

My Book

cover sample for A5,
with bleed margin 3mm

published by Re:VIEW

はじめての IoT 講座

THEToilet 著

2021-07-28 版 発行

はじめに

これは電子計算機研究会の IoT 講座用に作った技術同人誌です。

サークルに参加するメリットの一つに、興味があることについて学べる機会がある。これがあげられるとおもいます。自分も一年の時にサークルの先輩から、いろいろな勉強会を開催していただき。自分の知見をひろげることができました。

(@THEToilet)

電子計算機研究会とは

芝浦工業大学公認の技術サークルです^{*1}。主にゲームや Web アプリの制作活動やコンピューターサイエンスの勉強を行っています。

お問い合わせ先

本誌に関するお問い合わせ : toileito.wc.benki@gmail.com

想定読者

- IoT に興味はあるがなかなか手をだせない人
- 通信に興味がある人
- 電子計算機研究会に所属している人

^{*1} 電子計算機研究会 HP <http://den3.net>

目次

はじめに	ii
電子計算機研究会とは	ii
お問い合わせ先	ii
想定読者	ii
第1章　電子部品の準備	1
1.1　　電子部品の購入の方法	1
1.2　　本誌で利用する電子部品	2
おすすめ製品	2
第2章　環境構築	5
2.1　　ESP32 とは	5
2.2　　ESP32 の開発環境	5
2.3　　Arduino IDE のインストール	6
2.4　　ESP32 用ボードマネージャーのインストール	11
2.5　　ESP32 用ライブラリのインストール	14
2.6　　Hello ESP32!!	16
プログラムの記述	16
ブレッドボード	19
PC との接続	20
プログラムの書き込み	22
動作確認	23
シリアル通信とは	23

目次

第 3 章 電子部品を使ってみよう	25
3.1 部品説明	25
LED	25
ジャンプワイヤ	25
抵抗	25
タクトスイッチ	26
3.2 L チカしよう！	27
プログラムで L チカ	27
タクトスイッチで L チカ	28
コラム: チャタリング	29
3.3 応用問題: 状態遷移	30
第 4 章 センサーのデータを Web 上に公開しよう	31
4.1 センサーを使おう	31
I2C とは	31
4.2 Web に公開しよう	39
Wi-Fi と接続する	39
ambient について	39
第 5 章 WebAPI を使おう	45
WebAPI とは?	45
5.1 Weather API を使う	45
コラム: サーバクライアント	53
5.2 応用問題: Web サーバからの L チカ	53
第 6 章 応用編	54
6.1 外部からエアコンの電源を操作する	54
6.2 2 台の ESP32 を使ってピンポンする	54
6.3 VScode から ESP32 にスケッチを書き込む	55
付録 A トラブルシューティング	65
A.1 シリアルモニタで文字化けがする	65
A.2 プログラムが書き込めない	66

目次

A.3	プログラムを書き込んだが動作に反映されない	67
A.4	error: redefinition	67
	著者紹介	69

第1章

電子部品の準備

本章では本誌のサンプルを進めるにあたって必要な電子部品および、その購入方法について紹介します。

1.1 電子部品の購入の方法

電子部品の販売店が近くにあれば直接商品を見ながら購入するのが一番ですが、お店が近くになかったり、コロナ渦の問題などで直接行くことが難しい場合は、通販での購入をおすすめします。下記の5つは電子部品を通販で購入できるサイトです。特に秋月電子通商、千石電商そしてaitendoは秋葉原に店舗があるので、機会があれば行くことをおすすめします。

- 秋月電子通商
 - <https://akizukidenshi.com/catalog/>
- 千石電商
 - <https://www.sengoku.co.jp/>
- スイッチサイエンス
 - <https://www.switch-science.com/>
- Amazon
 - <https://www.amazon.co.jp/>
- aitendo
 - <https://www.aitendo.com/>

1.2 本誌で利用する電子部品

筆者が本誌に使用するサンプルを作成するにあたって購入した商品を紹介します（表1.1）。本誌のサンプルを進めるにあたって必要になるため、参考にしてください。

表 1.1: 必要な材料

品名	個数	参考価格	詳細情報
ESP32DevKitC	1 個	1230 円	
microUSB Type-B	1 本	約 300 円	
プレッドボード	2 個	280 円 × 2	
LED	1 袋	150 円	
ジャンプワイヤセット（オス・オス）	1 セット	220 円	
抵抗 100 & 10k	100 : 1 袋 10k : 1 袋	100 円 × 2	
タクトスイッチ	1 個	10 円	
温湿度センサ	1 個	300 円	
ディスプレイ	1 個	580 円	
計		約 3550 円	

おすすめ製品

今回筆者はすべて秋月の通販にて電子部品を購入をしましたが、同じ製品であればどの店舗で購入しても差し支えありません。しかし、本誌は以下の製品で動作確認をしているため基本的には以下の製品を購入することをおすすめします。

ESP32DevKitC

ESP32 - DevKitC - 32E ESP32 - WROOM - 32E 開発ボード
4 MB

<https://akizukidenshi.com/catalog/g/gM-15673/>

ブレッドボード

ブレッドボード 6穴版 E I C - 3 9 0 1

<https://akizukidenshi.com/catalog/g/gP-12366/>

備考: ESP32DevKitC は幅が広いため、6穴のブレッドボードを使うことをおすすめします。

LED

5mm赤色LED 625nm 7cd 60度 (10個入)

<https://akizukidenshi.com/catalog/g/gI-01318/>

ジャンプワイヤセット(オス・オス)

ブレッドボード・ジャンパワイヤ(オス-オス)セット 各種 合計60本以上

<https://akizukidenshi.com/catalog/g/gC-05159/>

抵抗

カーボン抵抗(炭素皮膜抵抗) 1 / 4W 10k (100本入)

<https://akizukidenshi.com/catalog/g/gR-25103/>

カーボン抵抗(炭素皮膜抵抗) 1 / 4W 100 (100本入)

<https://akizukidenshi.com/catalog/g/gR-25101/>

備考: 上記の抵抗は100本単位からしか購入できません。実際に使用するのはどちらの抵抗値とも3本以下なので必ずしも100本買う必要はありません。

タクトスイッチ

タクトスイッチ(緑色)

<https://akizukidenshi.com/catalog/g/gP-03651/>

備考: 色の選択は自由です。

温湿度センサ

温湿度センサ モジュール DHT11

<https://akizukidenshi.com/catalog/g/gM-07003/>

ディスプレイ

0.96インチ 128×64ドット有機ELディスプレイ（OLED）白色

<https://akizukidenshi.com/catalog/g/gP-12031/>

第 2 章

環境構築

この章では ESP32 を利用するために必要な環境構築手順を紹介します。本誌は、Windows 環境を想定しているので、Mac 環境の方は少々が違う点がある可能性があります。

2.1 ESP32 とは

ESP32 とは Espressif Systems 社が開発した SoC(System on a Chip) シリーズの名前です。ESP32 という名前の使われ方には様々あり、今回使用する ESP32DevKitC は ESP32 をユーザが利用しやすい形にした製品ですが、通称として ESP32 と呼ばれることもあります。そこで、本誌では ESP32DevKitC も含めて ESP32 と呼んでいます。

2.2 ESP32 の開発環境

ESP32 の開発環境としては主に以下の 3 つがあげられます。

- Arduino IDE
 - Arduino 用 IDE
- ESP-IDF
 - ESP32 純正 IDE
- MicroPython
 - Python で書ける

今回は利用者が多く、関連情報がネット上に多く見られる Arduino IDE を用いて開発を

進めたいと思います。

2.3 Arduino IDE のインストール

Arduino IDE をインストールするために以下のリンクにアクセスしてください

<https://www.arduino.cc/en/software>

ダウンロード画面（図 2.1）ではご自身の PC 環境にあったダウンロードリンクを選択してください。ここからの手順では、Windows10 でのダウンロードを想定しています。

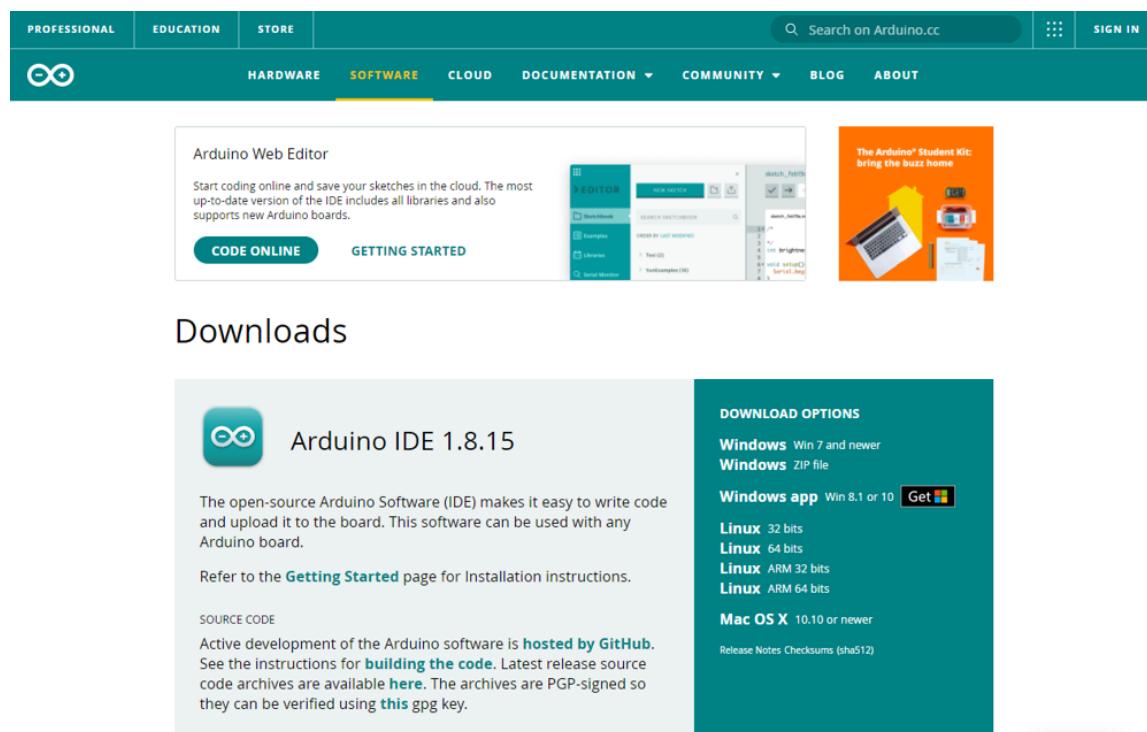


図 2.1: ArduinoIDE のダウンロード画面

ダウンロードリンクにアクセスすると、寄付金の金額選択画面に遷移します（図 2.2）。可能であれば寄付もできますが、JUST DOWNLOAD を選択することで次の画面に遷移します。

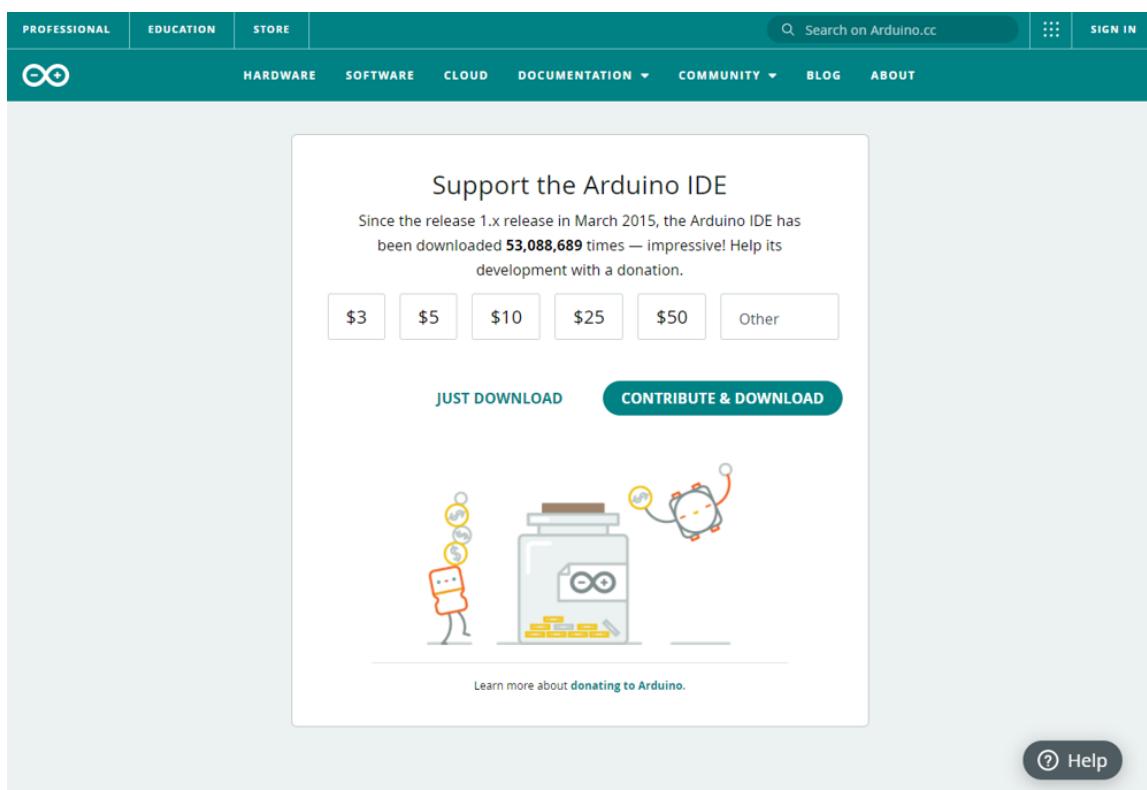


図 2.2: 寄付金の金額選択画面

JUST DOWNLOAD を選択するとブラウザ内で MicrosoftStore の画面に遷移します（図 2.3）。つぎは入手を選択すると、ブラウザのポップアップ上で Windows 側で MicrosoftStore を開く許可を求められるので許可をしてください。

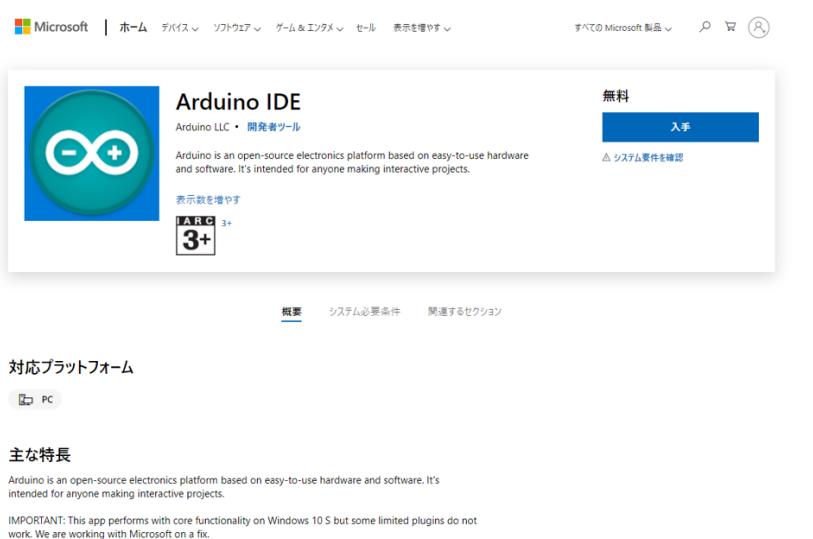


図 2.3: ブラウザで見る MicrosoftStore

Windows 上の MicrosoftStore です (図 2.4)。再度、入手を選択してください。

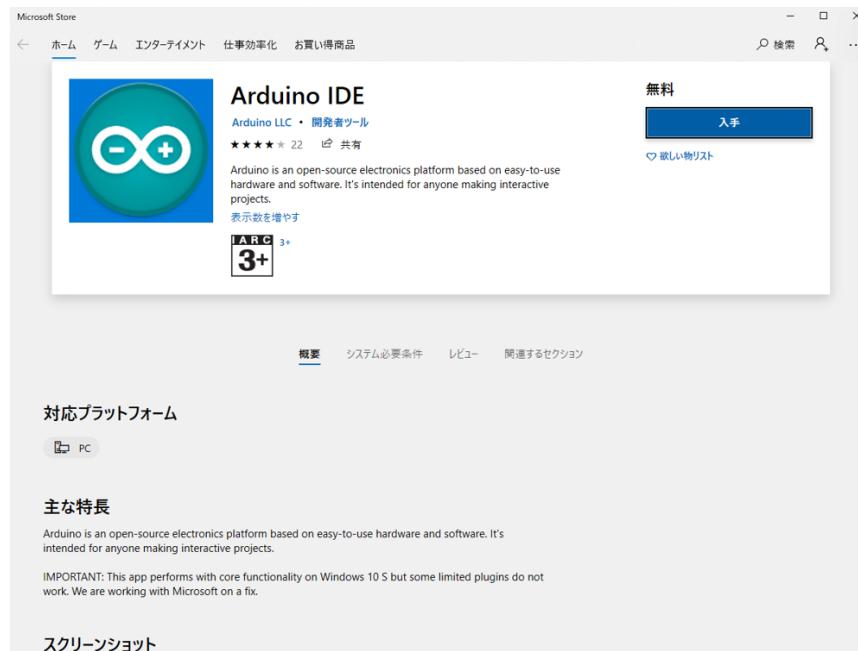


図 2.4: Windows で開いた MicrosoftStore

サインインについて尋ねられますが (図 2.5) 必要ありませんを選択した場合もダウンロードは開始されます。

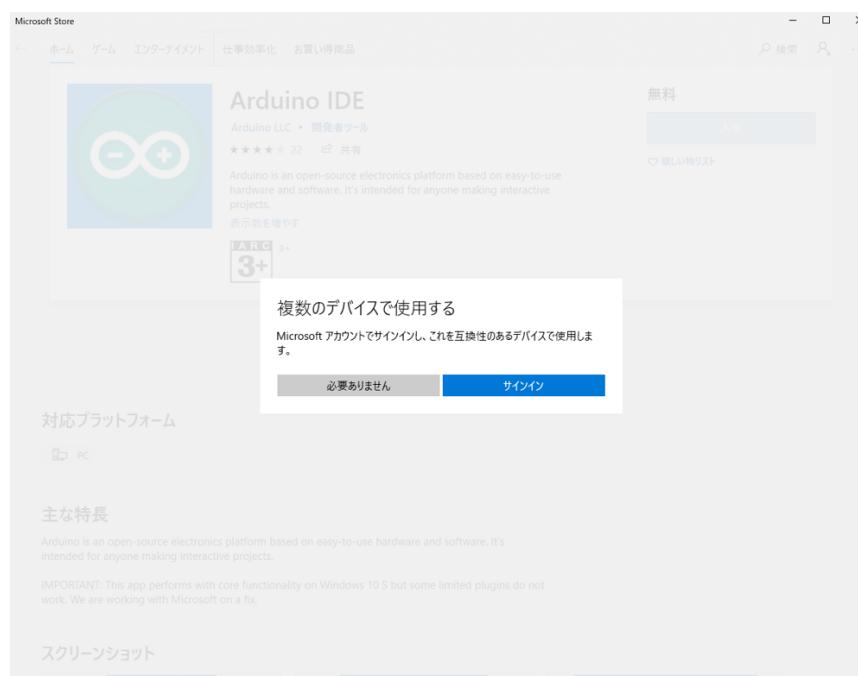


図 2.5: サインインの確認画面

図 2.6 では Arduino IDE のダウンロード状況を確認できます。



図 2.6: ダウンロードのキュー画面

ダウンロードが完了した後、検索窓にて Arduino IDE を検索し開いてください(図 2.7)。

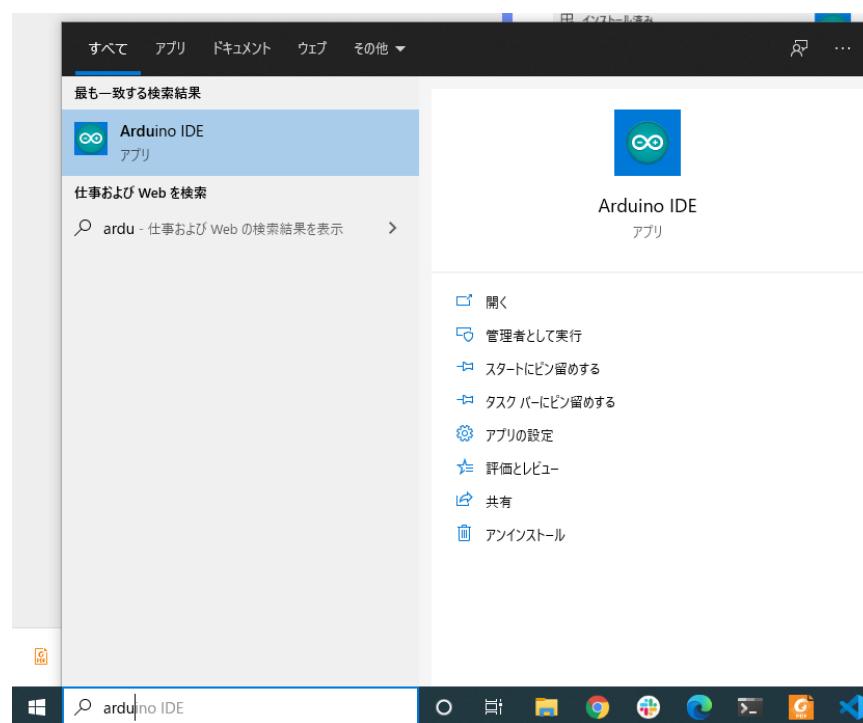


図 2.7: ArduinoIDE の検索

この際、セキュリティについての許可を求められるので(図 2.8) **アクセスを許可する**を選択すると、ブラウザのポップアップ上で Windows 側で MicrosoftStore を開く許可を求められるので許可をしてください。

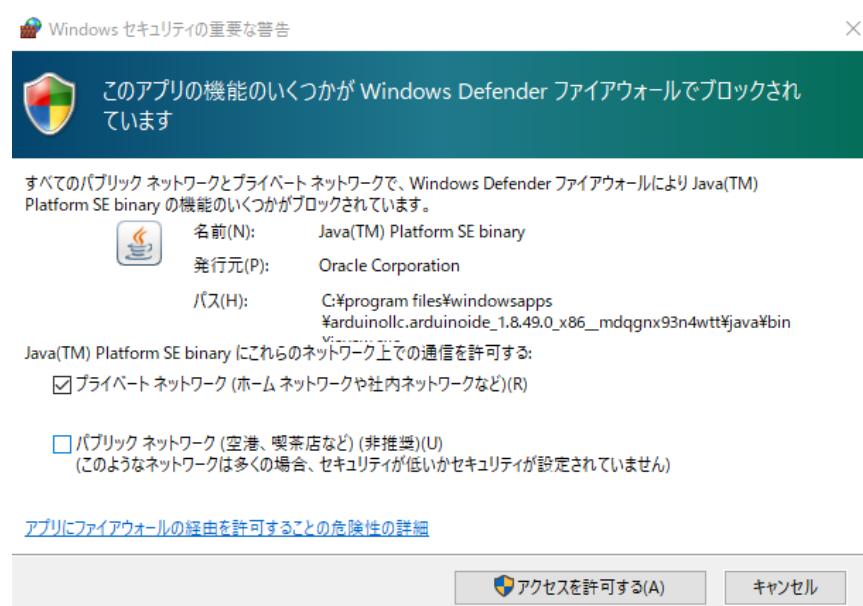


図 2.8: セキュリティの確認画面

Arduino IDE が起動すると、デフォルトの画面が表示されます（図 2.9）。

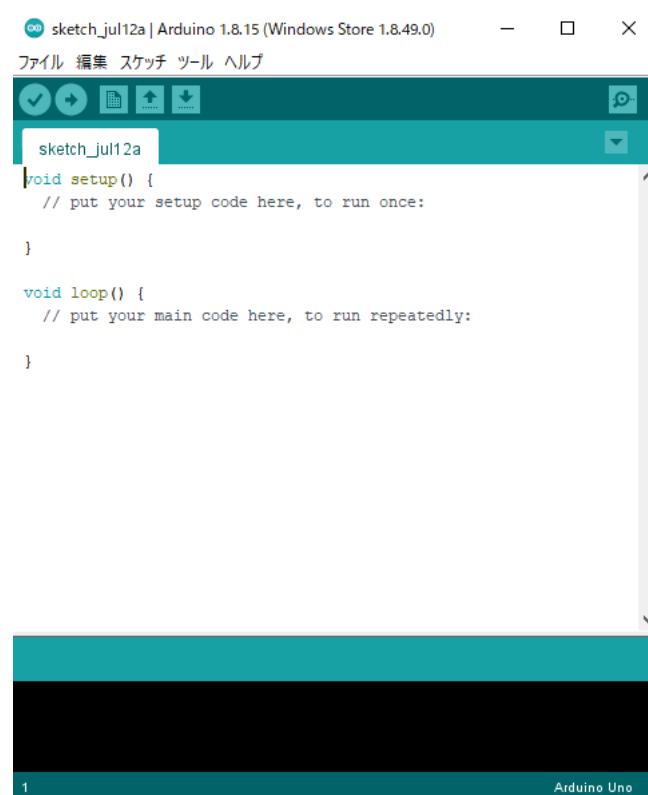


図 2.9: デフォルトのスケッチ画面

2.4 ESP32用ボードマネージャーのインストール

Arduino IDEにてESP32を使うために必要なボードマネージャーのインストールの流れを紹介します。

図2.10はESP32のボードマネージャーを追加するための手順であり、以下のリンクに記載されています。https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md

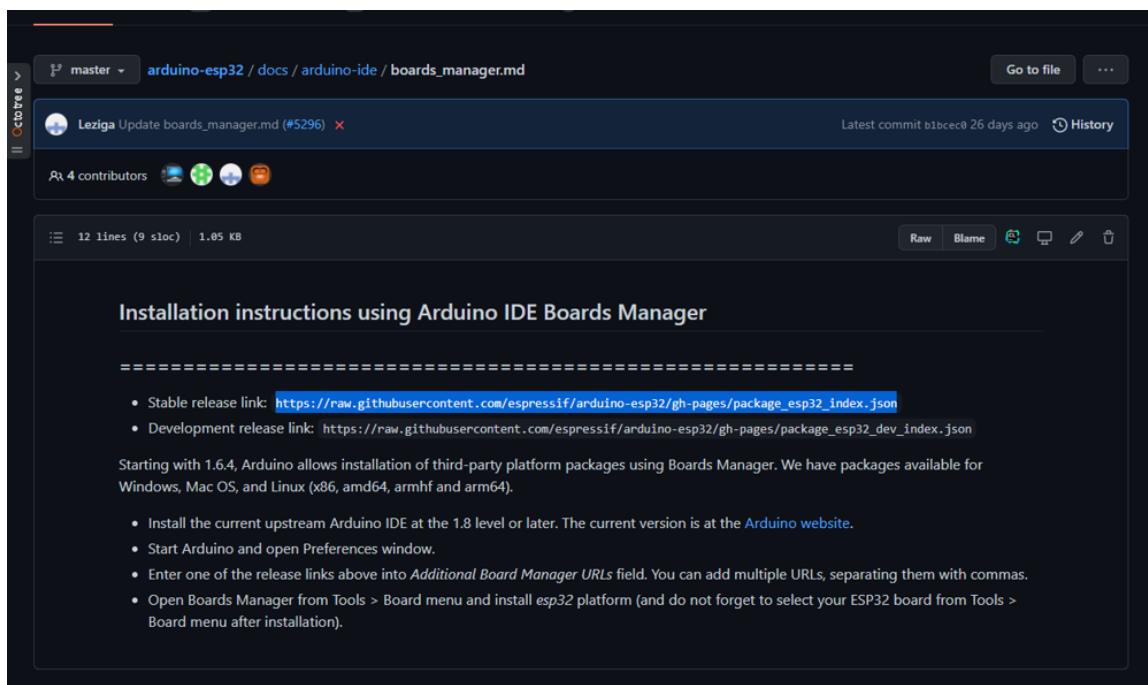


図2.10: ESP32をArduinoIDEで使うための設定

手順に従い以下のリンクをコピーしてください。以下のリンクには、図2.11のような情報が記載されています。

リスト 2.1: ボードマネージャーのリンク

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```



```
{  
  "packages": [  
    {  
      "name": "esp32",  
      "maintainer": "Espressif Systems",  
      "websiteURL": "https://github.com/espressif/arduino-esp32",  
      "email": "hristo@espressif.com",  
      "help": {  
        "online": "http://esp32.com"  
      },  
      "platforms": [  
        {  
          "name": "esp32",  
          "architecture": "esp32",  
          "version": "1.0.6",  
          "category": "ESP32",  
          "url": "https://github.com/espressif/arduino-esp32/releases/download/1.0.6/esp32-1.0.6.zip",  
          "archiveFileName": "esp32-1.0.6.zip",  
          "checksum": "SHA-256:982da9aaa181b6cb9c692dd4c9622b022ecc0d1e3aa0c5b70428ccc3c1b4556b",  
          "size": "51126692",  
          "help": {  
            "online": ""  
          },  
          "boards": [  
            {  
              "name": "ESP32 Dev Module"  
            },  
            {  
              "name": "WEMOS LoLin32"  
            },  
            {  
              "name": "WEMOS D1 MINI ESP32"  
            }  
          ],  
          "toolsDependencies": [  
            {  
              "packager": "esp32",  
              "name": "xtensa-esp32-elf-gcc",  
              "version": "1.22.0-97-gc752ad5-5.2.0"  
            },  
            {  
              "packager": "esp32",  
              "name": "esptool_py",  
              "version": "3.0.0"  
            },  
            {  
              "packager": "esp32",  
              "name": "askspiffs"  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```

図 2.11: ESP32用のボードマネージャ情報

Arduino IDE 側では、ファイル>環境設定を選択してください(図 2.12)

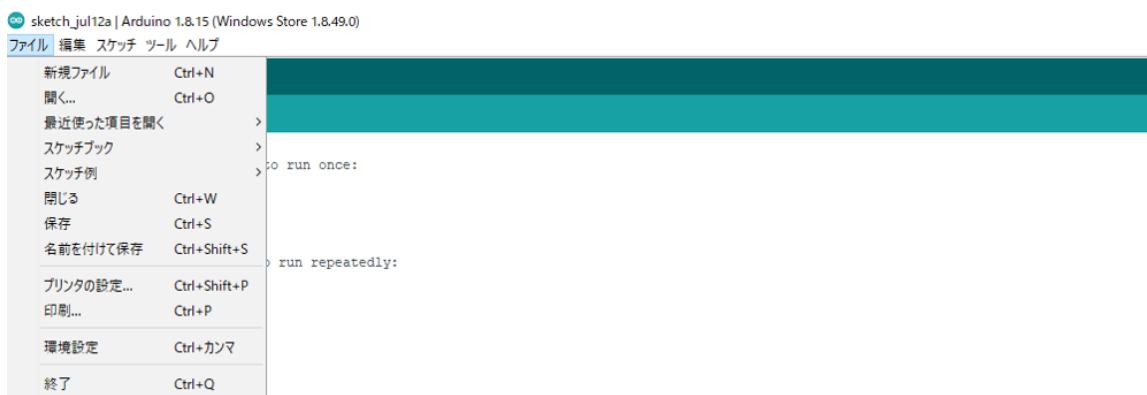


図 2.12: 環境設定を選択

環境設定の画面が表示されていることを確認してください(図 2.13)。

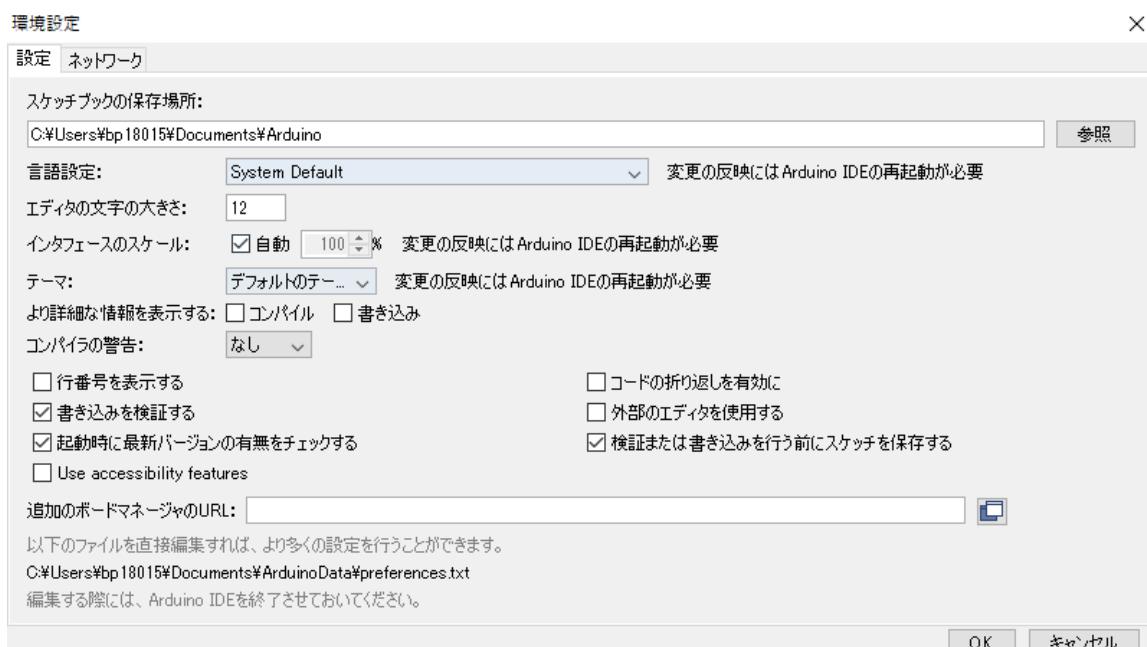


図 2.13: 環境設定の画面

次に先ほどコピーしたリスト 2.1 を追加ボードマネージャの URL の欄にペーストしてください(図 2.14)。



図 2.14: 追加ボードマネージャーの URL に貼り付ける

その後、OK を選択してください。

2.5 ESP32 用ライブラリのインストール

次に ESP32 用のライブラリを Arduino IDE にインストールします。

図 2.15 のように スケッチ > ライブラリのインクルード > ライブラリを管理 を選択してください。

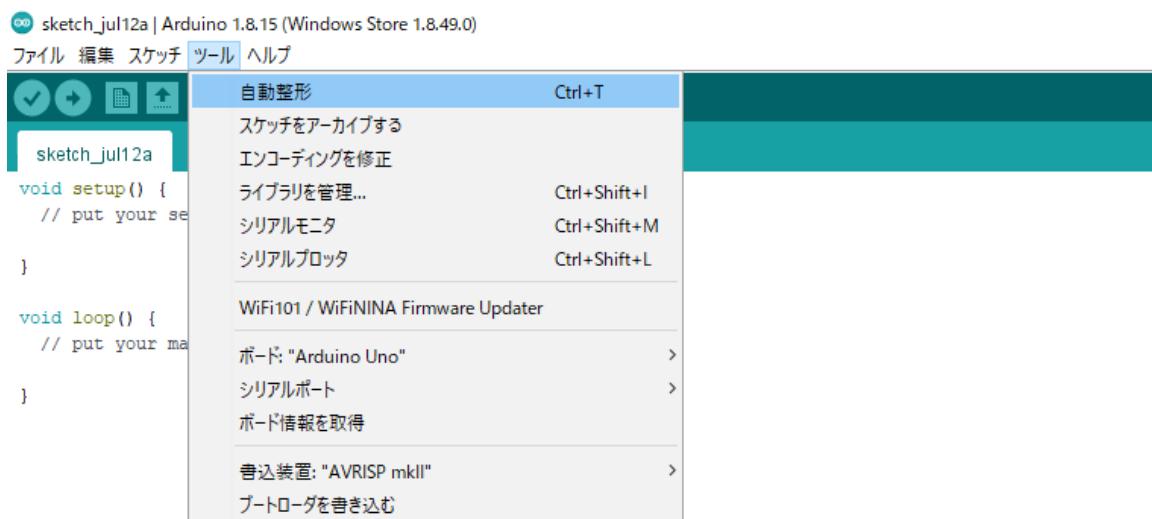


図 2.15: ライブラリの管理の選択

次に、検索窓に esp32 と入力しててきた esp32 ライブラリをインストールしてください（図 2.16）。



図 2.16: ESP32用ライブラリのインストール

2.6 Hello ESP32!!

ここで動作確認をするためにプログラミングでは定番のHello WorldをESP32でもやってみましょう。

プログラムの記述

HelloWorldを実行するため、新しくファイルを作成します。ファイル>新規ファイルを選択してください(図2.17)。

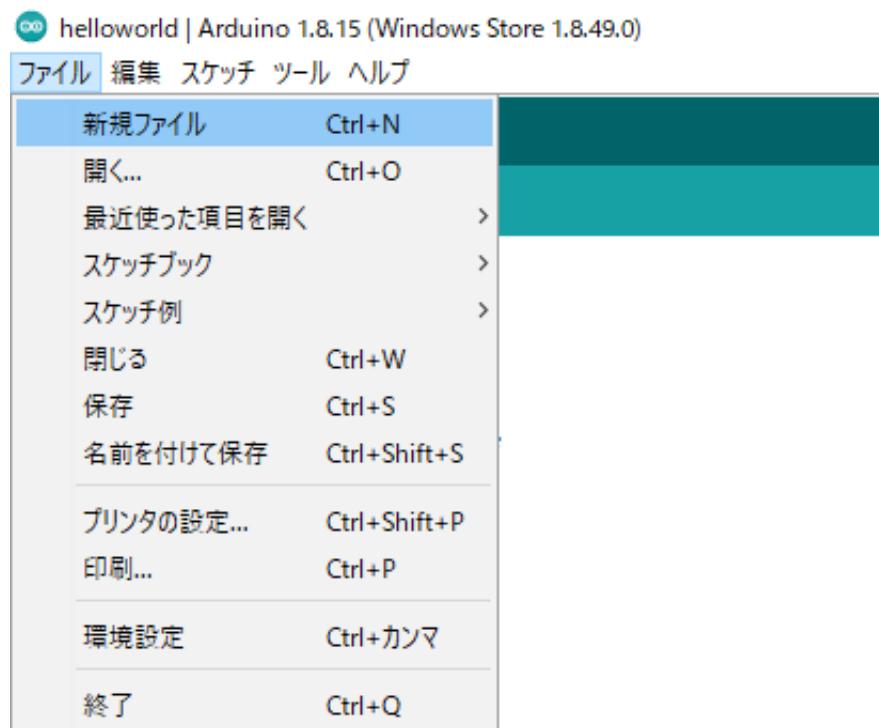


図2.17: 新規ファイルの作成

ファイルエクスプローラーが開かれるので、ファイル名にhelloworldと入力して保存してください(図2.18)。

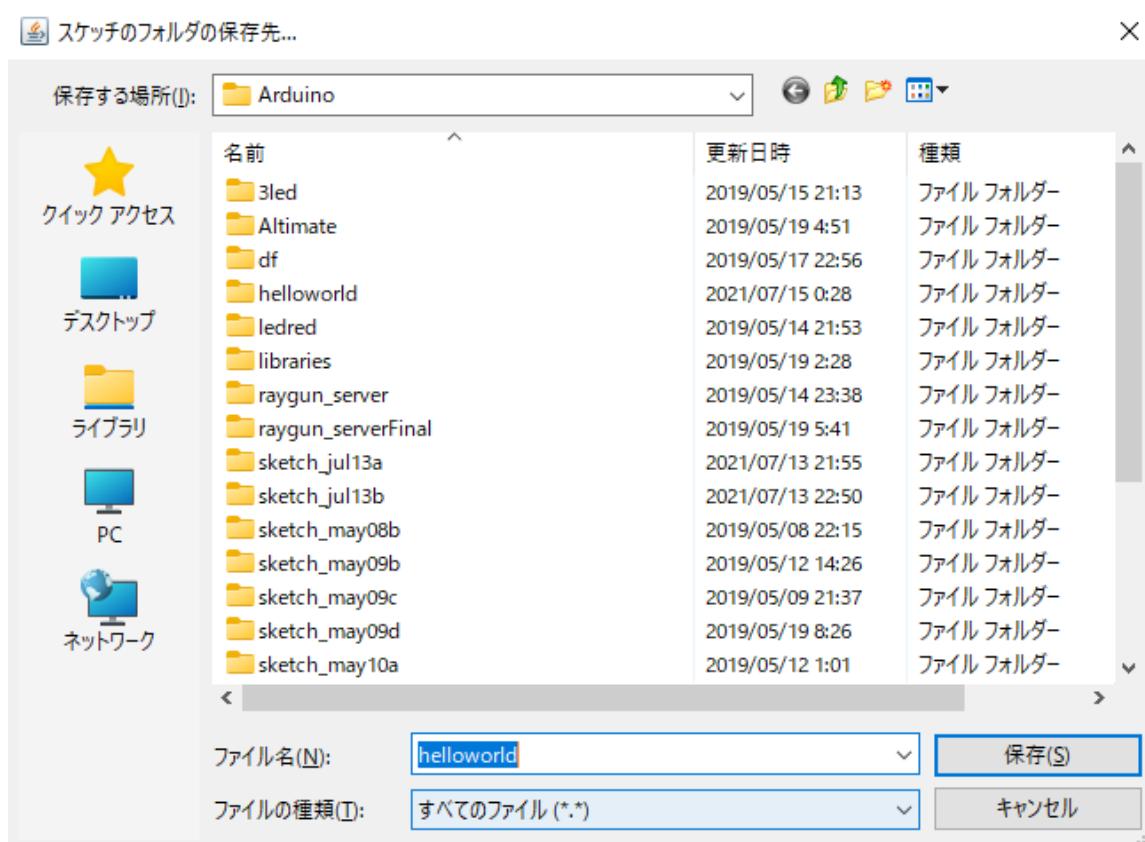


図 2.18: 新規ファイルの名前決定

完了すると図 2.19 のような画面が開きます。

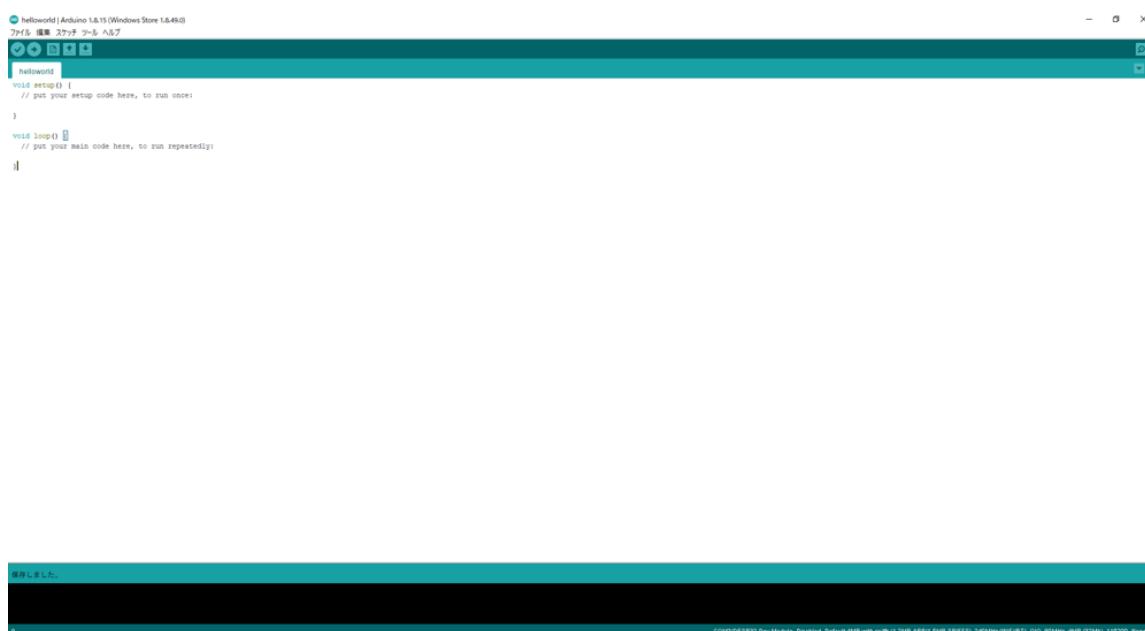


図 2.19: 新規ファイル作成完了画面

つぎに、リスト 2.2 を参考にして図 2.20 のようにプログラムを記述してください。

リスト 2.2: HelloWorld

```
void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("Hello,World");
    delay(3000);
}
```



図 2.20: helloworld のプログラムを記述

プログラムの説明

ここで、先ほど記述したプログラムの説明をします。まず、ESP32 のプログラムは大枠として* `setup()` * `loop()`

- Serial.begin(115200);
- Serial.println("Hello,World");
- delay(3000);

ブレッドボード

これからの動作確認のために ESP32 をブレッドボードにさしましょう。そこで、まずブレッドボードの説明をします。

ブレッドボードとは、電子回路のプロトタイプを組む際によく使われます。ブレッドボードにさした部品は再利用できるため、いろいろな回路を試すことができます。ブレッドボードの特徴図 2.21 のように、回路的につながっている部分とつながっていない部分に分かれているところがあります。最初は、このつながっている部分を忘れて、ショートしてしまう回路を作ってしまうことがあるので、注意してください。

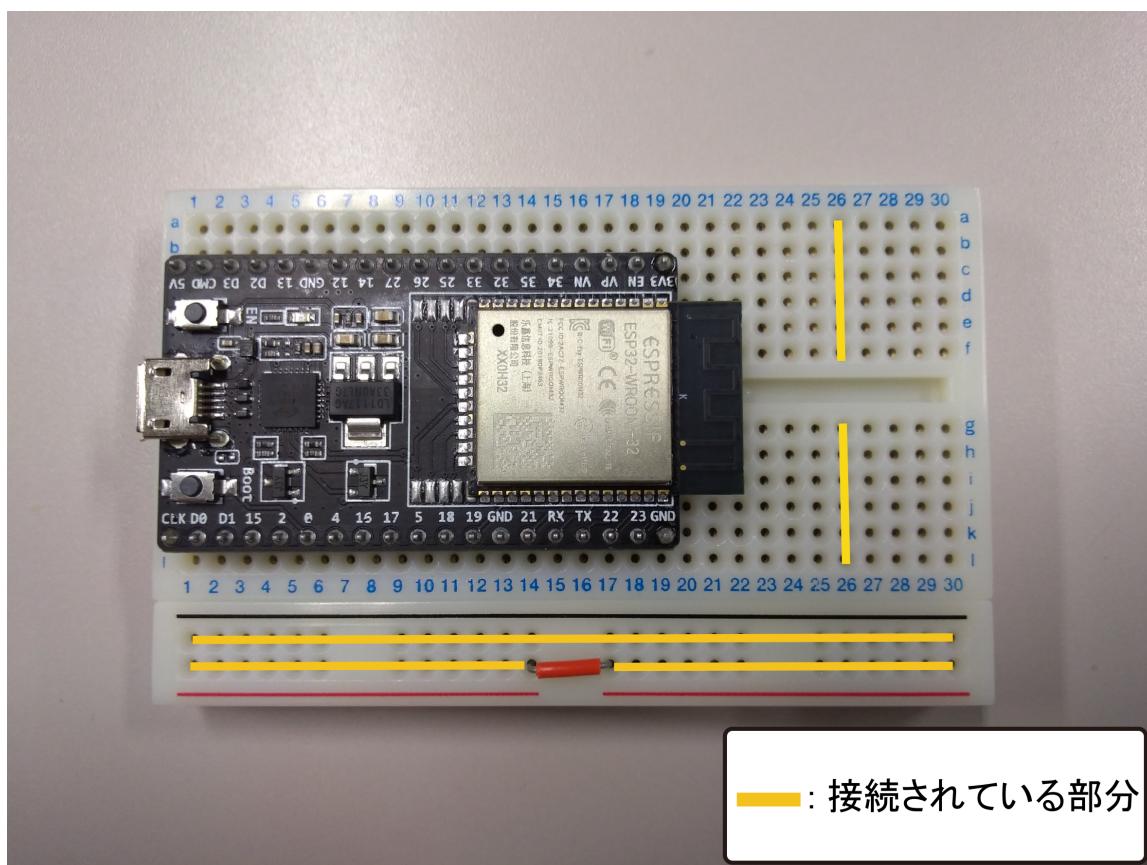


図 2.21: ブレッドボード

PCとの接続

つぎに、ESP32をPCと接続します。まず画像のよう microUSB Type-B をESP32とPCの間に接続してください。デバイスマネージャーをつかって(図2.22)、ESP32がつながっているポート番号を調べます。

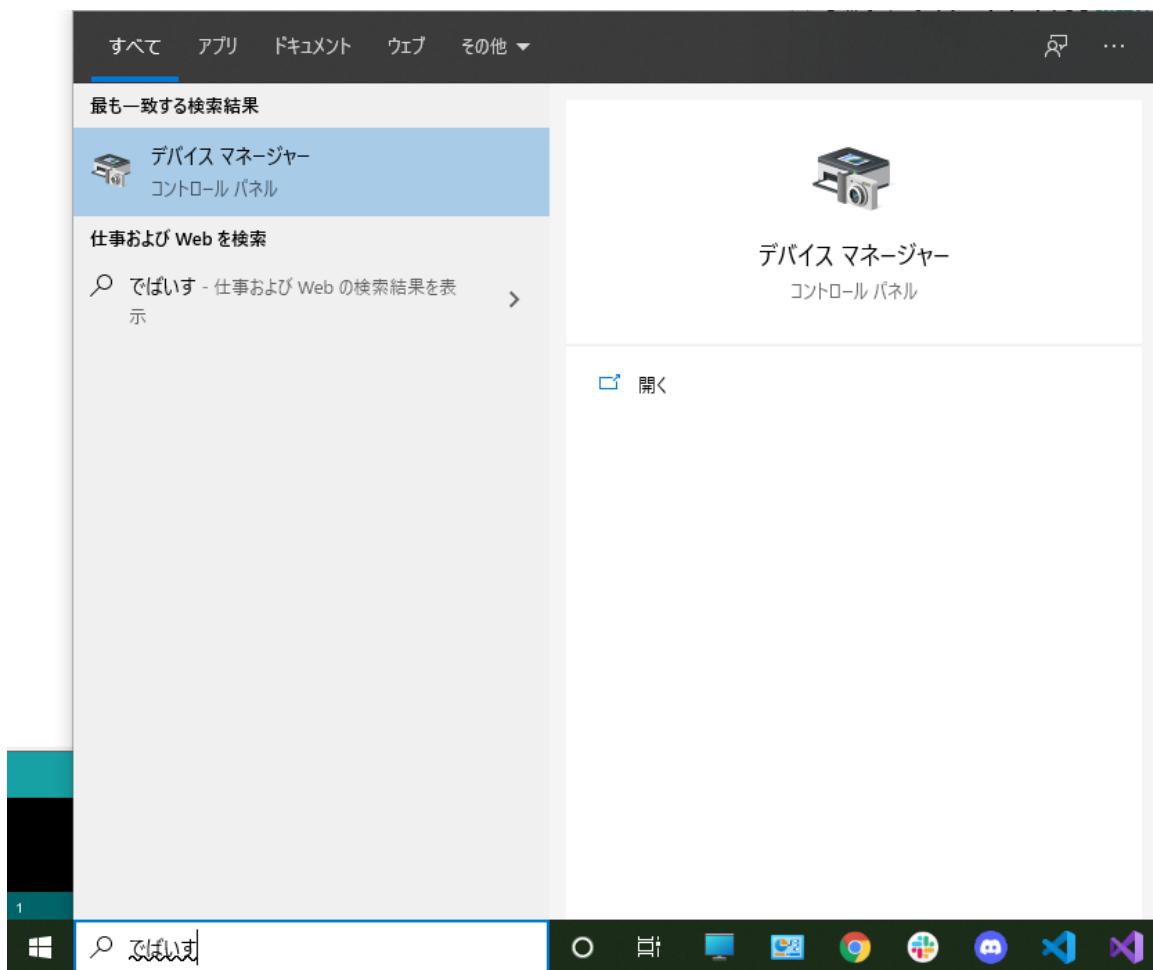


図2.22: デバイスマネージャーの検索

ESP32はという名前であり COM3 につながっていることがわかります(図2.23)。

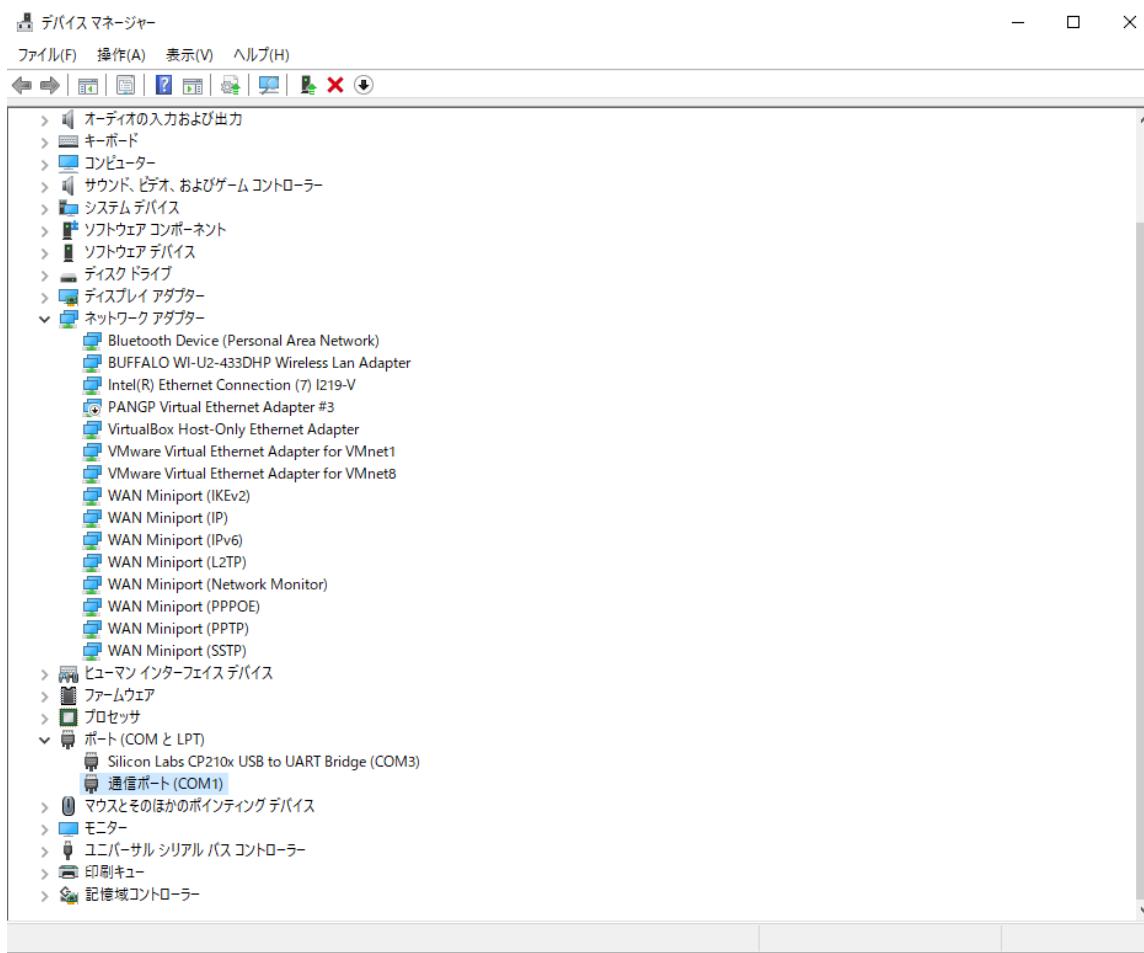


図 2.23: ESP32 の接続ポートを調べる

設定の確認

プログラムの書き込み

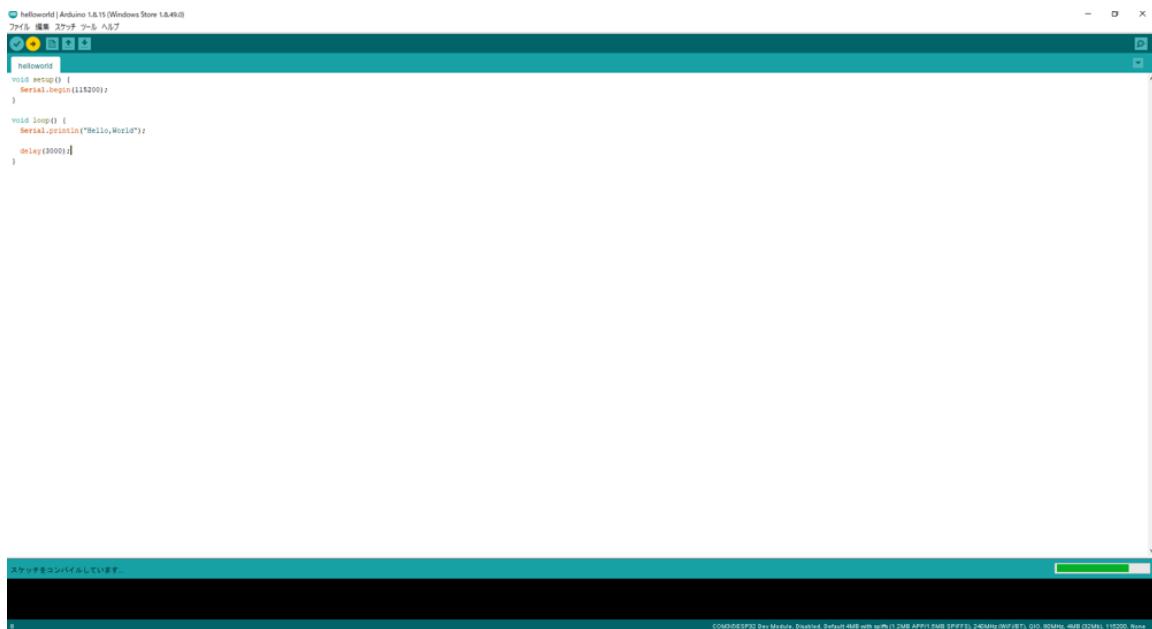


図 2.24: ESP32 にプログラムを書き込む

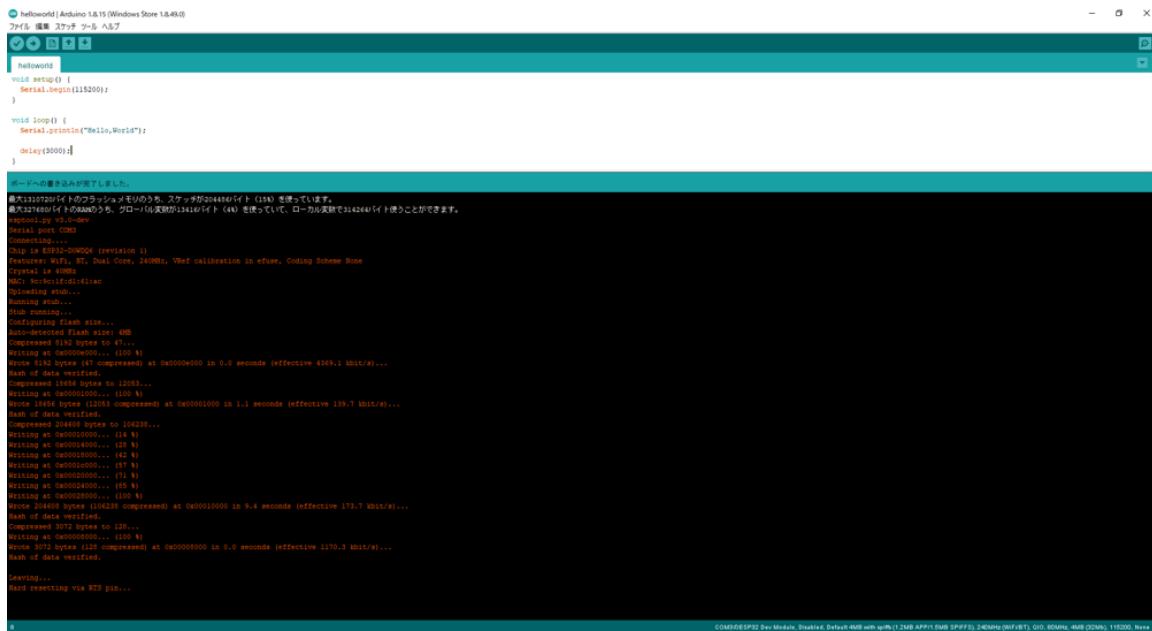


図 2.25: コンソール画面

動作確認

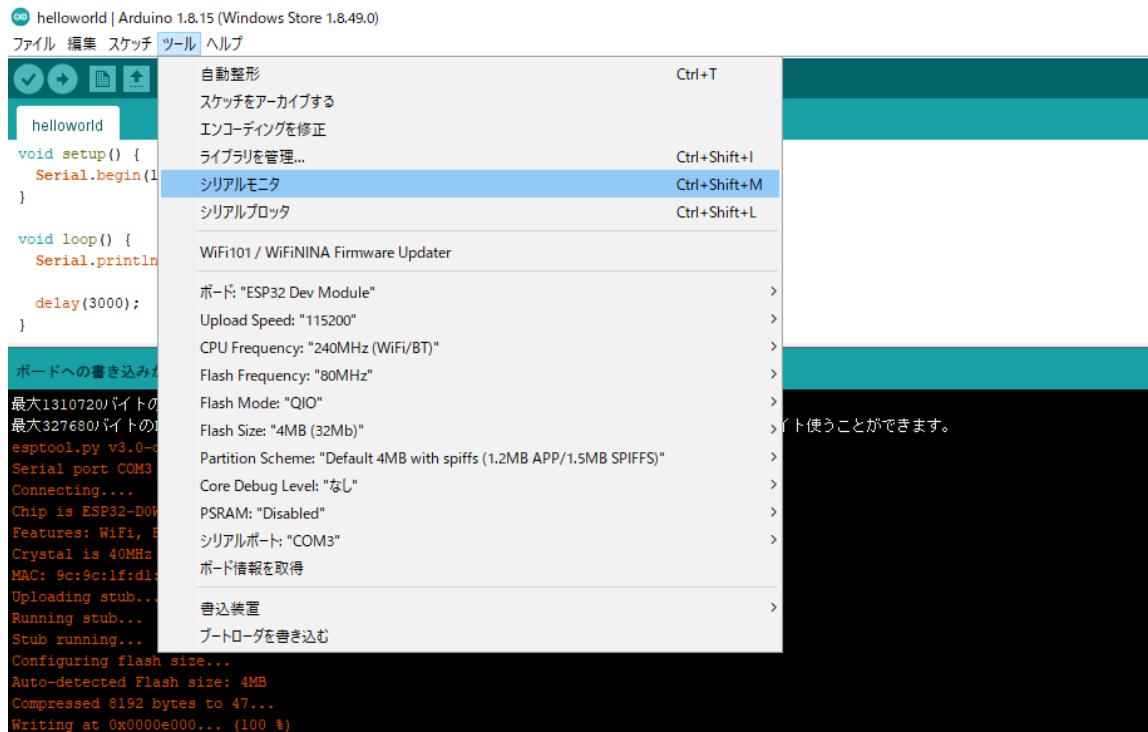


図 2.26: シリアルモニタの選択

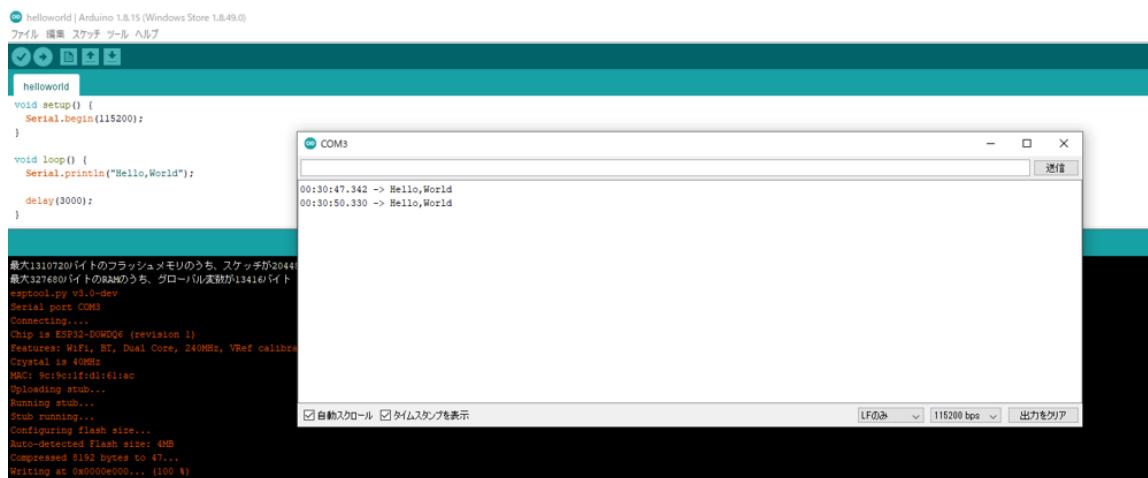


図 2.27: helloworld の表示成功

コラム: シリアル通信とは

ArduinoIDE はシリアルモニタを備えていて、Arduino とコミュニケーションする

ことができます。 ==[/column]

第3章

電子部品を使ってみよう

3.1 部品説明

ここでは、サンプルを作成するに際に使用する電子部品の説明を行います。

LED

- アノード

極性は端子の長いほうをアノードと呼び電源の+に接続する

- カソード

端子の短いほうをカソードと呼ぶカソード側は中の金属板が大きい * 点灯のために必要な情報** 順電圧 (vf) ** 順電流 (lf) ラズパイで利用する場合は順電圧が 2V 程度、順電流が 20mA 程度

ジャンプワイヤ

オススメ

抵抗

抵抗見分け方

タクトスイッチ

- プルアップとプルダウン

スイッチを利用すれば2つの値を切り替えられる回路を作れます。しかし、スイッチがオフの場合では、出力する端子が解放状態（何も接続されてない状態）になるこの場合周囲の雑音を拾ってしまい、値が安定しない状態になるそこで、プルダウンやプルアップを使って安定させる方法としてはGNDやVdd（電源）に接続しておく方法こうしておくことでスイッチがオフ状態のとき、出力端子に接続されている抵抗を介して値を安定させるスイッチOFF時に0Vに安定させる方法をプルダウン電圧がかかった状態に安定させる方法をプルアップと呼ぶ

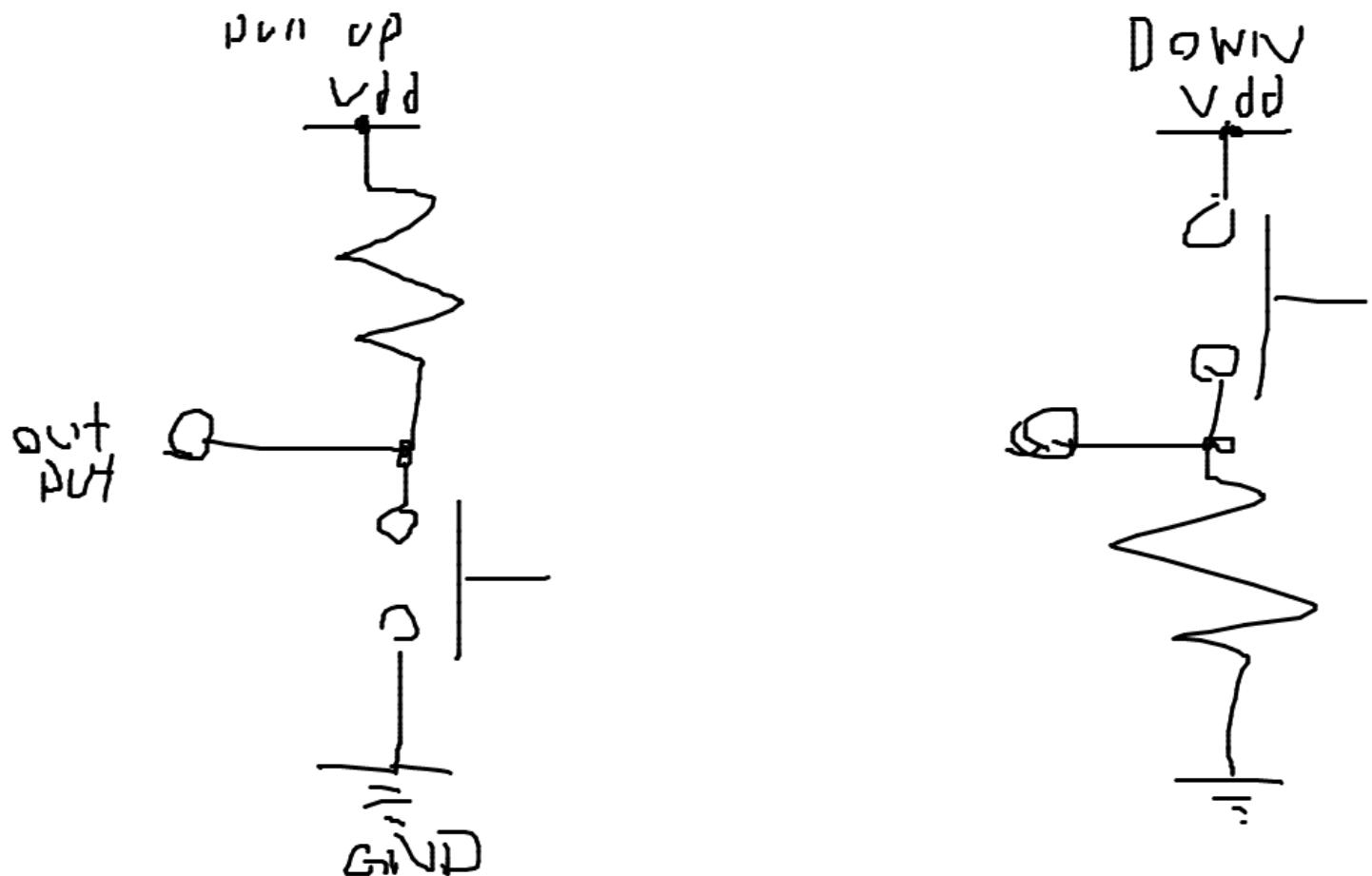


図 3.1: pullupdown

3.2 Lチカしよう！

Lチカとは、ハードにおける、プログラムのHelloWorldです。LEDをチカチカさせるだけですが、実際にLEDを光らせることができるとわくわくします。

プログラムでLチカ

Lチカですが、ESP32を使用することで、容易に実現できます。ArduinoIDEから新規作成を選択し、新たなファイルを作成して以下のプログラムを貼り付けてください。

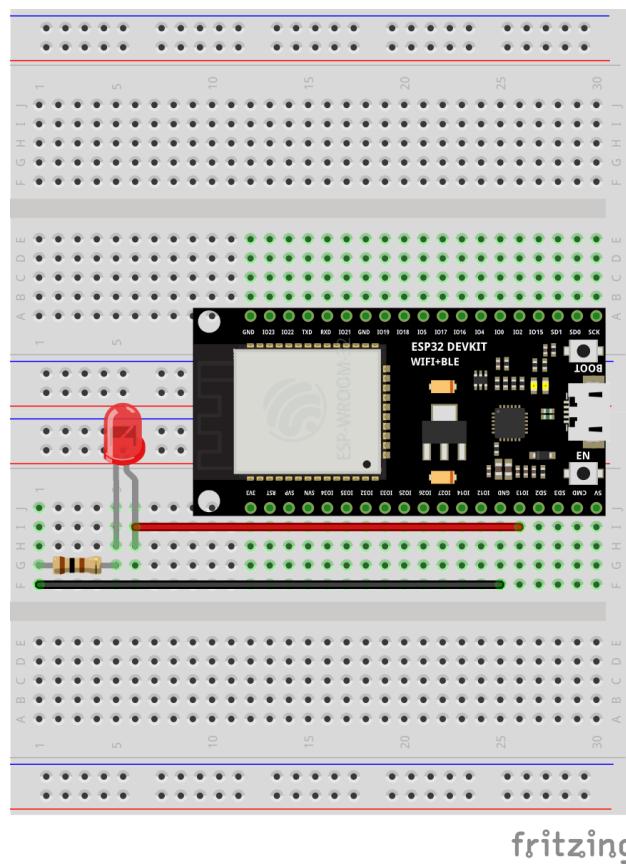


図 3.2: led2

リスト 3.1: Ltic

```
void setup() {  
    pinMode(13, OUTPUT);  
}  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000);  
}
```

```
void loop() {  
    digitalWrite(13, HIGH);  
    delay(100);  
    digitalWrite(13, LOW);  
    delay(100);  
}
```

タクトスイッチでLチカ

せっかくなので、スイッチを使用して、LED を光らせましょう同様に以下の画像を参考に電子回路を組み、プログラムを参考にして、ESP32 に書き込んでください。

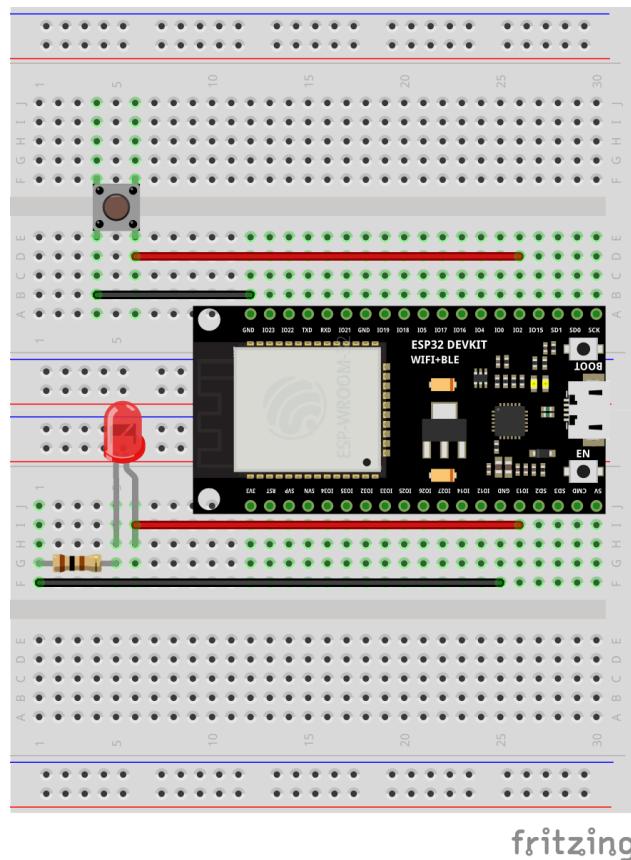


図 3.3: switch2

リスト 3.2: switch

```
void setup()
{
    Serial.begin(115200);
    pinMode(2, INPUT_PULLUP);
    pinMode(13, OUTPUT);
}

void loop()
{
    if (digitalRead(2) == LOW)
    {
        delay(100);
        digitalWrite(13, HIGH);
        delay(100);
        Serial.println("ON!");
    }
    if (digitalRead(2) == HIGH)
    {
        delay(100);
        digitalWrite(13, LOW);
        delay(100);
        Serial.println("OFF!");
    }
}
```

コラム: コラム: チャタリング

スイッチは金属板の接触によって、電流を通したり、通さなかったりしますが、これを行う際、複数回のオンオフが発生してしまいます。この対策としては、プログラム側で、delay をはさむことが挙げられます。==[/column]

3.3 応用問題: 状態遷移

二つの LED とスイッチを使用して、二つの LED の状態を以下のように変更してください

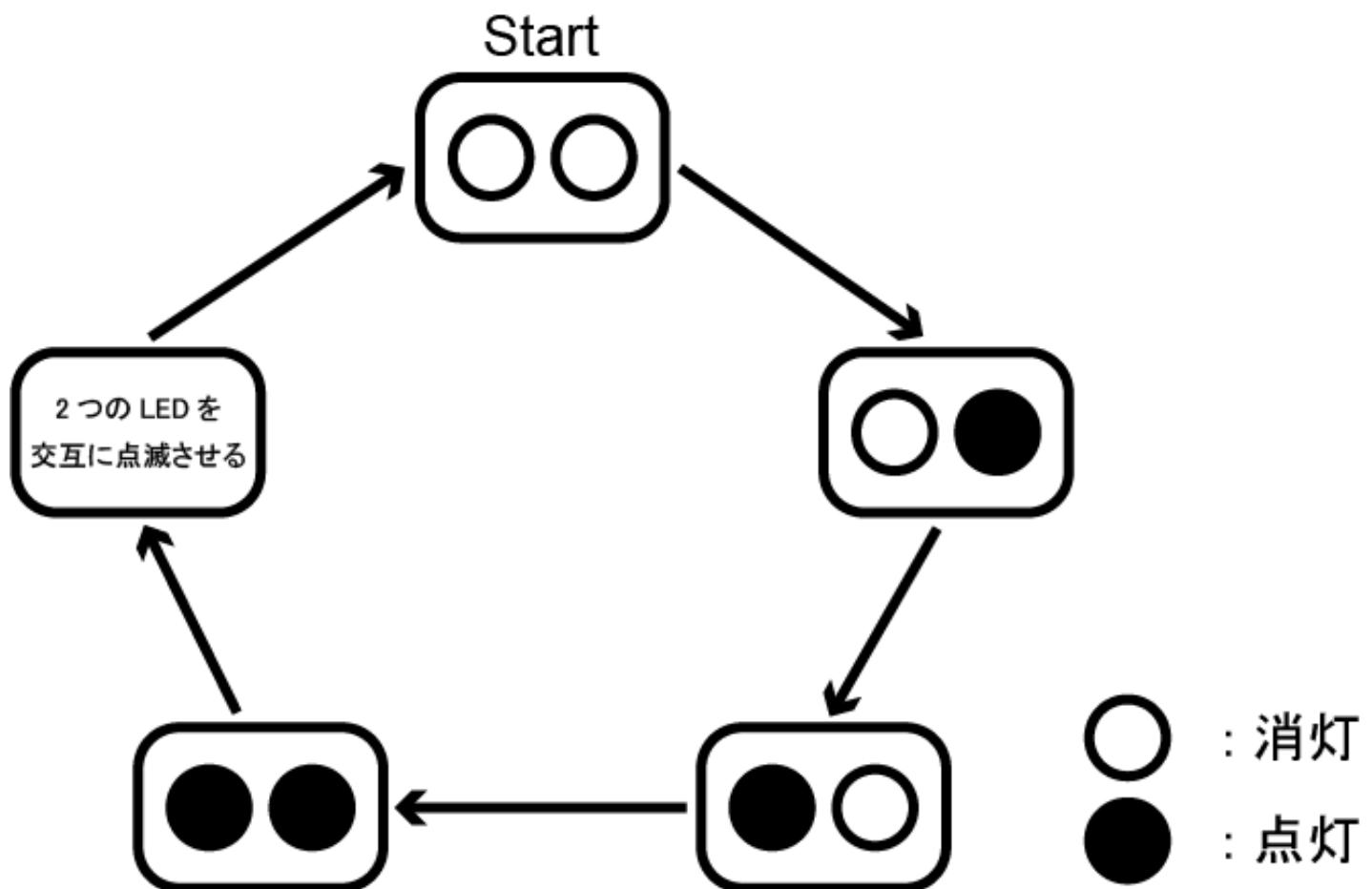


図 3.4: ouyou

2019 年度組み込み制作講座より引用

第4章

センサーのデータを Web 上に公開しよう



図 4.1: 今回の目標画面

4.1 センサーを使おう

I2C とは

4 本の線を接続するだけでセンサーヤや表示デバイスを手軽に利用できる I2C (Inter Integrated Circuit) IC 間で通信することを目的に、フィリップ社が開発したシリアル通信方式データのやり取りをする SDA (シリアルデータ) と、IC 間でタイミングを合わせるために利用する SCL (シリアルクロック) 2 本の線をつなげることで、互いにデータのやり取りえをするようになっている GND と電源にもつなげるので 4 本必要 I2C は各種デバイス

を制御するマスターと、マスターからの命令によって制御されるスレーブに分かれるマスターはマイコンに当たるデバイスを制御する場合に、対象デバイスを指定する必要がある各I2CデバイスにはI2Cアドレスが割り当てられているアドレスは16進数表記で0x03から0x77までの117個のアドレスが利用できる大体は製品出荷時にアドレスが割り当てられている

温湿度センサー

温度範囲 0~50 湿度範囲 20~90 動作電圧 3-5.5v 電流供給 0.5~2.5mA 読み取りタイミング一秒間隔毎秒センサー取得できる ArduinoIDE を使用した DHT11 / DHT22 温度および湿度センサーを備えた ESP32 <https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>

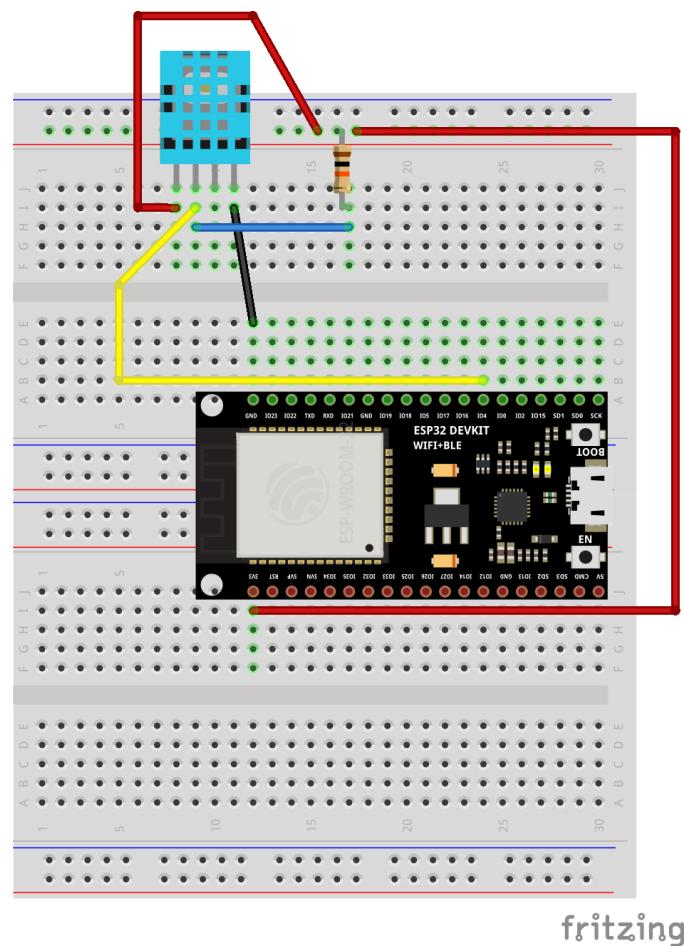


図 4.2: dht11

第4章 センサーのデータをWeb上に公開しよう 4.1 センサーを使おう

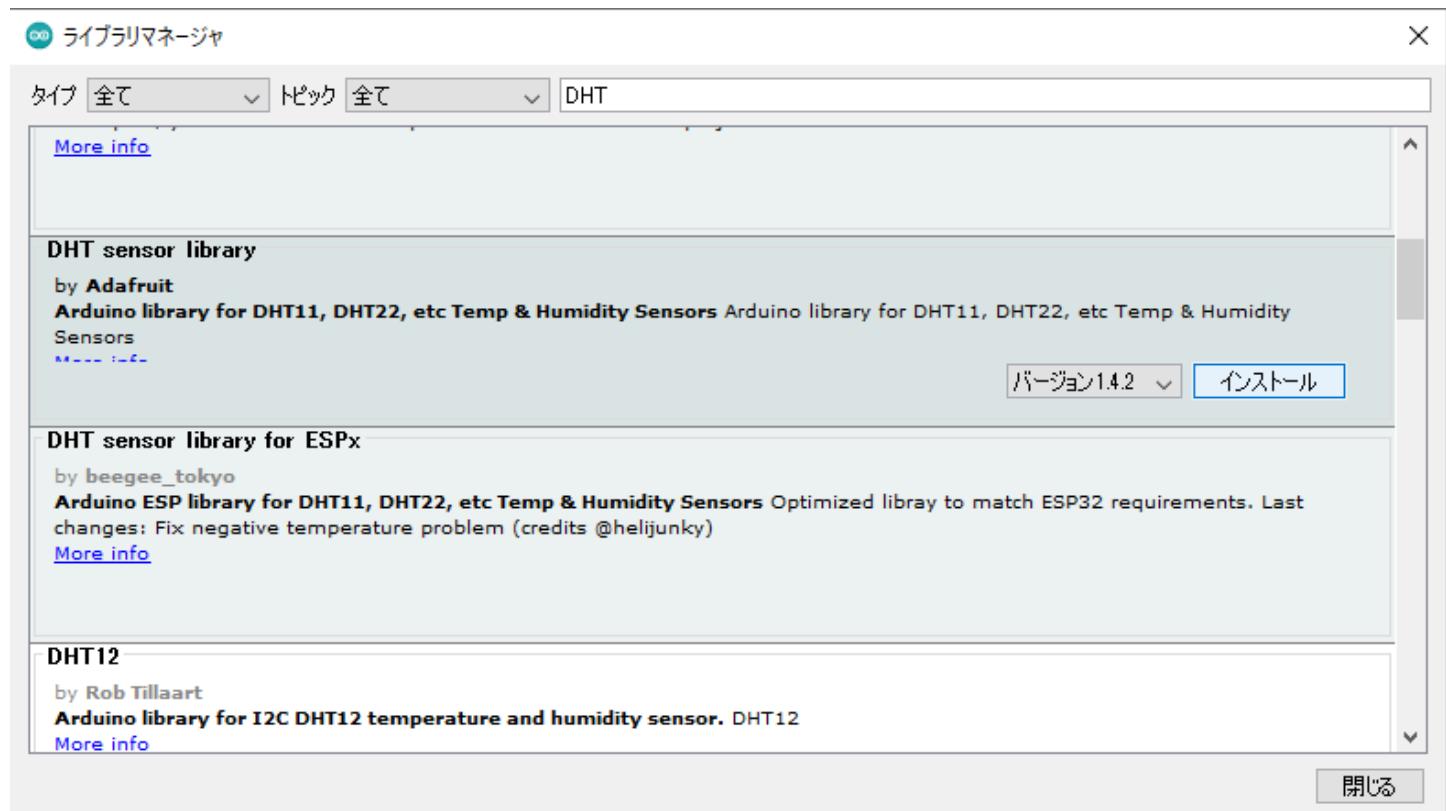


図 4.3: DHT11 用ライブラリのインストール

第4章 センサーのデータをWeb上に公開しよう 4.1 センサーを使おう

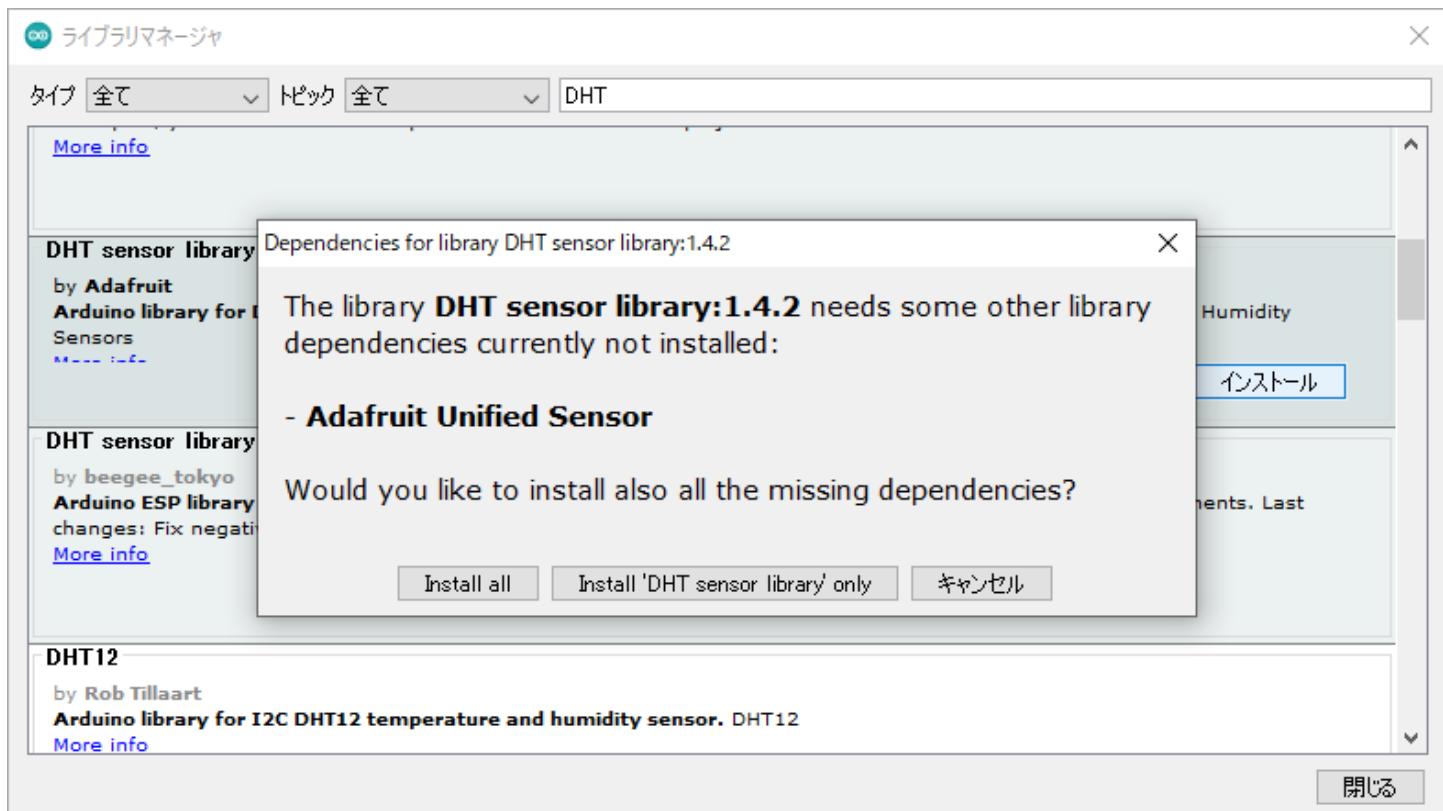


図 4.4: 依存ライブラリのインストール

LCD

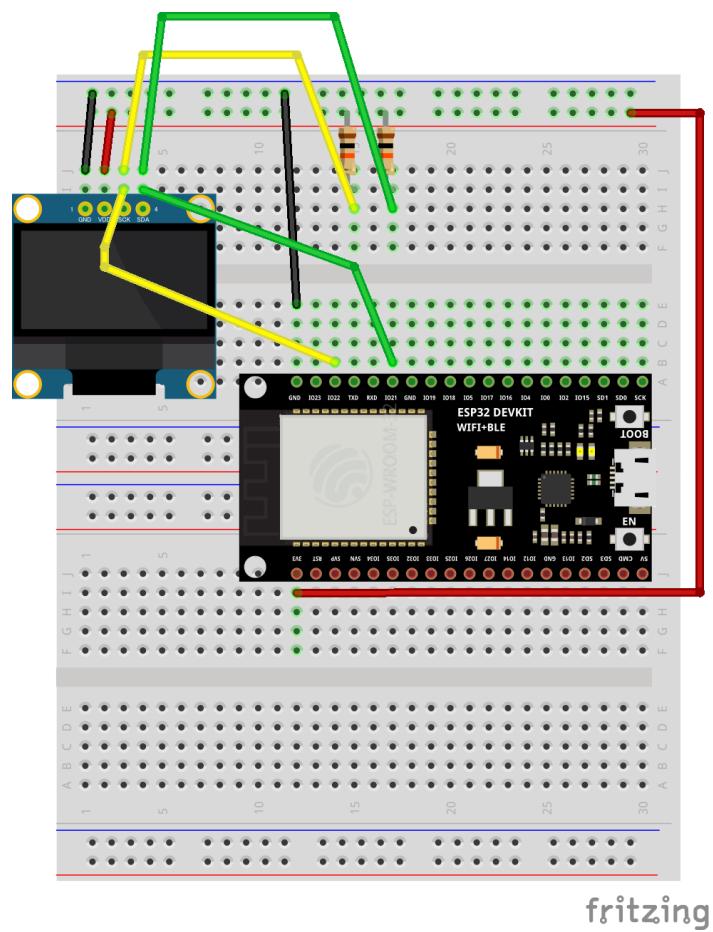


図 4.5: oled

第4章 センサーのデータをWeb上に公開しよう 4.1 センサーを使おう

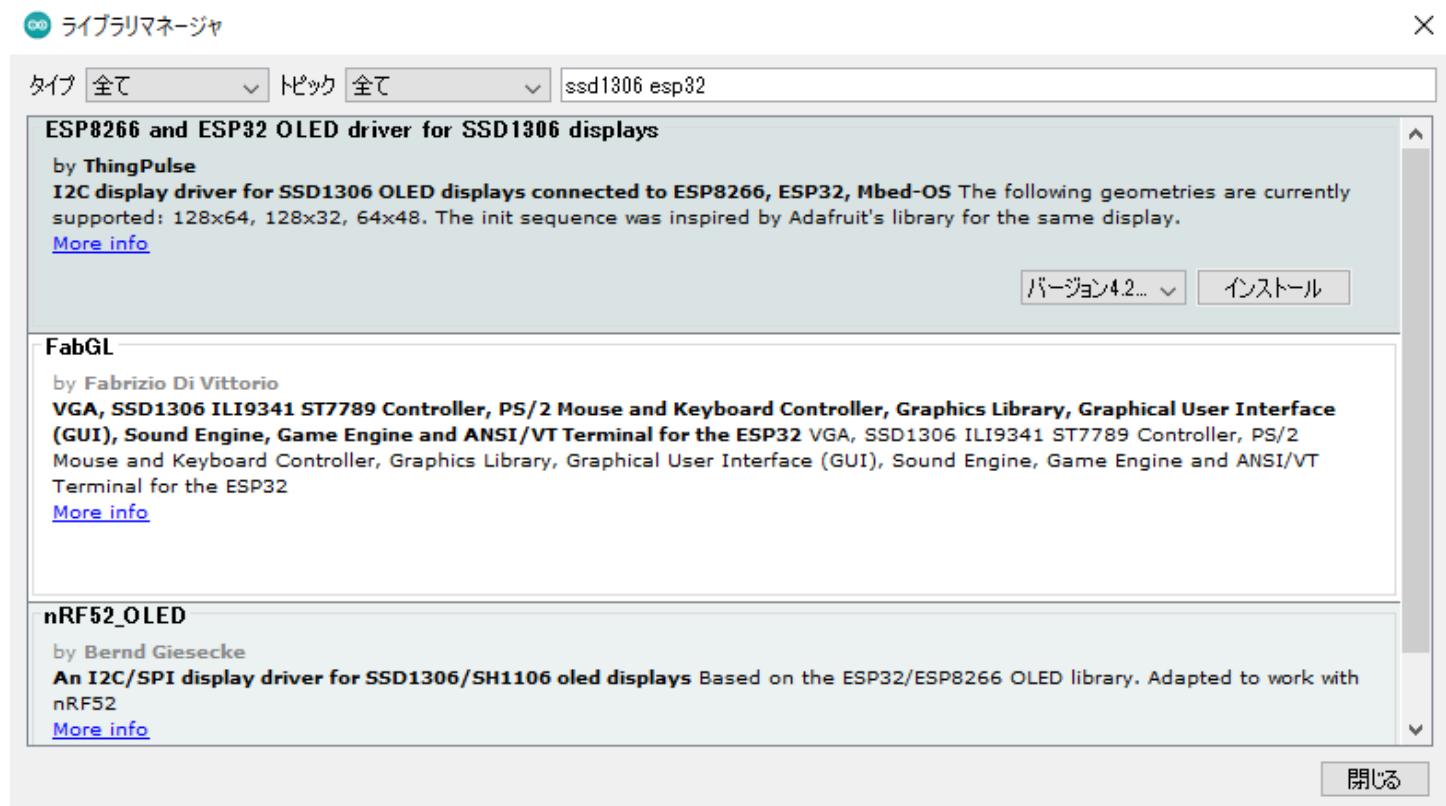


図 4.6: SSD1306 用ライブラリのインストール

リスト 4.1: oled

```
#include <Wire.h>
#include "SSD1306.h" //ディスプレイ用ライブラリを読み込み
using namespace std;

SSD1306 display(0x3c, 21, 22); //SSD1306インスタンスの作成 (I2Cアドレス, SDA, SCL)

void setup()
{
    display.init(); //ディスプレイを初期化
    display.setFont(ArialMT_Plain_16); //フォントを設定
    display.drawString(0, 0, "Hello World"); // (0,0) の位置に Hello World を表示
    display.display(); // 指定された情報を描画
}

void loop(){}
}
```

リスト 4.2: dht11

```
// Example testing sketch for various DHT humidity/temperature sensors written by ladyada
// REQUIRES the following Arduino libraries:
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor

#include "DHT.h"

#define DHTPIN 4      // Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// #define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(9600);
    Serial.println(F("DHTxx test!"));

    dht.begin();
}

void loop() {
    // Wait a few seconds between measurements.
    delay(2000);

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    float h = dht.readHumidity();
```

```
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
}

// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahrenheit = false)
float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity: "));
Serial.print(h);
Serial.print(F("% Temperature: "));
Serial.print(t);
Serial.print(F(" °C"));
Serial.print(f);
Serial.print(F(" °F Heat index: "));
Serial.print(hic);
Serial.print(F(" °C"));
Serial.print(hif);
Serial.println(F(" °F"));
}
```

リスト 4.3: tmp

```
Humidity: 56.00% Temperature: 24.50 °C 76.10 °F Heat index: 24.47 °
C 76.04 °F
Humidity: 56.00% Temperature: 24.50 °C 76.10 °F Heat index: 24.47 °C 76.04 °F
Humidity: 56.00% Temperature: 24.50 °C 76.10 °F Heat index: 24.47 °C 76.04 °F
```

4.2 Webに公開しよう

Wi-Fiと接続する

SSIDとは

アクセスポイントの名前 SSID (Service Set Identifier) とは IEEE802.11 (Wi-Fi 無線 LAN の通信規格) で定められているアクセスポイントの識別子のこと混線を避けるため名付けられている長さは最大 32 文字

ambientについて

Ambient は IoT データの可視化サービスです。 <https://ambidata.io/>

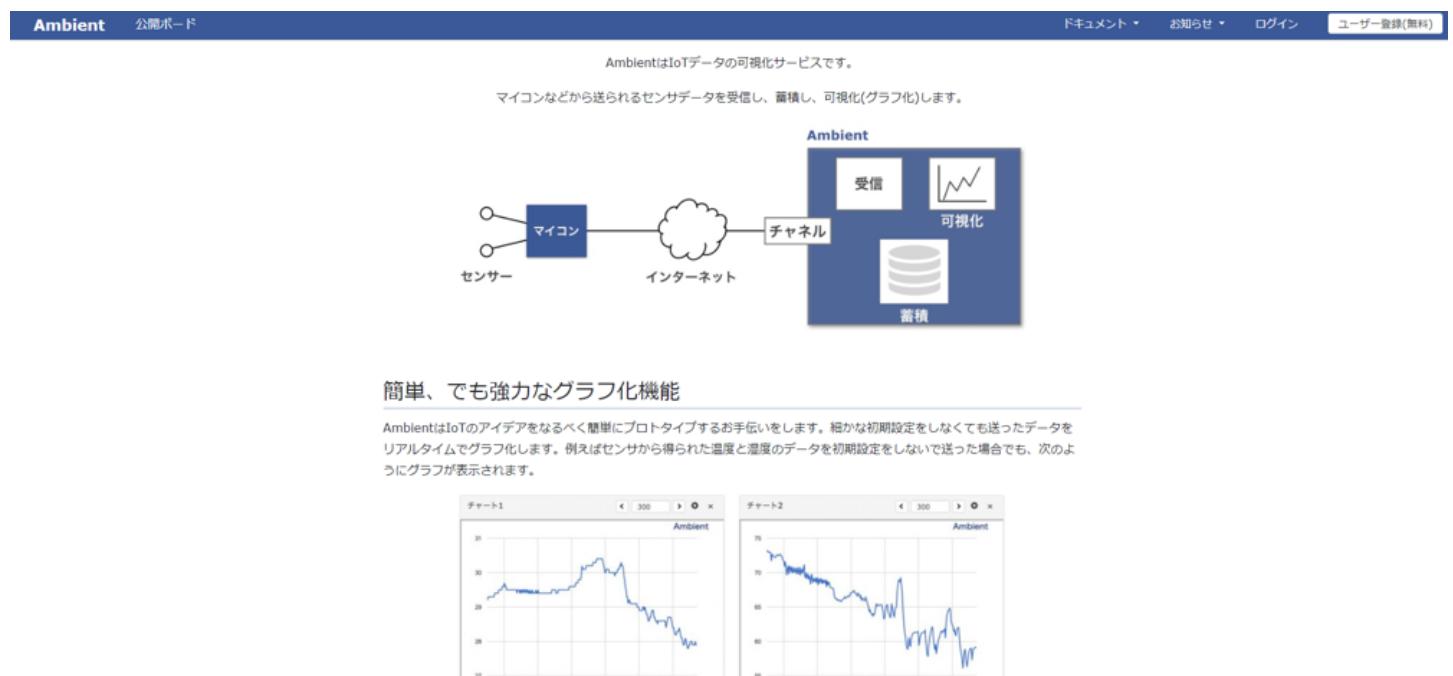


図 4.7: ambient のトップページ

第4章 センサーのデータをWeb上に公開しよう 4.2 Webに公開しよう



The screenshot shows the user registration page for Ambient. It features three input fields: 'メールアドレス' (Email Address), 'パスワード' (Password), and 'パスワード再入力' (Re-enter Password). Below these is a blue button labeled 'ユーザー登録(無料)' (Free User Registration) with the text '登録した時点で「Ambient利用規約」に書かれた内容に同意したものとします。' (By registering, you agree to the terms and conditions of the 'Ambient Usage Agreement'). At the bottom, there are links for 'AmbientData Inc.' and '利用規約 会社概要'.

図 4.8: ユーザ登録



図 4.9: 登録完了メール

第4章 センサーのデータをWeb上に公開しよう 4.2 Webに公開しよう



図 4.10: ログイン画面

チャネルを作成します。

The screenshot shows the channel creation completed screen. The header bar includes the 'Ambient' logo, a 'チャンネル一覧' button, and navigation links for 'ドキュメント', 'お知らせ', and a user profile icon with the email 'imanaka33@gmail.com'. Below the header, a message 'ユーザーキー: 2942a2a7b340998f8d' is displayed. A table lists a single channel entry: 'チャネル名' (チャネル39400), 'チャネルID' (39400), 'リードキー' (144e4e6ba54da05b), and 'ライトキー' (624168ad3f97c4b9). Action buttons 'ダウンロード', 'データ削除', and '設定' are shown next to the channel row. At the bottom left, there are buttons for 'チャネルを作る' and 'デバイスキーセット'. The footer contains links for 'AmbientData Inc.', '利用規約', and '会社概要'.

図 4.11: チャネル作成完了画面

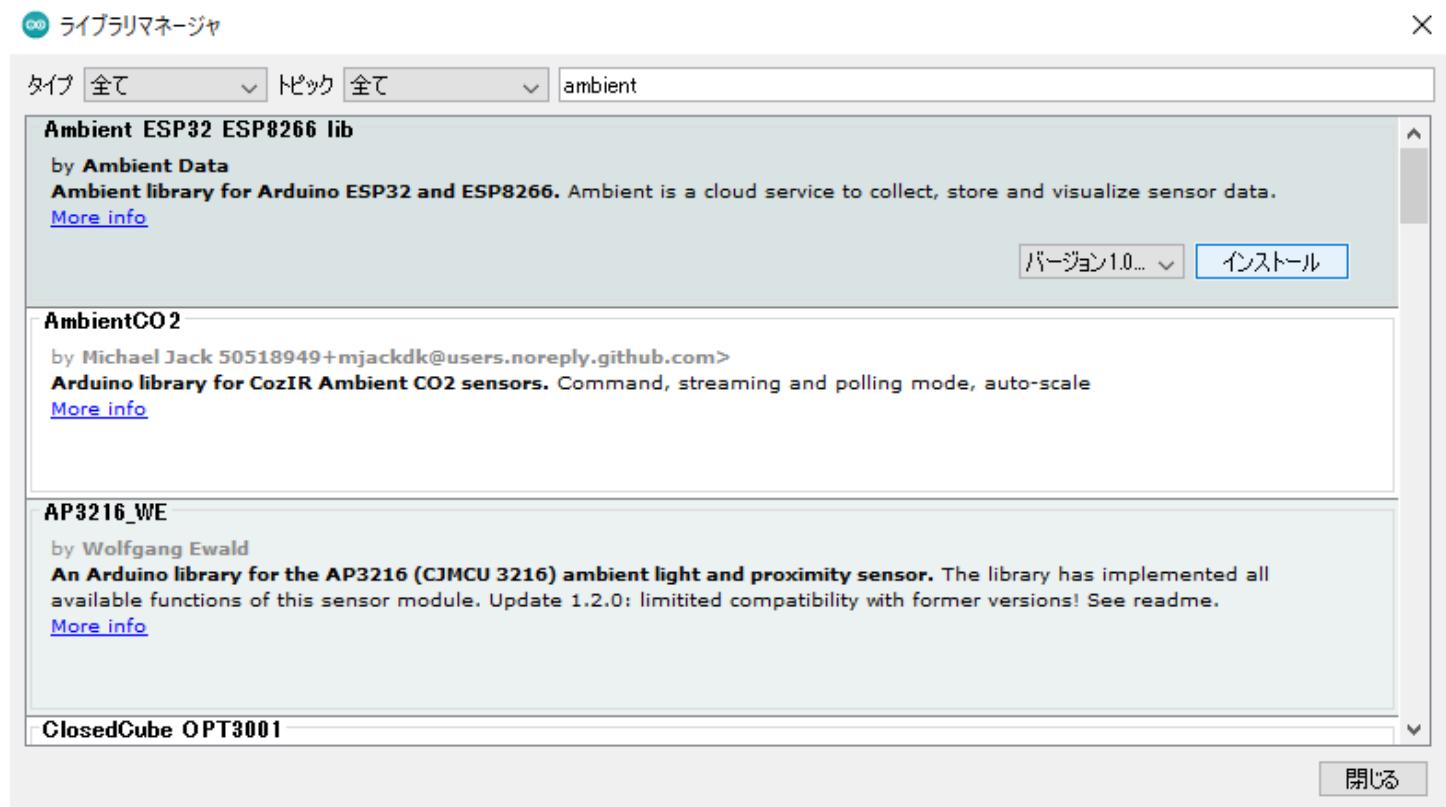


図 4.12: ambient 用ライブラリのインストール

ライブラリのインストール

回路図

コーディング

リスト 4.4: amibient

```
#include "Ambient.h"
#include <WiFi.h>

#include "DHT.h"

#define DHTPIN 4
#define PERIOD 30
WiFiClient client;
Ambient ambient;
#define DHTTYPE DHT11 // DHT 11

unsigned long delayTime;
```

```
float temp;
float pressure;
float humid;

unsigned int channelId = 39400; // AmbientのチャネルID(数字)
const char *writeKey = "624168ad3f97c4b9"; // ライトキー

DHT dht(DHTPIN, DHTTYPE);
void setup()
{
    Serial.begin(115200);
    WiFi.begin("elecom-b2809f-g", "fapd4rpfac3u"); // Wi-Fiの初期化

    while (WiFi.status() != WL_CONNECTED)
    { // Wi-Fiアクセスポイントへの接続待ち
        delay(500);
    }
    dht.begin();
    ambient.begin(channelId, writeKey, &client); // チャネルIDとライトキーを指定して
    Ambientの初期化
}

void loop()
{
    // Wait a few seconds between measurements.
    delay(2000);

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    float h = dht.readHumidity();
    // Read temperature as Celsius (the default)
    float t = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    float f = dht.readTemperature(true);

    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t) || isnan(f))
    {
        Serial.println(F("Failed to read from DHT sensor!"));
        return;
    }

    // Compute heat index in Fahrenheit (the default)
    float hif = dht.computeHeatIndex(f, h);
```

```
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);
ambient.set(1, h);
ambient.set(2, t);
ambient.set(3, f);

ambient.send(); // Ambientにデータを送信

delay(PERIOD * 1000);
}
```

第5章

WebAPI を使おう

WebAPI とは？

5.1 Weather API を使う

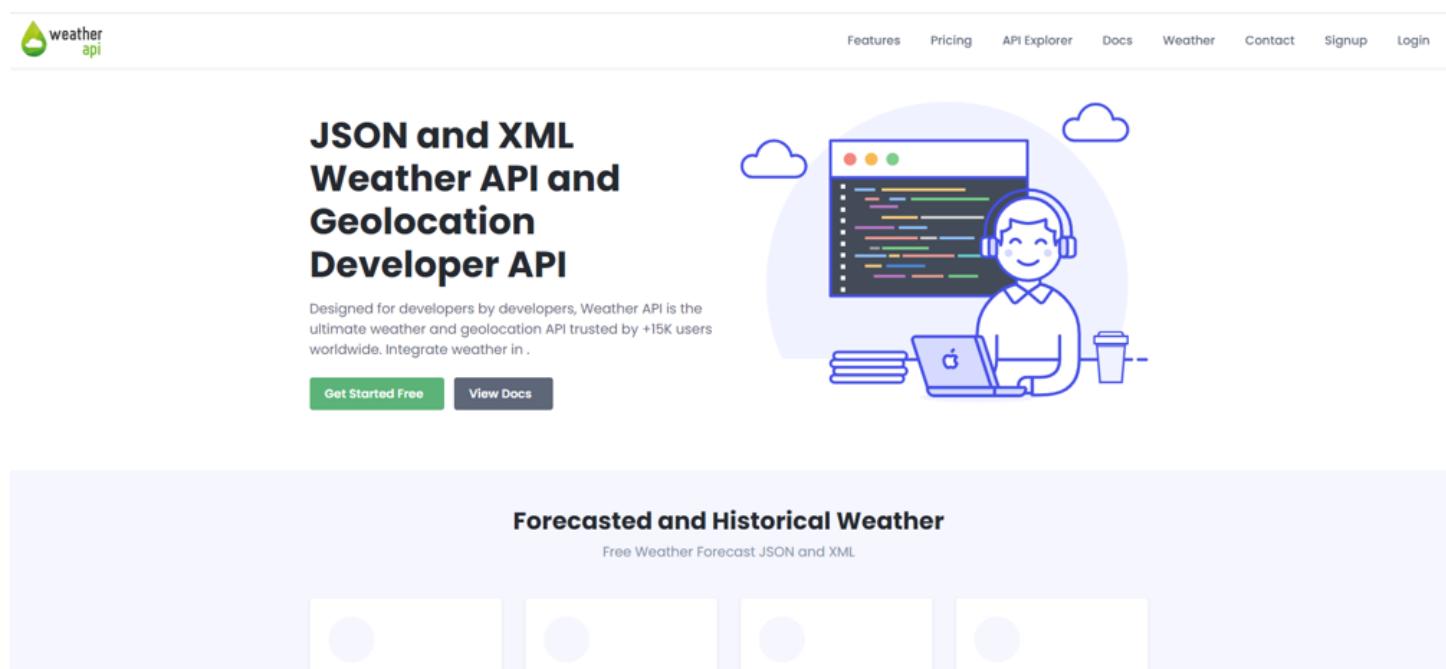


図 5.1: WeatherAPI のトップページ

The screenshot shows the 'Sign Up' page for Weather API. It includes fields for 'Email' (labeled as username), 'Password', and 'Retype Password'. Below these are CAPTCHA fields and checkboxes for accepting terms and conditions and privacy policy. A green 'Sign up' button is at the bottom.

図 5.2: 登録画面

The screenshot shows the 'Login' page for Weather API. It features fields for 'User Name' and 'Password', a 'Remember me next time' checkbox, and a green 'Log In' button. Below the login form is a promotional banner for quick and easy signups.

図 5.3: ログイン画面

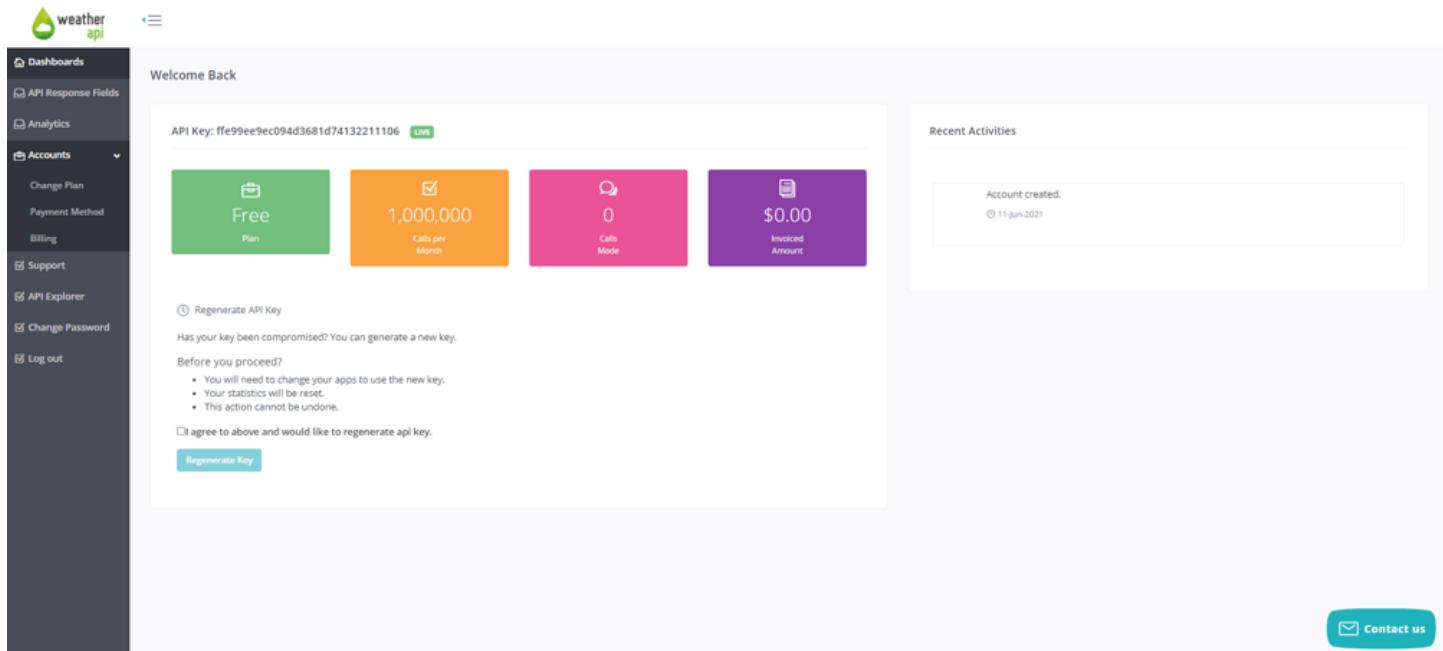


図 5.4: マイページ

The screenshot shows the 'Interactive API Explorer' page. It includes a header with links to Features, Pricing, API Explorer, Docs, Weather, Contact, and My Account. The main content area is titled 'Interactive API Explorer' and describes it as an interface for testing APIs. It shows fields for 'Your API Key' (ffe99ee9ec094d3681d7413221106), 'Protocol' (HTTPS), and 'Format' (JSON). Below these are two tables for parameters:

Parameter	Value	Type	Location	Description
q	Saitama	string	query	Pass US Zipcode, UK Postcode, Canada Postalcode, IP address, Latitude/Longitude (decimal degree) or city name. Visit request parameter section to learn more.

Parameter	Value	Type	Location	Description
aqi	no	string	query	Get air quality data

A 'Contact us' button is located at the bottom right.

図 5.5: API の試行ページ

Weather API <https://www.weatherapi.com/>

Call [https://api.weatherapi.com/v1/current.json?key=\[key\]&q=Saitama&aqi=no](https://api.weatherapi.com/v1/current.json?key=[key]&q=Saitama&aqi=no)

ResponeseCode 200

リスト 5.1: ResponesHeader

```
{  
    "Transfer-Encoding": "chunked",  
    "Connection": "keep-alive",  
    "Vary": "Accept-Encoding",  
    "CDN-PullZone": "93447",  
    "CDN-Uid": "8fa3a04a-75d9-4707-8056-b7b33c8ac7fe",  
    "CDN-RequestCountryCode": "FI",  
    "CDN-EdgeStorageId": "615",  
    "CDN-CachedAt": "2021-07-12 14:05:36",  
    "CDN-RequestPullSuccess": "True",  
    "CDN-RequestPullCode": "200",  
    "CDN-RequestId": "a45be49d32c7a76559a3f3920d337f53",  
    "CDN-Cache": "MISS",  
    "Cache-Control": "public, max-age=180",  
    "Content-Type": "application/json",  
    "Date": "Mon, 12 Jul 2021 12:05:36 GMT",  
    "Server": "BunnyCDN-FI1-615"  
}
```

リスト 5.2: ResponseBody

```
{  
    "location": {  
        "name": "Saitama",  
        "region": "Saitama",  
        "country": "Japan",  
        "lat": 35.91,  
        "lon": 139.66,  
        "tz_id": "Asia/Tokyo",  
        "localtime_epoch": 1626091536,  
        "localtime": "2021-07-12 21:05"  
    },  
    "current": {  
        "last_updated_epoch": 1626087600,  
        "last_updated": "2021-07-12 20:00",  
        "temp_c": 29.4,  
        "temp_f": 84.9,  
        "humidity": 55,  
        "pressure": 1013,  
        "wind_gust_kph": 12.0,  
        "wind_kph": 10.0,  
        "wind_dir": "ESE",  
        "clouds": 20,  
        "uv": 6.0  
    }  
}
```

```
"is_day": 0,  
"condition": {  
    "text": "Partly cloudy",  
    "icon": "//cdn.weatherapi.com/weather/64x64/night/116.png",  
    "code": 1003  
},  
"wind_mph": 7.6,  
"wind_kph": 12.2,  
"wind_degree": 162,  
"wind_dir": "SSE",  
"pressure_mb": 1010.0,  
"pressure_in": 30.3,  
"precip_mm": 0.0,  
"precip_in": 0.0,  
"humidity": 61,  
"cloud": 47,  
"feelslike_c": 32.1,  
"feelslike_f": 89.8,  
"vis_km": 10.0,  
"vis_miles": 6.0,  
"uv": 7.0,  
"gust_mph": 9.2,  
"gust_kph": 14.8  
}  
}
```

この形式を JSON(JavaScript Object Node)と言います。これらの形式が、WeatherAPI から帰ってくるため、ESP32 側で使えるようにしなければなりませんそこで、公開されているライブラリである `arduinoJSON` を利用します。

ライブラリのインストール

[** `arduinoJSON`] JSON ドキュメントを作る時にキャパシティを計算する必要がある
`ArduinoJson Assistant` <https://arduinojson.org/v6/assistant/>

The screenshot shows the ArduinoJson Assistant homepage. At the top, there's a navigation bar with links for Documentation, Assistant, Troubleshooter, Book, News, and a search bar. Below the navigation is a teal header bar with the title "ArduinoJson Assistant". A sub-header below it states: "The ArduinoJson Assistant is an online tool that computes the required `JsonDocument` capacity for a given document and generates a sample program." Below this are four numbered steps: 1 Configuration, 2 JSON, 3 Size, and 4 Program. Step 1 is currently active, titled "Step 1: Configuration". It contains dropdown menus for Processor (ESP32), Mode (Deserialize), and Input type (String). A note below says "This is most likely a bad choice, prefer `char*` or `stream`". A green info box at the bottom left says "This is the Assistant for ArduinoJson 6.18.1. Make sure the same version is installed on your computer." A "Next: JSON" button is at the bottom right. The footer of the page includes links for About, Contact, and Privacy, along with a newsletter sign-up form.

図 5.6: ArudinoAssistant のトップページ

This screenshot shows the same ArduinoJson Assistant homepage as the previous one, but the second step is now active. The title "Step 2: JSON" is at the top of the main content area. Below it, there's a section titled "Input" containing a large code block representing a JSON object. The code is as follows:

```

{
  "location": {
    "name": "Saitama",
    "region": "Saitama",
    "country": "Japan",
    "lat": 35.91,
    "lon": 139.66,
    "tz_id": "Asia/Tokyo",
    "localtime_epoch": 1626533912,
    "localtime": "2021-07-17 23:58"
  },
  "current": {
    "last_updated_epoch": 1626533100,
    "last_updated": "2021-07-17 23:45",
    "temp_c": 23.3,
    "temp_f": 73.9,
    "is_day": 0,
    "condition": {
      "text": "Clear",
      "icon": "https://cdn.weatherapi.com/weather/64x64/night/113.png",
      "code": 1000
    },
    "wind_mph": 3.8,
    "wind_kph": 6.1,
    "wind_degree": 250,
    "wind_dir": "WSW",
    "pressure_mb": 1019.0,
    "pressure_in": 30.6,
    "precip_mm": 0.0,
    "precip_in": 0.0,
    "humidity": 88,
    "cloud": 0,
    "feelslike_c": 24.9,
    "feelslike_f": 76.9,
    "vis_km": 16.0,
    "vis_miles": 9.0,
    "uv": 1.0,
    "gust_mph": 13.9,
    "gust_kph": 22.3
  }
}

```

Below the input field, it says "Input length: 660" and has a "Prettify" link. At the bottom are "Previous" and "Next: Size" buttons. The footer is identical to the previous screenshot.

図 5.7: JSON の大きさ設定

The screenshot shows the ArduinoJson Assistant interface. At the top, there are links for Documentation, Assistant, Troubleshooter, Book, News, and a Search bar. Below that is a teal header with the title "ArduinoJson Assistant". A sub-header states: "The ArduinoJson Assistant is an online tool that computes the required `JsonDocument` capacity for a given document and generates a sample program." Below the header, there are four numbered steps: 1 Configuration, 2 JSON, 3 Size (which is currently selected), and 4 Program.

Step 3: Size

Data structures	576	Bytes needed to stores the JSON objects and arrays in memory ?
Strings	441	Bytes needed to stores the strings in memory ?
Total (minimum)	1017	Minimum capacity for the <code>JsonDocument</code> .
Total (recommended)	1536	Including some slack in case the strings change, and rounded to a power of two

[Tweaks \(advanced users only\)](#)

[Previous](#) [Next: Program](#)

At the bottom of the main content area, there is footer information: "ArduinoJson is a JSON library for embedded C++. Simple, efficient, and versatile. Copyright 2014-2021 © Benoit Blanchon. Proofread by Grammarly". To the right, there is a newsletter sign-up form with fields for "Your email" and "Subscribe", and links for "About", "Contact", and "Privacy".

図 5.8: JSON のサイズ確認画面

ライブラリマネージャ

タイプ 全て トピック 全て arduinoJSON

ArduinoJson
by Benoit Blanchon バージョン6.18.0 **INSTALLED**
A simple and efficient JSON library for embedded C++. ArduinoJson supports ✓ serialization, ✓ deserialization, ✓ MessagePack, ✓ fixed allocation, ✓ zero-copy, ✓ streams, ✓ filtering, and more. It is the most popular Arduino library on GitHub ❤️❤️❤️❤️. Check out arduinojson.org for a comprehensive documentation.
[More info](#)

バージョンを選択 インストール 更新

cloud4rpipi-esp-arduino
by Cloud4RPi
Connect a board to the Cloud4RPi control panel using MQTT - <https://cloud4rpipi.io>. Cloud4RPi client library for ESP8266 and ESP32 based boards. Dependencies: ArduinoJson, PubSubClient.
[More info](#)

Constellation
by Sébastien Warin
Arduino/ESP library for Constellation 1.8 Arduino/ESP library for Constellation 1.8. This library use the Arduino JSON library (<https://github.com/bblanchon/ArduinoJson>) (version 5.x) to encode & decode JSON.
[More info](#)

閉じる

図 5.9: ArduinoJson 用ライブラリのインストール

リスト 5.3: 最初のプログラム

```
// String input;

StaticJsonDocument<1536> doc;

DeserializationError error = deserializeJson(doc, input);

if (error) {
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    return;
}

JsonObject location = doc["location"];
const char* location_name = location["name"]; // "Saitama"
const char* location_region = location["region"]; // "Saitama"
const char* location_country = location["country"]; // "Japan"
float location_lat = location["lat"]; // 35.91
float location_lon = location["lon"]; // 139.66
const char* location_tz_id = location["tz_id"]; // "Asia/Tokyo"
long location_localtime_epoch = location["localtime_epoch"]; // 1626533912
const char* location_localtime = location["localtime"]; // "2021-07-17 23:58"

JsonObject current = doc["current"];
long current_last_updated_epoch = current["last_updated_epoch"]; // 1626533100
const char* current_last_updated = current["last_updated"]; // "2021-07-17 23:45"
float current_temp_c = current["temp_c"]; // 23.3
float current_temp_f = current["temp_f"]; // 73.9
int current_is_day = current["is_day"]; // 0

JsonObject current_condition = current["condition"];
const char* current_condition_text = current_condition["text"]; // "Clear"
const char* current_condition_icon = current_condition["icon"];
int current_condition_code = current_condition["code"]; // 1000

float current_wind_mph = current["wind_mph"]; // 3.8
float current_wind_kph = current["wind_kph"]; // 6.1
int current_wind_degree = current["wind_degree"]; // 250
const char* current_wind_dir = current["wind_dir"]; // "WSW"
int current_pressure_mb = current["pressure_mb"]; // 1019
float current_pressure_in = current["pressure_in"]; // 30.6
int current_precip_mm = current["precip_mm"]; // 0
int current_precip_in = current["precip_in"]; // 0
int current_humidity = current["humidity"]; // 88
```

```
int current_cloud = current["cloud"]; // 0
float current_feelslike_c = current["feelslike_c"]; // 24.9
float current_feelslike_f = current["feelslike_f"]; // 76.9
int current_vis_km = current["vis_km"]; // 16
int current_vis_miles = current["vis_miles"]; // 9
int current_uv = current["uv"]; // 1
float current_gust_mph = current["gust_mph"]; // 13.9
float current_gust_kph = current["gust_kph"]; // 22.3
```

コラム: コラム: サーバクライアント

サーバ? クライアント? とは何==[/column]

5.2 応用問題: Web サーバからの L チカ

第6章

応用編

6.1 外部からエアコンの電源を操作する

6.2 2台のESP32を使ってピンポンする

サーバクライアント Interface 2018.9 より Wi-Fi ネットワークはアクセス・ポイント (AP)を中心としたネットワークアクセスポイントは多くの場合、インターネットなどの他のネットワークに接続しており、その場合はルータとも呼ばれるアクセス・ポイントに接続する端末をステーション (STA) という ESP32 を AP モードするには> WiFi.softAP(ssid, password); STA モードでアクセスポイントに接続するには> WiFi.begin(ssid, password);

WiFi のアクセスポイントがなくても ESP32 が 2 庁あれば、片方をアクセスポイントにして通信できる

6.3 VScode から ESP32 にスケッチを書き込む

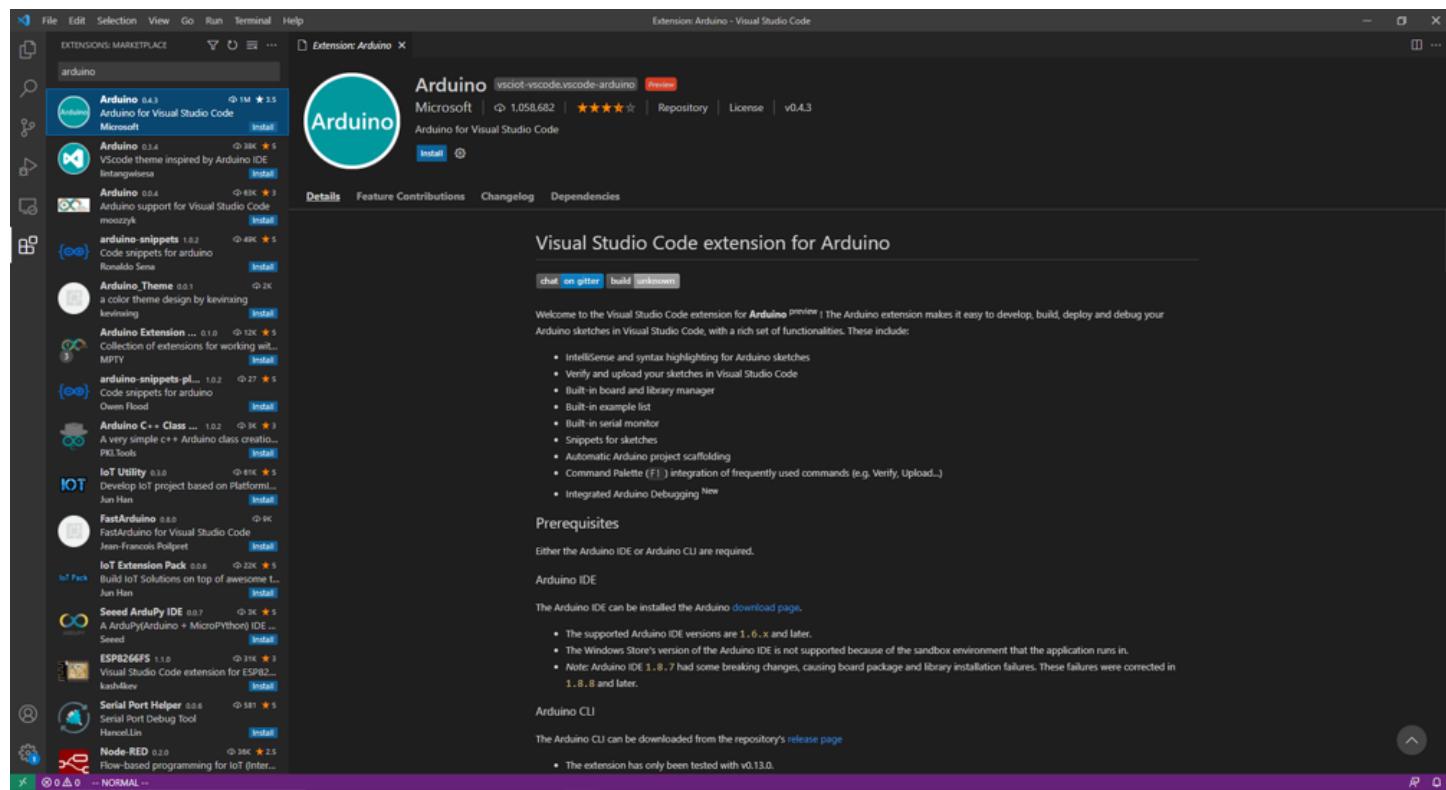


図 6.1: 1

コマンドパレット `ctrl shift P`

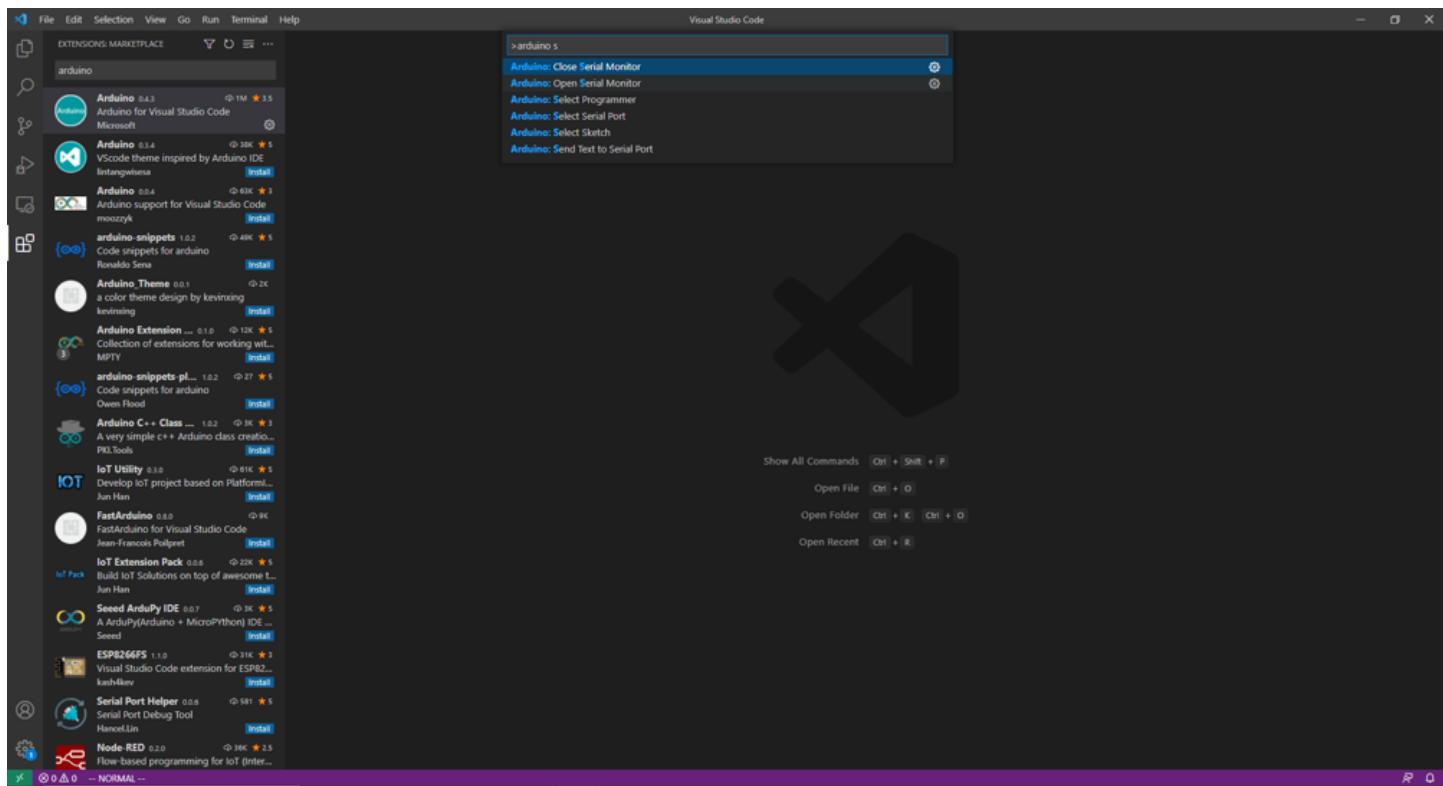


図 6.2: 2

arduino serial

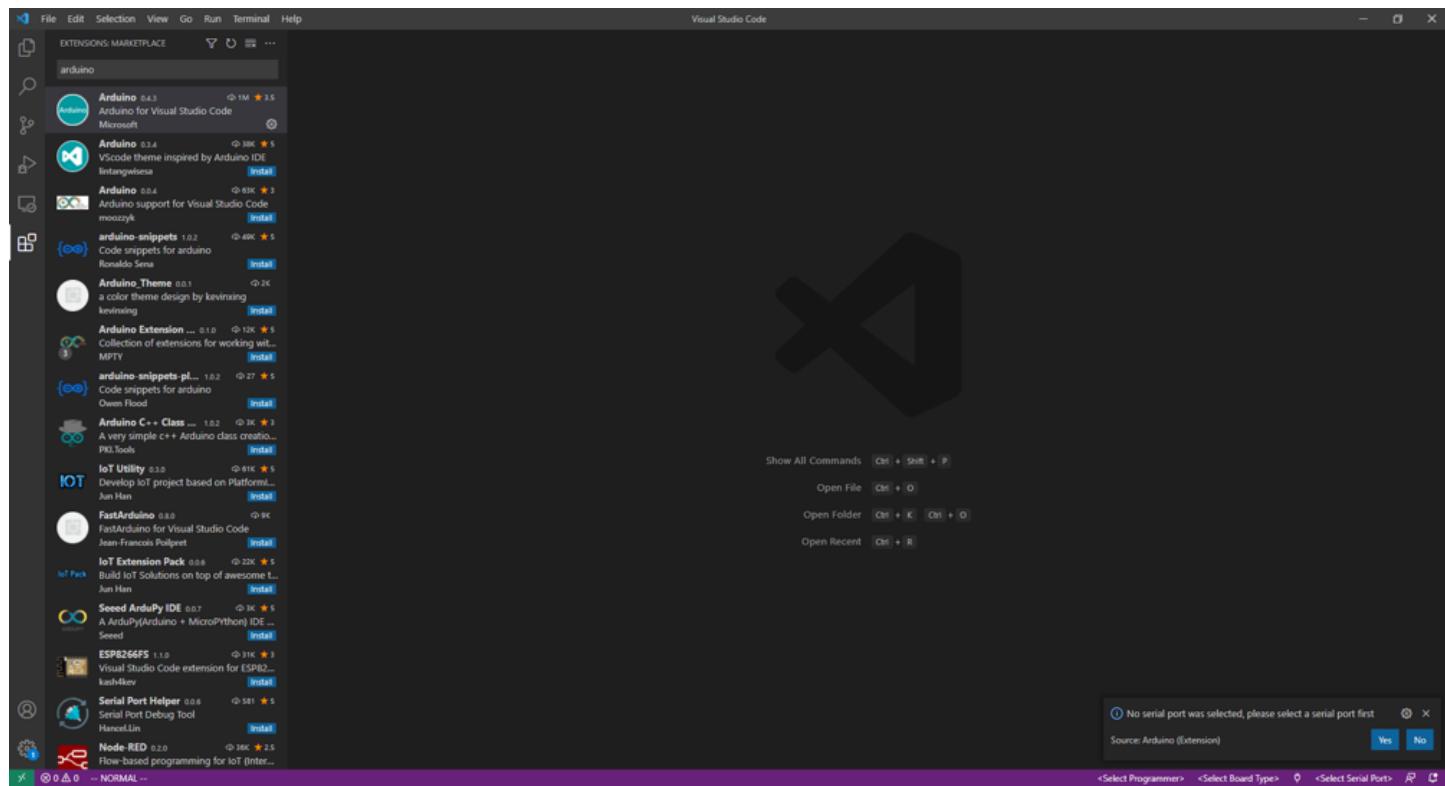


図 6.3: 3

/mnt/c/Users/Document/Arduino

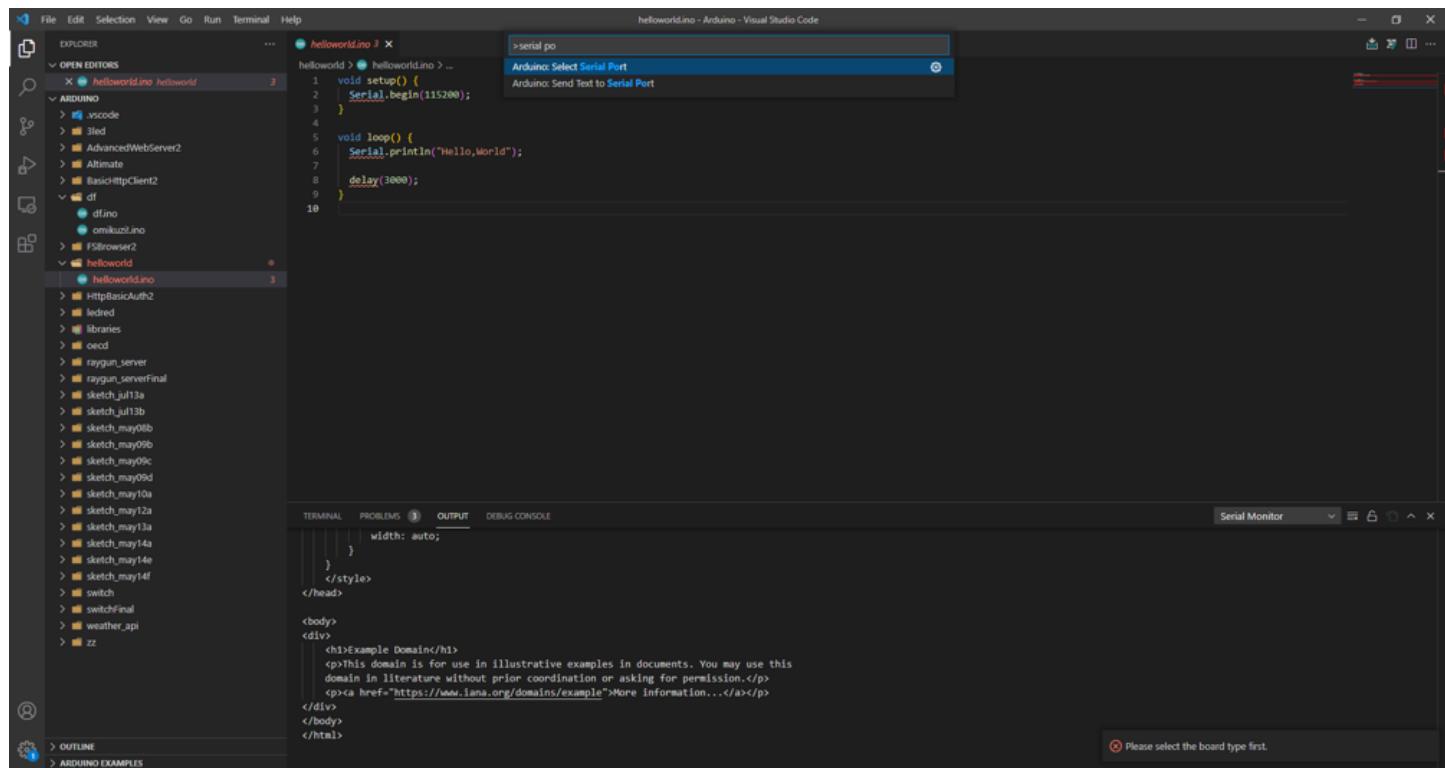


図 6.4: 4

serial port 選択

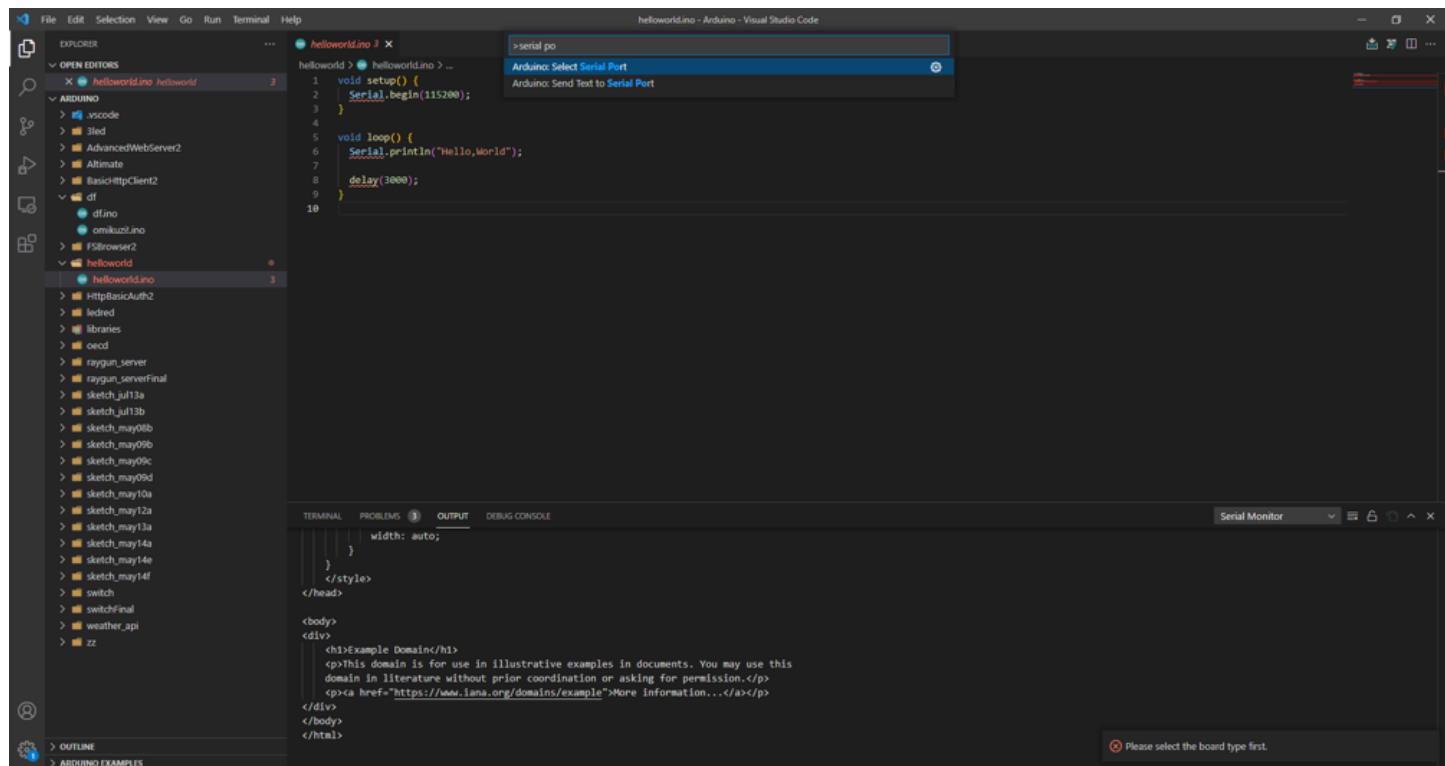


図 6.5: 5

ボード選択

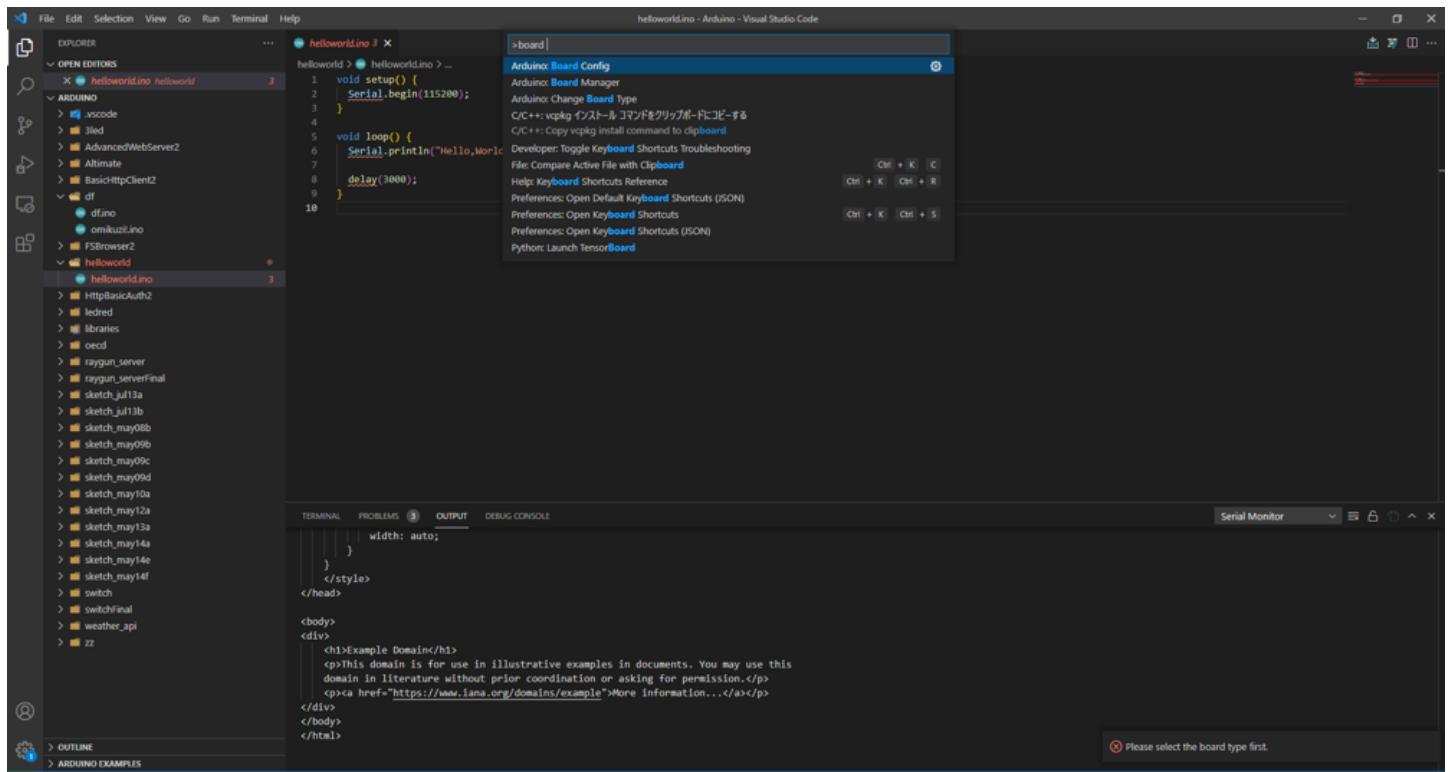


図 6.6: 6

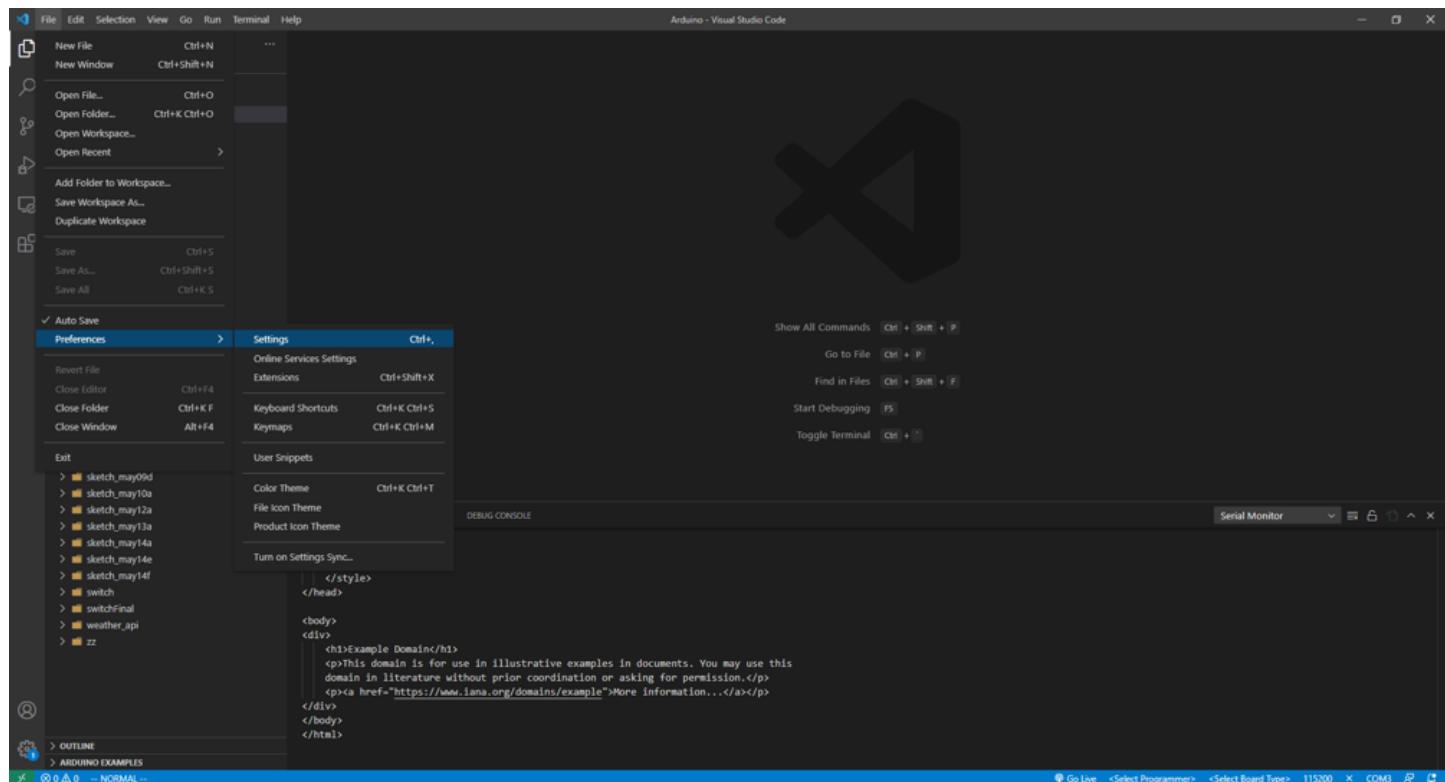


図 6.7: 7

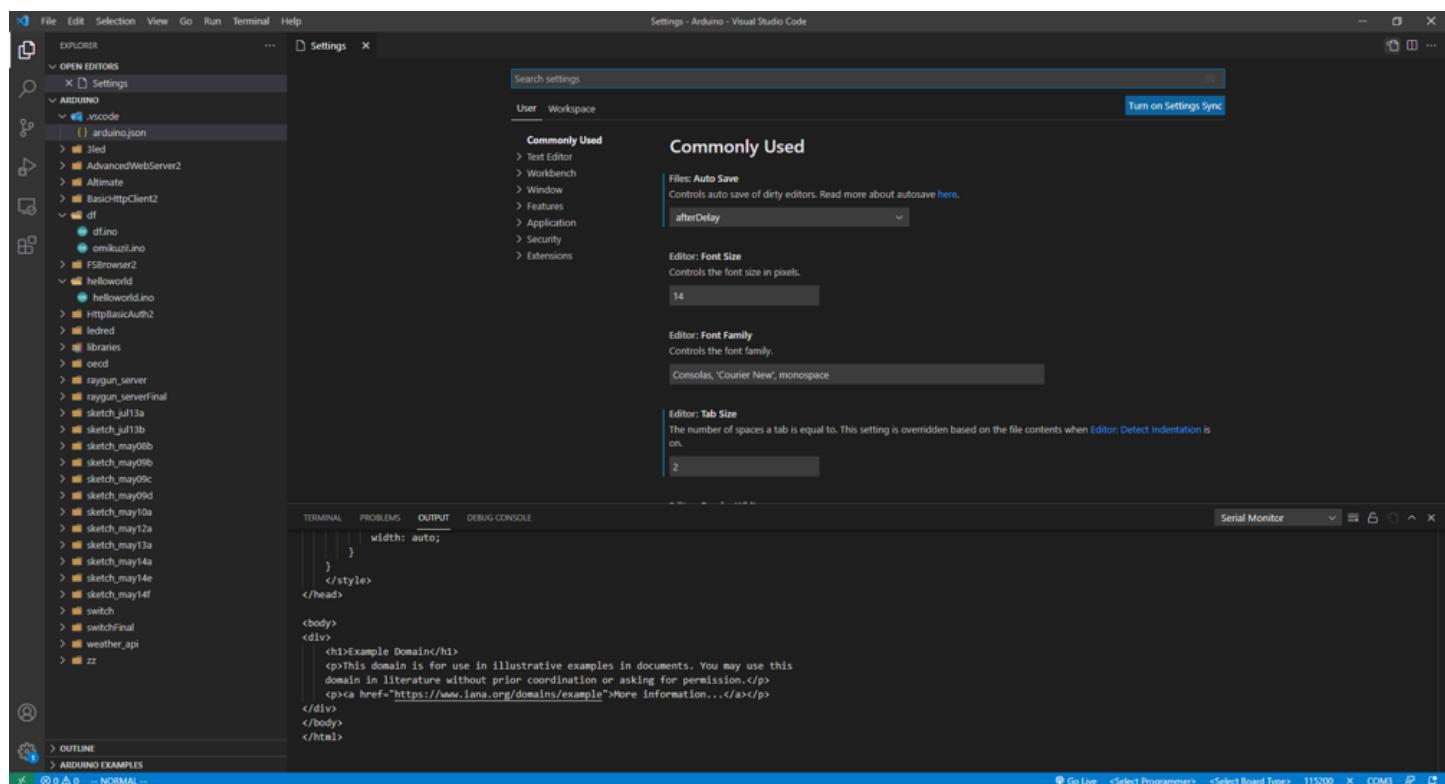


図 6.8: 8

リスト 6.1: json

```
"arduino.additionalUrls": [  
    "https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json"],
```

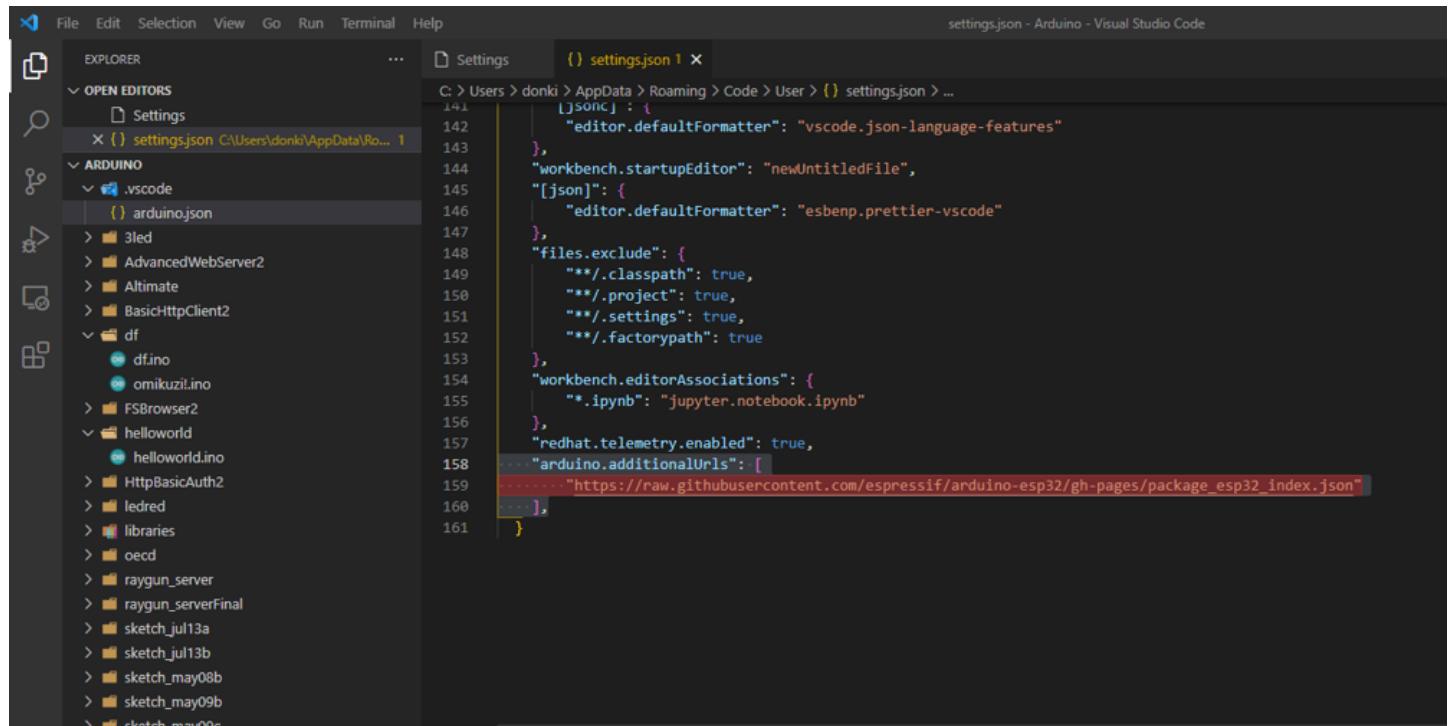


図 6.9: 9

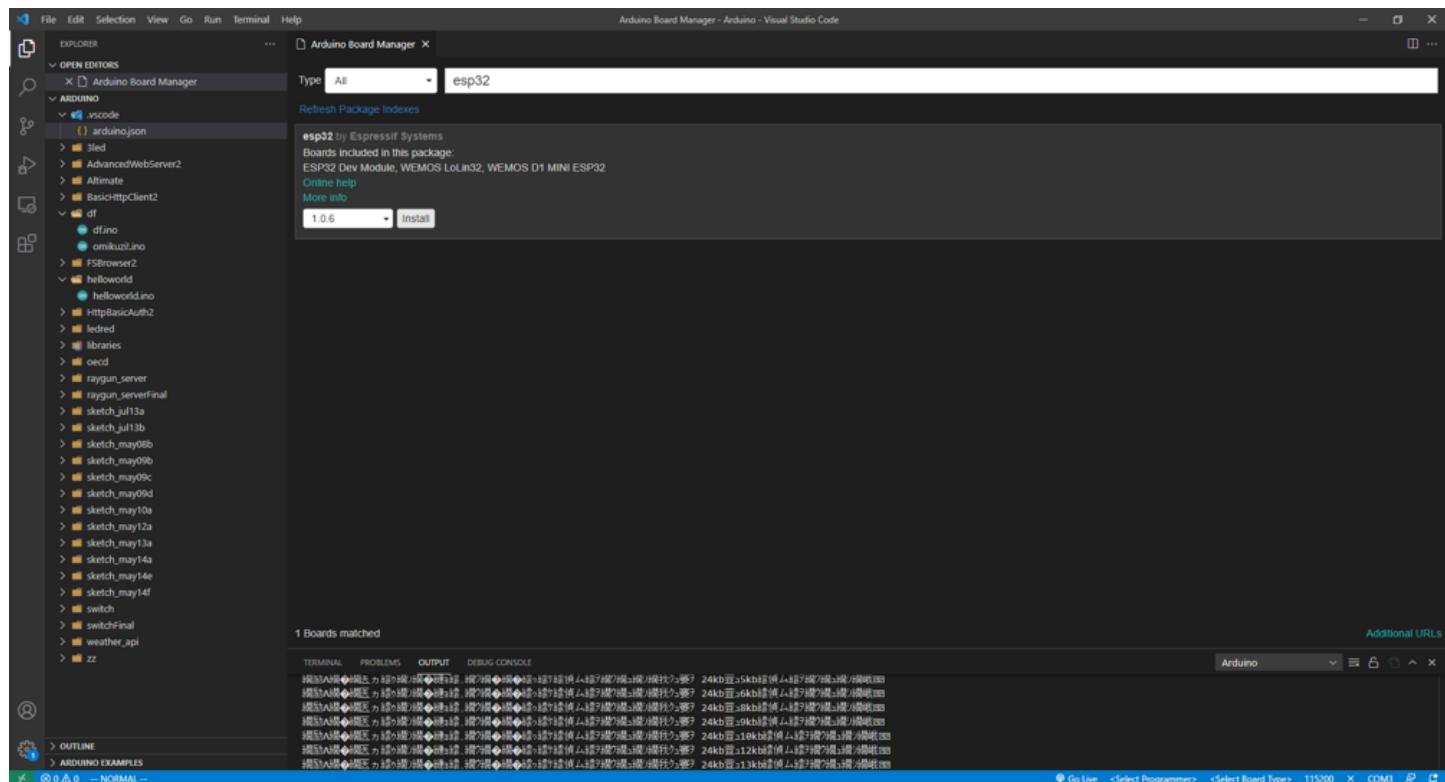


図 6.10: 10

arduino bord maneger

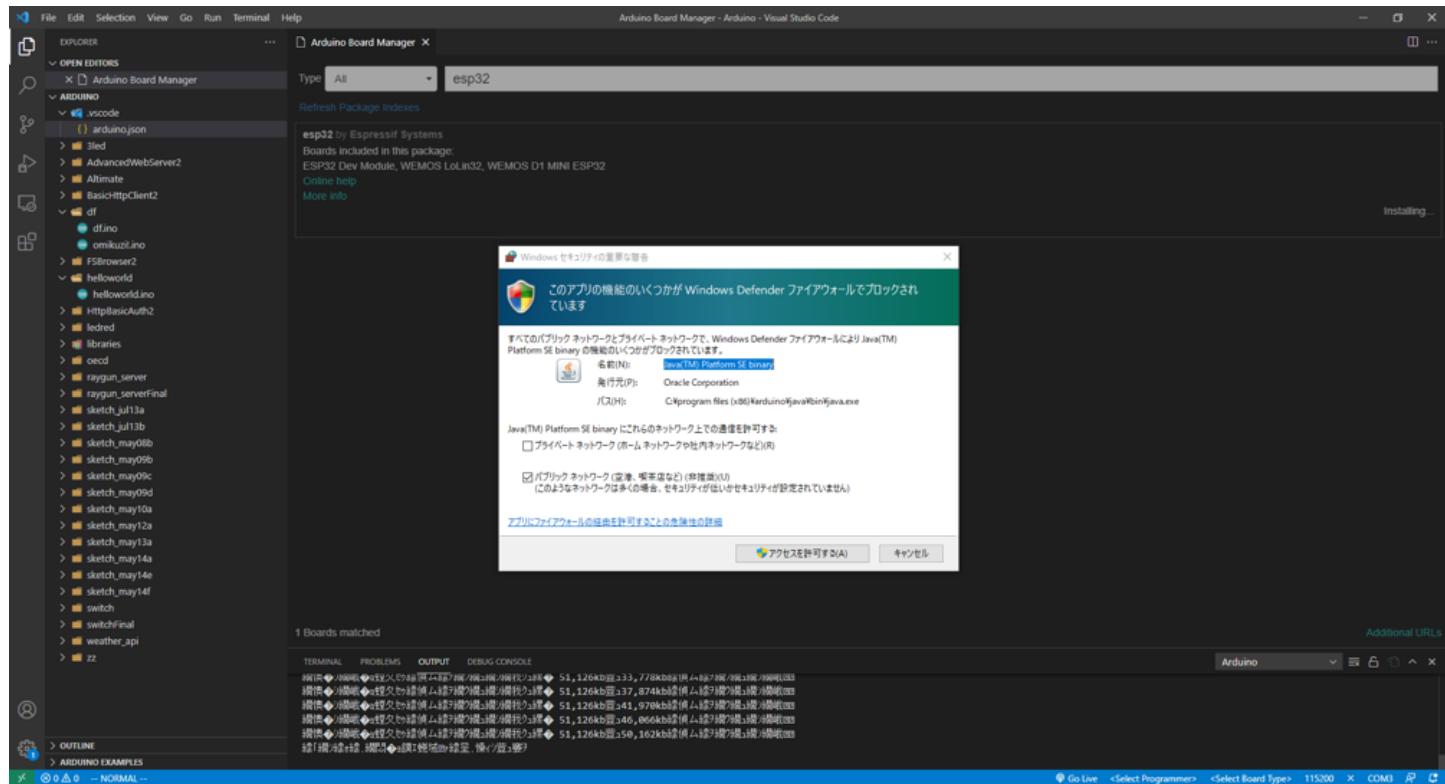


図 6.11: 11

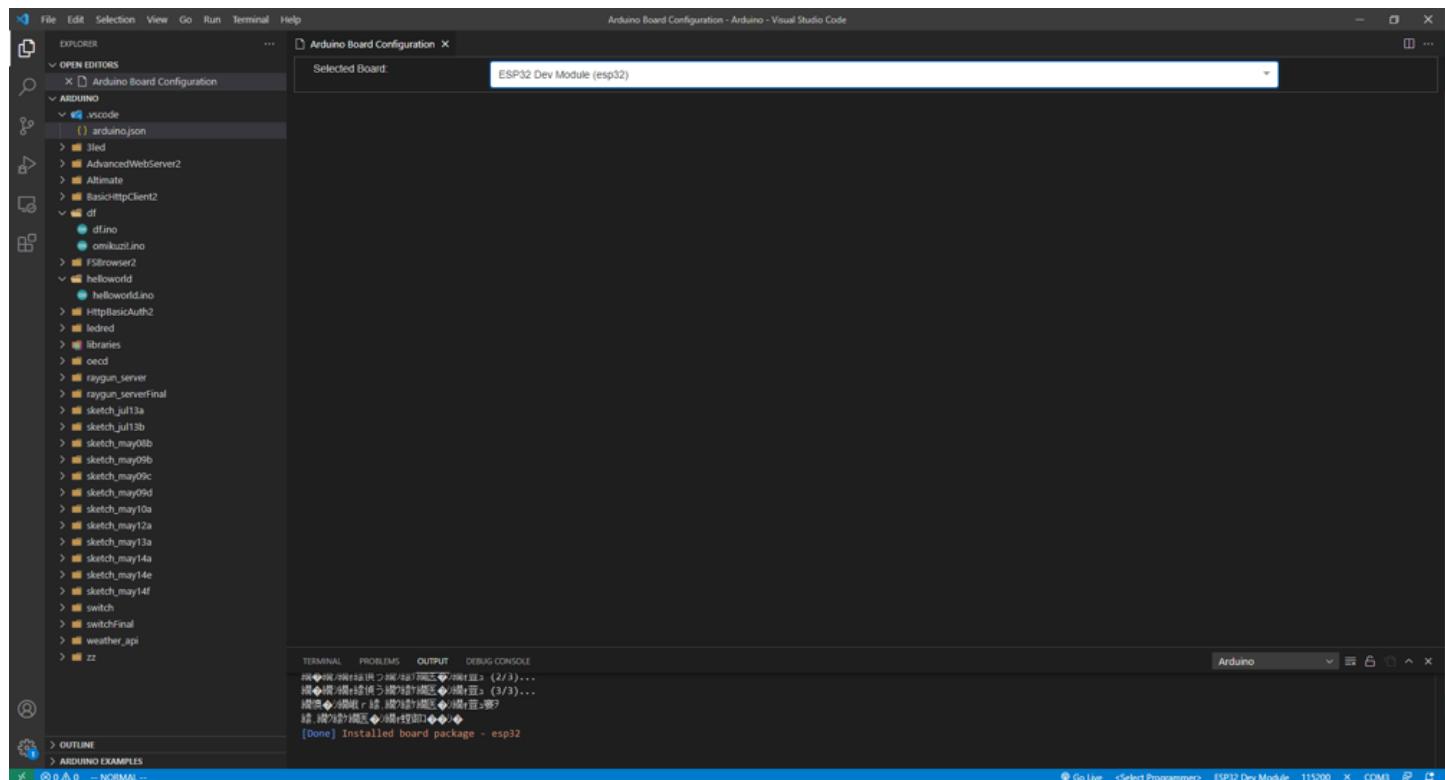


図 6.12: 12

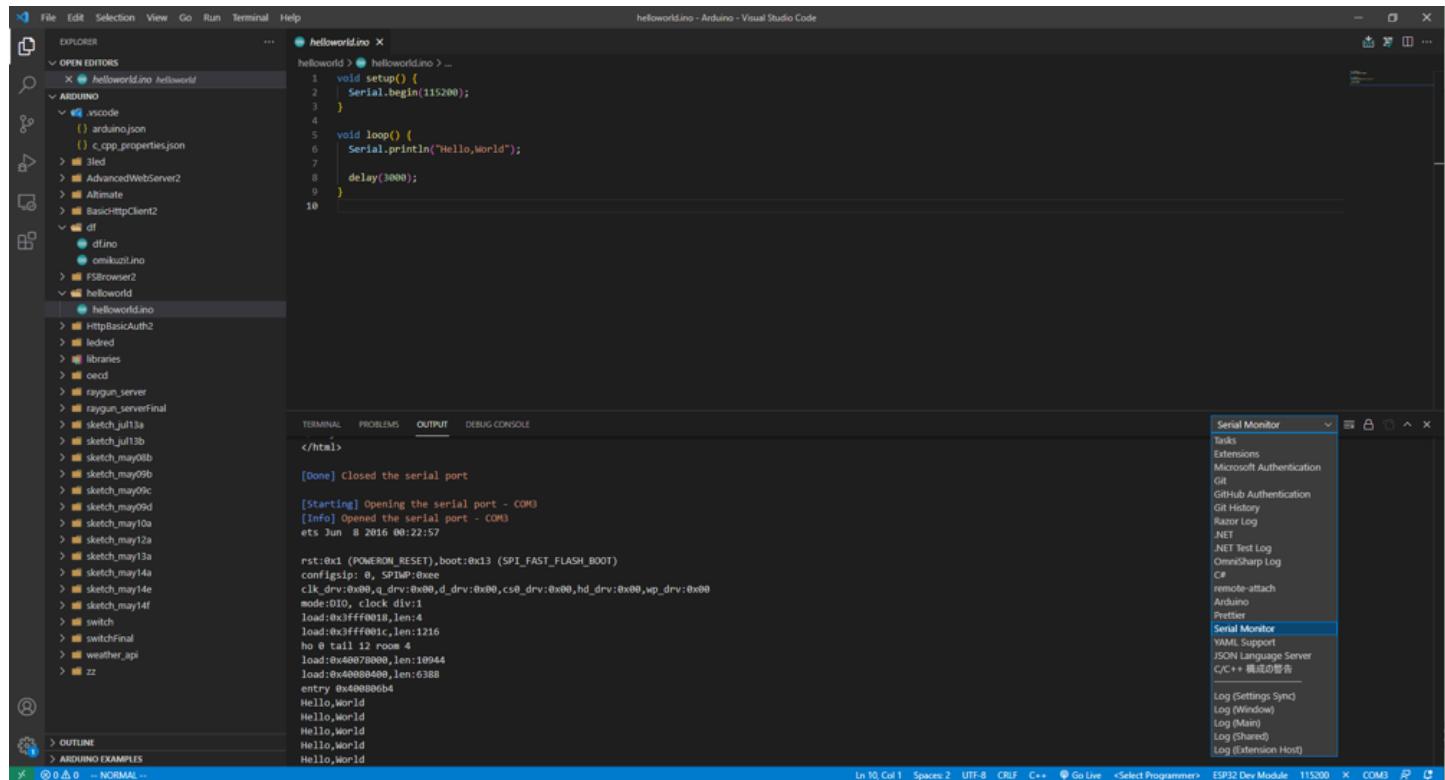


図 6.13: 13

Arduino IDE のほうを動かしているとうまくいかない

インクルードパスの設定 `c_cpp_properties.json` `ctrl shift P` select sketch でビルドしたいファイルを選択 <https://garretlab.web.fc2.com/arduino/introduction/vscode/>

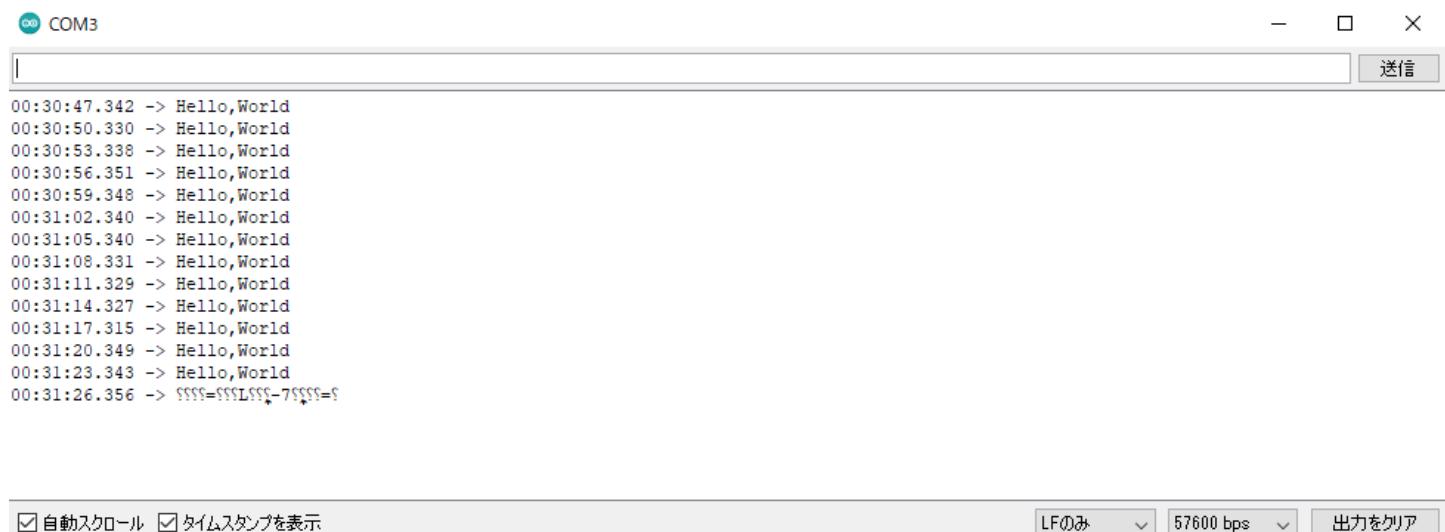


図 6.14: 14

付録 A

トラブルシューティング

A.1 シリアルモニタで文字化けがする



```
00:30:47.342 -> Hello,World
00:30:50.330 -> Hello,World
00:30:53.338 -> Hello,World
00:30:56.351 -> Hello,World
00:30:59.348 -> Hello,World
00:31:02.340 -> Hello,World
00:31:05.340 -> Hello,World
00:31:08.331 -> Hello,World
00:31:11.329 -> Hello,World
00:31:14.327 -> Hello,World
00:31:17.315 -> Hello,World
00:31:20.349 -> Hello,World
00:31:23.343 -> Hello,World
00:31:26.356 -> ?????=????L????-7????=?
```

自動スクロール タイムスタンプを表示 LFのみ 57600 bps 出力をクリア

図 A.1: 1

```
00:30:47.342 -> Hello,World  
00:30:50.330 -> Hello,World  
00:30:53.338 -> Hello,World  
00:30:56.351 -> Hello,World  
00:30:59.348 -> Hello,World  
00:31:02.340 -> Hello,World  
00:31:05.340 -> Hello,World  
00:31:08.331 -> Hello,World  
00:31:11.329 -> Hello,World  
00:31:14.327 -> Hello,World  
00:31:17.315 -> Hello,World  
00:31:20.349 -> Hello,World  
00:31:23.343 -> Hello,World  
00:31:26.356 -> ?????=????L?=?=????L?=?=????L?=?=Hello,World  
00:31:53.335 -> Hello,World  
00:31:56.328 -> Hello,World
```

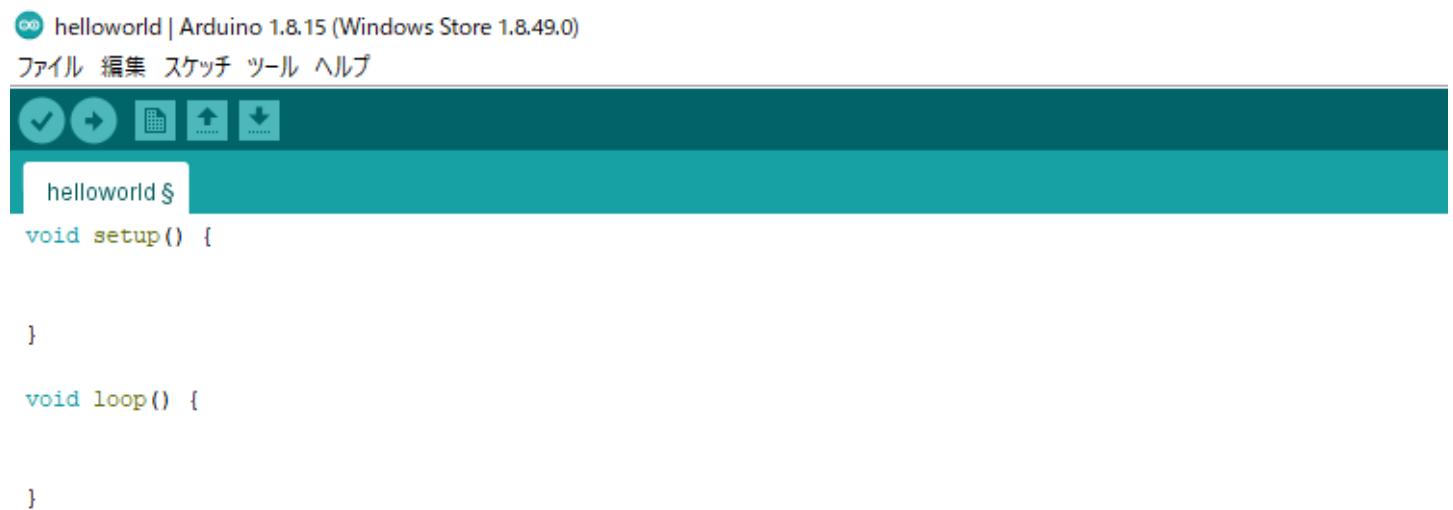
図 A.2: 2

Upload speed が間違っている可能性がある

A.2 プログラムが書き込めない

シリアルポートが間違っているかもしれない

A.3 プログラムを書き込んだが動作に反映されない



The screenshot shows the Arduino IDE interface. The title bar says "helloworld | Arduino 1.8.15 (Windows Store 1.8.49.0)". The menu bar includes "ファイル" (File), "編集" (Edit), "スケッチ" (Sketch), "ツール" (Tools), and "ヘルプ" (Help). Below the menu is a toolbar with icons for save, upload, and other functions. The main area shows the code for the "helloworld" sketch:

```
helloworld §  
void setup() {  
  
}  
  
void loop() {  
  
}
```

図 A.3: 3

プログラムの保存を忘れている Ctrl+S で保存してから読み込む

A.4 error: redefinition

```
c:\Users\donki\Documents\Arduino\helloworld\DHT11.ino: In function 'void setup()':  
DHT11:30:6: error: redefinition of 'void setup()'  
void setup() {
```

```
^
c:\Users\donki\Documents\Arduino\helloworld\helloworld.ino:1:6: note: 'void setup()'
previously defined here
void setup() {
^

c:\Users\donki\Documents\Arduino\helloworld\DHT11.ino: In function 'void loop()':
DHT11:37:6: error: redefinition of 'void loop()'
void loop() {
^

c:\Users\donki\Documents\Arduino\helloworld\helloworld.ino:5:6: note: 'void loop()',
void loop() {
^

exit status 1
```

- 解決法 Arduino コンパイルエラー (redefinition) 同じフォルダ内に setup() と loop() が重複している際に出るエラー Arduino はコンパイルをファルダ単位で行うため、このようなエラーが出る

著者紹介

THEToilet / @THEToilet

あとがきみたいなのにあこがれています。

執筆協力 / @raimu

はじめての IoT 講座

2021 年 7 月 12 日 初版第 1 刷 発行

著 者 THEToilet
