第１２章 BME280 環境センサーと Ambient IoT プラットフォーム

・Bosche BME280 センサーを使う

※製作例

《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ました。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ます。、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ます。
ただしアナログ入力は、センサーごとに入力ピンをとります。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ります。
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います。
また、電池駆動を考えて、電源電圧をアナログ入力で測定できるようにしてみましょう。

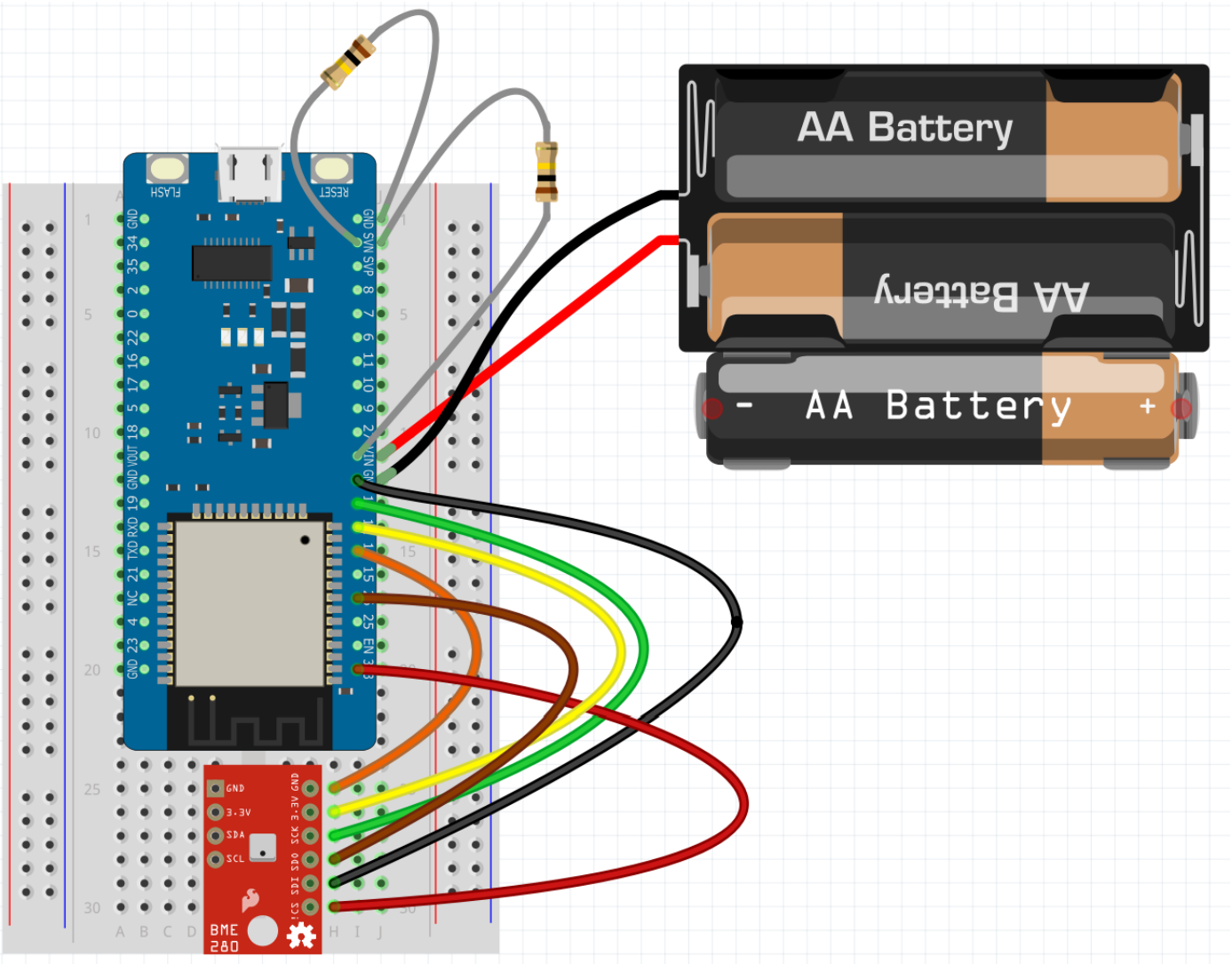
第 12 章 演習
【第 12 章の目標】
シリアル通信（SPI）アナログ値を取得する方法を知る。WiFi ステーション（クライアント）接続で、インターネット・サイトに接続し、シンプルな国産 IoT プラットフォーム「Ambient」を利用して、データのエントリと可視化を行う。

【1,ESP32 と電子工作部品との接続】
1.1.必要な部品

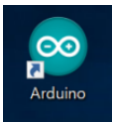
パーツ名	必要個数
Bosch BME280 センサー	1 個
ジャンパーワイヤー	橙色 2 個、黄色 1 個
ジャンパーコード	6 本（赤色、茶色、橙色、緑色、黄色、黒色 各 1 本ずつ） **※上記の色が揃わなければ別の色でも良い**
100kΩ抵抗(150kΩ等でも可)	2 個
電池ボックス	1 個（単 3 電池 3 個をセットする）

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。
BME280 センサーをブレッドボード（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。

① ジャンパーワイヤー（赤色）をブレッドボード 1 の i30 へ、もう片方を ESP32 の 3V3 ピンへ。
② 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。
③ ジャンパーワイヤ（茶色）をブレッドボード 1 の i28 へ、もう片方を ESP32 の IO26 ピンへ。
④ ジャンパーワイヤ（緑色）をブレッドボード 1 の i27 へ、もう片方を ESP32 の IO13 ピンへ。
⑤ ジャンパーワイヤ（黄色）をブレッドボード 1 の i26 へ、もう片方を ESP32 の IO12 ピンへ。
⑥ ジャンパーワイヤ（橙色）をブレッドボード 1 の i25 へ、もう片方を ESP32 の IO14 ピンへ。
⑦ 100kΩ抵抗(または 150kΩ抵抗)をブレッドボード 1 の j1 へ、もう片方を ESP32 の SEN_n ピン（右上から 2 番目）へ。
⑧ もうひとつの 100kΩ抵抗(または 150kΩ抵抗)をブレッドボード 1 の j2 へ、もう片方を ESP32 の VIN ピンへ。
⑨ 電池ボックスのワイヤ（赤色）をブレッドボード 1 の j11 へ
⑩ 電池ボックスのワイヤ（黒色）をブレッドボード 1 の j12 へ

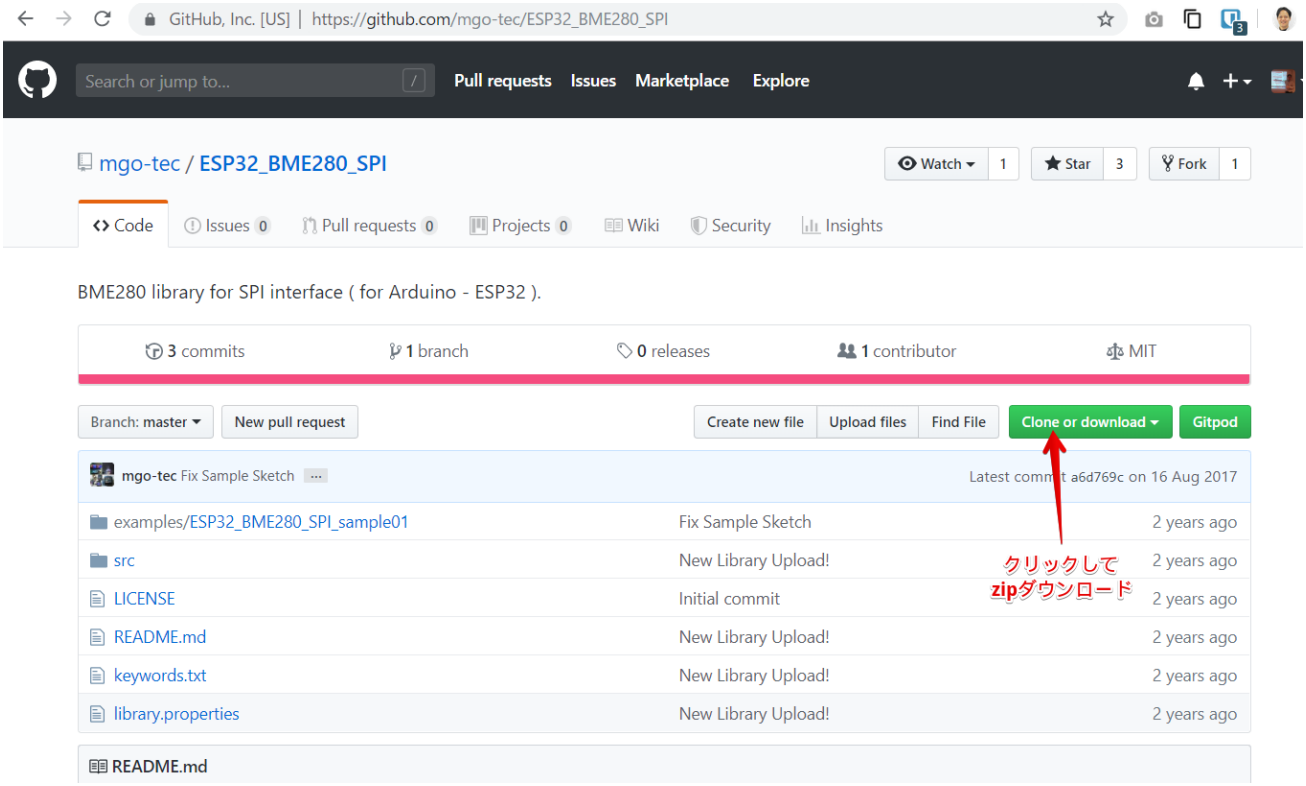


【2. Arduino スケッチのサンプルプログラムを実行】

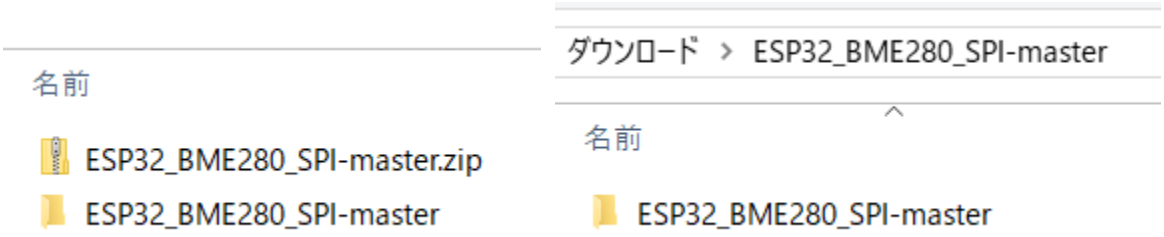


デスクトップのアイコンをダブルクリックして
Arduino IDE を起動します。

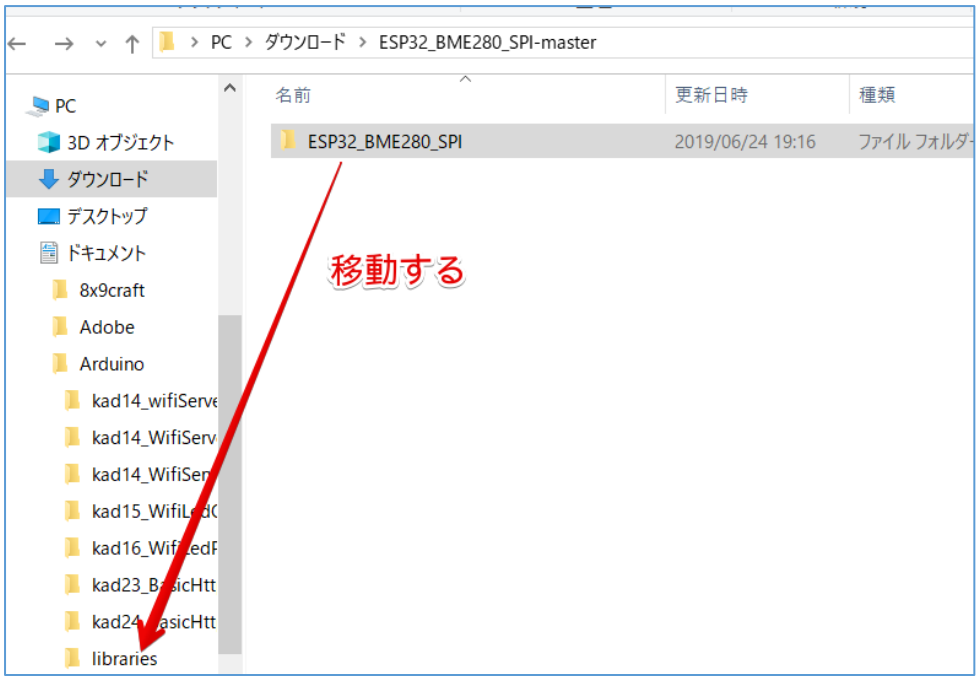
【課題 33】プロジェクト名を「kad33_BME280」として提出します。
のマイコン側プログラミングを行う前に、BME280 SPI 接続用のライブラリをダウンロードして libraries フォルダに格納しておく必要があります。
以下の github サイトでライブラリをダウンロードします。
https://github.com/mgo-tec/ESP32_BME280_SPI



ダウンロードした ZIP ファイルを解凍して、フォルダを開きます。



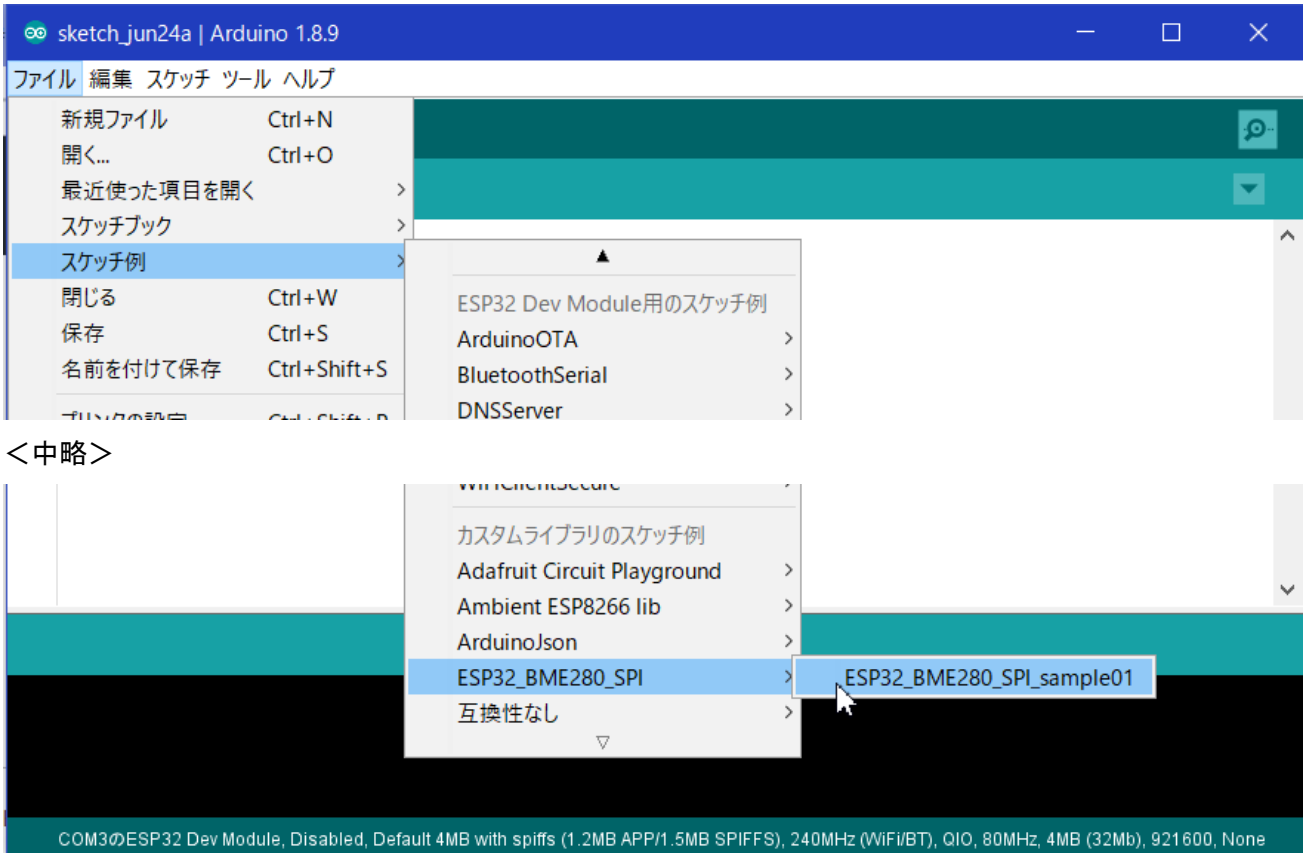
「ESP32_BME280_SPI-master」フォルダ内の「ESP32_BME280_SPI-master」フォルダを「ESP32_BME280_SPI」に書き換えて、ドキュメントフォルダ内の Arduino フォルダにある「Libraries」フォルダに移動します。



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。



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。

次に、SEN_n ピンのアナログ入力から電源電圧を計測するプログラムを追加する。

以下の手順で書き加え＆書き換えを行う。

手順① 3 行目あたりに以下の#define 文を追加

```
#include "ESP32_BME280_SPI.h"

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

const uint8_t SCLK_bme280 = 14;
const uint8_t MOSI_bme280 =13; //Master Output Slave Input ESP32=Master
const uint8_t MISO_bme280 =12; //Master Input Slave Output
const uint8_t CS_bme280 = 26; //CS pin

ESP32_BME280_SPI bme280spi(SCLK_bme280, MOSI_bme280, MISO_bme280, CS_bm
```

手順② loop()関数内を以下のように修正＆追加

```
27 void loop(){
28   delay(5000);//順序入れかえ
29   bme_get();//順序入れかえ
30   //電源電圧測定
31   float vbat = (analogRead(BATTERY) / 4095.0 * 3.3 + 0.1132) * 2.0;
32   //電源電圧表示
33   Serial.printf("vbat = %.2f\n", vbat);
34 }
```

31 行のアナログ入力→入力電圧の変換式は、特別な補正計算を加えている。

```
float vbat = (analogRead(BATTERY) / 4095.0 * 3.3 + 0.1132) * 2.0;
```
←コピペしてください

これは **ESP32 の AD コンバータが入力電圧に対してアナログ入力の値がリニアに取れない誤差を含む ESP32 自体の不具合**があって、その不具合を補正したもの。通常は無視してもよいが、今回は電池によるバッテリー電圧を計測するため補正計算を含めた変換式を使う。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
------------------------
vbat = 4.54[V] ←温度・湿度・大気圧につづけて電源電圧を表示 4.5V 前後（新しい電池の場合）
------------------------
Temperature = 27 ℃
Humidity = 41 %
Pressure = 1012 hPa
------------------------
vbat = 4.54[V]
------------------------
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
------------------------
vbat = 4.54[V]
------------------------
Temperature = 27 ℃
```

```
Humidity = 42 %
```

上記のように温度・湿度・大気圧・電源電圧が適切な値がでればデータ取得の準備は 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・電源電圧を 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 サイトで電源電圧・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）は前々回の課題 30「kad30_cdsAmbient」を流用して、d2, d3 d4 にエントリする形にする。

d1, d2, d3, d4 のデータソース(データ元)は、先ほどの課題「kad33_BME280」を

d1 へエントリしている光センサーの値を電源電圧（変数 vbat）に変える。

【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～d4 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。

```
53 void loop() {
54   delay(5000);//順序入れかえ
55   //kad33から電源電圧のアナログ入力→電圧 変換式
56   float vbat = (analogRead(BATTERY) / 4095.0 * 3.3 + 0.1132) * 2.0;
57   // 電源電圧の値をd1にセット
58   ambient.set(1, vbat); // データーがint型かfloat型であれば直接セットすることができます。
59
60   //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
61   float temperature = bme280spi.Read_Temperature();
62   float humidity = bme280spi.Read_Humidity();
63   float pressure = bme280spi.Read_Pressure();
64   // BME280の温度・湿度・大気圧の値をd2,d3,d4にセット
65   ambient.set(2, temperature);
66   ambient.set(3, humidity);
67   ambient.set(4, pressure);
68   // Ambientクラウドに一斉送信
69   ambient.send();
70
71   //電源電圧表示
72   Serial.printf("vbat = %.2f[V]\n", vbat);
73   // BME280の温度・湿度・大気圧の値をシリアルモニタに表示
74   Serial.printf("温度 = %.1f [℃]\n", temperature);
75   Serial.printf("湿度 = %.1f [%%]\n", humidity);  //%は半角で表示
76   Serial.printf("大気圧 = %.1f [hPa]\n", pressure);
77   delay(10000);
```

d1 電源電圧（kad33 と同じ）タイトル「バッテリー電圧」

・kad33 からアナログ入力→入力電圧の変換式を使って float 型の電源電圧の変数に代入する

・Serial.printf()関数を使って小数第 2 位までの浮動小数点数を書式表示

d2 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

d3 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

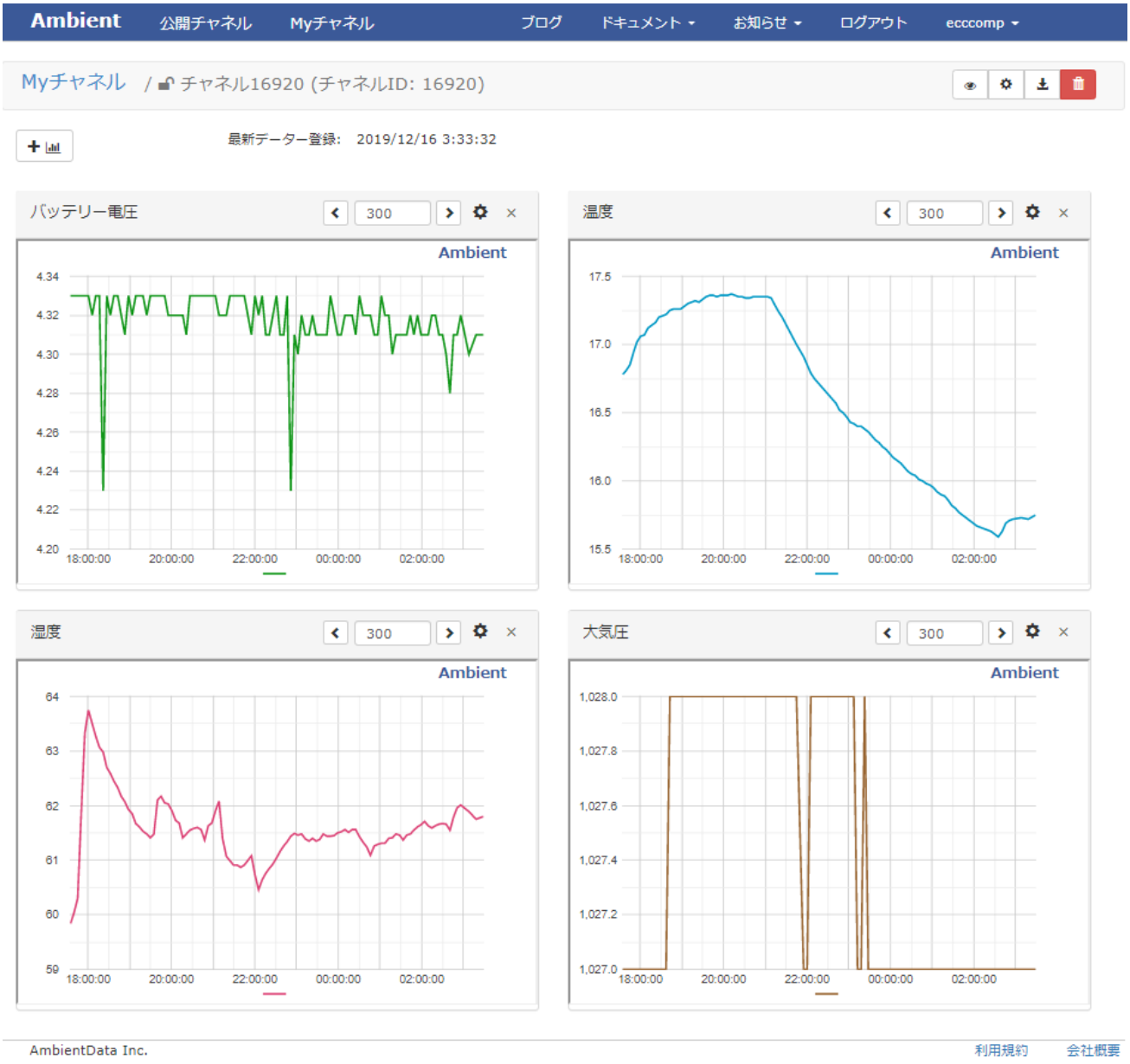
d4 大気圧（新しくグラフを追加）
・float 型の大気圧のための変数(pressure など)に bme280spi.Read_Pressure()を入れる
・Serial.printf()関数を使って小数第 1 位までの浮動小数点数を書式表示

【課題 34 実行結果】シリアルモニタ

グラフを追加して 4 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 4 つになる。
電源電圧（バッテリー電圧）・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 10～30℃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、先生に申告してください。
動作チェックします。

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

課題 34 で電源電圧・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。
電池で駆動するために、DeepSleep を行うようにしてみよう。
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 するようにしよう。

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）

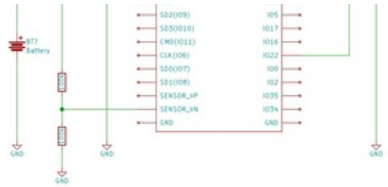


図3 温湿度センサデバイスの回路図

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、これを各自違うものに変える。先生に指示を受けてください。

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。

動作したら、USB シリアルのケーブルを抜いて、電池で駆動しても Ambient にグラフ表示されるか試してみよう。