Node MCU for farm code

```
#include <ESP8266WiFi.h>
#include <DHT.h>

#define DHTPIN D2           // Pin where the DHT11 is connected
#define DHTTYPE DHT11       // DHT 11
#define SOIL_PIN A0         // Pin where the Soil Moisture sensor is
connected
#define ANALOG_OUT D1       // Pin for analog output of Soil Moisture

DHT dht(DHTPIN, DHTTYPE);

// WiFi credentials
const char* ssid = "Test";           // Replace with your Wi-Fi SSID
const char* password = "xyz12345";   // Replace with your Wi-Fi Password

// ThingSpeak API credentials
const char* server = "api.thingspeak.com";
String apiKey = "WR1MY3YDXF4HM0GT";  // Replace with your ThingSpeak Write
API Key

void setup() {
  Serial.begin(9600);
  delay(10);

  dht.begin();
  pinMode(ANALOG_OUT, OUTPUT);  // Set up analog output pin

  // Connect to Wi-Fi
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

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

```
  }
  Serial.println();
  Serial.println("WiFi connected");
}

void loop() {
  // Read DHT11 sensor data
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Read soil moisture sensor value
  int soilMoistureValue = analogRead(SOIL_PIN);

  // Convert soil moisture to analog output range (0-255)
  int analogOutValue = map(soilMoistureValue, 0, 1023, 0, 255);
  analogWrite(ANALOG_OUT, analogOutValue);

  // Check if any reads failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    // return;
  }

  // Log sensor values to Serial Monitor
  Serial.print("Humidity: "); Serial.print(humidity);
  Serial.print(" %\t Temperature: "); Serial.print(temperature);
Serial.println(" *C");
  Serial.print("Soil Moisture: "); Serial.println(soilMoistureValue);
  Serial.print("Analog Output (Soil): "); Serial.println(analogOutValue);

  // Send data to ThingSpeak
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;

    if (client.connect(server, 80)) {
      String postStr = apiKey;
      postStr += "&field1=" + String(humidity);  // Field 1: Soil Moisture
      postStr += "&field2=" + String(temperature);       // Field 2:
Temperature
```

```
      postStr += "&field3=" + String(soilMoistureValue);           // Field
3: Humidity

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

      client.print("POST /update HTTP/1.1\n");
      client.print("Host: " + String(server) + "\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);

      Serial.println("Data sent to ThingSpeak");

      client.stop();
    } else {
      Serial.println("Connection to ThingSpeak failed");
    }
  } else {
    Serial.println("WiFi not connected");
  }

  // Delay for 20 seconds to comply with ThingSpeak's rate limit
  delay(20000);
}
```

NODE MCU FOR TANK CODE

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h> // Library for MLX90614 infrared temperature sensor


// Pin Definitions
#define TDS_PIN A0          // TDS Sensor connected to A1
#define TRIG_PIN D5         // Infrared sensor Trig pin
#define ECHO_PIN D6         // Infrared sensor Echo pin
#define ANALOG_OUT D3       // Analog output for water level (0-100%)


// Infrared Temperature Sensor
Adafruit_MLX90614 mlx = Adafruit_MLX90614();


// WiFi credentials
const char* ssid = "Test";            // Replace with your Wi-Fi SSID
const char* password = "xyz12345";    // Replace with your Wi-Fi Password


// ThingSpeak API credentials
const char* server = "api.thingspeak.com";
String apiKey = "WR1MY3YDXF4HM0GT";  // Replace with your ThingSpeak Write API Key

// Thingspeak Read:
unsigned long channelID = 2676023; // Replace with your ThingSpeak Channel ID
const char* readAPIKey = "X5RFW34FCEA1L6YH";


void setup() {
  Serial.begin(9600);
  delay(10);
```

```
  // Initialize sensors
  mlx.begin();


  pinMode(TDS_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(ANALOG_OUT, OUTPUT);


  // Connect to Wi-Fi
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  ThingSpeak.begin(client);
  pinMode(D7, OUTPUT);
}


void loop() {

  float field8Data = ThingSpeak.readFloatField(channelID, 8, readAPIKey);

  // Check if data is valid
  if (field8Data == NAN) {
    Serial.println("Error reading data from ThingSpeak!");
  } else {
    Serial.print("Field 8 Data: ");
    Serial.println(field8Data);
```

```
    // Set the D7 pin based on the value of Field 8 (assuming 0 or 1)
    if (field8Data == 1) {
      digitalWrite(D7, HIGH); // Output HIGH to D7 if Field 8 is 1
    } else {
      digitalWrite(D7, LOW); // Output LOW to D7 if Field 8 is 0
    }
  }

  // Read TDS Sensor Value
  int tdsRaw = analogRead(TDS_PIN);
  float tds = map(tdsRaw, 0, 1023, 0, 1000); // Example mapping (adjust
based on calibration)


  // Read Infrared Temperature Sensor
  float objectTemp = mlx.readObjectTempC(); // Temperature of the pointed
object
  float ambientTemp = mlx.readAmbientTempC(); // Ambient temperature


  // Read Distance from Infrared Sensor
  float distance = getDistance(); // Function to calculate distance using
Trig and Echo pins


  // Map distance to water level percentage (11cm = 100%, 5cm = 0%)
  int waterLevel = map(distance, 3.71, 8.71, 100, 0);
  waterLevel = constrain(waterLevel, 0, 100); // Ensure value stays within
0-100


  // Output water level percentage as analog signal
  int analogOutValue = map(waterLevel, 0, 100, 0, 255); // Scale to 0-255
for PWM
  analogWrite(ANALOG_OUT, analogOutValue);


  Serial.print("TDS (ppm): "); Serial.println(tds);
  Serial.print("Object Temperature (°C): "); Serial.println(ambientTemp);
```

```
  Serial.print("Distance (cm): "); Serial.println(distance);
  Serial.print("Water Level (%): "); Serial.println(waterLevel);
  Serial.print("Analog Output: "); Serial.println(analogOutValue);


  // Send data to ThingSpeak
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;


    if (client.connect(server, 80)) {
      String postStr = apiKey;
      // postStr += "&field4=" + String(turbidity);       // Field 1:
Turbidity
      postStr += "&field5=" + String(tds);             // Field 2: TDS
      postStr += "&field7=" + String(ambientTemp);     // Field 3: Object
Temperature
      postStr += "&field6=" + String(waterLevel);     // Field 6: Water
Level Percentage
      postStr += "\r\n\r\n";


      client.print("POST /update HTTP/1.1\n");
      client.print("Host: " + String(server) + "\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);


      Serial.println("Data sent to ThingSpeak");


      client.stop();
    } else {
      Serial.println("Connection to ThingSpeak failed");
```

```
    }
  } else {
    Serial.println("WiFi not connected");
  }

  // Delay for 20 seconds to comply with ThingSpeak's rate limit
  delay(20000);
}

// Function to calculate distance using Trig and Echo pins
float getDistance() {
  // Send 10us pulse to Trig pin
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Read Echo pin duration
  long duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate distance in cm (duration/58 for cm)
  float distance = duration / 58.0;
  return distance;
}
```

Arduino UNO FOR TANK CODE

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin definitions
const int motorPin = 7; // Motor control pin (connected to D7 of ESP8266)

const int turbidityPin = A0; // Analog pin for turbidity sensor

// LCD setup (16x2 LCD with I2C)
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address, width, and height of
the LCD

void setup() {
  // Initialize motor pin
  pinMode(motorPin, OUTPUT);

  // Initialize the LCD
  lcd.begin();
  lcd.print("Turbidity Sensor");

  // Wait for the screen to initialize
  delay(2000);
}

void loop() {
  // Read the D7 pin (if it is HIGH, turn on motor)
  int motorState = digitalRead(motorPin); // Read the state of the motor
control pin

  if (motorState == HIGH) {
    digitalWrite(motorPin, HIGH); // Turn on motor
    lcd.clear();
    lcd.print("Motor: ON");
  } else {
    digitalWrite(motorPin, LOW); // Turn off motor
    lcd.clear();
    lcd.print("Motor: OFF");
  }

  // Read turbidity sensor value (analog value)
  int turbidityValue = analogRead(turbidityPin);
```

```
  // Map the turbidity value to a readable range (adjust the mapping if
necessary)
  float turbidity = map(turbidityValue, 0, 1023, 0, 100);

  // Display turbidity value on LCD
  lcd.setCursor(0, 1); // Move to the second line
  lcd.print("Turbidity: ");
  lcd.print(turbidity);
  lcd.print(" NTU");

  // Wait for a short time before the next loop iteration
  delay(2000); // Adjust the delay as needed
}
```

Arduino UNO FOR Farm CODE

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin definition for soil moisture sensor (analog pin)
const int soilMoisturePin = A0; // Analog pin A0 for soil moisture sensor

// LCD setup (16x2 LCD with I2C)
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address, width, and height of
the LCD

void setup() {
  // Initialize the soil moisture sensor pin (analog)
  pinMode(soilMoisturePin, INPUT);

  // Initialize the LCD
  lcd.begin();
  lcd.print("Soil Moisture:");

  // Wait for the screen to initialize
  delay(2000);
}

void loop() {
  // Read the analog value from the soil moisture sensor
  int moistureValue = analogRead(soilMoisturePin);

  // Display the result on the LCD
  lcd.clear();
  lcd.setCursor(0, 0); // First line
  lcd.print("Soil Moisture:");

  // Map the analog value (0-1023) to a percentage (0-100)
  int moisturePercent = map(moistureValue, 0, 1023, 0, 100);

  // Display the moisture percentage on the second line
  lcd.setCursor(0, 1); // Second line
  lcd.print(moisturePercent);
  lcd.print("%");

  // Wait a bit before updating the display again
  delay(1000); // Adjust delay as needed
}
```