Soil Moisture Detecting and Watering System Using AWS IoT

Department of Computer Science, Western Michigan University, Kalamazoo - 49006, Michigan, USA;

Meghana Gadde, Sampath Talluri
meghana.gadde@wmich.edu, Sampath.talluri@wmich.edu

Abstract

Now a day's conservation of water is the main moto. We need water for our day to day activities, irrigation…etc., Farmers need more water for cultivation. If we can conserve water and supply adequately, we can get more crop yield with less water. The main objectives of this project is to design and execute an open source and cost effective automatic irrigation system that's capable of supplying the water to the fields whenever required. At the same time storing the details of the system into the database on an AWS, lets us to analyze and give guidance to farmer which crop can be cultivated with less water or existing water resource. The system contains a great elasticity by using wireless reliable technology to interconnecting various modules to the server of automatic irrigation system. This system allows us to analyze the seasonal water requirement for different crops and can reduce wastage of water.

1.Introduction

In this paper, using Cloud IoT we have designed and developed a system which precisely measure the soil moisture levels, salinity and temperature as they play a vital role in agriculture production. Using the wi-fi sensors, we can observe these parameters at real-time. This methodology falls under the study of internet of Things (IoT). In the past days, the ON and OFF of the water delivery motor within the irrigation system was controlled by switches and timers. In the current day wireless technology makes it simple to observe and control the agriculture parameters using IOT technology.

Different parameters are monitored in real time using IoT without any delay. IoT gives the information in all sectors of agriculture, healthcare, home appliances[1] etc. Cloud computing provides security, monitoring and maintenance of data. The major use of the Cloud technology is an efficient use of resources and cost reduction. The IoT connects things to the internet for communicating with the sensing devices with suitable protocols and exchanging data with each other by using wireless sensor networks[2].

2. Communication Technology

Out of many sensing units available in the wireless sensor network, one of them is developed in this paper. The sensing unit basically collects the information from the sensor and communicates with each other with the help of a gateway unit and then send the measured data to a cloud for further processing. Node MCU has on-chip wi-fi and will get IP address from the connection to the router.

The wi-fi on Node MCU provides approximately a order of 20-100m communication range with 2-54mbps transmission rate at 2.4GHz ISM band frequency.

3. IoT system for soil moisture monitoring

Microcontroller is a small and low-cost micro-computer and is designed to perform specific tasks of embedded systems like receiving remote signals.
The following materials were used in this project with IoT concepts:

Hardware Requirements

Microcontroller : Node MCU

Sensors : Soil Moisture Sensor

Actuators : Single Chanel Relay, Water Motor

Others : Bread Board, Jumper Cables, 9V Battery

Software Requirements

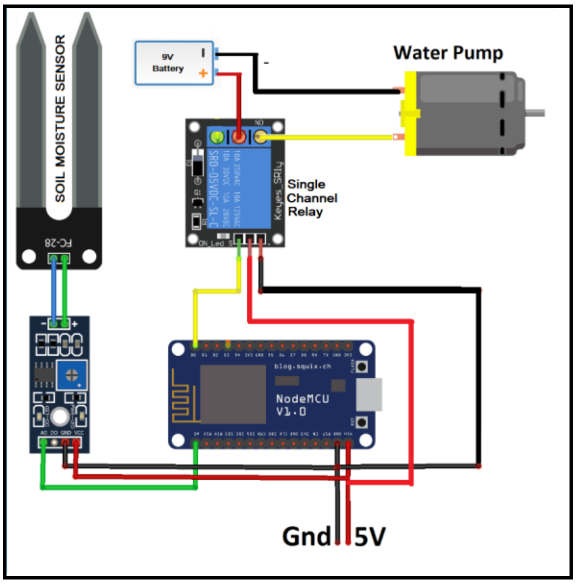
Operating System : Windows 7, 8 or 10

Language : Arduino Programming, PHP

Data Base : MySQL Server on AWS

Developer Tools : Arduino IDE, XAMP web server,

MySQL Workbench, AWS Educate Account



Circuit Diagram

3.1 NodeMCU V1.0:

NodeMCU has a built-in support for wi-fi connectivity which is an open source software and hardware development environment that is build around a very inexpensive System-on-a-chip called the ESP8266-12E chips.

This development board provides access to the GPIO(General purpose Input/Output) subsystem. ESP8266 is mostly used for development of IoT embedded applications.

.

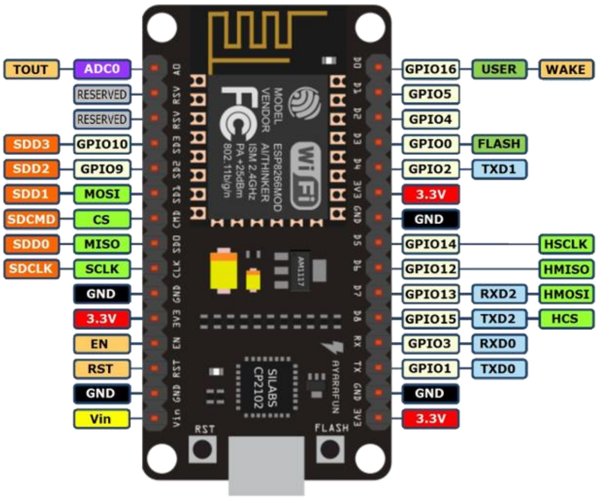


Figure 1. NodeMCU Board

The following table gives the comparison of some of the ESP8266 modules:

FEATURE	NodeMCU V0.9	NodeMCU V1.0	Wemos D1 mini	WemosD1 R2
Branded	NodeMCU	DOIT	Wemos	Wemos
GPIO pins	11	11	11	11
ADC	1	1	1	1
Antenna	PCB	PCB	PCB	PCB
ESP8266 module	ESP12 (AI-Thinker)	ESP12E (AI-Thinker)	ESP12E (Wemos)	ESP12E (Wemos)
USB to Serial	Yes	Yes	Yes	Yes
Serial Chip	CH340G	CH2102/CH340G	CH340G	CH340G
Breadboard friendly	Bad(covers many pins)	Very Good	Very Good (after Soldering)	Bad
Form factor	Big	Big	Medium	Very Big
Price	~ $6.40	~ $6	~ $6	~ $6.50
Application	Development Beginner	Development Beginner	Development Advanced (soldering Required)	Development Form compati-bilty with arduino shields

3.2 The soil moisture sensor:

The hygrometer or soil moisture sensor (Figure 2) consists mainly of two rods that are put into the soil to measure the moisture level and of a comparator circuit that returns the level of soil conductivity.

This circuit consists of six pins in total, where four pins have the functions of feeding the circuit and returning the soil moisture level and , two pins which are used to connect the comparison circuit to metal rod.

The soil moisture level can be determined using two ways:

1. An analog signal that can be used to estimate how humid the soil is.

2.A digital signal that basically informs if the ground is dry or not. ADC is an electronic device capable of generating a digital representation from an analog quantity, usually a signal represented by a voltage level or electric current.

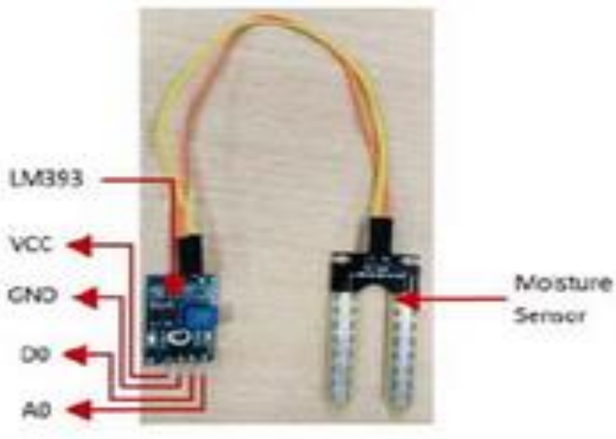


Figure 2. Soil moisture sensor (hygrometer) [20].

The following table gives the PIN Definitions of Soil Moisture Sensor:

PIN	DEFINITION
VCC	Power supply
GND	Ground
D0	Digital Output inter-face(0 or 1)
A0	Analog Output in-terface

3.3 The water motor:

The water motor (Figure 3) used for soil irrigation is made of injectable plastic and it is powered by direct current (DC). It is used to release water to the plants depending on the soil moisture levels. This motor is connected to relay which is an output pin of the microcontroller, which is an electrically driven switch. This relay is responsible for activating the water pump.

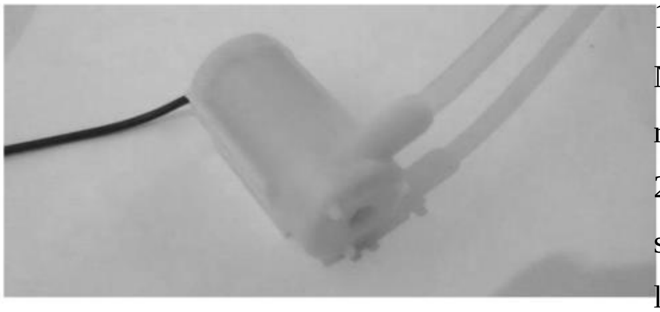


Figure 3. Submerged water pump.

3.4 Single channel Relay Module

In this project, we use a 5v single channel relay (figure 4). Relay is an electrically driven switch which is used to connect or disconnect a source. Here we used the relay to ON and OFF the water motor depending on the soil moisture level.

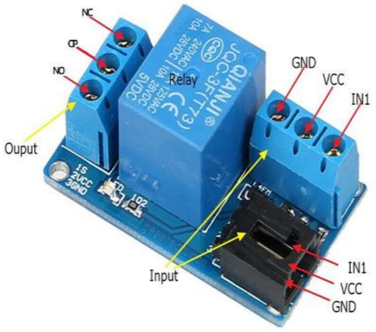


Figure 4. Single Channel Relay4

4. Design Considerations

1. Connect the Node MCU and XAMP Server to same WiFi Network.
2. Change WiFi Password in Arduino code.
3. Change the AWS MySQL Server Credentials in PHP Code.

5. Challenges Faced

1.When connecting the AC supply, the NodeMCU has short circuited and single channel relay has short circuited.

2.When sending code from NodeMCU sensor to server, a single spacebar at wrong place causes a lot of errors

3.When connecting AC supply of 110/240 volts, we faced a problem of motor disfunction

6. Conclusion and Future Scope

The combination of IoT microcontroller and AWS made it possible to develop a versatile IoT project using low-cost embedded systems. With the proposed project, the use of ESP8266 applied in a real-world problem became satisfactory, since with small lines of code, numerous applications can be achieved, opening several possibilities for future projects mainly with concepts of Internet of Things (IoT).

7. References

1. Zanella A, Bui N, Castellani A, Vangelista L,Zorzi M.
Internet of things for smart cities. IEEE Internet of things
Journal. 2014; 1(1):22–32.
2. Ghamri-Doudance Y, Minerva R. Guest editorial special issue on world forum on Internet of things Conference 2014. IEEE Internet of things Journal. 2015 Jun; 2(3):187–9.
3. Kantarci B, Mouftah HT. Trustworthy sensing for public safety in cloud-centric Internet of things. IEEE Internet of things Journal. 2014 Aug; 1(4):360–8.
4.Jucá S. and Pereira, R. “Practical applications of microcontrollers using free software: Practically
5.NodeMcu (2018). NodeMcu –An open-source firmware based on ESP8266wifi-soc. Available in:

http://www.nodemcu.com/index_en.html#fr_5474
7361d775ef1a3600000f. [AccessedMarch 2018].
6.https://www.geeetech.com/wiki/index.php/One-Channel_Relay_module
7.https://www.researchgate.net/publication/316727718_An_IoT_based_soil_moisture_monitoring_on_Losant_platform
8.https://pdfs.semanticscholar.org/9596/768538051e58eb1457c3d8f703aa640e62bc.pdf
9.https://www.researchgate.net/publication/307173266_Measurement_and_Monitoring_of_Soil_Moisture_using_Cloud_IoT_and_Android_System
10.https://forum.arduino.cc/index.php?topic=642612.0

8. Appendices

Board connection:

We connect jumper cable from AO pin in NodeMCU to AO pin in soil sensor

We connect jumper cable from D0 pin in NodeMCU to INN pin in single channel relay of input side

We connect jumper cable from 3.3V pin in NodeMCU to Vcc pin in soil sensor

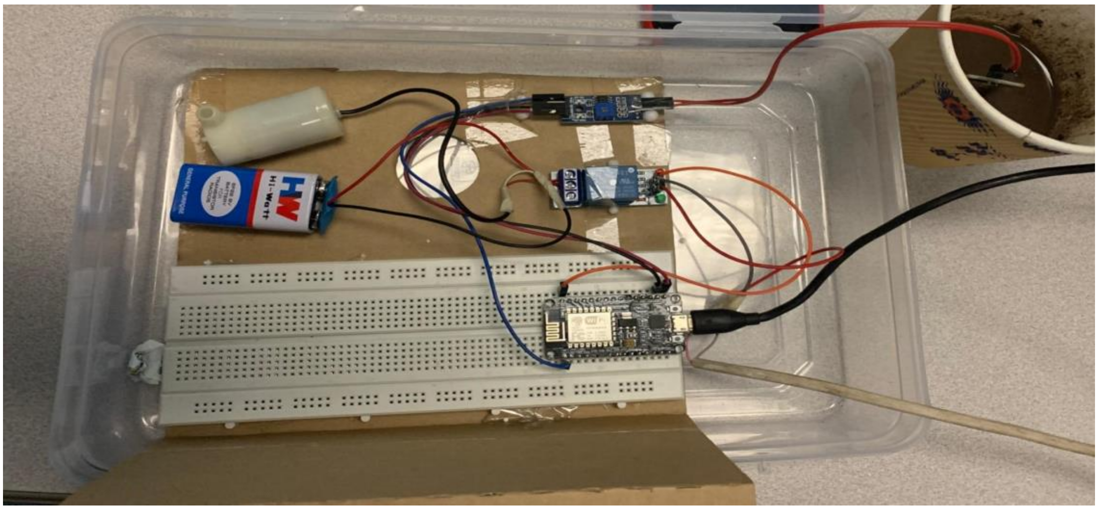
We connect jumper cable from GND pin in NodeMCU to GND pin in soil sensor

We connect jumper cable from GND and VCC pin in Single channel relay to Usb cable

We connect battery positive to CP in single channel relay of output side

We connect battery negative to motor negative

We connect motor positive to NC in single channel relay of output side



Insert code:

First, we connect all the components in the board

Then we connect usb cable to NodeMCU the other end to the computer

We should have Arduino software installed in your system and we have a Arduino structure to upload the code to NodeMCU

We just change the ssid, password and IP to connect to wifi.
Once the code is uploaded we go to tools->screen monitoring in Arduino software we get a output which Return IP address and soil moisture value and -1

Then we power supply the single channel relay to computer upLoad | Arduino 1.8.10

File Edit Sketch Tools Help

upLoad

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>

int Led_OnBoard = 2;                  // Initialize the Led_OnBoard
const int motorPin = D0;              // relay Output pin
const char* ssid = "MySpectrumWiFi68-2G";       // Your wifi Name
const char* password = "classywolf998";     // Your wifi Password
const char *host = "192.168.1.22";  //Your pc or server (database) IP, if you are a windows os user, open cmd, then type ipconfig then look at IPv4 A

void setup() {
  // put your setup code here, to run once:
  delay(1000);
  pinMode(Led_OnBoard, OUTPUT);       // Initialize the Led_OnBoard pin as an output
  pinMode(motorPin, OUTPUT);          // Initialise the motor Pin as an output
  //digitalWrite(motorPin, LOW);        // keep motor off initally
  Serial.begin(9600);
  WiFi.mode(WIFI_OFF);                //Prevents reconnection issue (taking too long to connect)
  delay(1000);
  WiFi.mode(WIFI_STA);                //This line hides the viewing of ESP as wifi hotspot
  WiFi.begin(ssid, password);         //Connect to your WiFi router
  Serial.println("");
  Serial.print("Connecting");
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(Led_OnBoard, LOW);
    delay(250);
    Serial.print(".");
    digitalWrite(Led_OnBoard, HIGH);
    delay(250);
  }

  digitalWrite(Led_OnBoard, HIGH);
  //If connection successful show IP address in serial monitor
```

Uploading...

```
Global variables use 29532 bytes (36%) of dynamic memory, leaving 52388 bytes for local variables. Maximum is 81920 bytes.
Uploading 274992 bytes from C:\Users\sampa\AppData\Local\Temp\arduino_build_859816/upLoad.ino.bin to flash at 0x00000000
................................................................................ [ 29% ]
...................................................................
```

25

codeWorking | Arduino 1.8.10

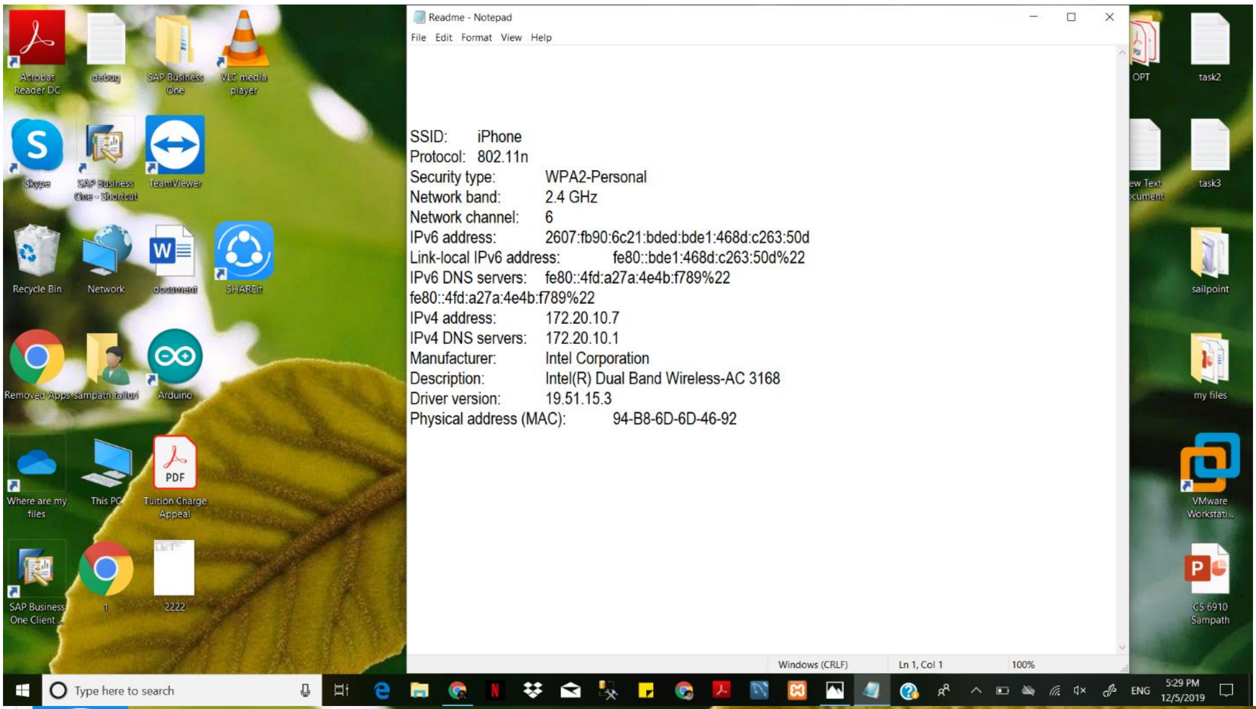
File Edit Sketch Tools Help

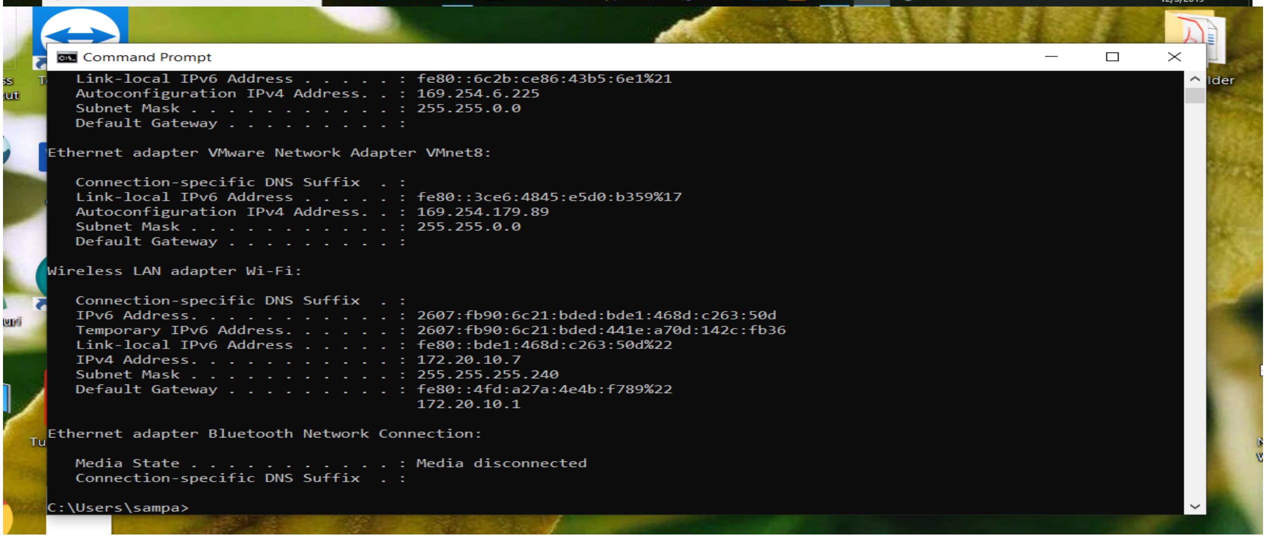
codeWorking

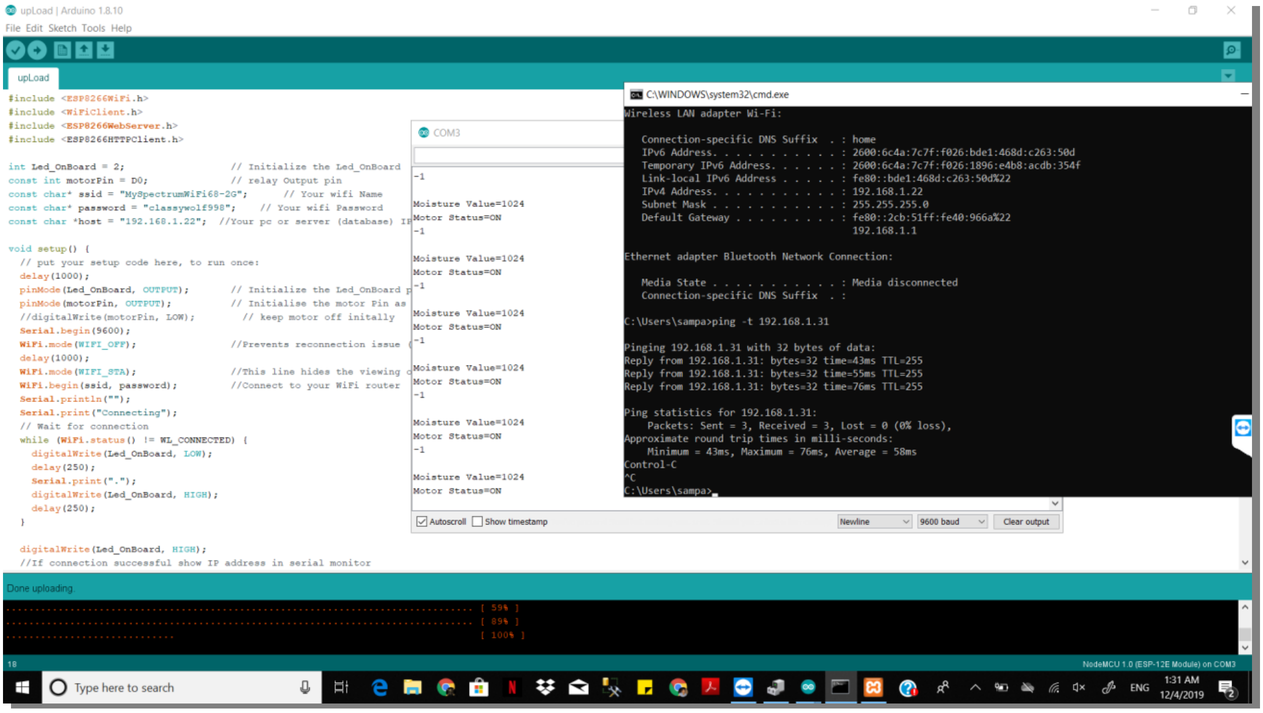
```
void loop() {
  // put your main code here, to run repeatedly:
  HTTPClient http;                                                        //Declare object of class HTTPClient

  String mValueSend, postData,motor;
  int rvalue=analogRead(A0);                                              //Read Analog value of Moisture Sensor
  mValueSend = String(rvalue);                                            //String to interger conversion

  if(rvalue >= 700){
    digitalWrite(motorPin, HIGH);                                         // turn ON motor (high value, low moisture)
    motor = "ON";
  }
  else{
    digitalWrite(motorPin, LOW);                                          // turn OFF motor (low value, High moisture)
    motor = "OFF";
  }
  postData = "svalue="+mValueSend;                                      //Post Data (moisture sensor value)

  http.begin("http://172.20.10.7/iot/InsertDB.php");                  //Specify request destination
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");  //Specify content-type header
  int httpCode = http.POST(postData);                                   //Send the request
  String payload = http.getString();                                    //Get the response payload


  Serial.println(httpCode);                                             //Print HTTP return code
  Serial.println(payload);                                              //Print request response payload

  Serial.println("Moisture Value=" + mValueSend);
  Serial.println("Motor Status=" + motor);

  http.end();  //Close connection

  delay(4000);  //Here there is 4 seconds delay plus 1 second delay below, so Post Data at every 5 seconds
  digitalWrite(Led_OnBoard, LOW);
  delay(1000);
```

```
................................................................................ [ 59% ]
................................................................................ [ 89% ]
.............................                                                    [ 100% ]
```

70


Readme - Notepad
File Edit Format View Help
SSID: iPhone
Protocol: 802.11n
Security type: WPA2-Personal
Network band: 2.4 GHz
Network channel: 6
IPv6 address: 2607:fb90:6c21:bded:bde1:468d:c263:50d
Link-local IPv6 address: fe80::bde1:468d:c263:50d%22
IPv6 DNS servers: fe80::4fd:a27a:4e4b:f789%22
fe80::4fd:a27a:4e4b:f789%22
IPv4 address: 172.20.10.7
IPv4 DNS servers: 172.20.10.1
Manufacturer: Intel Corporation
Description: Intel(R) Dual Band Wireless-AC 3168
Driver version: 19.51.15.3
Physical address (MAC): 94-B8-6D-6D-46-92
Windows (CRLF)
Ln 1, Col 1
100%
Type here to search
ENG
5:29 PM
12/5/2019


Command Prompt
Link-local IPv6 Address . . . . . : fe80::6c2b:ce86:43b5:6e1%21
Autoconfiguration IPv4 Address. . : 169.254.6.225
Subnet Mask . . . . : 255.255.0.0
Default Gateway . . . . :
Ethernet adapter VMware Network Adapter VMnet8:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::3ce6:4845:e5d0:b359%17
Autoconfiguration IPv4 Address. . : 169.254.179.89
Subnet Mask . . . . : 255.255.0.0
Default Gateway . . . . :
Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix . :
IPv6 Address. . . . . : 2607:fb90:6c21:bded:bde1:468d:c263:50d
Temporary IPv6 Address. . . . . . : 2607:fb90:6c21:bded:441e:a70d:142c:fb36
Link-local IPv6 Address . . . . . : fe80::bde1:468d:c263:50d%22
IPv4 Address. . . . . : 172.20.10.7
Subnet Mask . . . . : 255.255.255.240
Default Gateway . . . . : fe80::4fd:a27a:4e4b:f789%22
172.20.10.1
Ethernet adapter Bluetooth Network Connection:
Media State . . . . : Media disconnected
Connection-specific DNS Suffix . :
C:\Users\sampa>



Cloud connectivity:

1. Create an account in AWS Educate.
2. Sign in to the AWS Management Console and open the Amazon RDS console.
3. In the upper-right corner of the AWS Management Console, choose the AWS Region in which you want to create the DB instance.
4. In the navigation pane, choose Databases. Choose Create database.
5. In Choose a database creation method, choose Standard Create.
6. In Engine options, choose MySQL.
7. To enter your master password, do the following:

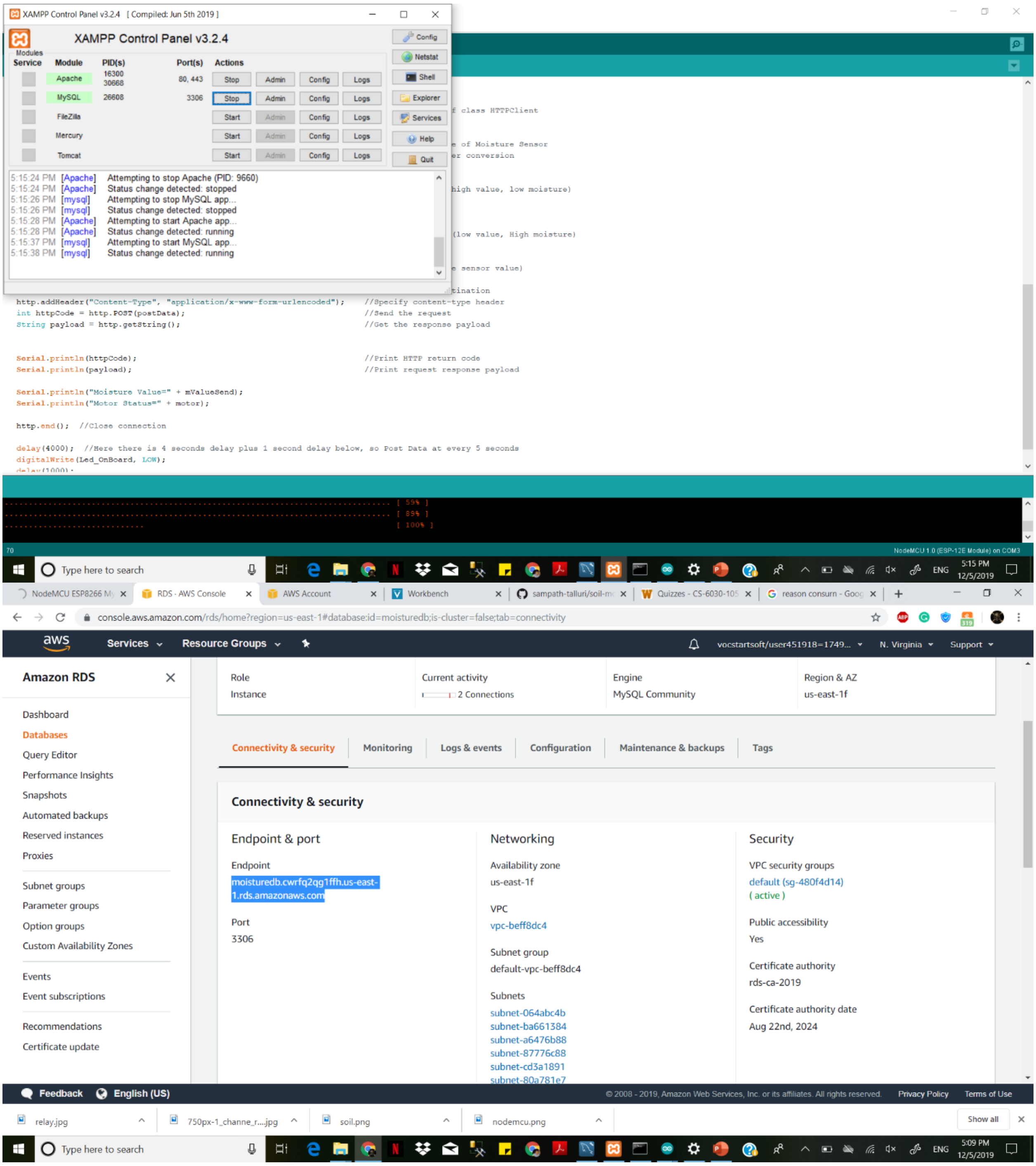
a. In the Settings section, open Credential Settings.
b. Clear the Auto generate a password check box.
c. (Optional) Change the Master username value and enter the same password in Master password and Confirm password.

8. Choose Create database.

Note:

1. Set Public accessibility to YES
2. Now you can connect the database form MySQL Workbench from your system and can create schema and tables.

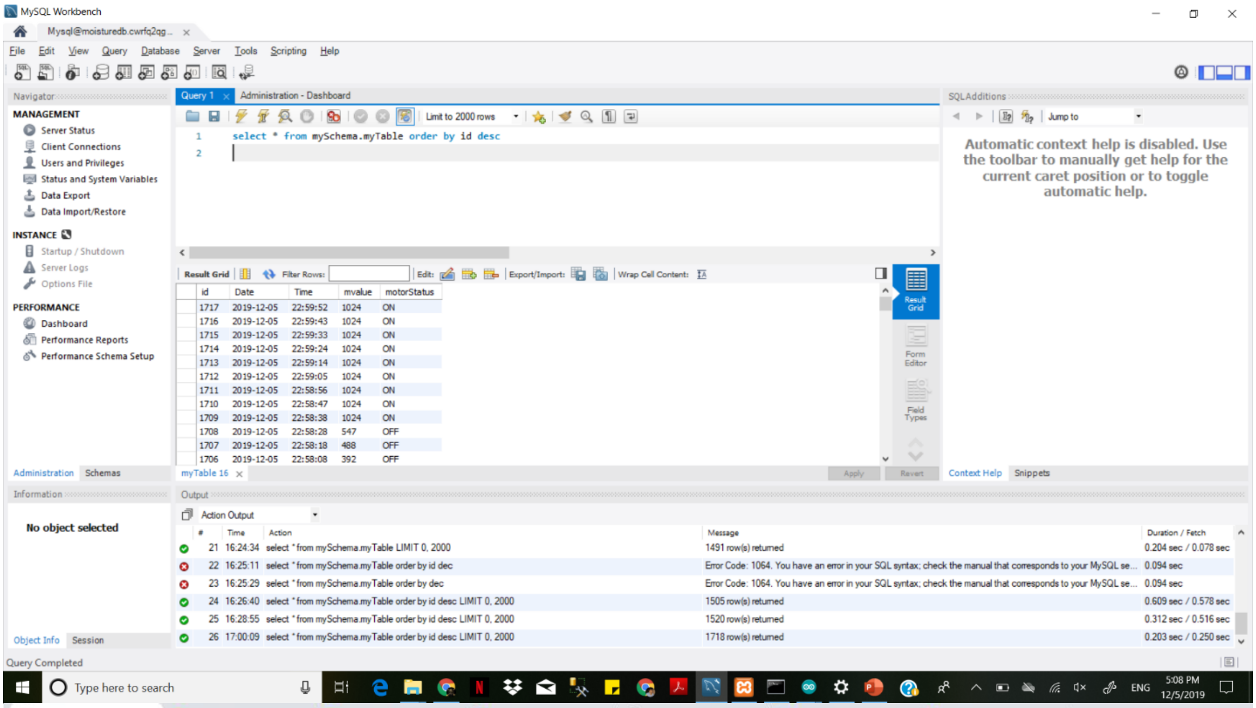
It takes 10-15 minutes to create database



Then we open the Mysql workbench in computer
Connect the AWS RDS to Mysql by using server name, username and password.
We get connected to cloud.

We also use Xampp server to monitor the data locally in our system.
Once the cloud is connected to Mysql and the data starts uploading to the cloud once the data is uploaded successfully we get the output in Arduino bode with soil moister value and motor status ON/OFF and also return 200 and ok which says the data is uploaded successfully.

Once we go to Mysql and retrieve the values from the cloud we get the Id, date, time and moisture value and motor status printed.


MySQL Workbench
select * from mySchema.myTable order by id desc
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
id Date Time mvalue motorStatus
1717 2019-12-05 22:59:52 1024 ON
1716 2019-12-05 22:59:43 1024 ON
1715 2019-12-05 22:59:33 1024 ON
1714 2019-12-05 22:59:24 1024 ON
1713 2019-12-05 22:59:14 1024 ON
1712 2019-12-05 22:59:05 1024 ON
1711 2019-12-05 22:58:56 1024 ON
1710 2019-12-05 22:58:47 1024 ON
1709 2019-12-05 22:58:38 1024 ON
1708 2019-12-05 22:58:28 547 OFF
1707 2019-12-05 22:58:18 488 OFF
1706 2019-12-05 22:58:08 392 OFF
No object selected
21 16:24:34 select * from mySchema.myTable LIMIT 0, 2000 1491 row(s) returned
22 16:25:11 select * from mySchema.myTable order by id dec Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...
23 16:25:29 select * from mySchema.myTable order by dec Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...
24 16:26:40 select * from mySchema.myTable order by id desc LIMIT 0, 2000 1505 row(s) returned
25 16:28:55 select * from mySchema.myTable order by id desc LIMIT 0, 2000 1520 row(s) returned
26 17:00:09 select * from mySchema.myTable order by id desc LIMIT 0, 2000 1718 row(s) returned
Query Completed


localhost/iot/
The database table has 1621 total entries.
Id Date Time Moisture Value Motor Status
1620 2019-12-05 22:44:42 1017 ON
1619 2019-12-05 22:44:32 1020 ON
1618 2019-12-05 22:44:23 1017 ON
1617 2019-12-05 22:44:15 1020 ON
1616 2019-12-05 22:44:04 1020 ON
1615 2019-12-05 22:43:54 1020 ON
1614 2019-12-05 22:43:45 1020 ON
1613 2019-12-05 22:43:36 1017 ON
1612 2019-12-05 22:43:27 1020 ON
1611 2019-12-05 22:43:18 1020 ON
1610 2019-12-05 22:43:08 1020 ON
1609 2019-12-05 22:42:58 1020 ON
1608 2019-12-05 22:42:48 1020 ON
1607 2019-12-05 22:42:37 1020 ON
1606 2019-12-05 22:42:28 1020 ON
1605 2019-12-05 22:42:17 1020 ON