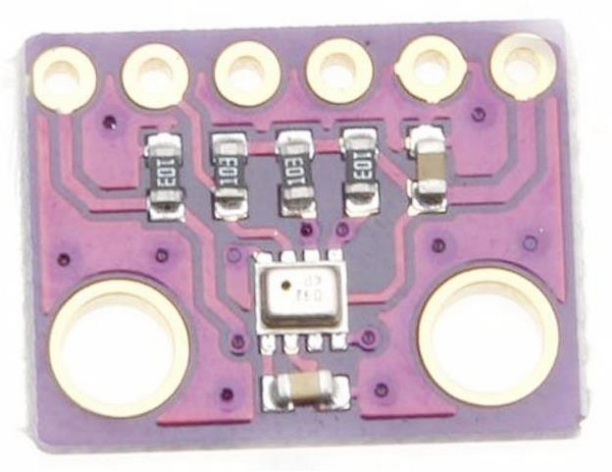
・Bosche BME280 センサーを使う



《何ができる？》前回まで、ボリューム（可変抵抗）や CdS センサ（光センサ）を使って、アナログ入力の値を Ambient クラウドサービスにアップロードした。ボリューム（可変抵抗）も光センサ（Cds センサ）もアナログ入力から値を取得している。アナログ入力を用いれば様々なアナログセンサーの値を取得することができる。
ただしアナログ入力は、センサーごとに入力ピン割り当てる。また ESP32 のアナログ入力の値は、ボードによってまちまちとなり、適切なセンサー値を得ることが難しい場合がある。
今回は、アナログ入力を使わずシリアル通信（SPI または I^2C）を行うセンサーボード BME280 で温度・湿度・大気圧のデータを取得して、Ambient クラウドにアップロードしグラフによる可視化を行いたいと思う。

【目標】

シリアル通信（SPI）アナログ値を取得する方法を知る。WiFi ステーション（クライアント）接続で、インターネット・サイトに接続し、シンプルな国産 IoT プラットフォーム「Ambient」を利用して、データのエントリと可視化を行う。

【1,ESP32 と電子工作部品との接続】

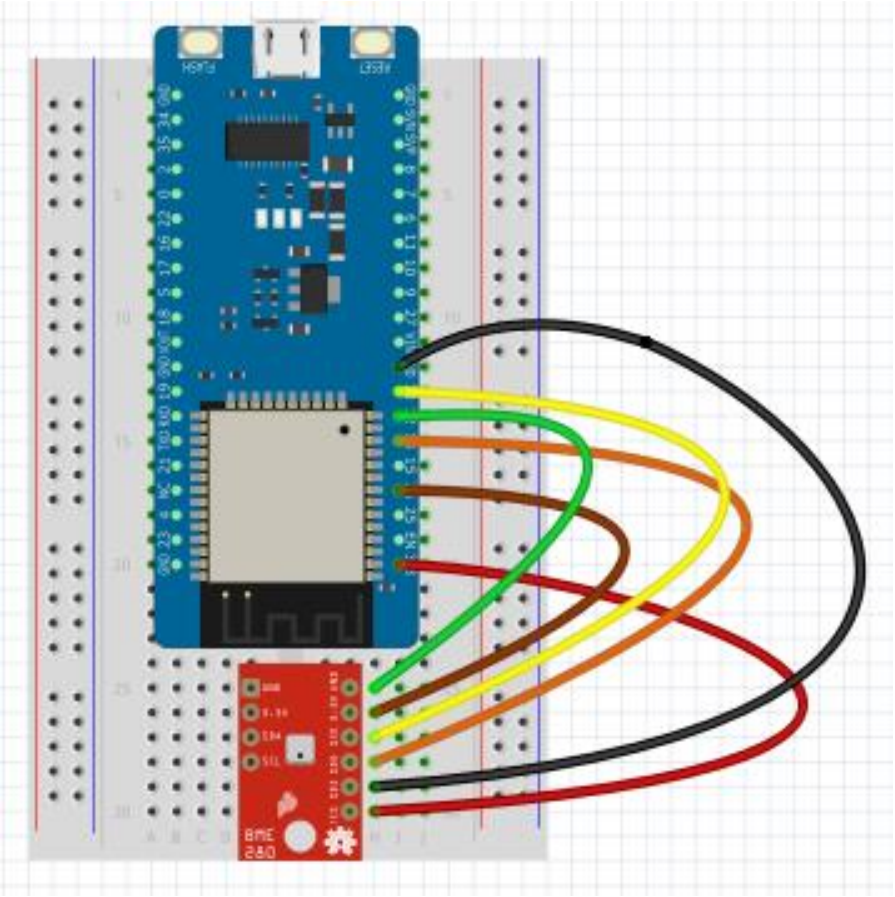
1.1.必要な部品

パーツ名	必要個数
Bosch BME280 センサー	1 個
ジャンパーコード	6 本（赤色、茶色、橙色、緑色、黄色、黒色 各 1 本ずつ） **※上記の色が揃わなければ別の色でも良い**

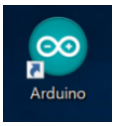
以下の接続を行う。以下の実体配線図を元に接続し、動作を確認すること。

① BME280 センサーをブレッドボード 1（ESP32 が載っている方）の g25 から g30 に差し込む。BME280 基板の裏のシルク印刷で VCC と印字されたピンが g30 に刺さるようにする。

② ジャンパーワイヤー（赤色）をブレッドボード 1 の i130 へ、もう片方を ESP32 の 3V3 ピンへ。
③ ジャンパーワイヤ（黒色）をブレッドボード 1 の i29 へ、もう片方を ESP32 の GND ピンへ。
④ ジャンパーワイヤ（橙色）をブレッドボード 1 の i28 へ、もう片方を ESP32 の IO14 ピンへ。
⑤ ジャンパーワイヤ（黄色）をブレッドボード 1 の i27 へ、もう片方を ESP32 の IO13 ピンへ。
⑥ ジャンパーワイヤ（茶色）をブレッドボード 1 の i26 へ、もう片方を ESP32 の IO26 ピンへ。
⑦ ジャンパーワイヤ（緑色）をブレッドボード 1 の i25 へ、もう片方を ESP32 の IO12 ピンへ。



【2. Arduino スケッチのサンプルプログラムを実行】

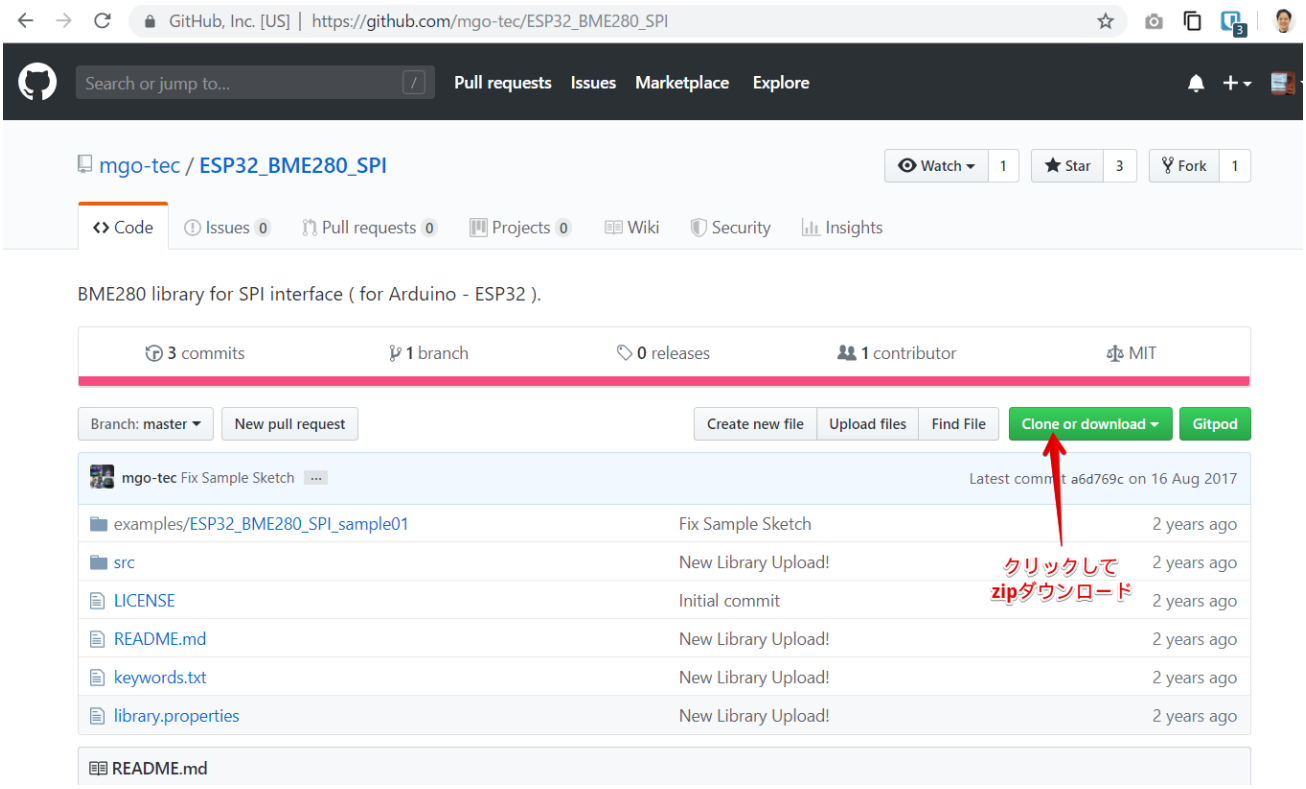


デスクトップのアイコンをダブルクリックして
Arduino IDE を起動する

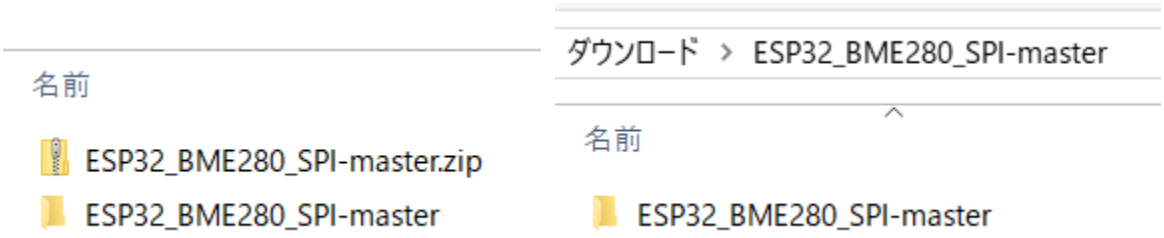
【課題 33】プロジェクト名「kad33_BME280」

のマイコン側プログラミングを行う前に、BME280 SPI 接続用のライブラリをダウンロードして libraries フォルダに格納しておく必要がある。

以下の github サイトでライブラリをダウンロードする。
https://github.com/mgo-tec/ESP32_BME280_SPI

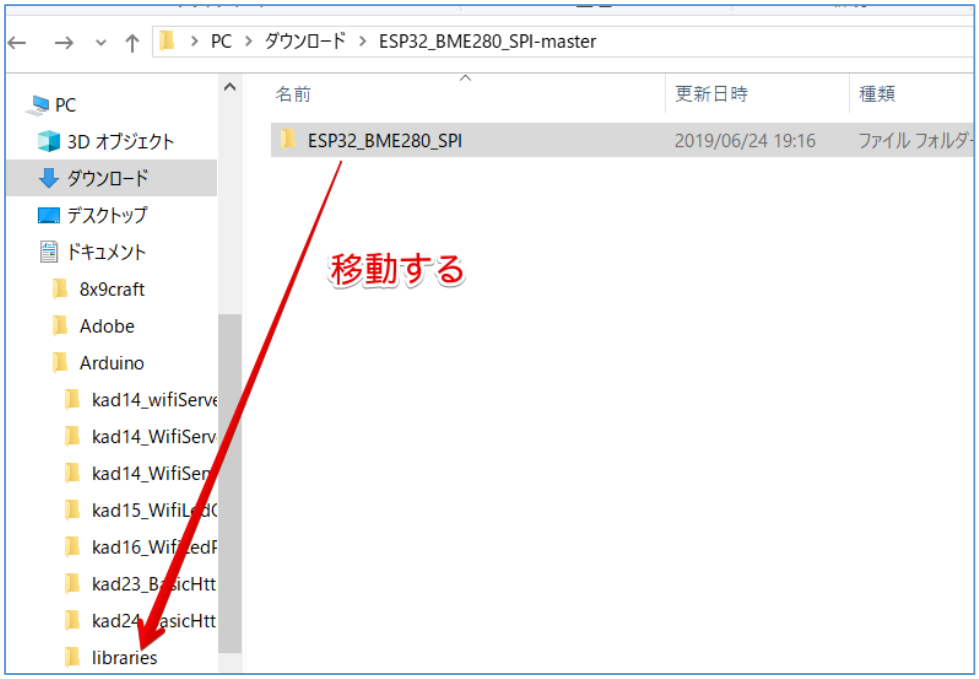


ダウンロードした ZIP ファイルを解凍して、フォルダを開く。



「ESP32_BME280_SPI-master」フォルダ内の「ESP32_BME280_SPI-master」フォルダを
「ESP32_BME280_SPI」に書き換えて、ドキュメントフォルダ内の Arduino フォルダにある「Libraries」

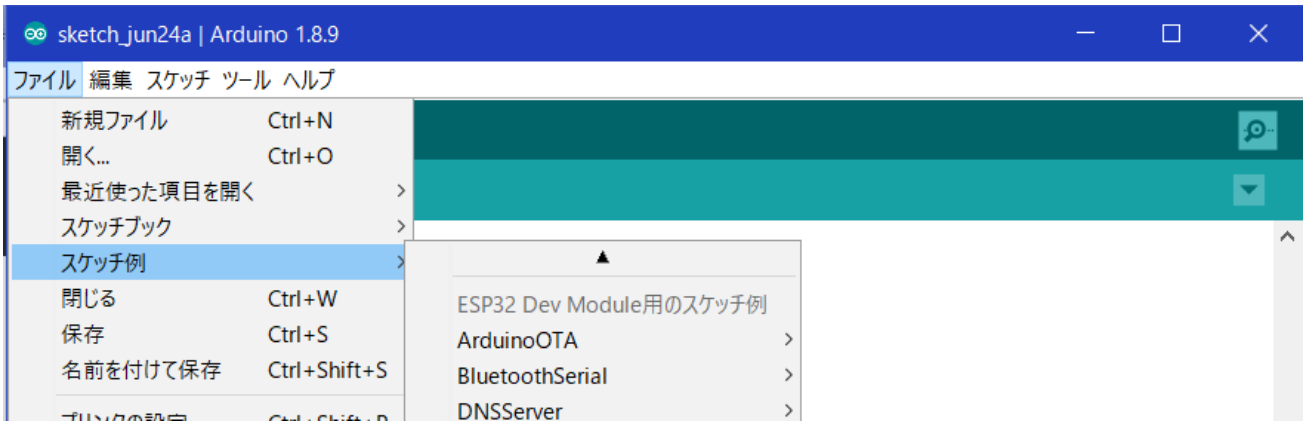
フォルダに移動する。



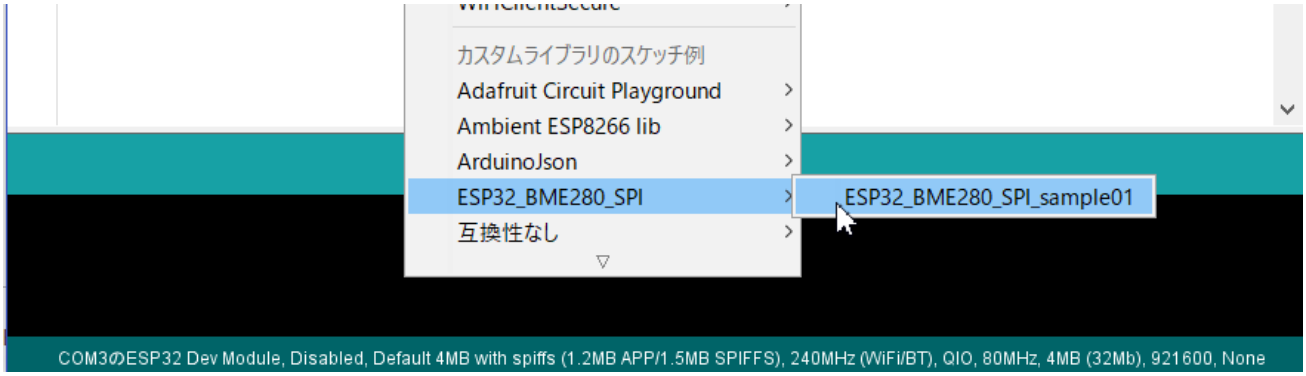
このとき、Arduino IDE を起動しているままならば、いったん Arduino IDE を終了させ、再度起動しなおす。

BME280 のライブラリが利用可能になっているかと BME280 センサーの動作確認のため、サンプルスケッチを開いてみる。

[ファイル]-[スケッチ例]-[ESP32_BME280_SPI]-[ESP32_BME280_SPI_sample01]を選ぶ。



＜中略＞



サンプルスケッチ「ESP32_BME280_SPI_sample01」を開いたら、そのままでは保存できないので、[ファイル]-［名前をつけて保存…]で、「**kad33_BME280**」というプロジェクト名に変えて保存する。

そのまま実行し、以下の様なシリアルモニタの表示となる。

シリアルモニタ

Temperature = 27 ℃　　←5 秒おきに温度・湿度・大気圧を表示
Humidity = 41 %
Pressure = 1012 hPa

Temperature = 27 ℃
Humidity = 41 %
Pressure = 1012 hPa

Temperature = 32 ℃　　← BME280 を指で押さえると、気温よりも温度が上がる
Humidity = 47 %　　← 湿度も少し上がる
Pressure = 1012 hPa　　←大気圧は指で押さえても変わらないようだ

Temperature = 27 ℃
Humidity = 42 %
Pressure = 1012 hPa

上記のように温度・湿度・大気圧が適切な値がでれば BME280 センサーボードの接続は OK。

【課題 34】 プロジェクト名「kad34_BME280_Ambient」

課題 33 の温度・湿度・大気圧を Ambient クラウドサービスにアップロードするプログラムを作る。その後 Ambient サイトで温度・湿度・大気圧をグラフで可視化するようにする。チャネル ID（ライトキーも）は前回の課題 31「kad31_cdsAmbient」を流用して、d1, d2, d3 にエントリする形にする。
d1, d2, d3 のデータソース(データ元)は、先ほどの課題「kad33_BME280」を温度・湿度・大気圧の順でエントリする。

【ポイント】Ambient は d1 から d8 まで 8 個のデータを同時にエントリできる。
d1～d3 は同時に send するようにせよ。そのほうが、通信量・省電力・リアルタイム性などが良い。

以下 loop()関数のみヒントを提示する。

```
52 void loop() {
53   delay(5000);//順序入れかえ最初に持ってくる
54
55   //ここからBME280 kad33のbme_get()を使わず温度・湿度・大気圧の変数をfloat型でセット
56   float temperature = bme280spi.[redacted];
57   float humidity = bme280spi.[redacted];
58   float pressure = bme280spi.[redacted];
59   // BME280の温度・湿度・大気圧の値をd1,d2,d3にセット
60   [redacted]
61
62
63   // Ambientクラウドに一斉送信
64   [redacted]
65
66   // BME280の温度・湿度・大気圧の値をシリアルモニタに表示
67   Serial.printf("温度 = [redacted];
68   Serial.printf("湿度 = [redacted];  //%は半角で表示
69   Serial.printf("大気圧 = [redacted];
70
71   delay(10000); //追加
72 }
```

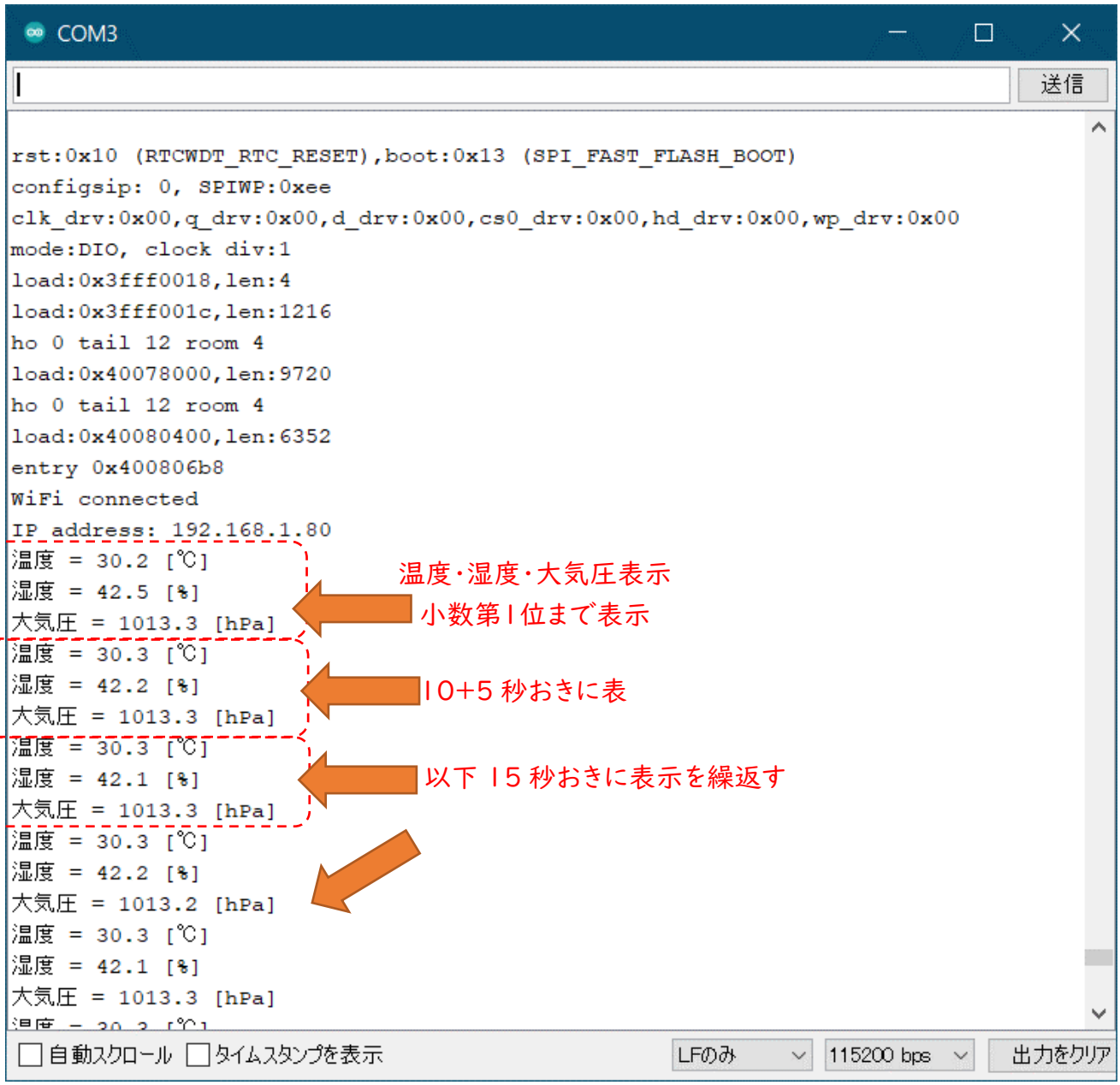
d1 温度 タイトル「温度」
- float 型の温度のための変数(temperature など)に bme280spi.Read_Temperature()を入れる
- Serial.printf()関数を使って小数第 1 位までの浮動小数点数を書式表示

d2 湿度（新しくグラフを追加）
- float 型の温度のための変数(humidity など)に bme280spi.Read_Humidity()を入れる
- Serial.printf()関数を使って小数第 1 位までの浮動小数点数を書式表示

d3 大気圧（新しくグラフを追加）
- float 型の大気圧のための変数(pressure など)に bme280spi.Read_Pressure()を入れる

・Serial.printf()関数を使って小数第１位までの浮動小数点数を書式表示

【課題 34 実行結果】シリアルモニタ



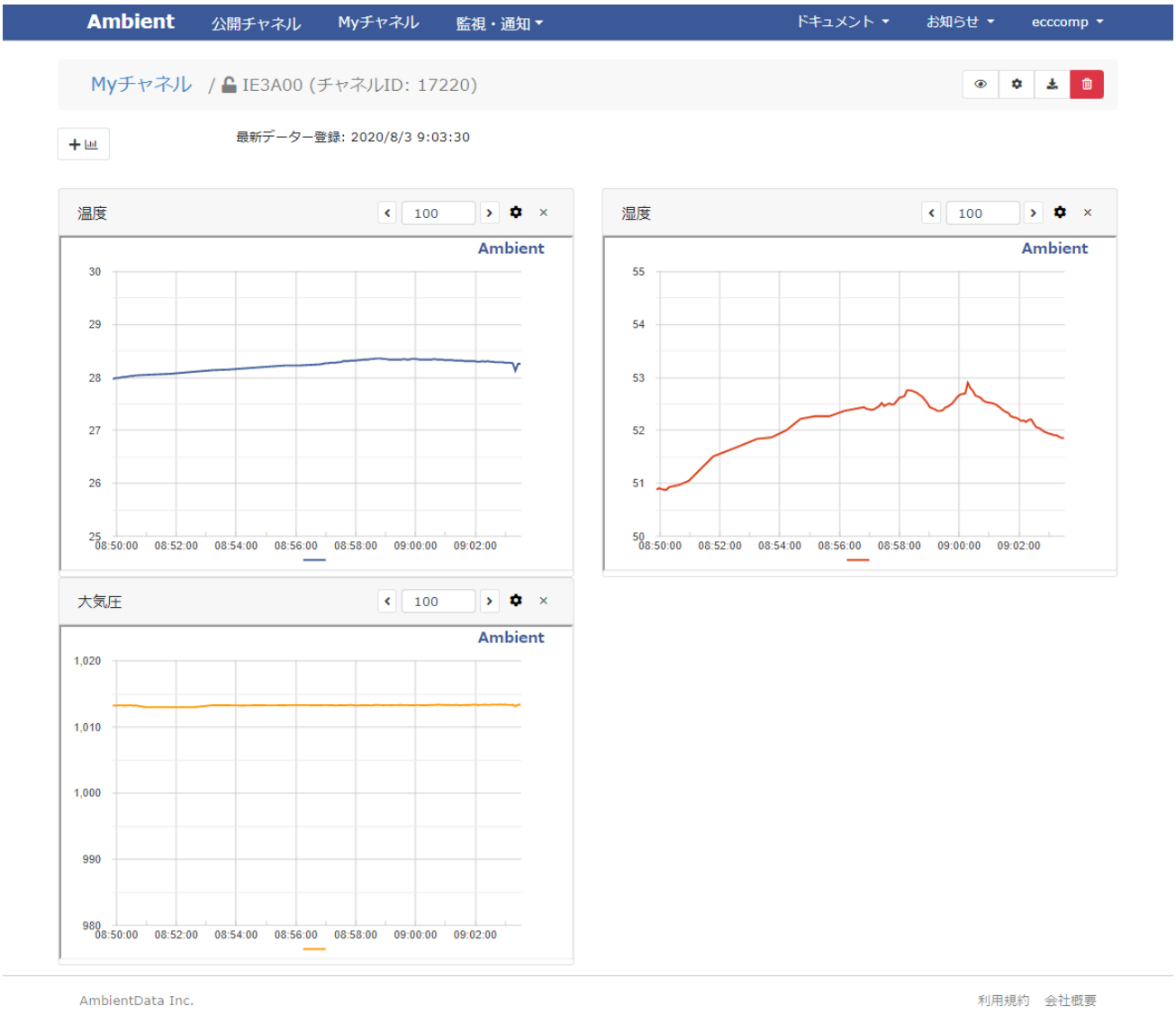
グラフを追加して 3 つのデータが同時に見られるようにしよう。

【課題 34 実行結果】Ambient グラフ

Ambient クラウド側のグラフは以下の 3 つになる。

温度・湿度・大気圧のグラフの Y 軸を調整して、変化が分かりやすいように工夫してみよう。

（例：温度だと 25～35℃にする等）



電源電圧・温度・湿度・大気圧のグラフのＹ軸を自分なりに工夫して調整できたら、動作チェックするので先生に申告する。

【課題 35】 プロジェクト名「kad35_BME280_DeepSleepAmbient」

課題 34 で温度・湿度・大気圧を Ambient クラウドサービスにアップロードすることができたが、センサー値をアップロードしていないときも WiFi に接続し無駄な電気を消費している。
電池などで駆動させることも考えて、DeepSleep を行うようにしてみる。
以下の記事にある「温湿度センサデバイスのプログラム」を元に、【課題 34】「kad34_BME280_Ambient」とソースを組み合わせてまずは 60 秒おきに DeepSleep するようにする。

ESP32 ではじめる IoT デバイス開発：一般記事 | gihyo.jp … 技術評論社
https://gihyo.jp/dev/column/01/iot/2019/esp32_iot?page=2 （←記事の 2 ページ目）

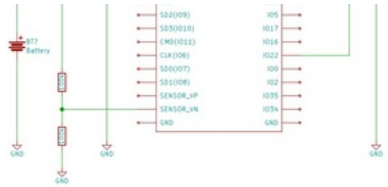


図3 温湿度センサデバイスの回路図

デバイスは単3乾電池3本で駆動します。電池電圧を抵抗で分圧してESP32のADコンバータ（SENSOR_VNピン）に加え，温度，湿度と合わせて測定するようにしました。

温湿度センサデバイスのソフトウェア

温湿度センサデバイスのプログラムは次のようになります。

```
#include
#include "Adafruit_Si7021.h"
#include

#define TIME_TO_SLEEP  60          // 測定周期（秒）

const char* ssid = "ssid";         // Wi-Fiルーターの SSID
const char* password = "password"; // Wi-Fiルーターのパスワード

WiFiClient client;
Ambient ambient;

unsigned int channelId = 100;      // Ambientのチャネル ID
const char* writeKey = "writeKey"; // ライトキー

Adafruit_Si7021 sensor = Adafruit_Si7021();

#define BATTERY 39                 // バッテリー電圧を測るピン

void setup(){
    unsigned long starttime = millis();
    Serial.begin(115200);
    while (!Serial) ;

    WiFi.begin(ssid, password);  // Wi-Fiネットワークに接続する
    while (WiFi.status() != WL_CONNECTED) {  // 接続したか調べる
        delay(500);
```

記事サイトをスクロールすると左の図のような「温湿度センサデバイスのプログラム」が見つかる。
これを元ソースとして【課題 35】「kad35_BME280_DeepSleepAmbient」を作ると良い。

このプログラム、なぜか 1 行目、3 行目が#iclude 命令のみでヘッダライブラリの指定が無い。追加修正しよう(2 行目も BME280 のヘッダに変更する必要がある)。

DeepSleep の期間を指定する TIME_TO_SLEEP マクロが 60 秒となっているが、確認するには長い為実行時には 15 秒程に短縮して良い。

Ambient のチャネル ID, ライトキー
WiFi の SSID、パスコードは書き換える。

この記事では、センサーは BME280 を使っておらず Si7021 というデジタル温湿度センサーを用いている。この部分を BME280 センサーのコードに差し替えるとうまく動作する。

【課題 35 実行結果】シリアルモニタ