

## 「IoT 制作演習 I」第 01 章 GPIO 基本 デジタル出力

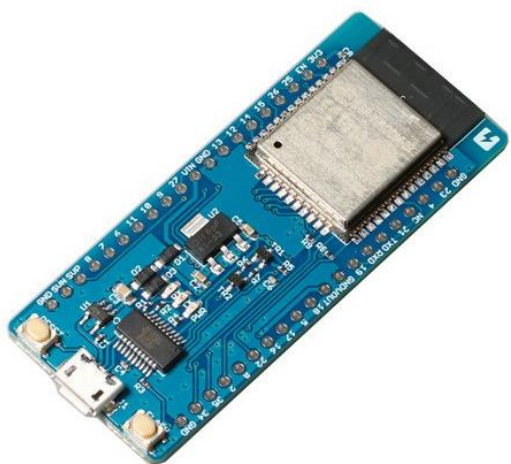
### 【授業の目標】

マイコンボード「ESPr Developer 32」が使えるようになる。

### 《何ができる?》

センサー（光や温度・湿度など）やアクチュエータ（モータなど）を制御することができる。

ESPr Developer 32 は、Wifi 接続や Bluetooth 接続ができるので、センサーからのデータをクラウドにアップロードしたり、スマホからモータなどを動かしたりできる。





Espressif Systems 社の ESP32-WROOM-32 というプロセッサを採用しており（以下、ESP32 と呼ぶ）、低消費電力のデュアルコア CPU を備える。

C/C++のマイコン用途の純正開発環境のほか、Arduino IDE 開発環境や、MicroPython ほか様々な開発言語の環境がある。

<https://www.switch-science.com/catalog/3210/>

ラズパイ、Arduino（アルデュイーノ）に続く新しい定番ボードとして今後普及が見込まれている。

IoT 開発で、よく用いられているボード「ラズパイ」と「Arduino（アルデュイーノ）」

	<p><b>ラズベリーパイ (Raspberry Pi 3 model B+)</b></p> <ul style="list-style-type: none"><li>・ボードコンピュータの代表格</li><li>・消費電力大</li><li>・SD カードに OS (Linux) を入れて動かす必要がある</li></ul> <p>ESP32 の方が安価、低消費、小型</p>
	<p><b>アルデュイーノ (Arduino Uno)</b></p> <ul style="list-style-type: none"><li>・マイコンボードの代表格</li><li>・安価でプロトタイプ作成に向く</li><li>・Wifi など無線機能はついていない</li></ul> <p>ESP の方が、無線通信機能や開発環境が充実している</p>

## 【第1章の目標】

開発環境をインストールして、実機（マイコンボード「ESP32」）につないだLEDを点滅させる（Lチカという）プログラムを作成し実行させる。

## 【1. 開発環境の構築】

- ① プロトタイプ開発で主流となっている Arduino IDE をインストールする  
Arduino IDE で用いられる開発言語は C/C++ ライクな文法の「Arduino 言語」でこれをもとに開発を行う。
- ② ESP32 で Arduino IDE を利用するには、「Arduino core for the ESP32」という機能拡張ライブラリをインストールする必要がある。

### 【1. 開発環境のインストール】

- ① Arduino IDE のインストール

Arduino サイトからインストーラをダウンロードする。

<https://www.arduino.cc/en/Main/Software>

## Downloads

自身の PC 内に既に Arduino 開発環境がある人は **不要**！

本科目の資料は version **1.8** で進めていく。  
version **2.0 以降** を使用する人は UI が変更されている為、各自で補完してください。



**Arduino IDE 1.8.13**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

**SOURCE CODE**

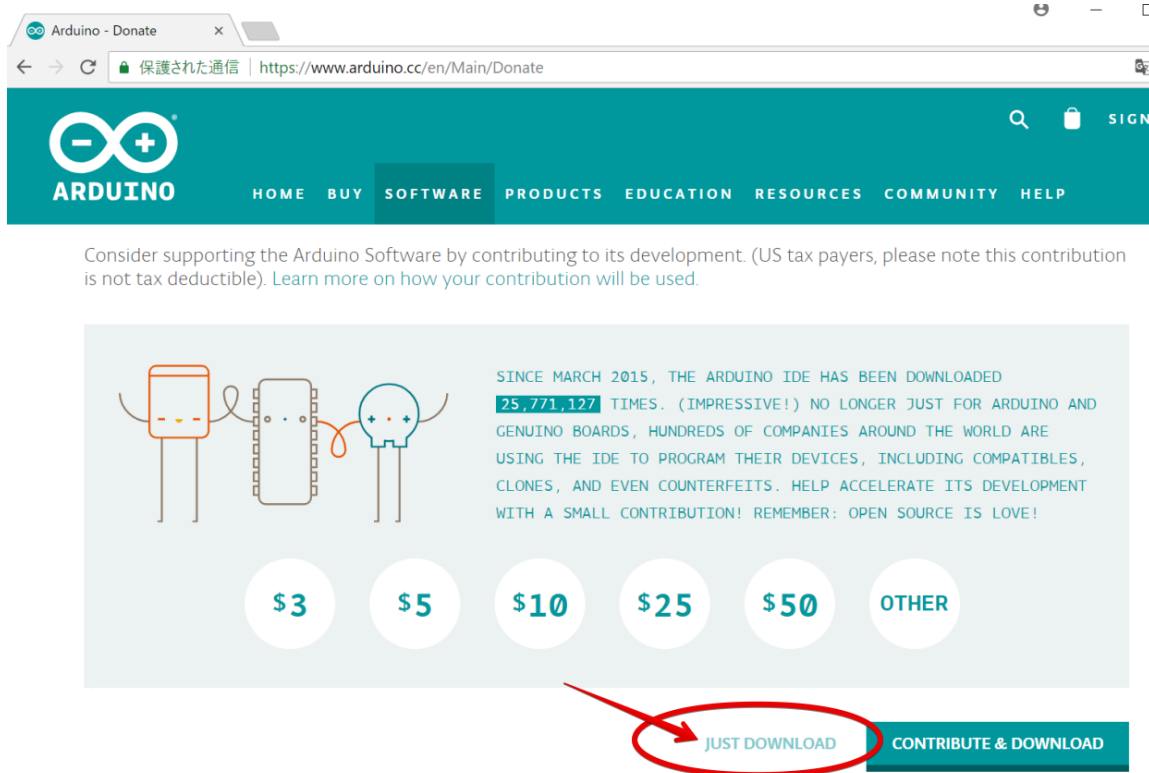
Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

**DOWNLOAD OPTIONS**

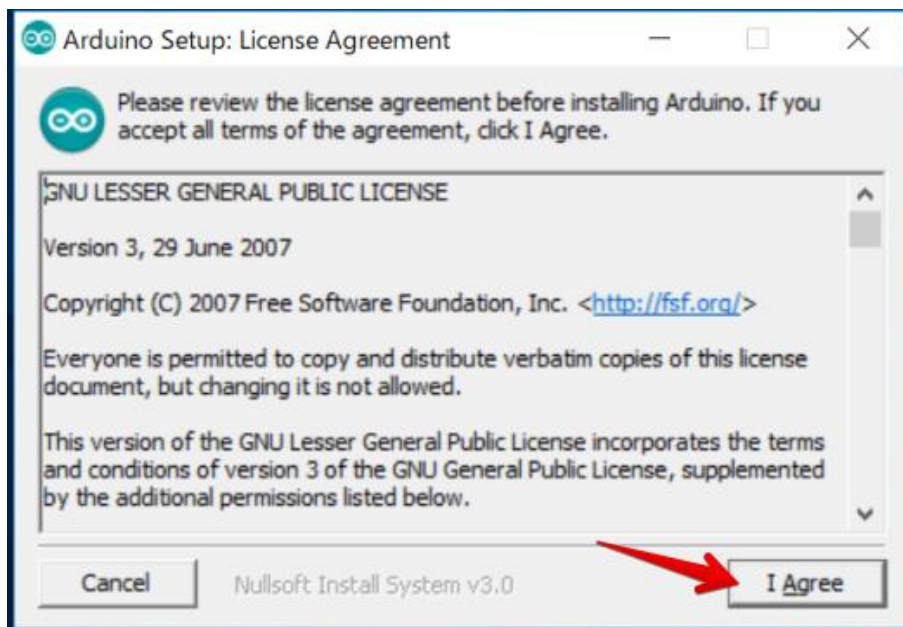
- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 **Get**
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

Release Notes Checksums (sha512)

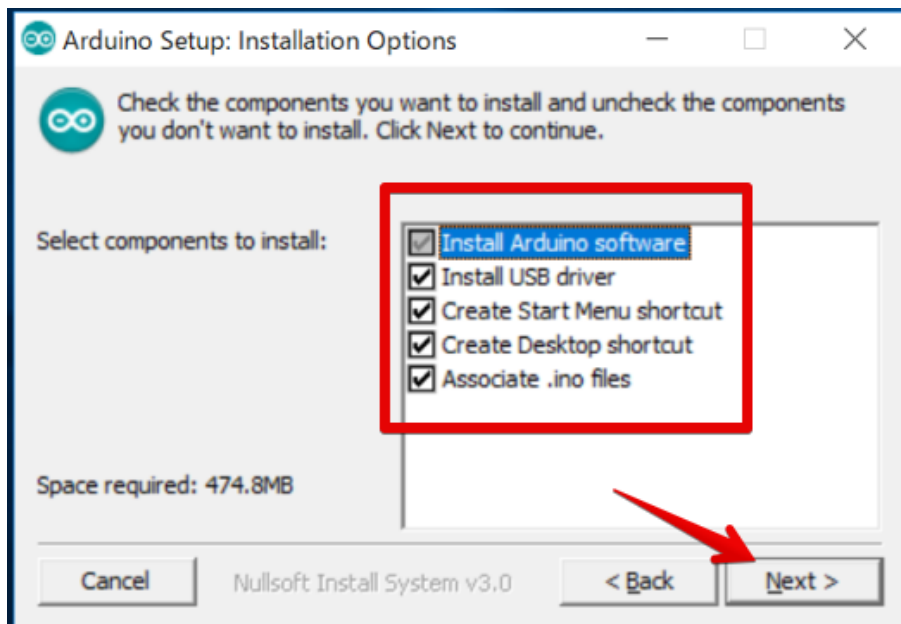
[Windows Installer for Windows XP and up]リンクをクリックする。



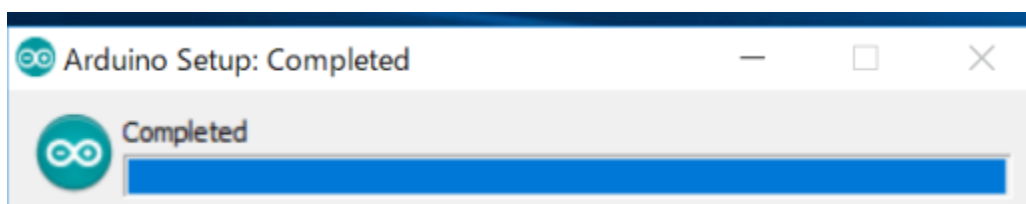
[JUST DOWNLOAD]をクリックしてインストーラの exe ファイルをダウンロードする。  
インストーラの exe ファイルをダブルクリックしてインストールをはじめる。管理者権限が必要になれば管理者ユーザーで続ける。ライセンス画面では[I Agree]をクリック



インストール・オプションの画面ではすべてのチェックが入ったままで[Next]をクリックして続ける



これ以降、保存する場所を指定する画面が出て、保存先はそのまま[Install]をクリック。  
インストールは数分かかる。



LLC ポート・USB のインストール許可ダイアログがでたら[YES]を選択する。  
Completed のメッセージが出れば完了。[Close]をクリックする。

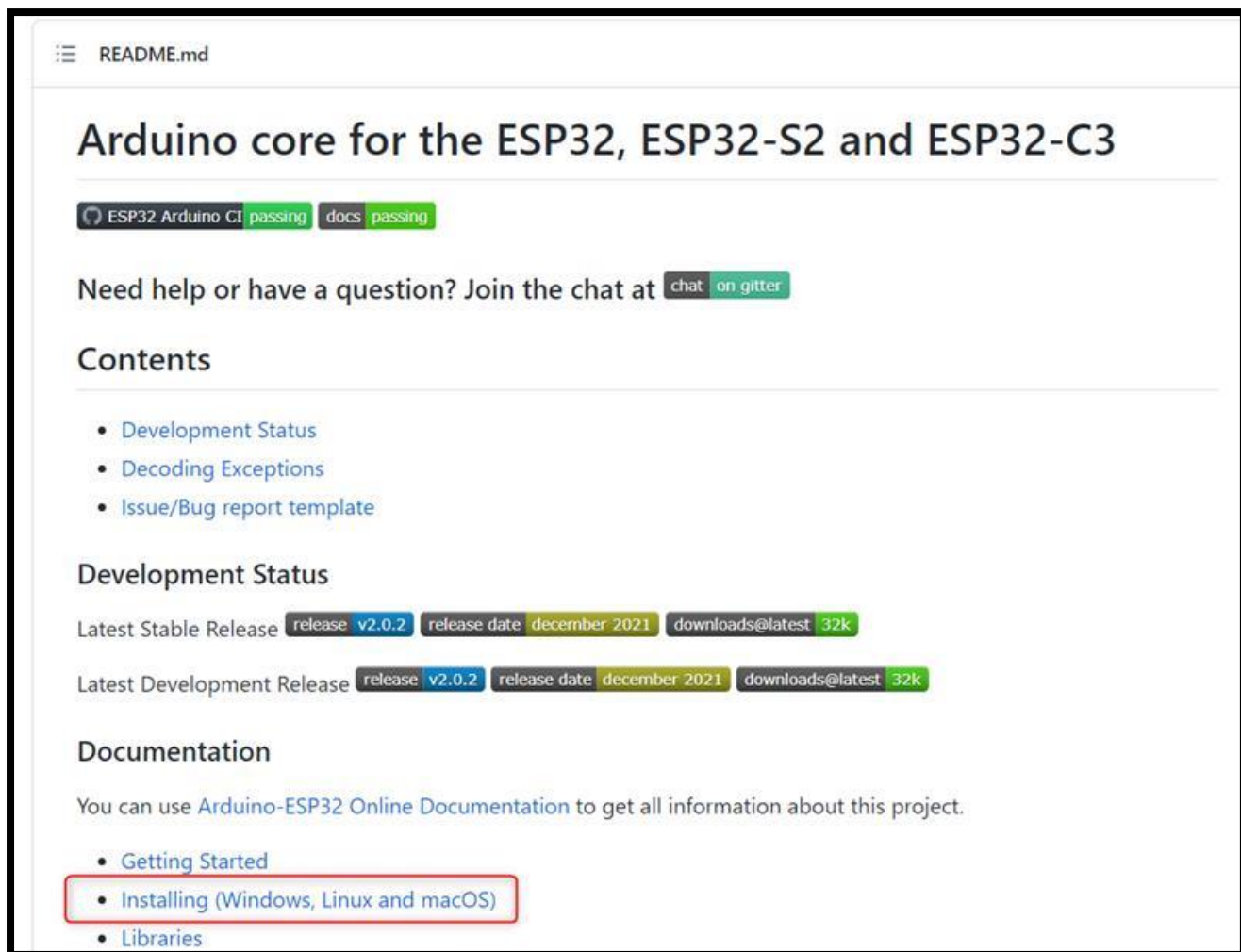
#### 【1. 開発環境のインストール】

② Arudino IDE に、「Arduino core for the ESP32」機能拡張ライブラリをインストールする

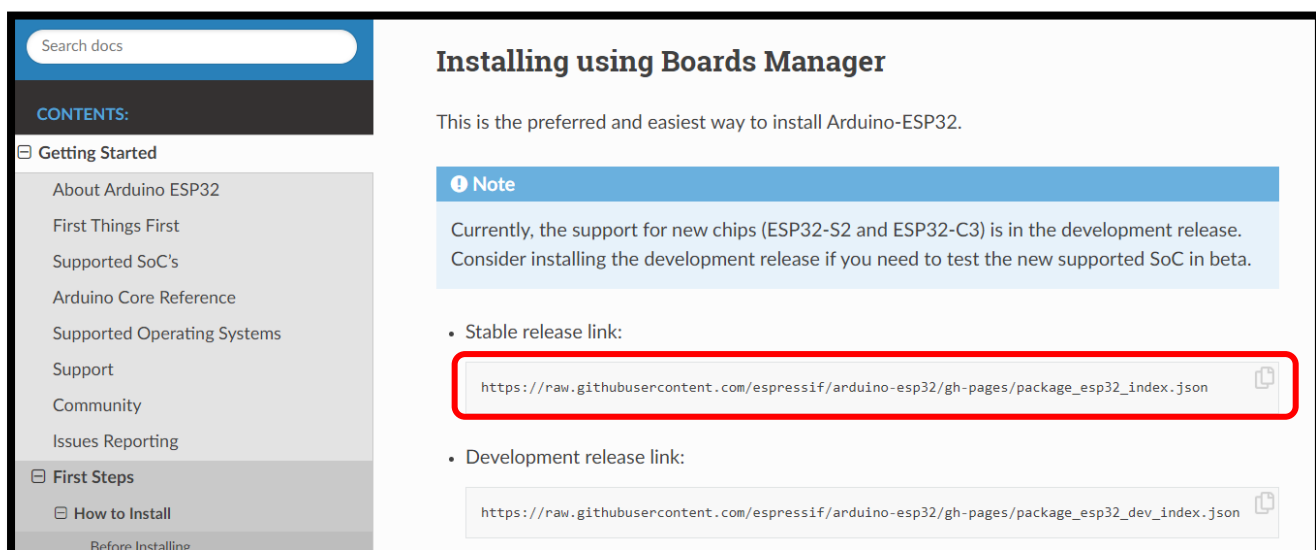
まず、ブラウザで以下のリンクの GitHub の Arduino core for the ESP32 のページを開く。

[GitHub - espressif/arduino-esp32: Arduino core for the ESP32](https://github.com/espressif/arduino-esp32)

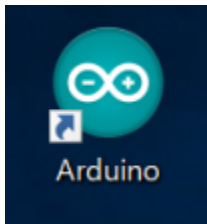
そのページでスクロールして下図の、赤枠のリンクをクリックして遷移する。



遷移先の Stable release link:の URL をコピーしておく。

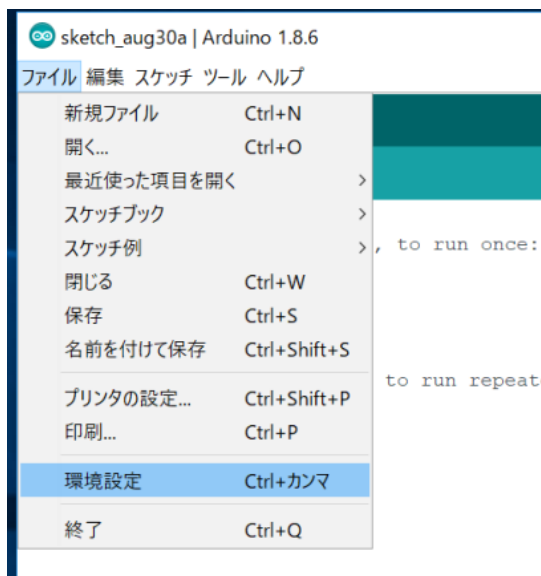


次に、Arudino IDE でコピーした URL を使って、ライブラリの設定を行う。

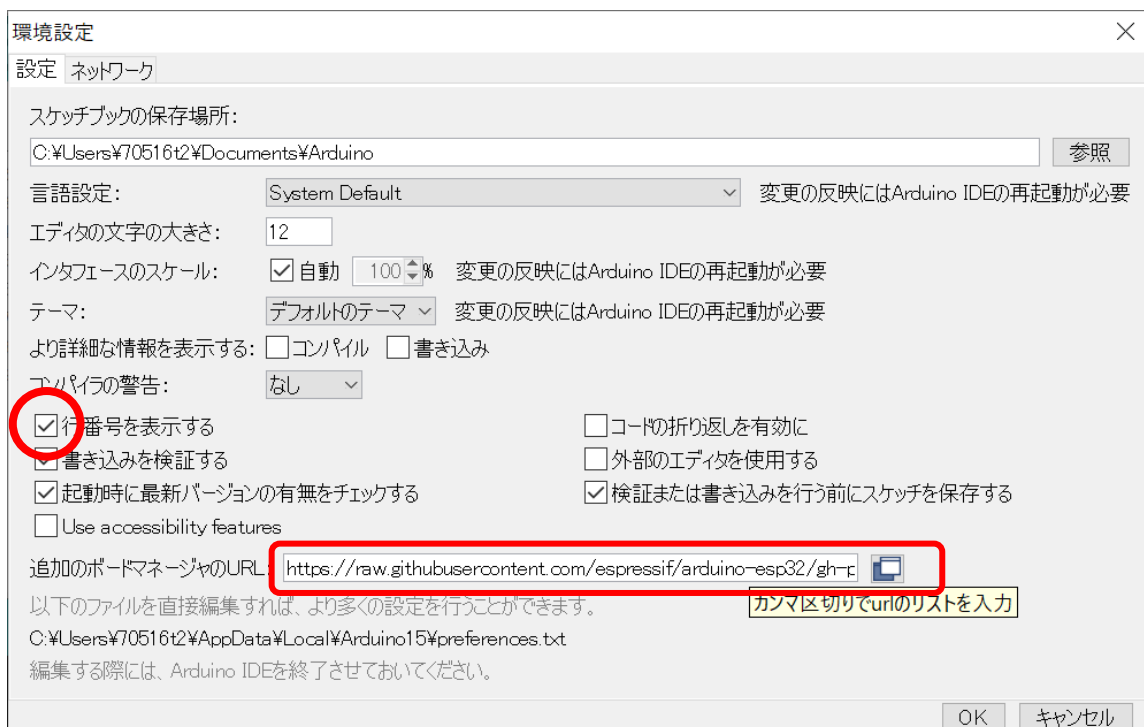


デスクトップに出来上がったアイコンをダブルクリックして Arduino IDE を起動する。

下図の様に[ファイル]-[環境設定]を選び、環境設定を開く。



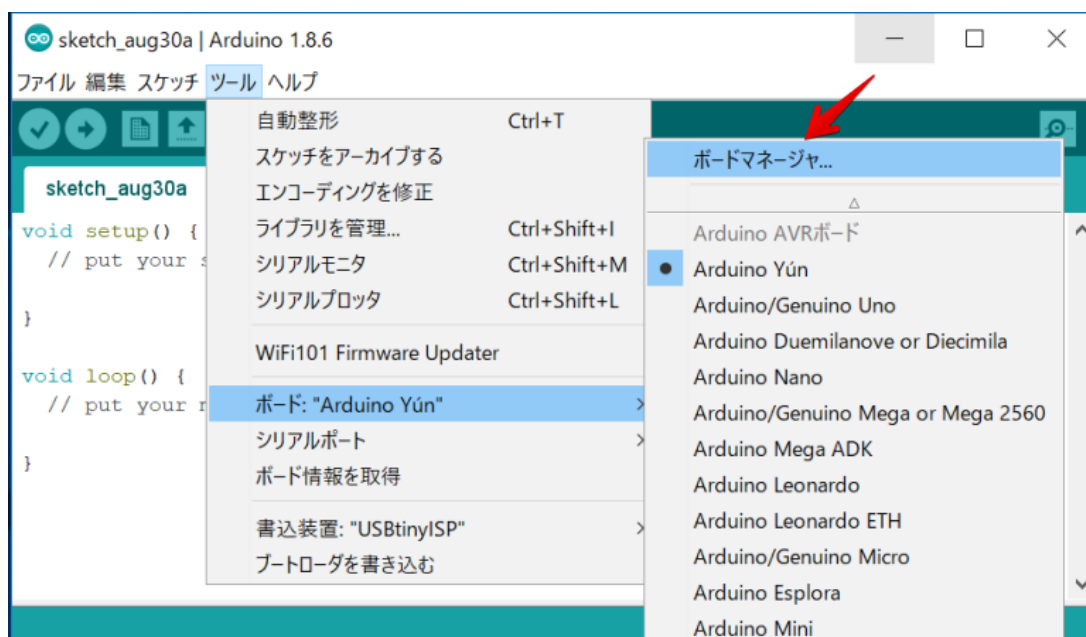
すると、下図の画面が表示されるので、下図の様に「追加のボードマネージャの URL:」欄に、先ほどコピーしておいた URL をペーストする。



ペーストが済めば、右下の[OK]ボタンをクリックして環境設定画面を閉じる。



次に、ツールメニューのボードマネージャを下図の様にクリックして開く。



すると、ボードマネージャが表示されて、ネット上のデータをダウンロードするので、少々待つ。

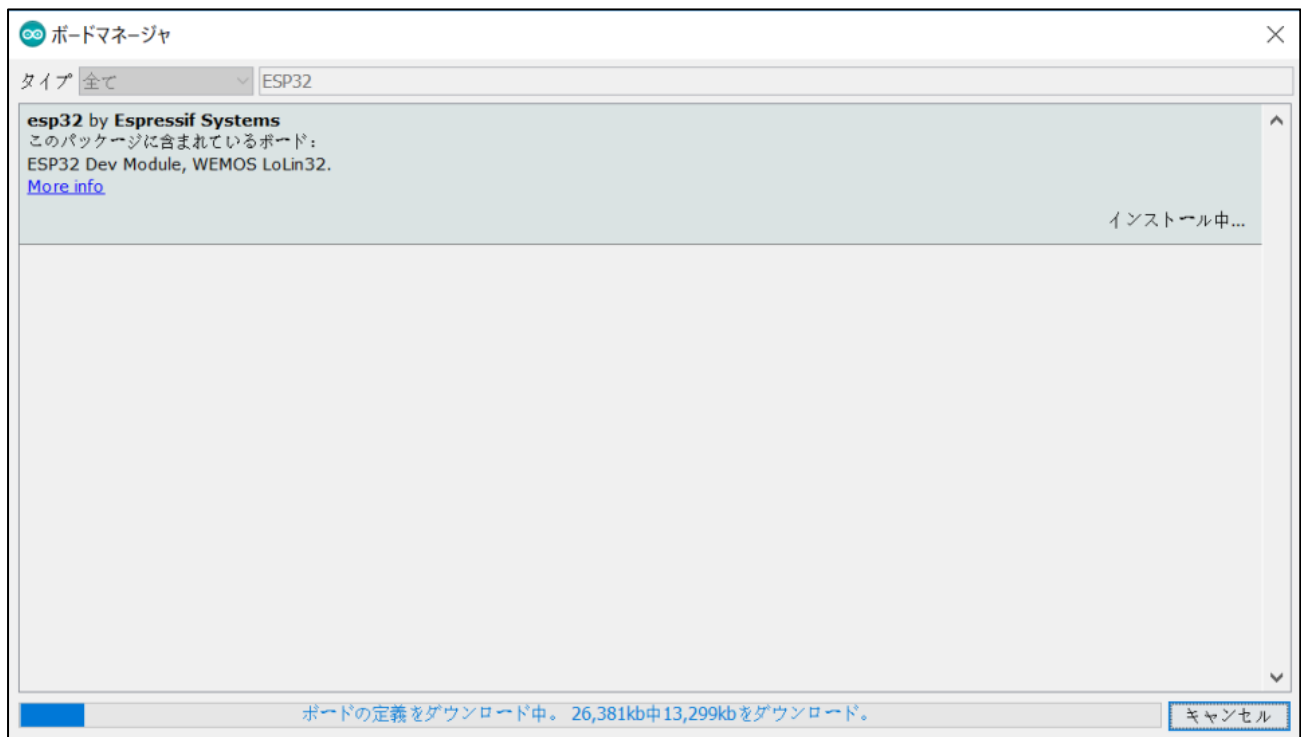


ダウンロードが終われば、検索フィールドをクリックし、「ESP32」と入力し、検索する。



ESP32 のパッケージが表示されるので、[インストール]をクリックしてください。

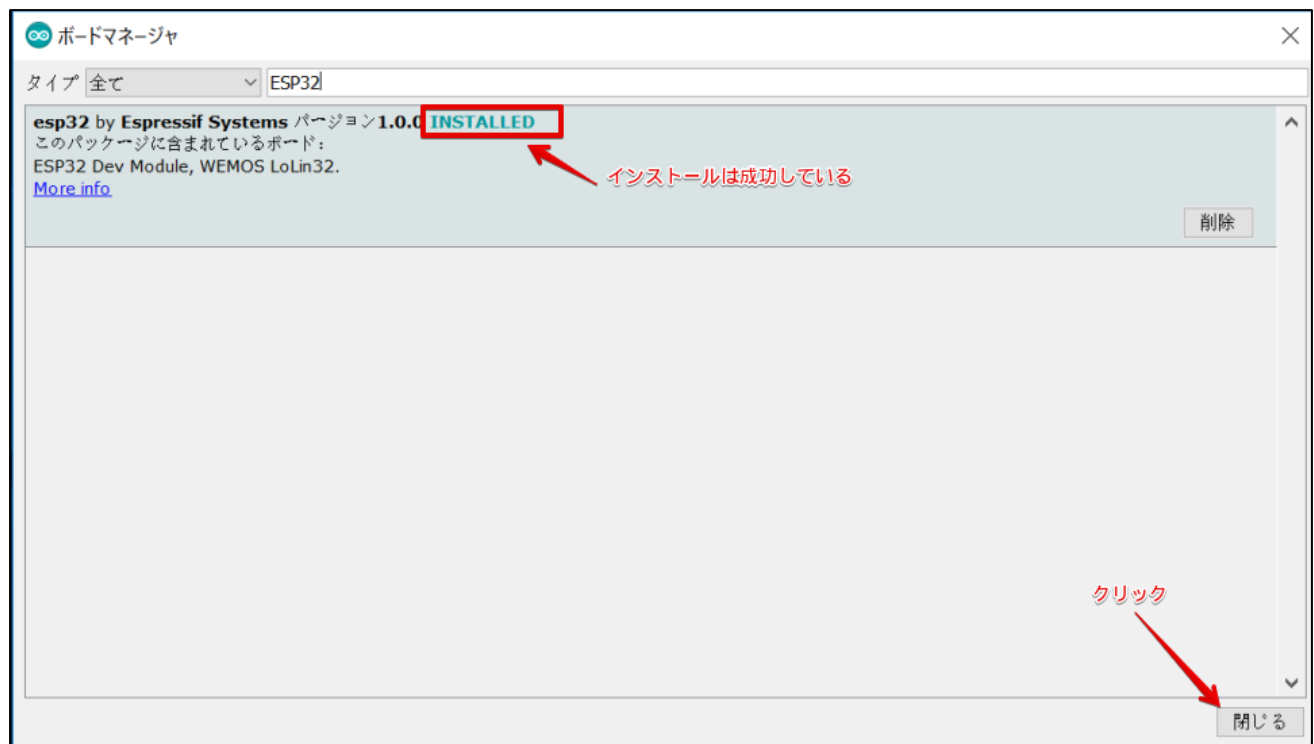
すると、下図の様にダウンロードが始まるので、少々待つ。





ダウンロードが完了して自動インストールが終了すると、下図のようになる。

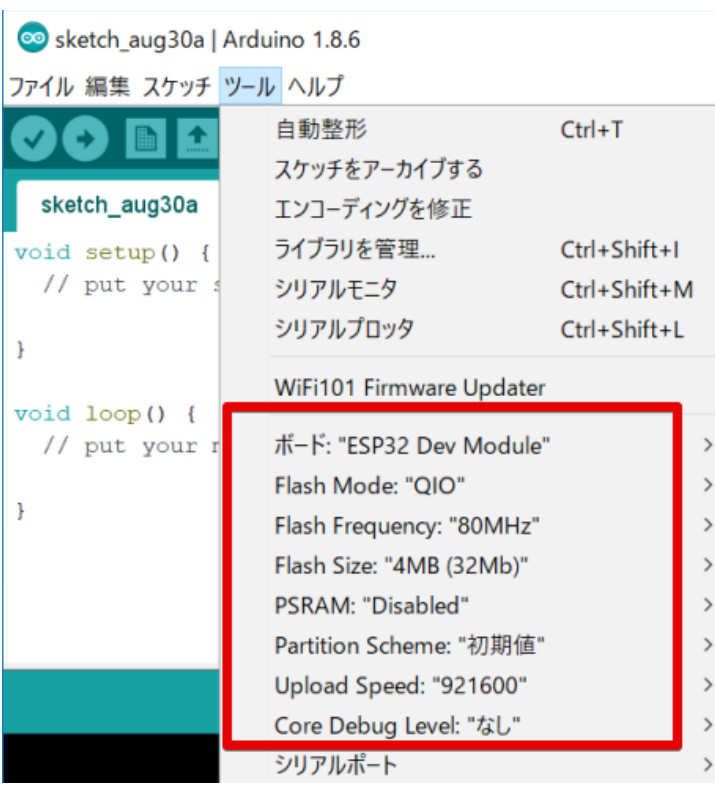
「INSTALLED」の文字があればインストールは完了している。[閉じる]をクリックする。



次に、[ツール]メニューから[ボード]を選び、メニューをスクロールして[ESP32 Dev Module]を見つけてクリックする。



ボードが“ESP32 Dev Module”になったら、ボードに書き込みための接続設定が以下の様になっているか確かめる。異なる場合は自分で設定すること。



ボード: “ESP32 Dev Module”

Flash Mode: “QIO”

Flash Size: “4MB (32Mb)”

Partition Scheme: “初期値”

Flash Frequency: “80MHz”

PSRAM: “Disabled”

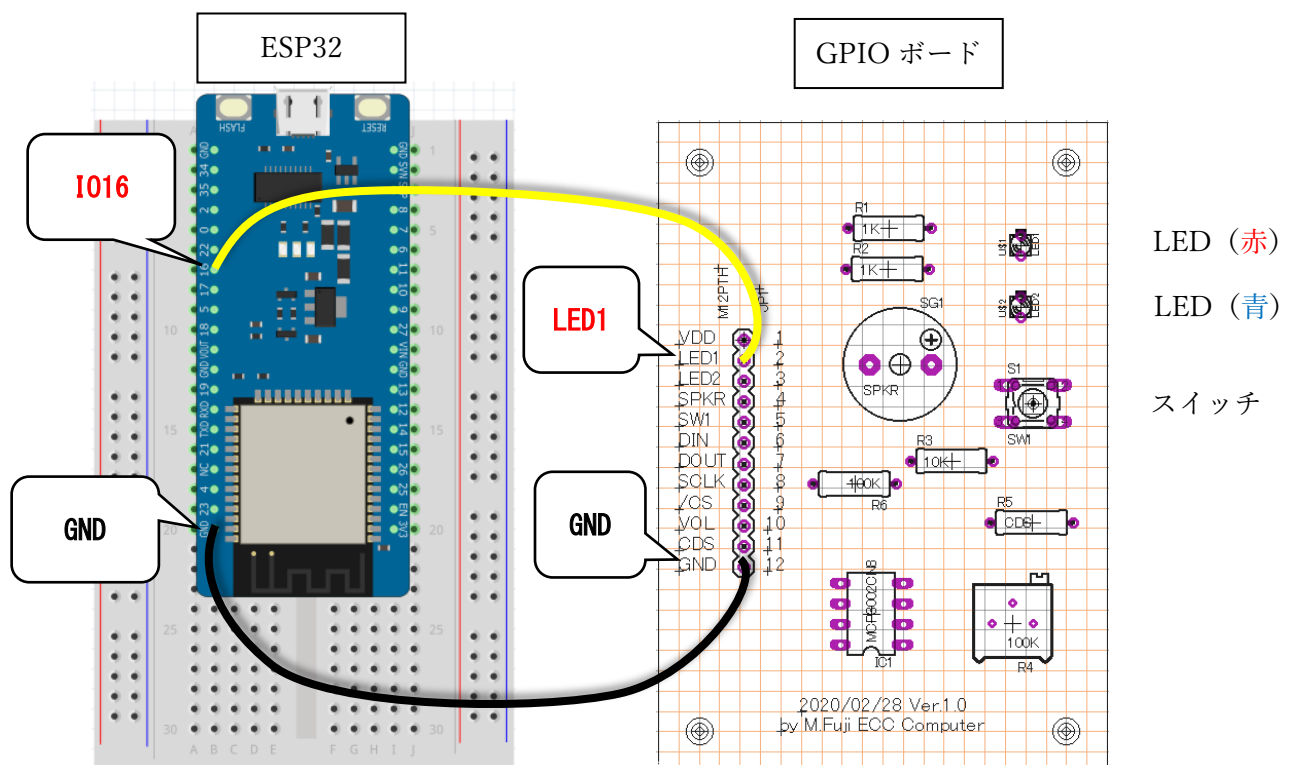
Upload Speed: “921600”

Core Debug Level: なし

シリアルポート: 自分の USB (COM) ポート

↑バージョンによって表示順が異なる場合がある。注意して設定を確かめること！

## 【2. ESP32 と電子工作部品との接続】



上記実体配線図を元に、5本のジャンパーコードの配線を行う。接続するピンは以下の一覧の通り。

- ① ジャンパーコード（黄）ESP32の **I016** – GPIOボード **LED1** (2番ピン)
- ② ジャンパーコード（黒）ESP32の **GND** – GPIOボード **GND** (12番ピン)

ジャンパーコードの色は電源（3V3,VOUT）が赤色、GNDは黒色が基本。

電源とGNDに赤黒以外の色は使用しないこと！！

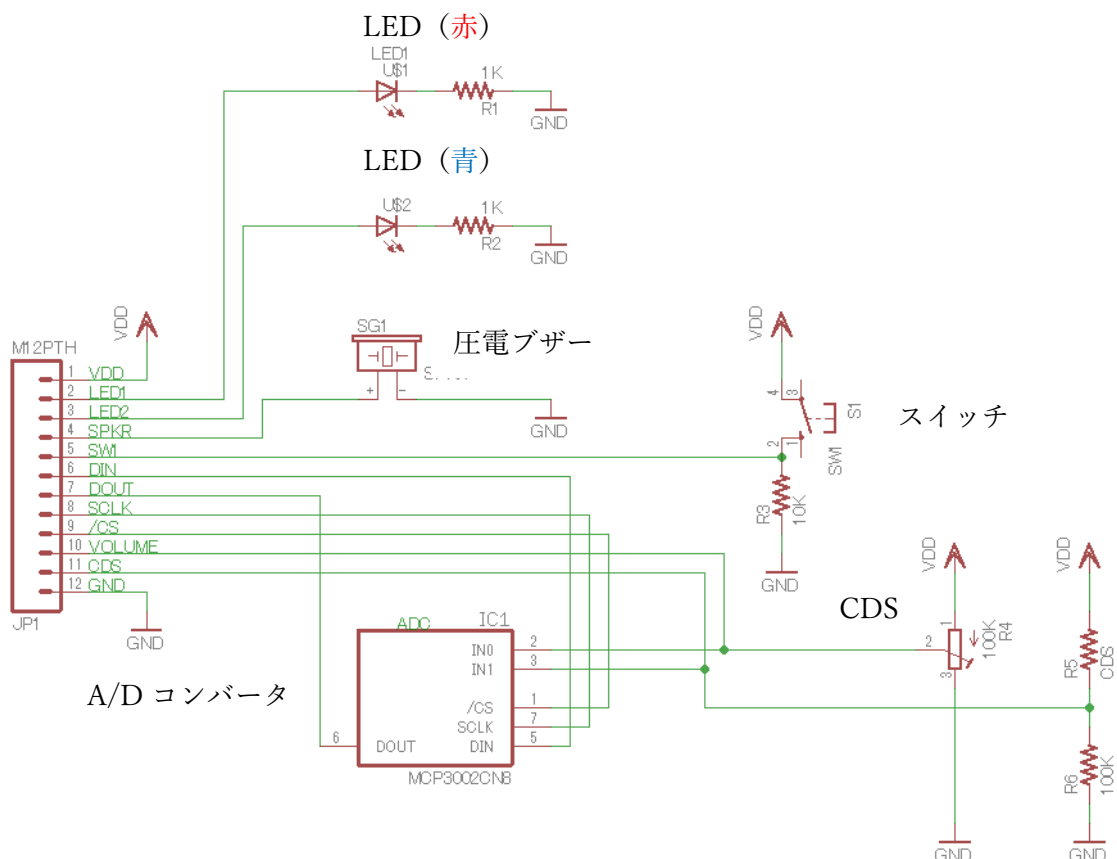
それ以外の信号線は何色を使用しても良く、部品ごとに色分けすると視覚的にわかりやすい。

### <<補足説明：GPIOボードについて>>

#### 使用目的

複数のクラスで授業が行われるため、自分が作成途中の回路を使うことが出来ないで汎用のGPIOボードを使用する。毎回結線する必要があるが最小限の作業で済むようにしている。

#### 回路図



## <ソフトウェア構成>

```
void setup()  
{  
  初期化処理を記述する  
}  
  
void loop()  
{  
  毎ループ実行する処理を記述する  
}
```

←setup 関数は実行時に一回だけ走る

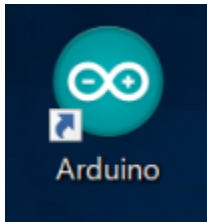
使用するポートの設定や変数の初期化等を行う

←loop 関数は繰り返し実行される

※終了しても永久に呼ばれる

### 【3. Arduino スケッチのサンプルプログラムをプログラム実行】

次に、Arudino IDE でコピーした URL 使って、ライブラリの設定を行う。



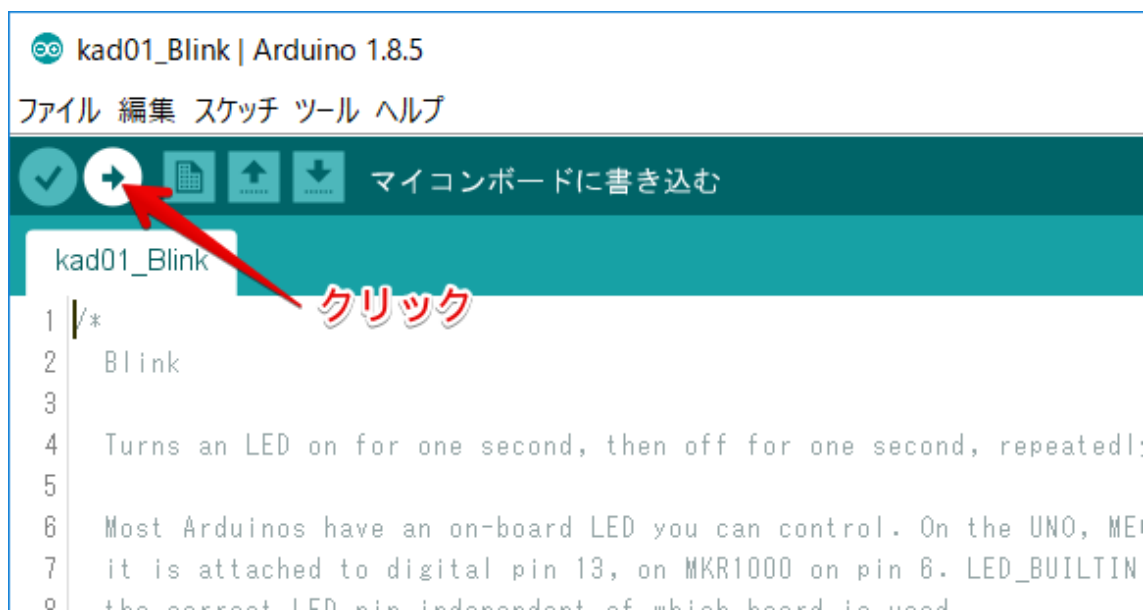
デスクトップに出来上がったアイコンをダブルクリックして Arduino IDE を起動する。

LED の点滅を行うサンプルプログラムを選ぶ。

[ファイル]メニューから[スケッチ例]—[01. Basics]—[Blink]を選ぶ。



→ボタンをクリックしてコンパイル&書き込みを行う。



LED\_BUILTIN が宣言されていないというメッセージが出てコンパイルを中止されてしまう。

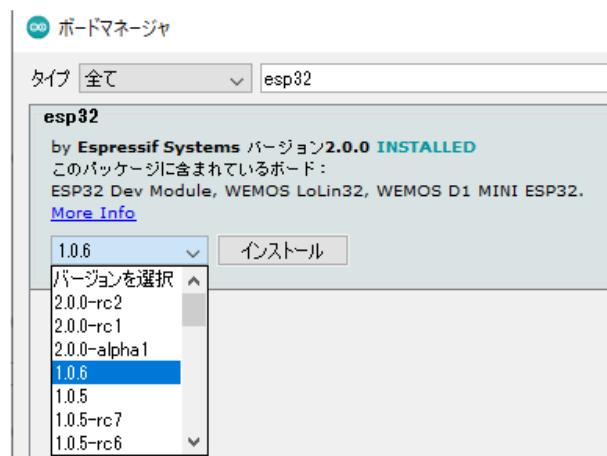
```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36 }
37
38 'LED_BUILTIN' was not declared in this scope
exit status 1
'LED_BUILTIN' was not declared in this scope
33 COM20のESP32 Dev Module, Disabled, Default, QIO
```

LED\_BUILTIN と書かれている部分（3 か所）を 16 に書きかえる。

```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(16, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(16, HIGH); // turn the LED on (HIGH is the voltage level)
34   delay(1000); // wait for a second
35   digitalWrite(16, LOW); // turn the LED off by making the voltage LOW
36   delay(1000); // wait for a second
37 }
```

LED が 1 秒おきに点滅すれば完成です。

※もし、それでもコンパイルエラーがでる場合はボードマネージャ:ESP32 のバージョンを 1.0.6 に下げてみる。



### 【課題 01】プロジェクト名 kad01\_Blink. ino

プロジェクト名を「kad01\_Blink」として提出する。

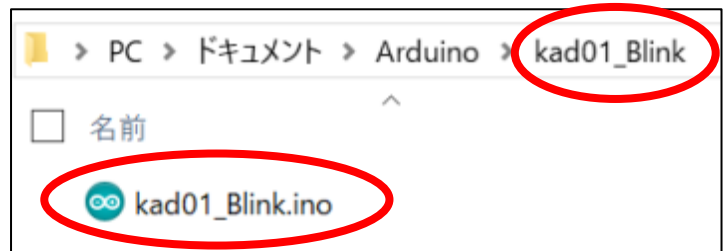
#### 《方法》

[ファイル]メニューから[名前をつけて保存]を選んで、ファイル名を「kad01\_Blink」とする。



※保存されたファイルは、ドキュメントフォルダの「Arduino」フォルダ内にできあがっている。

「kad01\_Blink」フォルダの中に「kad01\_Blink. ino」ファイルが保存されている。



### 【課題 02】プロジェクト名「kad02\_LED」

I016 番ピンに接続したLEDで200ミリ秒おきに点滅をくり返すスケッチを作成しなさい。

ただし、[ファイル]メニューから[新規ファイル]として作成されたテンプレートソースからプログラムを作成して試みる（課題01「kad01\_Blink」を参考にして作成すればよい）

### 【課題 03】プロジェクト名「kad03\_SOS」

I016 番ピンに接続したLEDでSOSのモールス信号が繰り返し点滅するようなスケッチを作成しなさい。

単音「・」200ミリ秒点灯

長音「ー」600ミリ秒点灯

消灯200ミリ秒

SOSのモールス信号「・・・——・・・」（トトトツーツーツートトト）

SOSの間は1秒あけて繰り返す（最後の長音の消灯時間はこの1秒に含めない）

※コードは関数やループ文を用いて、行数を極力減らす工夫をすること。