INTRODUCTION TO IoT

Monsoon 2021

LAB-8

Introduction:

MQTT (MQ Telemetry Transport) is a Machine-to-Machine connectivity protocol. This is fancy jargon which basically means its a way for devices to communicate. It allows data to be **published** and **subscribed** to by clients through one central server on topic channels. The **server** can be as simple as a local computer or Raspberry Pi (using an IP address) or larger and public like mqtt.thingspeak.com.

In this experiment, you will be testing the water level in a container using a water sensing module(RKI-2350) and uploading it to the Thingspeak channel using MQTT.

Part-1) ESP32 Publish sensor data to Thingspeak using ESP32

a) Introduction to thingspeak:

The platform is primarily aimed towards IoT Projects and data analytics using visuals. To get started with the free services of Thingspeak you will first need to Signup using your email ID, once that is done along with the email verification you will be greeted with a similar-looking page as showed in fig1.

Setting ThingsSpeak:

i. First login to the Thingspeak server. https://thingspeak.com/login
ii. If you are a new user, then create the new account.
iii. After login you show this type of webpage, i have already created three projects, but if you are a new user then click on the New Channel.
iv. After clicking the new channel then give the name of the new channel .
v. If you want to give the description, then write the description. and select the Field 1 and don't forget the click on checkbox, after clicking on the check box give the field name.
vi. Finally scroll down and click on save.

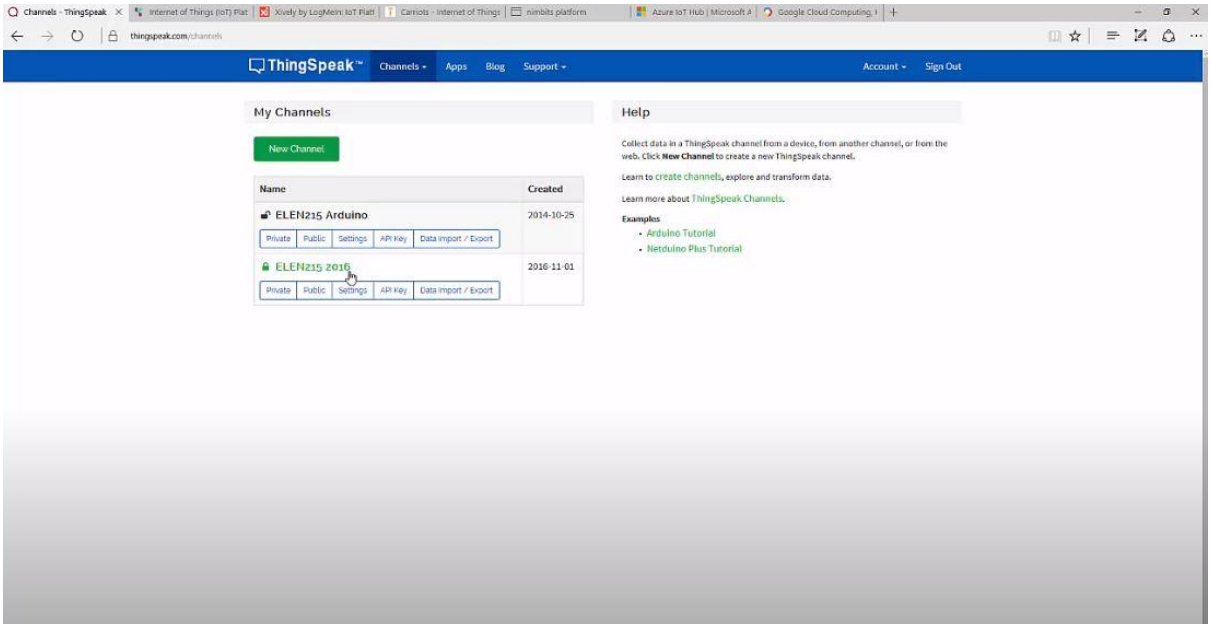

Figure 1

Now we will look at some of the terminologies:

- **Reading/Downloading Data:** Getting data on your ESP32 from the server is a read operation.

- **Writing/Uploading Data**: Sending data from your ESP8266/ESP32 to the server is a write operation.

- **API Key**: To have data security and to prevent anyone randomly from reading/writing data to your server there needs to be some sort of security/password and the API Key is something intended towards this. API Key is a long alphanumeric key which is needed to read/write data to the server. There are separate keys for reading and writing data. (Refer fig:2)

- **Channel**: A channel in thingspeak is a software counterpart of an IoT hardware device that you connect to Thingspeak, in our case an ESP32 will utilise one entire channel of our bandwidth. In a free account of thingspeak, you can have a maximum of 4 channels.

- **Field**: Each channel has 8 fields. A field is a variable and stores/shares a data type, for example when we send temperature and humidity from our device to the server, both the parameters will use one field each of the channel.

That's pretty much it about thingspeak!

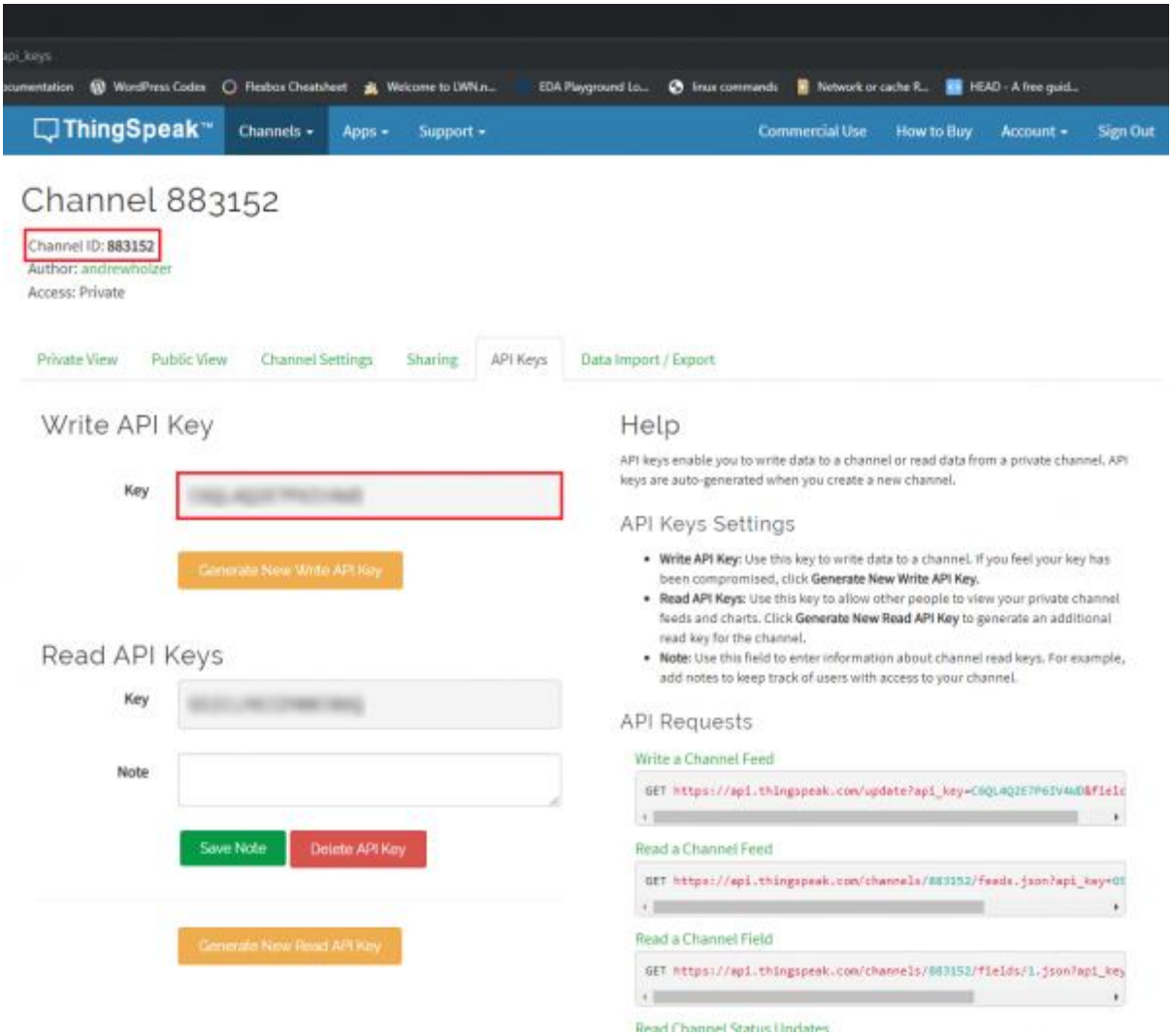

Figure 2

b) **Components required for this exercise:**

i) Water level sensor
ii) ESP32
iii) Breadboard
iv) Few jumper wires

c) Circuit connections:

i. Data pin of water level sensor is connected to any digital GPIO pin of the ESP32.
ii. The +VCC of water level sensor goes into +3.3V of the ESP32.
iii. GND of the water level sensor goes into Ground Pin (GND) of the ESP32.

d) Library Files:

Following 3 libraries will be required to run this code. Download the zip file, extract the same and copy this to your Arduino library folder.

i) Library 1: PubSubClient.h
ii) Library 2: WiFi.h
iii) Library 3: Thingspeak.h

e) Pseudocode:

i. Include the above-mentioned libraries.
ii. Create a char **ssid[]** = "_____" **, char password[] = "_____", const char* server = mqtt.thingspeak.com", char mqttUserName[] = "Water Level Sensing", char mqttPass[] = "MQTTPASS"** (Change to your MQTT API key from Account > MyProfile on Thingspeak.), **int writeChannelID = ChannelID ,char writeAPIKey[] = "apiKey"**
iii. Instantiate WifiClient object and a PubSubClient object as PubSubClient mqttClient(server, 1883, client)
iv. In the setup function:
a. Begin serial communication.
b. Define any digital pin as INPUT pin.
c. Begin wifi communication as: WiFi.begin(ssid,pass)
d. Wait till the wifi is connected by comparing WiFi.status() with WL_CONNECTED.
e. Set the MQTT broker details as: mqttClient.setServer (server, 1883)
v. In the loop function:
a. Check if the mqtt client is connected by checking if the value of mqttClient.connected() is not NULL. If not connected then call mqttConnect() function(already provided).
b. Call the loop to maintain the connection to the server with mqttClient.loop().
c. Read integer data from D1 using analogRead and store it in a variable sensorData.
d. Define an integer array fieldsToPublish[8] = {1,0,0,0,0,0,0,0}
e. Give a delay of 1 sec.
f. Print the value of sensorData on the serial monitor.
g. Call the mqttPublish() function as: mqttPublish(writeChannelID, writeAPIKey, dataToPublish, fieldsToPublish)

vi. In the mqttPublish function:
mqttPublish(long pubChannelID, char* pubWriteAPIKey, float dataArray[], int fieldArray[])
a. Define an integer variable, index=0 and an empty string dataString.
b. Instantiate a while loop checking if index by 1.
c. Print the dataString.
d. Create a topic string and publish data to ThingSpeak channel feed by defining a string topicString and appending "channels/" + String(pubChannelID) + "/publish/"+String(pubWriteAPIKey) to it.
e. Call the publish function of mqttClient as mqttClient.publish(topicString.c_str(), dataString.c_str()) .
f. Print the channel id of the channel to which the data is being published, i.e. print pubChannelID string.

f) Expected output:

The water level data should be plotted on the Thingspeak server in the Private View tab

Part-2) Control an LED using Thingspeak

a) Components required for this exercise:

i. Led
ii. ESP32
iii. Breadboard
iv. Few jumper wires

c) Circuit Diagram:

i. connect the LED with D1 pin.
ii. GND of the LED goes into Ground Pin (GND) of the ESP32.

d) Library Files:

i) Library 1: WiFi.h
ii) Library 2: Thingspeak.h
iii) Libraray3: Webserver.h

e) Pseudocode:

i. Include the above librararies
ii. Adjust the baud rate in the setup function Serial.begin().
iii. Define the pin at which the LED is connected to the ESP32
iv. Give your ssid and password. Then create the object of the **WiFiClient** class is client, this object is used to connect the device to the Thingspeak sever.
v. Make the LED pin as an OUTPUT pin and initially LOW the LED, then select the baudrate 115200 and and connect the ESP32 with wifi to the station mode. and print the local ip on the serial port. using **WiFi.localIP().**
vi. Now connect the client to the thingSpeak server using **ThingSpeak.begin(client).**
vii. Now read the value from thingspeak using ThingSpeak.readFloatField(channel, led1) to a variable
viii. In void loop() function, server.handleclient(); it is actually handle the client.
ix. ThingSpeak.readFloatField (channel_id , 1); this function return the int value where we pass on the HTML page link. here 1 mean Field number. upper we create the only field1.
x. if this function return the 1 then we on the led otherwise low.
xi. https://api.thingspeak.com/update?api_key=MOHD33LYGVXTG5UF&field1=0 this url is copy from the thingserver here api_key is different in your case last one &field1=0 mean we we press this url then send the 0 on your thingspeak server to the field1 and &field1=1 mean when we press this url then send the 1 on your thingspeak server to the field1. but we don't press the url using HTML code we only clink on the ON button to turn on the led and OFF button for turn off the led. (REFER FIG:3)

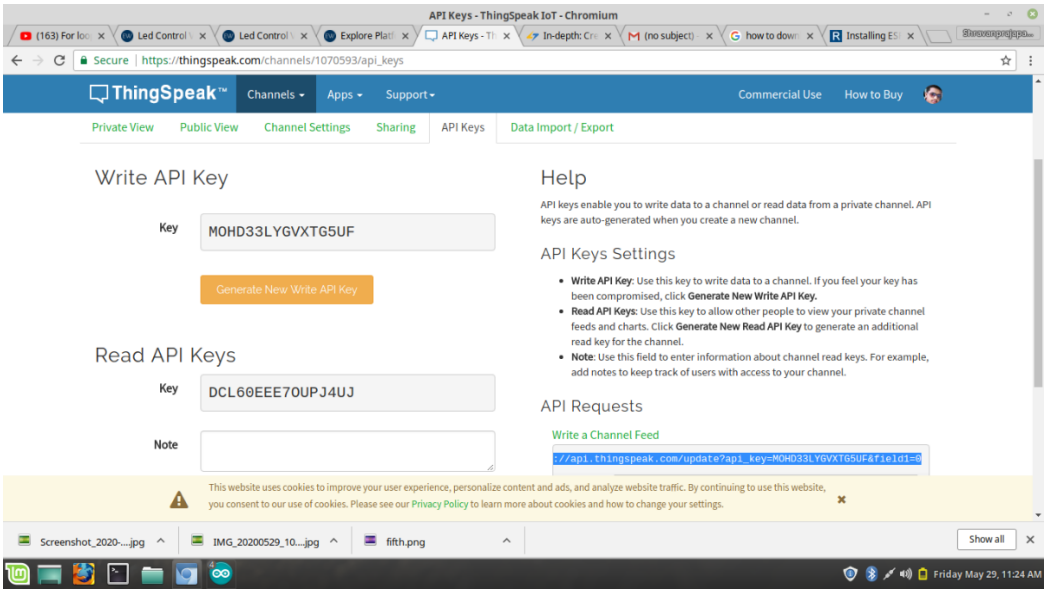

Figure 3

xii. Now go to the thingspeak server click on the API KEYS and copy the Write API KEY and paste the HTML code

xiii. The Html code looks something like this

```
String SendHTML(void){
 String ptr = "<!DOCTYPE html> <html>\n";
 ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
 ptr +="<title>LED Control</title>\n";
 ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
 ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";
 ptr +=".button {display: block;width: 80px;background-color: #1abc9c;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
 ptr +=".button-on {background-color: #1abc9c;}\n";
 ptr +=".button-on:active {background-color: #16a085;}\n";
 ptr +=".button-off {background-color: #34495e;}\n";
 ptr +=".button-off:active {background-color: #2c3e50;}\n";
 ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
 ptr +="</style>\n";
 ptr +="</head>\n";
 ptr +="<body>\n";
 ptr +="<h1>ESP32 with ThingSpeak Server</h1>\n";
 ptr +="<h3>Using Station(STA) Mode</h3>\n";
 ptr +="<h4>Control status For D1</h4>\n";
 ptr +="<a class=\"button button-on\" href=\"https://api.thingspeak.com/update?api_key=MOHD33LYGVXTG5UF&field1=1\">ON</a>\n";
```

```
 ptr +="<a class=\"button button-off\"
href=\"https://api.thingspeak.com/update?api_key=MOHD33LYGVXTG5UF&field1=0\">O
FF</a>\n";
 ptr +="</body>\n";
 ptr +="</html>\n";
 return ptr;
}
```

xiv. Finally, copy this IP on the Chrome browser.

f) Expected output:

The LED should be controlled using thingspeak.