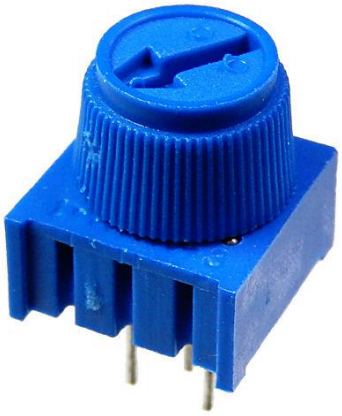
ボリューム（可変抵抗）を使う



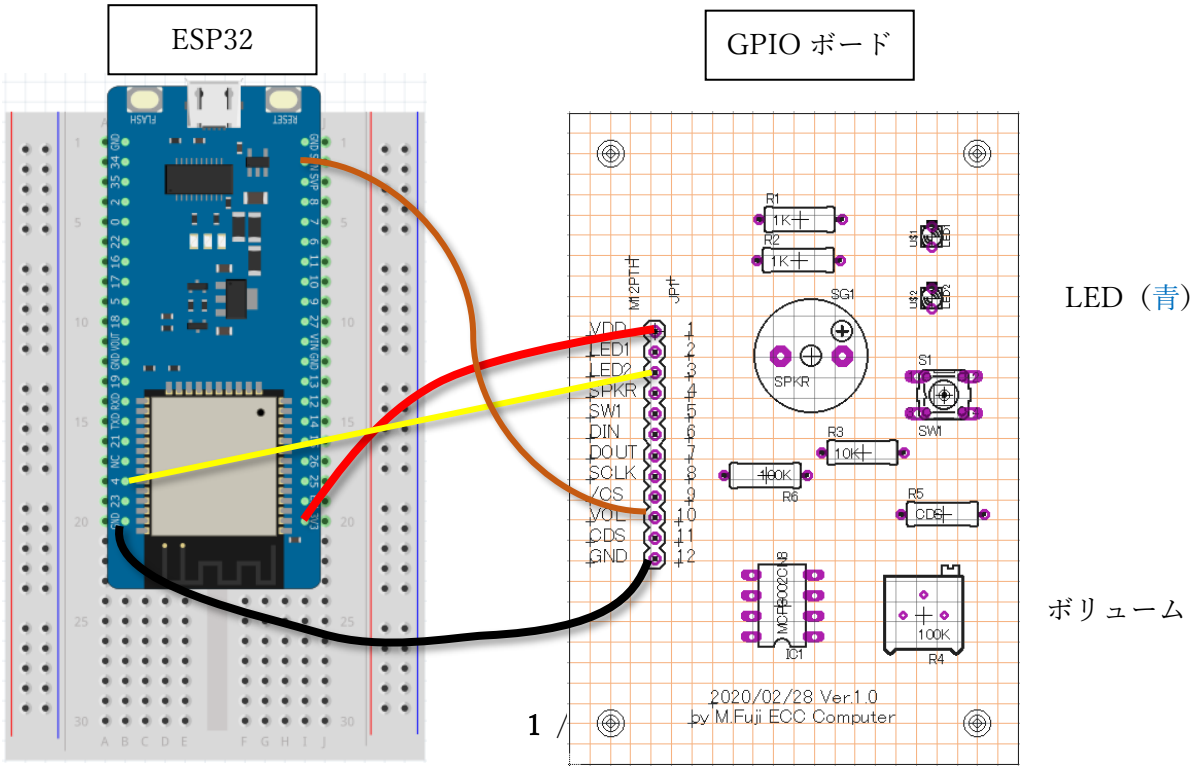
《何ができる？》今まで、光センサ（CdS センサ）を用いてアナログ入力から明るさの値を取得してきた。アナログ入力を用いれば様々なアナログセンサーの値を取得し、インターネットに向けてデータを転送しグラフの形で可視化することができる。
今回は、自分でつまみを変化させることのできるボリューム（可変抵抗）を用いて 0～4095 のアナログ入力値から電圧を算出し、データを蓄積する IoT のしくみを構築したい。

第１１回 演習

【第１１回の目標】

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。

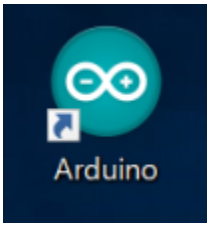
【1,ESP32 と電子工作部品との接続】



上記実体配線図を元に、４本のジャンパーコードの配線を行う。接続するピンは以下の一覧の通り。

① ジャンパーコード（白）ESP32 の IO39 － GPIO ボード LED1(10 番ピン)
　A3 ピン（IO39 ピン）＝「SEN_n」と表記のあるピン（図で言うと向かって右上から 2 番目）。
② ジャンパーコード（黄）ESP32 の IO4 － GPIO ボード LED2（3 番ピン）
③ ジャンパーコード（黒）ESP32 の GND － GPIO ボード GND(12 番ピン)
④ ジャンパーコード（赤）ESP32 の 3V3 － GPIO ボード VDD（1 番ピン）

【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 27】プロジェクト名「kad27_volume」
《サンプル①》を打ち込んで、動作確認をする。

```
1  const int volumePin = A3;//ボリューム（可変抵抗）A3ピン
2  const int bluePin = 4;  //青色LED IO4ピン
3  void setup() {
4    Serial.begin(115200);
5    pinMode(bluePin, OUTPUT);
6  }
7
8  void loop() {
9    int ana = analogRead(volumePin);  //ボリュームのアナログ入力計測
10   ledFlash(bluePin, 200);//計測を確認するため青色LED点滅
11
12   //アナログ入力の0-4095の値を0-3300[mV]にmap(比例配分)する
13   double v  = map(ana, 0,4095, 0,3300) / 1E3; //1000で割ってmVからVへ
14   //double v = 3.3 * ana / 4095; //この比例配分の式と同じ
15   Serial.printf("アナログ入力:%d, 単純変換値:%f[V]\n", ana, v);
16
17   delay(5000);//5秒おきに実行
18 }
```

10 行目に、ledFlash()というユーザー定義の関数が呼び出されているが、これは 22 行目以降に関数定義を書いておく。
20 行目から 24 行目まで Doc コメントという形式で記載されているが、これも打ち込むこと。打ち込む際に、/**[Enter]と打って、自動的にコメントの閉じまで生成されるのを確認しておく。

```
/**[Enter]
```

⇒

```
/**
 * ←カーソルがここに来る
 */
```

Doc コメントは元々JavaDoc から来ている。Java の API ドキュメントを自動生成するためのもの。

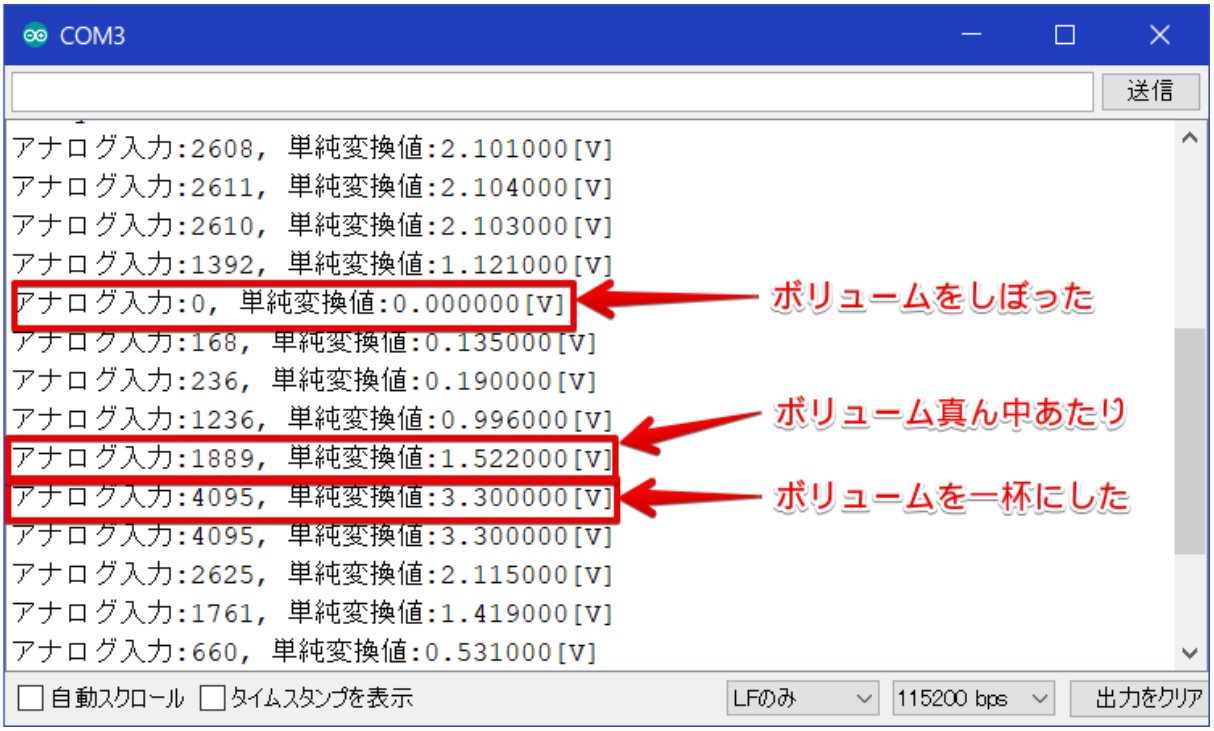
参考 URL https://www.javadrive.jp/javadoc/comment/index1.html

```
20 /**
21  * LEDを指定したミリ秒分点滅させる
22  * @param ledPin LEDの接続しているGPIOピン番号
23  * @param delayTime LEDを点灯させておく時間
24  */
25 void ledFlash(int ledPin, int delayTime){
26   digitalWrite(ledPin, HIGH);
27   delay(delayTime);
28   digitalWrite(ledPin, LOW);
29 }
```

※21～23 行のコメントもちゃんと打ち込むこと。後々API ドキュメントジェネレータをかけて自動生成のデータに使う予定。

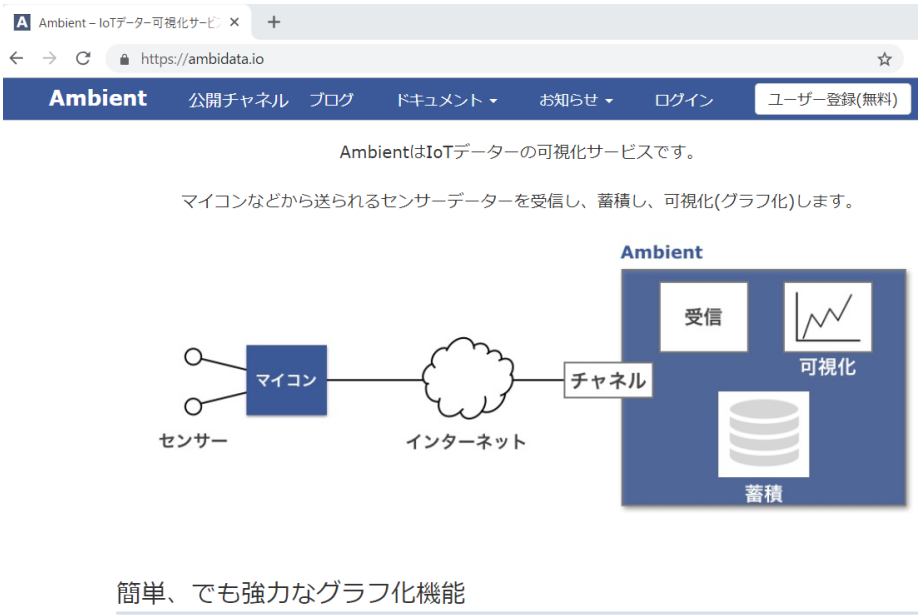
【サンプル①の実行結果】

シリアルモニタに以下の様な表示が出る。



【課題 28】 プロジェクト名「kad28_Ambient_ESP32」

ボリューム（可変抵抗）のアナログ値が取得できるようになったので、このデータをクラウドにエントリするプログラムを作る。

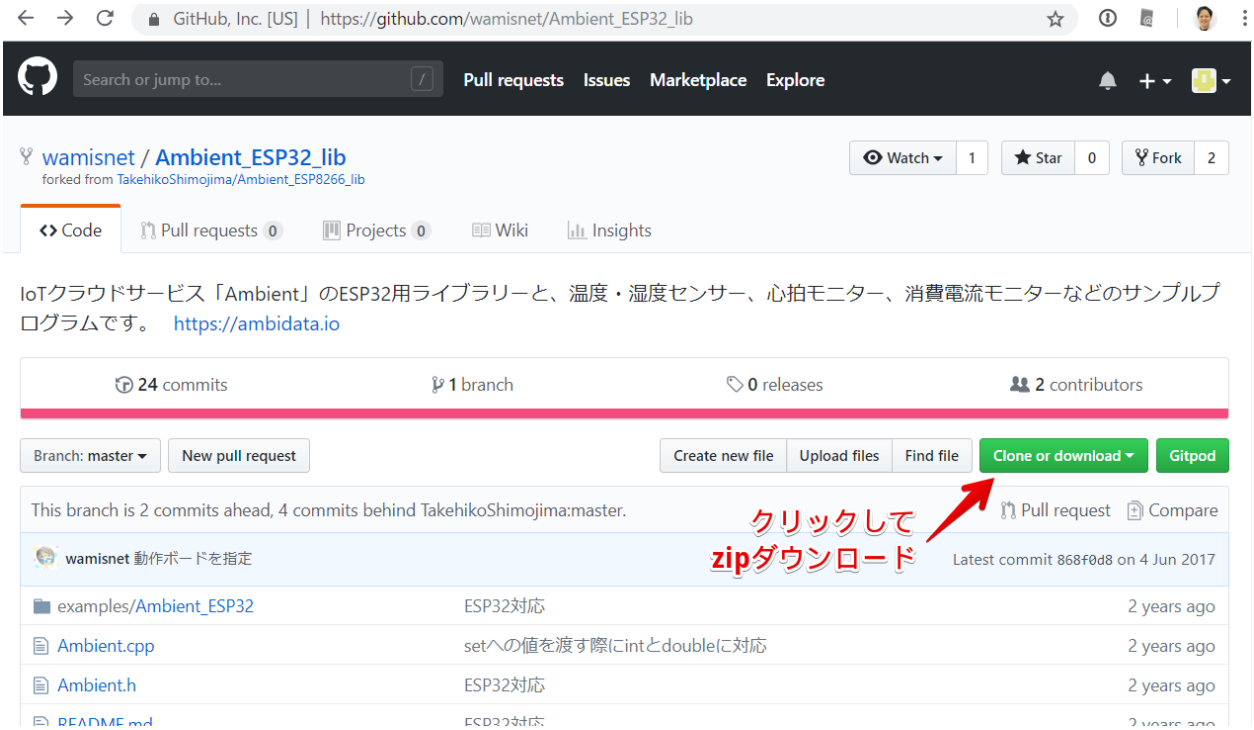


https://ambidata.io/ ユーザー登録（無料）を行う。適当なメールアドレスを使って入手する。

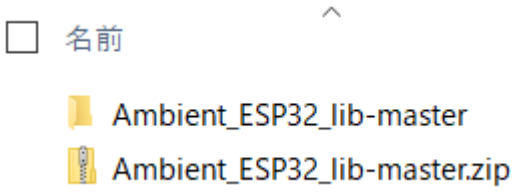
開発のおおまかな流れは、以下のチュートリアルサイトに記載されている。
https://ambidata.io/docs/gettingstarted/ クリックしてサイトを見る。

1. ユーザー登録(無料)
2. チャネル生成
3. マイコン側プログラミング
4. データ送信
5. 可視化(グラフ化)

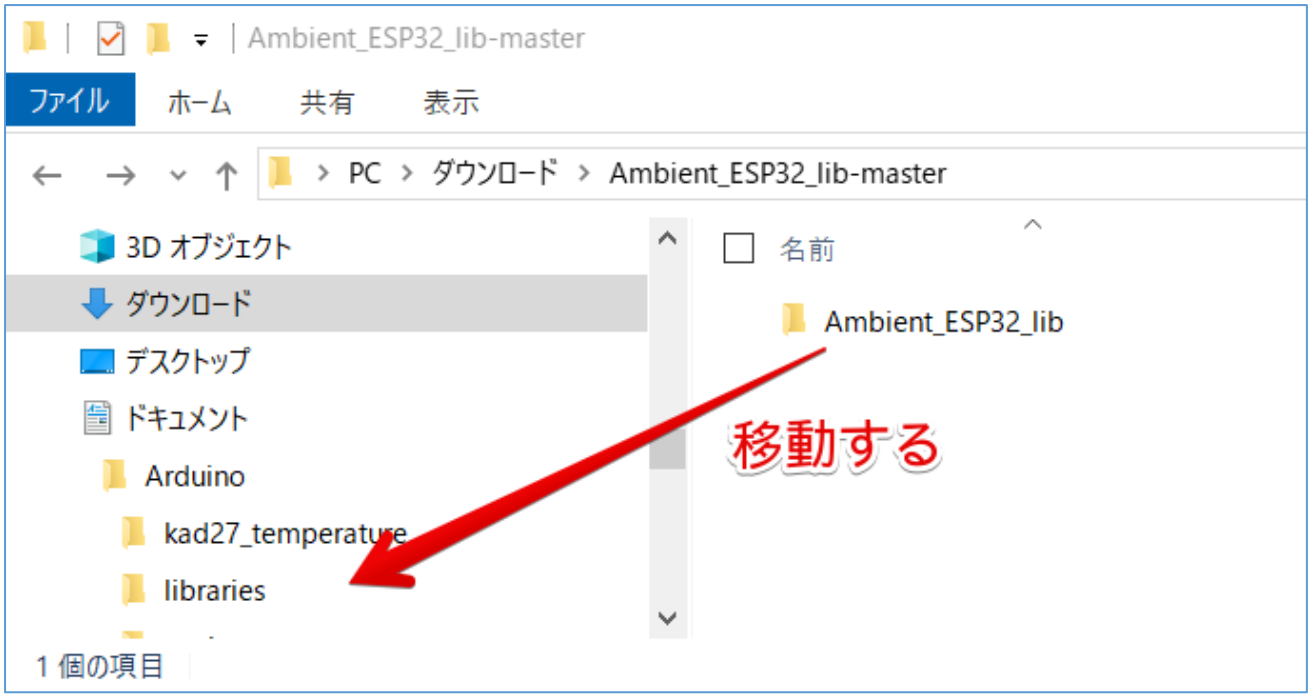
２のチャネル生成まで行うこと。
３のマイコン側プログラミングを行う前に、Ambient 用のライブラリをダウンロードして Library フォルダに格納しておく必要がある。
以下の github サイトでライブラリをダウンロードする。
https://github.com/wamisnet/Ambient_ESP32_lib



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。



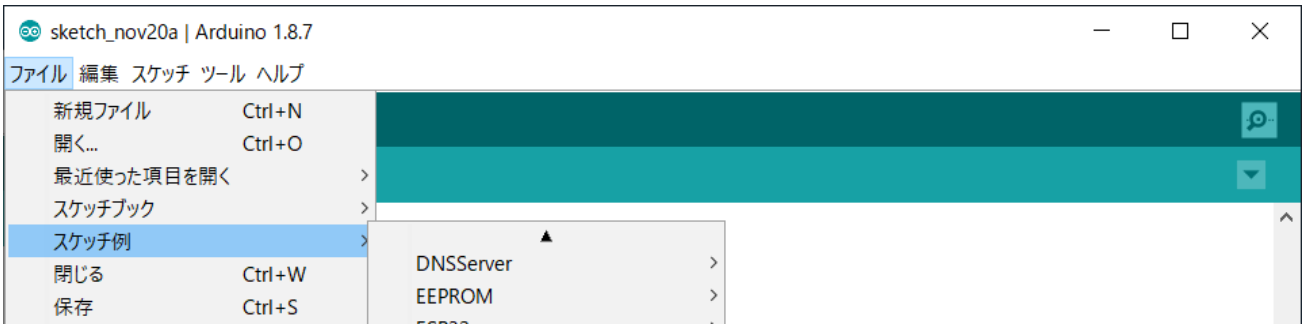
「Ambient_ESP32_lib-master」フォルダ内の
「Ambient_ESP32_lib-master」フォルダを
「Ambient_ESP32_lib」に書き換えて、ドキュメントフォルダ内の Arduino フォルダにある「libraries」フォルダに移動する。



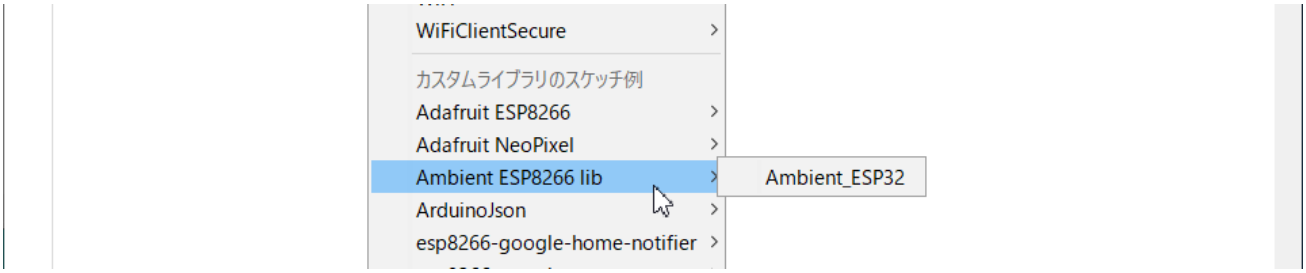
このとき、Arduino IDE を起動しているままならば、いったん Arduino IDE を終了させ、再度起動しなおす。

Ambient のライブラリが利用可能になっているかと Ambient の接続確認のため、サンプルスケッチを開いてみる。

[ファイル]-[スケッチ例]-[Ambient ESP8266 lib]-[Ambient_ESP32]を選ぶ。



＜中略＞



サンプルスケッチ「Ambient_ESP32」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、「kad28_Ambient_ESP32」というプロジェクト名に変えて保存する。

以下の表示で、SSID とパスコードを実習用に書き換える。

kad28_Ambient_ESP32

```
1 #include <WiFi.h>
2 #include "Ambient.h"
3
4 const char* ssid = "・・・ssid・・・";
5 const char* password = "・・・パスワード・・・";
6
7 unsigned int channelId = 100;
8 const char* writeKey = "ライトキー";
9
10 WiFiClient client;
11 Ambient ambient;
12
13 void setup(){
```

学内マイコン/IoT 演習用 WiFi の接続設定情報となるコード

// 接続先の SSID とパスワード 学内 CampusIOT
SSID 文字列： "CampusIoT-WiFi"
ハッシュ化されたパスフレーズ文字列：
"0b8b413f2c0fa6aa90e085e9431abbf1fa1b2bd2db0ecf4ae9ce4b2e87da770c"

7 行目のチャネル ID、8 行目のライトキーは、作成したチャネルのチャネル ID 値とライトキー文字列をコピー＆ペーストする。

課題 28 ソース書き換え

【宣言部】

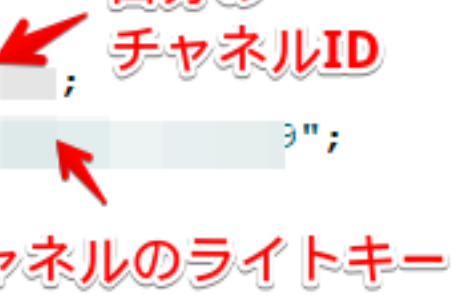
kad28_Ambient_ESP32

```
#include <WiFi.h>
#include "Ambient.h"

// 接続先のSSIDとパスワード 学内CampusIOT
const char* ssid = "CampusIoT-WiFi";
const char* password = "0b8b413f                a90e085e";

unsigned int channelId =      ;
const char* writeKey = "                    ";

WiFiClient client;
Ambient ambient;
```



【setup()メソッド部】

```
void setup(){
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi connected");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());

    ambient.begin(channelId, writeKey, &client);
}
```

【loop()メソッド部】

```
28 void loop(){
29   int rand;
30   rand = random(1000);
31
32     ambient.set(1, rand); // データーがint型かfloat型であれば、
33                           //直接セットすることができます。
34     ambient.send();
35     Serial.println(rand);
36     delay(5000);
37 }
```

追加

以上の書き換え・追加を行って、マイコンに書き込みを行う。

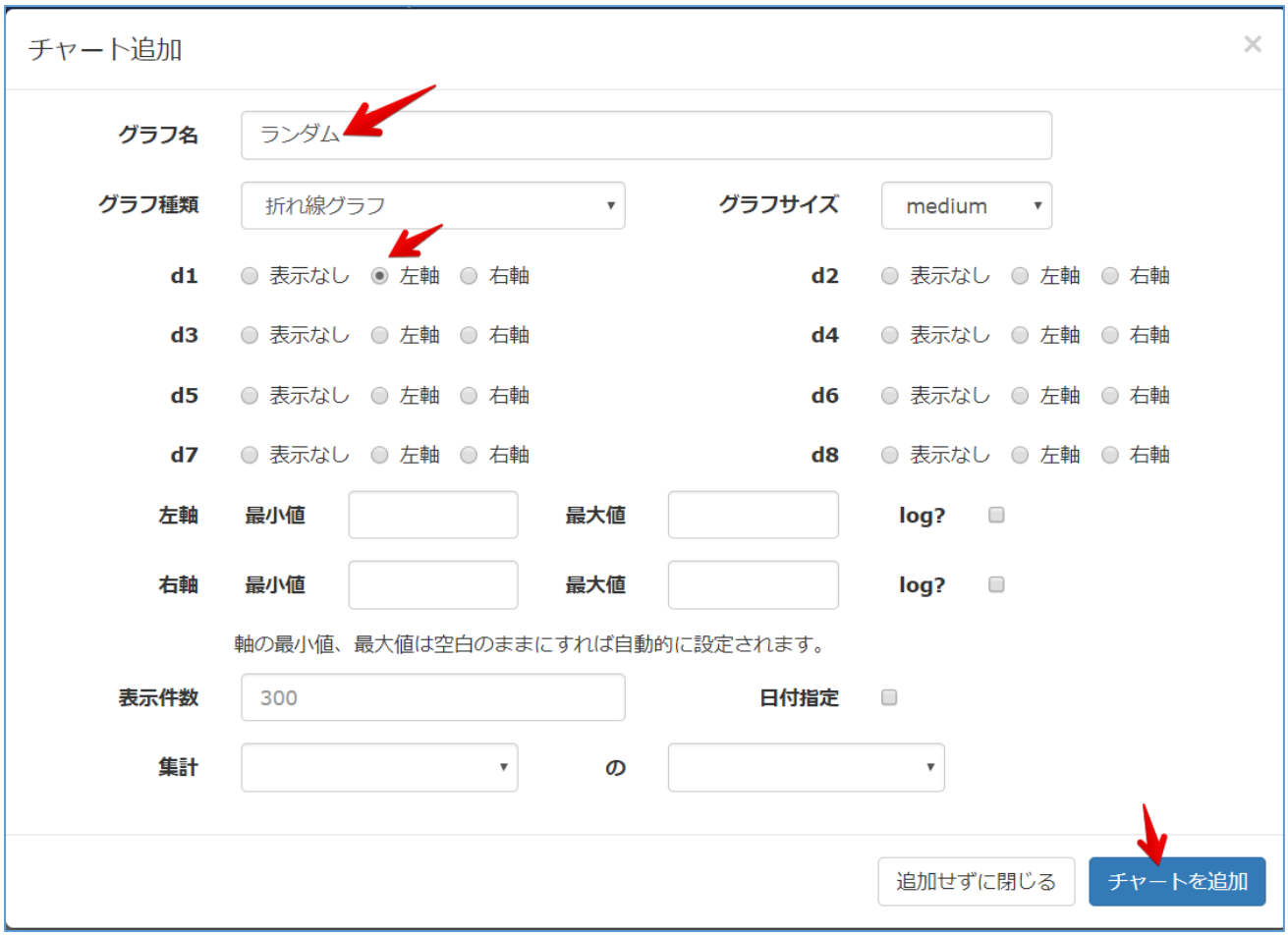
実行できたら、以下の様なシリアルモニタの表示となる。

シリアルモニタ
.....WiFi connected IP address: 10.101.0.x 195　　←5 秒おきに 0 から 999 までの乱数が表示される。 530 714 16 291 680 397 7 311

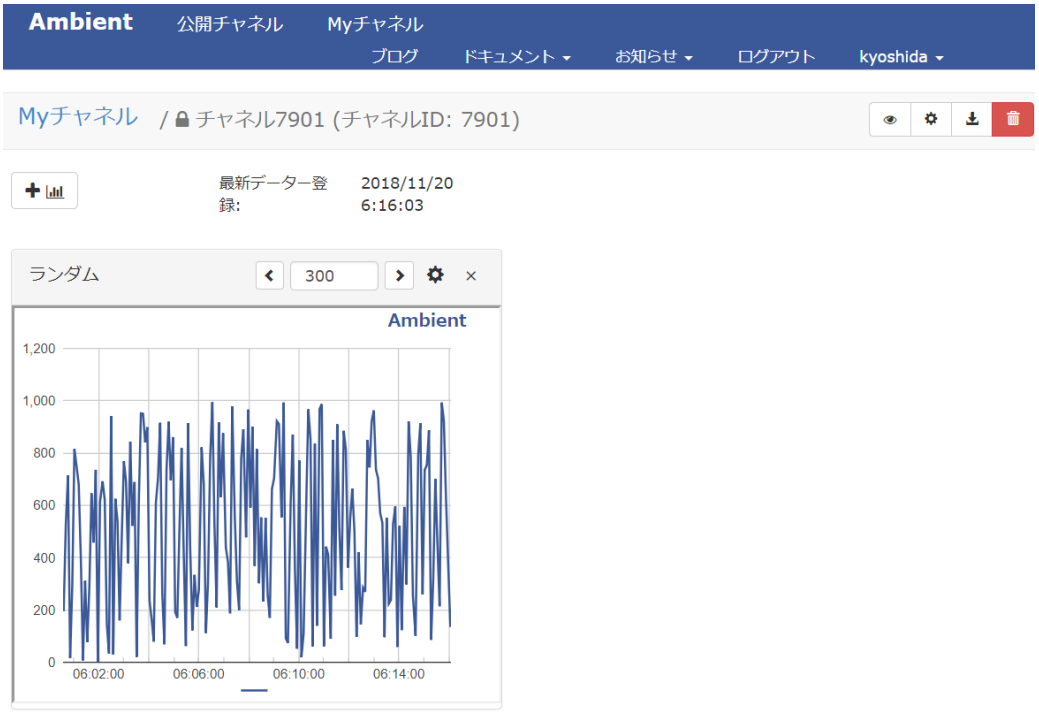
ブラウザで、Ambiet のチャネルの表示を行ってみる。最新データに日時があればデータが届いている。

[+]ボタンをクリックして、グラフの作成を行う。

グラフ名を「ランダム」などとし、d1 に〇左軸にチェックを入れ、[チャートを追加]ボタンをクリックしグラフを作成する。



以下の様なグラフが出たら課題 28 は完成。



【課題 29】 プロジェクト名「kad29_volumeAmbient」

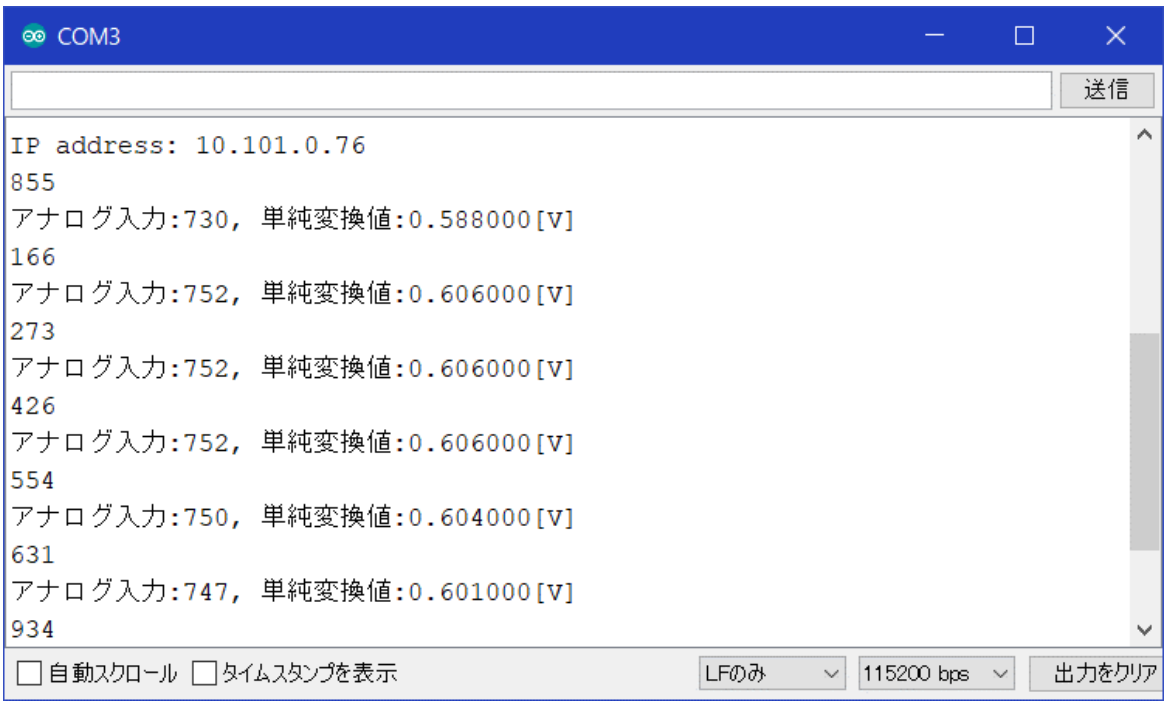
課題 28 のソースコードをそのままコピーして、ボリューム（可変抵抗）の電圧値を Ambient にアップロードせよ。チャネル ID（ライトキーも）はそのままで、d2 にエントリする形にする。

d1 へエントリしているランダム値はそのままにしておく。青色 LED の点滅処理も入れておくこと。

【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 と d2 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。

【課題 29 実行結果】



2 つ目の「電圧値」グラフを追加してランダムと電圧値が同時に見られるようにする。