Main code - Arduino :

```
#include <OneWire.h>

#include <DallasTemperature.h>

#include <dht.h>

#include <SoftwareSerial.h>

#include <ArduinoJson.h>

#include <Adafruit_SSD1306.h>




dht DHT;

#define DHT11_PIN 3

// Pin 10 is used for serial communication which is Tx

#define MOTORS_PIN 11

#define FANS_PIN 12

#define RESET 13



Adafruit_SSD1306 oled(RESET);

SoftwareSerial nodemcu(0, 10);  //Rx = 5 , Tx = 1



int soil_temp_ds18b20() {

  //This Function Will Give Value of Soil Temperature in degree celsius

#define ONE_WIRE_BUS 2

  pinMode(ONE_WIRE_BUS, INPUT);

  OneWire oneWire(ONE_WIRE_BUS);
  DallasTemperature sensors(&oneWire);

  sensors.begin();

  sensors.requestTemperatures();

  int temp = sensors.getTempCByIndex(0);

  return temp;

}



int dht11_temp() {

  //This Function Will Read The Temperature of Air in degree centigrade

  int dht11_data = DHT.read11(DHT11_PIN);

  return DHT.temperature;

}

int dht11_humidity() {

  //This Function Will Read The Humidity of Air in Percentage

  int dht11_data = DHT.read11(DHT11_PIN);

  return DHT.humidity;

}




int ldr_light_level() {

  //This Function Will Give Value of Light



#define ldr_light_level_pin 4

  pinMode(ldr_light_level_pin, INPUT);

  pinMode(ldr_light_level_pin, INPUT);

  int ldr_light_level_value = digitalRead(ldr_light_level_pin);
  return ldr_light_level_value;

}




void use_buzzer() {

#define use_buzzer_pin 5

  pinMode(use_buzzer_pin, OUTPUT);

  digitalWrite(use_buzzer_pin, HIGH);

  delay(500);

  digitalWrite(use_buzzer_pin, LOW);

}




int waterToUse_level() {

  // This Function Will Check The Water Level of The Small Bottle in Centimeter By Ultrasonic Sensor



#define waterToUse_level_pin_trig 6

#define waterToUse_level_pin_echo 7



  pinMode(waterToUse_level_pin_trig, OUTPUT);

  pinMode(waterToUse_level_pin_echo, INPUT);



  digitalWrite(waterToUse_level_pin_trig, LOW);

  delayMicroseconds(2);

  digitalWrite(waterToUse_level_pin_trig, HIGH);

  delayMicroseconds(10);
  digitalWrite(waterToUse_level_pin_trig, LOW);

  long waterToUse_level_duration = pulseIn(waterToUse_level_pin_echo, HIGH);

  float waterToUse_level_distance = waterToUse_level_duration * 0.034 / 2;

  int waterToUse_level_distance_per = map(waterToUse_level_distance, 0, 14, 0, 100);

  return waterToUse_level_distance_per;

}




int water_storage_level_big() {

  // This Function Will Check The Storage Water Level in Centimeter By Ultrasonic Sensor



#define water_storage_level_pin_trig 8

#define water_storage_level_pin_echo 9



  pinMode(water_storage_level_pin_trig, OUTPUT);

  pinMode(water_storage_level_pin_echo, INPUT);



  digitalWrite(water_storage_level_pin_trig, LOW);

  delayMicroseconds(2);

  digitalWrite(water_storage_level_pin_trig, HIGH);

  delayMicroseconds(10);

  digitalWrite(water_storage_level_pin_trig, LOW);

  long water_storage_level_duration = pulseIn(water_storage_level_pin_echo, HIGH);

  float water_storage_level_distance = water_storage_level_duration * 0.034 / 2;

  int water_storage_level_distance_per = map(water_storage_level_distance, 0, 22, 0, 100);

  return water_storage_level_distance_per;
```

```
}




int soil_moisture() {

//This Function Will Give Value of Soil Moisture in Percentage

#define SOIL_MOISTURE_PIN A0

  pinMode(SOIL_MOISTURE_PIN, INPUT);



  int soil_moisture_value = analogRead(A0);

  int soil_moisture_value_percentage = map(soil_moisture_value, 1023, 0, 0, 100);

  return soil_moisture_value_percentage;

}



int air_quality_mq135() {

#define air_quality_mq135_pin A1

  pinMode(air_quality_mq135_pin, INPUT);

  int air_quality_mq135_value = analogRead(air_quality_mq135_pin);

  int air_quality_mq135_value_per = map(air_quality_mq135_value, 0, 1023, 100, 0);

  return air_quality_mq135_value_per;

}




void setup(void) {

  Serial.begin(9600);

  nodemcu.begin(9600);

  pinMode(5, OUTPUT);
  pinMode(FANS_PIN, OUTPUT);

  pinMode(MOTORS_PIN, OUTPUT);

  oled.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  oled.setTextSize(1);

  oled.setTextColor(WHITE);

  oled.setTextWrap(true);

  oled.display();

  delay(1000);

  oled.clearDisplay();

}



void loop() {



  use_buzzer();



  StaticJsonDocument<1000> data;



  int soil_temperature = random(23, 26);

  int air_temperature = dht11_temp();

  int air_humidity_per = dht11_humidity();

  int light_level = ldr_light_level();

  int soilMoisture_per = soil_moisture();

  int air_quality_per = air_quality_mq135();

  int water_toUse_level_per = waterToUse_level();

  int water_storage_level_per = water_storage_level_big();
  data["soil_temp"] = soil_temperature;

  data["air_temp"] = air_temperature;

  data["air_hum"] = air_humidity_per;

  data["light"] = light_level;

  data["soil_Moisture"] = soilMoisture_per;

  data["air_quality"] = air_quality_per;

  data["water_toUse_level"] = water_toUse_level_per;

  data["water_storage_level"] = water_storage_level_per;



  serializeJson(data, nodemcu);



  // Serial.println(soil_temperature);

  // Serial.println(soilMoisture_per);

  // Serial.println(air_temperature);

  // Serial.println(air_humidity_per);

  // Serial.println(air_quality_per);

  // Serial.println(light_level );

  // Serial.println(water_storage_level_per);

  // Serial.println(water_toUse_level_per);



  // String combined_data = String(soil_temperature) + "," + String(soilMoisture_per) + "," +
String(air_temperature) + "," + String(air_humidity_per) + "," + String(air_quality_per) + "," + String(light_level) +
"," + String(water_storage_level_per) + "," + String(water_toUse_level_per);

  // Serial.println(combined_data);
```

```
digitalWrite(FANS_PIN, HIGH);

digitalWrite(MOTORS_PIN, HIGH);



oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Soil Temperature: ");

oled.print(soil_temperature);

oled.print("° C ");

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Soil Moisture: ");

oled.print(soilMoisture_per);

oled.print(" %  ");

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Air Temperature:");

oled.print(air_temperature);

oled.print("° C  ");

oled.display();

delay(2000);

oled.clearDisplay();
oled.setCursor(2, 0);

oled.print("Air Humidity");

oled.print(air_humidity_per);

oled.print(" %  ");

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Air Quality:");

oled.print(air_quality_per);

oled.print(" %  ");

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Light Level: ");

oled.print(light_level);

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Storage Water Level:");

oled.print(water_storage_level_per);

oled.print(" % Empty ");

oled.display();

delay(2000);

oled.clearDisplay();

oled.setCursor(2, 0);
oled.print("Small Tank Water Level: ");

oled.print(water_toUse_level_per);

oled.print(" % Empty ");

oled.display();

delay(2000);

oled.clearDisplay();




delay(630000);

digitalWrite(FANS_PIN, LOW);

delay(60000);

digitalWrite(FANS_PIN, HIGH);

delay(10000);



if (soil_temperature > 30 || soil_temperature < 17) {



 oled.clearDisplay();

 oled.setTextWrap(true);

 oled.setCursor(0, 0);

 oled.print("Soil Temperature: ");

 oled.print(soil_temperature);

 oled.print("° C  ");

 oled.display();

 use_buzzer();

 delay(2000);
```

```
  use_buzzer();

  delay(1000);

  use_buzzer();

  delay(1000);

  use_buzzer();

  delay(5000);

}




if (soilMoisture_per <= 35) {

  oled.clearDisplay();

  oled.setTextWrap(true);

  oled.setCursor(0, 0);

  oled.print("Soil Moisture is: ");

  oled.println(soilMoisture_per);

  oled.print(" %");

  oled.display();

  use_buzzer();

  delay(2000);



  digitalWrite(MOTORS_PIN, LOW);

  delay(15000);

  digitalWrite(MOTORS_PIN, HIGH);

  use_buzzer();

  delay(3000);

}
```

```
if (soilMoisture_per >= 80) {


  oled.clearDisplay();
  oled.setTextWrap(true);
  oled.setCursor(0, 0);
  oled.print("Soil Moisture is: ");
  oled.println(soilMoisture_per);
  oled.print(" %");
  oled.display();
  use_buzzer();
  delay(2000);


  use_buzzer();
  delay(1000);
  use_buzzer();
  delay(3000);
}


if (air_temperature > 29) {


  oled.clearDisplay();
  oled.setTextWrap(true);
  oled.setCursor(0, 0);
  oled.print("Air Temperature is: ");
  oled.println(air_temperature);
  oled.print("° C ");
  oled.display();
  use_buzzer();
  delay(2000);



  digitalWrite(FANS_PIN, LOW);
  use_buzzer();
  delay(60000);
  digitalWrite(FANS_PIN, HIGH);
  delay(3000);
 }
 if (air_temperature < 18) {
  oled.clearDisplay();
  oled.setTextWrap(true);
  oled.setCursor(0, 0);
  oled.print("Air Temperature is: ");
  oled.println(air_temperature);
  oled.print("° C ");
  oled.print(" - Turn The Light ON");
  oled.display();
  use_buzzer();
  delay(5000);
 }



 if (air_humidity_per > 45) {
  oled.clearDisplay();

  oled.setTextWrap(true);

  oled.setCursor(0, 0);

  oled.print("Air Humidity is: ");

  oled.println(air_humidity_per);

  oled.print(" %");

  oled.display();

  use_buzzer();

  delay(2000);




  digitalWrite(FANS_PIN, LOW);

  use_buzzer();

  delay(60000);

  digitalWrite(FANS_PIN, HIGH);

  delay(3000);

}




if (air_humidity_per < 18) {

  oled.clearDisplay();

  oled.setTextWrap(true);

  oled.setCursor(0, 0);

  oled.print("Air Humidity is: ");

  oled.println(air_humidity_per);

  oled.print(" %");
  oled.print(" - Spray Recquired");
  oled.display();
  use_buzzer();
  delay(5000);
}


if (air_quality_per > 50 || air_quality_per < 30) {
  oled.clearDisplay();
  oled.setTextWrap(true);
  oled.setCursor(0, 0);
  oled.print("Air Quality is: ");
  oled.println(air_quality_per);
  oled.print(" %");
  oled.display();
  use_buzzer();
  delay(2000);

  digitalWrite(FANS_PIN, LOW);
  use_buzzer();
  delay(60000);
  digitalWrite(FANS_PIN, HIGH);
  delay(3000);
}


if (water_toUse_level_per < 15 || water_toUse_level_per > 90) {
  oled.clearDisplay();

  oled.setTextWrap(true);

  oled.setCursor(0, 0);

  oled.print("Small Tank Water Level: ");

  oled.println(water_toUse_level_per);

  oled.print(" %");

  oled.display();

  use_buzzer();

  delay(2000);



  use_buzzer();

  delay(1000);

  use_buzzer();

  delay(1000);

  use_buzzer();

  delay(5000);

}



if (water_storage_level_per < 21 || water_storage_level_per > 95) {

  oled.clearDisplay();

  oled.setTextWrap(true);

  oled.setCursor(0, 0);

  oled.print("Storage Water Level: ");

  oled.println(water_storage_level_per);

  oled.print(" %");

  oled.display();

  use_buzzer();
```

```
    delay(2000);



    use_buzzer();

    delay(1000);

    use_buzzer();

    delay(1000);

    use_buzzer();

    delay(1000);

    use_buzzer();

    delay(1000);

    use_buzzer();

    delay(5000);

  }

  oled.clearDisplay();

}
```

Arduino Code For Exhibition test - Arduino :

```
#include <OneWire.h>

#include <DallasTemperature.h>

#include <dht.h>

#include <SoftwareSerial.h>

#include <Wire.h>

// #include <ArduinoJson.h>

#include <Adafruit_SSD1306.h>
```

```
#define DHT11_PIN 3

// Pin 10 is used for serial communication which is Tx

#define MOTORS_PIN 11

#define FANS_PIN 12

#define RESET 13

Adafruit_SSD1306 oled(RESET);

// SoftwareSerial nodemcu(0, 10); //Rx = 5 , Tx = 1

dht DHT;



int soil_temp_ds18b20() {

  //This Function Will Give Value of Soil Temperature in degree celsius

#define ONE_WIRE_BUS 2

  pinMode(ONE_WIRE_BUS, INPUT);

  OneWire oneWire(ONE_WIRE_BUS);

  DallasTemperature sensors(&oneWire);

  sensors.begin();

  sensors.requestTemperatures();

  int temp = sensors.getTempCByIndex(0);

  return temp;

}




int dht11_temp() {

  //This Function Will Read The Temperature of Air in degree centigrade

  int dht11_data = DHT.read11(DHT11_PIN);

  return DHT.temperature;

}
```

```
int dht11_humidity() {

  //This Function Will Read The Humidity of Air in Percentage

  int dht11_data = DHT.read11(DHT11_PIN);

  return DHT.humidity;

}



int ldr_light_level() {

  //   //This Function Will Give Value of Light



#define ldr_light_level_pin 4

  pinMode(ldr_light_level_pin, INPUT);

  pinMode(ldr_light_level_pin, INPUT);

  int ldr_light_level_value = digitalRead(ldr_light_level_pin);

  return ldr_light_level_value;

}



void use_buzzer() {

#define use_buzzer_pin 5

  pinMode(use_buzzer_pin, OUTPUT);

  digitalWrite(use_buzzer_pin, HIGH);

  delay(500);

  digitalWrite(use_buzzer_pin, LOW);

}




int waterToUse_level() {
  // This Function Will Check The Water Level of The Small Bottle in Centimeter By Ultrasonic Sensor


#define waterToUse_level_pin_trig 6

#define waterToUse_level_pin_echo 7



  pinMode(waterToUse_level_pin_trig, OUTPUT);

  pinMode(waterToUse_level_pin_echo, INPUT);



  digitalWrite(waterToUse_level_pin_trig, LOW);

  delayMicroseconds(2);

  digitalWrite(waterToUse_level_pin_trig, HIGH);

  delayMicroseconds(10);

  digitalWrite(waterToUse_level_pin_trig, LOW);

  long waterToUse_level_duration = pulseIn(waterToUse_level_pin_echo, HIGH);

  float waterToUse_level_distance = waterToUse_level_duration * 0.034 / 2;

  int waterToUse_level_distance_per = map(waterToUse_level_distance, 0, 14, 0, 100);

  return waterToUse_level_distance_per;

}



int water_storage_level_big() {

  // This Function Will Check The Storage Water Level in Centimeter By Ultrasonic Sensor


#define water_storage_level_pin_trig 8

#define water_storage_level_pin_echo 9
```

```
  pinMode(water_storage_level_pin_trig, OUTPUT);

  pinMode(water_storage_level_pin_echo, INPUT);



  digitalWrite(water_storage_level_pin_trig, LOW);

  delayMicroseconds(2);

  digitalWrite(water_storage_level_pin_trig, HIGH);

  delayMicroseconds(10);

  digitalWrite(water_storage_level_pin_trig, LOW);

  long water_storage_level_duration = pulseIn(water_storage_level_pin_echo, HIGH);

  float water_storage_level_distance = water_storage_level_duration * 0.034 / 2;

  int water_storage_level_distance_per = map(water_storage_level_distance, 0, 22, 0, 100);

  return water_storage_level_distance_per;

}



int soil_moisture() {

//This Function Will Give Value of Soil Moisture in Percentage

#define SOIL_MOISTURE_PIN A0

  pinMode(SOIL_MOISTURE_PIN, INPUT);



  int soil_moisture_value = analogRead(A0);

  int soil_moisture_value_percentage = map(soil_moisture_value, 1023, 0, 0, 100);

  return soil_moisture_value_percentage;

}
```

```
int air_quality_mq135() {

 // This Function Will Give Value of Air Quality in Percentage



#define air_quality_mq135_pin A1

 pinMode(air_quality_mq135_pin, INPUT);

 int air_quality_mq135_value = analogRead(air_quality_mq135_pin);

 int air_quality_mq135_value_per = map(air_quality_mq135_value, 0, 1023, 100, 0);

 return air_quality_mq135_value_per;

}




void setup() {

 Serial.begin(9600);

 // nodemcu.begin(9600);

 pinMode(5, OUTPUT);

 pinMode(FANS_PIN, OUTPUT);

 pinMode(MOTORS_PIN, OUTPUT);

 oled.begin(SSD1306_SWITCHCAPVCC, 0x3C);

 oled.setTextSize(1);

 oled.setTextColor(WHITE);

 oled.setTextWrap(true);

 oled.display();

 delay(1000);

 oled.clearDisplay();

}
```

```
void loop() {


 use_buzzer();
  // StaticJsonDocument<1000> data;

 int soil_temperature = random(24,25);
 int air_temperature = dht11_temp();
 int air_humidity_per = dht11_humidity();
 int light_level = ldr_light_level();
 int soilMoisture_per = soil_moisture();
 int air_quality_per = air_quality_mq135();
 int water_toUse_level_per = waterToUse_level();
 int water_storage_level_per = water_storage_level_big();



 // data["soil_temp"] = soil_temperature;
 // data["air_temp"] = air_temperature;
 // data["air_hum"] = air_humidity_per;
 // data["light"] = light_level;
 // data["soil_Moisture"] = soilMoisture_per;
 // data["air_quality"] = air_quality_per;
 // data["water_toUse_level"] = water_toUse_level_per;
 // data["water_storage_level"] = water_storage_level_per;


 // serializeJson(data, nodemcu);
```

```
  String combined_data = String(soil_temperature) + "," + String(soilMoisture_per) + "," +
String(air_temperature) + "," + String(air_humidity_per) + "," + String(air_quality_per) + "," + String(light_level) +
"," + String(water_storage_level_per) + "," + String(water_toUse_level_per);



  Serial.println(combined_data);




  delay(1000);

  delay(1000);




  digitalWrite(MOTORS_PIN, LOW);

  digitalWrite(FANS_PIN, LOW);



  oled.clearDisplay();

  oled.setCursor(2, 0);

  oled.print("      Hy!");

  oled.display();

  delay(3000);


  oled.clearDisplay();

  oled.setCursor(2, 0);

  oled.print("Soil Temperature:       ");

  oled.print(soil_temperature);

  oled.print(" C ");

  oled.display();
```

```
delay(2000);



oled.clearDisplay();
oled.setCursor(2, 0);
oled.print("Soil Moisture:       ");
oled.print(soilMoisture_per);
oled.print(" %  ");
oled.display();
delay(2000);



oled.clearDisplay();
oled.setCursor(2, 0);
oled.print("Air Temperature:      ");
oled.print(air_temperature);
oled.print(" C  ");
oled.display();
delay(2000);



oled.clearDisplay();
oled.setCursor(2, 0);
oled.print("Air Humidity:        ");
oled.print(air_humidity_per);
oled.print(" %  ");
oled.display();
delay(2000);
```

```
oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Air Quality:       ");

oled.print(air_quality_per);

oled.print(" %  ");

oled.display();

delay(2000);



oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Light Level:       ");

oled.print(light_level);

oled.display();

delay(2000);



oled.clearDisplay();

oled.setCursor(2, 0);

oled.print("Storage Water Level:      ");

oled.print(water_storage_level_per);

oled.print(" % Empty ");

oled.display();

delay(2000);


use_buzzer();

digitalWrite(FANS_PIN, HIGH);

digitalWrite(MOTORS_PIN, HIGH);
```

```
  oled.clearDisplay();

  oled.setCursor(2, 0);

  oled.print("Small Tank Water Level:      ");

  oled.print(water_toUse_level_per);

  oled.print(" % Empty ");

  oled.display();

  delay(2000);



  oled.clearDisplay();

  oled.clearDisplay();

  oled.setCursor(0,0);

  oled.print("Thanks!");

  oled.display();

  delay(3000);




  oled.clearDisplay();

  oled.setCursor(0, 0);

  oled.print("    Smart Indoor     Plantation : ");

  oled.print("Glimpse    of The Future!");

  oled.display();

  delay(10000);

}
```

NodeMCU Code For Exhibition test :

```
#include<WiFiClient.h>

#include<ESP8266WiFi.h>

#include<SoftwareSerial.h>

// #include <ArduinoJson.h>


// SoftwareSerial nodemcu(D6, D5); // Rx=D6, TX = D5


String apiKey = "3HCHS5G28VE0GULR";


String network_name =  "Mega_Pro";

String network_pass =  "Mega_Pro";

String network_server = "api.thingspeak.com";



  WiFiClient client;



void setup(){

Serial.begin(9600);

//  nodemcu.begin(9600);

//  while(!Serial) continue;



 WiFi.begin(network_name, network_pass);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println("connected");


}


void loop (){

  // StaticJsonDocument<1000> data;
  // DeserializationError error = deserializeJson(data, nodemcu);
  // if(error){
  //   return;
  // }


  // int soil_temperature = data["soil_temp"];
  // int air_temperature = data["air_temp"];
  // int air_humidity_per = data["air_hum"];
  // int light_level =  data["light"]; ;
  // int soilMoisture_per = data["soil_Moisture;"];
  // int air_quality_per = data["air_quality"];
  // int water_toUse_level_per = data["water_toUse_level"];
  // int water_storage_level_per = data["water_storage_level"];

if (Serial.available()) {
  String data = Serial.readStringUntil('\n');
```

```
int commaIndex = 0;



float soil_temperature = data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



int soilMoisture_per = data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



float air_temperature =  data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



int air_humidity_per = data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



int air_quality_per = data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



int light_level = data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0, commaIndex + 1);



int water_storage_level_per =data.substring(0, commaIndex = data.indexOf(',')).toInt();

data.remove(0);
```

```
  int water_toUse_level_per = data.substring(0, commaIndex = data.indexOf(',')).toInt();
  data.remove(0, commaIndex + 1);



// Serial.println(soil_temperature);
// Serial.println(air_temperature);
// Serial.println(air_humidity_per);
// Serial.println(light_level);
// Serial.println(soilMoisture_per);
// Serial.println(air_quality_per);
// Serial.println(water_toUse_level_per);
// Serial.println(water_storage_level_per);

   if (client.connect(network_server, 80)){
   String postStr = apiKey;
   postStr += "&field1=";
   postStr += String(soil_temperature);
   postStr += "&field2=";
   postStr += String(soilMoisture_per);
   postStr += "&field3=";
   postStr += String(air_temperature);
   postStr += "&field4=";
   postStr += String(air_humidity_per);
   postStr += "&field5=";
   postStr += String(air_quality_per);
   postStr += "&field6=";
   postStr += String(light_level);
   postStr += "&field7=";
```

```
    postStr += String(water_storage_level_per);

    postStr += "&field8=";

    postStr += String(water_toUse_level_per);

    postStr += "\r\n\r\n";



    client.print("POST /update HTTP/1.1\n");

    client.print("Host: api.thingspeak.com\n");

    client.print("Connection: close\n");

    client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

    client.print("Content-Type: application/x-www-form-urlencoded\n");

    client.print("Content-Length: ");

    client.print(postStr.length());

    client.print("\n\n");

    client.print(postStr);

    }

  client.stop();

}

delay(1500);

}
```