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

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"

#define relay 16

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

#define WLAN\_SSID "Home"

#define WLAN\_PASS "quietsquash890"

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

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

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

#define AIO\_USERNAME "commproject"

#define AIO\_KEY "7ec9f408f631436a90bd6004dcb51174"

/\*\*\*\*\*\*\*\*\*\*\*\* 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/Light");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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);

pinMode(relay,OUTPUT);

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

// 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);

uint16\_t state = atoi((char \*)onoffbutton.lastread);

digitalWrite(relay,state);

}

}

// 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!");

}