SPRINT – 2

DATE	**03 November 2022**
TEAM ID	**PNT2022TMID27433**
PROJECT NAME	**Smart Farmer - IoT Enabled Smart Farming Application**

SMART FARMER PROGRAM IN WOKWI USING ESP32

Description:

This Smart Irrigation System is used to help farmers in the irrigation process. The System provides data on the parameters which can be used to monitor the condition of the field to maintain and protect the crops. The parameters like temperature, humidity, the water level in the field, etc., can be accessed through the system. The sensors in the system monitor the parameters and provide them to the farmer through the Wi-Fi module to the IBM cloud to take the necessary measures.

Program:

```
#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";
char pubTopic[] = "iot-2/evt/status1/fmt/json";
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 char 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;
#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, NULL, 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 WiFi
  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:
```

```
pinMode(led, OUTPUT);
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(receivedCallback);
   if (client.subscribe(subTopic))
```

```
        {
            Serial.println("subscribe to cmd OK");
        }
        else
        {
            Serial.println("subscribe 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());

// 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 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)
```

```
{
  digitalWrite(g, 1);
  s.write(90);
  Serial.println("Normal Condition");
  Serial.println("Water Partially Flows");
}
```

/*If the temperature is greater than 40 OR the humidity 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)
{
  digitalWrite(r, 1);
  s.write(180);
  Serial.println("High Temperature or Low humid condition");
  Serial.println("Water Fully Flows");
  lcd.clear();
  lcd.println("Drain the water");
  delay(2000);
}
```

/*If the level of water is MORE in the field it will be indicated by distance sensor for less than 10cm, then the YELLOW light will be turned ON indicating the high water level */

```
else if (distance < 10)
{
  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(2000);
    lcd.clear();
  }
```

/*If the temperature is less than 30 OR the humidity 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)
  {
    digitalWrite(b, 1);
    s.write(0);
    Serial.println("Cool Temperature or High Humid Condition");
    Serial.println("Water Does Not Flow");
  }

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

  lcd.setCursor(1, 0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" degree");
```

```
lcd.setCursor(1, 1);
lcd.print("Distance: ");
lcd.print(distance);
lcd.print(" cm");
lcd.setCursor(1, 2);
lcd.print("Moisture: ");
lcd.print(h);
lcd.print(" %");
delay(3000);
lcd.clear();

//Sending payload:
Serial.println("");
if (now - lastMsg > 5000)
{
   lastMsg = now;
   String payloadt = "{\"d\":{\"Name\":\"" DEVICE_ID "\"";
   payloadt += ",\"Temperature\":";
   payloadt += t;
   payloadt += "}}";
   Serial.print("Sending payload t: ");
   Serial.println(payloadt);

   if (client.publish(pubTopic, (char *)payloadt.c_str()))
   {
      Serial.println("Publish ok for t");
   }
```

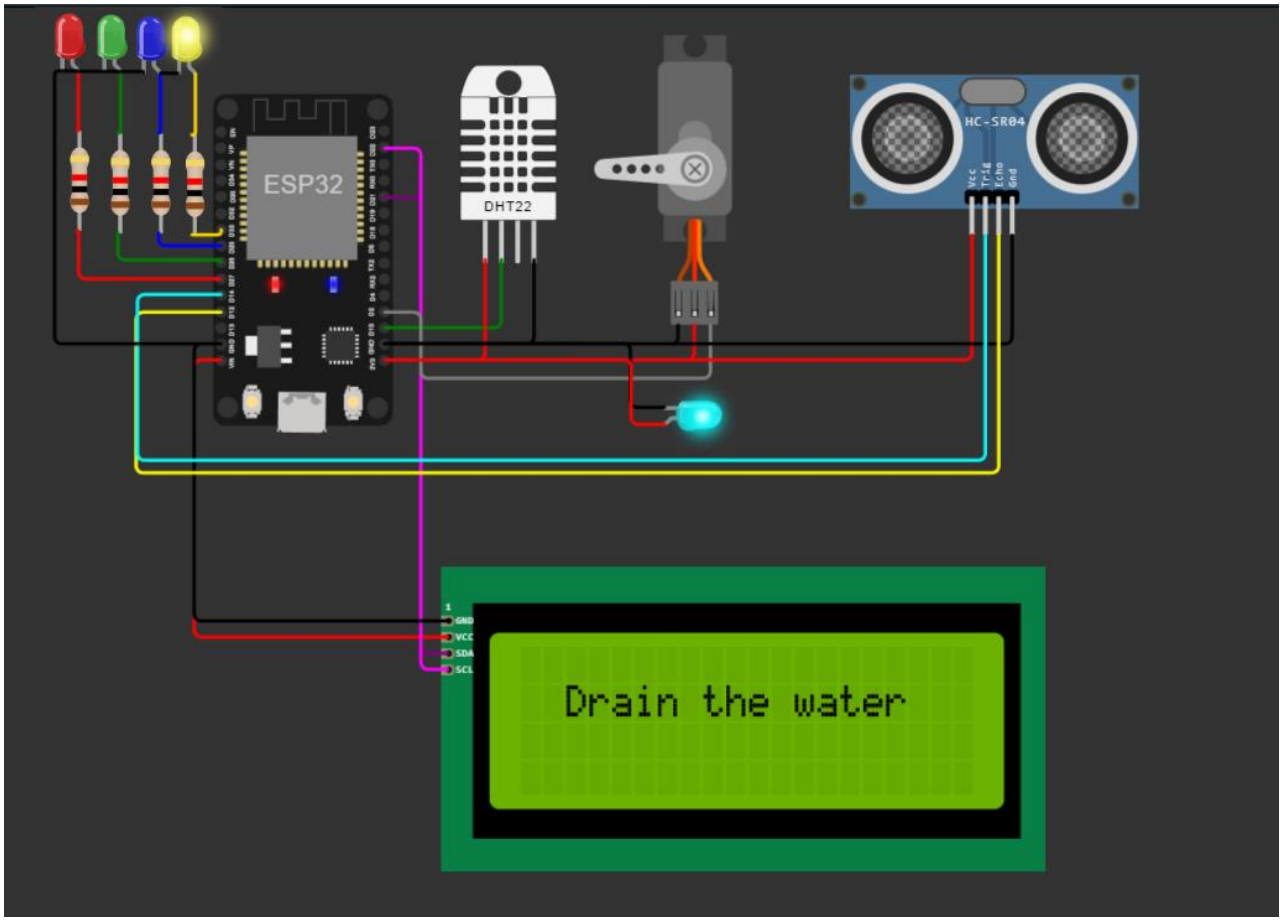
```
    else
    {
      Serial.println("Publish failed");
    }
    delay(3000);

    String payloadh = "{\"d\":{\"Name\":\"" DEVICE_ID "\"";
    payloadh += ",\"Humidity\":";
    payloadh += h;
    payloadh += "}}";
    Serial.print("Sending payload h: ");
    Serial.println(payloadh);

    if (client.publish(pubTopic, (char *)payloadh.c_str()))
    {
      Serial.println("Publish ok for h");
    }
    else
    {
      Serial.println("Publish failed");
    }
    delay(2000);
  }

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

Circuit:



Link to Project in Wokwi:

https://wokwi.com/projects/347430571578753618

Note:

The model done in Tinkercad in Sprint-1 cannot send data to IBM cloud, because it does not have any Wi-fi module. So, we used Wokwi to design our project again in this Sprint-2 as it has ESP32 module to send data to IBM cloud for visualisation. But the drawback of using Wokwi is that it does not have soil moisture and DC motor which we have installed in the Tinkercad design. So, the design in the Tinkercad is our original idea and as the Wokwi simulator does not have those sensors we cannot install those sensors here.