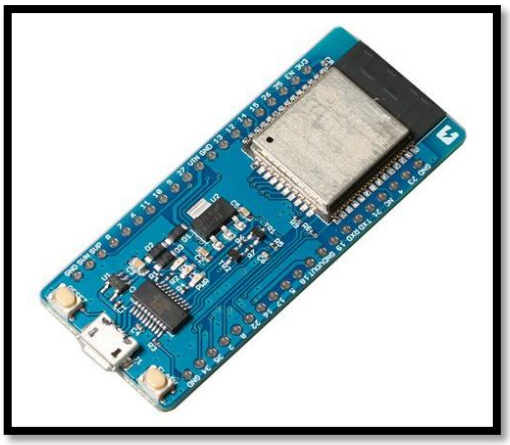
ESP32 とは



上海の Espressif Systems 社によって開発されている
集積回路製品（SoC）のマイクロコントローラ。
Raspberry Pi とは違い起動に OS が必要ない分、消費電力が
少なく、Wi-Fi や Bluetooth も搭載している上に低価格。
世界中で電子工作に使用される Arduino の開発環境を使用
する事ができ、Arduino 言語によってプログラミングし、
GPIO ピンから様々な電子部品を配線して制御できる。

Arduino 言語とは

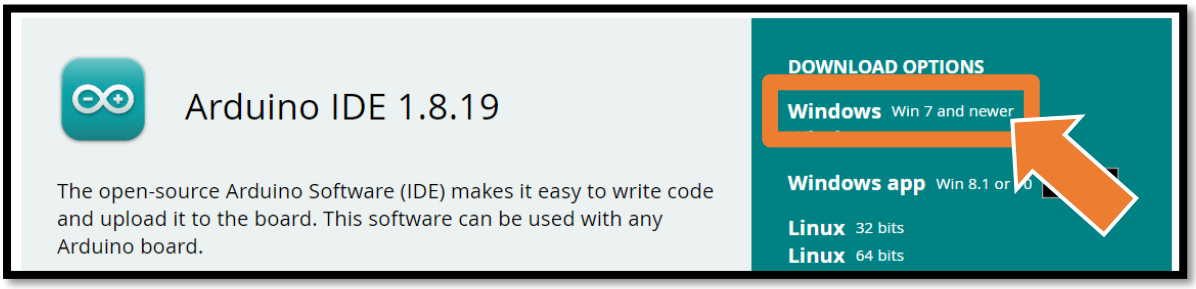
Arduino 言語は C 言語と C++をベースにしたプログラミング言語で、どちらの文法も殆ど使用できる。
C 言語をベースにした Java を既に学んでいる人であれば、違和感なく記述できる。

Arduino IDE インストール

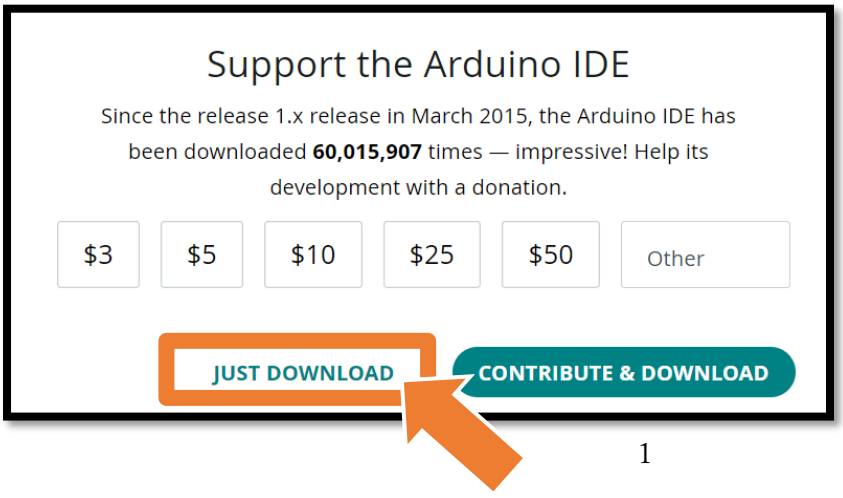
ESP32 の開発環境を構築していく。Windows10 での構築手順なので、実習機で行うことを推奨する。
※既に Arduino IDE が実習機にインストールされている場合は⑤から行うこと。

① ブラウザを起動し、[Software | Arduino]へアクセスする。

② DOWNLOAD OPTIONS の[Windows Win7 and newer]をクリックする。

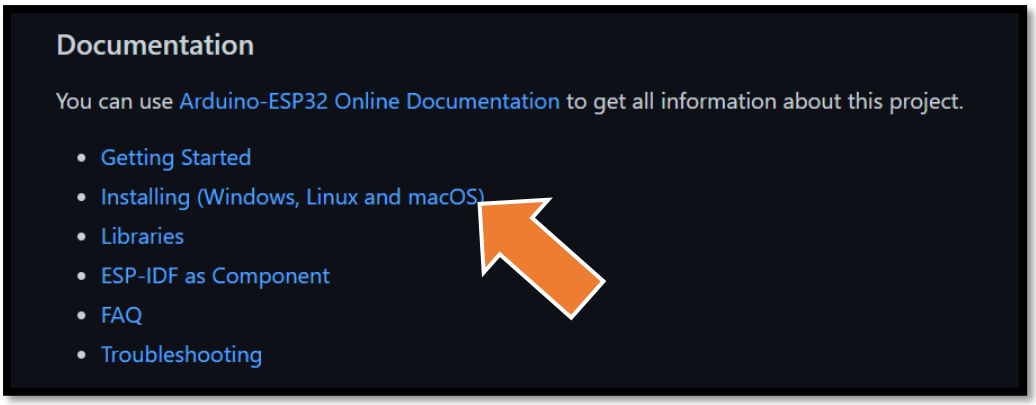


③ [JUST DOWNLOAD]をクリックするとインストーラーのダウンロードが始まる。



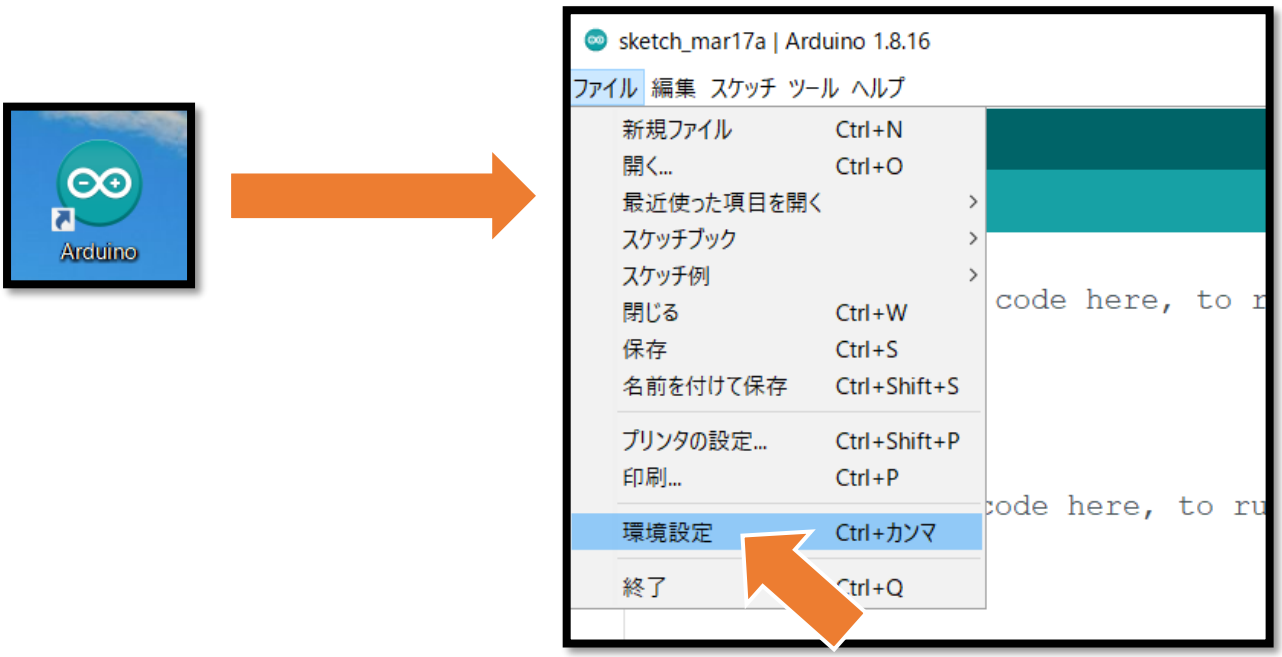
④　ダウンロードした[Arduino-1.~.~~-windows.exe]を実行してインストールする。
※全てデフォルトのまま進めて良い。

⑤　インストール中に[GitHub - espressif/arduino-esp32: Arduino core for the ESP32]にアクセスし、ページ下部の Documentation 欄の[Installing(Windows, Linux and macOS)]をクリックする。

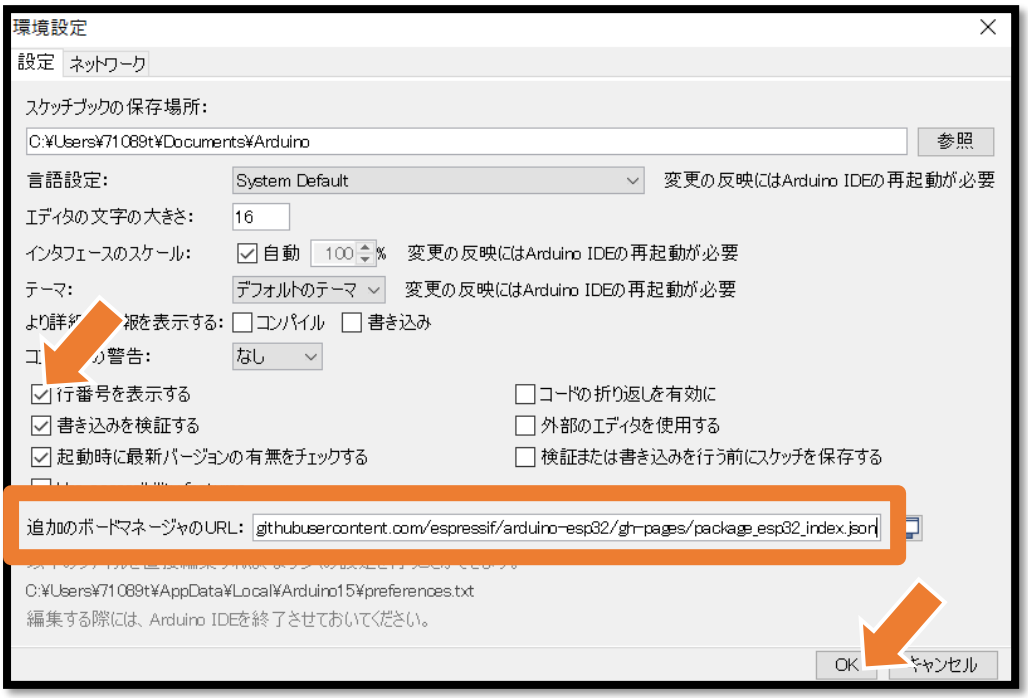


⑥　移動したリンク先の Stable release link:の下にある URL をコピーする。

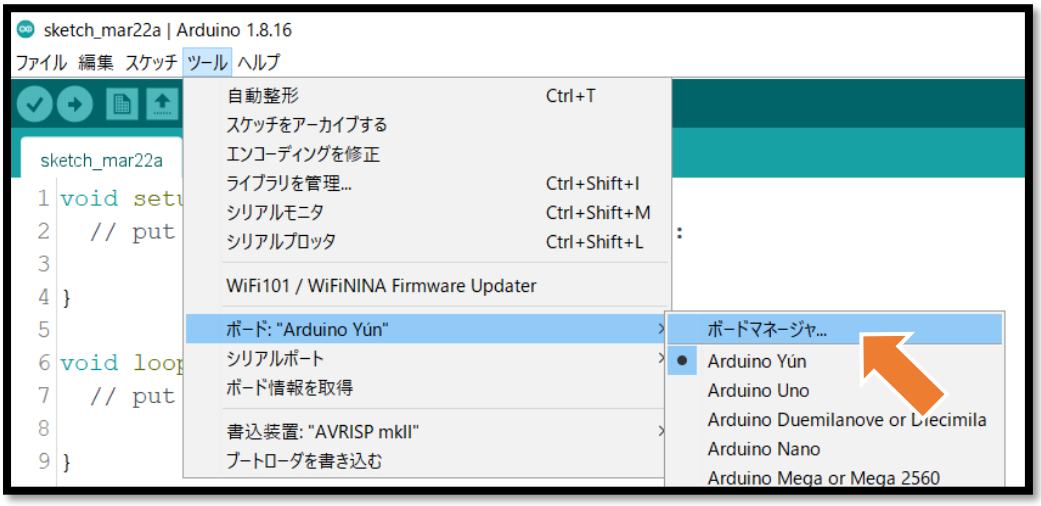
⑦　インストール後、Arduino IDE を起動する。
その後、ファイルタブの[環境設定]をクリックする。



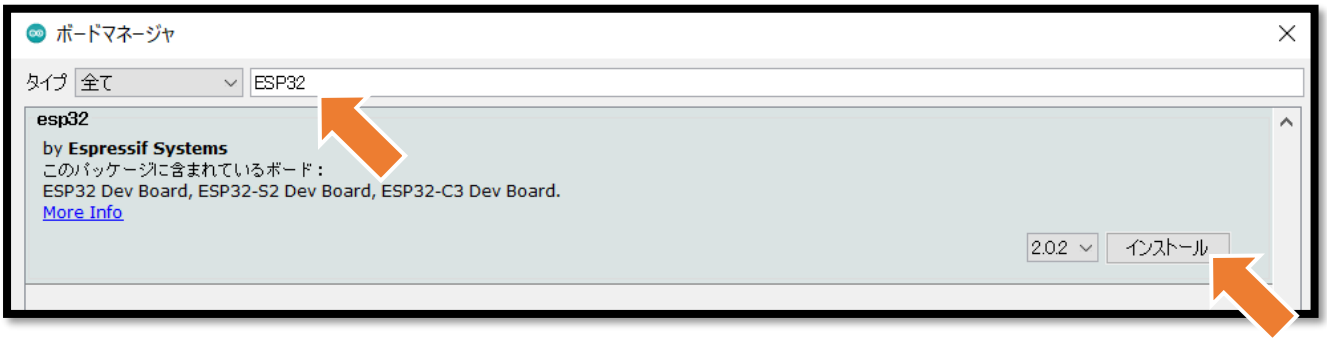
⑧　[行番号を表示する]チェックボックスをクリックし、チェックを入れる。
その後、[追加のボードマネージャの URL]欄に先ほどコピーした URL をペーストする。
その後、[OK]をクリックしてウィンドウを閉じる。



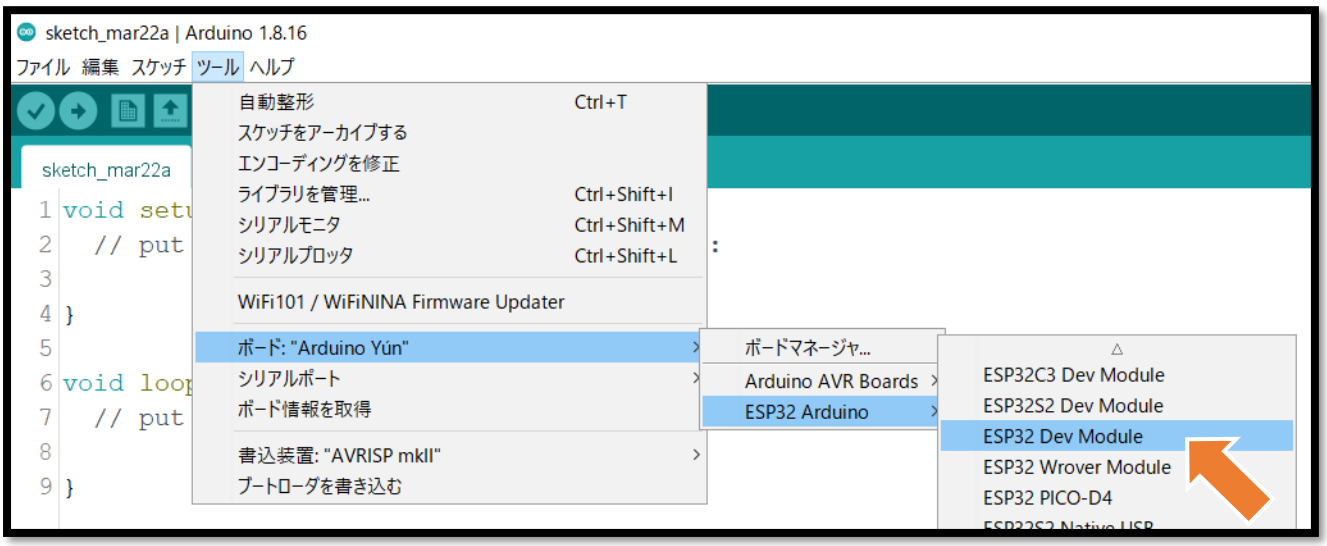
⑨　ツールタブ内のボードメニューにある[ボードマネージャ…]をクリックする。



その後、ボードマネージャの入力欄に[ESP32]と入力し、インストールをクリックする。



⑩　インストール後、ボードメニューに追加された[ESP32 Arduino]パッケージの中から
[ESP32 Dev Module]を選択する。　※似たものが多いので注意

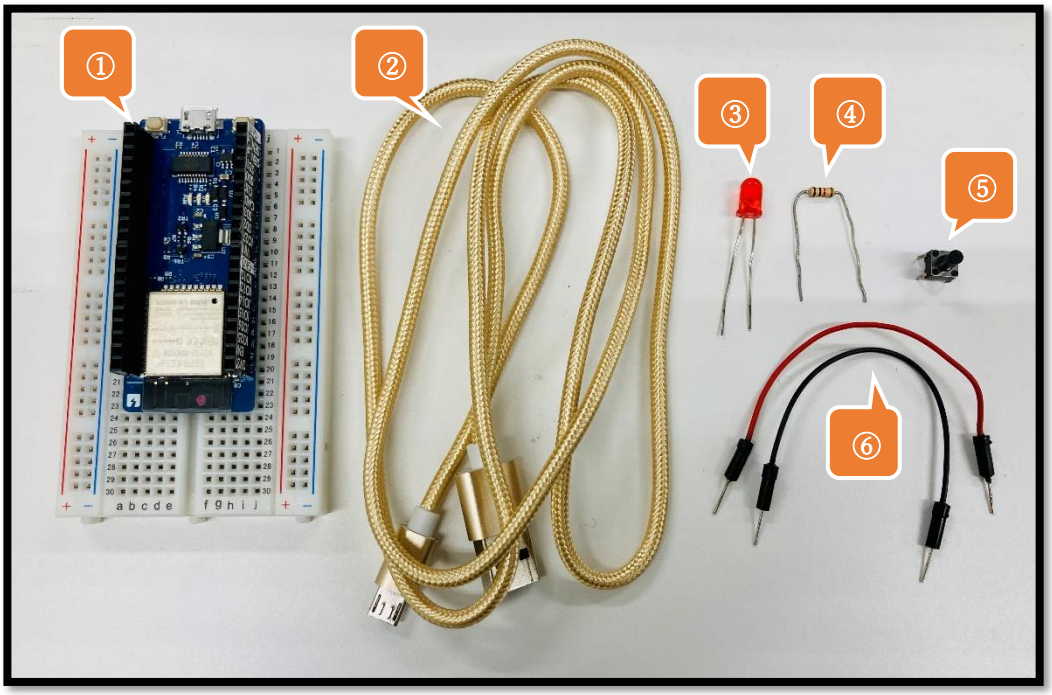


機材準備

今回使用する機材を所定の位置から借りて持って来る。※授業終了時には必ず元の場所に戻すこと。

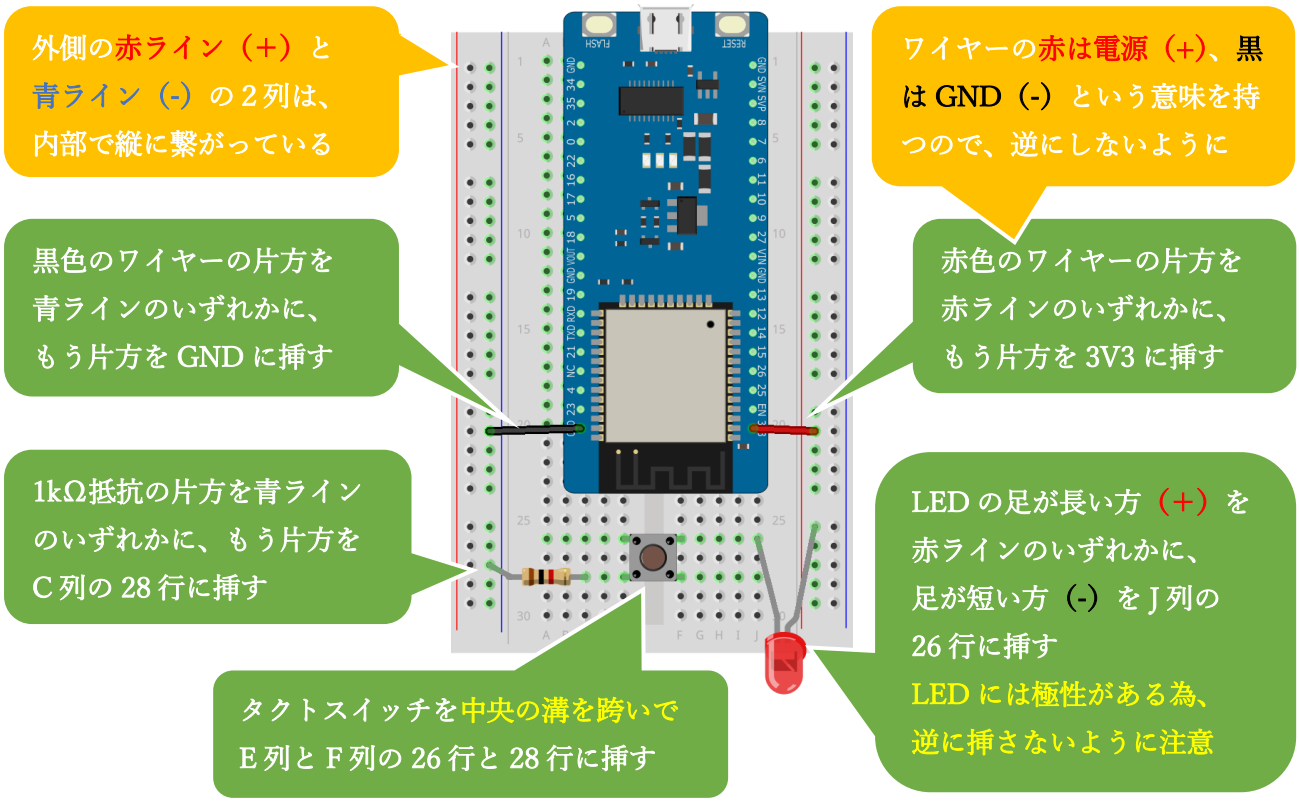
① ESP32（ブレッドボード付）
② Micro-B USB ケーブル
③ 赤色 LED
④ 1kΩ抵抗（茶・黒・赤・金）　※他の抵抗と間違えないように
⑤ タクトスイッチ
⑥ ジャンパーワイヤー（赤と黒を 1 本ずつ）

① と ② は毎回必要。自分の番号の物を取ること



練習 1 （ハードウェア制御）

ESP32 に Micro-B USB ケーブルを差し込まずに以下の図のとおりに配線を行いなさい。



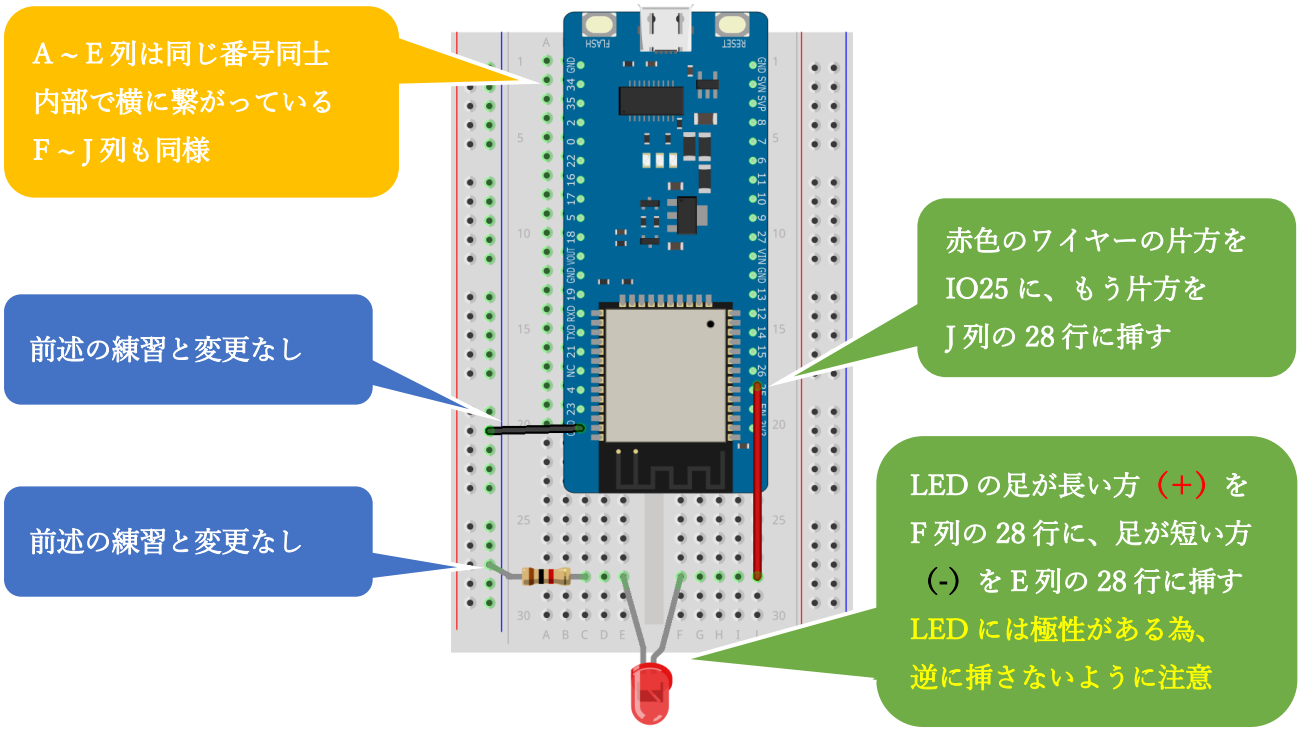
配線を行った後、Micro-B USB ケーブルを ESP32 と実習機に繋ぎ通電する。
その後、タクトスイッチを押すと LED が点灯し、離すと消灯するか確認しなさい。
※点灯しない場合は以下の理由が考えられます。
・単純に配線を間違えている。
・LED かスイッチが壊れている
・ジャンパーワイヤーが断線している。
・ESP32 のコネクタの接触不良
対処法として、LED とスイッチとジャンパーワイヤーを取り換えてみる（壊れていたら講師に渡す）
それでも上手くいかなければ ESP32 を空き番号や隣の人のものと変えてみてください。

この練習では ESP32 から出力されている電源を使用し、物理的に電子回路を組んだだけなので
「スイッチを押すと LED が点灯する」という動作しか行えず、LED を自動で点灯・消灯させられない。
LED を自動で点灯・消灯させるには LED の電源（+）側を GPIO ピンに変えて、プログラムで電圧の
出力を切り替えなければならない。…ということをこれまで Raspberry Pi で演習してきました。

Raspberry Pi の時と同様に LED を制御する為に、ESP32 へプログラムの書き込みを行いましょう。

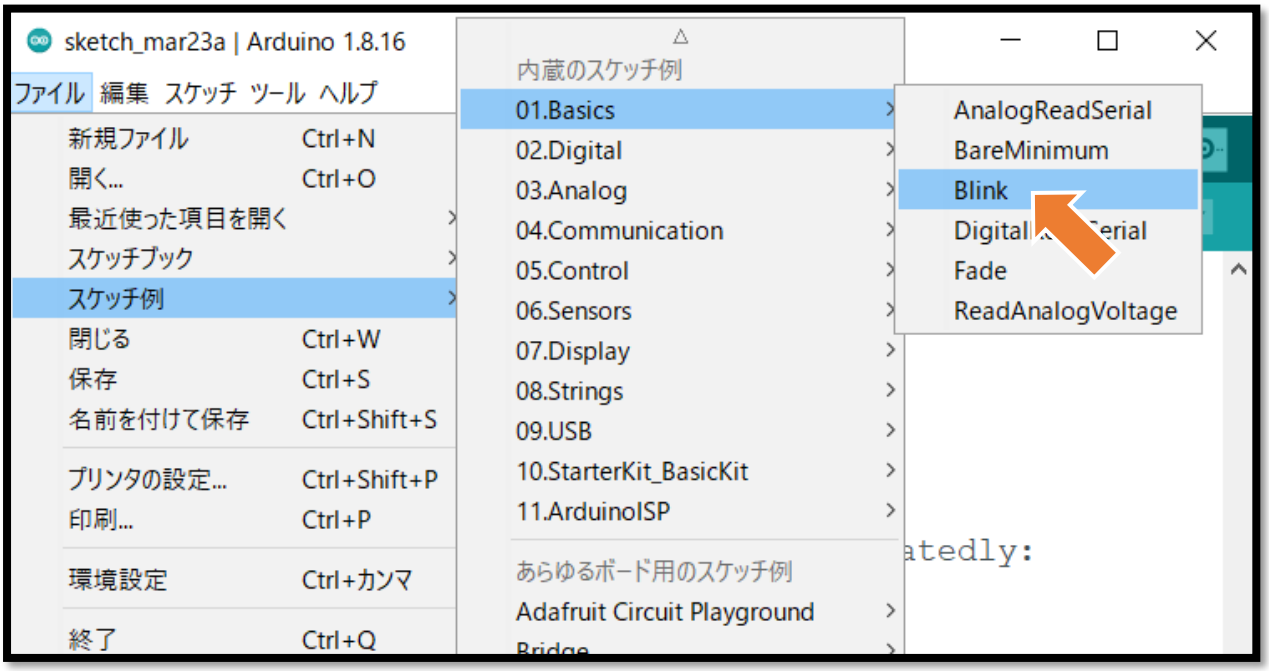
練習 2（ソフトウェア制御）

ESP32 に Micro-B USB ケーブルを差し込まずに以下の図のとおりに配線を変更しなさい。



配線を行った後、Micro-B USB ケーブルを ESP32 と実習機に繋ぎ通電する。
以下の手順に従い、LED を 1 秒間隔で点灯と消灯が繰り返されるかシリアルモニタ上でも確認しなさい。

① Arduino IDE を起動し、ファイルタブ内のスケッチ例メニューにある[01.Basics]->[Blink]を開く。



② 開かれたサンプルソースを確認する。
グレーアウトしている/* */で囲まれた文字列はこのサンプルソースのヘッダーコメントであり、説明やリファレンスの URL などが記載されている。

Blink

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/
```

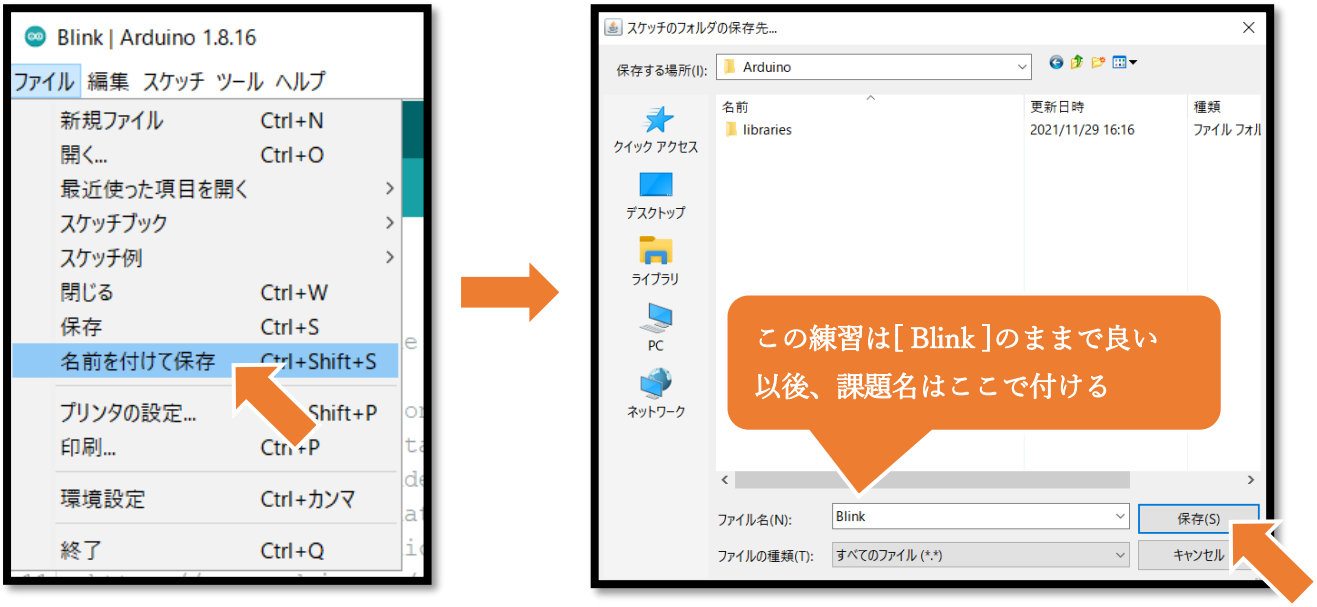
③ 続きを確認する。setup と loop という 2 つのメソッドが存在しているが、用途は以下の通りである。
・setup … ESP32 の起動時に一度だけ呼び出されるメソッド。主に初期化処理などを記述する。
・loop … setup 実行後に繰り返し呼び出されるメソッド。Java での while(true){～}の部分に該当する。
このメソッド内に繰り返し実行し続けるメインの処理を記述していく。

Blink

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

このように Arduino IDE にはスケッチ例に様々なサンプルソースが搭載されており、電子制御専用の開発環境なので、初期化メソッドとメインループメソッドがテンプレートとして用意されている。

④ サンプルソース[Blink]に処理を記述する前に、まずは名前を付けて保存する。
保存先はデフォルトのまま[Document]->[Arduino]フォルダの中で良い。
※[libraries]フォルダの中ではなく、同階層に保存すること



⑤ 以下のようにサンプルソースから追記・修正する。

Blink §

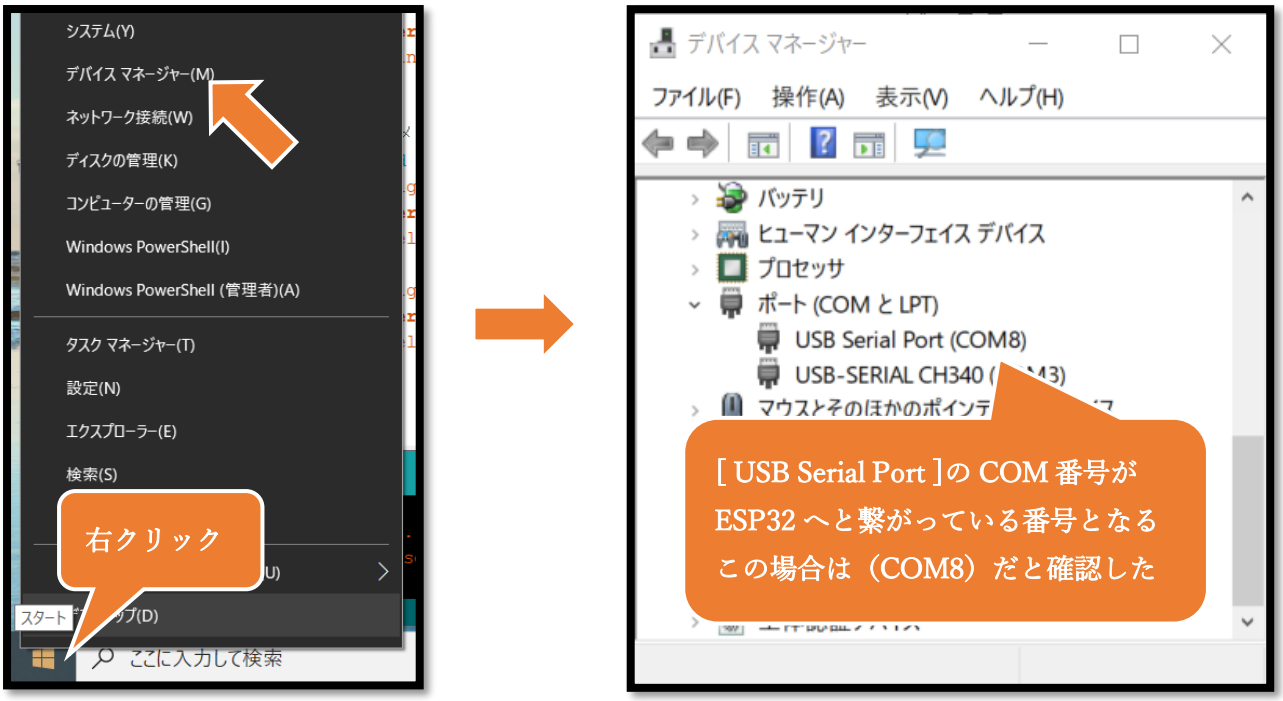
```
/*
  Blink
  Date : 2022/01/01
  Author : IE1A 99 K.Murakami
*/

int LED_BUILTIN = 25; //LEDのピン番号

// 起動時に一度だけ呼び出されるメソッド
void setup() {
  Serial.begin(115200); // シリアル通信の転送レート（bps）を設定
  pinMode(LED_BUILTIN, OUTPUT); // LEDのピンをデジタル出力に設定
}

// メインループメソッド
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // LEDにHIGHを出力し点灯する
  Serial.println("点灯");    // シリアルモニタに出力
  delay(1000);     // 1秒待機

  digitalWrite(LED_BUILTIN, LOW);    // LEDにLOWを出力し消灯する
  Serial.println("消灯");    //  シリアルモニタに出力
  delay(1000);     // 1秒待機
}
```

setup メソッド内でも loop メソッド内でも使用するので、大外に LED のピン番号の変数を記述する
Java でいうフィールド部分に該当する

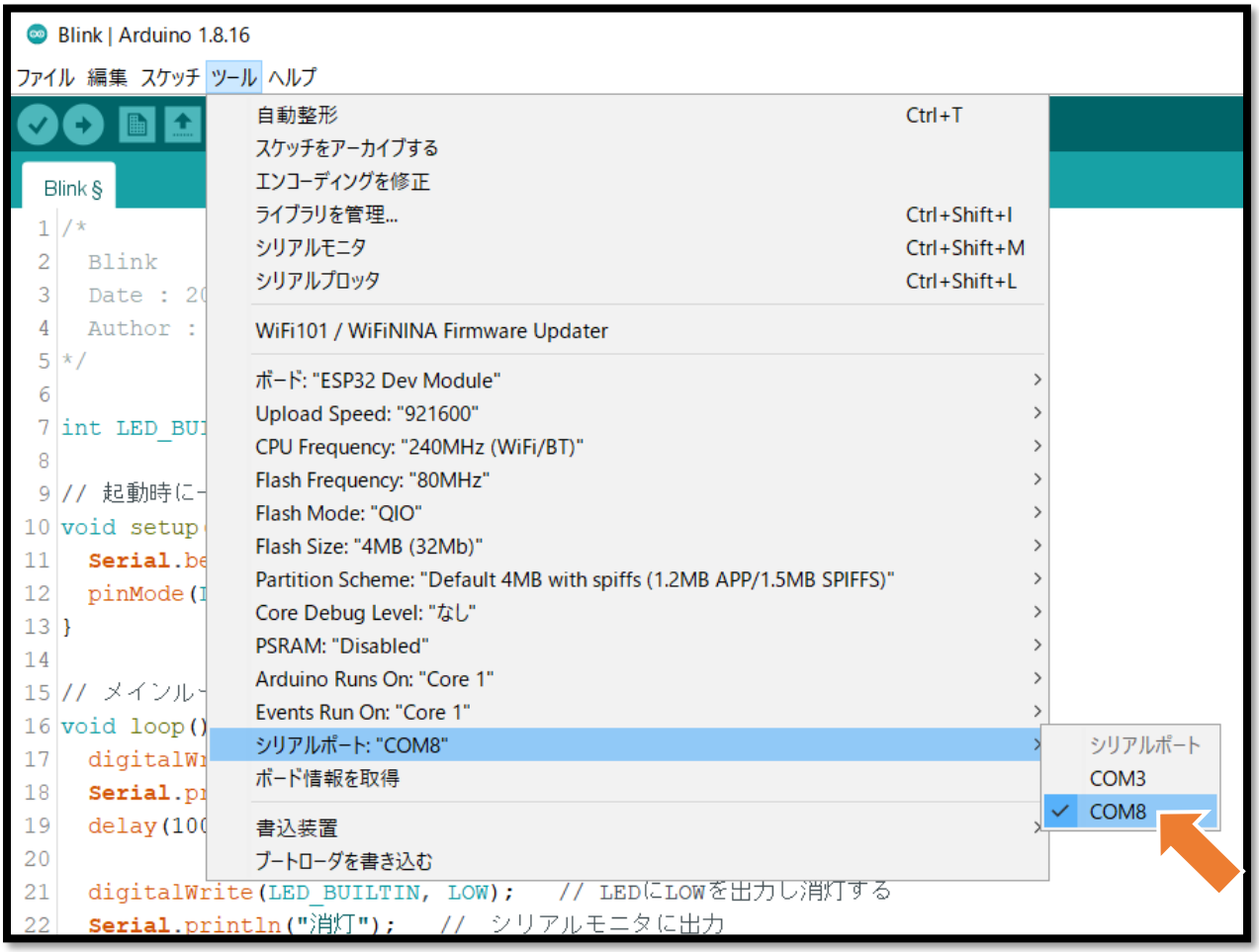
Raspberry Pi の時と同様、GPIO ピンにデジタル出力（OUTPUT）を設定する

Java の Thread.sleep(1000); と同じ意味

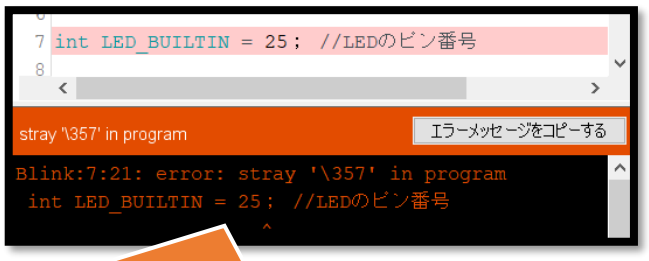
⑥ 書き込み先のシリアルポート番号を調べる為に、デバイスマネージャーを開いて確認する。



シリアルポート番号を確認したら、Arduino IDE のツールタブ内のシリアルポートメニューで確認したポート番号を選択する。



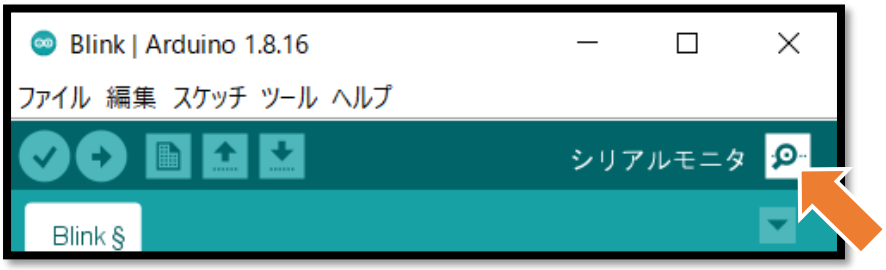
⑦ Arduino IDE の左上部にある（➡）ボタンをクリックして ESP32 へプログラムを書き込む。
書き込む前にコンパイルが行われ、エラーの場合は書き込みが中断される。



エラーメッセージ（セミコロンが全角）

コンパイル&書き込み完了メッセージ

⑧ 書き込み完了後、Arduino IDE の右上部にある[🔍]ボタンをクリックしてシリアルモニタを開く。
Serial.begin メソッドで設定した bps に合わせると、Serial.println メソッドの内容が表示される。



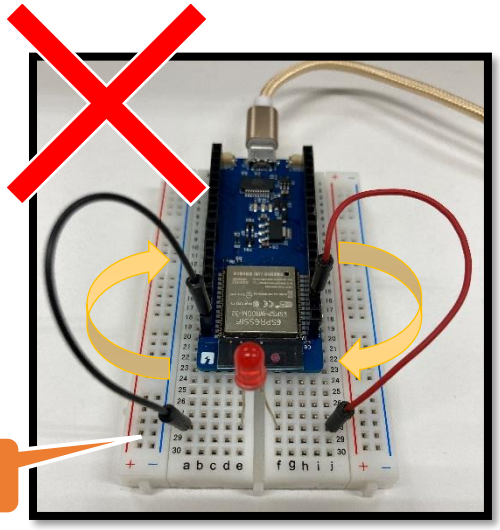
Serial.println(“点灯”); と、
Serial.println(“消灯”); が、
１秒間隔で実行されて出力される
シリアルモニタに出力することで、
処理内容を目で確認することができる

Serial.begin(115200);
と記述しているので同じ転送レートに設定

抵抗

抵抗とは、電子回路上で電気の流れを妨げる役割を持つ電子部品のこと。
抵抗には様々な種類があり、それぞれに抵抗値が設定されている。
その抵抗値を利用することで、電圧と電流の値を変えることができる。
抵抗値は皮膜上の色の羅列で確認することができる。
カラー抵抗早見表！（freelab.jp）

配線上での注意



この後の課題で配線を行いますがその際に LED の回路上で、
GPIO ピンから GND の間に抵抗を必ず入れること。
もし抵抗を入れなかった場合、オームの法則により
電流(A) = 電圧(V) / 抵抗(Ω)
の抵抗の値が 0Ω になる為、電圧値を 0 で割ることにより
電流が無限に増幅し、LED が耐え切れずに壊れます。
右の図のように抵抗を入れずに直結しないよう注意すること。
※一瞬で壊れるので興味本位でも絶対に試さないように。

ワイヤーと LED の間に抵抗を最低 1 つは入れる必要がある

課題（BlinkLed2）

前述の練習を別名保存して 2 つの LED が 1 秒間隔で点灯と消灯を繰り返すプログラムに修正しなさい。
2 つ目の LED 回路を追加する必要がある為、追加で必要な部品を考えて所定の位置から持って来ること。
配線を行う際は Micro-B USB ケーブルを ESP32 に差し込まずに、上記の注意点に気を付けて行うこと。
また、以下の 4 つの条件を満たすこと。

① 1 つ目の LED の回路は練習のまま変更しないこと
② 2 つ目の LED は IO26 番から出力して制御すること
③ 2 つ目の LED の出力信号用ジャンパーワイヤーの色は赤・黒以外の色を選択すること
④ 2 つ目の LED の抵抗値は 10kΩ（茶・黒・橙・金）にすること → 2 つ目の明るさはどうなった？

※課題が完成した人は ESP32 にプログラムを書き込んだ状態で講師を呼び、チェックしてもらうこと。

授業終了前に

Raspberry Pi の時と同様に使用した機材や部品類は元あった場所に必ず戻し現状復帰すること。
ブレッドボード上の部品類やワイヤーは外し忘れることが多い為、しっかり確認して元に戻すように。