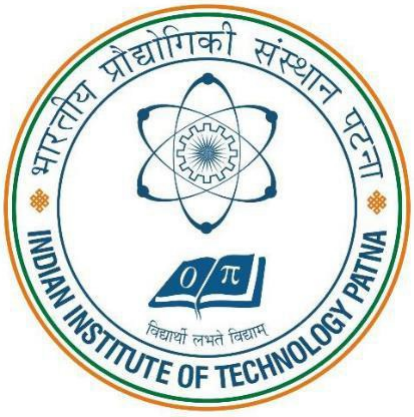
Indian Institute of Technology Patna



Embedded Systems(EE512)
PROJECT

Topic:

IoT based Smart Irrigation System using ESP8266 NodeMCU And Arduino Uno

Submitted by: Aditya Shah**(**2011mt02**)**
(MTech-Mechatronics)

Under the supervision of:
Dr. Udit Satija
(Assistant Professor)
Department of Electrical Engineering
IIT Patna

OBJECTIVE:

- **To implement IoT based Smart Irrigation System using ESP8266 NodeMCU and Arduino Uno**

BACKGROUND:

In this Project, I will work with Smart irrigation System that has wide scope to automate the complete irrigation system. Here I will be building an IoT based Irrigation System using ESP8266 NodeMCU Module,Arduino Uno , Soil Moisture Sensor Module and DHT11 Sensor. It will automatically irrigate based on the moisture level in the soil, Temperature and Humidity of ambient and also send the Data to ThingSpeak Server to keep track of the land condition. The System also consist of a water pump module which will be used to sprinkle water on the land depending upon the land environmental condition such as Moisture, Temperature and Humidity.

Apparatus Used:

- **ESP8266 NodeMCU Development Board**

Description:

NodeMCU ESP8266 is an open-source Lua based firmware and development board specially targeted for IoT based applications. It includes firmware that runs on the **ESP8266 Wi-Fi SoC** from Espressif Systems and hardware which is based on the ESP-12 module, and like this, it can also be programmed using Arduino IDE and can act as both WiFi Hotspot or can connect to one. It has one Analog Input Pin, 16 Digital I/O pins along with the capability to connect with serial communication protocols like SPI, UART, and I2C. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. Its applications include prototyping for IoT devices, low powered battery-operated applications, and projects requiring I/O interface with Bluetooth and WiFi capabilities.

SPECIFICATIONS

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz

➢ **3V to 12V Mini DC Submersible Pump**

Description:

This DC operated mini submersible water pump is ideal for small vending machines and other applications, in which a small amount of water has to be pumped. The operating voltage for this DC pump is between 3 to 12V and can be easily controlled with development boards like Arduino, Raspberry Pi, ESP, and other microcontrollers, so also frequently used in DIY electronics projects and hobby projects. The pumping height for this mini **DC water pump** is between 40 to 110 cm.

SPECIFICATIONS OF THE MINI DC SUBMERSIBLE PUMP

- Operating voltage (Vcc): 3V to 12V
- Pumping Height: between 40 to 110 cm.

- **Soil Moisture Sensor Module**

Description: The soil moisture sensor is commonly used in smart agriculture or other garden automation projects to measure the moisture content present in the soil. It consists of 4 pins in which two pins, Vcc and Gnd are connected to supply voltage. The remaining two pins are digital (D0) and analog (A0) are the output pins. When the moisture content present in the soil goes beyond the threshold level, the output of the digital pin (D0) will go low (the output of the digital pin is either logic 0 or 1). The threshold value of the sensor module can be set by varying the onboard potentiometer. The analog output pin can be used to calculate the approximate level of moisture content present in the soil.

SPECIFICATIONS OF THE SOIL MOISTURE SENSOR

- Operating voltage (Vcc): 3.3V to 5V
- Analog Output voltage: 0 to 4.2V@ Vcc = 5V
- Digital Output Voltage: 0V or 5V @ Vcc=5v
- Current consumption: 32mA

- **DHT11 Temperature and Humidity Sensor**

Description: It's is a basic and low-cost digital temperature and humidity sensor. This comes in a blue perforated plastic enclosure. This has three pin,which are V_{cc}, Gnd and Data pin.

DHT11 Sensor Specifications:

- Temperature measurement accuracy: ±2.0
- Output Signal Digital
- Temperature Range 0 to 50º C
- Humidity Range: 20 to 90% RH
- Humidity Accuracy: ± 5% RH
- Power Supply: 3.3 to 5V
- Size: 2.3cm x 1.2cm x 0.5cm.

- **5V 10A Relay Module**

Description: This **5V Relay module** helps us to switch (control) AC/DC loads from a microcontroller like Arduino, PIC, ARM etc. We can turn on or turn off loads that consume upto 10A using this Relay module. The module also comes with a power indicator led and signal indicator led. Input/Trigger voltage: 5V

The specifications of the 5V relay module are:

- Current consumption: 20mA maximum
- AC load voltage: upto 250V
- DC load voltage: upto 30V
- Load current: upto 10A

Since this Relay Module require 5 volt Input for energizing its coil, I used Arduino uno for actuation because node-mcu can provide a max voltage of 3.3 V only (Since I was not having any external module to be integrated with node -mcu to give the control command to relay).

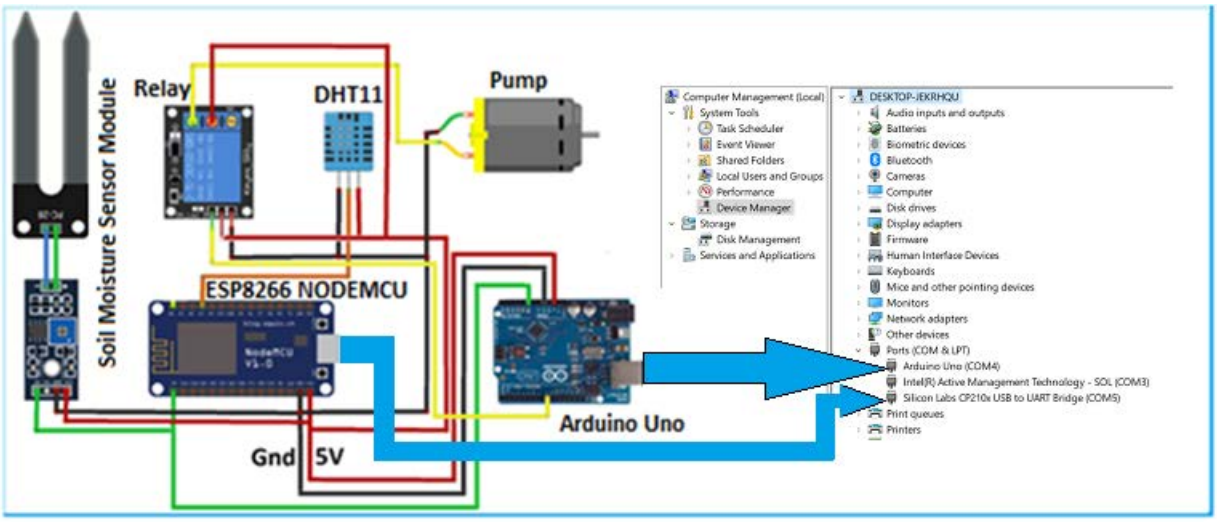
- **UNO R3 Development Board ATmega328P ATmega16U2**

Description: . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

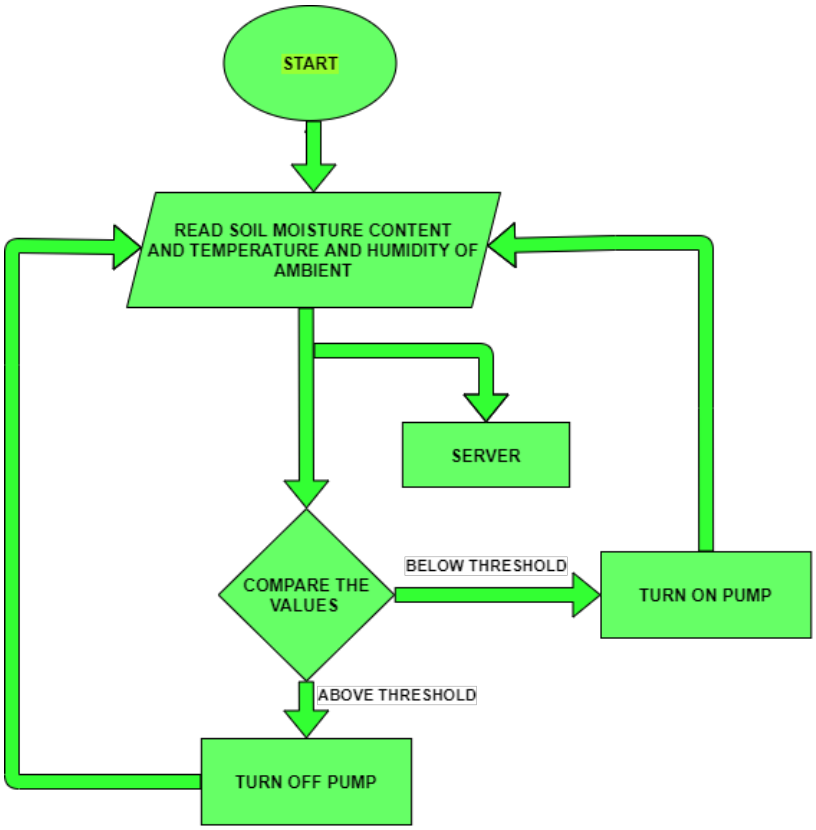
Specifications for this item:

Brand Name :Generic
Is Assembly Required: false
Item Weight :299 grams
Material: UNO R3 Development Board ATmega328P ATmega16U2 with USB cable for Arduino
Model Number: ADU-0004
Special Features: Digital I/O Pins: 14 (of which 6 provide PWM output) , Flash Memory: 32 KB (ATmega328P) of which 0.5 KB used by bootloader. , SRAM: 2 KB (ATmega328P), Operating Voltage 5V

Circuit Diagram:



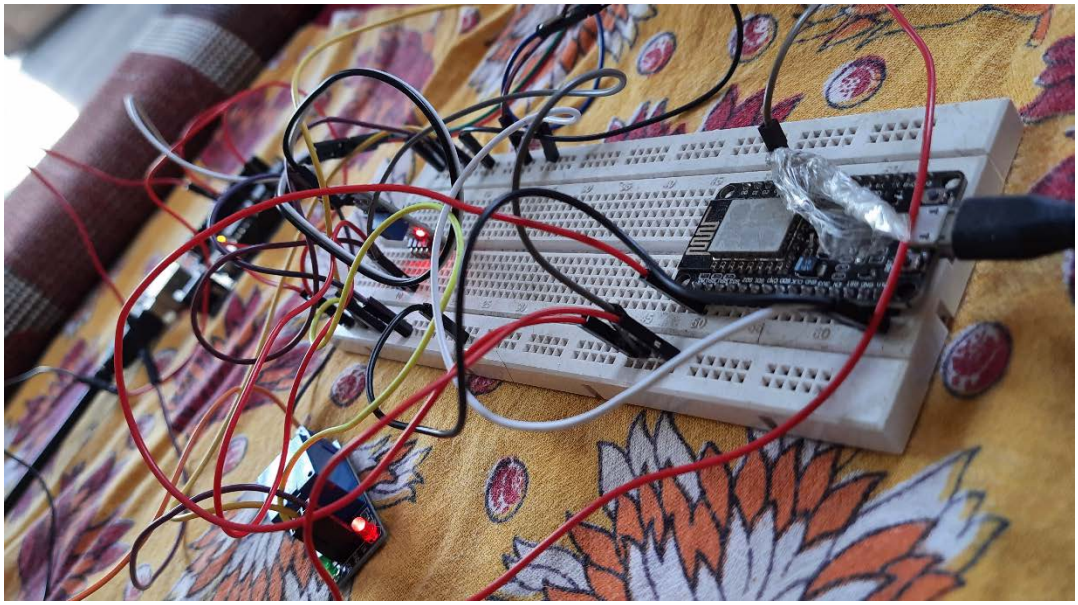
Flow Diagram:



Name: Aditya Shah Roll: 2011mt02

Experimental Setup:

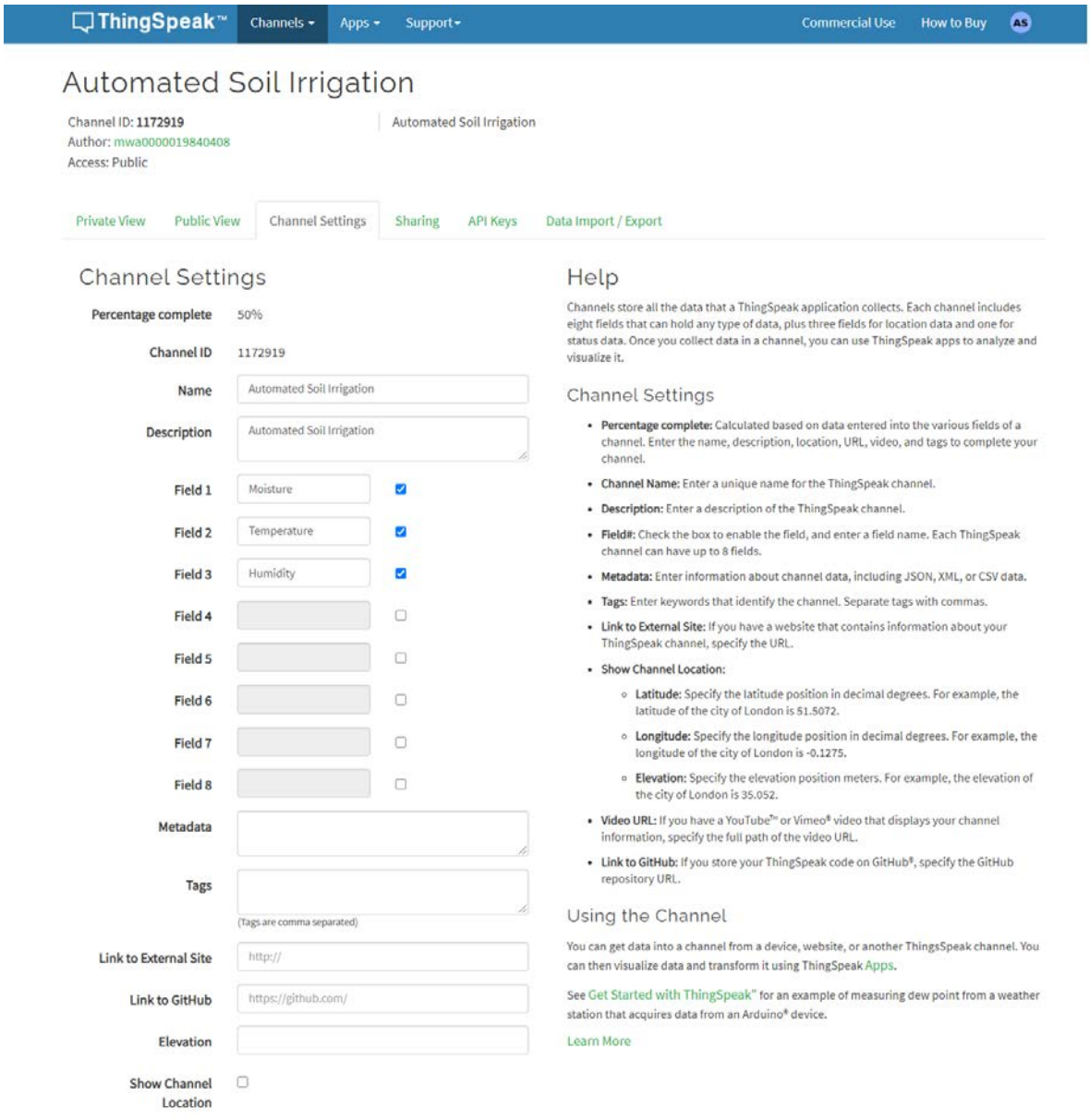
Click on below Image for getting more Visualization of Setup:



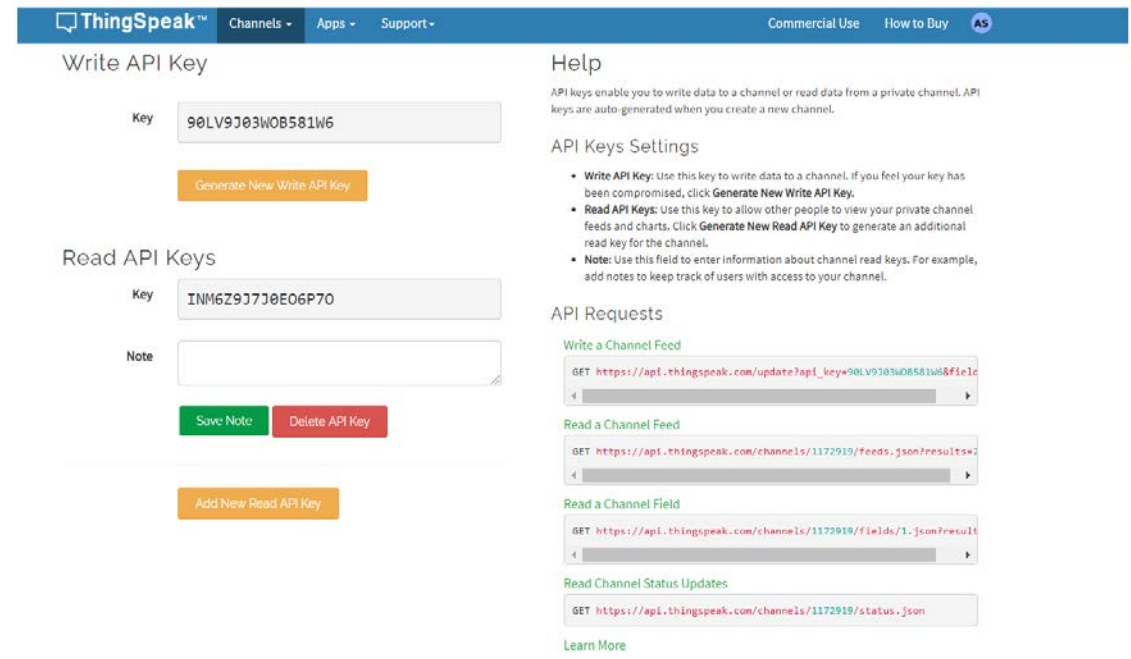
Circuit Visualization

Experimental Setup

Setting up of Channel :



API KEYS: This enable us to write data to a channel or read data from a private channel. API keys are auto-generated when we create a new channel.



Coding

The code for both Arduino Uno and ESP8266 NodeMCU is given below:

ESP8266 NodeMCU-IOT

```
#include <DHT.h>
#include <ESP8266WiFi.h>
String apiKey = "90LV9J03WOB581W6";     //  My API key
const char* server = "api.thingspeak.com";
const char *ssid =  "Alexa";     // My WiFi Name
const char *pass =  "ES-Project"; // My WiFi Password
#define DHTPIN D3          // GPIO Pin where the dht11 is connected
DHT dht(DHTPIN, DHT11);
WiFiClient client;


const int moisturePin = A0;             // moisteure sensor pin
unsigned long interval = 10000;
unsigned long previousMillis = 0;
unsigned long interval1 = 1000;
unsigned long previousMillis1 = 0;
int moisturePercentage;              //moisture reading
float h;                  // humidity reading
float t;                  //temperature reading

void setup()
{
  Serial.begin(115200);
  delay(10);
      dht.begin();
```

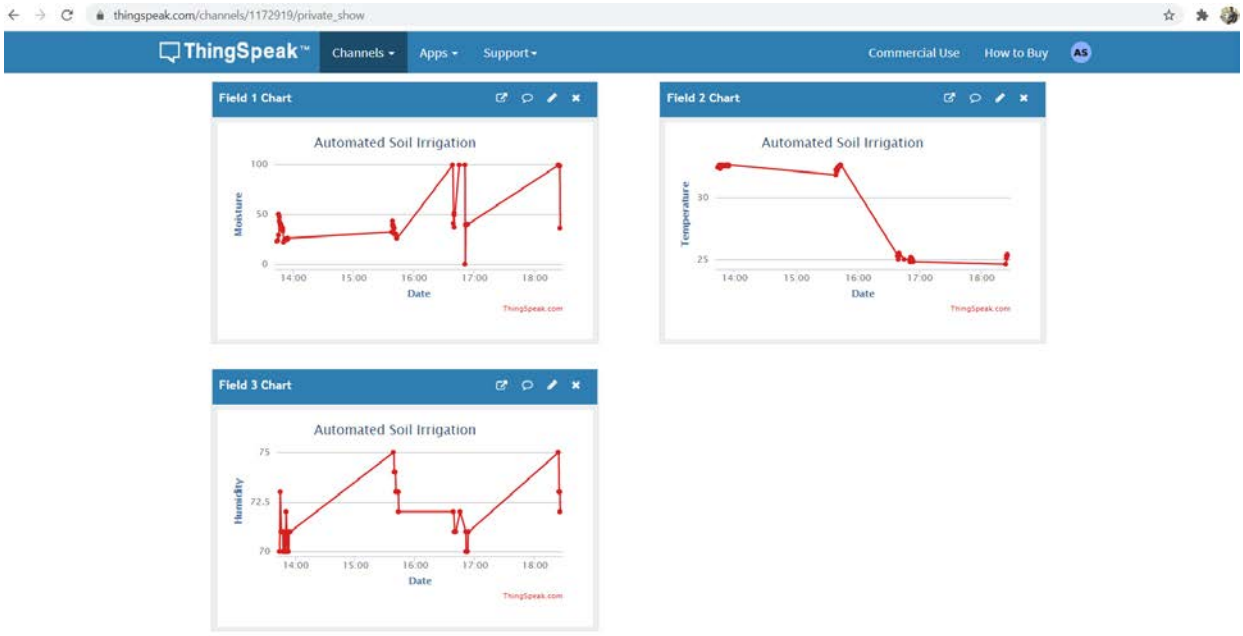
```
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");              // print ... till not connected
  }
  Serial.println("");
  Serial.println("WiFi connected");
}

void loop()
{
  unsigned long currentMillis = millis(); // grab current time

  h = dht.readHumidity();     // read humiduty
  t = dht.readTemperature();     // read temperature

  if (isnan(h) || isnan(t))
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  moisturePercentage = (100.00-((analogRead(moisturePin)/(1023.00))*100.00));


  if ((unsigned long)(currentMillis - previousMillis1) >= interval1) {
    Serial.print("Soil Moisture is  = ");
    Serial.print(moisturePercentage);
    Serial.println("%");
    previousMillis1 = millis();
  }

if ((unsigned long)(currentMillis - previousMillis) >= interval) {

  sendThingspeak();           //send data to thing speak
  previousMillis = millis();
  client.stop();
}

}

void sendThingspeak() {
  if (client.connect(server, 80))
  {
    String postStr = apiKey;              // add api key in the postStr
string
    postStr += "&field1=";
    postStr += String(moisturePercentage);    // add mositure readin
    postStr += "&field2=";
    postStr += String(t);                 // add tempr readin
    postStr += "&field3=";
    postStr += String(h);                  // add humidity readin
    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());           //send lenght of the string
    client.print("\n\n");
    client.print(postStr);                      // send complete string
    Serial.print("Moisture Percentage: ");
    Serial.print(moisturePercentage);
    Serial.print("%. Temperature: ");
    Serial.print(t);
    Serial.print(" C, Humidity: ");
    Serial.print(h);
    Serial.println("%. Sent to Thingspeak.");
  }}
```

ARDUINO CODINGS-ACTUATION

```
const int motorPin = 8;
  int moisturePercentage;
  const int moisturePin = A0;
unsigned long interval1 = 1000;
unsigned long previousMillis1 = 0;

void setup() {
   Serial.begin(115200);
  delay(10);
  pinMode(motorPin, OUTPUT);

}

void loop() {

  unsigned long currentMillis = millis(); // grab current time
  moisturePercentage = (100.00-((analogRead(moisturePin)/(1023.00))*100.00));


  if ((unsigned long)(currentMillis - previousMillis1) >= interval1) {
    Serial.print("Soil Moisture is  = ");
    Serial.print(moisturePercentage);
    Serial.println("%");
    previousMillis1 = millis();
  }
  if (moisturePercentage < 35.00) {
      digitalWrite(motorPin, LOW);
         } else {
      digitalWrite(motorPin, HIGH);
    }}
```

Checking of Connection:



It worked correctly and thus, the microcontroller is sending the Data to ThingSpeak Server to keep track of the land condition i.e, Temperature ,Humidity and moisture information, in a interval of 10 seconds as programmed in Node-MCU.

Working:

Here in this Project,I have used Arduino Uno and NodeMcu , Sensor as discussed above and Pump Module.So basically ,the sensor will provide the Ambient condition data to NodeMcu and Arduino Uno .The NodeMcu after getting the data from the sensor, will send the data to server ,to visualize the ambient condition graphically. The only reason of using Arduino Uno ,is the inability of NodeMCU of providing 5V O/P (NodeMcu at max can give 3.3V) and for actuating the Relay ,we need 5 V for it's proper functioning. So I used the Arduino Uno,which after analyzing the sensor data, will actuate the pump when the ambient parameter crosses below the predefined threshold value .

Click on below Image to see the Working of Automated Irrigation System:

Observation:

During the experiment, it was observed that the relay was not working with node-mcu,so for actuation we used Arduino uno and for sending the ambient condition to server ,it was verified that the data gets to server at every 10 s and the same has been checked in serial monitor. Also it was found that sometimes the DHT 11 was unable to read the temperature and humidity condition ,due to its internal problem ,so we are taking the data after some periodic interval and since the ambient condition doesn't change drastically in short span, so it works well with situation.

Conclusions: So through the Project, I learnt about the two microcontroller and also how to interface various sensor used ,to those microcontroller. I also understood how to setup the channel in ThingSpeak and how to connect the same with NodeMCU ,to send the data of ambient parameters to Server at every predefined time periodically. Thus, with this I successfully implemented the Automated Soil Irrigation System.