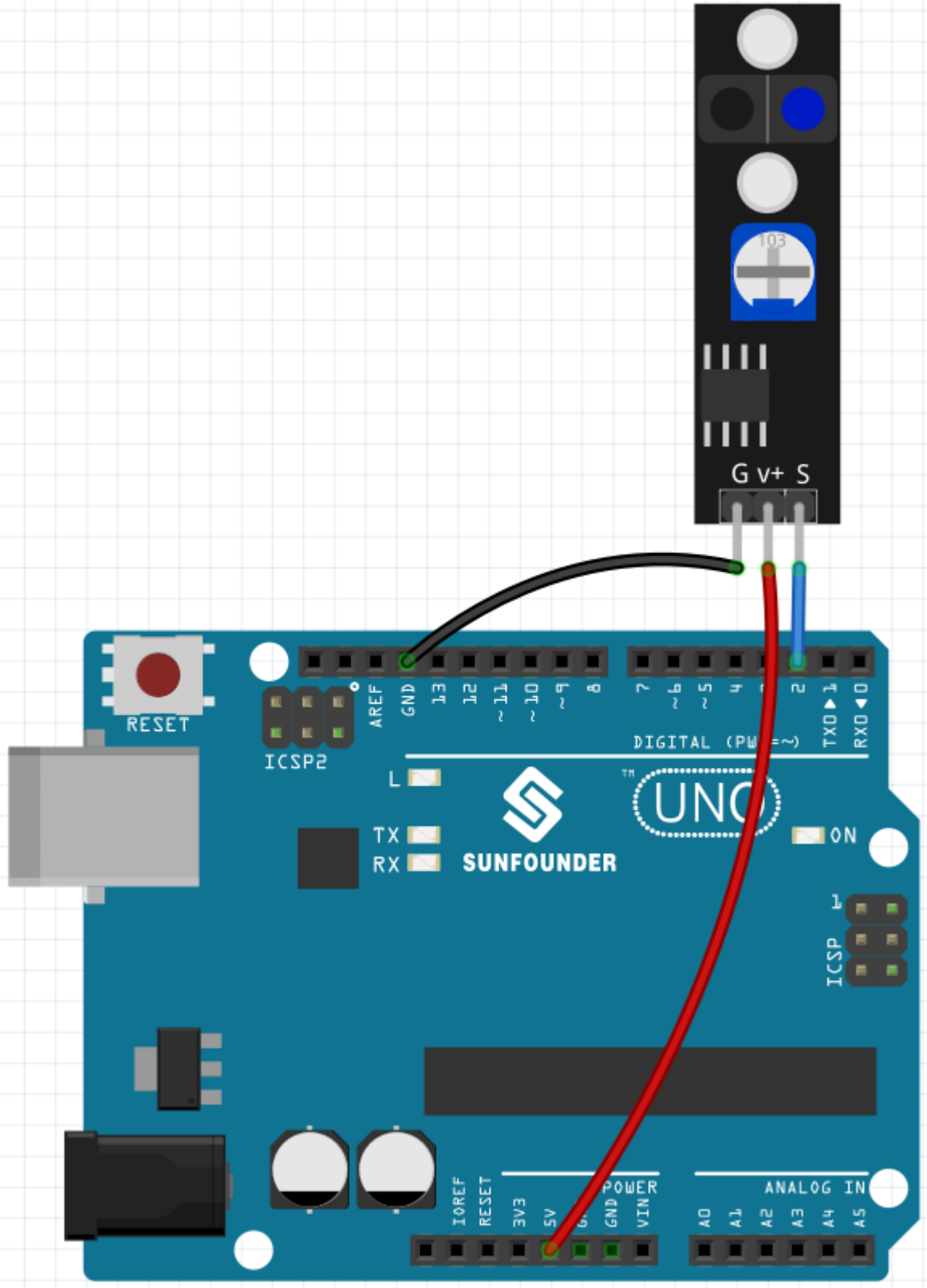
以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* **ボード** | |
| **ジャンパーワイヤー** | |
| **ライン追跡モジュール** | |

### 7.24.2 回路の作成

これはデジタルの Line Tracking モジュールであり、黒い線が検出されると 1 を出力し、白い線が検出されると 0 の値を出力します。さらに、モジュール上のポテンショメータを通じて感知距離を調整することができます。

次の図に従って回路を組み立ててください。




**注釈:** プロジェクトを開始する前に、モジュールの感度を調整する必要があります。

上記の図に従って配線し、R3 ボードに電源を供給します（USB ケーブルを直接挿入するか、9V の電池ボタンケーブルを使用）。コードをアップロードせずに。

机の上に黒い電気テープを貼り、Line Track モジュールを机から 2cm の高さに置きます。

センサーを下に向け、モジュールの信号 LED が白いテーブルで点灯し、黒いテープで消灯することを確認します。

そうでない場合、モジュール上のポテンショメータを調整して、上記の効果を実現できるようにします。

### 7.24.3 プログラミング

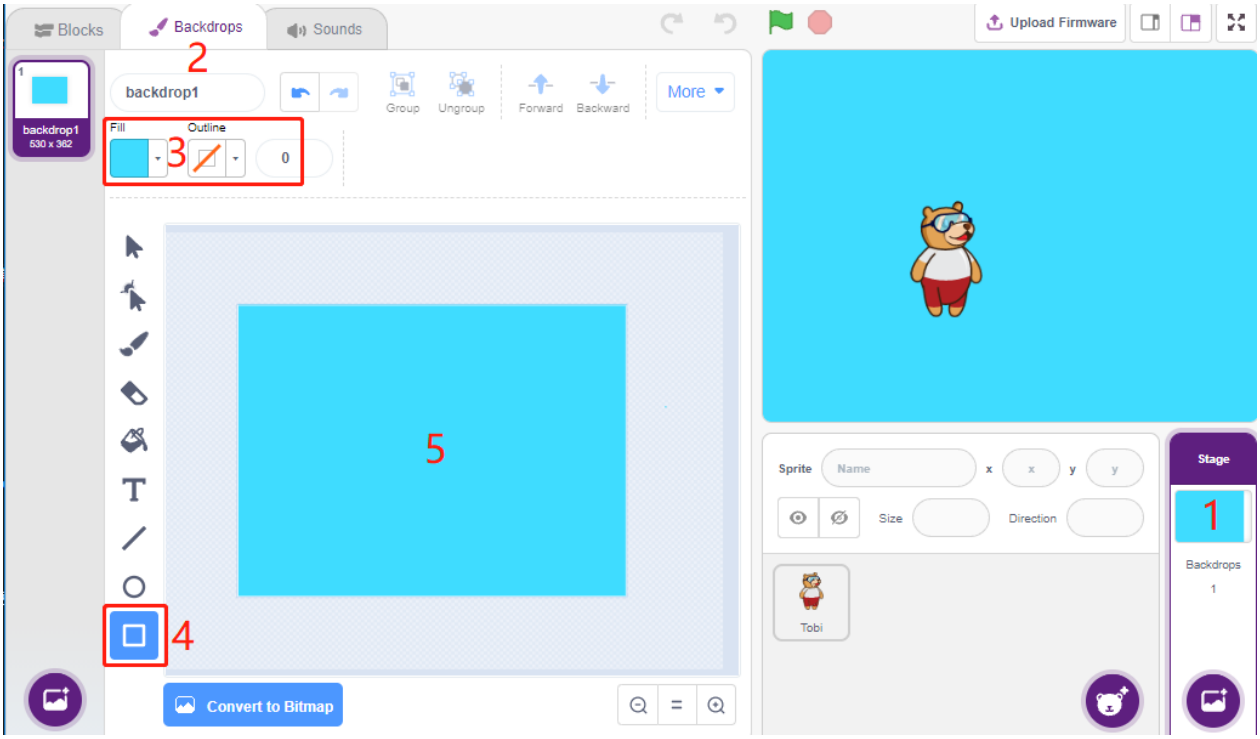
ここでは、 **Heart** 、 **Square Box** 、 **Arrow1** の 3 つのスプライトを作成する必要があります。

- **Heart**：舞台の真ん中で止まっていて、 **Arrow1** スプライトに触れるとゲームが終了します。
- **Square Box**：コスチュームが 2 種類あり、黒と白で、Line Tracking モジュールの値に応じてコスチュームを切り替えます。
- **Arrow**：任意の位置から舞台の中央に向かって黒/白で飛びます。その色が **Square Box** スプライトの色と一致する場合、それはブロックされ、ランダムな位置から舞台の中央に再び飛ぶ。その色が **Square Box** スプライトの色と一致しない場合、 **Heart** スプライトを通過し、ゲームが終了します。

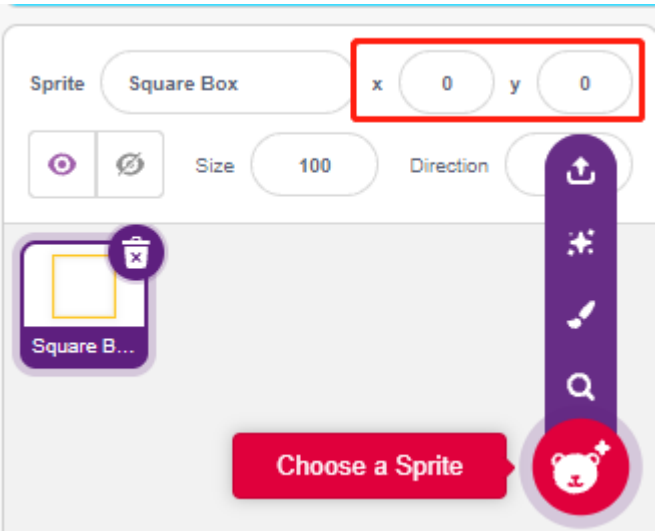
**1. Square Box スプライトを追加する**

Arrow1 と Square Box スプライトの両方が白いコスチュームを持っているため、それらを舞台に表示するために、今、背景を黒、白、赤を除く任意の色で塗りつぶします。

- **Backdrop1** をクリックして、その **Backdrops** ページに移動します。
- 塗りつぶす色を選択します。
- 描画ボードと同じサイズの矩形を描画するために **Rectangle** ツールを使用します。

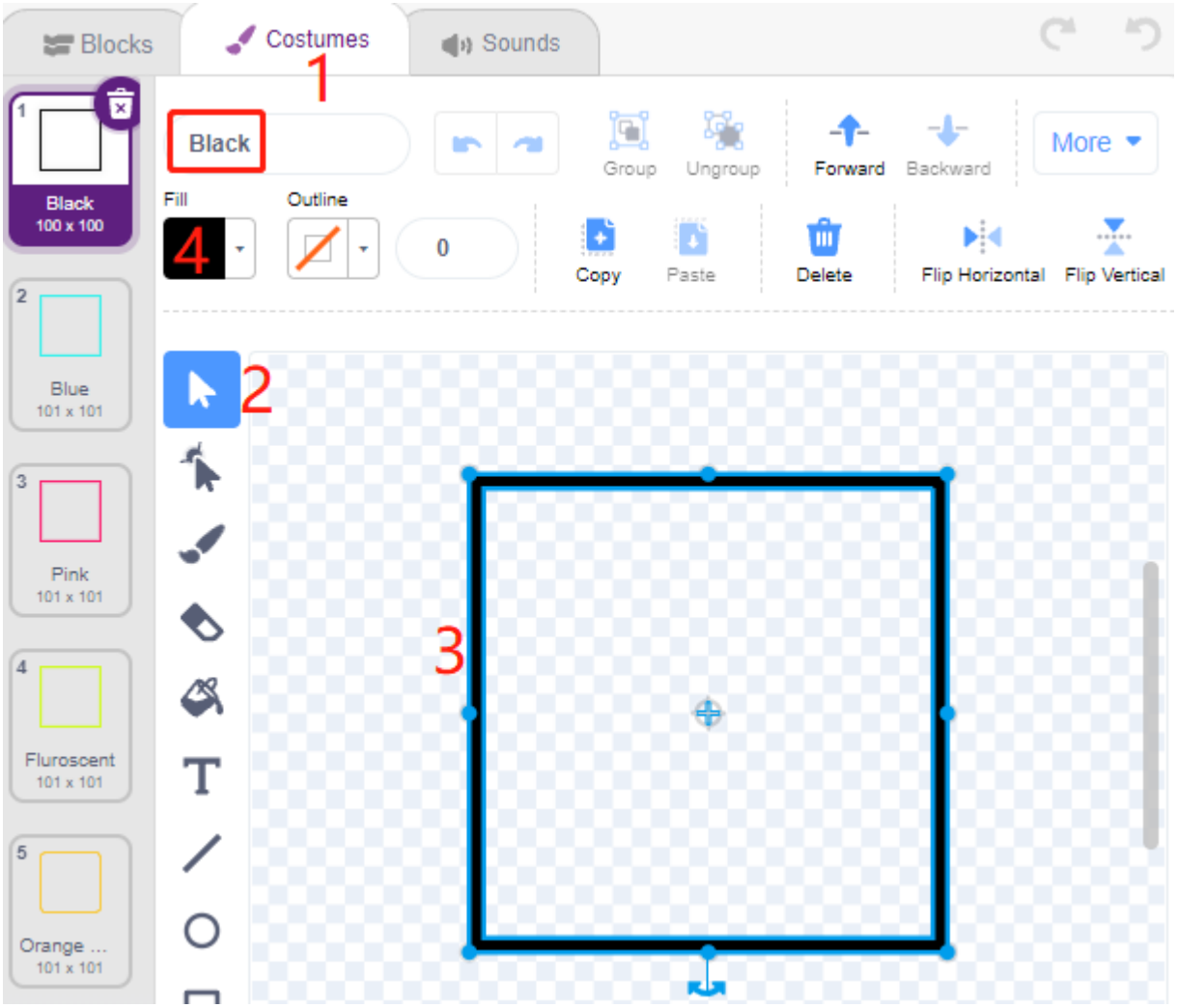



デフォルトのスプライトを削除し、 **Choose a Sprite** ボタンを使用して **Square Box** スプライトを追加し、その x と y を (0, 0) に設定します。

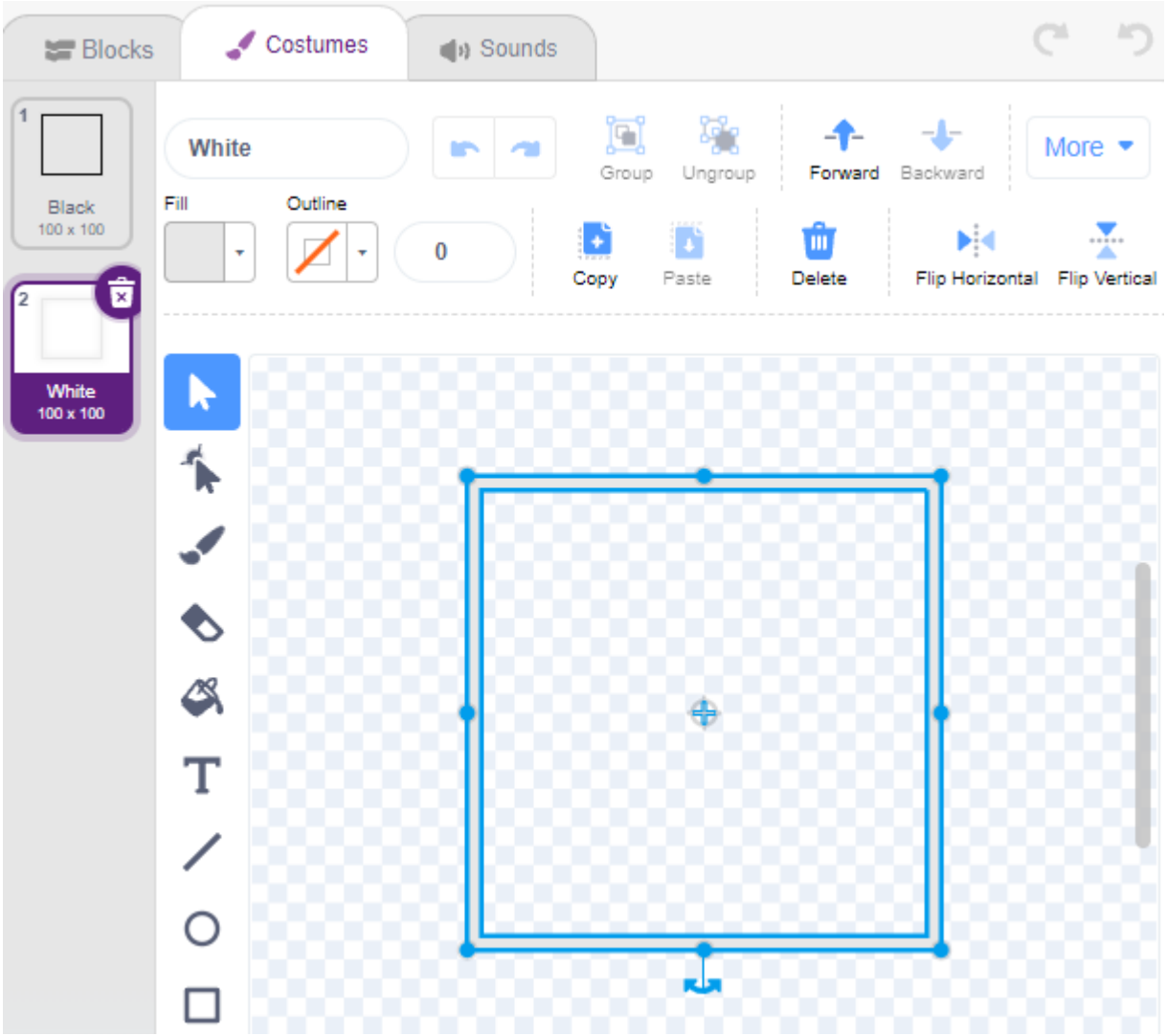



**Square Box** スプライトの **Costumes** ページに移動して、黒と白のコスチュームを設定します。

- 選択ツールをクリックします。
- キャンバス上の矩形を選択します。
- 塗りつぶしの色を黒に設定します。
- そして、コスチュームの名前を **Black** とします。

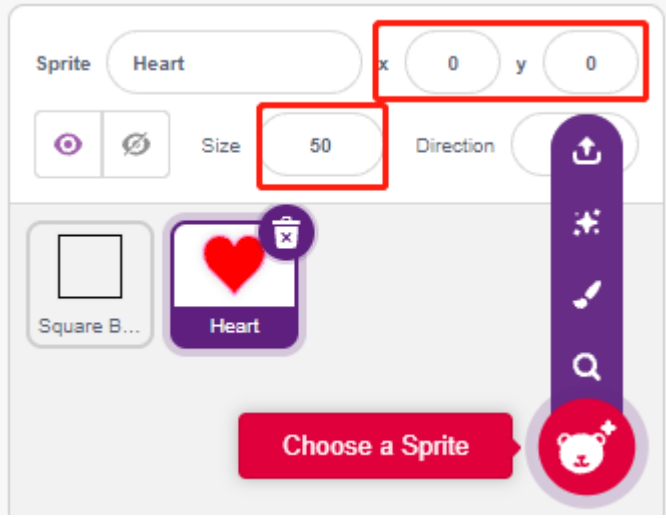



2 番目のコスチュームを選択し、塗りつぶしの色を白に設定し、その名前を White に設定し、残りのコスチュームを削除します。

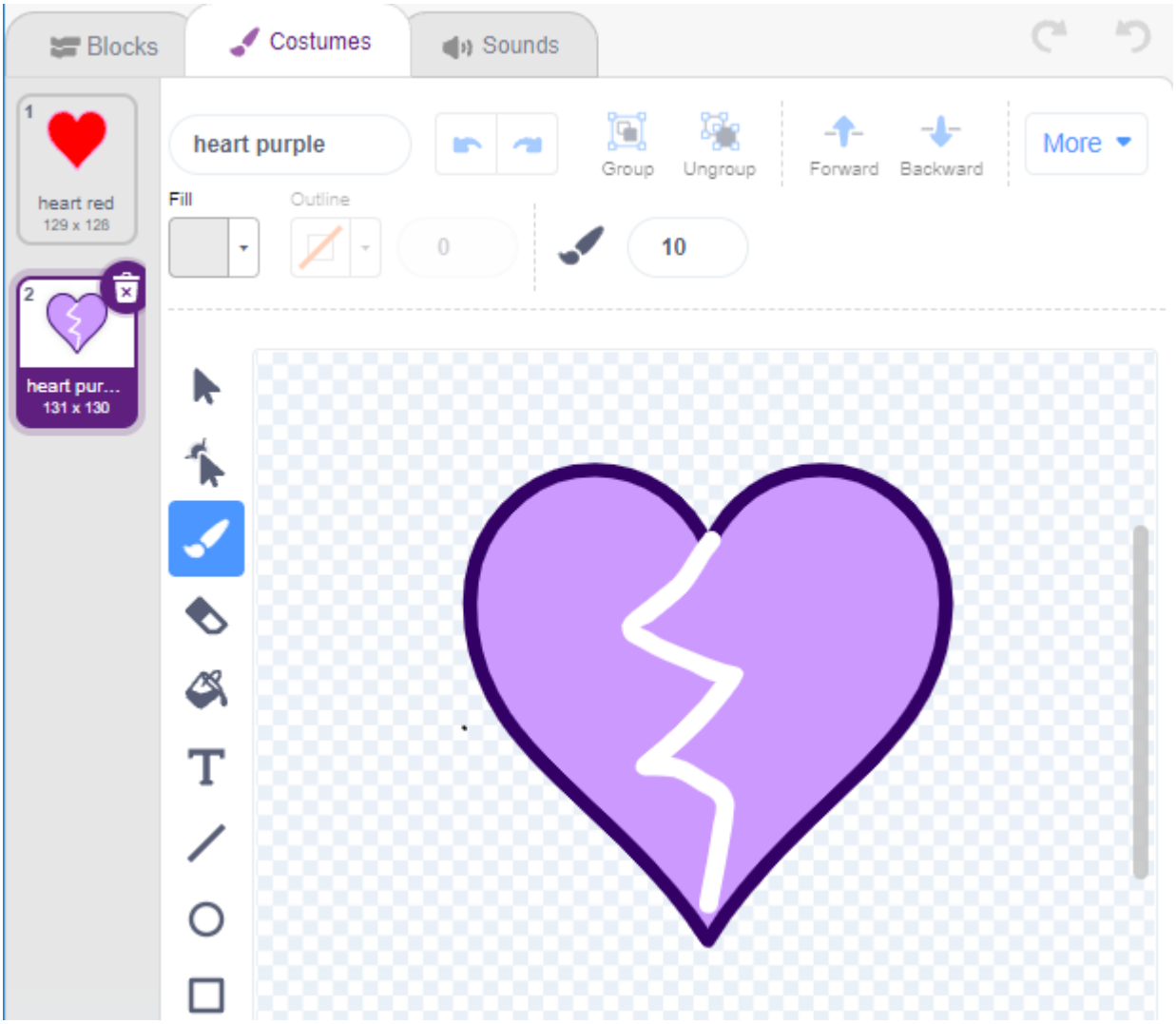



**2. Heart スプライトの追加**

**Heart** スプライトを追加し、その位置を (0, 0) に設定し、Square Box の中に位置しているようにサイズを縮小します。

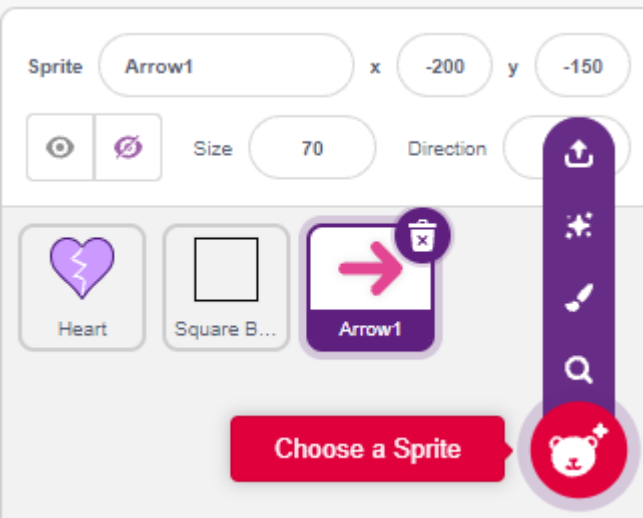



**Costumes** ページで、紫色のハートのコスチュームを破損しているように調整します。

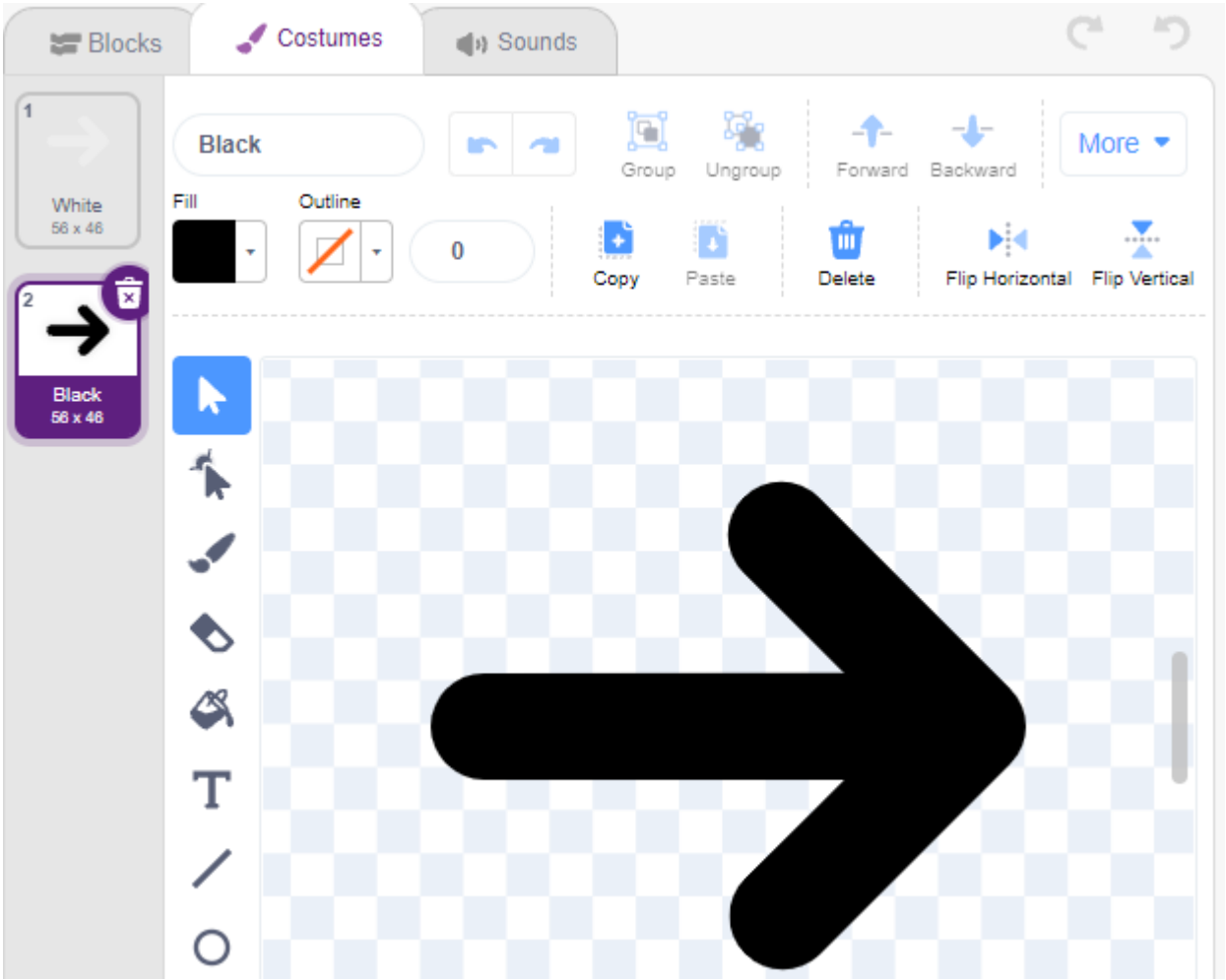


### 3. Arrow1 スプライトの追加

**Arrow1** スプライトを追加します。



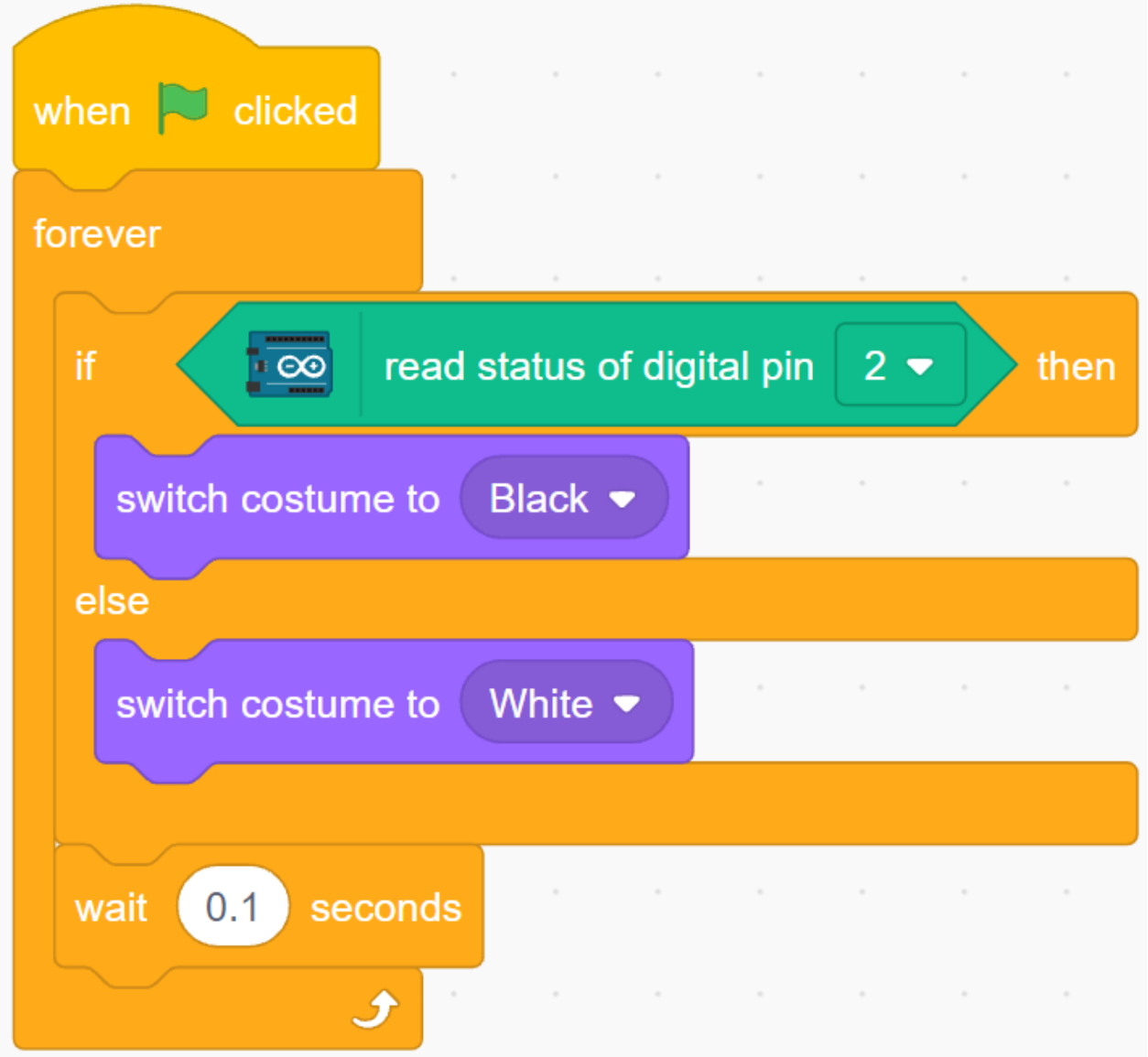

**Costumes** ページで、右向きのコスチュームを保持・複製し、その色を黒と白に設定します。



#### 4. Square Box スプライトのスクリプト作成

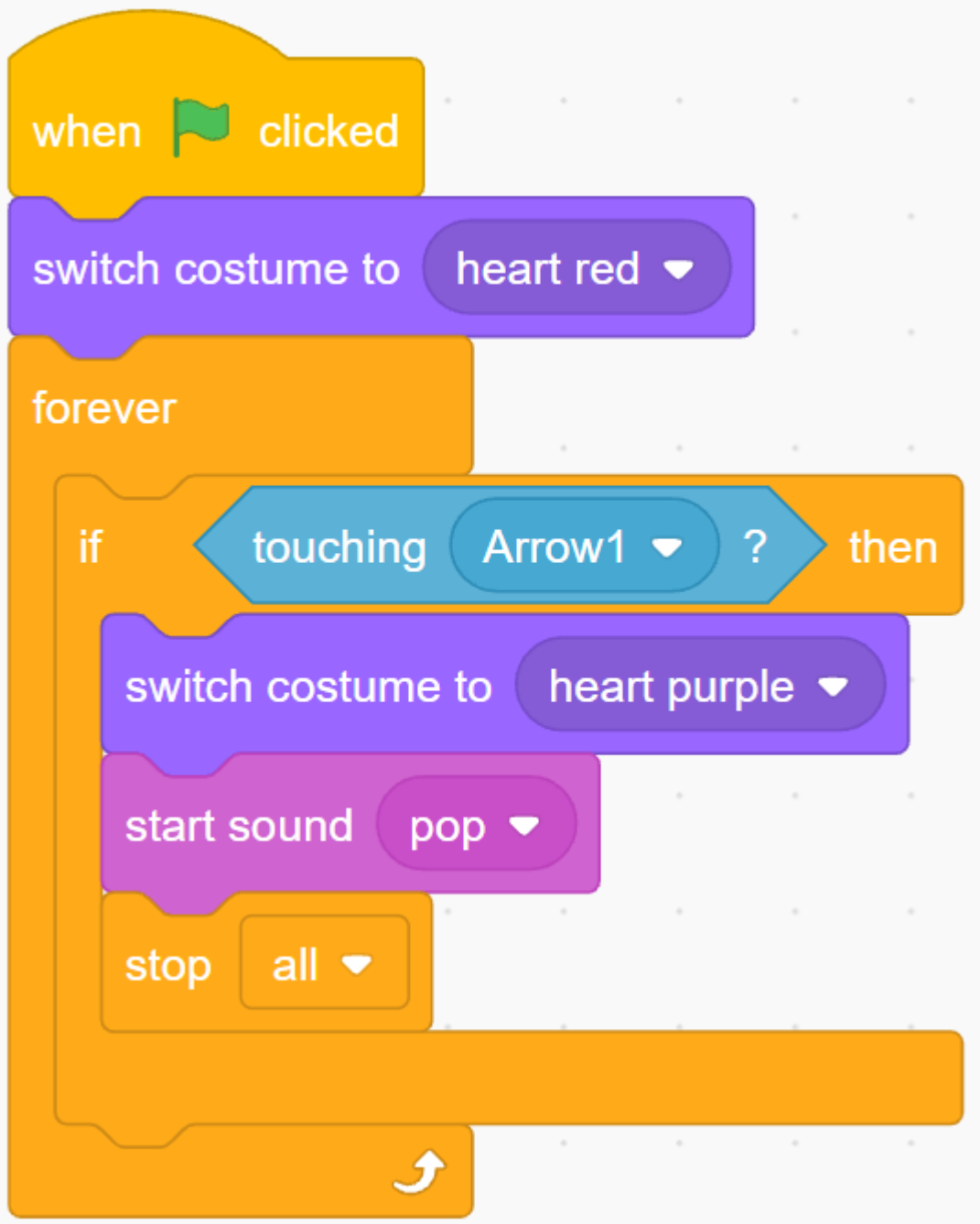
**Blocks** ページに戻り、 **Square Box** スプライトのスクリプトを作成します。

- デジタルピン 2（ラインフォローモジュール）の値が 1（黒い線が検出された場合）の場合、コスチュームを **Black** に切り替えます。
- それ以外の場合は、コスチュームを **White** に切り替えます。




#### 5. Heart スプライトのスクリプト作成

**Heart** スプライトは **Square Box** 内に保護されており、デフォルトでは赤いコスチュームです。Arrow1 スプライトが触れた場合、ゲームは終了します。

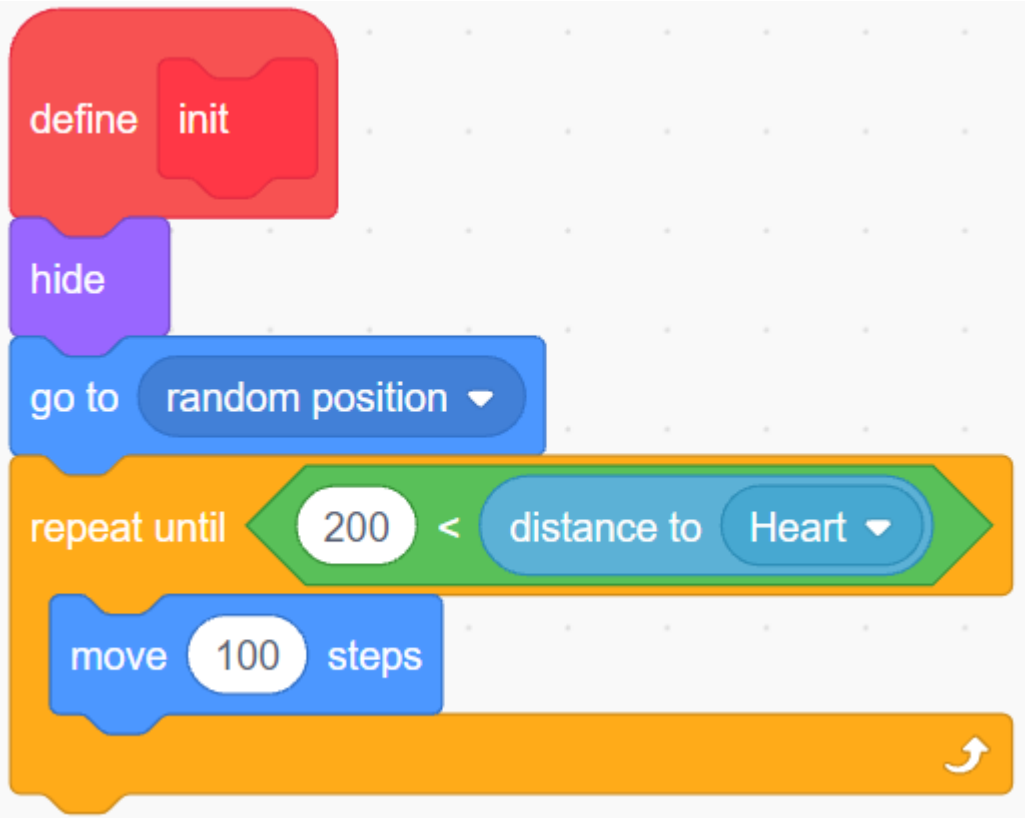



**6. Arrow1 スプライトのスクリプト作成**

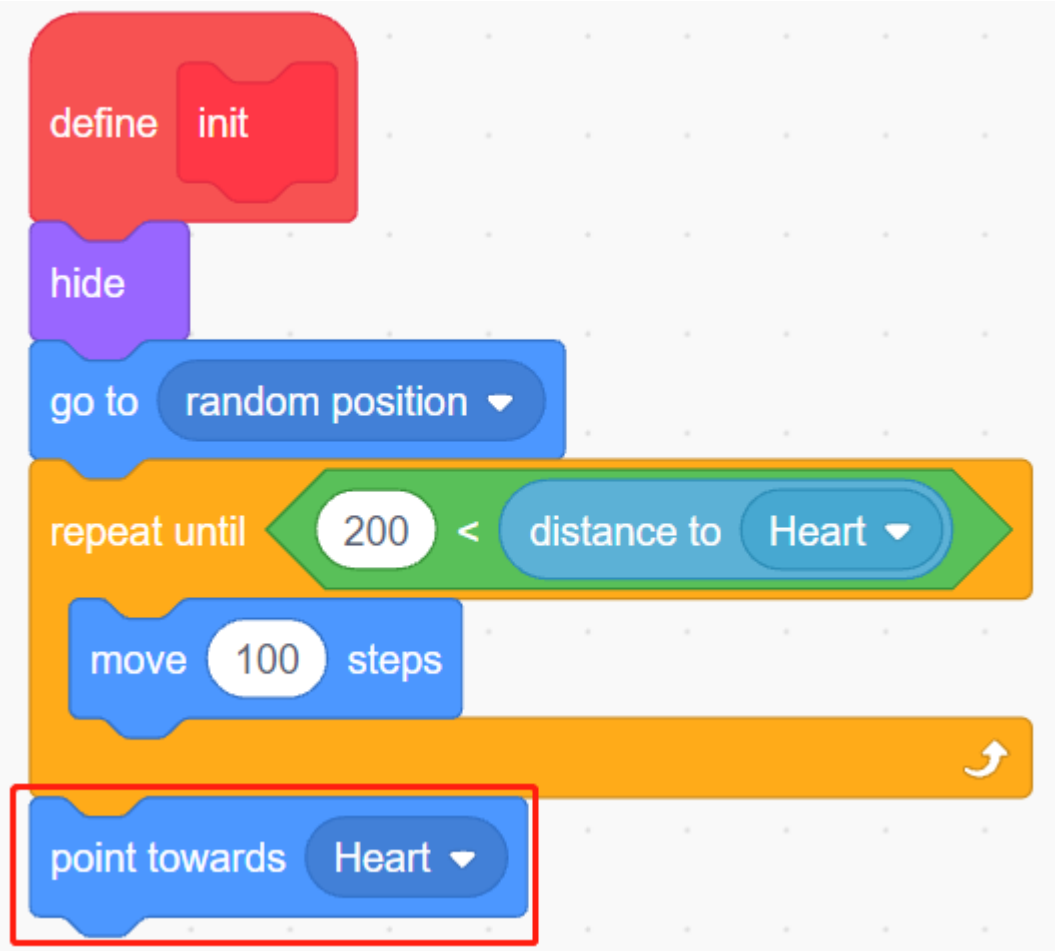
緑のフラグがクリックされたとき、 **Arrow1** スプライトを非表示にし、クローンを作成します。


[init] ブロックを作成して、 **Arrow1** スプライトの位置、向き、色を初期化します。

ランダムな位置で現れ、それと **Heart** スプライトとの距離が 200 未満の場合、距離が 200 以上になるまで外向きに移動します。

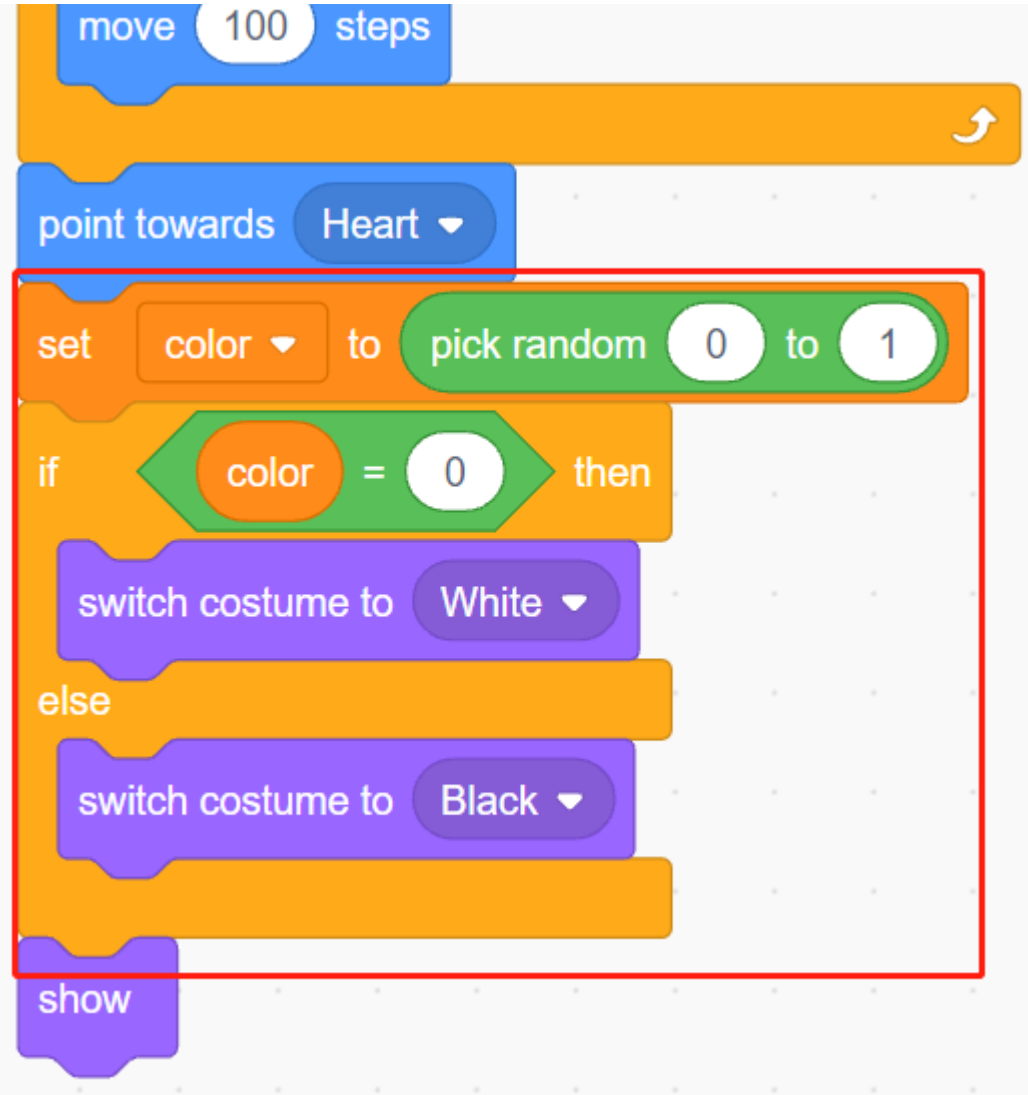



**Heart** スプライトの方向に向けてその方向を設定します。

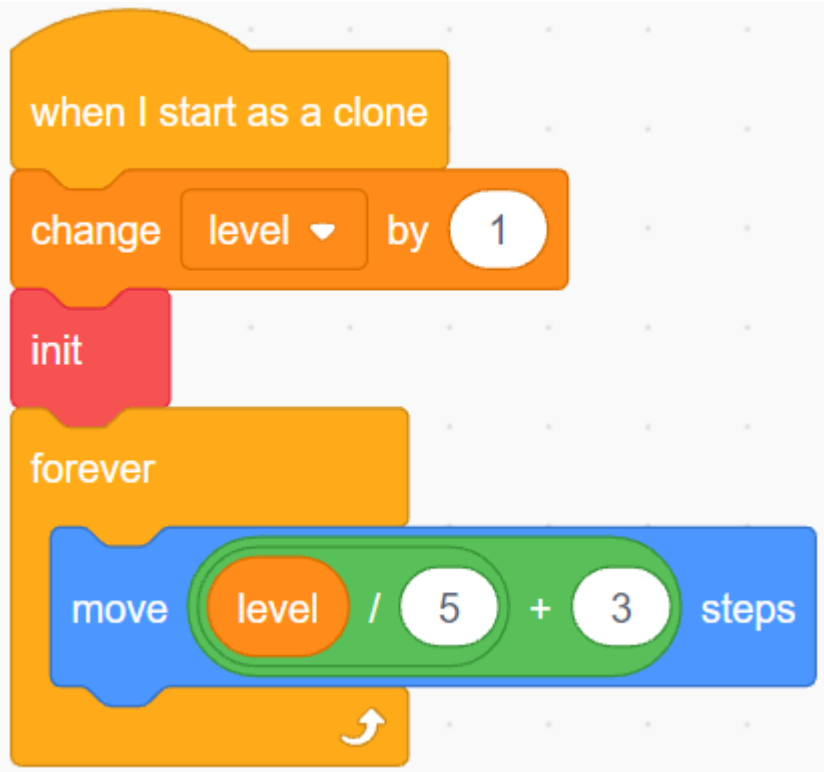



色を黒/白の間でランダムに切り替えます。

- 変数の色が 0 の場合、コスチュームを **White** に切り替えます。
- 変数の色が 1 の場合、コスチュームを **Black** に切り替えます。

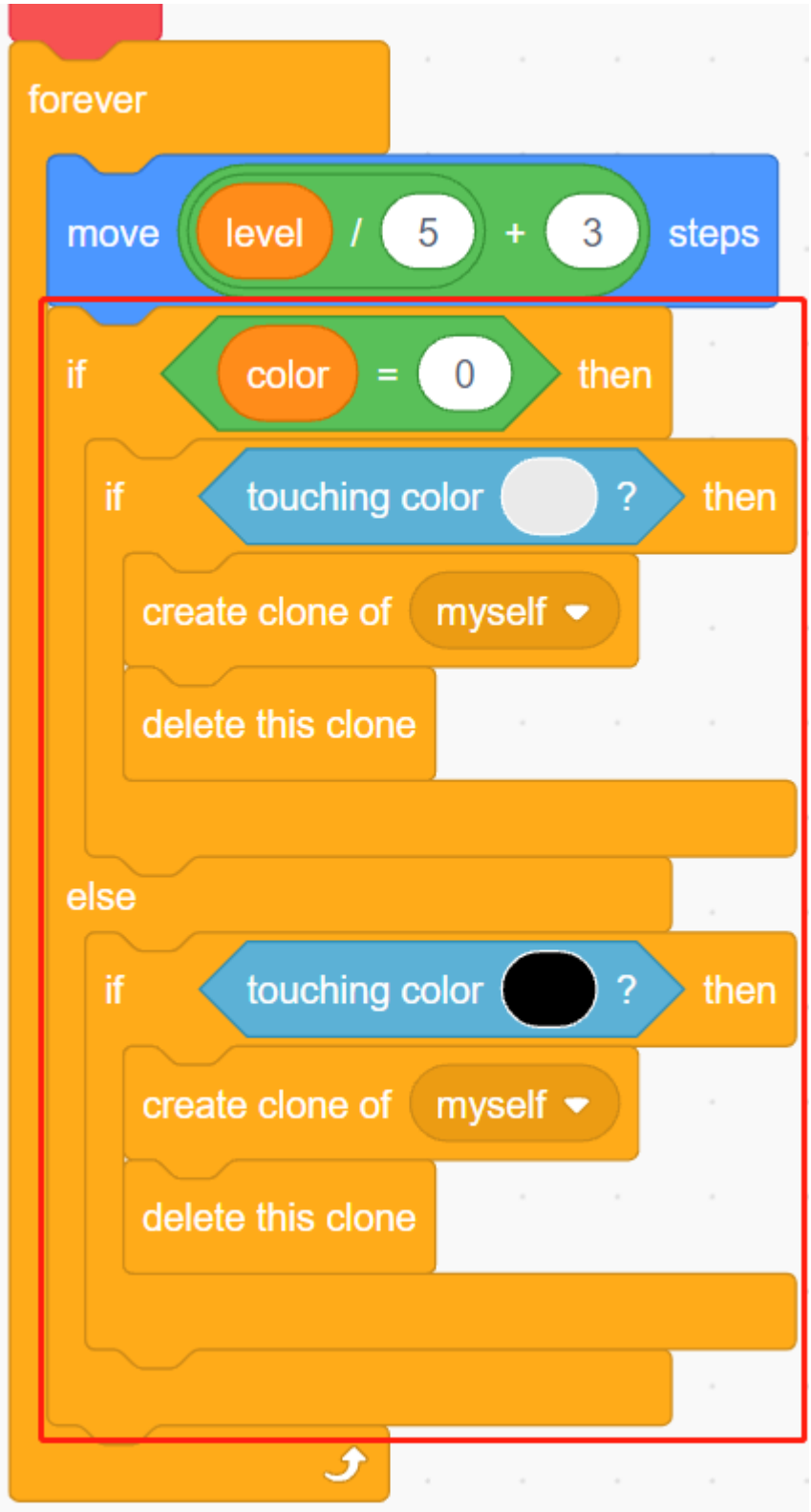



これで、移動を開始します。変数 **level** の値が増加すると、移動速度が速くなります。

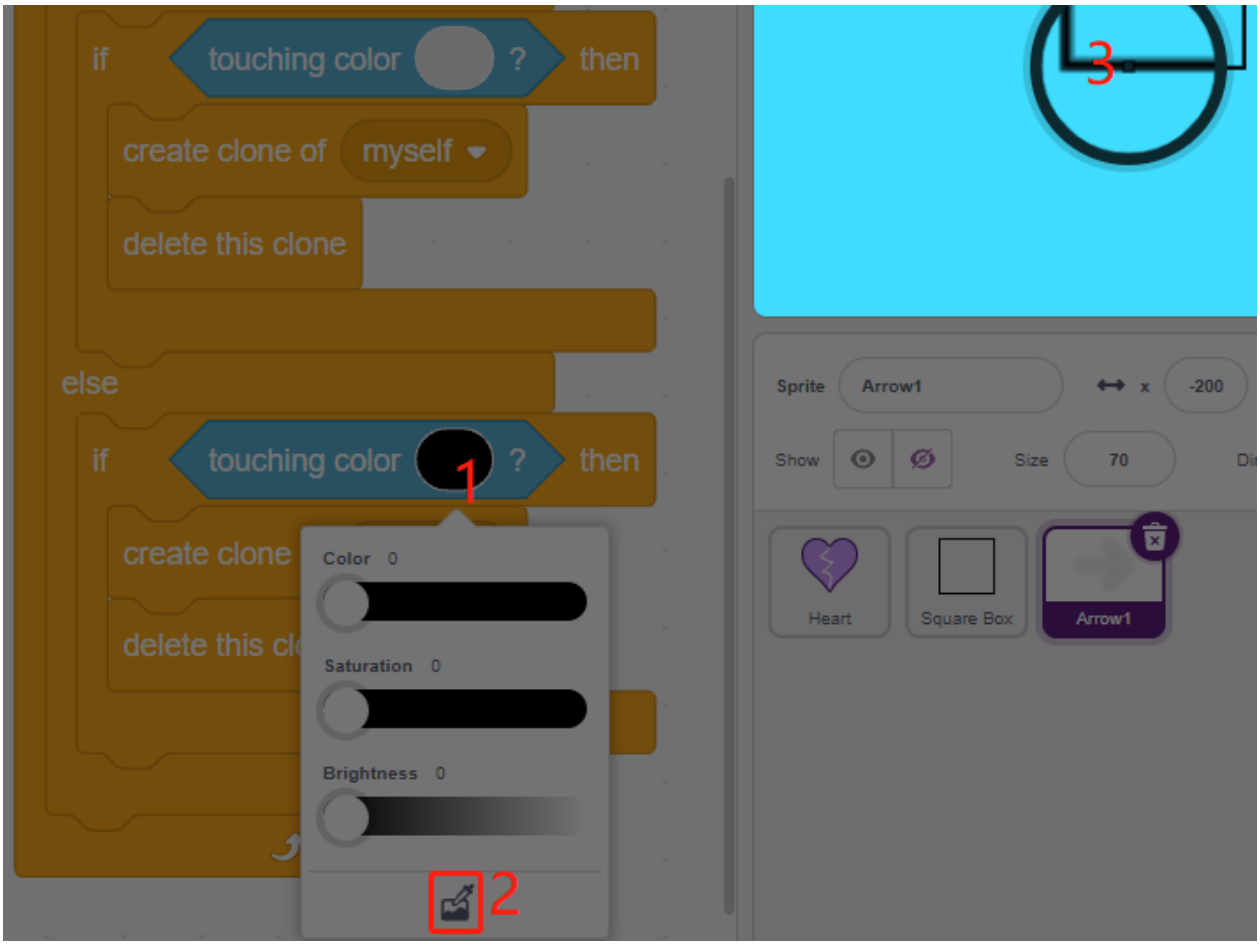



**Square Box** スプライトとの衝突効果を設定します。

- **Arrow1** スプライトと **Square Box** スプライトの色が同じ（Line Track モジュールの値に応じて変更される）場合、新しいクローンが作成され、ゲームは続行されます。
- 彼らの色が一致しない場合、 **Arrow1** スプライトは移動を続け、 **Heart** スプライトに当たるとゲームが終了します。



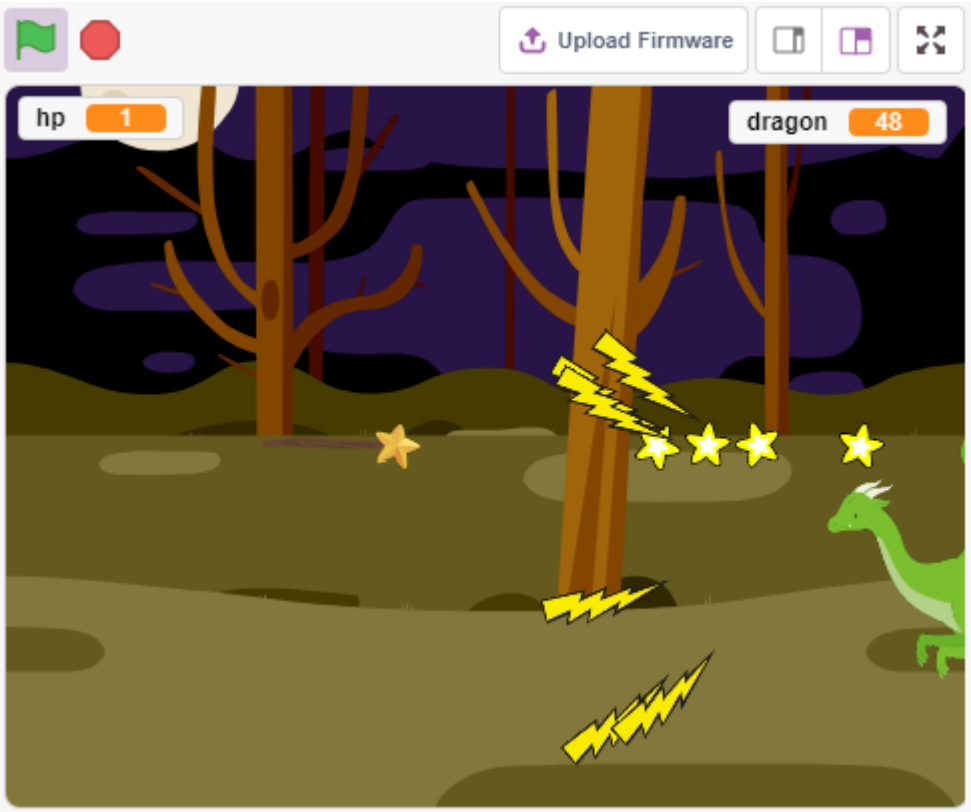

**注釈:** 2 つの [touch color()] ブロックは、Square Box の黒/白のコスチュームをそれぞれ別々に取得する必要があります。




## 7.25 2.22 GAME - ドラゴン討伐

このゲームでは、ジョイスティックを使ってドラゴンを討伐します。

緑色をクリックすると、ドラゴンは右側で上下に浮かび、時々火を吹きます。魔法の杖の動きをジョイスティックで制御し、ドラゴンに向かって星の攻撃を打ち出しつつ、ドラゴンが放つ炎を避けて、最終的に倒す必要があります。




### 7.25.1 必要な部品

このプロジェクトには以下の部品が必要です。

全セットを購入することは非常に便利です。リンクは以下の通りです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| ジャンパーワイヤー | |
| ジョイスティックモジュール | - |

### 7.25.2 回路の作成

ジョイスティックは、ベース上で旋回するスティックを持つ入力デバイスで、制御するデバイスにその角度や方向を報告します。ジョイスティックは、ビデオゲームやロボットの制御によく使用されます。

コンピュータに完全な動きを伝えるために、ジョイスティックはスティックの位置を 2 つの軸で測定する必要があります – X 軸（左から右）と Y 軸（上から下）です。

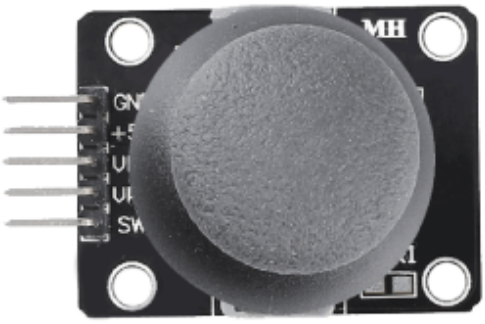
ジョイスティックの動きの座標は以下の図に示されています。

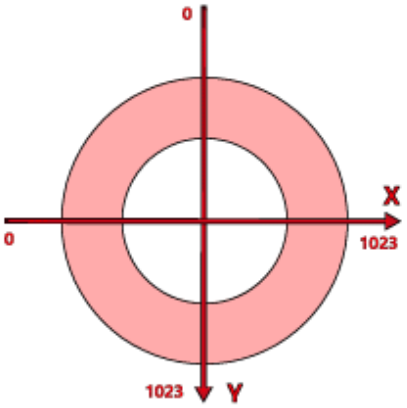
---

**注釈:**

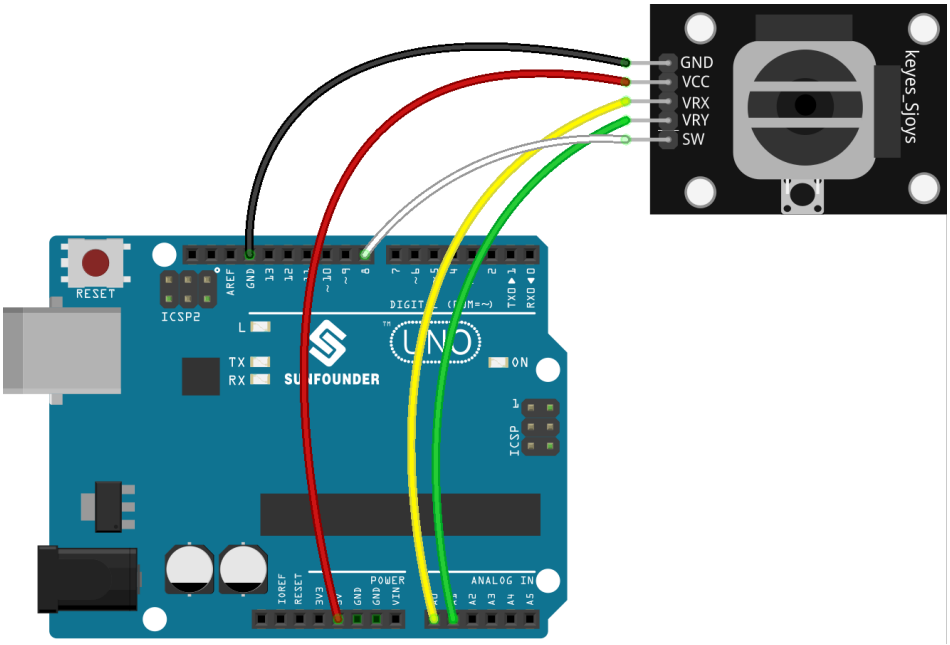
- x 座標は左から右へ、範囲は 0-1023。
- y 座標は上から下へ、範囲は 0-1023。

---




以下の図に従って回路を組み立ててください。

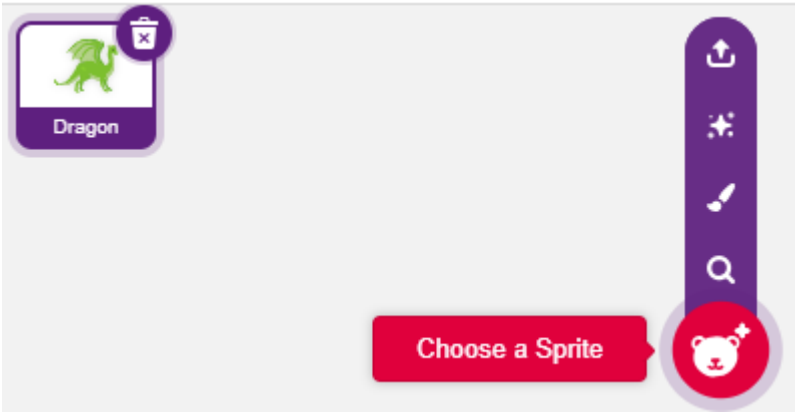



### 7.25.3 プログラミング

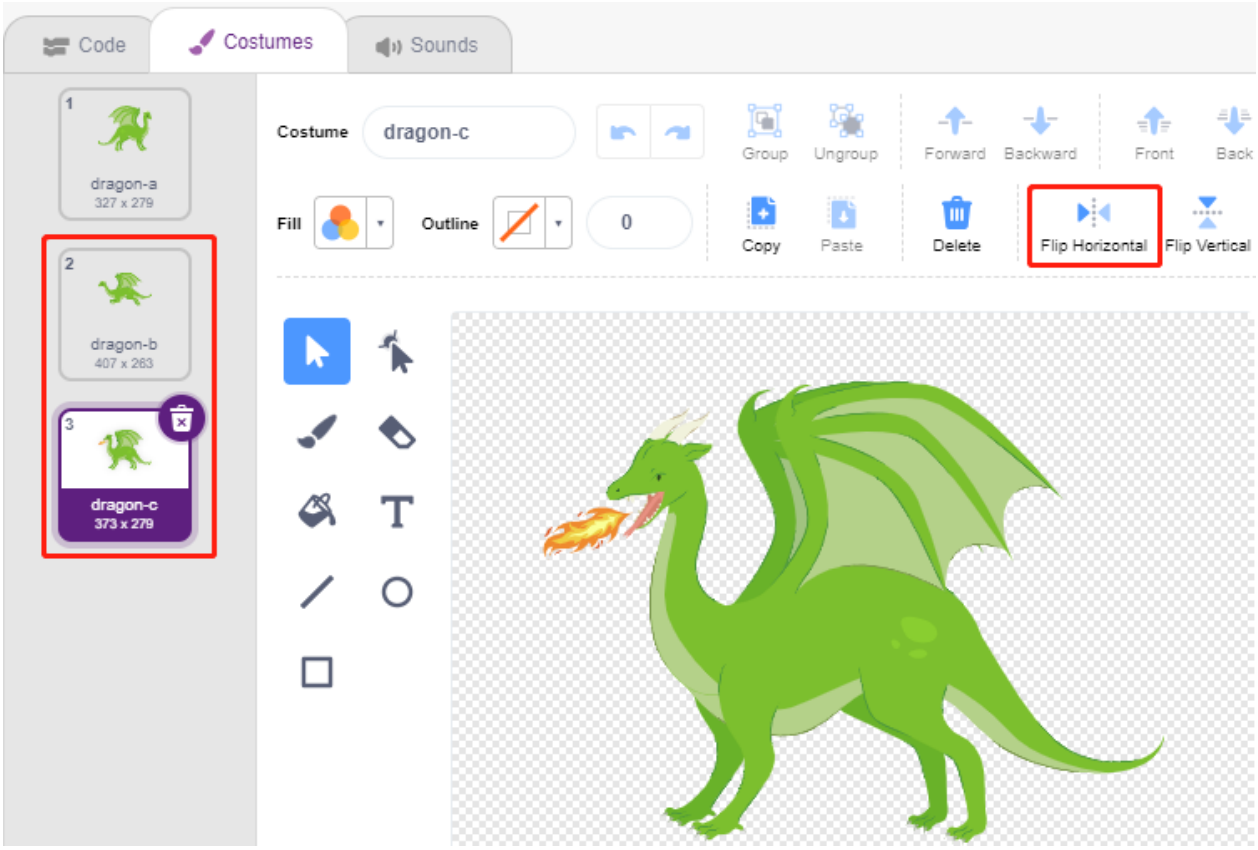
#### 1. ドラゴン

**Woods** の背景は **Choose a Backdrop** ボタンから追加します。

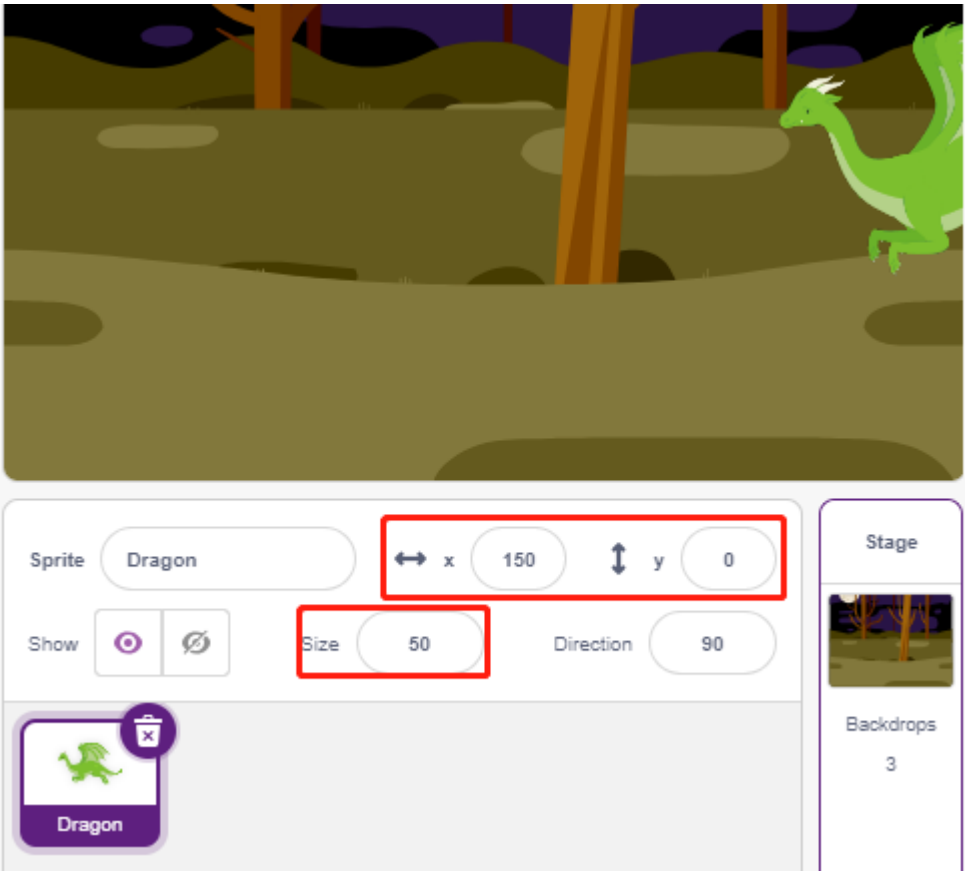

- デフォルトのスプライトを削除し、 **Dragon** スプライトを追加します。



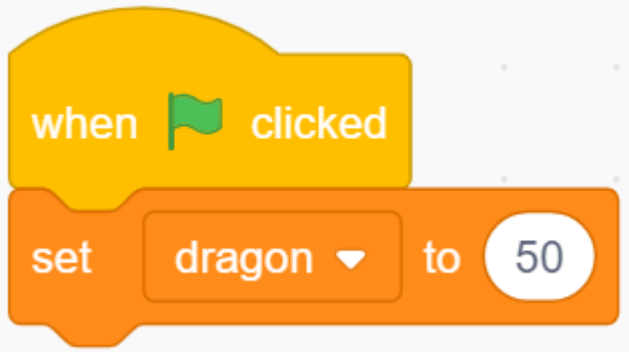

- **Costumes** ページに移動し、dragon-b と dragon-c を水平に反転します。



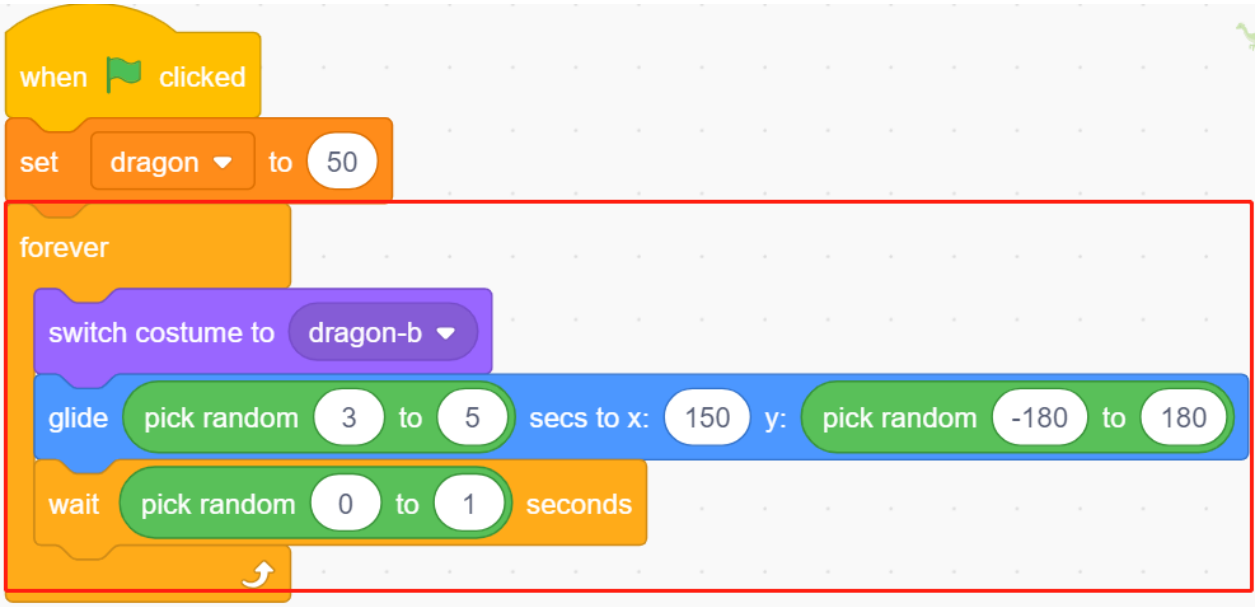

- サイズを 50% に設定します。




- 変数 - **dragon** を作成して、ドラゴンのライフポイントを記録し、初期値を 50 に設定します。




- 次に、スプライトのコスチュームを **dragon-b** に切り替え、 **Dragon** スプライトを一定の範囲で上下に動かします。

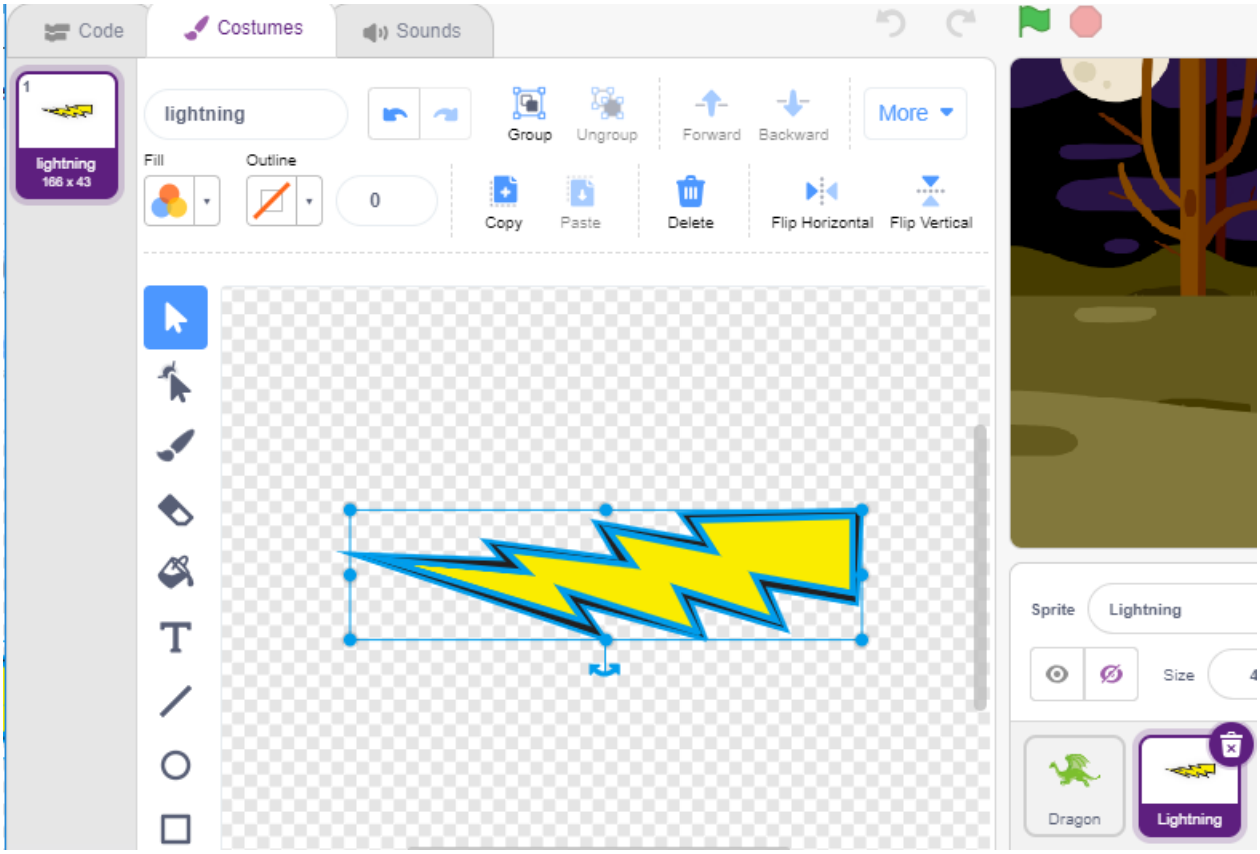



- **Dragon** スプライトが吹き出す火として、 **Lightning** スプライトを追加します。Costumes ページで 90 ° 時計回りに回転させ、 **Lightning** スプライトが正しい方向に動くようにします。

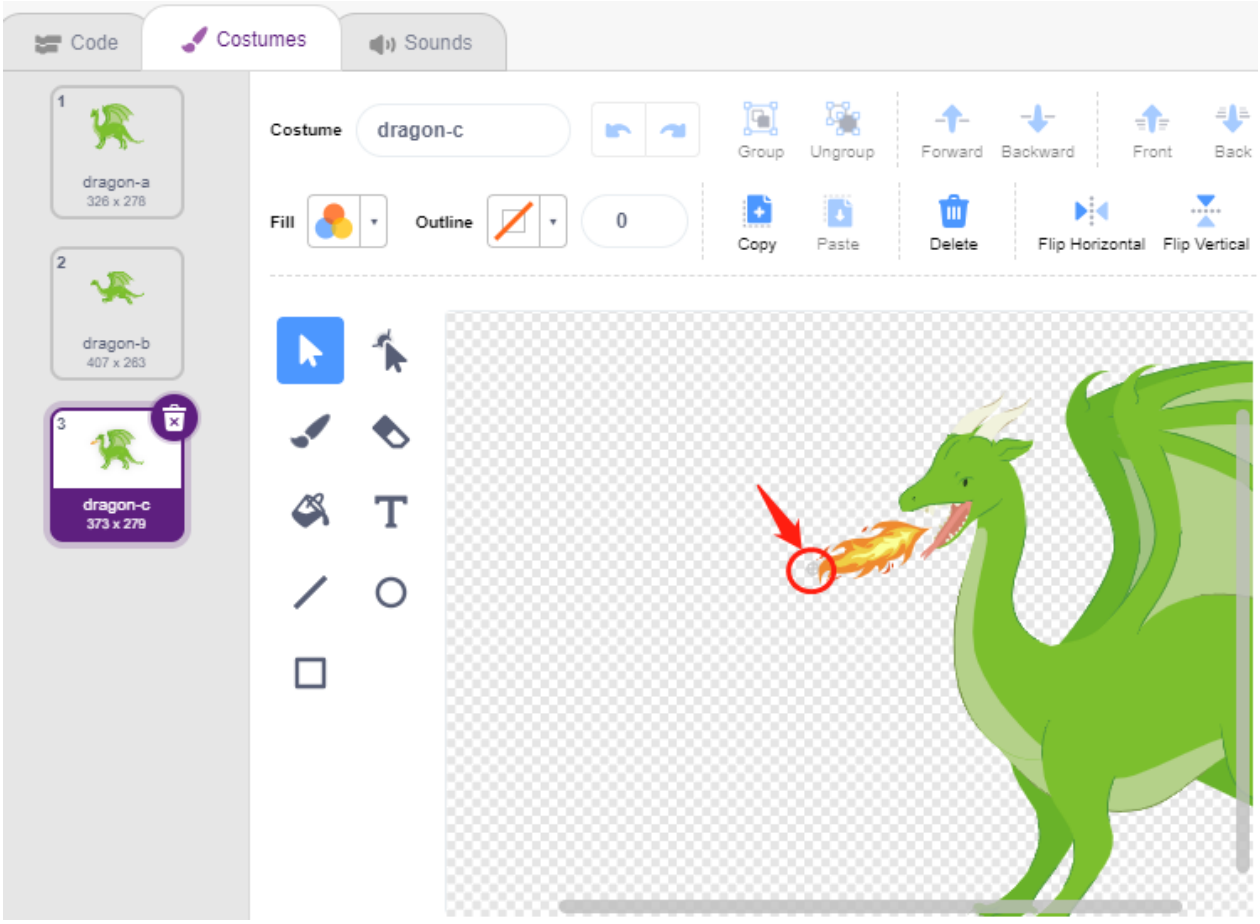
---

**注釈:** **Lightning** スプライトのコスチュームを調整する際、中心から外れてしまうことがありますが、それは避ける必要があります！ 中心点はスプライトの真ん中に合わせてください！

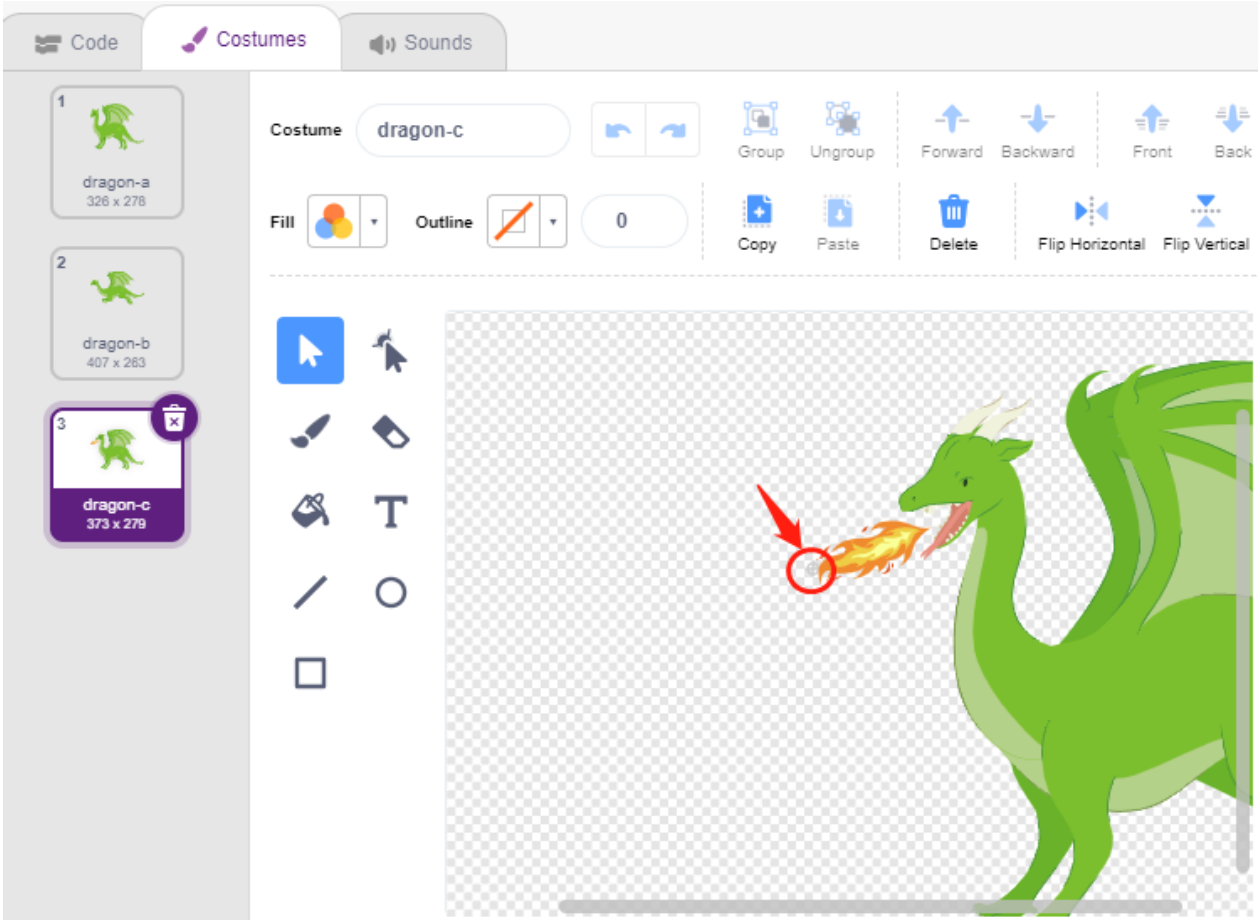
---



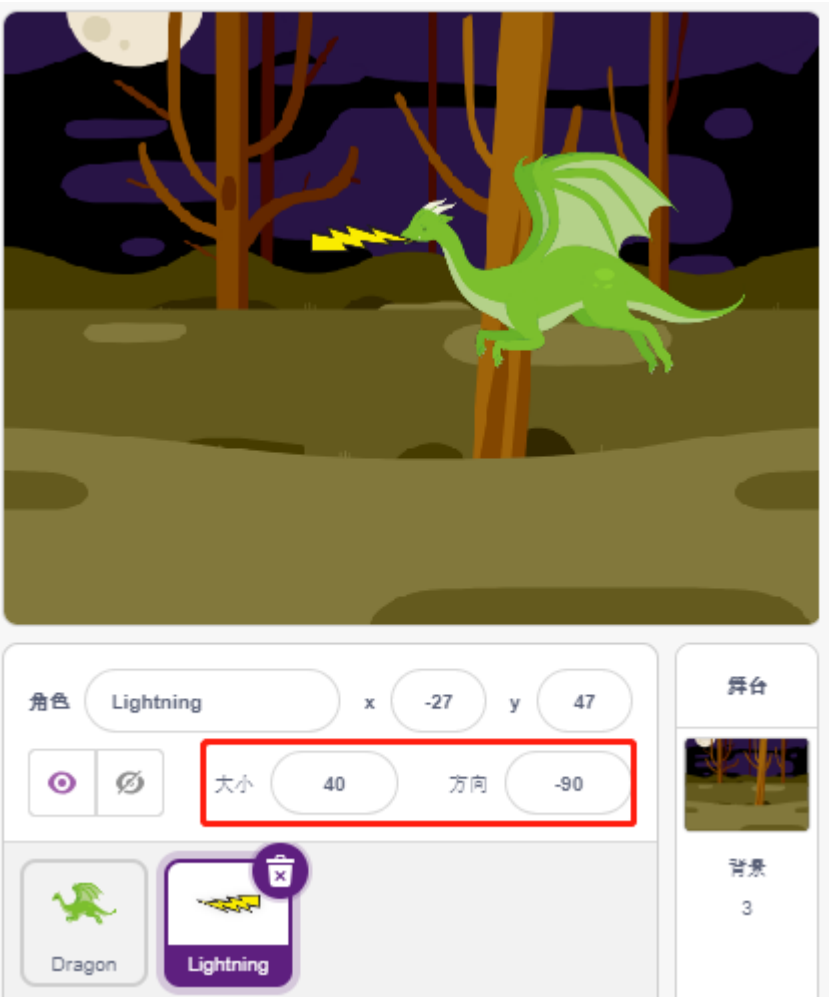

- まず、 **Dragon** スプライトの **dragon-c** コスチュームを調整し、その中心点が火の尾に位置するようにします。これにより、 **Dragon** スプライトと **Lightning** スプライトの位置が正確になり、 **Lightning** がドラゴンの足元から発射されるのを防ぎます。



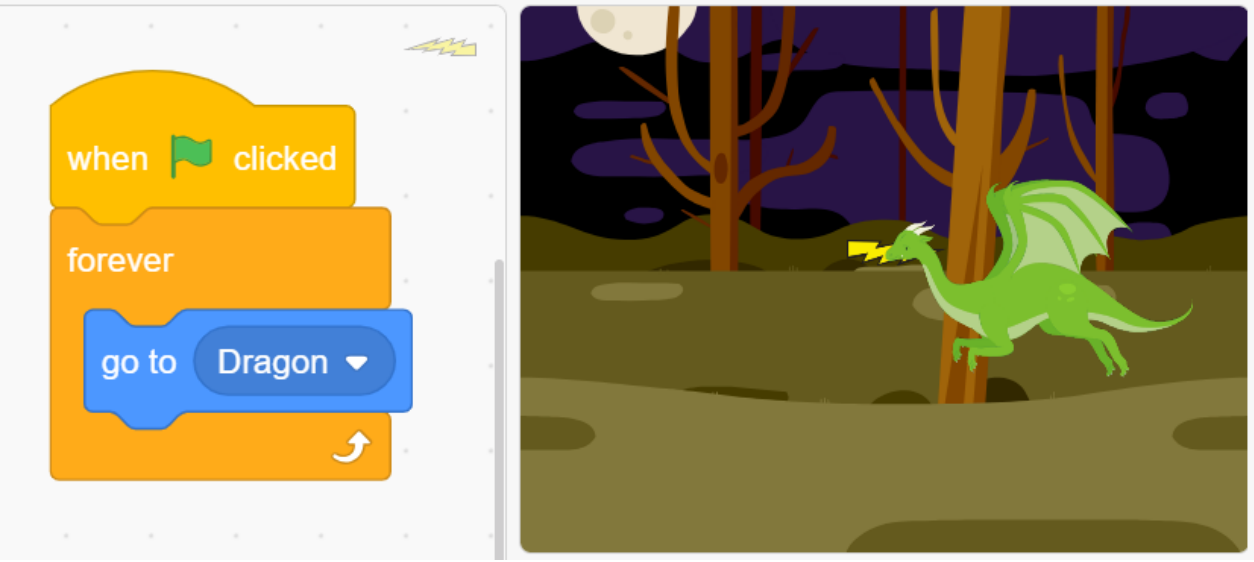

- 対応するように、 **dragon-b** はドラゴンの頭を中心点と一致させる必要があります。



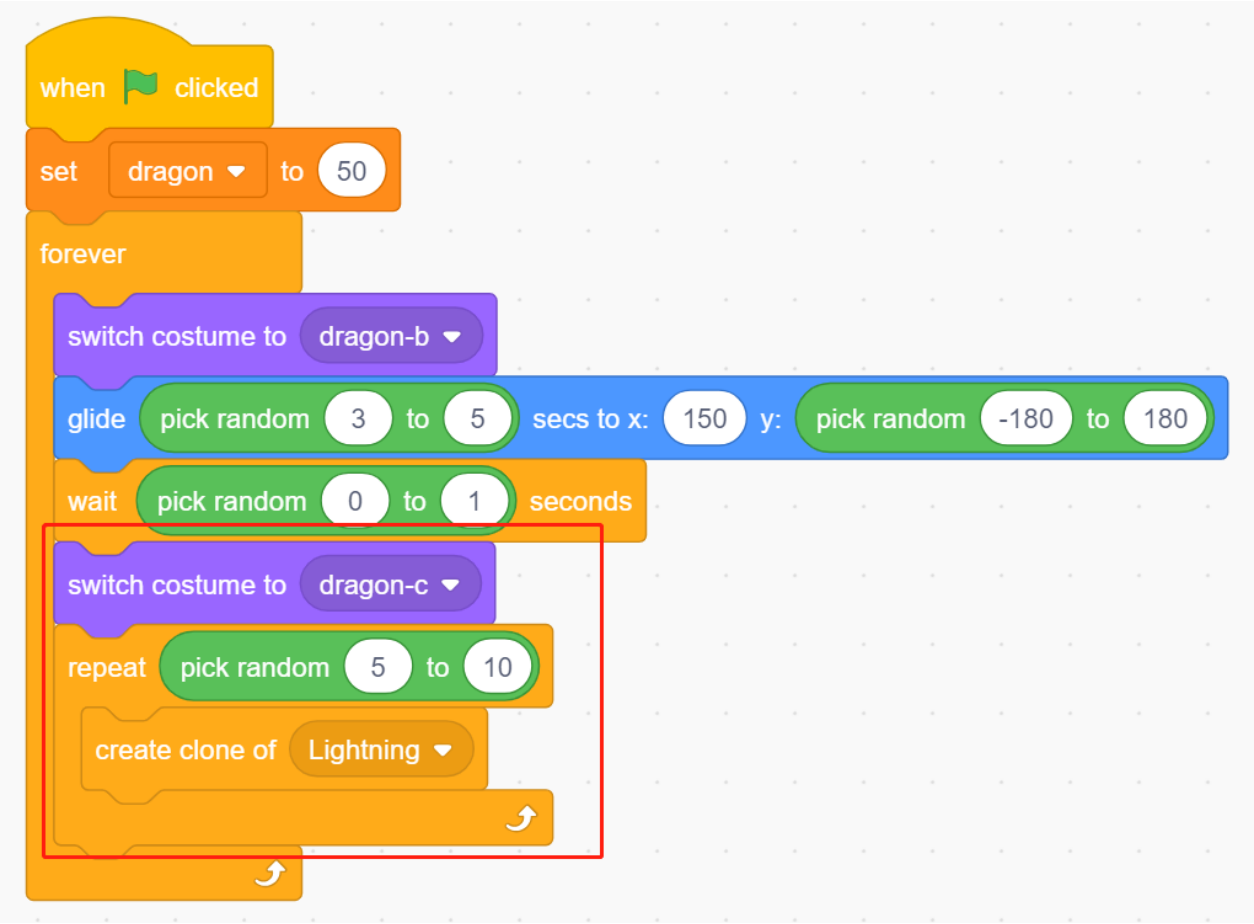

- **Lightning** スプライトのサイズと方向を調整し、画像がより調和のとれたものになるようにします。



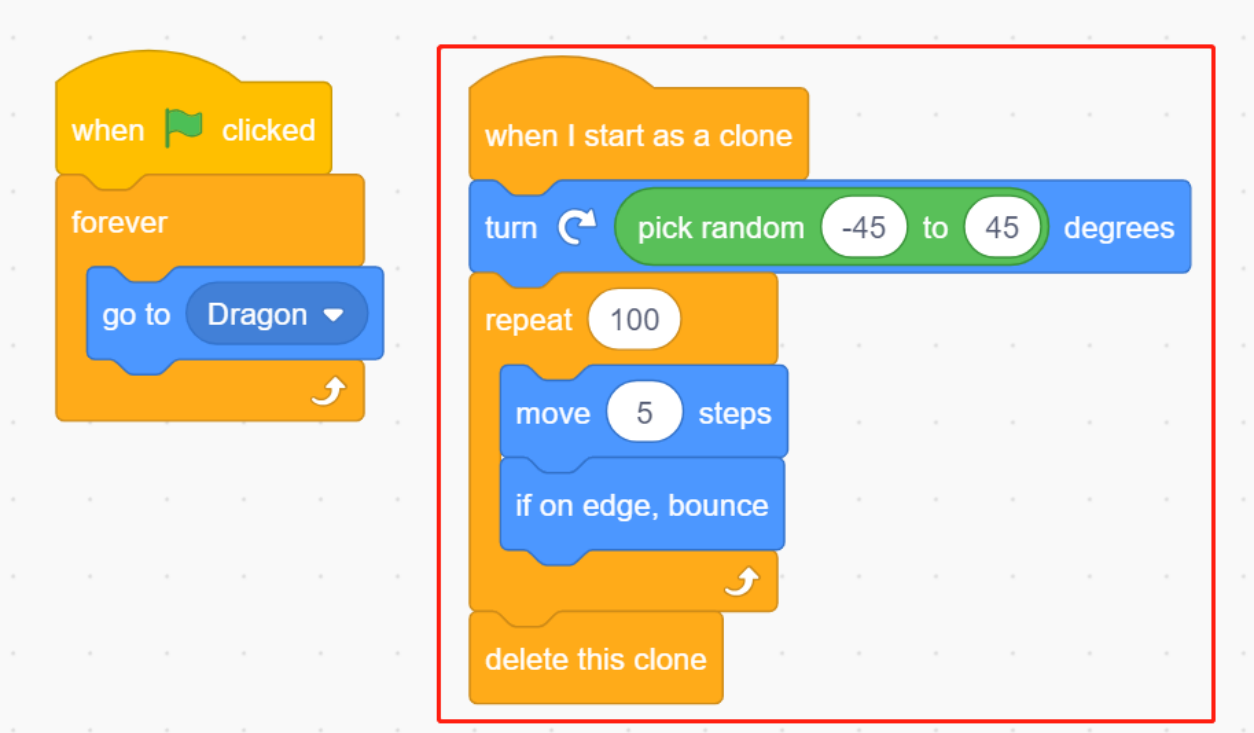

- 今度は **Lightning** スプライトをスクリプト化します。これは簡単で、 **Dragon** スプライトを常に追従させるだけです。この時点で緑の旗をクリックすると、 **Dragon** が稲妻を口にくわえて動き回るのが見えるでしょう。



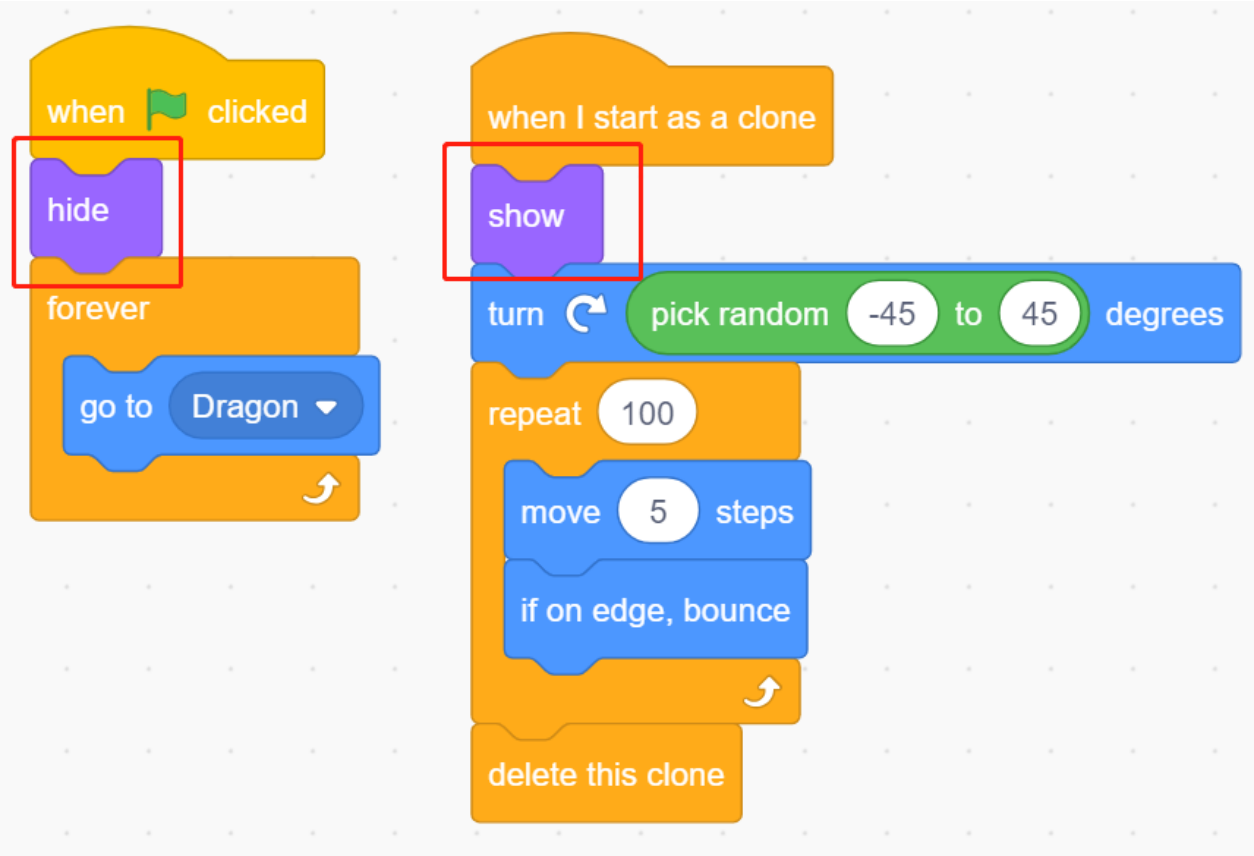
- **Dragon** スプライトに戻り、口から火を吹き出すようにします。口の中の火を発射しないよう注意し、**Lightning** スプライトのクローンを作成します。




- **Lightning** スプライトをクリックし、 **Lightning** のクローンをランダムな角度で発射します。一定の時間が経過すると、壁から跳ね返り消えます。



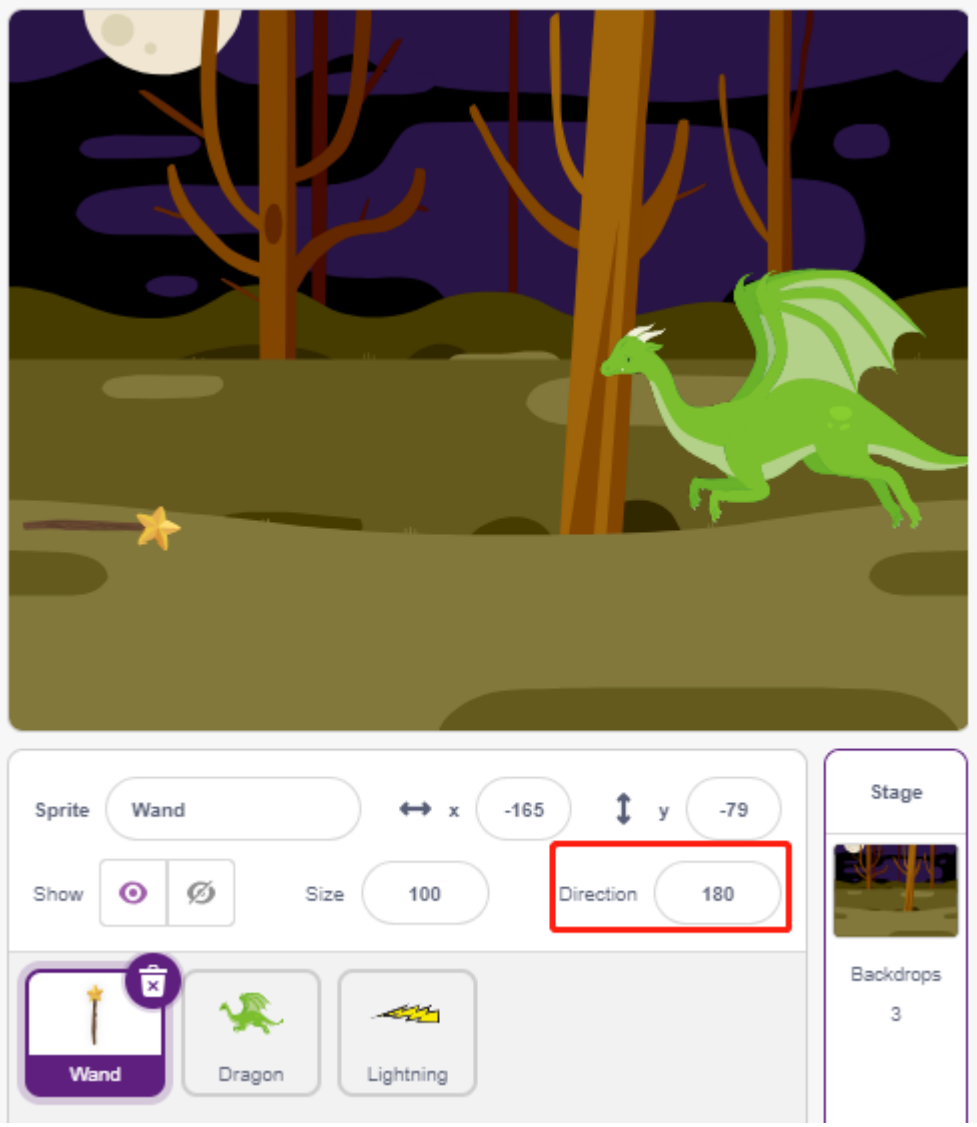

- **Lightning** スプライトで、その本体を隠し、クローンを表示します。



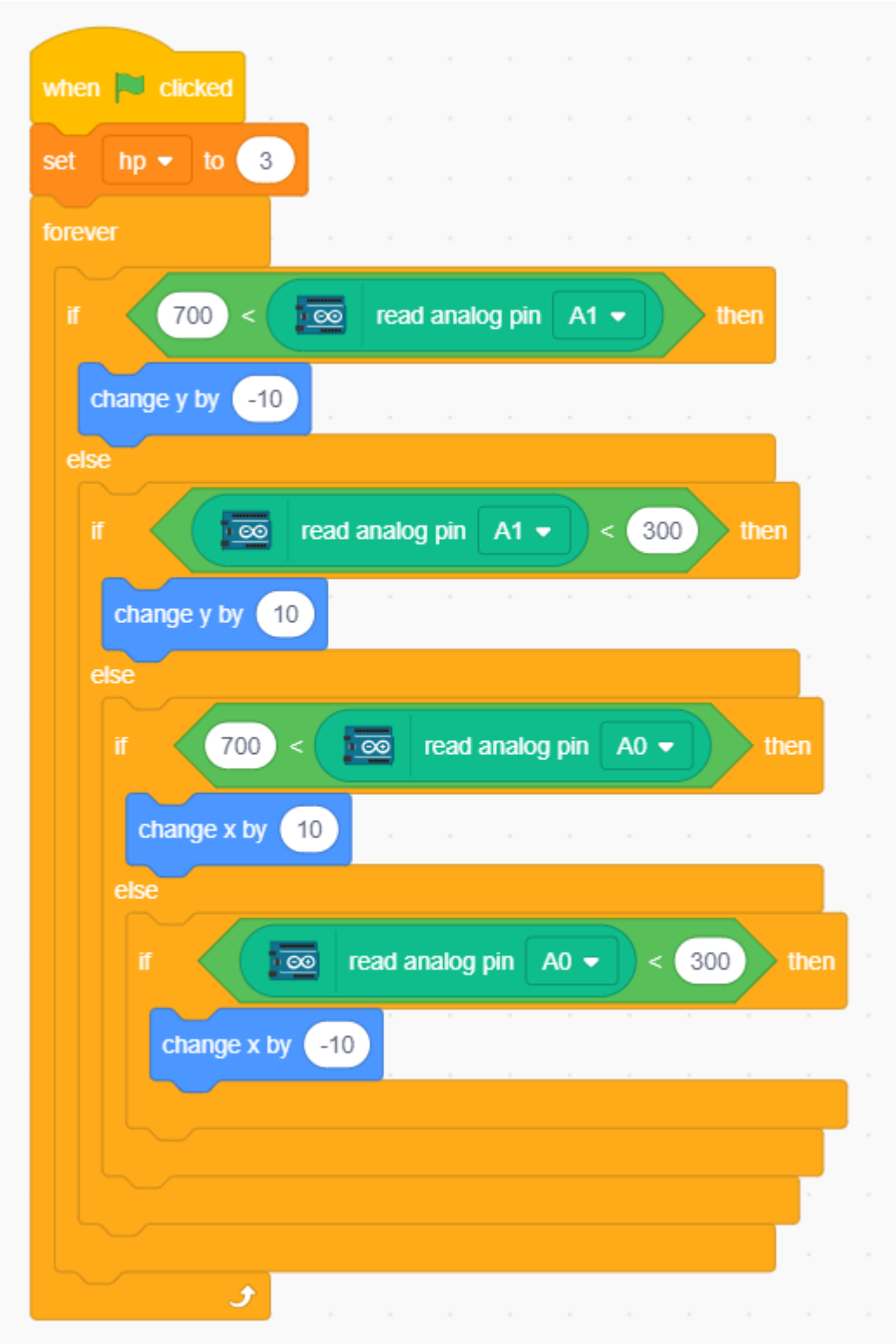
これでドラゴンは上下に動きながら火を吹き出すことができます。

**2. Wand**

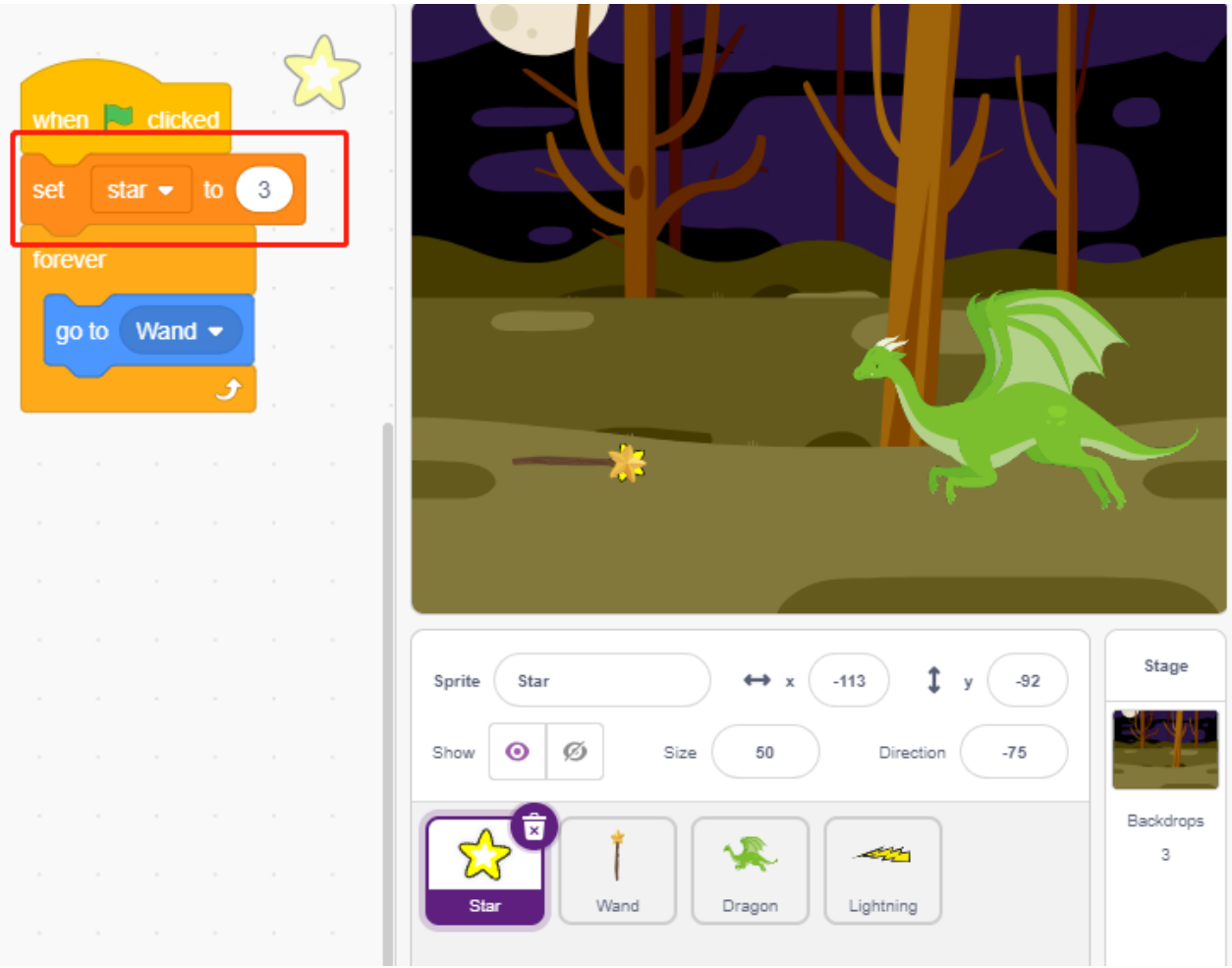
- **Wand** スプライトを作成し、方向を 180 にして右を向かせます。



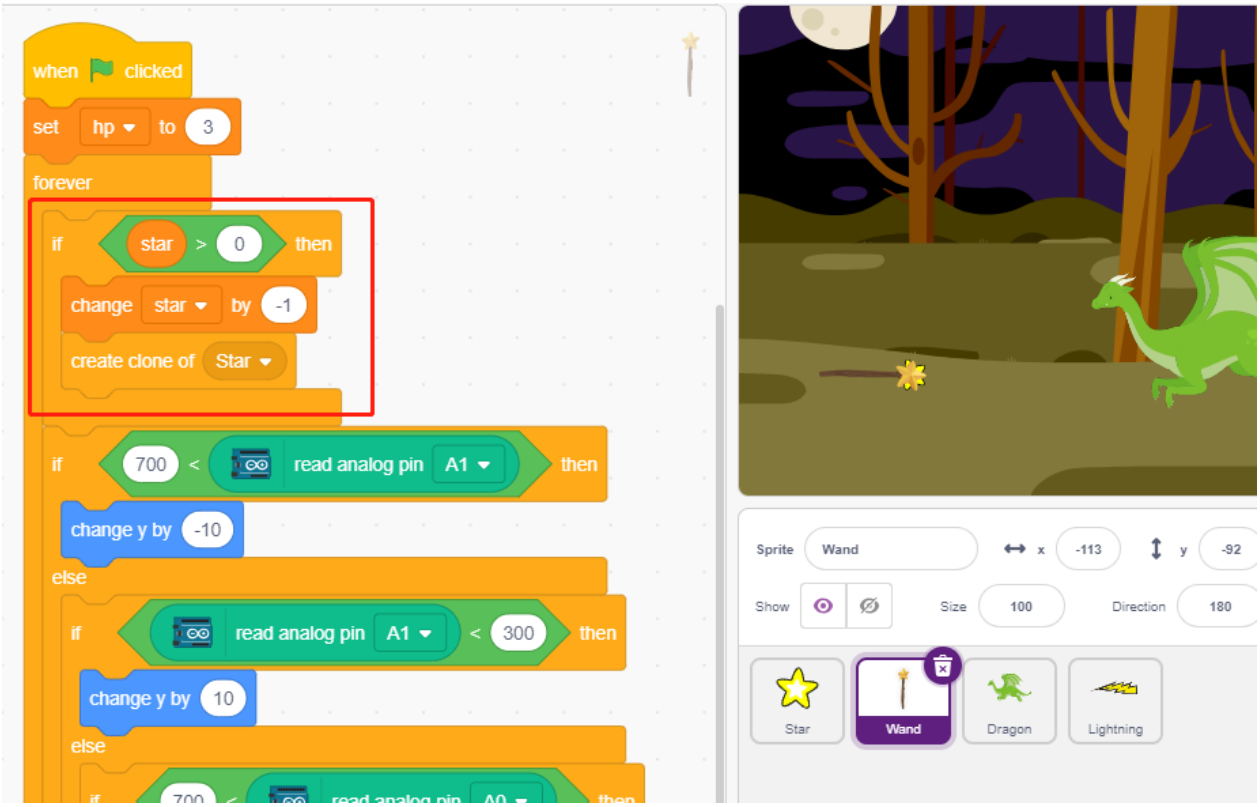

- 変数 **hp** を作成して、そのライフ値を記録します。初期設定は 3 にします。次に、ジョイスティックの値を読み取り、それを使用してワンドの動きを制御します。



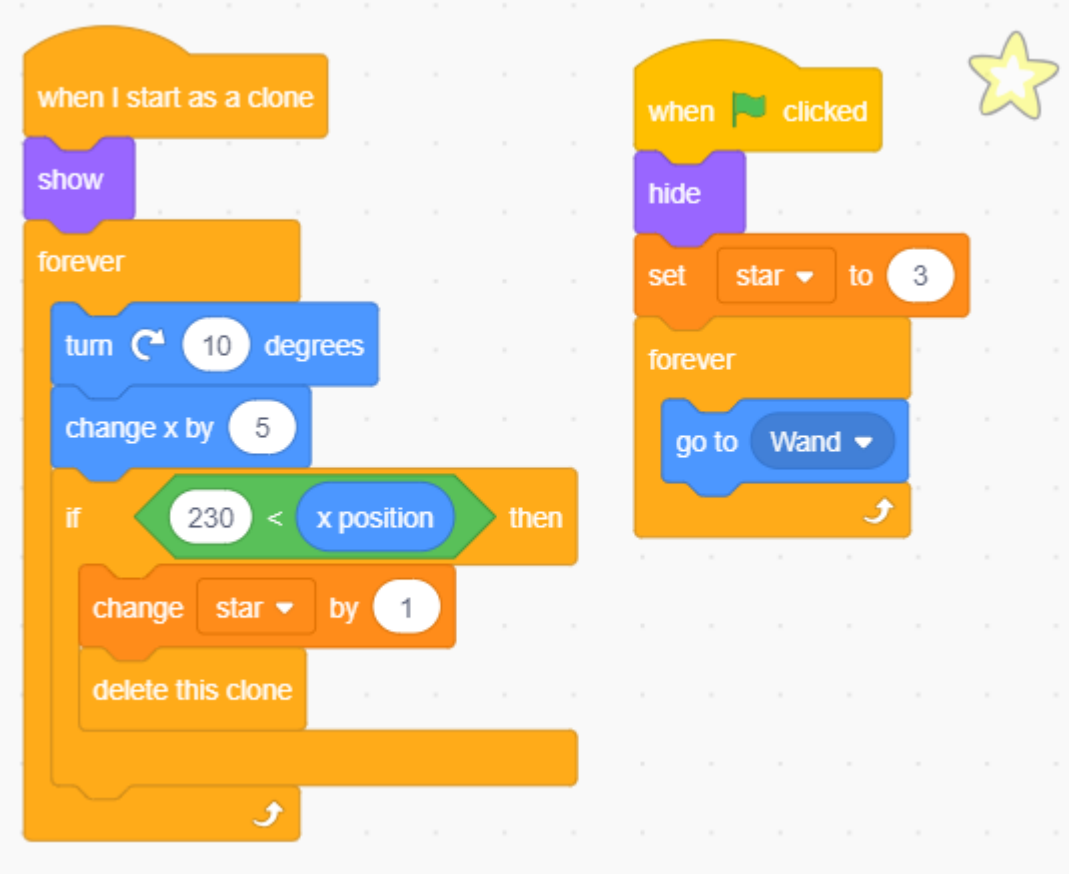

- ドラゴンは雷を持っており、それを砕くワンドには「魔法の弾丸」があります！ **Star** スプライトを作成し、サイズを変更して、常に **Wand** スプライトに従い、星の数を 3 つに制限します。




- **Wand** スプライトが星を自動的に撃つようにします。 **Wand** スプライトが星を撃つ方法は、ドラゴンが火を吹き出す方法と同じです。クローンを作成するだけです。




- **Star** スプライトに戻り、そのクローンが右に回転して撃つようにスクリプトを書きます。ステージの外に出た後に消え、星の数を回復します。 **Lightning** スプライトと同様に、本体を隠してクローンを表示します。




これで、星の弾丸を撃つワンドができました。

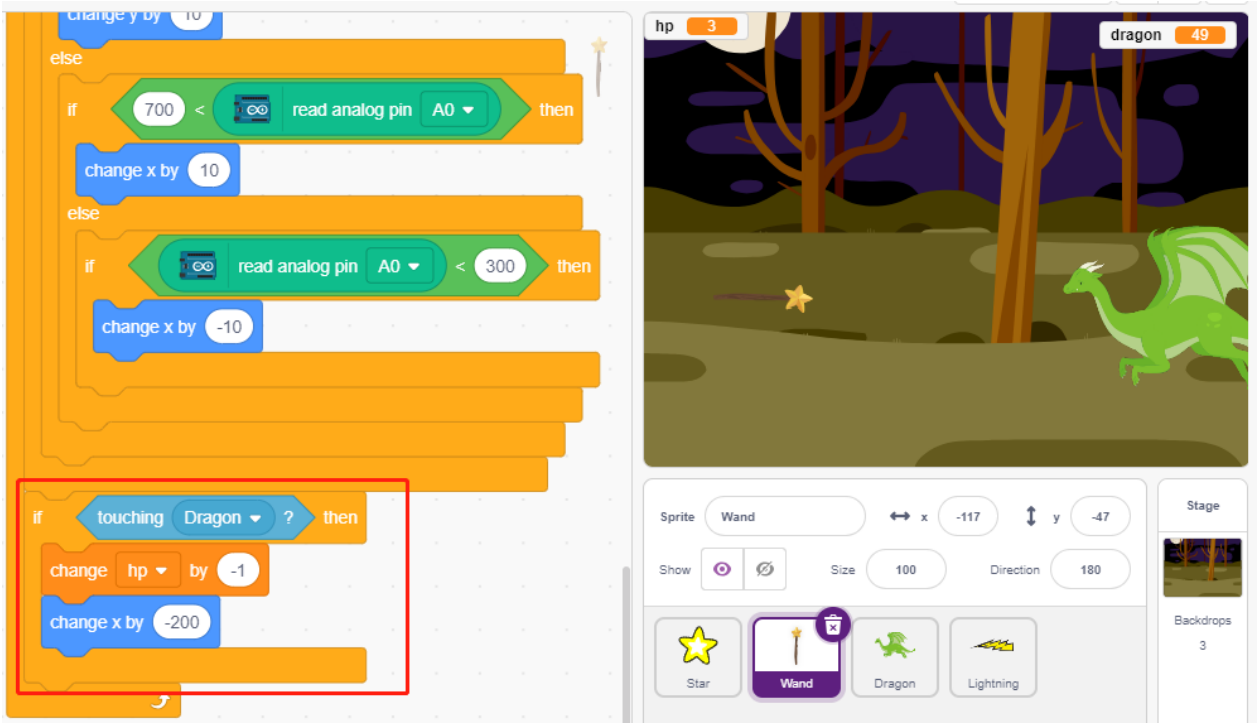
**3. Fight!**

ワンドとドラゴンは現在まだ互いに対立していますが、それらを戦わせることにします。ドラゴンは強力であり、ワンドはドラゴンに対して聖戦を行う勇敢な人です。彼らの間の相互作用は、以下の部分で構成されます。

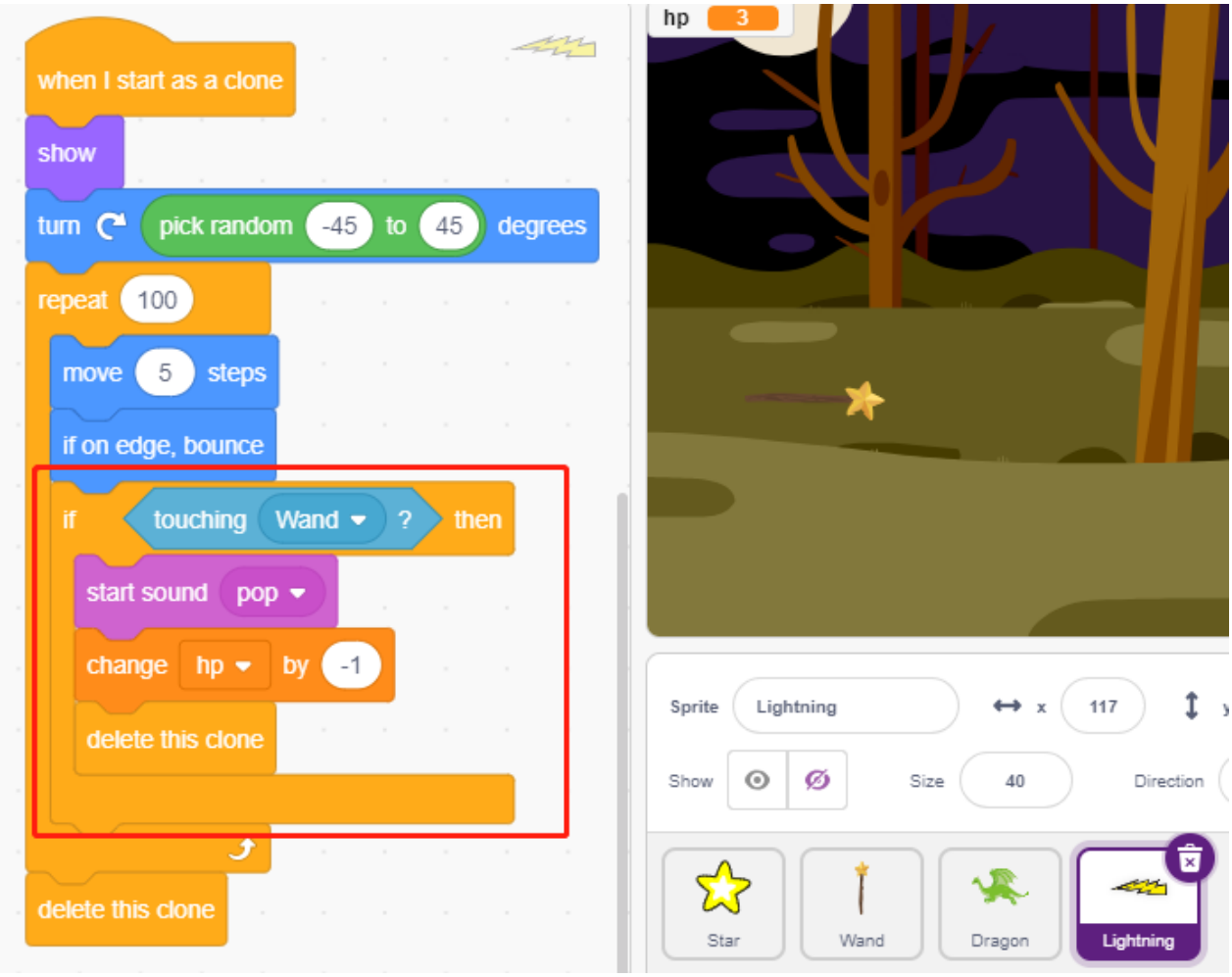
1. ワンドがドラゴンに触れると、ワンドは後ろに打ち退けられ、ライフポイントを失います。
2. 雷がワンドに当たると、ワンドはライフポイントを失います。
3. 星の弾丸がドラゴンに当たると、ドラゴンはライフポイントを失います。

その後、各スプライトのスクリプトを変更する作業に進みます。

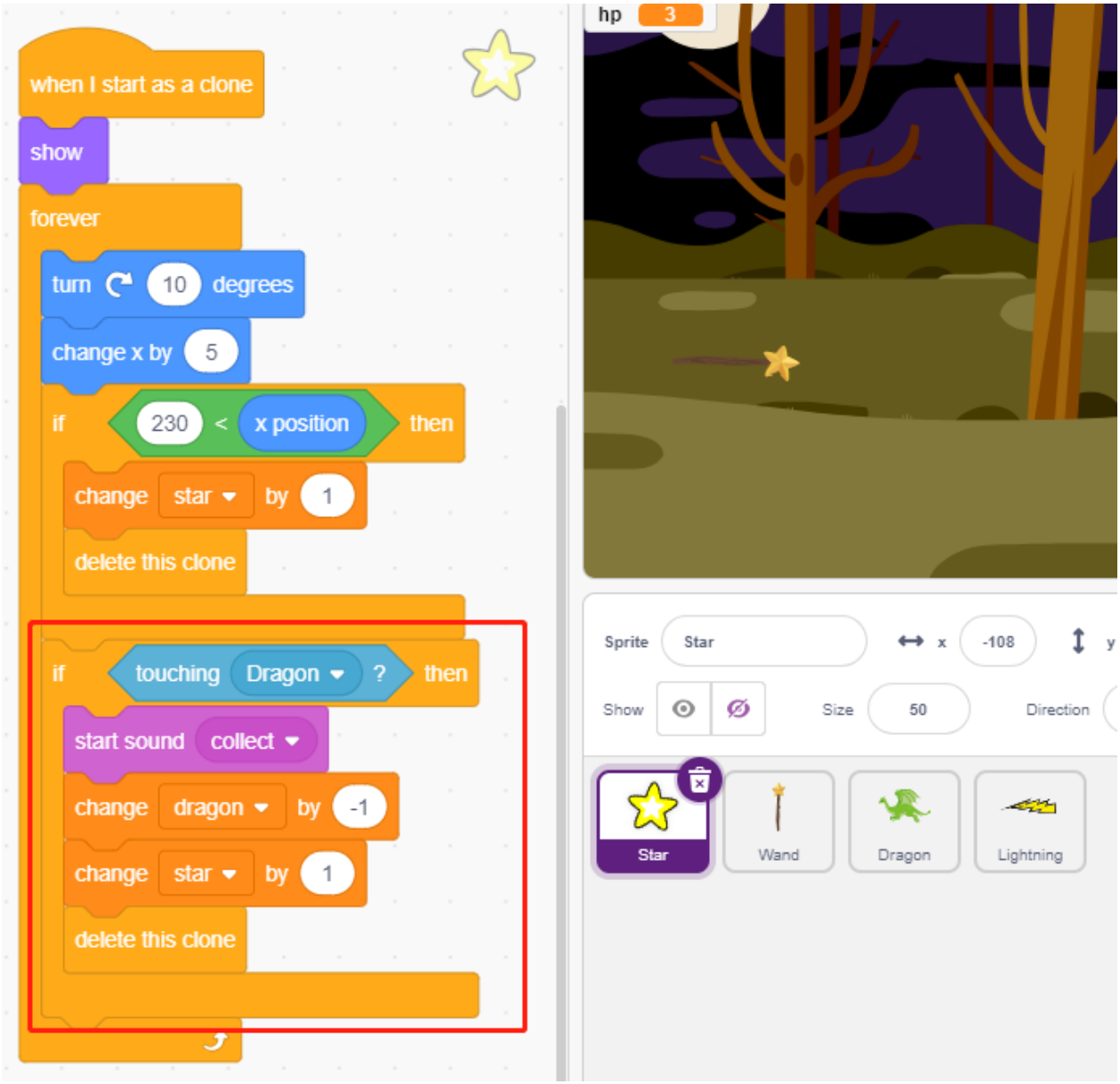
- **Wand** が **Dragon** に当たると、後ろに打ち退けられ、ライフポイントを失います。




- **Lightning** （ **Lightning** スプライトのクローン）が **Wand** スプライトに当たると、ポップ音を鳴らして消え、 **Wand** はライフポイントを失います。



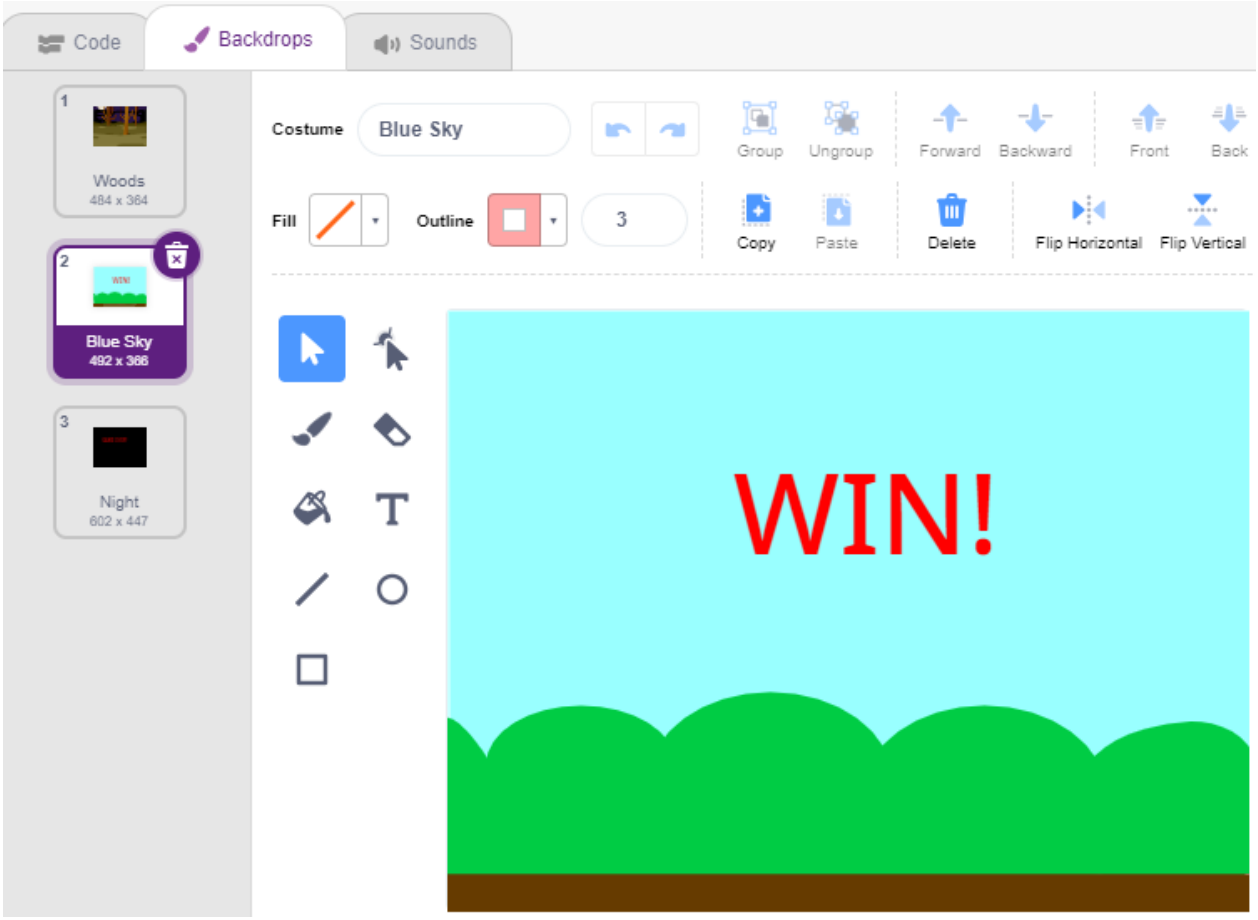

- **Star** （ **Star** スプライトのクローン）が **Dragon** に当たると、収集音を発して消え、 **Star** の数を回復し、 **Dragon** はライフポイントを失います。



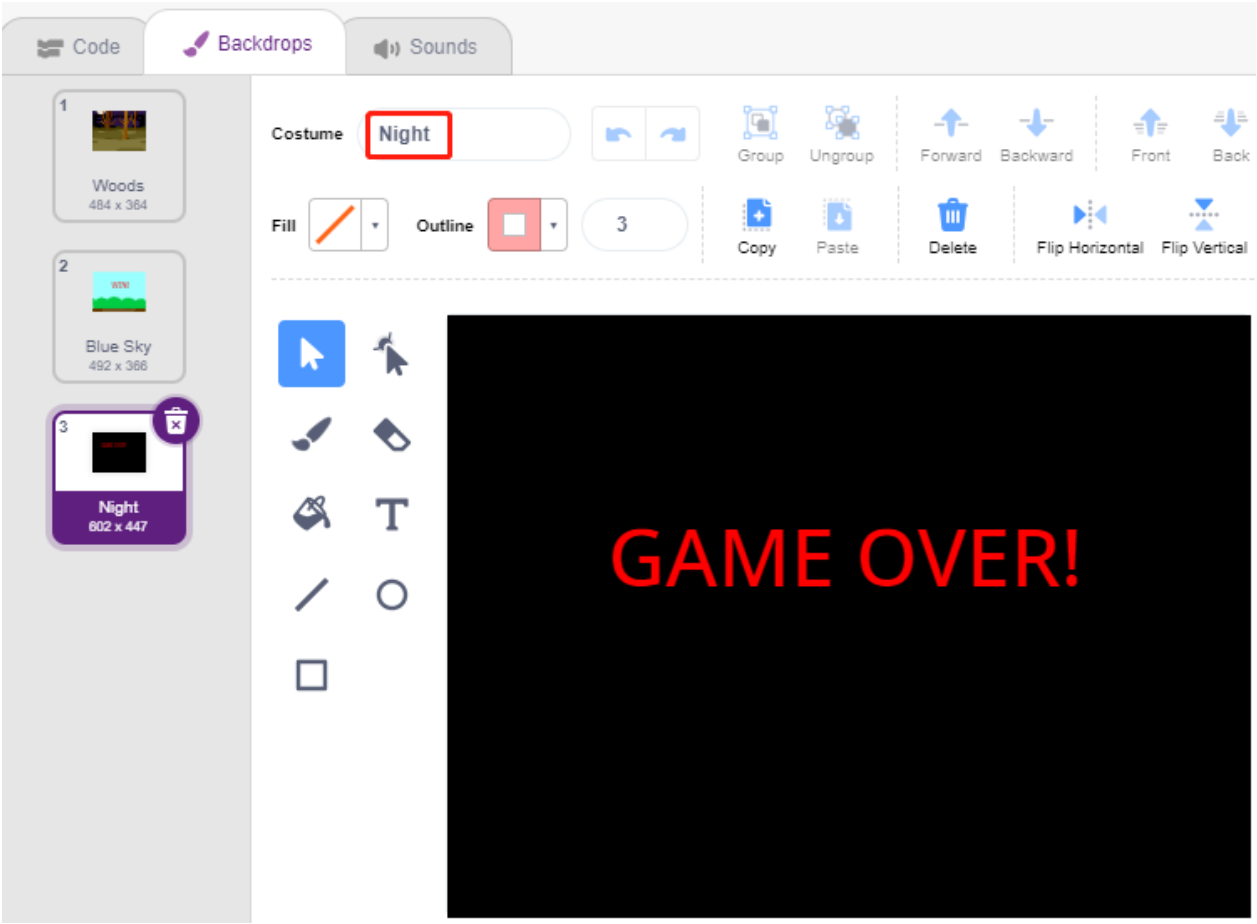

### 4. ステージ

**Wand** と **Dragon** の戦いは最終的に勝者と敗者に分かれることになりますが、それはステージで表現します。

- **Blue Sky** の背景を追加し、それに「WIN!」の文字を書き、ドラゴンが倒され、夜明けが来たことを示します。




- そして、次のように空白の背景を修正して、ゲームが失敗し、全てが闇になることを示します。




- これらの背景を切り替えるスクリプトを書きます。緑の旗がクリックされると、 **Woods** の背景に切り替えます。ドラゴンのライフポイントが 1 未満の場合、ゲームが成功し、背景を **Blue Sky** に切り替えます。 **Wand** のライフポイントが 1 未満の場合、 **Night** の背景に切り替え、ゲームが失敗します。


**3. Scratch で車を操作する**

以下のプロジェクトはプログラミングの難易度の順に書かれています。これらの本を順番に読むことをおすすめします。

各プロジェクトでは、回路の組み立て方やプログラムの手順をステップバイステップで非常に詳細に教えています。最終結果を達成するための方法も含まれています。

もちろん、スクリプトを直接開いて実行することもできますが、関連する資料を github からダウンロードしたことを確認する必要があります。

ダウンロードが完了したら、それを解凍します。 **アップロードモード** を参照して、個々のスクリプトを直接実行します。

## 7.26 3.1 車をテストする

このセクションでは、車を前進させるためのスクリプトの書き方を学びますが、まずは **カープロジェクト** を参照して車の組み立て方や基本的な理解を得る必要があります。

プロジェクトを開始する前に、**アップロードモード** で PictoBlox を使用する手順を知っておく必要があります。

### 7.26.1 必要な部品

このプロジェクトでは、以下の部品が必要です。

キットでまとめて購入すると便利です。以下にリンクを掲載しています。

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから、部品を個別に購入することもできます。

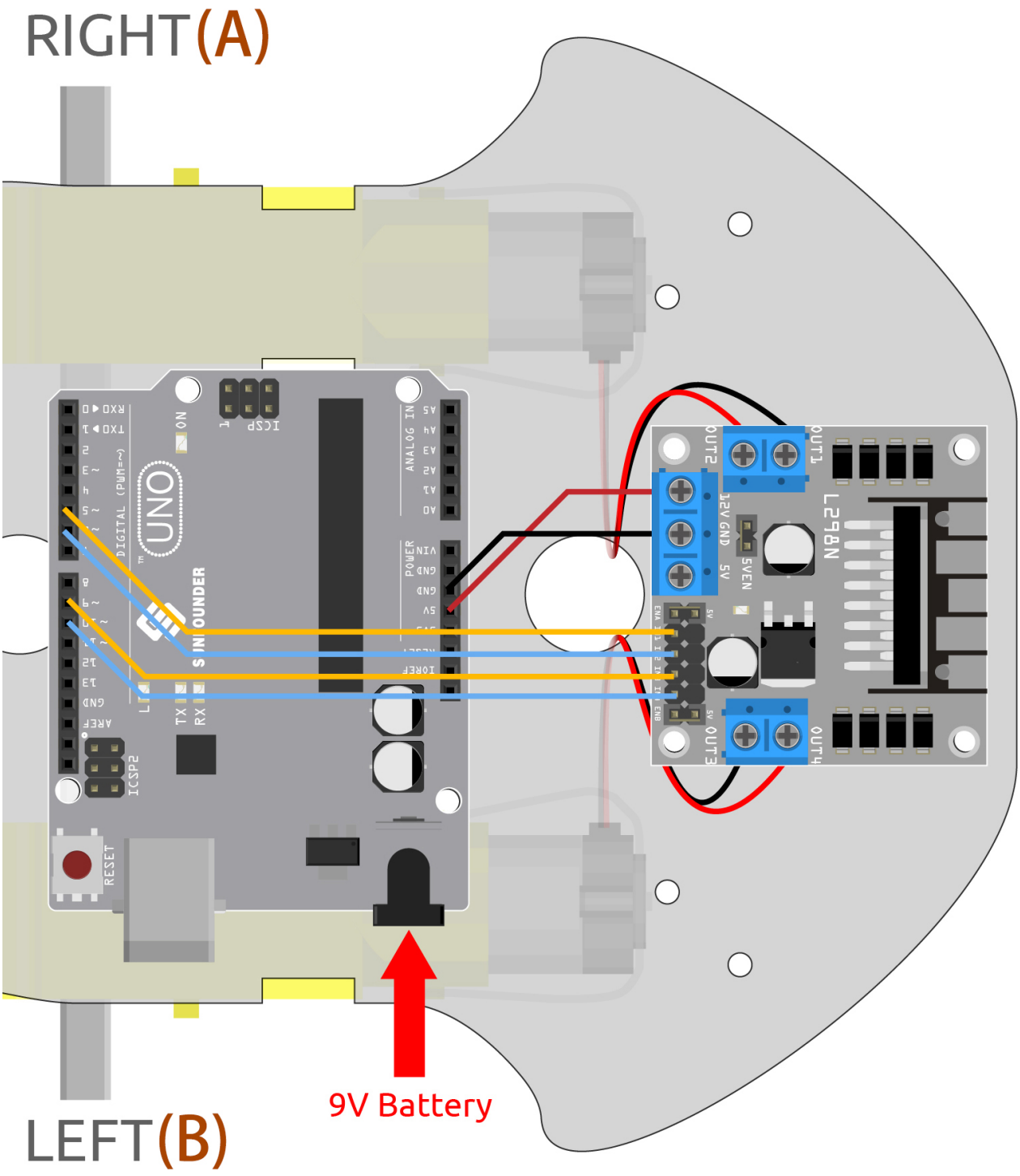
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| *L298N* **モジュール** | |
| *TT* **モーター** | - |

### 7.26.2 回路の作成

L298N モータードライバモジュールは、DC モーターやステッピングモーターを駆動するための高出力モータードライバモジュールです。L298N モジュールは、最大 4 つの DC モーターや、2 つの DC モーターの方向と速度を制御することができます。

以下の図に従って、L298N モジュールと R3 ボードの間にワイヤを接続してください。

| L298N | R3 ボード | モーター |
|---|---|---|
| IN1 | 5 | |
| IN2 | 6 | |
| IN3 | 9 | |
| IN4 | 10 | |
| OUT1 | | 右モーターの黒ワイヤ |
| OUT2 | | 右モーターの赤ワイヤ |
| OUT3 | | 左モーターの黒ワイヤ |
| OUT4 | | 左モーターの赤ワイヤ |

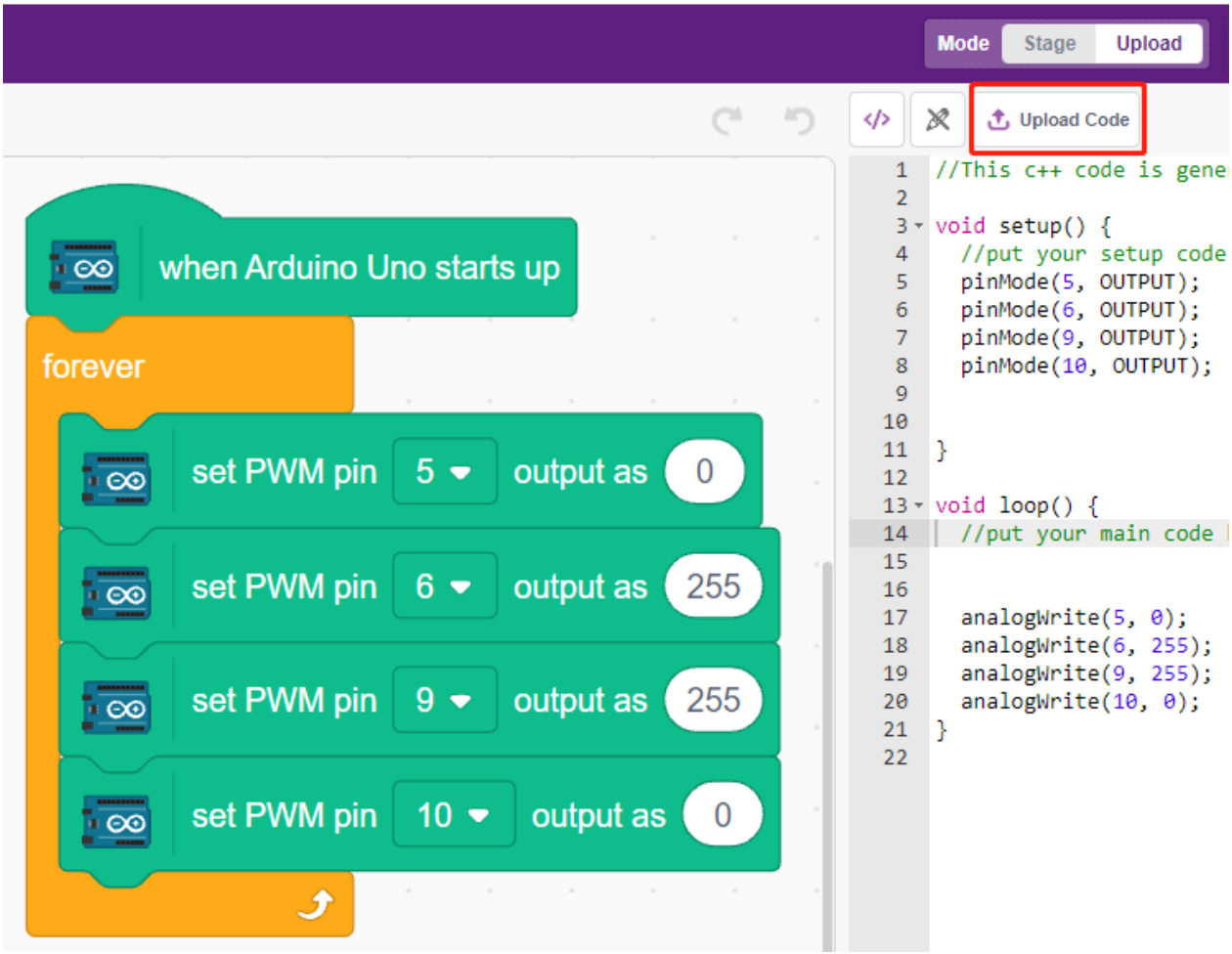

RIGHT(A)
LEFT(B)
9V Battery
L298N
OUT1
OUT2
OUT3
OUT4
12V
GND
5V
5VEN
UNO
SUNFOUNDER
ANALOG IN
POWER
DIGITAL (PWM=~)
ICSP
ICSP2
RESET
TX
RX
ON

### 7.26.3 プログラミング

#### 1. 車を前進させる

上記の配線に基づいて、ピン 5 と 6 は右モーターの回転を制御するために使用され、ピン 9 と 10 は左モーターの回転を制御するために使用されることがわかります。それでは、車を前進させるスクリプトを書いてみましょう。

Arduino Uno をボードとして選択した後、 **アップロードモード** に切り替え、以下の図に従ってスクリプトを書いてください。




**Upload Code** ボタンをクリックして、R3 ボードにコードをアップロードします。完了すると、車の 2 つのモーターが前進します（車を地面に置くと、直線上で前進しますが、2 つのモーターの速度が少し異なるため、カーブするかもしれません）。

もし両方のモーターが前進しない、または以下の状況が発生した場合、2 つのモーターの配線を調整する必要があります。

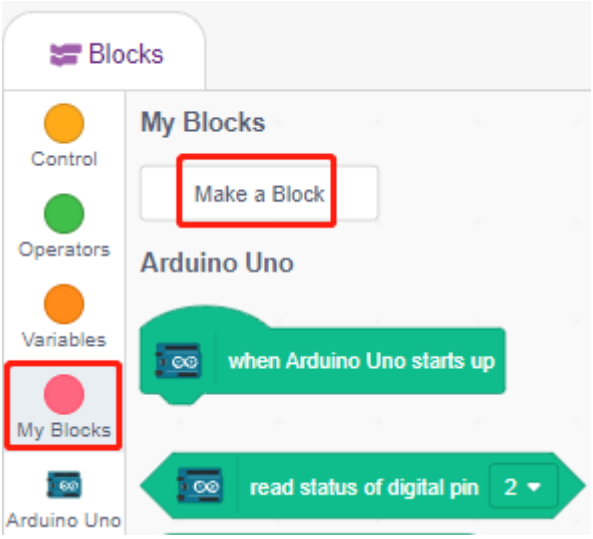
- 両方のモーターが同時に後退する場合（左モーターが時計回り、右モーターが反時計回りに回転する場合）、左と右のモーターの配線を同時に交換します。OUT1 と OUT2、OUT3 と OUT4 を交換します。

- 左モーターが後退する場合（時計回りの回転）、左モーターの OUT3 と OUT4 の配線を交換します。
- 右モーターが後退する場合（反時計回りの回転）、右モーターの OUT1 と OUT1 の配線を交換します。

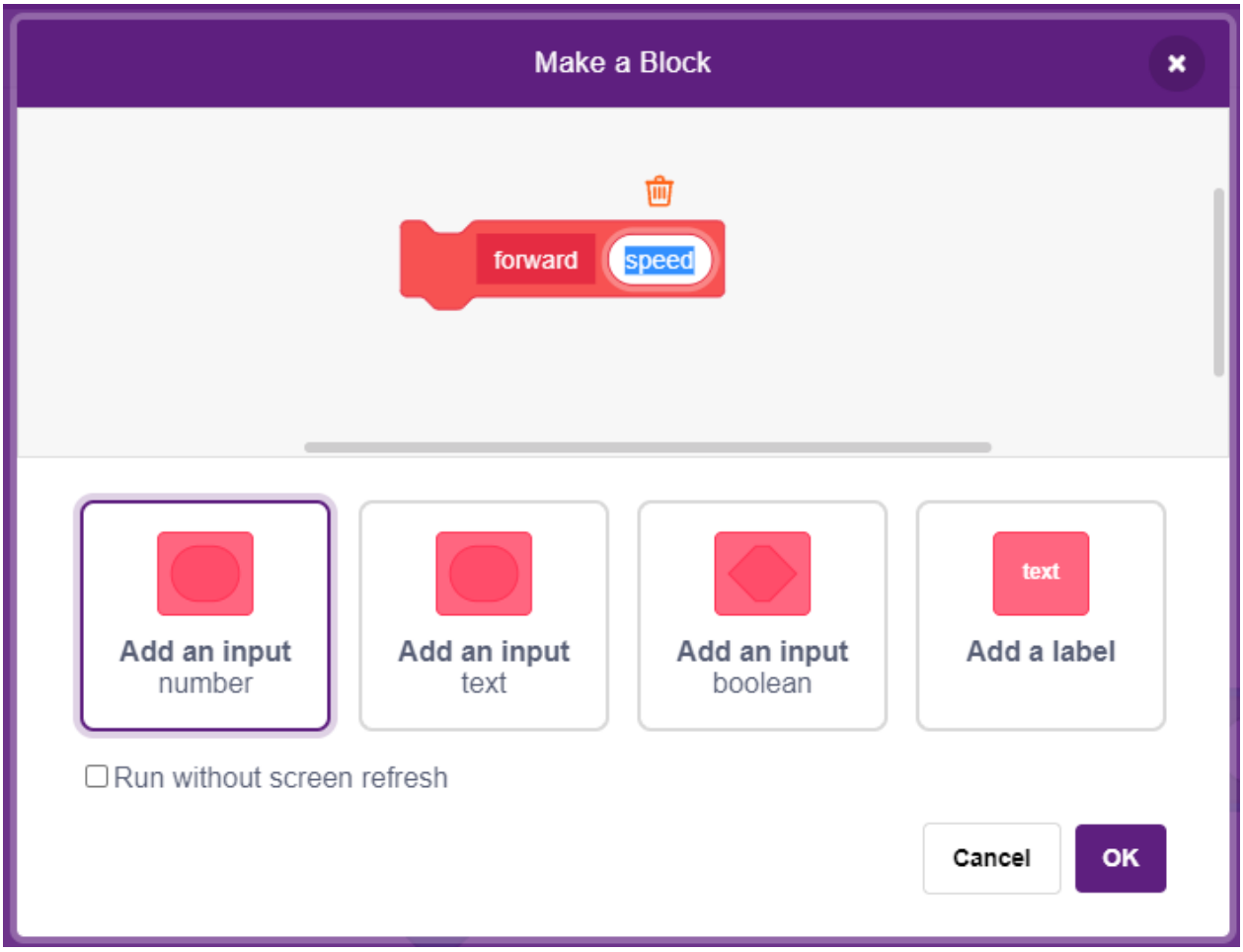
**2. ブロックを作成**

スクリプトをよりクリーンで使いやすくするため、前進の動作を制御するすべてのブロックを 1 つのブロックにまとめ、使用するときはこのブロックを直接呼び出します。

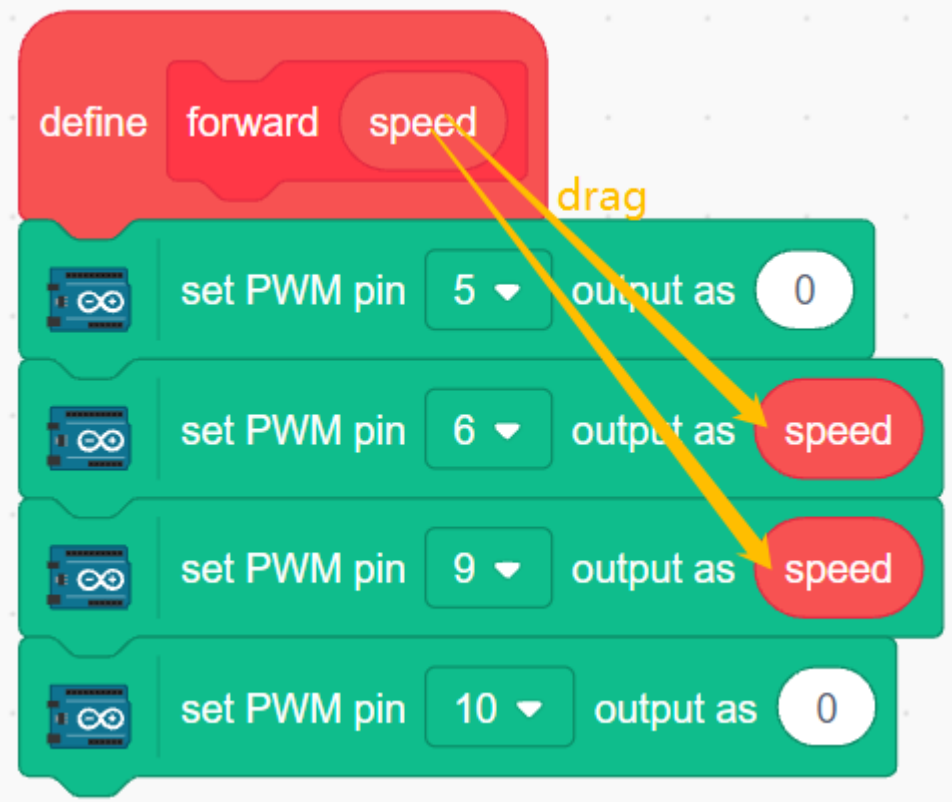
**My Blocks** パレットの **Make a Block** をクリックします。




ブロックの名前を **forward** に設定し、 **Add an input** にチェックを入れ、入力名を **speed** に設定します。

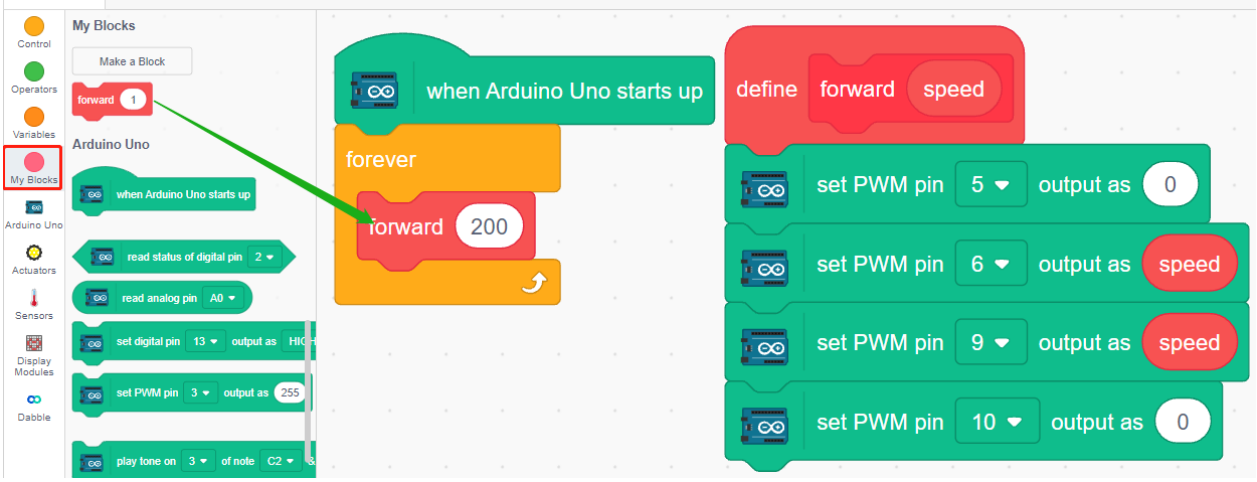



車を前進させるためのブロックを **forward** にドラッグ&ドロップします。pin6 と pin9 にパラメーター **speed** を追加する必要があります。




作成したブロックを [Forward] ブロックの **forward** で呼び出します。Upload モードでは、最初に [When Arduino Uno starts up] ブロックを追加する必要があります。

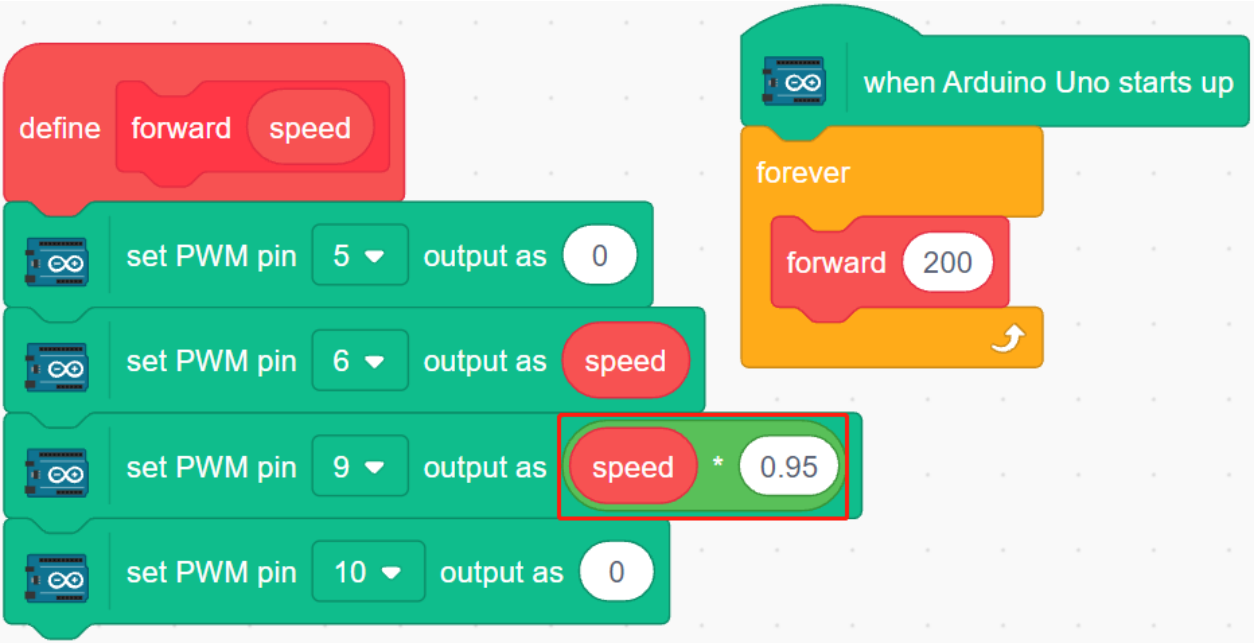
- モーターの回転速度の範囲は 100～255 です。




### 3. モーターの速度を調整

2 つのモーターの速度にわずかな違いがあるため、車がまっすぐ進むことができない場合、車ができるだけまっすぐ進むように、左と右のモーターに異なる速度を設定します。

私の車は右前方にゆっくりと進むので、左のモーターの速度を少し下げます。




## 7.27 3.2 動き

このプロジェクトは *3.1* **車をテストする** に基づいて車を全方向に移動させるものです。

プログラミングを始める前に、L298N の動作原理を確認しましょう。

ENA と IN1、IN2 の動作関係は以下の通りです。

| ENA | IN1 | IN2 | **右モータ (A) の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

ENB と IN3、IN4 の動作関係は次の通りです。

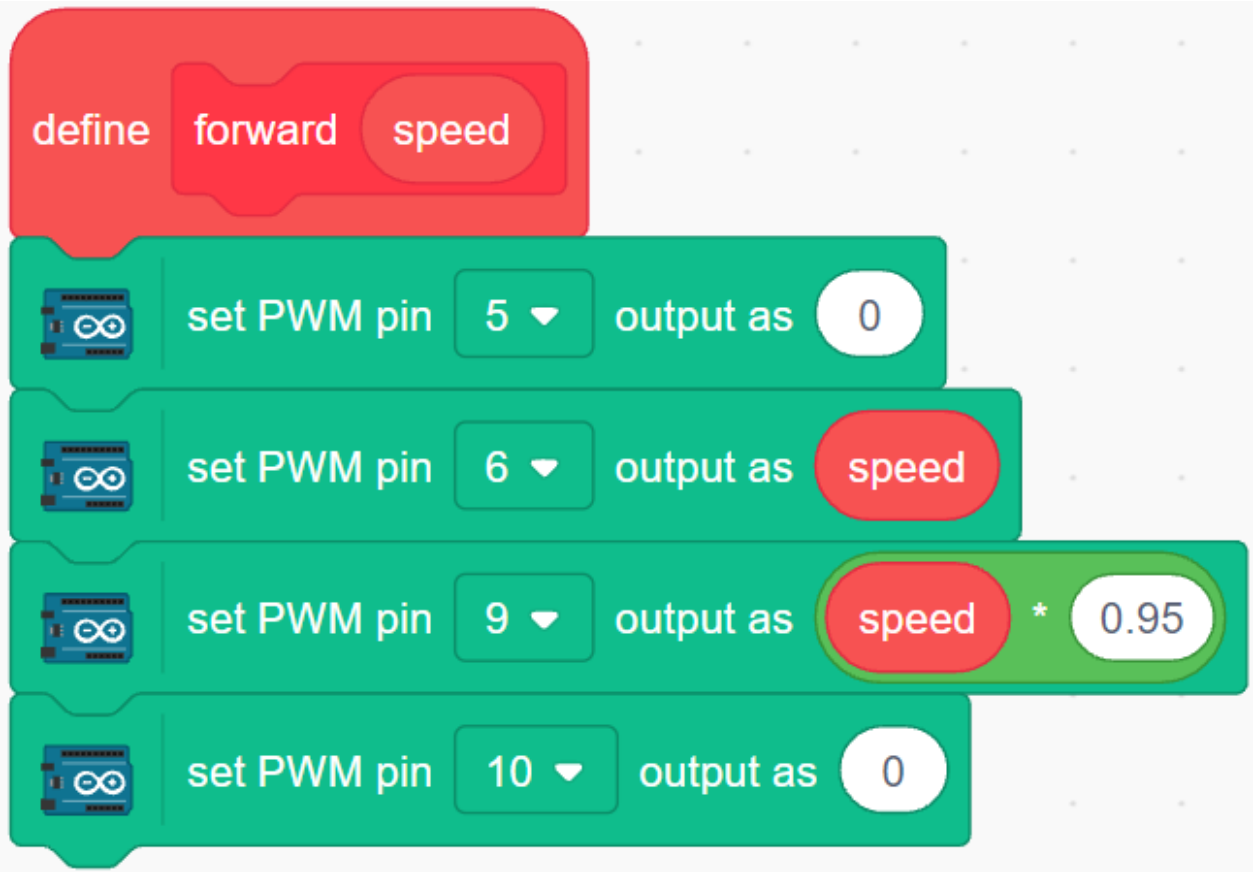
| ENB | IN3 | IN4 | **左モータ (B) の状態** |
|---|---|---|---|
| 0 | X | X | 停止 |
| 1 | 0 | 0 | ブレーキ |
| 1 | 0 | 1 | 時計回りに回転 |
| 1 | 1 | 0 | 反時計回りに回転 |
| 1 | 1 | 1 | ブレーキ |

### 7.27.1 プログラミング

次に、車が前進、後退、左右回転、及び停止するためのブロックを作成します。

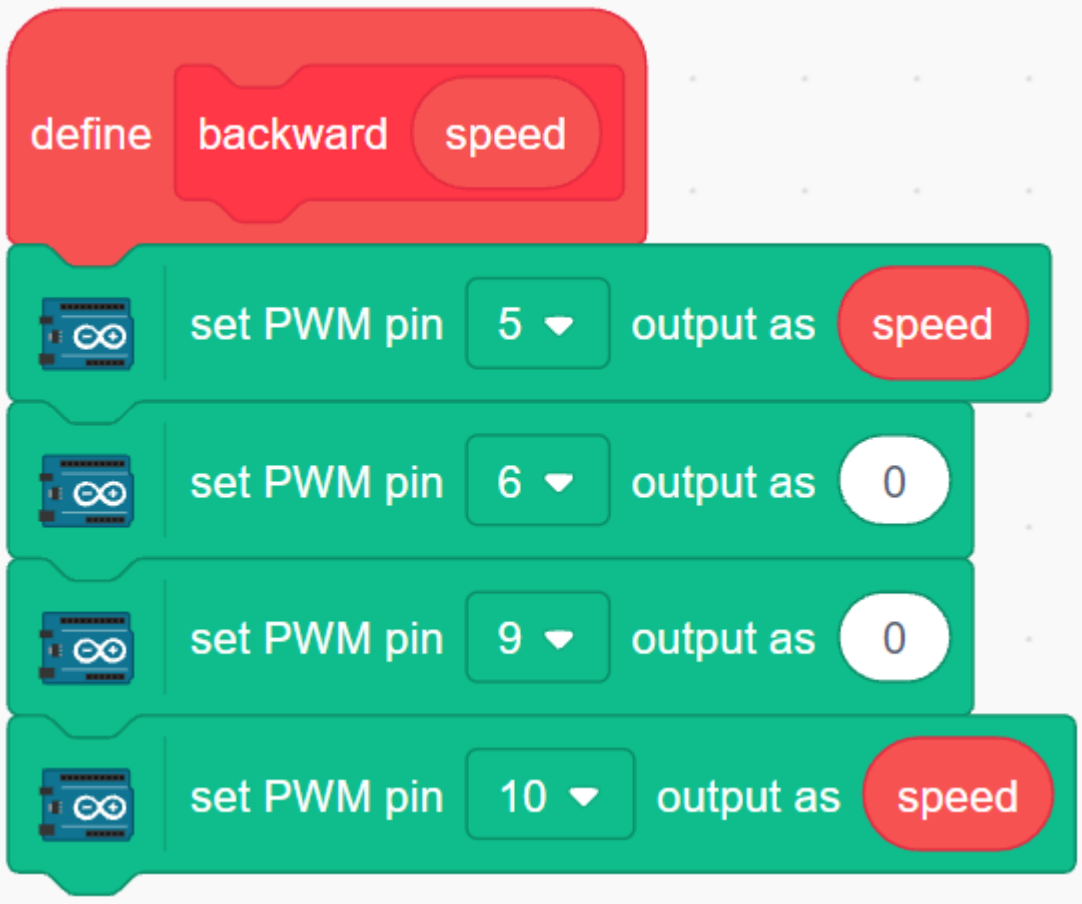
**1. 前進**

右モータは時計回り、左モータは反時計回りに回転して車を前進させます。



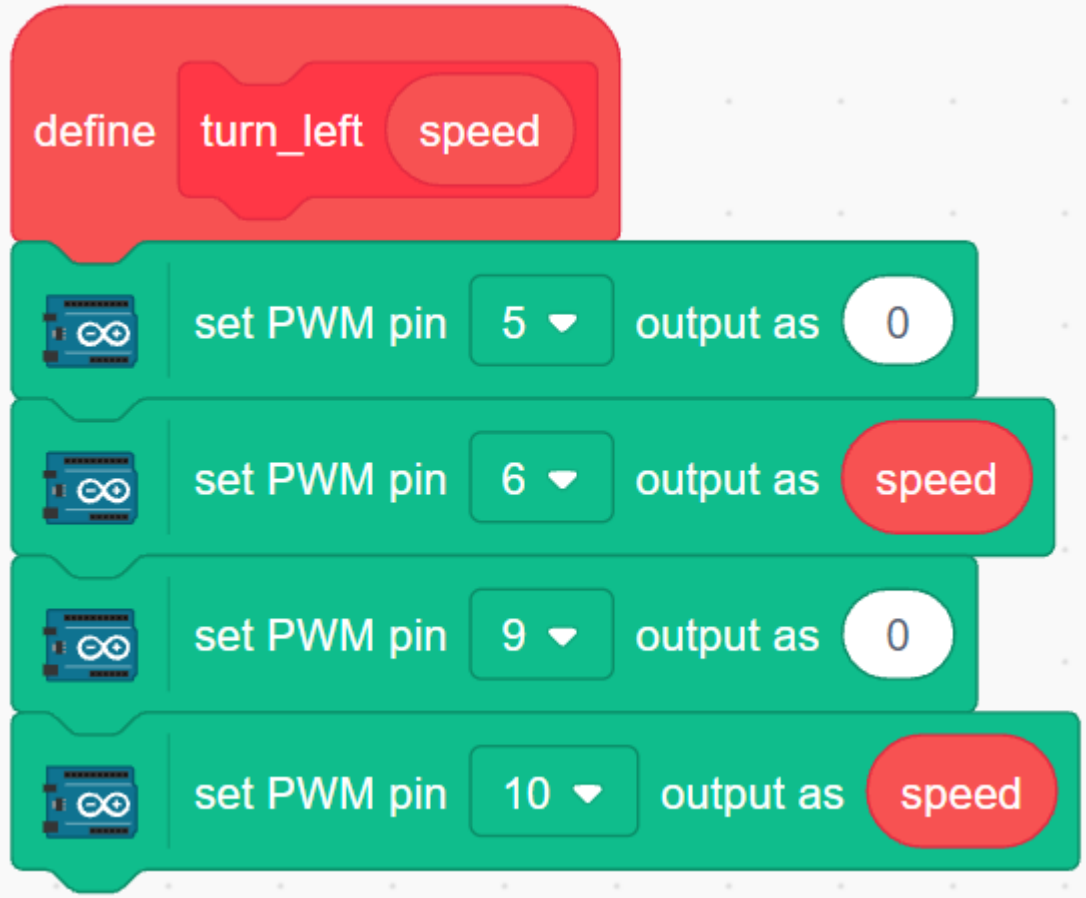

**2. 後退**

後退はその逆で、右モータは反時計回り、左モータは時計回りに回転が必要です。



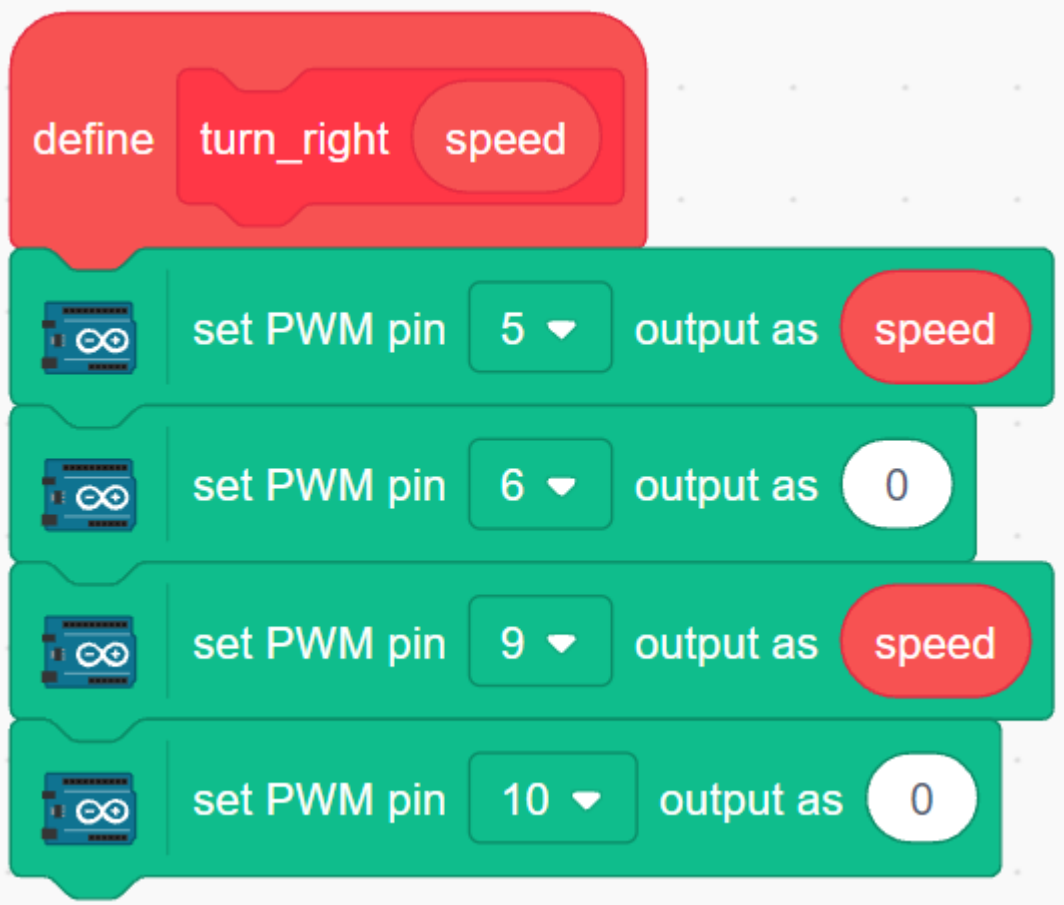

**3. 左に曲がる**

左右のモータを同時に時計回りに回転させて車を左に曲がらせます。



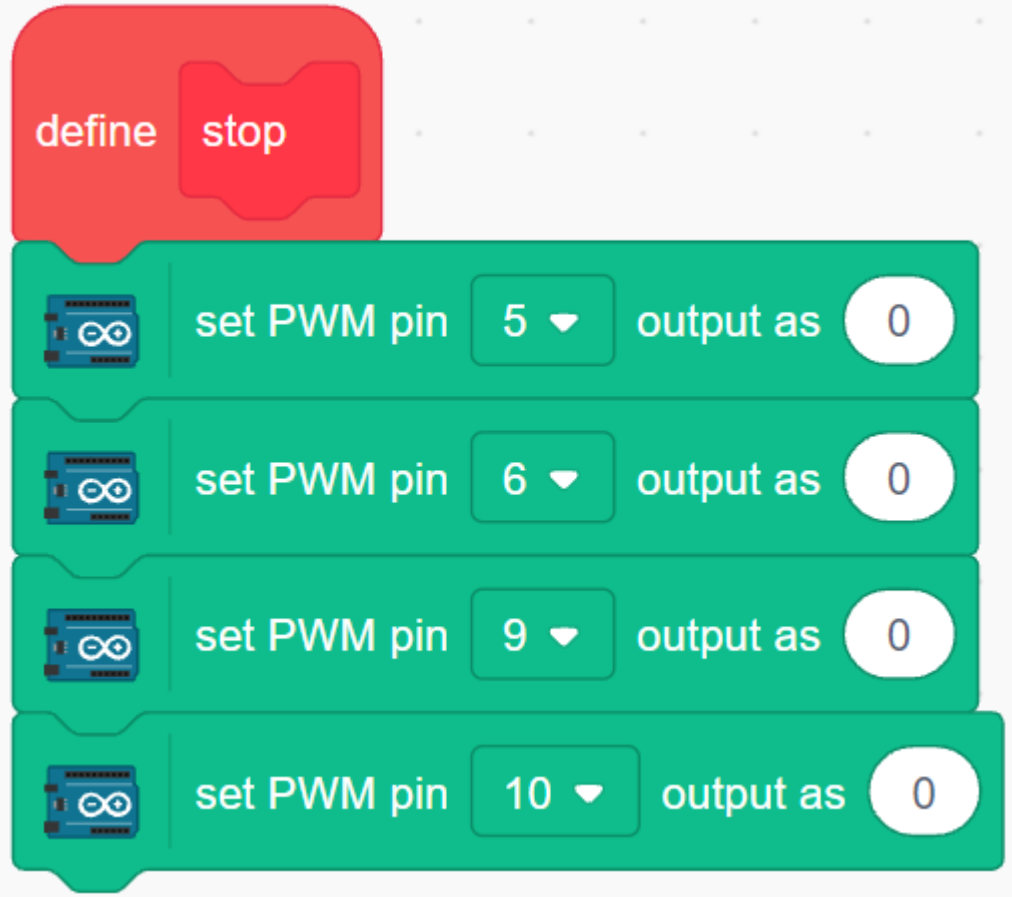

**4. 右に曲がる**

同様に、左右のモータを反時計回りに回転させて車を右に曲がらせます。



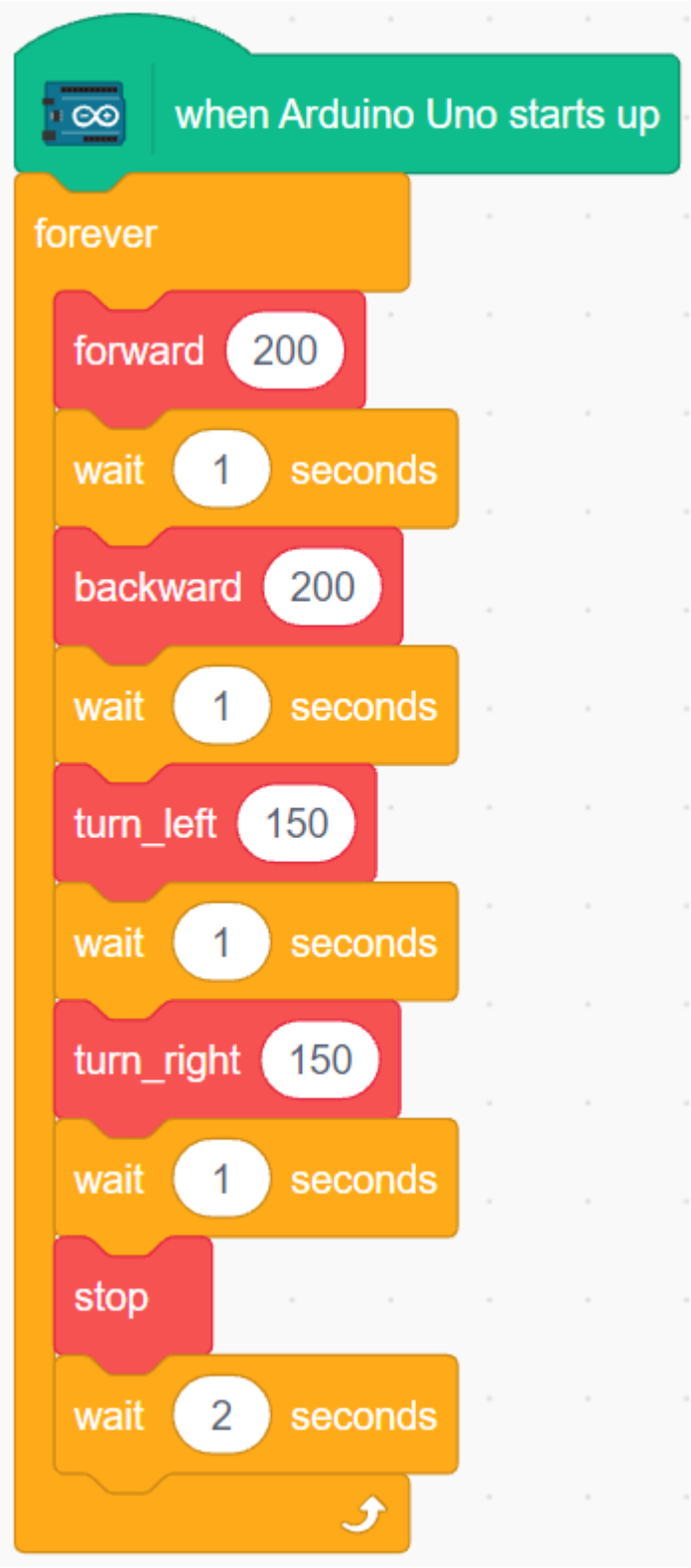

**5. 停止**

全モータの設定を 0 にして車を停止させます。




**6. 車を動かす**

車を前進、後退、左右に 1 秒間動かしてから停止します。すべてのブロックが [Forever] ブロック内に配置されているので、車が上記の動作を繰り返すのを見ることができます。


when Arduino Uno starts up
forever
forward 200
wait 1 seconds
backward 200
wait 1 seconds
turn_left 150
wait 1 seconds
turn_right 150
wait 1 seconds
stop
wait 2 seconds

## 7.28 3.3 黒い線を追う

この車にはライン追跡モジュールが搭載されており、車が黒い線を追跡するようにすることができます。

プロジェクトを開始する前に、黒い線テープを使ってカーブマップを作成する必要があります。推奨されるラインの幅は 0.8-1.5cm で、曲がり角は 90 度未満であってはなりません。

### 7.28.1 必要な部品

このプロジェクトでは、以下の部品が必要です。

全体のキットを購入すると非常に便利です。リンクは以下の通りです。

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することも可能です。

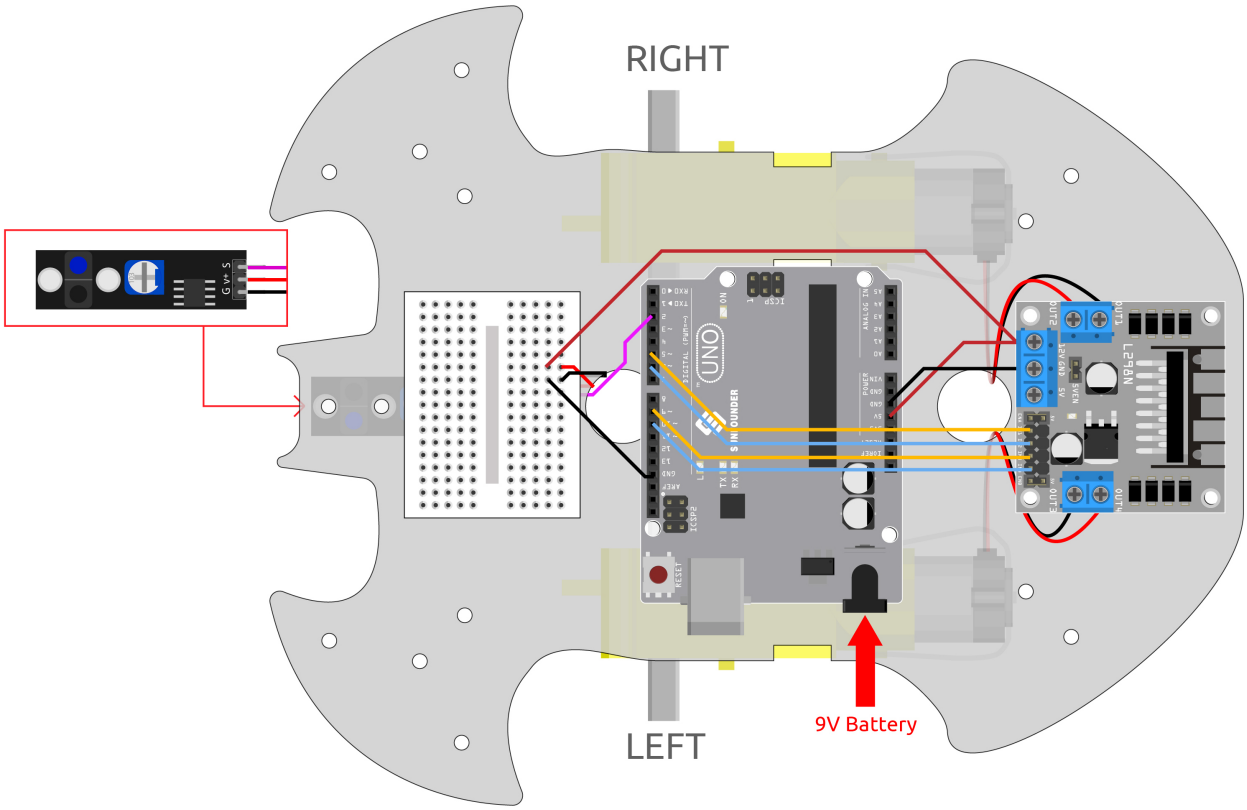
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| ライン追跡モジュール | |

### 7.28.2 回路の作成

これはデジタルライン追跡モジュールで、黒い線を検出すると 1 を出力し、白い線を検出すると 0 の値を出力します。さらに、モジュール上のポテンショメータを通じて感知距離を調整することができます。

以下の図に従って回路を組み立ててください。

| ライン追跡モジュール | R3 ボード |
|---|---|
| S | 2 |
| V+ | 5V |
| G | GND |




### 7.28.3 モジュールの調整

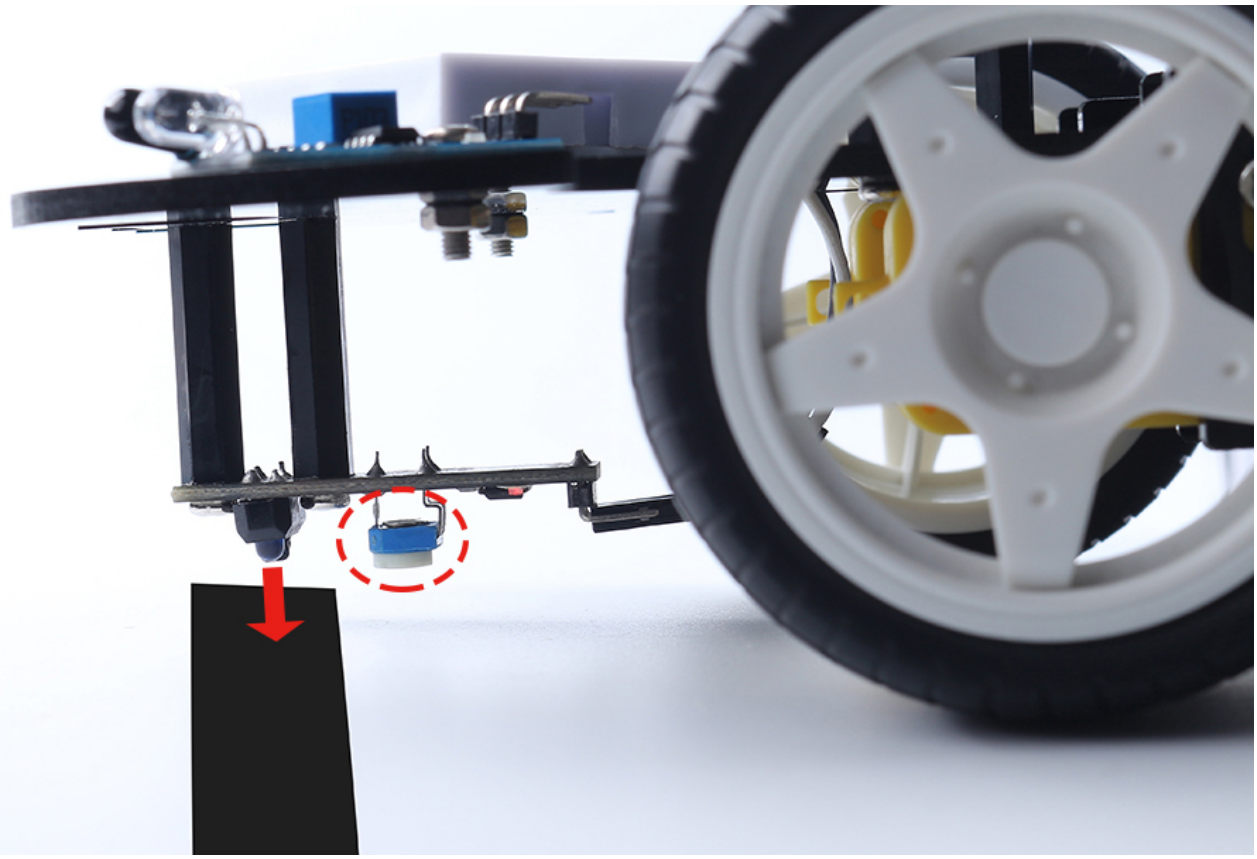
プロジェクトを開始する前に、モジュールの感度を調整する必要があります。

上記の図に従って配線し、R3 ボードを電源に接続してください（USB ケーブルまたは 9V の電池ボタンケーブルで直接接続）。コードをアップロードすることなく電源を入れます。

テーブルに黒い電気テープを貼り、カートをその上に置きます。

モジュールのシグナル LED を観察して、白いテーブル上で点灯し、黒いテープ上で消灯することを確認してください。

そうでない場合は、モジュール上のポテンショメータを調整して、上記の効果が得られるようにします。

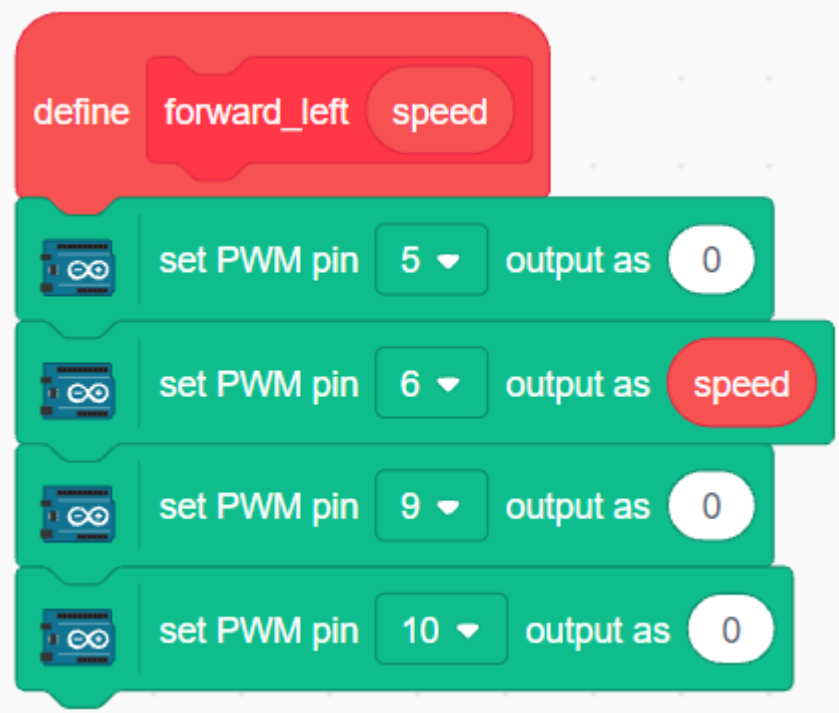


### 7.28.4 プログラミング

左前または右前に車を動かす 2 つのブロックを作成します。

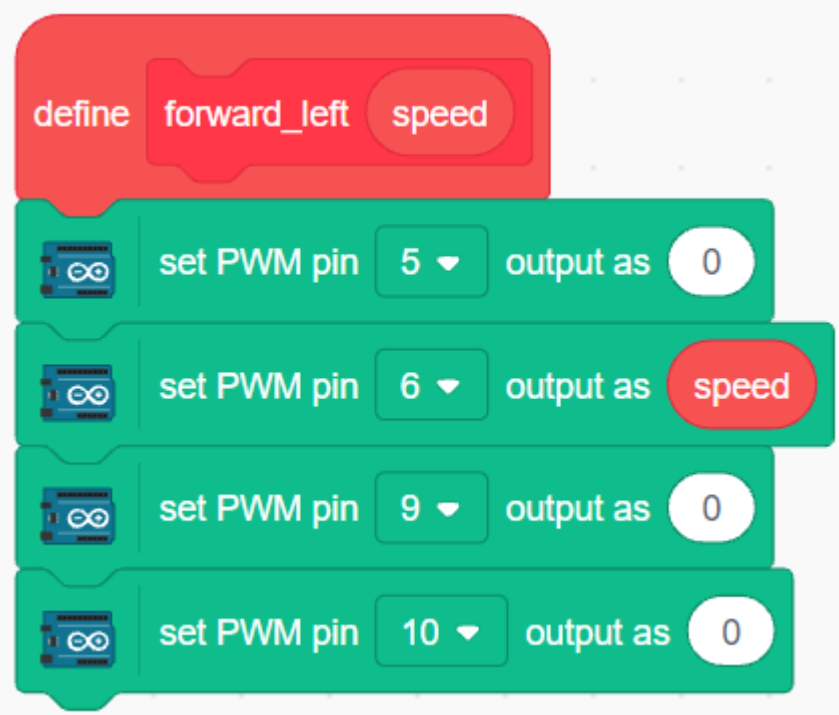
**1. 左前方に移動**

右のモーターが時計回りに回転し、左のモーターが動かないと、車は左前方にわずかに移動します。




**2. 右前方への移動**

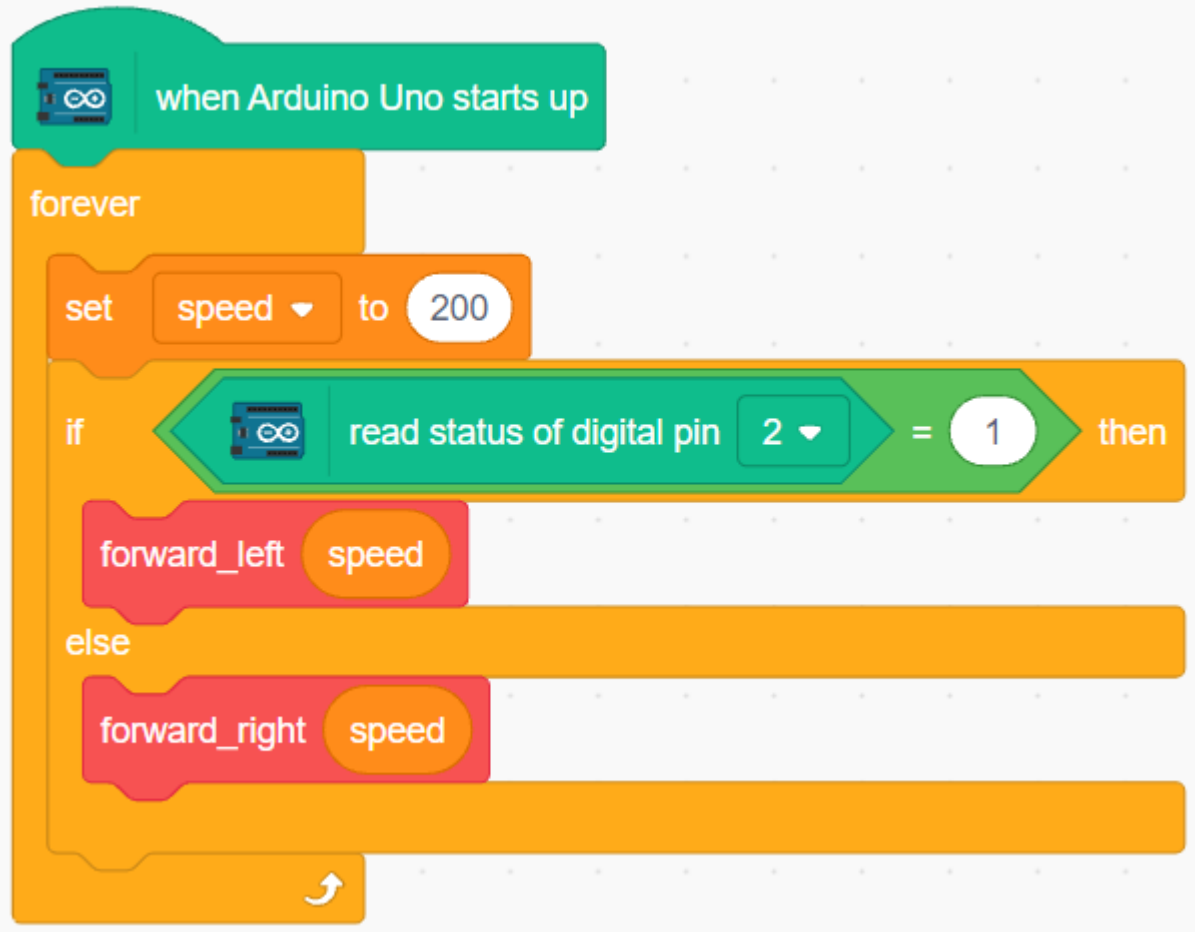
同様に、左モータが反時計回りに回転し、左モータが動かないと、車は右方向にわずかに移動します。




**3. ライン追跡**

ライン追跡モジュールの値を読み取り、1 の場合は黒い線が検出されたことを意味するので、車を左に進めます。

そうでなければ、右方向に進みます。




R3 ボードにコードをアップロードした後、ライン追跡モジュールを車の下の黒い線と一致させると、車が線を追跡するのを見ることができます。

## 7.29 3.4 手を追う

この車をペットと考えてみてください。手を振ると、車はあなたの方へ走ってきます。

### 7.29.1 必要な部品

このプロジェクトには、以下の部品が必要です。

全体のキットを購入すると非常に便利です。リンクは以下の通りです。

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することも可能です。

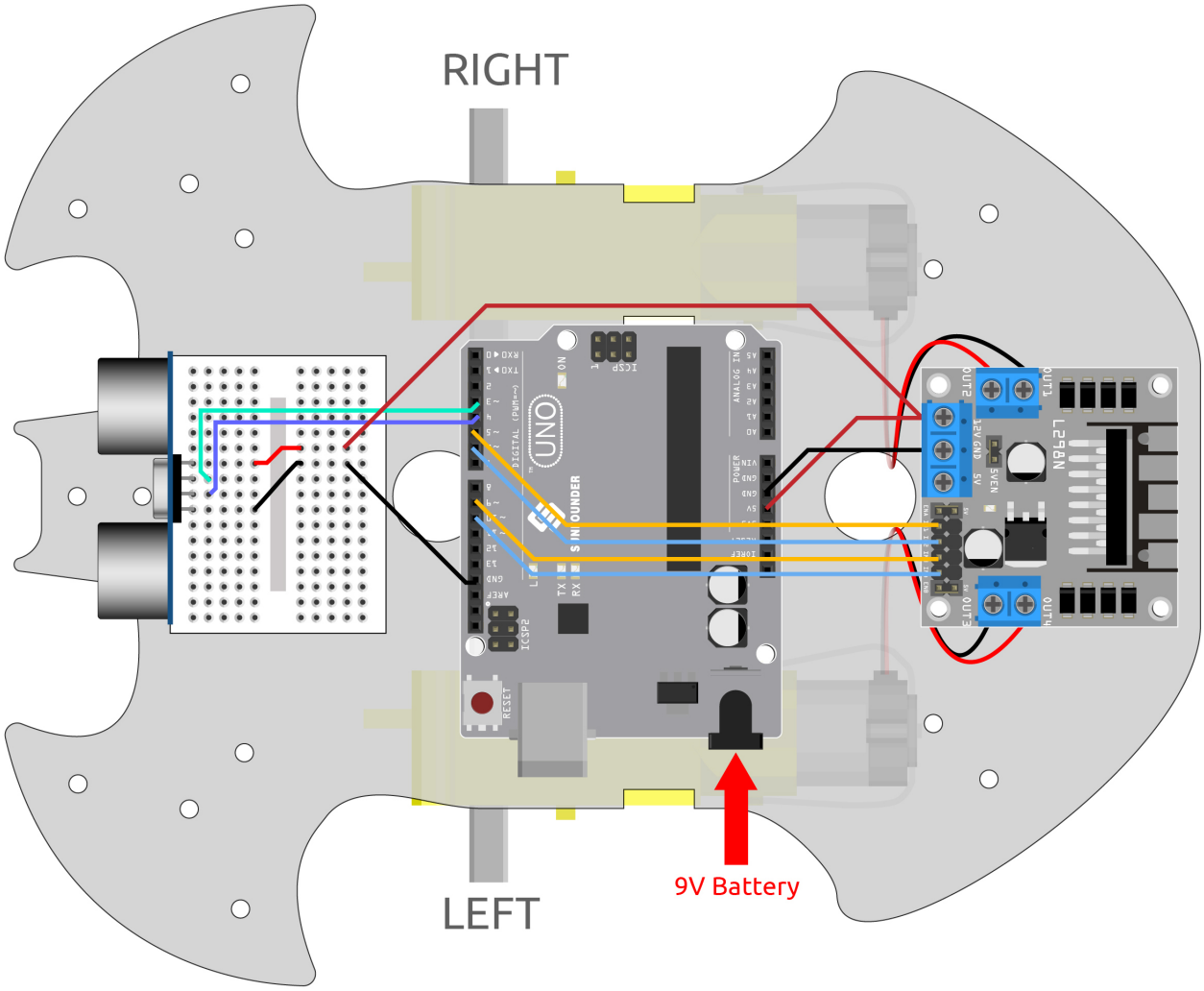
| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 超音波モジュール | |

### 7.29.2 回路の作成

超音波センサーモジュールは、超音波を使用して物体までの距離を測定する道具です。2 つのプローブがあります。1 つは超音波を送るためのもので、もう 1 つは波を受信して送受信の時間を距離に変換し、デバイスと障害物との距離を検出するためのものです。

以下の図に従って回路を組み立ててください。

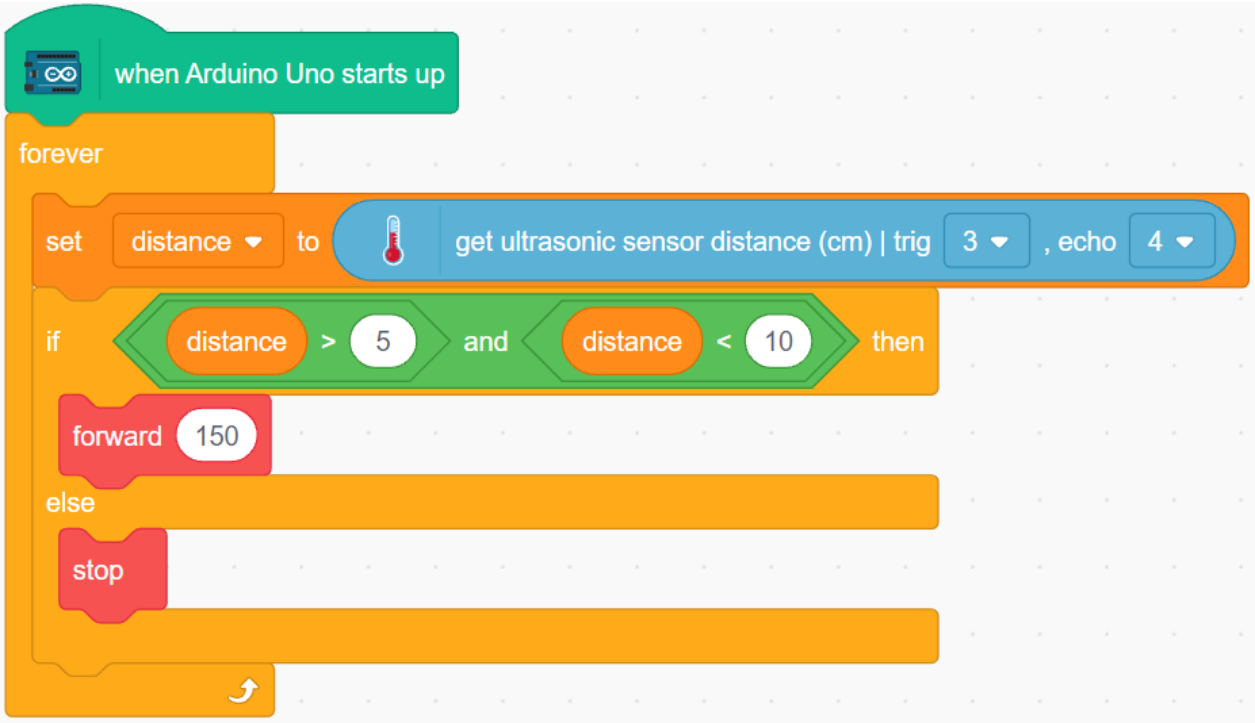
| 超音波モジュール | R3 ボード |
|---|---|
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |




### 7.29.3 プログラミング

車が前進し停止するブロックを作成します。


車の前に手をかざして、超音波モジュールの値を読み取ります。手の検出距離が 5-10cm の場合、車を前進させ、それ以外の場合は停止させます。




## 7.30 3.5 障害物回避

車の前部には 2 つの赤外線障害物回避モジュールが取り付けられており、近くの障害物を検出するのに使用できます。

このプロジェクトでは、車は自由に前進し、障害物に遭遇するとそれを避けて他の方向に移動することができます。

### 7.30.1 必要な部品

このプロジェクトには、以下の部品が必要です。

全体のキットを購入することは非常に便利です。以下がリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
| --- | --- |
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 障害物回避モジュール | |

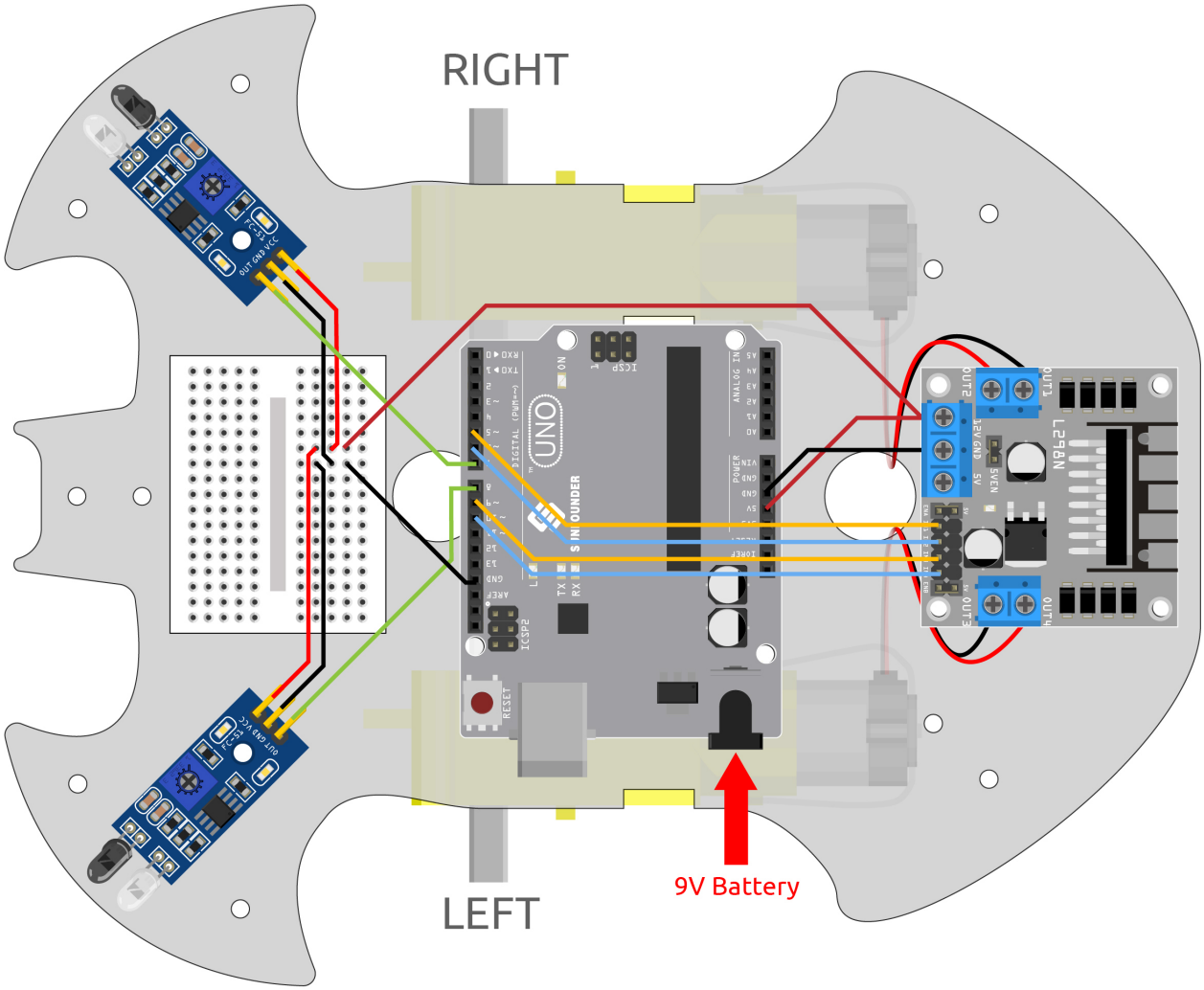
### 7.30.2 回路の作成

障害物回避モジュールは、距離調整可能な赤外線近接センサーで、通常は出力が高く、障害物を検出すると低くなります。

以下の図に従って回路を組み立ててください。

| 左 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 8 |
| GND | GND |
| VCC | 5V |

| 右 IR モジュール | R3 ボード |
| --- | --- |
| OUT | 7 |
| GND | GND |
| VCC | 5V |




### 7.30.3 モジュールの調整

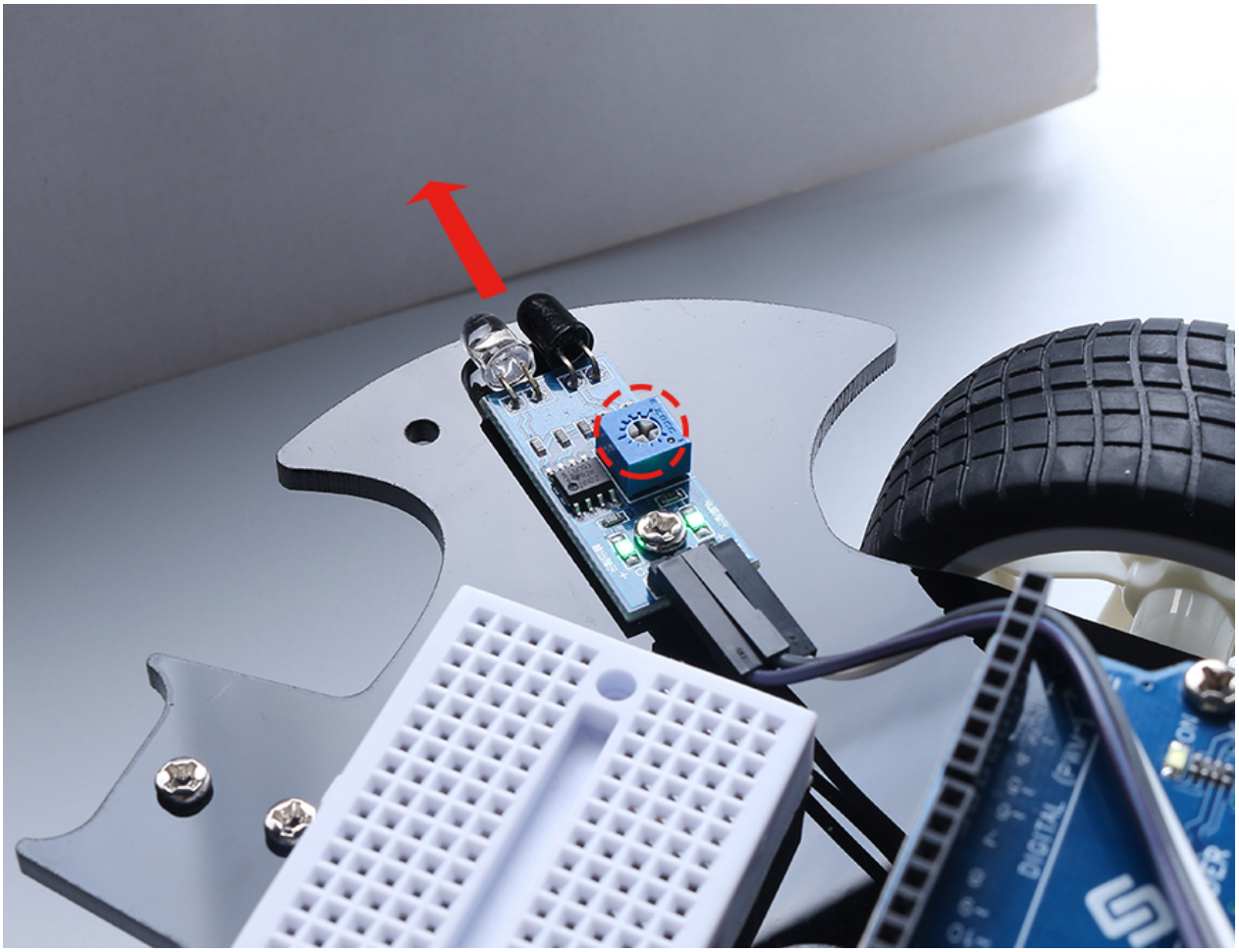
プロジェクトを開始する前に、モジュールの検出距離を調整する必要があります。

上記の図に従って配線し、R3 ボードに電源を供給します（USB ケーブルを直接挿入するか、9V のバッテリーケーブルを取り付ける）。コードをアップロードせずに電源を入れます。

赤外線障害物回避の前に約 5cm のノートや他の平らな物を置きます。

その後、モジュール上のポテンショメータを回して、モジュール上の信号インジケータがちょうど点灯するように、最大検出距離 5cm に調整します。

もう一つの赤外線モジュールも同じ方法で調整します。



### 7.30.4 プログラミング

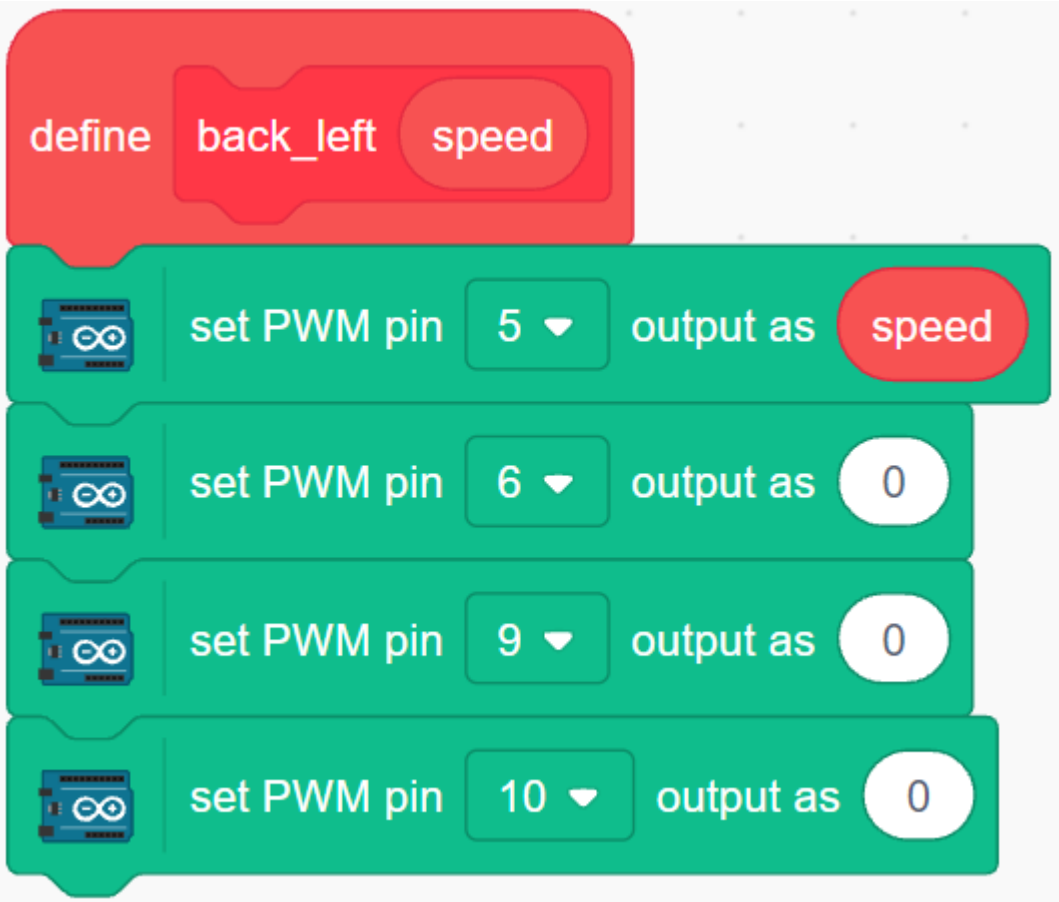
達成したい効果：

- 左の IR モジュールが障害物を検出すると、車は左に後退します。
- 右の IR モジュールが障害物を検出すると、車は右に後退します。
- 両方の IR モジュールが障害物を検出すると、車は直接後退します。
- それ以外の場合、車は前進します。

対応するブロックを作成してください。

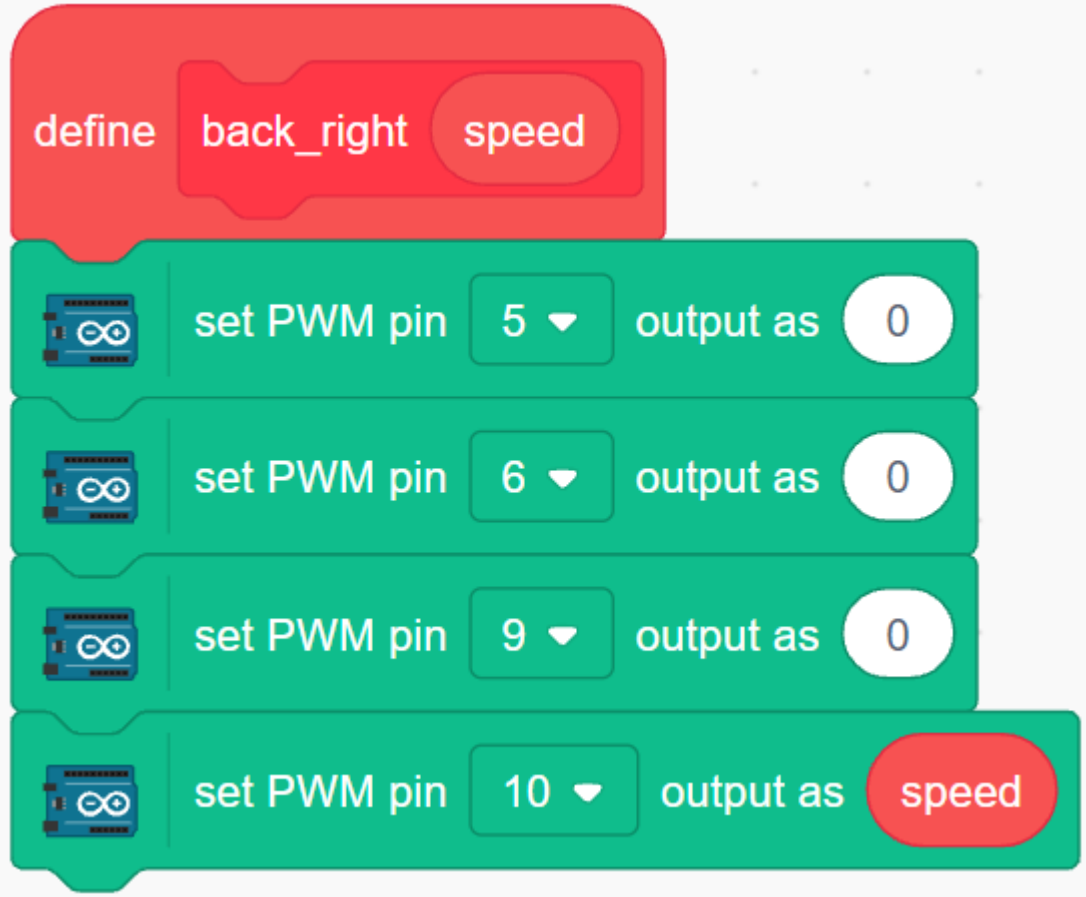
**1. 車が左に後退する**

右のモータが反時計回りに回転し、左のモータが回転しない場合、車は左に後退します。

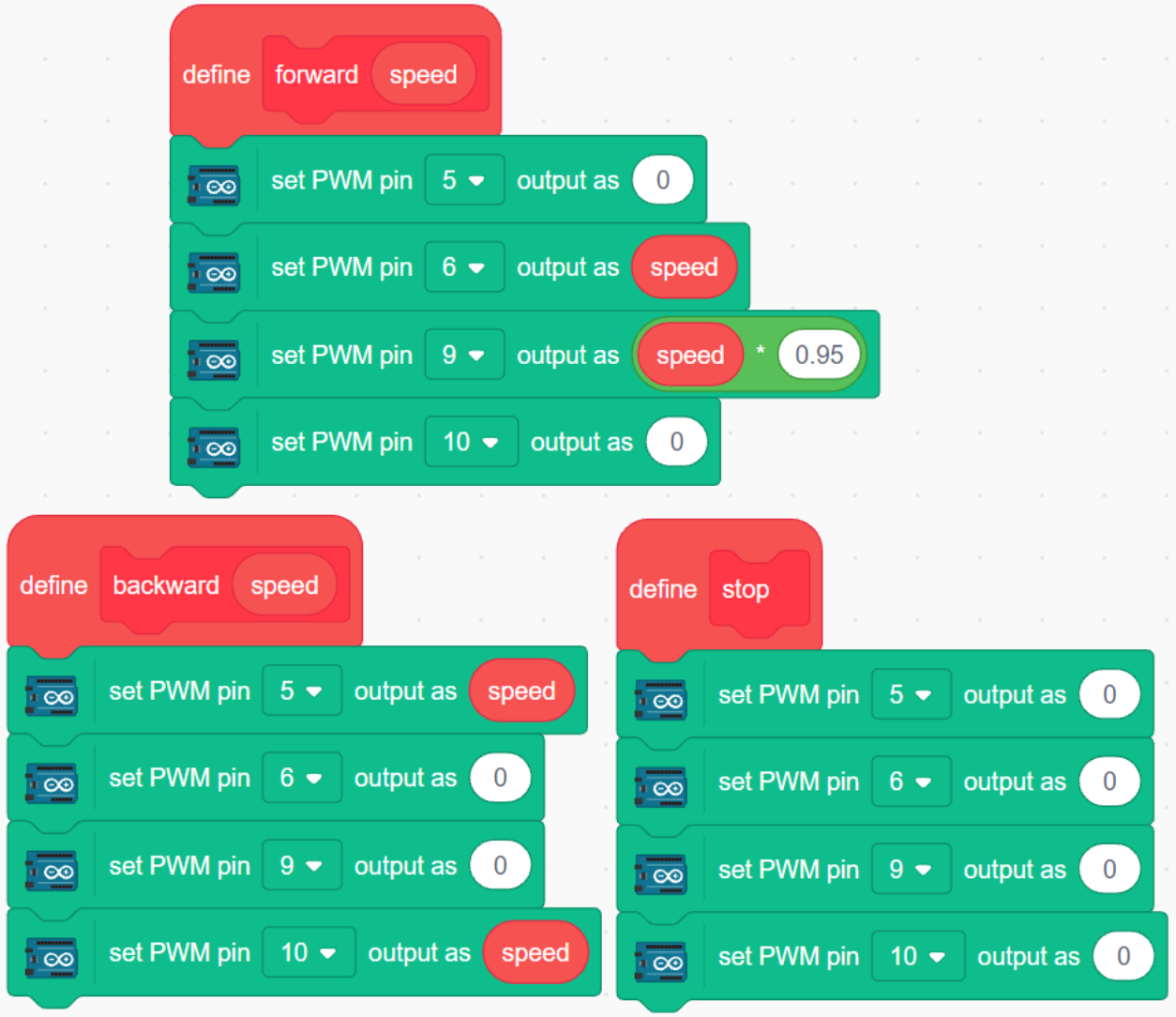



**2. 車が右に後退する**

左のモータが時計回りに回転し、右のモータが回転しない場合、車は右に後退します。

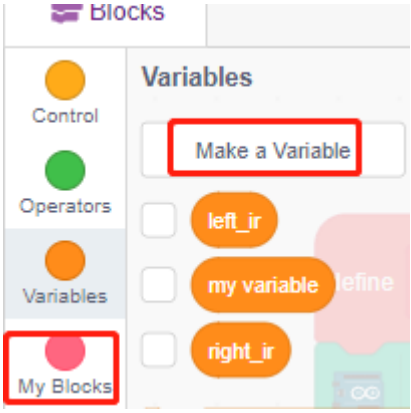



3. 車は前進、後退し、停止します

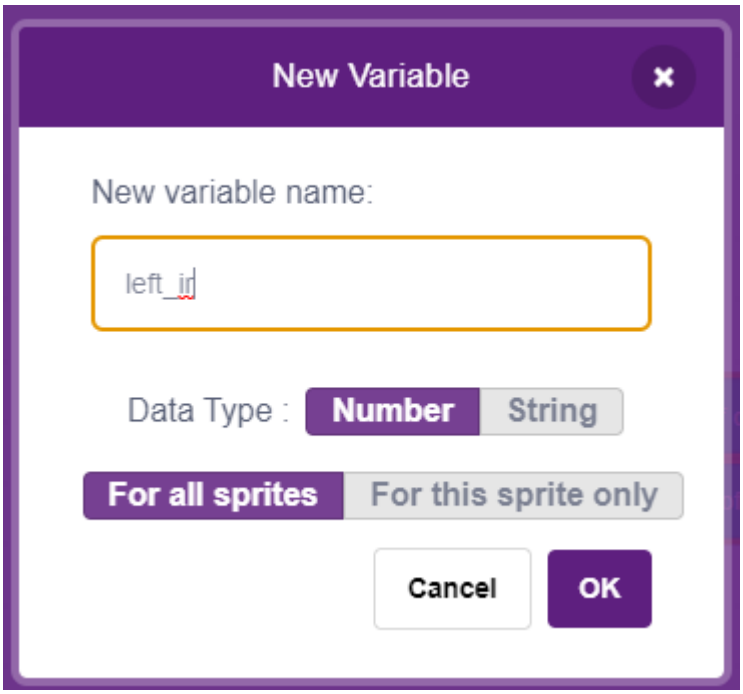



**4. 2 つの IR モジュールの値を読み取る**

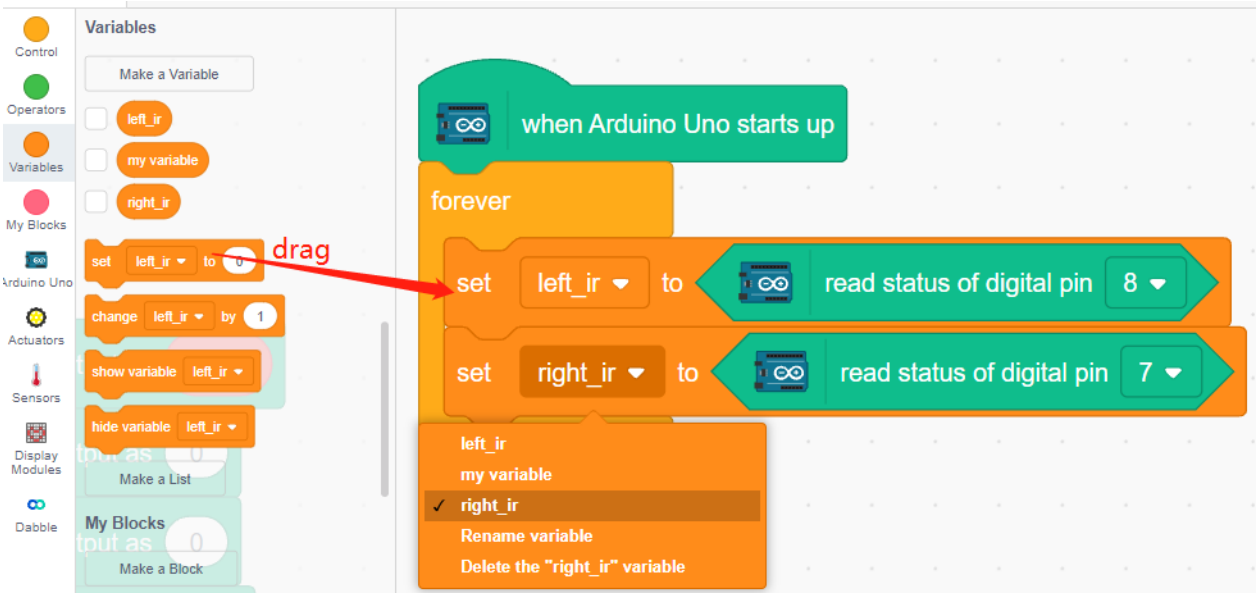
**Variables** パレットで **Make a variable** をクリックします。




変数名を入力し、 **OK** をクリックして新しい変数を作成します。

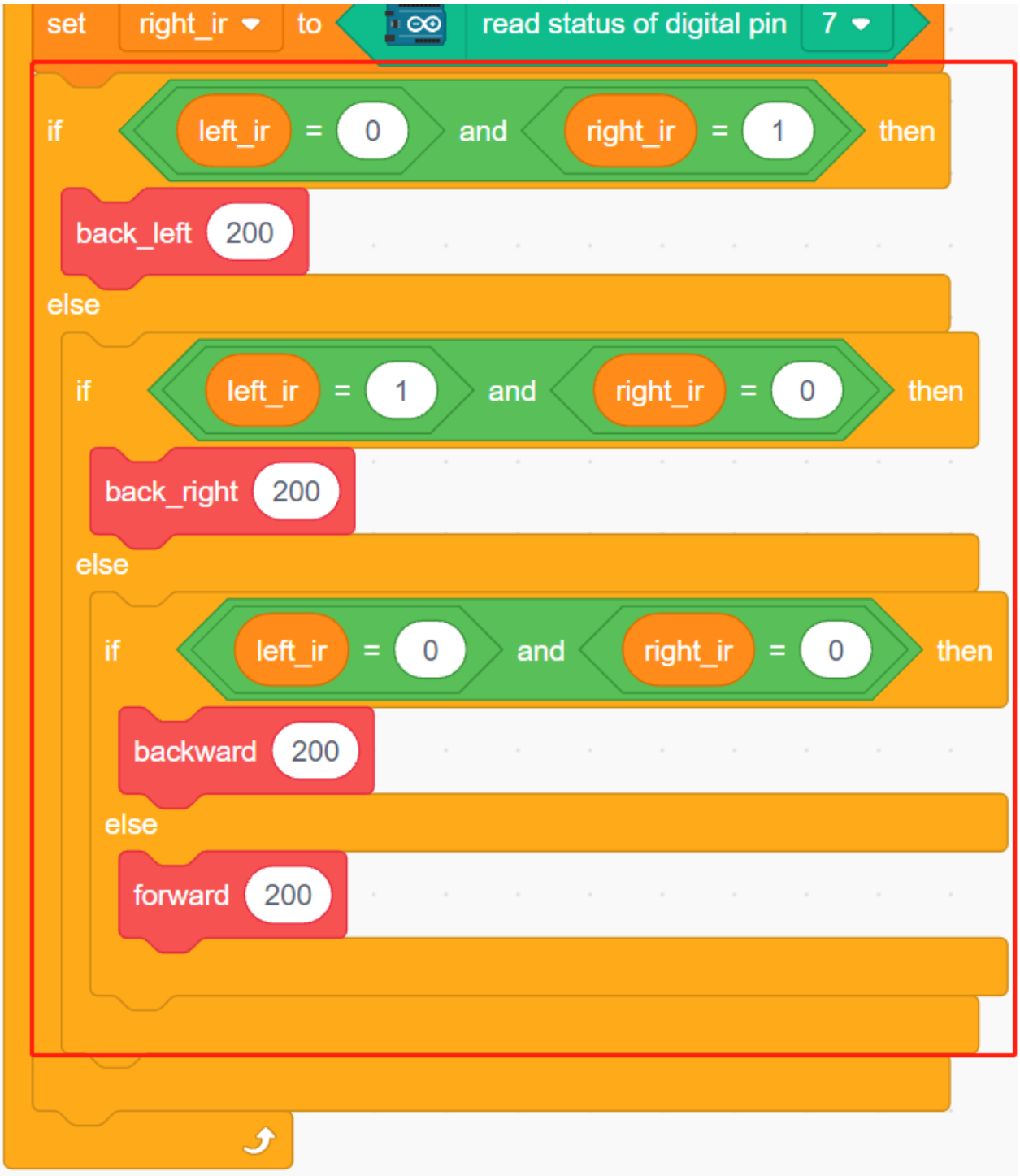



左および右の IR 障害物回避モジュールの値を読み取り、2 つの新しい変数に保存します。




**5. 障害物を避ける**

- 左の IR モジュールが 0（障害物検出）で、右の IR モジュールが 1 の場合、車は左にバックアップします。
- 右の IR モジュールが 0（障害物検出）の場合、車は右にバックアップします。
- 2 つの IR モジュールが同時に障害物を検出すると、車は後退します。
- それ以外の場合、車は前進を続けます。


set right_ir to read status of digital pin 7
if left_ir = 0 and right_ir = 1 then
back_left 200
else
if left_ir = 1 and right_ir = 0 then
back_right 200
else
if left_ir = 0 and right_ir = 0 then
backward 200
else
forward 200

## 7.31 3.6 あなたの手を追跡する 2

*3.4* **手を追う** のプロジェクトでは、超音波モジュールのみが使用されており、前方の手だけを追跡することができます。

このプロジェクトでは、2 つの IR 障害物回避モジュールを同時に使用することで、車があなたの手を左または右に追跡できるようになります。

### 7.31.1 必要な部品

このプロジェクトに必要な部品は以下のとおりです。

一式のキットを購入することは非常に便利です。以下がリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* **ボード** | |
| *L298N* **モジュール** | |
| *TT* **モーター** | - |
| **超音波モジュール** | |
| **障害物回避モジュール** | |

### 7.31.2 回路の作成

超音波モジュールと 2 つの IR 障害物回避モジュールを同時に接続します。

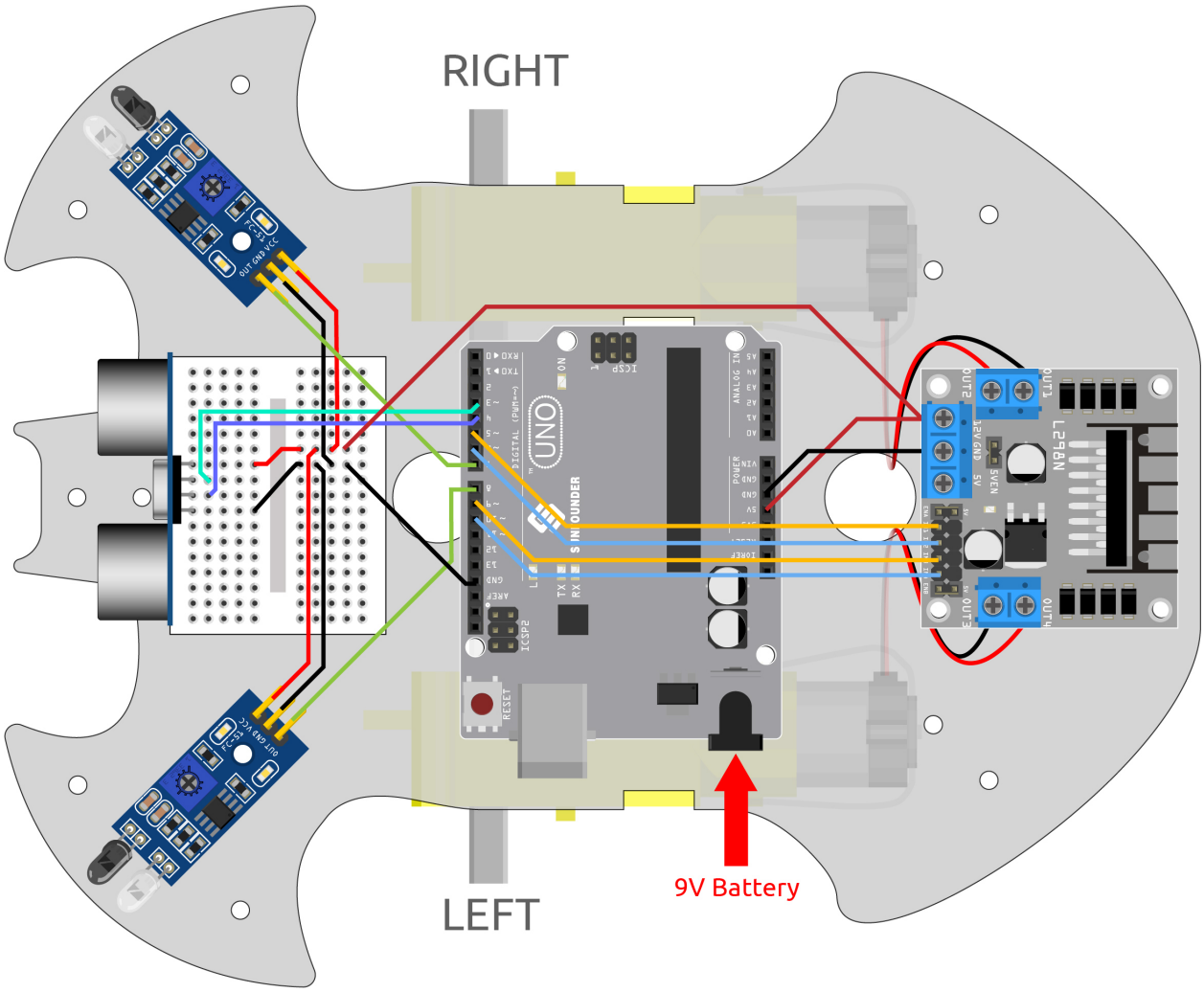
超音波モジュールと R3 ボード間の配線は以下の通りです。

| 超音波モジュール | R3 ボード |
|---|---|
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |

2 つの IR 障害物回避モジュールと R3 ボード間の配線は以下の通りです。

| 左 IR モジュール | R3 ボード |
|---|---|
| OUT | 8 |
| GND | GND |
| VCC | 5V |

| 右 IR モジュール | R3 ボード |
|---|---|
| OUT | 7 |
| GND | GND |
| VCC | 5V |



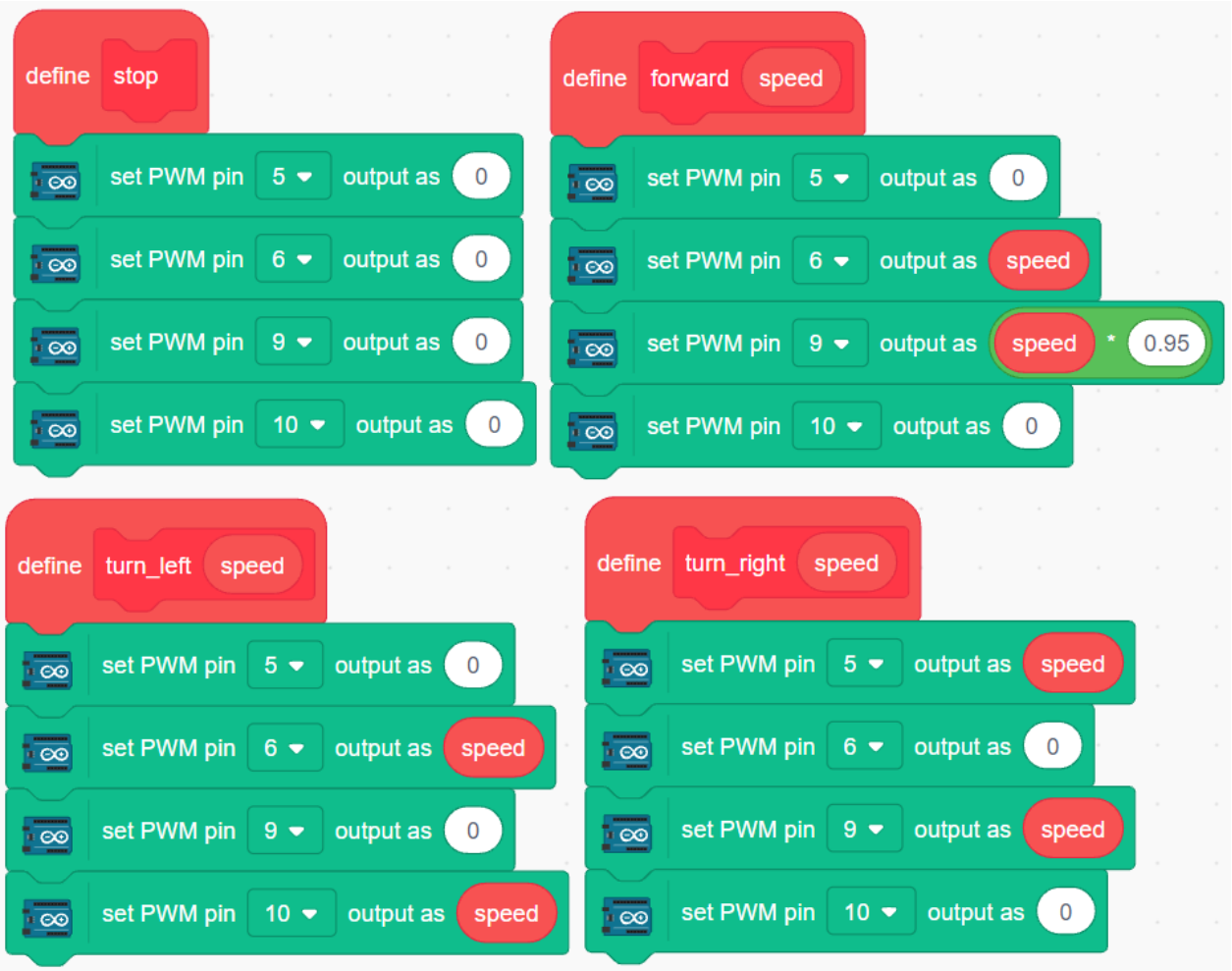

### 7.31.3 プログラミング

このプロジェクトで達成したい効果は以下の通りです

- 超音波は、前方で 5-10cm の距離で手を検出し、車を追跡させます。
- 左の赤外線モジュールが手を検出すると、左に曲がります。
- 右の IR モジュールが手を検出すると、右に曲がります。

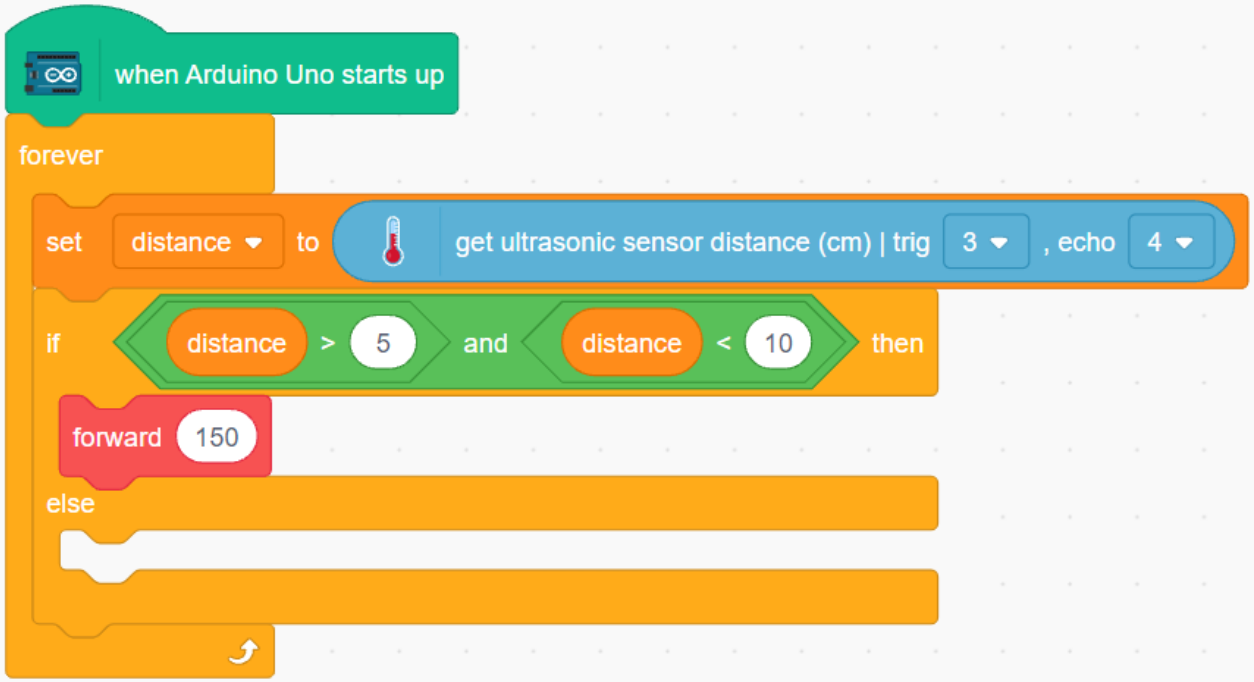
**1. ブロックを作成する**

前進、左折、右折、停止するためのブロックを作成します。




**2. 前進するための追跡**

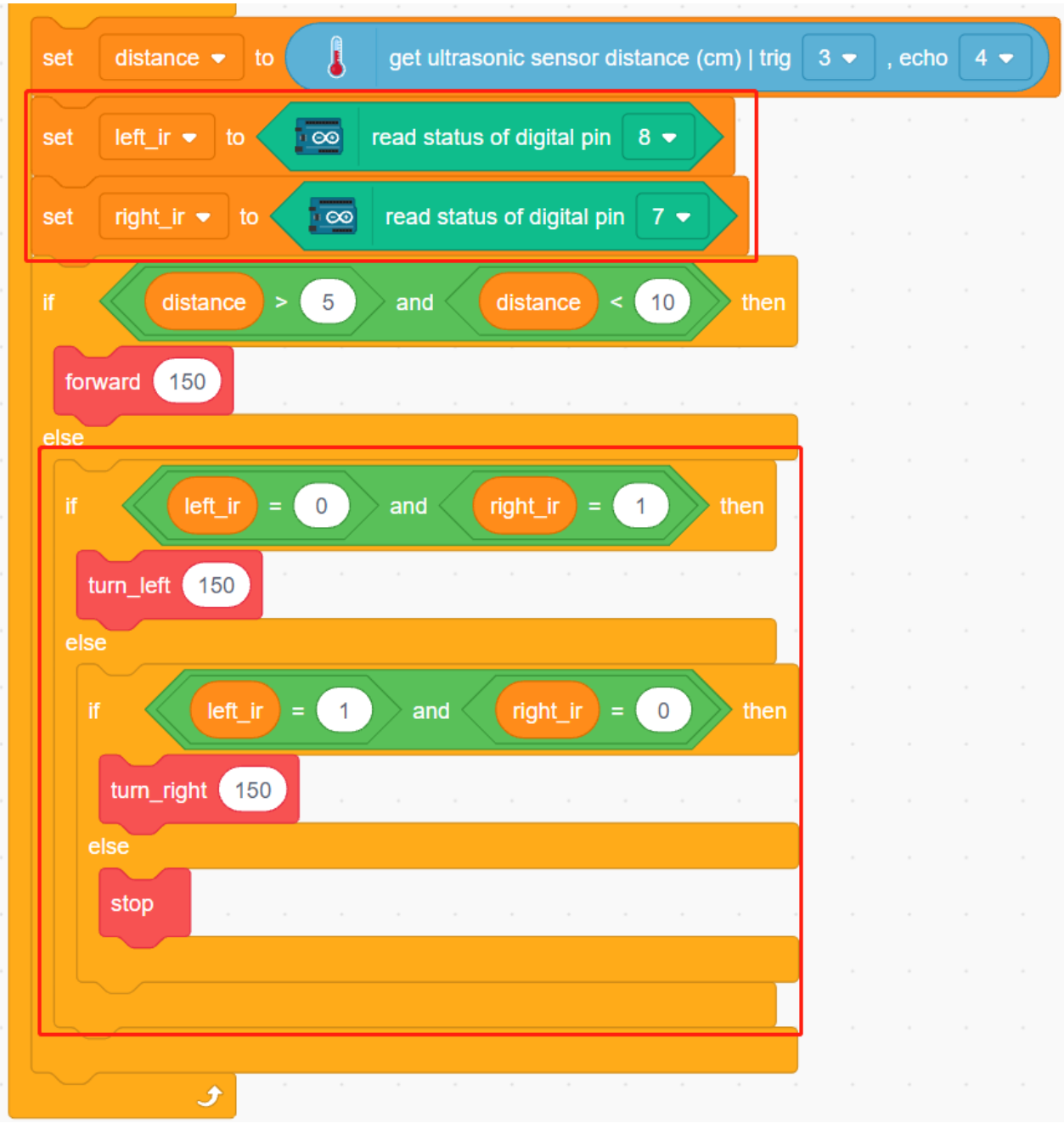
超音波の値を読み取り、手が 5-10cm の距離で検出された場合、車を追跡させます。




### 3. 左と右に追跡する

左と右の IR モジュールの値を読み取ります。

- 左の IR モジュールが手を検出した場合、左に曲がります。
- 右の IR モジュールが手を検出した場合、右に曲がります。
- 両方の IR モジュールと超音波モジュールが手を検出しない場合、車を停止させます。




## 7.32 3.7 障害物回避 2

*3.5* **障害物回避** プロジェクトでは、2 つの IR 障害物回避モジュールのみが障害物回避のために使用されていましたが、IR 障害物回避モジュールの検出距離は短く、車が障害物を避けるのが遅すぎることがあります。

このプロジェクトでは、遠距離検出のために超音波モジュールも追加します。これにより、車はより遠くの障害物を感知して判断を下すことができます。

### 7.32.1 必要な部品

このプロジェクトには、以下の部品が必要です。

キット全体を購入すると確実に便利です。こちらがリンクです：

| 名前 | このキットのアイテム | リンク |
|---|---|---|
| 3 in 1 Starter Kit | 380+ | |

以下のリンクから個別に購入することもできます。

| コンポーネントの紹介 | 購入リンク |
|---|---|
| *SunFounder R3* ボード | |
| *L298N* モジュール | |
| *TT* モーター | - |
| 超音波モジュール | |
| 障害物回避モジュール | |

### 7.32.2 回路の作成

超音波モジュールと 2 つの IR 障害物回避モジュールを同時に接続します。

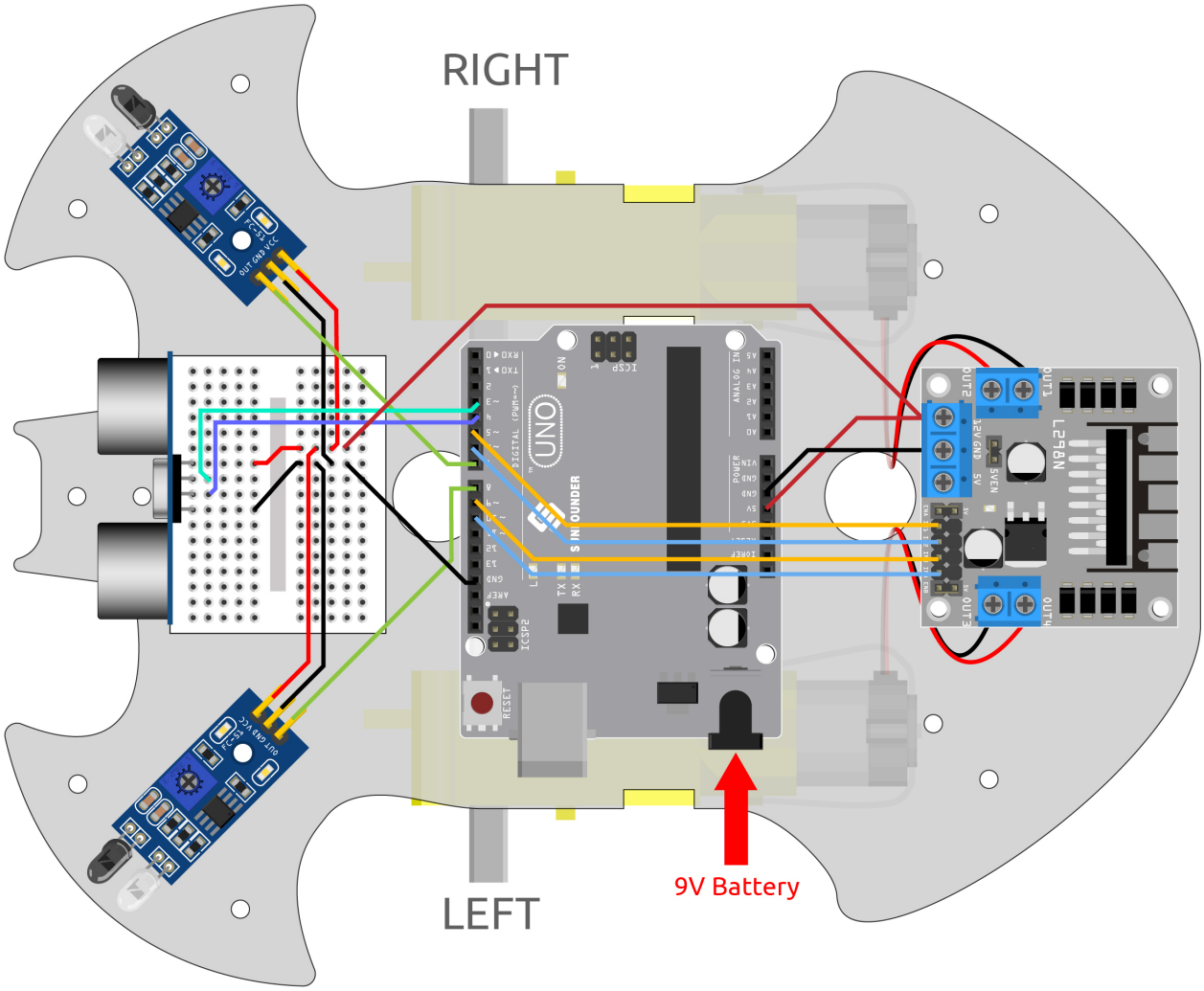
超音波を R3 ボードに次のように接続します。

| 超音波モジュール | R3 ボード |
|---|---|
| Vcc | 5V |
| Trig | 3 |
| Echo | 4 |
| Gnd | GND |

2 つの IR 障害物回避モジュールの R3 ボードへの配線は次のとおりです。

| 左 IR モジュール | R3 ボード |
|---|---|
| OUT | 8 |
| GND | GND |
| VCC | 5V |

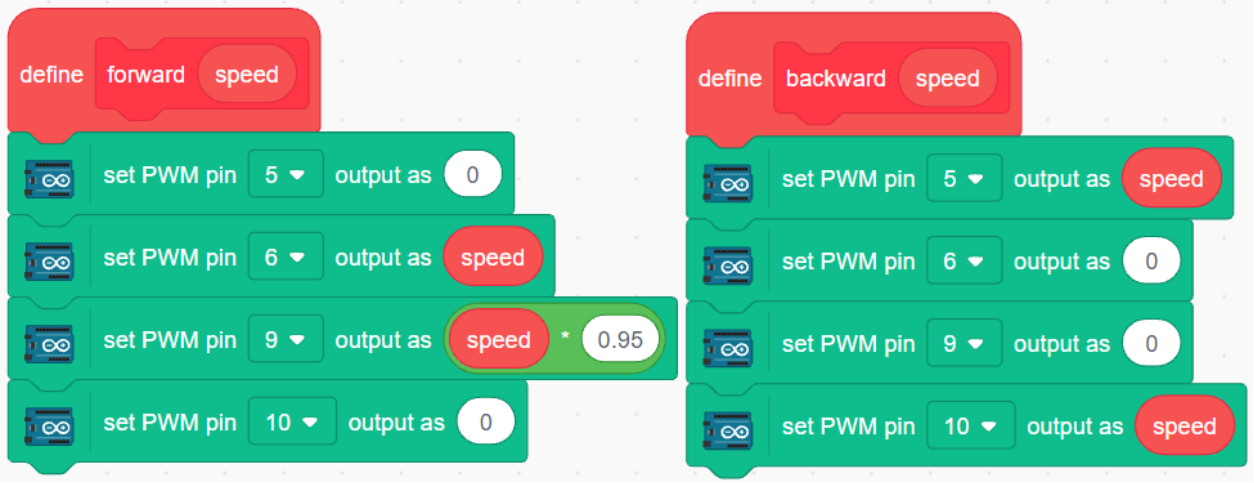
| 右 IR モジュール | R3 ボード |
|---|---|
| OUT | 7 |
| GND | GND |
| VCC | 5V |



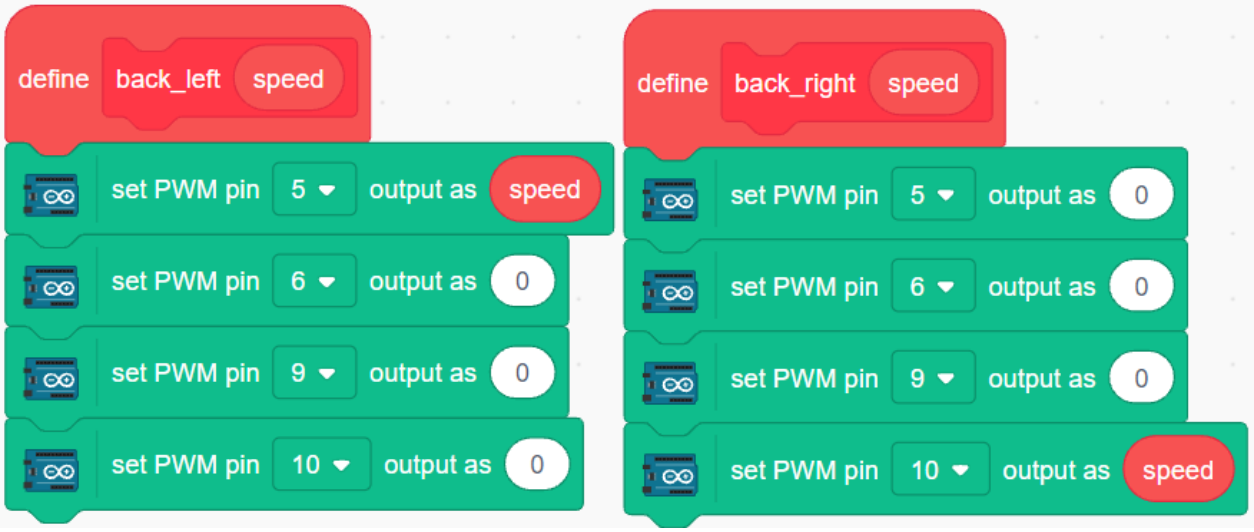
### 7.32.3 プログラミング

**1. 関数を作成する**

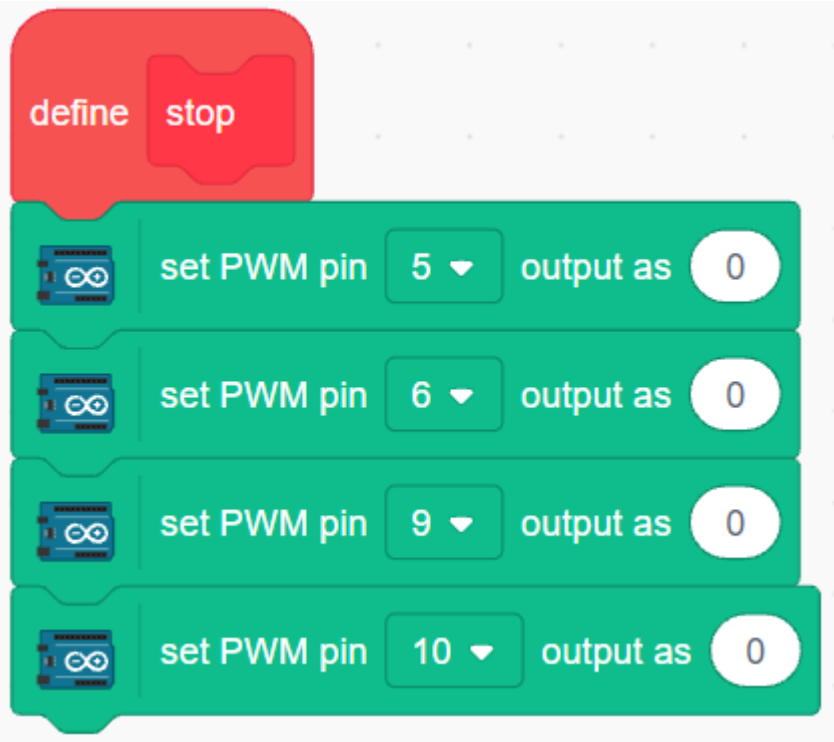
車を前後に動かします。




車を左後ろおよび右後ろに動かします。



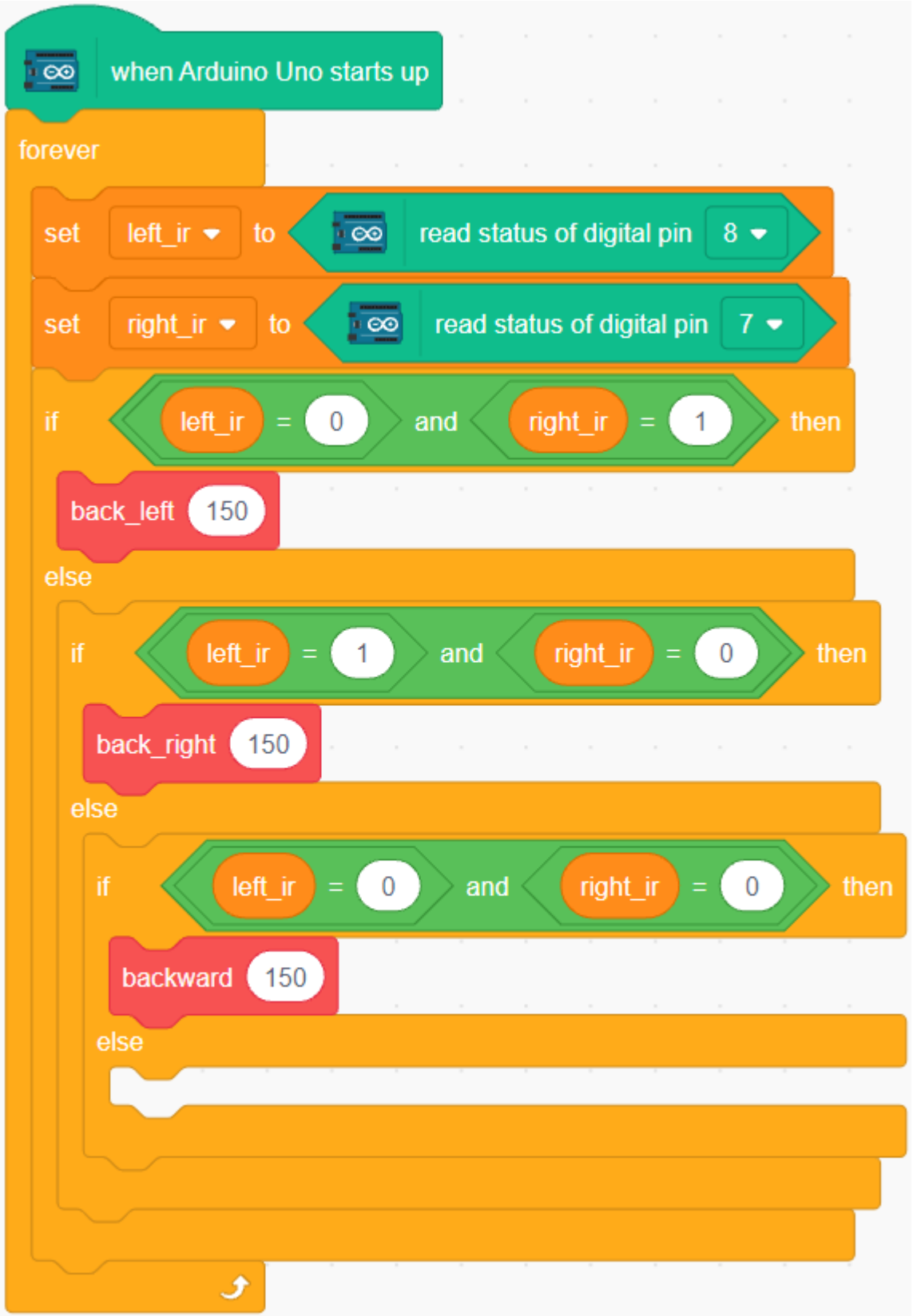

車を停止させます。




**2. 緊急障害物回避**

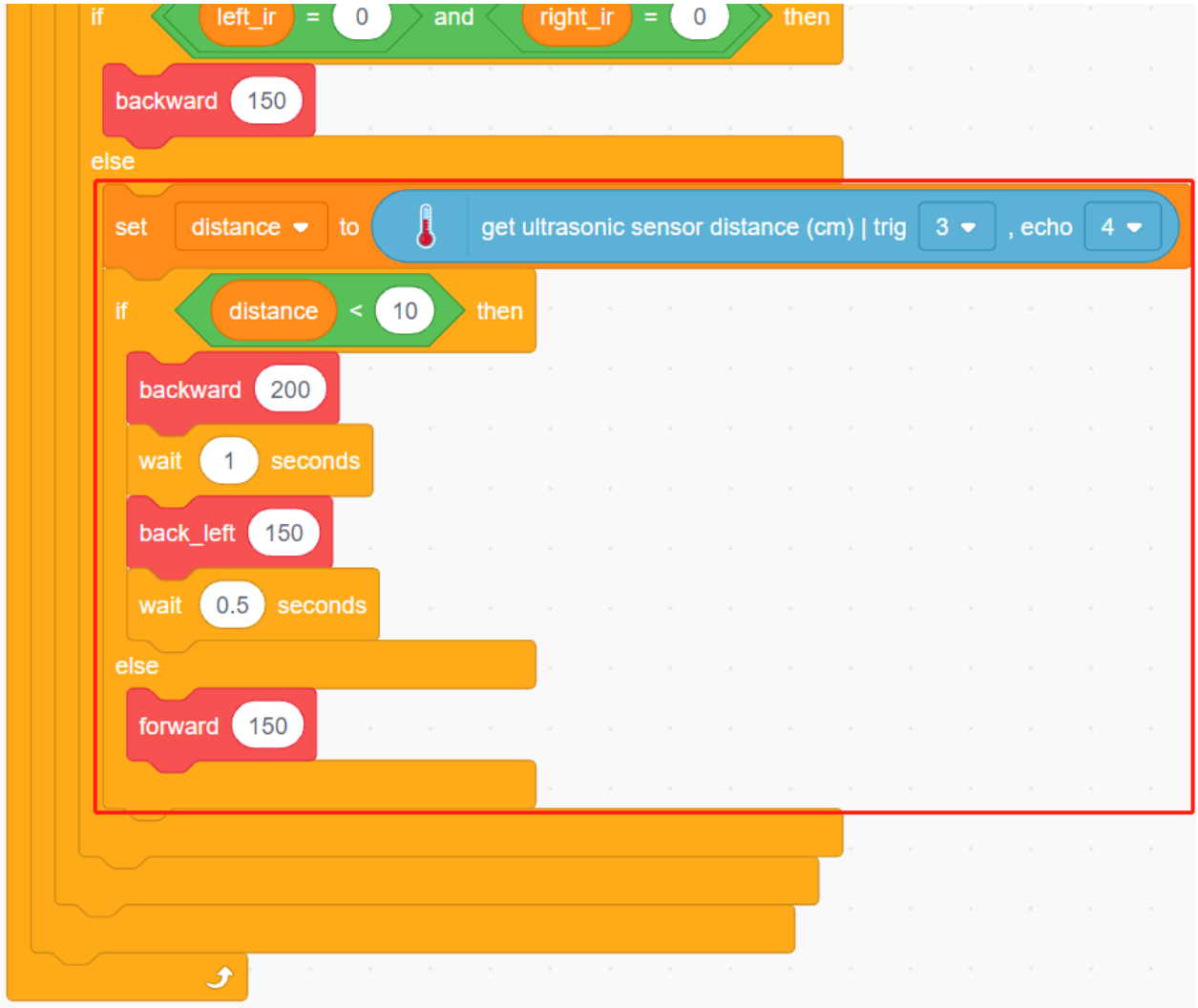
車の 2 つの赤外線障害物回避モジュールは、緊急障害物回避のために使用され、短距離、角度、または比較的小さな障害物での障害物を検出します。

- 左の赤外線モジュールが障害物を検出すると、車は左に後退します。
- 右の IR モジュールが障害物を検出すると、車は右後ろに後退します。
- 2 つのモジュールが同時に障害物を検出すると、車は直接後ろに後退します。


when Arduino Uno starts up
forever
set left_ir to read status of digital pin 8
set right_ir to read status of digital pin 7
if left_ir = 0 and right_ir = 1 then
back_left 150
else
if left_ir = 1 and right_ir = 0 then
back_right 150
else
if left_ir = 0 and right_ir = 0 then
backward 150
else

3. 長距離障害物回避

超音波モジュールの値を読み取り、検出された値が 10 未満の場合、車は後退します。それ以外の場合は前進を続けます。

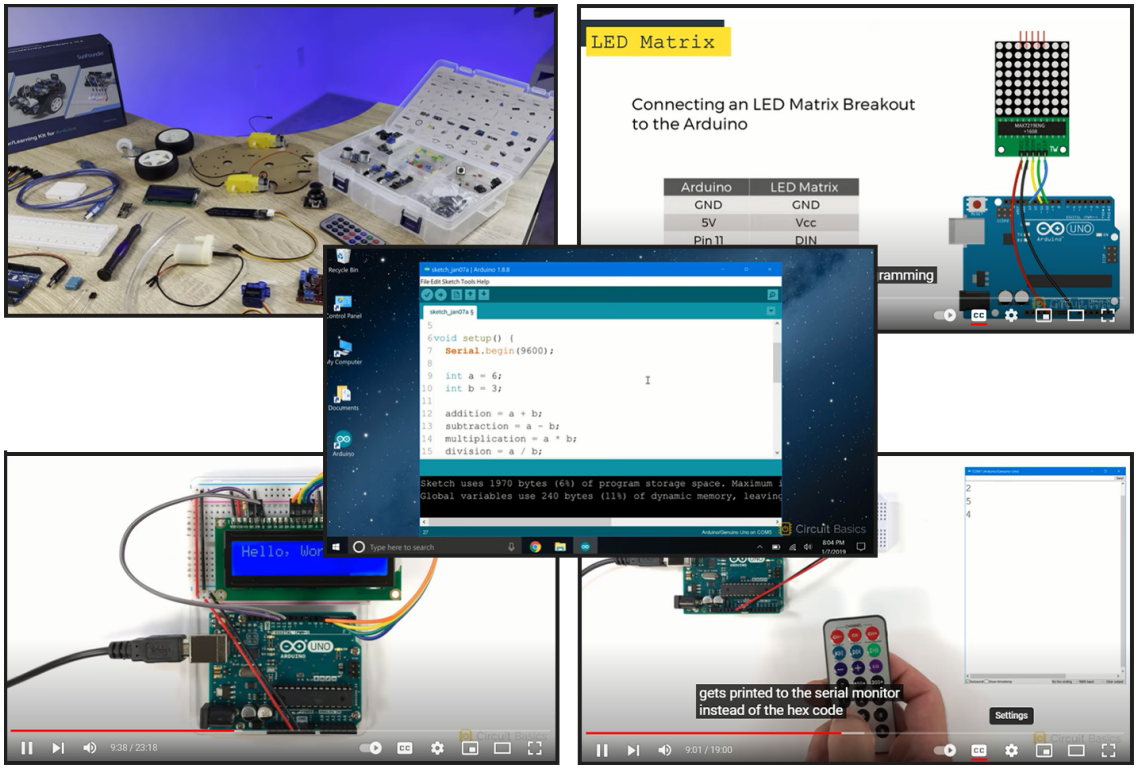


# 第 8 章

# ビデオ講座

- **基本的なプロジェクト** セクション用

  オンラインドキュメントの内容が理解しにくいと感じた場合、進行形式のビデオ講座に従うことで学びの体験を向上させることができます。これらのビデオにより、Arduino の学びがさらに魅力的で視覚的になります。

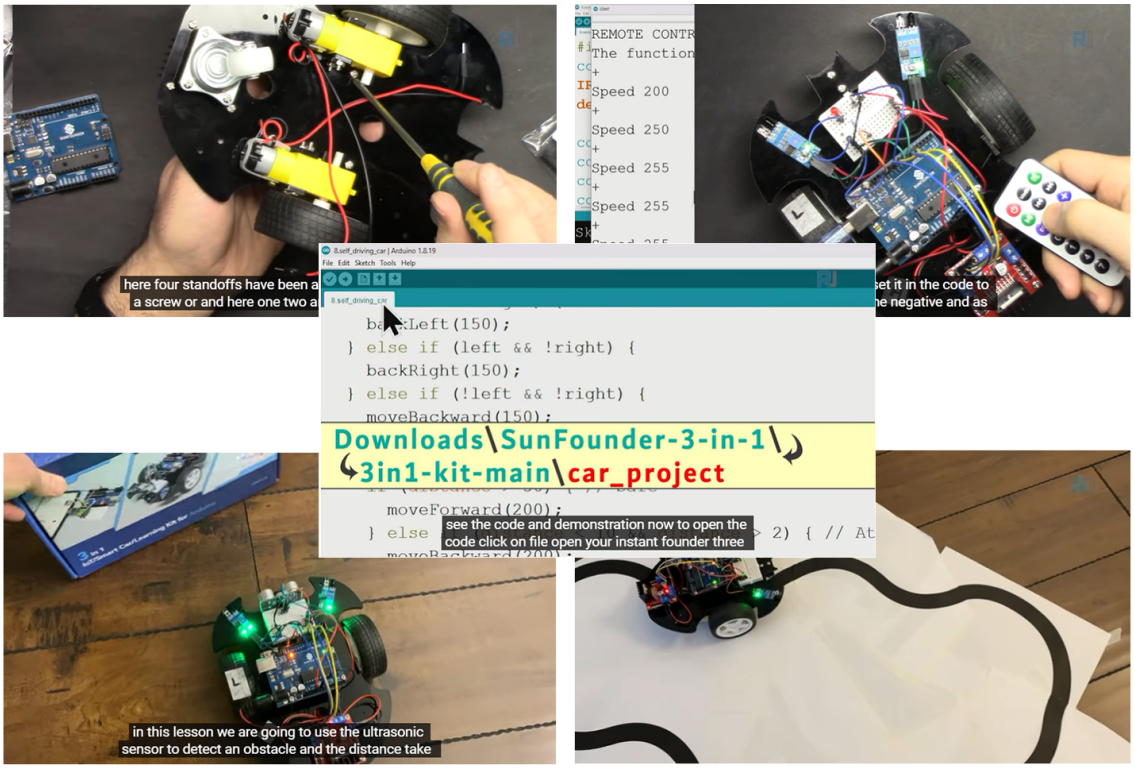



ビデオ講座へのアクセスはこちら: 。

これらのビデオでは、Arduino の概念や原則を楽しくインタラクティブに提示する魅力的な実験やプロジェクトを発見します。ビデオを視聴し、実践的な活動に参加することで、Arduino とともに興奮と楽しさを感じる学びの経験を持つことができます。

- **カープロジェクト** セクション用

  より多くのガイダンスや実践的な経験を得るために、次の YouTube プレイリストを視聴することをおすすめします: 。

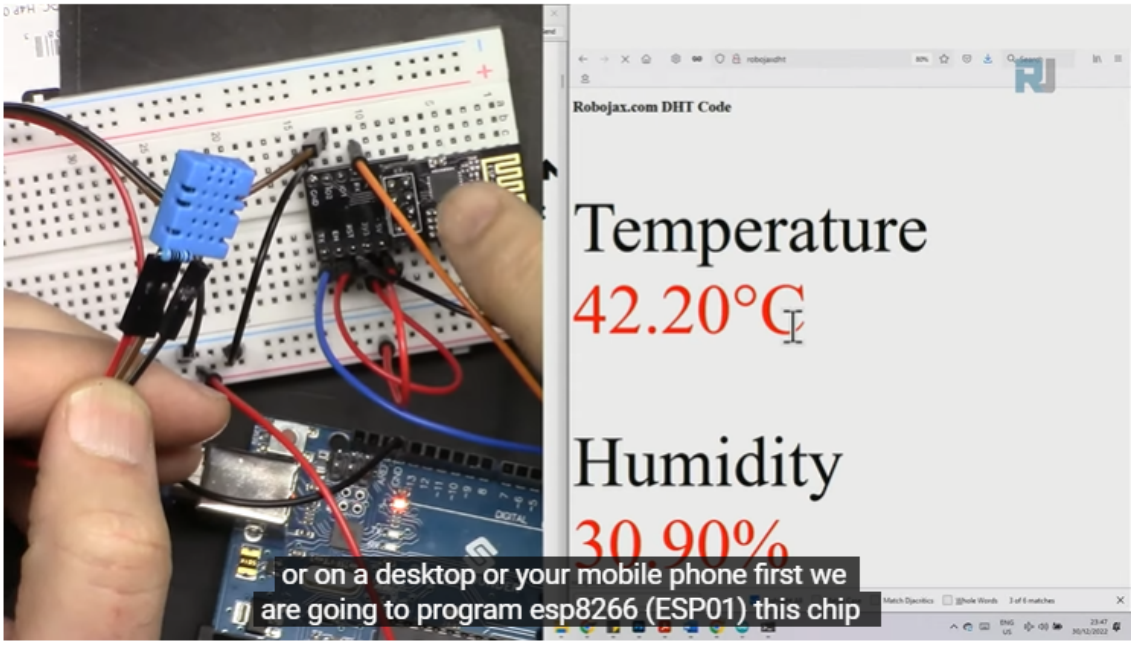



  これらのビデオで、魅力的なビデオレッスンを通じてロボティクスと Arduino の基礎を学びます。ステップバイステップで、モーターや障害物回避モジュール、ライントラッキングモジュール、赤外線受信機の仕組みを理解しながら、ロボットカーを組み立てます。車がさまざまな機能を実現する方法を探索し、ロボティクスとテクノロジーの世界でクリエイティブさを発揮してください。

- WiFi 機能について

  オンラインチュートリアルの *IoT* **プロジェクト** セクションでは、IoT プラットフォーム Blynk との通信方法を学びます。

  では、Web サーバーの作成とセンサーデータのアップロードの方法を指導されます。このチュートリアルでは、WiFi を使用して Arduino プロジェクトと Web サーバーとの接続を確立する方法を学びます。


Robojax.com DHT Code
Temperature
42.20°C
Humidity
30.90%
or on a desktop or your mobile phone first we
are going to program esp8266 (ESP01) this chip

# 第 9 章

# よくあるご質問（FAQ）

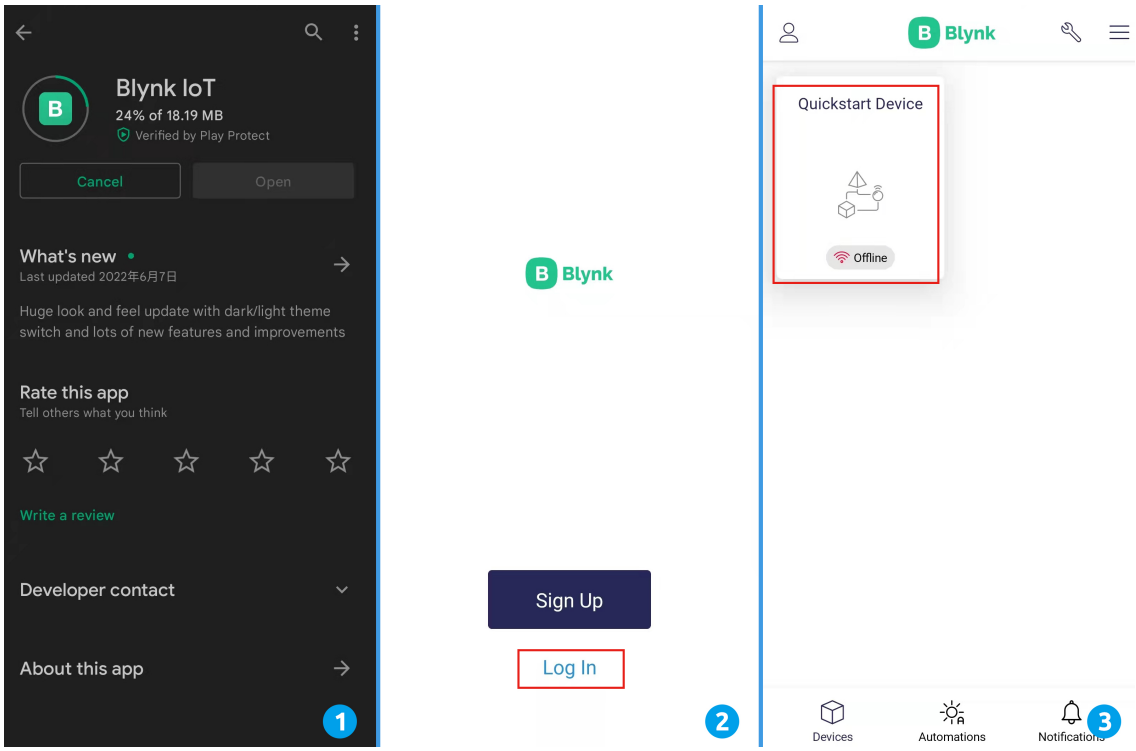
## 9.1 モバイルデバイスで Blynk を使用する方法は？

---

**注釈:** データストリームはウェブ上の Blynk でのみ作成できるため、ウェブ上でデータストリームを作成するために異なるプロジェクトを参照し、次に以下のチュートリアルに従ってモバイルデバイスの Blynk でウィジェットを作成する必要があります。

---

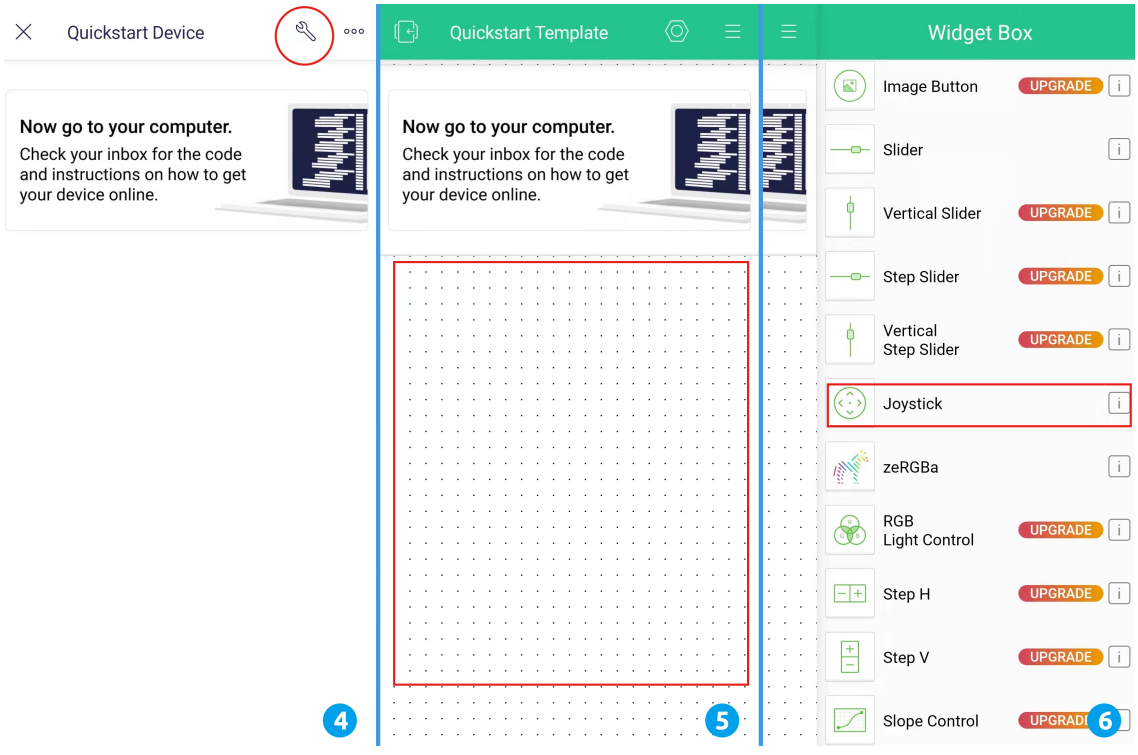
1. お使いのモバイルデバイスで Google Play または APP Store を開き、"Blynk IoT"（Blynk(legacy) ではない）を検索してダウンロードします。
2. APP を開いた後、ログインします。このアカウントは、ウェブクライアントで使用されるアカウントと同じものでなければなりません。
3. 次に、**Dashboard** に移動します（まだ持っていない場合は作成します）。モバイル用とウェブ用の **Dashboard** は互いに独立していることがわかります。




4. **編集** アイコンをクリックします。

5. 空白のエリアをクリックします。

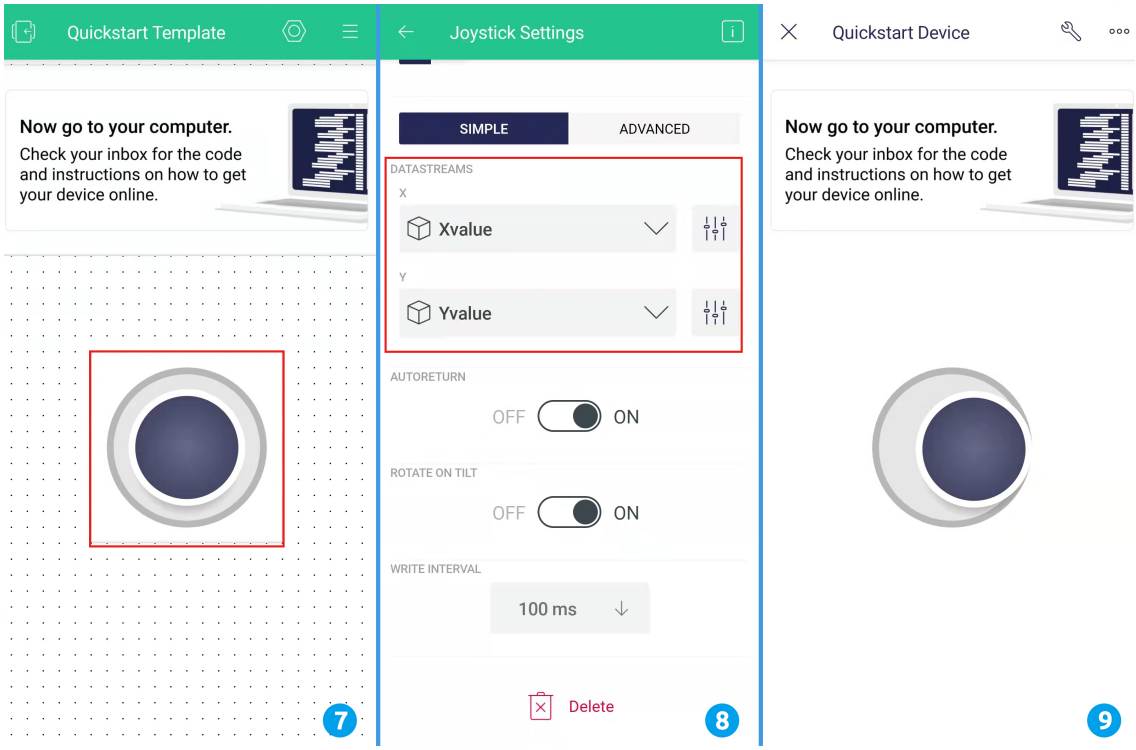
6. ウェブページ上と同じウィジェットを選択します。例えば、 **Joystick** ウィジェットを選択します。



7. 今、空白のエリアに **Joystick** ウィジェットが表示されますので、それをクリックします。

8. **Joystick** の設定が表示されるので、ウェブページで設定した **Xvalue** および **Yvalue** データストリームを選択します。各プロジェクトの各ウィジェットが異なるデータストリームに対応していることに注意してください。

9. **Dashboard** ページに戻り、必要に応じて **Joystick** を操作できます。

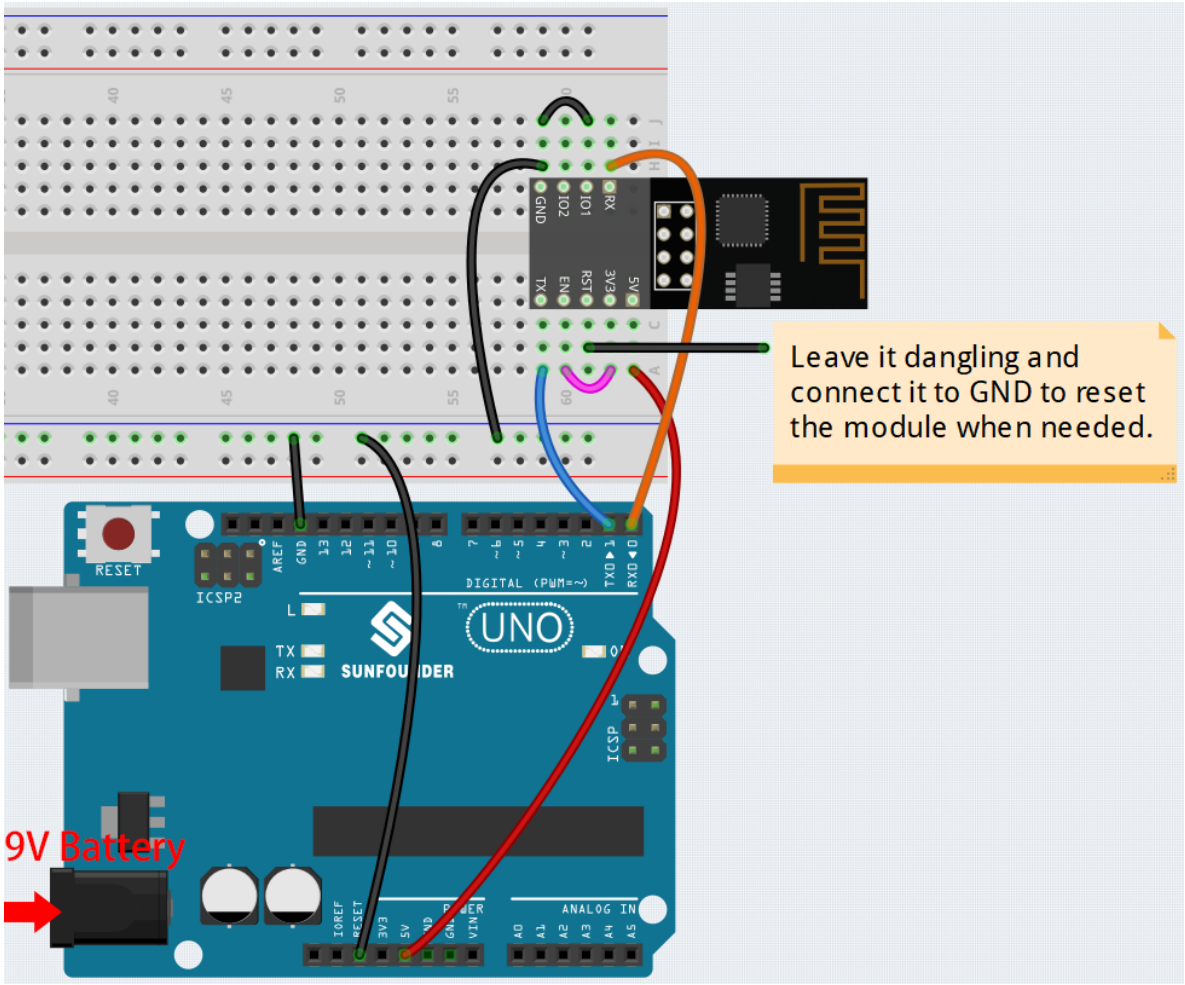



## 9.2 ESP8266 モジュールのファームウェアを再書き込みする方法は?

### 9.2.1 R3 でのファームウェアの再書き込み

#### 1. 回路の作成

ESP8266 と SunFounder R3 ボードを接続します。




### 2. ファームウェアの書き込み

- **Windows** を使用している場合、以下の手順でファームウェアを書き込んでください。

  1. ファームウェアと書き込みツールをダウンロードします。

     – ESP8266 **ファームウェア**

  2. 展開すると、4 つのファイルが表示されます。

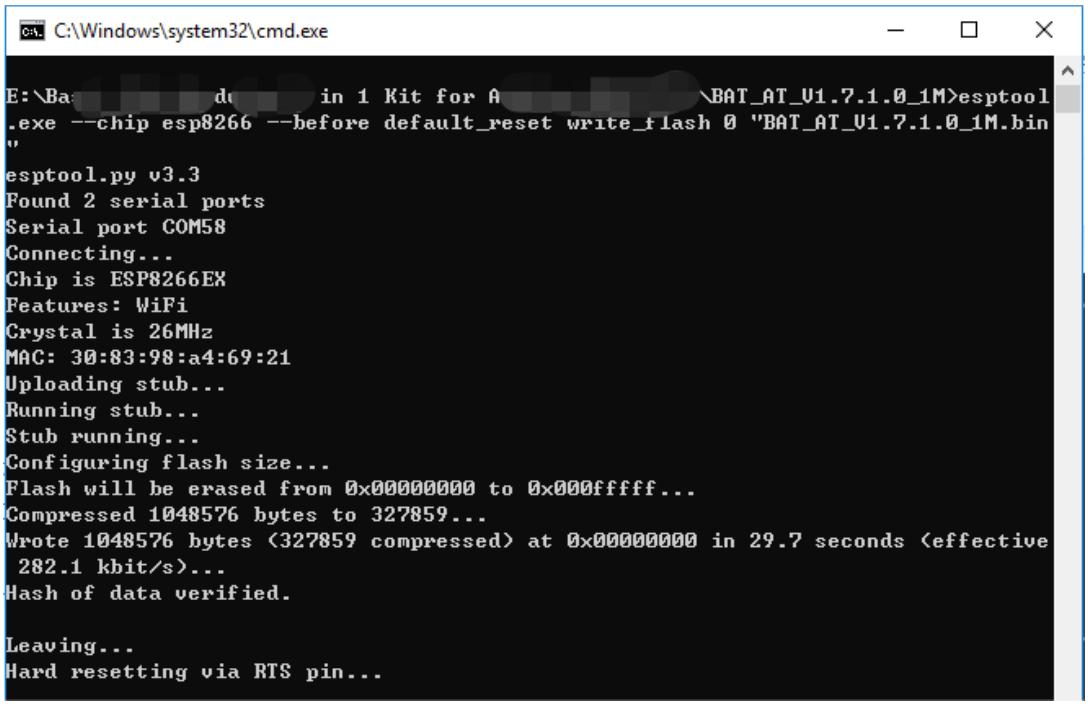
     – `BAT_AT_V1.7.1.0_1M.bin`: ESP8266 モジュールに書き込むファームウェア。

     – `esptool.exe`: Windows 用のコマンドラインユーティリティ。

     – `install_r3.bat`: Windows システム用のコマンドパッケージ。このファイルをダブルクリックすると、ファイル内のすべてのコマンドが実行されます。

     – `install_r4.bat`: `install_r3.bat` と同じですが、UNO R4 ボード専用です。

  3. `install_r3.bat` をダブルクリックしてファームウェアの書き込みを開始します。以下のプロンプトが表示されたら、ファームウェアが正常にインストールされたことを示します。




---

**注釈:** 書き込みに失敗した場合は、以下の点を確認してください。

- ESP8266 アダプターの RST を GND に挿入してから抜き、ESP8266 モジュールをリセットします。
- 配線が正しいか確認してください。
- コンピュータがボードを正しく認識しているか、またポートが占有されていないことを確認してください。
- install.bat ファイルを再度開いてください。

---

- **Mac OS** を使用している場合、以下の手順でファームウェアを書き込んでください。

1. Esptool をインストールするには、以下のコマンドを使用します。Esptool は Espressif チップの ROM ブートローダーと通信するための Python ベースのオープンソース、プラットフォーム独立のユーティリティです。

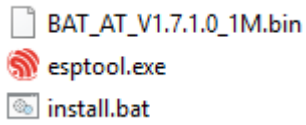
```
python3 -m pip install --upgrade pip
python3 -m pip install esptool
```

2. esptool が正しくインストールされていれば、 `python3 -m esptool` を実行すると [usage: esptool] のようなメッセージが出力されます。

3. ファームウェアをダウンロードします。

- ESP8266 **ファームウェア**

4. 展開すると、3 つのファイルが表示されます。

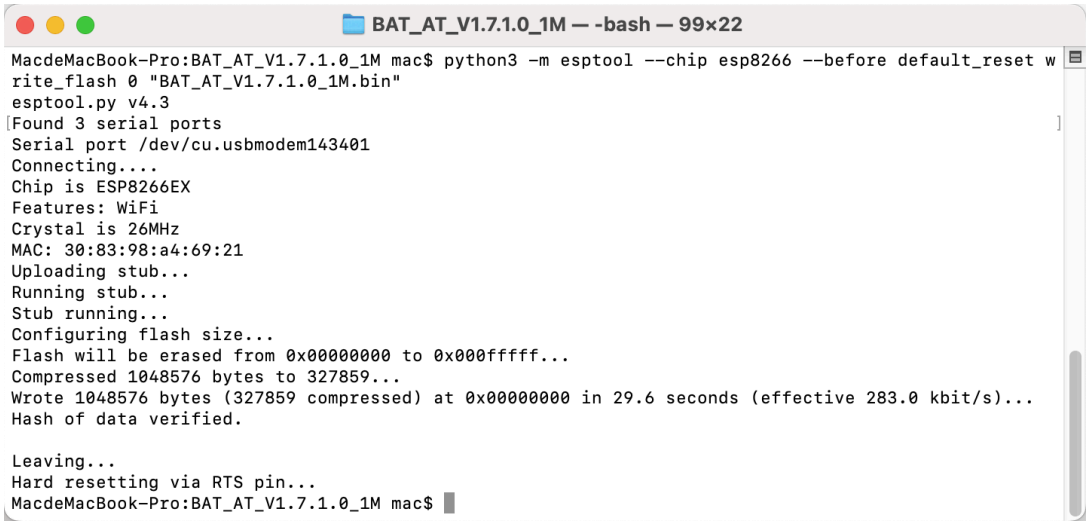



   - BAT_AT_V1.7.1.0_1M.bin: ESP8266 モジュールに書き込むファームウェア。
   - esptool.exe: Windows 用のコマンドラインユーティリティ。
   - install_r3.bat: Windows システム用のコマンドパッケージ。
   - install_r4.bat: install_r3.bat と同じですが、UNO R4 ボード専用です。

5. ターミナルを開き、 cd コマンドを使用してダウンロードしたファームウェアのフォルダに移動し、次のコマンドを実行して既存のファームウェアを消去し、新しいファームウェアを再書き込みします。

```
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. 以下のプロンプトが表示されたら、ファームウェアが正常にインストールされたことを示します。




---

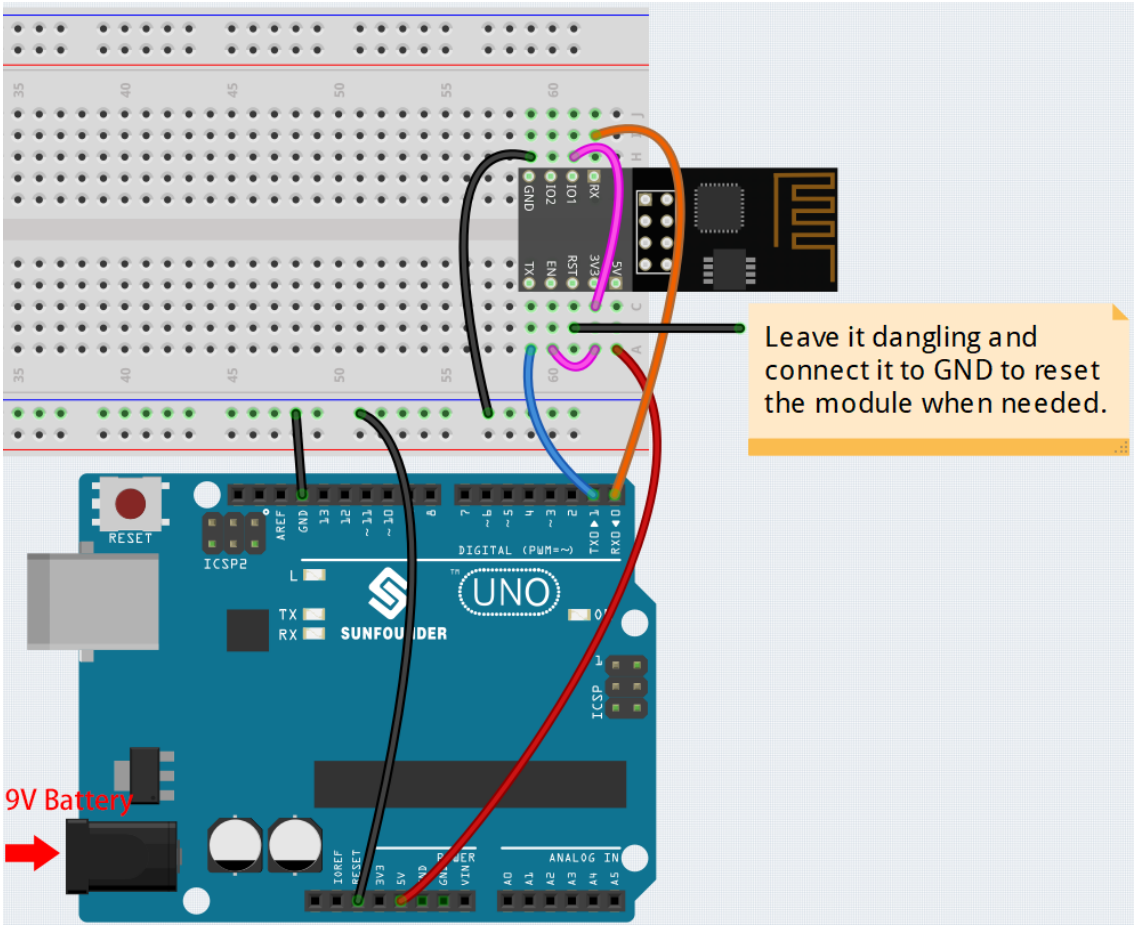
**注釈:** 書き込みに失敗した場合は、以下の点を確認してください。

- ESP8266 アダプターの RST を GND に挿入してから抜き、ESP8266 モジュールをリセットします。
- 配線が正しいか確認してください。

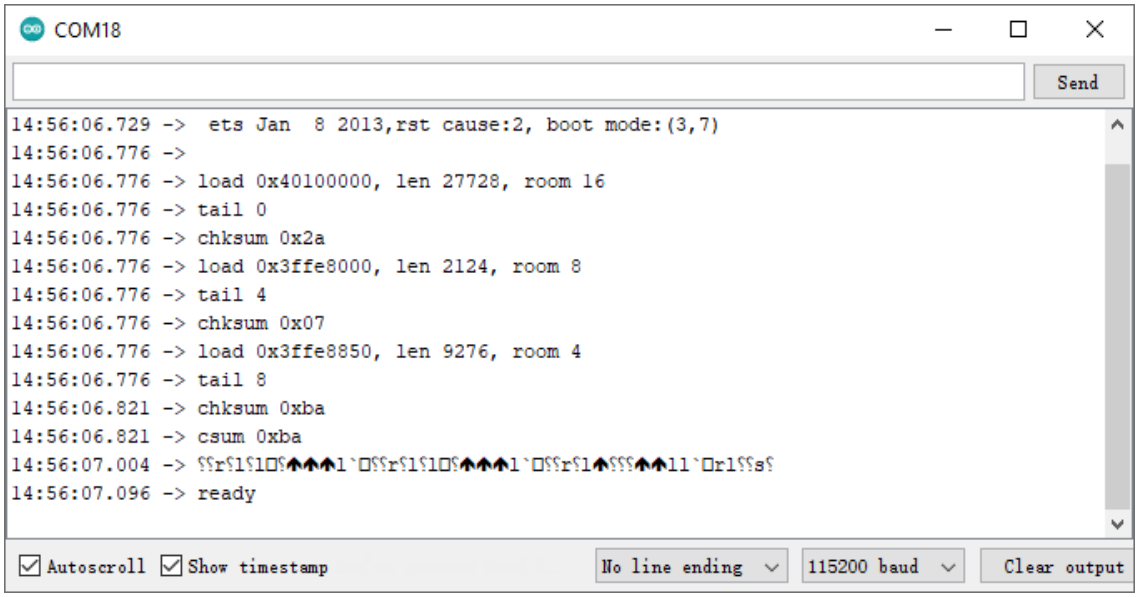
- コンピュータがボードを正しく認識しているか、またポートが占有されていないことを確認してください。
- install.bat ファイルを再度開いてください。

### 3. テスト

1. 元の配線の基盤上で、IO1 を 3V3 に接続します。



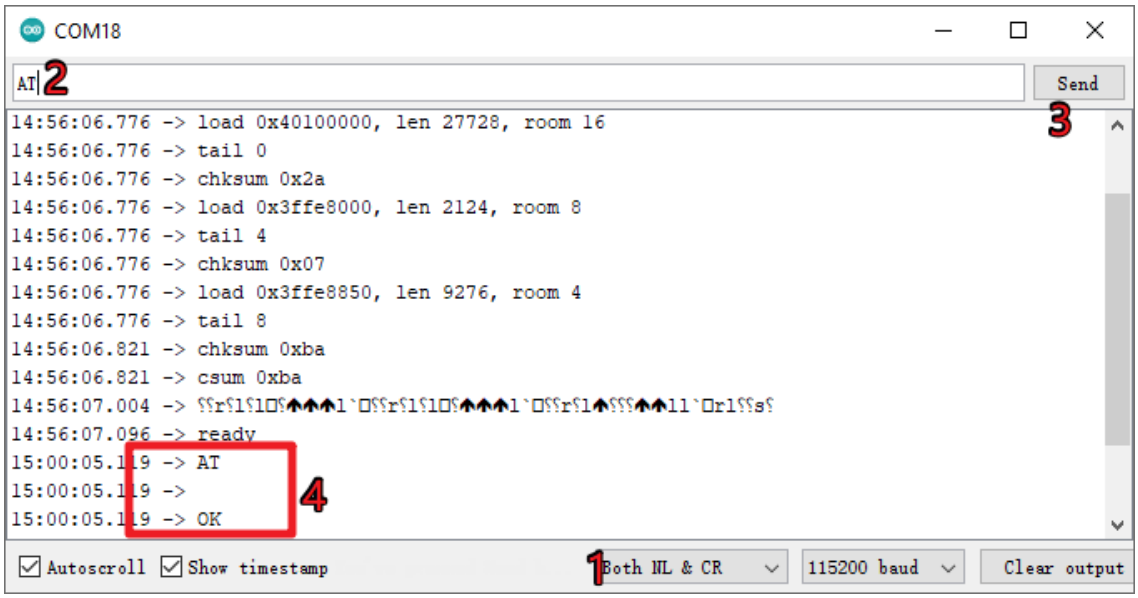

2. 右上の虫眼鏡アイコン（シリアルモニタ）をクリックし、ボーレートを **115200** に設定すると、ESP8266 モジュールに関する情報が表示されます。




**注釈:**

- ready が表示されない場合は、ESP8266 モジュールをリセットして（RST を GND に接続）、シリアルモニタを再度開いてみてください。

3. **NEWLINE DROPDOWN BOX** をクリックし、ドロップダウンオプションで both NL & CR を選択し、AT を入力します。OK が返された場合、ESP8266 が R3 ボードと正常に接続されていることを示します。

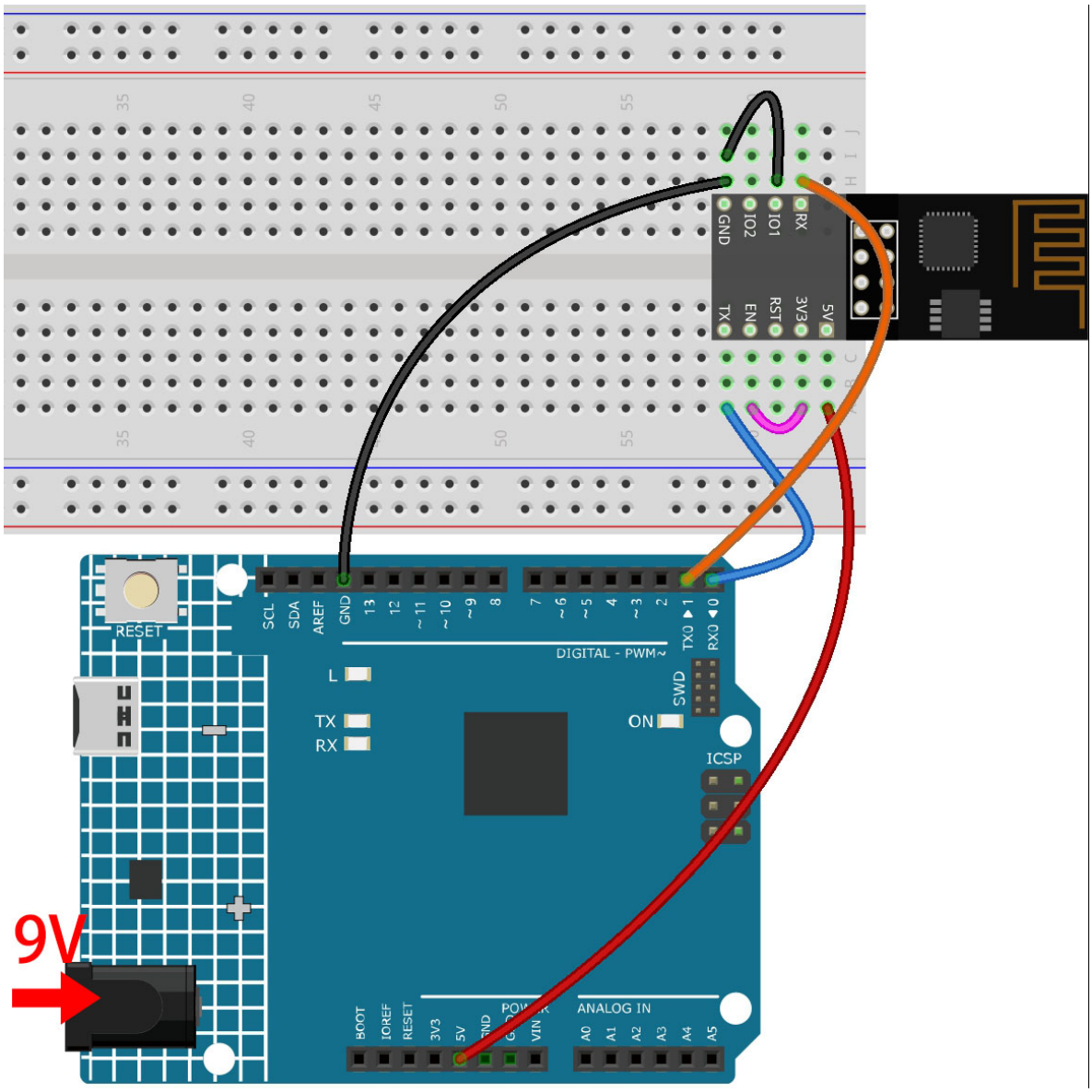



次に、*1.1 ESP8266* **の設定** に従って、ESP8266 モジュールの動作モードとボーレートを設定することができます。

### 9.2.2 R4 でファームウェアを再書き込み

#### 1. 回路を作成する

ESP8266 と Arduino UNO R4 ボードを接続します。




#### 2. R4 に以下のコードをアップロードする

```
void setup() {
    Serial.begin(115200);
    Serial1.begin(115200);
}
```

(次のページに続く)

(前のページからの続き)

```
void loop() {
    if (Serial.available()) {      // シリアル（USB）から何かが入力された場合
        Serial1.write(Serial.read());   // それを読み取り、Serial1（ピン 0 & 1）で送信する
    }
    if (Serial1.available()) {     // Serial1（ピン 0 & 1）から何かが入力された場合
        Serial.write(Serial1.read());   // それを読み取り、シリアル（USB）で送信する
    }
}
```

**3. ファームウェアの書き込み**

- **Windows** を使用している場合は、以下の手順でファームウェアを書き込む方法に従ってください。

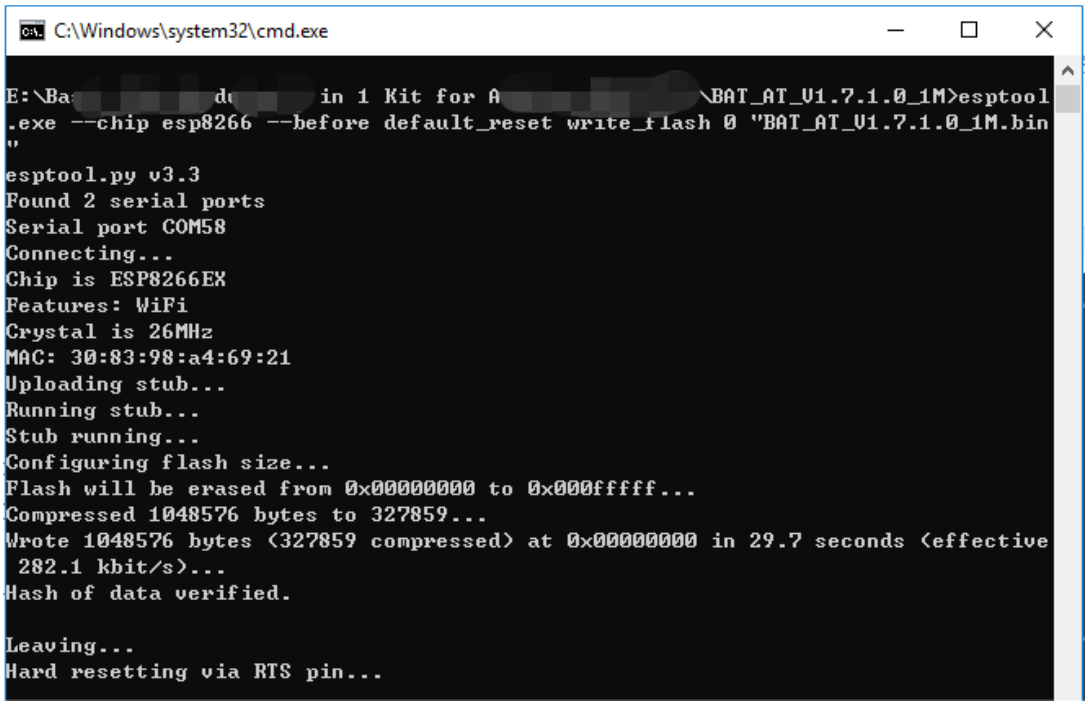
  1. ファームウェアと書き込みツールをダウンロードします。

     - ESP8266 Firmware

  2. 解凍すると、4 つのファイルが表示されます。

     - BAT_AT_V1.7.1.0_1M.bin: ESP8266 モジュールに書き込むファームウェア。
     - esptool.exe: Windows 用のコマンドラインユーティリティ。
     - install_r3.bat: Windows システム用のコマンドパッケージ。このファイルをダブルクリックすると、ファイル内のすべてのコマンドが実行されます。
     - install_r4.bat: install_r3.bat と同じですが、UNO R4 ボード専用です。

  3. install_r4.bat をダブルクリックしてファームウェアの書き込みを開始します。以下のプロンプトが表示されたら、ファームウェアが正常にインストールされました。




---

**注釈:** 書き込みに失敗した場合は、以下の点を確認してください。

- ESP8266 Adapter の RST を GND に挿入してから取り外すことで、ESP8266 モジュールをリセットします。
- 配線が正しいか確認してください。
- コンピュータがボードを正しく認識しているか、およびポートが占有されていないか確認してください。
- install.bat ファイルを再度開きます。

---

- **Mac OS** システムを使用している場合、以下の手順に従ってファームウェアを書き込む方法に従ってください。

1. Esptool をインストールするには、以下のコマンドを使用します。Esptool は Python ベースのオープンソース、プラットフォームに依存しないユーティリティで、Espressif チップの ROM ブートローダーと通信します。

```
python3 -m pip install --upgrade pip
python3 -m pip install esptool
```

2. esptool が正しくインストールされている場合、 `python3 -m esptool` を実行すると [usage: esptool] というメッセージが出力されます。

3. ファームウェアをダウンロードします。

   - ESP8266 Firmware

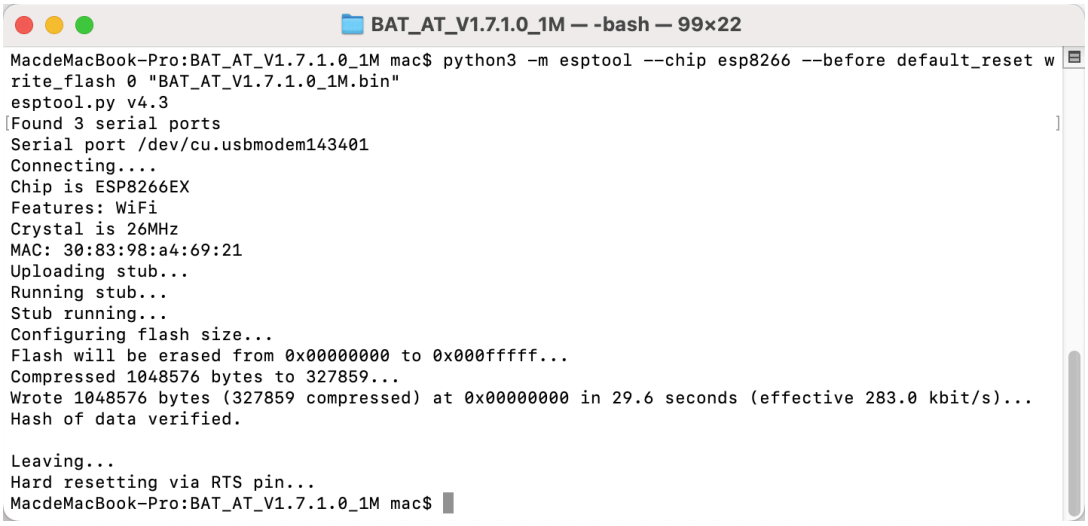
4. 解凍すると、4 つのファイルが表示されます。

   - BAT_AT_V1.7.1.0_1M.bin: ESP8266 モジュールに書き込むファームウェア。
   - esptool.exe: Windows 用のコマンドラインユーティリティ。
   - install_r3.bat: Windows システム用のコマンドパッケージ。
   - install_r4.bat: install_r3.bat と同じですが、UNO R4 ボード専用です。

5. ターミナルを開き、ダウンロードしたファームウェアのフォルダに移動するために cd コマンドを使用します。その後、以下のコマンドを実行して既存のファームウェアを消去し、新しいファームウェアを再書き込みします。

```
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. 以下のプロンプトが表示されたら、ファームウェアが正常にインストールされました。




---

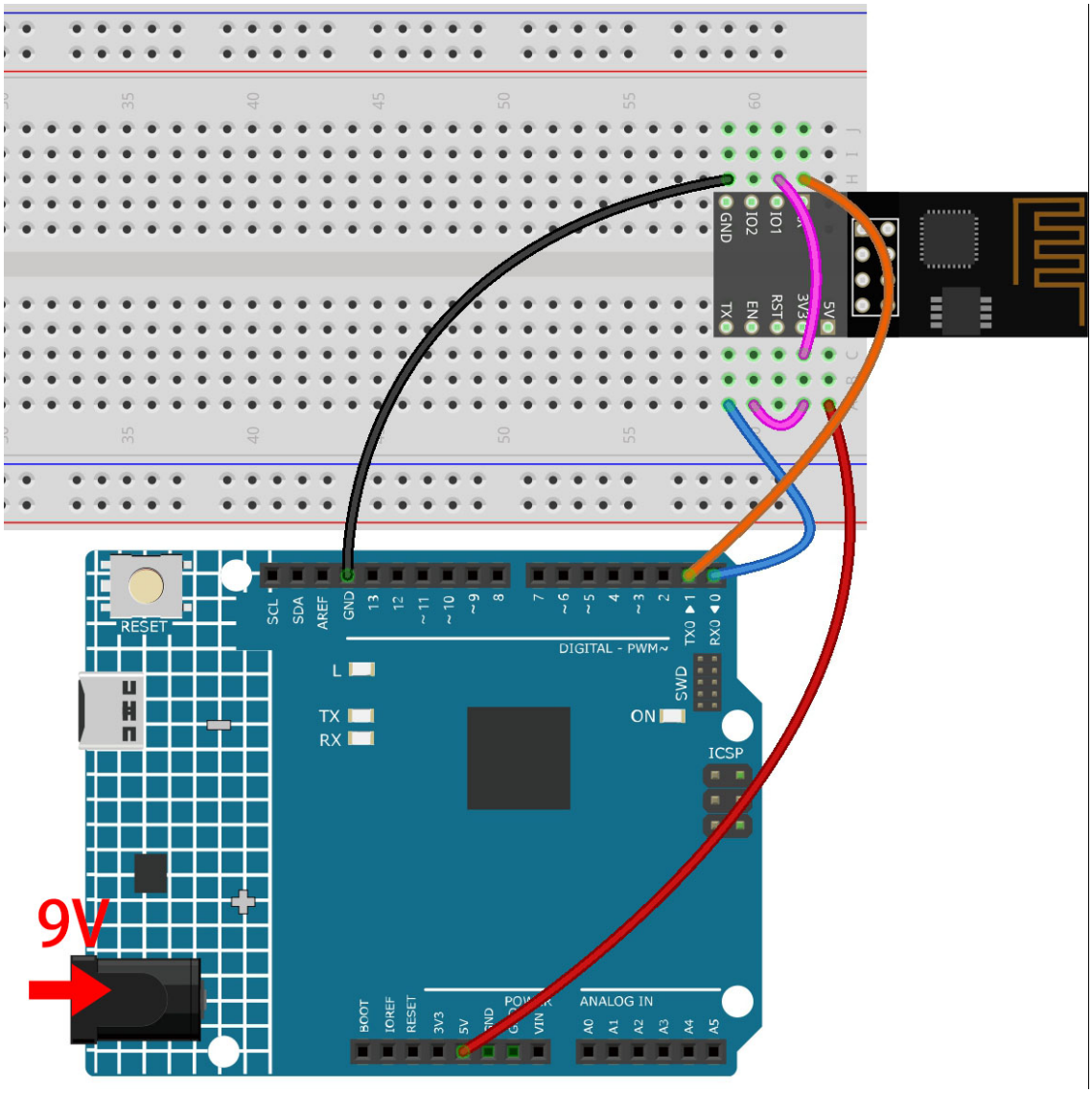
**注釈:** 書き込みに失敗した場合は、以下の点を確認してください。

- ESP8266 Adapter の RST を GND に挿入してから取り外すことで、ESP8266 モジュールをリセットします。
- 配線が正しいか確認してください。
- コンピュータがボードを正しく認識しているか、およびポートが占有されていないか確認してください。

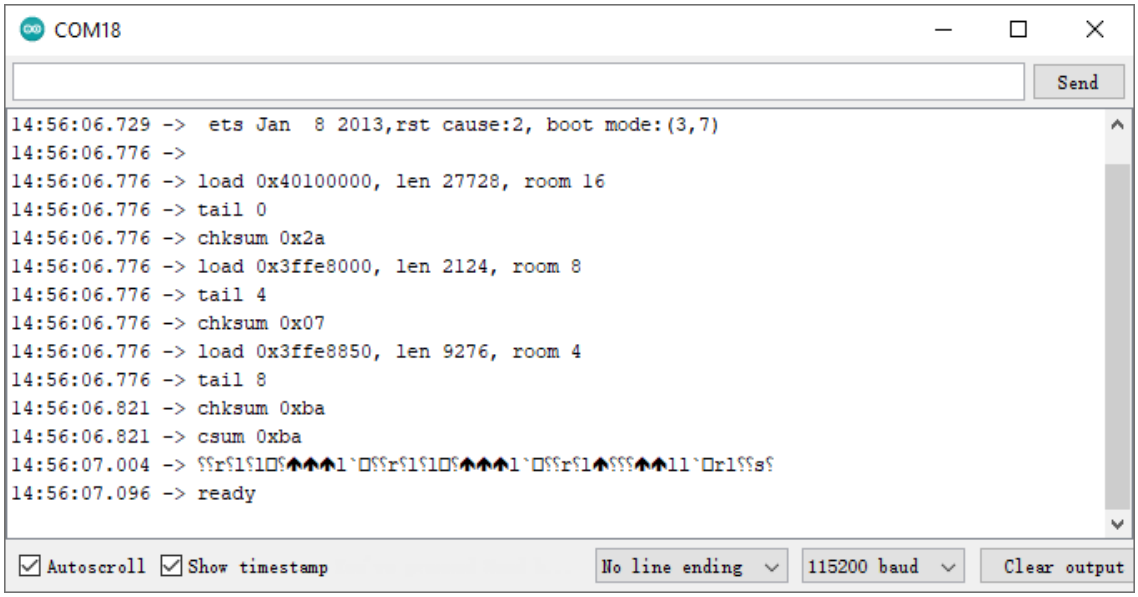
– install.bat ファイルを再度開きます。

### 4. テスト

1. 元の配線に基づいて、IO1 を 3V3 に接続します。



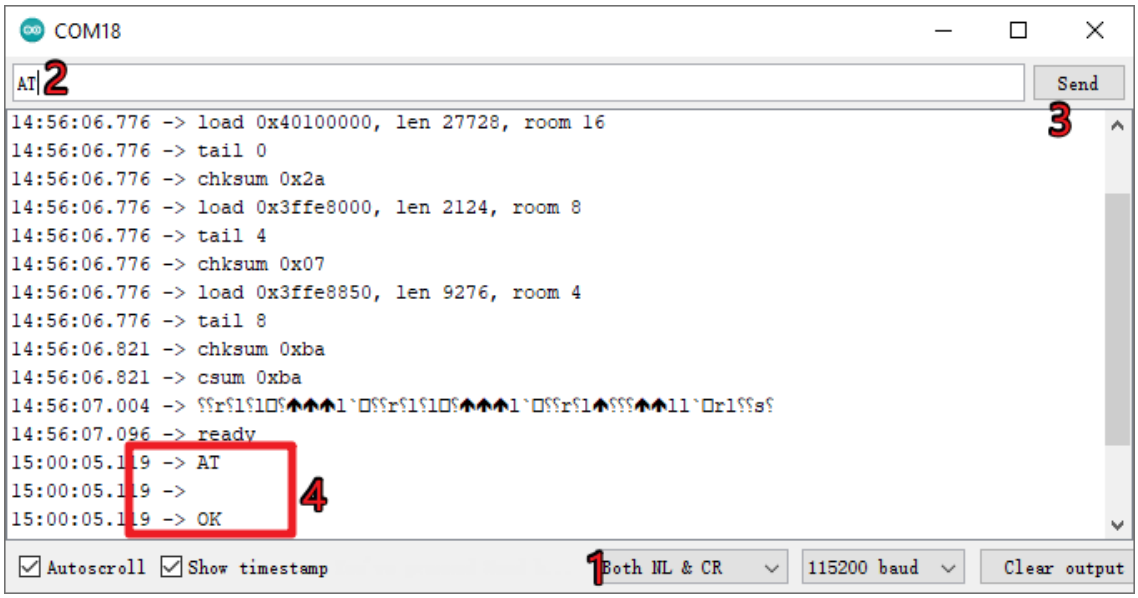

2. 右上隅の虫眼鏡アイコン（シリアルモニタ）をクリックし、ボーレートを **115200** に設定すると、ESP8266 モジュールに関する情報が表示されます。




**注釈:**

- ready が表示されない場合、ESP8266 モジュールをリセット（RST を GND に接続）し、シリアルモニタを再度開くことができます。

3. **NEWLINE DROPDOWN BOX** をクリックし、ドロップダウンオプションで both NL & CR を選択し、AT を入力します。OK が返されると、ESP8266 がボードとの接続を正常に確立したことを意味します。




次に、 *1.1 ESP8266* **の設定** を参照して、ESP8266 モジュールの動作モードとボーレートを設定できます。

# 第 10 章

# ありがとうございます

製品を評価してくれた評価者、チュートリアルの提案をしてくれたベテラン、そして私たちをフォローしサポートしてくれるユーザーの皆様、ありがとうございます。皆様からの貴重な提案は、私たちがより良い製品を提供するための動機となっています。

**特別な感謝**

- レン・デイビスソン
- カレン・ダニエル
- フアン・デラコスタ

少しだけお時間を割いて、このアンケートに回答していただけますか？

---

**注釈:** アンケートを提出した後、結果を表示するためにページの先頭に戻ってください。

---

# 第 11 章

# 著作権通知