

TALLER 4. Protocolo MQTT

Inicio

El objetivo de este taller es entender el funcionamiento del protocolo de transmisión MQTT a través de la implementación de diferentes brókers, subscriptores y publicadores.

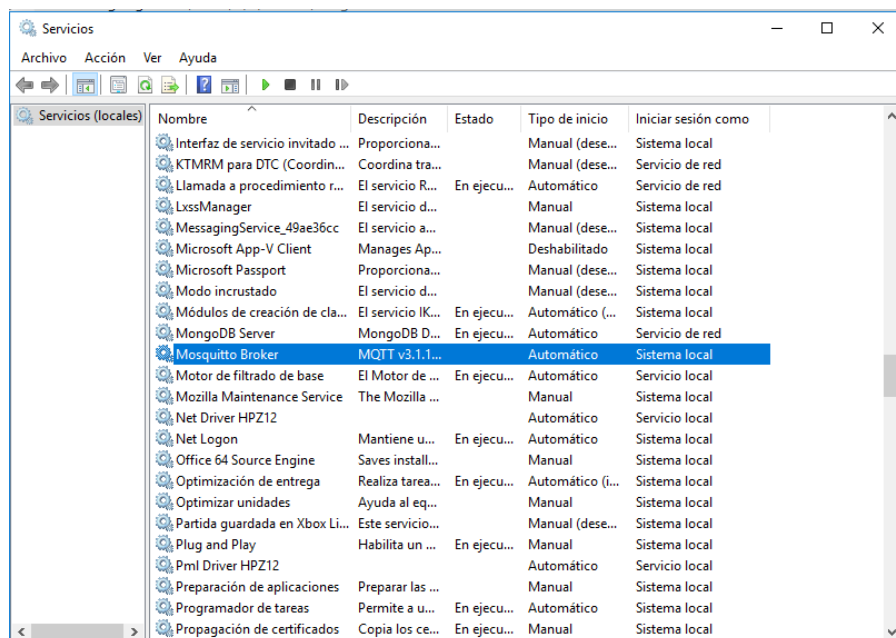
Se trabajará sobre objeto iot del proyecto grupal, realizando el envío de los datos en el formato JSON definido.

Se trabajará en 3 ambientes diferentes:

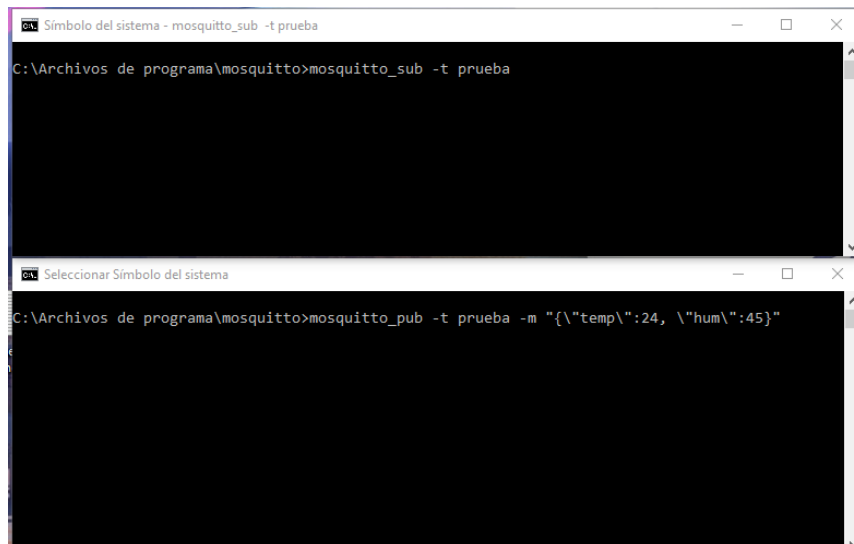
1. Bróker local - Mosquitto
2. Bróker en la nube - Maquiatto, Myqtt
3. Plataforma IoT que implemente el protocolo Mqtt – Cayenne, Ubidots.

Procedimiento – Bróker Local

1. **Broker Mosquitto.** Descargar mosquitto para Windows, del siguiente enlace: <https://mosquitto.org/download/>, instalarlo e iniciar el servicio, lo cual se puede hacer desde la ventana de Servicios de Windows.

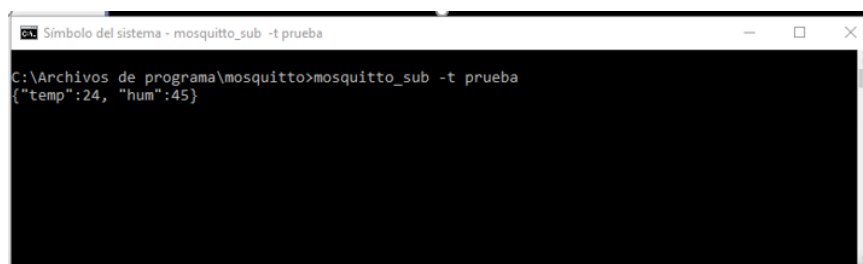


2. **Cientes Mosquitto.** Abrir dos ventanas de comandos de Windows, una para ejecutar un publicador de mosquitto y la otra para ejecutar un subscriptor.

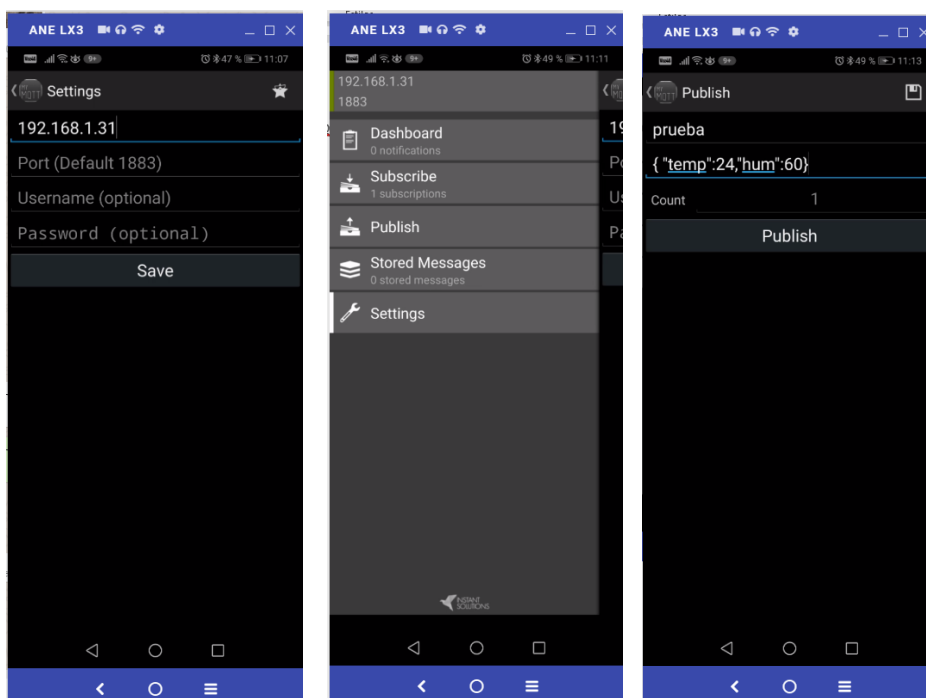


El comando `mosquitto_sub` lanza un subscriptor el cual se subscribe al tópic `prueba`.

El comando `mosquitto_pub` lanza un publicador, que publicará un mensaje en el tópic `prueba`. Al ejecutar el publicador el mensaje aparecerá en la ventana del subscriptor



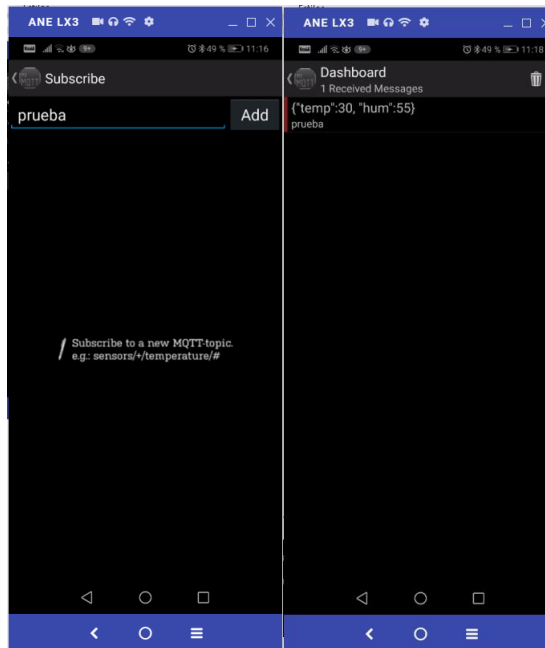
3. **Ciente smartphone.** Instale un cliente MQTT en su Smartphone, como MyMQTT y configúrelo para que se conecte al bróker local. Los datos que debe tener en cuenta para esta conexión son: la ip del bróker, el nombre del tópic.



Cuando se envía un mensaje desde el smartphone este aparece en lo subscriptores, y cuando se configura MyMqtt como subscritor el recibirá los mensajes enviados al tópico

```
Símbolo del sistema - mosquitto_sub -t prueba

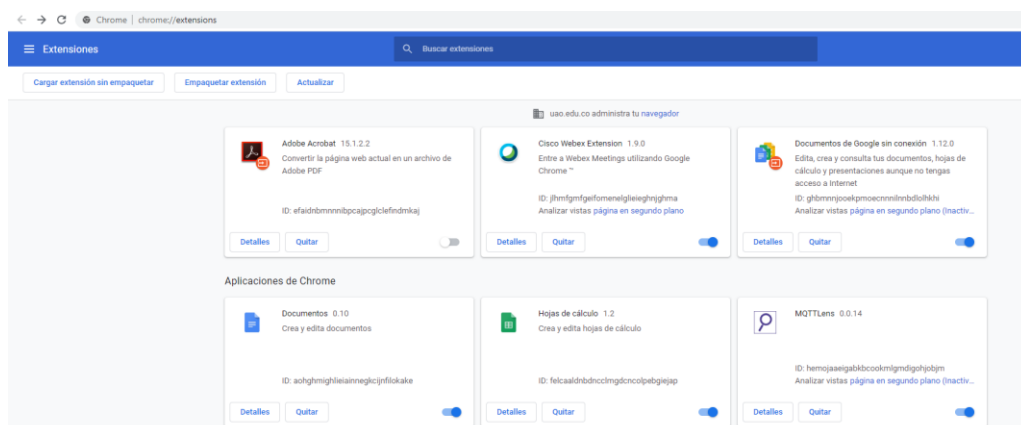
C:\Archivos de programa\mosquitto>mosquitto_sub -t prueba
{"temp":24, "hum":45}
{"temp":24,"hum":60}
```



```
Símbolo del sistema

C:\Archivos de programa\mosquitto>mosquitto_pub -t prueba -m "{\"temp\":24, \"hum\":45}"
C:\Archivos de programa\mosquitto>mosquitto_pub -t prueba -m "{\"temp\":30, \"hum\":55}"
C:\Archivos de programa\mosquitto>
```

4. Cliente MQTTLens. Instalar MQTTLens. Esta herramienta es un complemento de Chrome.



Configurar MQTTLens para que se conecte al bróker. Los datos que debe tener en cuenta para esta conexión son: la ip del bróker, el nombre del tópic.

Add a new Connection

Connection Details

Connection name

Mosquitto

Connection color scheme

Hostname

tcp://localhost

Port

1883

Client ID

lens_de10aSNGkbe50yXKdy7uuAEJmBM

Generate a random ID

Session

☒ Clean Session

Automatic Connection

☒ Automatic Connection

Keep Alive

120 seconds

Credentials

Username

Enter username

Password

Enter password

Last-Will

CANCEL

CREATE CONNECTION

<

Connection: Mosquitto

Subscribe

topic

0 - at most once

SUBSCRIBE

Publish

topic

0 - at most once

☐ Retained

PUBLISH

Message

Subscriptions

Publicar datos y recibir como subscriptor.

5. **Plataforma hardware (Arduino).** En nuestro hardware (plataforma+ sensores) incluir las librerías y el código que permite la conexión al bróker.

Antes de hacer la programación del Arduino se debe realizar la configuración de la conexión a la red wifi. Para esto siga el procedimiento que se encuentra en el siguiente link: <https://docs.arduino.cc/retired/getting-started-guides/ArduinoYun> en la sección **Configuring the onboard WiFi**

Para la conexión al bróker se debe usar la librería pubsubclient. Esta debe ser incluida en el IDE de Arduino.

A continuación, se muestran un sketch de ejemplo para publicar datos en el bróker en un tópico denominado **topico1**

```
#include <ArduinoJson.h>
#include <SPI.h>
#include <Bridge.h>
#include <BridgeClient.h>
#include <PubSubClient.h>

#define mqttUser ""
#define mqttPass ""
#define mqttPort 1883

char mqttBroker[] = "ip del equipo donde se encuentra mosquitto";
char mqttClientId[] = "ArduinoYun01"; //puede ser cualquier nombre
char inTopic[] = "topico1";

//función que se ejecuta cuando llega un mensaje en caso de subscribirse a un
tópico
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

BridgeClient BClient;
PubSubClient client(BClient);

void reconnect() {
    // Este ciclo se ejecuta mientras esté conectado al broker
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Intenta conectarse
        if (client.connect(mqttClientId, mqttUser, mqttPass)) {
```

```

        Serial.println("connected"); // cuando se conecta publica este
mensaje...
        //se leen los sensores y se construye el Json
        int sensor1 = analogRead(A0);
        int sensor2 = analogRead(A1);
        String variable;

        DynamicJsonDocument doc(1024);

        doc["sensor1"] = sensor1;
        doc["sensor2"] = sensor2;
        doc["idnodo"] = 1;
        doc["timestamp"] = 1655133862;

        serializeJson(doc, variable);
        int lon = variable.length()+1;
        Serial.println(variable); //muestra en la consola el Json
        //convierte el string del json a un arreglo de caracteres
        char datojson[lon];
        variable.toCharArray(datojson, lon);
        //publica el json
        client.publish(inTopic,datojson);
        //se desconecta del broker
        client.disconnect();
        delay(5000);

        //client.subscribe("topic2"); //en caso de querer subscribirse aun
topico
    } else { //en caso de falla de conexión
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        // espera 5 segundos y lo intents de nuevo
        delay(5000);
    }
}
}
}

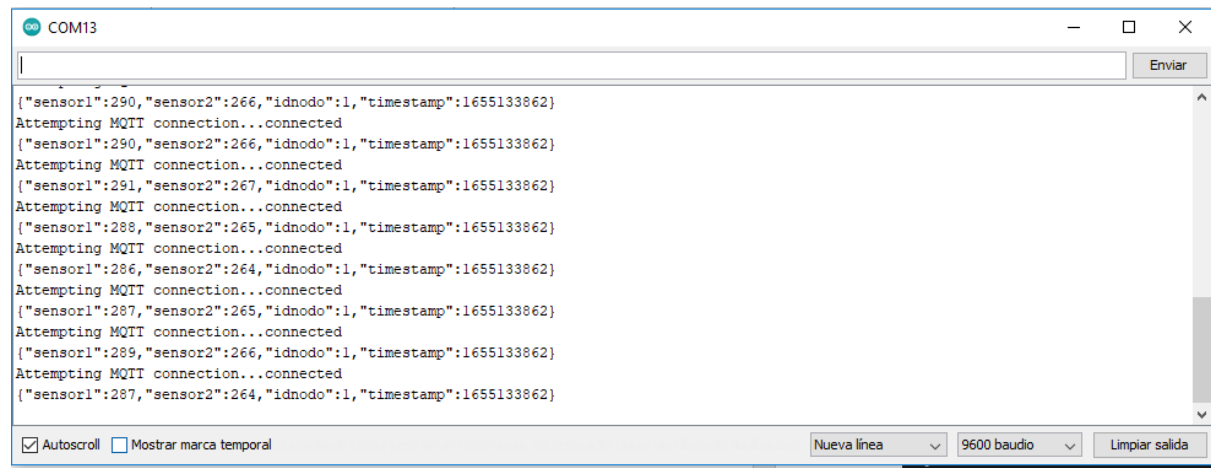
void setup()
{
    Serial.begin(57600);
    Bridge.begin();

    client.setServer( mqttBroker, mqttPort );
    client.setCallback( callback );
    Serial.println("Setup done");
    delay(1500);
}

```

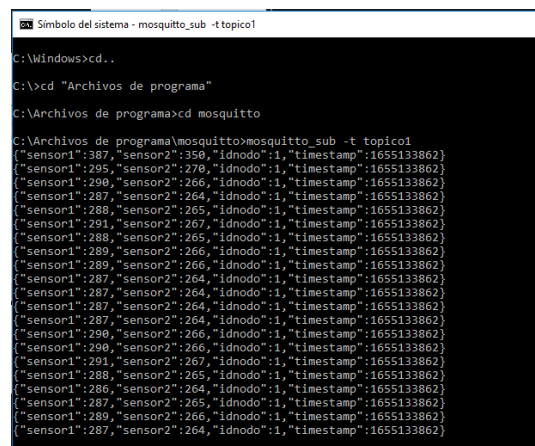
```
void loop()
{
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
```

Cuando se ejecuta el sketch, se puede observar que se envían los datos:

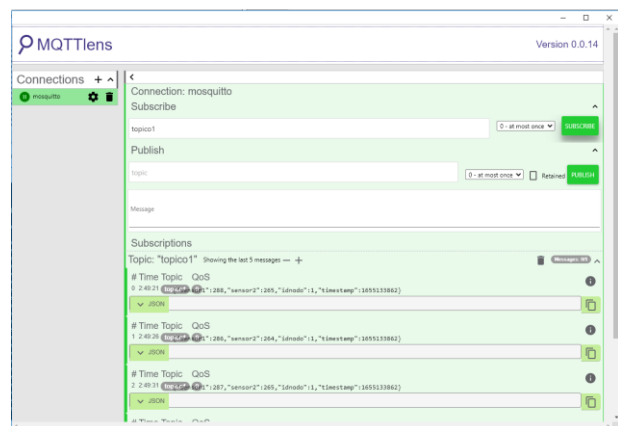


```
COM13
{"sensor1":290,"sensor2":266,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":290,"sensor2":266,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":291,"sensor2":267,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":288,"sensor2":265,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":286,"sensor2":264,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":287,"sensor2":265,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":289,"sensor2":266,"idnodo":1,"timestamp":1655133862}
Attempting MQTT connection...connected
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
```

y se reciben en los diferentes clientes subscriptores:



```
Simbolo del sistema - mosquitto_sub -t topico1
C:\Windows>cd .
C:\>cd "Archivos de programa"
C:\Archivos de programa>cd mosquitto
C:\Archivos de programa\mosquitto>mosquitto_sub -t topico1
{"sensor1":387,"sensor2":350,"idnodo":1,"timestamp":1655133862}
{"sensor1":295,"sensor2":270,"idnodo":1,"timestamp":1655133862}
{"sensor1":290,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":288,"sensor2":265,"idnodo":1,"timestamp":1655133862}
{"sensor1":291,"sensor2":267,"idnodo":1,"timestamp":1655133862}
{"sensor1":288,"sensor2":265,"idnodo":1,"timestamp":1655133862}
{"sensor1":289,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":289,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":290,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":290,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":291,"sensor2":267,"idnodo":1,"timestamp":1655133862}
{"sensor1":288,"sensor2":265,"idnodo":1,"timestamp":1655133862}
{"sensor1":286,"sensor2":264,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":265,"idnodo":1,"timestamp":1655133862}
{"sensor1":289,"sensor2":266,"idnodo":1,"timestamp":1655133862}
{"sensor1":287,"sensor2":264,"idnodo":1,"timestamp":1655133862}
```

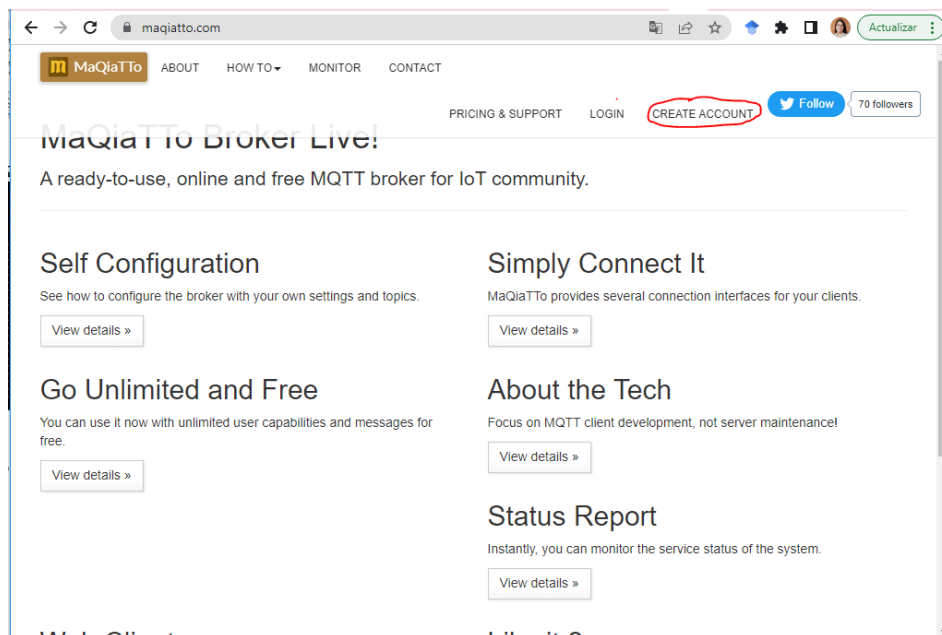


6. Seleccione otra plataforma hardware y realice el sketch para conectarse al bróker y publicar el json con los datos de los sensores, de la misma manera como se hizo con el Arduino Yun.

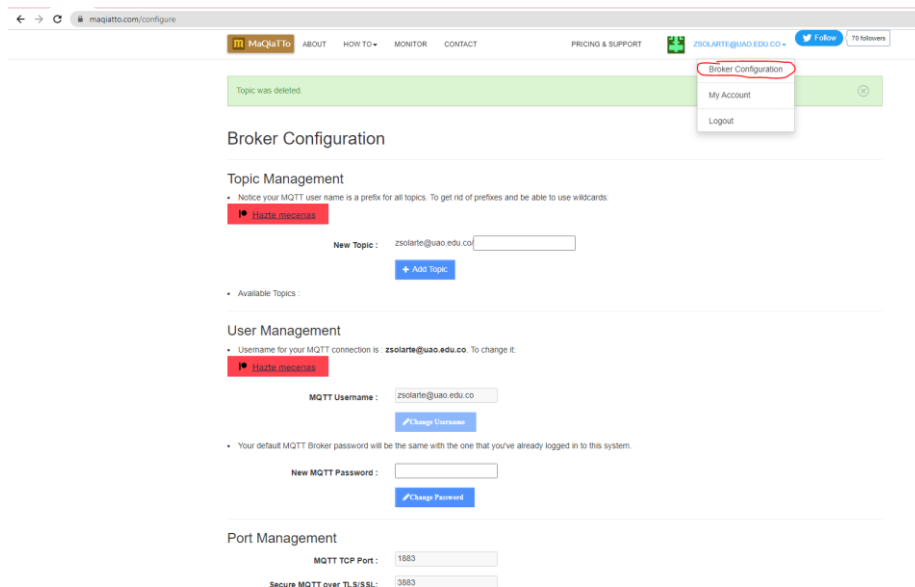
Procedimiento – Bróker en la nube

Desarrollar todo el proceso usando un bróker en la nube. En este taller usaremos maqiatto.

7. Lo primero es crear una cuenta:



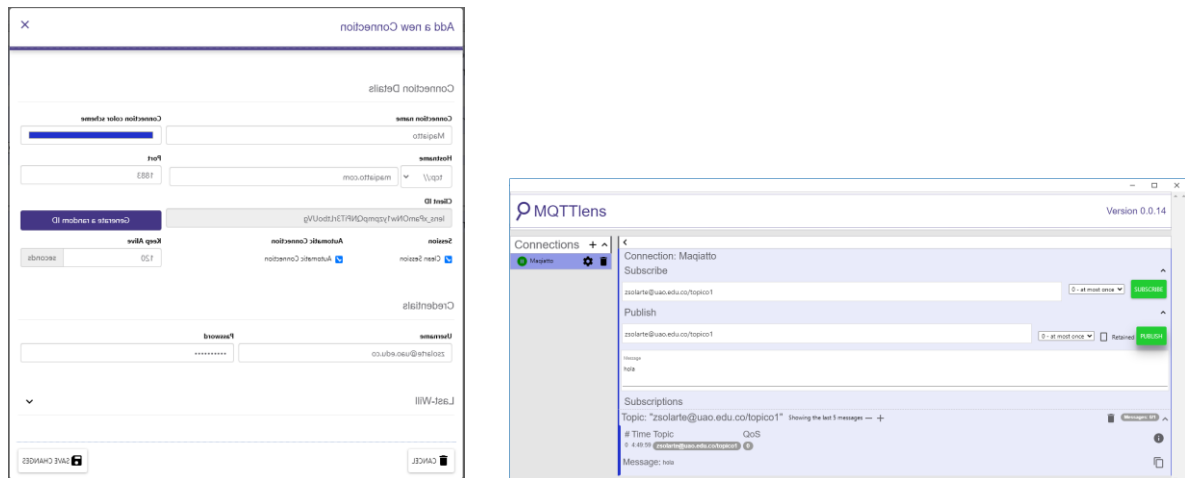
8. Después de ingresar se debe configurar el bróker, definiendo el usuario, el password y el tópicó que se deberá usar.



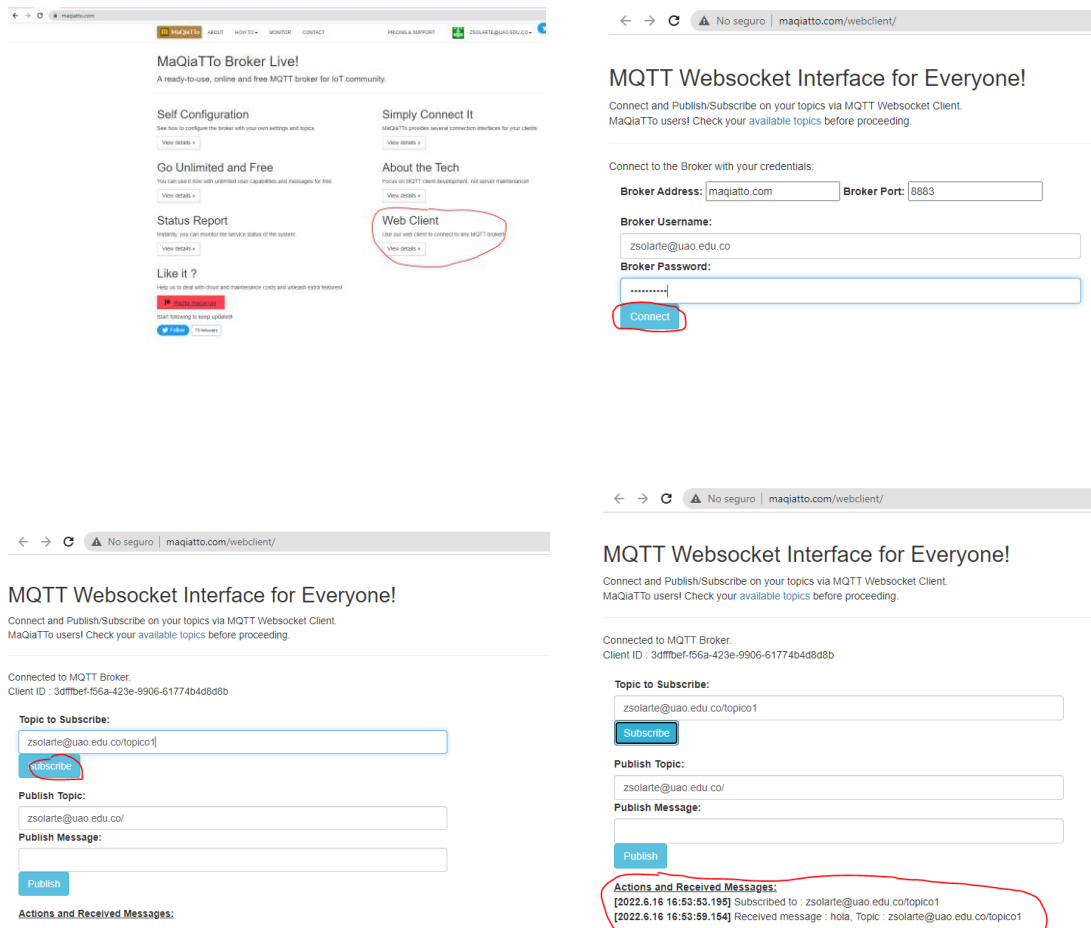
Por defecto el username es el correo electrónico, el password es el que se usó para ingresar al sistema y el tópicó es cualquier nombre al que se le antepone el correo electrónico seguido del carácter slash /. En mi caso usaré el tópicó zsolarte@uao.edu.co/topico1.

9. En todos los clientes usados debemos cambiar los parámetros de conexión y realizar las pruebas de publicación y suscripción.

En mqttLens quedaría de la siguiente manera:



Y en maqiatto se ven las publicaciones de la siguiente manera:



Se debe realizar la configuración del Arduino para que se conecte al bróker maqiatto, quedaría el sketch de la siguiente manera:

```

#include <ArduinoJson.h>
#include <SPI.h>
#include <Bridge.h>
#include <BridgeClient.h>
#include <PubSubClient.h>

#define mqttUser "zsolarte@uao.edu.co"
#define mqttPass "xxxx" //el password para conectarse a maqiatto
#define mqttPort 1883

char mqttBroker[] = "maqiatto.com";
char mqttClientId[] = "ArduinoYun01"; //puede ser cualquier nombre
char inTopic[] = "zsolarte@uao.edu.co/topico1";

//función que se ejecuta cuando llega un mensaje en caso de subscribirse a un
tópico
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived ");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

BridgeClient BClient;
PubSubClient client(BClient);

void reconnect() {
    // Este ciclo se ejecuta mientras esté conectado al broker
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Intenta conectarse
        if (client.connect(mqttClientId, mqttUser, mqttPass)) {
            Serial.println("connected"); // cuando se conecta publica este
mensaje...
            //se leen los sensores y se construye el Json
            int sensor1 = analogRead(A0);
            int sensor2 = analogRead(A1);
            String variable;

            StaticJsonDocument<256> doc;

            doc["sensor1"] = sensor1;
            doc["sensor2"] = sensor2;
            doc["idnodo"] = 1;
            doc["timestamp"] = 1655133862;

```

```

        serializeJson(doc, variable);
        int lon = variable.length()+1;
        Serial.println(variable); //muestra en la consola el json
        //convierte el string del json a un arreglo de caracteres
        char datojson[lon];
        variable.toCharArray(datojson, lon);
        //publica el json
        client.publish(inTopic,datojson);
        //se desconecta del broker
        client.disconnect();
        delay(5000);

        //client.subscribe("topic2"); //en caso de querer subscribirse aun
topico
    } else { //en caso de falla de conexión
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        // espera 5 segundos y lo intents de nuevo
        delay(5000);
    }
}
}
}

void setup()
{
    Serial.begin(57600);
    Bridge.begin();

    client.setServer( mqttBroker, mqttPort );
    client.setCallback( callback );
    Serial.println("Setup done");
    delay(1500);
}

void loop()
{
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

```

Los datos enviados por el arduino y recibidos en el bróker:



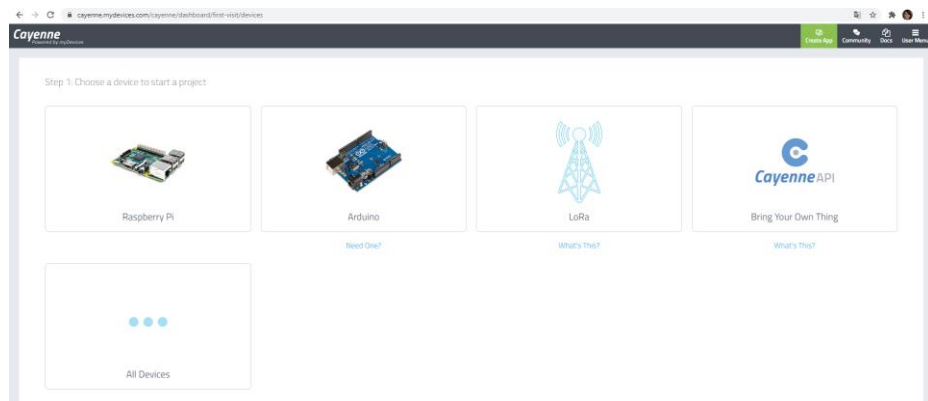
- ## Procedimiento – Plataforma de IoT con soporte Mqtt

-
- The screenshot shows the myDevices website. The header is dark blue with the myDevices logo on the left and navigation links (CAPABILITIES, SOLUTIONS & DEVICES, NO CODE IOT, SUCCESS, PARTNER) on the right. A red button labeled 'VIEW ALL SUCCESS STORIES' is centered below the header. The main content area is white and features a large heading 'Connect with Us' and social media icons for Twitter, Instagram, LinkedIn, and Facebook. The footer is dark blue and contains the myDevices logo, a list of links (myDevices, Platform, About, News, Careers, Contact, Industries, Hospitality, Healthcare, Campus, Workplace, Food Service, IoT in a Box, Capabilities, Hardware, Sign In, Device Manufacturers, Support, Cayenne, Features, Docs, Forum, Sign Up Free, Sign In), and copyright information (© 2020 myDevices, Inc.). The 'Cayenne' link is highlighted with a red circle.

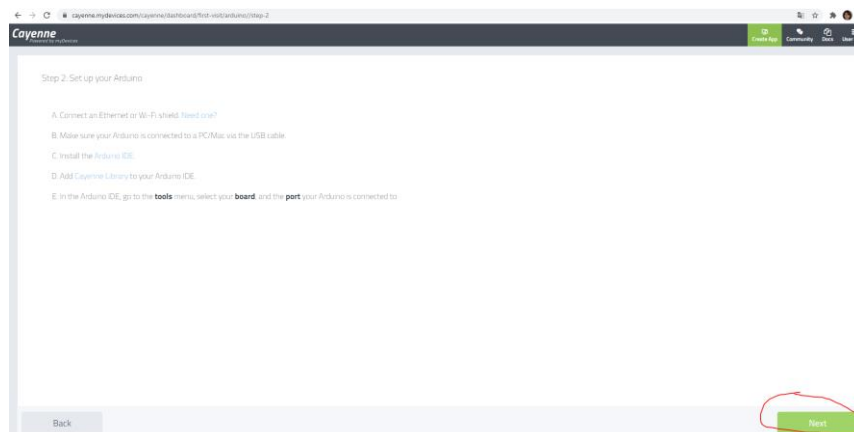


En este punto se debe crear la cuenta.

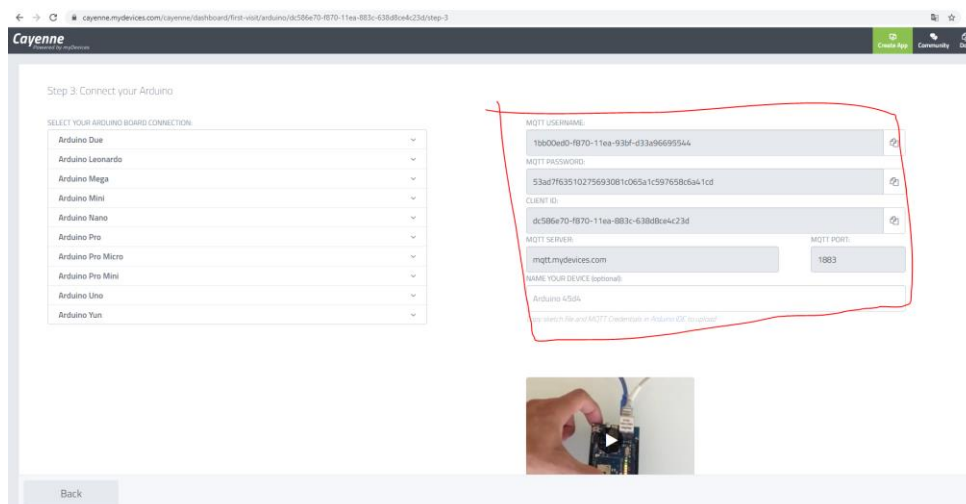
Ya habiendo ingresado, se muestra este dashboard.



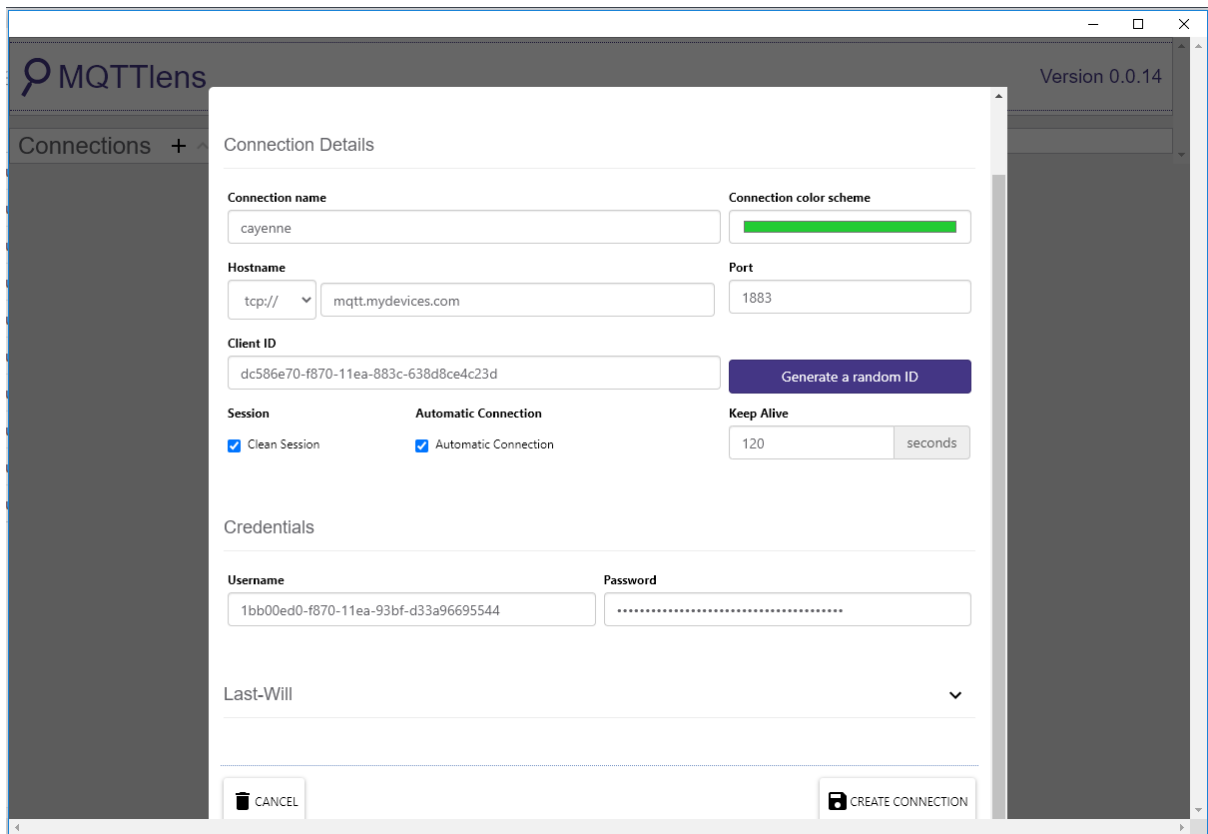
Seleccionamos Arduino, aunque esto no es tan relevante.



Estos son los parámetros para conectarse al bróker de mqtt de cayenne.



12. Desde un cliente como MQTTLens nos conectamos:



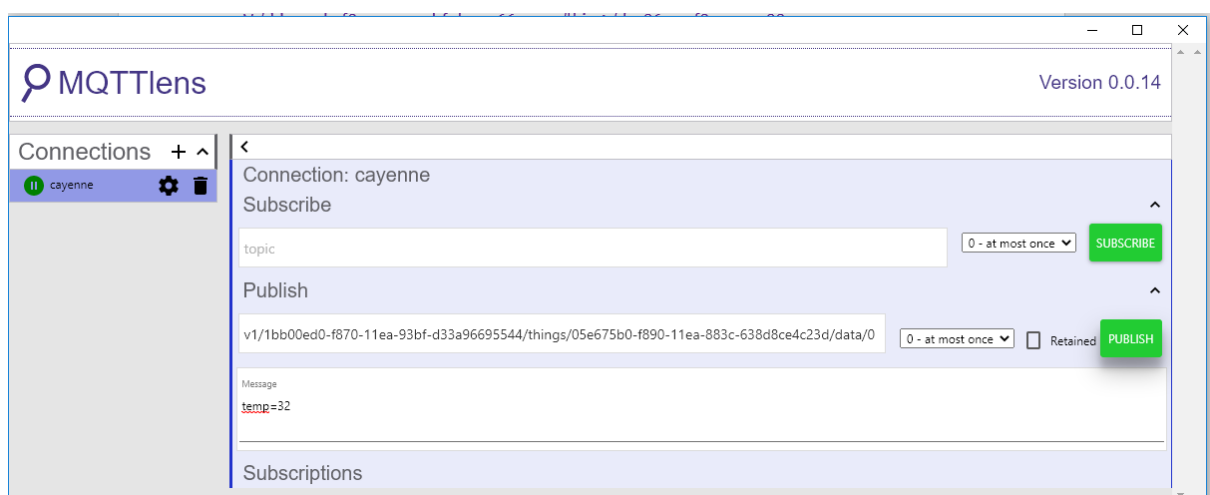
El t pico ser  el siguiente:

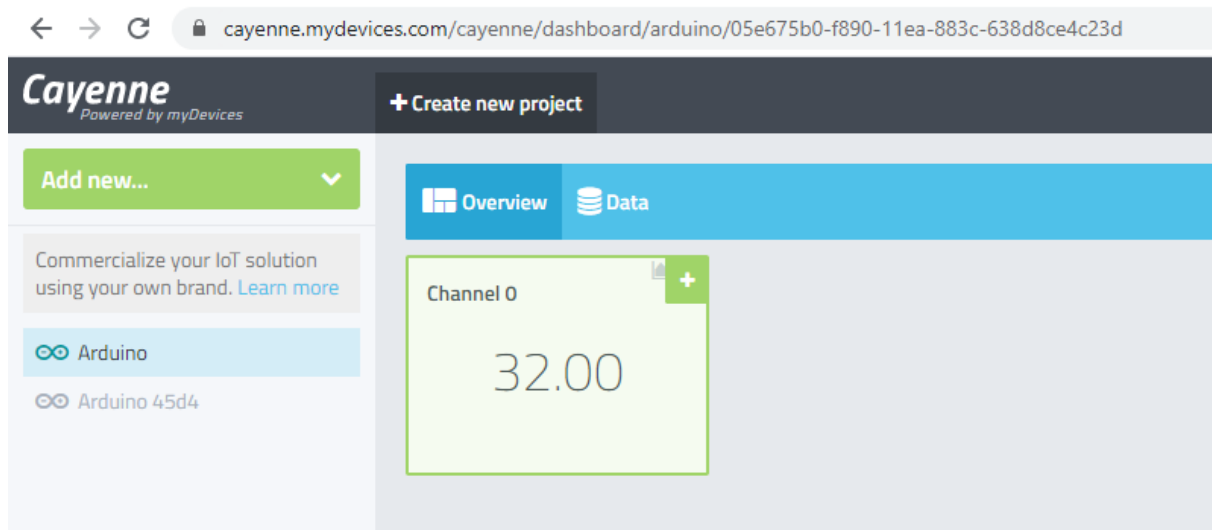
`v1/username/things/clientID/data/canal`

Se deben cambiar los valores con los adecuados, entregados por cayenne:

En mi caso ser a:

`V1/1bb00ed0-f870-11ea-93bf-d33a96695544/things/dc586e70-f870-11ea-883c-638d8ce4c23d/data/o`





Después de que aparece el canal con el dato, este debe añadirse al dashboard dando clic en el signo +.

También se pueden enviar datos en formato json, a través del siguiente tópico:

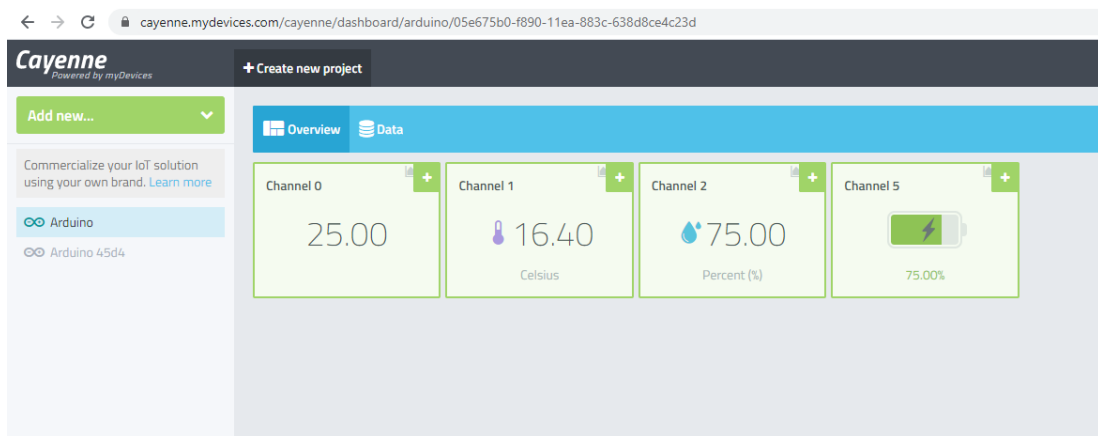
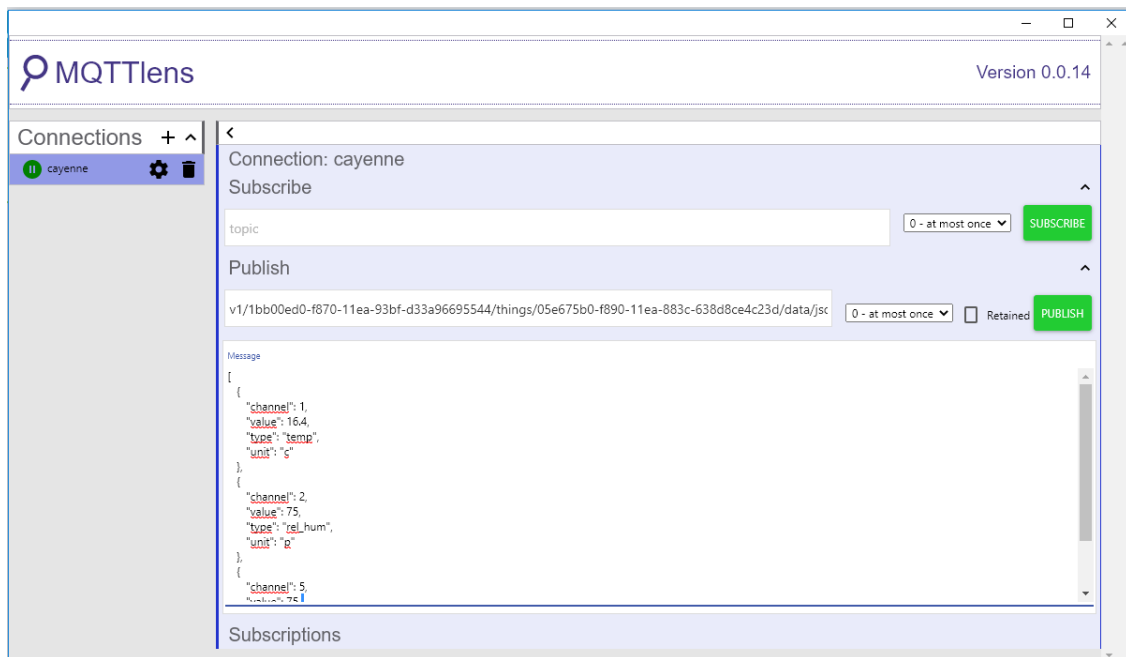
`v1/username/things/clientID/data/json`

En mi caso sería:

`V1/1bb00edo-f870-11ea-93bf-d33a96695544/things/dc586e70-f870-11ea-883c-638d8ce4c23d/data/json`

El mensaje sería un arreglo de json

```
[
  {
    "channel": 1,
    "value": 16.4,
    "type": "temp",
    "unit": "c"
  },
  {
    "channel": 2,
    "value": 75,
    "type": "rel_hum",
    "unit": "p"
  },
  {
    "channel": 5,
    "value": 75,
    "type": "batt",
    "unit": "v"
  }
]
```



13. En el Arduino el sketch quedaría de la siguiente manera (aquí lo importante es cambiar el formato del json, ya que debe ser un arreglo):

```
#include <ArduinoJson.h>
#include <SPI.h>
#include <Bridge.h>
#include <BridgeClient.h>
#include <PubSubClient.h>

#define mqttUser "b061af80-8e96-11e7-b546-bf6ccbcd8710"
#define mqttPass "910727b8aa2ba53687e67096cb46f8ec8500aaf6"
#define mqttPort 1883

char mqttBroker[] = "mqtt.mydevices.com";
char mqttClientId[] = "34557a00-edc5-11ec-8da3-474359af83d7";
char inTopic[] = "v1/b061af80-8e96-11e7-b546-bf6ccbcd8710/things/34557a00-edc5-11ec-8da3-474359af83d7/data/json";
```



```

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

BridgeClient BClient;
PubSubClient client(BClient);

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(mqttClientId, mqttUser, mqttPass)) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            int sensor1 = analogRead(A0);
            int sensor2 = analogRead(A1);
            String variable;

            StaticJsonDocument<256> arreglo; //Se crea el arreglo
            StaticJsonDocument<256> json1; //Se crea el json del primer canal
            StaticJsonDocument<256> json2; //Se crea el json del segundo canal

            json1["channel"] = 0;
            json1["value"] = sensor1;
            json2["channel"] = 1;
            json2["value"] = sensor2;

            arreglo.add(json1); //Se añaden los json al arreglo
            arreglo.add(json2);

            serializeJson(arreglo, variable);
            int lon = variable.length()+1;
            Serial.println(variable);
            char datojson[lon];
            variable.toCharArray(datojson, lon);
            client.publish(inTopic,datojson);
            client.disconnect();
            delay(5000);
            // ... and resubscribe
            //client.subscribe("topic2");
        } else {

```


14. Seleccione otra plataforma hardware y realice el sketch para conectarse al bróker de Cayenne y publicar el json con los datos de los sensores, de la misma manera como se hizo con el Arduino Yun.

ENTREGA

Realiza un documento en donde se evidencie todo el procedimiento realizado.