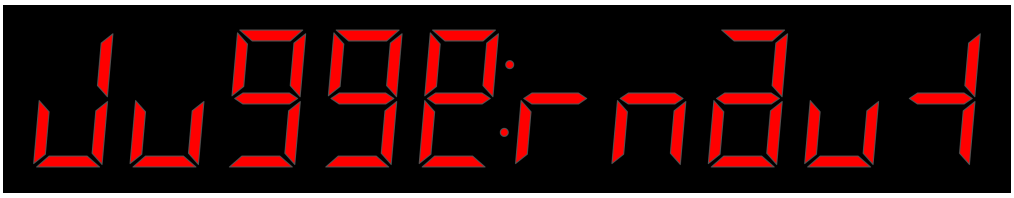

装置名	Red or Blue
装置番号	icc2025-001
作問者	Norimasa TAKANA
作問日	2025-10-20
制限時間	15m00s

1. 作問者より一言

やっぱり最初は定番のこれ！ 落ち着いてよくプログラムを読めばどっちを切るべきか分かるはずです．

2. 回路

回路の配線図を図 2 に示す．

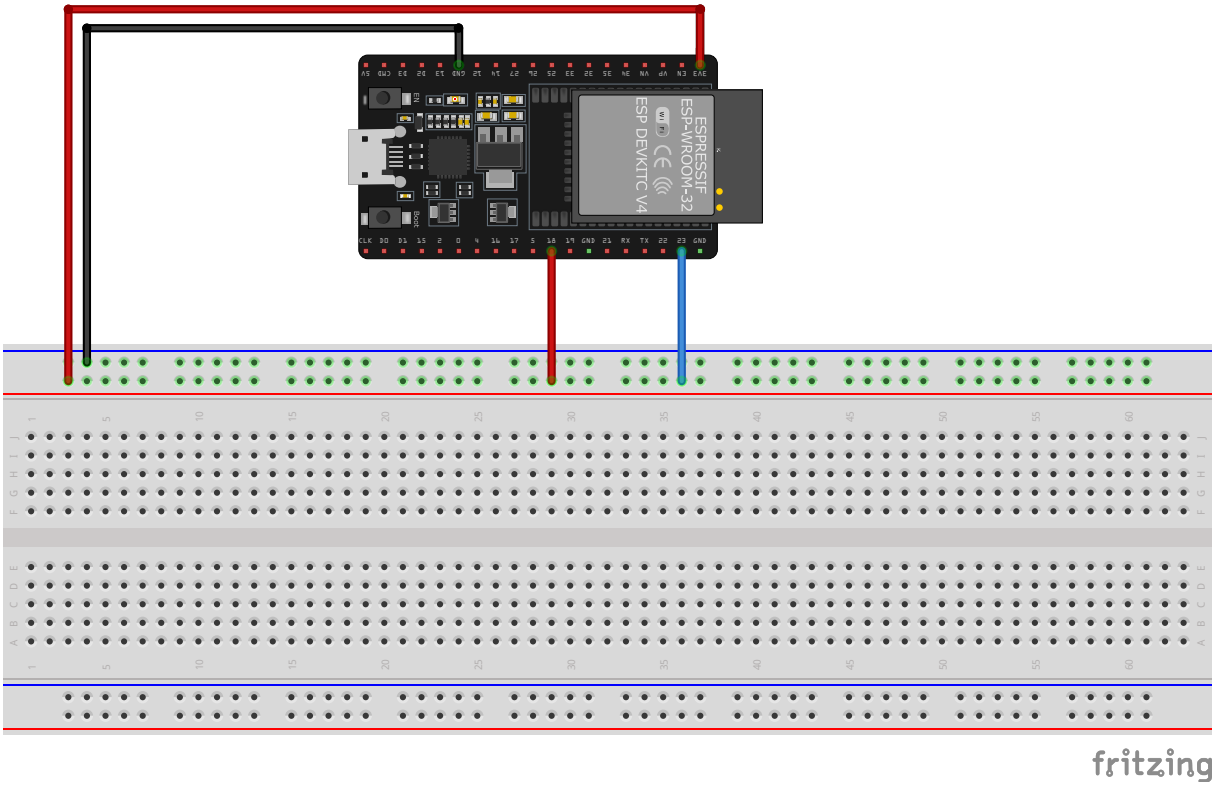

図 2: 配線図

3. ソースコード

装置に書き込まれたプログラムを以下に示す．

```cpp
struct Challenge RedOrBlue = {
    .gaming = red_or_blue,
    .setup_pin = setup_rob,
    .time_limit = 300, // 300 秒 = 5 分
};

// pin assgin
const uint8_t RED_WIRE = 23; // 赤いワイヤが刺さっている 23 番
const uint8_t BLUE_WIRE = 18;// 青いワイヤが刺さっている 18 番

void setup_rob(void) {
    // INPUT_PULLDOWN: Vcc(+) に繋がってたら HIGH それ以外は LOW
    pinMode(RED_WIRE, INPUT_PULLDOWN);
    pinMode(BLUE_WIRE, INPUT_PULLDOWN);
}

void red_or_blue(void *pvParameters) {
    // 最初はどっちも false
    bool flag1 = false;
    bool flag2 = false;

    while (true) {
        // もし，RED_WIRE 番のピンが LOW だったら { } の中へ
        if (digitalRead(RED_WIRE) == LOW) {
            flag1 = true;
        // そうじゃなかったら↓の { } の中へ
        } else {
            flag1 = false;
        }

        // もし，BLUE_WIRE 番のピンが LOW だったら { } の中へ
        if (digitalRead(BLUE_WIRE) == LOW) {
            flag2 = true;
        // そうじゃなかったら↓の { } の中へ
        } else {
            flag2 = false;
        }

        // もし，flag1 が true なら { } の中へ
        if (flag1) {
            succeeded(); // 解除成功！
        }

        // もし，flag2 が true なら { } の中へ
        if (flag2) {
            failed(); // 解除失敗……
        }
    } // while (true) { の行まで戻る
```

```
49 }
```

4. 解き方

4.1. 配線を確認する

何にせよまずは配線を確認しましょう． 図 2 を見てみると，どうやら赤い線と青い線が装置から出ているみたいです．

次に書かれているプログラムを見てみると，`RED_WIRE` や `BLUE_WIRE` という文字が見えます． 関係がありそうです． しかし，ここで焦って糠喜びしてはいけません． プログラムではデータを入れる“箱”，変数にプログラマが好きな名前を付けられますが，爆弾魔が赤いワイヤに正直に `RED_WIRE` と付けるとは限りません．

やってみよう

- 手元の装置を見て，赤いワイヤ・青いワイヤがマイコンのピンの何番に繋がっているか調べよう．
  - 赤いワイヤ: _____
  - 青いワイヤ: _____
- プログラムを見て， `RED_WIRE` , `BLUE_WIRE` がそれぞれ何番を表すのか調べよう．
  - `RED_WIRE` : _____
  - `BLUE_WIRE` : _____

4.2. 解除失敗の条件を調べる

成功の条件を調べるために回路をガチャガチャしてたらうっかり `failed()` の方に入って爆発……そうならないために先に失敗する条件を確認しましょう．解除が失敗する（= 爆弾が爆発する）条件は `failed()` を呼び出してしまうことでした．この関数が呼び出されると LED が赤になりブザーが鳴ってタイマが停止します．

4.2.1. 爆発の条件

まず， `failed()` がどこにあるか調べましょう．これはすぐに見つかるはずです．

cpp
```cpp
// もし，flag2 が true なら { } の中へ
if (flag2) {
    failed(); // 解除失敗……
}
```

まだ if という構文に慣れないでしょうが，

cpp
```cpp
if (この中身が true なら) {
    この中身が実行される（true じゃなければ無視される）
}
```

という意味です．つまり， `flag2` とやらが `true` じゃなければ良さそうです．

まとめ

- `flag2` が `true` → **爆発**
- `flag2` が `false` → **爆発しない**

4.2.2. flag2 が true になるには？

`flag2` は変数というものです．変数はデータに名前を付けて記憶しておく“箱”のようなものでした． 以下のように定義されています．

cpp

```cpp
18 // 最初はどっちも false
19 bool flag1 = false;
20 bool flag2 = false;
```

最初は `false` なのでいきなり爆発したりはしなさそうです． では，どこで `true` になるのでしょうか？変数に値を入れるには `=` を使うので，`flag2 = true` というコードを探せば良さそうです． 見つかりましたか？

正解は 33 行目のここです．

cpp

```cpp
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
35 } else {
36     flag2 = false;
37 }
```

条件が書いてありますね．`digitalRead(BLUE_WIRE) == LOW` とあります． `digitalRead` 関数はこういう関数でした．

定義: digitalRead

マイコンのピンの状態を**「読む」**関数．Vcc(+) に繋がってたら `HIGH` それ以外は `LOW` を返す．

ここまでの話をまとめるとこうなります．

まとめ

- `BLUE_WIRE` のピンの入力が `LOW` → **爆発**
- `BLUE_WIRE` のピンの入力が `HIGH` → **爆発しない**

4.2.3. 回路をもう 1 回見る

`BLUE_WIRE` のピンの入力が `LOW` → **爆発** ということは今は `LOW` ではなさそうです．今 `BLUE_WIRE` のピンから出ているワイヤはどこへ繋がっているのでしょう？

やってみよう

ブレッドボードの Vcc(+) と繋がっている部分を全てペンでなぞって整理しよう．

4.3. 解除成功の条件を調べる

失敗の条件と同じ方法で，今度は成功する条件を調べましょう．`succeeded()` が呼び出されれば成功です．

やってみよう

以下の空欄や選択肢を埋めながら，今度は解除が成功するときの条件を考えてみましょう．

- `succeeded()` は `flag1・flag2` が `true・false` のときに呼び出される．
- `flag1・flag2` が `true` になる条件は，＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ == `HIGH・LOW` のときである．
- `INPUT_PULLDOWN` のピンは Vcc(+) に繋がっているとき `HIGH・LOW`，繋がってないとき `HIGH・LOW` となる．

4.4. こっちだと思う線を引っこ抜け！

もうここまで来れば大丈夫でしょう．赤と青，こっちだと思う線を引っこ抜いてみましょう！

結果

- 引っこ抜いた線: 赤・青
- 結果: 成功・失敗
- 残り時間: ＿＿＿＿＿＿＿＿