

SAWACHI エンジニア養成講座（デバイス編）

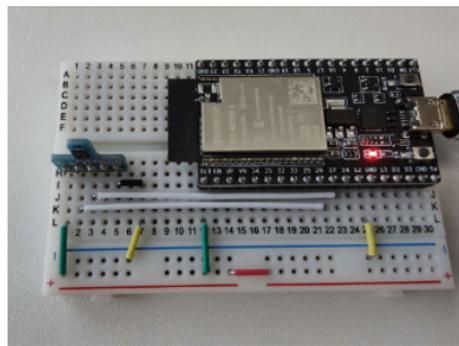
2025年6月27日
IoP技術者コミュニティ

1. 概要

1.1. 本講座の全容

温湿度データを計測するデバイスを製作し、SAWACHI を経由してデータ取得とグラフ化を行います。

温度データを計測 & 蓄積



温度データを取得 & グラフ化



図 1: SAWACHI を経由して温湿度データを取得し、グラフ化する



図 2: データを表示するダッシュボード「詳細分析画面」

1.2. SAWACHI の仕組み

SAWACHI には用途別に 2 つの API が用意されています

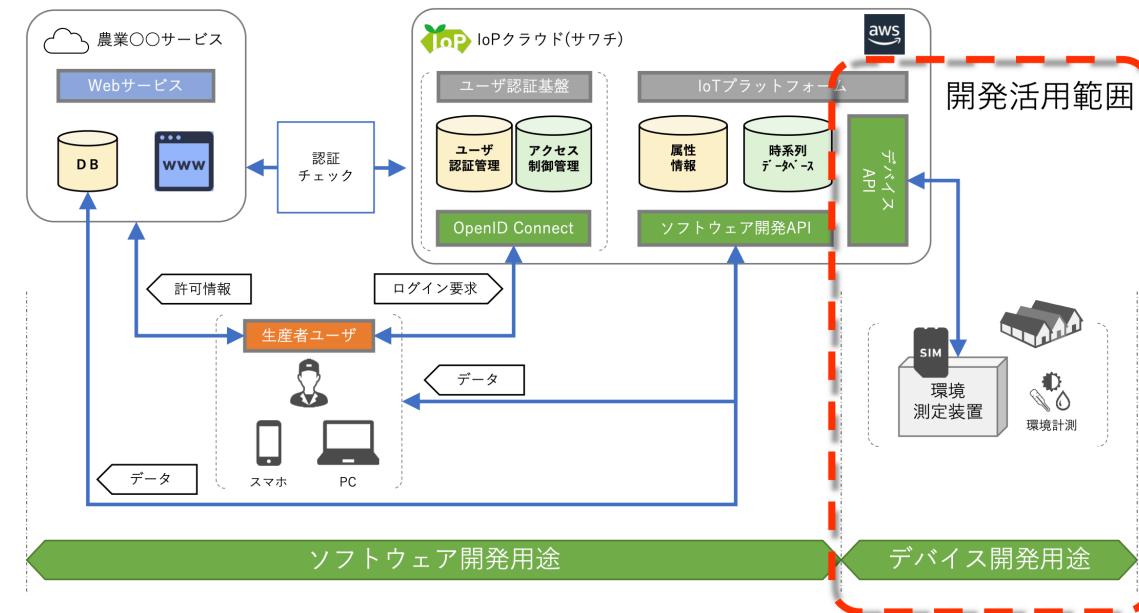


図 3: IoP クラウド連携における開発範囲
出典: 【配布用】SAWACHI 技術資料ライブラリ API 編
30_SAWACHI デバイス API を活用した開発手法説明

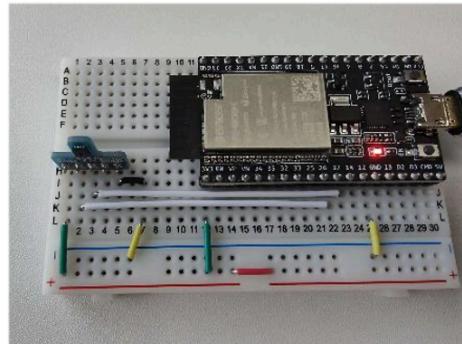
補足:

API は、Application Programming Interface の略で、ソフトウェア同士が通信するためのインターフェースです。インターフェースとは、接点や境界を意味します。

1.3. 本講座の内容

MQTT と呼ばれる通信方式で温湿度データを送信します

制作する温湿度計測デバイス

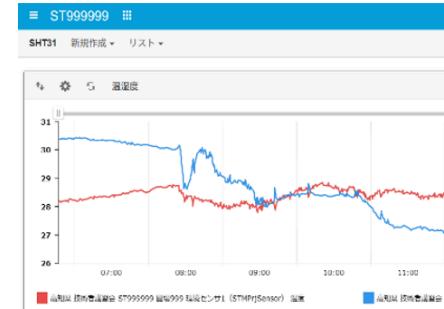


温湿度データ送信
MQTT

SAWACHI



温湿度データの確認



※SAWACHIに搭載されているグラフアプリ

図 4: 温湿度データの送信は MQTT を使用します

1.4. 全体手順図

講座の前半でデバイスの製作を行い、後半に SAWACHI に接続します

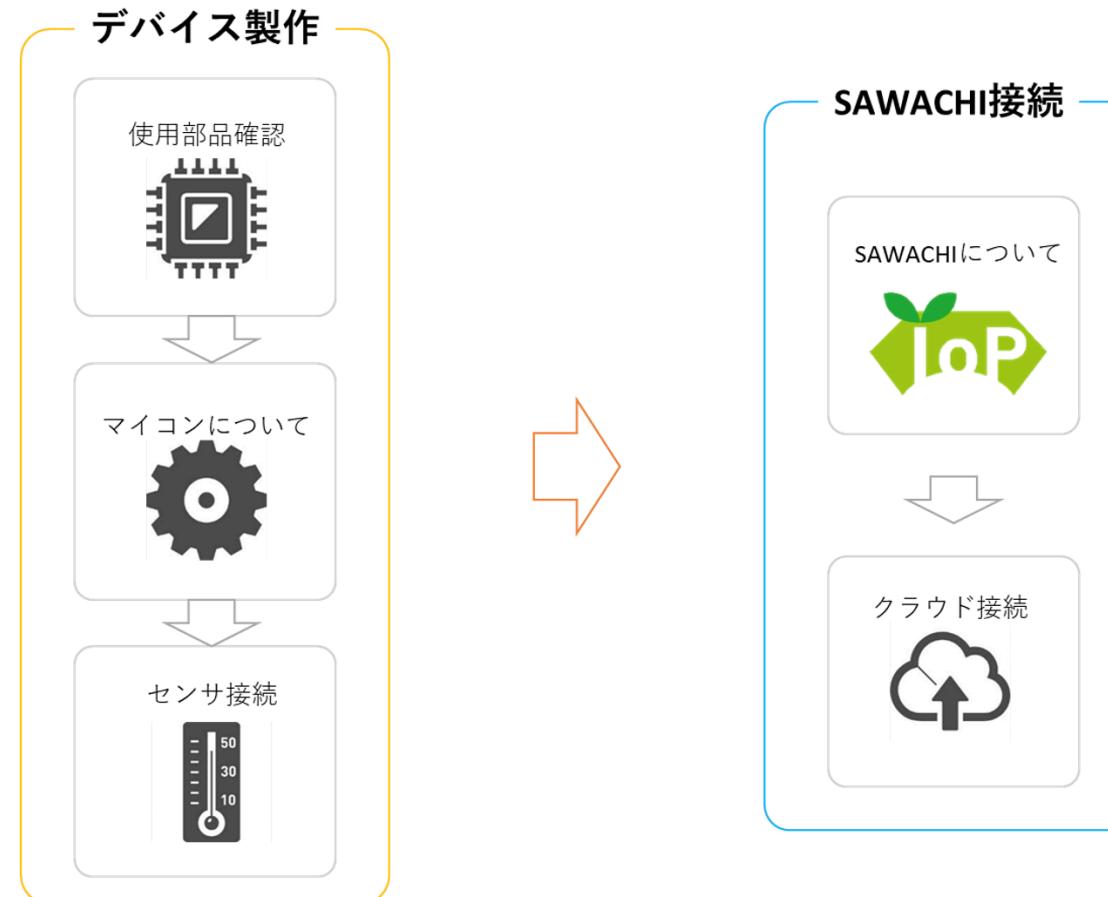


図 5: 講座の全体手順図

1.5. 本講座の難易度

後半にかけて難易度が高く感じられるかもしれません

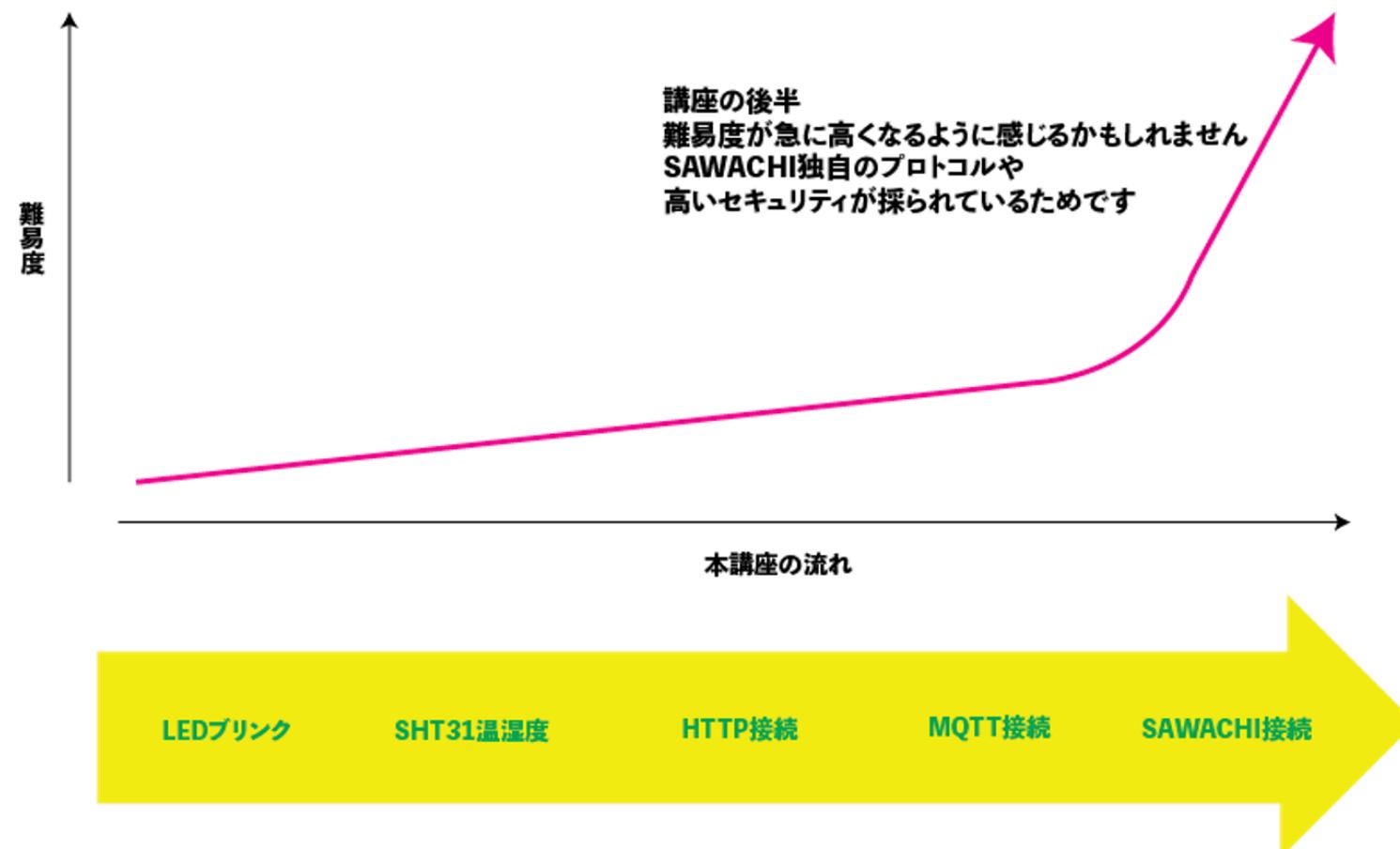


図 6: 講座の難易度

事前準備は完了していますか？

[事前準備資料](#)

1.7. マイコンについて

1.7.1. マイコンとは

マイコンは、マイクロコントローラの略称で、コンピュータの機能を持つ小型の集積回路です。主に組み込みシステムや IoT デバイスで使用されます。

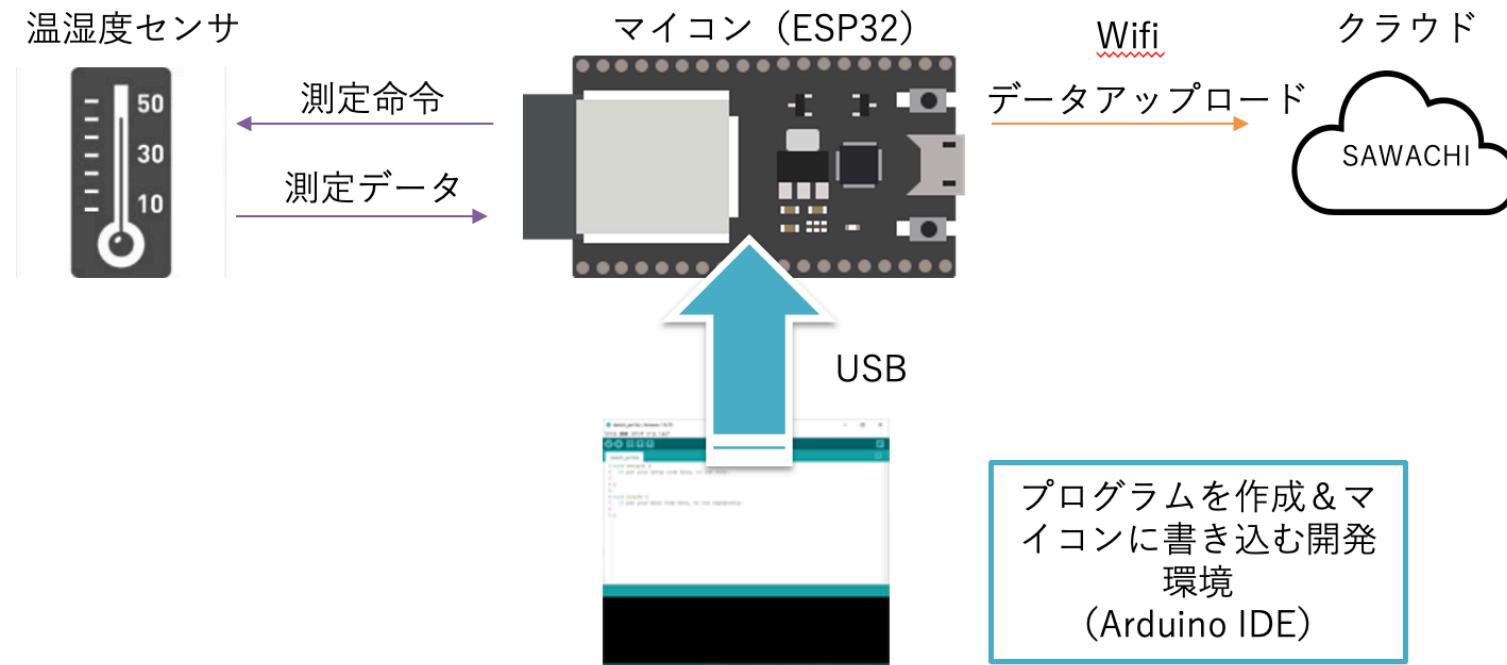


図 7: 本講座でのマイコンの役割

1.8. Arduino と ESP32

Arduino は専門家でなくても扱いやすいマイコンです。 Arduino IDE を使用して、簡単にプログラムを書いて動かすことができます。

ESP32 は、Wi-Fi や Bluetooth 機能を備えた Arduino よりさらに高性能なマイコンです。
ESP32 は Arduino IDE でプログラムできます

Arduino	 
ESP32	 <ul style="list-style-type: none">・低消費電力・Wi-Fi、Bluetooth内蔵・Arduino互換（Arduino IDEが使用可能）・商用利用を想定して製作されている・約1800円

図 8: Arduino と ESP32

1.9. Arduino IDE とは

Arduino IDE は、Arduino や ESP32 のプログラムを書くための統合開発環境です。

プログラムを書いて、マイコンにアップロードすることができます。

C 言語に似た Arduino 言語を使用して、マイコンを制御するプログラムを書くことができます。

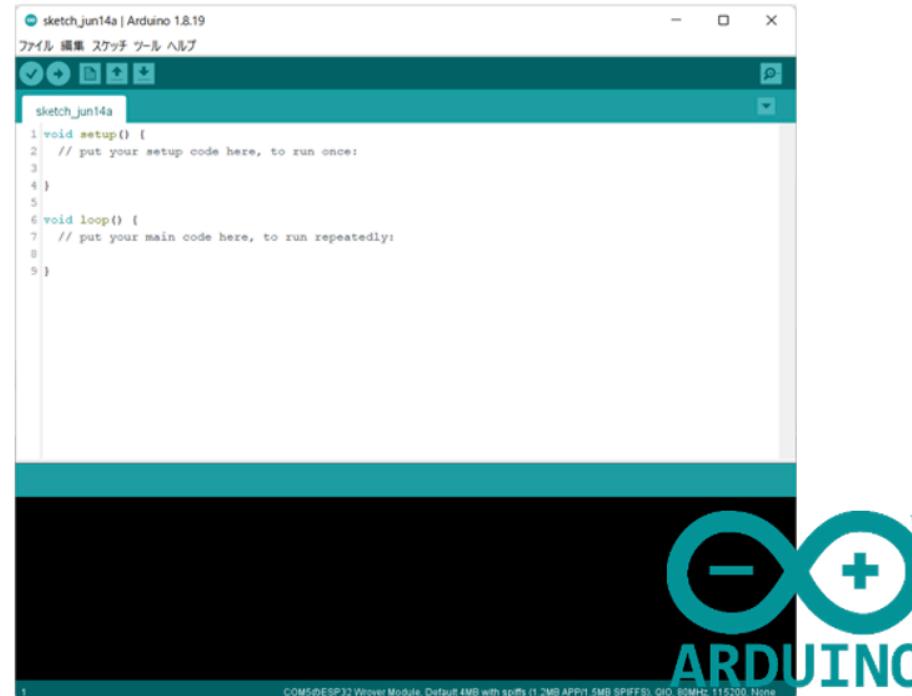


図 9: Arduino IDE の画面

1.10. ESP32 を使用した LED 点灯

1.10.1. ESP32 を使用した LED 点灯

Arduino IDE を使用して、ESP32 の GPIO ピンを制御し、LED を点灯させるプログラムを書きます。

GPIO ピン General Purpose Input/Output の略で、マイコンの入出力ピンのことです。

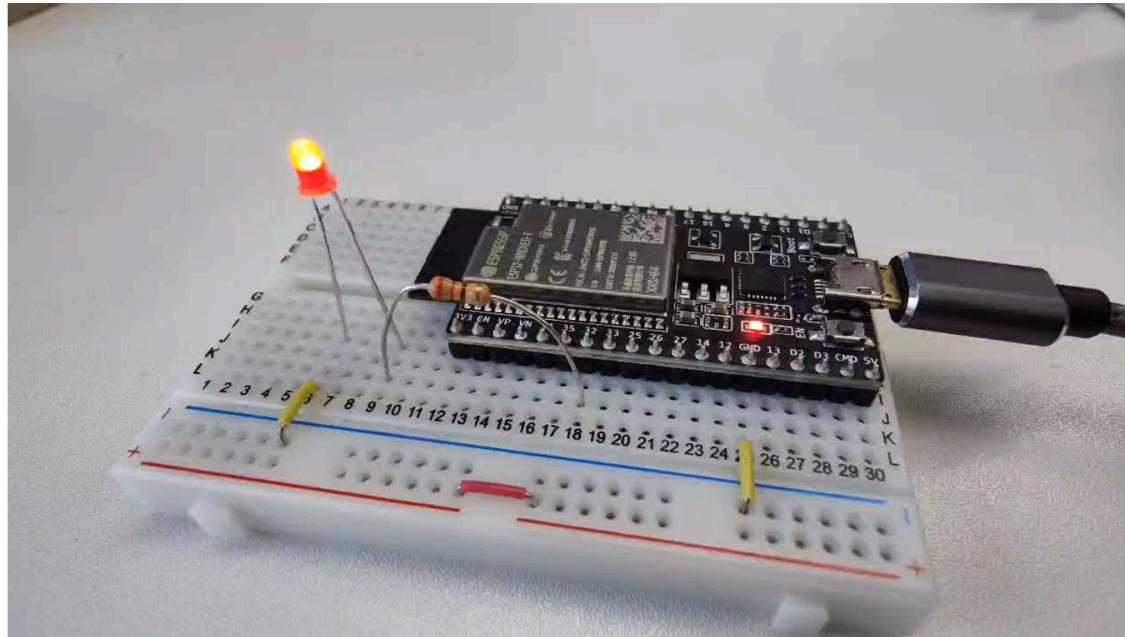


図 10: LED が点灯している様子 (通称 L チカ)

[LED 点灯のサンプルコードはこちら](#)

1.11. ブレッドボードについて

1.11.1. ブレッドボードについて

ブレッドボードは、電子回路を組み立てるためのプロトタイピングボードです。

はんだ付けせずに部品を差し込むだけで回路を組むことができます。
GPIO ピンを使用して、LED やセンサーなどの部品を接続するのに便利です。

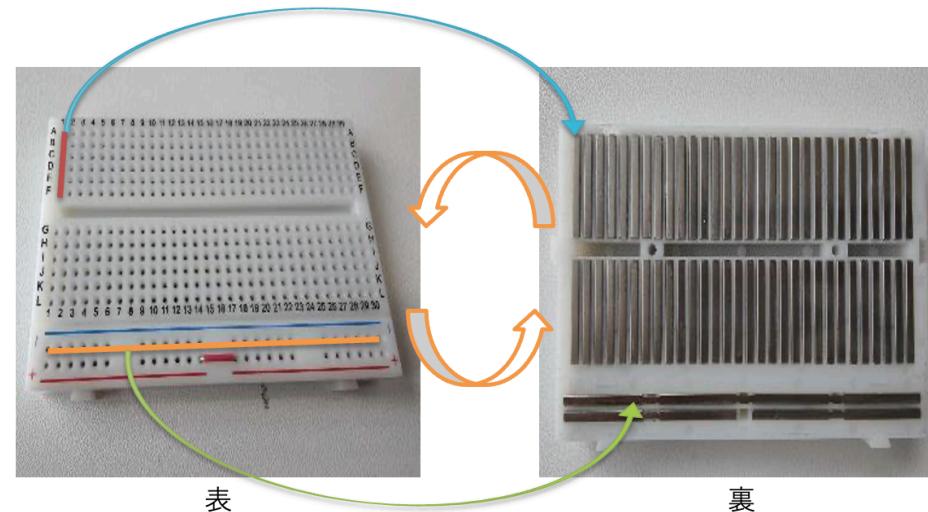


図 11: ブレッドボード: 背面を見ると、内部の配線がわかります。
同じ番号の A-F 列、G-L 列はそれぞれ内部で導通しています。
+, - は電源用の接続です。横一列に導通しています。

1.12. 使用する部品

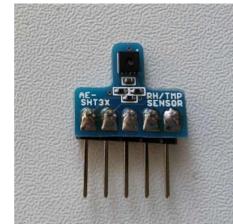
1.12.1. 使用する部品

本講座では、以下の部品を使用します。

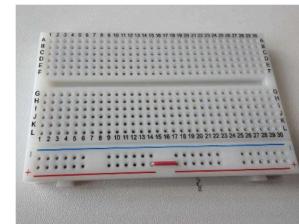
- ESP32-WROVER-E
- 温湿度センサー (AE-SHT31)
- ブレッドボード
- 抵抗
- LED
- USB ケーブル (Type-A)
- ジャンパー線



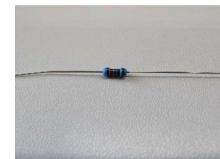
ESP32-WROVER-E
マイコン搭載の開発キット



AE-SHT31
高精度温湿度センサ



ブレッドボード
電子回路試作基板



カーボン被膜抵抗
150Ω～1kΩ
の被膜抵抗



赤色LED



シリアル通信用
Micro USBケーブル



ジャンパー線
ブレッドボード
結線用線

図 12: 使用する部品

1.13. ブレッドボードの番号

ブレッドボードの穴には番号が振られています。
行は A から L まで、列は 1 から 30 までです。
アルファベットと数字を組み合わせて、各穴を特定します。

例) A-15, H-28

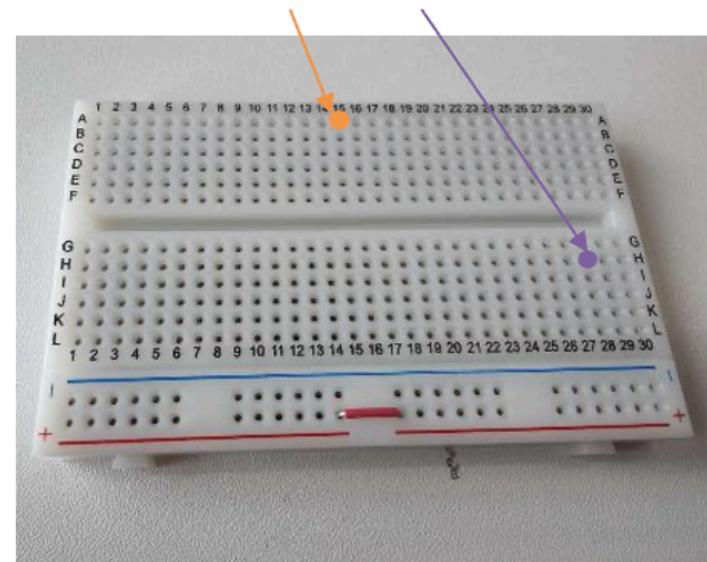


図 13: ブレッドボードの指定例

これ以降の結線図では、アルファベットと数字を組み合わせて位置を表現します。

2. LED 点灯

プログラム不要で LED を点灯させてみましょう

2.1. LED 点灯

まずプログラム不要で LED を点灯させる回路を組みます。ブレッドボードで、LED と抵抗を接続します。

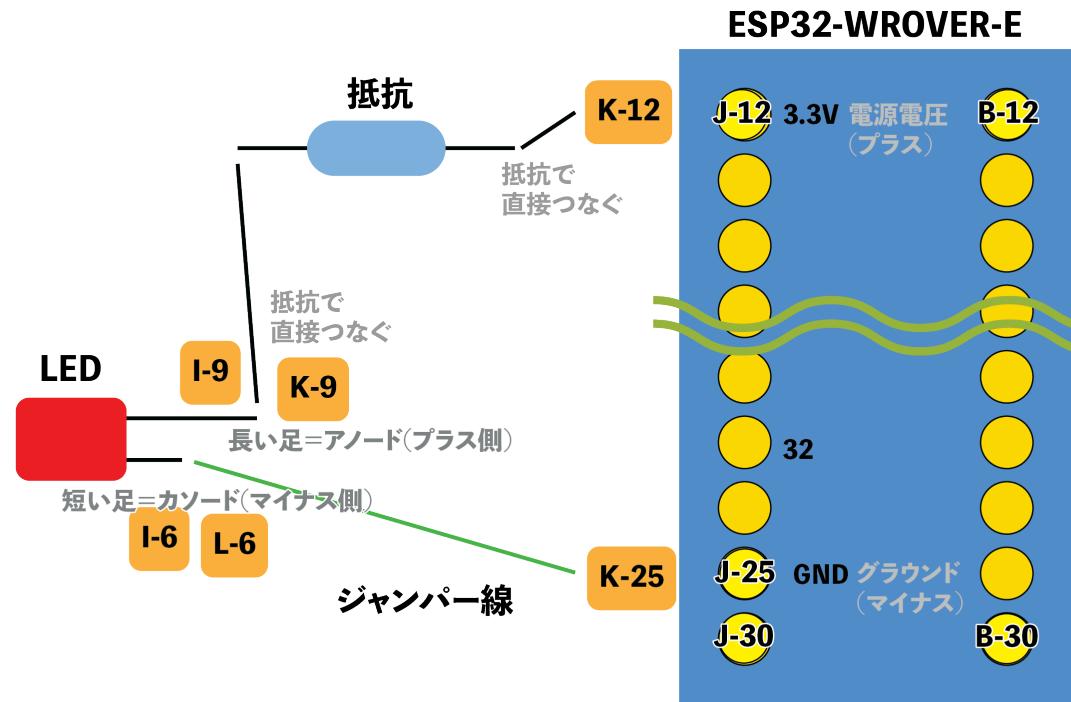


図 14: LED 点灯の結線図

LED の長い方の足がアノード、短い方の足がカソードです。

アノードはプラス側、カソードはマイナス側に接続します。

USB ケーブルを接続して、ESP32 に電源を供給すると LED が点灯します。

2.2. ブレッドボードへの取り付け

ブレッドボードに ESP32 を取り付ける際は、
向きをよく確認し、奥までしっかりと差し込みます。

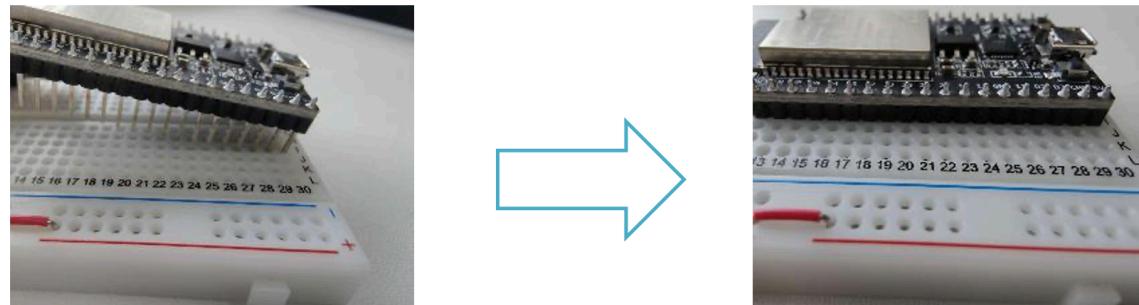


図 15: ブレッドボードに ESP32 を取り付ける

2.3. ブレッドボードの利点

ブレッドボードを使用する利点は、
はんだ付けせずに部品を差し込むだけで回路を組むことができる点です。
回路の変更や修正が容易で、試作や実験に適しています。
また、部品の交換も簡単に行えます。

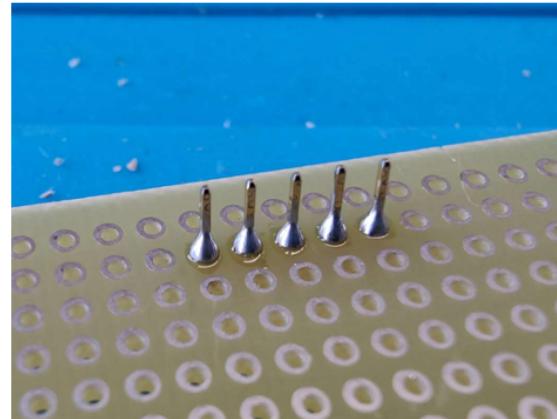


図 16: はんだ付け: 上手なはんだ付けには熟練の技術が必要です。
ブレッドボードの利点は、はんだ付けせずに部品を差し込むだけで
回路を組むことができる点です。

2.4. LED の点灯の様子

LED が点灯しましたか？

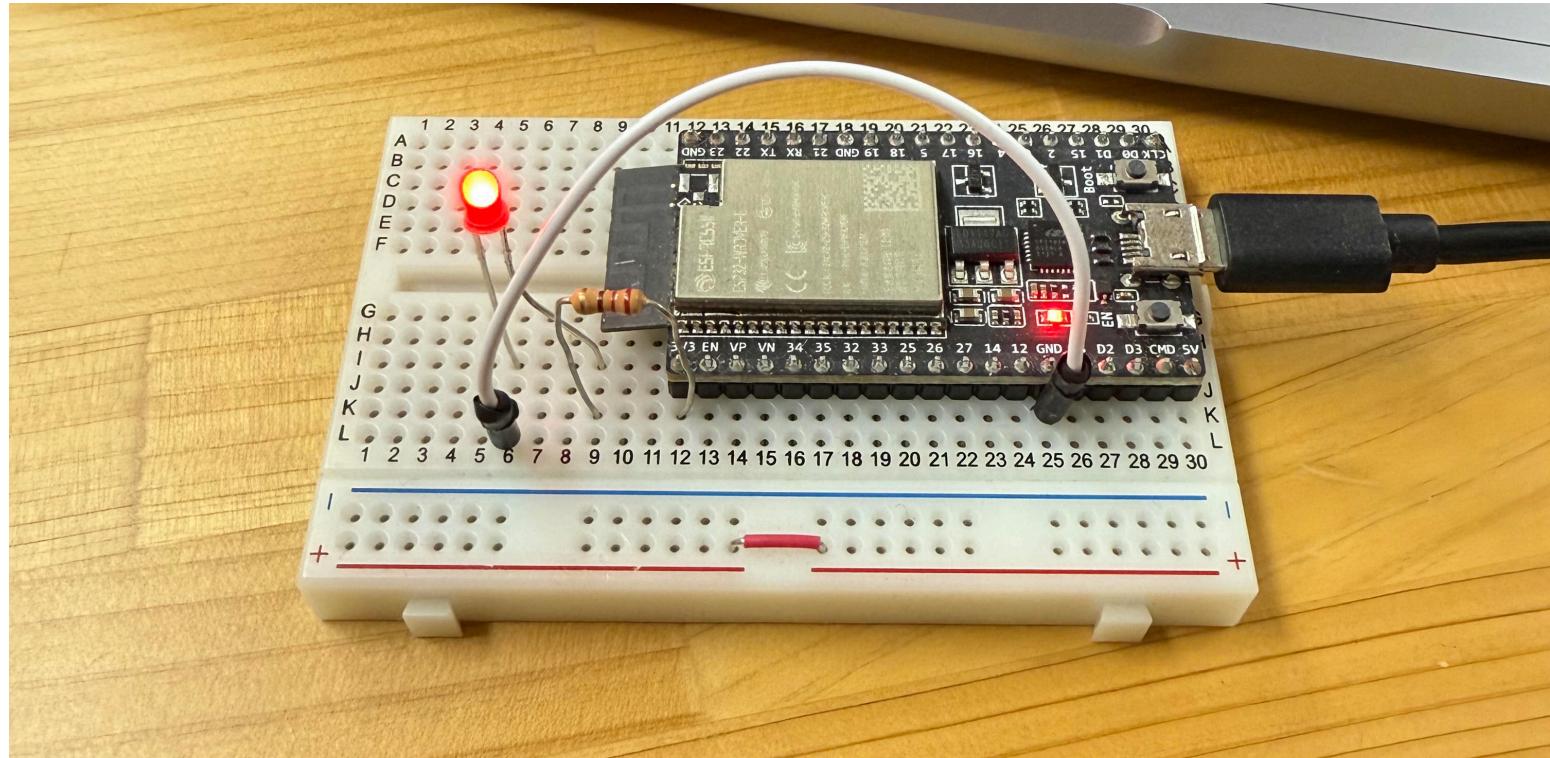


図 17: LED が点灯している様子

LED は点灯しましたか？

次はいよいよ **Arduino IDE** を使用して、
プログラムを書いて LED を点滅させます。

3. LED 点滅

プログラムを書いて LED を点滅させてみましょう

3.1. Arduino IDE のライセンスについて

3.1.1. GPL ライセンス v2.0

Arduino IDE は、オープンソースのソフトウェアであり、
GNU General Public License (GPL) バージョン 2 に基づいて配布されています。
GPL ライセンスは、**ソフトウェアの自由な使用、コピー、変更を許可しますが、**
ソフトウェアを再配布する場合は、同じライセンス条件を適用する必要があります。
(コピーレフト)

GPL ライセンスのソフトウェアにはソースコードの公開が求められます。

3.2. 様々なライセンス

今回使用するソフトウェアやライブラリそれぞれにライセンスがあります。

Arduino IDE GPL v2.0

espressif/arduino-esp32 LGPL v2.1

knolleary/pubsubclient MIT License

bblanchon/ArduinoJson MIT License

基本的に制約の強い規約に従ってライセンスが適用されます。

今回は GPL v2.0 が適用されることになります。

実際に商用利用で組み込み機器を開発する場合は、ライセンスの内容をよく確認する必要があります。

法的な問題を避けるために、専門家に相談することをお勧めします。

3.3. LED 点滅

Arduino IDE を使用して、ESP32 の GPIO ピンを制御し、LED を点滅させるプログラムを書きます。
下記のように結線を行います。

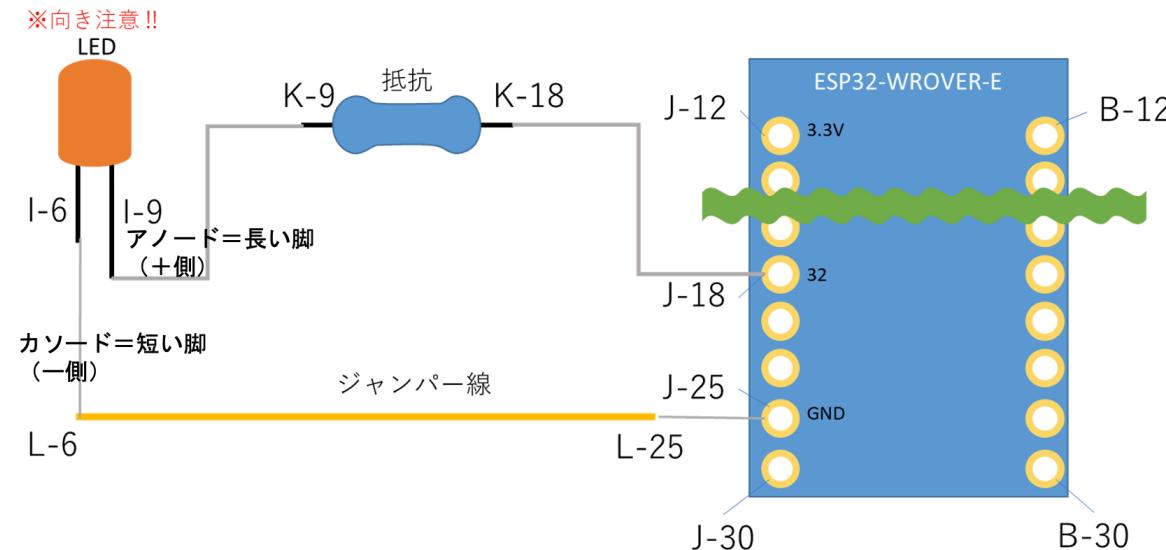


図 18: LED 点滅の結線図

ESP32 の 32 番ピンが LED のアノードに接続されていることに注目してください

正しく配線できていますか?

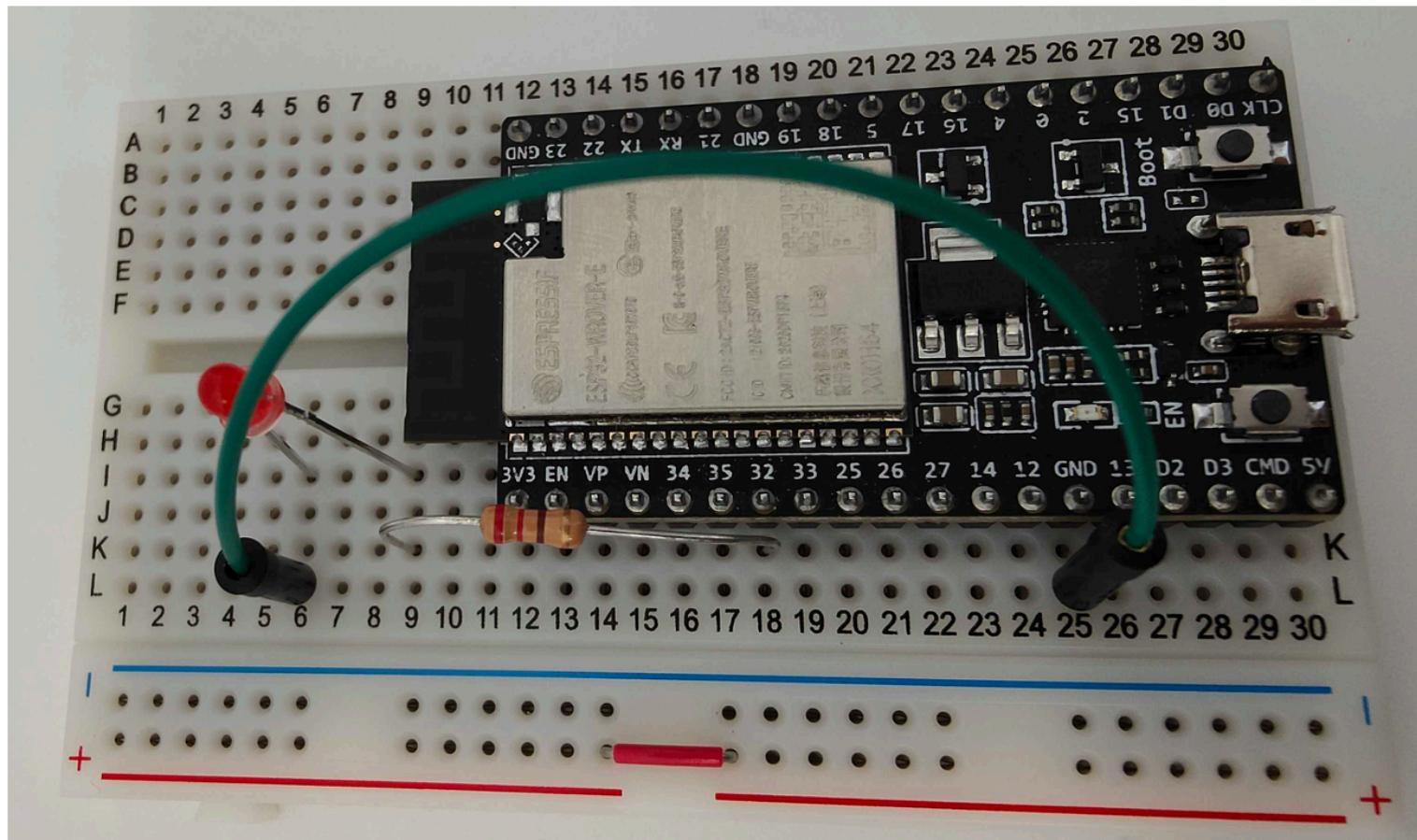


図 19: 結線の写真

3.4. Arduino IDE でプログラムを書く

3.4.1. Arduino 言語の特徴

Arduino 言語は、C 言語に似た構文を持つプログラミング言語です。2 つの主な関数、`setup()` と `loop()` を使用してプログラムを構成します。

`setup()`:

プログラムの初期化を行う関数で、マイコンが起動したときに一度だけ実行されます。

`loop()`:

`setup()` 関数の後に実行される関数で、マイコンが起動している間、繰り返し実行されます。



```
1 #define LED_PIN 32
2
3 void setup() {
4     // put your setup code here, to run once:
5     pinMode(LED_PIN,OUTPUT);
6 }
7
8 void loop() {
9     // put your main code here, to run repeatedly:
10    digitalWrite(LED_PIN,HIGH);
11    delay(1000);
12    digitalWrite(LED_PIN,LOW);
13    delay(1000);
14 }
```

図 20: Arduino IDE のコードエディタ

3.5. ボード設定の確認

Arduino IDE で ESP32 を使用するためには、ボードの設定を確認する必要があります。ボードの設定は、ツールメニューから行います。

ボードメニューから、使用する ESP32 Wrover Module を選択します。

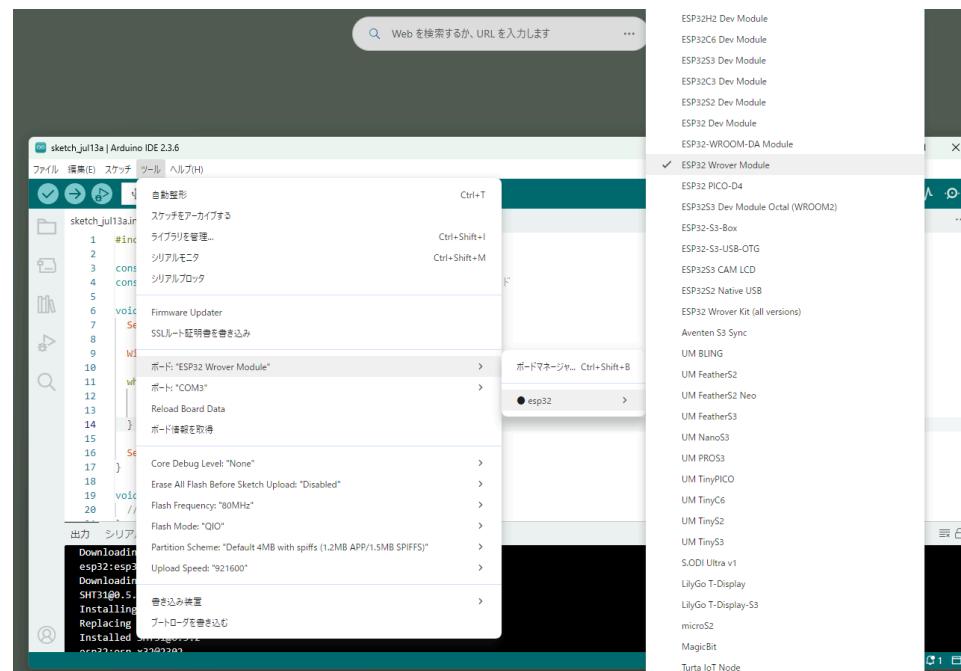


図 21: Arduino IDE のボード設定

もしボードの設定が表示されない場合は、ツールメニューからボードマネージャーを開き、ESP32 のボードをインストールしてください。

[事前準備資料](#)

3.6. シリアルポートの設定

Arduino IDE で ESP32 を使用するためには、シリアルポートの設定を確認する必要があります。シリアルポートの設定は、ツールメニューから行います。

シリアルポートメニューから、ESP32 が接続されているポートを選択します。(COM3)

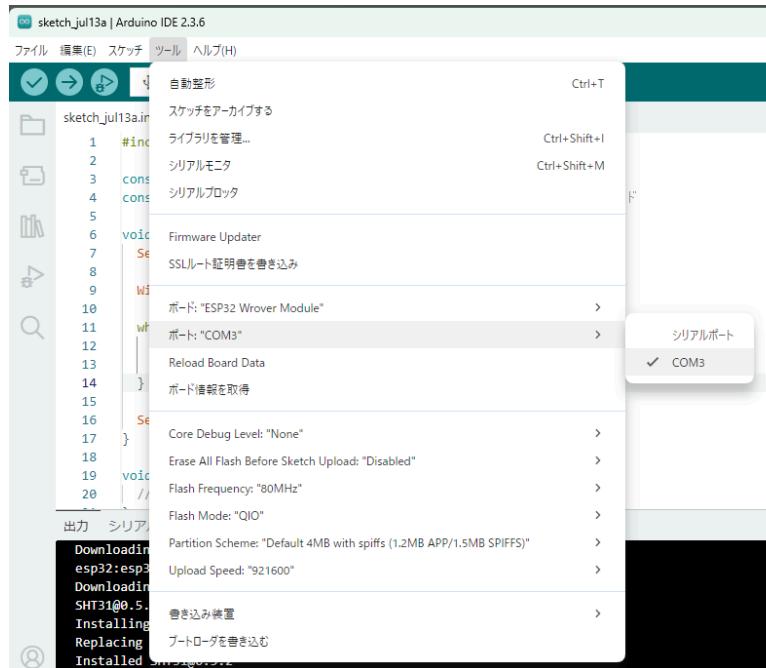


図 22: Arduino IDE のシリアルポート設定

もしシリアルポートの設定が表示されない場合は、USB ケーブルが正しく接続されているか確認してください。また、ドライバが正しくインストールされているかも確認してください。

3.7. LED 点滅のプログラム

Arduino IDE で、ESP32 の GPIO ピンを制御し、LED を点滅させるプログラムです。
以下のコードを Arduino IDE にコピーしてください。

setup() 関数で、GPIO ピンのモードを設定し、
loop() 関数で、LED を点灯・消灯させる処理を繰り返します。

[LED 点滅のサンプルコードはこちら](#)

3.8. プログラムのアップロード

プログラムを書いたら、書き込みボタンをクリックして、ESP32 にプログラムをアップロードします。

アップロードが完了すると、LED が点滅するはずです。



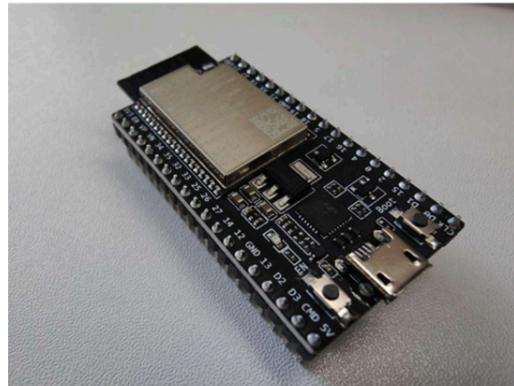
図 23: Arduino IDE の書き込みボタン(→のアイコン)

書き込みに失敗する場合、ユーザー名に全角文字が含まれていることが原因の可能性があります。可能であれば半角英数字のみのユーザー名を使用してください。

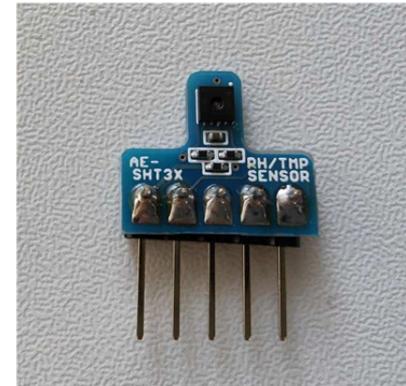
4. 温湿度の取得

4.1. 使用する機器

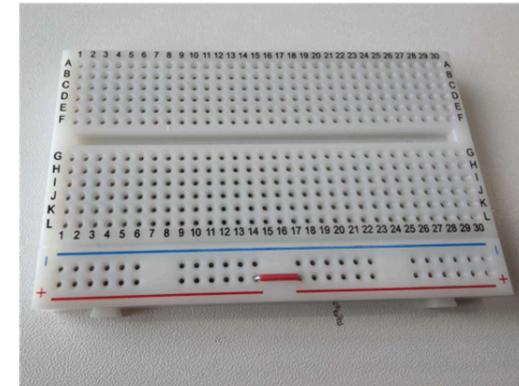
- ESP32-WROVER-E
- 温湿度センサー (AE-SHT31)
- ブレッドボード
- USB ケーブル (Type-A)
- ジャンパー線



ESP32-WROVER-E



AE-SHT31



ブレッドボード



USB Type A ケーブル



ジャンパー線

図 24: 使用する機器

4.2. ESP32 と温湿度センサーの接続

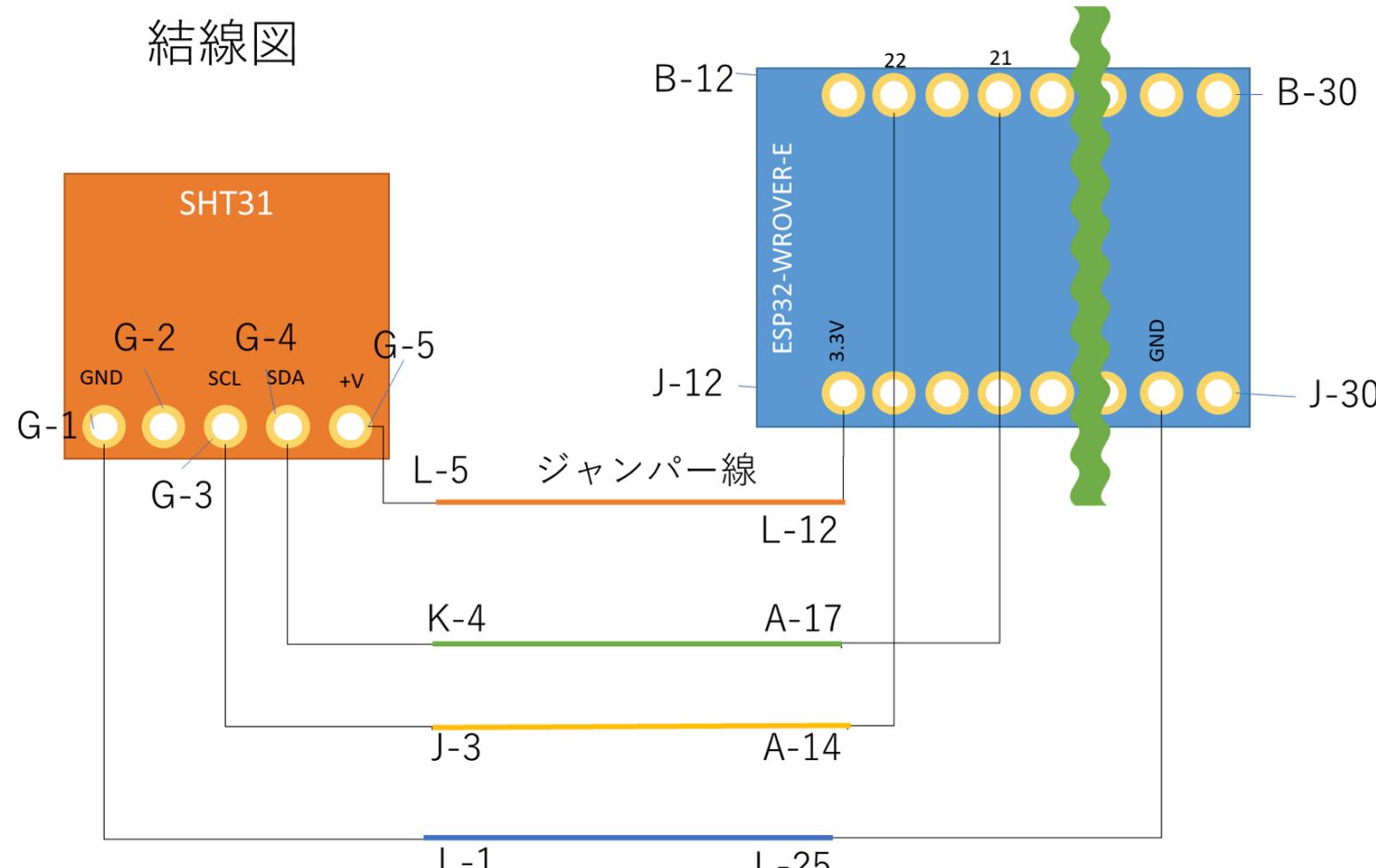


図 25: 結線図

4.3. 結線

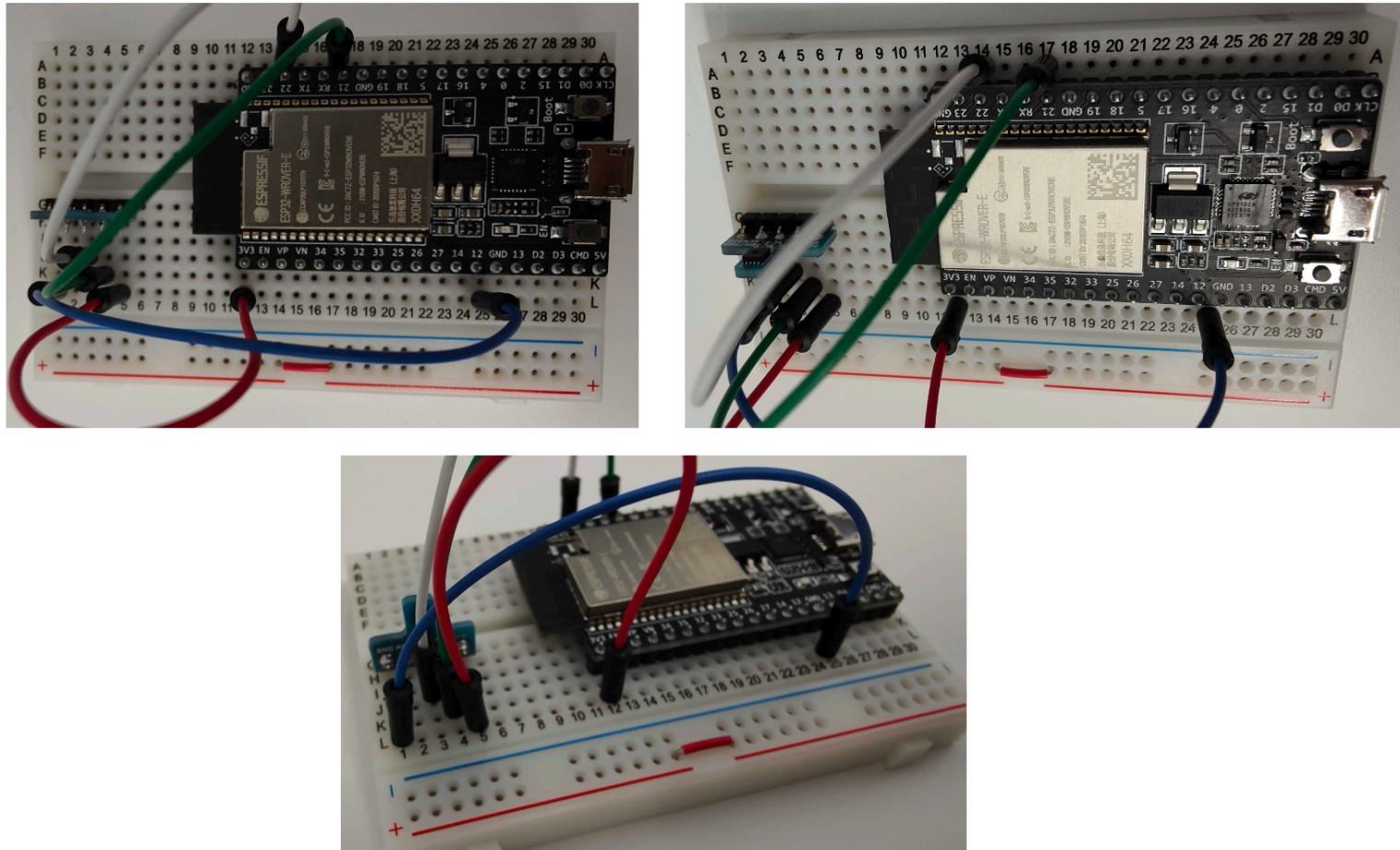


図 26: 結線の写真

4.4. SHT31 のアドレス確認

もし配線が間違っていないのに接続できない場合は、SHT31 のアドレスを確認してください。

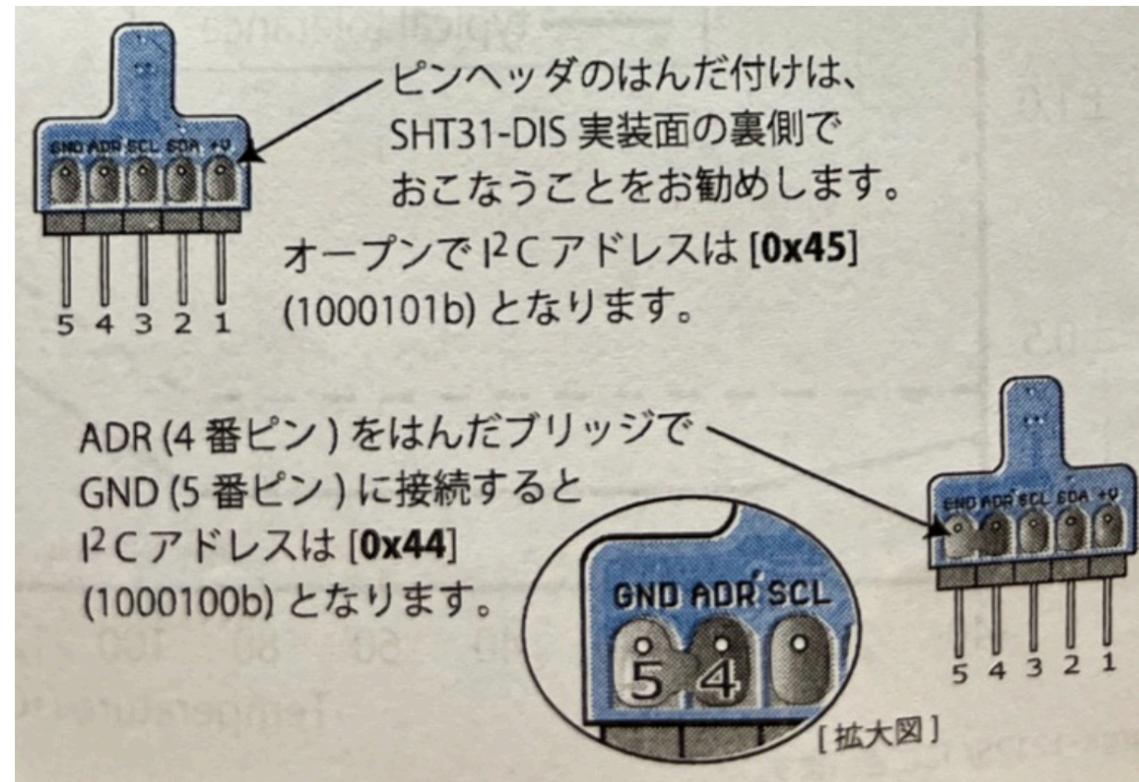


図 27: SHT31 のアドレス確認

4.5. 温湿度取得のプログラム

温湿度センサーからデータを取得するプログラムです。
以下のコードを Arduino IDE にコピーしてください。

setup()関数で、I2C 通信の初期化とセンサーの初期化を行い、
loop()関数で、温湿度データを取得してシリアルモニタに出力します。

[温湿度取得のサンプルコードはこちら](#)

4.6. シリアルモニタとは

シリアルモニタは、Arduino IDE の一部で、マイコンと PC 間のシリアル通信を行うためのツールです。シリアルモニタを使用して、マイコンからの情報やセンサーデータを確認できます。シリアルモニタは、ツールメニューから開くことができます。



図 28: ツールメニューからシリアルモニタを選択

4.7. ボーレート

シリアルモニタを使用する際は、ボーレートを設定する必要があります。ボーレートは、シリアル通信の速度を表す単位です。

115200bps (ビット/秒) を使用してください。

シリアルモニタの右下にあるボーレートのドロップダウンメニューから設定できます。



図 29: シリアルモニタのボーレート設定

4.8. シリアルモニタで確認

Arduino IDE のシリアルモニタを使用して、温湿度センサーからのデータを確認します。シリアルモニタは、Arduino IDE のツールメニューから開くことができます。

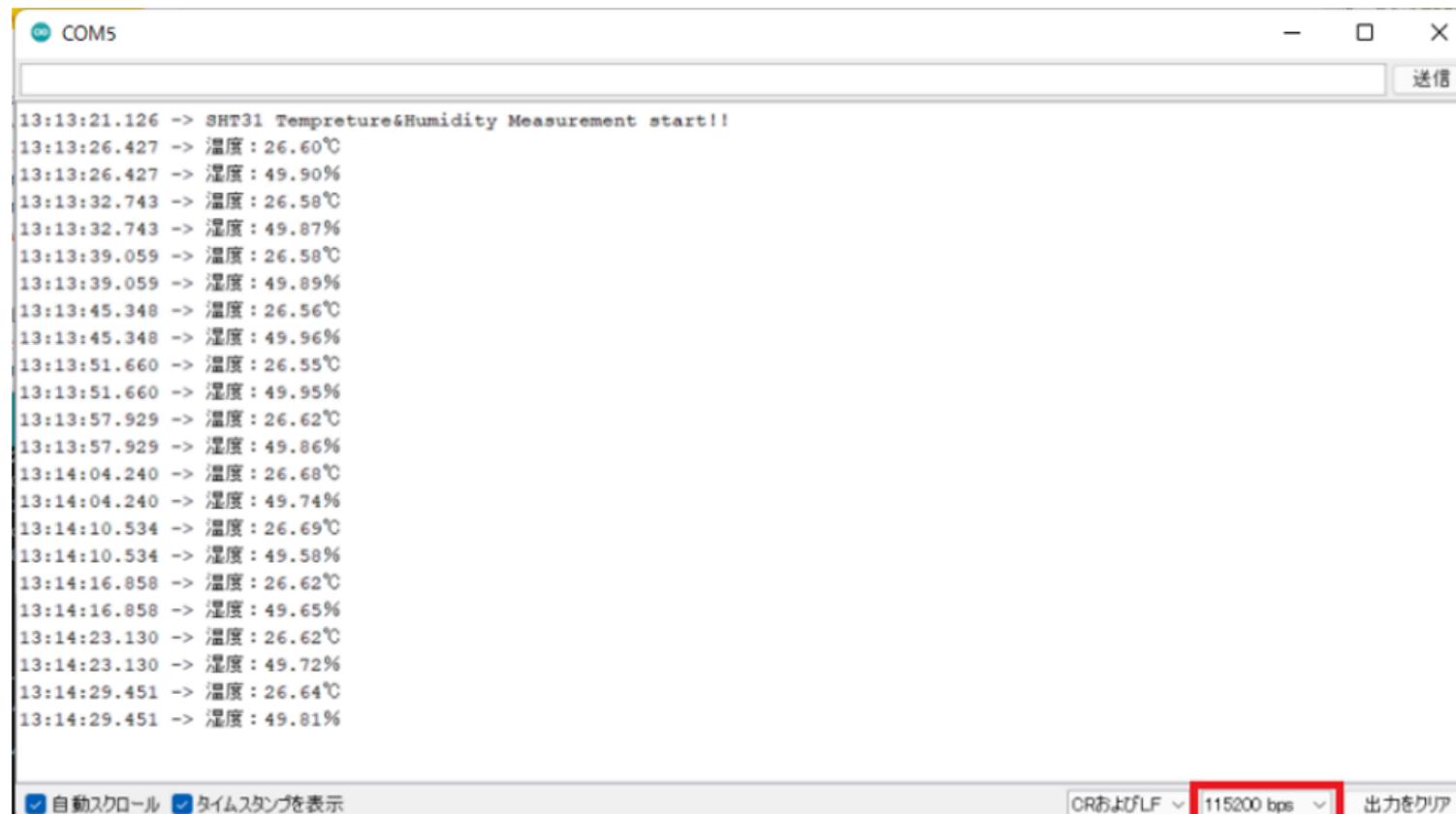


図 30: シリアルモニタの画面

5. MQTT 通信

IoT に適した通信方式を学びます

5.1. HTTP 通信とは異なる方式

IoP クラウドでは、環境測定装置とクラウド側の通信の仕組みを総じて「デバイス API」と呼んでいます。一般的な「HTTP 通信による API」とは異なる方式が採用されています。

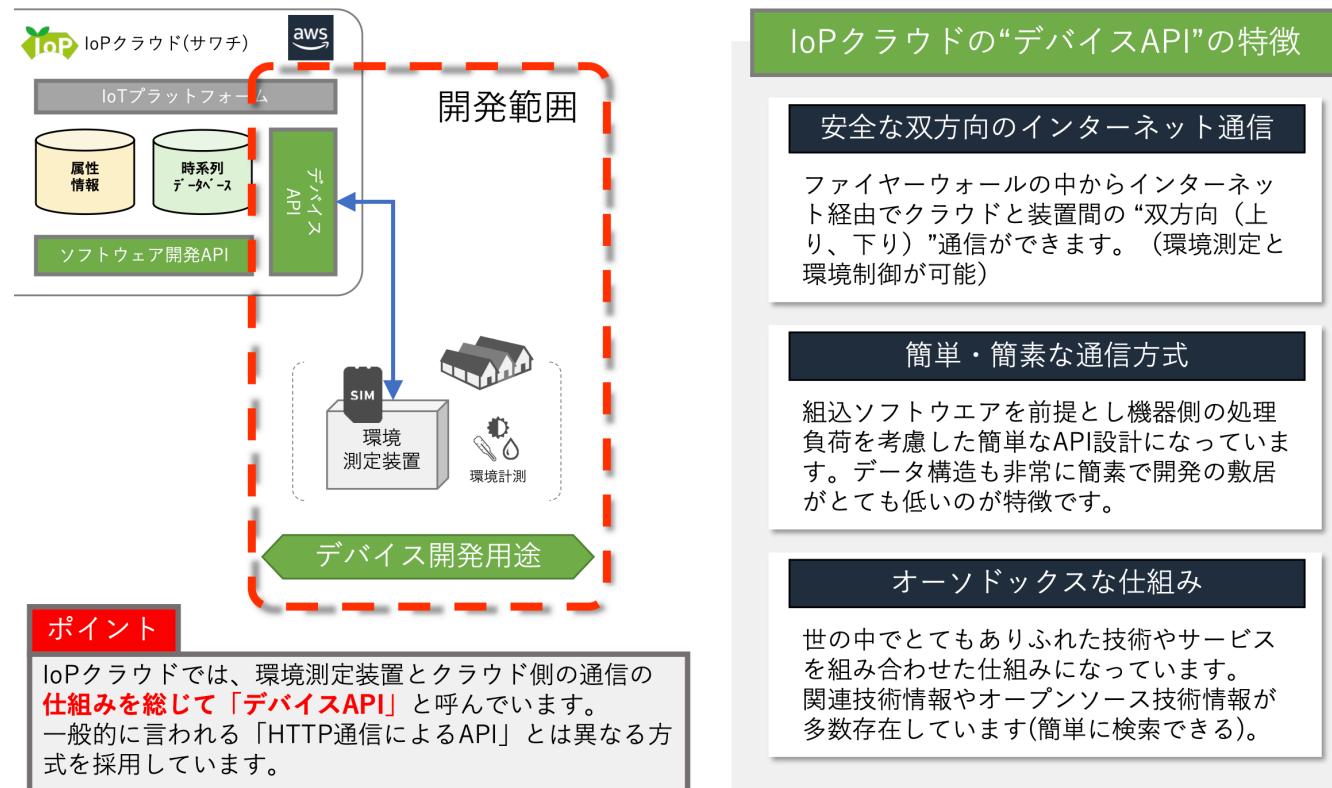


図 31: デバイス開発用途の API について

出典: 【配布用】 SAWACHI 技術資料ライブラリ API 編
30_SAWACHI デバイス API を活用した開発手法説明

5.2. HTTP 通信を確認

HTTP 通信は、Web ブラウザで Web サイトを閲覧する際に使用される通信方式です。クライアント（ブラウザや IoT デバイス）からサーバーにリクエストを送り、サーバーからレスポンスを受け取ります。この通信方式は、**リクエスト/レスポンス型**と呼ばれます。下記のコードを試してみましょう

HTTP 通信のコード

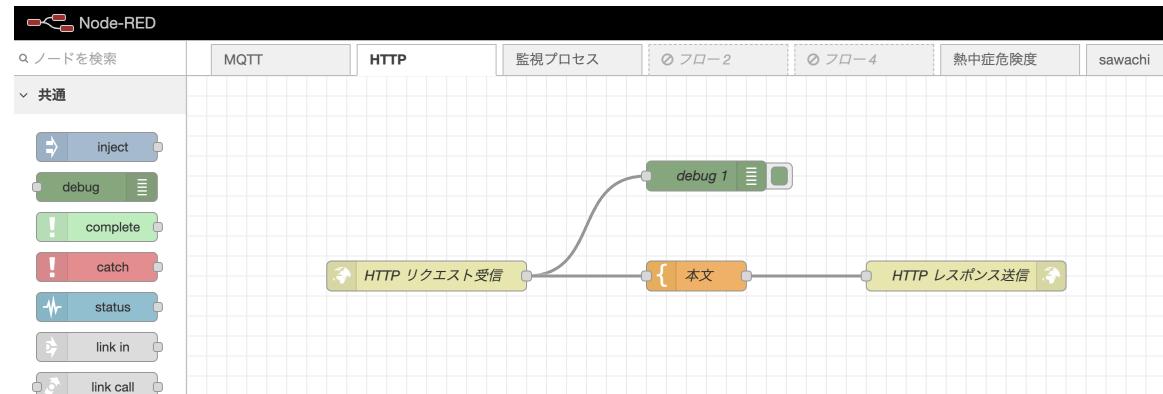


図 32: 同じネットワーク上の **Node-RED** を使用して、
ESP32 からの HTTP 通信を確認します。

Node-RED Node-RED は、IoT アプリケーションに適した機能を持つローコード開発ツールです。

5.3. MQTT 通信

MQTT は、**Message Queuing Telemetry Transport** の略で、軽量なメッセージングプロトコルです。Broker による双方向通信が可能で、IoT デバイスとクラウド間の通信に適しています。

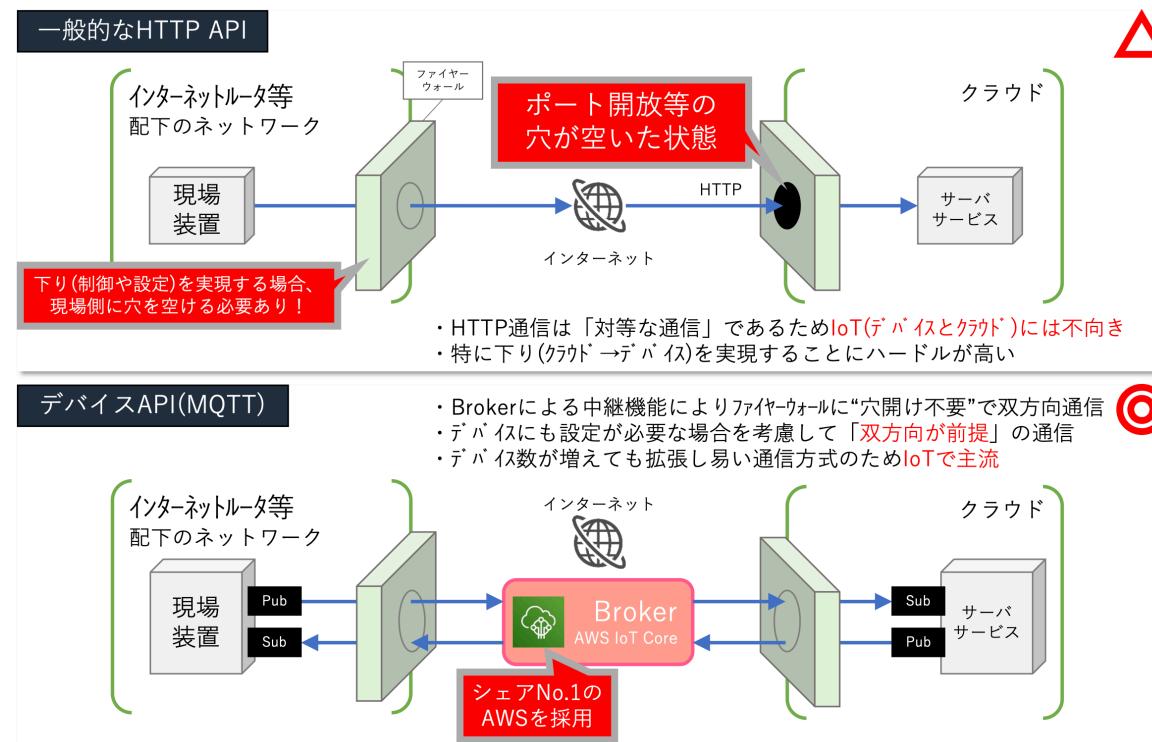


図 33: IoT クラウドが MQTT を採用する理由
出典: 【配布用】SAWACHI 技術資料ライブラリ API 編
30_SAWACHI デバイス API を活用した開発手法説明

5.4. MQTT の構成

環境測定装置は Publisher として SAWACHI の提供する Broker に接続します。topic、broker などの用語がわからなくなったらこの図を思い出してください。

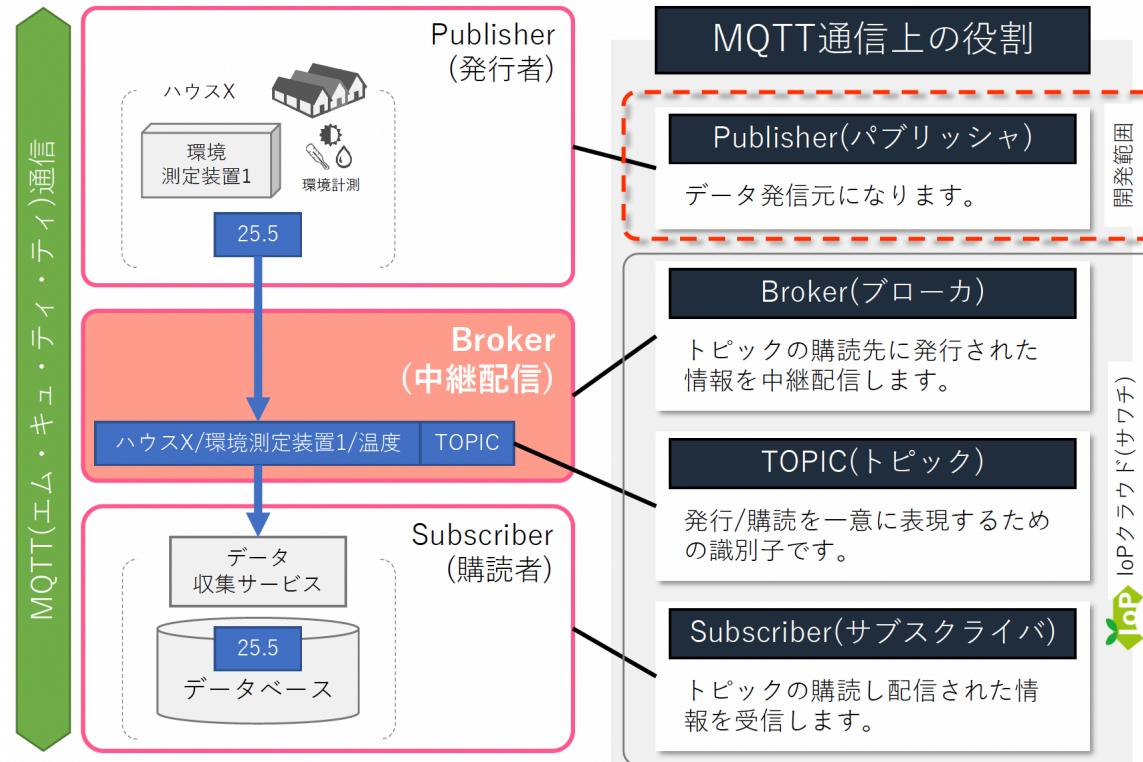


図 34: Pub/Sub 方式の MQTT 通信を採用
出典: 【配布用】SAWACHI 技術資料ライブラリ API 編
30_SAWACHI デバイス API を活用した開発手法説明

5.5. PubSubClient のインストール

MQTT 通信を行うために、**PubSubClient** ライブラリを Arduino IDE にインストールします。
PubSubClient は、MQTT プロトコルを使用してメッセージを送受信するためのライブラリです。
インストール方法は以下の通りです。

1. ツールメニューからライブラリを管理を選択します。
 2. 検索バーに「PubSubClient」と入力します。
 3. **PubSubClient**を見つけて、インストールボタンをクリックします。

<https://www.arduino.cc/reference/en/libraries/pubsubclient/>



図 35: PubSubClient のインストール

5.6. MQTT 送信

MQTT がどのようなものかを理解するために、Node-RED を使用して MQTT の送信テストを行います。

以下のコードを Arduino IDE にコピーしてください。

MQTT 送信テストのコード

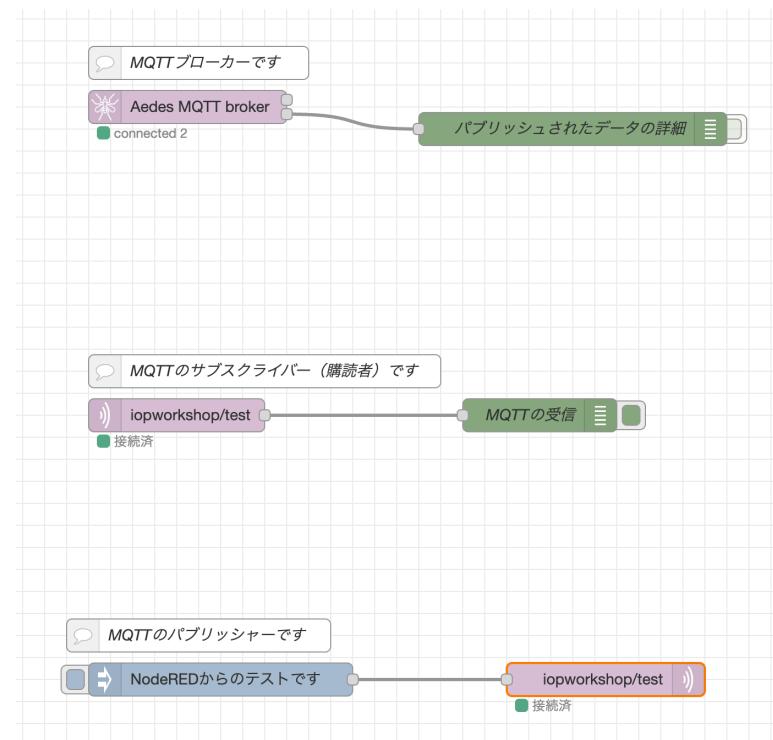


図 36: Node-RED を使用して MQTT の送信テストを行います。

5.7. MQTT 受信

次に、Node-RED を使用して MQTT の受信テストを行います。以下のコードを Arduino IDE にコピーしてください。

MQTT 受信テストのコード

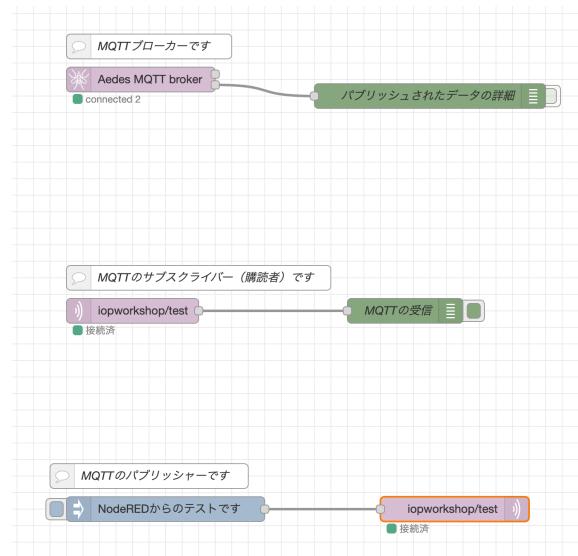


図 37: (再掲) Node-RED を使用して MQTT の受信テストを行います。

このように、MQTT を使用すると、送信だけでなく受信も行うことができます。

5.8. (補足) ボタンを使った MQTT 送信デモ

ESP32 の GPIO ピンを使用して、ボタンを押すと MQTT でメッセージを送信するデモを行います。

プログラムされたタイミングだけでなく、**IoT デバイスの状態に応じて MQTT メッセージを送信することができます。**

ボタンを使った MQTT 送信デモ

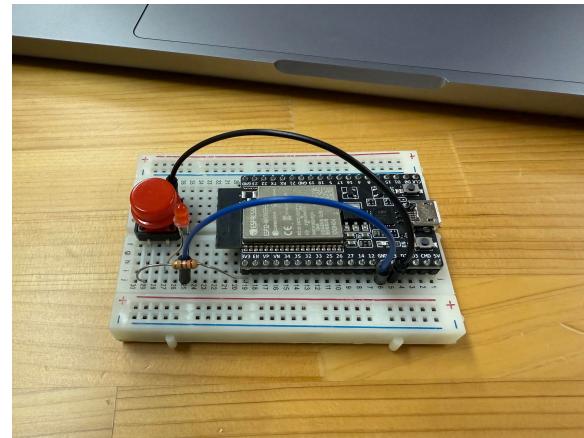


図 38: ボタンを押すと MQTT でメッセージが送信されるデモ

5.9. Qsu プロトコル

SAWACHI では、環境測定装置とクラウド側の通信のために JSON ベースの Qsu プロトコルを策定しています。プロトコルとは、通信のルールや手順を定めたものです。

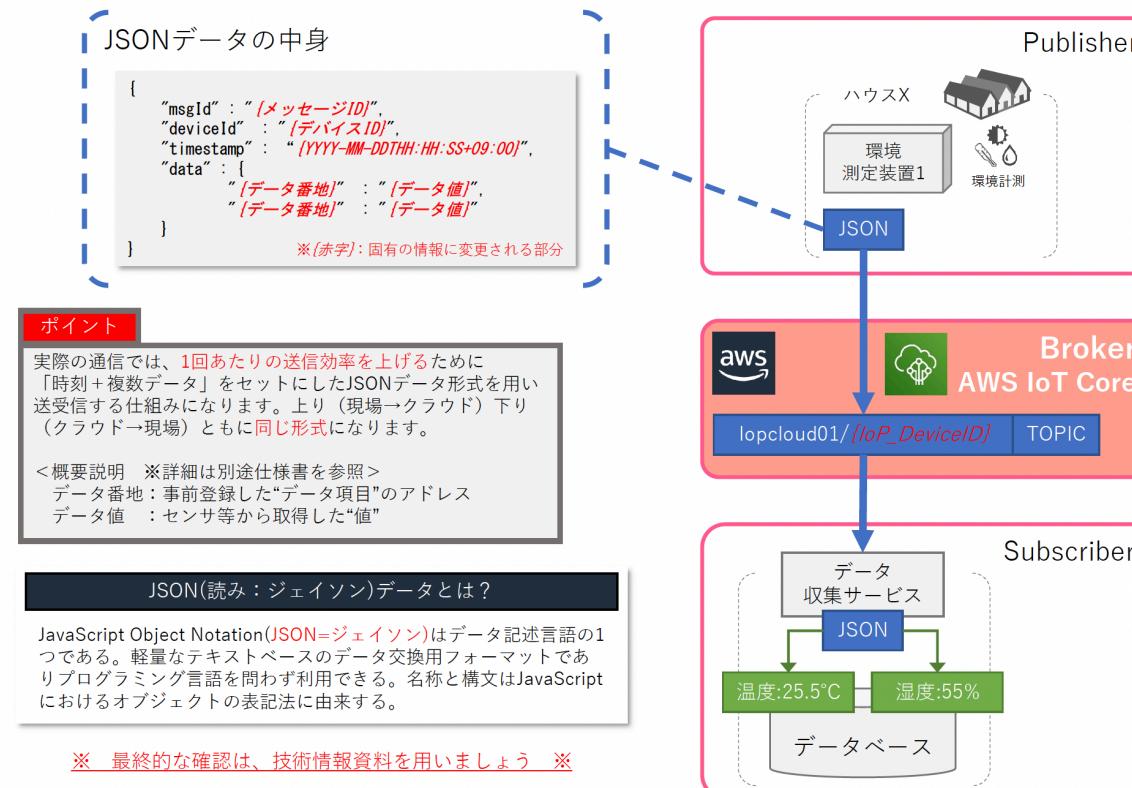


図 39: デバイス API のデータ連携の概要
出典: 【配布用】 SAWACHI 技術資料ライブラリ API 編
30_SAWACHI デバイス API を活用した開発手法説明

5.10. Qsu プロトコル演習

今回は、温度と湿度を Qsu フォーマットに従って送信します。 ArduinoJson ライブラリを PubSubClient と同様にインストールしてください。

温湿度センサーのデータを SAWACHI に送信します。以下のコードを Arduino IDE にコピーしてください。

Qsu プロトコルでのデータ送信



図 40: ArduinoJson ライブラリのインストール

5.11. Qsu プロトコルの確認

Qsu プロトコルで送信されたデータは、データは JSON 形式で送信されます。以下のような形式でデータが送信されます。

```
{"msgId": "P000000001", "deviceId": "some_device_id", "data": {"400": "1", "1000": "27.34", "1001": "71.14"}}
```

The screenshot shows a log entry from a Node-RED node. The title of the node is "MQTTの受信" (MQTT Receive). The log message is timestamped at "6/26/2025, 1:32:44 PM". The message content is: "ノード: MQTTの受信" followed by "iopworkshop/test : msg.payload : Object". Below this, the payload is shown as an object with properties: "msgId: "P000000001"" and "deviceId: "some_device_id"". Under the "data" property, there is another object with properties: "400: "1"" and "1000: "27.34"" and "1001: "71.14"".

```
MQTTの受信
6/26/2025, 1:32:44 PM ノード: MQTTの受信
iopworkshop/test : msg.payload : Object
  ▼object
    msgId: "P000000001"
    deviceId: "some_device_id"
  ▼data: object
    400: "1"
    1000: "27.34"
    1001: "71.14"
6/26/2025, 1:32:54 PM ノード: MQTTの受信
```

図 41: Qsu プロトコルで送信されたデータの例

6. SAWACHI でのデータ確認

詳細分析画面 = モデルメソッドを使用してデータを確認します

6.1. デバイス API のセキュリティ

SAWACHI では、デバイス API のセキュリティを確保するために、以下のような仕組みを採用しています。

1. セキュリティ管理が可能な PaaS(AWS IoT Core)で構築
2. MQTT broker は PaaS が提供する管理されたサービスを利用
3. セキュリティが担保され発行された**クライアント証明書**を利用
4. セキュリティポリシーの厳格な管理によりクライアントの制限が可能
5. MQTT Pub/Sub に厳密なアクセス制御を施し末端までセキュアな仕組み
6. 許可されたクライアントからしかアクセスできない仕組み

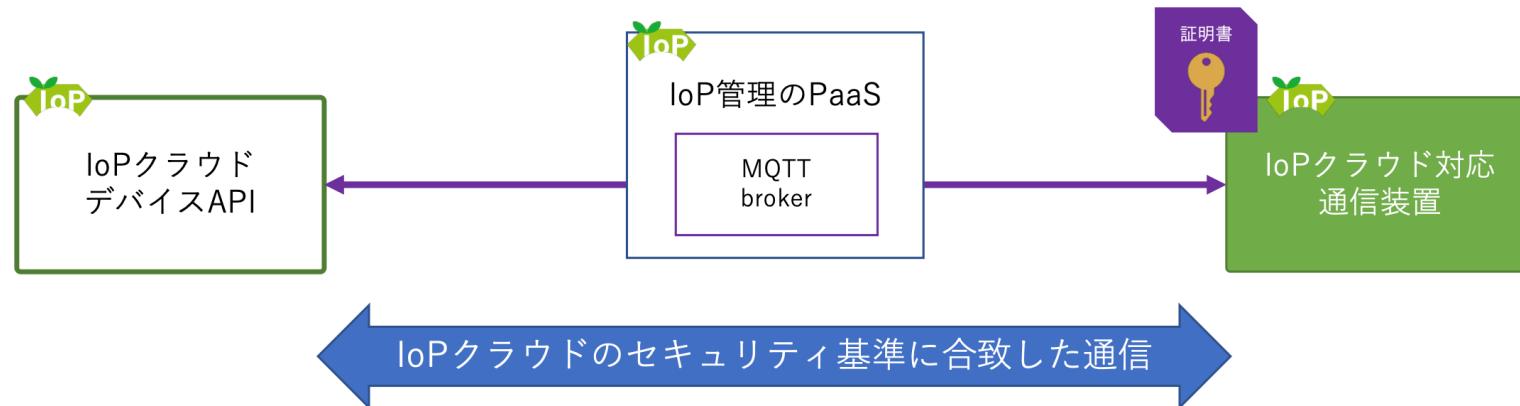


図 42: デバイス API について
出典: 【配布用】 10_SAWACHI API の概要

6.2. データの送信のプログラム

各参加者ごとに証明書を用意しています (PASTEME.txt)
下記コードに証明書を貼り付けてください。

[SAWACHIへのデータ送信](#)

6.3. 詳細分析画面へのアクセス

SAWACHI の詳細分析画面にアクセスするには、
ブラウザで以下の URL を開きます。
(ユーザー名、パスワードは別途お伝えします)



図 43: 詳細分析画面のログイン画面

6.4. データの確認

詳細分析画面は非常に多機能ですが、
今回は表示に必要な設定を既に行っています。
データが確認できれば、今回の講座は成功です!

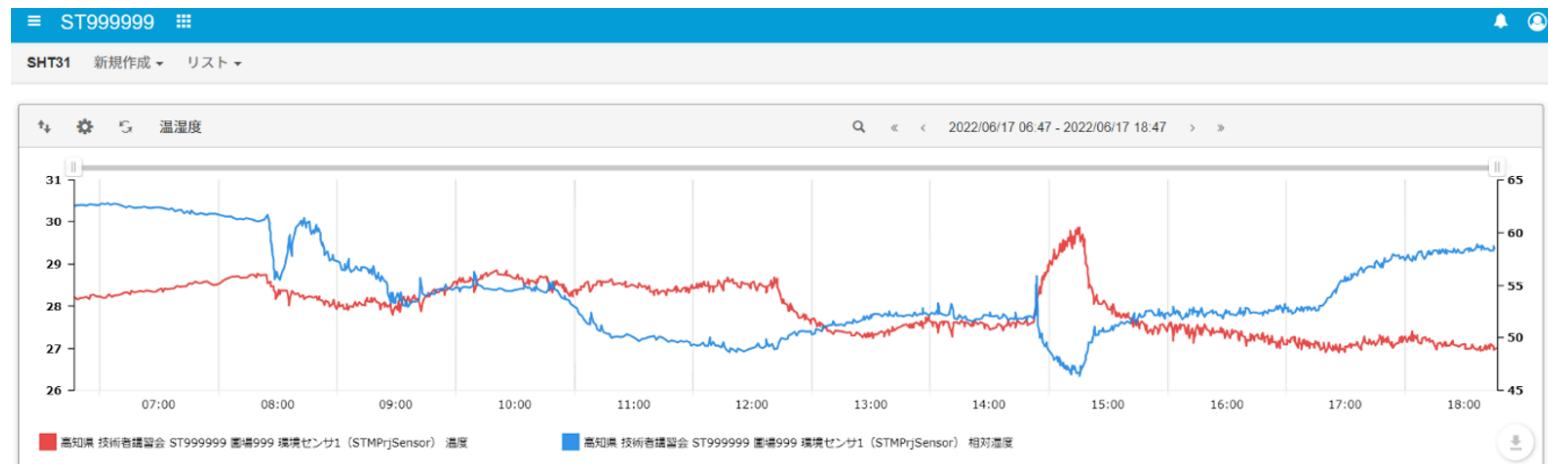


図 44: 詳細分析画面のデータ表示例