```
#include <ESP8266WiFi.h>
#include <DallasTemperature.h>call
#include <OneWire.h>
#include "DHT.h"
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <ArduinoJson.h>
const char *ssid =  "Galaxy-M20";     // Enter your WiFi Name
const char *pass =  "ac312124"; // Enter your WiFi Password
WiFiClient client;
#define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "aschoudhary" // Your Adafruit IO Username
#define MQTT_PASS "1ac95cb8580b4271bbb6d9f75d0668f1" // Adafruit IO AIO key
const char server[] = "api.openweathermap.org";
```

```
String nameOfCity = "Jaipur,IN";

String apiKey = "e8b22b36da932dce8f31ec9be9cb68a3";

String text;

const char* icon="";

int jsonend = 0;

boolean startJson = false;

int status = WL_IDLE_STATUS;

#define JSON_BUFF_DIMENSION 2500

unsigned long lastConnectionTime = 10 * 60 * 1000;     // last time you connected to the server, in milliseconds

const unsigned long postInterval = 10 * 60 * 1000;  // posting interval of 10 minutes  (10L * 1000L; 10 seconds delay for testing)

const int ldrPin = D1;

const int ledPin = D0;

const int moisturePin = A0;  // moisteure sensor pin
const int motorPin = D8;

float moisturePercentage;             //moisture reading

int temperature, humidity, soiltemp;

#define ONE_WIRE_BUS 4   //D2 pin of nodemcu

#define DHTTYPE DHT11   // DHT 11

#define dht_dpin D4

DHT dht(dht_dpin, DHTTYPE);

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

const unsigned long Interval = 50000;

unsigned long previousTime = 0;

//Set up the feed you're publishing to

Adafruit_MQTT_Client mqtt(&client, MQTT_SERV, MQTT_PORT, MQTT_NAME, MQTT_PASS);

Adafruit_MQTT_Publish Moisture = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/Moisture");  // Moisture is the feed name where you will publish your data
Adafruit_MQTT_Publish Temperature = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/Temperature");

Adafruit_MQTT_Publish Humidity = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/Humidity");

Adafruit_MQTT_Publish SoilTemp = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/SoilTemp");

Adafruit_MQTT_Publish WeatherData = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME "/f/WeatherData");

//Set up the feed you're subscribing to

 Adafruit_MQTT_Subscribe LED = Adafruit_MQTT_Subscribe(&mqtt, MQTT_NAME "/f/LED");

 Adafruit_MQTT_Subscribe Pump = Adafruit_MQTT_Subscribe(&mqtt, MQTT_NAME "/f/Pump");

void setup()

{

  Serial.begin(9600);

  delay(10);

  dht.begin();

  sensors.begin();

  mqtt.subscribe(&LED);
  mqtt.subscribe(&Pump);

  pinMode(motorPin, OUTPUT);

  pinMode(ledPin, OUTPUT);

  pinMode(ldrPin, INPUT);

  digitalWrite(motorPin, LOW); // keep motor off initally

  digitalWrite(ledPin, HIGH);

  text.reserve(JSON_BUFF_DIMENSION);

  Serial.println("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)

  {

    delay(500);

    Serial.print(".");              // print ... till not connected
  }

  Serial.println("");

  Serial.println("WiFi connected");

}

void loop()

{

 unsigned long currentTime = millis();

 MQTT_connect();

 if (millis() - lastConnectionTime > postInterval) {

    // note the time that the connection was made:

    lastConnectionTime = millis();

    makehttpRequest();

  }

//}

 int ldrStatus = analogRead(ldrPin);
```

```
  if (ldrStatus <= 200) {
     digitalWrite(ledPin, HIGH);
     Serial.print("Its DARK, Turn on the LED : ");
     Serial.println(ldrStatus);
  }
  else {
    digitalWrite(ledPin, LOW);
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);
   }
moisturePercentage = ( 100.00 - ( (analogRead(moisturePin) / 1023.00) * 100.00 ) );
Serial.print("Soil Moisture is  = ");
Serial.print(moisturePercentage);
Serial.println("%");
```

```
if (moisturePercentage < 35) {

  digitalWrite(motorPin, HIGH);         // tun on motor

}

if (moisturePercentage > 38) {

  digitalWrite(motorPin, LOW);          // turn off mottor

}

 temperature = dht.readTemperature();

 humidity = dht.readHumidity();

 //Serial.print("Temperature: ");

 //Serial.print(temperature);

 //Serial.println();

 //Serial.print("Humidity: ");

 //Serial.print(humidity);

 //Serial.println();

 sensors.requestTemperatures();
 soiltemp = sensors.getTempCByIndex(0);

// Serial.println("Soil Temperature: ");

// Serial.println(soiltemp);

if (currentTime - previousTime >= Interval) {

   if (! Moisture.publish(moisturePercentage)) //This condition is used to publish the Variable
(moisturePercentage) on adafruit IO. Change thevariable according to yours.

       {

         }

   if (! Temperature.publish(temperature))

       {

        }

   if (! Humidity.publish(humidity))

       {

       //delay(30000);

        }
   if (! SoilTemp.publish(soiltemp))

     {

        }

   if (! WeatherData.publish(icon))

     {

       }

        previousTime = currentTime;

}

Adafruit_MQTT_Subscribe * subscription;

while ((subscription = mqtt.readSubscription(5000))) //Dont use this one until you are conrolling
something or getting data from Adafruit IO.

    {

   if (subscription == &LED)

    {

     //Print the new value to the serial monitor
```

```
  Serial.println((char*) LED.lastread);
    if (!strcmp((char*) LED.lastread, "OFF"))
   {
     digitalWrite(ledPin, LOW);
    }
    if (!strcmp((char*) LED.lastread, "ON"))
      {
     digitalWrite(ledPin, HIGH);
     }
 }
if (subscription == &Pump)
  {
  //Print the new value to the serial monitor
  Serial.println((char*) Pump.lastread);
```

```
      if (!strcmp((char*) Pump.lastread, "OFF"))
       {
        digitalWrite(motorPin, HIGH);
       }
     if (!strcmp((char*) Pump.lastread, "ON"))
       {
        digitalWrite(motorPin, LOW);
       }
     }
   }
  delay(9000);
 // client.publish(WeatherData, icon)
}
void MQTT_connect()
```

```
{
  int8_t ret;
  // Stop if already connected.
  if (mqtt.connected())
  {
    return;
  }
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) // connect will return 0 for connected
  {
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0)
```

```
      {
        // basically die and wait for WDT to reset me
        while (1);
      }
  }
}
void makehttpRequest() {
  // close any connection before send a new request to allow client make connection to server
  client.stop();
  // if there's a successful connection:
  if (client.connect(server, 80)) {
    client.println("GET /data/2.5/forecast?q=" + nameOfCity + "&APPID=" + apiKey +
"&mode=json&units=metric&cnt=2 HTTP/1.1");
    client.println("Host: api.openweathermap.org");
    client.println("User-Agent: ArduinoWiFi/1.1");
    client.println("Connection: close");
    client.println();
    unsigned long timeout = millis();
    while (client.available() == 0) {
     if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
     }
    }
    char c = 0;
    while (client.available()) {
     c = client.read();
     // since json contains equal number of open and close curly brackets, this means we can determine
when a json is completely received  by counting
```

```
// the open and close occurences,

//Serial.print(c);

if (c == '{') {

  startJson = true;        // set startJson true to indicate json message has started

  jsonend++;

}

if (c == '}') {

  jsonend--;

}

if (startJson == true) {

  text += c;

}

// if jsonend = 0 then we have have received equal number of curly braces

if (jsonend == 0 && startJson == true) {

  parseJson(text.c_str());  // parse c string text in parseJson function
```

```
    text = "";                // clear text string for the next time
    startJson = false;        // set startJson to false to indicate that a new message has not yet started
   }
  }
 }
 else {
  // if no connction was made:
  Serial.println("connection failed");
  return;
 }
}
//to parse json data recieved from OWM
void parseJson(const char * jsonString) {
 //StaticJsonBuffer<4000> jsonBuffer;
```

```
  const size_t bufferSize = 2*JSON_ARRAY_SIZE(1) + JSON_ARRAY_SIZE(2) + 4*JSON_OBJECT_SIZE(1) + 3*JSON_OBJECT_SIZE(2) + 3*JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) + 2*JSON_OBJECT_SIZE(7) + 2*JSON_OBJECT_SIZE(8) + 720;

  DynamicJsonBuffer jsonBuffer(bufferSize);

//  DynamicJsonDocument(bufferSize);

  // FIND FIELDS IN JSON TREE

  JsonObject& root = jsonBuffer.parseObject(jsonString);

  if (!root.success()) {

    Serial.println("parseObject() failed");

    return;

  }

  JsonArray& list = root["list"];

  JsonObject& nowT = list[0];

  JsonObject& later = list[1];

  JsonObject& tommorow = list[2];

//  String conditions = list.weather.main;
  // including temperature and humidity for those who may wish to hack it in

  String city = root["city"]["name"];

  String weatherNow = nowT["weather"][0]["description"];

  String weatherLater = later["weather"][0]["description"];

  String list12 = later["weather"][0]["list"];

  Serial.println(list12);

  Serial.println(weatherLater);

  if(weatherLater == "few clouds"){

    icon = "Few Clouds";

    Serial.print(icon);

  }

  else if(weatherLater == "rain"){

    icon = "Rain";

    Serial.print(icon);
  }

  else if(weatherLater == "broken clouds"){

    icon = "Broken Clouds";

    Serial.print(icon);

  }

  else {

    icon = "Sunny";

    }

}
```