

**SMART AGRICULTURE SYSTEM**

**Name: Vedant Badiger**

**College: Bangalore Institute of Technology**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NUMBER** |
| **TABLE OF GRAPH** | **3 - 4** |
| **ABSTRACT** | **5** |
| **CHAPTERS**  (Introduction, Components Used, Libraries used, code, Circuit diagram) | **6-16** |
| **CONCLUSION** | **17** |

**TABLE OF GRAPH**

|  |  |
| --- | --- |
| GRAPH | INFORMATION |
|  | **Graph of moisture vs time recorded on thingspeak.com** |
|  | **Graph of temperature vs time recorded on thingspeak.com** |

|  |  |
| --- | --- |
|  | **Graph of humidity vs time recorded on thingspeak.com** |

**ABSTRACT**

Most of the farmers use large portions of farming land and it becomes very difficult to reach and track each corner of large lands. Sometime there is a possibility of uneven water sprinkles. This result in the bad quality crops which further leads to financial losses. In this scenario the Smart Irrigation System using Latest Iot Technologies helpful and leads to ease of farming.

The **Smart agriculture System** has wide scope to automate the complete irrigation system. Here I am building a **IoT based Irrigation System** using ESP8266 NodeMCU Module and DHT11 Sensor. It will not only automatically irrigate the water based on the moisture level in the soil but also send the Data to ThingSpeak Server to keep track of the land condition. The System will consist a water pump which will be used to sprinkle water on the land depending upon the land environmental condition such as Moisture, Temperature and Humidity.

**CHAPTERS**

**INTRODUCTION**

Smart Agriculture developing model is a real time monitoring system It monitor the soil properties like temperature, humidity soil moisture PH etc. It is possible to control many operations of the field remotely from anywhere, anytime by IOT. It offers a futuristic way of life in which an individual gets to control his electronic devices using a smart phone, it also offers an efficient use of energy. It applied in all areas of industry, including smart agriculture, smart parking, smart building environmental monitoring, healthcare transportation and many more.

**COMPONENTS REQUIRED**

* NodeMCU ESP8266
* Soil Moisture Sensor Module
* Water Pump Module
* Relay Module
* DHT11
* Connecting Wires

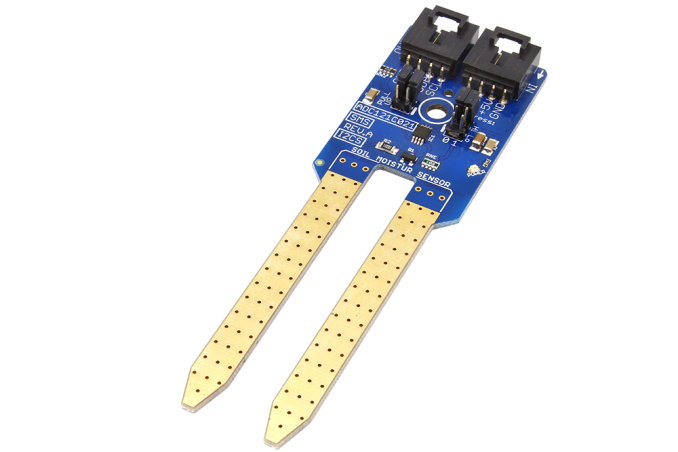
**NodeMCU ESP8266 :**

NodeMCU 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.



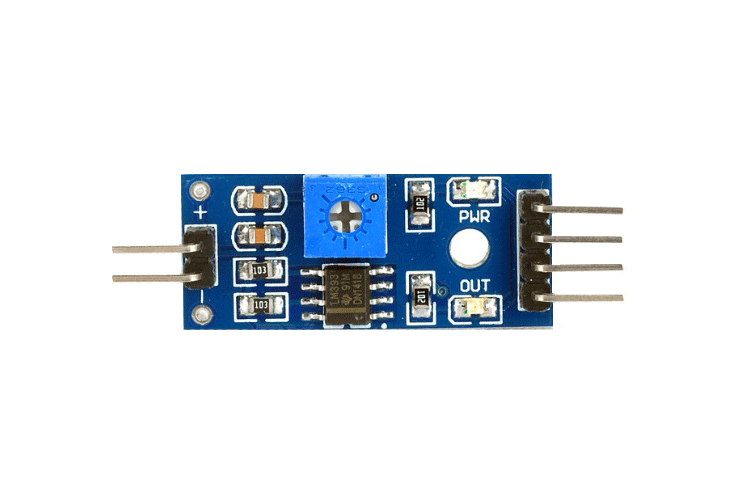
**Soil Moisture Sensor Module :**

**Soil moisture sensors** measure the volumetric water content in soil. A small charge is placed on the electrodes and electrical **resistance** through the sensor is measured. As water is used by plants or as the soil moisture decreases, water is drawn from the sensor and resistance increases. Conversely, as soil moisture increases, resistance decreases.



**LM393 Driver :**

The LM393 series are **dual independent precision voltage comparators capable of single or split supply operation**. These devices are designed to permit a common mode range-to-ground level with single supply operation.



**Water Pump Module :**

The water pump can be defined as a pump which uses the principles like mechanical as well as hydraulic throughout a piping system and to make sufficient force for its future use. They have been approximately in one structure otherwise another because of early civilization. At present these pumps are utilized within a wide range of housing, farming, municipal, and manufacturing applications.



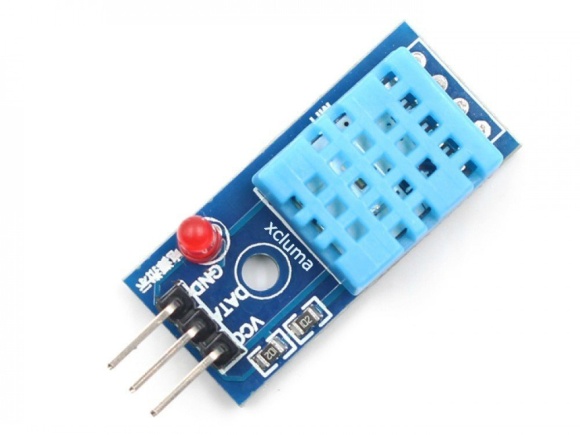
**Relay Module :**

Relays are switches that open and close circuits electromechanically or electronically. Relays **control one electrical circuit by opening and closing contacts in another circuit**. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized.



**DHT11 :**

The DHT11 is a **basic, ultra low-cost digital temperature and humidity sensor**. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).



**LIBRARIES USED**

**#include<DHT.h> :**

Before you can **use** the **DHT11** on the Arduino, you'll need to install the DHTLib **library**. It has all the **functions** needed to get the humidity and temperature readings from the sensor. It's easy to install, just download the DHTLib.

**#include<ESP8266WiFi.h> :**

ESP8266 is all about Wi-Fi. If you are eager to connect your new ESP8266 module to a Wi-Fi network to start sending and receiving data, this is a good place to start. If you are looking for more in depth details of how to program specific Wi-Fi networking functionality, you are also in the right place.

**CODE**

* Since we are using the ThingSpeak Server, the API Key is necessary in order to communicate with server

String apiKey = "6L29ZADH3G4OIN4D"; // Enter your Write API key here

const char\* server = "api.thingspeak.com";

* The next Step is to write the Wi-Fi credentials such as SSID and Password.

const char \*ssid = "D-Link"; // Enter your WiFi Name

const char \*pass = "vasudev123890"; // Enter your WiFi Password

* Define the DHT Sensor Pin where the DHT is connected and Choose the DHT type.

#define DHTPIN D3

DHT dht(DHTPIN, DHT11);

* The moisture sensor output is connected to Pin A0 of ESP8266 NodeMCU. And the motor pin is connected to D0 of NodeMCU.

const int moisturePin = A0;

const int motorPin = D0;

* We will be using millis() function to send the data after every defined interval of time here it is 10 seconds. The delay() is avoided since it stops the program for a defined delay where microcontroller cannot do other tasks.

unsigned long interval = 10000;

unsigned long previousMillis = 0;

* Set motor pin as output, and turn off the motor initially. Start the DHT11 sensor reading.

pinMode(motorPin, OUTPUT);

digitalWrite(motorPin, LOW);

dht.begin();

* Try to connect Wi-Fi with given SSID and Password and wait for the Wi-Fi to be connected and if connected then go to next steps.

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

* Define the current time of starting the program and save it in a variable to compare it with the elapsed time.

unsigned long currentMillis = millis();

* Read temperature and humidity data and save them into variables

h = dht.readHumidity();

t = dht.readTemperature();

* If DHT is connected and the ESP8266 NodeMCU is able to read the readings then proceed to next step or return from here to check again.

if (isnan(h) || isnan(t))

{

Serial.println("Failed to read from DHT sensor!");

return;

}

* Read the moisture reading from sensor and print the reading.

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 the moisture reading is in between the required soil moisture range then keep the pump off or if it goes beyond the required moisture then turn the pump ON.

if (moisturePercentage < 50)

{

digitalWrite(motorPin, HIGH); }

if (moisturePercentage > 50 && moisturePercentage < 55)

{

digitalWrite(motorPin, HIGH);

}

if (moisturePercentage > 56)

{

digitalWrite(motorPin, LOW);

}

* Now after every 10 seconds call the sendThingspeak() function to **send the moisture, temperature and humidity data to ThingSpeak server.**

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

{

sendThingspeak();

previousMillis = millis();

client.stop();

}

* In the sendThingspeak() function we check if the system is connected to server and if yes then we prepare a string where moisture, temperature, humidity reading is written and this string will be sent to ThingSpeak server along with API key and server address.

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";

* Finally the **data is sent to ThingSpeak server** using client.print() function which contains API key, server address and the string which is prepared in previous step.

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");

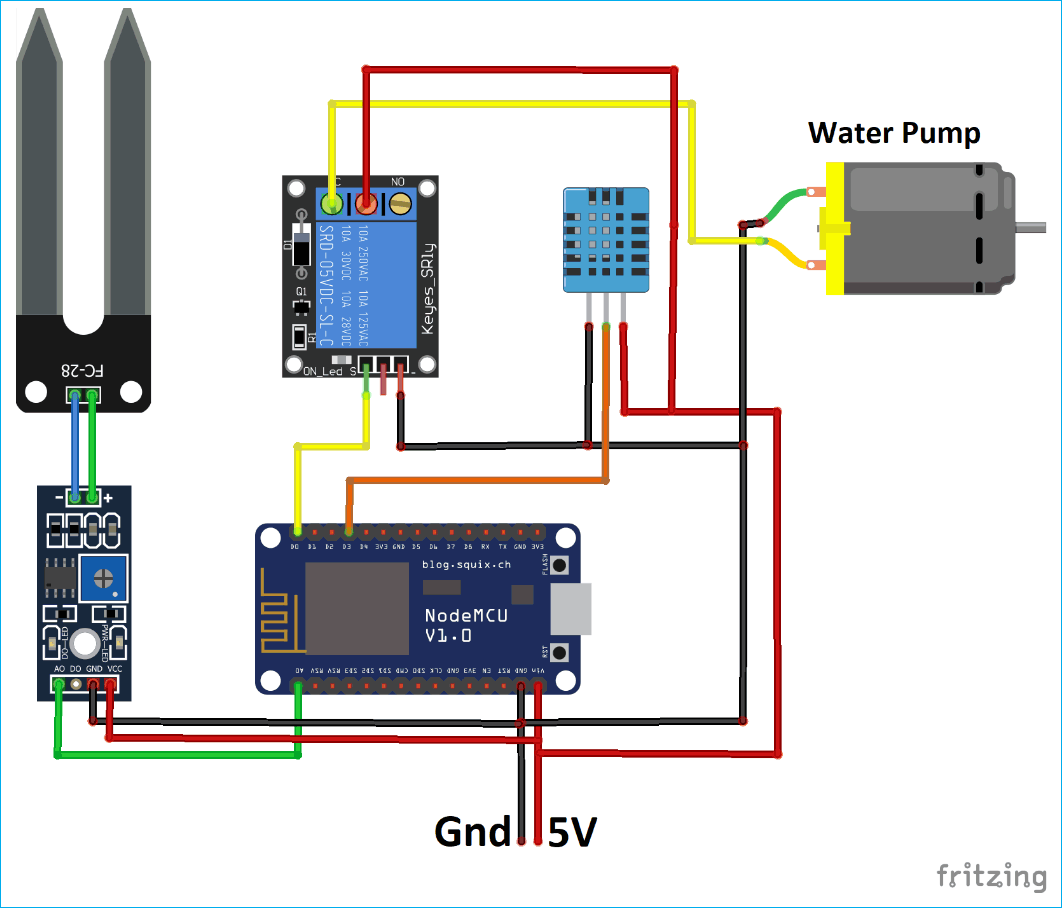
client.print("Content-Length: ");

client.print(postStr.length()); //send lenght of the string

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

client.print(postStr);

**CIRCUIT DIAGRAM**





**CONCLUSION**

The proposed approach of smart garden monitoring is based upon NodeMCU microcontroller, Data Recording, and the internet of things. It provides real-time statistics of farm environmental factors, so the local users and farmers are able to treat their plants in a well manner. Soil temperature, moisture, and relative humidity are considered environmental factors. The result are stored in the form of graph in Thingspeak website. This approach can remedy the problems of gardening that are faced in urban areas due to lack of farmers.