

My Book

cover sample for A5,
with bleed margin 3mm

published by Re:VIEW

はじめての IoT 講座

THEToilet 著

2021-07-23 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サークルであり、制作活動や日々の勉強を行ってい
ます

お問い合わせ先

本書に関するお問い合わせ : toilet.wc@gmail.com

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
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 Arduino IDE のインストール	6
2.3 ESP32 用ボードマネージャーのインストール	17
2.4 動作確認	17
ブレッドボード	17
シリアル通信とは	28
第3章 電子部品を使ってみよう	29
3.1 部品説明	29
ジャンプワイヤ	29
抵抗	29

目次

	タクトスイッチ	30
3.2	Lチカしよう！	31
	タクトスイッチでLチカ	32
	チャタリング	34
	応用問題	34
第4章	取得データをWebに公開しよう！	35
4.1	センサーを使おう	35
	I2Cとは	35
4.2	Webに公開しよう	41
	ambientについて	42
第5章	APIを使おう！	47
5.1	Weather APIを使う	47
	APIとは？	53
	サーバクライアント	53
	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	55
6.2	2台のESP32を使ってピンポンする	55
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	63
A.1	シリアルモニタで文字化けがする	63
A.2	プログラムが書き込めない	64
A.3	プログラムを書き込んだが動作に反映されない	65
A.4	error: redefinition	65
著者紹介		67

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/>
- SWITCHSCIENCE
 - <https://www.switch-science.com/>

- Amazon.co.jp
 - <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。本誌のサンプルを勧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	
ブレッドボード	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

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 - 32
E開発ボード 4MB

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

ブレッドボード

ブレッドボード 6穴版 EIC - 3901

<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 7cd60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・ジャンパーウイヤ(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 / 4W10k (100本入)

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

カーボン抵抗(炭素皮膜抵抗) 1 / 4W100 (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説明します。
Windows 環境を想定しているので、Mac 環境の方は少しやり方が違う
かもしれません。

2.1 ESP32 とは

ESP32 ってなに？？？
Espressif Systems 社が開発した SoC(System on a Chip) シリーズの名
前環境開発環境として

- Arduino IDE
- ESP-IDF
- MicroPython

などがありますが、今回は Arduino IDE を用いて開発を進めていきた
いと思います。

2.2 Arduino IDE のインストール

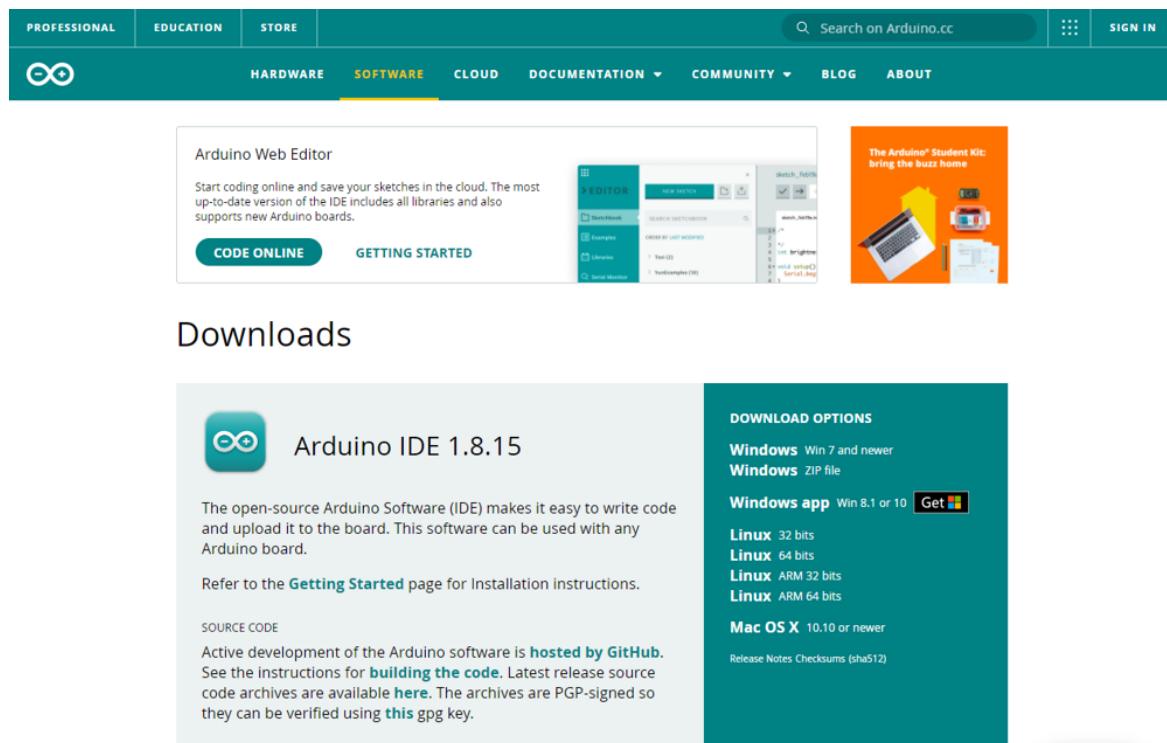


図 2.1: 1

第2章 環境構築

2.2 Arduino IDE のインストール

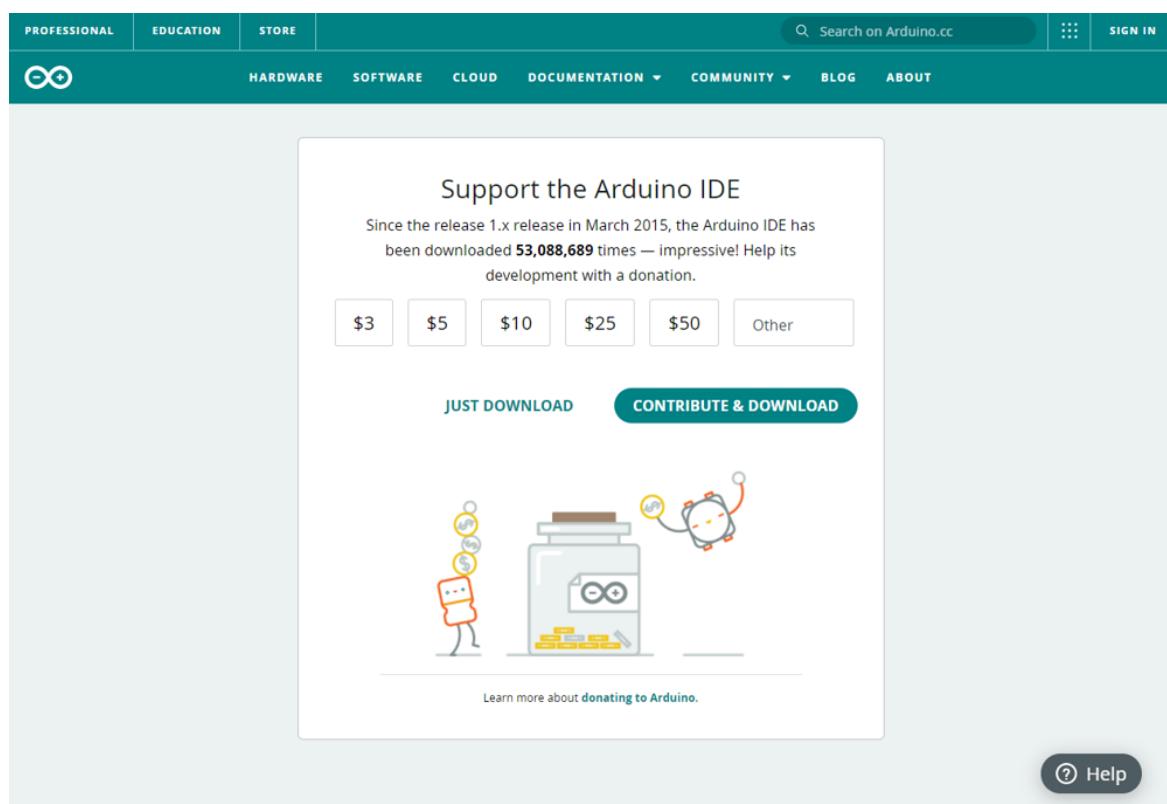


図 2.2: 2

第2章 環境構築

2.2 Arduino IDE のインストール

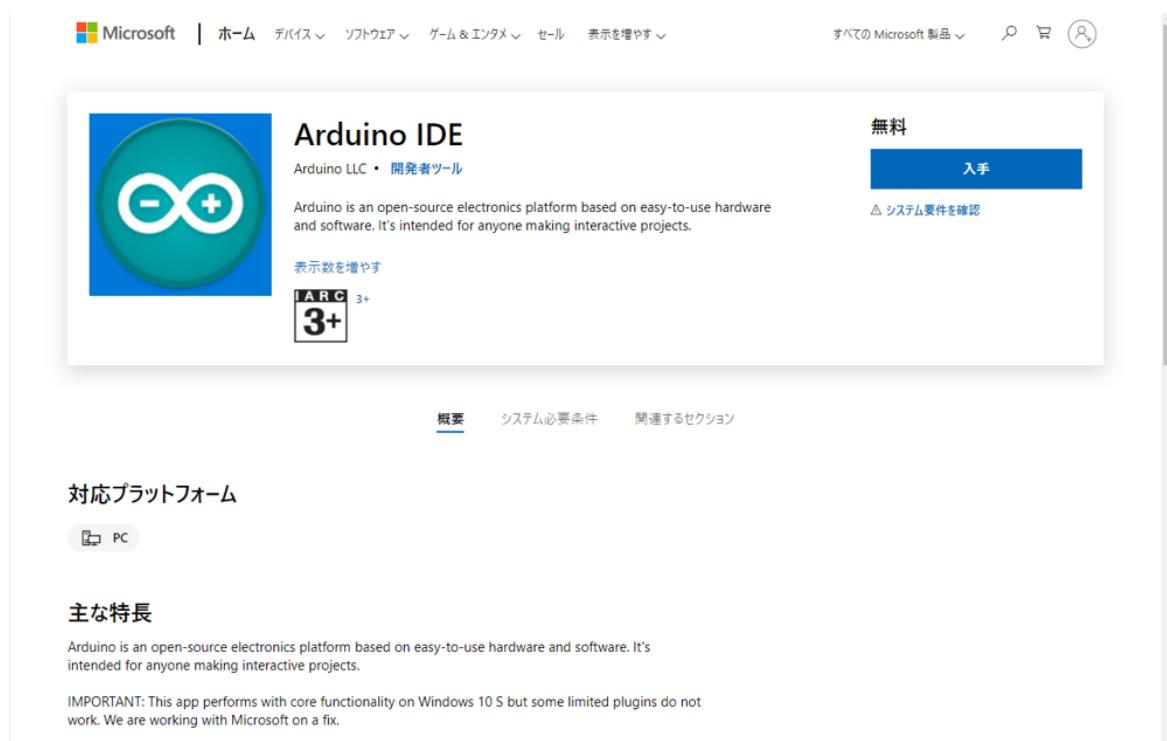


図 2.3: 3

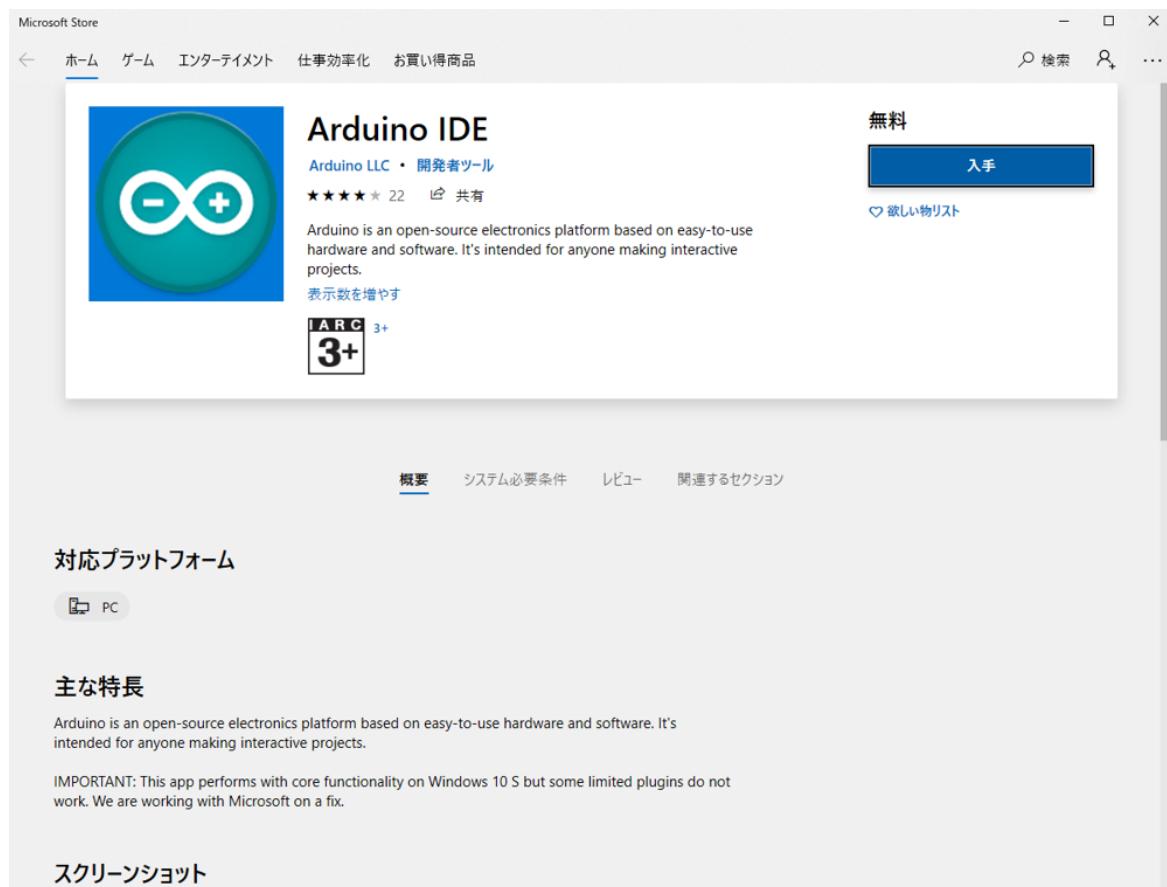


図 2.4: 4

第2章 環境構築

2.2 Arduino IDE のインストール

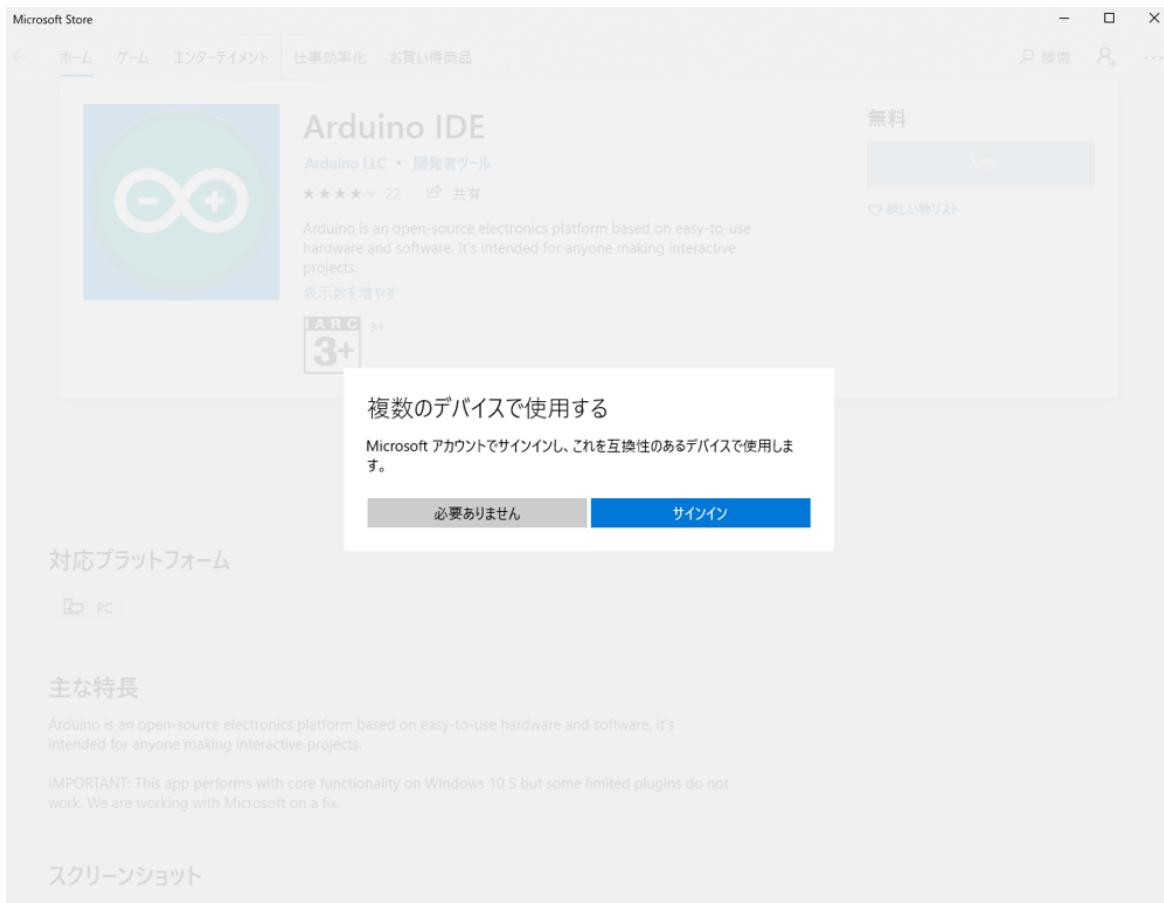


図 2.5: 5



図 2.6: 6

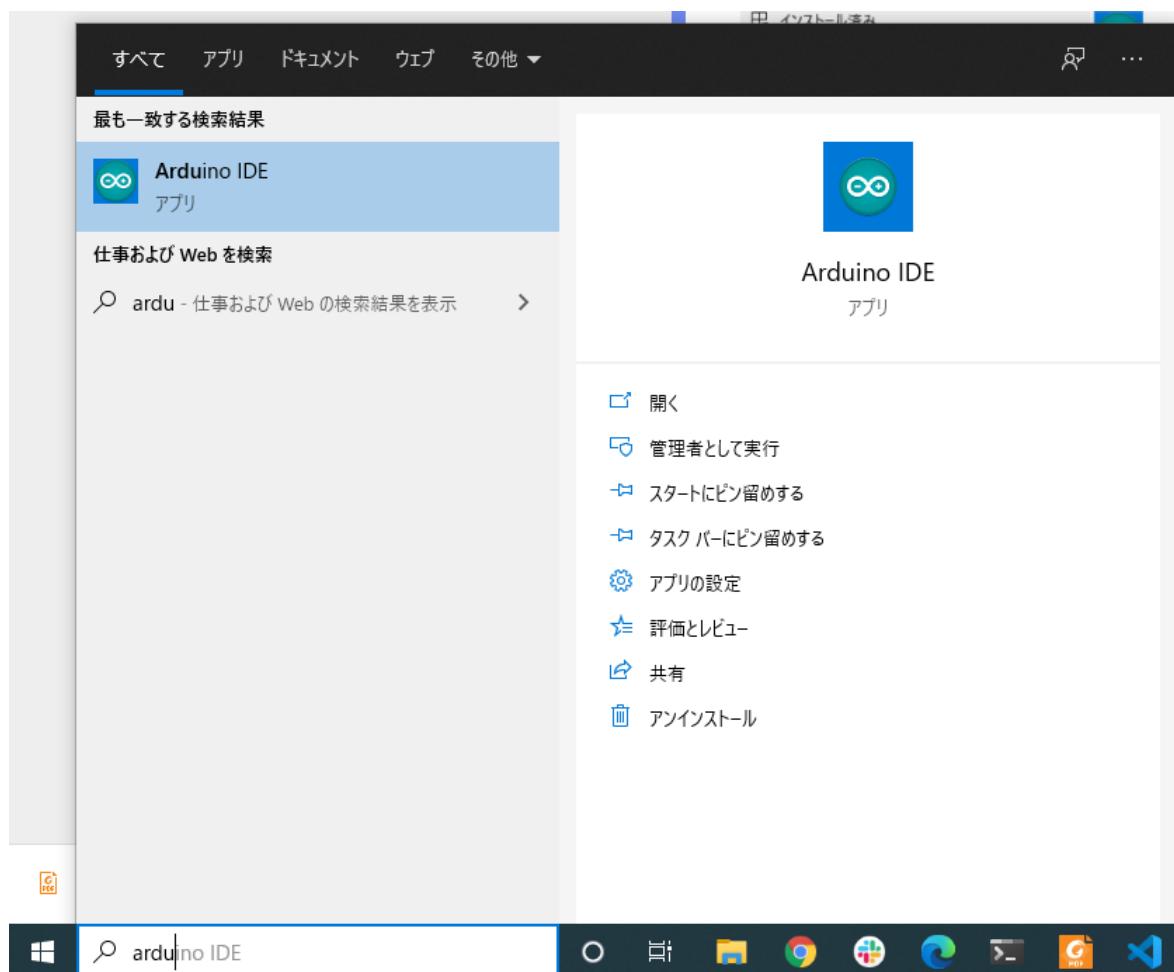


図 2.7: 7

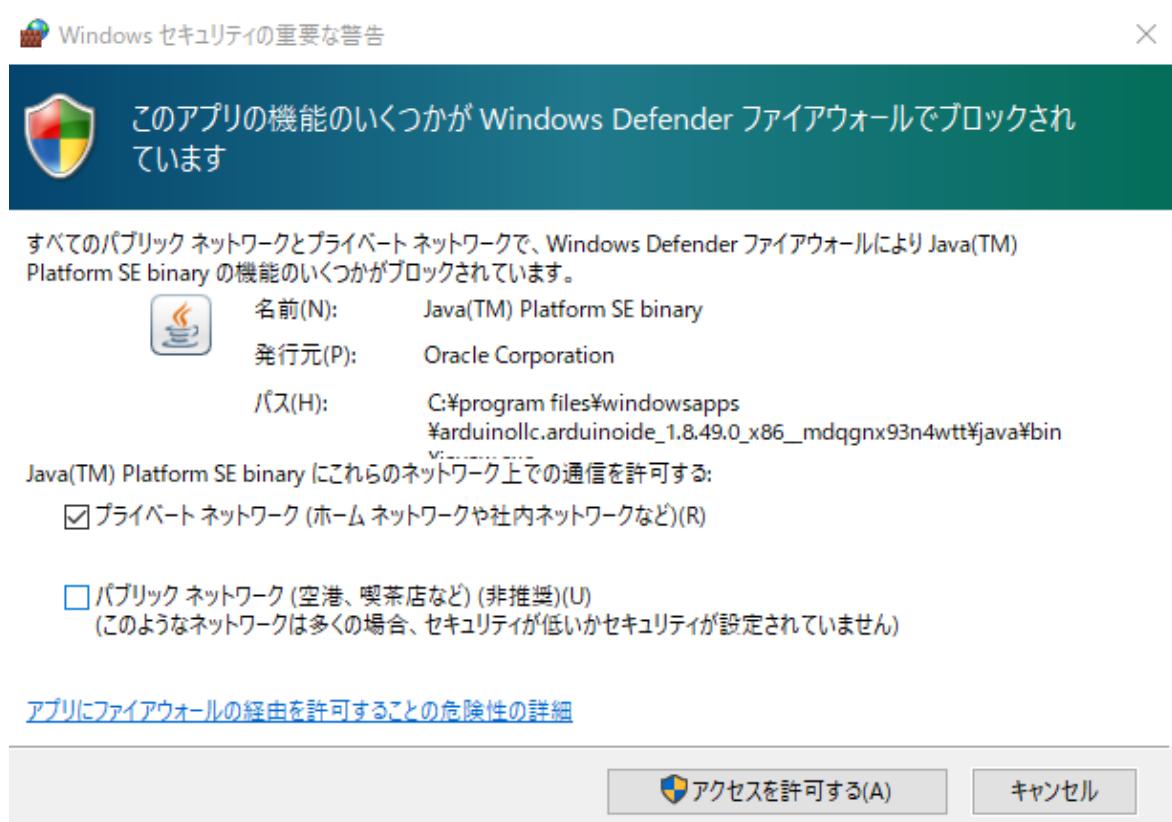


図 2.8: 8

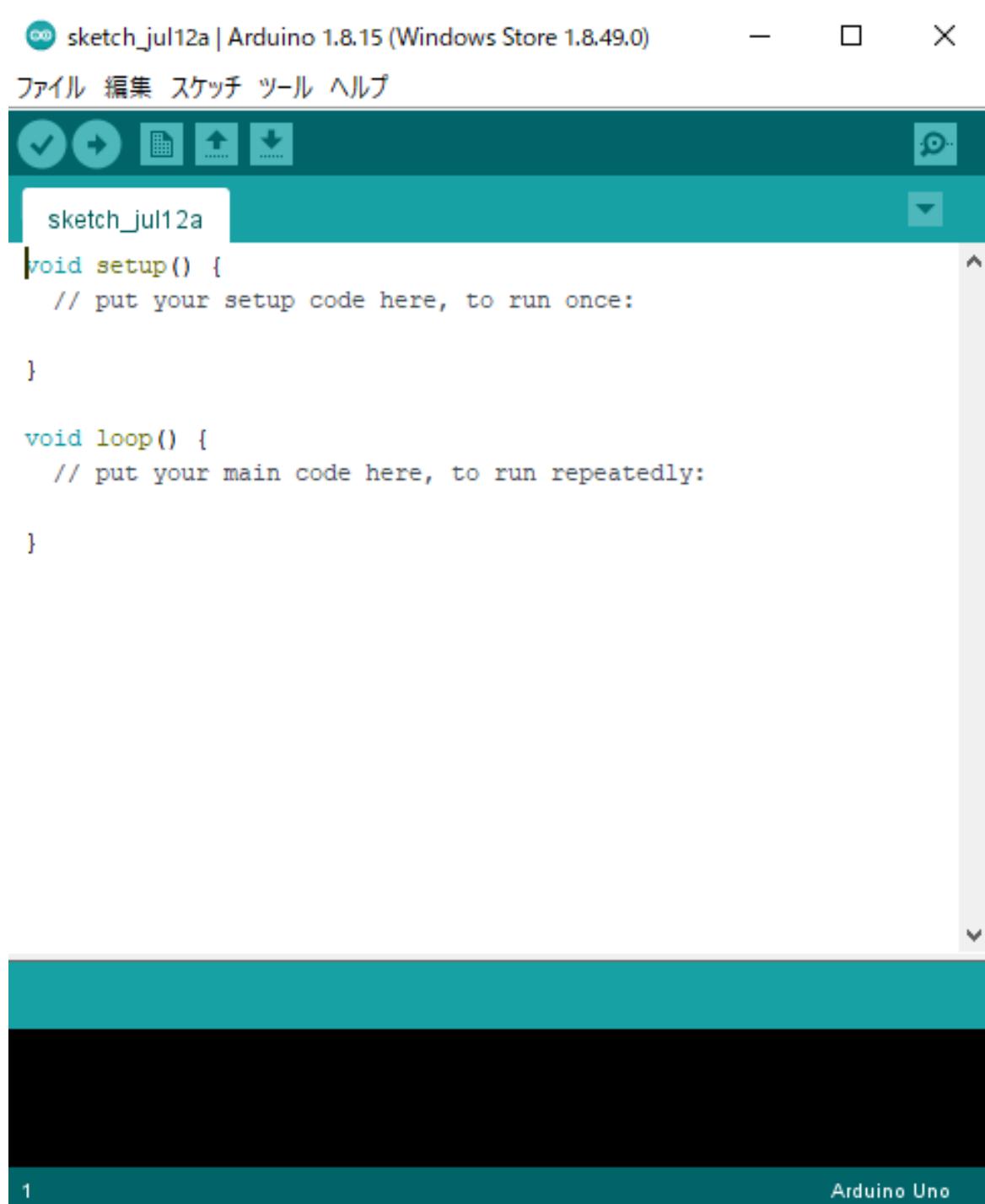


図 2.9: 9

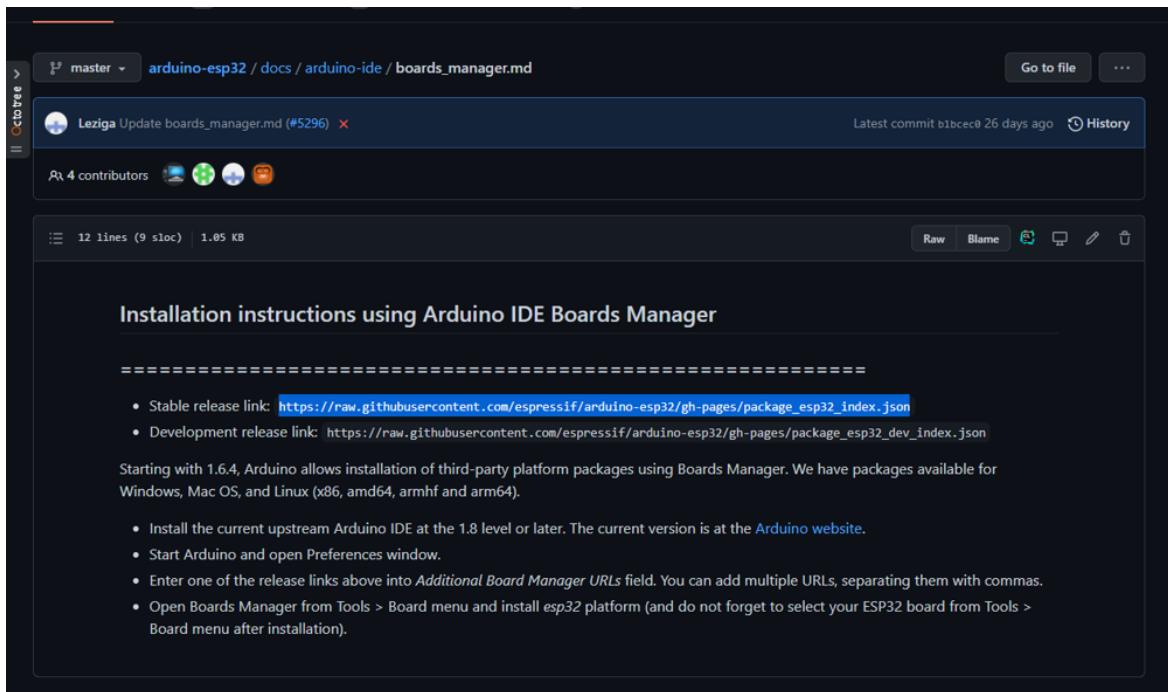


図 2.10: 10



The screenshot shows a JSON configuration file for the esp32 package. The file includes details about the package itself, its platforms, boards, and tools dependencies.

```
{
  "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-256:982da9aa181b6cb9c692dd4c9622b022ecc0d1e3aa0c5b70428ccc3c1b4556b",
          "size": "51126662",
          "help": {
            "online": ""
          }
        }
      ],
      "boards": [
        {
          "name": "ESP32 Dev Module"
        },
        {
          "name": "WEMOS LoLin32"
        },
        {
          "name": "WEMOS D1 MINI ESP32"
        }
      ],
      "toolsDependencies": [
        {
          "packager": "esp32",
          "name": "xtensa-esp32-elf-gcc",
          "version": "1.22.0-87-gc752ad5-5.2.0"
        },
        {
          "packager": "esp32",
          "name": "esptool_py",
          "version": "3.0.0"
        },
        {
          "packager": "esp32",
          "name": "mkspiffs",
          "version": "1.0.0"
        }
      ]
    }
  ]
}
```

図 2.11: 11

第2章 環境構築

2.2 Arduino IDE のインストール



図 2.12: 12

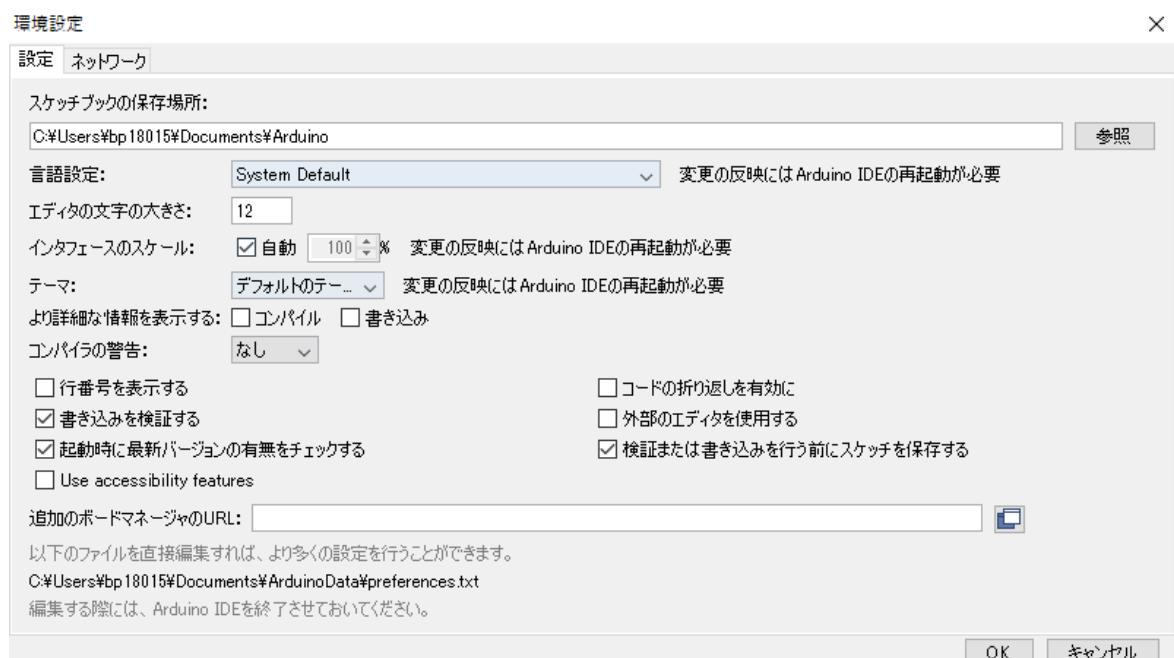


図 2.13: 13

https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md

2.3 ESP32用ボードマネージャーのインストール

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

2.4 動作確認

ここで動作確認をするために定番の HelloWorld を行いましょう

ブレッドボード

まず ESP32 をブレッドボードにさしましょう



図 2.14: 14

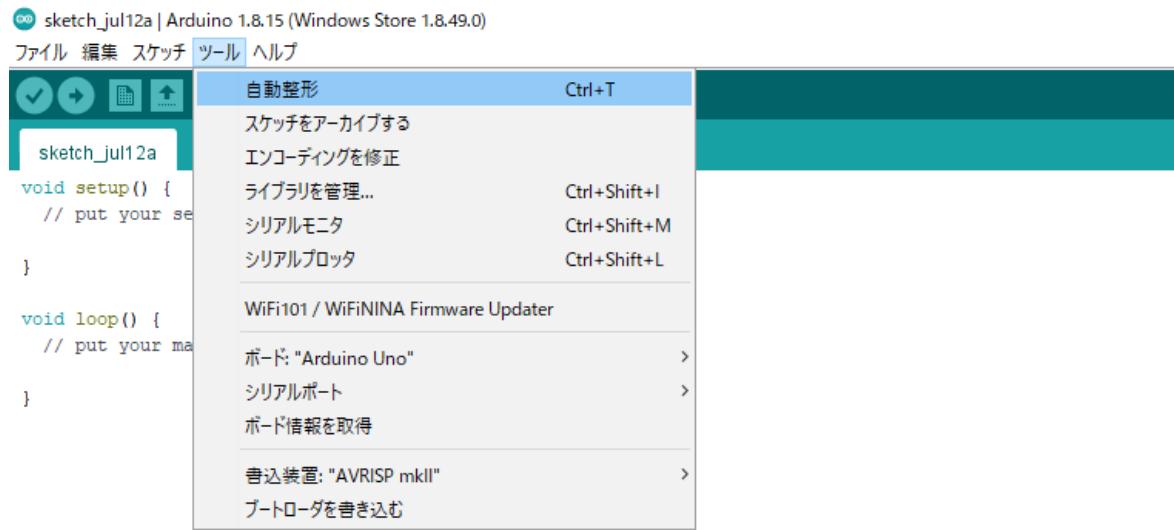


図 2.15: 15



図 2.16: 17

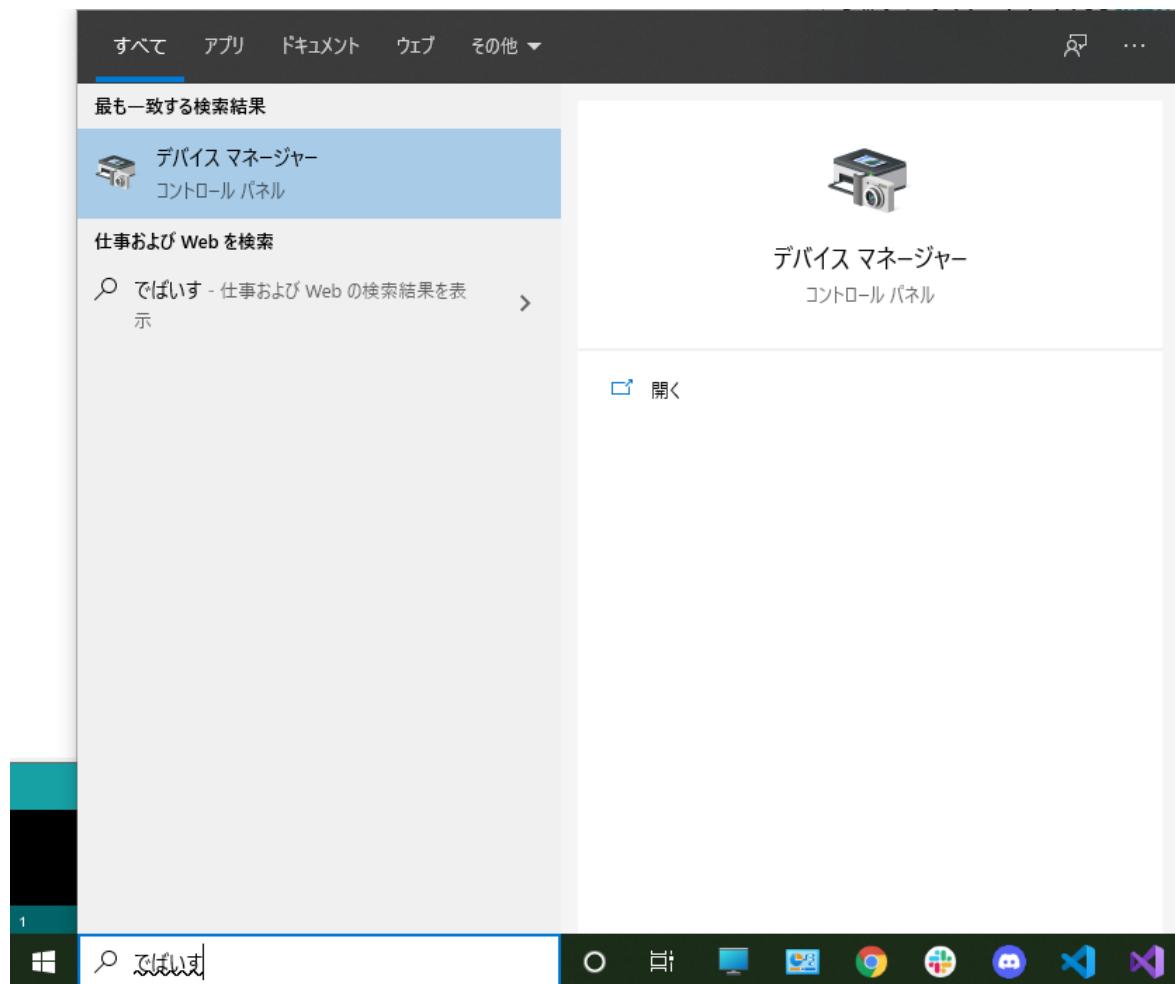


図 2.17: 18

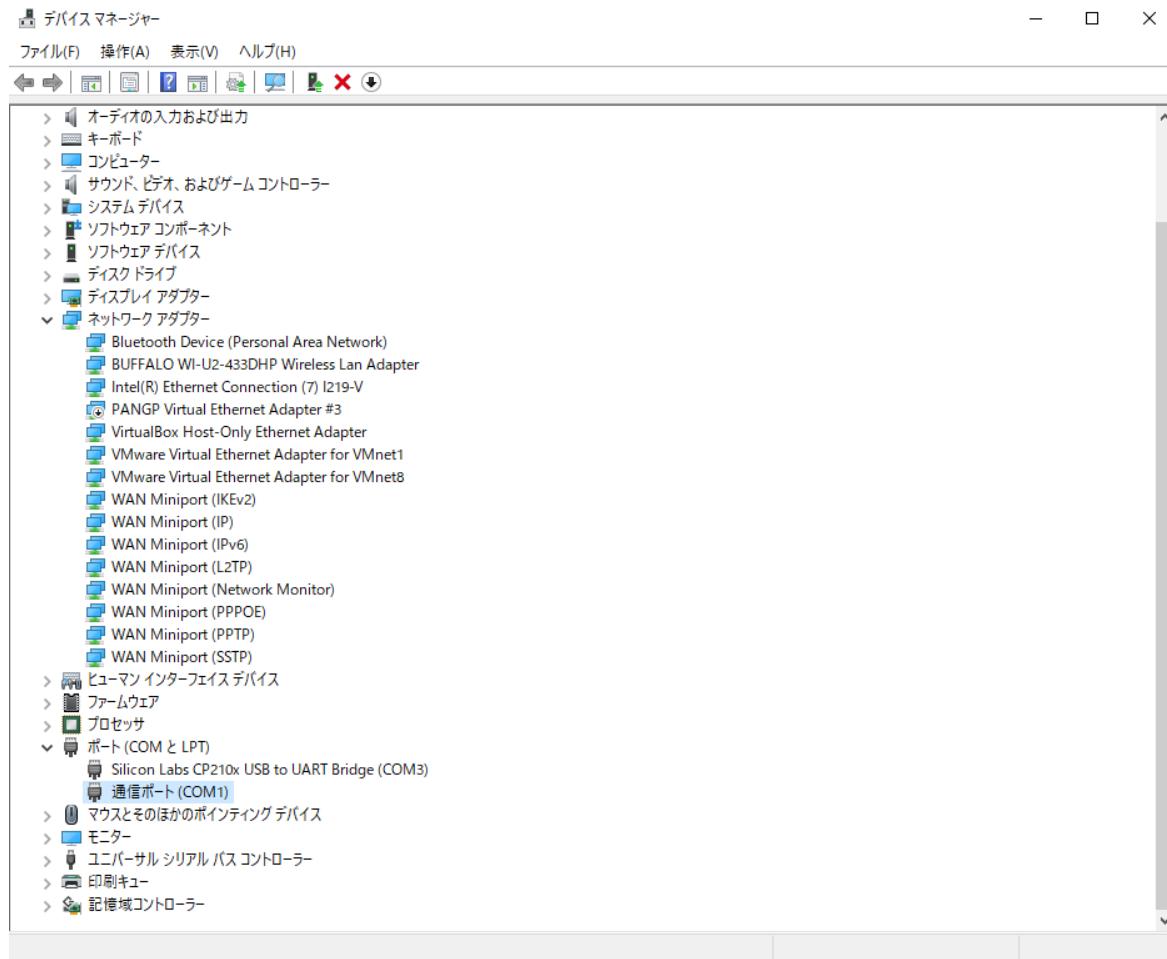


図 2.18: 19

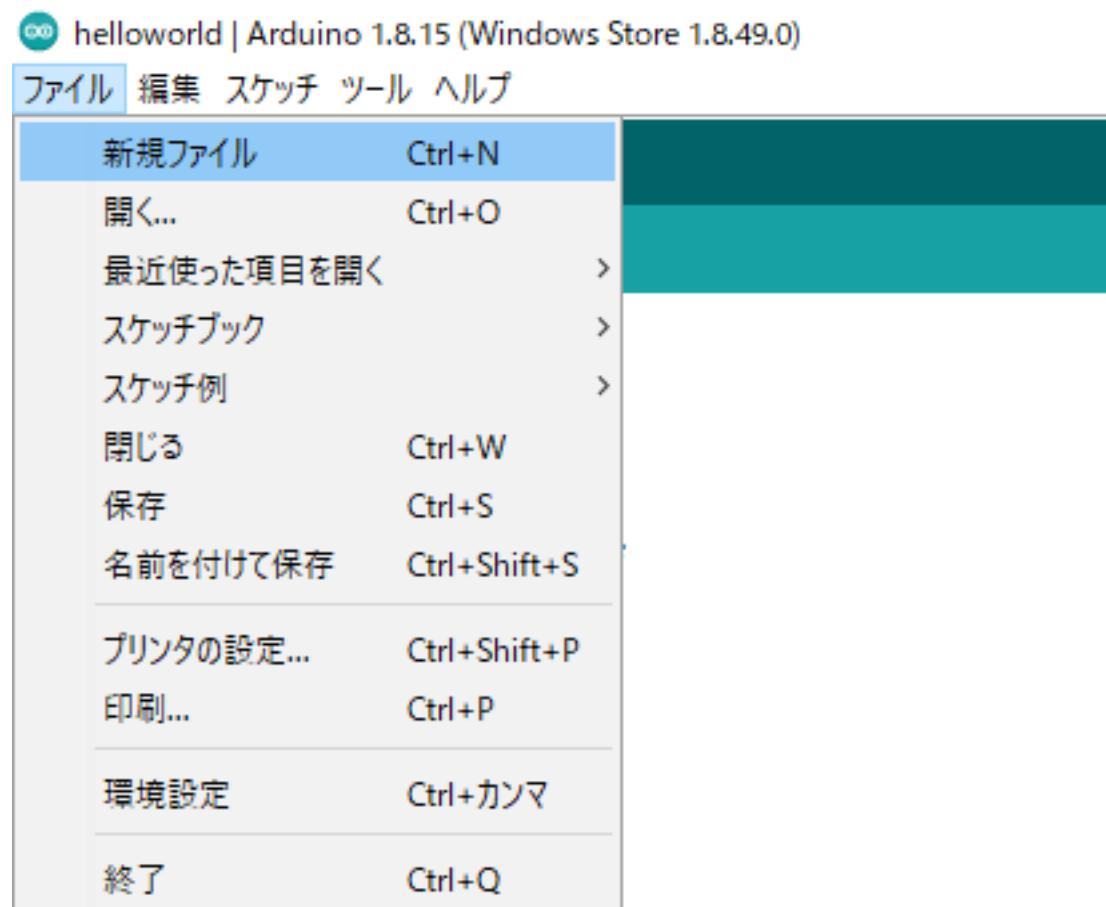


図 2.19: 20

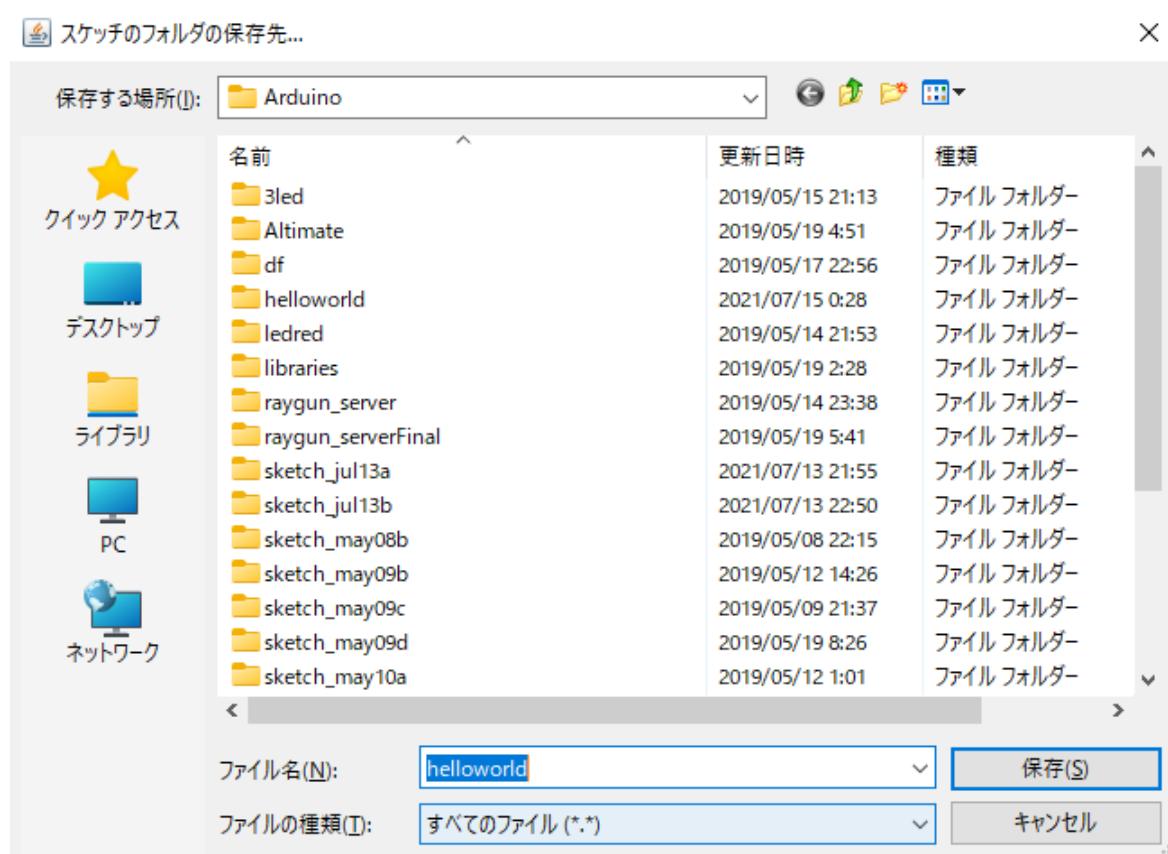


図 2.20: 21

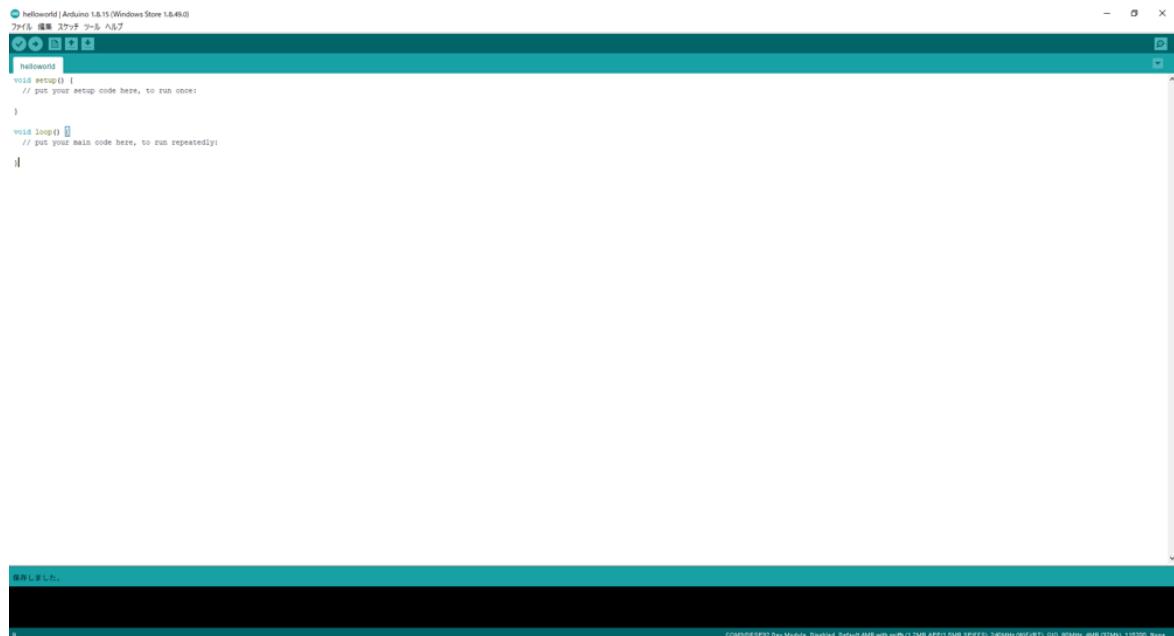


図 2.21: 22



図 2.22: 23

第2章 環境構築

2.4 動作確認

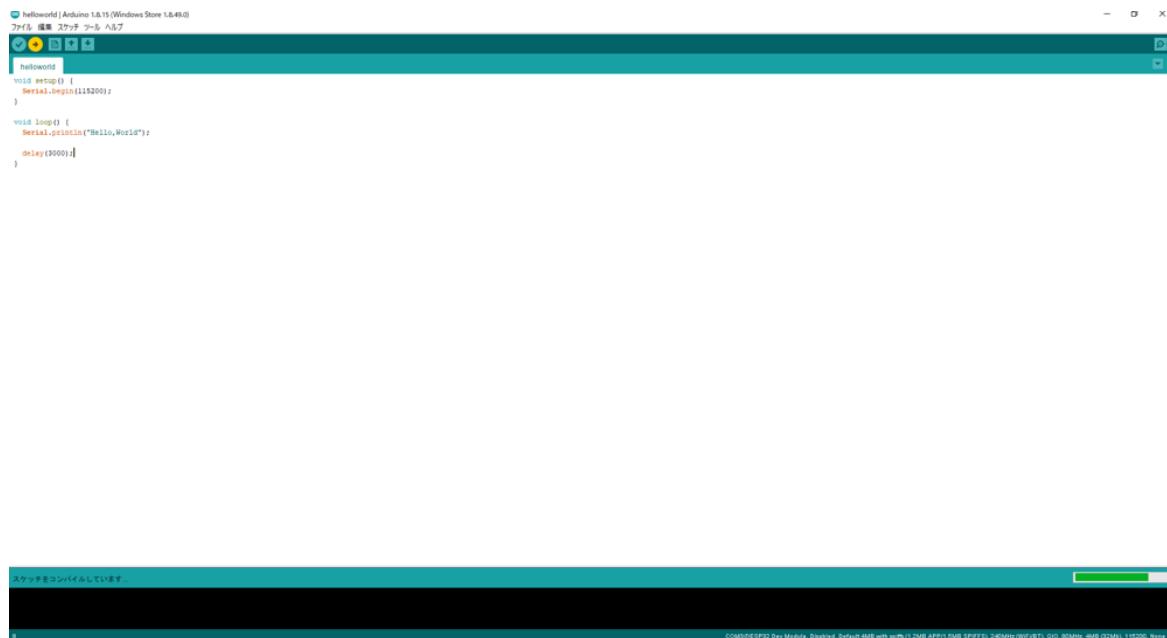


図 2.23: 24

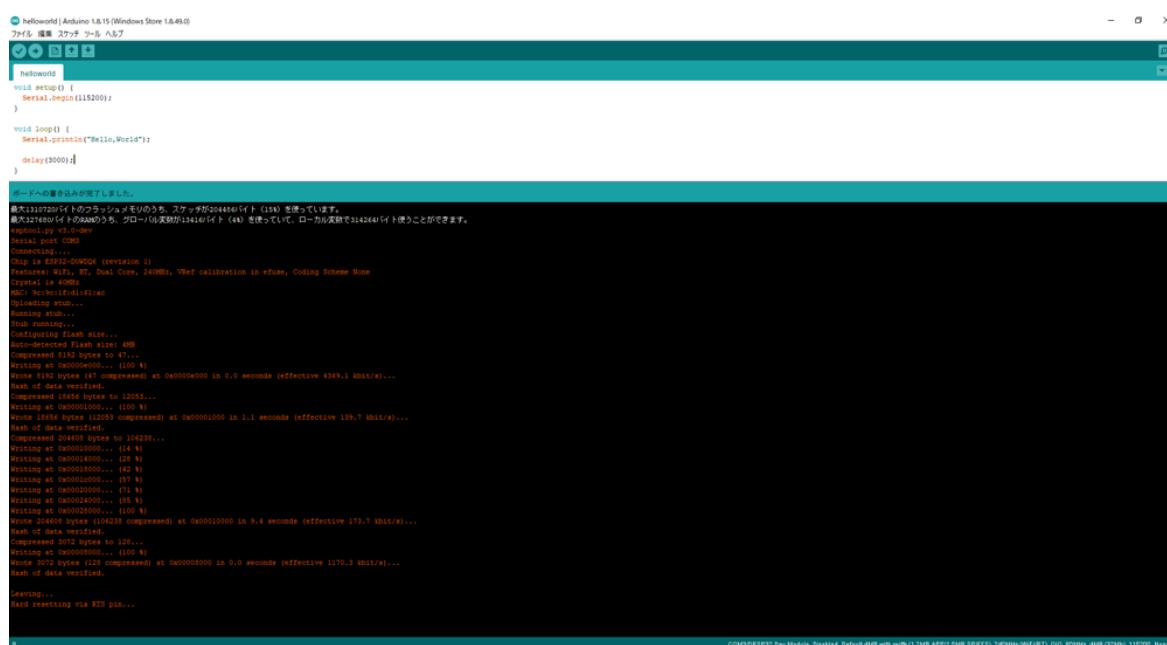


図 2.24: 25

第2章 環境構築

2.4 動作確認

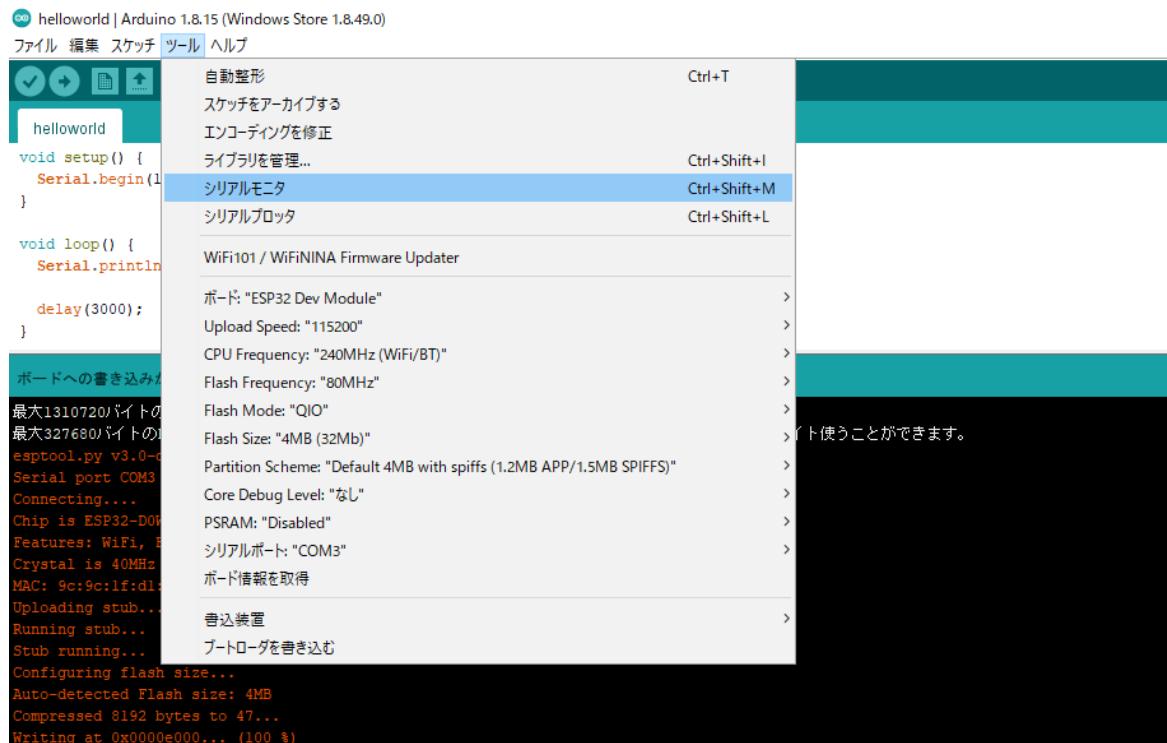


図 2.25: 26

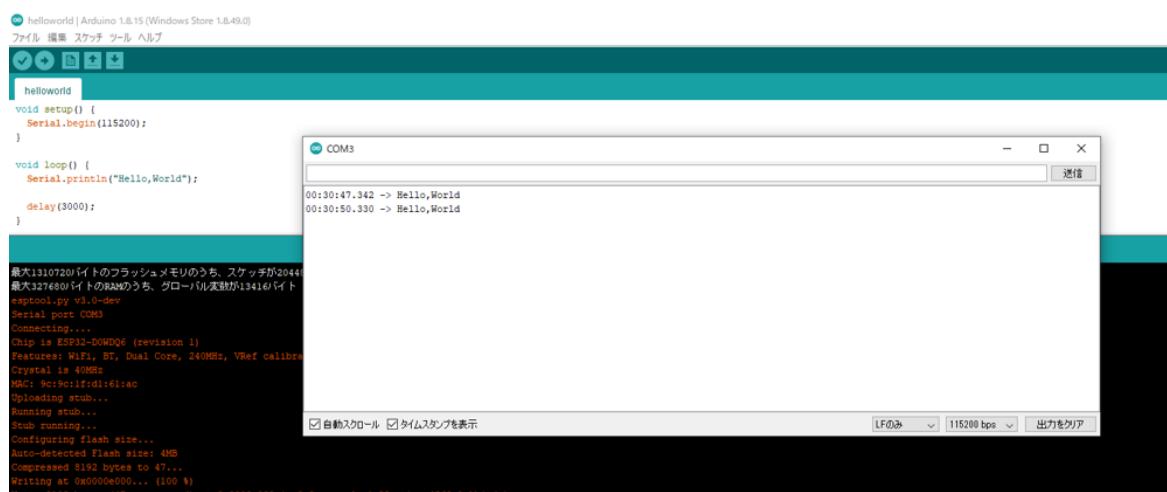


図 2.26: 27

リスト 2.1: 最初のプログラム

```
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    Serial.println("Hello,World");  
    delay(3000);  
}
```

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。

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

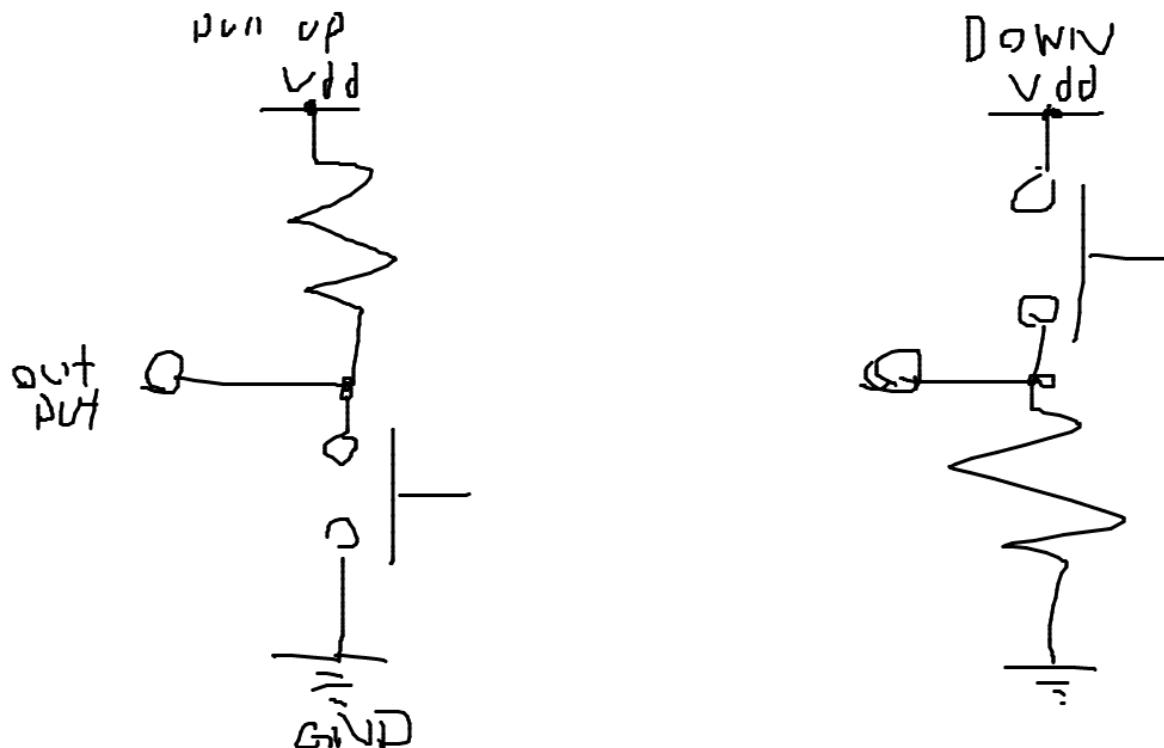


図 3.1: pullupdown

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。

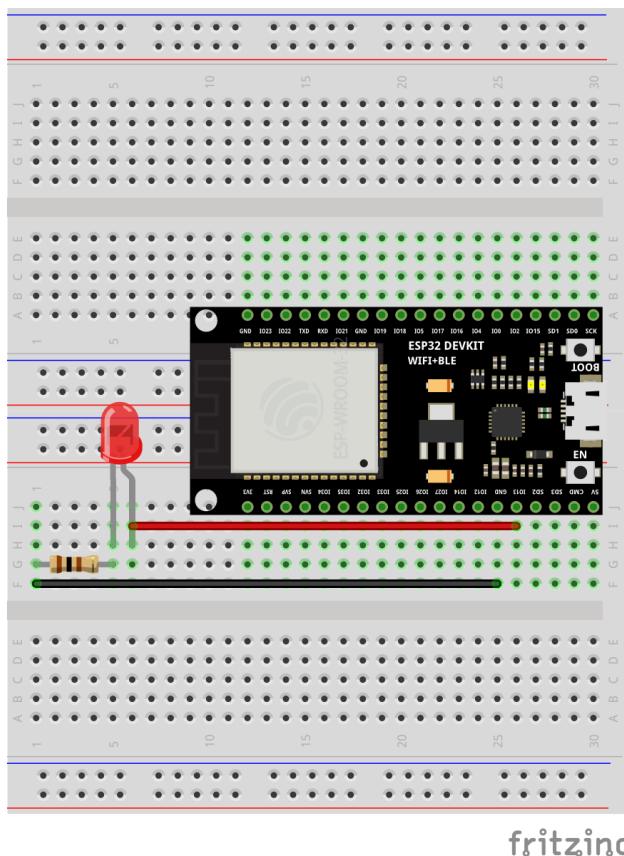


図 3.2: led2

リスト 3.1: Ltic

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

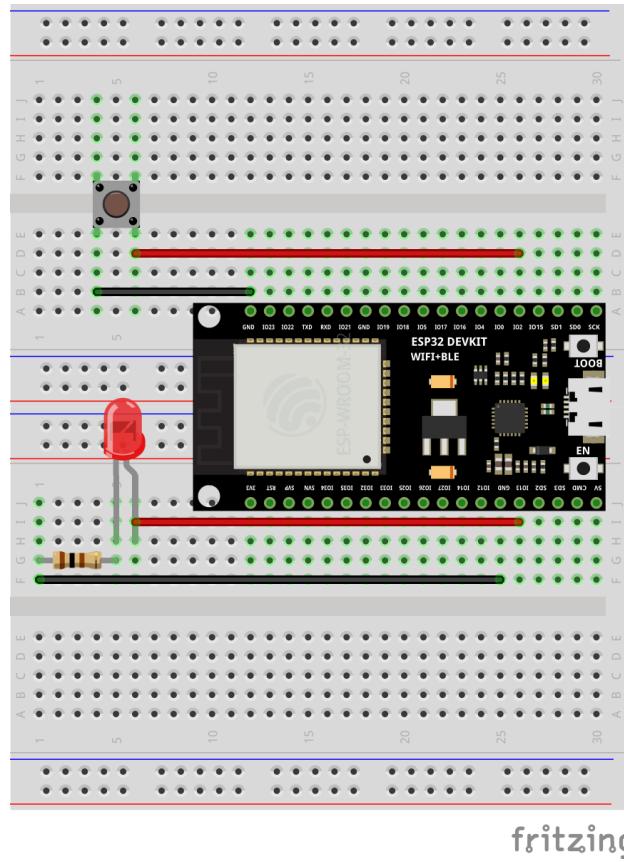


図 3.3: switch2

リスト 3.2: switch

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

void loop()
{
    if (digitalRead(2) == LOW)
    {
        delay(200);
        Serial.println("ON!");
    }
}
```

```
if (digitalRead(2) == HIGH)
{
    delay(200);
    Serial.println("OFF!");
}
```

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]

応用問題

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

第4章

取得データを Web に公開しよう！



図 4.1: 6

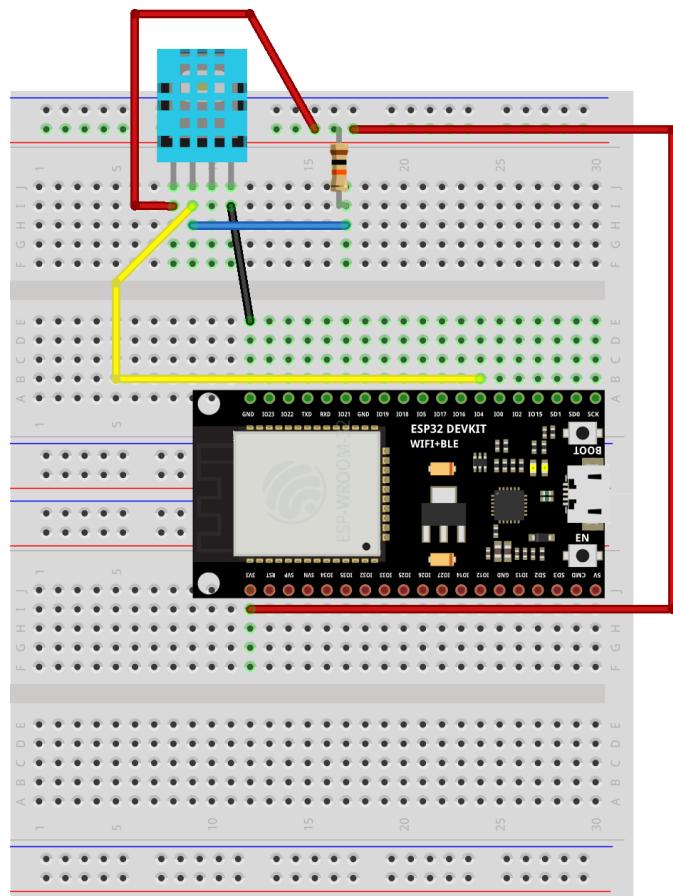
4.1 センサーを使おう

I2C とは

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/>



fritzing

図 4.2: dht11

LCD

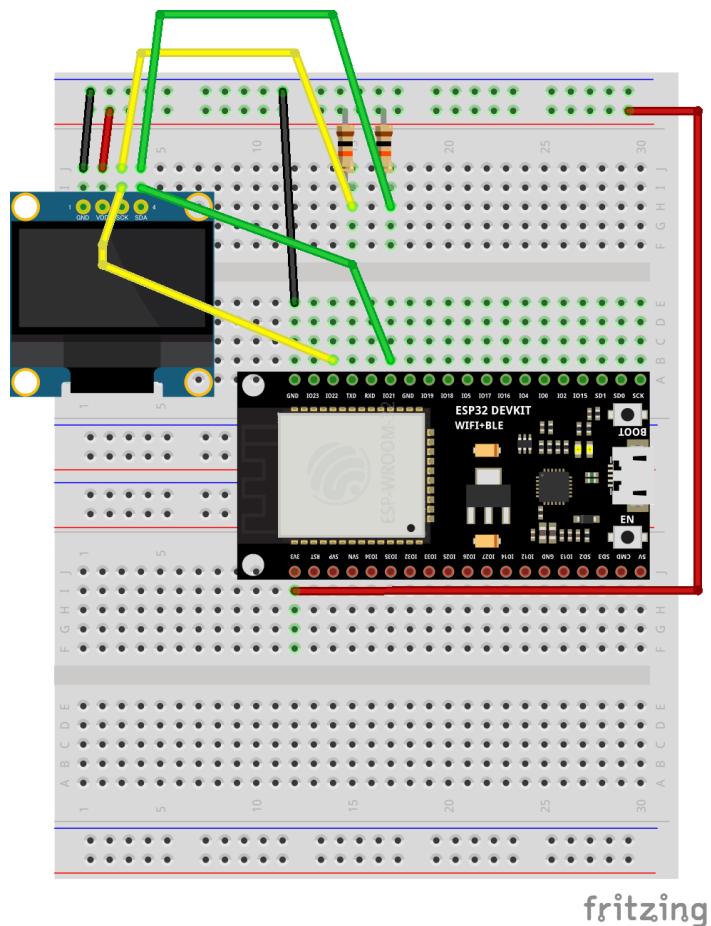


図 4.3: oled

第4章 取得データを Web に公開しよう！

4.1 センサーを使おう

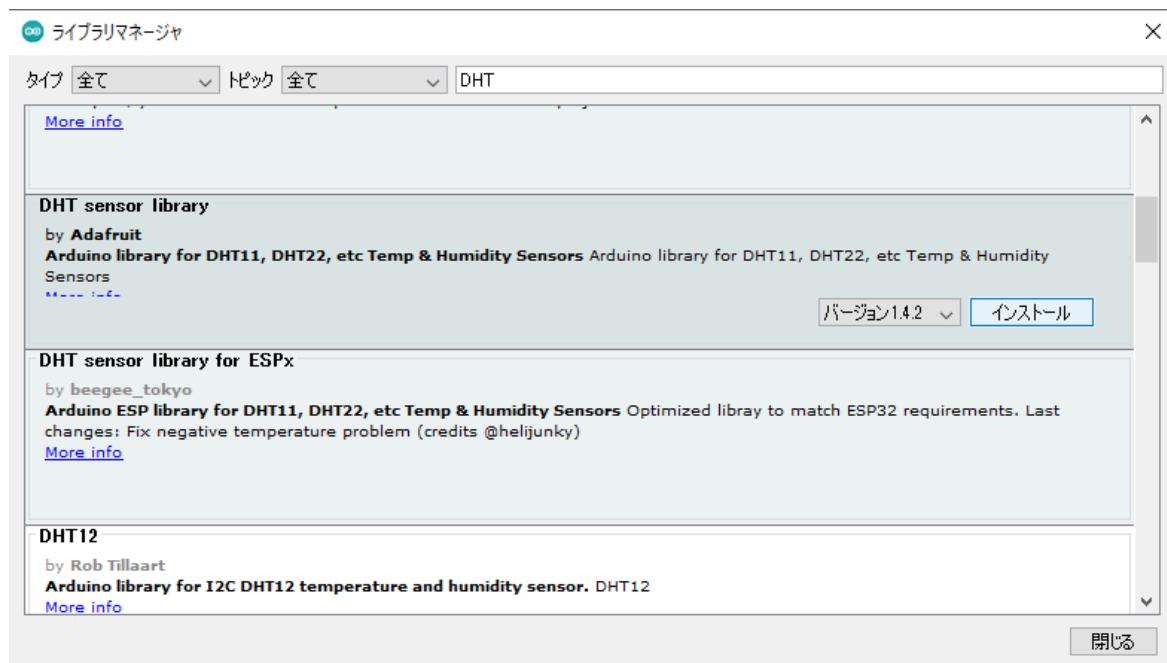


図 4.4: 9

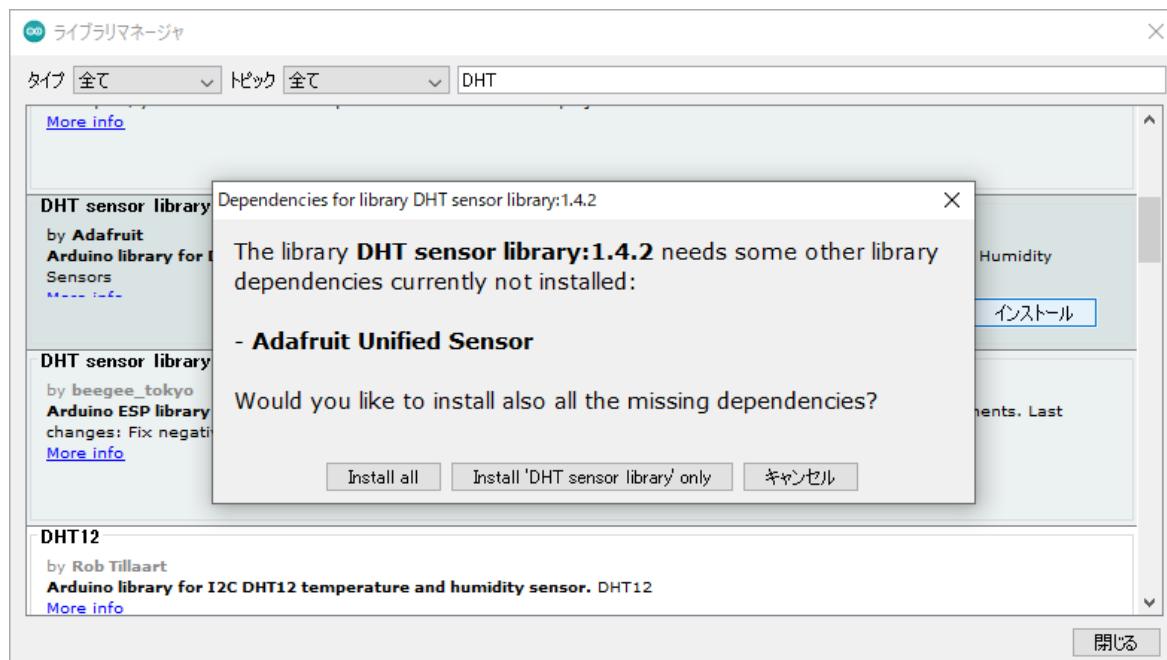


図 4.5: 8

リスト 4.1: dht11

```
// Example testing sketch for various DHT humidity/temperature sensors written
// 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
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster processors.
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 4.2: 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  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F  
Humidity: 57.00% Temperature: 24.50 °C 76.10 °  
F Heat index: 24.49 °C 76.09 °F
```

4.2 Web に公開しよう

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/>

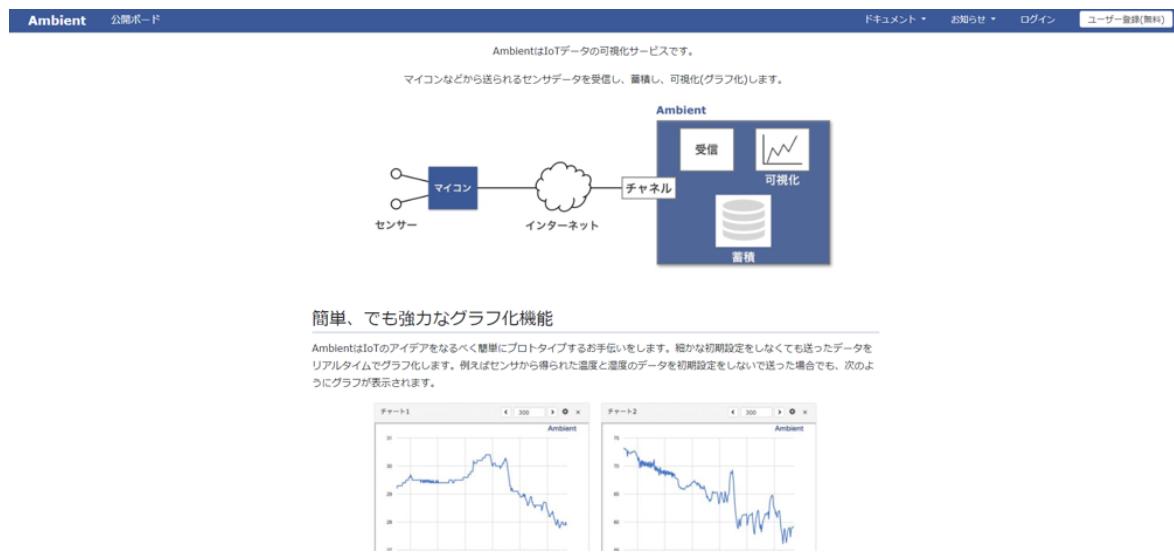


図 4.6: 1

The screenshot shows the user registration form for Ambient. It includes fields for 'メールアドレス' (Email Address), 'パスワード' (Password), and 'パスワード再入力' (Password Confirmation). Below these fields is a button labeled 'ユーザー登録(無料)' (User Registration (Free)) with the note: '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。' At the bottom, there are links for '利用規約' (Terms of Use) and '会社概要' (Company Profile).

図 4.7: 2

第4章 取得データをWebに公開しよう！

4.2 Webに公開しよう



図 4.8: 3



図 4.9: 4

チャネルを作成します。



図 4.10: 5

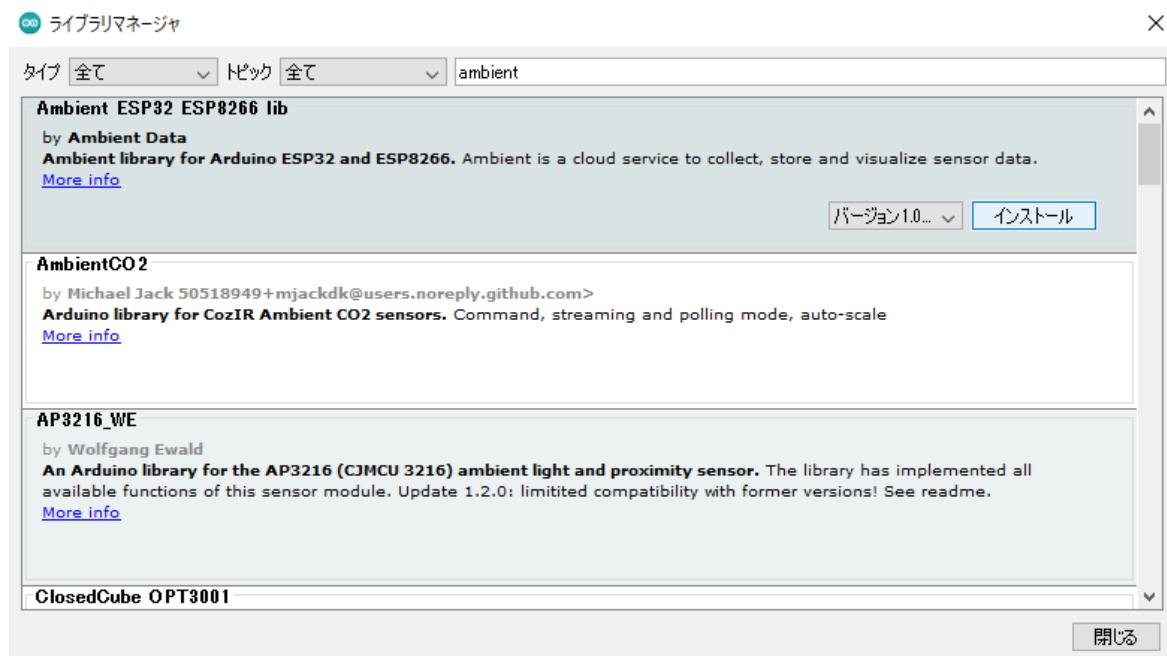


図 4.11: 7

ライブラリのインストール

回路図

コーディング

リスト 4.3: ambient

```
#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にデーターを送信

delay(PERIOD * 1000);
}
```

第5章

API を使おう！

5.1 Weather API を使う

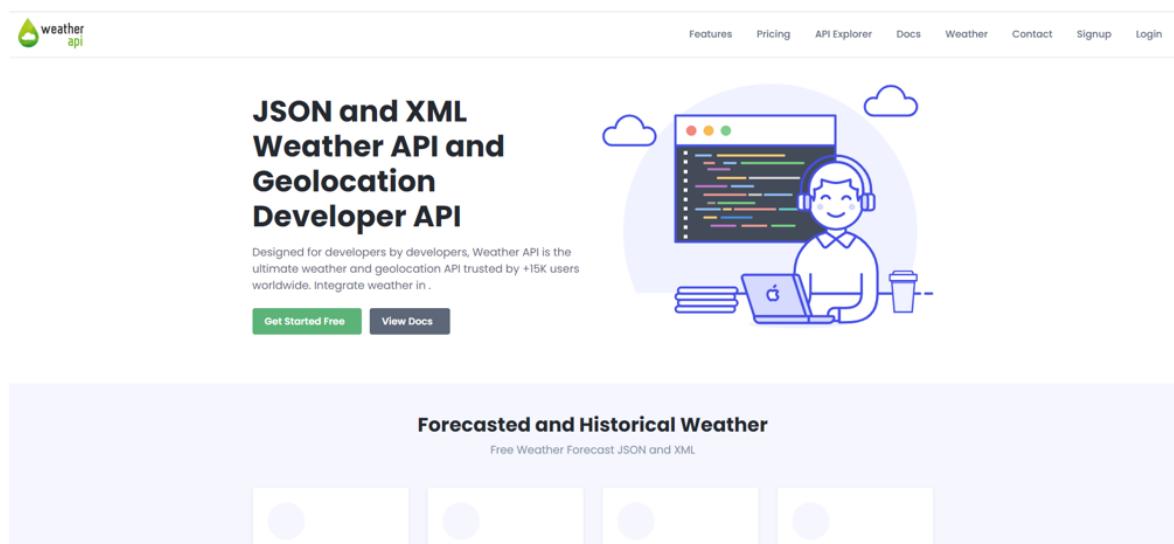


図 5.1: 1

第5章 API を使おう！

5.1 Weather API を使う

The screenshot shows the 'Sign Up' form on the Weather API website. The form includes fields for Email, Password, and Confirm Password. There are several validation messages in red: 'Email is required.', 'Password cannot be left blank!', and 'Confirm Password cannot be left blank!'. A reCAPTCHA field is present with the message 'I'm not a robot'. Below the form is a note about reading T&C's and Privacy Policy, and a green 'Sign up' button.

図 5.2: 2

The screenshot shows the 'Login' form on the Weather API website. It features fields for User Name and Password, and a 'Remember me next time' checkbox. Below the form is a blue banner with the text 'Quick and Easy Signup for Free Weather API' and a 'Get Started' button. The banner also contains a brief description of the service and a 'Contact us' button.

図 5.3: 3

第5章 API を使おう！

5.1 Weather API を使う

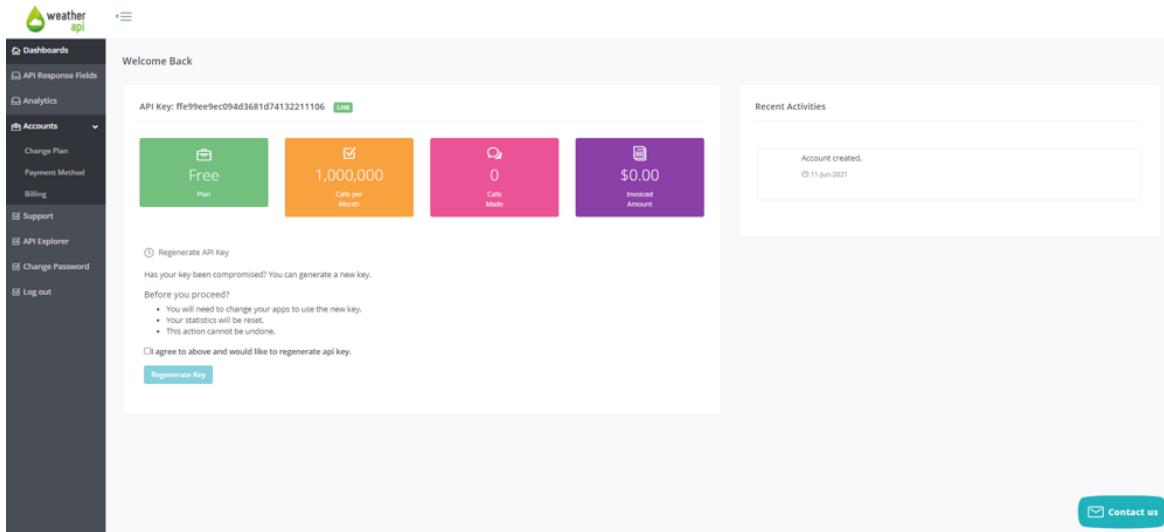


図 5.4: 4

The screenshot shows the Weather API Interactive API Explorer. The top navigation bar includes links for 'Features', 'Pricing', 'API Explorer' (selected), 'Docs', 'Weather', 'Contact', and 'My Account'. The main section is titled 'Interactive API Explorer' and contains a note about the tool's purpose. It features input fields for 'Your API Key' (set to 'ff69ee9ec094d3681d7413221106'), 'Protocol' (set to 'HTTPS'), and 'Format' (set to '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 degrees) or city name. Visit request parameter section to learn more.

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

At the bottom right is a 'Contact us' button.

図 5.5: 5

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)
ResponseCode 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",  
        "temperature": 25.5,  
        "humidity": 65,  
        "wind": 10,  
        "pressure": 1013,  
        "clouds": 30,  
        "rain": 0,  
        "forecast": [{"date": "2021-07-13", "temp": 26, "precip": 0},  
                    {"date": "2021-07-14", "temp": 27, "precip": 0},  
                    {"date": "2021-07-15", "temp": 28, "precip": 0},  
                    {"date": "2021-07-16", "temp": 29, "precip": 0},  
                    {"date": "2021-07-17", "temp": 30, "precip": 0}],  
        "weather": "Partly Cloudy",  
        "atmosphere": "Normal",  
        "air_quality": 50  
    }  
}
```

```
"temp_c": 29.4,  
"temp_f": 84.9,  
"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

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 23:58:00"

JsonObject current = doc["current"];
long current_last_updated_epoch = current["last_updated_epoch"]; // 1626533120
const char* current_last_updated = current["last_updated"]; // "2021-07-17 23:58:00"
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
```

API とは？

コラム: サーバクライアント
サーバ? クライアント? とは何

Web サーバからのレスポンス

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

6.3 VScode から ESP32 にスケッチを書き込む

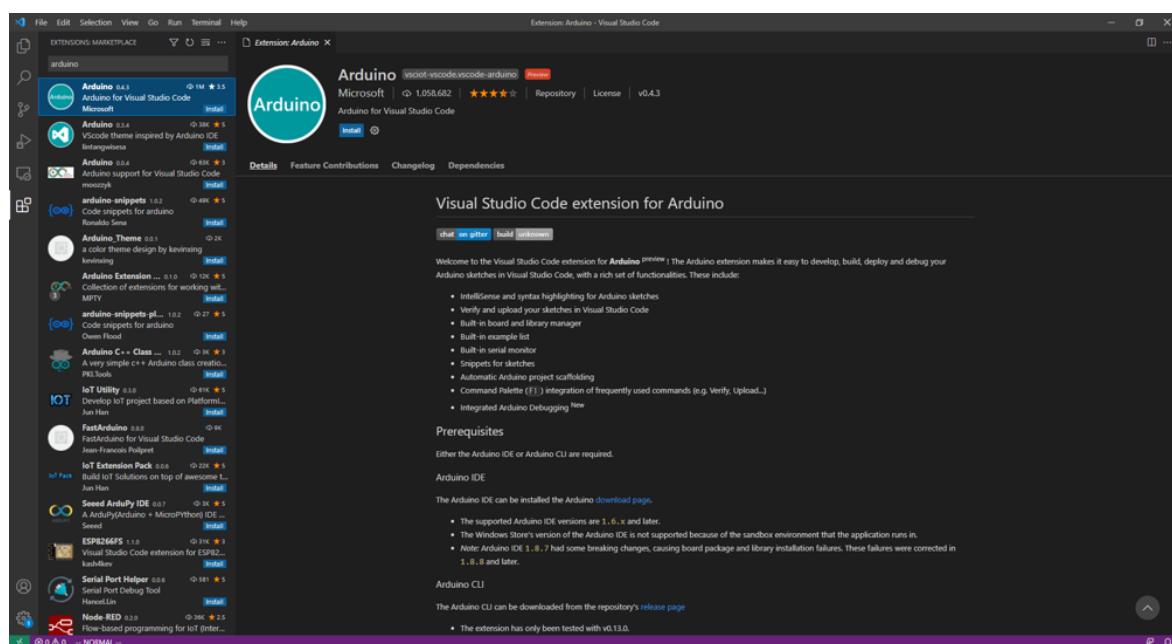


図 6.1: 1

コマンドパレット `ctrl shift P`

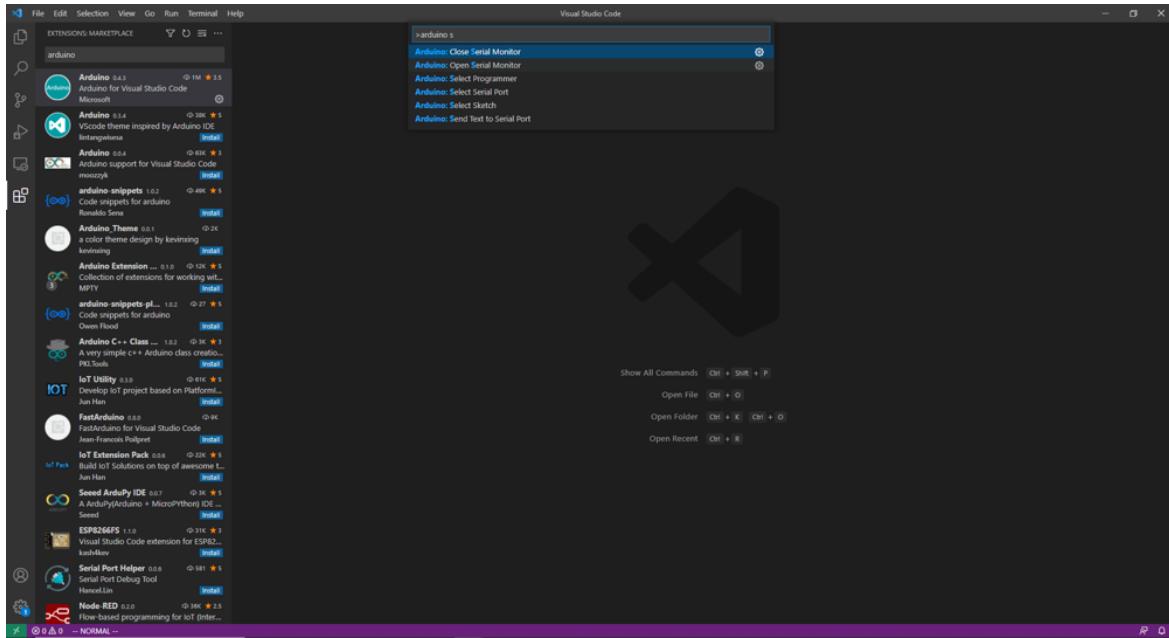


図 6.2: 2

arduino serial

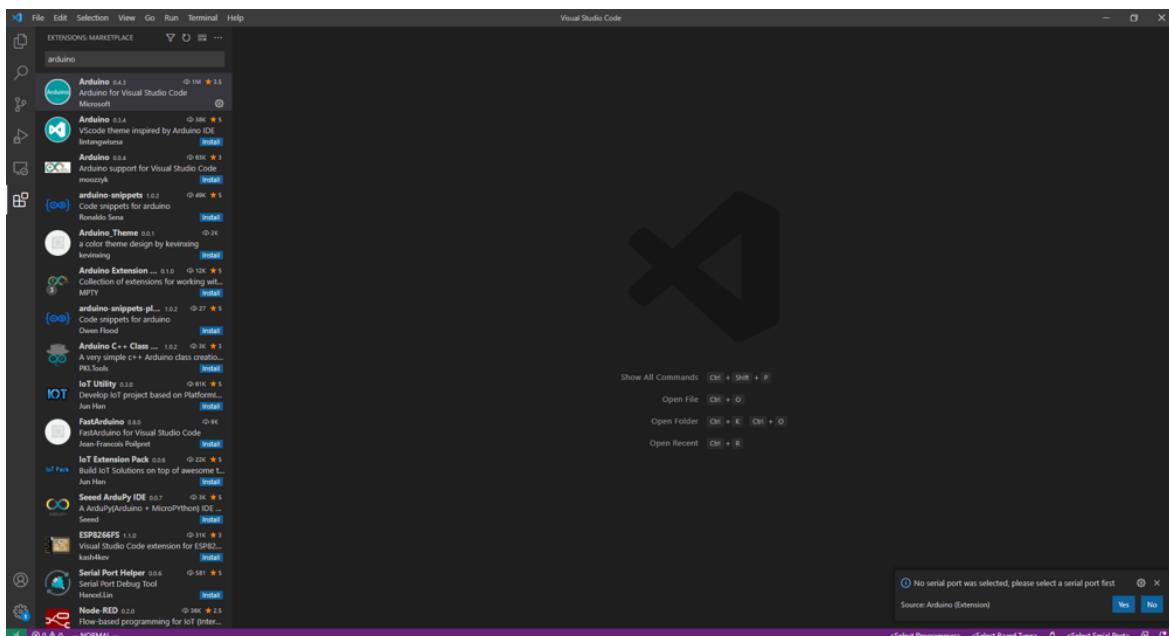


図 6.3: 3

/mnt/c/Users/Document/Arduino

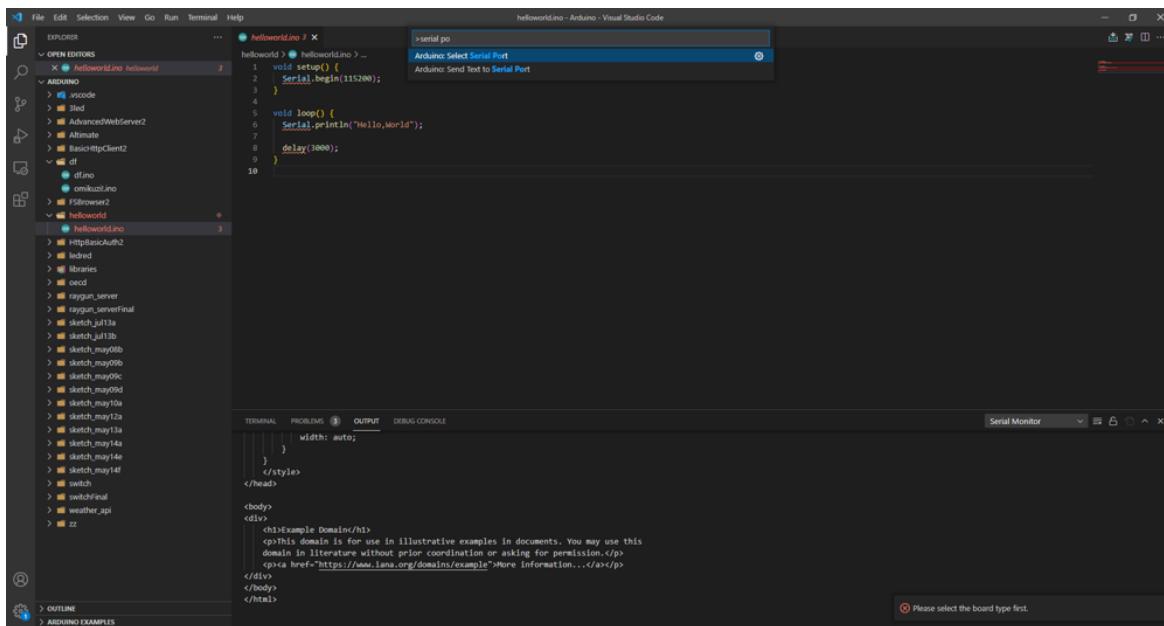


図 6.4: 4

serial port 選択

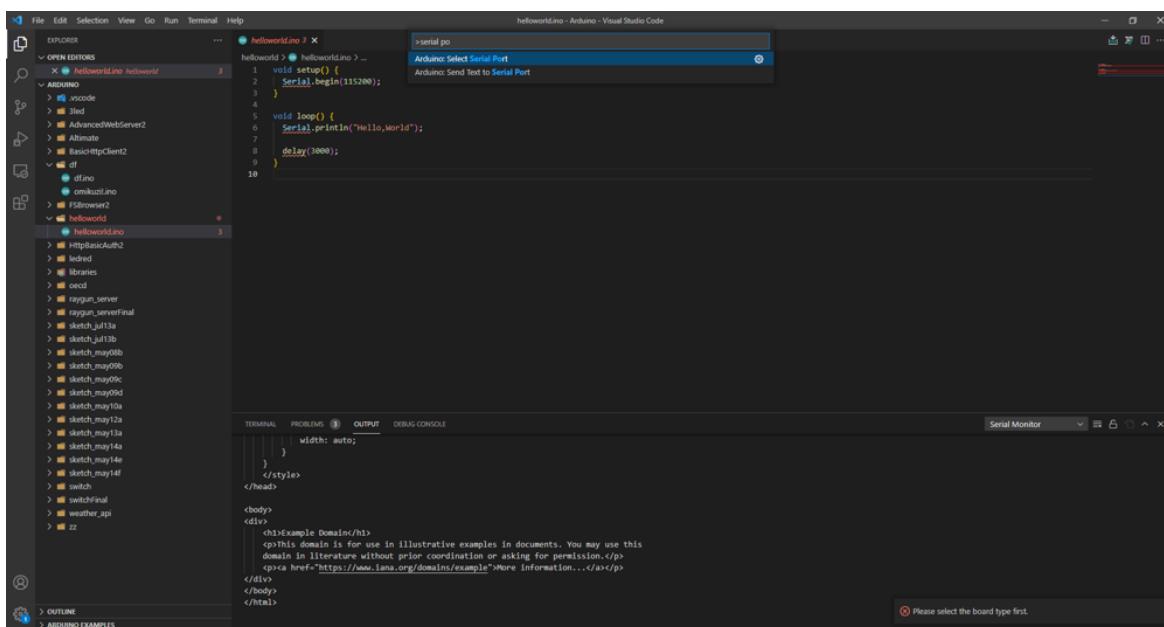


図 6.5: 5

ポート選択

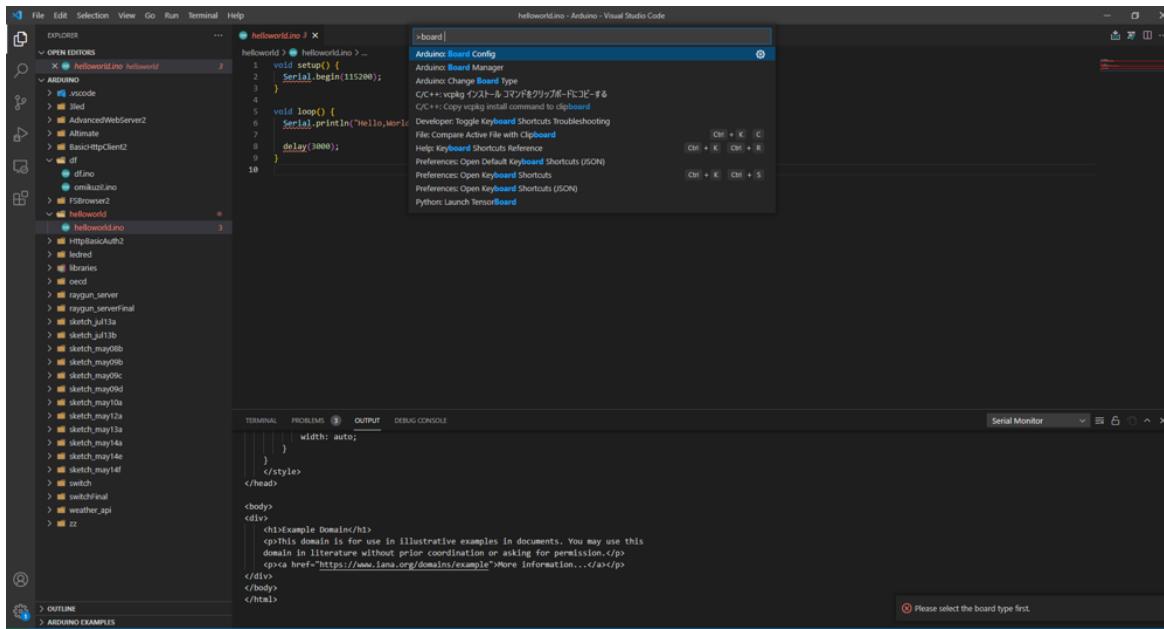


図 6.6: 6

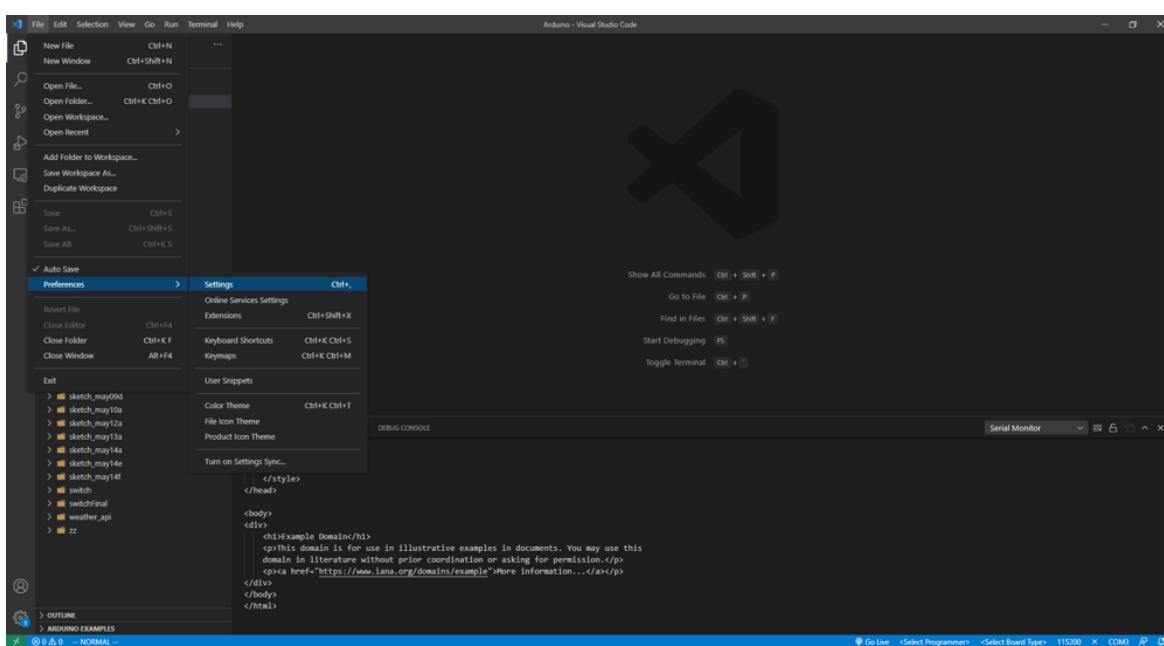


図 6.7: 7

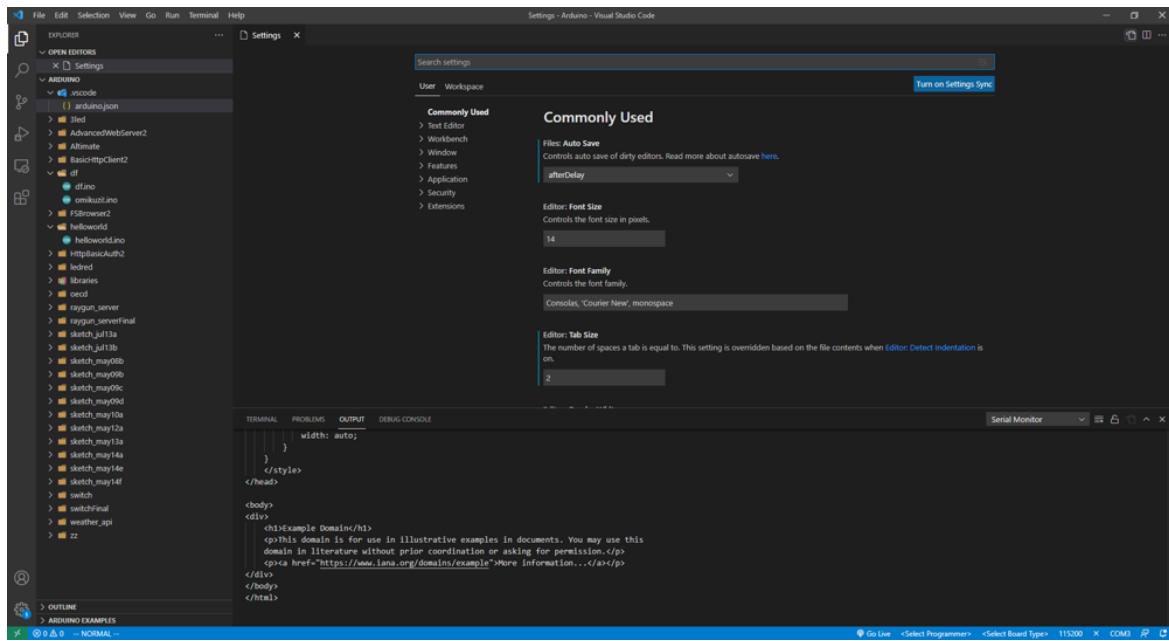


図 6.8: 8

リスト 6.1: json

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

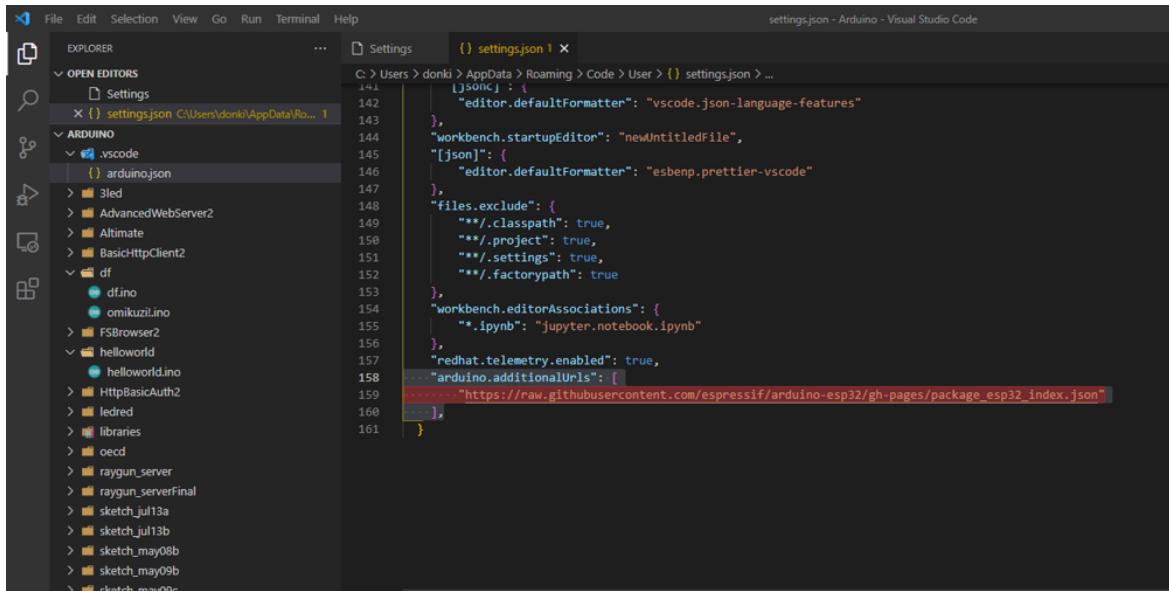


図 6.9: 9

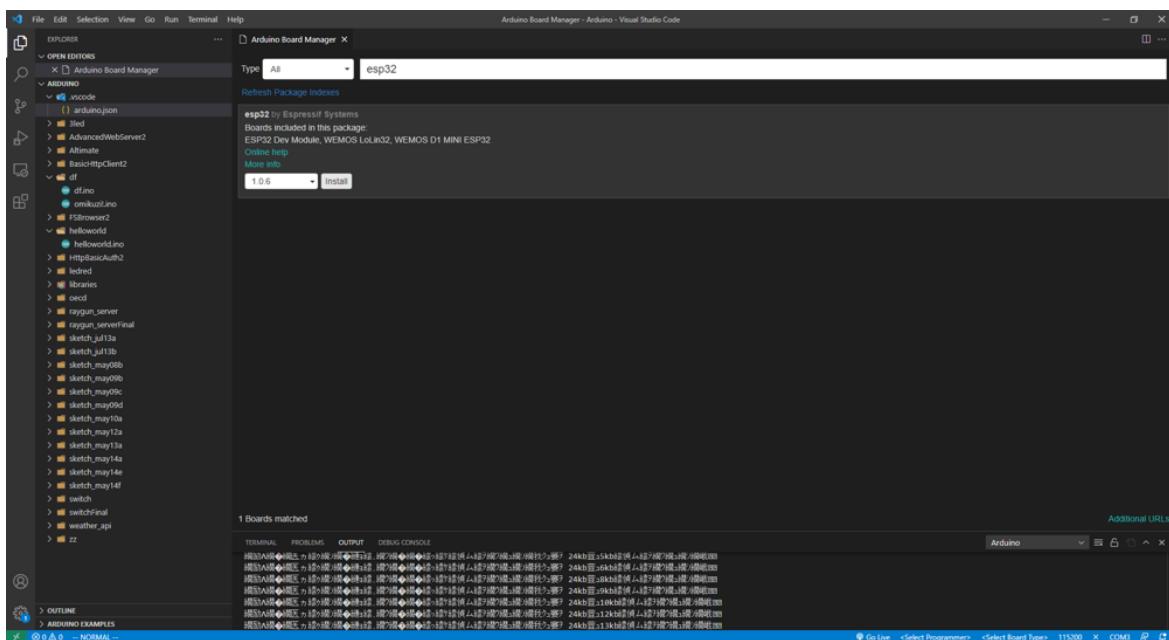


図 6.10: 10

arduino bord maneger

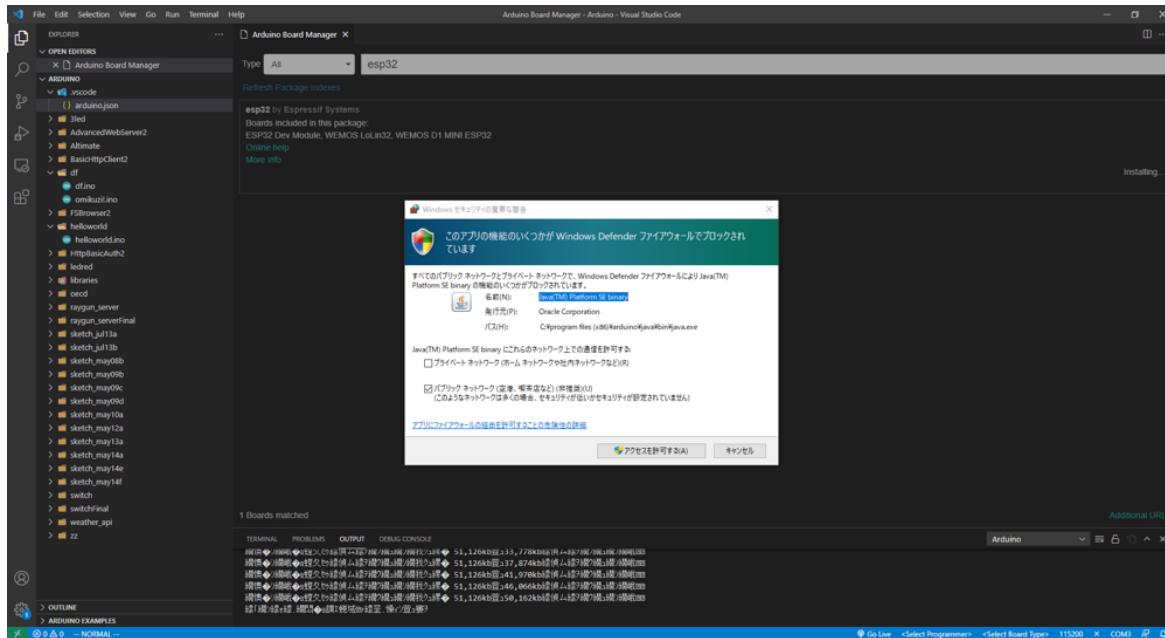


図 6.11: 11

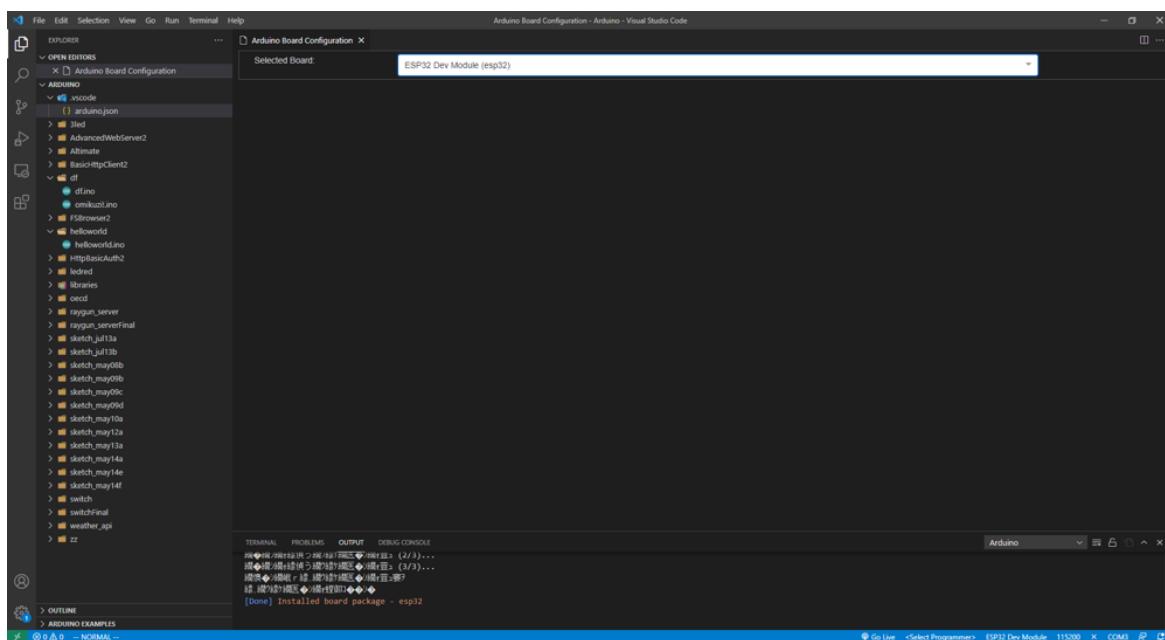


図 6.12: 12

第6章 応用編

6.3 VScode から ESP32 にスケッチを書き込む

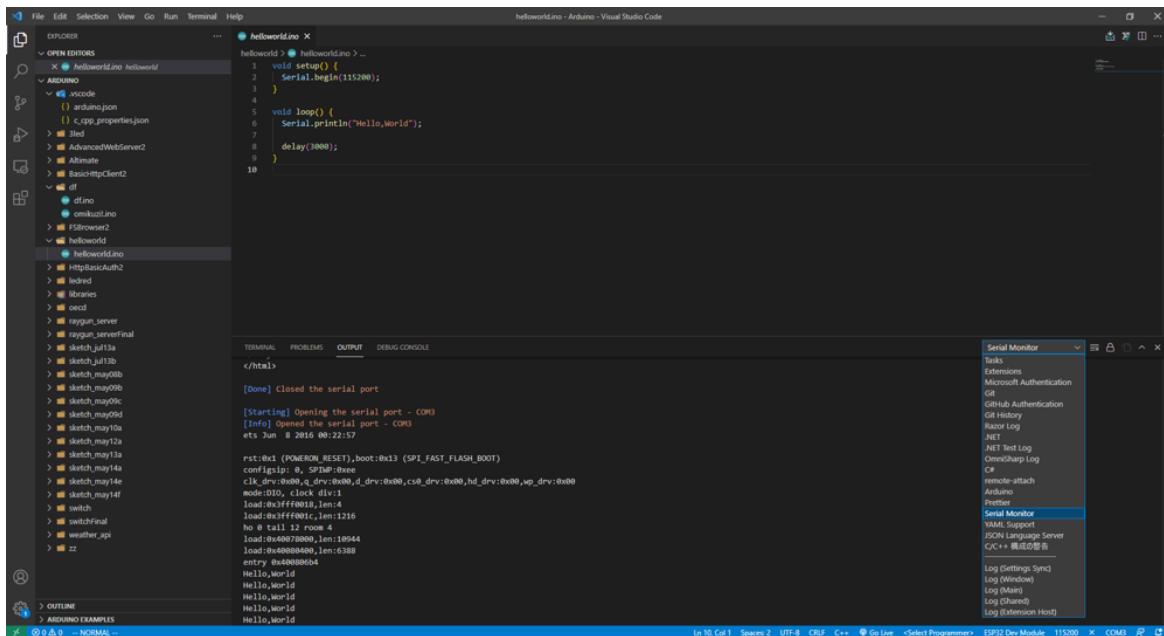


図 6.13: 13

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/introdu>

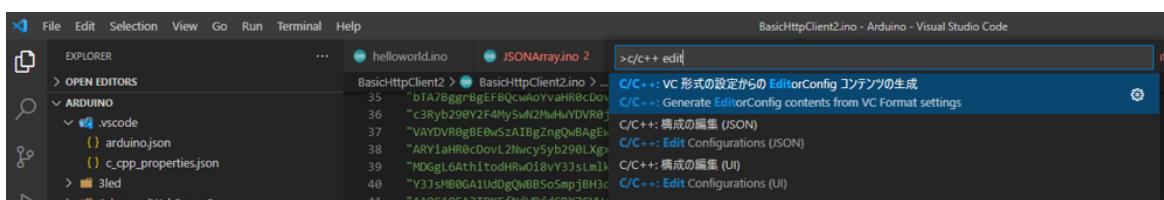


図 6.14: 14

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



図 A.1: 1



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

付録A トラブルシューティングプログラムを書き込んだが動作に反映されない

A.3 プログラムを書き込んだが動作に反映されない



```
helloworld | Arduino 1.8.15 (Windows Store 1.8.49.0)
ファイル 編集 スケッチ ツール ヘルプ

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 '
```

```
DHT11:30:6: error: redefinition of 'void setup()'
void setup() {
^

c:\Users\donki\Documents\Arduino\helloworld\helloworld.ino:1:6: no
void setup() {
^

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

c:\Users\donki\Documents\Arduino\helloworld\helloworld.ino:5:6: no
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きました。

はじめての IoT 講座

2021 年 7 月 12 日 初版第 1 刷 発行

著 者 THEToilet
