

# IoT概論

## 温湿度モニタリング機器を 作ってみよう

国際情報農学研究室

高草木 和史

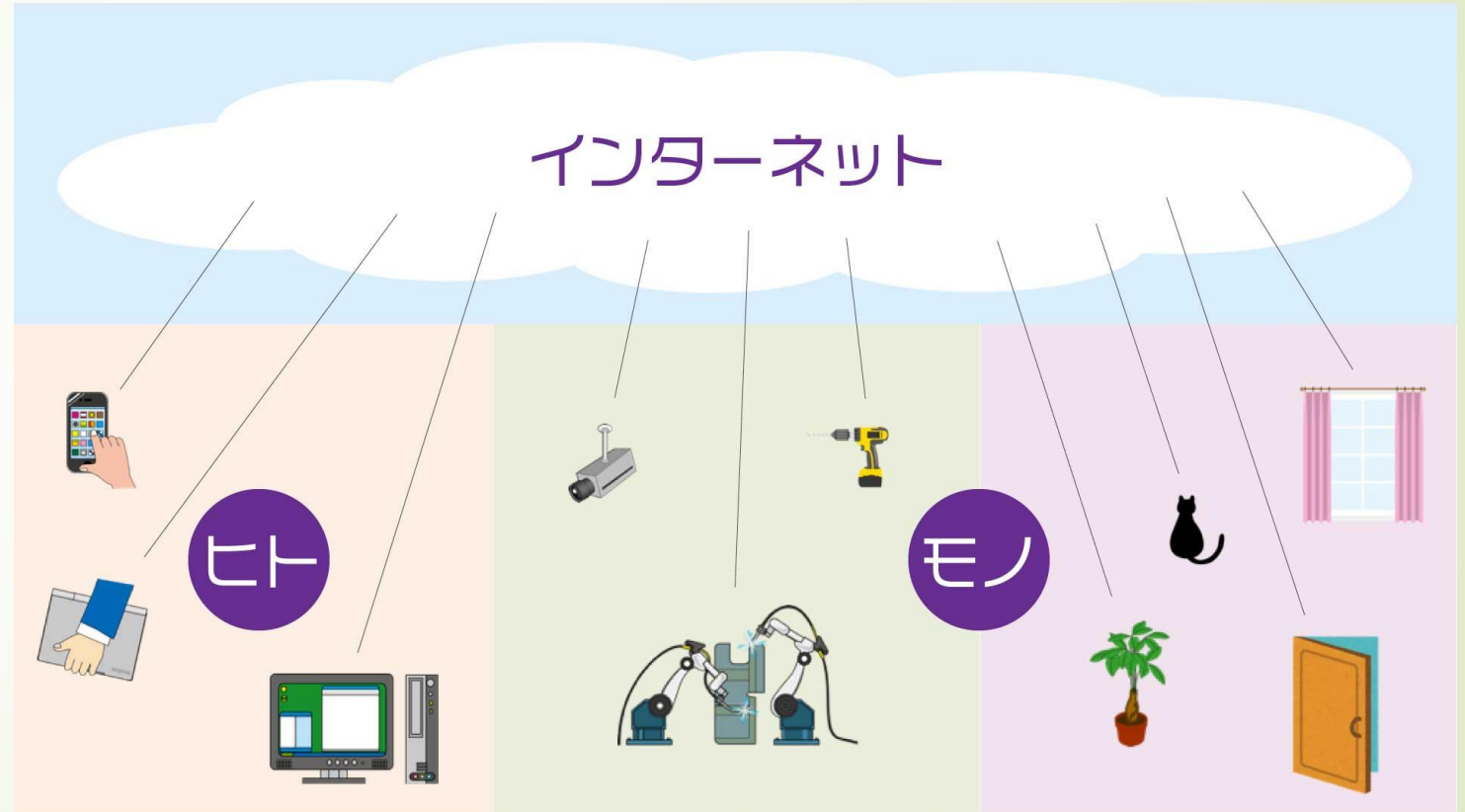
# そもそもIoTとは？

## Internet of Things:

モノがインターネット経由で  
通信することを意味する

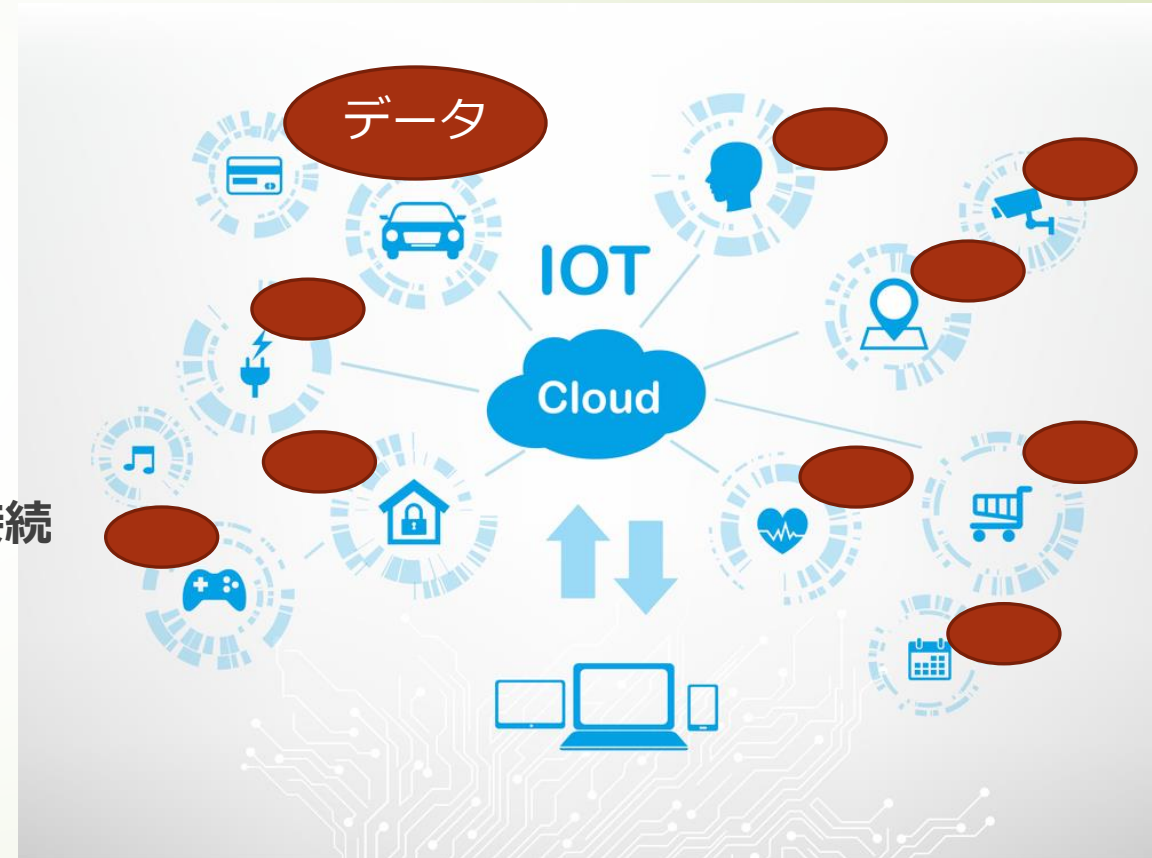
モノのインターネット

以前は、インターネットはコン  
ピュータ同士を接続するためのも  
のだった



# IoTを支える3技術

- デバイス技術  
モノからデータを得る技術
- システム技術  
データを送受信してクラウドに接続
- 応用技術  
データの分析と処理を行う



# IoT概論の目的

- ▶ IoTのハードに実際に触れる
- ▶ 自分で組み立てを行う
- ▶ IoTの基礎的な成り立ちを知る
  - ▶ センサーからデータを得る
  - ▶ 得たデータを可視化するまで

# 全体の構成

## 準備編

- ① Arduino IDEのインストール、環境設定
- ② Lチカを試してみよう

## 実践編

- ③ 温湿度の計測をしよう
- ④ 測った温湿度をグラフ化してみよう

デバイス技術

システム技術

IoT

# 要件

- ▶ 開発環境
  - ▶ Arudino version 1.8.12
- ▶ ライブラリ
  - ▶ Esp8266 version 2.4.2

# Arduinoのインストール

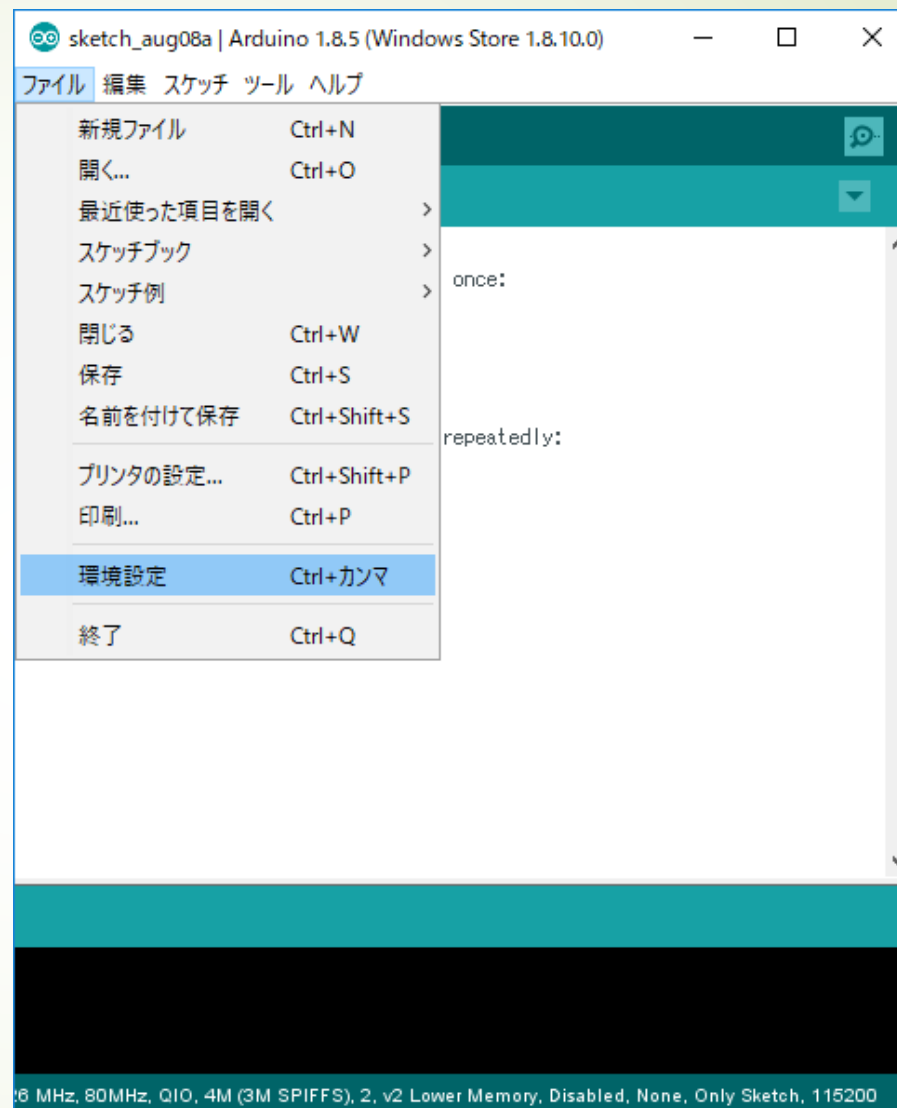
Arduino IDEのダウンロードをArduinoのwebページで行います。

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

1. [SOFTWARE] メニュー の[DOWNLOADS] を選択
2. [Previous Releases] → **[Previous Release (1.8.13)] をクリック**
3. [1.8.12]の中で自分のバージョンをダウンロード
4. Contribute to the Arduino Software の画面はお金を払いたくない場合は、右下の[JUST DOWNLOAD]をクリックしましょう。
5. ダウンロードができればArduinoを開きましょう。
6. ダウンロードしたインストーラのファイル名は Arduino-1.8.12-windows.exeのような名前です。

# ボードマネージャの追加

- ▶ 左図のように「ファイル」メニューから、環境設定画面を開く
- ▶ チェックボックスの下の、追加のボードマネージャのURLを追加します。





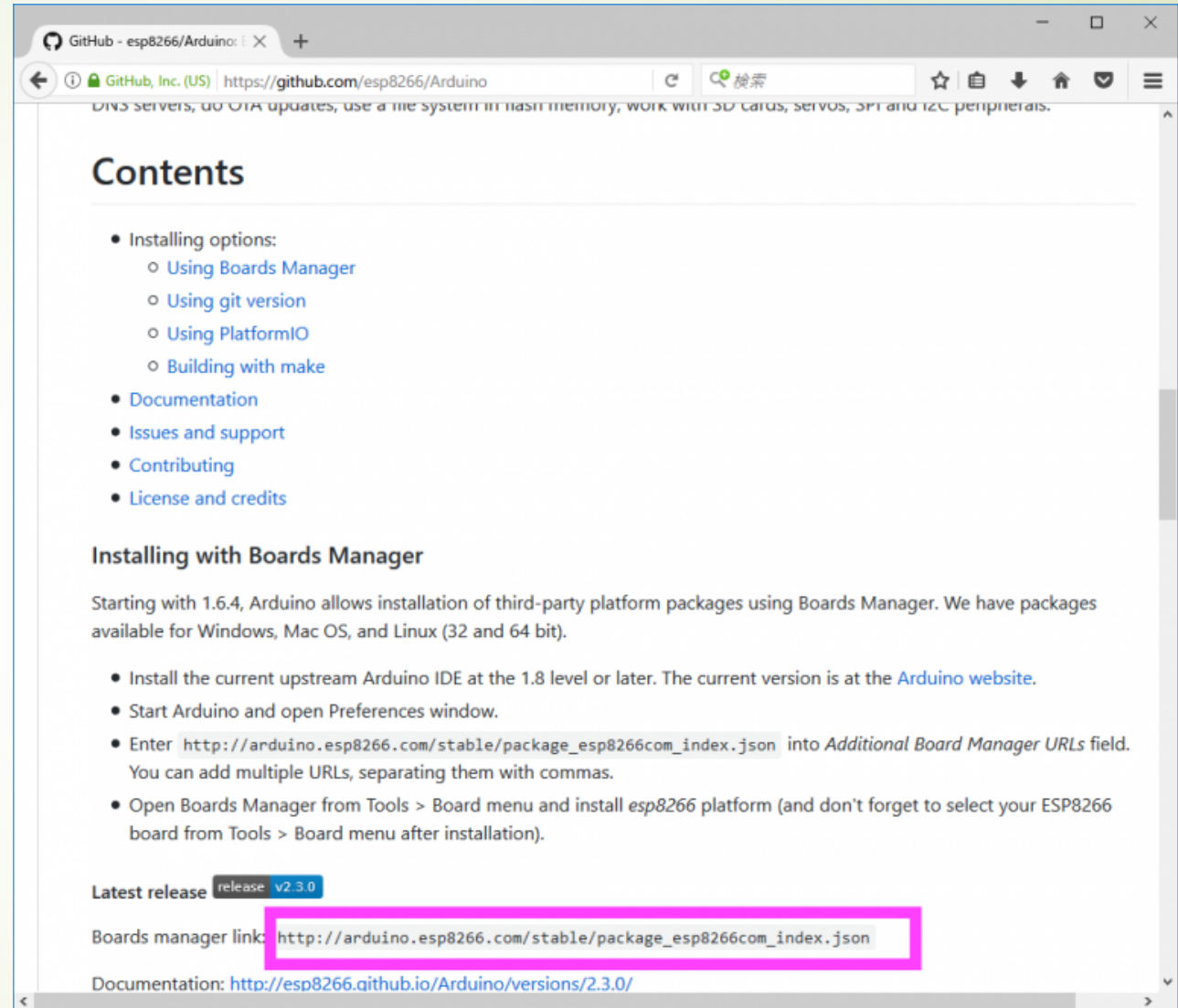
# 追加のボードマネージャ

➡ <https://github.com/esp8266/Arduino>

を開いて下の方にスクロールすると、ボードマネージャのリンクがあるのでコピーして、

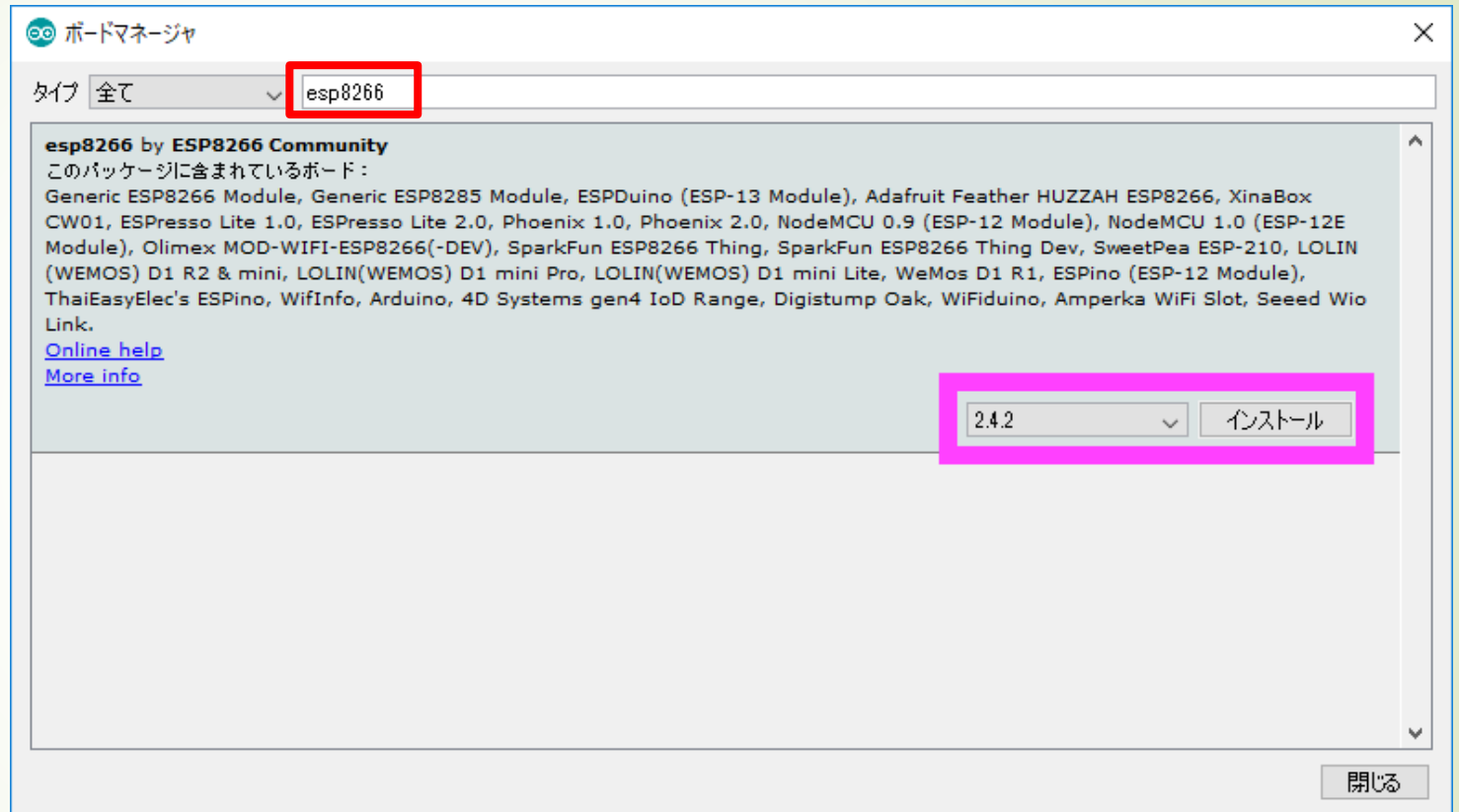
前頁の“追加のボードマネージャのURL”のところにペーストします。

OKボタンを押して環境設定を完了します。



# ライブラリの追加

- [ツール] → [ボード] → [ボードマネージャ] でボードマネージャ画面を開く
- 上の検索欄に"esp8266"を記入する
- 紫枠のようにversion2.4.2に変更しインストールを行う。

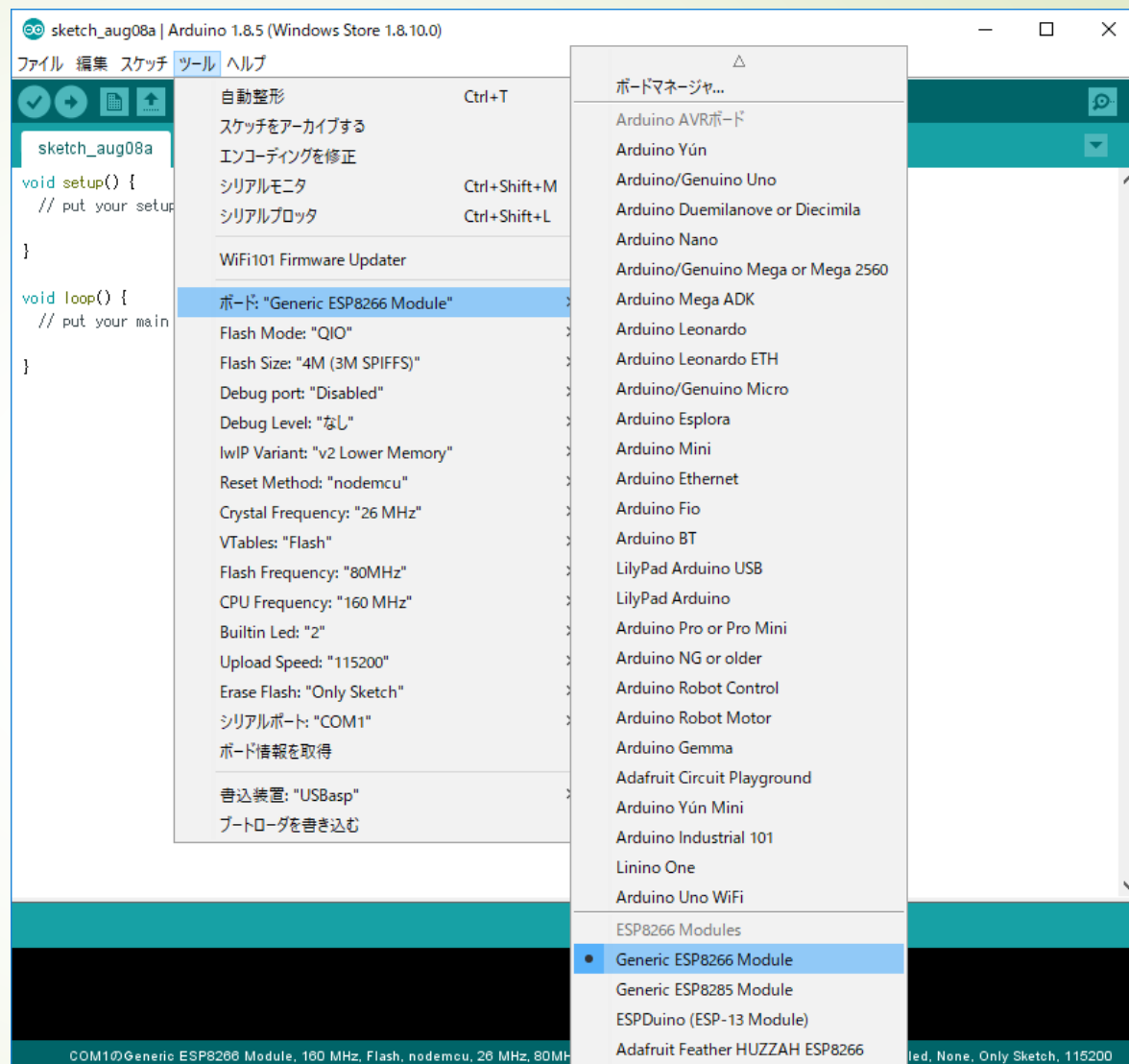


# ボードの選択

「ツール」 → 「ボード」

で下の方にスクロールすると、

[Generic ESP8266 Module]という項目が現れるのでそれを選択する

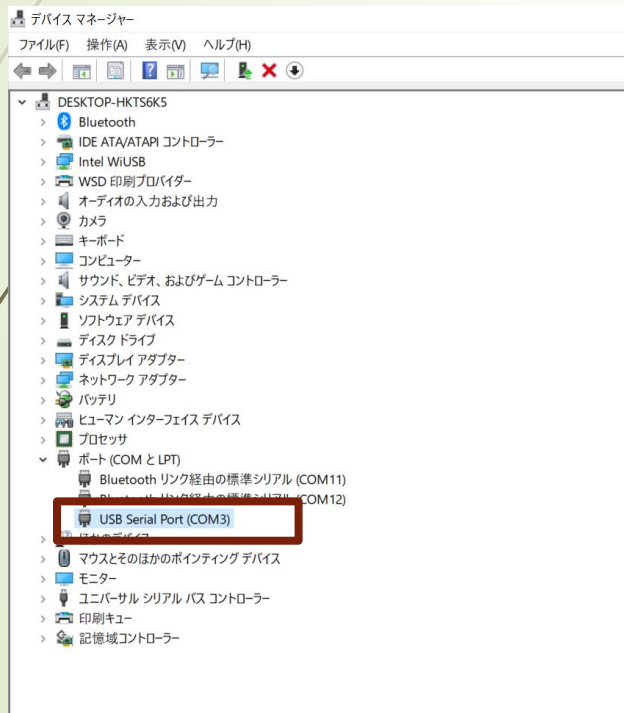


# 接続の確認

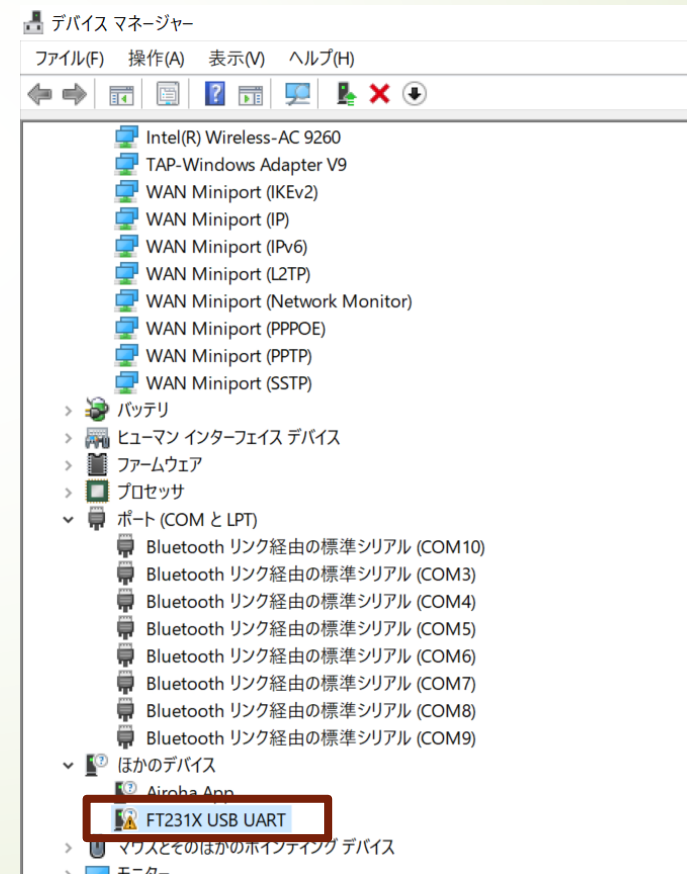
- ▶ 開発ボードとパソコンをつないで、シリアルポートを選択する。
- ▶ そのとき追加前から増えたシリアルポートを選択する。
- ▶ シリアルポートが追加されない場合は、
  - ・ 環境設定のURL確認
  - ・ ツールの設定が正しくなっているか確認
  - ・ ケーブルを交換する

# シリアルポートが読み込めないとき 1

上手くいくケース  
ポートに'USB Serial Port  
(COM番号)'が現れる

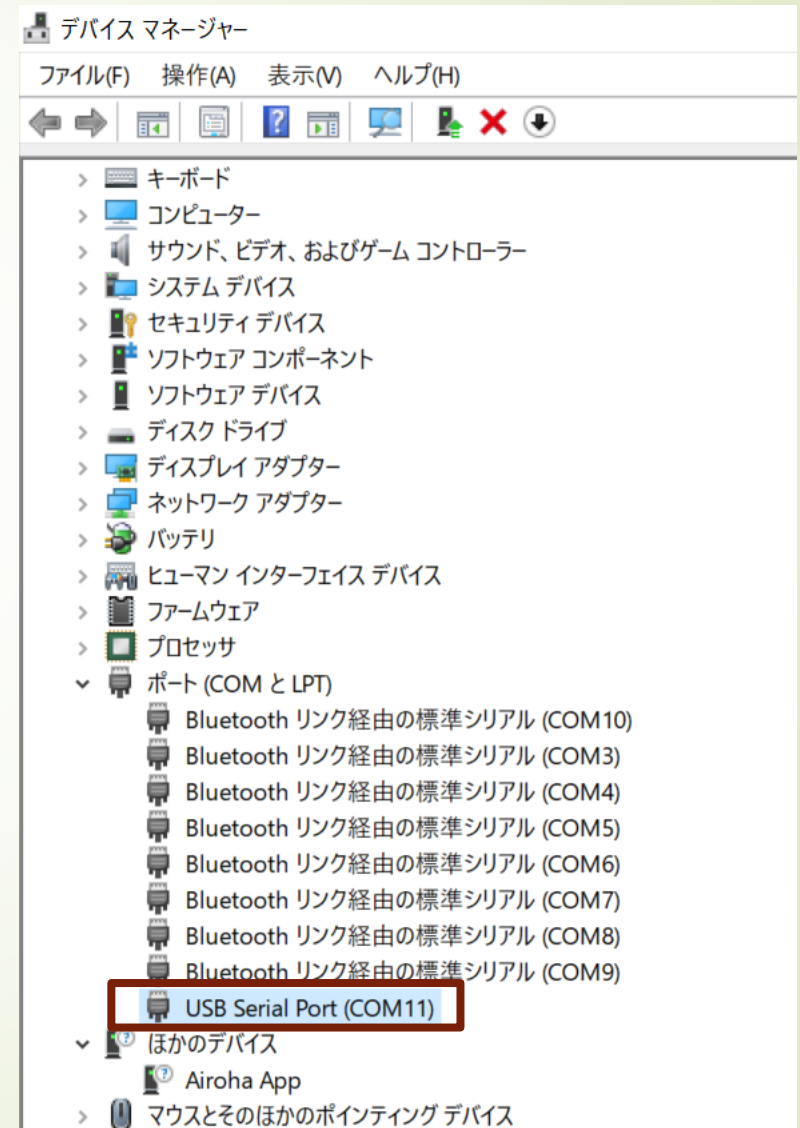


上手くいかないケース  
ポートに'USB Serial Port (COM番号)'  
が現れない



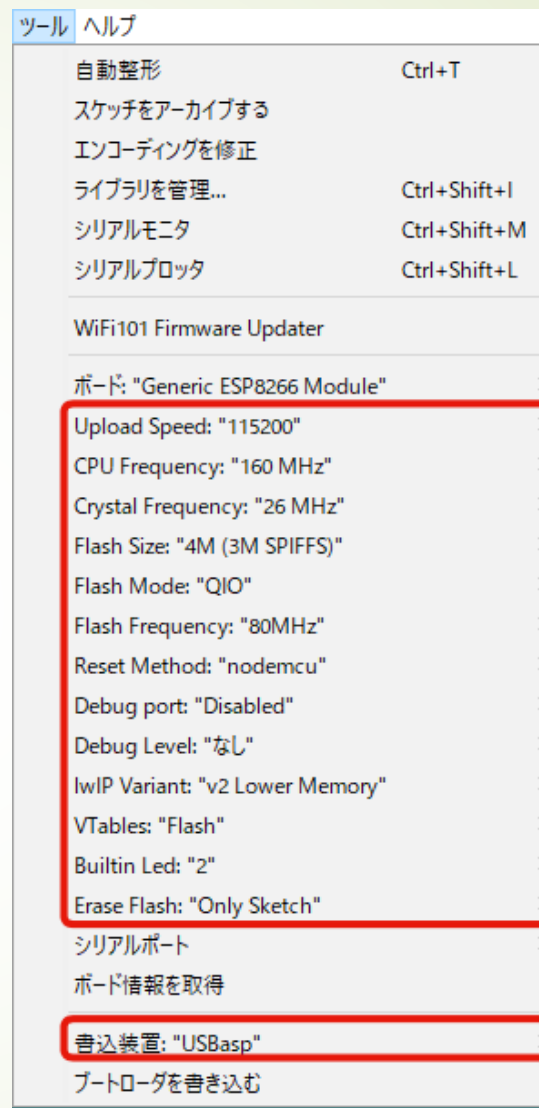
# シリアルポートが読み込まれないとき 2

- ▶ <https://www.monoxit.com/tecmat/windows/ftdiusb/>
- ▶ なんのドライバーをインストールする必要があるかを、確認して手動で更新する
  - ▶ 例では、FTDI社のドライバーをインストールしている
- ▶ このエラーはパソコンの規格によって異なる
- ▶ ‘読み込まれないドライバ名 手動’ などとググってがんばる



# ツール設定の変更

- ツールを選択して、赤枠と同じ設定に変更する



# 全体の構成

## 準備編

- ① Arduino IDEのインストール、環境設定
- ② **Lチカを試してみよう**

## 実践編

- ③ 温湿度の計測をしよう
- ④ 測った温湿度をグラフ化してみよう

デバイス技術

システム技術



# Lチカ デジタル出力

- ▶ Lチカは“LEDチカチカ”の略である
- ▶ マイコン工作において、動作確認をする際にLEDをつないでチカチカと点滅させることからきている。

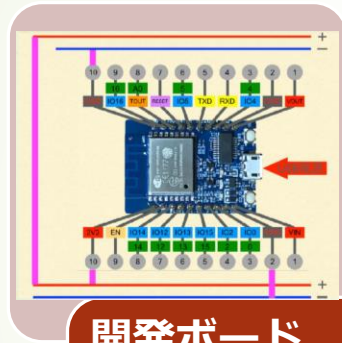


# LEDをチカチカさせよう！ ⇒LEDに電気信号を送る！



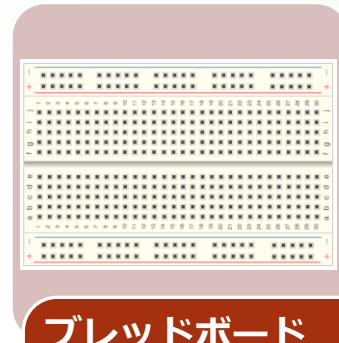
## Arduino

- 命令の記述 (プログラミング)



## 開発ボード

- 命令の記憶
- 命令を電気信号に翻訳



## ブレッドボード

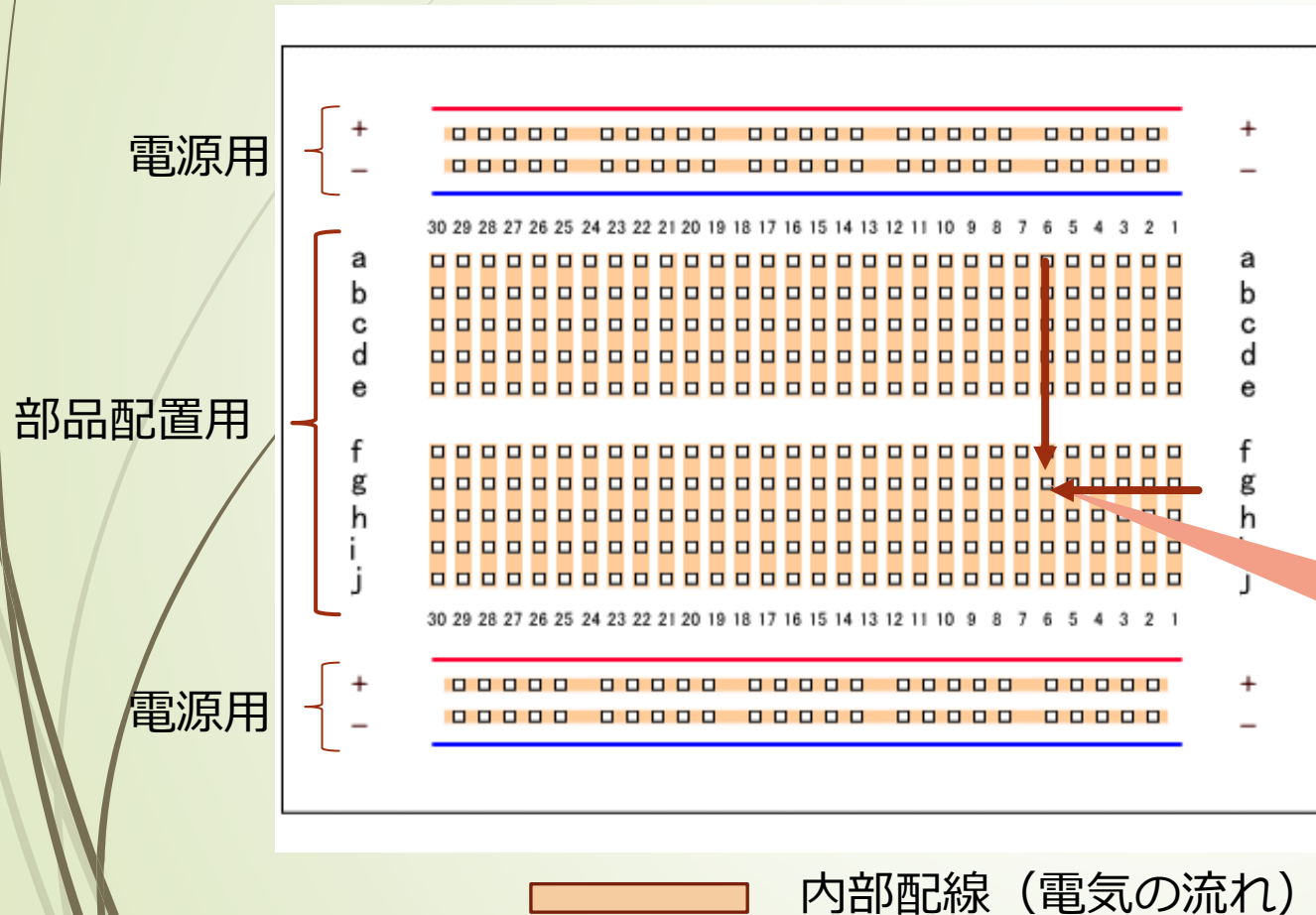
- 電気信号が通る道/場



## LED

- 電気信号を受け取り点滅

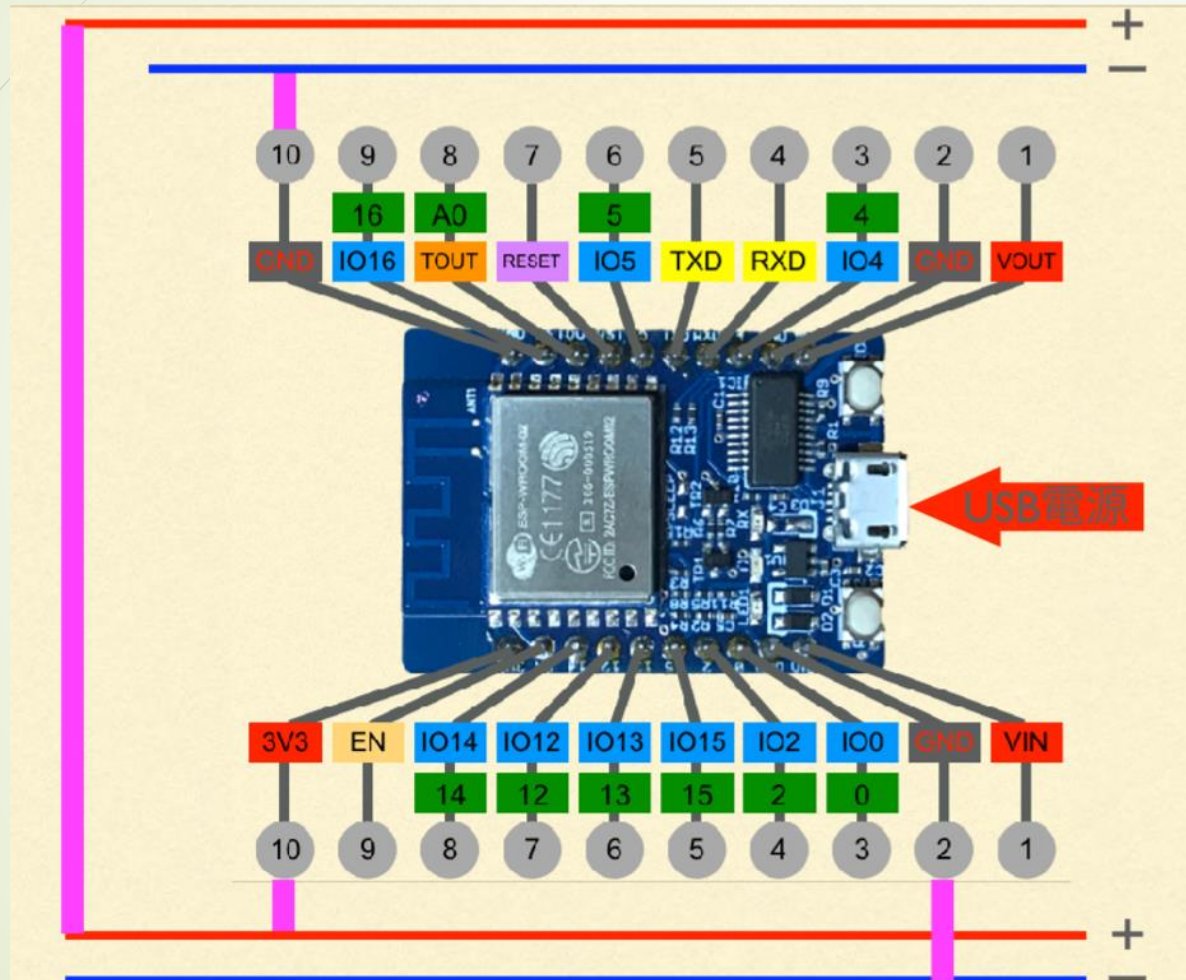
# ブレッドボードの配線



- 上から二列、下から二列は電源用。  
配線は横になっている
- 中の10列は部品を配置するためにある。配置する位置はアルファベットと数字によって座標で表せる  
配線は縦になっている。

座標表示はG 6

# 開発ボードの配線

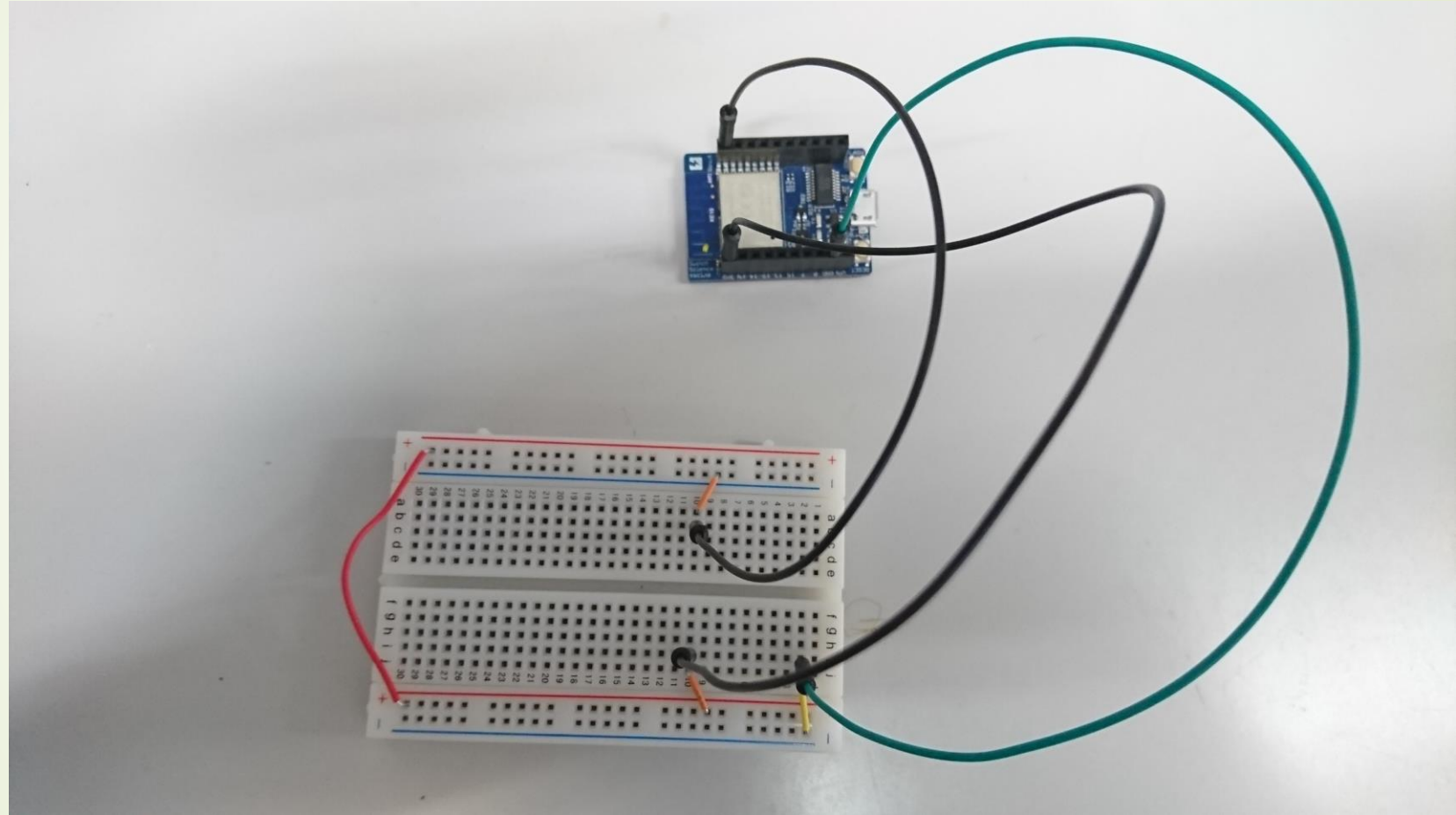


命令を処理したり記憶したりする場所

開発ボードのピンには、それぞれ異なる機能がついている  
今回のマニュアル内で  
上段を↑、下段を↓で表す。

いまは、開発ボードと  
ブレッドボードがつながっていないので電気を通す

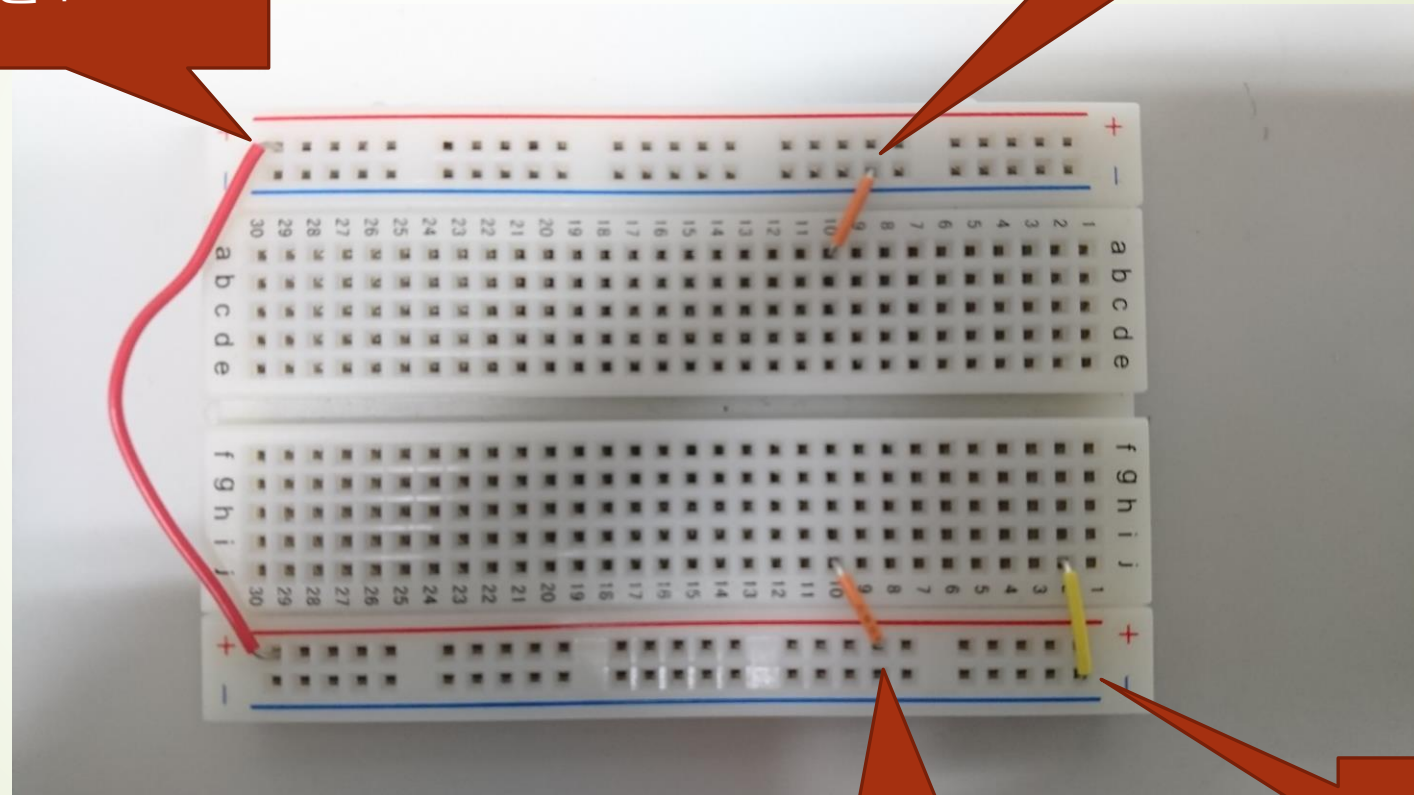
# 準備：開発ボードとブレッドボードの配線



# ブレッドボードの配線

+と+

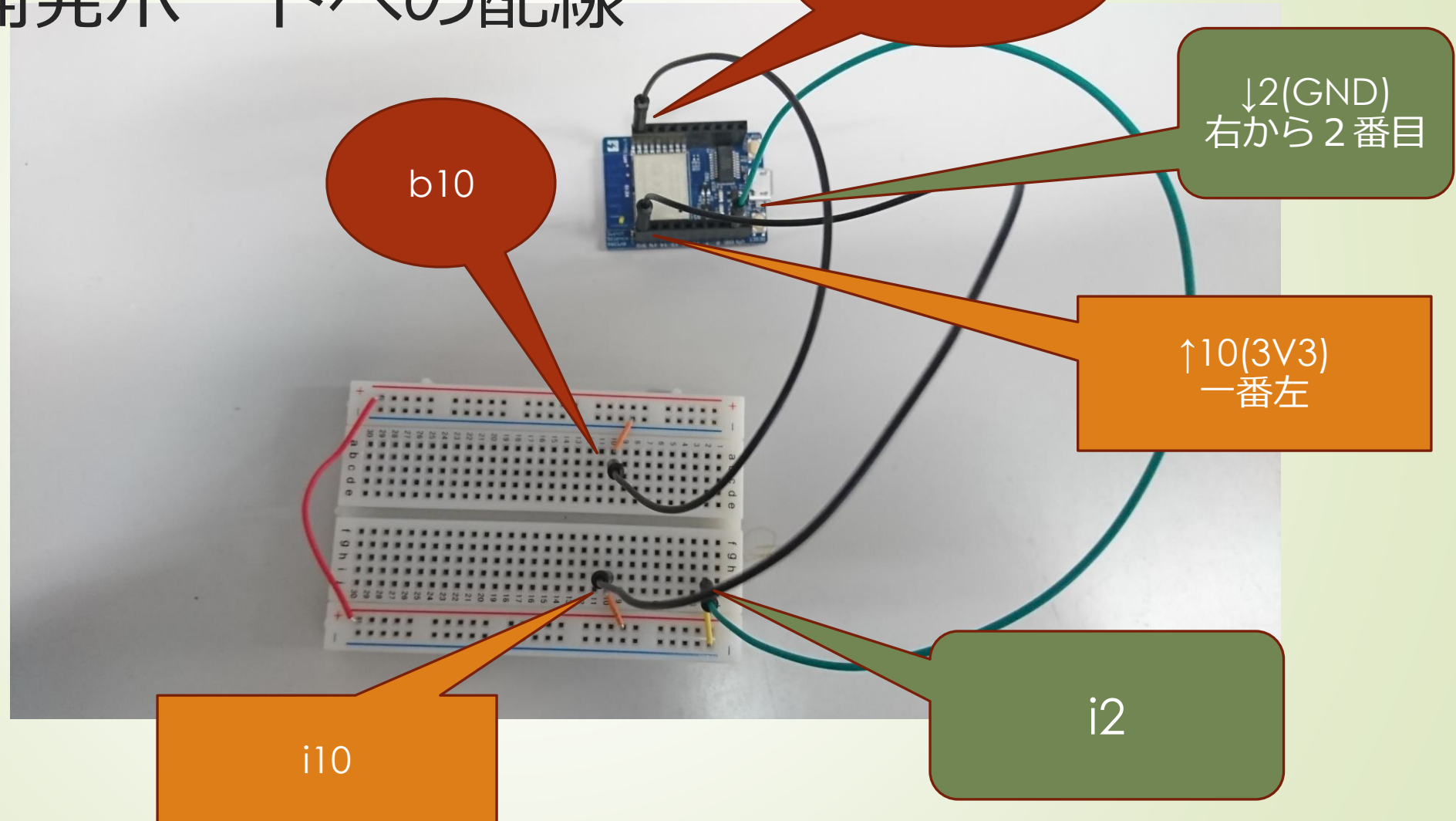
-とa10



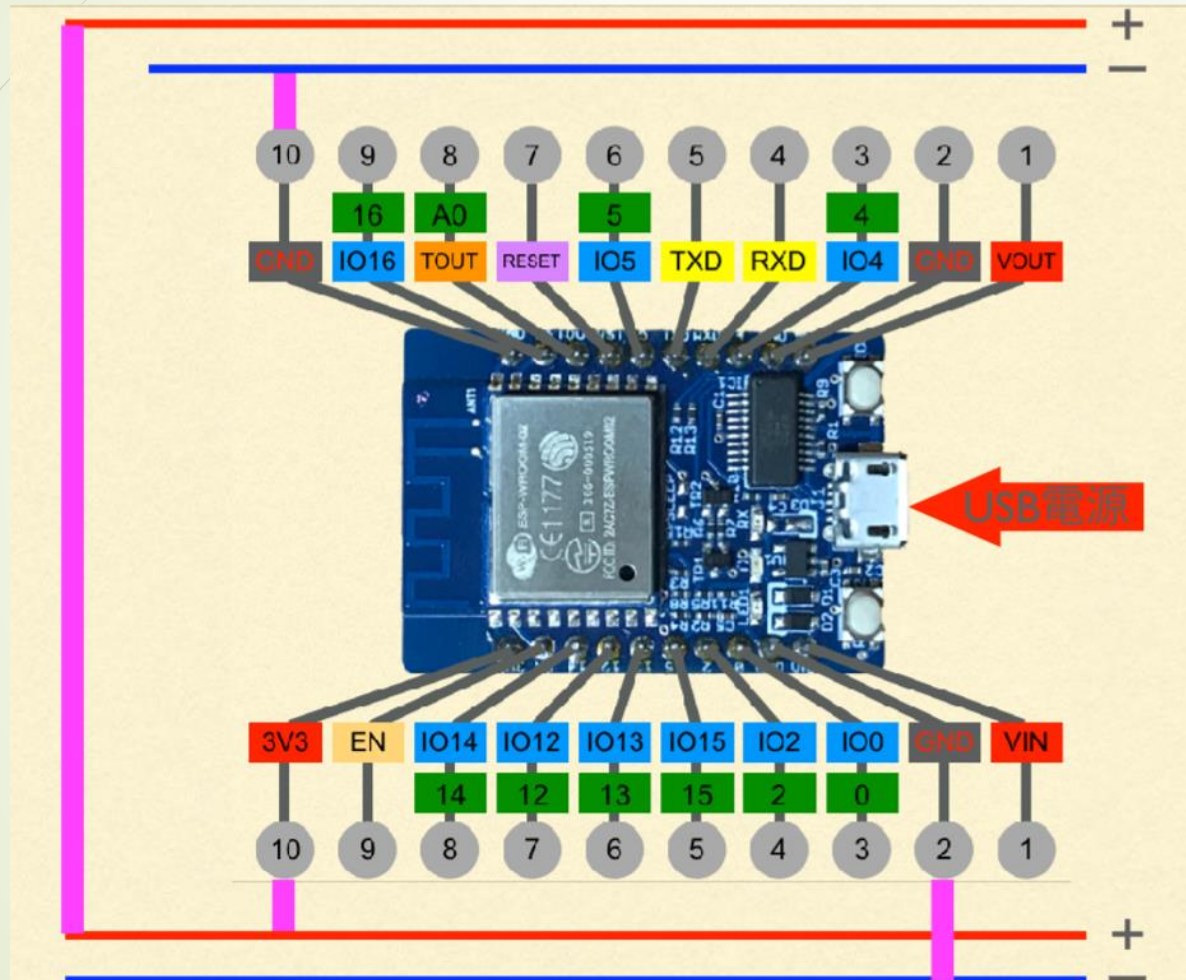
+とj10

-とj2

(足が短いやつ)  
開発ボードへの配線



# 開発ボードの配線

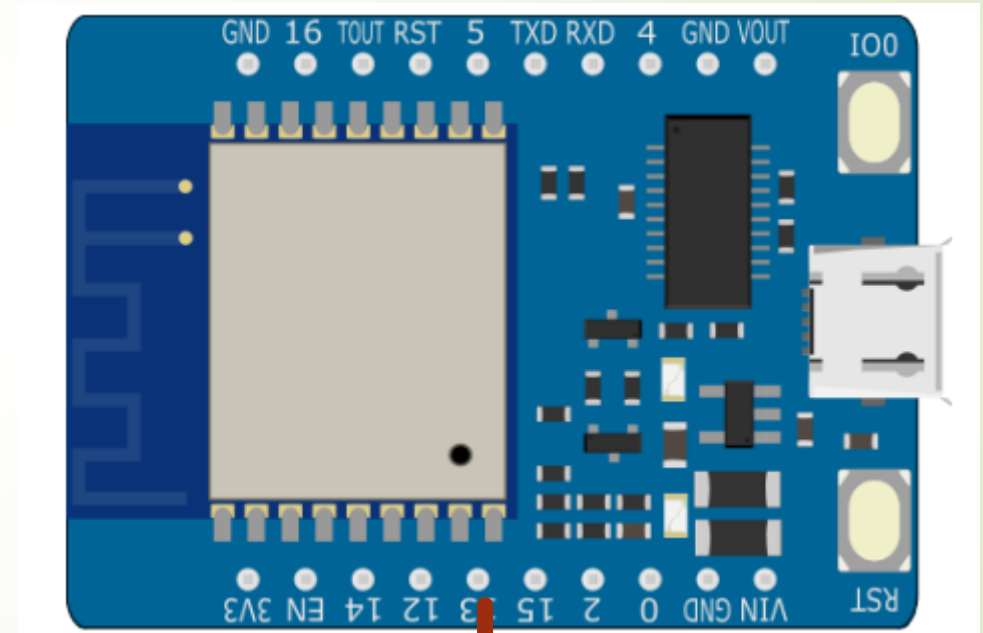


ブレッドボード内の赤線沿いには、電圧がかかっている

青線沿いはGNDに接続しているため、電圧がかかっていない状態



# Lチカ回路のイメージ



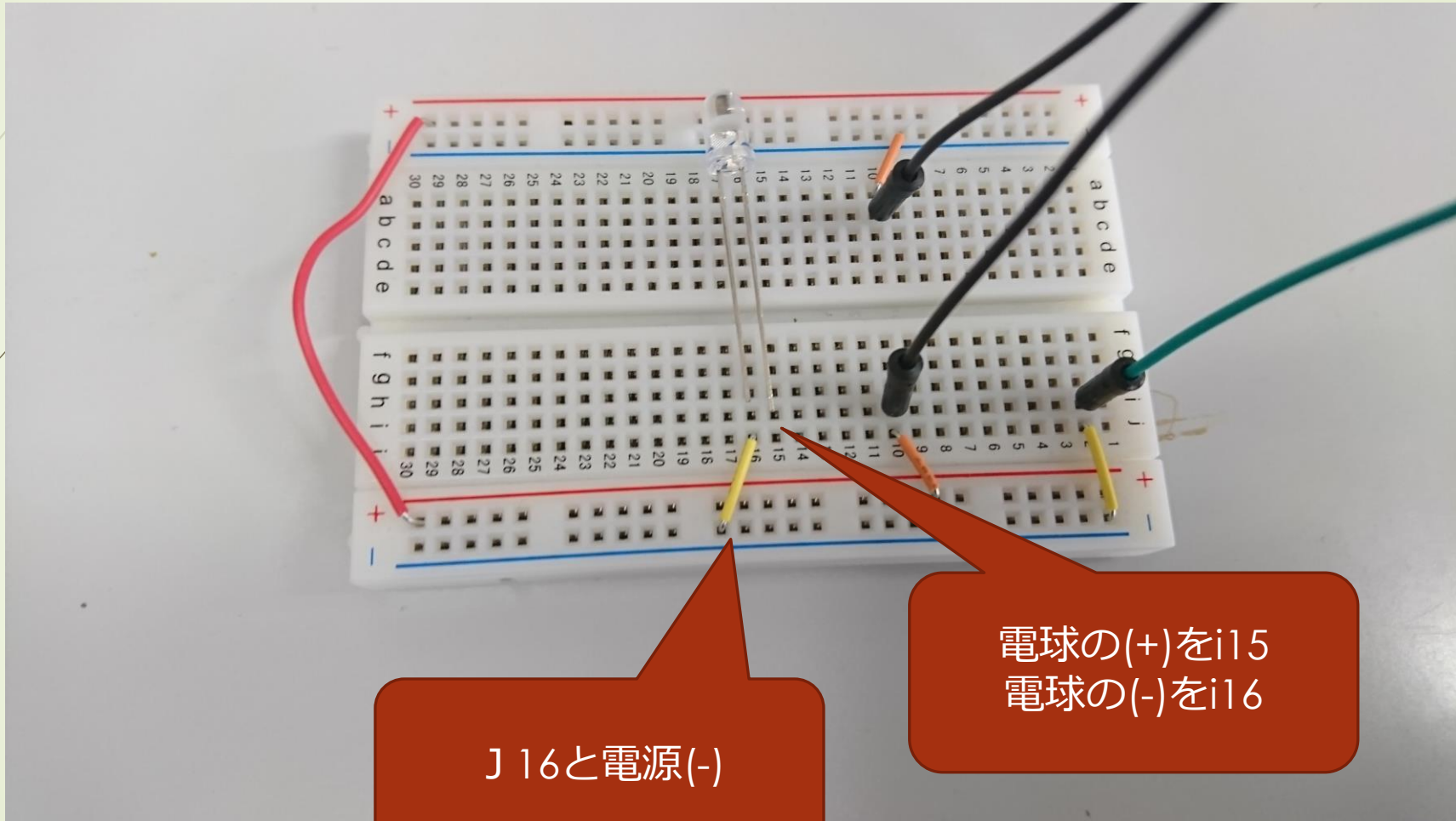
# LEDの極性

短い方が-

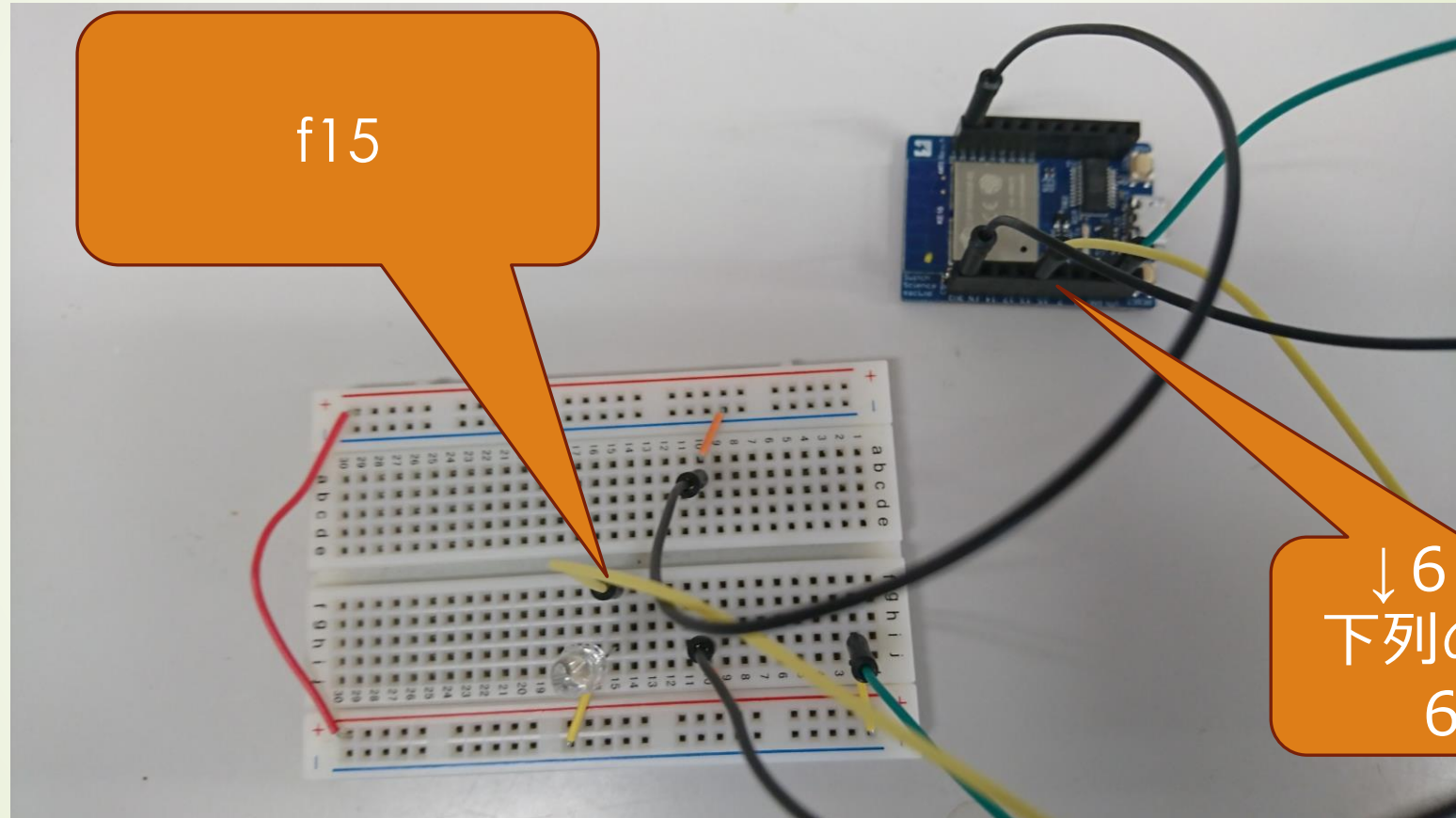


足の長い方が+

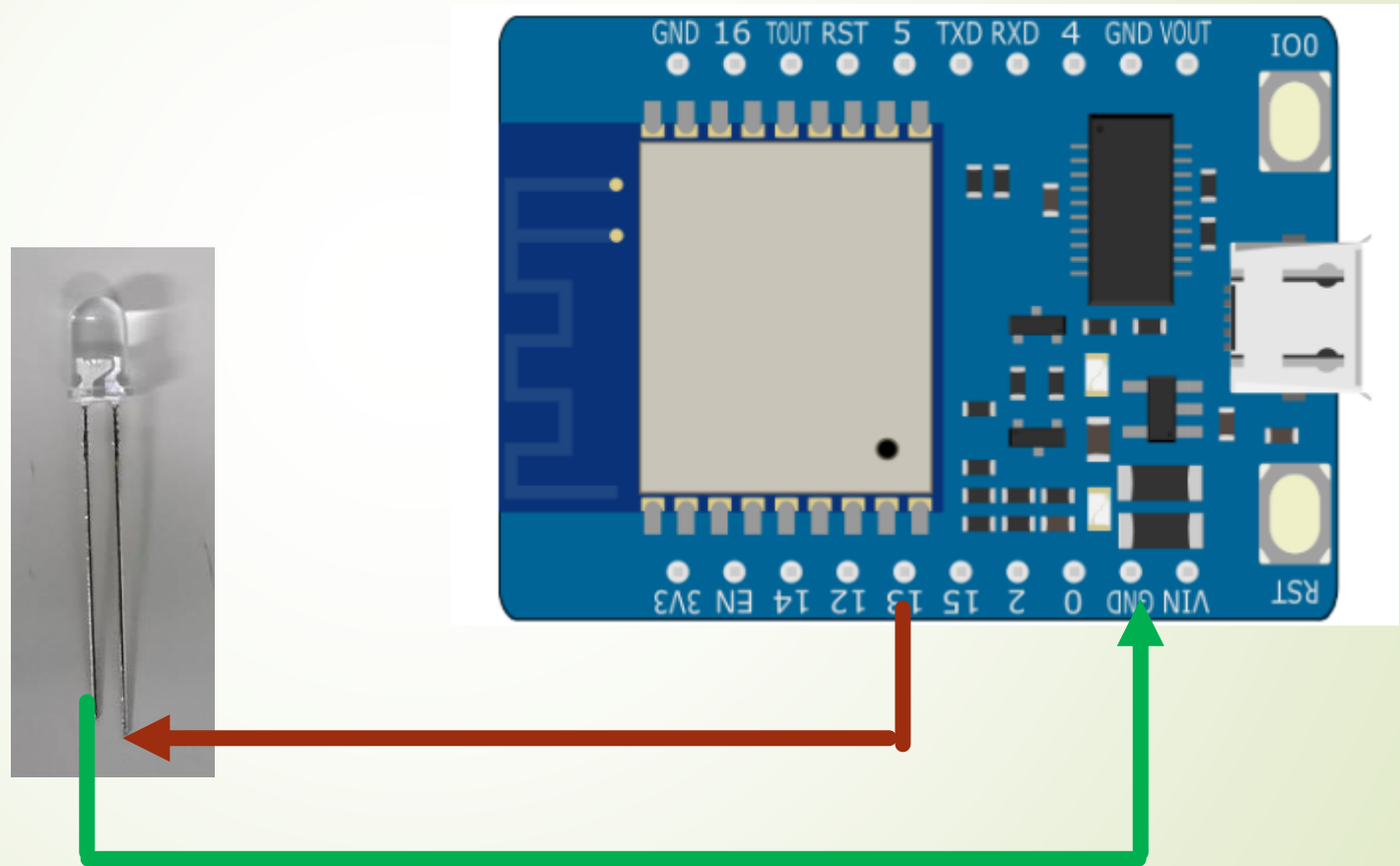
# Lチ力配線



# Lチカ ボードと配線



## 回路のイメージ



# Arduinoのプログラムの構造

ファイル 編集 スケッチ ツール ヘルプ



sketch\_may07a §

```
void setup() {
```

最初に一回だけ行う

```
}
```

```
void loop() {
```

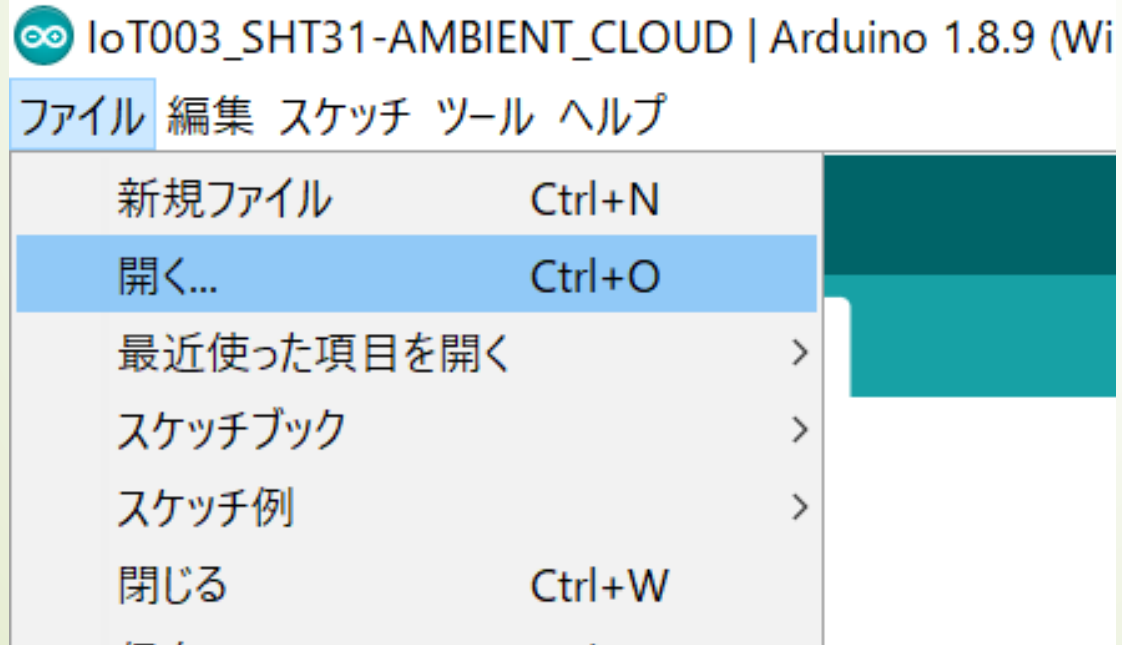
繰り返し行う

```
|
```

```
}
```

# ソースコードの開き方

- ▶ IDEを起動して[ファイル]メニューの[開く]で開く。
- ▶ Lチカでは、デスクトップにある[ソースコード]のフォルダ IoT1-LEDを開いてください



# Lチカのソースコード

IoT-LED-blink | Arduino 1.8.9 (Windows Store 1.8.21.0)

ファイル 編集 スケッチ ツール ヘルプ



IoT-LED-blink

```
#define LED 13

void setup() {
  Serial.begin(115200);
  delay(100);

  Serial.println("");
  Serial.println("LED BLINK TEST");

  pinMode(LED, OUTPUT);
}

void loop() {
  Serial.println("On");
  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second

  Serial.println("Off");
  digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

上のループで点灯  
下のループで消灯



# コンパイルと読込

矢印マークを  
クリックすると  
マイコンへの  
書き込みが始ま  
る



```
IoT-LED-blink | Arduino 1.8.9 (Windows Store 1.8.21.0)
ファイル 編集 スケッチ ツール ヘルプ

IoT-LED-blink
#define LED 13

void setup() {
  Serial.begin(115200);
  delay(100);

  Serial.println("");
  Serial.println("LED BLINK TEST");

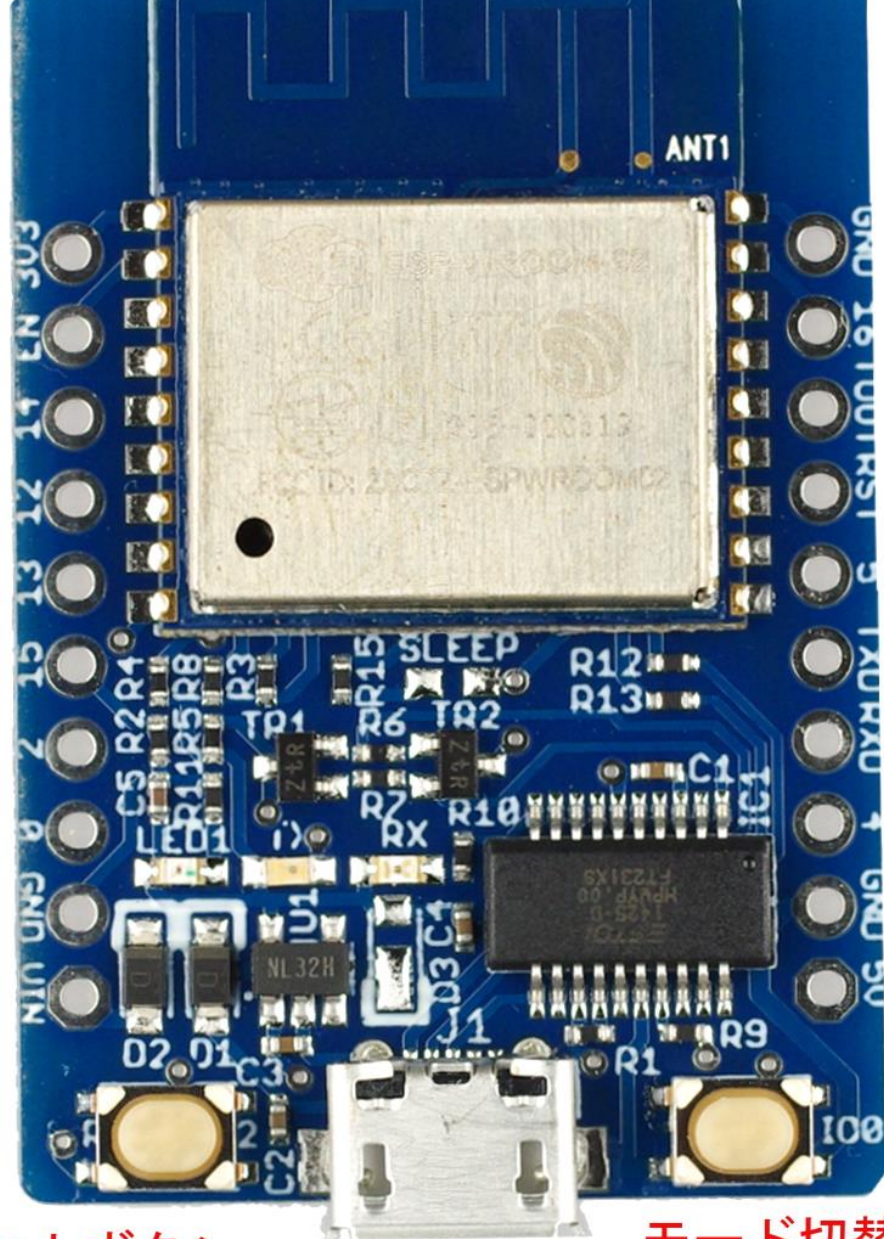
  pinMode(LED, OUTPUT);
}

void loop() {
  Serial.println("On");
  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second

  Serial.println("Off");
  digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

# 34 コンパイルとプログラムの書き込み

- ▶ シリアルコンソールが開いていたら閉じる
- ▶ ESPr DeveloperにUSBケーブルを接続する
- ▶ ツールからシリアルポートを選択
- ▶ シリアルモニターを開き、Developerのリセットボタンを押す
- ▶ **マイコンのFLASHボタンを押しながら、コンパイル&プログラミングの書き込みボタンをクリック**
- ▶ 書き込みが完了



リセットボタン

モード切替ボタン  
押すとIO0:L  
放すとIO0:H

# 書き込み完了

書き込みが完了すると下のところにお知らせが来る

```
Serial.println("Off");  
digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW  
delay(1000);           // wait for a second  
}
```

ボードへの書き込みが完了しました。

[ 64% ]

[ 96% ]

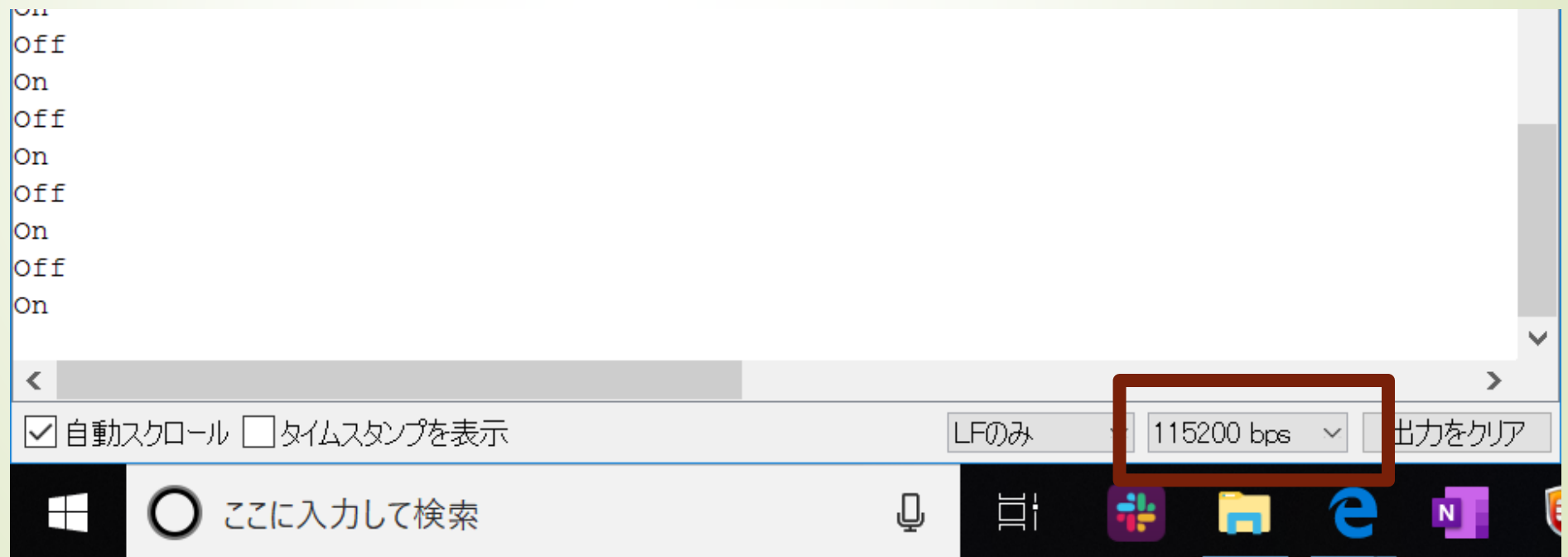
[ 100% ]

# コンソール出力

```
COM10
|
On
Off
On
Off
On
Off
On
Off
r□□1$?r??#□?n?□??▲□?▲□??□p?<????8□??δ??□p▲□▲?n?n?□?;?n?□??▲□?▲b?#1`□$`□?p?n?□?▲□?□?□?1□?□?□?▲□?▲1
LED BLINK TEST
On
Off
On
Off
On
Off
On
Off
```

# コンソール出力の方法

シリアルポートを開いて  
右下の枠のところ、115200bpsになっているか確認



# よくあるエラー

- ▶ シリアルポートが読み込めない
  - ▶ P.13-14をよく読む
- ▶ 書込装置をUSBaspに変更できない
  - ▶ Arduinoのバージョンを確認。1.8.12以下にする

# 全体の構成

## 準備編

- ① Arduino IDEのインストール、環境設定
- ② Lチカを試してみよう

## 実践編

- ③ **温湿度の計測をしよう**
- ④ 測った温湿度をグラフ化してみよう

デバイス技術

システム技術

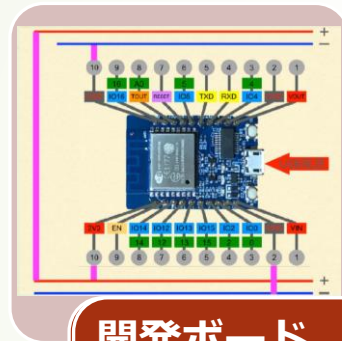
# 温湿度を測ろう！

## ⇒ 温湿度計から電気信号を受けとる！



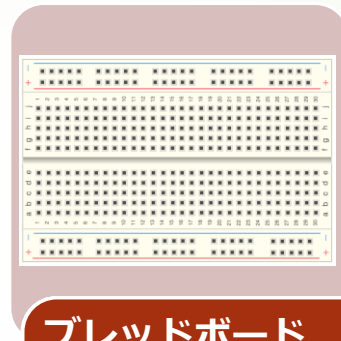
**Arduino**

- 温湿度の表示



**開発ボード**

- 温湿度計からの電気信号の翻訳



**ブレッドボード**

- 電気信号が通る道/場



**温湿度計**

- 温湿度を測り電気信号にする

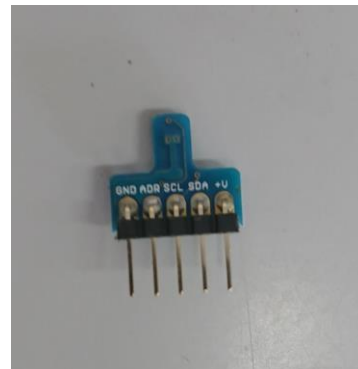
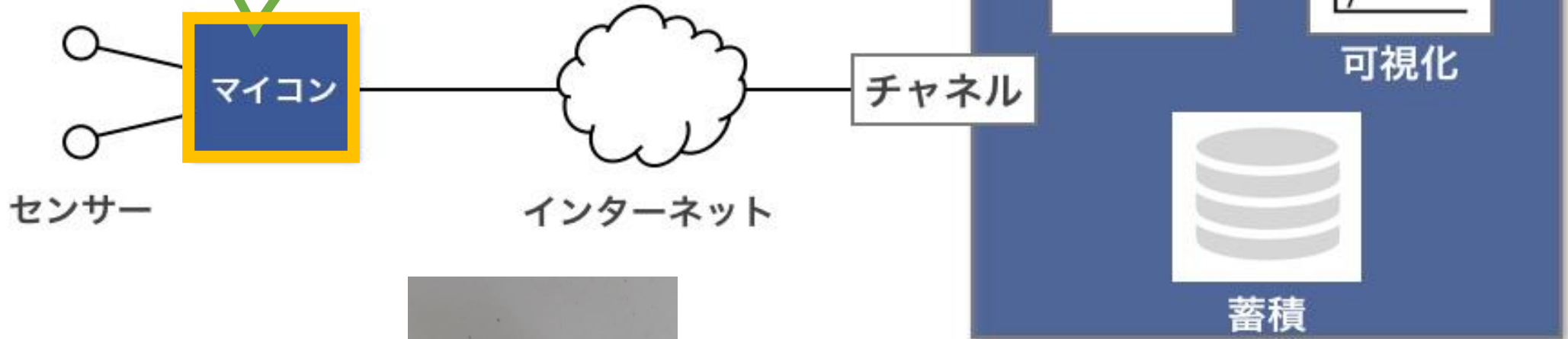


# 全体の構造

命令を処理する  
(ESP-WROOM32,  
switch science社)

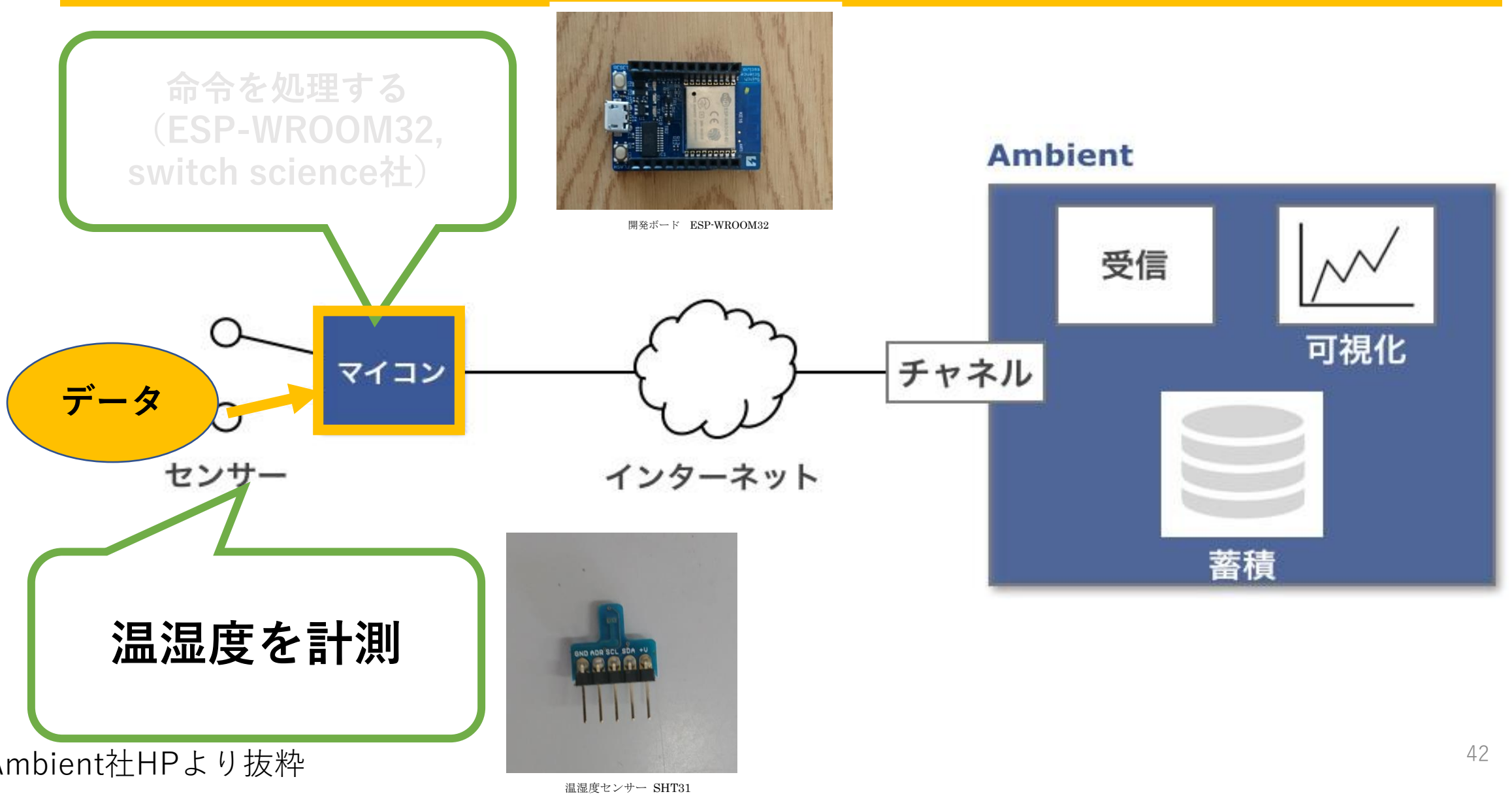


開発ボード ESP-WROOM32

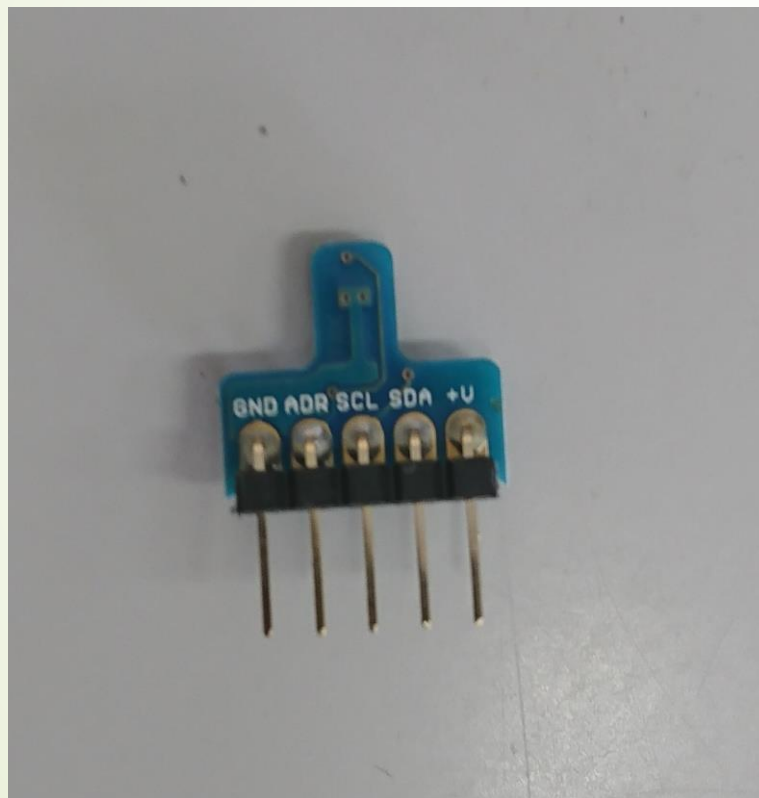


温湿度センサー SHT31

# 全体の構造

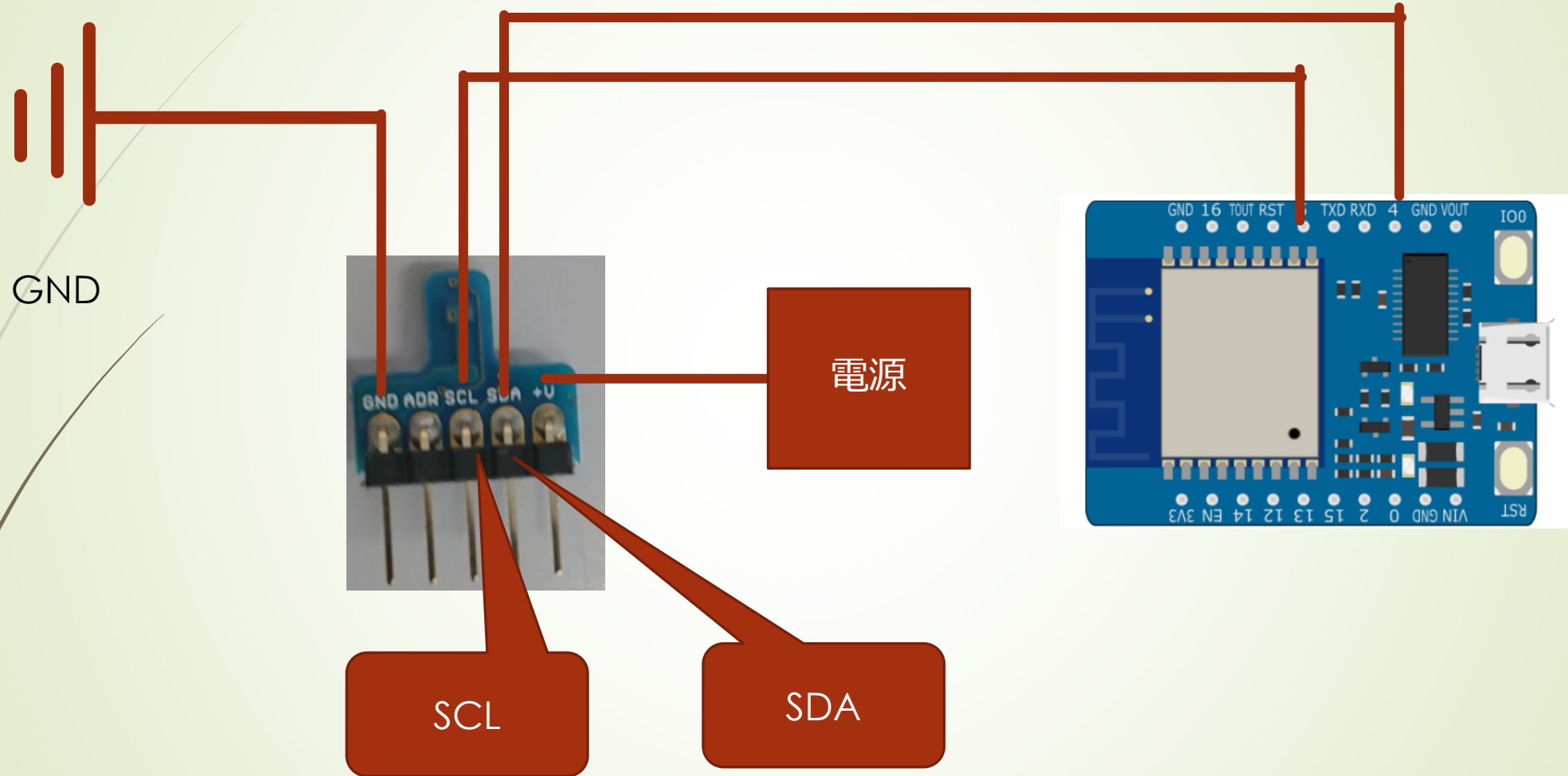


# 温湿度センサーをつなげる



- SENSIRION社製
- 温度  $-40^{\circ}\text{C}\sim+125^{\circ}\text{C}$  精度： $\pm 0.3^{\circ}\text{C}$   
(@ $0^{\circ}\text{C}\sim 90^{\circ}\text{C}$ )
- 湿度 0%~100% 精度： $\pm 2\%$   
(@ $0^{\circ}\text{C}\sim 90^{\circ}\text{C}$ )
- 価格 950円 (秋月電子通商  
<http://akizukidenshi.com/catalog/g/gK-12125/>)

## 作成する回路



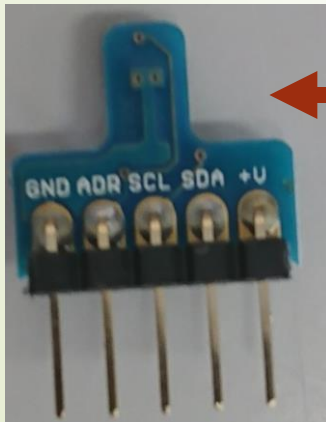
# 通信の仕方

信号 2 本で通信を行う

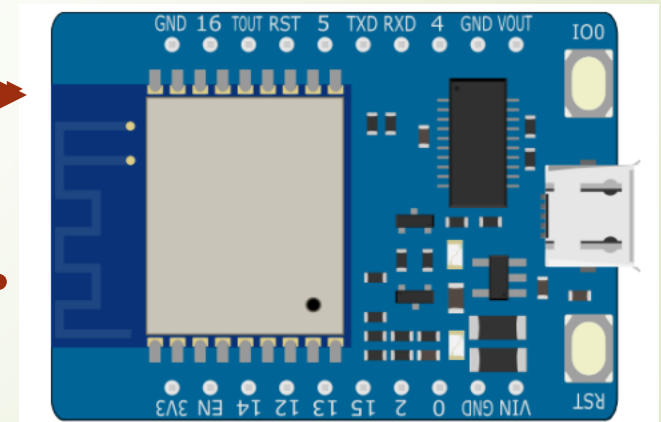
SCL (シリアルクロック線) : 同期をとるための信号線

SDA (シリアルデータ線) : SCLに同期してデータの転送に用いる信号線

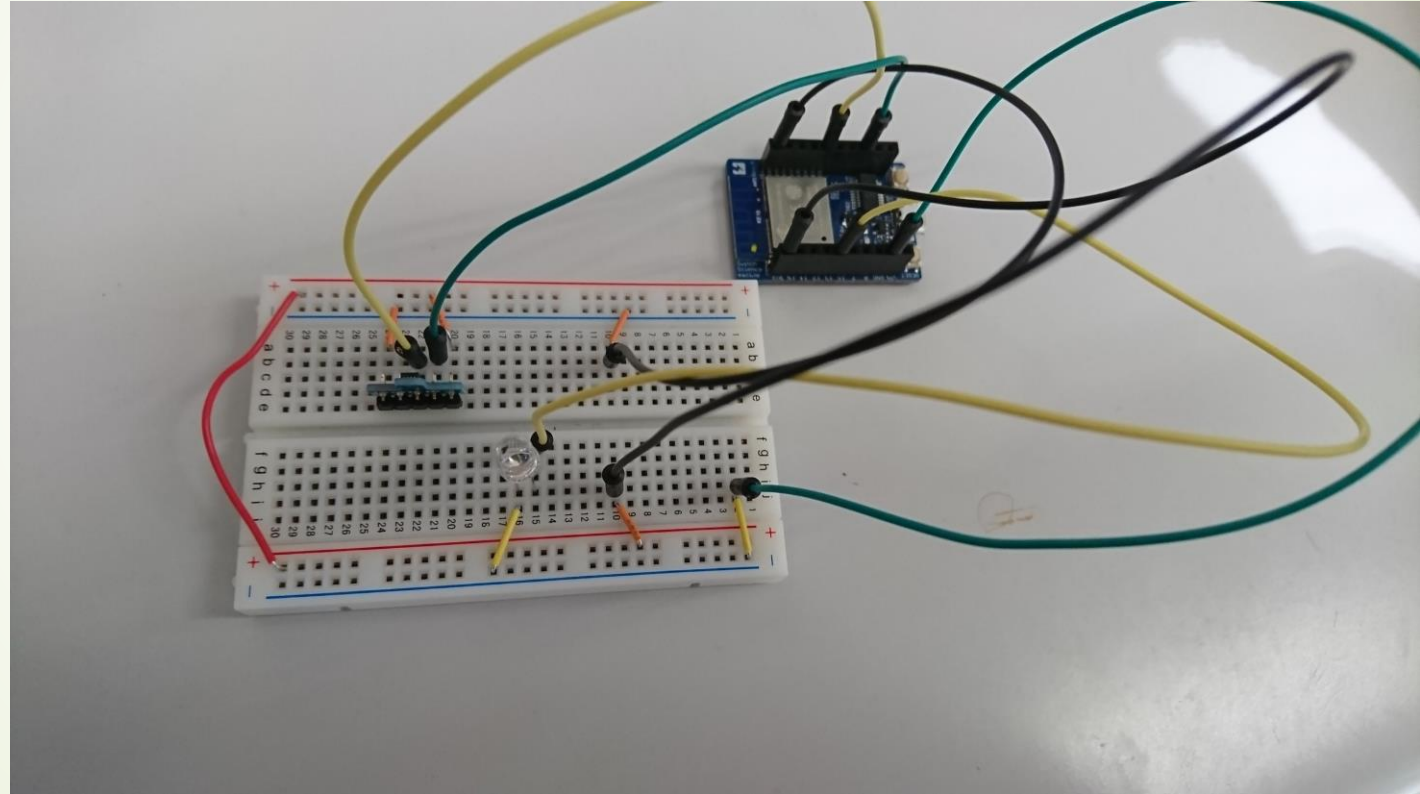
SDA  
(SCLがLowのときデータ変更)  
(SCLがHighのときデータ転送)



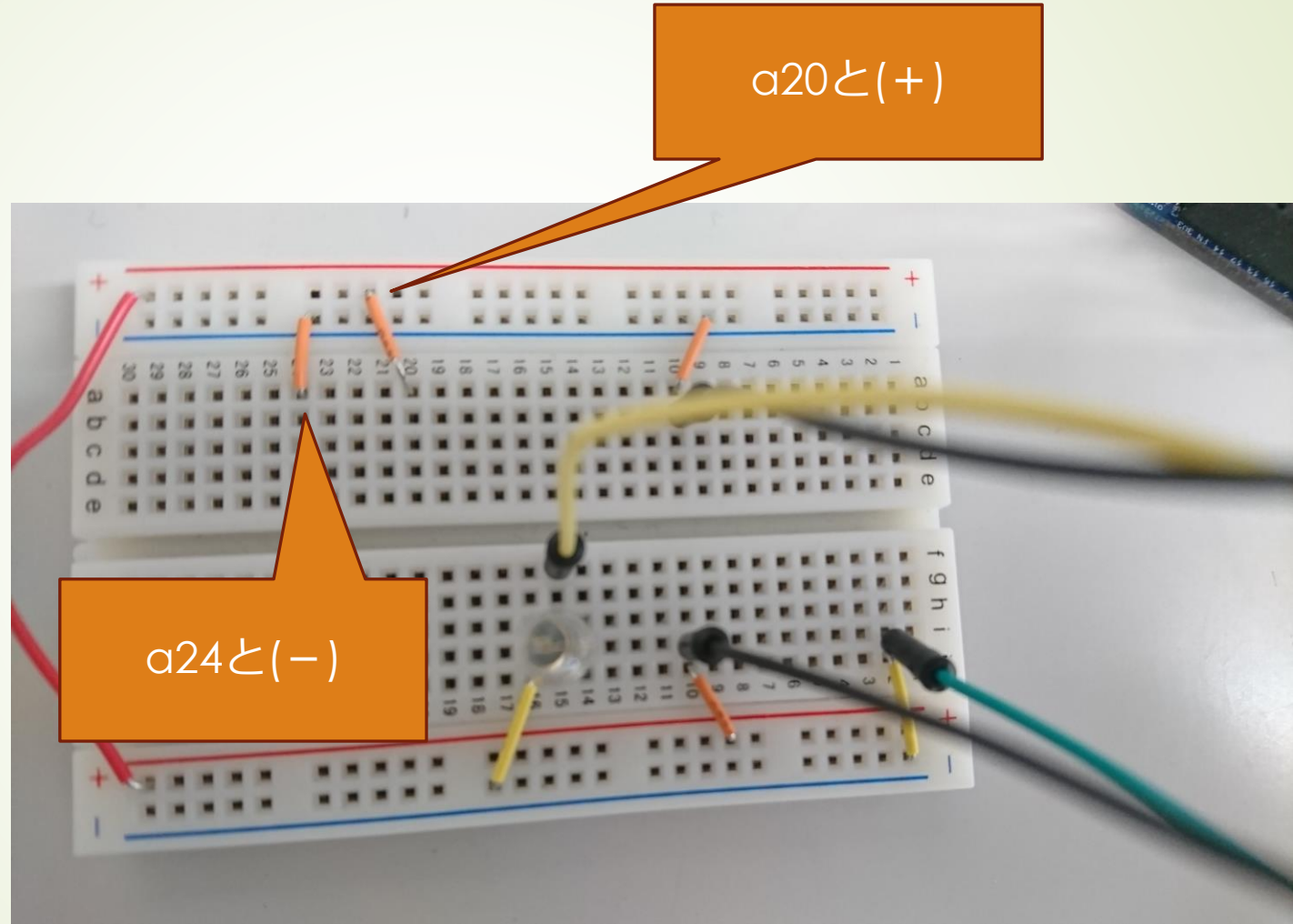
SCL  
LowとHighの切り替えをする



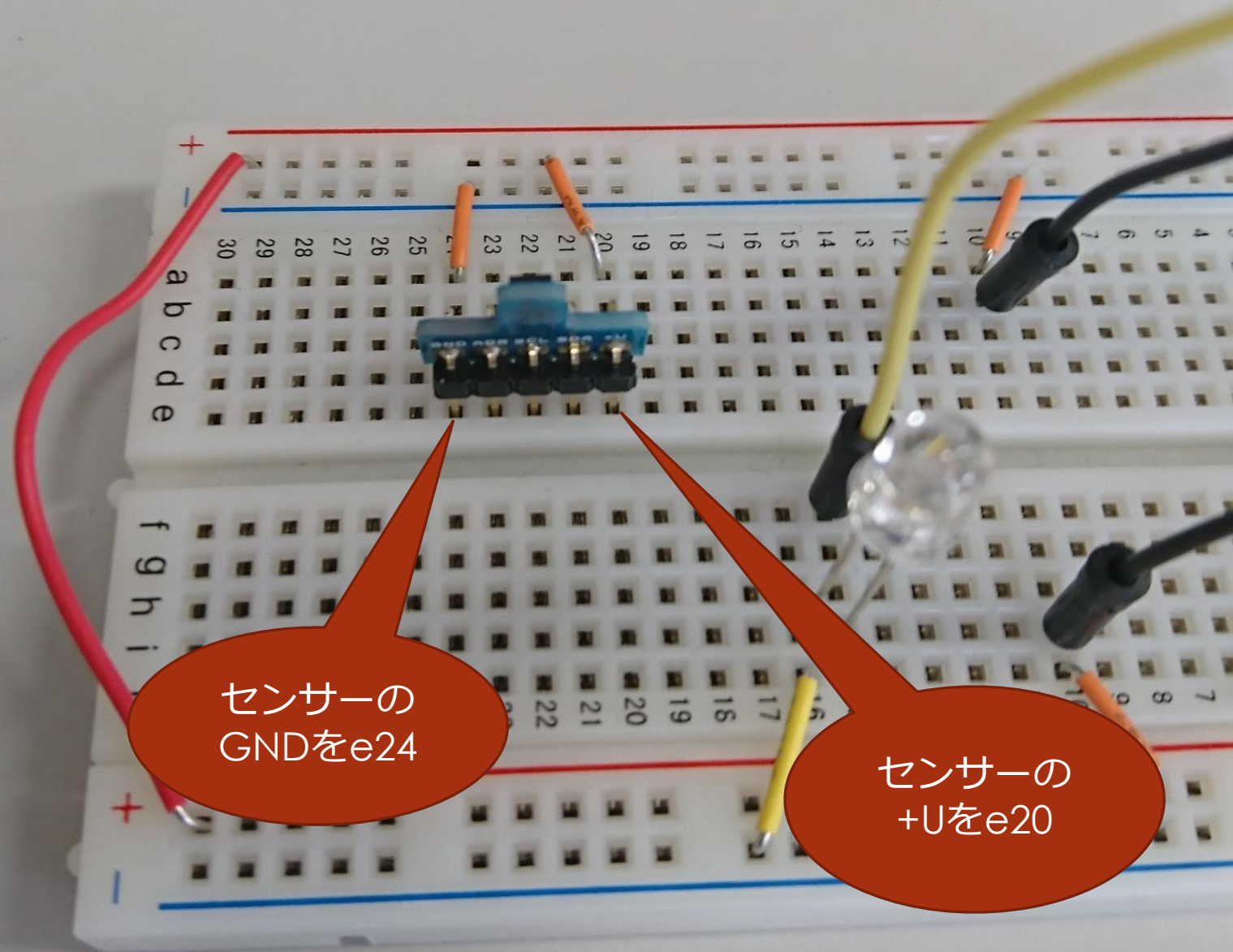
# 温湿度センサー配線 完成図



## 配線



# 48 センサー取り付け

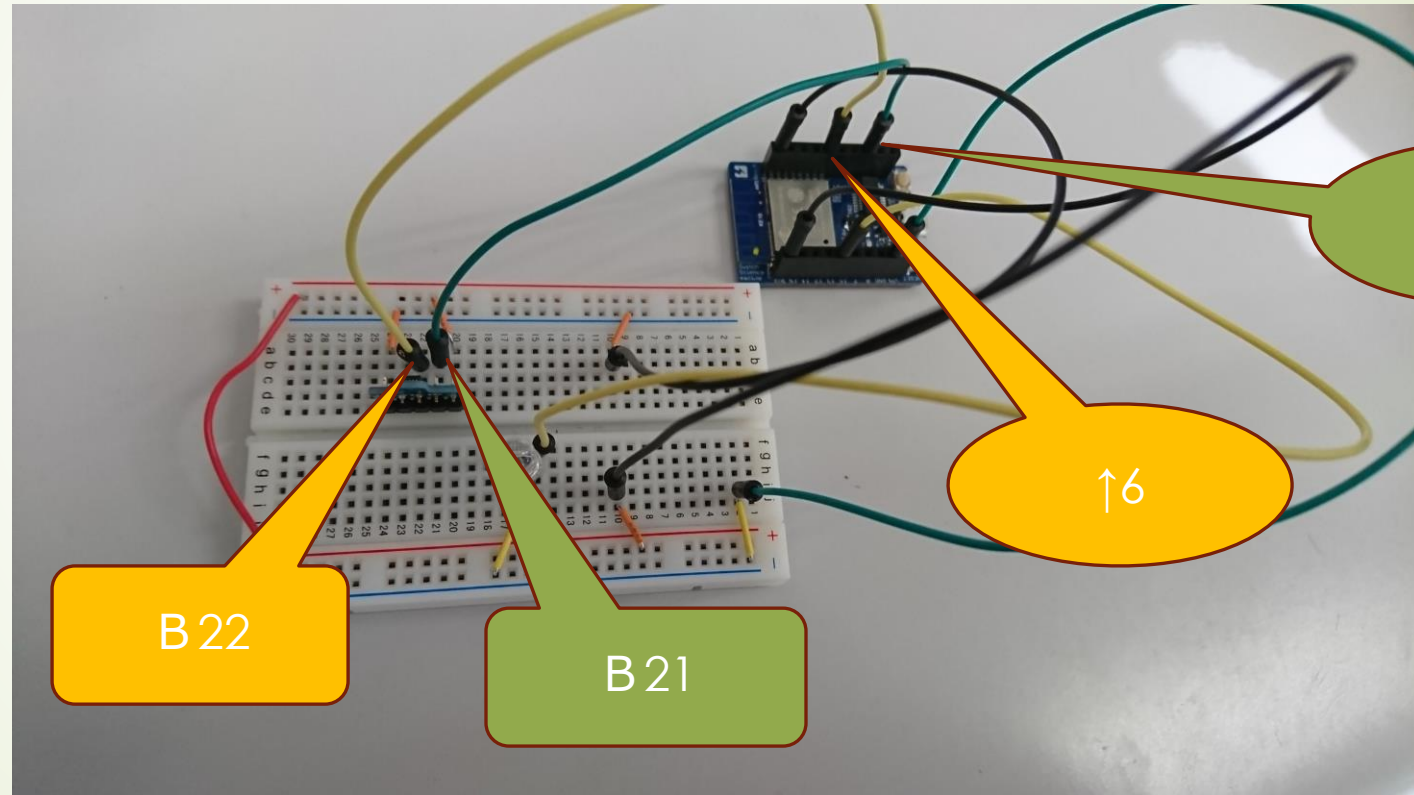


センサーの  
GNDをe24

センサーの  
+Uをe20

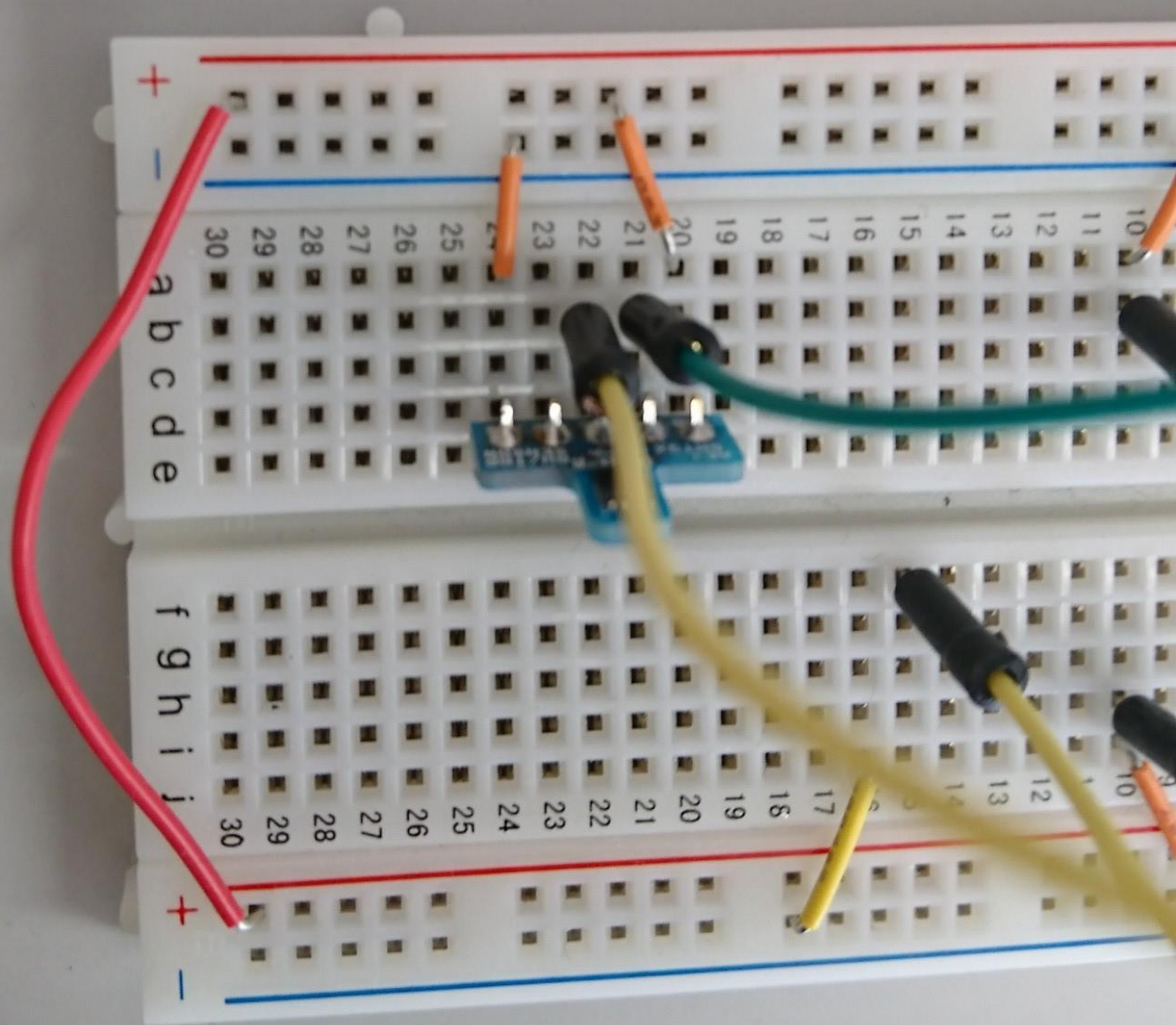


# センサーと開発ボードとの配線



# 完成イメージ 斜め上から見た図

- ▶ センサー周辺の配線確認



# コンパイルと読込しよう

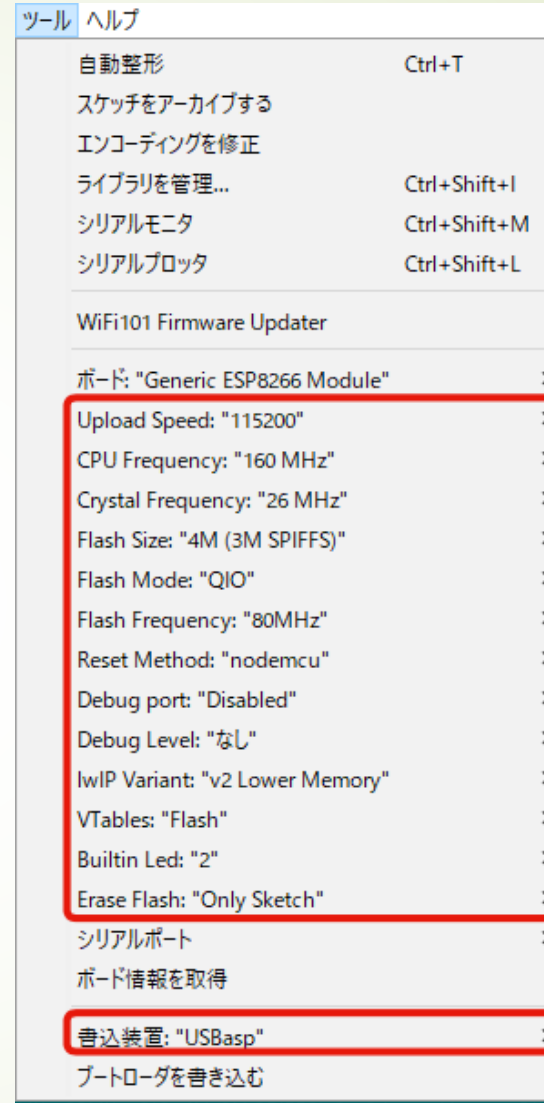
- ▶ [ソースコード]のIoT2\_sht31を開きましょう。
- ▶ 矢印をクリックしてコンパイル&読込をしましょう。

# ツール設定の確認

- ▶ ツールを選択して、赤枠と同じ設定に変更する
- ▶ 設定がリセットされる可能性がある

松岡修造  
MATSUOKA SHUZO

大丈夫!  
キミなら  
できる!



# 温湿度測定のコソール出力

COM10

```
temperature: 23.78 DegC,  humidity: 27.49 %
```

```
temperature: 23.79 DegC,  humidity: 27.42 %
```

```
temperature: 23.79 DegC,  humidity: 27.45 %
```

```
r□□l??r??#□?n?□??↑□?↑□??□p?<????8□??ø??□p↑□↑?nn?□?;?nE??↑□?↑b?#l`□$`□?p?n??□↑□  
SHT31 Test!!
```

```
temperature: 23.79 DegC,  humidity: 27.42 %
```

```
temperature: 23.79 DegC,  humidity: 27.46 %
```

```
temperature: 23.79 DegC,  humidity: 27.48 %
```

```
temperature: 23.79 DegC,  humidity: 27.54 %
```

```
temperature: 23.81 DegC,  humidity: 27.51 %
```

```
temperature: 23.79 DegC,  humidity: 27.52 %
```

# 全体の構成

## 準備編

- ① Arduino IDEのインストール、環境設定
- ② Lチカを試してみよう

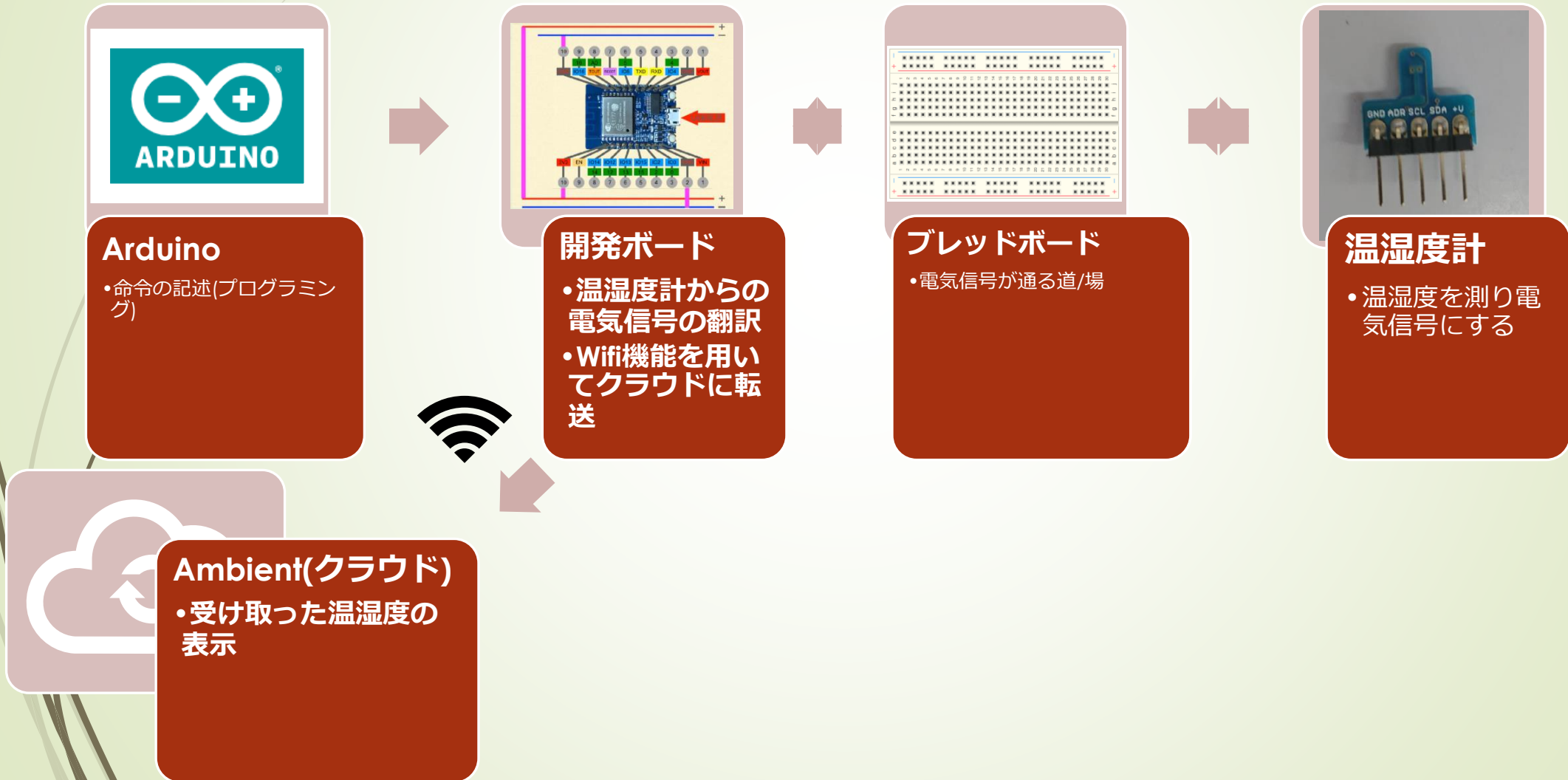
## 実践編

- ③ 温湿度の計測をしよう
- ④ 測った温湿度をグラフ化してみよう

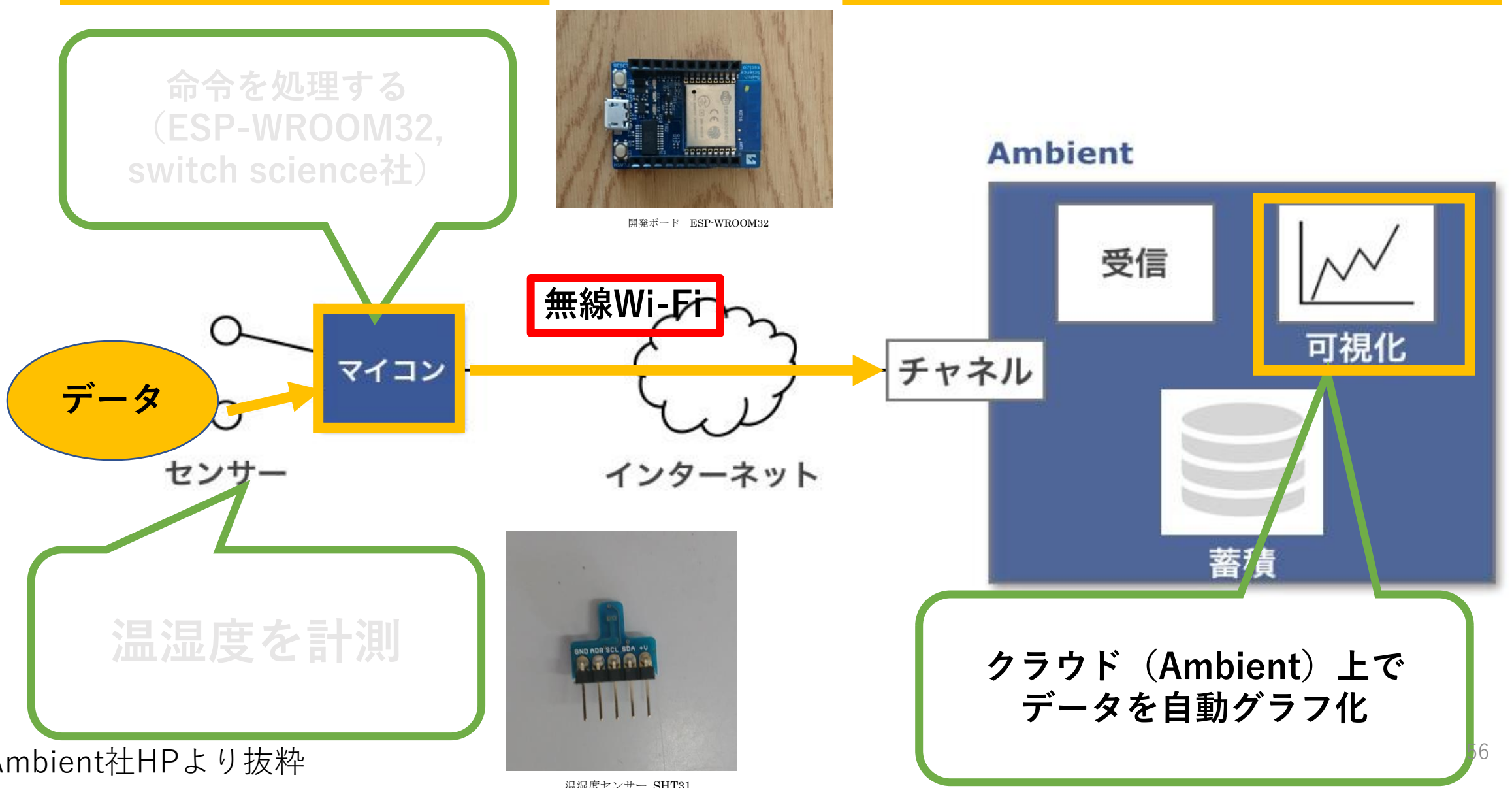
デバイス技術

システム技術

# 温湿度をクラウドにアップしよう！ ⇒ 温湿度計から電気信号をクラウドに飛ばす！



# 全体の構造





# Ambientの準備・設定

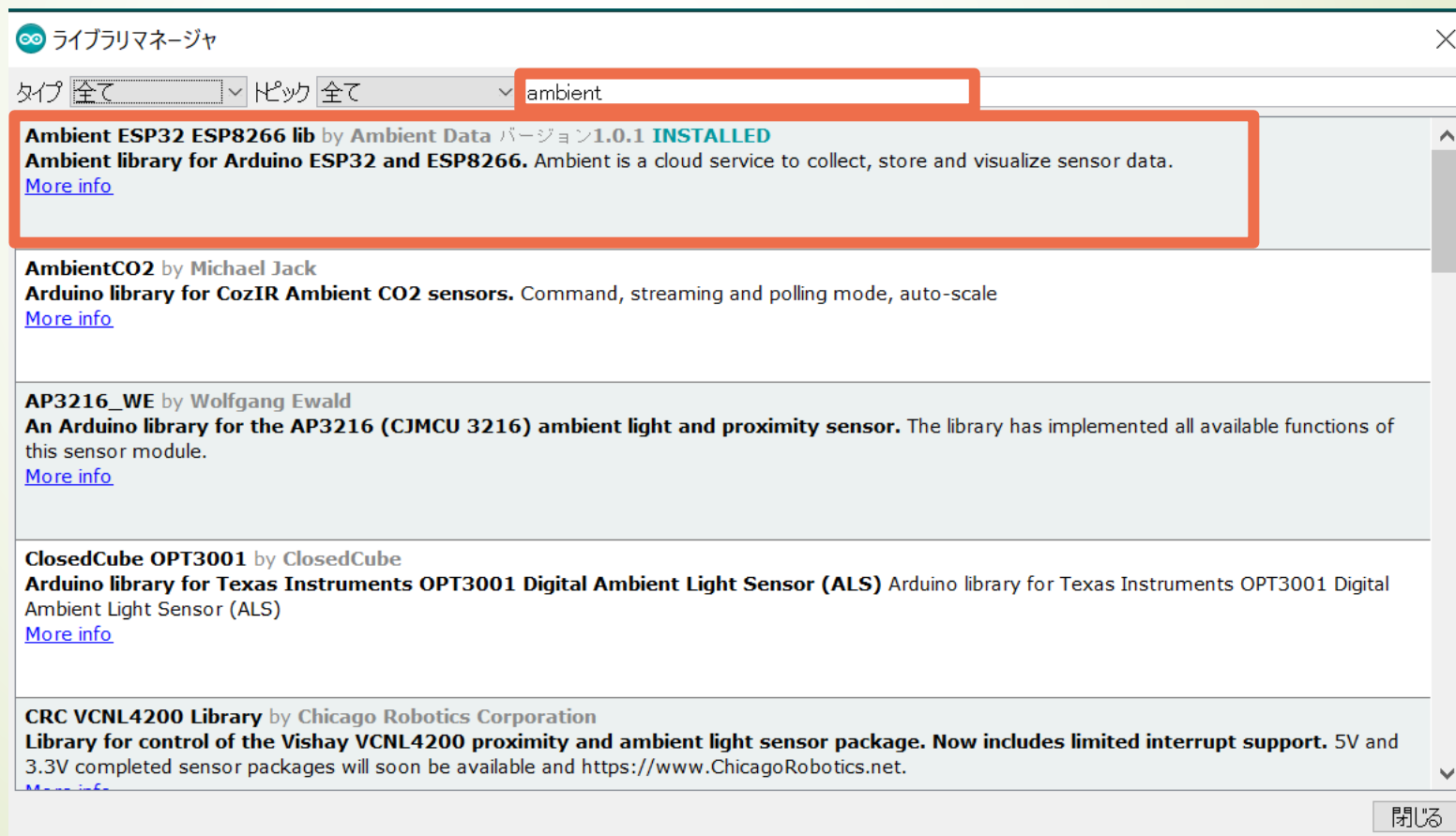
- ▶ ユーザ登録

<https://ambidata.io/usr/signup.html>

チャンネルを作り、チャンネルIDとライトキーを得る

# Ambientライブラリのインストール

- ▶ 「ツール」メニューの「ライブラリを管理...」を選択し、ライブラリマネージャ
- ▶ 検索窓に、'ambient'を入力し、表示された「Ambient ESP32 ESP8266 lib」をインストールする



# コードの変更

IoT3\_SHT31\_CLOUD | Arduino 1.8.10

ファイル 編集 スケッチ ツール ヘルプ



IoT3\_SHT31\_CLOUD

```
#include <ESP8266WiFi.h>
#include "Ambient.h"
#include "AE_SHT31.h"

#define LED 13
#define SDA 4
#define SCL 5

#define PERIOD 30

// ----- customize here -----
// WiFi Connection
const char* ssid = "wifiのSSIDに変更する";
const char* password = "wifiのパスワードに変更する";

// Ambient Channel Info
unsigned int channelId = チャンネルIDに変更;
const char* writeKey = "writekeyに変更";
// -----

WiFiClient client;
Ambient ambient;
// SHT31のアドレスを設定
AE_SHT31 SHT31 = AE_SHT31(0x45, SDA, SCL);
```

# Ambientソースコードについて

- ▶ IoT3\_SHT31\_CLOUD を読み込む
- ▶ コンパイル&書き込みボタンを押し、プログラミングを読み込む

# 最終的に得られるグラフ

