

# Juggernaut

---

爆弾解除でプログラミングと電子回路を学ぼう

2025-11-12

International Cybersecurity Challenge TOKYO 2025 @ホテルニューオータニ幕張

Norimasa TAKANA

# 目次

## 本講義資料の目次

1. はじめに .....	5
1.1 自己紹介 .....	6
1.2 今回の内容 .....	7
2. 事前知識 .....	8
2.1 Juggernaut とは .....	9
2.2 競技の流れ .....	10
2.3 装置の説明 .....	12
2.4 マイコンって何？ .....	13
2.5 マイコンでできること .....	14
2.6 回路に関する基礎知識 .....	15
2.7 Vcc(+) と GND(-) .....	16
2.8 回路の例 .....	17

# 目次

## 本講義資料の目次

2.9 部品紹介: ブレッドボード .....	18
2.10 部品紹介: フルカラー LED .....	19
2.11 部品紹介: トグルスイッチ .....	20
2.12 部品紹介: タクトスイッチ .....	21
2.13 回路のまとめ .....	22
2.14 プログラムに関する基礎知識 .....	23
2.15 ピンの出力 .....	24
2.16 電圧が高いか低いか .....	26
2.17 ピンの入力 .....	27
2.18 プログラムの基本中の基本 .....	30
2.19 プログラムのまとめ .....	32
3. 問題 1: 赤か青か .....	33

# 目次

## 本講義資料の目次

3.1 問題文 .....	34
3.2 概要 .....	35
3.3 配線図 .....	36
3.4 プログラム .....	37
4. 問題 2: unknown sensor .....	40
4.1 配線図 .....	41
4.2 プログラム .....	42
4.3 解説 .....	47
5. 問題 3: ULTRASONIC .....	48
5.1 配線図 .....	49
5.2 プログラム .....	50
6. 付録 .....	56

# 目次

## 本講義資料の目次

6.1 参考文献 .....	57
----------------	----

# 1. はじめに

---

# はじめに

## 自己紹介

**名前:** 高名 典雅 (Norimasa TAKANA)

**所属:** 筑波大学 理工情報生命学術院

**専門分野:** システムソフトウェア, セキュリティ

**経歴:**

- 船橋市立大穴中学校
- 国立木更津工業高等専門学校
- 筑波大学 情報学群 情報科学類
- 筑波大学 理工情報生命学術院
- セキュリティ・キャンプ 講師
- 未踏 スーパークリエータ



図 1: 講師近影

# はじめに

## 今回の内容

今回は爆弾解除を通してプログラミングと電子回路に入門します。  
主な流れはこんな感じです。

- 事前知識の説明 (20 分)
  - ▶ この競技の概要
  - ▶ 回路の基礎
  - ▶ プログラミングの基礎
  - ▶ 問題 1
- 問題 2 (15 分)
  - ▶ 問題に挑戦してみよう
  - ▶ 解説
- 問題 3 (15 分)
  - ▶ 問題に挑戦してみよう
  - ▶ 解説
- まとめ

## 2. 事前知識

---

# 事前知識

Juggernaut とは

回路とプログラムを読み解いて**爆弾**を解除する競技！

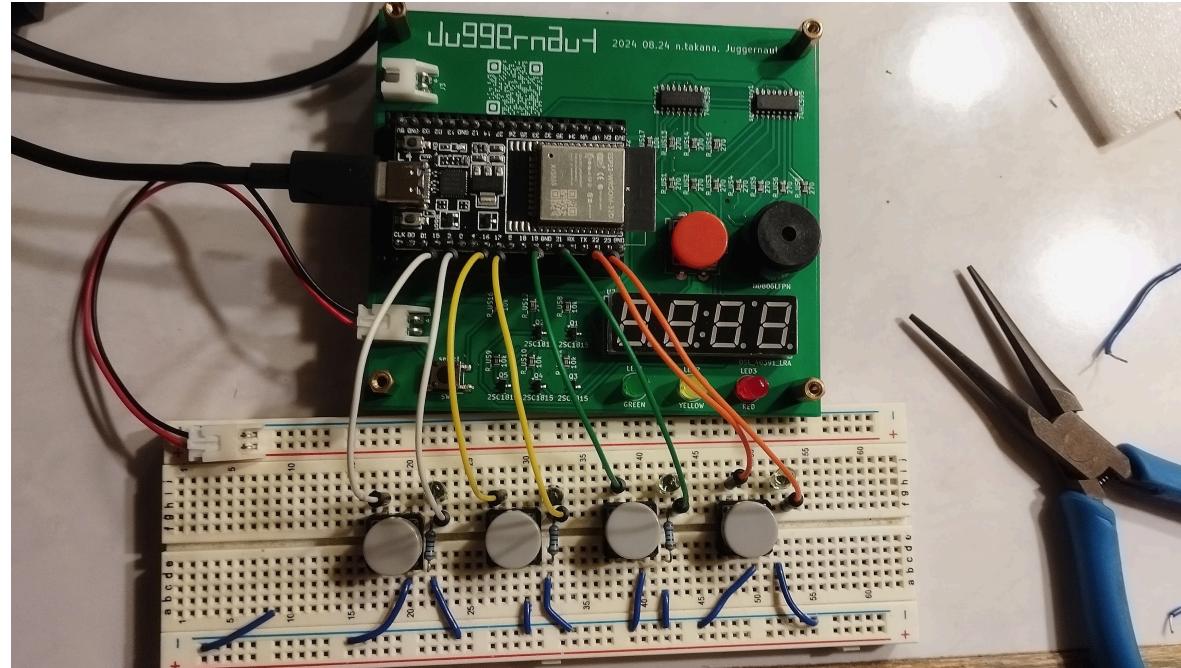


図 2: 競技に使う装置

# 事前知識

## 競技の流れ

1. 回路が接続された装置とプログラムが書かれた問題用紙が渡される
2. ボタンを押してスタート！
  - カウントダウンが始まります
  - 信号が黄色になります
3. プログラムに書かれた条件を満たすと→解除成功！
  - カウントダウンが始まります
  - 信号が緑に変わります
4. 時間切れ or 失敗条件を満たすと→解除失敗……
  - 解除失敗を知らせるブザーが鳴ります
  - 信号が赤に変わります

# 事前知識

## 競技の流れ

### 成功したら……

成功したときの残り秒数がそのまま加点になる

例: 残り 50 秒で成功 → +50 点

### 失敗したら……

失敗したときの残り秒数がそのまま減点になる

例: 残り 47 秒で失敗 → -47 点

解き方が分かっても他のチームにバレないように解きましょう。

# 事前知識

## 装置の説明

装置には、以下のものが付いています。

- 緑・黄・赤の LED
- 残り時間を示す 7 セグ LED
- ブザー
- スタートボタン（ブザーの隣）
- セレクトボタン
- マイコン

# 事前知識

## マイコンって何？

言うまでもなく世の中はコンピュータで溢れています。

- しかし、コンピュータはスマホのような高性能で多機能なものだけではありません。エアコンや炊飯器にも昔から非常に小さな機能だけを持ったコンピュータが入っています。

- このような小さな機能を持ったコンピュータを **microcomputer**、マイコンと呼びます。



図 3: ESP32

# 事前知識

## マイコンでできること

- ・ ピンに電圧をかける（出力）
- ・ ピンの電圧を測る（入力）
- ・ 手に入れたデータを使って別の処理をする

# 事前知識

## マイコンでできること

- ・ ピンに電圧をかける（出力）
- ・ ピンの電圧を測る（入力）
- ・ 手に入れたデータを使って別の処理をする

### こんなことができる

- ・ 暗くなったりある時刻になつたら電気をつける
- ・ センサを繋げて人が通った回数を数える
- ・ 時間になつたら花に水をあげる

# 事前知識

## マイコンでできること

- ・ ピンに電圧をかける（出力）
- ・ ピンの電圧を測る（入力）
- ・ 手に入れたデータを使って別の処理をする

### こんなことができる

- ・ 暗くなったりある時刻になつたら電気をつける
- ・ センサを繋げて人が通った回数を数える
- ・ 時間になつたら花に水をあげる

※適切に回路を組んでそれを制御するプログラムを書けたら

# 事前知識

## 回路に関する基礎知識

### 回路って何だろう？

【回】 カイ（クワイ）・エ（エ）・まわる・まわす・めぐる・かえる

- 向きを変えて、もとにもどる。まわす。まわる。めぐる。かえす。
- 《名・造》 ひとまわり。度数。

みち【道・路・途】

- 人や車などが通る（ように設けた）所。通り路。道路。往来。
- [道・途] 途中。

電気的な素子（電気部品）を導線で結んだループ状の電気の通り路。

# 事前知識

## Vcc(+) と GND(-)

電池には + と - があります。電流はこの + から出ていって - に入っています。さっきの豆電球もそうでしたね。

- これは + のほうが電気を流そうとする圧力が高いからです。これを**電圧**と呼びます。水道の水と同じで電気は圧力の高い方から低い方へと流れていきます。

### 💡 ヒント

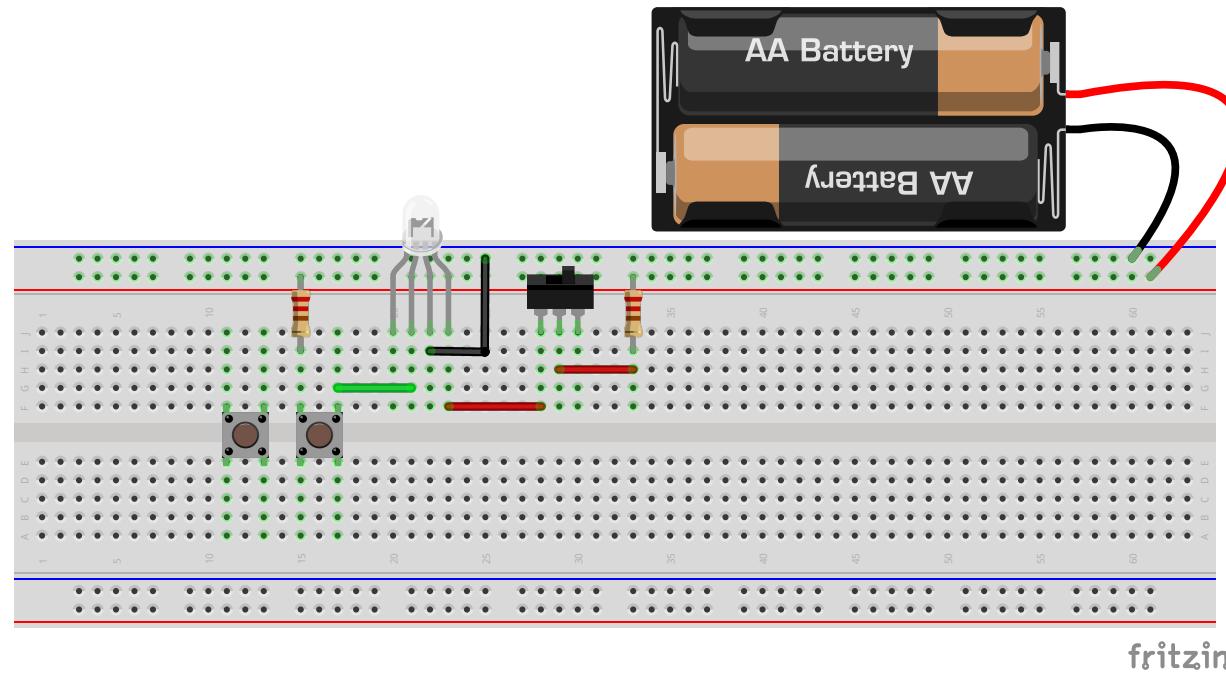
電圧が高いとはどういうことでしょうか？

水道の蛇口を思い出してください。蛇口をたくさんひねると水は勢いよく出てきます。逆に少ししかひねらないとチョロチョロとしか出てこないでしょう。

# 事前知識

## 回路の例

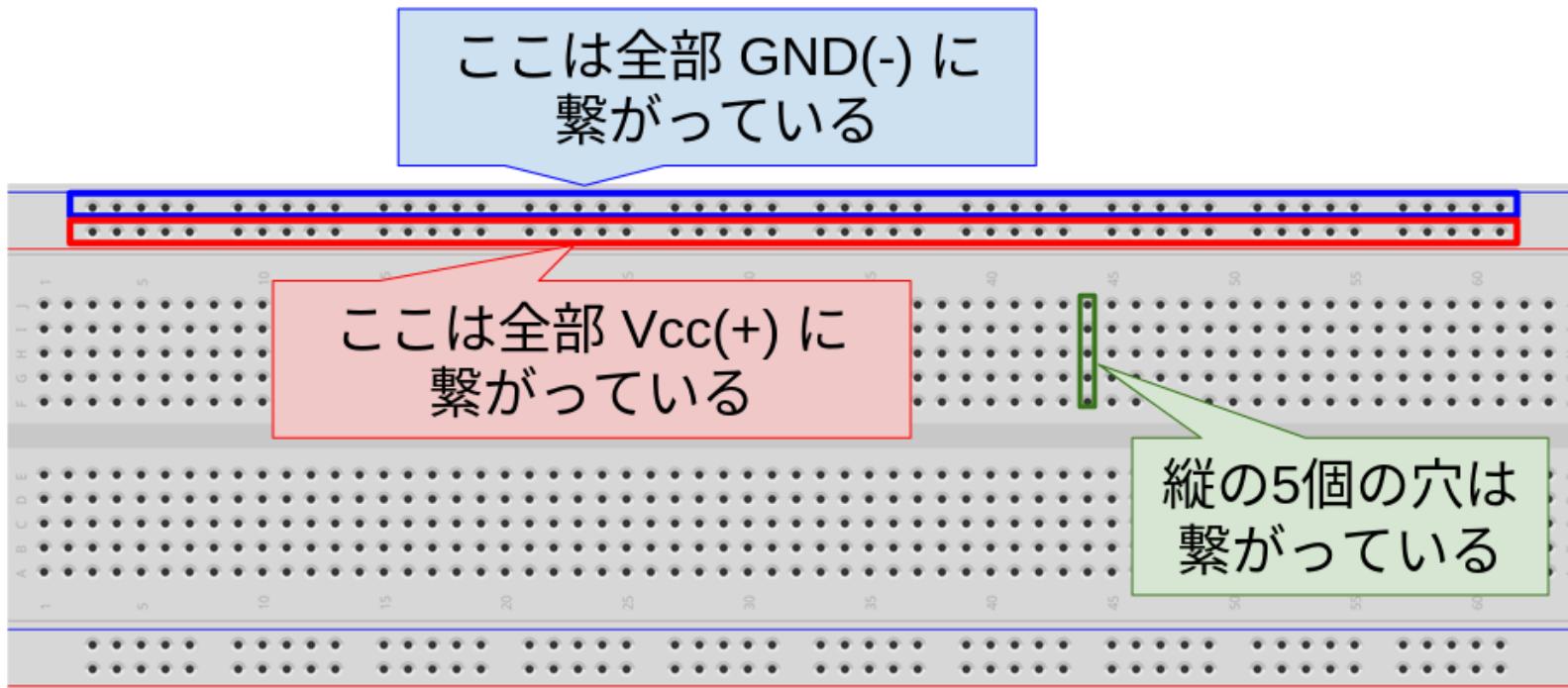
以下のような回路を用意してみました。実際にスイッチを押したり切り替えたりしてみましょう。LEDが点いたり消えたりします。



# 事前知識

## 部品紹介: ブレッドボード

部品やワイヤが刺さっているところをブレッドボードと呼びます。



# 事前知識

## 部品紹介: フルカラー LED

図 6 のように、色の三原色である赤・緑・青のプラス端子と共にマイナス端子からなってます。

配った回路だとマイナスは既に GND に繋がれているので、赤・緑・青のそれぞれに電圧をかけければ電気が流れて光りそうです。

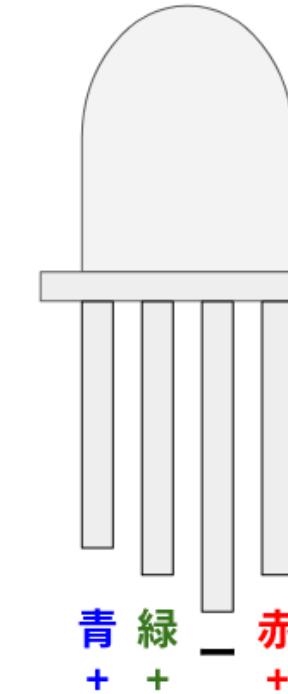


図 6: フルカラー LED

# 事前知識

## 部品紹介: トグルスイッチ

右側の銀色の棒が飛び出たスイッチがトグルスイッチです。

図 7 のように右に倒れているときは左 2 つが、左に倒れているときは右 2 つのピンが繋がるようになります。

これを使って 1 番と 2 番に繋げたいものを接続すると、右に倒れているときは ON、左に倒れているときは OFF となります。

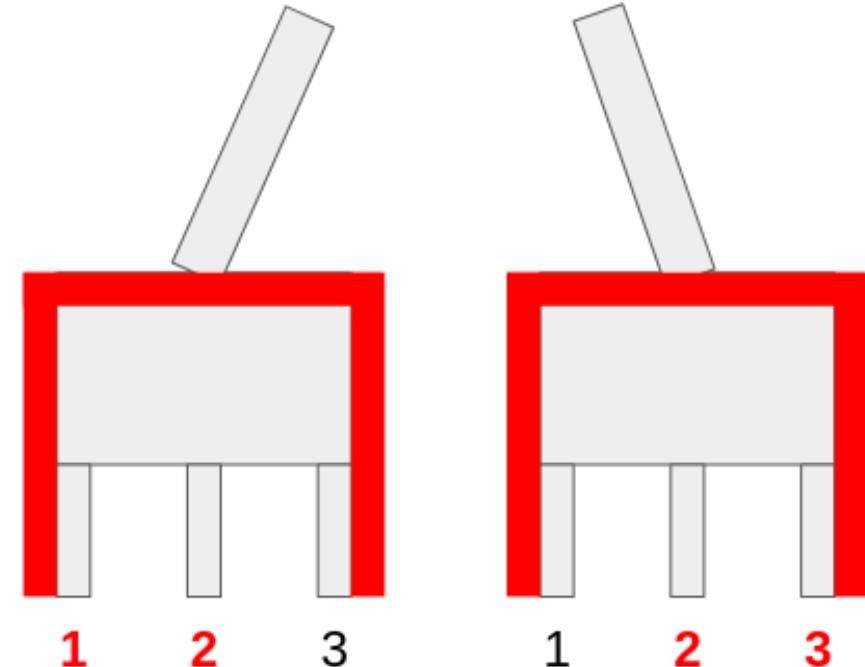


図 7: トグルスイッチの動き

# 事前知識

## 部品紹介: タクトスイッチ

左側にある押しボタンのようなものがタクトスイッチ<sup>1</sup>です。

図 8 のように押していないときは足の縦の部分だけが繋がっています。押しているときは横の足も、つまり全てのピンが繋がります。

これを使って横のピン同士に繋げたいものを接続すると、押している間だけ ON になります。

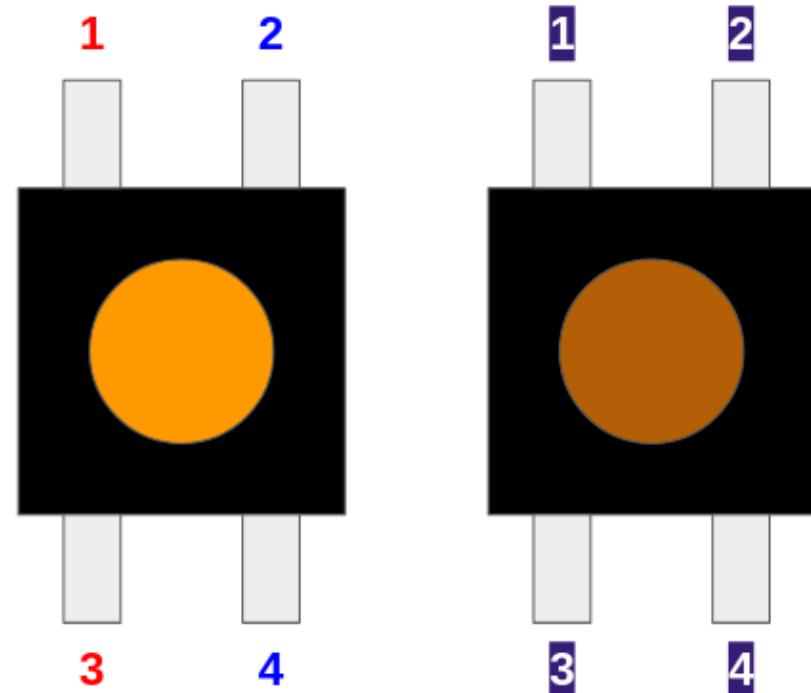


図 8: プッシュスイッチの動き

<sup>1</sup> またはプッシュスイッチ

# 事前知識

## 回路のまとめ

### まとめ

- ・ 回路はループの形になっている電気の通り路.
- ・ 電気は電圧の高いところから低いところに流れる.
- ・ ブレッドボードは縦 5 つの穴と上の横一直線が繋がっている.

# 事前知識

## プログラムに関する基礎知識

回路が作れるようになると色々なことができます。さらに、ここにマイコンを使った制御を加えると、複雑な条件を付けたりデータを保持したり時間を使った処理をしたりできます。

マイコンがどんな制御をすれば良いかを記述するのが**プログラム**です。

例えば **LED** が点灯しているときにボタンを押したら消灯、点灯しているときに押したら点灯するプログラムを考えてみましょう。以下の処理が必要です。

- LED を点灯・消灯させる
- ボタンが押されたかを確かめる
- 今 LED がどっちの状態なのか記憶しておく

# 事前知識

## ピンの出力

LED を付けたり消したりするには電気を流す・流さないをマイコンでコントロールできれば良さそうです。

- ところで電気は**電圧の高いところから低いところに流れ**るのでした。つまり特定の場所のピンの電圧を高い方にできれば良さそうです。
- このような機能は `digitalWrite(pin, HIGH/LOW)` という関数（機能）で実現できます。`pin` はマイコンのピン番号を指定します。番号は足の付け根に書いてあります。

# 事前知識

## ピンの出力

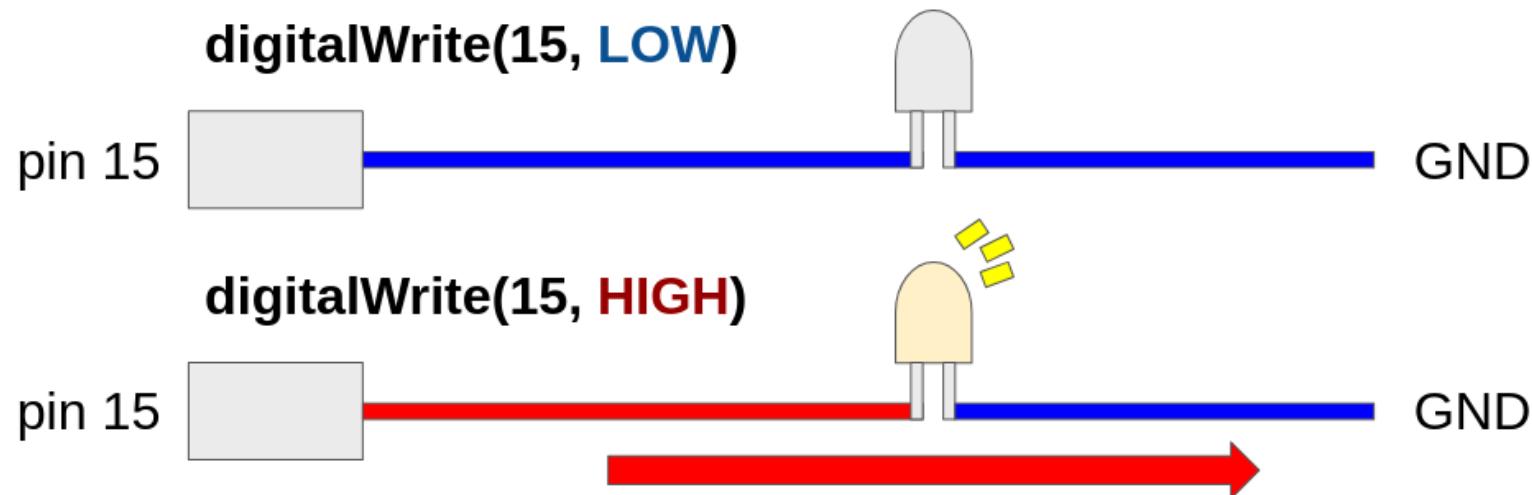


図 9: digitalWrite を使った LED の点灯<sup>1</sup>

HIGH, LOW は電圧の高い低いです。 HIGH を指定すれば LED がつき LOW を指定すれば消えそうです。

<sup>1</sup> 実際は LED にかかるて良い電流が決まっているので抵抗が必要

# 事前知識

## 電圧が高いか低いか

電圧には高い低いがあり、これがいわゆる  $V$  (ボルト) で表されるのですが、今回は **高いか低いか**にのみ着目します。

- プログラムでは電圧が高い時を `HIGH`、低い時を `LOW` と表します。  $V_{cc}$  に繋がっていれば `HIGH`、 $GND$  に繋がっていたら `LOW` だと思いましょう。
- 
- 
- 
- 
-

# 事前知識

## ピンの入力

- 今度はボタンの入力です。電圧の高い低いを検知する機能として、`digitalRead(pin)` という関数があります。指定したピン番号の電圧が `HIGH` か `LOW` かを判定します。
- もし、ピンが `Vcc` と繋がってたら `HIGH`、`GND` と繋がってたら `LOW` になります。

# 事前知識

## ピンの入力

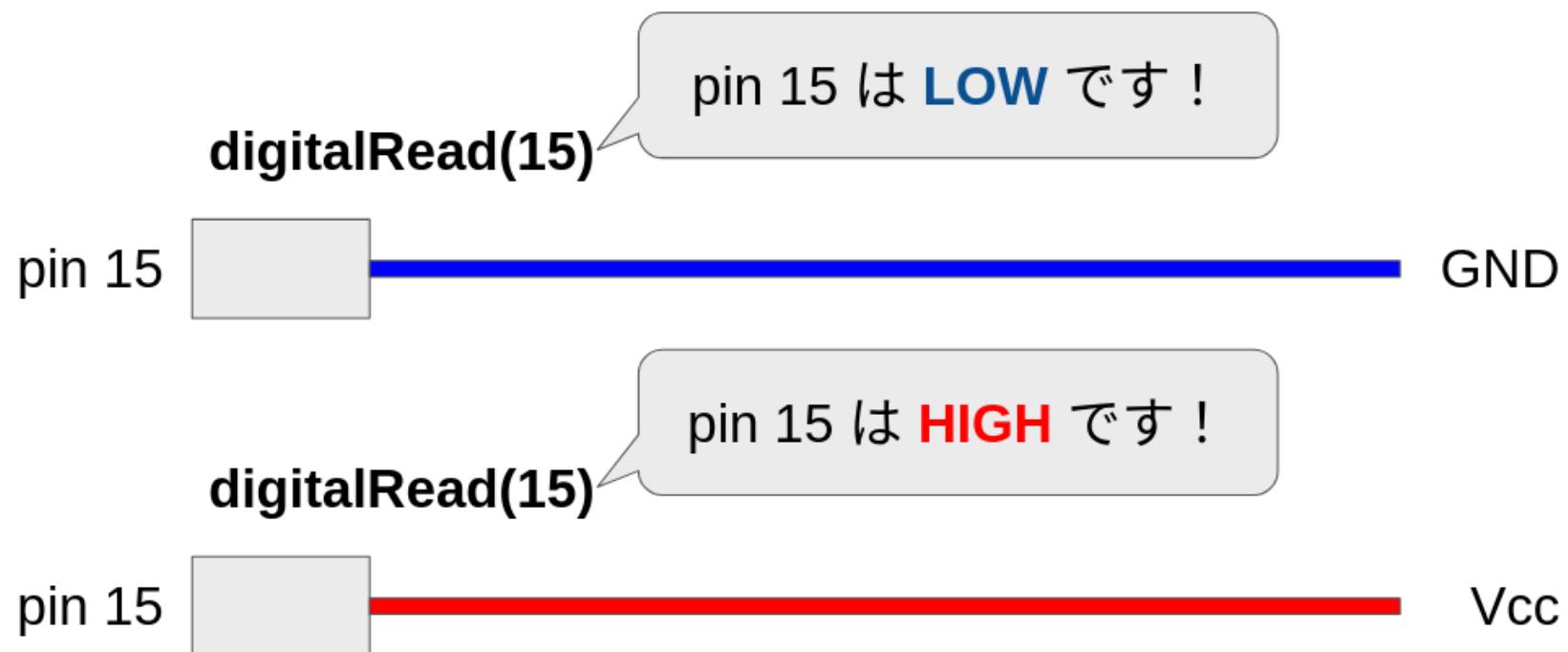


図 10: `digitalRead` を使った電圧の読み込み

# 事前知識

## ピンの入力

そして、ボタンの片方を Vcc, 片方をマイコンのピンに繋げれば、ボタンを押したときだけ HIGH になるためボタンを押しているかが電圧から分かることになります。

### ! 重要

このように電圧の高い・低いは電気を流して何かをするだけではなく、ものの状態を伝える（今回ならボタンを押す）ときにも使われます。

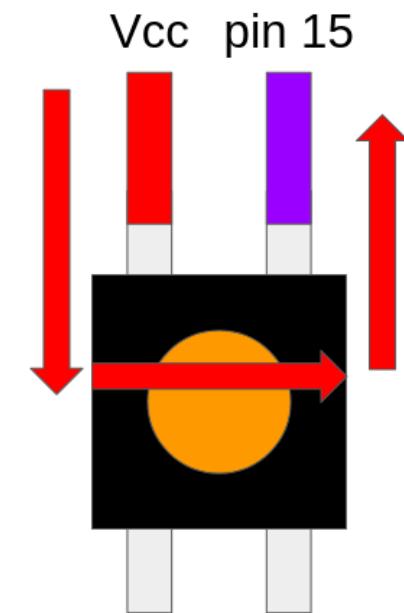


図 11: ボタンを押しているときの電気の流れ

# 事前知識

## プログラムの基本中の基本

```
1 ++
2 void red_or_blue(void *pvParameters) {
3     // 最初はどちらも false
4     bool flag1 = false;
5     bool flag2 = false;
6
7     while (1) {
8         // もし、RED_WIRE 番のピンが LOW だったら { } の中へ
9         if (digitalRead(RED_WIRE) == LOW) {
10             flag1 = true;
11             // そうじゃなかったら↓の { } の中へ
12         } else {
13             flag1 = false;
14         }
15
16         // もし、BLUE_WIRE 番のピンが LOW だったら { } の中へ
17         if (digitalRead(BLUE_WIRE) == LOW) {
```

# 事前知識

## プログラムの基本中の基本

```
18     flag2 = true;
19     // そうじゃなかったら↓の { } の中へ
20 } else {
21     flag2 = false;
22 }
23
24 // もし、flag1 が true なら { } の中へ
25 if (flag1) {
26     succeeded(); // 解除成功！
27 }
28
29 // もし、flag2 が true なら { } の中へ
30 if (flag2) {
31     failed(); // 解除失敗.....
32 }
33 }
34 }
```

# 事前知識

## プログラムのまとめ

### まとめ

- ・ プログラムは上から下に順に実行される
- ・ `digitalRead` である点の電圧を読み取ることができる
- ・ `digitalWrite` である点に電圧をかけることができる
- ・ `while(true)` は `{}` の中をぐるぐる実行し続ける
- ・ `if(条件)` は条件に応じて実行内容を変えられる
- ・ `flag = true` のように名前を付けてデータを保存できる  
これを**変数**と呼ぶ

### 3. 問題 1: 赤か青か

---

# 問題 1: 赤か青か

## 問題文

次の URL に置いてあります。

[https://github.com/Alignof/Juggernaut\\_icc2025\\_workshop](https://github.com/Alignof/Juggernaut_icc2025_workshop)

# 問題 1: 赤か青か

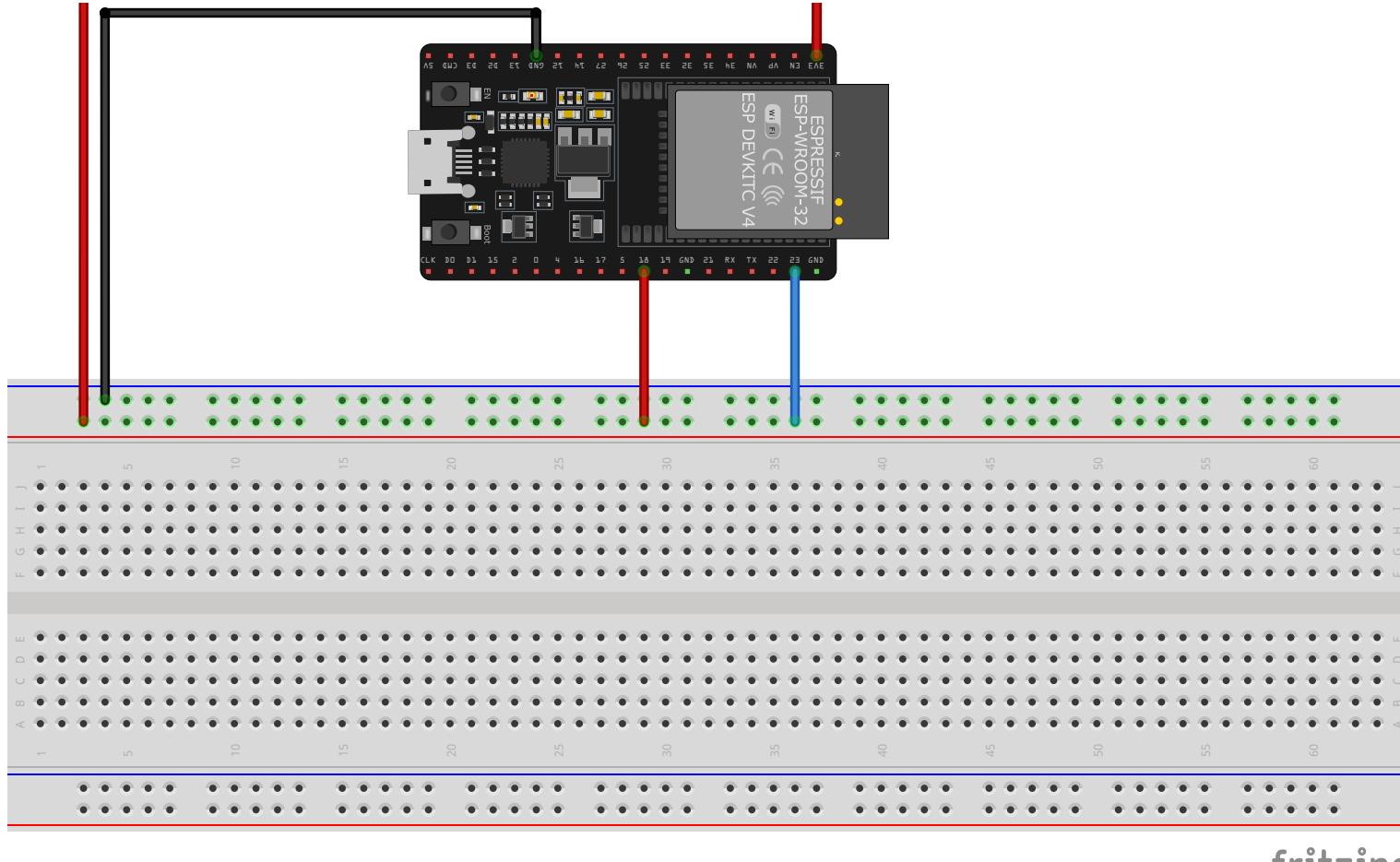
## 概要

赤い線と青い線が出ています。どちらを切ればよいでしょうか？



# 問題 1: 赤か青か

## 配線図



# 問題 1: 赤か青か

## プログラム

```
1 ++
2 struct Challenge RedOrBlue = {
3     .gaming = red_or_blue,
4     .setup_pin = setup_rob,
5     .time_limit = 300, // 300 秒 = 5 分
6 };
7
8 // giver pin assgin
9 const uint8_t RED_WIRE = 23; // 赤いワイヤが刺さっている 23 番
10 const uint8_t BLUE_WIRE = 18; // 青いワイヤが刺さっている 18 番
11
12 void setup_rob(void) {
13     pinMode(RED_WIRE, INPUT_PULLDOWN); // Vcc(+) に繋がってたら HIGH それ以外は LOW
14     pinMode(BLUE_WIRE, INPUT_PULLDOWN);
15 }
16
17 void red_or_blue(void *pvParameters) {
```

# 問題 1: 赤か青か

## プログラム

```
18 // 最初はどちらも false
19 bool flag1 = false;
20 bool flag2 = false;
21
22 while (1) {
23     // もし、RED_WIRE 番のピンが LOW だったら { } の中へ
24     if (digitalRead(RED_WIRE) == LOW) {
25         flag1 = true;
26         // そうじゃなかったら↓の { } の中へ
27     } else {
28         flag1 = false;
29     }
30
31     // もし、BLUE_WIRE 番のピンが LOW だったら { } の中へ
32     if (digitalRead(BLUE_WIRE) == LOW) {
33         flag2 = true;
34         // そうじゃなかったら↓の { } の中へ
```

# 問題 1: 赤か青か

## プログラム

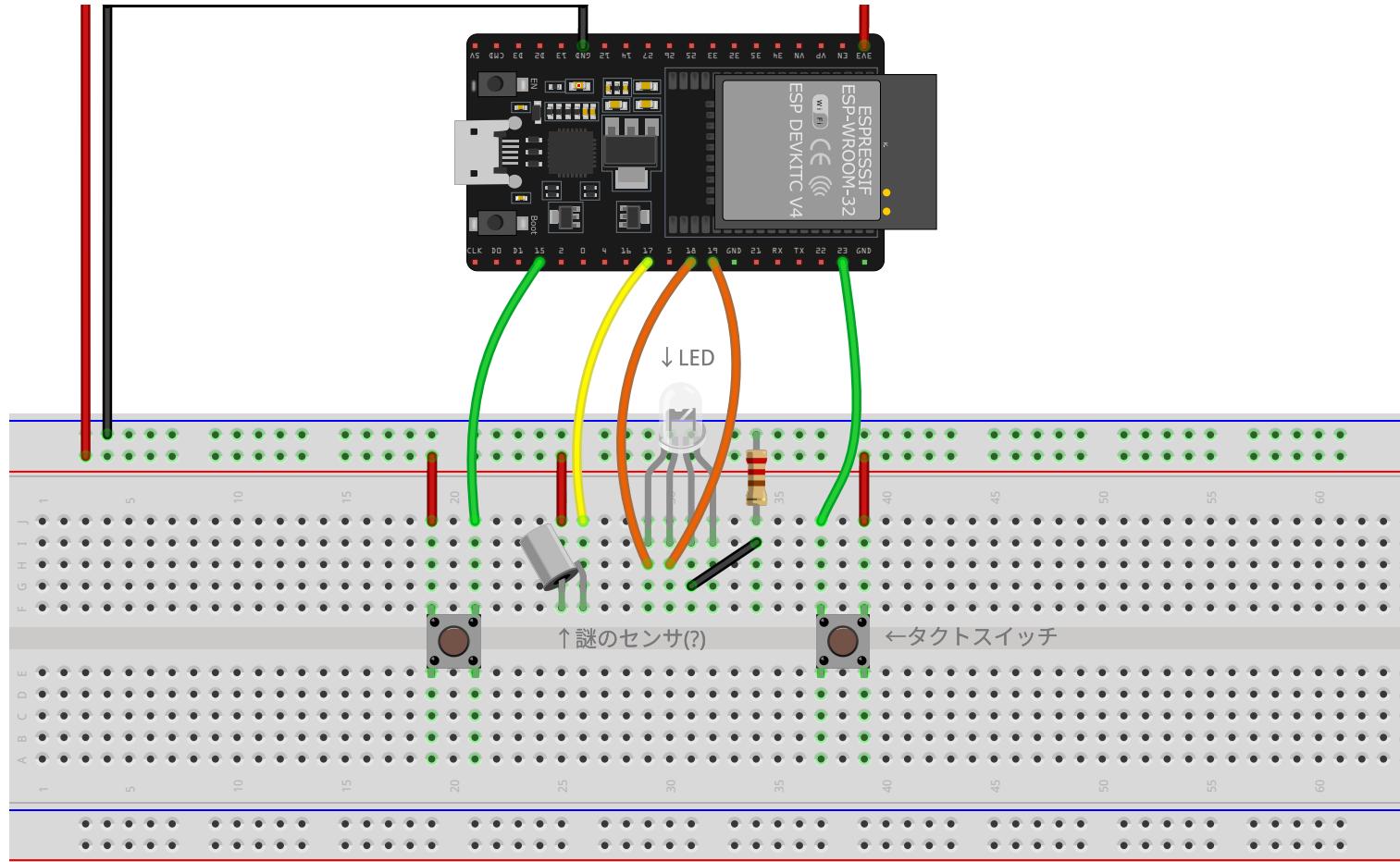
```
35 } else {
36     flag2 = false;
37 }
38
39 // もし、flag1 が true なら {} の中へ
40 if (flag1) {
41     succeeded(); // 解除成功！
42 }
43
44 // もし、flag2 が true なら {} の中へ
45 if (flag2) {
46     failed(); // 解除失敗.....
47 }
48 }
49 }
```

## 4. 問題 2: unknown sensor

---

# 問題 2: unknown sensor

## 配線図



# 問題 2: unknown sensor

## プログラム

```
1 ++
2 struct Challenge UnknownSensor = {
3     .gaming = unknown_sensor,
4     .setup_pin = setup_unknown,
5     .time_limit = 900,
6 };
7
8 // giver pin assgin
9 const uint8_t SENSOR = 17;
10 const uint8_t LED_B = 18;
11 const uint8_t LED_G = 19;
12 const uint8_t BUTTON_LEFT = 15;
13 const uint8_t BUTTON_RIGHT = 23;
14
15 void setup_unknown(void) {
16     pinMode(LED_B, OUTPUT); // LED を点灯させるためのピン
17     pinMode(LED_G, OUTPUT); // LED を点灯させるためのピン
```

# 問題 2: unknown sensor

## プログラム

```
18     pinMode(SENSOR, INPUT_PULLDOWN); // センサの値を読み取るピン
19     pinMode(BUTTON_LEFT, INPUT_PULLDOWN); // ボタンの入力を読み取るピン
20     pinMode(BUTTON_RIGHT, INPUT_PULLDOWN); // ボタンの入力を読み取るピン
21 }
22
23 void unknown_sensor(void *pvParameters) {
24     // 最初は全部 false
25     bool flag1 = false;
26     bool flag2 = false;
27     bool flag3 = false;
28
29     while(1) {
30         // BUTTON_LEFT 番のピンが HIGH なら true そうでなければ false
31         flag1 = (digitalRead(BUTTON_LEFT) == HIGH);
32
33         // BUTTON_RIGHT 番のピンが HIGH なら true そうでなければ false
34         flag2 = (digitalRead(BUTTON_RIGHT) == HIGH);
```

# 問題 2: unknown sensor

## プログラム

```
35      // SENSOR 番のピンが HIGH なら true そうでなければ false
36      flag3 = (digitalRead(SENSOR) == LOW);
37
38      // もし、flag1 の中身が true だったら { } の中へ
39      if (flag1) {
40          digitalWrite(LED_G, HIGH);
41          // そうじゃなかったら↓の { } の中へ
42      } else {
43          digitalWrite(LED_G, LOW);
44
45      }
46
47      // もし、flag3 の中身が true だったら { } の中へ
48      if (flag3) {
49          digitalWrite(LED_B, HIGH);
50          // そうじゃなかったら↓の { } の中へ
51      }
```

# 問題 2: unknown sensor

## プログラム

```
52 } else {
53     digitalWrite(LED_B, LOW);
54 }
55 }
56
57 // もし、flag3 と flag2 の*両方*が true なら
58 if(flag3 && flag2) {
59     succeeded();
60 }
61
62 // もし、flag3 と flag1 の*両方*が true なら
63 if(flag3 && flag1) {
64     failed();
65 }
66
67 delay(100);
68 }
```

# 問題 2: unknown sensor

## プログラム

```
69 }  
70
```

# 問題 2: unknown sensor

解説

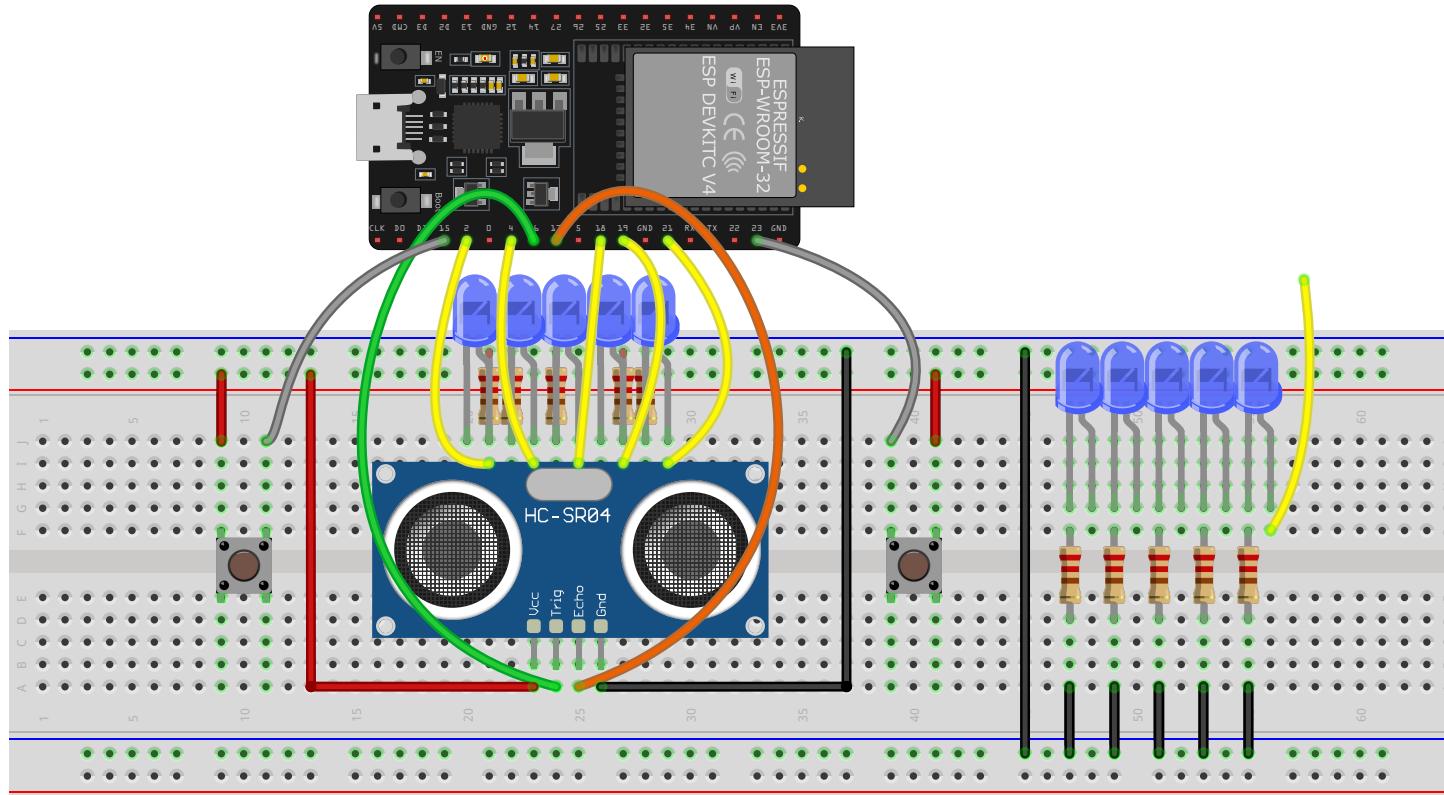


## 5. 問題 3: ULTRASONIC

---

# 問題 3: ULTRASONIC

## 配線図



↑ LEDの部分は分かりづらいですがこうなってます。

fritzing

# 問題 3: ULTRASONIC

## プログラム

```
1 ++
2 struct Challenge Ultrasonic = {
3     .gaming = ultrasonic,
4     .setup_pin = setup_dist,
5     .time_limit = 900,
6 };
7
8 // giver pin assgin
9 const uint8_t TRIG = 16;
10 const uint8_t ECHO = 17;
11 const uint8_t LED1 = 2;
12 const uint8_t LED2 = 4;
13 const uint8_t LED3 = 18;
14 const uint8_t LED4 = 19;
15 const uint8_t LED5 = 21;
16 const uint8_t LEFT_BUTTON = 15;
17 const uint8_t RIGHT_BUTTON = 23;
```

# 問題 3: ULTRASONIC

## プログラム

```
18
19 void setup_dist(void) {
20     pinMode(TRIG, OUTPUT); // 出力のためのピン
21     pinMode(LED1, OUTPUT); // 出力のためのピン
22     pinMode(LED2, OUTPUT); // 出力のためのピン
23     pinMode(LED3, OUTPUT); // 出力のためのピン
24     pinMode(LED4, OUTPUT); // 出力のためのピン
25     pinMode(LED5, OUTPUT); // 出力のためのピン
26     pinMode(ECHO, INPUT); // 入力のためのピン
27     pinMode(LEFT_BUTTON, INPUT_PULLDOWN); // ボタンの入力を読むためのピン
28     pinMode(RIGHT_BUTTON, INPUT_PULLDOWN); // ボタンの入力を読むためのピン
29 }
30
31 void ultrasonic(void *pvParameters) {
32     bool flag1 = false;
33     bool flag2 = false;
34     bool flag3 = false;
```

# 問題 3: ULTRASONIC

## プログラム

```
35     float duration, cm;  
36  
37     while(1) {  
38         flag1 = (digitalRead(RIGHT_BUTTON) == HIGH);  
39         flag2 = (digitalRead(LEFT_BUTTON) == HIGH);  
40  
41         digitalWrite(TRIG, LOW);  
42         delayMicroseconds(2);  
43         digitalWrite(TRIG, HIGH);  
44         delayMicroseconds(10);  
45         digitalWrite(TRIG, LOW);  
46  
47         // ECHO が HIGH になっている時間を返す（マイクロ秒）。  
48         duration = pulseIn(ECHO, HIGH);  
49         // duration を 2 で割って係数 0.0344 をかける。  
50         // 係数 0.0344 は  $331.5 * (0.61 * t)$  ( $t$  は摂氏温度, 今回は20度と想定) の結果の単位を合わせたもの。
```

# 問題 3: ULTRASONIC

## プログラム

```
51 // これは一般的に約 340 m/s と習う「あの」速度を温度に合わせて厳密に計算したものです.  
52 cm = (duration / 2) * 0.0344;  
53  
54 // 先に LED1~LED5 までのピンを全部 LOW にしておく  
55 digitalWrite(LED1, LOW);  
56 digitalWrite(LED2, LOW);  
57 digitalWrite(LED3, LOW);  
58 digitalWrite(LED4, LOW);  
59 digitalWrite(LED5, LOW);  
60 if (10 <= cm) { //もし cm の値が 10 以下だったら  
61 | digitalWrite(LED1, HIGH); // LED1 番のピンを HIGH に  
62 }  
63 if (15 <= cm) { //もし cm の値が 15 以下だったら  
64 | digitalWrite(LED2, HIGH);  
65 }  
66 if (20 <= cm) { //もし cm の値が 20 以下だったら  
67 | digitalWrite(LED3, HIGH);
```

# 問題 3: ULTRASONIC

## プログラム

```
68      }
69      if (25 <= cm) { //もし cm の値が 25 以下だったら
70          digitalWrite(LED4, HIGH);
71      }
72      if (30 <= cm) { //もし cm の値が 30 以下だったら
73          digitalWrite(LED5, HIGH);
74      }
75
76      if (flag1 && 15 < cm && cm <= 20) {
77          flag3 = true;
78      }
79
80      // succeeded
81      if(flag3) {
82          succeeded();
83      }
84  }
```

# 問題 3: ULTRASONIC

## プログラム

```
85     // failed
86     if(flag1 && flag2) {
87         failed();
88     }
89
90     delay(100);
91 } // while (true) { の行まで戻る
92 }
```

## 6. 付録

---

# 付録

## 参考文献

