

## 「IoT 制作演習 I」第 13 章 MQTT を用いた PubSub 通信①

### MQTT で 1 対多通信を行う

MQTT (MQ Telephony Transport) は、パブリッシュ／サブスクライブ型と呼ばれる 1 対多通信が可能なプロトコルである。実際に Arduino IDE で実装してみる。

#### 演習①

##### 【目標】

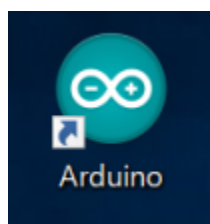
MQTT で 1 対多通信を行えるようにする。

##### 【1. ESP32 と電子工作部品との接続】

###### 1. 1. 必要な部品

今までに使った部品

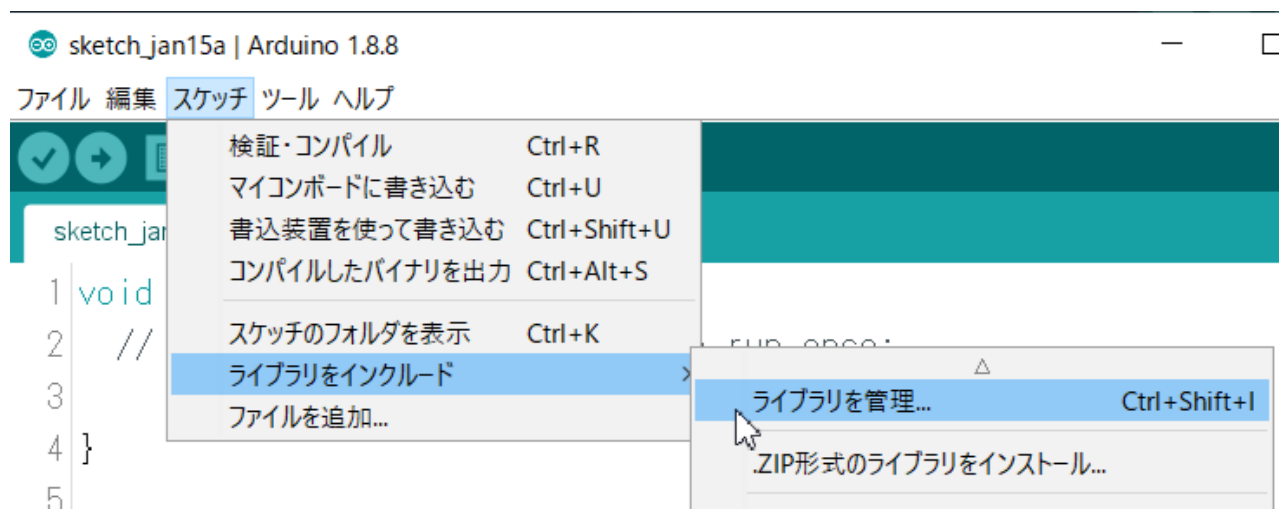
##### 【2. Arduino スケッチのサンプルプログラムを実行】



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。

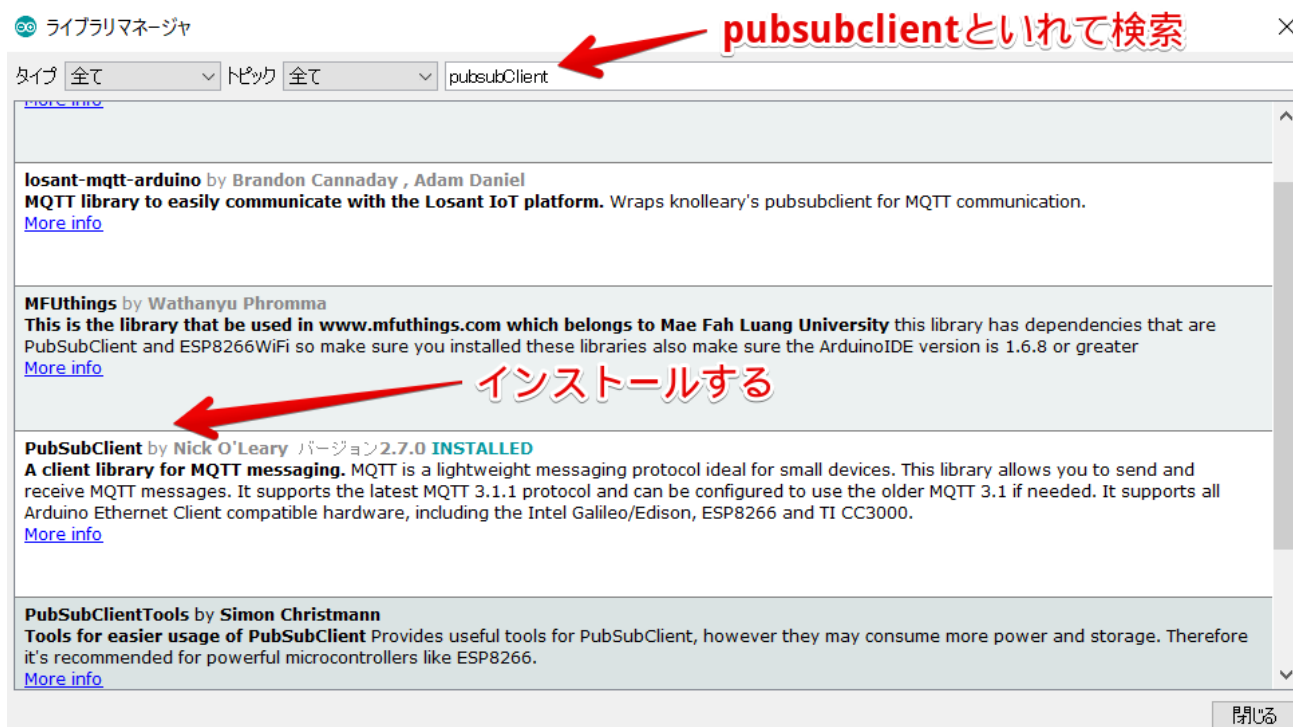
ライブラリ「PubSubClient」をダウンロードしましょう。

メニュー[スケッチ]—[ライブラリをインクルード]—[ライブラリを管理...]を選択



「pubsubclient」と入れて検索し、Nick O'Leary の PubSubClient をインストールする。

(次の図のようにスクロールさせて見つける必要があります。バージョン 2.8.0 になりました。)



### 【チーム課題 01】プロジェクト名「team01\_mqtt\_esp32」

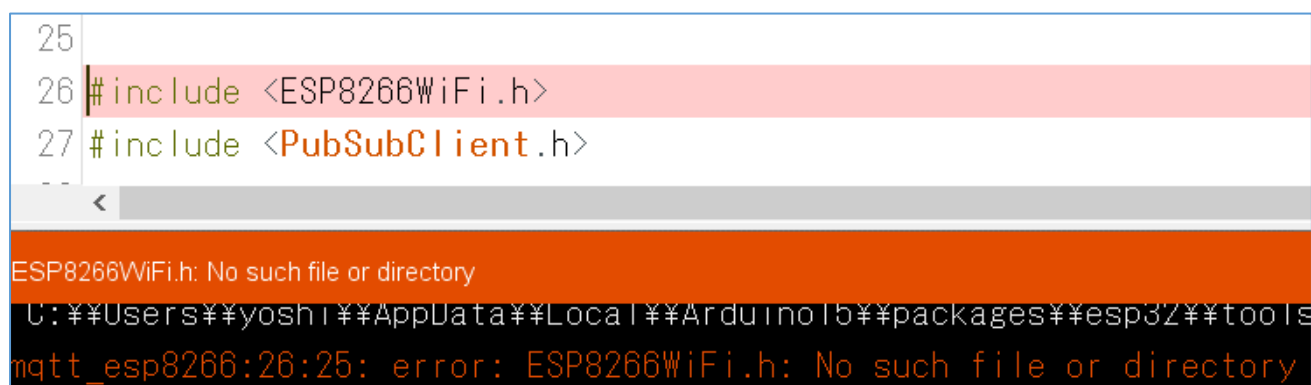
ファイル > スケッチ例 > 「PubSubClient」 > 「mqtt\_esp8266」を開いて動作確認をする。  
名前をつけて本課題として保存する。

ただし、若干作業が必要。



コンパイルのみ行って、エラーをつぶしていくこと。

#### エラー①



ESP8266 用のコアライブラリを読み込んでいるようなので、ESP32 用のヘッダーファイルに書き換える。

## エラー②

```
109 void setup() {
110     pinMode(BUILTIN_LED, OUTPUT);    // Initialize the BUILTIN_LED pin
    <
'BUILTIN_LED' was not declared in this scope
exit status 1
'BUILTIN_LED' was not declared in this scope
```

BUILTIN\_LED マクロが定義されていない。#define で BUILTIN\_LED を I016 番ピンに定義して、赤色 LED が光るようにする。

## エラー③（このエラーが出ない場合もある）

```
127     ++value;
128     snprintf(msg, 50, "hello world #%ld", value);
    <
format '%ld' expects argument of type 'long int', but argument 4 has type 'int' [-Werror=format=]
exit status 1
format '%ld' expects argument of type 'long int', but argument 4 has type 'int' [-Werror=format=]
```

書式指定子%ldが、変数 value の型に一致しないようなので、value の型に合うように修正する。

その他、コンパイルエラーが出るようであれば適宜修正する。

修正が済めば Wi-Fi 接続を行う。33. 34 行目を以下の情報をもとに書きかえる。

学内マイコン/IoT 演習用 WiFi の接続設定情報	
ssid :	"CampusIoT-WiFi"
password:	"0b8b413f2c0fa6aa90e085e9431abbf1fa1b2bd2db0ecf4ae9ce4b2e87da770c";

```
31 // Update these with values suitable for your network.
32
33 const char* ssid = ".....";
34 const char* password = ".....";
35 const char* mqtt_server = "broker.mqtt-dashboard.com";
```

〔ポイント〕 35 行目では接続する MQTT ブローカーを指定している（ココを変えれば様々な MQTT ブローカーにアクセスできる）。

【チーム課題 01 実行結果】 以下の様な Publish のメッセージが出つづけていれば OK

COM21

Connecting to school8x9express

....

WiFi connected

IP address:

192.168.3.215

Attempting MQTT connection...connected

Publish message: hello world #1

Message arrived [inTopic] 0

Publish message: hello world #2

Publish message: hello world #3

Publish message: hello world #4

Publish message: hello world #5

Publish message: hello world #6

Publish message: hello world #7

☐ 自動スクロール ☐ タイムスタンプを表示

←MQTT ブローカーへの接続

←1 回目の Publish メッセージ

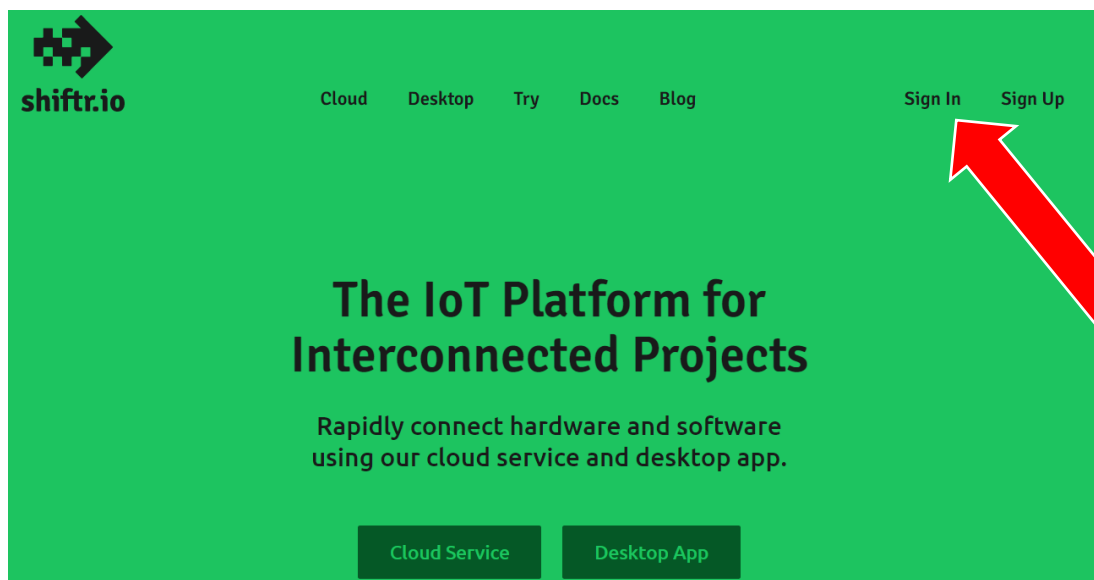
←1 度だけ Subscribe が来る。

←2 回目の Publish メッセージ

...

## 【チーム課題 02】 プロジェクト名「team02\_mqtt\_shiftr」

サンプルソースの MQTT ブローカーはパブリックで自由に Subscribe するのが難しいため、MQTT メッセージ・ブローカーのクラウド・サービスを行っている shiftr.io を使う。



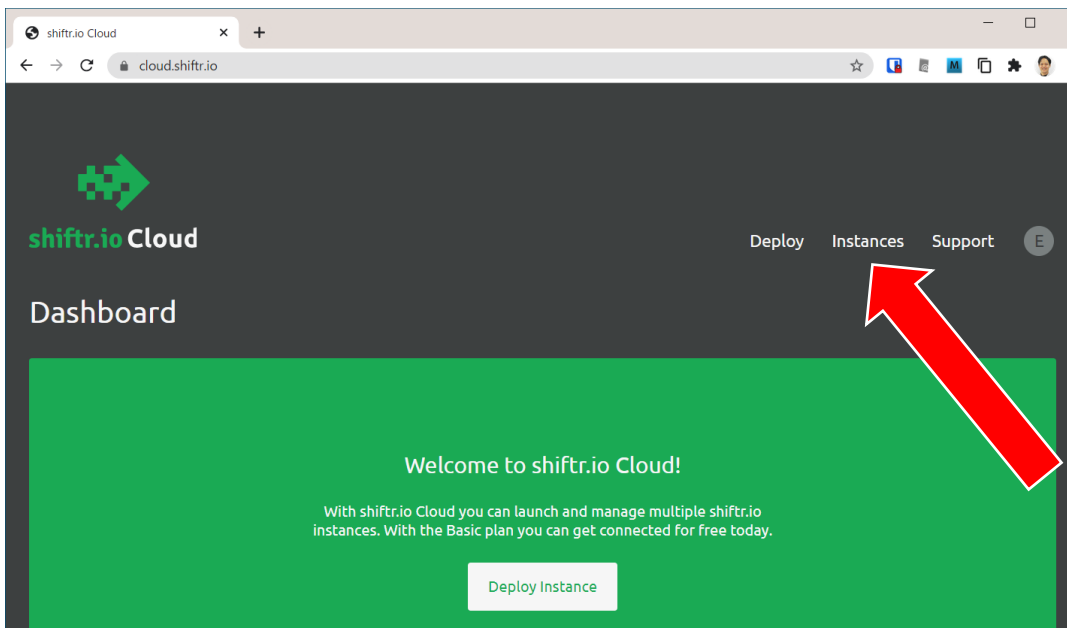
こちらに接続して、Publish/Subscribe を行う。

<https://www.shiftr.io/> にアクセスし右上の[Sign In] ボタンをクリックして、以下のユーザーでログインする。

Email Address: **eccandroid10a@gmail.com**

Password: **Nakazakicho2020**

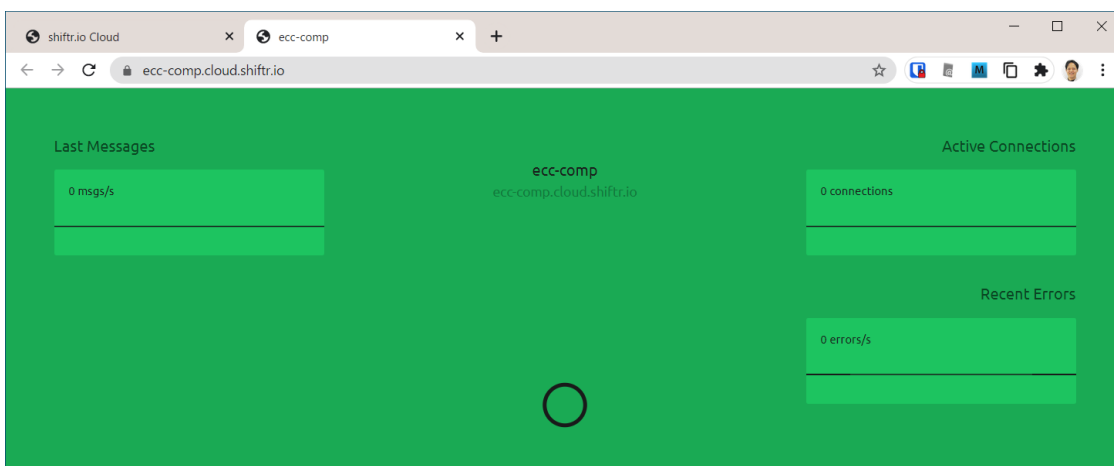
ログインできたらダッシュボード上の「Instances」リンクをクリック。



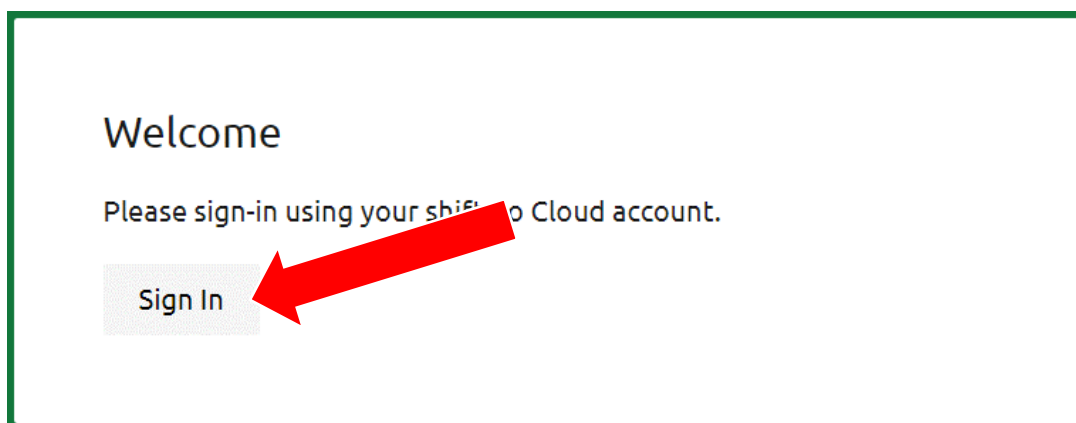
以下の様に、MQTT ブローカーのインスタンスの一覧が表示される。画面をスクロールさせてインスタンス「ecc-comp3」を見つける。インスタンス「ecc-comp3」の Web サービス URL「ecc-comp3.cloud.shifttr.io」をクリックする。**※クリックしてから 6 時間が有効期限となる**



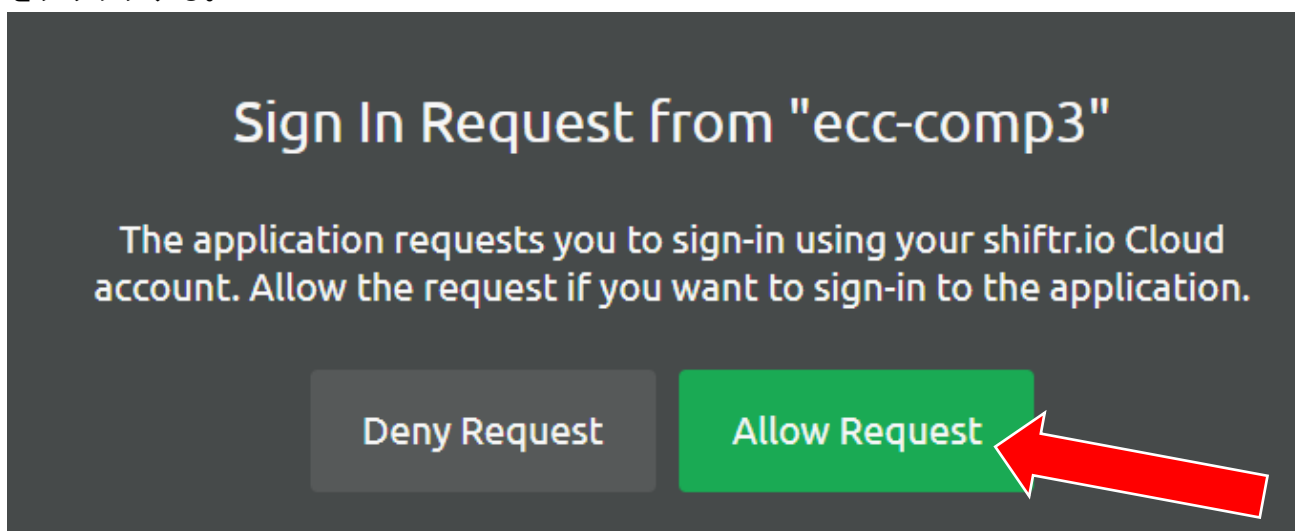
以下のように新しいタブで、「ecc-comp.cloud.shifttr.io」の Web サービス画面が表示される。



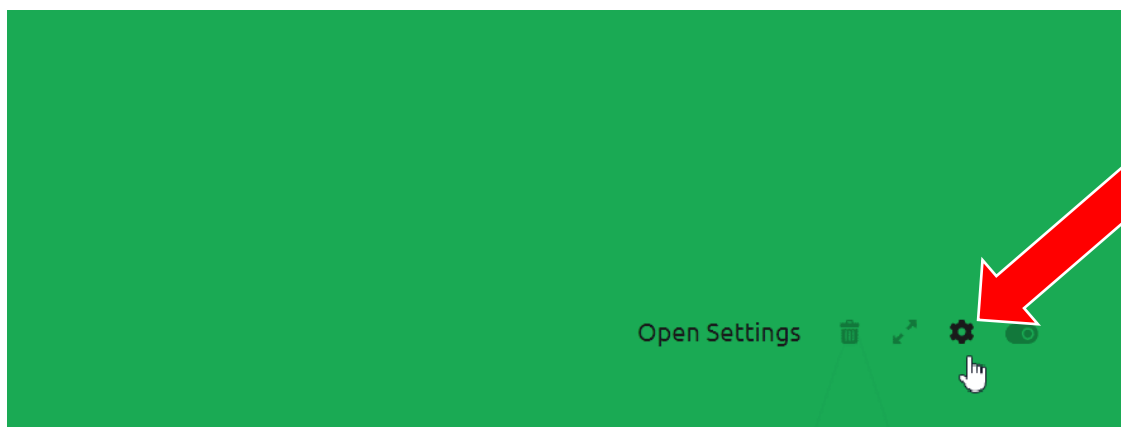
もし、サインインするような画面が出たら、再度サインインする。



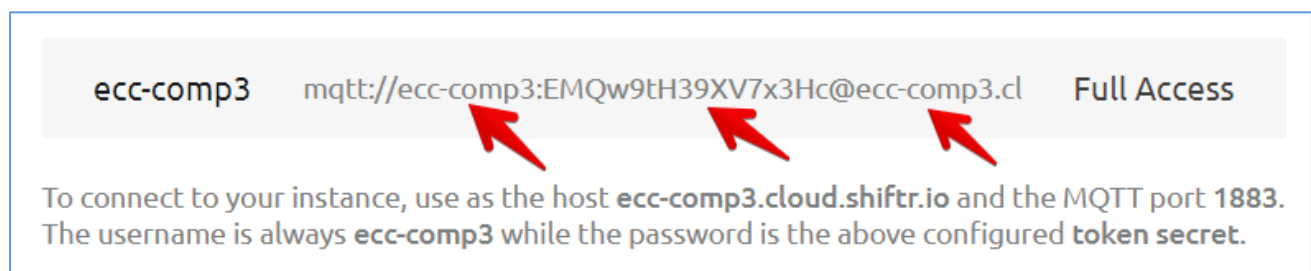
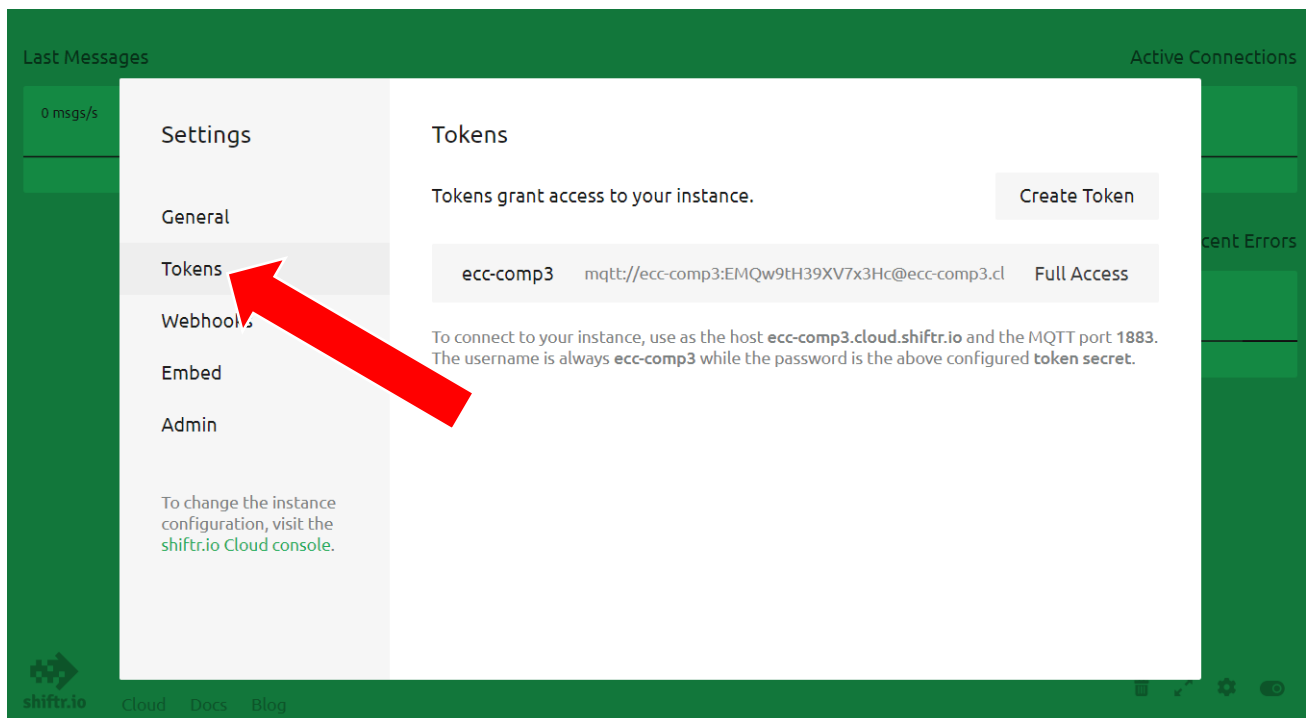
サインインした場合、リクエストを要求されたら Allow(許可)するようにしておくため「Allow Request」をクリックする。



Web サービス画面の右下の歯車をクリックして設定を表示させる。



次の図のように、「Settings」ダイアログ画面内の「Tokens」項目をクリックして、トークン(接続情報)を表示させる。3つの矢印の部分が、課題作成ソースに必要な情報になる。

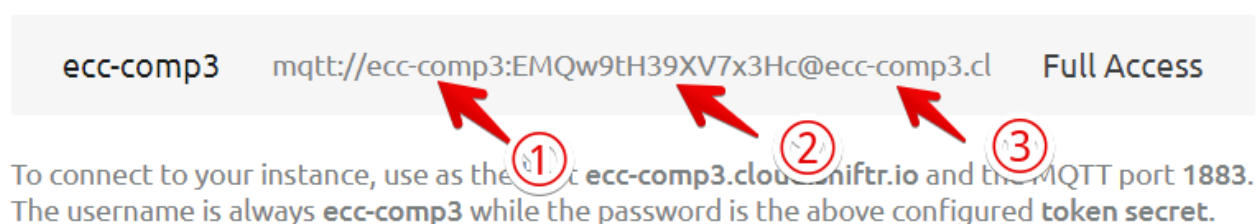


※↑上図のリンクをクリックすると詳細情報が表示されるが、変更可能だが実習中にはあまり表示させない方がいいと思う（間違えて書き換えると全員に迷惑がかかる…）。

という訳で以下に接続情報のテキストを貼り付けておくので、ここからコピー＆ペーストして下さい

```
mqtt://ecc-comp3:EMQw9tH39XV7x3Hc@ecc-comp3.cloud.shiftr.io
```

3つの矢印の部分は以下のようになる。



- ①キー（ユーザー名）
- ②シークレット（パスワード）
- ③MQTT ブローカーのサーバーアドレス

## 課題ソース作成

前回のチーム課題 01 のソースをコピーして作成し、以下のように改造する。

- (1) グローバル変数 mqtt\_server に MQTT ブローカーのサーバーアドレスを指定する。

```
//const char* mqtt_server = "broker.mqtt-dashboard.com";  
const char* mqtt_server = "ecc-comp3.cloud.shiftr.io";
```

- (2) reconnect() 関数内で、MQTT サービスにアクセスする①ユーザー名と②パスワードを指定する。

```
// Attempt to connect  
// if (client.connect(clientId.c_str())) {  
//ecc-comp3用①ユーザー名と②パスワード  
if (client.connect(clientId.c_str(), "ecc-comp3", "EMQw9tH39XV7x3Hc")) {
```

- (3) reconnect() 関数内で、クライアント ID を「クラス名+出席番号 2 桁」にする。

```
//String clientId = "ESP8266Client-";  
//clientId += String(random(0xffff), HEX);  
String clientId = "se2a00";
```

- (4) reconnect() 関数内で、トピックを“クラス名/チーム名”に、送信メッセージを“自分の名前 (ローマ字)”にして、Publish (発信・投稿) するように指定する。

```
// client.publish("outTopic", "hello world");  
client.publish("se2a/team-a", "Yoshida");
```

↑ここでは、トピックが“se2a/team-a” = SE2A クラスのチーム A 班、送信メッセージは“Yoshida” = 吉田先生の名前 (ローマ字)

- (5) reconnect() 関数内で、トピックを“クラス名/#”にして、クラス名トピックに該当するすべてのトピックを Subscribe (購読) するように指定する。

```
// client.subscribe("inTopic");  
client.subscribe("se2a/#");
```

↑ここでは、se2a トピック内のすべてのトピックのメッセージが受信される。

- (6) loop () 関数内で、Hello World の文字列を自分の名前 (ローマ字) に変えておく。



```

    lastMsg = now;
    ++value;
//    snprintf (msg, 50, "hello world #%d", value);
    snprintf (msg, 50, "Yoshida #%d", value);
    Serial.print("Publish message: ");

```

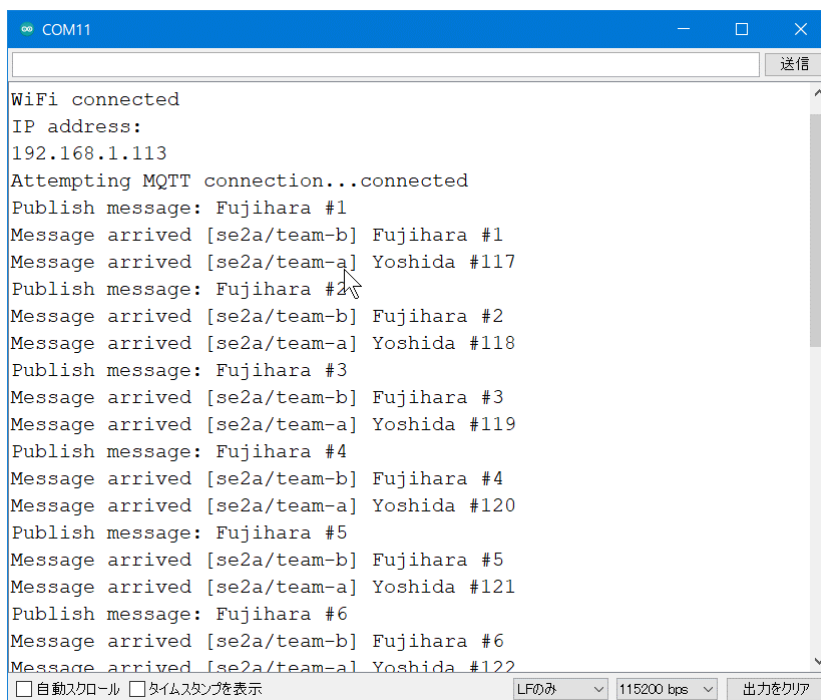
(7) loop () 関数内で、トピックを“クラス名/チーム名”にして、Publish (発信・投稿) するように指定する。

```

    Serial.print("Publish message: ");
    Serial.println(msg);
//    client.publish("outTopic", msg);
    client.publish("se2a/team-a", msg);
}
}

```

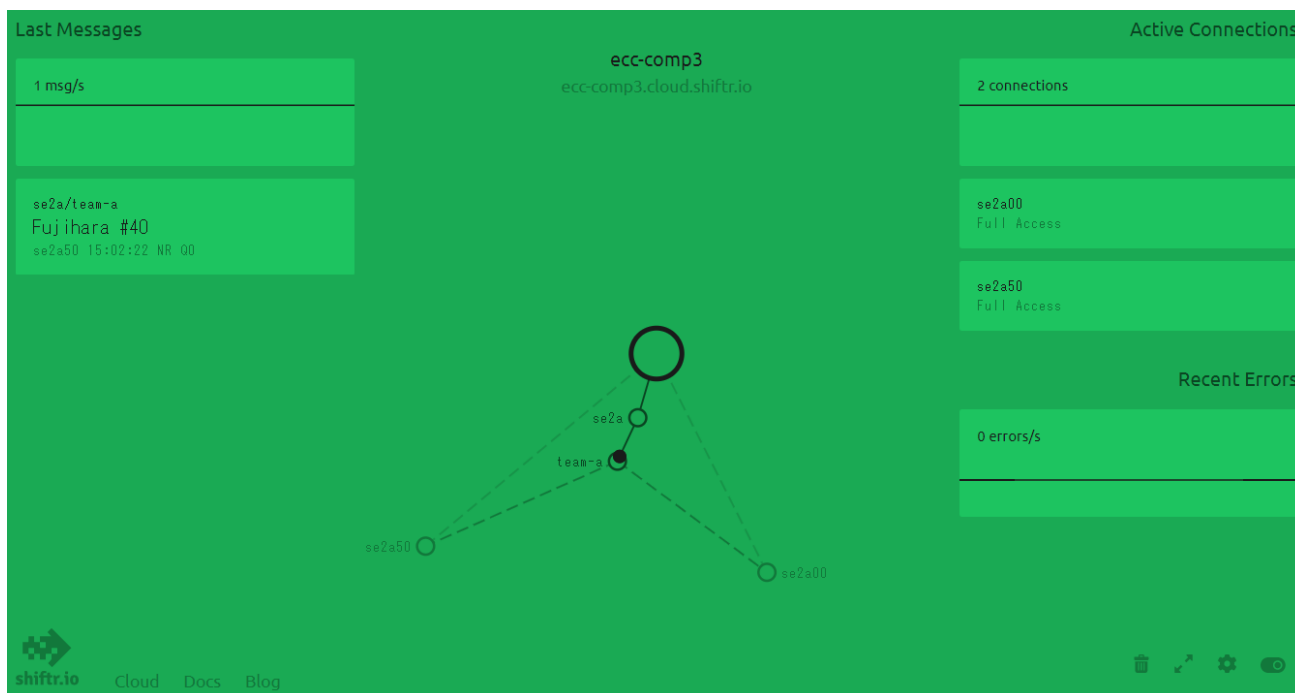
#### 【チーム課題 02 実行結果】



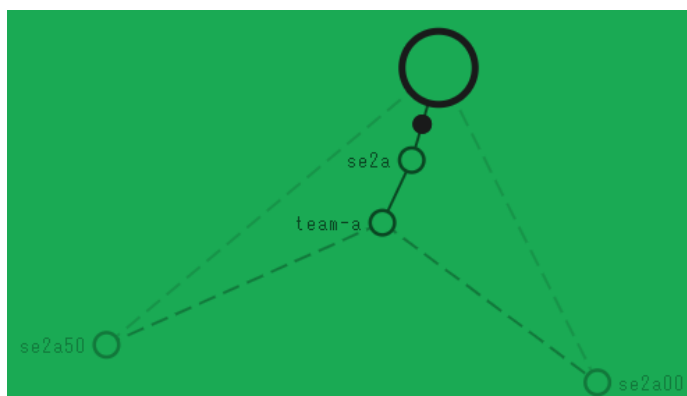
shiftr.io のリアルタイム画面で全員の Pub/Sub 動作を確認できる。

以下の URL をクリックして表示させてみよう。

<https://ecc-comp3.cloud.shiftr.io/>



ここでは、クラス se2a のチーム team-a に se2a00 の Yoshida、se2a50 の Fujihara がいる。



```
Attempting MQTT connection...connected
Publish message: Yoshida #1
Message arrived [se2a/team-a] Yoshida #1
Publish message: Yoshida #2
Message arrived [se2a/team-a] Yoshida #2
Message arrived [se2a/team-a] Fujihara
Publish message: Yoshida #3
Message arrived [se2a/team-a] Fujihara #1
Message arrived [se2a/team-a] Yoshida #3
Publish message: Yoshida #4
Message arrived [se2a/team-a] Fujihara #2
Message arrived [se2a/team-a] Yoshida #4
Publish message: Yoshida #5
Message arrived [se2a/team-a] Fujihara #3
Message arrived [se2a/team-a] Yoshida #5
Publish message: Yoshida #6
Message arrived [se2a/team-a] Fujihara #4
Message arrived [se2a/team-a] Yoshida #6
Publish message: Yoshida #7
```

☐ 自動スクロール ☐ タイムスタンプを表示

LFのみ

2 人とも、Subscribe (受信・購読) はトピック「se2a/#」をしているのでクラス内のすべてのトピックのメッセージが受信されている。

### 【チーム課題 03】プロジェクト名「team03\_mqtt\_shiftr\_team」

MQTT ブローカー・インスタンス「ecc-comp3」でパブリッシュ／サブスクライブができれば、チームごとに MQTT ブローカー・インスタンスを変えて実行する。前回の課題のソースをコピーして作成する。shiftr.io サイトで「instances」のリンクをクリックして、チームごとに設定を変えて各チームでトピックがやりとりされているか確かめること。



Instances の MQTT ブローカー・インスタンス一覧画面。チーム名（インスタンス）の Web サービス URL をクリックしてリアルタイムモニターを表示させる。

動作確認は、各々の Web サービス URL のページのリアルタイムモニターで確認できる。

team-a の Web サービス URL <https://team-a.cloud.shiftr.io/>

team-b の Web サービス URL <https://team-b.cloud.shiftr.io/>

team-c の Web サービス URL <https://team-c.cloud.shiftr.io/>

team-d の Web サービス URL <https://team-d.cloud.shiftr.io/>

team-e の Web サービス URL <https://team-e.cloud.shiftr.io/>

team-f の Web サービス URL <https://team-f.cloud.shiftr.io/>