SMART FARMER -

IOT ENABLED SMART FARMING APPLICATION

FINAL CODE

```
#include <Wire.h>            //Includes the library for connections
#include <ESP32Servo.h>      //Includes the library for Servo motor
#include <LiquidCrystal_I2C.h> //Includes the library for LED
#include <DHTesp.h>          //Includes the library for DHT22 sensor

// WiFi libraries:
#include <WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>

#define ORG "oqy2ad" // Organization ID of IBM Cloud
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "NodeMCU"
#define TOKEN "123456789"

// Publishing Event in Watson IOT platform:
char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; //
oqy2ad.messaging.internetofthings.ibmcloud.com
char pubTopic[] = "iot-2/evt/status1/fmt/json";
char subTopic[] = "iot-2/cmd/command/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

const char *ssid = "Wokwi-GUEST";
const char *password = "";

const int led = 4;
const int servoPin = 2;
const int echo = 12;
const int trig = 14;
const int r = 27;
const int g = 26;
const int b = 25;
```

```
const int y = 33;
const int sec = 0;
const int dht = 15;
long lastMsg = 0;
Servo s;
String data3;

void callback(char *subTopic, byte *payload, unsigned int payloadLength);

#define I2C_ADDR 0x27
#define LCD_COLUMNS 20
#define LCD_LINES 4

LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES);
DHTesp dhtSensor;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);

void setup()
{
  Serial.begin(115200);
  Wire.begin();
  pinMode(A0, INPUT);          // Temperature Sensor
  pinMode(trig, OUTPUT);       // Ultra sonic Trigger
  pinMode(echo, INPUT);        // Ultra sonic Echo
  pinMode(b, OUTPUT);          // BLUE light for LED
  pinMode(g, OUTPUT);          // GREEN light for LED
  pinMode(r, OUTPUT);          // RED light for LED
  pinMode(y, OUTPUT);          // YELLOW light for LED
  pinMode(led, OUTPUT);        // LED for Motor Indication
  s.attach(servoPin, 500, 2400); // Servo Motor
  lcd.init();                  // LCD Display
  lcd.setBacklight(0);
  dhtSensor.setup(dht, DHTesp::DHT22);

  Serial.println();
  // Connecting the ESP32 with WiFi:
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password, 6);
  while (WiFi.status() != WL_CONNECTED)
  {
```

```
        delay(500);
        Serial.print(".");
    }
    Serial.println("");

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

    // Connecting to IBM Cloud:
    if (!client.connected())
    {
        Serial.print("Reconnecting client to ");
        Serial.println(server);
        while (!client.connect(clientId, authMethod, token))
        {
            Serial.print(".");
            delay(500);
        }

        client.setCallback(callback);
        if (client.subscribe(subTopic))
        {
            Serial.println("Subscription to cmd OK");
        }
        else
        {
            Serial.println("Subscription to cmd FAILED");
        }

        Serial.println("Bluemix connected");
        Serial.println("");
    }
}

float readDistanceCM()
{
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    int duration = pulseIn(echo, HIGH);
    return duration * 0.034 / 2;
```

```
}

void loop()
{

   client.loop();
   long now = millis();

   // Temperature:
   TempAndHumidity data = dhtSensor.getTempAndHumidity();
   float t = data.temperature;
   float h = data.humidity;
   Serial.println("Temperature: " + String(t) + " degrees");
   Serial.println("Moisture: " + String(h) + " %");

   // Ultrasonic sensor:
   float distance = readDistanceCM();
   Serial.print("Measured distance: ");
   Serial.println(readDistanceCM());

   // Soil Moisture:
   int soil = random(0, 100); // As there is no soil moisture sensor, random
function is used for it.
   Serial.println("Soil Moisture: " + String(soil) + "%");

   // LCD Display:
   lcd.setBacklight(1);
   lcd.clear();

   digitalWrite(b, 0);
   digitalWrite(g, 0);
   digitalWrite(r, 0);
   digitalWrite(y, 0);

   // Conditions:
   /*If the temperature is Greater than 30 and less than 40 and also humidity or
soil moisture is greater than 30 and
   less than 70 then the GREEN light will be turned ON indicating the Normal
condition */
   if (t > 30 & t < 40 && h > 30 & h<70 | soil> 30 & soil < 70)
   {
      digitalWrite(g, 1);
      s.write(90);
```

```
      Serial.println("Normal Condition");
      Serial.println("Water Partially Flows");
      lcd.setCursor(3, 1);
      lcd.println("ON Motor");
      delay(1000);
      lcd.clear();
   }

   /*If the temperature is greater than 40 OR the humidity or soil moisture is
less than 30, then the RED light will
   be turned ON indicating the Hot or Low humid condition */
   else if (t > 40 | h < 30 | soil < 30)
   {
      digitalWrite(r, 1);
      s.write(180);
      Serial.println("High Temperature or Low humid condition");
      Serial.println("Water Fully Flows");
      lcd.setCursor(3, 1);
      lcd.println("ON Motor");
      delay(1000);
      lcd.clear();
   }

   /*If the level of water is MORE in the field it will be indicated by distance
sensor for less than
   10cm and soil moisture is greater than 70, then the YELLOW light will be
turned ON indicating the high water level */
   else if (distance<10 & soil> 70)
   {
      digitalWrite(y, 1);
      s.write(0);
      Serial.println("Water Does Not Flow");
      Serial.println("Water is Full in the field");
      lcd.setCursor(2, 1);
      lcd.println("Drain the water");
      delay(1000);
      lcd.clear();
   }

   /*If the temperature is less than 30 OR the humidity or soil moisture is
greater than 70, then the BLUE light will
   be turned ON indicating the Cool or High humid condition */
   else if (t<30 | h> 70 | soil > 70)
```

```
{
   digitalWrite(b, 1);
   s.write(0);
   Serial.println("Cool Temperature or High Humid Condition");
   Serial.println("Water Does Not Flow");
   lcd.setCursor(3, 1);
   lcd.println("OFF Motor");
   delay(1000);
   lcd.clear();
}

else
{
   digitalWrite(b, 1);
   s.write(0);
   Serial.println("Water Does Not Flow");
}

// Sending payload:
Serial.println("");
if (now - lastMsg > 1000)
{
   lastMsg = now;

   // Payload for Parameters:
   String payload = "{\"Name\":\"" DEVICE_ID "\"";
   payload += ",\"Temperature\":";
   payload += t;
   payload += ",\"Humidity\":";
   payload += h;
   payload += ",\"Distance\":";
   payload += distance;
   payload += ",\"SoilMoisture\":";
   payload += soil;
   payload += "}";
   Serial.print("Sending payload: ");
   Serial.println(payload);
   Serial.println("");
   if (client.publish(pubTopic, (char *)payload.c_str()))
   {
      Serial.println("Publish ok for payload");
   }
   else
```

```
    {
      Serial.println("Publish failed");
    }
  }
  Serial.println("-------------------------------------------------");

  lcd.setCursor(1, 0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" degree");
  lcd.setCursor(1, 1);
  lcd.print("Humidity: ");
  lcd.print(h);
  lcd.print(" %");
  lcd.setCursor(1, 2);
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print(" cm");
  lcd.setCursor(1, 3);
  lcd.print("Soil Moisture: ");
  lcd.print(soil);
  lcd.print(" %");
  delay(5000);
  lcd.clear();
}

void callback(char *subTopic, byte *payload, unsigned int payloadLength)
{
  Serial.println("----------------------Callback!!----------------------");
  Serial.print("Callback invoked for topic:");
  Serial.println(subTopic);
  for (int i = 0; i < payloadLength; i++)
  {
    data3 += (char)payload[i];
  }
  Serial.println("Data:" + data3);
  if (data3 == "motoron")
  {
    Serial.println("Motor is ON");
    digitalWrite(led, 1);
  }
  else
  {
```

```
      Serial.println("Motor is Off");
      digitalWrite(led, 0);
    }
    data3 = "";
    Serial.println("-----------------------------------------------------");
}
```