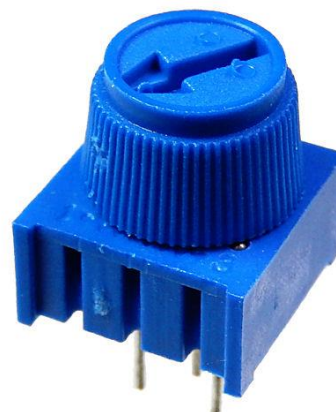


ボリューム（可変抵抗）を使う

《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。

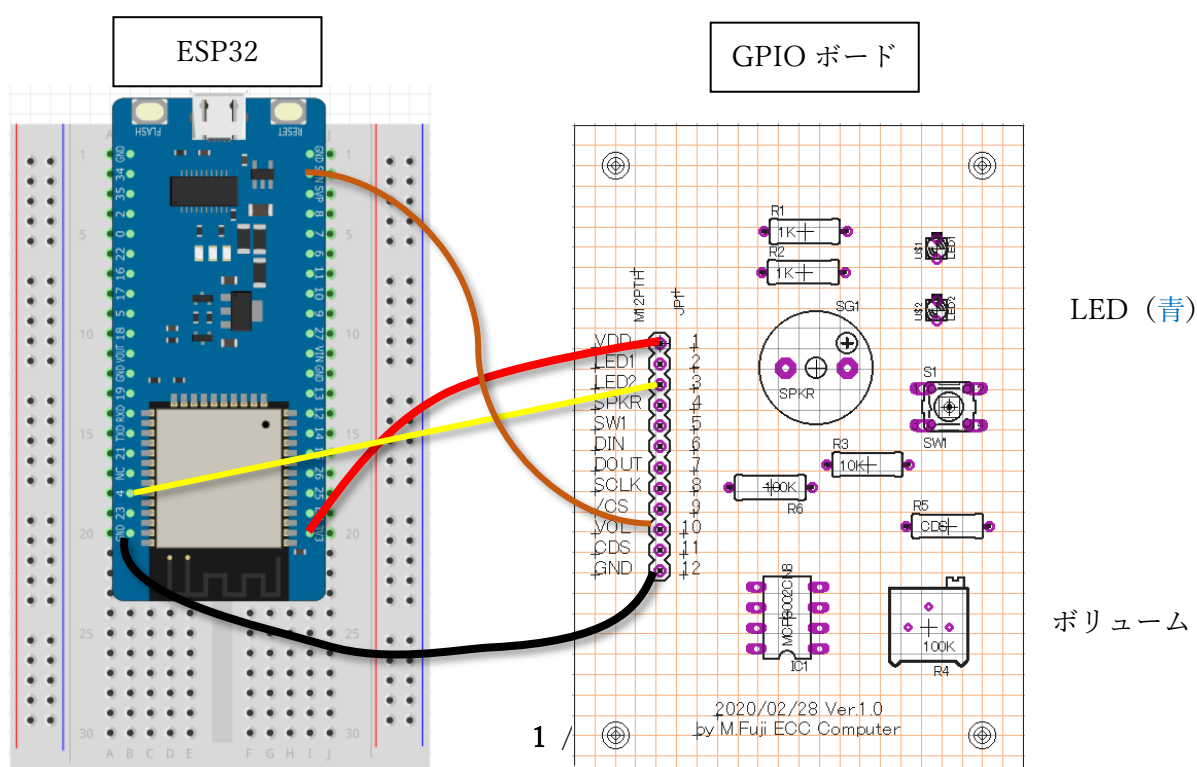


第 11 回 演習

【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、データのエンタリと可視化を行う。

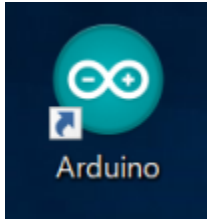
【1, ESP32 と電子工作部品との接続】



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 I039 – GPIOボード LED1(10番ピン)
A3ピン（I039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 I04 – 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。

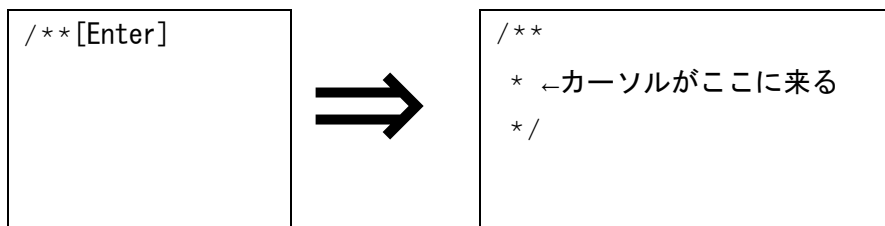
【課題 27】プロジェクト名「kad27_volume」

《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。



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。

COM3

送信

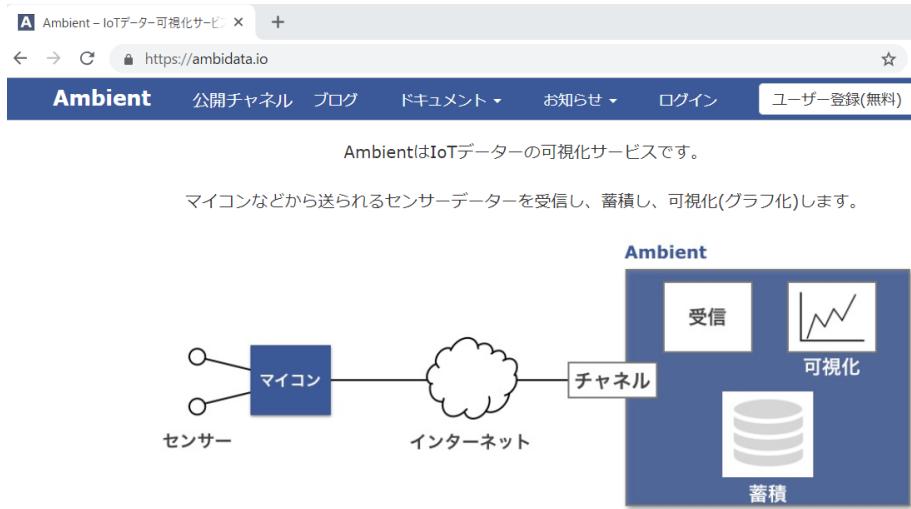
アナログ入力:2608, 単純変換値:2.101000 [V]
 アナログ入力:2611, 単純変換値:2.104000 [V]
 アナログ入力:2610, 単純変換値:2.103000 [V]
 アナログ入力:1392, 単純変換値:1.121000 [V]
 アナログ入力:0, 単純変換値:0.000000 [V] ← ボリュームをしぼった
 アナログ入力:168, 単純変換値:0.135000 [V]
 アナログ入力:236, 単純変換値:0.190000 [V]
 アナログ入力:1236, 単純変換値:0.996000 [V]
 アナログ入力:1889, 単純変換値:1.522000 [V] ← ボリューム真ん中あたり
 アナログ入力:4095, 単純変換値:3.300000 [V] ← ボリュームを一杯にした
 アナログ入力:4095, 単純変換値:3.300000 [V]
 アナログ入力:2625, 単純変換値:2.115000 [V]
 アナログ入力:1761, 単純変換値:1.419000 [V]
 アナログ入力:660, 単純変換値:0.531000 [V]

☐ 自動スクロール ☐ タイムスタンプを表示

LFのみ 115200 bps 出力をクリア

【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。



簡単、でも強力なグラフ化機能

<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. チャンネル生成
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のチャン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

← → ↻ GitHub, Inc. [US] | https://github.com/wamisnet/Ambient_ESP32_lib ☆ ⓘ 📄 👤

🐱 Search or jump to... / Pull requests Issues Marketplace Explore 🔔 + 👤

🔑 wamisnet / Ambient_ESP32_lib
forked from TakehikoShimajima/Ambient_ESP8266_lib

👁 Watch 1 ★ Star 0 🍴 Fork 2

🔗 Code 📄 Pull requests 0 📁 Projects 0 📖 Wiki 📊 Insights

IoTクラウドサービス「Ambient」のESP32用ライブラリーと、温度・湿度センサー、心拍モニター、消費電流モニターなどのサンプルプログラムです。 <https://ambidata.io>

📄 24 commits 🌿 1 branch 📦 0 releases 👤 2 contributors

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾ Gitpod

This branch is 2 commits ahead, 4 commits behind TakehikoShimajima:master. 📄 Pull request 📄 Compare

👤 wamisnet 動作ボードを指定 Latest commit 868f0d8 on 4 Jun 2017

📁 examples/Ambient_ESP32	ESP32対応	2 years ago
📄 Ambient.cpp	setへの値を渡す際にintとdoubleに対応	2 years ago
📄 Ambient.h	ESP32対応	2 years ago
📄 README.md	ESP32対応	2 years ago

クリックして
zipダウンロード

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。

☐ 名前

📁 Ambient_ESP32_lib-master
📦 Ambient_ESP32_lib-master.zip

ダウンロード > Ambient_ESP32_lib-master

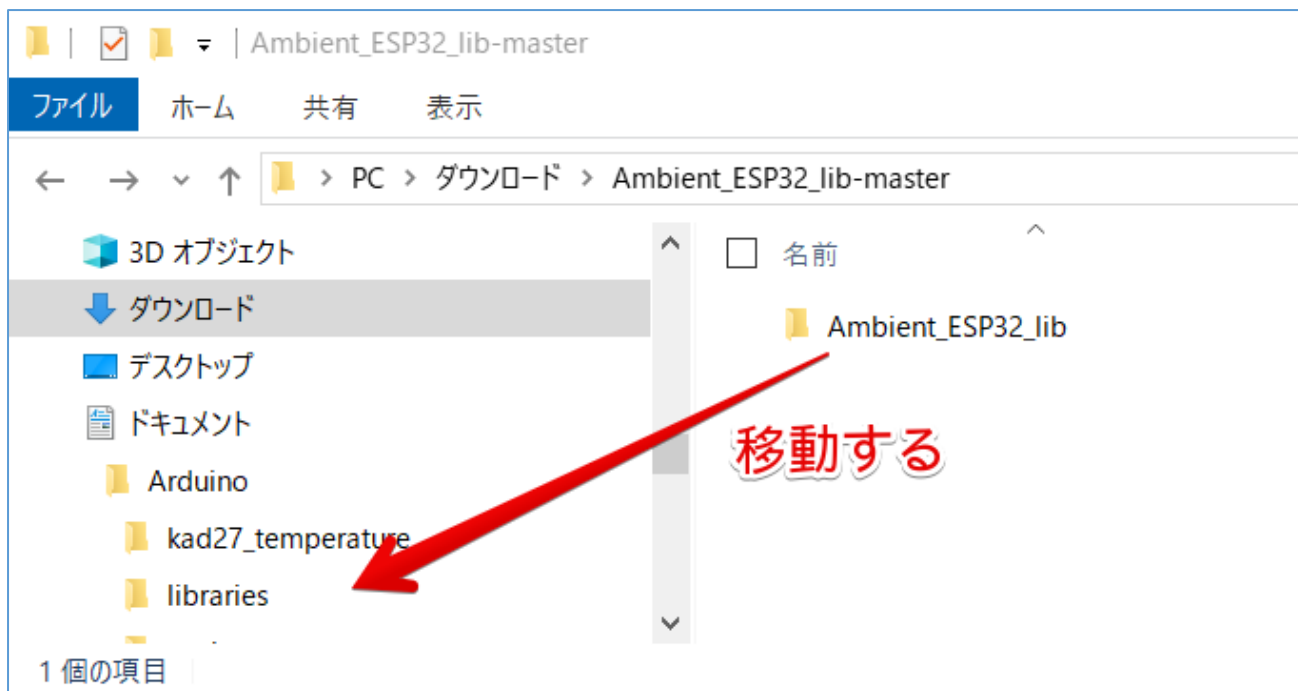
☐ 名前

📁 Ambient_ESP32_lib-master

「Ambient_ESP32_lib-master」フォルダ内の

「Ambient_ESP32_lib-master」フォルダを

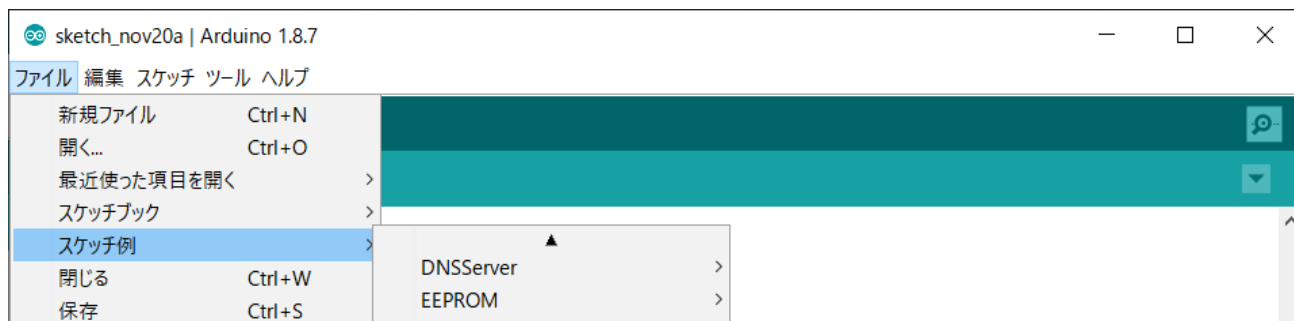
「Ambient_ESP32_lib」に書き換えて、ドキュメントフォルダ内の Arduino フォルダにある「libraries」フォルダに移動する。



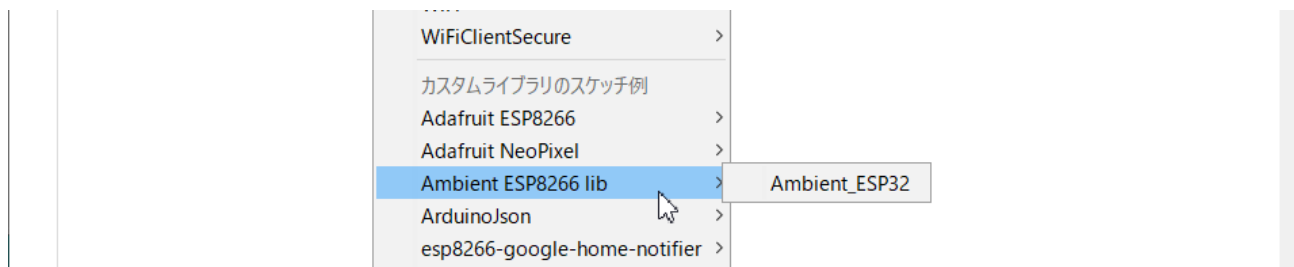
このとき、Arduino IDE を起動しているままならば、いったん Arduino IDE を終了させ、再度起動しなおす。

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ワード文字列:

"0b8b413f2c0fa6aa90e085e9431abbf1fa1b2bd2db0ecf4ae9ce4b2e87da770c"

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

Ambient

公開チャンネル

Myチャンネル

ブログ

ドキュメント

お知らせ

ログアウト

kyoshida

Myチャンネル

ユーザーキー:

writeKeyにコピー＆ペーストする

チャンネル名	チャンネルID	リードキー	ライトキー	作成日
🔒 チャンネル7901	7901	c23ab7d3e389822f	8088d1c1d6d5349a	2018/11/20
🔒				

チャンネルを作る

channelIdにコピー＆ペーストする

AmbientData Inc.

利用

課題 28 ソース書き換え

【宣言部】

kad28_Ambient_ESP32

```
1 #include <WiFi.h>
2 #include "Ambient.h"
3
4 // 接続先のSSIDとパスワード 学内CampusIoT
5 const char* ssid = "CampusIoT-WiFi";
6 const char* password = "0b8b413f[redacted]a90e085e!";
7
8 unsigned int channelId = [redacted];
9 const char* writeKey = "2[redacted]";
10
11 WiFiClient client;
12 Ambient ambient;
```

自分のチャンネルID

自分のチャンネルのライトキー

【setup() メソッド部】

```
13 void setup(){
14     Serial.begin(115200);
15     WiFi.begin(ssid, password);
16
17     while (WiFi.status() != WL_CONNECTED) {
18         delay(500);
19         Serial.print(".");
20     }
21     Serial.println("WiFi connected");
22     Serial.print("IP address: ");
23     Serial.println(WiFi.localIP());
24
25     ambient.begin(channelId, writeKey, &client);
26 }
```

追加する

【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); ←追加
36   delay(5000);
37 }
```

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 のチャン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。

Ambient 公開チャンネル Myチャンネル

ブログ ドキュメント ▾ お知らせ ▾ ログアウト kyoshida ▾

Myチャンネル / 🔒 チャンネル7901 (チャンネルID: 7901) 👁 ⚙ ⬇ 🗑



最新データ 2018/11/20
—登録: 6:16:03

日時が表示されていれば
データが受信されている

クリックしてグラフの追加

[+]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○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。

チャート追加

グラフ名

ランダム

グラフ種類

折れ線グラフ

グラフサイズ

medium

d1

☐ 表示なし

☒ 左軸

☐ 右軸

d2

☐ 表示なし

☐ 左軸

☐ 右軸

d3

☐ 表示なし

☐ 左軸

☐ 右軸

d5

☐ 表示なし

☐ 左軸

☐ 右軸

d7

☐ 表示なし

☐ 左軸

☐ 右軸

d4

☐ 表示なし

☐ 左軸

☐ 右軸

d6

☐ 表示なし

☐ 左軸

☐ 右軸

d8

☐ 表示なし

☐ 左軸

☐ 右軸

左軸

最小値

最大値

log?

☐

右軸

最小値

最大値

log?

☐

軸の最小値、最大値は空白のままにすれば自動的に設定されます。

表示件数

300

日付指定

☐

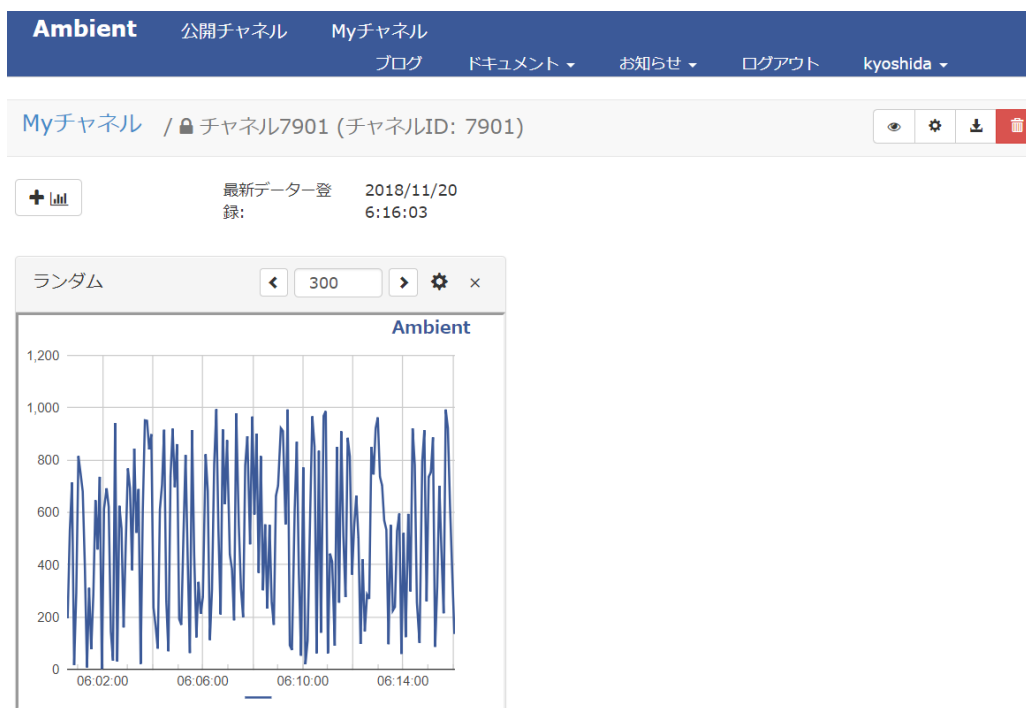
集計

の

追加せずに閉じる

チャートを追加

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」

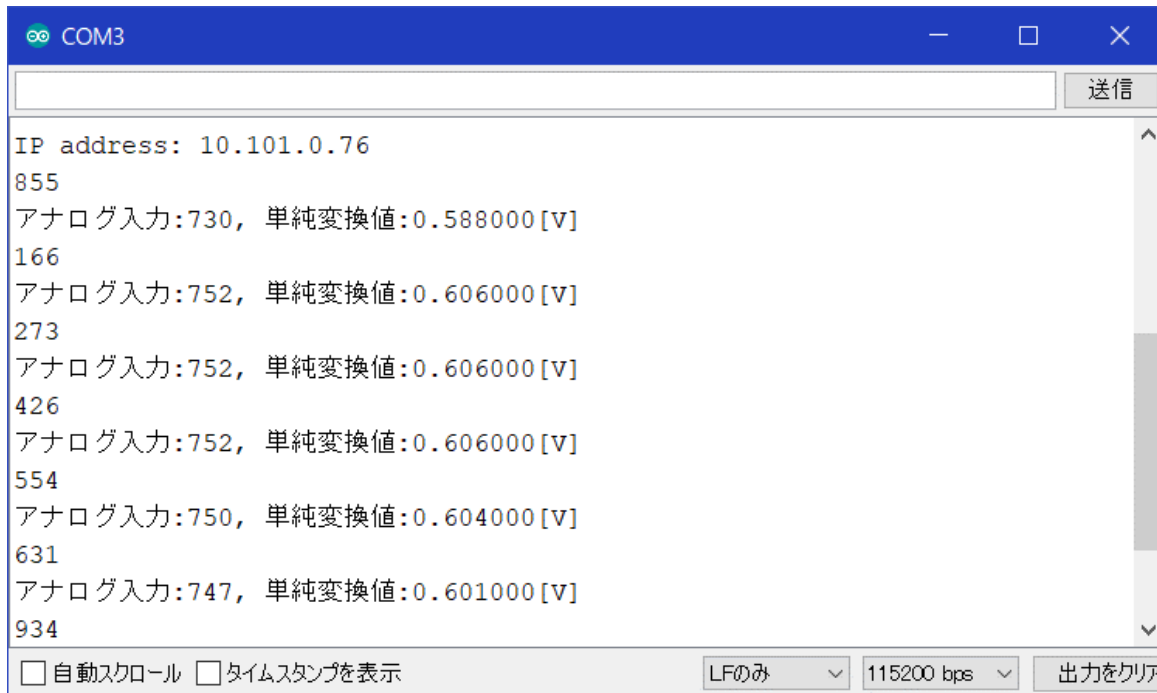
課題 28 のソースコードをそのままコピーして、ボリューム（可変抵抗）の電圧値を Ambient にアップロードせよ。チャンネル ID（ライトキーも）はそのまま、d2 にエントリする形にする。

d1 ヘ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。

