# ABSTRACT FOR SMART IRRIGATION

# SMART IRRIGATION SYSTEM USING IoT


## ABSTRACT

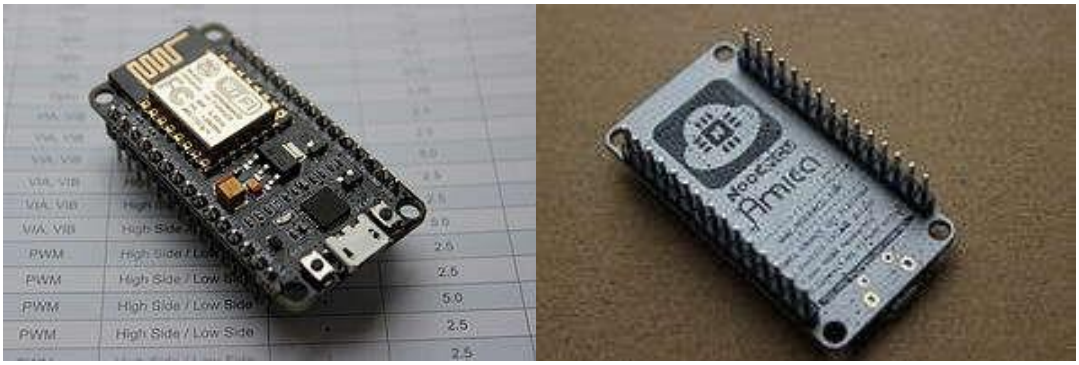
The concept of this project is to allow the owners of fields to control and observe the growth of their plants in their farms. This is achieved by using a smart platform of IoT and solenoid valves to control the flow of water based on the moisture of the soil and gives real time surveillance to the owners who stay far away from the farms. This project also allows surveillance on the personnel and their crops so as to not occur losses. It is easy to use for anyone with a Smartphone and doesn't require maintenance once set up.


## INTRODUCTION

This project has been designed for surveillance of irrigation systems in farms without the need of manual checking of irrigation systems. For example, if you are staying in Bangalore, and have your farm in Andhra Pradesh or elsewhere and it is not possible for you to go to the farms every time to keep a tab on the plants. Instead, this project allows you to check up on your plants using a simple IoT system. The positive part of this project is that, the node used to connect the system to your smart device, also controls the flow of water from the pump and also the timing intervals in between the irrigation cycles. In this paper we will be discussing all about the project as to how it is constructed and how it works.

These are the main components used in this project:

**Node MCU (ESP8266) Wifi Module:** NodeMCU is an advanced API for hardware input/output device which can be dramatically reduces the work for configuring manipulative hardware. It uses a code like Arduino but rather is an interactive script called Lua. It is an open source IoT platform. It runs on a firmware of ESP8266 WiFi Soc produced by Espressif systems. NodeMCU has 16 input/output pins and hence 16 nodes can be connected to a single node. The ESP8266 is Wi-Fi Soc which is integrated with a Tensilica Xtensa LX106 core which is widely used in IoT applications." NodeMCU" refers in default to the firmware rather than the development kits. ESP8266 is an inbuilt WiFi module which can also be used as an individual module as a Wifi module.

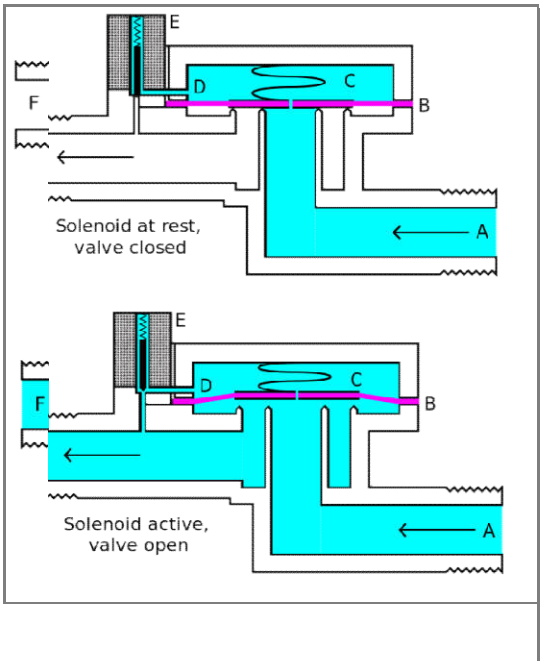



***Meghana Gupta Arakere et al. Smart irrigation system using iot***

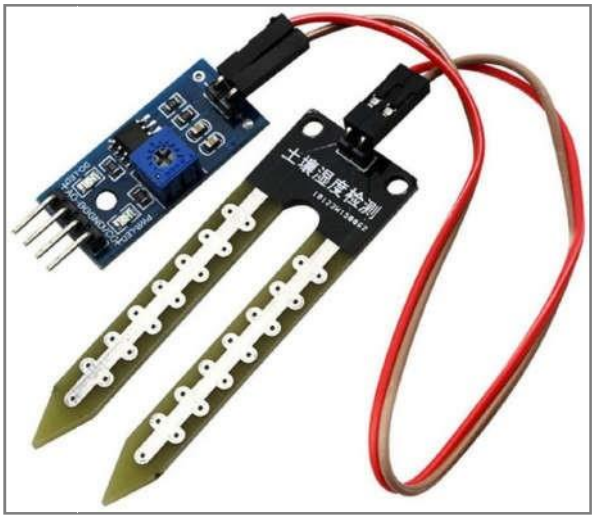
**Solenoid Valve:** A solenoid valve is an electromechanical device in which the solenoid uses an electric current to generate a magnetic field and thereby operate a mechanism which regulates the opening of fluid flow in a valve. Solenoid valves differ in the characteristics of the current they use, the strength of the magnet field they generate, the mechanism they use to regulate the fluid and the characteristics of the fluid they control. The mechanisms vary from linear action plunger type actuators to pivoted armature actuators and rocker actuators. The valve can use a two-port design to regulate a flow or use a three or more port design to switch flows between ports. Multiple solenoid valves can be placed together on a manifold. Solenoid valves are the most frequently used control elements in fluidics. Their tasks are to shut off, release, dose, distribute or mix fluids. They are found in many application

areas. Solenoids offer fast and safe switching, high reliability, long service life, good medium compatibility of the materials used, low control power and compact design.

**Operation:** Ordinary valves can have many ports and fluid paths. A 2-way valve, for example, has 2 ports; if the valve is open, then the two ports are connected and fluid may flow between the ports; if the valve is closed, then ports are isolated. If the valve is open when the solenoid is not energized, then the valve is termed normally open (N.O.). Similarly, if the valve is closed when the solenoid is not energized, then the valve is termed normally closed. There is also 3-way and more complicated designs. A 3-way valve has 3 ports; it connects one port to either of the two other ports (typically a supply port and an exhaust port).Solenoid valves are also characterized by how they operate. A small solenoid can generate a limited force. If that force is sufficient to open and close the valve, then a direct acting solenoid valve is possible.




**Soil Moisture sensor:** Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners. Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.



---

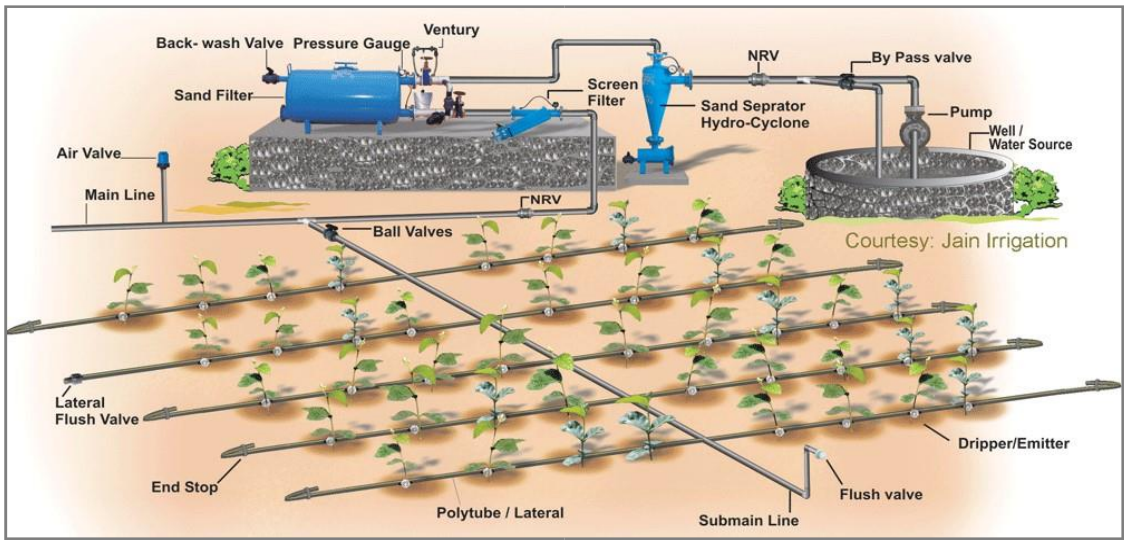
**Drip irrigation system:** Drip irrigation is a type of micro-irrigation system that has the potential to save water and nutrients by allowing water to drip slowly to the roots of plants, either from above the soil surface or buried below the surface. The goal is to

place water directly into the root zone and minimize evaporation. Drip irrigation systems distribute water through a network of valves, pipes, tubing, and emitters. Depending on how well designed, installed, maintained, and operated it is, a drip irrigation system can be more efficient than other types of irrigation systems, such as surface irrigation or sprinkler irrigation.

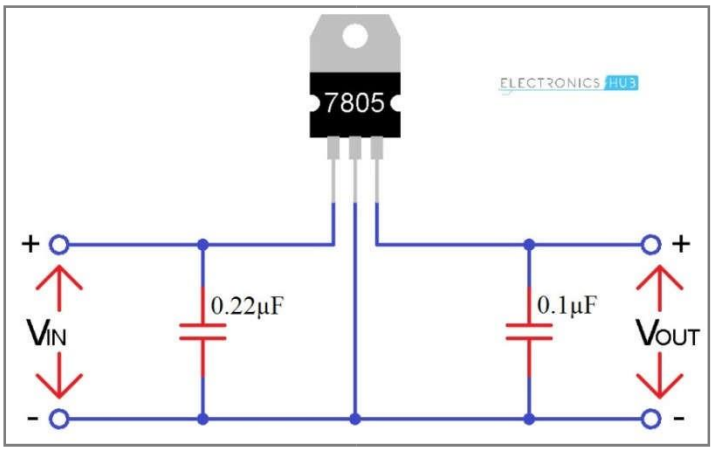
Components used in drip irrigation (listed in order from water source) include:

- Pump or pressurized water source
- Water filter(s) or filtration systems: sand separator, Fertigation systems (Venturi injector) and chemigation equipment (optional)
- Backwash controller (Backflow prevention device)
- Pressure Control Valve (pressure regulator)
- Distribution lines (main larger diameter pipe, maybe secondary smaller, pipe fittings)
- Hand-operated, electronic, or hydraulic control valves and safety valves
- Smaller diameter polyethylene tube (often called "laterals")
- Poly fittings and accessories (to make connections)
- Emitting devices at plants (emitter or dripper, micro spray head, inline dripper or inline drip tube)

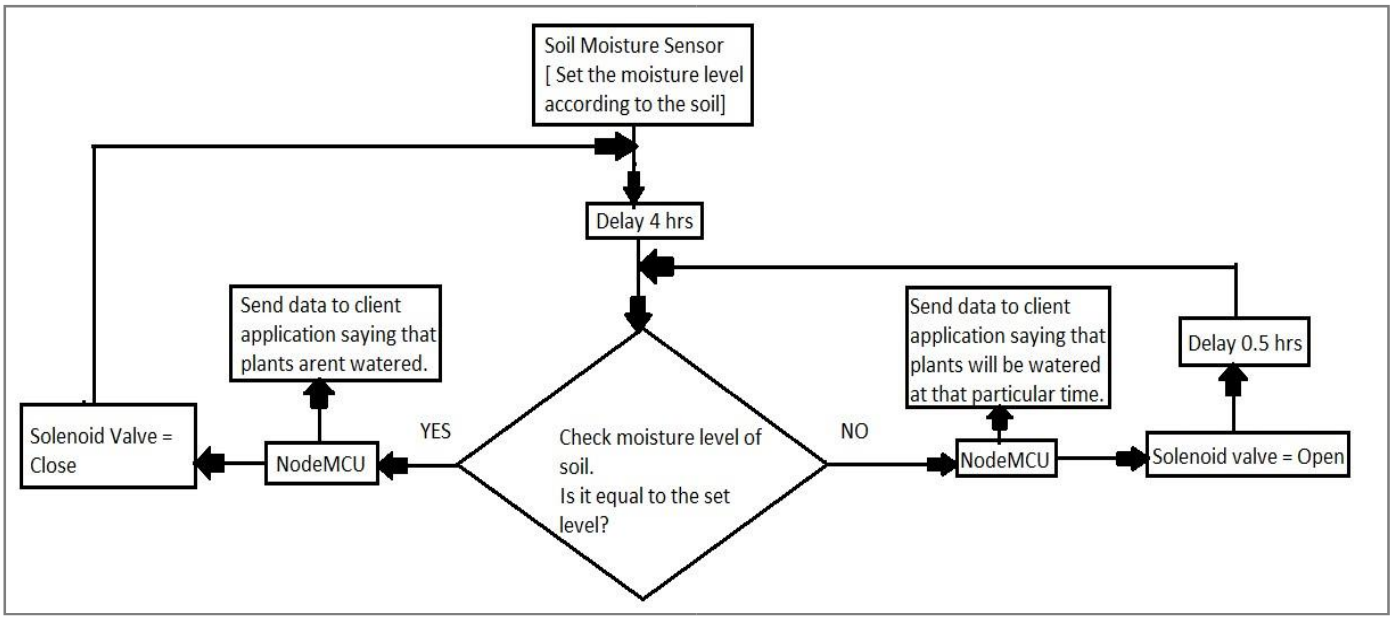
**Operation:** In drip irrigation systems, pump and valves may be manually or automatically operated by a controller.
Most large drip irrigation systems employ some type of filter to prevent clogging of the small emitter flow path by small waterborne particles. New technologies are now being offered that minimize clogging. Some residential systems are installed without additional filters since potable water is already filtered at the water treatment plant. Virtually all drip irrigation equipment manufacturers recommend that filters be employed and generally will not honor warranties unless this is done. Last line filters just before the final delivery pipe are strongly recommended in addition to any other filtration system due to fine particle settlement and accidental insertion of particles in the intermediate lines. Drip and subsurface drip irrigation is used almost exclusively when using recycled municipal wastewater. Regulations typically do not permit spraying water through the air that has not been fully treated to potable water standards. Because of the way the water is applied in a drip system, traditional surface applications of timed-release fertilizer are sometimes ineffective, so drip systems often mix liquid fertilizer with the irrigation water. This is called fertigation; fertigation and chemigation (application of pesticides and other chemicals to periodically clean out the system, such as chlorine or sulfuric acid) use chemical injectors such as diaphragm pumps, piston pumps, or aspirators. The chemicals may be added constantly whenever the system is irrigating or at intervals. Fertilizer savings of up to 95% are being reported from recent university field tests using drip fertigation and slow water delivery as compared to timed-release and irrigation by micro spray heads. Properly designed, installed, and managed, drip irrigation may help achieve water conservation by reducing evaporation and deep drainage when compared to other types of irrigation such as flood or overhead sprinklers since water can be more precisely applied to the plant roots. In addition, drip can eliminate many diseases that are spread through water contact with the foliage. Finally, in regions where water supplies are severely limited, there may be no actual water savings, but rather simply an increase in production while using the same amount of water as before. In very arid regions or on sandy soils, the preferred method is to apply the irrigation water as slowly as possible. Pulsed irrigation is sometimes used to decrease the amount of water delivered to the plant at any one time, thus reducing runoff or deep percolation. Pulsed systems are typically expensive and require extensive maintenance. Therefore, the latest efforts by emitter manufacturers are focused on developing new technologies that deliver irrigation water at ultra-low flow rates, i.e. less than 1.0 liter per hour. Slow and even delivery further improves water use efficiency without incurring the expense and complexity of pulsed delivery equipment. An emitting pipe is a type of drip irrigation tubing with emitters pre-installed at the factory with specific distance and flow per hour as per crop distance. An emitter restricts water flow passage through it, thus creating head loss required (to the extent of atmospheric pressure) in order to emit water in the form of droplets. This head loss is achieved by friction/turbulence within the emitter.



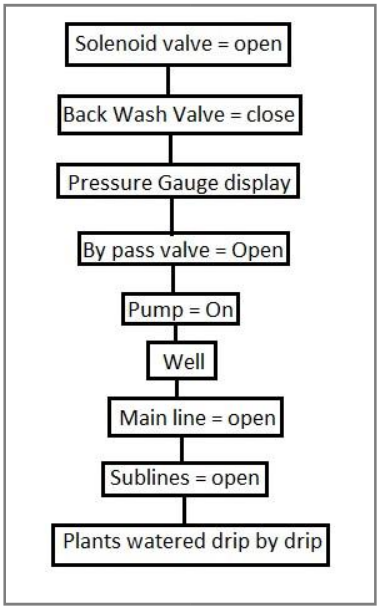
**Voltage Regulator (7805):** 7805 Voltage Regulator IC is a regulated power supply is very much essential for several electronic devices due to the semiconductor material employed in them have a fixed rate of current as well as voltage. The device may get damaged if there is any deviation from the fixed rate. One of the important sources of DC Supply is Batteries. But using batteries in sensitive electronic circuits is not a good idea as batteries eventually drain out and lose their potential over time. Also, the voltages provided by batteries are typically 1.2V, 3.7V, 9V and 12V. This is good for circuits whose voltage requirements are in that range. But, most of the TTL IC's work on 5V logic and hence we need a mechanism to provide a consistent 5V Supply. Here comes the 7805 Voltage Regulator IC to the rescue. It is an IC in the 78XX family of linear voltage regulators that produce a regulated 5V as output. 7805 is a three terminal linear voltage regulator IC with a fixed output voltage of 5V which is useful in a wide range of applications. We will be using the 7805 to reduce the 220V to 5V for the NodeMCU.




**Working of this Project:** In this project, the soil moisture sensors are placed at every few feet with the probes in the ground. The soil moisture sensor is set to a particular level based on the type of soil and the crops being grown. This project is useful for crops that need constant irrigation in small amounts of water. Every 4 hours, the soil moisture is checked through the sensor. The sensors value is then sent to the NodeMCU or the WiFi module. Since this acts like an arduino but can be connected to the WiFi, the sensor reading are checked and matched to the set value of reading. If the value of the soil moisture sensor is equal to the set value, then a command is sent to the solenoid valve to close. A message is then sent to the client's mobile or app which is connected to the NodeMCU saying that the plants haven't been watered because the soil moisture level is already equal to the set value. The solenoid valve is generally closed. The moisture is then checked after 4 hours. If the moisture of the soil is less, then the NodeMCU sends a command to the solenoid valve to open. When the solenoid valve is opened, the drip irrigation system starts to drip the water at the roots. It allows water for half an hour and after half an hour; the moisture level is again checked and sent to the NodeMCU. A message is sent to the client device saying that the plants have been watered at that particular time. The process then repeats all over. The intention of this project is to provide the clients far away from the fields, a chance to keep an eye on their plants.



Once the solenoid valve is open, the back wash valve is closed, the pressure gauge starts fluctuating as the pressure is built up to pump the water. Once the particular pressure is obtained, the by-pass valve opens and pump gets switched on. The pump then pumps out water from the well and the water flows through the main line and the sub-lines and the water is delivered to the roots of the plant, drop by drop. When the solenoid valve closes, all these processes are shut off hence shutting off the water supply to the plants.

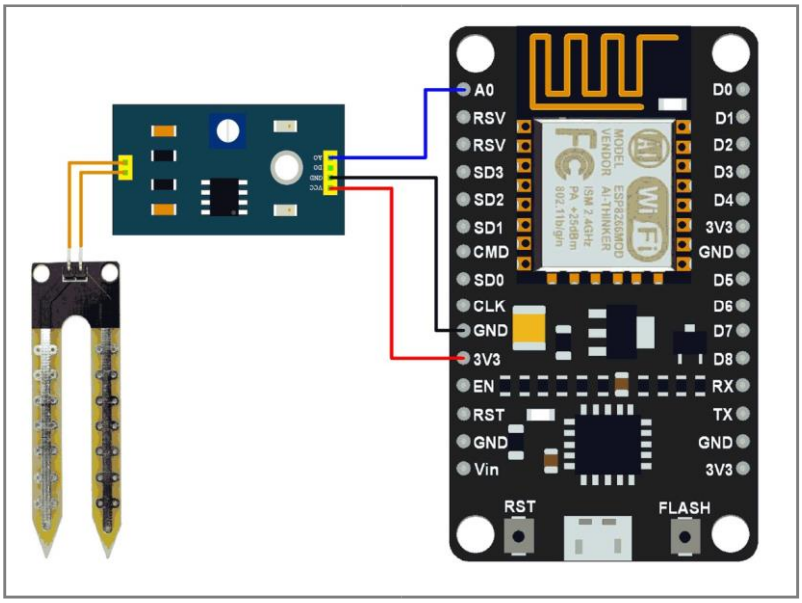



**Node MCU connection with smart device (MQTT Box)**

We will be using MQTT box to connect the solenoid valve as well as the soil moisture sensor. MQTT is lightweight publishsubscribe based messaging protocol.

- It is quicker (faster) than other request-response based APIs like HTTP.
- It is developed on the base of TCP/IP protocol.
- It allows remote location devices to connect, subscribe, and publish etc. to a specific topic on the server with the help of message broker.
- MQTT Broker/Message broker is a module in between the sender and the receiver. It is an element for message validation, transformation and routing.
- The broker is responsible for distributing messages to the interested clients (subscribed clients) of their interested topic.

**Method of connecting MQTT box to NodeMCU**



Here, the analog output of soil moisture sensor is processed using ADC. The moisture content in terms of percentage is displayed on the serial monitor. The output of the soil moisture sensor changes in the range of ADC value from 0 to 1023. This can be represented as moisture value in terms of percentage using formula given below.

Analog Output = ADC Value / 1023
Moisture in percentage = 100 – (Analog output * 100)

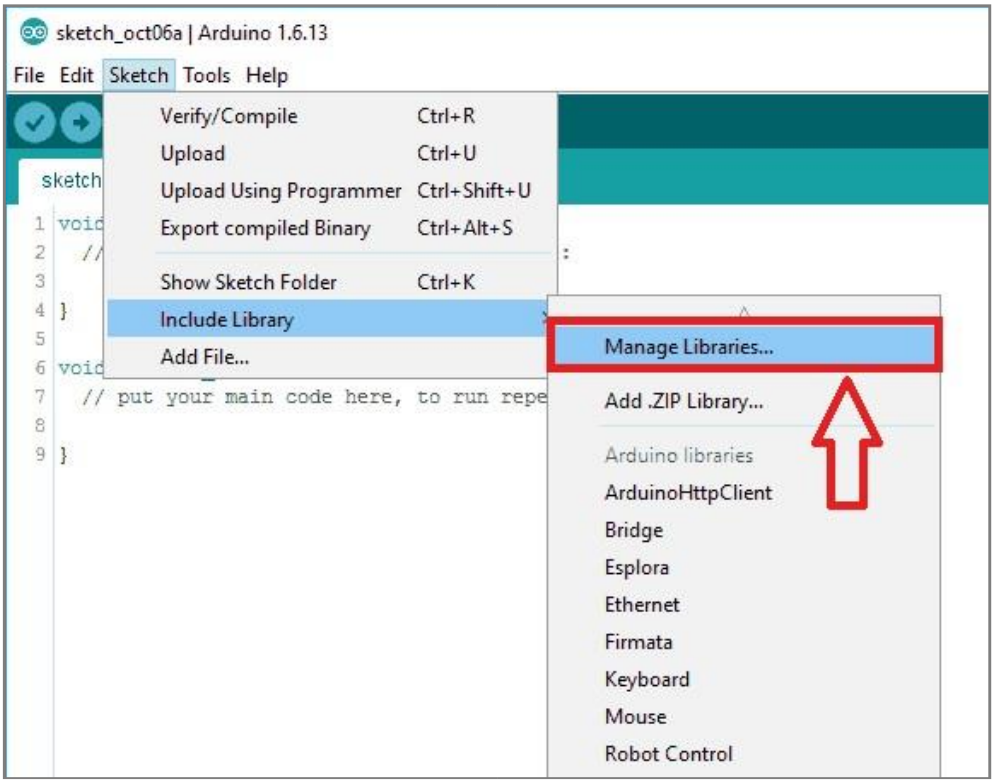
For zero moisture, we get maximum value of 10-bit ADC, i.e. 1023. This in turn gives ~0% moisture.

***Steps:***

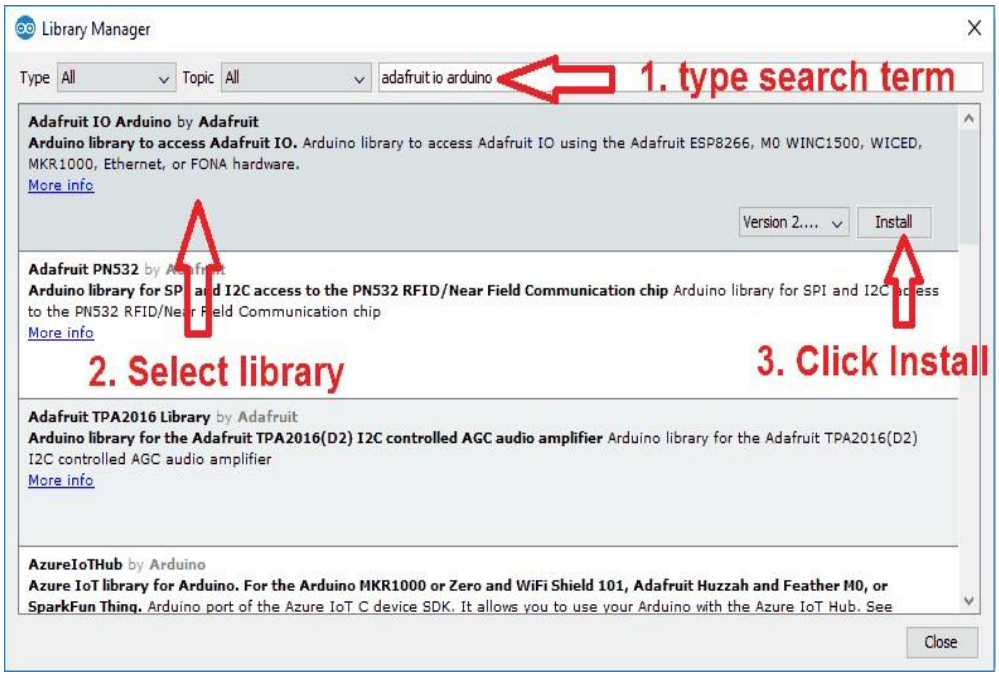
Install required libraries

First refer Getting Started with NodeMCU using Arduino IDE if you are not installed NodeMCU board packages in Arduino IDE. Here we are using Adafruit libraries for above example. We will need to install the Adafruit IO**,** Adafruit MQTT**,** and Arduino Http Client libraries using the Arduino Library Manager.

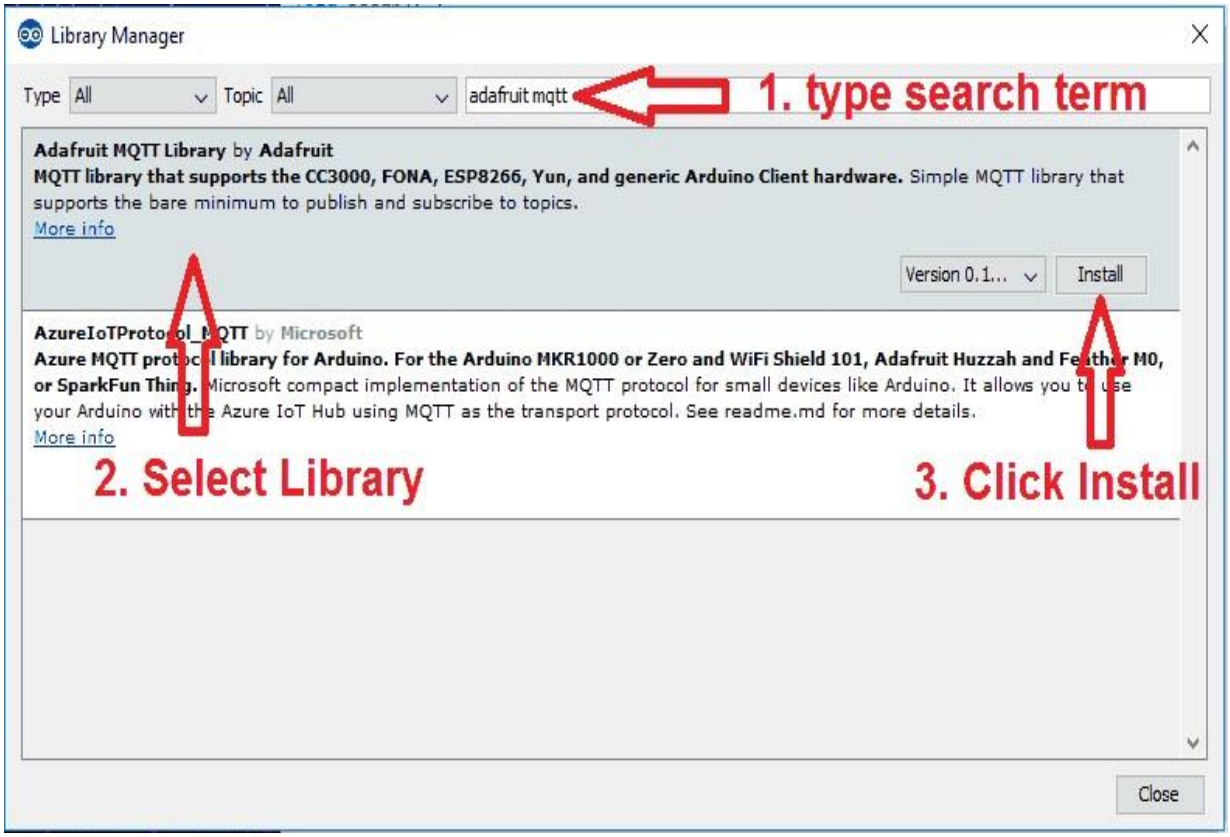
Open the Arduino IDE and navigate to **Sketch -> Include Library -> Manage Libraries…**



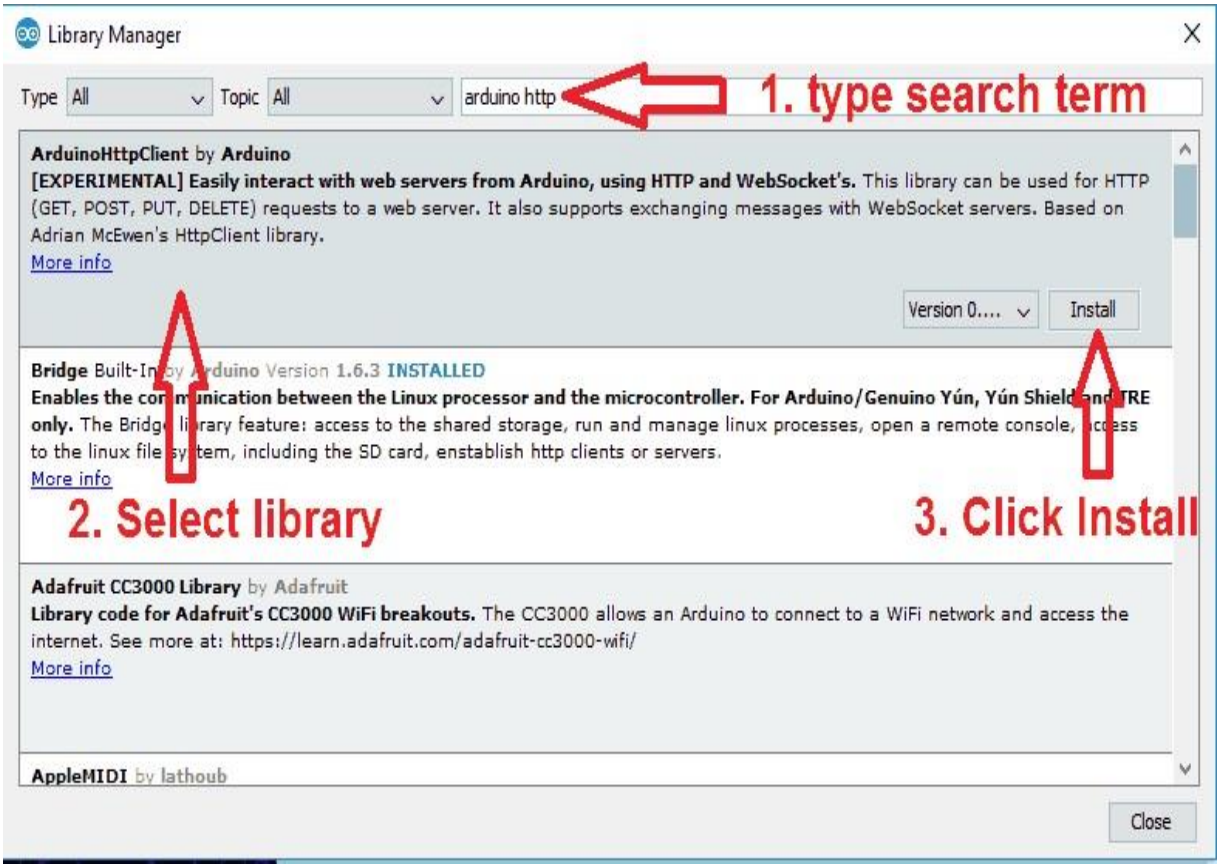

Library Manager Window will pop up. Now enter **Adafruit IO Arduino** into the search box, and click Install on the **Adafruit IO Arduino library** option to install version 2.6.0 or higher.



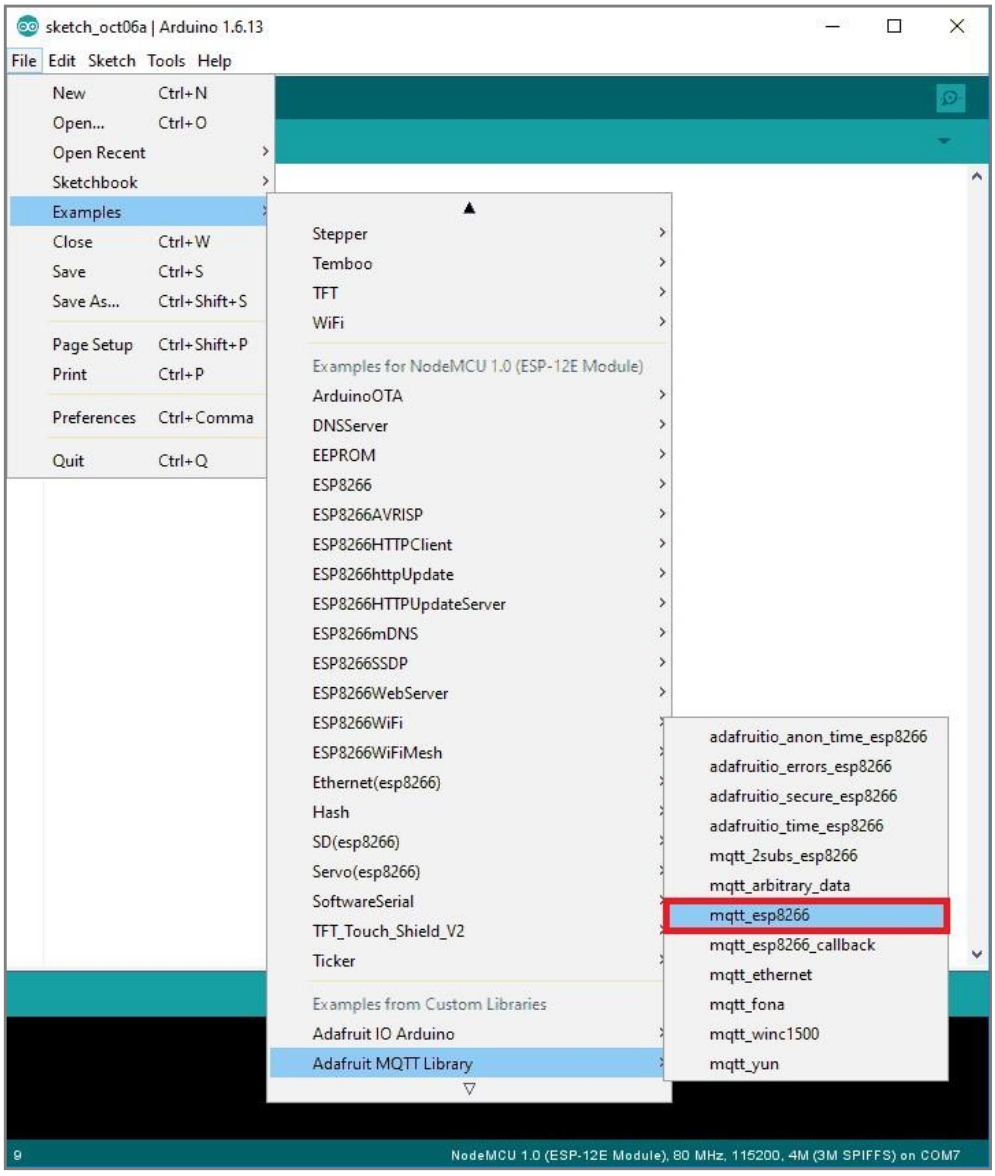

Now enter **Adafruit MQTT** into the search box, and click Install on the **Adafruit MQTT library**option to install version 0.17.0 or higher.



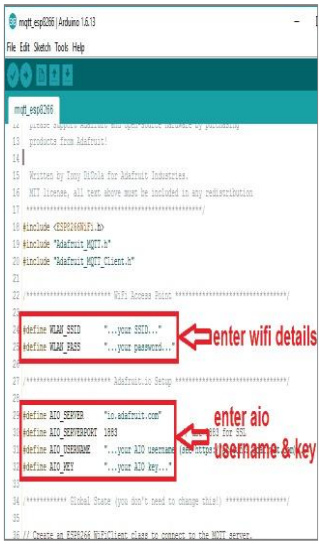

Now enter **Arduino Http Client** into the search box, and click Install on the **ArduinoHttpClient**library option to install version 0.3.0 or higher.




Now open example of Adafruit mqtt Io dashboard. To open it navigate to **File -> Examples -> Adafruit MQTT Library -> mqtt_esp8266**




Now edit the Wifi and Adafruit Io credentials with correct information of example as shown in below image.




We have modified mqtt_esp8266 example as per our above example as below
Arduino Sketch for MQTT Client

```
/************************************************
 Adafruit MQTT Library ESP8266 Example

 Must use ESP8266 Arduino from:
https://github.com/esp8266/Arduino

 Works great with Adafruit's Huzzah ESP board & Feather
 ----> https://www.adafruit.com/product/2471
 ----> https://www.adafruit.com/products/2821

 Adafruit invests time and resources providing this open source code,
Please support Adafruit and open-source hardware by purchasing
Products from Adafruit!

 Written by Tony DiCola for Adafruit Industries.
 MIT license, all text above must be included in any redistribution
 ************************************************/
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID    "...your SSID..."
#define WLAN_PASS    "...your password..."

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER   "io.adafruit.com"
#define AIO_SERVERPORT 1883          // use 8883 for SSL
#define AIO_USERNAME  "...your AIO username (see https://accounts.adafruit.com)..." #define
AIO_KEY     "...your AIO key..."

/************ Global State (you don't need to change this!) ******************/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiFlientSecure for SSL
//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/

// Setup a feed called 'potValue' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish potValue = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/potValue");

// Setup a feed called 'ledBrightness' for subscribing to changes.
Adafruit_MQTT_Subscribe ledBrightness = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/ledBrightness");

/*************************** Sketch Code ************************************/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration //
for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();

uint8_t ledPin = D6; uint16_t
potAdcValue = 0;
```

```
uint16_t ledBrightValue = 0;

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

 Serial.println(F("Adafruit MQTT demo"));

 // Connect to WiFi access point.
 Serial.println(); Serial.println();
 Serial.print("Connecting to ");
 Serial.println(WLAN_SSID);

 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
delay(500);
  Serial.print(".");
 }
 Serial.println();

 Serial.println("WiFi connected");
 Serial.println("IP address: "); Serial.println(WiFi.localIP());

 // Setup MQTT subscription for ledBrightness feed.
mqtt.subscribe(&ledBrightness);
}

void loop() {
 // Ensure the connection to the MQTT server is alive (this will make the first
 // connection and automatically reconnect when disconnected). See the MQTT_connect
// function definition further below.
 MQTT_connect();

 // this is our 'wait for incoming subscription packets' busy subloop
// try to spend your time here

 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription(200))) {
  if (subscription == &ledBrightness) {
Serial.print(F("Got LED Brightness : "));
   ledBrightValue = atoi((char
*)ledBrightness.lastread);
Serial.println(ledBrightValue);    analogWrite(ledPin,
ledBrightValue);
  }
 }

 // Now we can publish stuff!
 uint16_t AdcValue = analogRead(A0);
 if((AdcValue > (potAdcValue + 7)) || (AdcValue < (potAdcValue - 7))){   potAdcValue
= AdcValue;
  Serial.print(F("Sending pot val "));
  Serial.print(potAdcValue);
  Serial.print("...");
  if (! potValue.publish(potAdcValue)) {
   Serial.println(F("Failed"));
  } else {
   Serial.println(F("OK!"));
  }
 }
 // ping the server to keep the mqtt connection alive
 // NOT required if you are publishing once every KEEPALIVE seconds
```

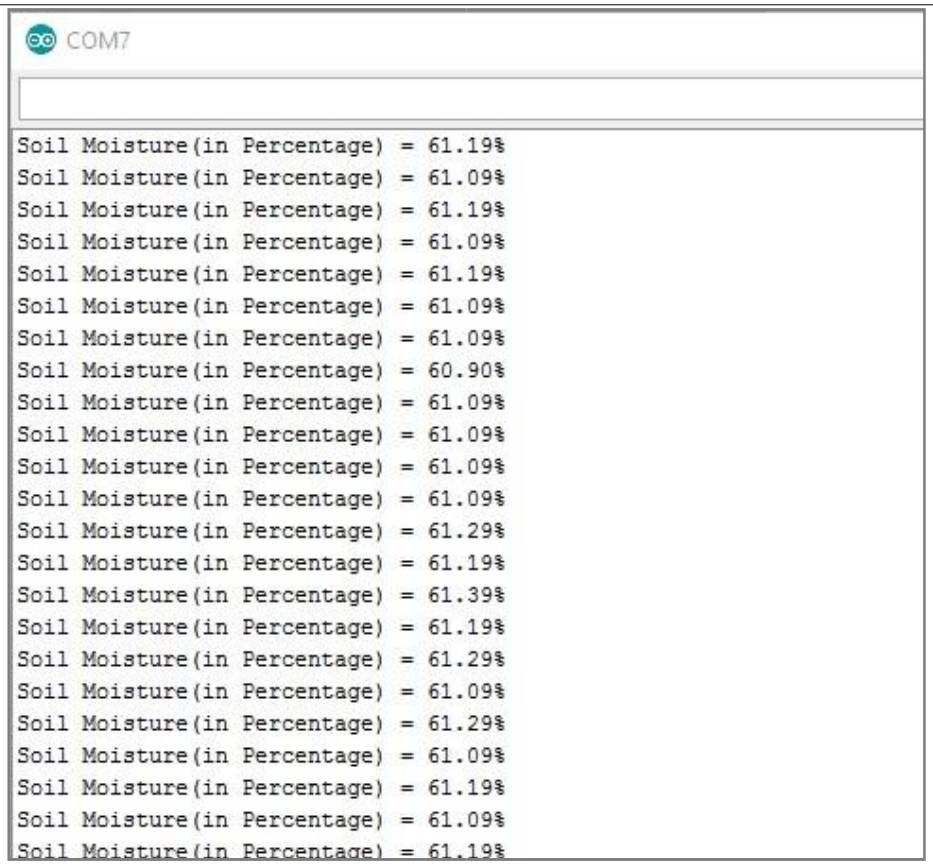
```
 /*  if(!
mqtt.ping()) {
mqtt.disconnect();
```

```
 }
 */
}

// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
 int8_t ret;

 // Stop if already connected.
 if (mqtt.connected()) {
  return;
 }

 Serial.print("Connecting to MQTT... ");

 uint8_t retries = 3;
 while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 seconds...");
    mqtt.disconnect();
    delay(5000); // wait 5 seconds
    retries--;
    if (retries == 0) {
     // basically die and wait for WDT to reset me
     while (1);
    }
 }
 Serial.println("MQTT Connected!");
}
```

Arduino Sketch for Soil Moisture

```
const int sensor_pin = A0; /* Connect Soil moisture analog sensor pin to A0 of NodeMCU */

void setup() {
 Serial.begin(9600); /* Define baud rate for serial communication */
}

void loop() {
 float moisture_percentage;

 moisture_percentage = ( 100.00 - ( (analogRead(sensor_pin)/1023.00) * 100.00 ) );

 Serial.print("Soil Moisture(in Percentage) = ");
 Serial.print(moisture_percentage);
 Serial.println("%");

 delay(1000);
}
```

Arduino Serial Output Window
Arduino Serial monitor output window for Soil Moisture




Arduino Sketch for Solenoid Valve

```
#include "CayenneDefines.h"
#include "CayenneWiFi.h"
#include "CayenneWiFiClient.h"
#define CAYENNE_PRINT Serial // Comment this out to disable prints and save space
#define RELAY_PIN 14 // RELAY PIN

const int sensorPin= A0; //sensor pin connected to analog pin A0
float liquid_level; int liquid_percentage; int top_level = 512;
int bottom_level = 3;

// Cayenne authentication token. This should be obtained from the Cayenne Dashboard.
char token[] = ""; // Insert your token here char
ssid[] = ""; // Insert your SSID here char pwd[] =
""; // Insert your SSID password here

void setup() {  Serial.begin(115200);
 Cayenne.begin(token, ssid, pwd);  pinMode(sensorPin,
INPUT);
 pinMode(RELAY_PIN, OUTPUT);
}

void loop() {
 liquid_level = analogRead(sensorPin);
 liquid_percentage = ((liquid_level-bottom_level)/top_level)*100;
 Serial.println(liquid_level);
delay(100);  Cayenne.run();
```



```
CAYENNE_OUT(V10)
{
  Cayenne.virtualWrite(V10, liquid_percentage);
}
```

```
CAYENNE_IN(V1)
{
 // get value sent from dashboard
 int currentValue = getValue.asInt(); // 0 to 1

 // assuming you wire your relay as normally open  if
(currentValue == 0) {
  digitalWrite(RELAY_PIN, HIGH);
 } else {
  digitalWrite(RELAY_PIN, LOW);
 }
}
```

**Conclusion**

Thus this project has a very useful application for all those people who have farms and like to contribute to the agriculture of country but are poised with a lack of personnel and lack of time from their daily duties. This project also allows surveillance on the personnel and their crops so as to not occur losses. It is easy to use for anyone with a Smartphone and doesn't require maintenance once set up.

## REFERENCES


1. https://www.instructables.com/id/IoT-Water-Control-and-Monitor-Using-NodeMCU-Cayenn/
2. https://www.electronicwings.com/nodemcu/nodemcu-mqtt-client-with-arduino-ide
3. https://www.electronicwings.com/nodemcu/soil-moisture-sensor-interfacing-with-nodemcu
4. https://www.omega.co.uk/prodinfo/solenoid-valve.html
5. https://tameson.com/solenoid-valve-types.html
6. http://www.circuitstoday.com/arduino-soil-moisture-sensor
7. https://www.nodemcu.com/index_en.html
8. https://www.instructables.com/id/IoT-Water-Control-and-Monitor-Using-NodeMCU-Cayenn/
9. https://www.instructables.com/id/Interface-Moisture-Sensor-With-NodeMCU/
10. https://www.jameco.com/Jameco/workshop/learning-center/voltage-regulator.html


*******