**EMBEDDED SYSTEMS**

**(EL-419)**

**LABORATORY MANUAL**

**SPRING 2019**

****

**Engr. Aneela Sabir**

## Introduction to MQTT and device control over internet

**(LAB # 06)**

Student Name: \_\_\_Kamran,Aqib Khan,Moiz Hussain\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Section: \_\_A\_\_

Date performed: \_\_\_\_\_\_\_27/2\_\_\_\_\_\_, 2019

**MARKS AWARDED: \_\_\_\_\_\_\_\_/ 10**

## Lab # 06: Introduction to MQTT and device control over internet

**Learning Objectives:**

1. Introduction to MQTT server
2. Introduction to IFTTT Android Application
3. How to use Adafruit.io
4. Device control using internet
5. Using Google Assistant to control devices

#### **Equipment Required:**

1. NodeMCU ESP8266 Breakout Board
2. USB-A to micro-USB Cable
3. Jumper wires
4. LED
5. Bread board
6. Arduino IDE

**Adafruit Io:**

# **Overview**

Adafruit IO is a system that makes data useful. Our focus is on ease of use, and allowing simple data connections with little programming required.

# **Getting Started**

If you haven't already, log into your Adafruit account and then head over to [io.adafruit.com](https://io.adafruit.com/). There are two main fields that you would be using in the lab.

1. Feeds
2. Dashboard

**Feeds:**

Feeds are the core of the Adafruit IO system. The feed holds **metadata** about the data you push to Adafruit IO. This includes settings for whether the data is public or private, what license the stored sensor data falls under, and a general description of the data. The feed also contains the sensor **data**values that get pushed to Adafruit IO from your device.

You will need to create one feed for each unique source of data you send to the system. For example, if you have a project with one temperature sensor and two humidity sensors, you would need to create three feeds. One feed for the temperature sensor, and one feed for each humidity sensor. First, let's take a look at how to create a new feed using Adafruit IO.

[**https://learn.adafruit.com/adafruit-io-basics-feeds/creating-a-feed**](https://learn.adafruit.com/adafruit-io-basics-feeds/creating-a-feed)

**Dashboard:**

Dashboards allow you to visualize data and control Adafruit IO connected projects from any modern web browser. Widgets such as charts, sliders, and buttons are available to help you quickly get your IoT project up and running without the need for any custom code.

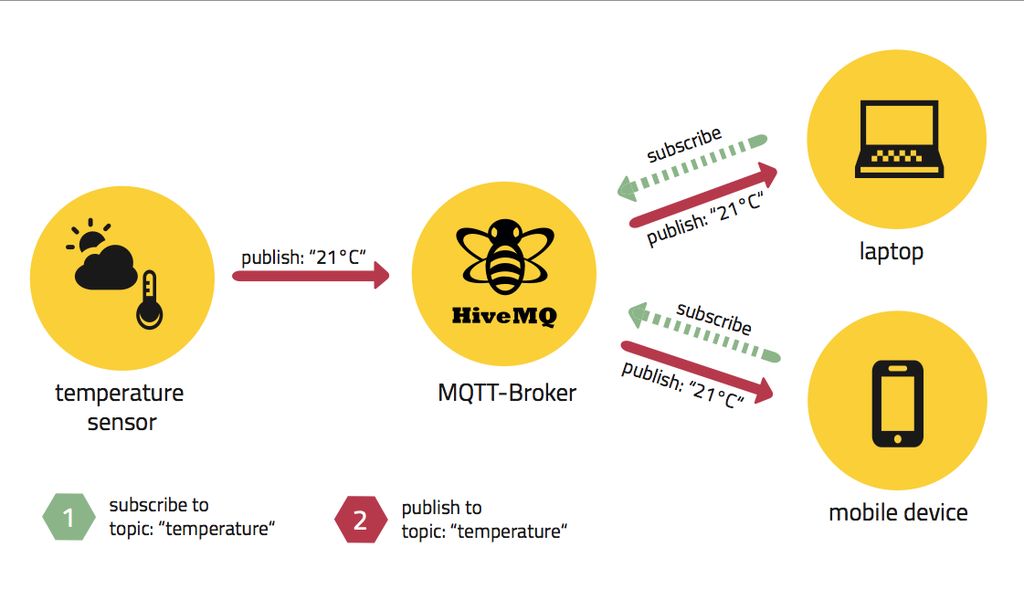
**Creating a dashboard:**

[**https://learn.adafruit.com/adafruit-io-basics-dashboards/creating-a-dashboard**](https://learn.adafruit.com/adafruit-io-basics-dashboards/creating-a-dashboard)

**MQTT**

**Introduction:**

MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.

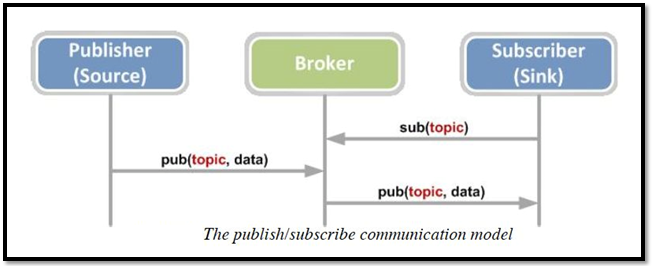


MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that provides resource-constrained network clients with a simple way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used for machine-to-machine (M2M) communication and plays an important role in the Internet of Things (IoT).

History :

MQTT was developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech; now Cirrus Link) in 1999 for the monitoring of an oil pipeline through the desert. The goals were to have a protocol, which is bandwidth-efficient and uses little battery power, because the devices were connected via satellite link and this was extremely expensive at that time.

The protocol uses a publish/subscribe architecture in contrast to HTTP with its request/response paradigm. Publish/Subscribe is event-driven and enables messages to be pushed to clients. The central communication point is the MQTT broker, it is in charge of dispatching all messages between the senders and the rightful receivers. Each client that publishes a message to the broker, includes a topic into the message. The topic is the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client. Therefore the clients don’t have to know each other, they only communicate over the topic. This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.



**Why MQTT?**

1. It is a publish/subscribe protocol

2. It has Multiple Quality of Service levels (QOS)

3. It has at-least-once and exactly-once semantics

4. It has a low overhead (2 bytes at minimum)

5. It supports offline messaging

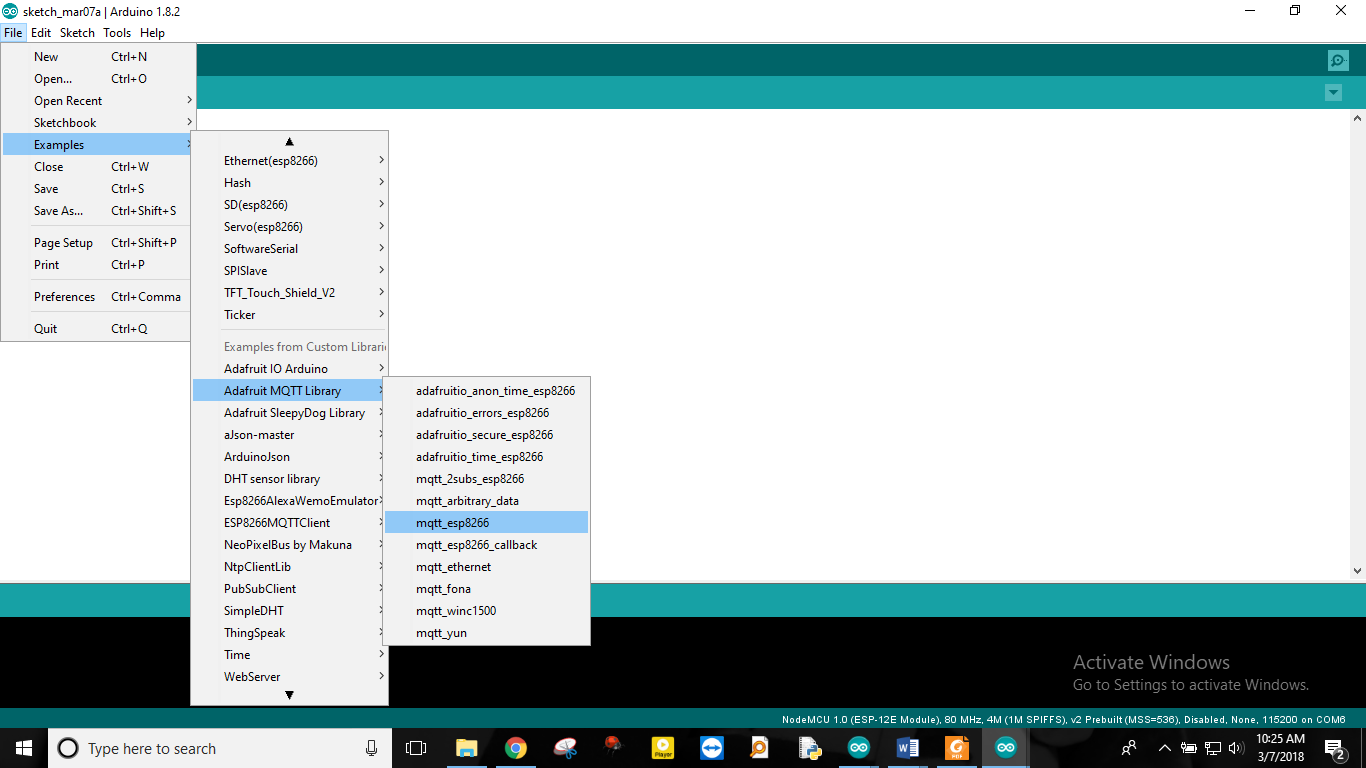
6. It retains messages, like a key/value store

**Publish / Subscribe:**

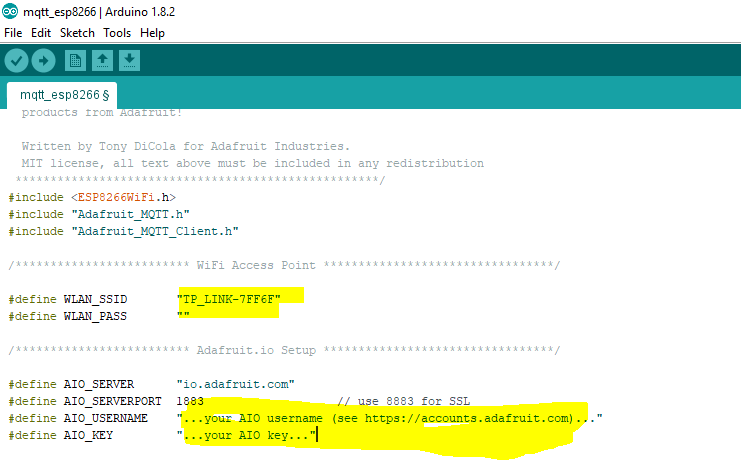
The publish / subscribe (often called pub-sub) pattern lies at the heart of MQTT. It’s based around a message broker, with other nodes arranged around the broker in a star topology. This is a very different model to the standard client/server approach, and at first it might seem a little strange, but the decoupling it provides is a huge advantage in many situations.

Clients can publish or subscribe to particular topics which are somewhat like message subjects. They are used by the broker to decide who will receive a message. Topics in MQTT have a particular syntax. They are arranged in a hierarchy using the slash character (/) as a separator, much like the path in a URL. So a temperature sensor in your kitchen might publish to a topic like ‘sensors/temperature/home/kitchen’.

**Using MQTT Library in Arduino IDE:**



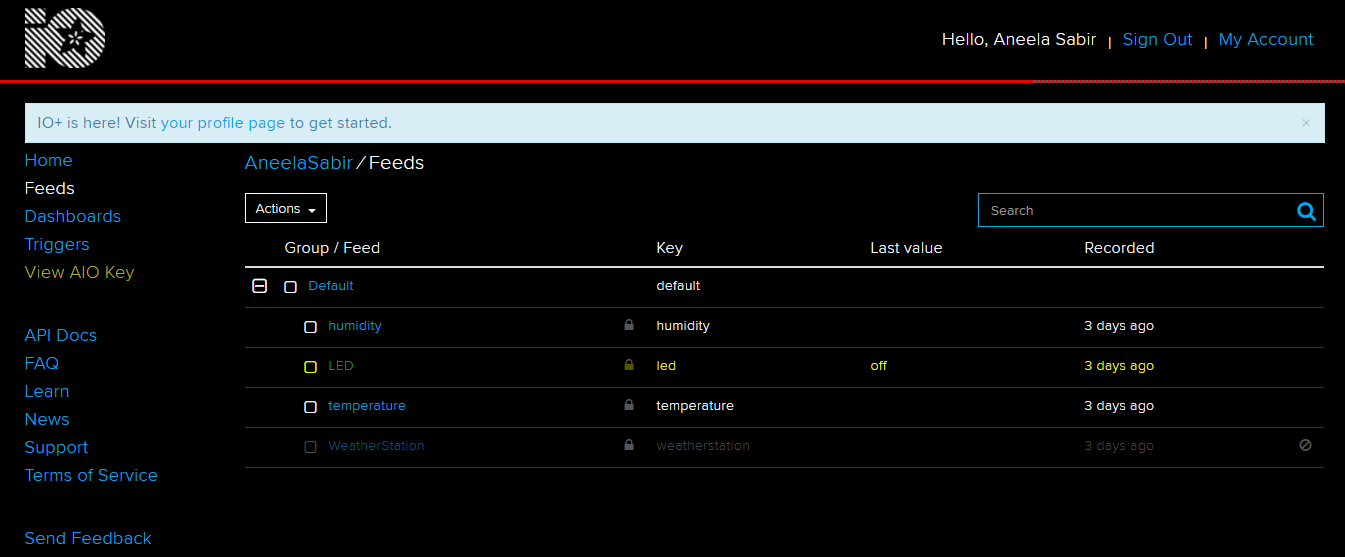
You need to modify highlited things in your code:



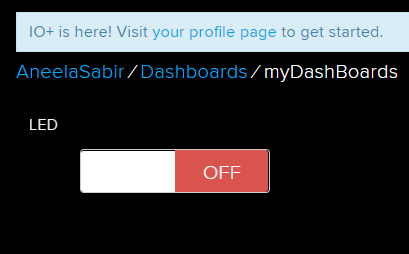


## ****Testing the code****

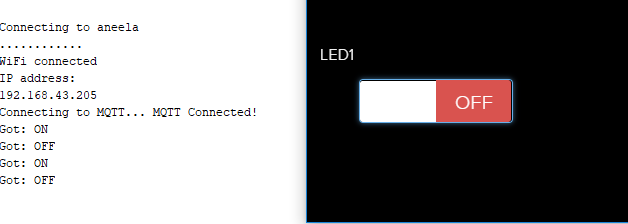
1. Create a Field in Adafruit io and name it LED.



1. Add a switch in your dashboard under LED Field:

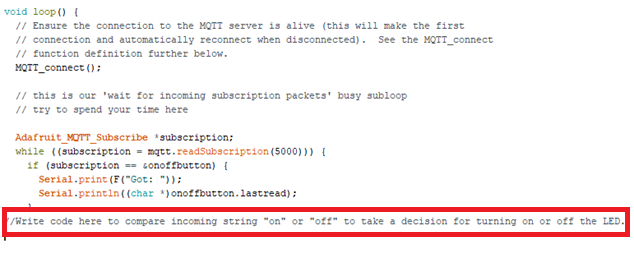


1. Check serial monitor and change the status of LED in your Adafuit io dashboard.



**Task 1:**

Modify example code to turn on and off Built in LED of NodeMCU ESP8266 module using Adafruit io.



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Adafruit MQTT Library ESP8266 Example

Must use ESP8266 Arduino from:

https://github.com/esp8266/Arduino

Works great with Adafruit's Huzzah ESP board & Feather

----> https://www.adafruit.com/product/2471

----> https://www.adafruit.com/products/2821

Adafruit invests time and resources providing this open source code,

please support Adafruit and open-source hardware by purchasing

products from Adafruit!

Written by Tony DiCola for Adafruit Industries.

MIT license, all text above must be included in any redistribution

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WiFi Access Point \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define WLAN\_SSID "moghees"

#define WLAN\_PASS "gct\*2016"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883 // use 8883 for SSL

#define AIO\_USERNAME "moiz23"

#define AIO\_KEY "5da51ad995d54d668c7ee9b85c61ca45"

/\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// or... use WiFiFlientSecure for SSL

//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Feeds \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Setup a feed called 'photocell' for publishing.

// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

Adafruit\_MQTT\_Publish photocell = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/photocell");

// Setup a feed called 'onoff' for subscribing to changes.

Adafruit\_MQTT\_Subscribe onoffbutton = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/lab6");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sketch Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT\_connect();

void setup() {

Serial.begin(115200);

delay(10);

Serial.println(F("Adafruit MQTT demo"));

pinMode(LED\_BUILTIN, OUTPUT);

// Connect to WiFi access point.

Serial.println(); Serial.println();

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Setup MQTT subscription for onoff feed.

mqtt.subscribe(&onoffbutton);

}

uint32\_t x=0;

void loop() {

// Ensure the connection to the MQTT server is alive (this will make the first

// connection and automatically reconnect when disconnected). See the MQTT\_connect

// function definition further below.

MQTT\_connect();

// this is our 'wait for incoming subscription packets' busy subloop

// try to spend your time here

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(5000))) {

if (subscription == &onoffbutton) {

Serial.print(F("Got: "));

Serial.println((char \*)onoffbutton.lastread);

String value=String((char \*)onoffbutton.lastread);

if(value=="ON"){digitalWrite(LED\_BUILTIN, LOW);}

if(value=="OFF"){digitalWrite(LED\_BUILTIN, HIGH);}

}

}

// Now we can publish stuff!

Serial.print(F("\nSending photocell val "));

Serial.print(x);

Serial.print("...");

if (! photocell.publish(x++)) {

Serial.println(F("Failed"));

} else {

Serial.println(F("OK!"));

}

// ping the server to keep the mqtt connection alive

// NOT required if you are publishing once every KEEPALIVE seconds

/\*

if(! mqtt.ping()) {

mqtt.disconnect();

}

\*/

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT\_connect() {

int8\_t ret;

// Stop if already connected.

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0) {

// basically die and wait for WDT to reset me

while (1);

}

}

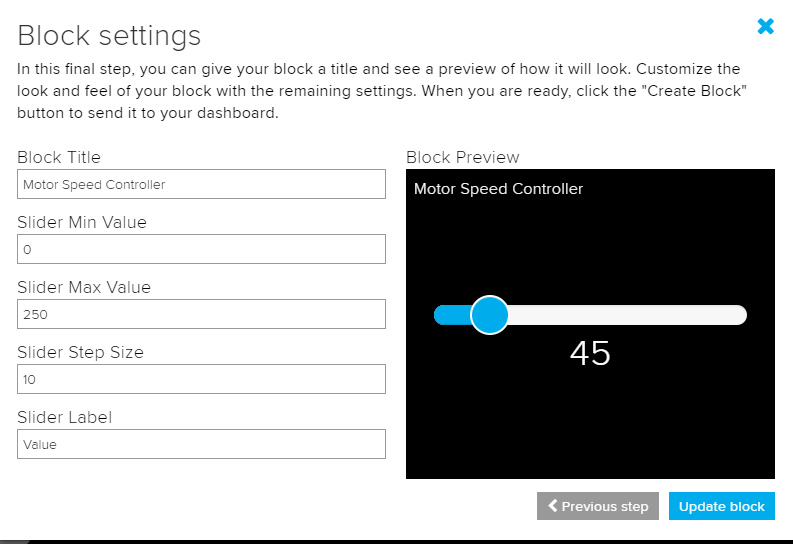
Serial.println("MQTT Connected!");

}

**Task 2:**

**2.1** Create another feed into your Adafruit io and name it Motor Speed Controller.

**2.2** Add a slider in dashboard as shown in fig. below. You need to adjust the brightness of LED using the slider movement.



**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Adafruit MQTT Library ESP8266 Example

Must use ESP8266 Arduino from:

https://github.com/esp8266/Arduino

Works great with Adafruit's Huzzah ESP board & Feather

----> https://www.adafruit.com/product/2471

----> https://www.adafruit.com/products/2821

Adafruit invests time and resources providing this open source code,

please support Adafruit and open-source hardware by purchasing

products from Adafruit!

Written by Tony DiCola for Adafruit Industries.

MIT license, all text above must be included in any redistribution

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WiFi Access Point \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define WLAN\_SSID "moghees"

#define WLAN\_PASS "gct\*2016"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Adafruit.io Setup \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883 // use 8883 for SSL

#define AIO\_USERNAME "moiz23"

#define AIO\_KEY "5da51ad995d54d668c7ee9b85c61ca45"

/\*\*\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// or... use WiFiFlientSecure for SSL

//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Feeds \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Setup a feed called 'photocell' for publishing.

// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

Adafruit\_MQTT\_Publish photocell = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/photocell");

// Setup a feed called 'onoff' for subscribing to changes.

Adafruit\_MQTT\_Subscribe onoffbutton = Adafruit\_MQTT\_Subscribe(&mqtt, AIO\_USERNAME "/feeds/pwmmotor");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sketch Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT\_connect();

void setup() {

Serial.begin(115200);

delay(10);

Serial.println(F("Adafruit MQTT demo"));

pinMode(LED\_BUILTIN, OUTPUT);

// Connect to WiFi access point.

Serial.println(); Serial.println();

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Setup MQTT subscription for onoff feed.

mqtt.subscribe(&onoffbutton);

}

uint32\_t x=0;

void loop() {

// Ensure the connection to the MQTT server is alive (this will make the first

// connection and automatically reconnect when disconnected). See the MQTT\_connect

// function definition further below.

MQTT\_connect();

// this is our 'wait for incoming subscription packets' busy subloop

// try to spend your time here

Adafruit\_MQTT\_Subscribe \*subscription;

while ((subscription = mqtt.readSubscription(5000))) {

if (subscription == &onoffbutton) {

Serial.print(F("Got: "));

Serial.println((char \*)onoffbutton.lastread);

String value=String((char \*)onoffbutton.lastread);

int pwmVal = value.toInt();

analogWrite(LED\_BUILTIN, pwmVal);

}

}

// Now we can publish stuff!

Serial.print(F("\nSending photocell val "));

Serial.print(x);

Serial.print("...");

if (! photocell.publish(x++)) {

Serial.println(F("Failed"));

} else {

Serial.println(F("OK!"));

}

// ping the server to keep the mqtt connection alive

// NOT required if you are publishing once every KEEPALIVE seconds

/\*

if(! mqtt.ping()) {

mqtt.disconnect();

}

\*/

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT\_connect() {

int8\_t ret;

// Stop if already connected.

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0) {

// basically die and wait for WDT to reset me

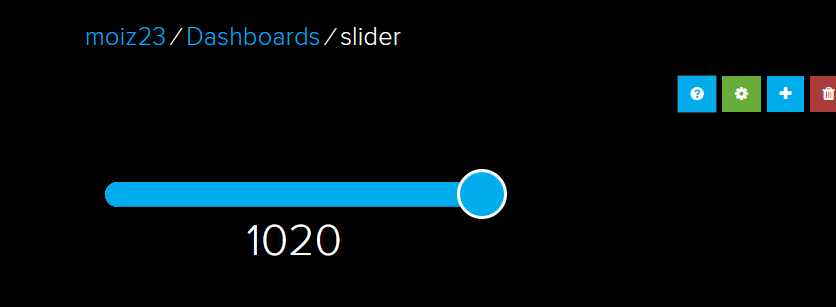
while (1);

}

}

Serial.println("MQTT Connected!");

}



**IFTTT Android Application:**

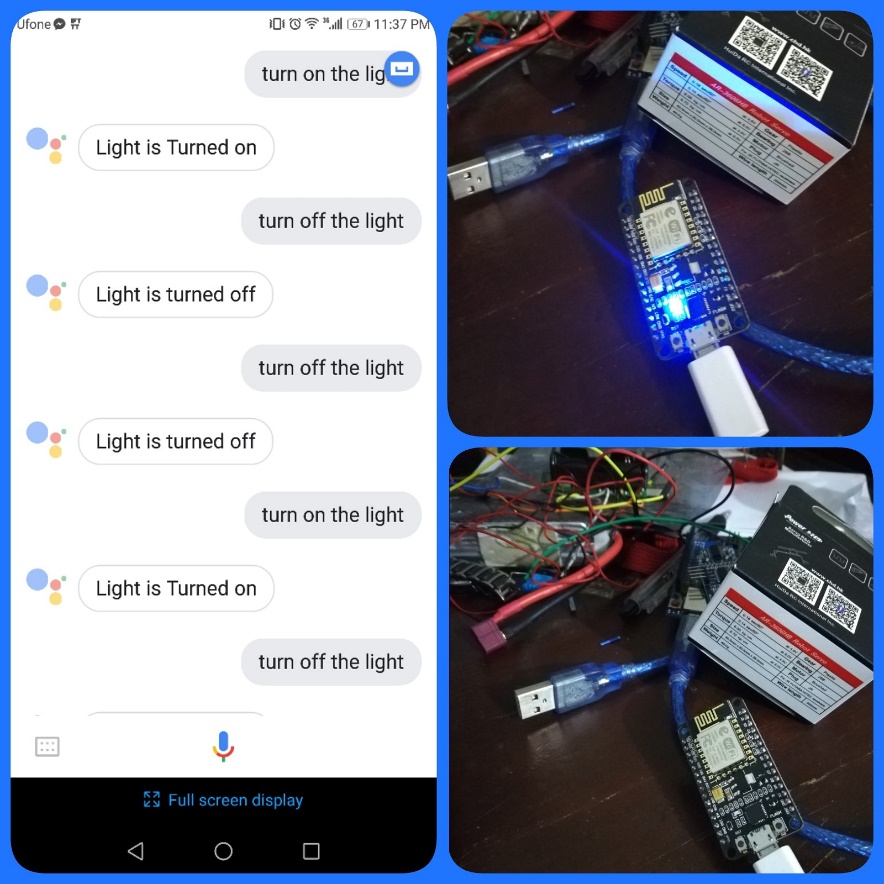
**If This Then That**, also known as **IFTTT**, is a free [web](https://en.wikipedia.org/wiki/World_Wide_Web)-based service to create chains of simple [conditional statements](https://en.wikipedia.org/wiki/Conditional_(computer_programming)), called applets. An applet is triggered by changes that occur within other web services such as [Gmail](https://en.wikipedia.org/wiki/Gmail), [Facebook](https://en.wikipedia.org/wiki/Facebook), [Telegram](https://en.wikipedia.org/wiki/Telegram_(messaging_service)), [Instagram](https://en.wikipedia.org/wiki/Instagram), or [Pinterest](https://en.wikipedia.org/wiki/Pinterest). For example, an applet may send an e-mail message if the user [tweets](https://en.wikipedia.org/wiki/Twitter) using a hashtag, or copy a photo on Facebook to a user's archive if someone tags a user in a photo.

**Using IFTTT Application to Control devices using Google Assistant:**

1. Sign up to IFTTT app.
2. Create a new applet to turn on an LED.

For creating new applet, you can see the video uploaded on SLATE.

1. Create sencond applet for turning off the LED.
2. Upload Task 1 code into your NodeMCU ESP8266 module.
3. Use google Assistant to test the code. Your response may look like as shown in figure below:

****

**Task 3:** Use Google Assistant to turn on/off the LED connected with your NodeMCU ESP8266 module.

**References:**

http://www.instructables.com/id/Introduction-to-MQTT/

<http://www.instructables.com/id/Control-Your-Projects-With-Google-Assistant-and-Ad/>