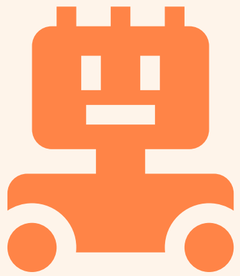
SW-KIT-100

卓上小型ロボットキット チュートリアル

株式会社シサクウェア
ShisakuWare

このチュートリアルでできること

- Arduinoの基本を理解する
- 電子回路の基礎を学ぶ
- センサーの使い方を学ぶ
- モーターを制御する
- ライントレースロボットを作る



ロボット紹介

机の上で気軽にロボット開発を始められる、
小型ロボットのベースキットです。

当社のロボット開発基板（SW-BRD-100）
を使っています。

連続回転サーボモータを採用しており、
前進・後退・旋回など、ロボットの基本動作を
手軽に試すことができます。

LED、ブザー、距離センサー、通信機能など、
用途に応じて自由に拡張できるため、
シンプルな走行ロボットから
自律動作を行うロボットまで幅広く対応します。

学習の流れ

1. Arduinoとは

2.基盤に備え付けのLED・カラーLED・ブザーを動かしてみよう

3. LED点滅

4. センサー読み取り

5. モーター制御

6. ロボット走行

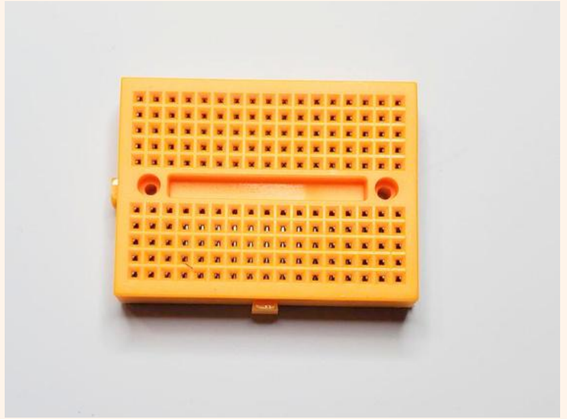
7. ライントレース

8.まとめ

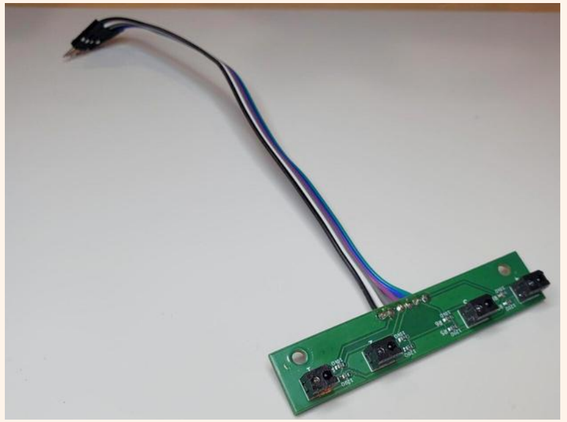
必要なもの

- ブレッドボード
- LED
- 抵抗
- ジャンプワイヤー
- ラインセンサー
- 当社のロボットキット

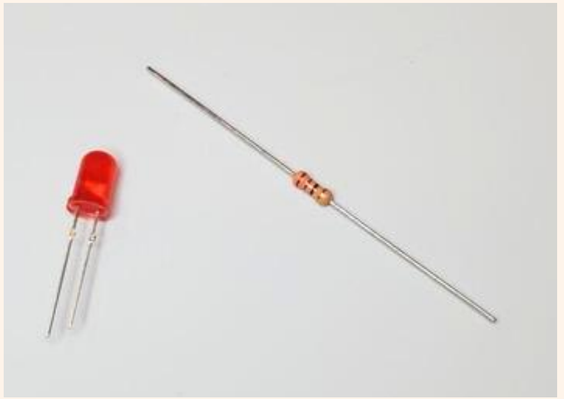
3.5×4.5ブレッドボード



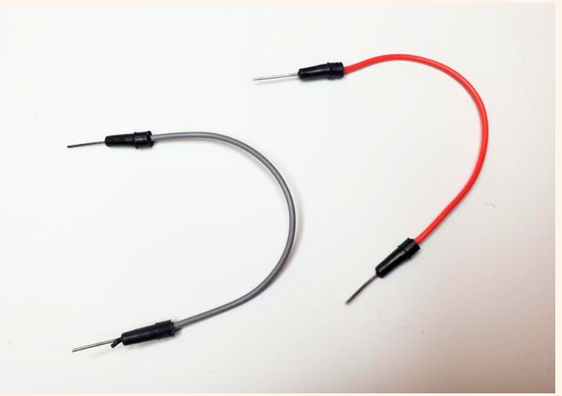
ライントレーサ基盤



LED／220Ω抵抗



ジャンプワイヤー



LEDと抵抗の選定例

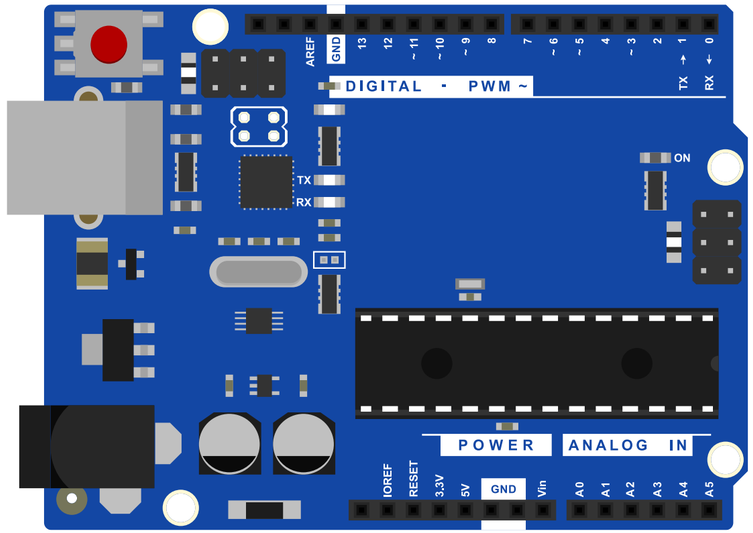
電源電圧：3.3V
LED順方向電圧：2.0V
定格電流：10mA（0.01A）
R = (3.3 - 2.0) / 0.01 = 130Ω

標準抵抗値の220Ωの抵抗を使用する。

1. Arduino基礎

Arduinoとは

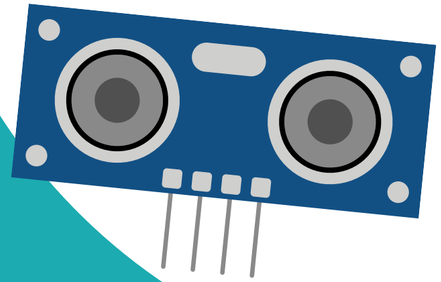
Arduino（アルドゥイーノ）は、
初心者でも簡単に電子工作やプログラミングができ、
電子回路を簡単に制御できるマイコンボードです。

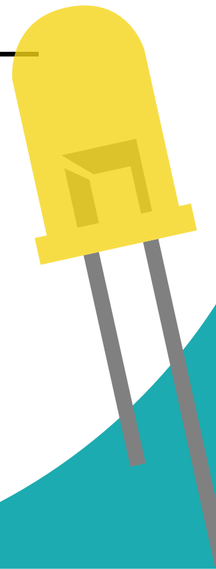


ARDUINO™

Arduinoでできること

- **LED制御**：明るさの調整（PWM制御）、点滅パターン、カラー変化など
- **センサー読み取り**：温度、湿度、明るさ、距離、人の動き（人感センサー）などを取得
- **モーター制御**：サーボモーター、DCモーター、ステッピングモーターを動かし、ロボットやラジコンを作成
- **ロボット制御**：モーターを複数組み合わせて、物を掴んだり移動させたりするアームを制御



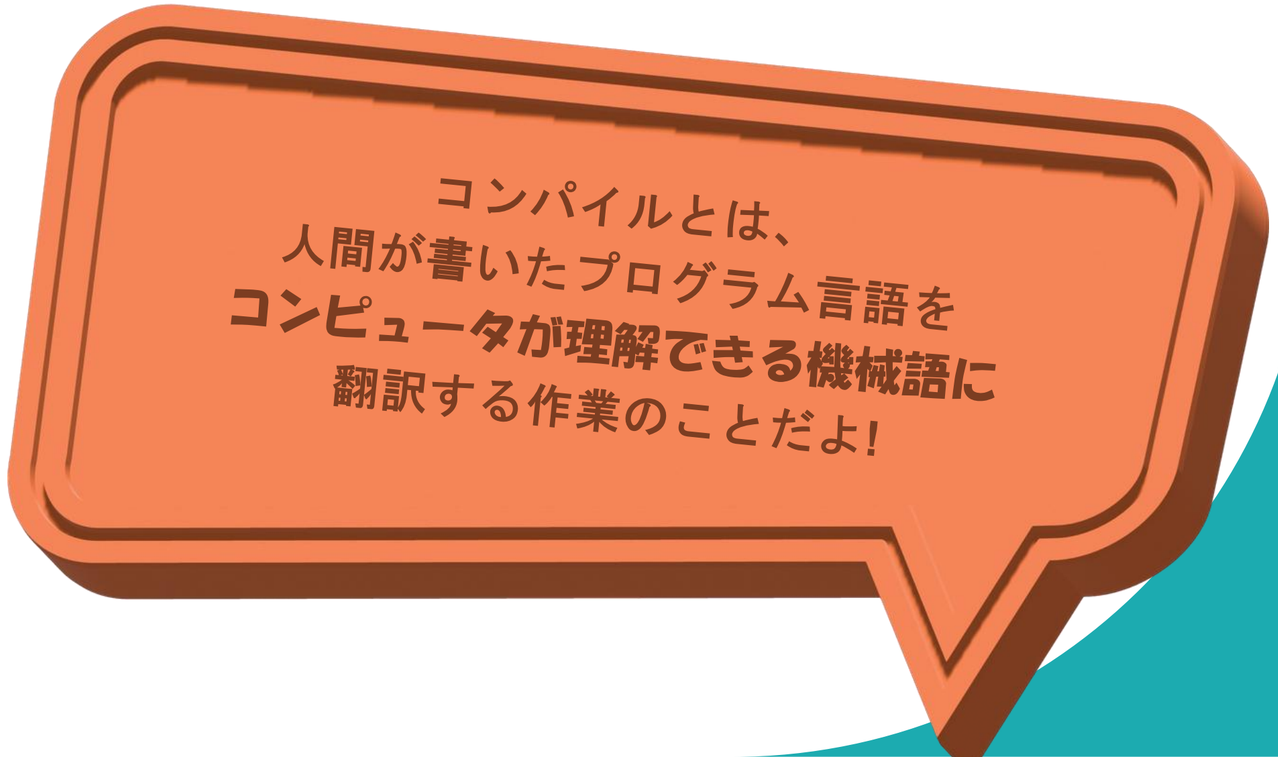


Arduino IDEとは

Arduino IDEとはプログラムを書く無料のソフト

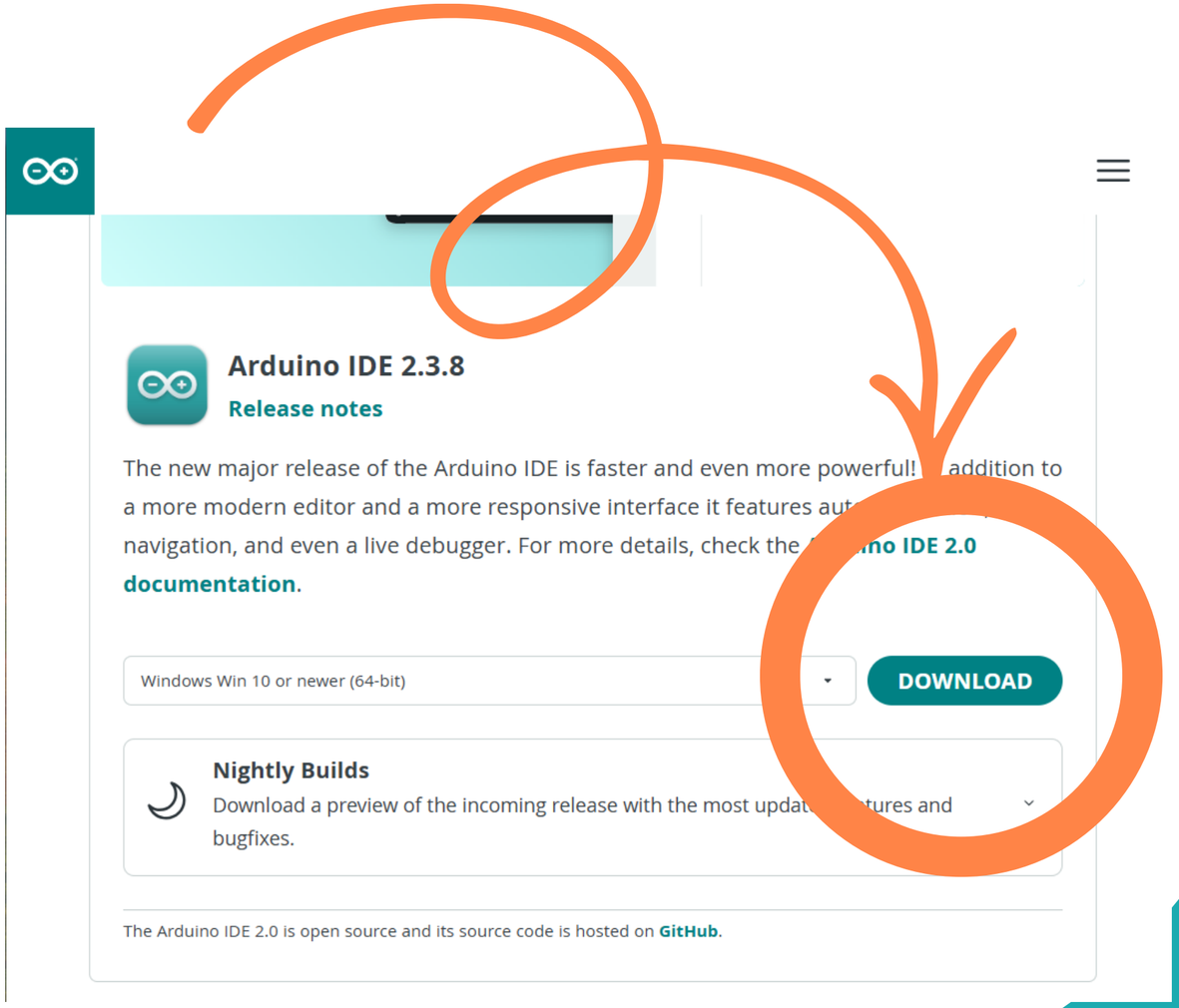
説明内容

- ✔コードを書く
- ✔コンパイル
- ✔Arduinoに書き込み



Arduino IDEをインストールしよう！

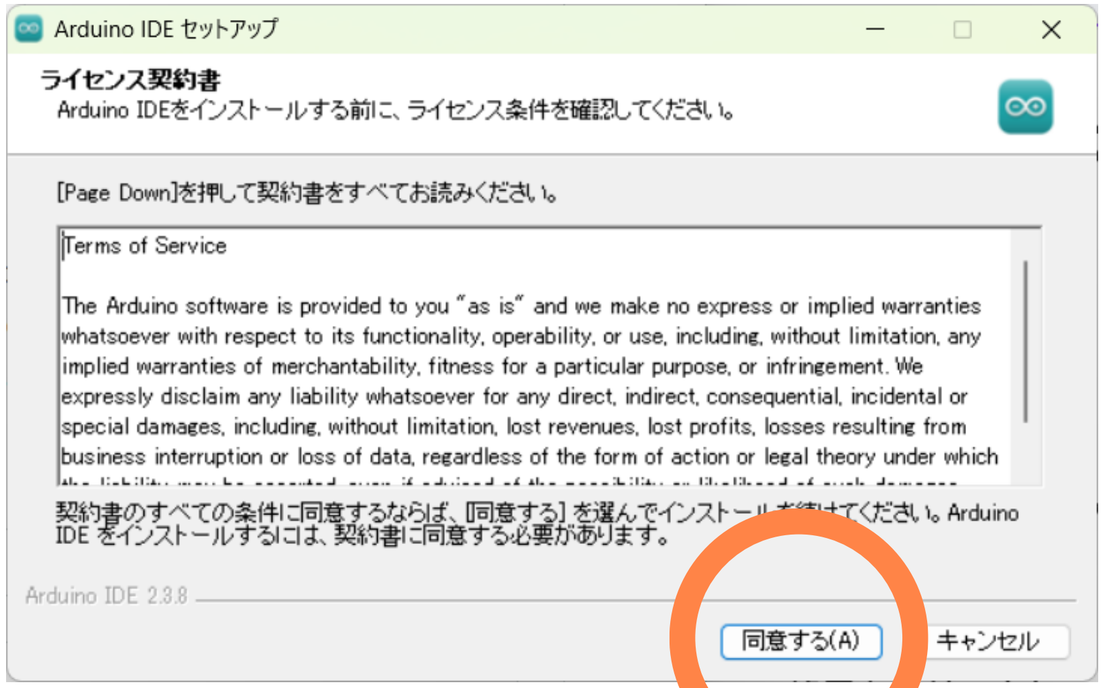
1. Arduino公式にアクセス
https://www.arduino.cc/en/software/
2.使用するOSを選択
3. Downloadをクリック



Arduino IDEをインストールしよう！

4.インストーラー起動
ダウンロードされたファイルを開き
arduino-ide_x.x.x.exe
をダブルクリック

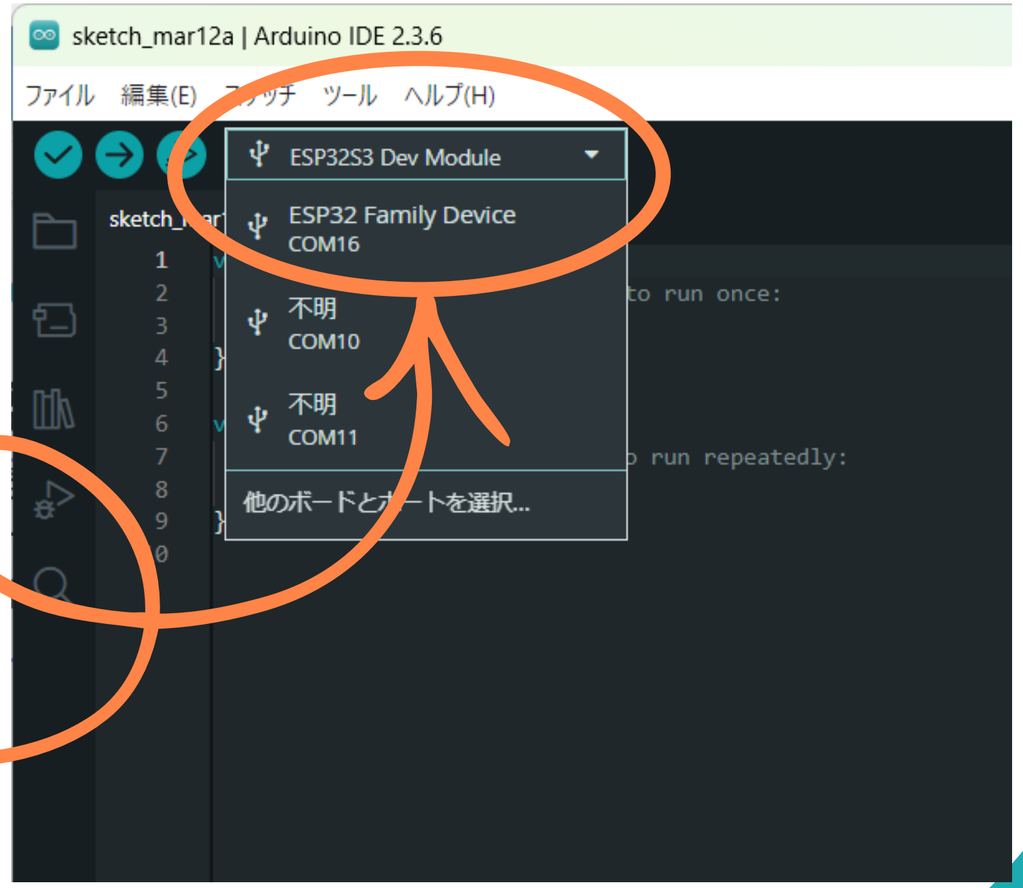
5.セットアップ開始
- 同意するをクリック
- 次へ をクリック
- インストールをクリック



6.終了をクリックしてインストール完了！

開発基盤に接続

1. 開発基盤(SW-BRD-100)をUSB接続
2. ツール → ボード
3. esp32 → ESP32 Dev Moduleを選択
4. 接続されているCOMポートを選択



Arduinoプログラム構造

setup → 電源接続時に最初に1回だけ実行

loop → 繰り返し実行

Arduinoライブラリ

Arduinoライブラリは、
LEDやセンサー、モーターなどの電子部品を
簡単に制御するための、事前に作成されたプログラム。
#include文で読み込むだけで機能が利用可能になります。

ライブラリの導入方法

ライブラリマネージャー

1.Arduino IDEで「ツール」＞「ライブラリを管理...」を選択。
2.検索窓に目的のライブラリ名を入力。
3.インストールをクリック。

2. 基盤に備え付けの LED・カラーLED・ブザーを 動かしてみよう

サンプルプログラム


から、研究・試作開発まで幅広くご利用いただけます。

マニュアル

マニュアル

サンプル例

Test_Blink.zip：出荷時サンプル用。

Blink.zip：LEDの点滅例

Tone.zip：圧電ブザーの駆動例

RGBLED.zip：カラーLEDの点滅例

Servo.zip：SG90を使ったサーボモータの駆動例

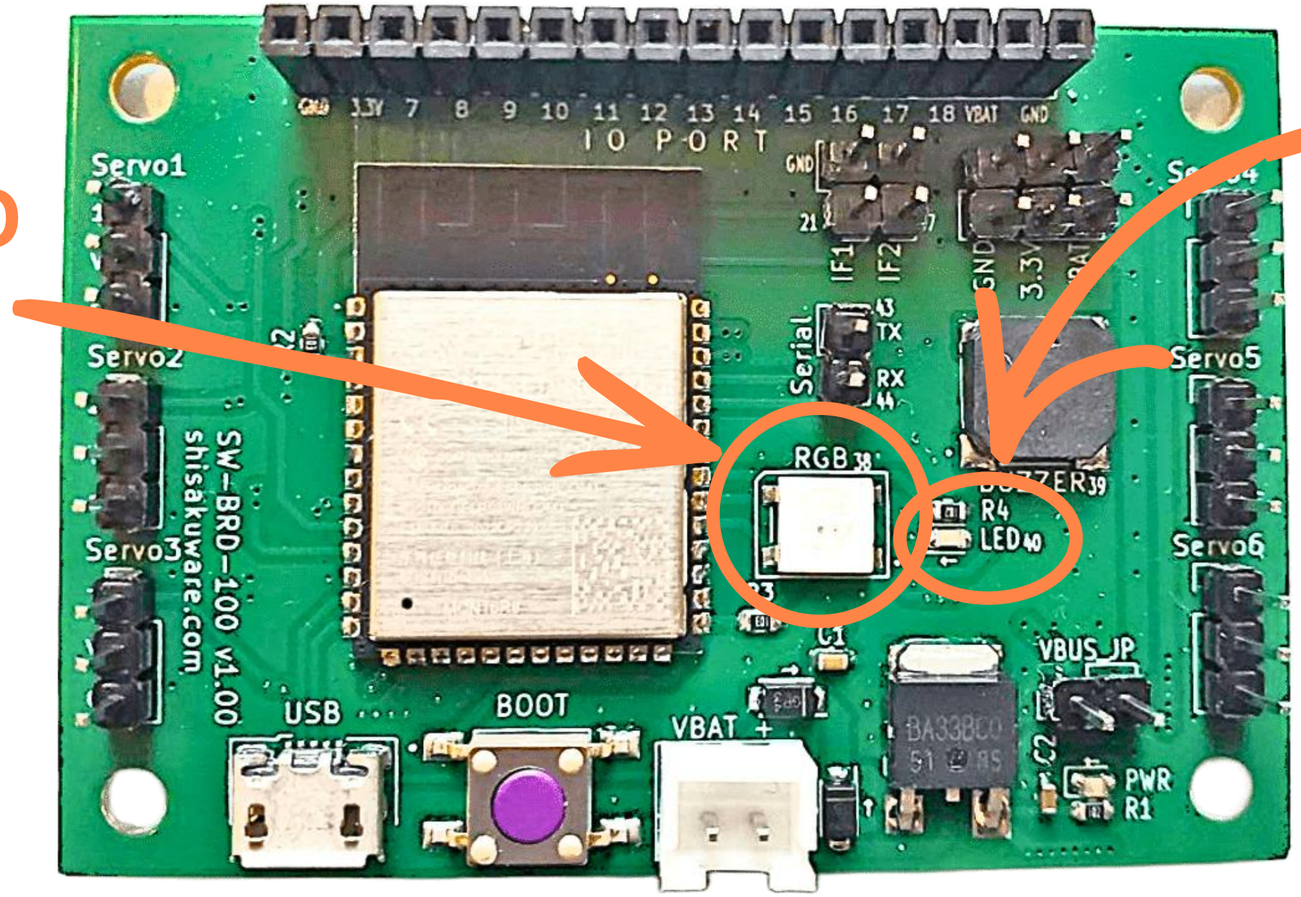
RotationServo.zip：FS90Rを使ったサーボモータの駆動例

https://shisakuware.com/product/sw-brd-100/

当社ホームページにサンプルプログラムがございます。
クリックしていただけますとダウンロードが始まります。

LEDとは

LED（Light Emitting Diode）は電流を流すと光る部品。
光の3原色（赤・緑・青）を一つの電子部品にしたカラーLEDもあります。



カラーLED

LED

基盤の電子部品には番号が
振られていて
プログラムではこの番号を
指定して操作するよ！
（例）LED→40番

Pt.1-40番のLEDを点滅させる

```
6
7   void setup() {
8
9     pinMode(40, OUTPUT);     // 40番のピンを出力モードに変更
10
11  }
12
13  void loop() {
14
15    digitalWrite(40, HIGH); // 40番のピンをON
16    delay(500);
17
18    digitalWrite(40, LOW); // 40番のピンをOFF
19    delay(500);
20  }
21
```

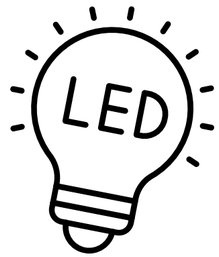
サンプル例：Blink.zipのプログラム

pinMode()とは
ピンの役割を決める
INPUT　：入力
OUTPUT：出力
入力 → センサーを読む
出力 → LEDやモーターを動かす
書き方：　pinMode(ピン番号, 役割)

digitalWrite()とは
ピンをON / OFFする
HIGH：ON
LOW：OFF
書き方：　digitalWrite(ピン番号, 状態)

Pt.2-40番のLEDを点滅させる

delay()とは
プログラムを一時停止する命令
書き方：　delay(時間);
時間＝ミリ秒(ms)



基盤をUSBでパソコンに接続しボードを選択。
右矢印の書き込みをクリック。
書き込み完了すると、40番のLEDが点滅します。

38番のカラーLEDの色を変える

```
#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel led(1, 38, NEO_GRB+NEO_KHZ800);

void setup() {

  led.begin();

}

void loop() {

  // 赤
  led.setPixelColor(0, led.Color(10, 0, 0)); // R, G, B
  led.show();
  delay(500);

  // 緑
  led.setPixelColor(0, led.Color(0, 10, 0));
  led.show();
  delay(500);

  // 青
  led.setPixelColor(0, led.Color(0, 0, 10));
  led.show();
  delay(500);

  // 赤緑青　＝白色
  led.setPixelColor(0, led.Color(10, 10, 10));
  led.show();
  delay(500);

  // 消す
  led.clear(); led.show();
  delay(500);
}
```

ライブラリマネジャーから
[Adafruit Neopixel by Adafruit]をインストール。
LED を使うための機能を読み込む。
書き方：　#include <ライブラリ名>

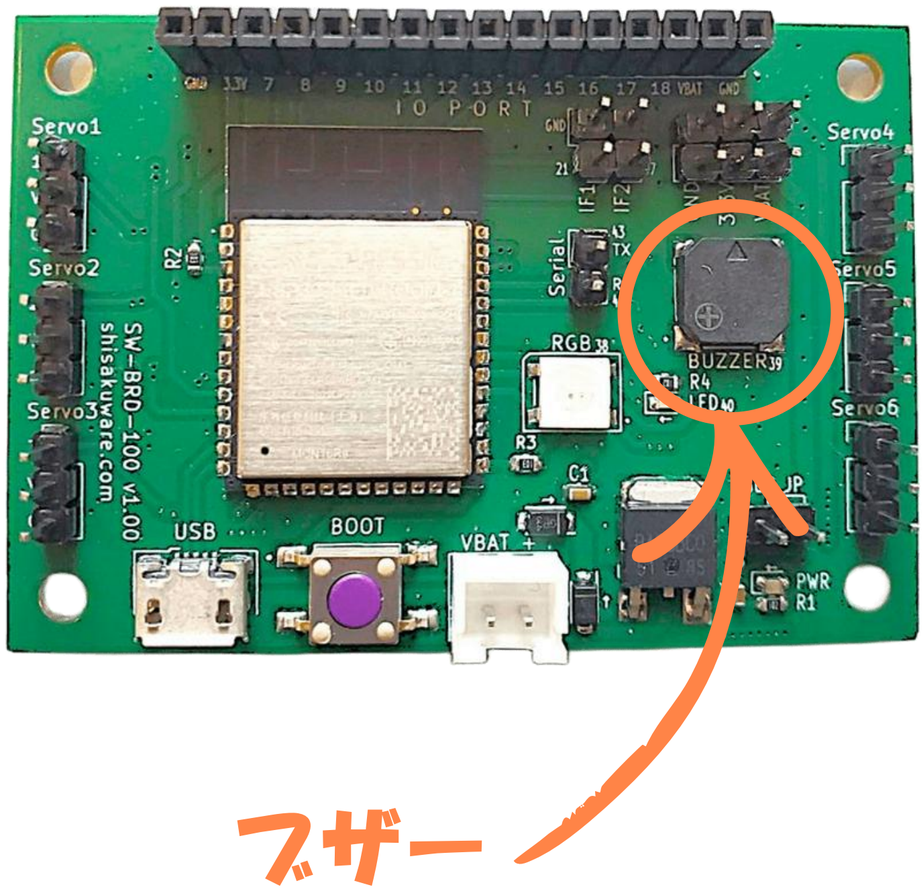
Adafruit_NeoPixel led(1, 38, NEO_GRB + NEO_KHZ800);
NeoPixel LED を使うための設定を作っている。
□構造
Adafruit_NeoPixel　変数名　(LED数, 接続ピン, LEDタイプ)

begin();　初期化
setPixcelColor();　色設定(RGBの（0～255）で値を指定)
show();　LED表示
clear();　LED消灯

書き込み完了すると、38番のカラーLEDが点灯します。

サンプル例：RGBLED.zipのプログラム

ブザー（BUZZER）とは



電子部品のブザーは、
電気信号を音（警告音・確認音）
に変換する音響部品です。
Arduino IDEに標準で備わっている
tone()関数で操作できます。

関数とは
処理をまとめたプログラム。
tone()のカッコ内に
指定した値を
入力するだけで処理を
行ってくれる機能。
digitalWrite()も関数のひとつ。

39番のブザーを鳴らす

```
void setup() {
}

void loop() {

  tone(39, 261, 200);delay(200); // ド
  tone(39, 293, 200);delay(200); // レ
  tone(39, 329, 200);delay(200); // ミ
  tone(39, 349, 200);delay(200); // ファ
  tone(39, 391, 200);delay(200); // ソ
  tone(39, 440, 200);delay(200); // ラ
  tone(39, 493, 200);delay(200); // シ
  tone(39, 523, 200);delay(200); // ド

  delay(1000);
}
```

サンプル例：Tone.zipのプログラム

tone(39, 261, 200);
書き方：　tone(ピン番号, 周波数, 鳴らす時間);
時間＝ミリ秒(ms)
noTone(ピン番号);　ピン番号の音を止める

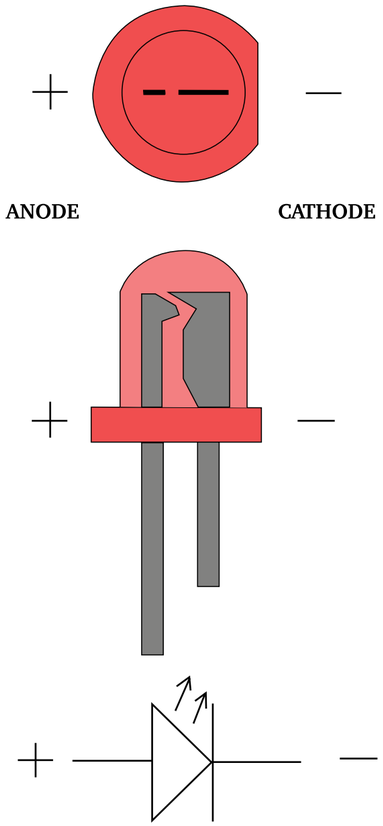
周波数とは音の高さ
261→ド
293→レ
329→ミ
349→ファ
391→ソ
440→ラ
493→シ
523→ド

基盤をUSBでパソコンに接続し
ボードを選択。
右矢印の書き込みをクリック。
書き込み完了すると、
39番のブザーが鳴ります。

3. LED点滅（Lチカ）

LED（発光ダイオード）とは

- 電流を流すと光る半導体素子
- 極性あり（＋と−を逆にすると光らない）
- 低消費電力で長寿命



足の長い方が
アノード（+）

足の短い方が
カソード（-）

LEDの接続方法

- LED + 抵抗を直列に接続
- 抵抗は必ず必要（電流制限）

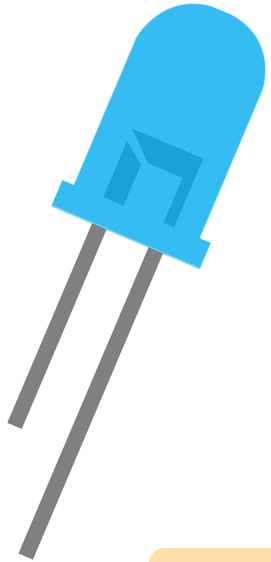
なぜ抵抗が必要か？

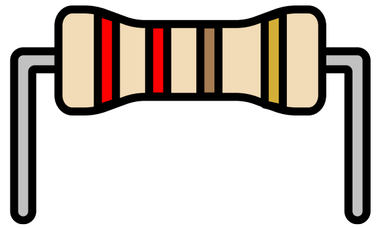
- 電流を適切な値に制限するため
- LEDの破損防止

今回使用したLED定格と抵抗値

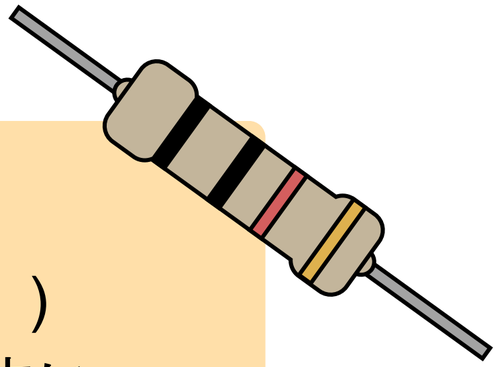
電源電圧：3.3V
LED順方向電圧：2.0V
定格電流：10mA（0.01A）
R = (3.3 - 2.0) / 0.01 = 130Ω
標準抵抗値の220Ωの抵抗を使用する。

LED回路





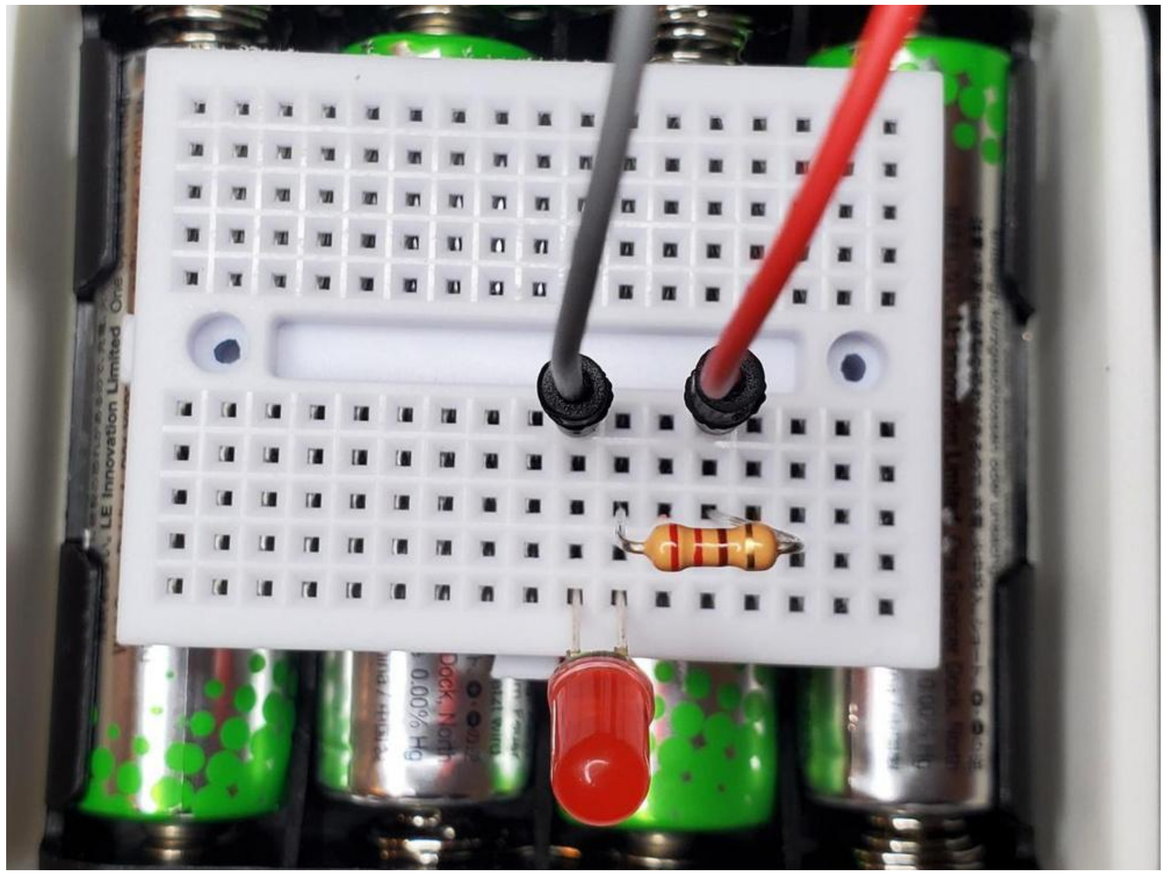
LEDには必ず抵抗が必要です。
理由：電流が流れすぎるとLEDが壊れる



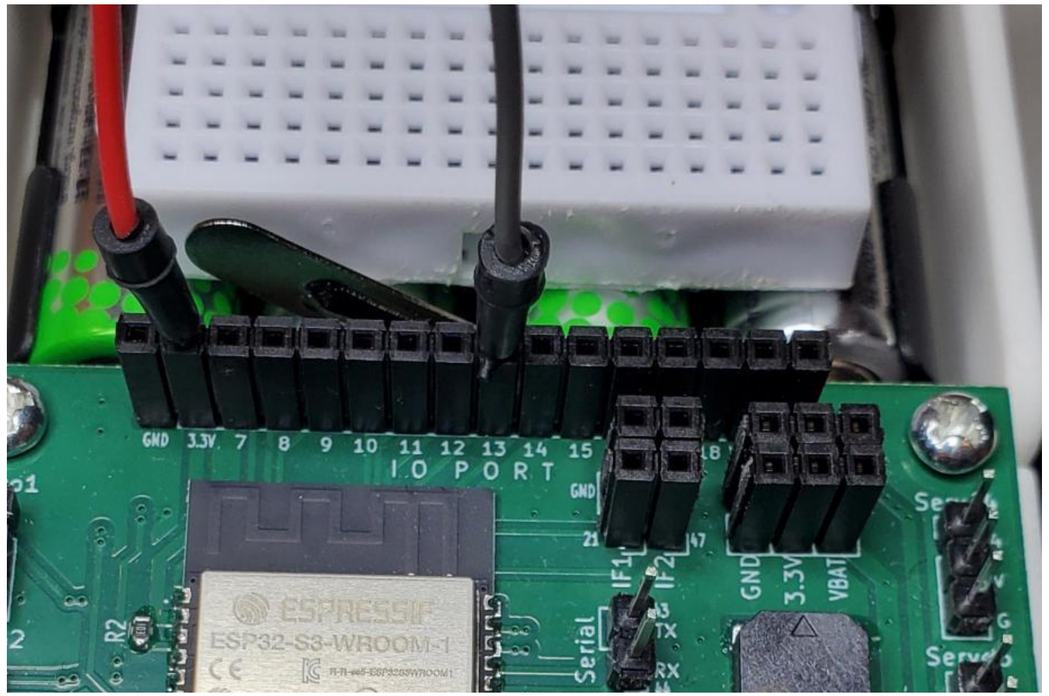
抵抗(電気抵抗)とは
電流の流れにくさを表した数値(単位：Ω［オーム］)
値が大きいほど電気が流れにくく、小さいほど流れやすい。

回路を流れる「電圧」と「電流」がわかれば、以下の式で計算できます。
抵抗（Ω）＝電圧（V）÷電流（A）

LEDをプログラムで点滅させよう
＊Lチカ回路＊

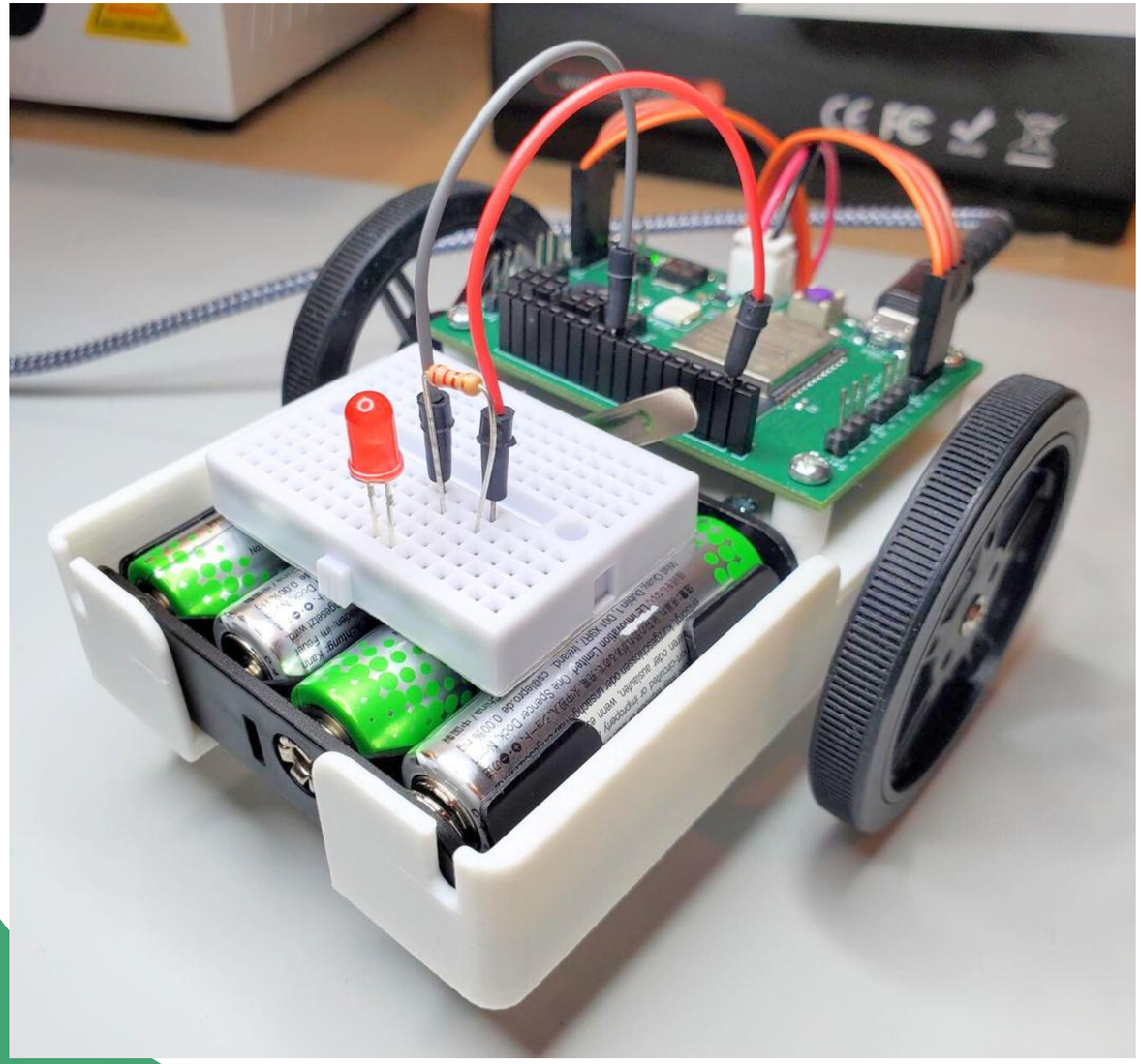


ブレッドボードは縦5点が
内部の端子により接続されている



ロボットキットのピンからLEDを制御する。
3.3V電源とLEDの間に抵抗を直列に挿入し、アノード側に接続する。カソード側はGNDに接続する。

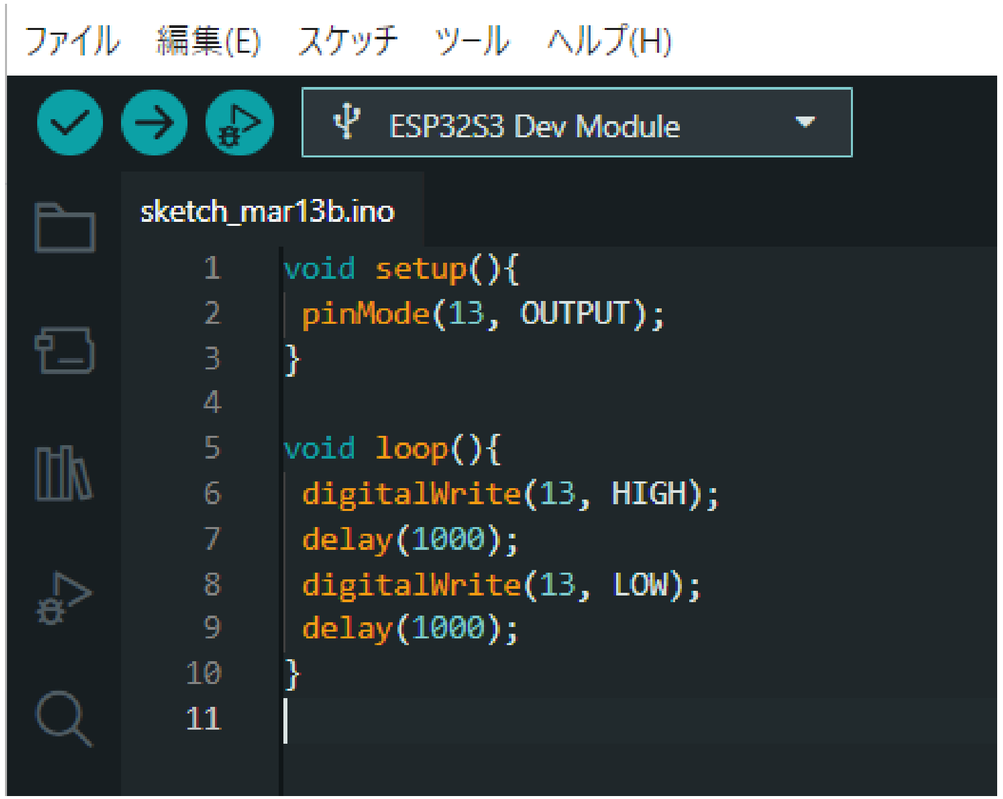
LEDをプログラムで点滅させよう
＊Lチカ回路＊



アノード（＋）→電源側
カソード（−）→GND側

電流は「3.3V → 抵抗 → LED → GND」の順に流れる

Lチカプログラム



- 13番のピン番号を出力に設定
- digitalWrite()でON/OFF
- delay()で1秒間(1000ミリ秒)停止

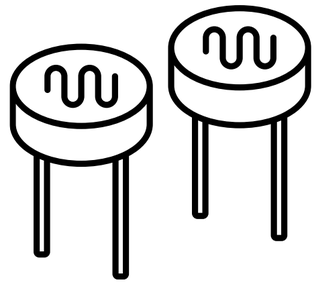
基盤をUSBでパソコンに接続しボードを選択。
右矢印の書き込みをクリック。
書き込み完了すると、LEDが点滅します。

4. センサー

センサーとは

温度、光、距離、人の動きなど、周囲の物理的な変化を検出し、
電気信号としてマイコンボードに入力する電子部品

人感センサー：赤外線で人の動きを検知
（例）人が近づいたらLEDを点灯する
超音波距離センサー：超音波を出しそれが物体に反射して戻ってくるまでの時間を測り距離を計算する
（例）ロボットの障害物検知
光センサー：周囲の明るさを検知
（例）自動点灯



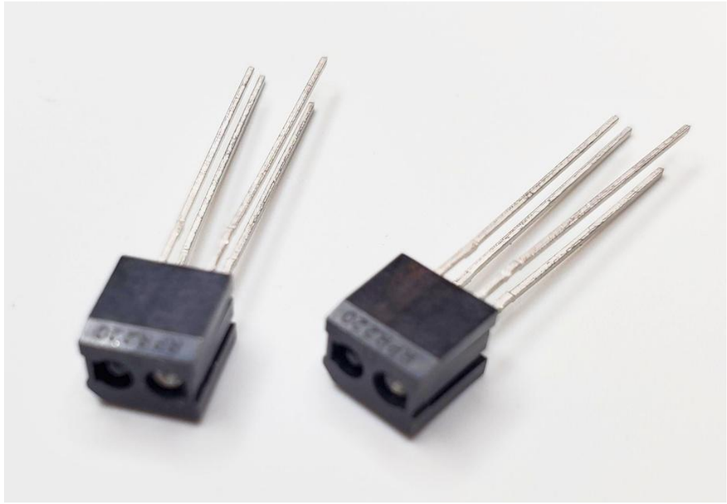
フォトリフレクタ

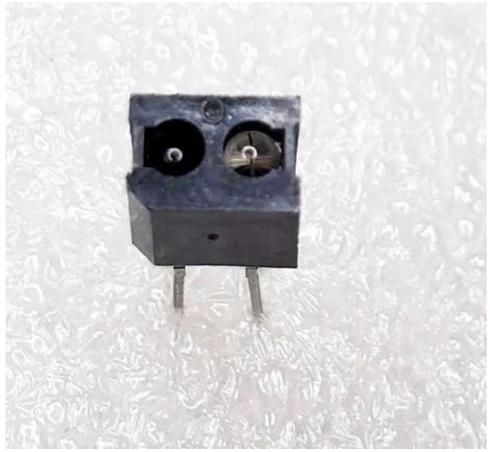
LEDから発行する赤外線（目に見えない光）を出して、
その反射光を感知するセンサー。

反射光検出の仕組み

白 → 光を反射
黒 → 光を吸収
反射量の差で色や物体を判別

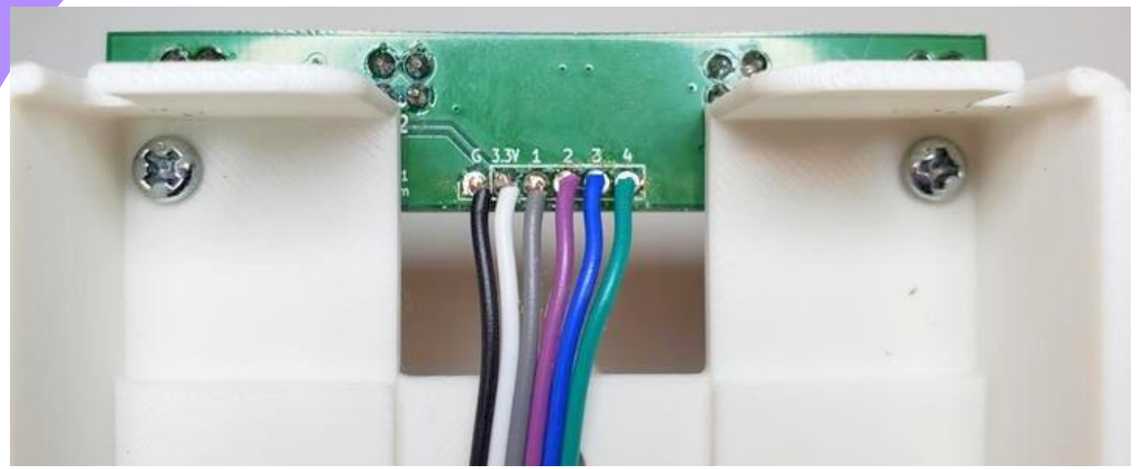
物があるとき：
光が跳ね返ってくるので、センサーが「物がある！」と気づきます。
物がないとき:
光はどこまでも飛んでいくので、
センサーには何も戻ってきません。

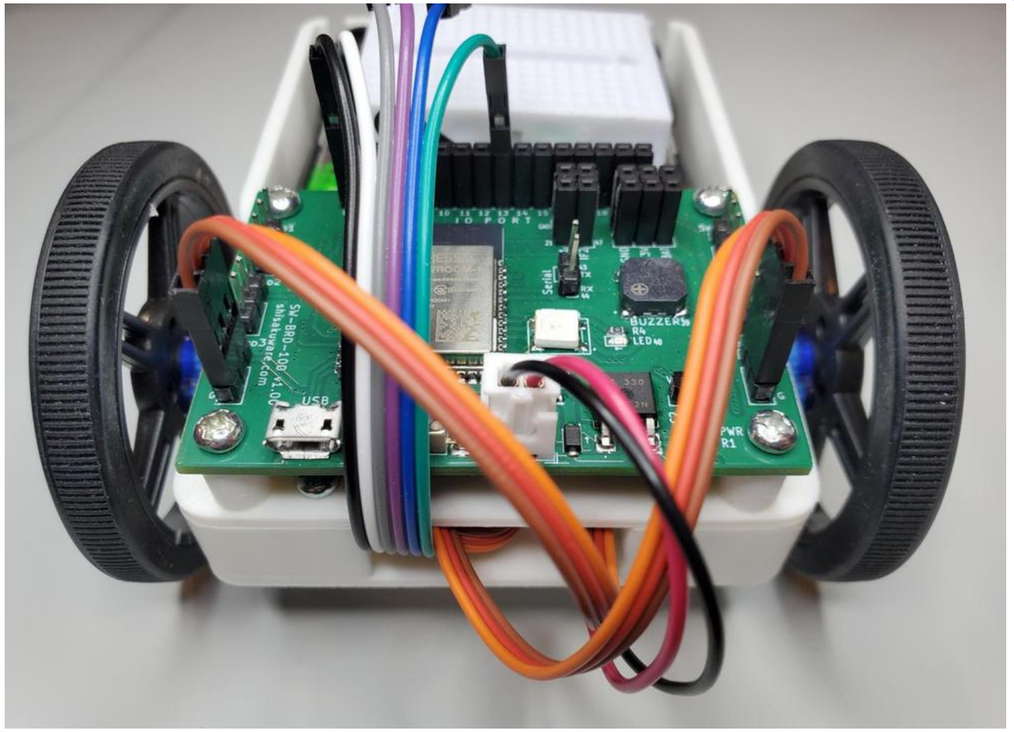


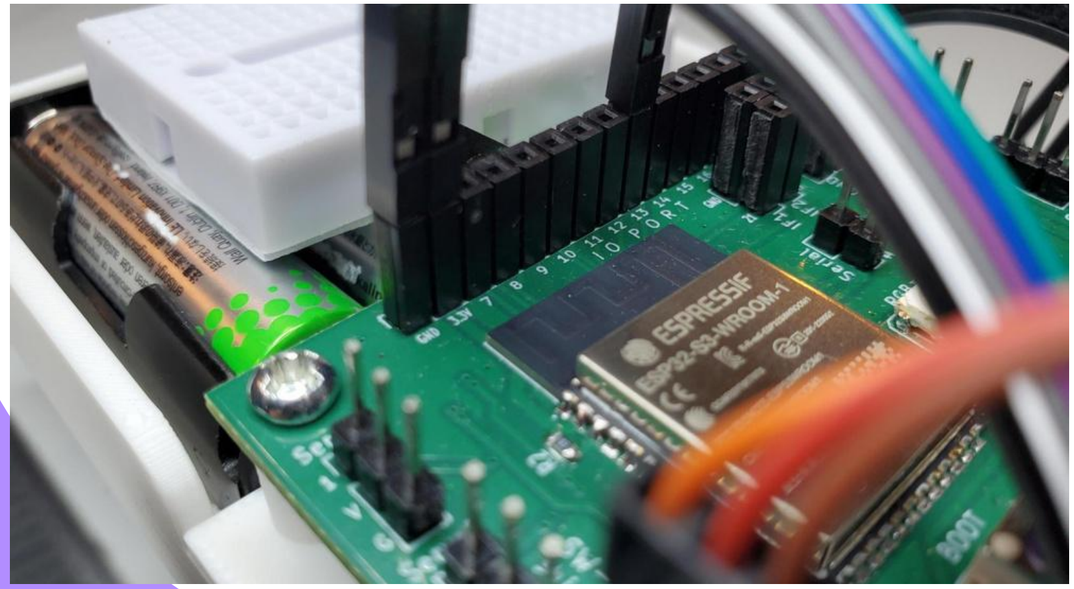


フォトリフレクタが搭載されているライントレーサ基盤を
使用して反射光の値を見ていきます。

配線例







ピン配置例

3.3V：3.3V

GND：GND

４　　：13番ピン

センサー読み取り用に4番ピンを開発基盤の13番のソケットに繋ぎます。

Serial通信

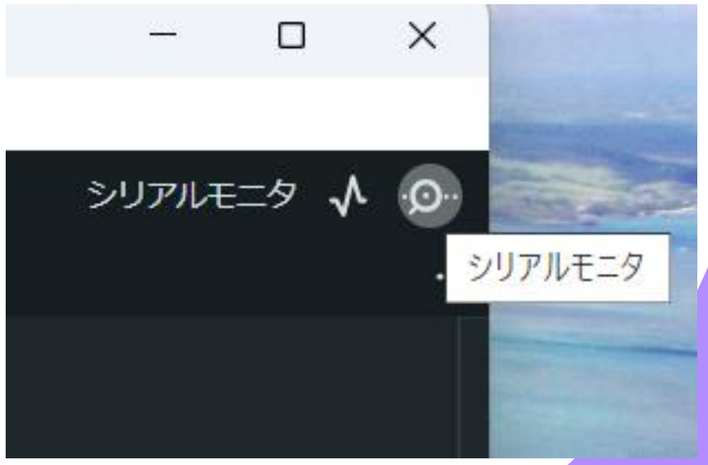
Arduinoとパソコンがデータをやり取りする機能

用途

- センサー値の確認
- プログラムの動作確認
- デバッグ（間違いを見つけ修正する作業）

Serial.begin(通信速度);　Serial通信を開始する命令
Serial.print();　パソコンにデータを表示する命令
Serial.println();　改行つきでデータをパソコンに表示する命令
（例）Serial.print("Hello");　Hello と表示する

シリアルモニタとは
Arduinoから送られたデータを見る画面
開き方：[ツール→シリアルモニタ]または[右上の虫眼鏡アイコン]



センサー値を読み込み表示するプログラム

センサーの値を読み取ってパソコンに表示するプログラム

```
1
2  int sensor;
3
4  void setup() {
5    Serial.begin(9600);
6  }
7
8  void loop() {
9    sensor = analogRead(13);
10   Serial.println(sensor);
11   delay(1000);
12 }
13
```

analogRead(ピン番号);とは
センサーのアナログな情報（明るさ、温度など）
を、読み取る命令です

① 変数の宣言
int sensor;
センサーの値を保存するための変数

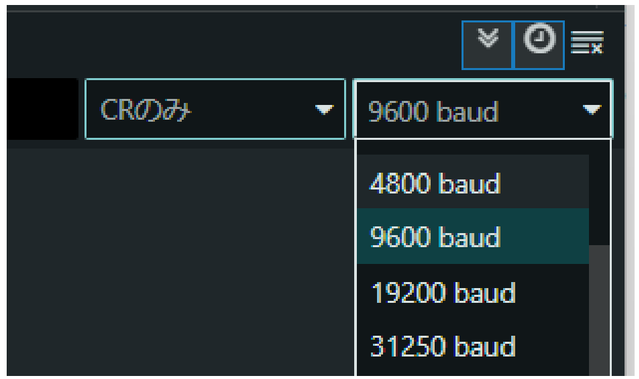
② setup()
Serial.begin(9600);
パソコンと通信を開始する

③ loop()
センサー値を読む
sensor = analogRead(13);
13番ピンのセンサー値を読み取りsensorに保存する

④ シリアルモニタに表示
Serial.println(sensor);
センサー値をシリアルモニタに表示す
る

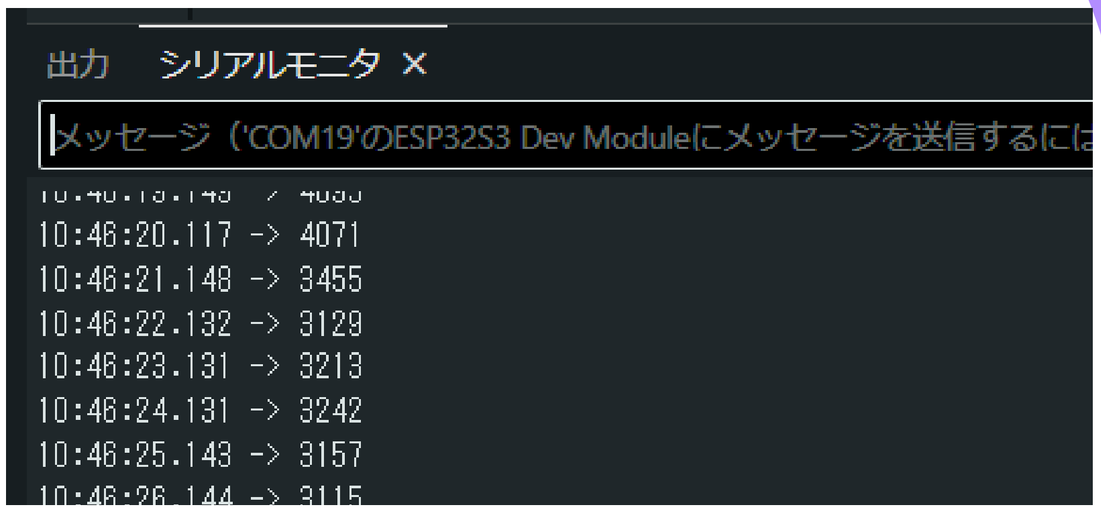
シリアルモニタを開きセンサー値を確認する

黒いものと白いものにセンサーを当ててセンサーの値の変化を確かめてみましょう。



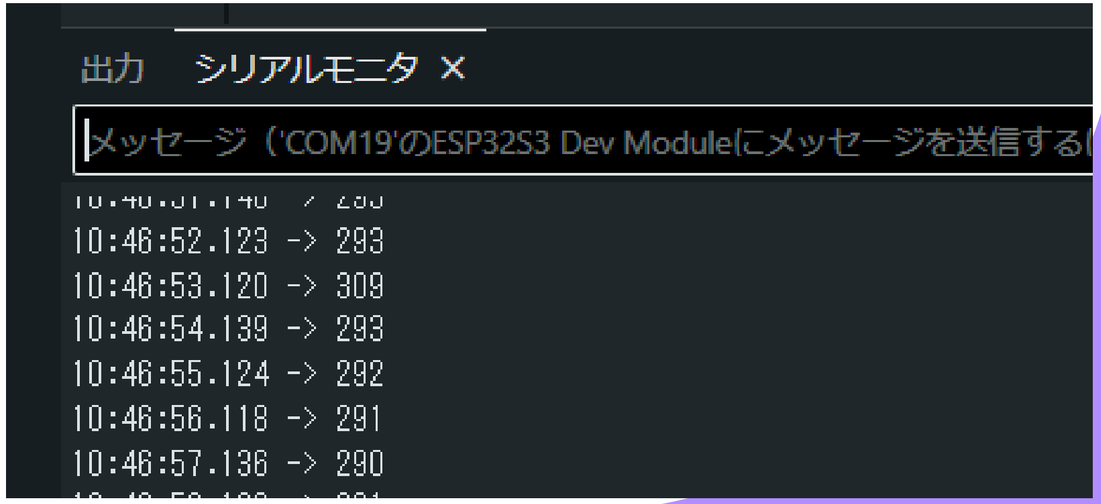
シリアルモニタを開いたら
右端の通信速度がプログラム
で設定した速度と同じか確認

黒のラインを検出



このライントレーサ基盤は
黒を検出したとき値が大きくなるセンサー設定

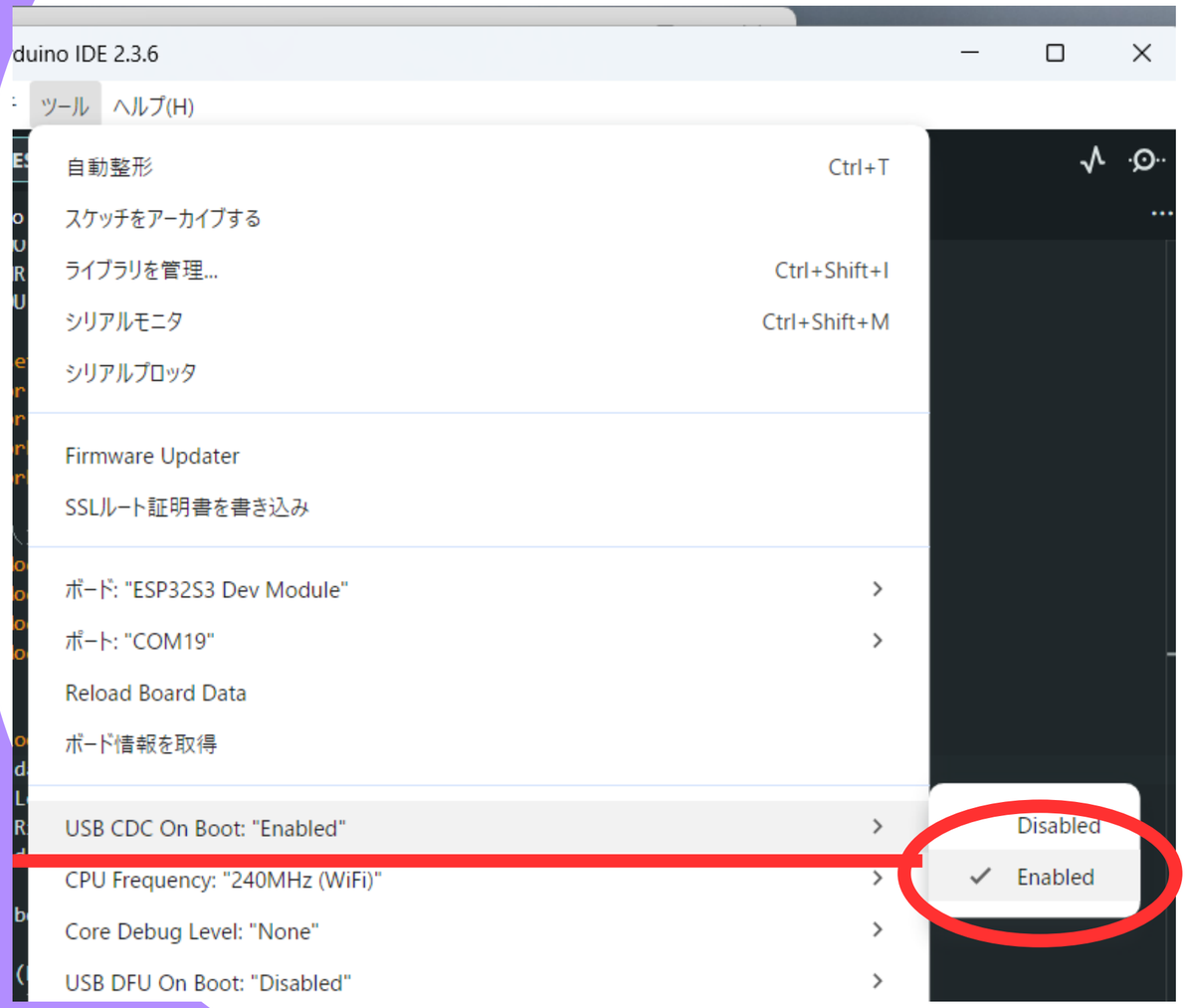
白の床を検出



フォトリフレクタの値
大きい → 黒
小さい → 白

黒 → 光を吸収 → 値が大きい
白 → 光を反射 → 値が小さい

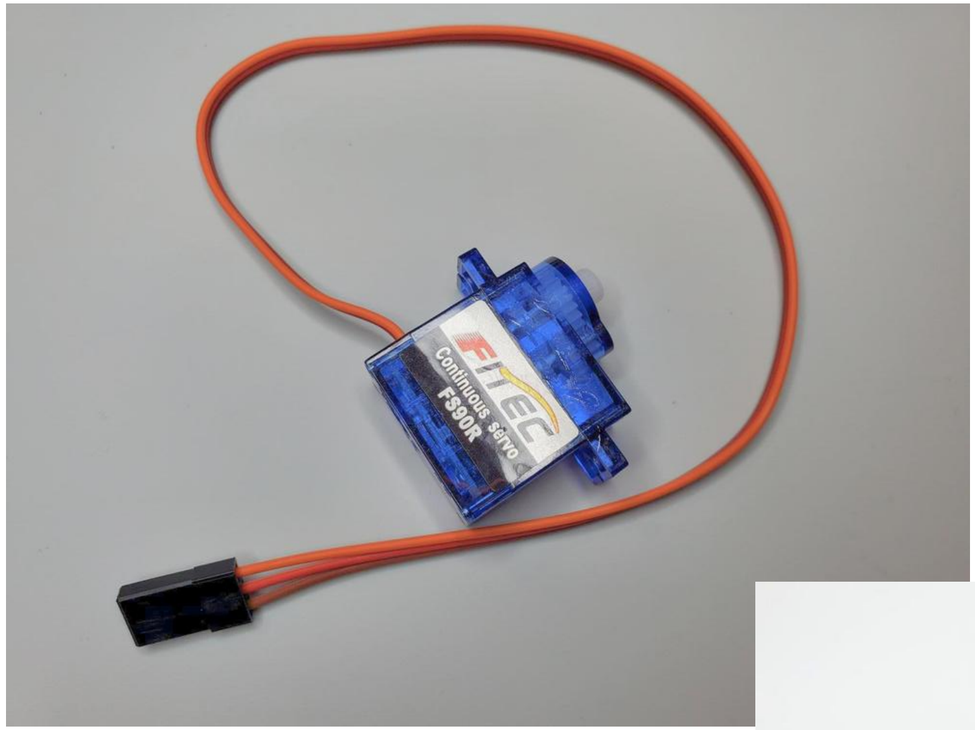
シリアルモニタにメッセージが表示されない場合



ツール ＞ USB CDC On Boot: ＞ “Enabled”
Enabledを選択

5. モーター制御

マイクロサーボモーター

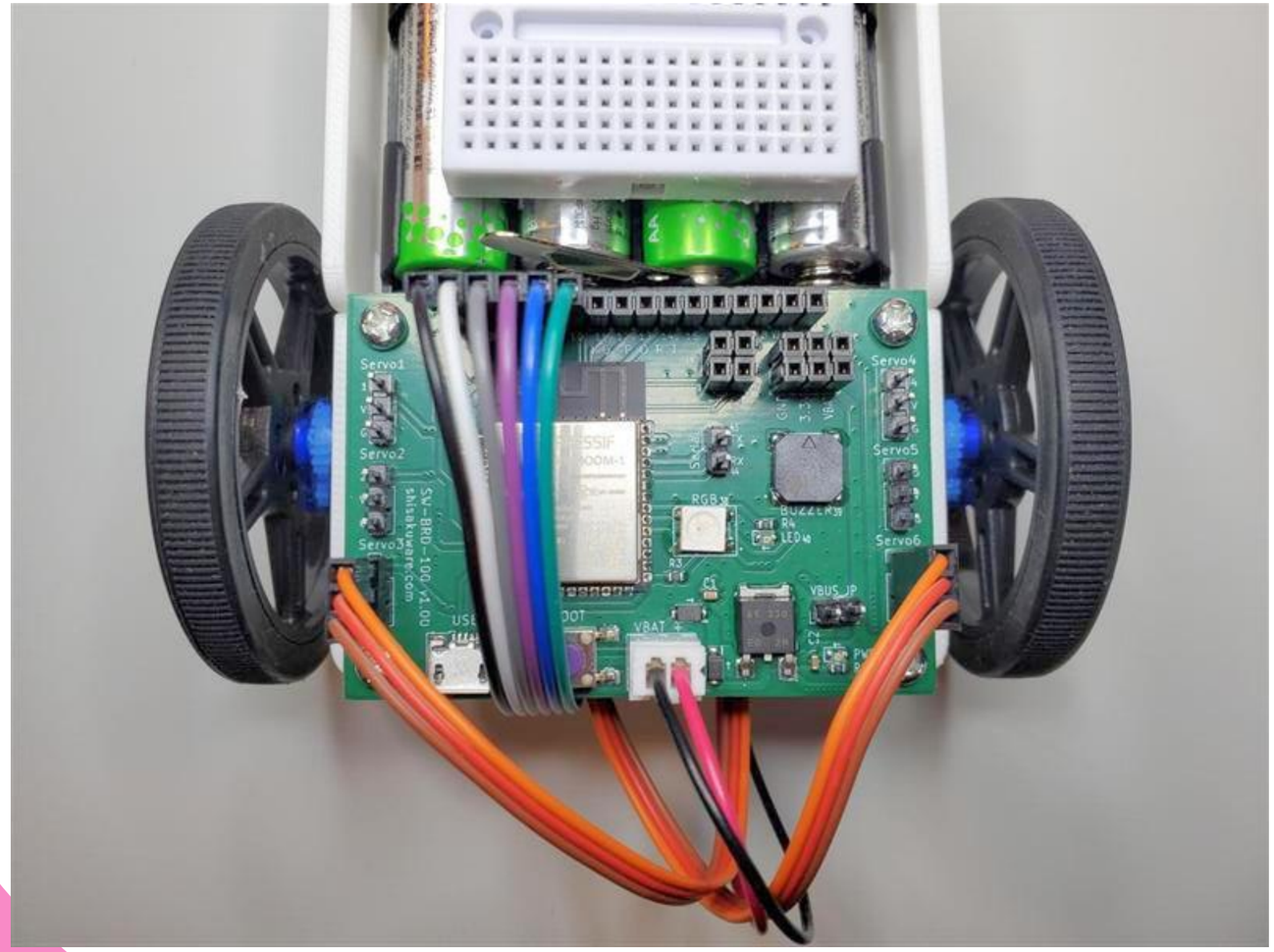


マイクロサーボモーターとは
指定した角度に動くモーター
電子工作やロボット制作でよく使われます。

サーボモーターの特徴
- マイコンから角度を指定して動かせる
- 指定した位置で止まる
- ロボットの関節などに使われる

サーボモーターの種類
① 通常型サーボ
0°～180°の範囲で動く
指定した角度で止まる
② 連続回転サーボ
360°以上回り続ける

マイクロサーボモーターの配線・接続



赤　　：電源
茶／黒　　：GND
オレンジ／黄：信号

配線例
Servo3：左のモーター
Servo6：右のモーター

マイクロサーボの種類

通常のマイクロサーボ（位置制御サーボ）

特徴

- 回転できる角度は 約0°～180°
- 指定した角度に 正確に動いて止まる
- その位置を 保持する

用途

- ロボットの関節
- ラジコン飛行機の舵
- カメラの向きを動かす装置（パン・チルト）

連続回転マイクロサーボ

特徴

- 角度ではなく 回転し続ける
- 信号によって 正転・逆転・停止を制御できる
- 回転速度も調整できる

用途

- ロボットの車輪
- コンベア
- 回転する機構の駆動

通常のマイクロサーボプログラム

サーボモーターを0°、90°、180° の角度に動かす

```ino
# include <ESP32Servo.h>

Servo servo1;

void setup() {
  servo1.setPeriodHertz(50);
  servo1.attach(1, 400, 2500);  // 1ピンにServoを接続。極性に注意
}

void loop() {
  servo1.write(0); // 角度[deg]
  delay(1000);

  servo1.write(90);
  delay(1000);

  servo1.write(180);
  delay(1000);

  servo1.write(90);
  delay(1000);
}
```

サンプル例：Servo.zipのプログラム

ライブラリマネジャーから
「ESP32Servo by Kebin..etc」ver.3.0.9 をインストール
サーボモータを使うための機能を読み込む。
書き方：#include <ESP32Servo.h>

Servo servo1;
servo1 という名前でサーボモーターを操作できるようにする

servo1.setPeriodHertz(50);
サーボ用の信号設定
50Hz → サーボモーターの標準信号

servo1.attach(1, 400, 2500);
1：接続ピン
400：最小パルス
2500：最大パルス
1番ピンにサーボを接続する設定

servo1.write(角度);
サーボモーターを
指定した角度に動かす

連続回転マイクロサーボプログラム

サーボモーターを回転させる

```ino
# include <ESP32Servo.h>

Servo servo1;

void setup() {
  servo1.setPeriodHertz(50);
  servo1.attach(1, 500, 2400); // ピン1のサーボを動かす
}

void loop() {
  for (int pos = 90; pos <= 180; pos += 1) {
    servo1.write(pos);
    delay(20);
  }
  for (int pos = 180; pos >= 0; pos -= 1) {
    servo1.write(pos);
    delay(20);
  }
  for (int pos = 0; pos < 90; pos += 1) {
    servo1.write(pos);
    delay(20);
  }
}
```

サンプル例：RotationServo.zipのプログラム

#include <ESP32Servo.h>
サーボモーターを動かすための機能を読み込む

Servo servo1;
servo1という名前でサーボモーターを操作する

servo1.setPeriodHertz(50);
servo1.attach(1, 500, 2400);
サーボモーターの設定を行う
・PWM信号設定（50Hz）・ピン1にサーボ接続

servo1.write(角度);
サーボモーターを動かす命令

for (int pos = 90; pos <= 180; pos += 1)
for文（繰り返し）
90°から180°まで1度ずつ動かす
つまり
90 → 91 → 92 → … → 180

動きの流れ
① 90° → 180°
② 180° → 0°
③ 0° → 90°
これを 繰り返す

6. ロボット走行

二輪ロボット

2つの車輪で移動し、バランスを取りながら自律走行するロボットのことです。

仕組み

左右の車輪を別々のモーターで動かし、速度差をつけることで前進・後退・旋回を制御します。

制御例

前進

左モーター → 前

右モーター → 前

左旋回

左モーター → 停止

右モーター → 前

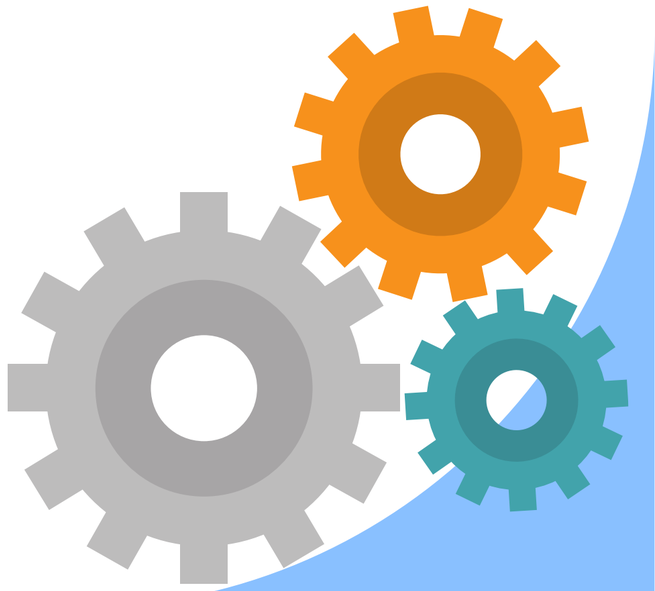
右旋回

左モーター → 前

右モーター → 停止

遠隔操作二輪駆動ロボット

Shisaku Rover



Pt1.-サーボモーター制御プログラム

左右の車輪で方向を制御する

```
1   #include <ESP32Servo.h>
2
3   Servo leftServo;
4   Servo rightServo;
5
6   void setup() {
7     leftServo.setPeriodHertz(50);
8     rightServo.setPeriodHertz(50);
9
10    leftServo.attach(1, 500, 2400);
11    rightServo.attach(4, 500, 2400);
12  }
13
```

#include <ESP32Servo.h>
サーボモーターを動かすための機能を読み込む

Servo leftServo;
Servo rightServo;
各サーボモーターに操作する名前を設定

leftServo.attach(1, 500, 2400);
rightServo.attach(4, 500, 2400);
左のモーターを1番ピン、右のモーターを4番ピンに設定

Pt2.-サーボモーター制御プログラム

左右の車輪で方向を制御する

```
3
4   void loop() {
5
6     // 前進
7     leftServo.write(100);
8     rightServo.write(80);
9     delay(2000);
0
1     // 左旋回
2     leftServo.write(90); // 停止
3     rightServo.write(80);
4     delay(2000);
5
6     // 右旋回
7     leftServo.write(100);
8     rightServo.write(95); // 停止
9     delay(2000);
0
1   }
2
```

前進

leftServo.write(100);
rightServo.write(80);
左サーボ → 前回転
右サーボ → 前回転

右旋回

leftServo.write(100);
rightServo.write(95);
左サーボ → 前回転
右サーボ → 停止

左旋回

leftServo.write(90);
rightServo.write(80);
左サーボ → 前回転
右サーボ → 停止

write()の値は0 ～ 180 の範囲

90 → 停止
90より大きい → 正回転
90より小さい → 逆回転

サーボ停止位置の調整方法

連続回転サーボは個体差があり、完全停止の位置が90°から
少しずれることがあります。

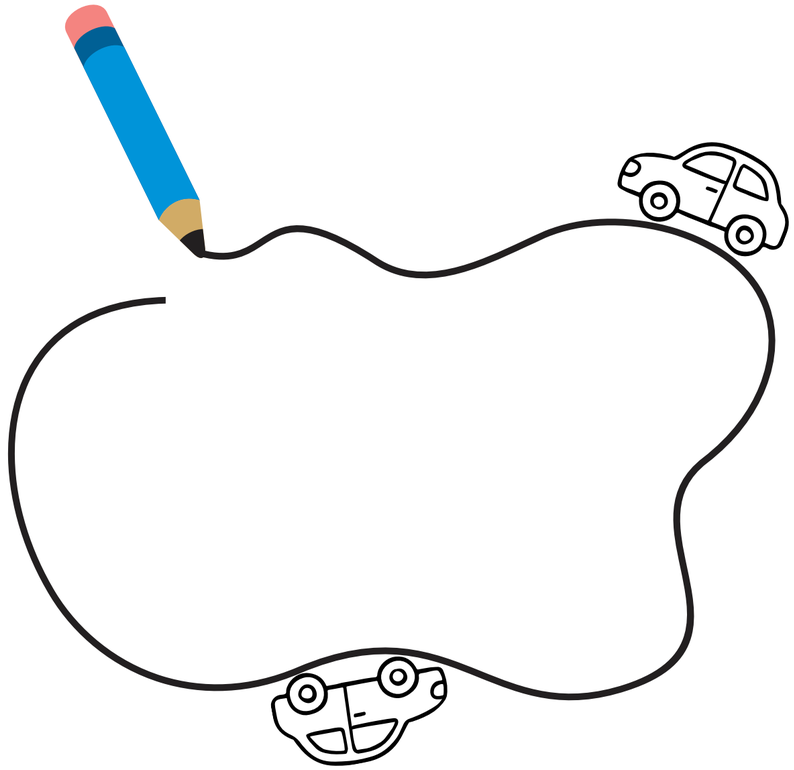
連続回転サーボは
90° = 停止
しない場合があります。サーボごとに停止する角度が少し違います。
（例）rightServo.write(92);
のように調整すると止まります。
85〜95の範囲で調整してください。

7. ラインとレース

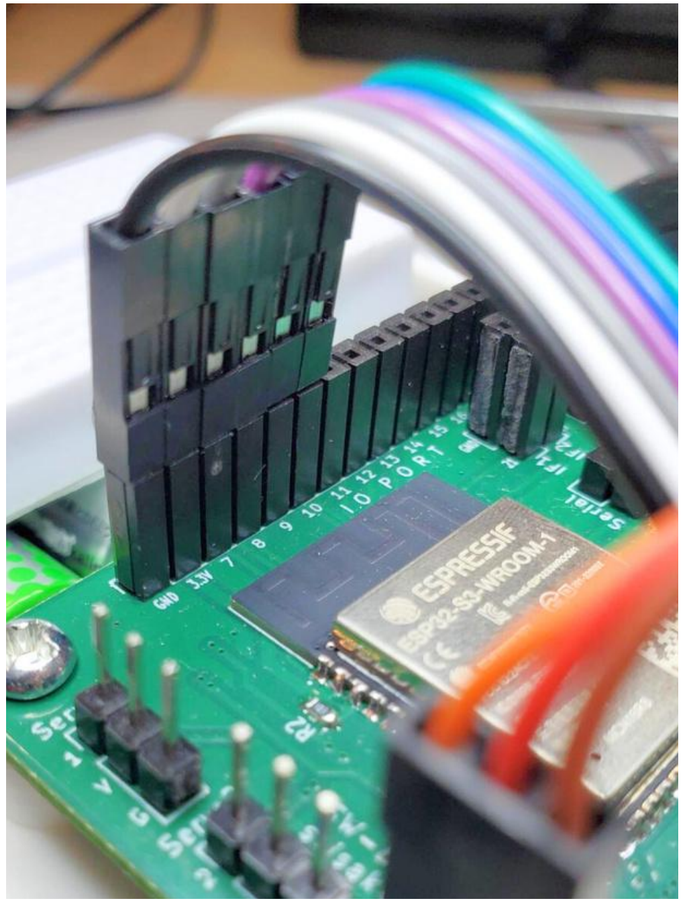
ライントレースとは

床に書かれた線（黒い線）をセンサーで読み取り、
その線に沿ってロボットが走る技術

ライントレーサ基板
4.センサーで使用した
フォトリフレクタが
搭載されているライントレーサ基盤
を使用し、反射光の値を読み取り
黒い線に沿ってロボットが走る
プログラムを書いていきます。



センサー・ピン配置



センサーを動かすために3.3Vに繋ぎます。
GNDは電流の戻り道として機能します。

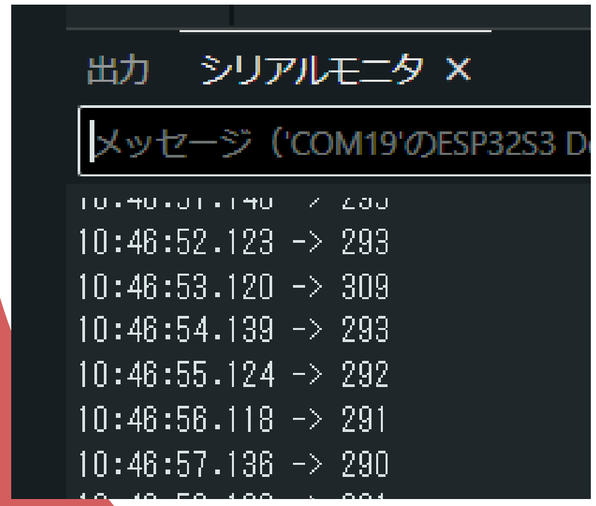
ピン配置例

3.3V：3.3V
GND：GND
１　：7番ピン
２　：8番ピン
３　：9番ピン
４　：10番ピン

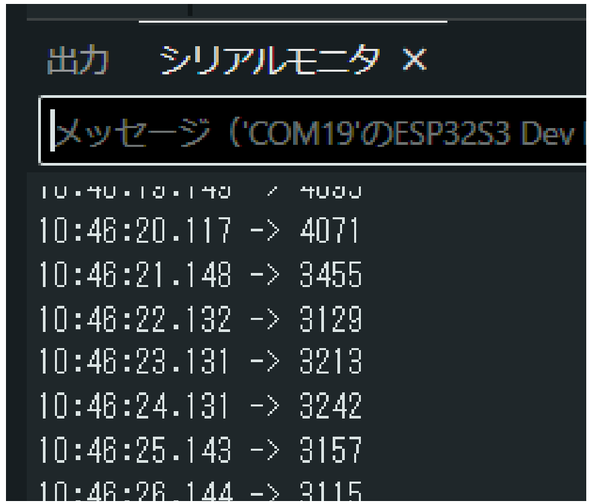
しきい値を決める

しきい値とは判定の境界線（ボーダーライン）のことです。
センサーが受け取る光の量は、周りの明るさや距離によって変化します。
その変化の状態に対して、「ここより光が強ければ『ON（あり）』、弱ければ『OFF（なし）』とみなす！」
という決まりを作るのが「しきい値を決める」という作業です。

白の床を検出



黒のラインを検出



4.センサーのプログラムで
黒いものと白いものにセンサーを当ててセンサーの
値の違いを確かめてみましょう。

白い床： 光がよく反射する（数値：300）
黒い線： 光を吸収する（数値：3000）
しきい値：例えば、その中間の 「1500」 くらいに設定します。こうすることで、「1500より大きければ黒、小さければ白」とはっきり区別できるようになります。ですが、照明の環境やラインの濃さや太さによりライントレース走行の挙動が変わりますので、しきい値を増減して調整してください。

Pt.1-ライントレース制御プログラミング

センサーの値を見てロボットの進む方向を決める

```
#include <ESP32Servo.h>

Servo motorLeft3;   // 左モーター
Servo motorRight6;  // 右モーター

int ONE = 7;
int TWO = 8;   // 内左のセンサー
int THR = 9;   // 内右のセンサー
int FOU = 10;

void setup() {
  motorLeft3.setPeriodHertz(50);
  motorLeft3.attach(3, 400, 2500);
  motorRight6.setPeriodHertz(50);
  motorRight6.attach(6, 400, 2500);

  // 入力に設定
  pinMode(ONE, INPUT);
  pinMode(TWO, INPUT);
  pinMode(THR, INPUT);
  pinMode(FOU, INPUT);
}

```

#include <ESP32Servo.h>
サーボモータを使うための機能を読み込む。

Servo motorLeft3;
Servo motorRight6;
左右のモーターに名前を付けて操作できるようにする

int ONE = 7;
int TWO = 8;
int THR = 9;
int FOU = 10;
ライントレーサ基盤のピン番号とロボット開発基盤に配線したピン番号が連携しているか確認し、それぞれの変数に定義する。

Pt.2-ライントレース制御プログラミング

```
#include <ESP32Servo.h>

Servo motorLeft3;   // 左モーター
Servo motorRight6;  // 右モーター

int ONE = 7;
int TWO = 8;   // 内左のセンサー
int THR = 9;   // 内右のセンサー
int FOU = 10;

void setup() {
  motorLeft3.setPeriodHertz(50);
  motorLeft3.attach(3, 400, 2500);
  motorRight6.setPeriodHertz(50);
  motorRight6.attach(6, 400, 2500);

  // 入力に設定
  pinMode(ONE, INPUT);
  pinMode(TWO, INPUT);
  pinMode(THR, INPUT);
  pinMode(FOU, INPUT);
}

```

motorLeft3.setPeriodHertz(50);
サーボ用の信号設定
50Hz → サーボモーターの標準信号

motorLeft3.attach(3, 400, 2500);
3：接続ピン
400：最小パルス
2500：最大パルス
3番ピンにサーボを接続する設定

pinMode(ONE, INPUT)
pinMode(TWO, INPUT)...
定義済みのピン番号（ONE, TWO など）を入力モードに設定し、ライントレースのセンサ入力として使用する。

Pt.3-ライントレース制御プログラミング

```
53
54  int borderline = 500;
55
56  void loop() {
57    int left_1 = analogRead(ONE);
58    int left_2 = analogRead(TWO);
59    int right_3 = analogRead(THR);
60    int right_4 = analogRead(FOU);
61
62    // 全部黒 → 停止
63    if (left_1 >= borderline && left_2 >= borderline &&
64        right_3 >= borderline && right_4 >= borderline) {
65      stop();
66    }
67    // 中央2つで制御
68    else if (right_3 >= borderline && left_2 >= borderline){
69      forward();
70    } else if (right_3 >= borderline && left_2 <= borderline){
71      turnRight();
72    } else if (right_3 <= borderline && left_2 >= borderline){
73      turnLeft();
74    } else {
75      turnRight(); // ライン見失い
76    }
77
78    delay(10);
79  }
80
```

int borderline = 500;
しきい値を500に設定する。

int left_1 = analogRead(ONE);
int left_2 = analogRead(TWO);
int right_3 = analogRead(THR);
int right_4 = analogRead(FOU);
analogRead()で各センサの値を取得し、それぞれ変数に保存する。
4つあるセンサのうち、中央の2つを用いてラインの検出を行う。

```
if (left_1 >= borderline && left_2 >= borderline &&
 right_3 >= borderline && right_4 >= borderline) {
stop();
}
```
すべてのセンサーがしきい値の500より大きいとき、停止する。（すべて黒）

Pt.4-ライントレース制御プログラミング

```
53
54  int borderline = 500;
55
56  void loop() {
57    int left_1 = analogRead(ONE);
58    int left_2 = analogRead(TWO);
59    int right_3 = analogRead(THR);
60    int right_4 = analogRead(FOU);
61
62    // 全部黒 → 停止
63    if (left_1 >= borderline && left_2 >= borderline &&
64        right_3 >= borderline && right_4 >= borderline) {
65      stop();
66    }
67    // 中央2つで制御
68    else if (right_3 >= borderline && left_2 >= borderline){
69      forward();
70    } else if (right_3 >= borderline && left_2 <= borderline){
71      turnRight();
72    } else if (right_3 <= borderline && left_2 >= borderline){
73      turnLeft();
74    } else {
75      turnRight(); // ライン見失い
76    }
77
78    delay(10);
79  }
80
```

else if (right_3 >= borderline && left_2 >= borderline){
forward();
}
中央右のセンサーの値がボーダーラインより大きい（黒）、また、中央左のセンサーの値がボーダーラインより大きいとき（黒）、前進する。

else if (right_3 >= borderline && left_2 <= borderline){
turnRight();
}
中央右のセンサーの値がボーダーラインより大きく（黒）、また、中央左のセンサーの値がボーダーライン小さいとき（白）、右に旋回する。

else if (right_3 <= borderline && left_2 >= borderline){
turnLeft();
}
中央右のセンサーの値がボーダーラインより小さく（白）、また、中央左のセンサーの値がボーダーラインより大きいとき（黒）、左に旋回する。

Pt.5-ライントレース制御プログラミング

```
53
54  int borderline = 500;
55
56  void loop() {
57    int left_1 = analogRead(ONE);
58    int left_2 = analogRead(TWO);
59    int right_3 = analogRead(THR);
60    int right_4 = analogRead(FOU);
61
62    // 全部黒 → 停止
63    if (left_1 >= borderline && left_2 >= borderline &&
64        right_3 >= borderline && right_4 >= borderline) {
65      stop();
66    }
67    // 中央2つで制御
68    else if (right_3 >= borderline && left_2 >= borderline){
69      forward();
70    } else if (right_3 >= borderline && left_2 <= borderline){
71      turnRight();
72    } else if (right_3 <= borderline && left_2 >= borderline){
73      turnLeft();
74    } else {
75      turnRight(); // ライン見失い
76    }
77
78    delay(10);
79  }
80
```

else {
turnRight();
}
ラインを見失ったとき、右に旋回。

if(条件){}
条件が成立したら、{ }内の処理が実行されます。
else if(条件){}
if文の条件が成立しなかった場合に、次の条件をチェックします。
else{}
ifやelse ifで提示したどの条件にも当てはまらない場合に実行されます。

Pt.6-ライントレース制御プログラミング

```
53
54  void forward() {
55    motorLeft3.write(100);
56    motorRight6.write(80);
57  }
58
59  void turnLeft() {
60    motorLeft3.write(90);
61    motorRight6.write(80);
62  }
63
64  void turnRight() {
65    motorLeft3.write(100);
66    motorRight6.write(90);
67  }
68
69  void stop() {
70    motorLeft3.write(90);
71    motorRight6.write(90);
72  }
```

```
void forward() {
 motorLeft3.write(100);
 motorRight6.write(80);
}
```
前進するプログラムにforward()という名前を付けたもの。

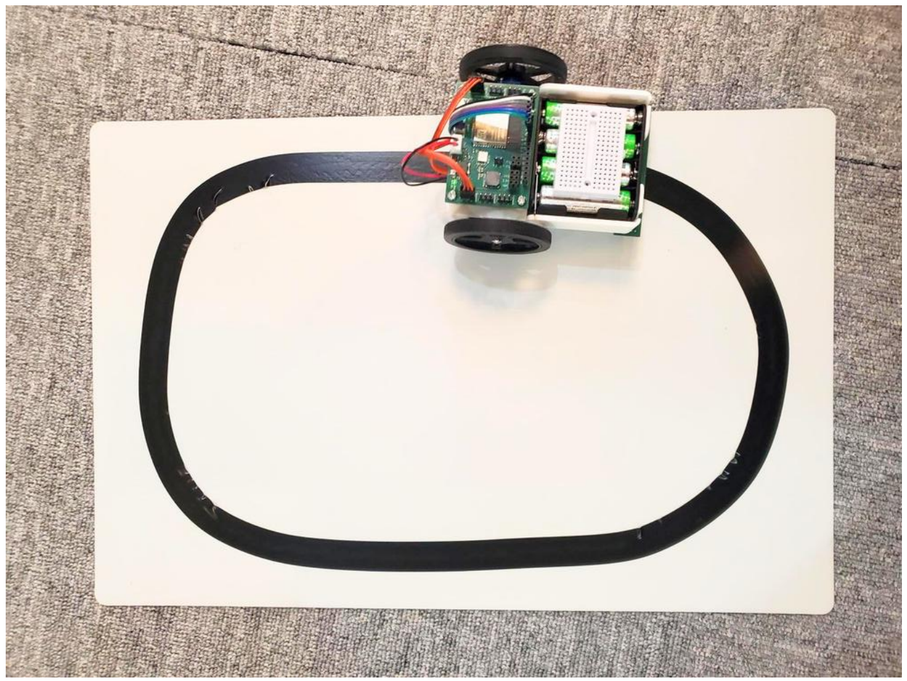
```
void turnLeft() {
 motorLeft3.write(90);
 motorRight6.write(80);
}
```
左旋回するプログラムにturnLeft()という名前を付けたもの。

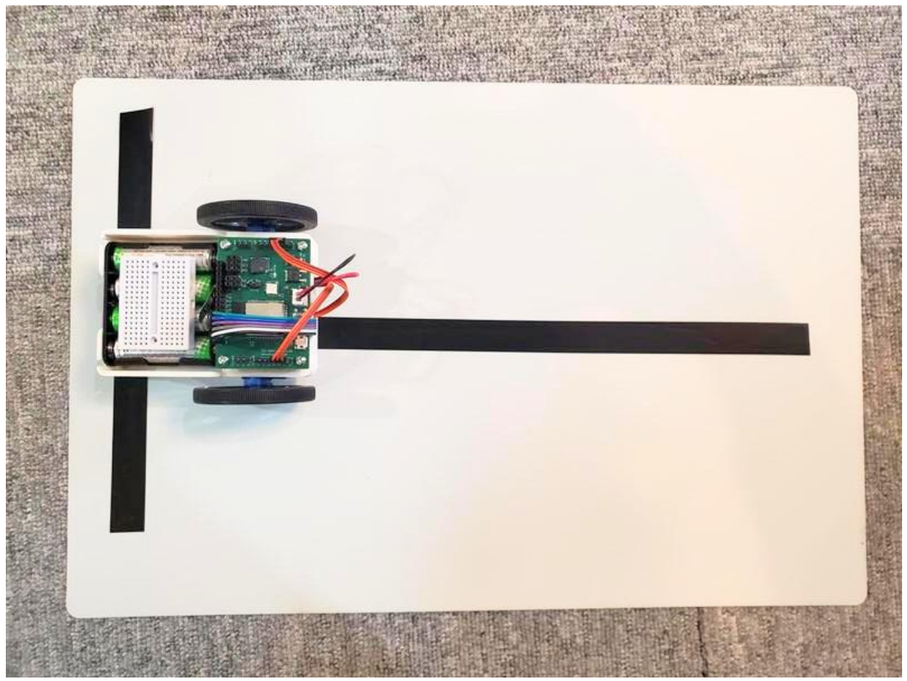
```
void turnRight() {
 motorLeft3.write(100);
 motorRight6.write(90);
}
```
右旋回するプログラムにturnRight()という名前を付けたもの。

```
void stop() {
 motorLeft3.write(90);
 motorRight6.write(90);
}
```
停止するプログラムにstop()という名前を付けたもの。

Pt.7-ライントレース制御プログラミング

Arduinoにプログラムを書き込み、
黒いテープで線路を作り、電源を入れて走らせてみましょう





8. まとめ

学んだこと

Arduino基礎
Arduinoのプログラムはsetup()とloop()で構成されており、繰り返し処理によって動作することを理解した。また、pinMode()やdigitalWrite()などを用いて、マイコンから外部機器を制御できることを学んだ。

LED制御
LEDの点灯・消灯を通して、デジタル出力（HIGH/LOW）の基本を理解した。また、プログラムによって点滅周期を制御できることを学んだ。

センサー
センサーからの値をanalogRead()で取得し、環境に応じて値が変化することを確認した。また、しきい値を用いることで、黒と白のような状態判定が可能であることを学んだ。
しきい値は理論値ではなく、実際のセンサ値のばらつきを考慮して調整する必要があることを理解した。

ロボット制御
センサの入力に応じてモーターの動作を変えることで、条件分岐によるロボット制御が可能であることを学んだ。

ライントレース
複数のセンサを用いてラインの位置を判断し、その結果に応じて進行方向を制御することでライントレースが実現できることを学んだ。
また、しきい値の設定によって走行の安定性が大きく変化することを実験を通して理解した。

トラブルシューティング

Arduino基礎

1.プログラムが動かない

原因：setup()やloop()の書き忘れ・構文エラー

対策：エラーメッセージを確認し、最低限の構造を守る

- 意図しない繰り返し動作

原因：loop()が常に実行されることを考慮していない

対策：フラグ変数やif文で状態管理を行う

LED制御

- LEDが点灯しない

原因：配線ミス、抵抗の付け忘れ、ピン番号の指定ミス

対策：回路図を確認し、正しいピンとGND接続をチェック

- 点滅速度が不安定

原因：delay()の値ミスや処理時間の影響

対策：シンプルなコードで動作確認、必要ならmillis()で制御

- LEDが壊れる

原因：電流制限抵抗なし

対策：必ず適切な抵抗（例：220Ω）を使用

センサー

- しきい値で誤判定

原因：固定値が環境に合っていない

対策：実測値をログ出力して調整（Serial.println）

- 値が変化しない

原因：配線ミス、アナログピン指定ミス

対策：ピン番号と接続確認

- センサーが反応しない、シリアルモニタに文字がひょうじされないとき

原因：Arduinoでのツール未設定

対策：ツール ＞ USB CDC On Boot: ＞ “Enabled”　を選択

ロボット制御

- モーターが動かない

原因：電源不足、配線ミス

対策：別電源が電池切れしていないか、配線されているか確認

- 思った方向に動かない

原因：左右モーターの配線逆

対策：回転方向を確認し、コードまたは配線を修正

トラブルシューティング

ライントレース
・ラインを見失う
原因：しきい値不適切、センサ位置ずれ
対策：しきい値調整、センサ高さ・位置の最適化
・コースによって動作が変わる
原因：床の色・照明条件の違い
対策：環境ごとに再調整