SPRINT 2

CONNECTING TO IBM CLOUD FROM ESP32 MICROCONTROLLER AND UPDATING DATA TO IT PERIODICALLY

Date	4th November 2022
Team ID	PNT2022TMID12810
Project Name	SmartFarmer – IoT Enabled Farming Application
Submitted by	Tharun G, Tamilselvan S

WOKWI PROJECT LINK: https://wokwi.com/projects/348487506111496786

PROGRAM (ESP32 CODE):

The following program uploaded in ESP32 microcontroller gets the data (temperature and humidity) from DHT22 sensor. Also, it gets connected to IBM cloud through MQTT protocol and updates the data periodically. It follows publish subscribe communication model.

```
// Importing the necessary libraries
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <WiFi.h>
#include <ESP32Servo.h>
#include "PubSubClient.h"
#define L LOW
#define H HIGH
#define DHTPIN 4
#define DHTTYPE DHT22
#define servoPin 2
#define violetPin 25
// Since motor is not available in wokwi platform,
// violet led is used instead of motor. However the connections
// remains same for motor if used.

// Cloud Credentials
```

```
#define ORG "c54g8s"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "24_0A_C4_00_01_10"
#define TOKEN "projectDONEbyTharunTeam12345"

char server[] = ORG".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/test/fmt/String";
char AUTH[] = "use-token-auth";
char token[] = TOKEN;
char CLIENTID[] = "d:"ORG":"DEVICE_TYPE":"DEVICE_ID;

DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;

// Temperature and humidity variables
float temperature = 0;
float humidity = 0;

// SSID and Password for WiFi connection
char SSID[] = "Wokwi-GUEST";
char PASSWORD[] = "";

Servo servo;
// angle position of servo motor
int deg = 0;

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);
```

```
// Connecting to Cloud through MQTT Protocol
void mqttConnect()
{
  if (!client.connected())
  {
    Serial.print("Client trying to reconnect to ");
    Serial.println(server);
    while (!client.connect( CLIENTID, AUTH, TOKEN))
    {
      Serial.print(".");
      delay(500);
    }
    Serial.println();
    if (client.subscribe(topic))
    {
      Serial.println("Subscription Success!");
    }
    else
    {
      Serial.println("Subscription Failed!");
    }
  }
}

// Setup function - run only once
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID, PASSWORD);
  pinMode(violetPin, OUTPUT);
```

```
digitalWrite(violetPin, LOW);

// Connecting to WiFi
Serial.print("Trying to connect to WiFi.");
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println();

Serial.print("Connected to ");
Serial.print(SSID);
Serial.print("(IP Address: ");
Serial.print(WiFi.localIP());
Serial.println(")");

Serial.print("MAC Address: ");
Serial.println(WiFi.macAddress());

// DHT22 sensor and Servo motor configuration
dht.begin();
servo.attach(servoPin, 500, 2400);
sensor_t sensor;
dht.temperature().getSensor(&sensor);
Serial.println("------------------------------------");
Serial.print("Temperature Sensor - Resolution : ");
Serial.print(sensor.resolution);
Serial.println("°C");
Serial.println("------------------------------------");
dht.humidity().getSensor(&sensor);
```

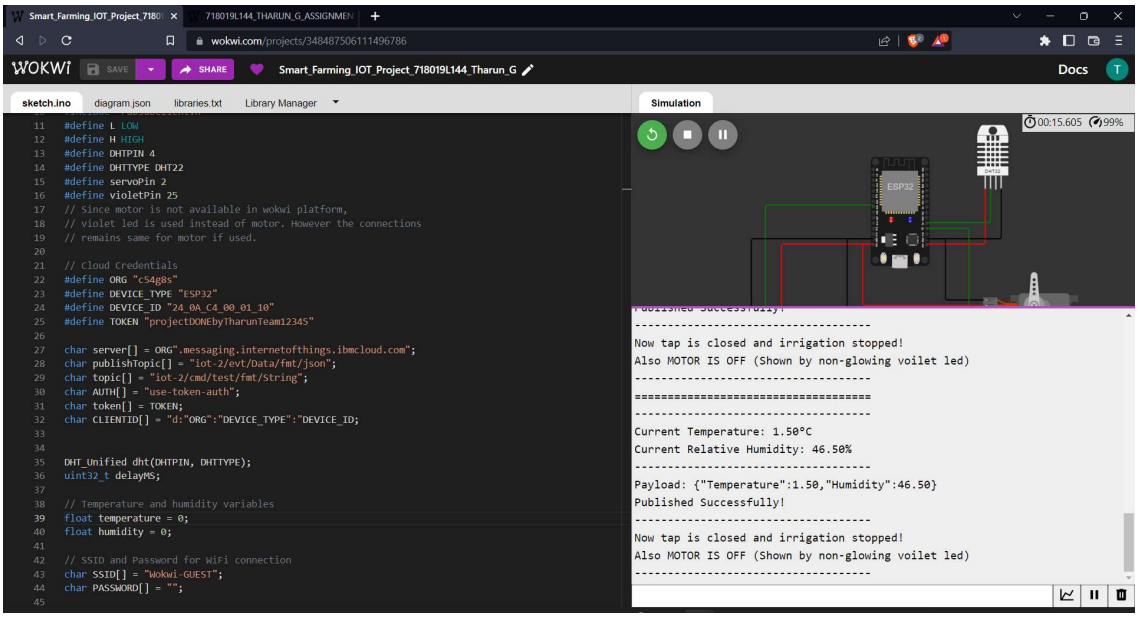
```
  Serial.print("Humidity Sensor - Resolution : ");
  Serial.print(sensor.resolution);
  Serial.println("%");
  Serial.println("------------------------------------");
  delayMS = sensor.min_delay / 1000;

}

// Loop function - run continuously
void loop() {

  // Getting temperature and humidity values at the moment
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  Serial.println("====================================");
  Serial.println("------------------------------------");
  if (isnan(event.temperature))
  {
    temperature = 0;
    Serial.println("Got error while reading temperature!");
  }
  else
  {
    temperature = event.temperature;
    Serial.print("Current Temperature: ");
    Serial.print(event.temperature);
    Serial.println("°C");
  }
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity))
```

```
{
  humidity = 0;
   Serial.println("Got error while reading humidity!");
}
else
{
  humidity = event.relative_humidity;
  Serial.print("Current Relative Humidity: ");
  Serial.print(event.relative_humidity);
  Serial.println("%");
}

Serial.println("-----------------------------------");

if (temperature>0 && humidity>0)
{
  String payload = "{\"Temperature\":";
  payload += String(temperature,2);
  payload += ",\"Humidity\":";
  payload += String(humidity,2);
  payload += "}";
  Serial.print("Payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char *) payload.c_str()))
  {
    Serial.println("Published Successfully!");
  }
  else
  {
    Serial.println("Publish Failed!");
```

```
  }
}
Serial.println("-----------------------------------");

// Controlling tap and motor based on certain conditions
if ( ((temperature < 27)||(temperature == 0)) && ((humidity > 30)||(humidity == 0)) )
{
  digitalWrite(violetPin, LOW);
  Serial.println("Now tap is closed and irrigation stopped!");
  Serial.println("Also MOTOR IS OFF (Shown by non-glowing voilet led)");
  for (; deg >= 0; deg -= 1)
  {
    servo.write(deg);
    delay(15);
  }
}
else
{
  digitalWrite(violetPin, HIGH);
  Serial.println("Now tap is open and irrigation occurs!");
  Serial.println("Also MOTOR IS ON (Shown by glowing voilet led)");
  for (; deg <= 90; deg += 1)
  {
    servo.write(deg);
    delay(15);
  }
}
Serial.println("-----------------------------------");
if (!client.loop())
{
```

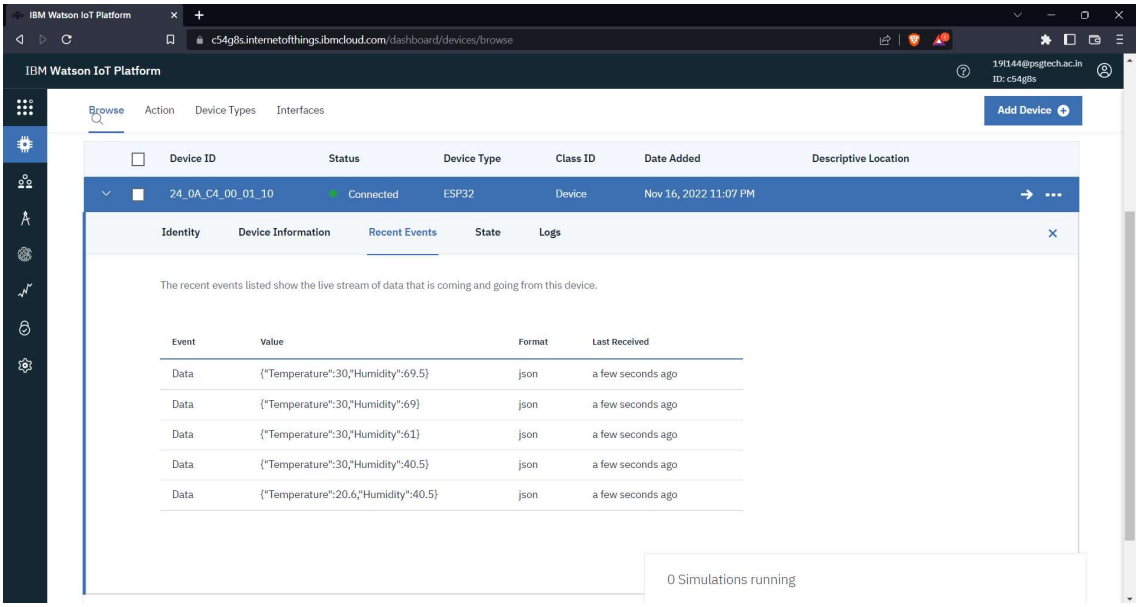
```
    mqttConnect();
  }
  delay(delayMS + 1000);
}
```

OUTPUT (SERIAL MONITOR):



From the above screenshot, we can infer that temperature and humidity values are published to IBM cloud periodically and if ESP32 gets disconnected, it tries again to connect to cloud using the provided credential such token, auth-method, device id, device type etc…

OUTPUT (IN IBM WATSON IOT PLATFORM – CLOUD):



From the above screenshot, we can infer that ESP32 is connected to IBM Cloud (Status: **Connected**) and the updated sensor values are observed in **Recent Events** in IBM WATSON IOT PLATFORM.