A

Project Report

On

"SMART IRRIGATION SYSTEM USING IOT"

Submitted

In partial Fulfillment of the Requirements

For the Degree of

BACHELOR OF TECHNOLOGY

By

Bhuvnesh Singh (1500431007)

Amit Kumar (1500431003)

Ajeet Singh (1500431002)

Dharmendra kumar (1400431014)

Under the supervision of

Dr. Dushyant singh

To the

P G Department of electronics and communication Engineering

RBS ENGINEERING TECHNICAL CAMPUS BICHPURI

AGRA

April, 2019

CERTIFICATE

It is Certified that **BHUVNESH SINGH** (1500431007), **AMIT KUMAR** (1500431004) , **AJEET SINGH** (1500431002), **DHARMENDRA KUMAR** (1400431014) have carried out the research work presented in this thesis entitled "**SMART IRRIGATION SYSTEM USING IOT**" for the award of Bachelor of Technology from Dr.A.P.J. Abdul Kalam Technical University, Uttar Pradesh, Lucknow under my supervision .The thesis embodies results of original work, and studies as are carried out by the students themselves and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/ Institute.

Signature

DR. DUSHYANT SINGH

Associate Professor

P.G .Department of

Electronics & Communication

Place :

Date :

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

S.N0.	Student Name	Roll No.	Signature
1	BHUVNESH SINGH	1500431007	
2	AMIT KUMAR	1500431003	
3	AJEET SINGH	1500431002	
4	DHARMENDRA KUMAR	1400431014	

ABSTRACT

The main objective of this project is to develop an automation irrigation system using an Nodemcu board with Internet being remotely controlled by any computer Operating System. So that agricultural Lands are irrigated automatically without physical present of farmer. As technology is advancing so irrigations are also getting smarter. Modern irrigation pumps are gradually shifting from Conventional switches to centralized control system, involving remote controlled switches. Presently, conventional pump switches located in different parts of the agriculture land are Makes it difficult for the user to go near them to operate and physically present on those areas. Even more it becomes more difficult for the elderly or physically handicapped people to do so. Remote controlled irrigation automation system provides a most modern solution with smart Phones for those persons who want to do agriculture without physically present on that place. In order to achieve this, a IOT (Internet of thing) module is interfaced to the Nodemcu board at The receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF Commands to the receiver where loads are connected. By touching the specified location on the GUI, the loads can be turned ON/OFF remotely through this technology. The loads are operated By IOT board through Relay Module. Along with this we use a soil sensor. Which detect whether soil is dry or wet. When soil Condition is dry soil sensor give command to IOT module to start the pump. When soil becomes Wet it gives command to stop the water pump. It works in accordance with the soil condition. This project is complete smart project for advance irrigation.

Keywords: Automation, NodeMCU(ESP8266), Nodejs server, IOT, blynk App grafana etc.

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to prevent report of **B.Tech** Project undertaken during **B.Tech Final year**. We owe special depth of gratitude to **Dr. Dushyant Singh**, P.G. Department of Electronics & Communication Engineering, **RBS Engineering Technical Campus**, Agra (Affiliated to **Dr. A.P.J. Abdul Kalam Technical University, Lucknow**) for his constant support and guidance throughout the course of our work. His constant motivation and guidance helped us in completing the project successfully.

We would like to extend our deep sense of gratitude to other respected faculty members of the department for their kind support and cooperation. We would like to thank our family members as well as friends for always showing faith in our work.

BHUVNESH SINGH

AMIT KUMAR

AJEET KUMAR

DHARMENDRA KUMAR

TABLE OF CONTENTS

List of Symbols & Abbreviations

- ➢ = : Equals, is the same as, results in
- ➢ > : Is greater than, is larger than
- ➢ ↑ : Increase, rise, growth
- ➢ < : Is less than, is smaller than
- ➢ + : And, also, as well as, in addition to, plus
- ➢ **DC** : Direct current
- ➢ **GND** : Ground
- ➢ **GPIO** : General-purpose input/output
- ➢ **HTTP** : Hypertext Transfer Protocol
- ➢ **ID** : Identifier
- ➢ **IoT** : Internet of Things
- ➢ **JSON** : Javascript Object Notation
- ➢ **LCD** : Liquid Crystal Display
- ➢ **LED** : Light-emitting diode
- ➢ **OSI** : Open Systems Interconnection
- ➢ **PCB** : Printed Circuit Board
- ➢ **REST** : Representational State Transfer
- ➢ **TCP** : Transmission Control Protocol
- ➢ **TTL** : Transistor–transistor logic
- ➢ **UART** : Universal serial receiver/transmitter
- ➢ **USB** : Universal Serial Bus

List of Figures

Chapter-1

INTRODUCTION

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION

India is the country of village and agriculture plays an important role for development of country. In our country, agriculture depends on the monsoons which has insufficient source of water. So the irrigation is used in agriculture field. In Irrigation system, depending upon the soil type, water is provided to plant. In agriculture, two things are very important, first to get information of about the fertility of soil and second to measure moisture content in soil. Nowadays, for irrigation, different techniques are available which are used to reduce the dependency of rain. And mostly this technique is driven by electrical power and on/off scheduling. In this technique, water level indicator placed in water reservoir and soil moisture sensors are placed root zone of plant and near the module and gateway unit handles the sensor information and transmit data to the controller which in turns the control the flow of water through the valves.In the India population crossing 1.35 billion in 2017, a balance between the optimum population growth and a healthy of nation is far to be achieved.The rising population, there is a need for increased agricultural production. Irrigated agriculture has been an extremely important source increased agricultural production.Now a days people wants to observe their work from anywhere on their digital devices such as Smartphone and tablet or laptop.

Figure 1.1 ancient irrigation system

This project is for to create an IOT based automated irrigation systems which turns the pumping motor ON and OFF on detecting the moisture content and sufficient water level and pass data through IOT platform.

1.2 MOTIVATION

For continuously increasing demand and decrease in supply of food necessities, it's important to rapid improvement in production of food technology. Agriculture is only the source to provide this. This is the important factor in human societies to growing and dynamic demand in food production. Agriculture plays the important role in the economy and development, like India. Due to lack of water and scarcity of land water result the decreasing volume of water on earth, the farmer use irrigation. Irrigation may be defined as the science of artificial application of water to the land or soil that means depending on the soil type, plant are to be provided with water.

Figure 1.2 modern irrigation system

The purpose of irrigation is to supply adequate amount of water when rainfall is not sufficient or timely to meet the crops' water needs. Since the 1980s, one of Dole's priorities has been to develop a more scientific approach to better schedule the irrigation of bananas and to apply water only when and where necessary as a way to compensate for water deficiencies in the soil.

In order to assess the need for irrigation, several factors must be considered:

1. Smart irrigation systems can optimize water levels based on things such as soil moisture and weather predictions.
2. The smart irrigation system will help you have better control of your landscape and irrigation needs as well as peace of mind that the smart system can make decisions independently if you are away.

3. You will save a significant amount of money on your water bills because through intelligent control and automation, your smart irrigation system will optimize resources so that everything gets what it needs without needless waste.

1.3 AREA OF UTILITY

1. The primary focus of this project is to help the farmers and reduce their work.
2. This module can be implemented in perennial plant irrigation land and gardening land.
3. To save water and reduce human intervention in the agriculture field.
4. Continuously Monitoring the status of sensors and provide signal for taking necessary action.
5. To get the output of soil water sensor and provide water to crop.
6. To observe other parameters for better yield.
7. To increase productivity.
8. To visualize water level and soil moisture in real time.

1.4 PROBLEM ANALYSIS

The process of utilizing technology in farming and cultivation requires deep knowledge of agricultural processes, biology, chemistry, and empirical knowledge. There are many parameters which must be taken into consideration and investigated in depth when designing a system that should improve cultivation procedures by making the whole process more effective and sustainable.

In order to design and build a precision agriculture system that can be widely used by many users and applied in different contexts, many questions need to be addressed. Some of these questions are:

1. Is it feasible to design a system that will accommodate every possible scenario in an agricultural context and do so for all possible users?
2. Is automation in agriculture really useful and in what part or parts of the cultivation process (e.g. seed planting, growing, harvesting, selling) can it be applied?
3. What is the cost of the cultivation process and how can this cost be reduced by automating one or more parts of this process?
4. What is the most costly component of this process that could be reduced? How and how much could this cost be reduced?

5. Are geographic parameters such as location, altitude, solar exposure, ground and air moisture, ground and air temperature, mineral content of the soil, the (micro-) climate, or the season, sufficient to make a significant difference in the way that a crop is cultivated?
6. What are the sensitivities of the crop that should be taken care of when cultivating?
7. What types of plants are to be planted and how long will this crop be planted in this location? What is the planned rotation of crops? What are the plans for applying fertilizer to this location? What is the level of the farmer's empirical knowledge?
8. Are there any abnormalities regarding the location, season of the year, previous crops in a specific field, or a combination of all these aspects which need to be considered as part of an informed decision making procedure by the farmer?

1.5 REVIEW OF RELATED LITERATURE

Karan kansara (2015) proposed an automated irrigation system where the humidity and temperature sensors are used to sense the soil conditions and based on that microcontroller will control the water flow. Farmer will be intimated through GSM. This system doesn't monitor the nutrient content in the soil [1].

Archana and Priya (2016) proposed a paper in which the humidity and soil moisture sensors are placed in the root zone of the plant. Based on the sensed values the microcontroller is used to control the supply of water to the field. This system doesn't intimate the farmer about the field status [2].

Sonali D.Gainwar and Dinesh V. Rojatkar (2015) proposed a paper in which soil parameters such as humidity, moisture and temperature are measured for getting high yield from soil. This system is fully automated which turns the motor pump ON/OFF as per the level of moisture in the soil. The current field status is not intimated to the farmer[3].

S.Reshma and B.A.Sarath (2016) proposed an IOT based automatic irrigation system using wireless sensor networks in which various sensors are used to measure the soil parameters. This system provides a web interface to the user to monitor and control the system remotely. Weather monitoring is not done in this system[4].

In A Remote Measurement and Control System for Greenhouse Based on GSM-SMS the proposed system introduced a GSM-SMS remote measurement and control system for greenhouse based on PC-based database system connected with base station. Base station is developed by using a microcontroller, GSM module, sensors and actuators. In practical operation, the central station receives and sends messages through GSM module. Criterion value of parameters to be measured in every base station is set by central station, and then in base stations parameters including the air temperature, the air humidity[5].

Indu et al. (2013) mainly focuses on reviews in the field of remote monitoring and control, the technology used and their potential advantages. The paper proposes an innovative GSM/Bluetooth based remote controlled embedded system for irrigation. The system sets the irrigation time depending on the temperature and humidity reading from sensors and type of crop and can automatically irrigate the field when unattended. Information is exchanged between far end and designed system via SMS on GSM network. A Bluetooth module is also interfaced with the main microcontroller chip which eliminates the SMS charges when the user is within the limited range of few meters to the designated system. The system informs users about many conditions like status of electricity, dry running motor, increased temperature, water content in soil and smoke via SMS on GSM network or by Bluetooth[5].

R.Suresh et al. (2014) mentioned about using automatic microcontroller based rain gun irrigation system in which the irrigation will take place only when there will be intense requirement of water that save a large quantity of water. These systems bring a change to management of field resource where they developed a software stack called Android is used for devices that include an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. Mobile phones have almost become an integral part of us serving multiple needs of humans. This application makes use of the GPRS feature of mobile phone as a solution for irrigation control system. These system covered lower range of agriculture land and not economically affordable[6].

In IOT SMS alarm system based on SIM900A [7], an IOT alarm system based on SIM900A module of SIMCOM Company was designed for greenhouse. The system can gather environmental parameters such as air temperature and air humidity. Meanwhile, with

the use of AT command, this system can also realize SMS automatic sending and receiving, environmental parameters overrun alarm and insufficient balance alarm[7].

1.6 CHAPTER SUMMARY AND CONCLUSION

In this chapter, we did talk about the brief introduction of our project and also analysis of recent developed project the are also based on irrigation system. From this chapter we will identify how our project will different with recent developed project.

This project shows project problem analysis and also find area of utilities e.g.-

- The primary focus of this project is to help the farmers and reduce their work.
- This module can be implemented in perennial plant irrigation land and gardening land.
- To save water and reduce human intervention in the agriculture field.

Chapter-2

METHODOLOGY

CHAPTER 2 METHODOLOGY

2.1 BLOCK DIAGRAM

Here we can see the block diagram of this project. This diagram shows all the working and functionality of our project and also shows how the data will be flow in this project:-

2.1.1 FOR NODEJS SERVER

From the block diagram we will see that nodemcu-1 consist of three sensors connected that are DHT11 sensor, soil moisture sensor and also soil temperature sensor (DS18B20). Here the nodemcu-1 collect the data from all three sensors and the data are sending to the node JS server into the database that are created by MongoDB database . This database is also connect with nodemcu-2 and also connected to the the mobile and web application mobile and web applications are used to show the data online and everyone can see that and also monitor. Nodemcu-2 Retrieve the data from the node JS server. Nodemcu-2 connected to the the railway that also connected with the pump at a used to control pump ON and OFF. The data that are stored by node JS server also seen on the Oled display. When the soil moisture is passes the the threshold value the relay will on that by the pump also ON.

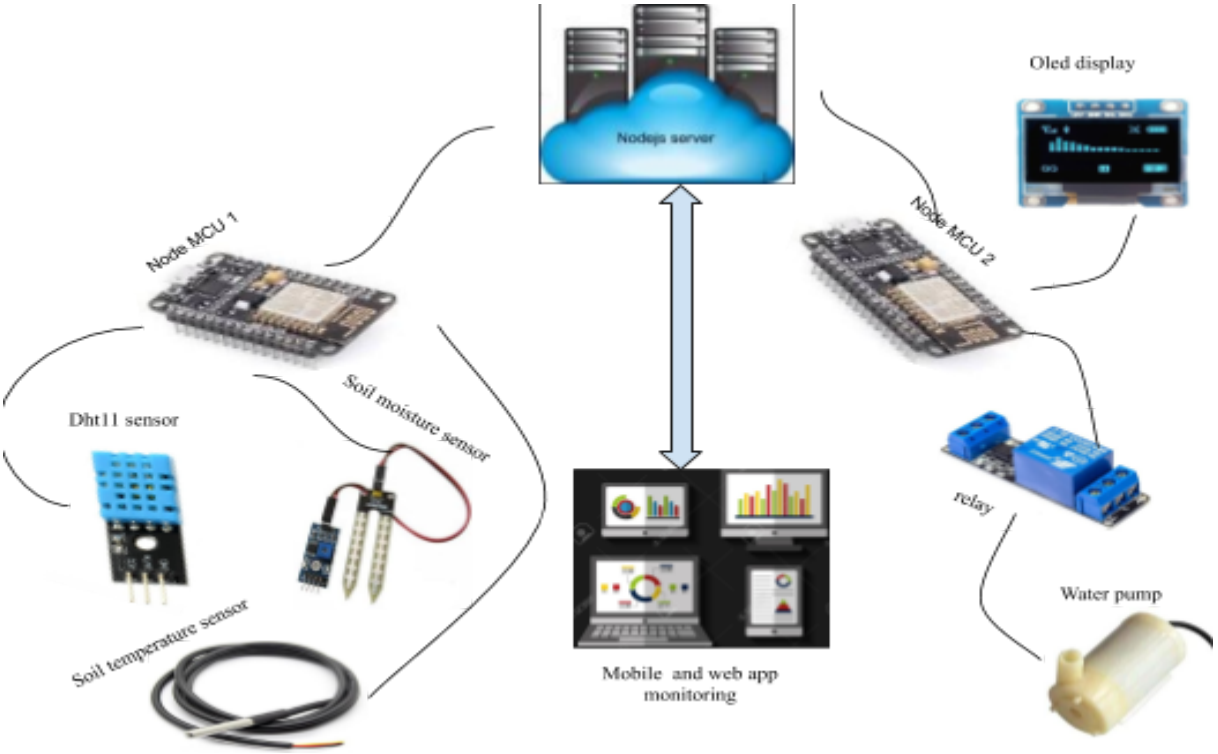

Figure 2.1 Block diagram using NodeJS server

2.1.2 FOR BLYNK WEB SERVER

From the block diagram we will see that nodemcu-1 consist of three sensors connected that are DHT11 sensor, soil moisture sensor and also soil temperature sensor (DS18B20). Here the nodemcu-1 collect the data from all three sensors and the data are sending to the Blynk Web server into the database that are created by online blynk web server developer. This database is also connect with nodemcu-2 and also connected to the the mobile and web application mobile and web applications are used to show the data online and everyone can see that and also monitor. Nodemcu-2 Retrieve the data from the node JS server. Nodemcu-2 connected to the the railway that also connected with the pump at a used to control pump ON and OFF. The data that are stored by node JS server also seen on the Oled display. When the soil moisture is passes the the threshold value the relay will on that by the pump also ON. if the system will fails then we use the manual control of pump and lamp and also we control the pump and lamp by the mobile application.

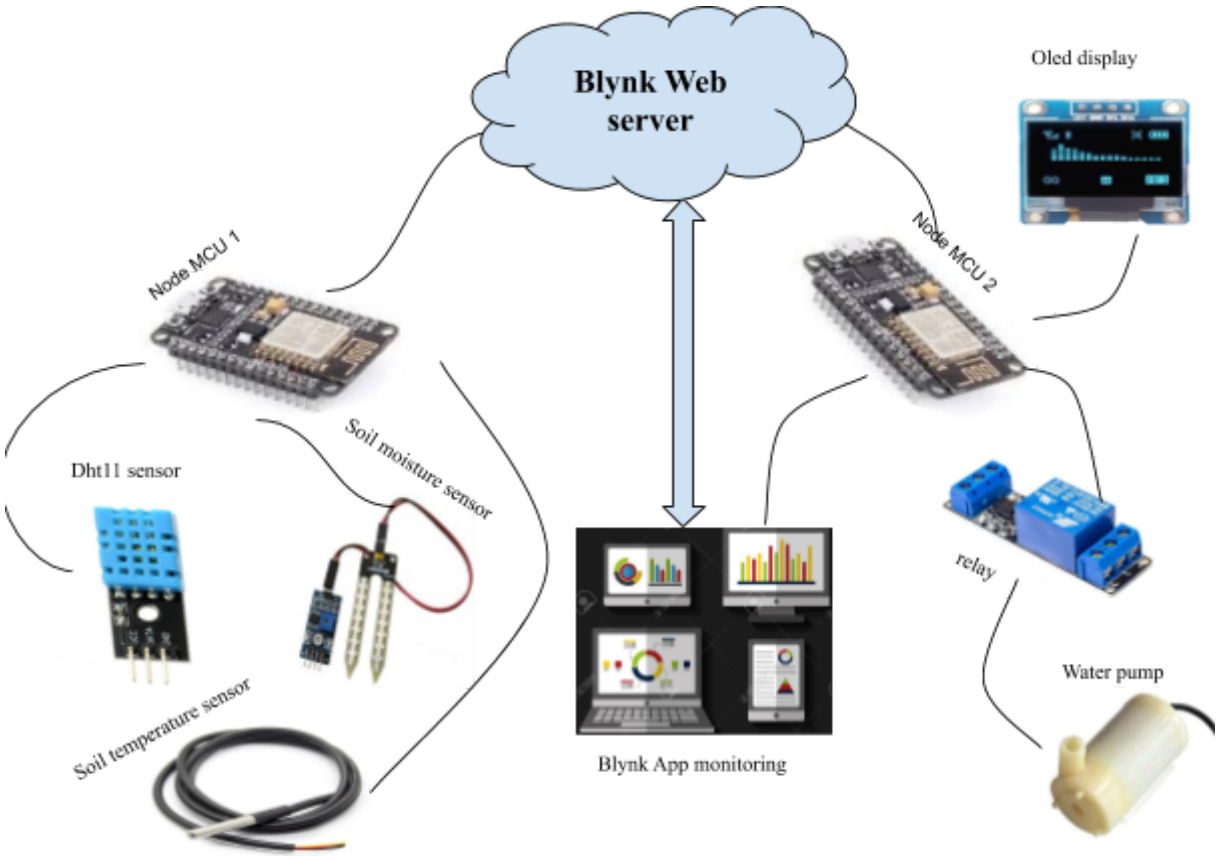

Figure 2.2 Block diagram using Blynk Web Server

2.2 FLOW CHART

Here the flow chart of our project shows data flowing in our project. The flow of data explains how the data sends from one block to another and also explain the basic ideas about the data flowing.

2.2.1 FOR NODEJS SERVER

From the block diagram we will see that nodemcu-1 consist of three sensors connected that are DHT11 sensor, soil moisture sensor and also soil temperature sensor (DS18B20). Here the nodemcu-1 collect the data from all three sensors and the data are sending to the node JS server into the database that are created by MongoDB database . This database is also connect with nodemcu-2 and also connected to the the mobile and web application mobile and web applications are used to show the data online and everyone can see that and also monitor. Nodemcu-2 Retrieve the data from the node JS server. Nodemcu-2 connected to the the railway that also connected with the pump at a used to control pump ON and OFF. The data that are stored by node JS server also seen on the Oled display. When the soil moisture is passes the the threshold value the relay will on that by the pump also ON.

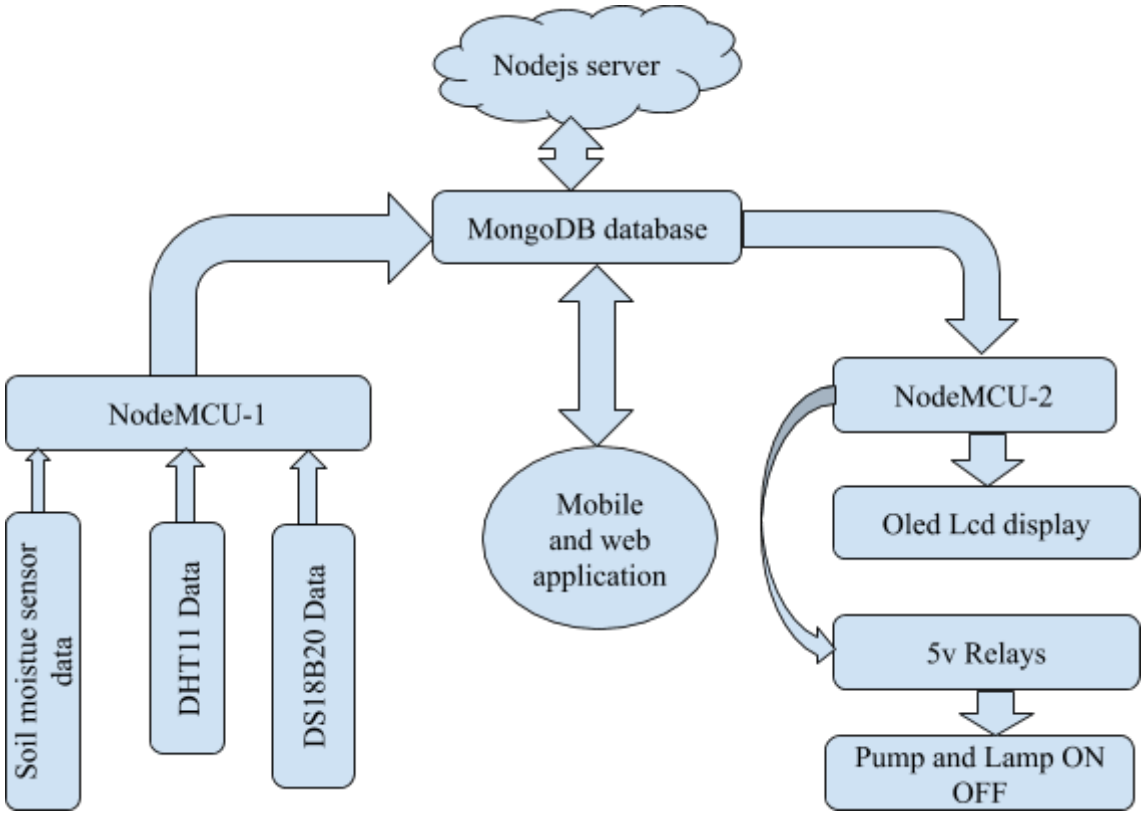

Figure 2.3 Flow Graph using NodeJS server

2.2.2 FOR BLYNK WEB SERVER

It is also similarly like above flow chart except that we use blynk web server in place of node js server.From the block diagram we will see that nodemcu-1 consist of three sensors connected that are DHT11 sensor, soil moisture sensor and also soil temperature sensor (DS18B20). Here the nodemcu-1 collect the data from all three sensors and the data are sending to the Blynk Web server into the database that are created by online blynk web server developer. This database is also connect with nodemcu-2 and also connected to the the mobile and web application mobile and web applications are used to show the data online and everyone can see that and also monitor. Nodemcu-2 Retrieve the data from the node JS server. Nodemcu-2 connected to the the railway that also connected with the pump at a used to control pump ON and OFF. The data that are stored by node JS server also seen on the Oled display. When the soil moisture is passes the the threshold value the relay will on that by the pump also ON. if the system will fails then we use the manual control of pump and lamp and also we control the pump and lamp by the mobile application.

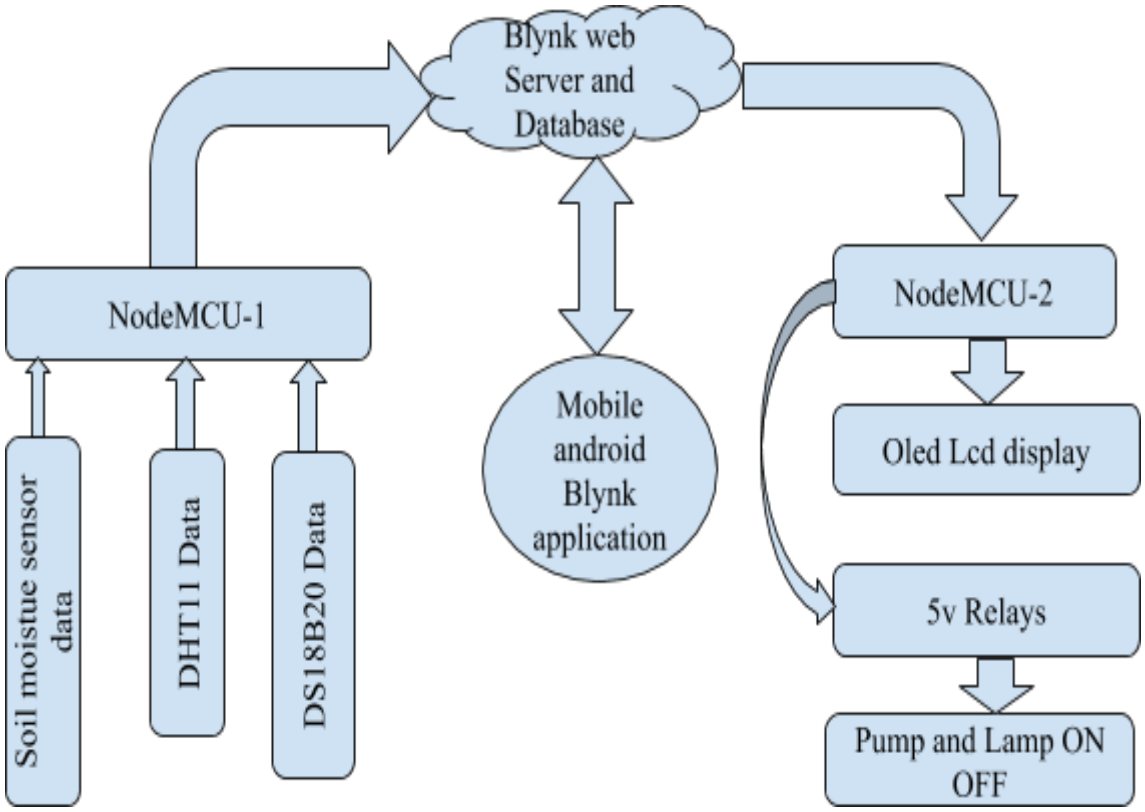

Figure 2.4 Flow Graph using Blynk Web Server

Chapter-3

Hardware Used

CHAPTER 3 HARDWARE USED

3.1 COMPONENT REQUIRED

In this project there are some hardware components and devices will use for the performing project circuit design. These are some hardware which are given below:-

1. **NODE MCU (ESP8266)**
2. **OLED LCD LED Display Module**
3. **DHT11 TEMPERATURE SENSOR**
4. **YL-69 sensor and LM393 Comparator**
5. **DS18B20 - High Quality Waterproof Digital Temperature Sensor**
6. **5V Relay**
7. **Submersible Water Pump**
8. **Transistor (BC-547)**
9. **Diode (IN4007)**
10. **Relay Circuit**

3.1.1 NODE MCU (ESP8266)

ESP8266 is a complete and self-contained Wi-Fi network solutions that can carry software applications, or through another application processor uninstall all Wi-Fi networking capabilities. ESP8266 when the device is mounted and as the only application of the application processor, the flash memory can be started directly from an external Move. Built-in cache memory will help improve system performance and reduce memory requirements. Another situation is when wireless Internet access assume the task of Wi-Fi adapter, you can add it to any microcontroller-based design, and the connection is simple, just by SPI / SDIO interface or central processor AHB bridge interface. Processing and storage capacity on ESP8266 powerful piece, it can be integrated via GPIO ports sensors and other applications specific equipment to achieve the lowest early in the development and operation of at least occupy system resources. The ESP8266 highly integrated chip, including antenna switch balun, power management converter, so with minimal external circuitry, and includes front-end module, including the entire solution designed to minimize the space occupied by PCB.

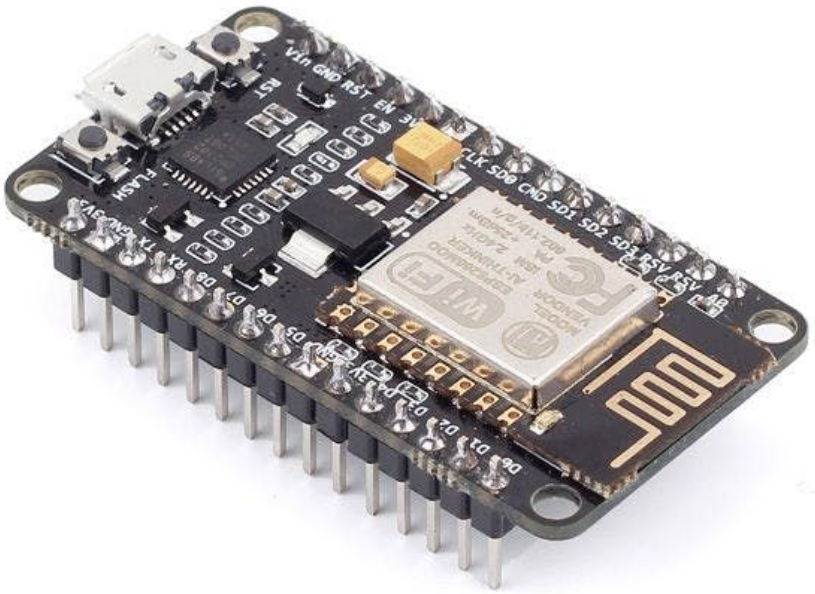

Figure 3.1 NodeMCU Structure

The NodeMCU ESP-12E is the integrated version of the popular ESP8266, a Serial to Wi-Fi System On a Chip (SoC) that appeared for the first time in 2013, been released on following year. The ESP8266 was developed by the Shanghai-based company Espressif Systems, an IC manufacturer focused on the development of RF chips, particularly Wi-Fi.

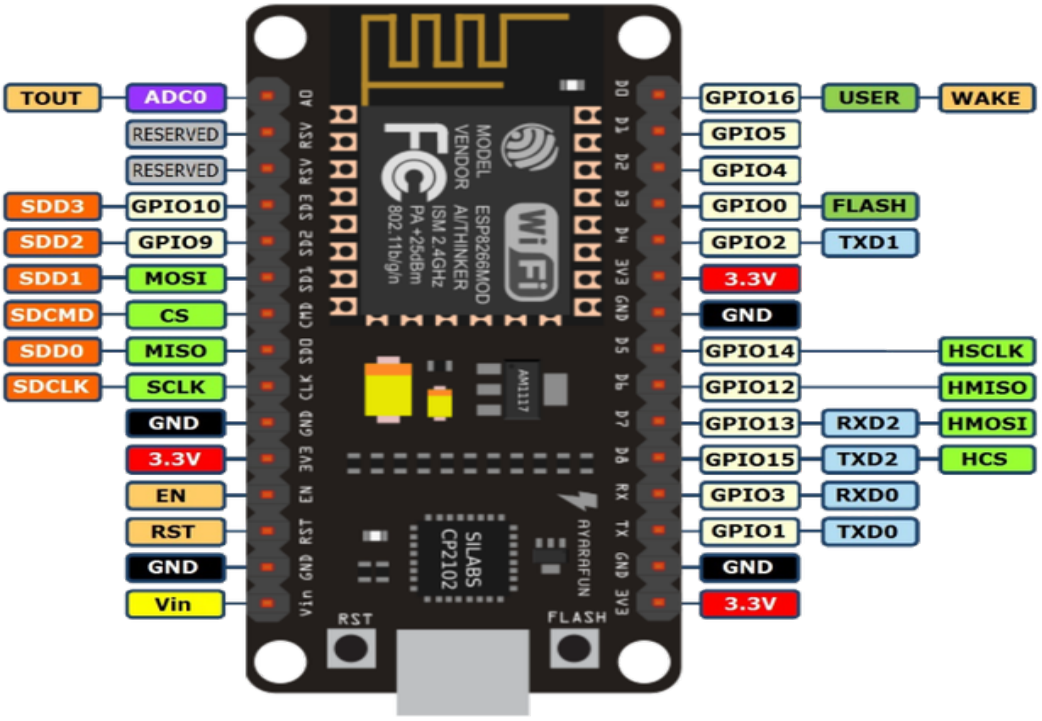

Figure 3.2 NodeMCU with GPIO Pin diagram

There are several modules in the market that use the ESP8266 chip, they are named ESP-NN, where NN is a number 01, 02, ... 12, sometimes followed by a letter. These modules typically carry the ESP8266 SoC, flash memory, a crystal, and in most cases, an onboard antenna. In the link you can find the full list of ESP8266 based devices found in the market. The 2 more important modules are without doubt, the ESP-01 and the ESP-12E.

Here, we will use the ESP-12E Development Board (NodeMCU DevKit 1.0). This development board for the ESP8266 SoC inside the ESP-12E module is out-of-the-box ready for you to connect it to your computer, install USB drivers, and start writing programs that connect to your Wi-Fi network!

3.1.1.1 BLOCK DIAGRAM OF ESP8266

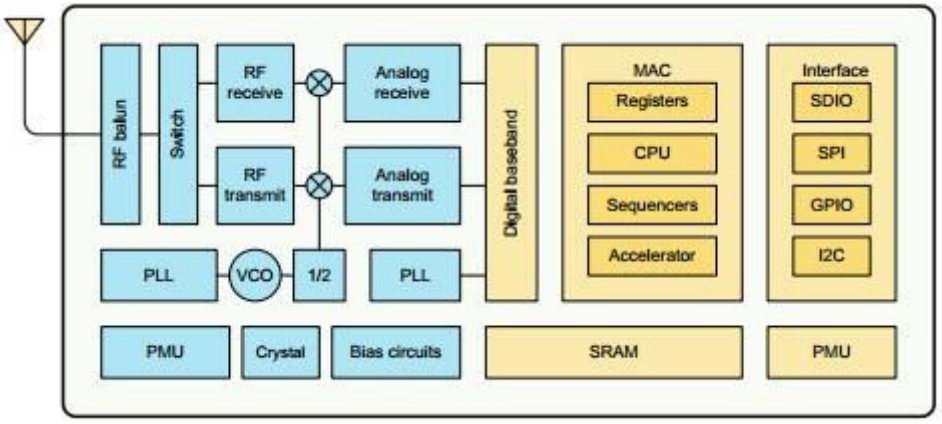

Figure 3.3 Block diagram of ESP8266

3.1.1.2 TECHNICAL SPECIFICATION

1. Support STA/AP/STA+AP 3 working modes;
2. Built-in TCP/IP protocol stack, support multiple-channel TCP Client connection (max 5);
3. 0～D8, SD1～SD3: used for GPIO, PWM (D1-D8), IIC, ect; the driven ability can be arrived at 15mA;
4. AD0: one-way 10 bits ADC;
5. Power input: 4.5V~9V(10VMAX), support USB powered and USB debug;
6. Working current: ≈70mA(200mA MAX, continue), standby＜200uA;
7. Transmission data rate: 110-460800 bps;
8. Support UART/GPIO data communication interface;
9. Support update firmware remotely (OTA);
10. Support Smart Link;
11. Working temperature:-40℃～＋125℃；

12. Driven mode: double large-power H bridge driven
13. Weight: 7g.

3.1.2 OLED LCD LED DISPLAY MODULE

OLED modules are available in wide variety of sizes and features. The one we're going to use in this tutorial is a mono color 128x64 OLED module. This type of module is available in following sizes (In order that you see on the pictures):

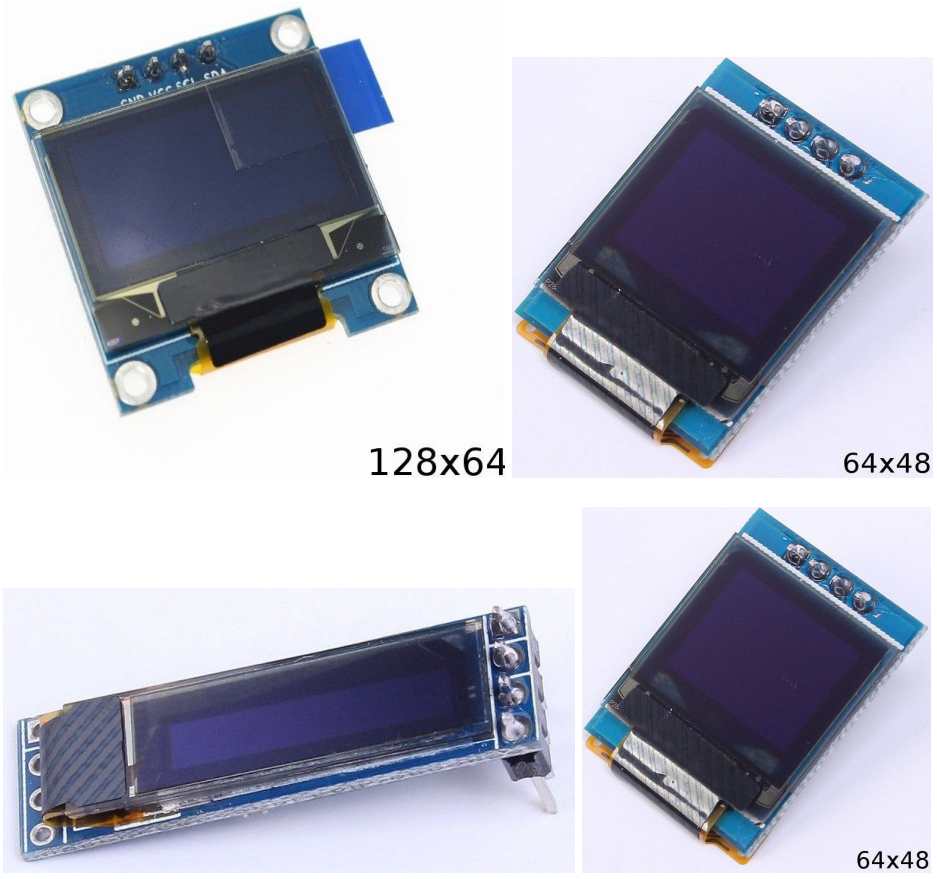

Figure 3.4 OLED LCD LED Display Module

1. 128x64
2. 128x32
3. 96x16
4. 64x48
5. 64x32

As all these modules support I2C protocol as a mean for communication, the code and wiring of all of them is exact same. The only difference is that you have to consider the size of the display on your code so that the contents that you're going to display, fit properly on it.The inter-integrated circuit (IIC) which is normally called I2C (I squared C) developed by Philips on 80s as a data exchange bus used to transfer data between the central processing unit (CPU) or microcontroller unit (MCU) of a device and peripheral chips. It was basically targeted for TV application. Due to its simplicity, it became so popular that after awhile it became one of the primary mechanisms of data transfer for CPUs and MCUs and peripheral devices that are not necessary part of the same PCB board and are connected to it via wire (e.g. sensors, display modules, etc.).

I2C consists of a communication bus made of two wire that supports bidirectional data transfer between a master and several slave devices. Typically the master node is in charge of controlling the bus – which is actually done by generating a synchronization signal on the serial clock line (SCL) . It's a signal that would be sent continuously by master during the transfer and all other nodes connected to the bus will use it to sync their communication and detect the speed of the bus. Data is transferred between the master and slave through a serial data (SDA) line. The transmission speed can be up to 3.4 Mbps. All devices that want to transfer data via I2C should have a unique address and can operate as either transmitter or receiver depending on the function of the device. For example an OLED display module is a receiver which accepts some data and displays them, while a temperature sensor is a transceiver that sends captured temperature via I2C bus. Normally a master device is the device that initiates a data transfer on the bus and generates the clock signals to permit the transfer. During that transfer, any device addressed by this master is considered a slave and reads that data.

3.1.3 DHT11 TEMPERATURE SENSOR

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability. This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance. Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program

memory, internal sensors detect signals in the process, we should call these calibration coefficients.

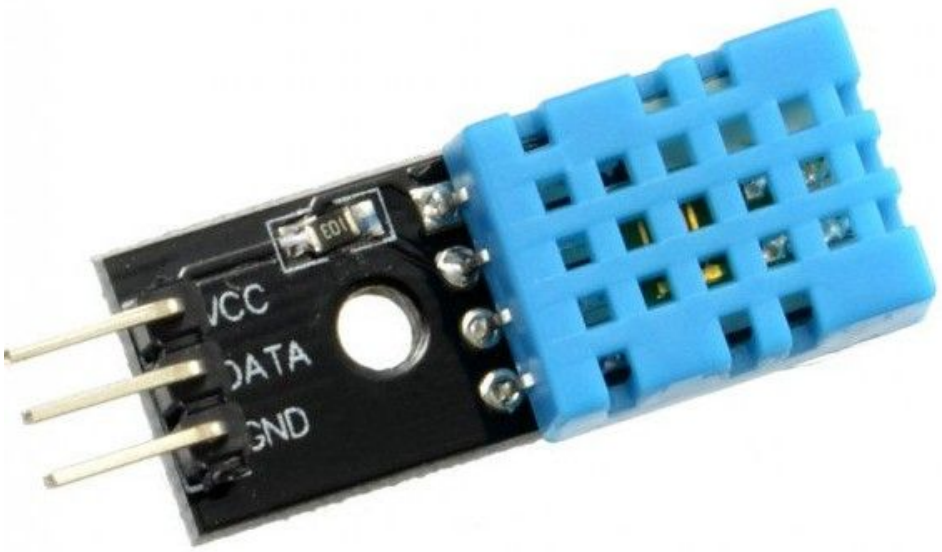

Figure 3.5 DHT11 Temperature and Humidity sensor

The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

3.1.3.1 SPECIFICATION

1. Supply Voltage: +5 V
2. Temperature range :0-50 °C error of ± 2 °C
3. Humidity :20-90% RH ± 5% RH error
4. Interface: Digital

3.1.4 YL-69 SENSOR & LM393 COMPARATOR

This moisture sensor can read the amount of moisture present in the soil surrounding it. It's a low tech sensor, but ideal for monitoring an urban garden, or your pet plant's water level.This is a must have tool for a connected garden. This sensor uses the two probes to pass current through the soil, and then it reads that resistance to get the moisture level. More water makes the soil conduct electricity more easily (less resistance), while dry soil conducts electricity poorly (more resistance). It will be helpful to remind you to water your indoor plants or to monitor the soil moisture in your garden.

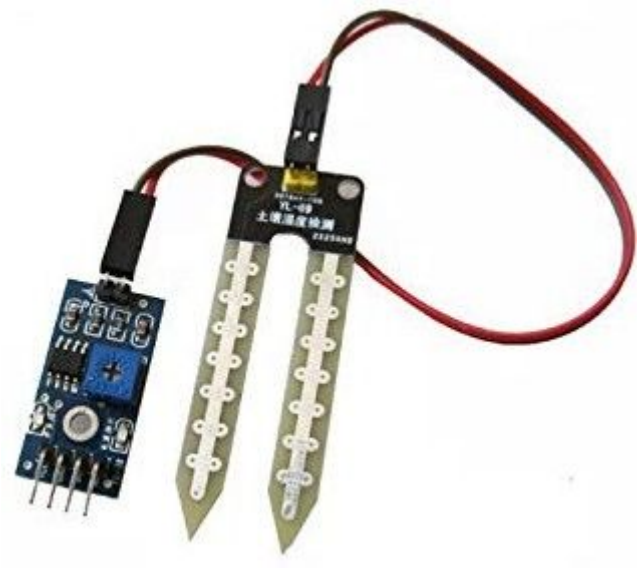

Figure 3.6 YL-69 sensor and LM393 Comparator

A Chinese built YL-69 sensors (Figure 3.6) come with a 'middle-man' circuit which allows to get two outputs: one is an analog readout of the resistance between the sensor's probes and the second is a digital output (essentially, HIGH or LOW, 5v or 0v) depending on whether the humidity is above or below a threshold which can in turn be adjusted by a built-in POTS. The YL-69 sensor has two pins which need to be wired to be the two pins on the YL-38 Bridge. On the other end of the YL-38 have four pins which represent VCC, GND, D0 and A0. VCC and GND are power pins which should set to 3.3/5V and ground respectively. A0 is an analog output. D0 is a digital output.

3.1.5 DS18B20 - DIGITAL TEMPERATURE SENSOR

This Maxim-made item is a digital thermo probe or sensor that employs DALLAS DS18B20. Its unique 1-wire interface makes it easy to communicate with devices. It can converts temperature to a 12-bit digital word in 750ms (max). Besides, it can measures temperatures from -55°C to +125°C (-67F to +257F). In addition, this thermo probe doesn't require any external power supply since it draws power from data line. Last but not least, like other common thermo probe, its stainless steel probe head makes it suitable for any wet or harsh environment. The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor.In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Figure 3.7 DS18B20 - High Quality Waterproof Digital Temperature Sensor

3.1.5.1 FEATURES

Power supply range:	3.0V to 5.5V
Operating temperature range:	-55°C to +125°C (-67F to +257F)
Storage temperature range:	-55°CC to +125°C (-67F to +257F)
Accuracy over the range of - 10°C to +85°C:	±0.5°C
3-pin 2510 Female Header Housing	
Waterproof Stainless steel sheath	
Stainless steel sheath	
Size of Sheath:	6*50mm
Connector:	RJ11/RJ12, 3P-2510, USB.
Pin Definition:	RED: VCC Yellow: DATA Black: GND
Cable length:	1 meter, 2m, 3m, 4m are available upon request

3.1.5.2 APPLICATION

The DS18B20 Digital Temperature Probe provides 9 to 12 bit (configurable) temperature readings which indicate the temperature of the device. Information is sent to/from the DS18B20 over a 1- Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source. Because each DS18B20 contains a unique silicon serial number, multiple DS 18B20s can exist on the same 1Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

3.1.6 5V RELAY

Relay is an electromagnetic device which is used to isolate two circuits electrically and connect them magnetically. They are very useful devices and allow one circuit to switch another one while they are completely separate. They are often used to interface an electronic circuit (working at a low voltage) to an electrical circuit which works at very high voltage. For example, a relay can make a 5V DC battery circuit to switch a 230V AC mains circuit. Thus a small sensor circuit can drive, say, a fan or an electric bulb.

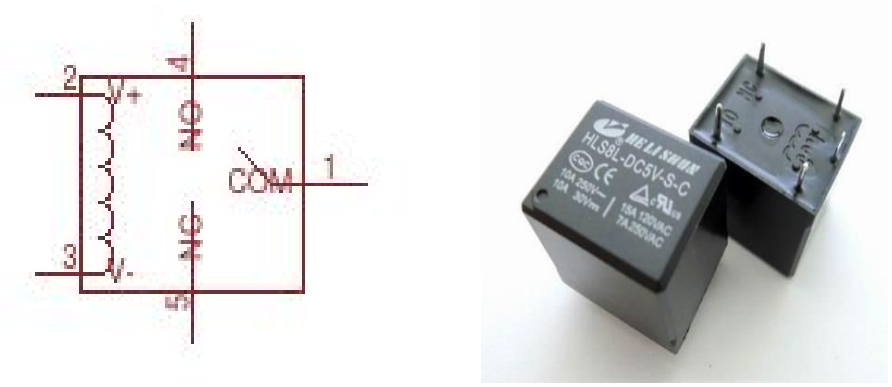

Figure 3.8 5V Relay module

A relay switch can be divided into two parts: input and output. The input section has a coil which generates magnetic field when a small voltage from an electronic circuit is applied to it. This voltage is called the operating voltage. Commonly used relays are available in different configuration of operating voltages like 6V, 9V, 12V, 24V etc. The output section consists of contactors which connect or disconnect mechanically. In a basic relay there are three contactors: normally open (NO), normally closed (NC) and common (COM). At no

input state, the COM is connected to NC. When the operating voltage is applied the relay coil gets energized and the COM changes contact to NO.

3.1.7 SUBMERSIBLE WATER PUMP

A submersible pump (or sub pump, electric submersible pump) is a device which has a hermetically sealed motor close-coupled to the pump body. The whole assembly is submerged in the fluid to be pumped. The main advantage of this type of pump is that it prevents pump cavitations, a problem associated with a high elevation difference between pump and the fluid surface. Small DC Submersible water pumps push fluid to the surface as opposed to jet pumps having to pull fluids. Submersibles are more efficient than jet pumps. It is usually operated between 3v to 12v.

Figure 3.9 5V Water Pump

3.1.7.1 SPECIFICATION

1. Voltage : 2.5-10V
2. Maximum lift : 40-110cm / 15.75"-43.4"
3. Flow rate : 80-120L/H
4. Outside diameter : 7.5mm / 0.3"
5. Inside diameter : 5mm / 0.2"
6. Diameter : Approx. 24mm / 0.95"
7. Length : Approx. 45mm / 1.8"
8. Height : Approx. 30mm / 1.2"
9. Material : Engineering plastic

10. Driving mode : DC design, magnetic driving
11. Continuous working life for 500 hours

3.1.8 TRANSISTOR (BC-547)

A BC547 transistor is a negative-positive-negative (NPN) transistor that is used for many purposes. Together with other electronic components, such as resistors, coils, and capacitors, it can be used as the active component for switches and amplifiers. Transistors has an emitter terminal, a base or control terminal, and a collector terminal. In a typical configuration, the current flowing from the base to the emitter controls the collector current. A short vertical line, which is the base, can indicate the transistor schematic for an NPN transistor, and the emitter, which is a diagonal line connecting to the base, is an arrowhead pointing away from the base.

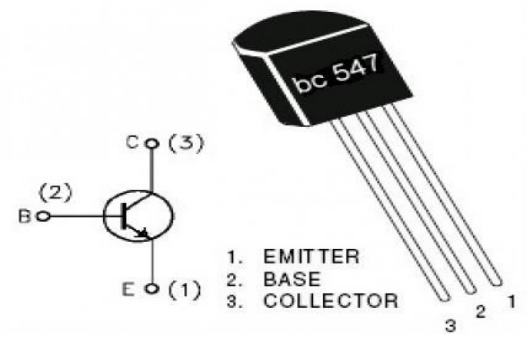

Figure 3.10 BC547 transistor

3.1.9 DIODE (IN4007)

In electronics, a diode is a two-terminal electronic component that conducts primarily in one direction (asymmetric conductance), it has low (ideally zero) resistance to the flow of current in one direction, and high (ideally infinite) resistance in the other. A semiconductor diode is a crystalline piece of semiconductor material with a p–n junction connected to two electrical terminals.

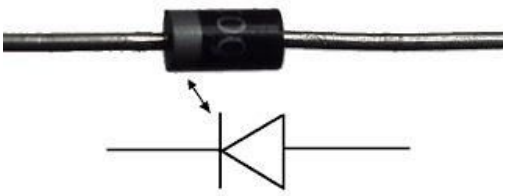

Figure 3.11 Diode (IN4007)

3.1.10 RELAY CIRCUIT

The relay circuit consists of relay switch, diode and transistor as shown in figure 3.12. This circuit controls 18-24v DC solenoid valve or 2.5-10v submersible motor. The control signal from controller to the base of transistor controls ON-OFF of actuators. The diode prevents the reverse flow of current in input end of the relay switch. At output end of relay switch a series connection of battery source and actuator. It is used because of actuators needs supply of 3-24v DC, but controller output signal is of 3.3v DC.

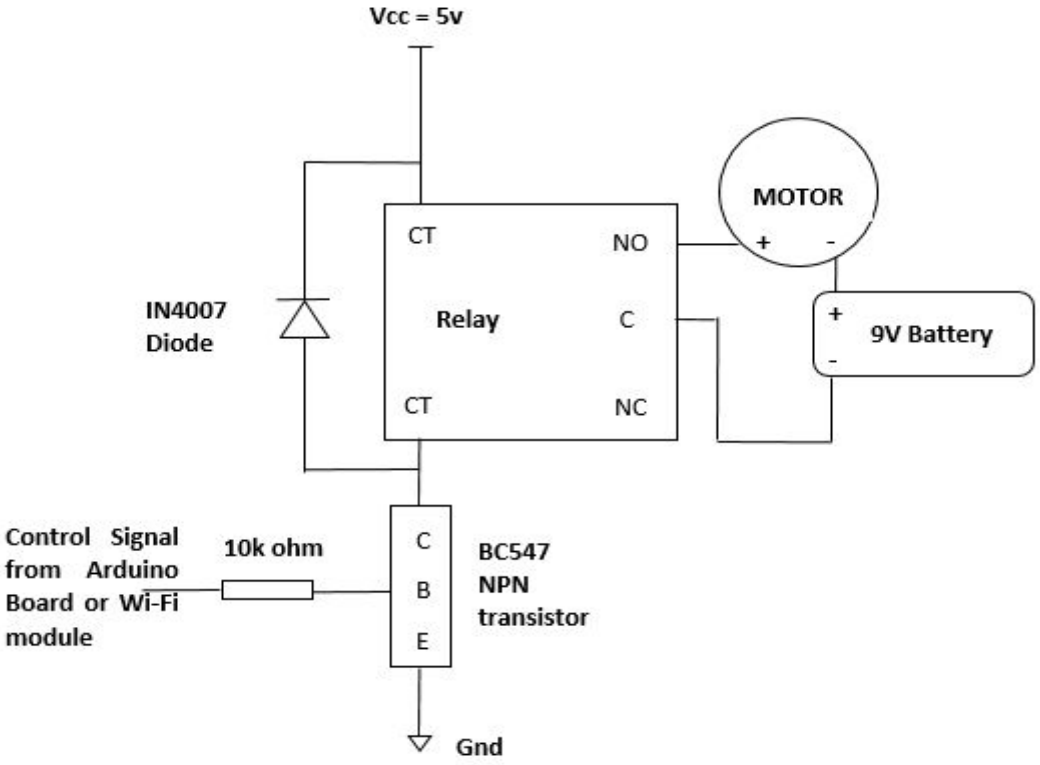

Figure 3.12 Relay Circuit

Chapter-4
SOFTWARE USED

CHAPTER 4 SOFTWARE USED

4.1 MONGODB

4.1.1 INTRODUCTION

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format)

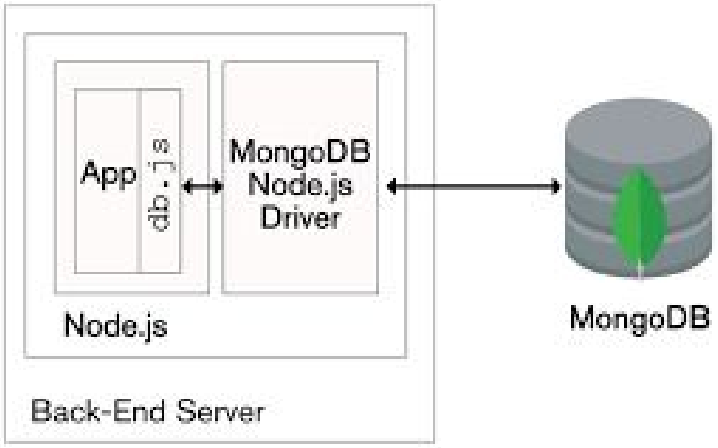

Figure 4.1 MongoDB block diagram

A simple MongoDB document Structure:

{

title: 'MY PROJECT',

by: 'BHUVNESH SINGH',

url: 'www.xyz.com',

type: 'NoSQL'

}

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

4.1.2 FEATURES OF MONGODB

1. **Document Oriented**: MongoDB stores the main subject in the minimal number of documents and not by breaking it up into multiple relational structures like RDBMS. For example, it stores all the information of a computer in a single document called Computer and not in distinct relational structures like CPU, RAM, Hard disk, etc.
2. **Indexing**: Without indexing, a database would have to scan every document of a collection to select those that match the query which would be inefficient. So, for efficient searching Indexing is a must and MongoDB uses it to process huge volumes of data in very less time.
3. **Scalability**: MongoDB scales horizontally using sharding (partitioning data across various servers). Data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that resides across many physical servers. Also, new machines can be added to a running database.
4. **Replication and High Availability**: MongoDB increases the data availability with multiple copies of data on different servers. By providing redundancy, it protects the database from hardware failures. If one server goes down, the data can be retrieved easily from other active servers which also had the data stored on them.
5. **Aggregation**: Aggregation operations process data records and return the computed results. It is similar to the GROUP BY clause in SQL. A few aggregation expressions are sum, avg, min, max, etc.

6. **Distributed data** Since multiple copies of data are stored across different servers, recovery of data is instant and safe even if there is a hardware failure.

4.1.3 LANGUAGE SUPPORT BY MongoDB:

MongoDB currently provides official driver support for all popular programming languages like C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang.

4.2 NODE JS SERVER

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

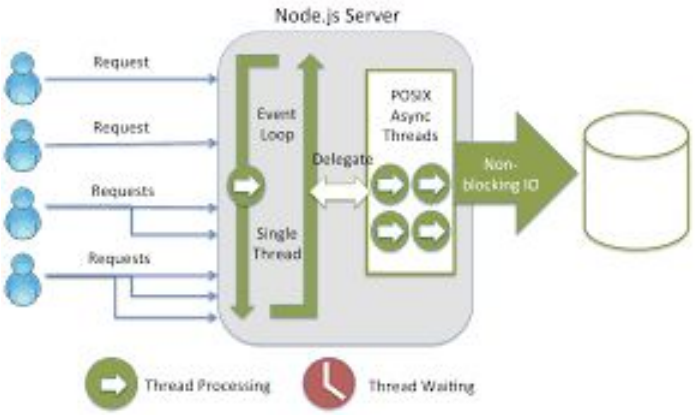

Figure 4.2 Node.JS Server Block Diagram

Though .js is the standard filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

Figure 4.3 Node.js Icon

The Node.js distributed development project, governed by the Node.js Foundation, is facilitated by the Linux Foundation's Collaborative Projects program.

4.2.1 WHAT IS NODE.JS?

Node.js is an open source server framework, completely free, and used by thousands of developers around the world. Node.js is an open source runtime environment for server-side and networking applications and is single threaded.

- It uses Google JavaScript V8 Engine to execute code.
- It is a cross platform environment and can run on Microsoft Windows Linux, FreeBSD and IBM
- It provides an event driven architecture and non-blocking I/O that is optimised & scalable.

4.2.2 WHY NODE.JS ?

These are the following reasons for using Node.js:-

1. Node.js uses Asynchronous programming.
2. Node.js eliminates the waiting, and simply proceeds with the next request.
3. Node.js runs single threaded, non-blocking, asynchronously programming, which is very memory efficient.

4.3 ARDUINO IDE

1. Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
2. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
3. It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
4. A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
5. Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.

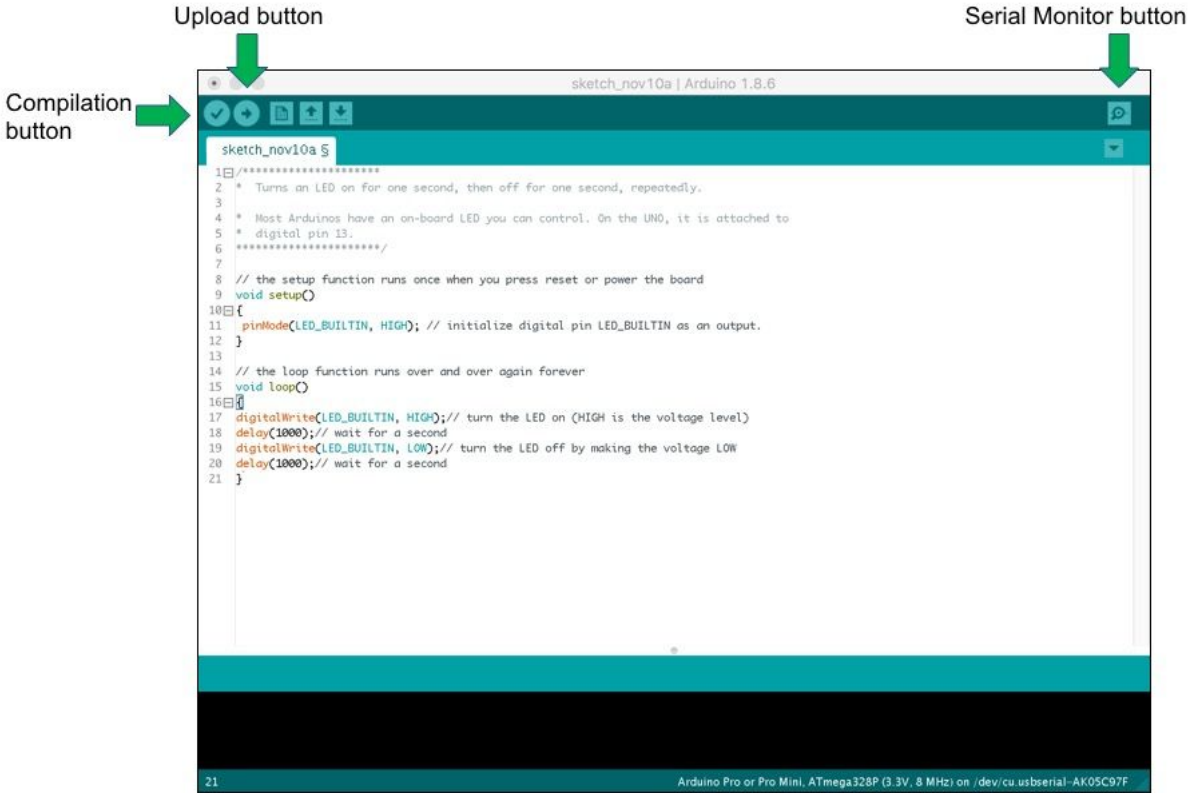

Figure 4.4 Arduino IDE software visualization

6. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
7. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
8. This environment supports both C and C++ languages.

4.3.1 CONCLUSION

There are plenty of other features available to consider on the IDE. But, having used many different types of microcontrollers and having been involved in multiple programming environments, it is shocking how simple the Arduino and its IDE is! In less than two minutes, you can get a simple C++ program uploaded onto the Arduino and have it running.

4.4 ROBO 3T

Robo 3T is a free and lightweight GUI for MongoDB. It is a MongoDB management tool which has a shell-centric cross-platform and is supported by JSON i.e. JavaScript Object Notation. This tool is not typical of MongoDB's other administrative tools of user Interface i.e. its shell could get embedded in Mongo Shell with a whole lot of access in both Mongo CLI and Mongo GUI. With the help of this mongo shell, a user could view, edit and delete mongo documents. Moreover, Robo 3T is a volunteered open source project and it's totally free of cost to the public.

Figure 4.5 Robo 3T ICON

It could be re-disseminate and could get re-modified as well by following the TOS of General public license version 3, which has been published by Free Software Foundation. This software has been promulgated and could be redistributed for the purpose of helping people who could get assistance from it, that's why it bears no Warranty of wholesaling it, as per the rules by GNU.

4.4.1 WHY ROBO 3T?

Robo 3T is a free and machine friendly software, that uses a small number of one's resources available on a machine. It's highly appreciated and recognized as the world-famous project with highest success ratio in giving prime output. The user doesn't have to go through the messy procedure of using tables and rows, which is typically used in rational databases. Unlike them, it is built on architecture Mongo collections and Mongo documents.

4.5 GRAFANA

Grafana is a beautiful dashboard for displaying various Graphite metrics through a web browser. Grafana is nice because it is simple to set up and maintain and is easy to use and displays metrics in a very nice Kibana like display style. I would like to walk readers through the basics of this tool because although it is a very new project (beginning of 2014) it has an enormous amount of potential and I honestly believe that there is enough functionality to put it into production.

Figure 4.6 Grafana ICON

The guys over at Hosted Graphite have just recently released hosted Grafana as part of their standard plan. If you are interested you can check it out over at Hosted Graphite. One very nice feature of this product offering is that you do not need to worry about any of the behind the scenes details or intricacies of how all of the Graphite components work together. You just click a few buttons and you are ready to go. Highly recommended if you just need something that works, go check it out. Anyway, Grafana gives you the ability to bolt on all kinds of bells and whistles to your graphs. For example, there is a nice little node.js tool called statsd written by the guys over at etsy that sort of expands the capabilities of Graphite. And since it hooks right in with Graphite, it makes it very simple to represent and output the various metrics into Grafana.

If you do choose to roll your own Grafana solution, there are a few gotchas that I was not aware of that I'd like to cover. The first, which should seem easy enough is that you need to run but Graphite and Grafana simultaneously. So that means you either need to create two virtual hosts depending on which web server you choose to use or you need to to create to different port bindings. The Grafana documentation has a few examples of how to create new port binding, which uses Apache as the web server but it is pretty easy to find examples of

configs on Github and other sites if you are interested in using nginx as your web server instead. The important thing to remember is that you will want to have two sites listed in your sites-enabled directory inside of the apache directory. One for Graphite (which I moved to port 8080 for simplicity) and one for Grafana (which I stuck on port 80).

Figure 4.7 Grafana dashboard software visualization

If you choose to use authentication there are a few extra headers that need to be written as well so that Grafana is accessible correctly. This is easy to add to either your nginx or you apache config.

4.6 BLYNK APP SERVER

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

There are three major components in the platform:

1. **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
2. **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
3. **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to space the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a Blynk of an eye.

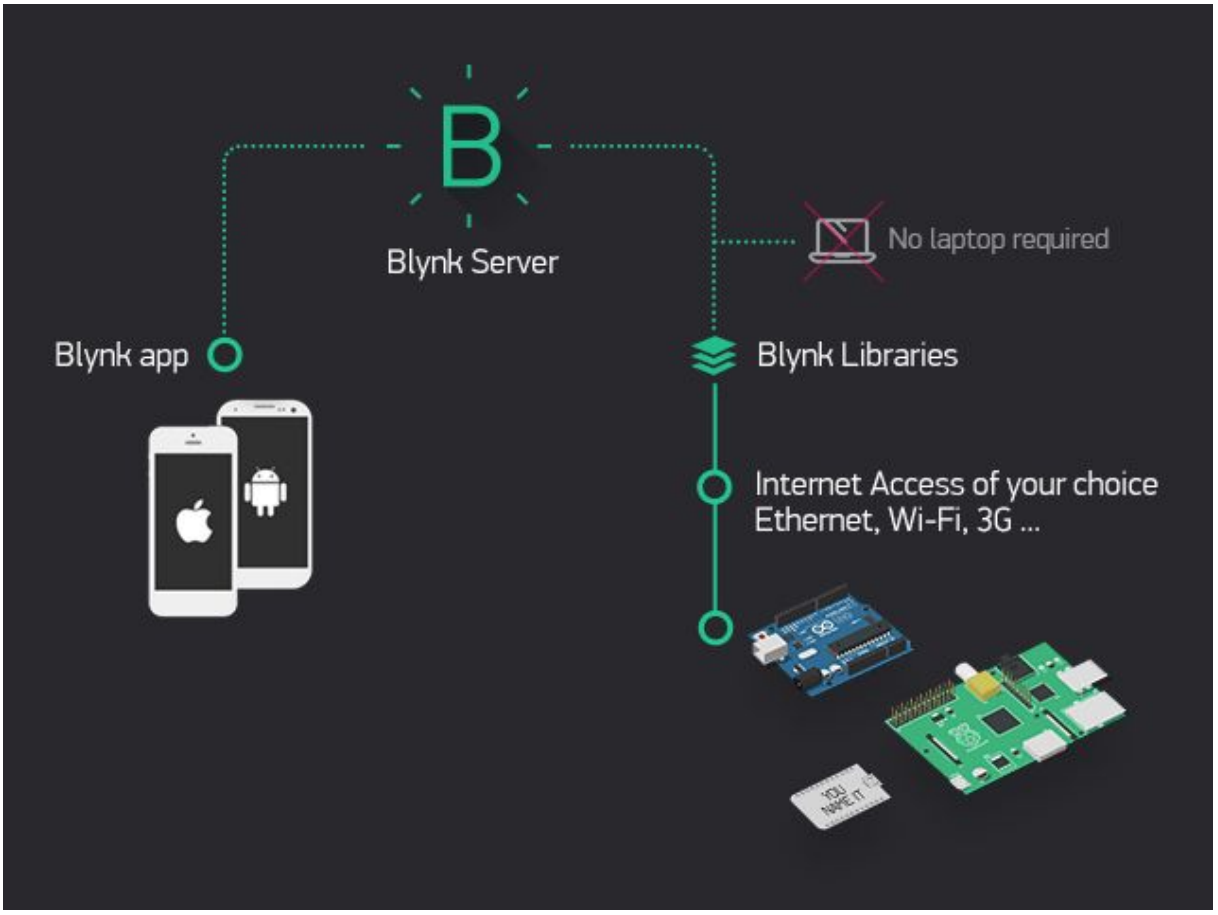

Figure 4.8 Blynk WEb Server working visualization

4.6.1 FEATURES

1. Similar API & UI for all supported hardware & devices
2. Connection to the cloud using:
    i. WiFi
    ii. Bluetooth and BLE
    iii. Ethernet
    iv. USB (Serial)
    v. GSM
3. Set of easy-to-use Widgets
4. Direct pin manipulation with no code writing
5. Easy to integrate and add new functionality using virtual pins
6. History data monitoring via SuperChart widget

7. Device-to-Device communication using Bridge Widget
8. Sending emails, tweets, push notifications, etc.

You can find example sketches covering basic Blynk Features. They are included in the library. All the sketches are designed to be easily combined with each other.

4.6.2 WHAT DO I NEED TO BLYNK?

At this point you might be thinking Just a couple of things, really:

1. Hardware:-Blynk works over the Internet. This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, raspberry pi with WiFi dongle, Particle Photon or SparkFun Blynk Board. But even if you don't have a shield, you can connect it over USB to your laptop or desktop (it's a bit more complicated for newbies, but we got you covered). What's cool, is that the list of hardware that works with Blynk is huge and will keep on growing.

2. A Smartphone:-The Blynk App is a well designed interface builder. It works on both iOS and Android, here, ok?

Chapter-5
PROJECT PROCESS

CHAPTER 5 PROJECT PROCESS

5.1 INSTALLING THE SOFTWARES

Here the process will start with software installation. We will install software one by one. There are some software that will installed for complete the project.

Note:- *In this project we did use linux ubuntu operating system.*

I. Node.js server
II. MongoDB
III. Arduino IDE
IV. Prometheus
V. Robo 3T

5.1.1 INSTALL NODEJS SERVER

Ubuntu 16.04 contains a version of Node.js in its default repositories that can be used to easily provide a consistent experience across multiple systems. At the time of writing, the version in the repositories is v4.2.6. This will not be the latest version, but it should be quite stable and sufficient for quick experimentation with the language.

In order to get this version, we just have to use the apt package manager. We should refresh our local package index first, and then install from the repositories:

$ sudo apt-get update

$ sudo apt-get install nodejs

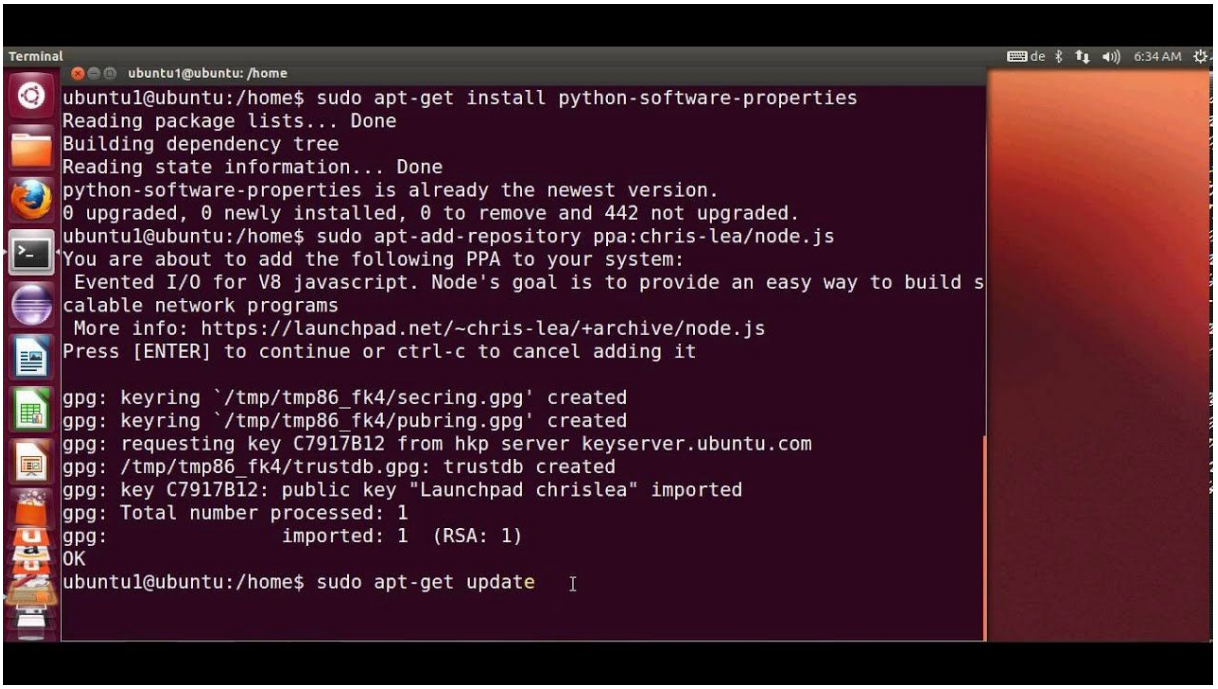

Figure 5.1 node.js server installation

If the package in the repositories suits your needs, this is all you need to do to get set up with Node.js. In most cases, you'll also want to also install npm, which is the Node.js package manager. You can do this by typing:

$ sudo apt-get install npm

This will allow you to easily install modules and packages to use with Node.js. Because of a conflict with another package, the executable from the Ubuntu repositories is called nodejs instead of node. Keep this in mind as you are running software. To check which version of Node.js you have installed after these initial steps, type:

$ node -v

Once you have established which version of Node.js you have installed from the Ubuntu repositories, you can decide whether or not you would like to work with different versions, package archives, or version managers. Next, we'll discuss these elements along with more flexible and robust methods of installation.

5.1.2 INSTALL MongoDB

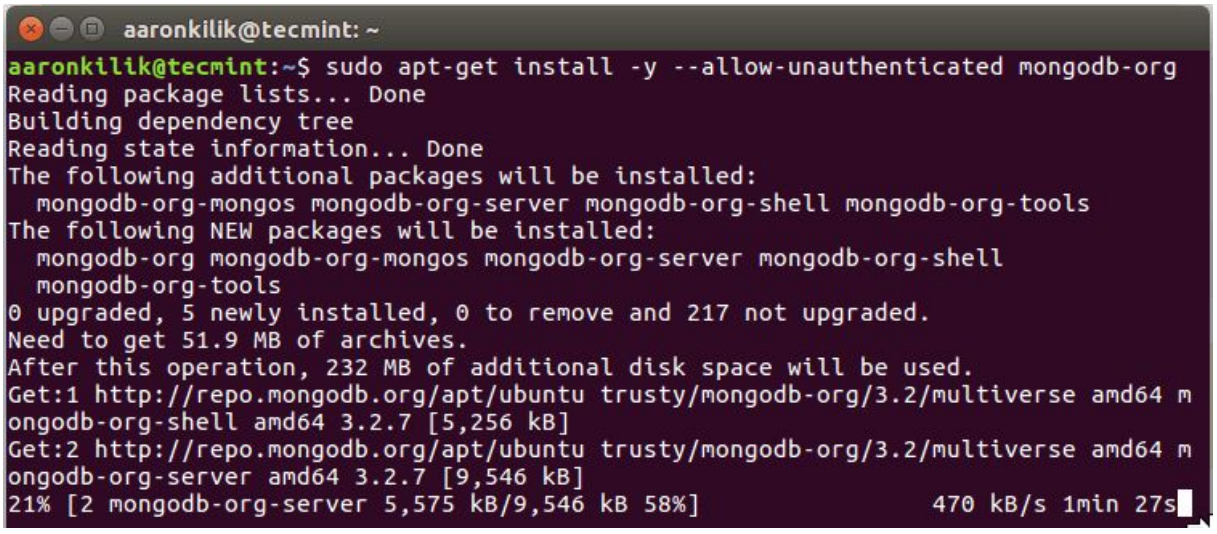

Figure 5.2 mongoDB installing

1. Install the latest stable version of MongoDB:

```
> sudo apt-get install -y --allow-unauthenticated  mongodb-org
```

2. Install a specific release of MongoDB:
3. You must specify each component package specifically with their version number, check the following example:

```
> sudo apt-get install -y mongodb-org=3.4 mongodb-org-server=3.4 mongodb-org
```

5.1.3 INSTALL ARDUINO IDE

1. Go to the link: https://www.arduino.cc/en/Main/Software .

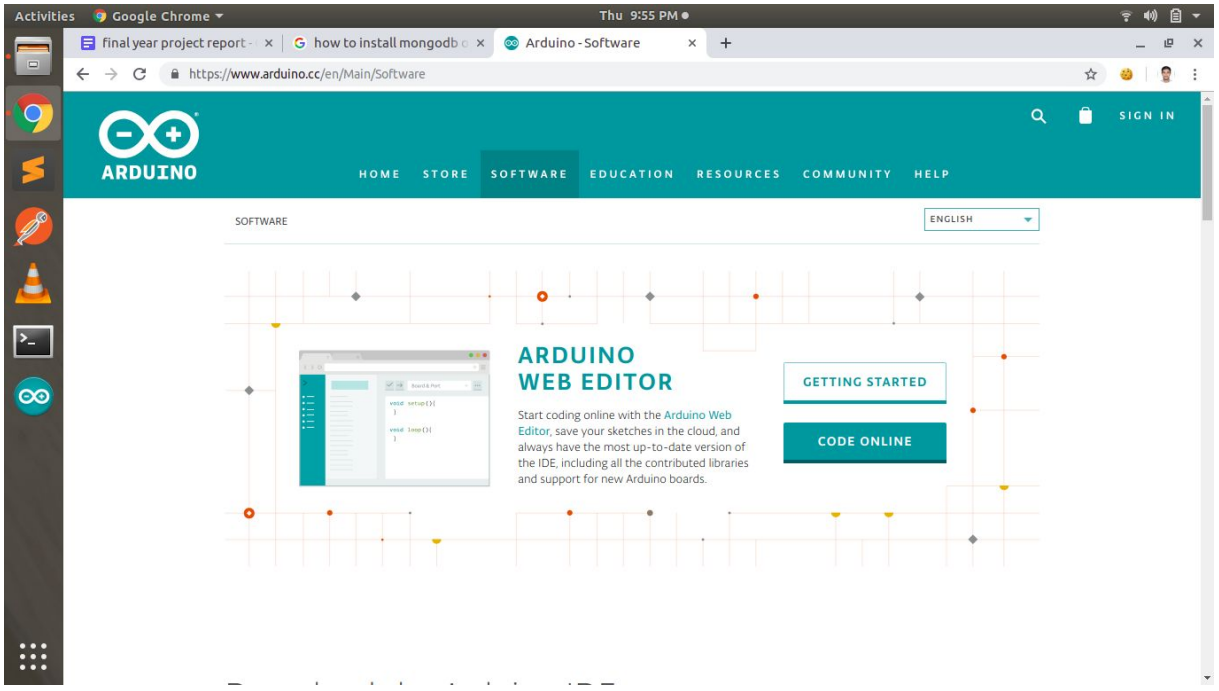

Figure 5.3 Arduino installing link open

2. You can choose between the 32, 64 and ARM versions. It is very important that you choose the right version for your Linux distro. Clicking on the chosen version brings you to the donation page and then you can either open or save the file. Please save it on your computer.

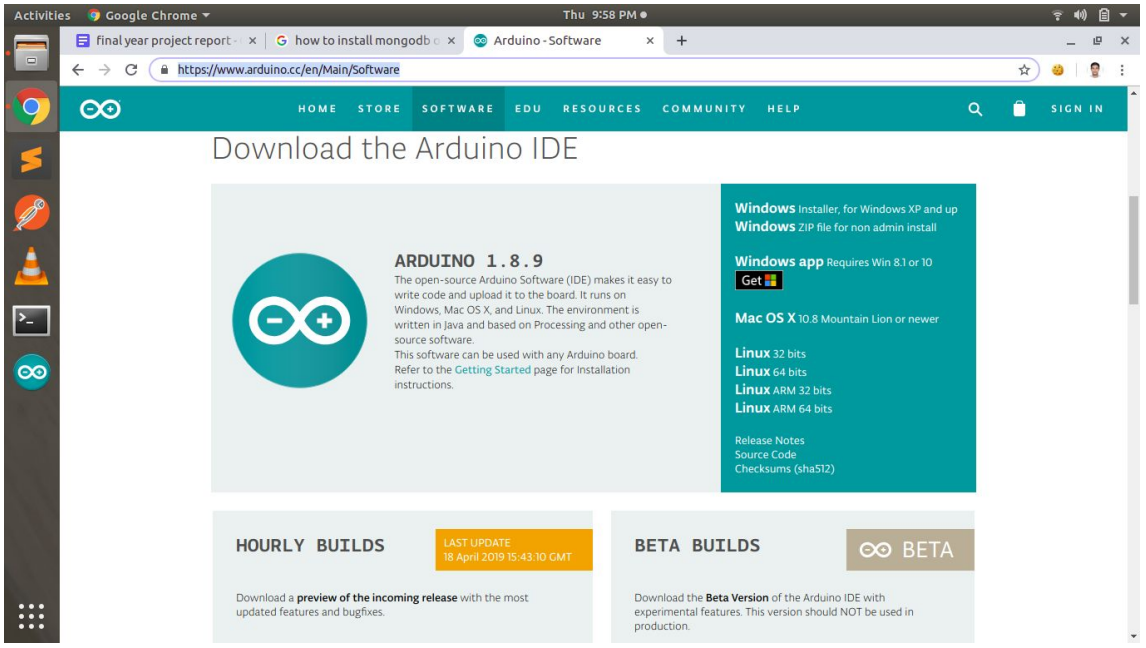

Figure 5.4 arduino ide downloading

3. The file is compressed and you have to extract it in a suitable folder, remembering that it will be executed from there.

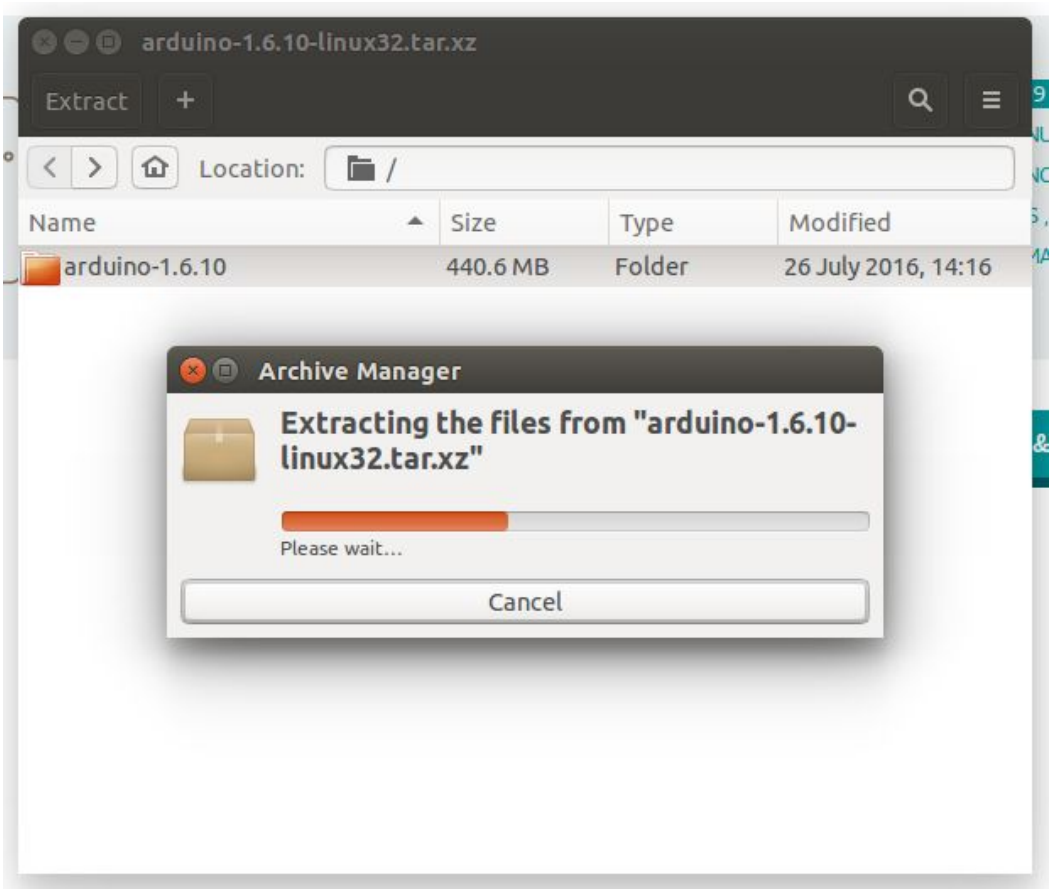

Figure 5.5 Arduino file extracting

4. Open the arduino-1.6.x folder just created by the extraction process and spot the install.sh file. Right click on it and choose Run in Terminal from the contextual menu. The installation process will quickly end and you should find a new icon on your desktop.

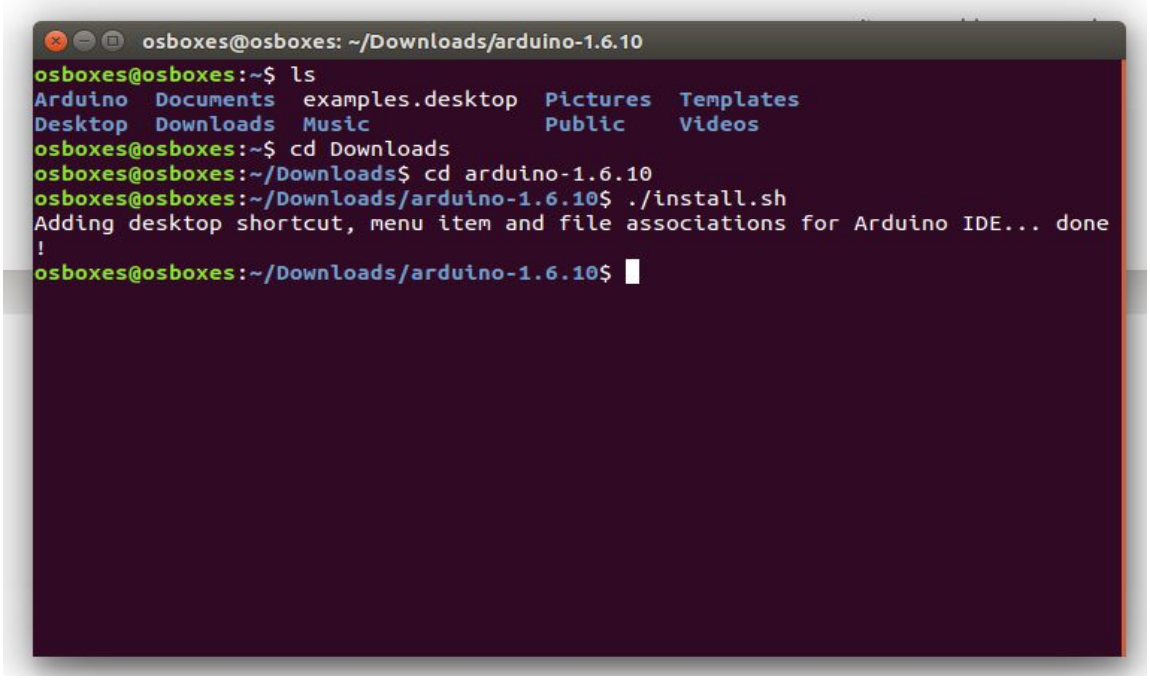

Figure 5.6 Arduino software open command

5.1.4 INSTALL PROMETHEUS

Prometheus is a free and open source software ecosystem that allows us to collect metrics from our applications and stores them in a database, especially a time-series based DB. It is a very powerful monitoring system suitable for dynamic environments. Prometheus is written in Go and use query language for data processing. Prometheus provides metrics of CPU, memory, disk usage, I/O, network statistics, MySQL server and Nginx.

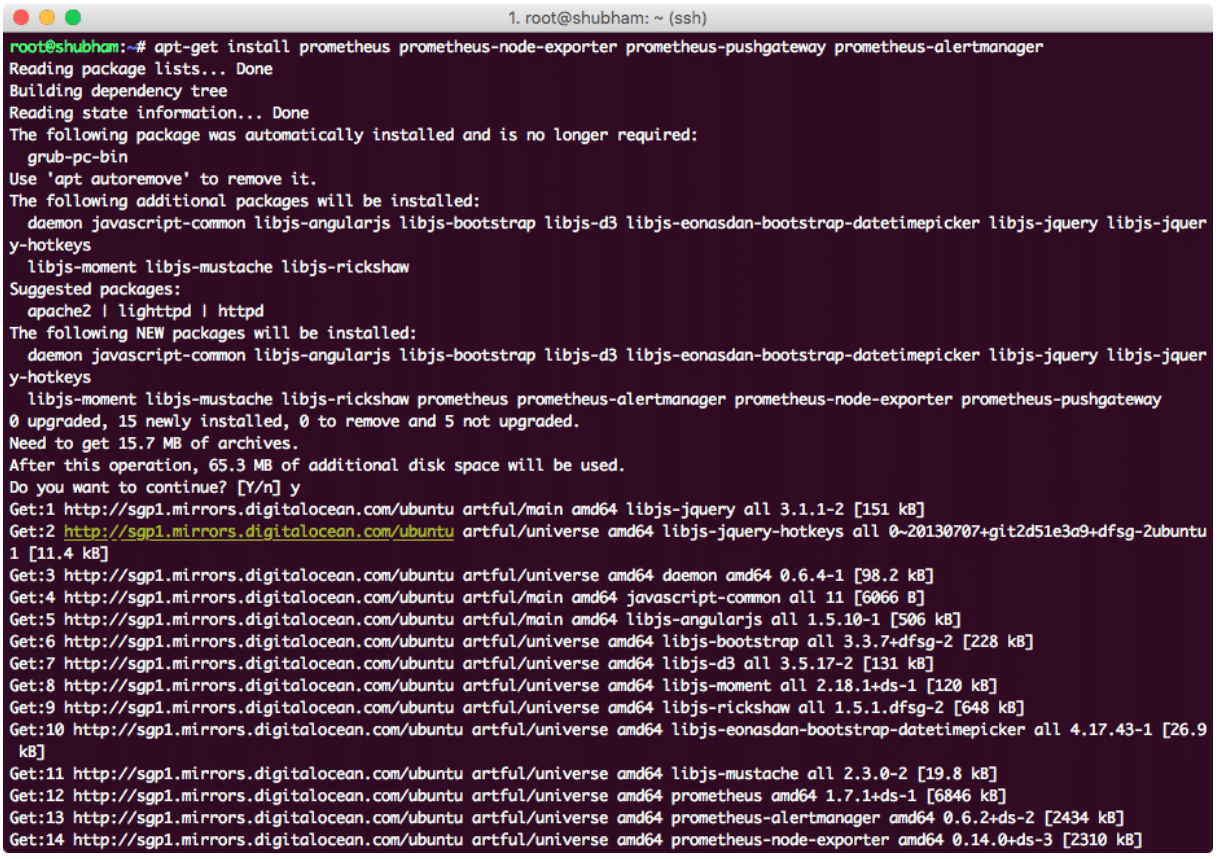

Figure 5.7 Prometheus installing

First, download and add the GPG key with the following command:

```
$ wget https://s3-eu-west-1.amazonaws.com/deb.robustperception.io/41EFC99D.gpg | sudo apt-key add -
```

Next, update the repository and install Prometheus with the following command:

```
$ sudo apt-get update -y
$ sudo apt-get install prometheus prometheus-node-exporter prometheus-pushgateway prometheus-alertmanager-y
```

Once the installation is completed, start Prometheus service and enable it to start on boot time with the following command:

```
$ sudo systemctl start prometheus
```

```
$ sudo systemctl enable prometheus
```

You can also check the status of Prometheus service with the following command:

```
$ sudo systemctl status prometheus
```

5.1.5 INSTALL ROBO 3T

1. Download the robomongo: https://robomongo.org/download
2. Extract the .tar.gz downloaded from above
3. Change directory to extracted folder.
4. You'll find a bin folder. Go in there, then double click on robomongo.

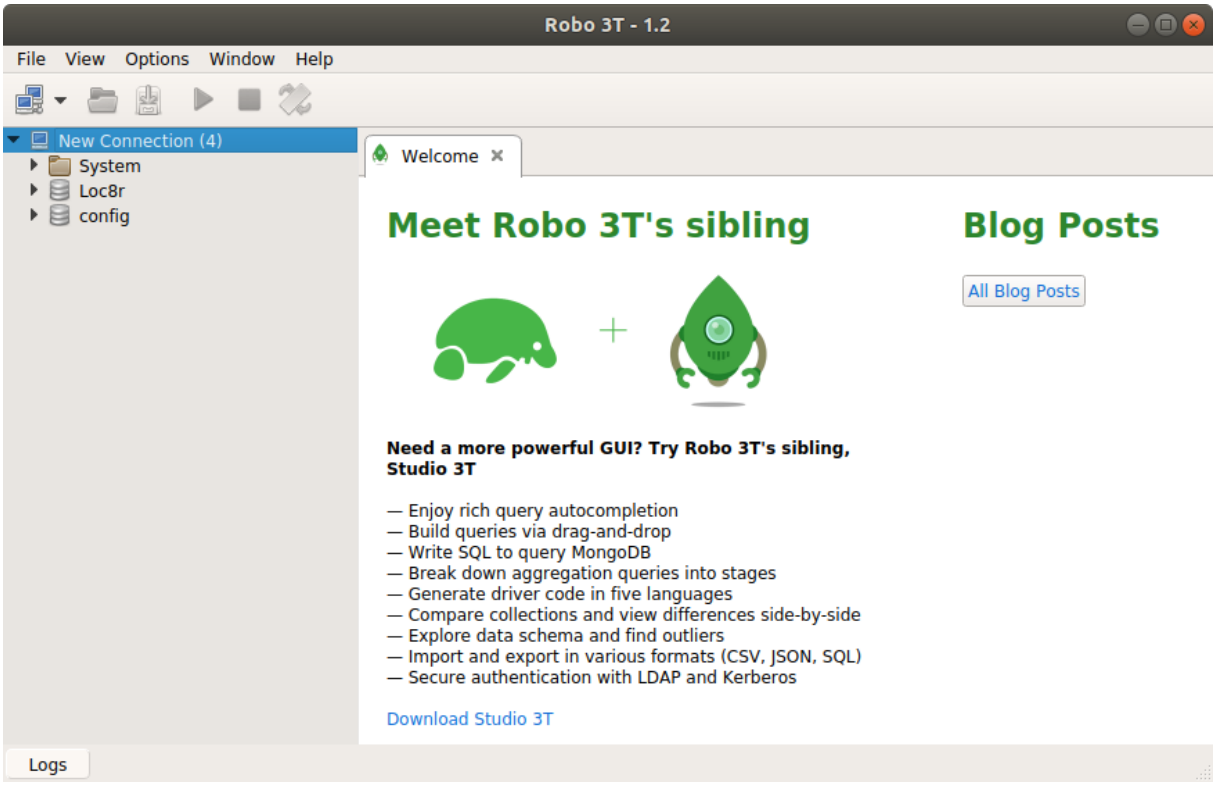

Figure 5.8 Robo3t dashboard

Anytime you'll wanna run robomongo, you will have to do something like this from terminal:

```
$ /path/to/robomongo_dir/bin/robomongo
```

You might wanna add a link to robomongo from your `/usr/bin` which will allow you to do something like this anywhere from terminal:

5.2 CIRCUIT PROCESSING

Here we will discuss about the circuit processing and will show how all devices & sensors modules connection with nodemcu microcontroller one by one.

The circuit is consist between two different parts that are given below:-

1. **Controlling & monitoring section**
2. **Sensors section**

5.2.1 SENSOR SECTION

In this section circuit all the sensors are connected as following:-

5.2.1.1 CAPTURING AIR TEMPERATURE & HUMIDITY

One of most used sensors for capturing weather data is the DHT22 (or it's brother DHT11), a digital relative humidity and temperature 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).

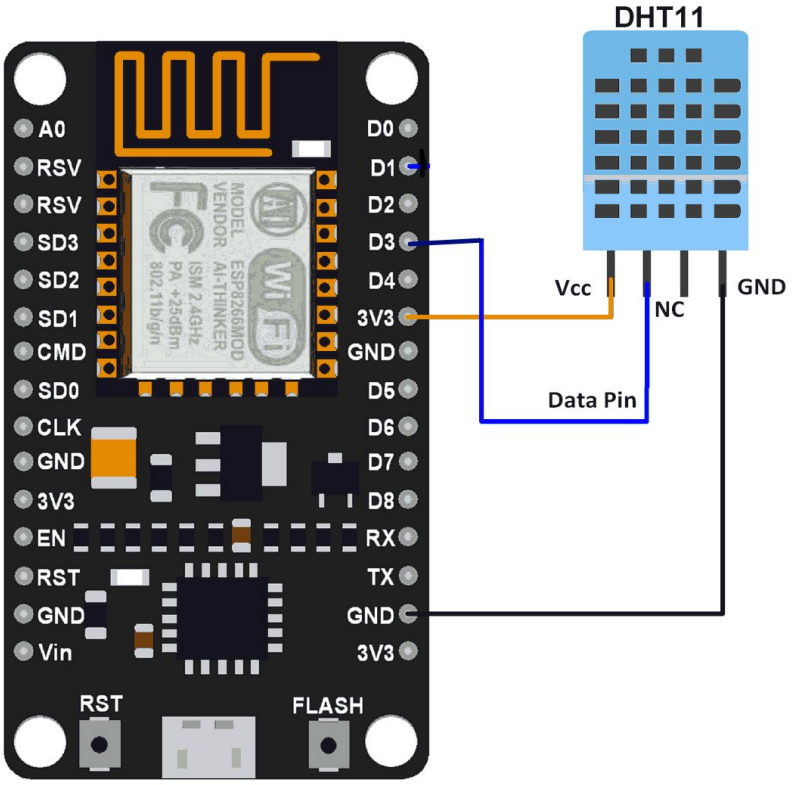

Figure 5.9 installing the temperature and humidity sensors

The sensor should be powered between 3.3V and 5V and will work from -40oC to +80oC with an accuracy of +/- 0.5oC for temperature and +/-2% for relative Humidity. It is

also important to have in mind that the its sensing period is in average 2 seconds (minimum time between readings). The site of Adafruit provides a lot of information about both, DHT22 and its brother DHT11. The DHT22 has 4 pins (facing the sensor, pin 1 is the most left):

- VCC (we will connect to 3.3V from NodeMCU);
- Data out;
- Not connected and
- Ground.

Once usually you will use the sensor on distances less than 20m, a 10K resistor should be connected between Data and VCC pins. The Output pin will be connected to NodeMCU pin D3 (see the diagram above).

5.2.1.2 CAPTURING SOIL MOISTURE & HUMIDITY

There we have explored a DIY type of sensor; here let's use a electronic one, very common in the market: the YL-69 sensor and LM393 Comparator module soil medium Hygrometer.

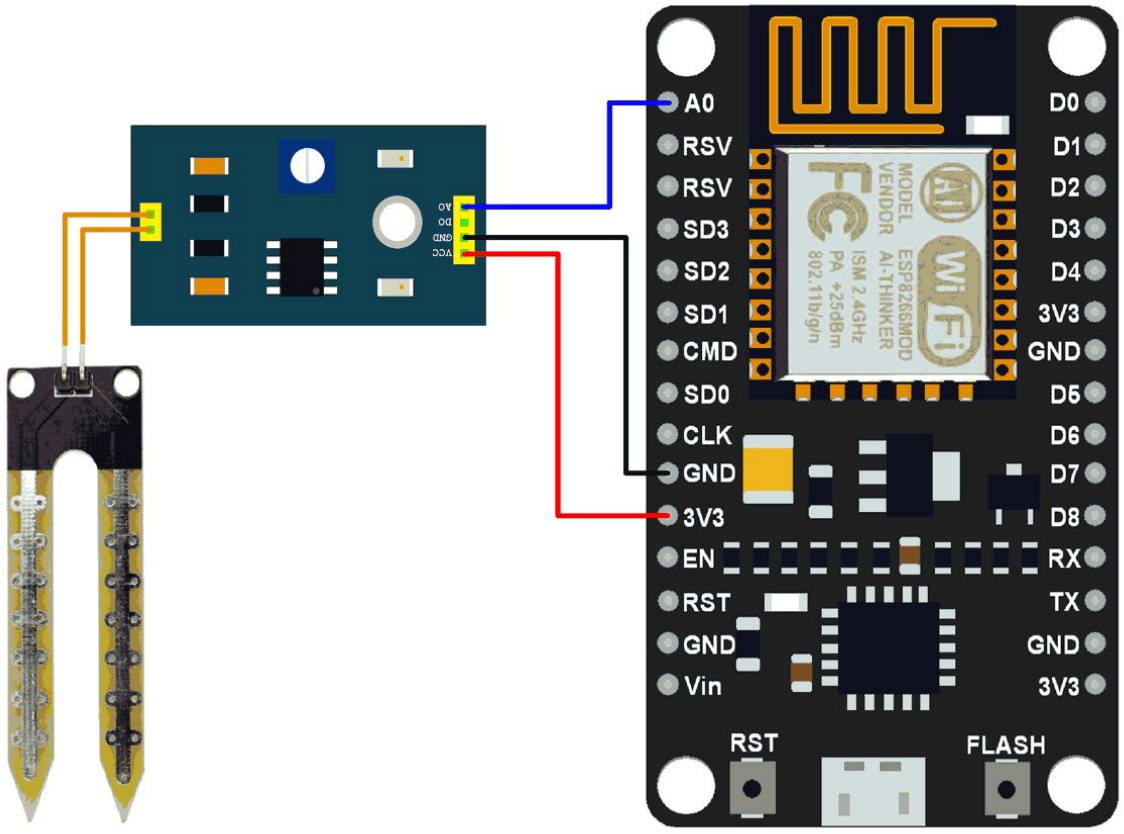

Figure 5.10 installing the soil moisture and humidity sensors

The LM393 module has 2 outputs, one digital (D0) that can be set-up using the potentiometer that exist on it and an analog one (A0). This module can be sourced with 3.3V,

what is very convenient when working with an NodeMCU. What we will do is install the LM393 4 pins as below:

1. LM393 A0 output to A0 NodeMCU A0 input
2. LM393 VCC to NodeMCU VCC or to NodeMCU GPIO D3*
3. LM393 GND to NodeMCU GND
4. LM393 D0 open

It's important to highlight that the correct is to connect the Sensor VCC to a Digital Pin as output, so the LM393 will be powered only when we need a read. This is important no only to save power, but also to protect the probes from corrosion. the NodeMCU did not worked well the soilMoisterVcc PIN connected. Also I had eventual errors due the power consumption. So, I powered the LM393 direct to VCC (5V), the code does not need to be changed. It is worked fine.

5.2.1.3 CAPTURING SOIL TEMPERATURE

We will use on this project, a waterproofed version of the DS18B20 sensor. It is very useful for remote temperature on wet conditions, as an humid soil. The sensor is isolated and can take measurements until 125 degree C.

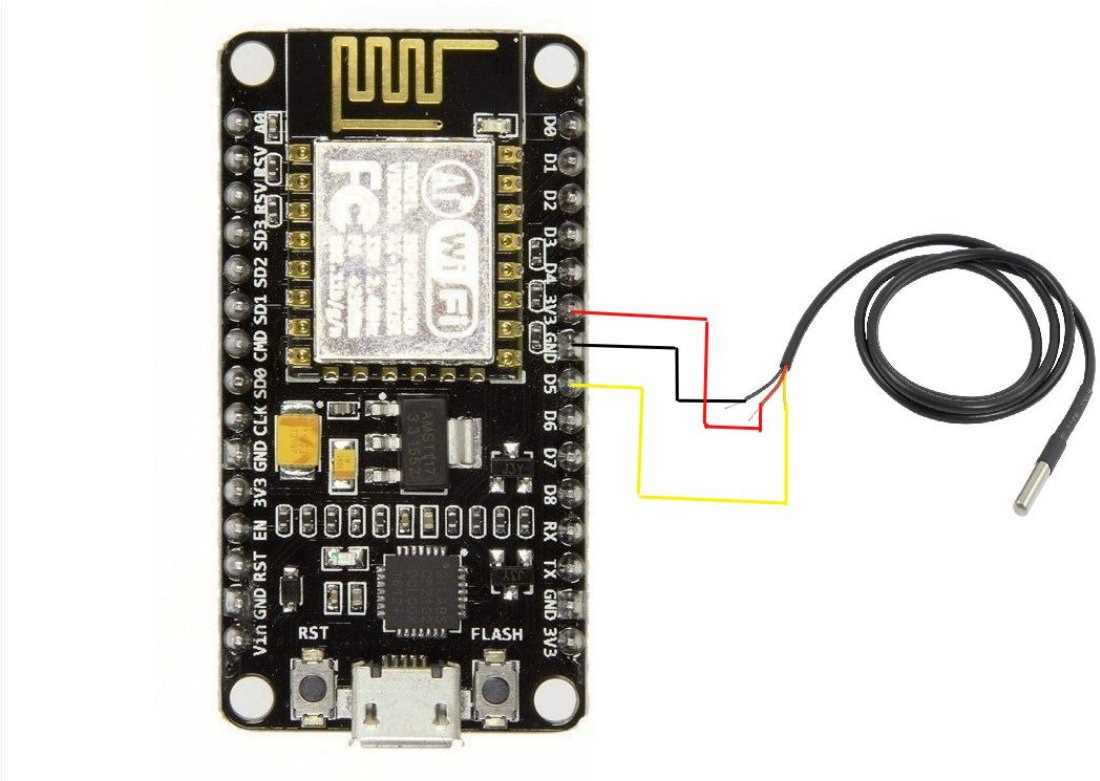

Figure 5.11 installing the Soil Temperature sensors

5.2.1.4 COMPLETE THE SENSOR HARDWARE

To complete the sensors hardware all the sensors are easily mounted on a single PCB shown as figure 5.12 below:-

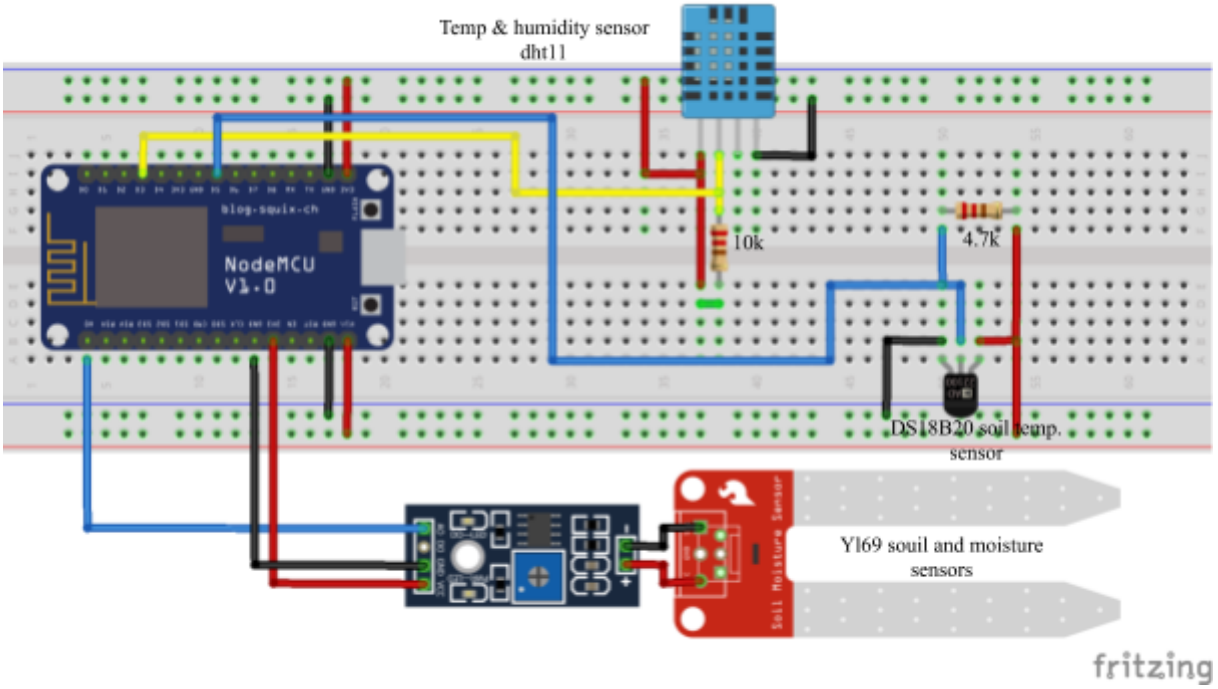

Figure 5.12 installing the sensor full hardware circuit

The circuit diagram of this circuit can be shown as below:-

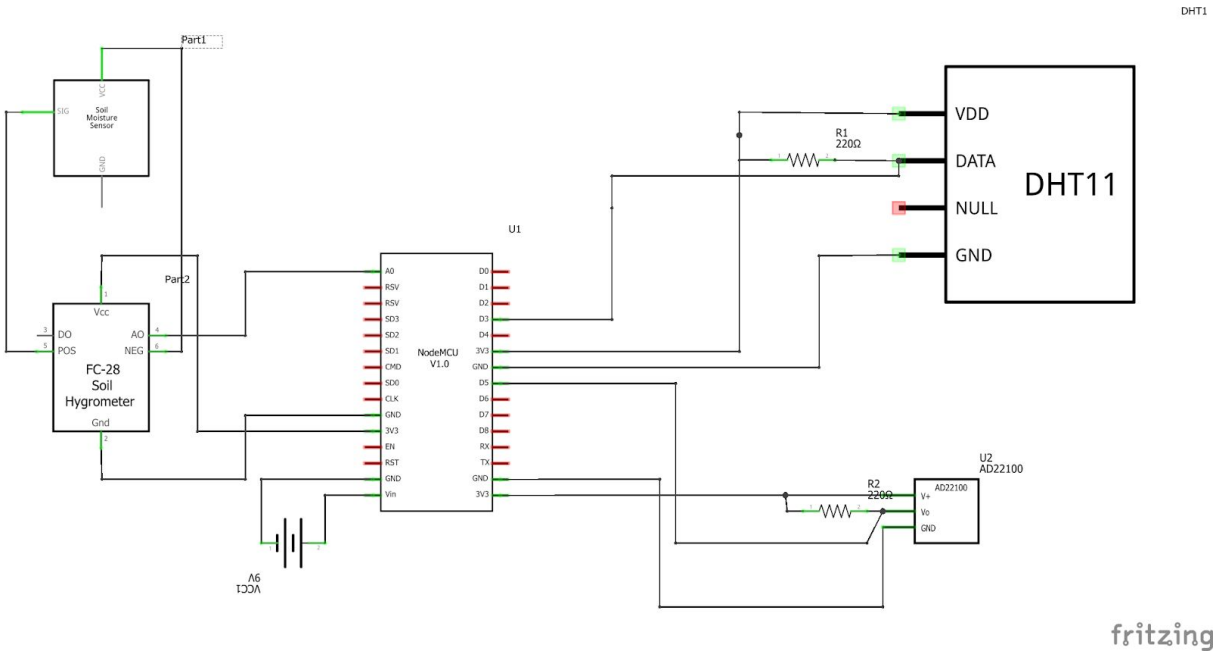

Figure 5.13 circuit Diagram Of Sensors Part PCB

5.2.2 CONTROLLING & MONITORING SECTION

This section is used to control and show the data on LED screen and manual control with button as shown as following:-

5.2.2.1 INSTALLING THE OLED

In order to verify the collected data locally, a small OLED Display should be installed. It is a I2C display, so we will connect it to the NodeMCU I2C pins, using:-

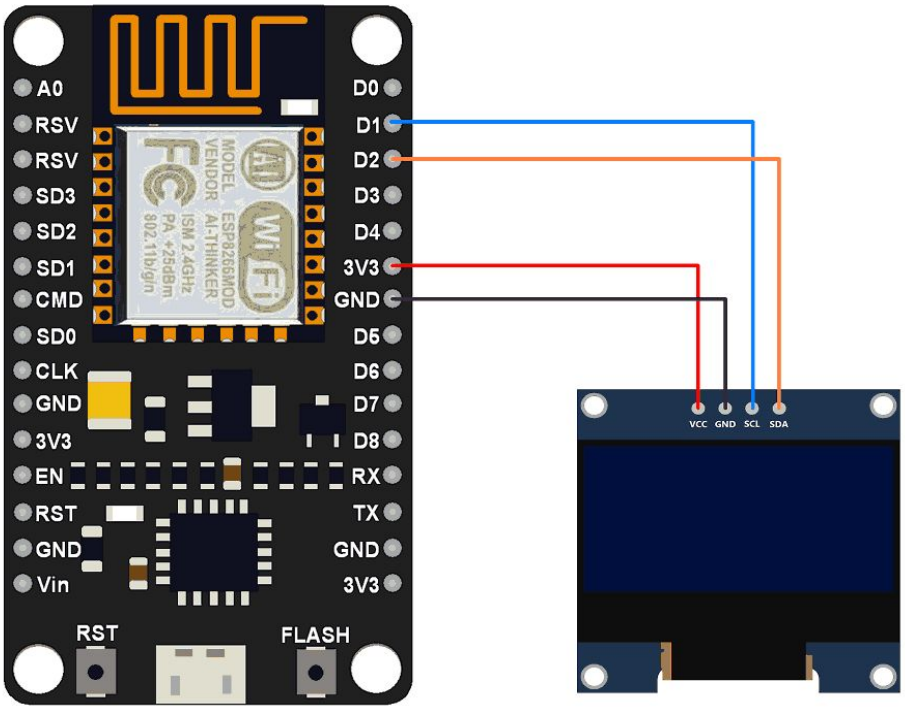

Figure 5.14 installing the Oled display

1. SCL ==> D1 (5)
2. SDA ==> D2 (4)
3. 128 pixels at horizontal and 64 pixels at vertical. So if you use 8x8 characters, we will get a "16X8" Display (8 lines of 16 characters each).

The SSD1306 can be powered with 5V (external) or 3.3V directly from the NodeMCU module. The first option was the chosen one (5V).

5.2.2.2 FULL HARDWARE OF CONTROLLING SECTION

Following the above diagrams, complete the required system HW installing the buttons and LEDs.

- **LED**

Note that LEDs connected on NodeMCU, are for testing only. They will "simulate", the Pump (Red LED) and the Lamp (Green LED). For the final circuity the Relays will connected to those outputs as described on the next Step.

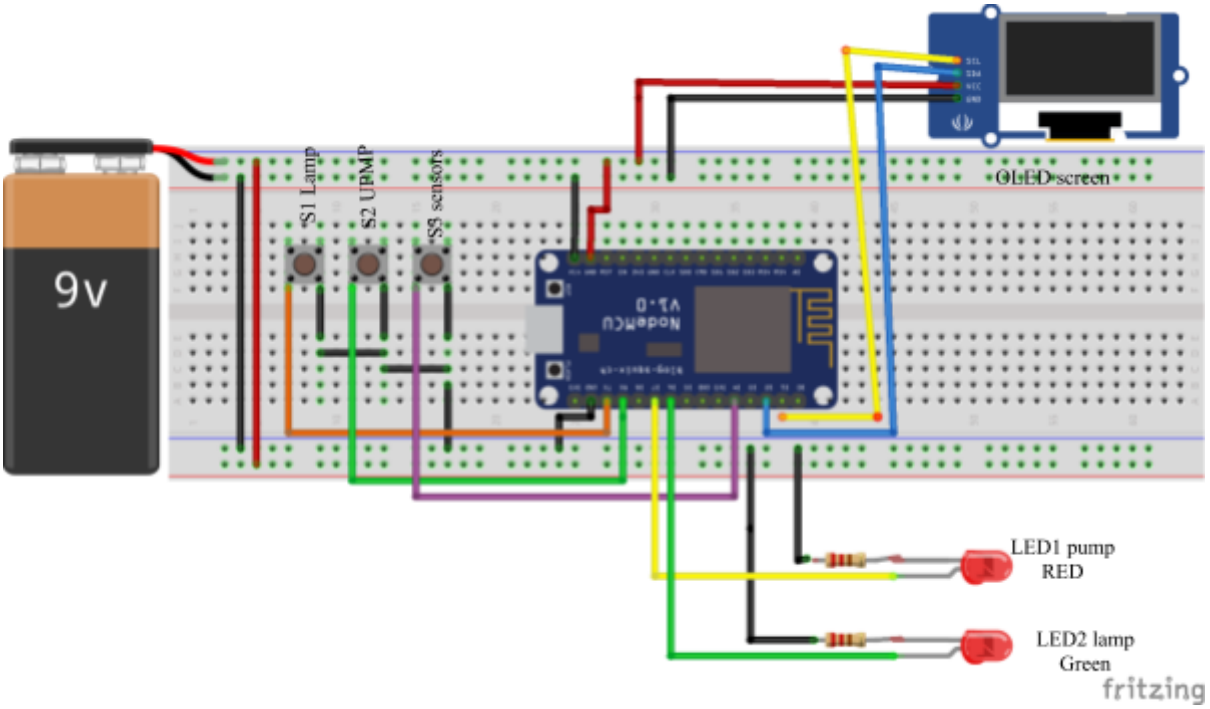

Figure 5.15 controlling and monitoring section

The block diagram of this circuit can be shown as below:-

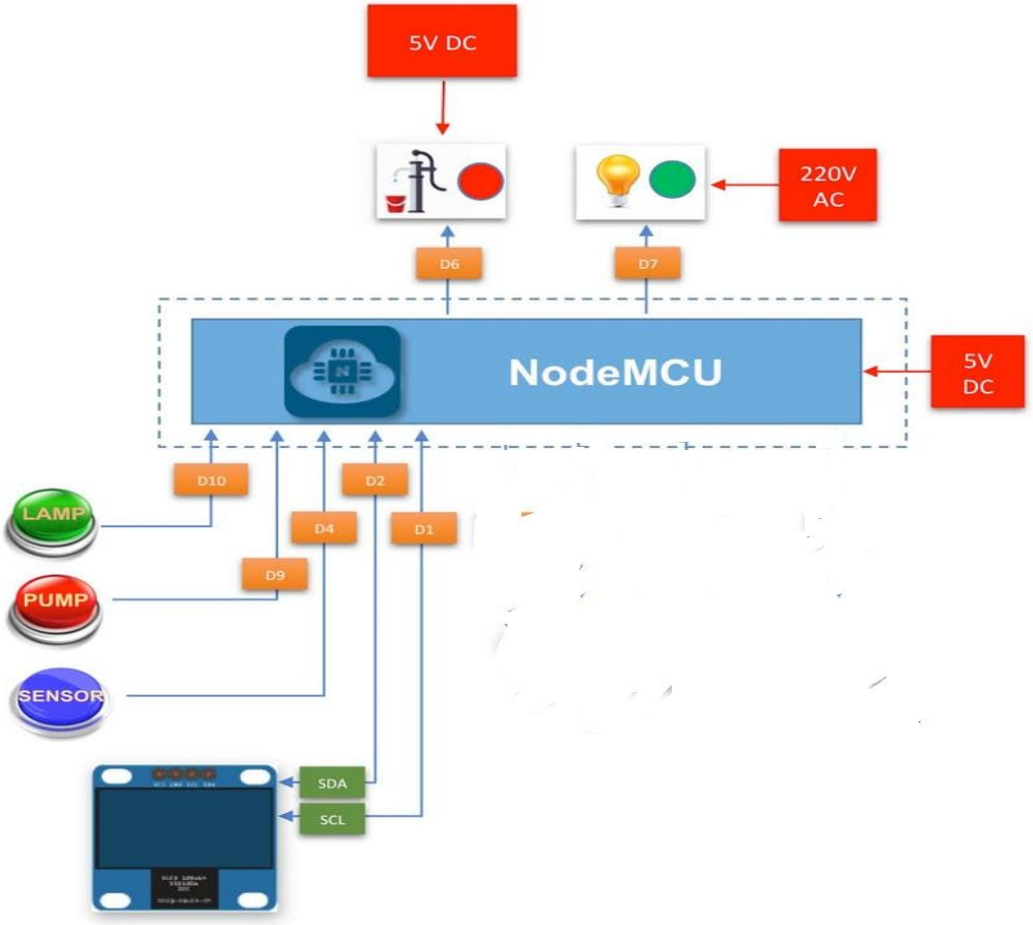

Figure 5.16 Block Diagram of controlling and monitoring section

The block diagram of this circuit can be shown as below:-

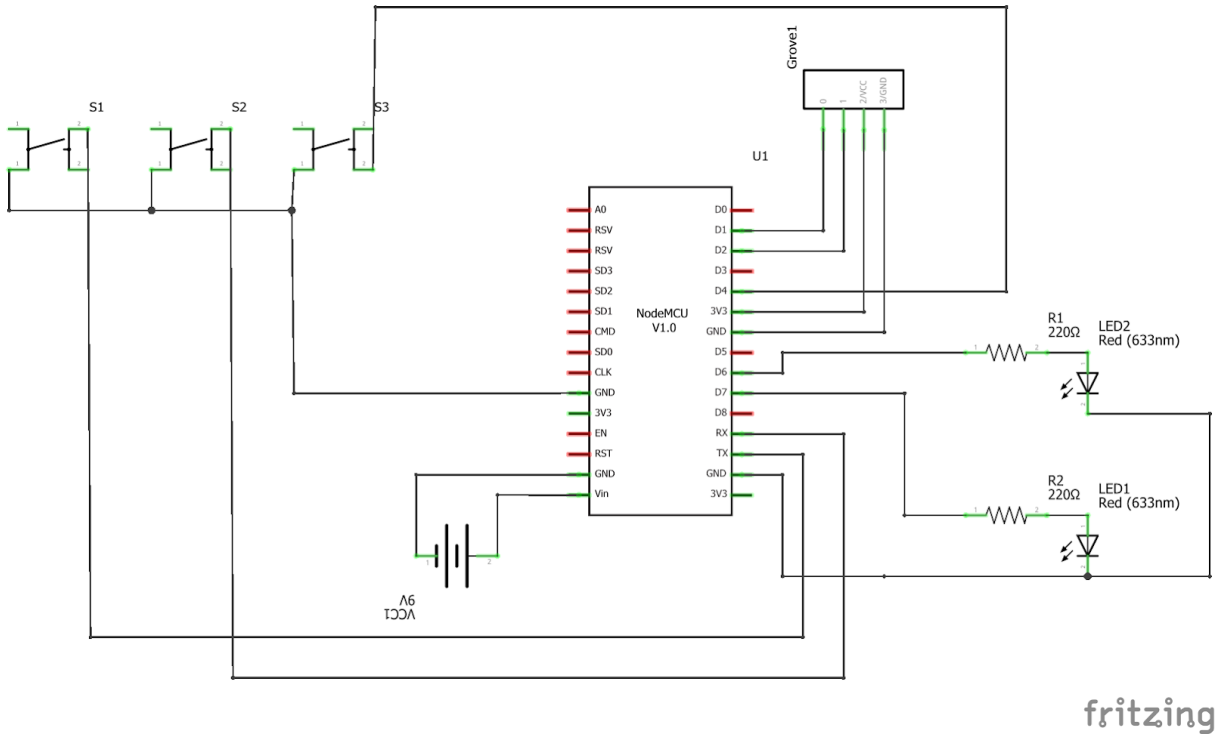

Figure 5.17 Circuit Diagram of controlling and monitoring section

❖ **Buttons**

Based on the readings of sensors, an operator could be also decide manually control the Pump and/or Lamp. For that, three push-buttons will be incorporate to the project:

1. RED: Pump Manual Ctrl
2. GREEN: Lamp manual Ctrl
3. YELLOW: Sensor Read Button (To update sensors, "light on" the OLED and present data (explained at next step)

The buttons will work on a "toggle" mode: If an actuator is "ON", pressing the button will "Turn-OFF" it and vice versa. The button's logic will be "normally closed", what means that NodeMCU Input will be constantly "HIGH". Pressing the button, a "LOW" will be applied at the specific pin (please see the above block diagram).

Chapter-6
RESULT & ANALYSIS

CHAPTER 6 RESULT & ANALYSIS

6.1 UPLOADING THE SOFTWARE IN NODEMCU

Step 1: Connect your NodeMCU to your computer

You need a USB micro B cable to connect the board. Once you plugged it in, a blue LED will start flashing. If your computer is not able to detect the NodeMCU board.

Step 2: Open Arduino IDE

You need to have at least Arduino IDE version 1.6.4 to proceed with this. Go to File > Preferences. In the "Additional Boards Manager URLs" field, type (or copy-paste) “http://arduino.esp8266.com/stable/package_esp8266com_index.json”. Don't forget to click OK!

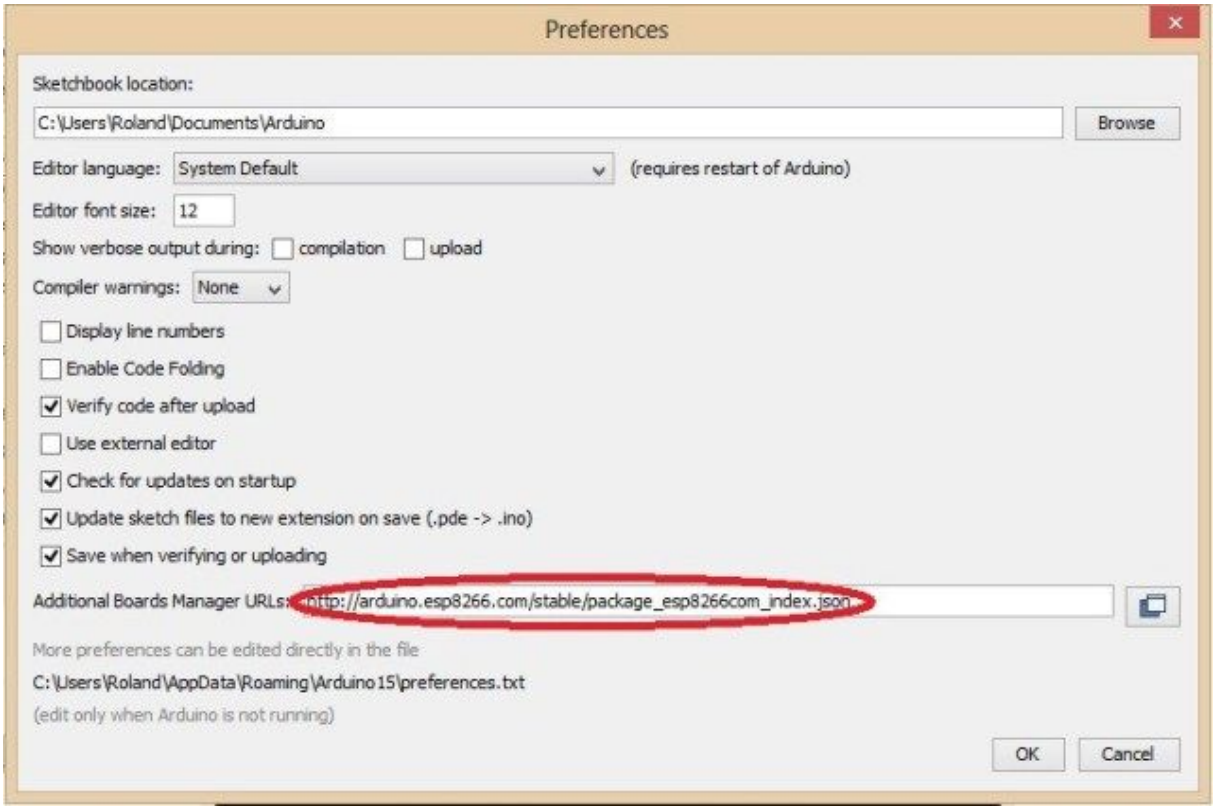

Figure 6.1 Arduino ide preference change

Then go to Tools > Board > Board Manager. Type "esp8266" in the search field. The entry "esp8266 by ESP8266 Community" should appear. Click that entry and look for the install button on the lower right.

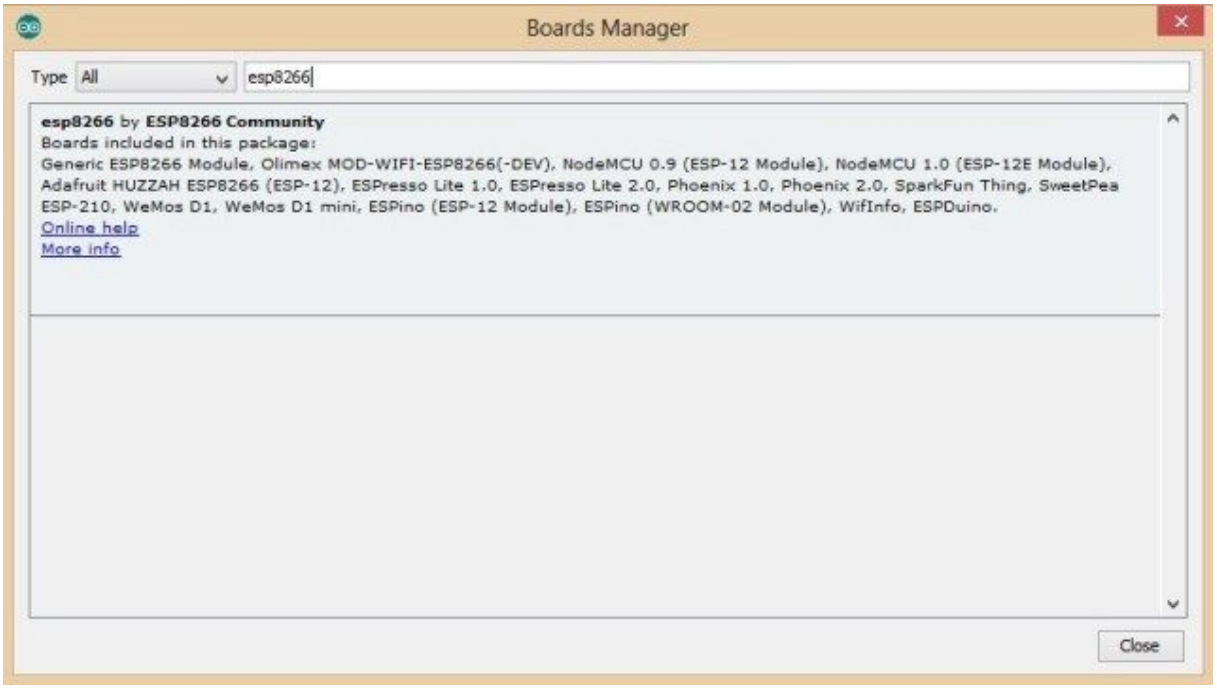

Figure 6.2 arduino ide set board manager

Once the download is complete, you can start coding!

Step 3: Downloading the Software File

1. To download the software file you should go to my github library first **"https://github.com/enggb/irrigation-system-server.git".**
2. Download the folder and find folder "esp8266_programming_using_arduino".
3. Got to files of folder for both nodemcu.

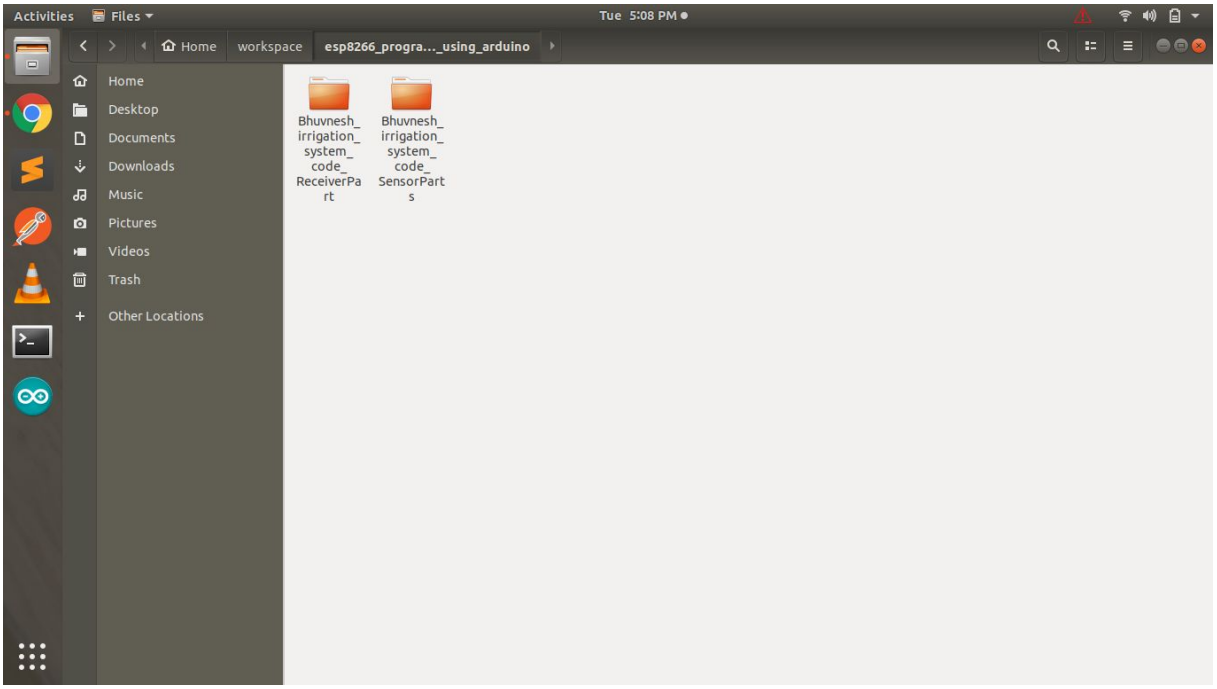

Figure 6.3 coding folders of nodemcu

Step 3: Uploading the Software File

1. Open the folders and uploading the files as shown in figure.
2. Upload bhuvnesh_irrigation_system_code_SensorPartprogramming in sensors circuit.

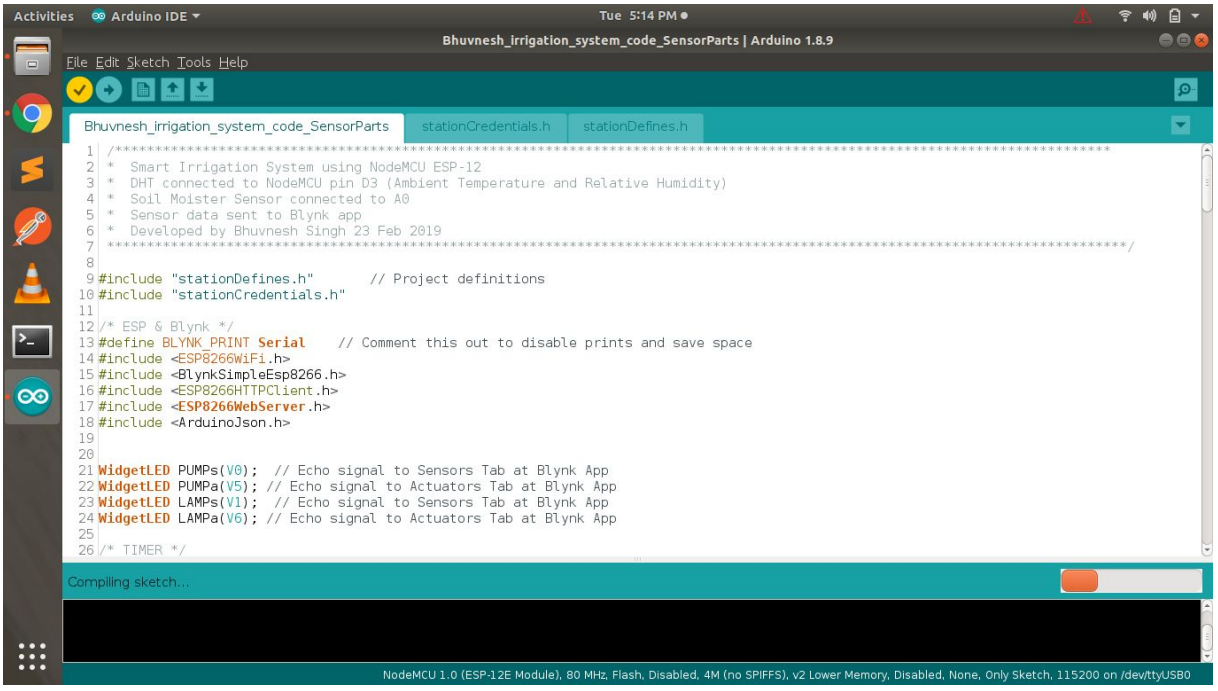

Figure 6.4 uploading sensors part programming

3. Upload bhuvnesh_irrigation_system_code_Receiverpart programming in controlling & monitoring circuit.

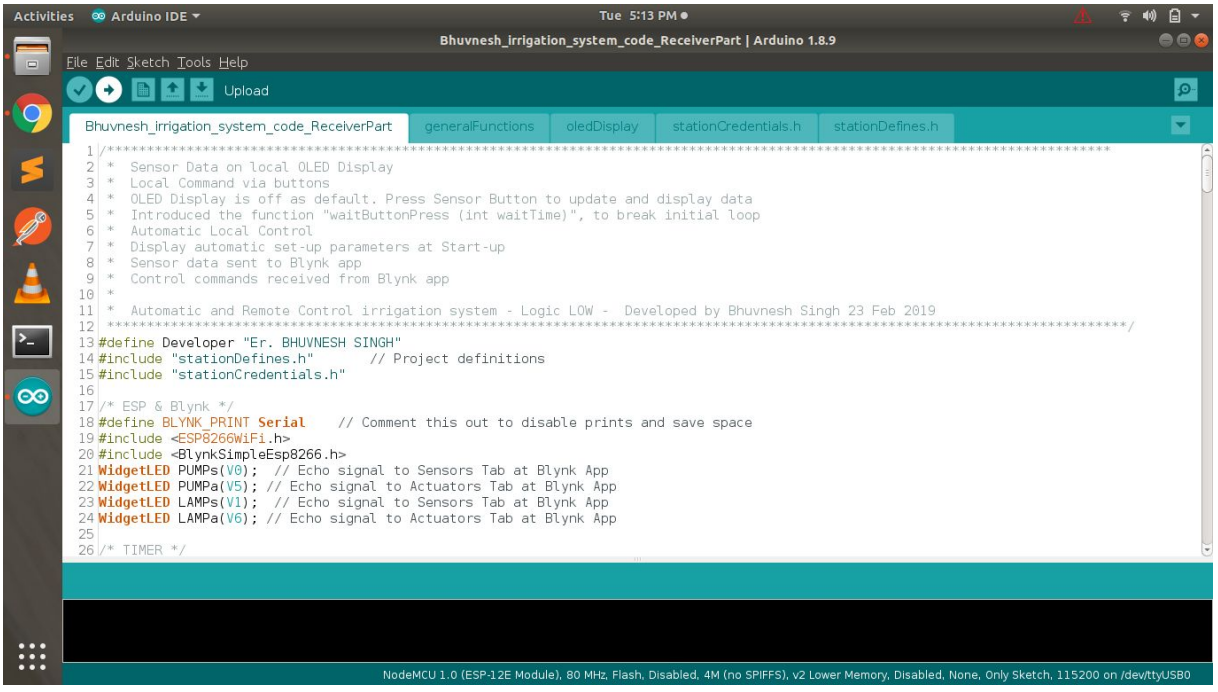

Figure 6.5 uploading monitoring part programming

The result of the our project shown in two different categories or shown in two different forms which are given as following:-

1. **Laptop or Web View**
2. **Mobile View Using Blynk App**

6.2 LAPTOP OF WEB VIEW

For the laptop view we will follow the following steps which are given below:-

Step 1. Start the server and database first

1. Type the "sudo mongod --dbpath ~/.ISS.db/" in the terminal to open the database.

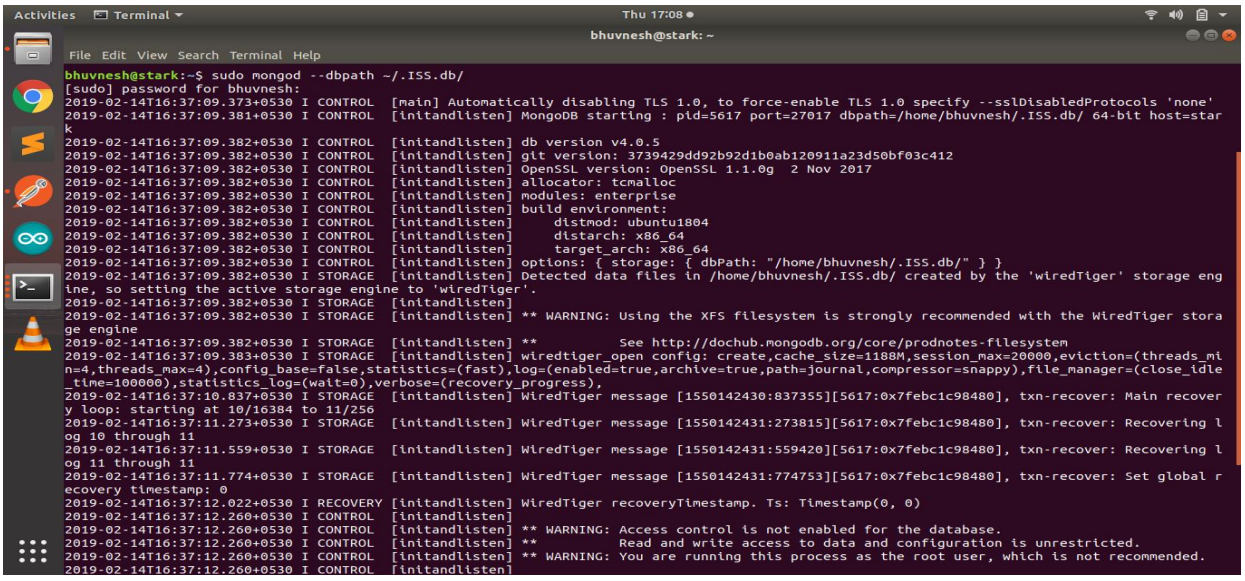

Figure 6.6 mongoDB open command in terminal

2. Type the "cd workspace/irrigation system server and then hit node index.js" in the terminal to open the server.

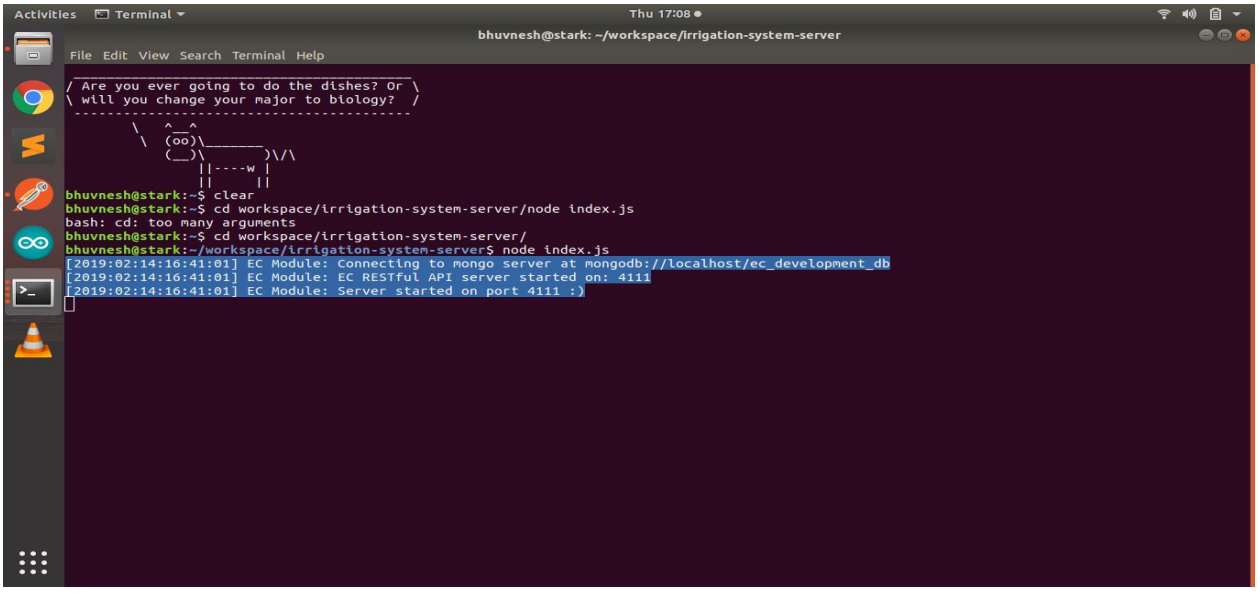

Figure 6.7 Nodejs server open command in terminal

3. Type the “cd workspace/prometheus-2.5.0.linux-amd64 ./prometheus” in the terminal to open the prometheus program.

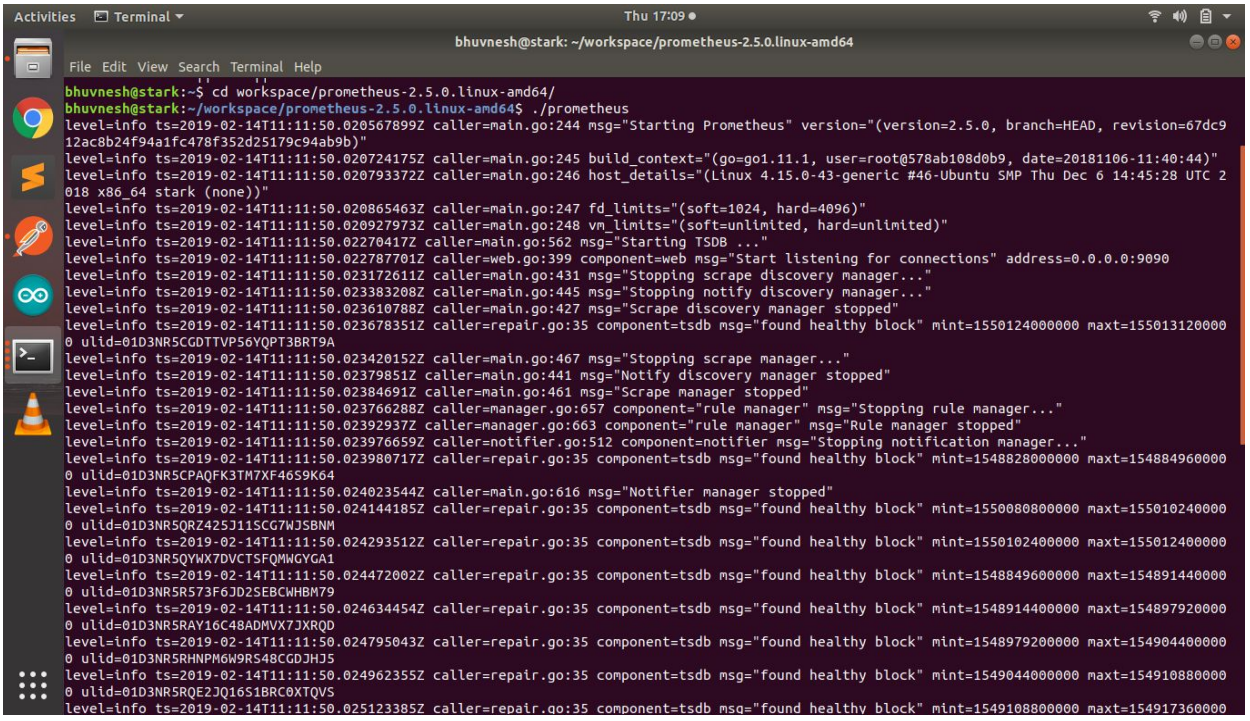

Figure 6.8 prometheus software open command in terminal

4. Type the “cd workspace/irrigation system server/ prometheus node connect.js” in the terminal to open the prometheus file.

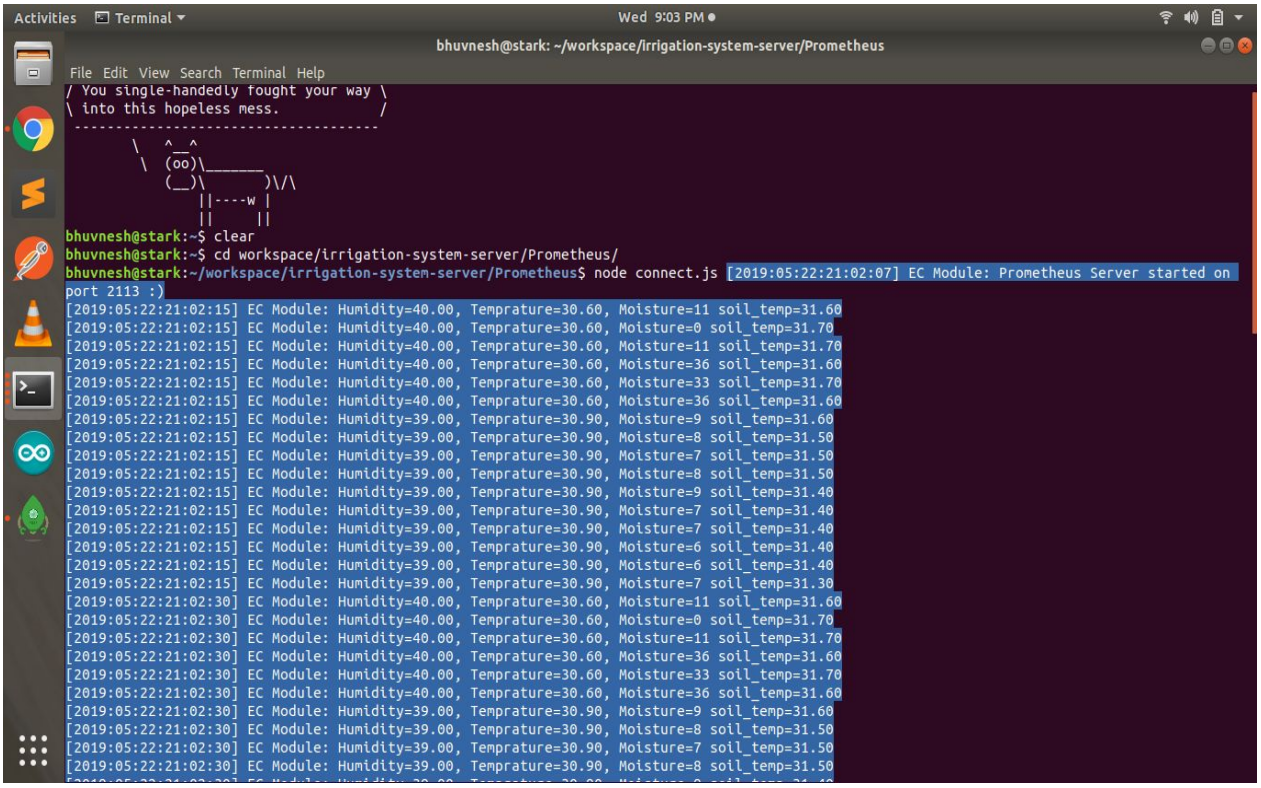

Figure 6.9 Nodejs file open command in terminal

5. Type the “cd workspace/robo3t-1.2.1-linux-x86_64-3e50a65/ bin and then ./robo3t” in the terminal to open the prometheus file.

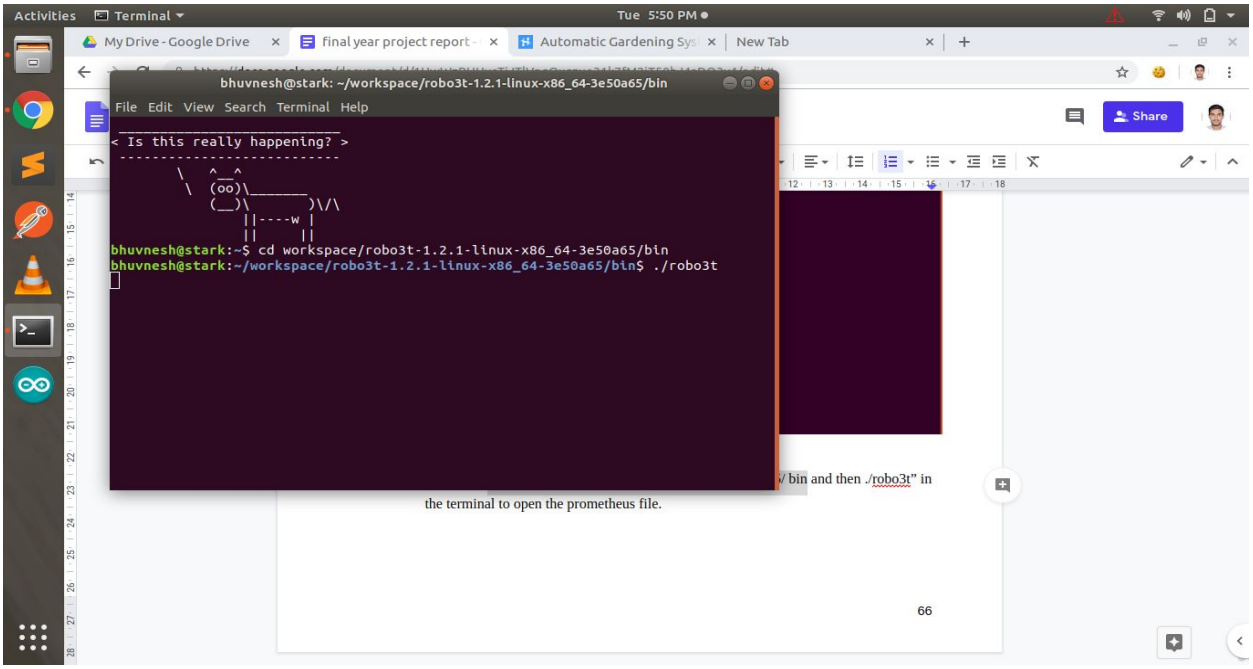

Figure 6.10 Robo3t software open command in terminal

Step 2. Connect the circuits project to power supply

1. Connect the sensor circuit to the 9v battery for turning on the NodeMcu-1 to collect the information from sensors and send to the database server.

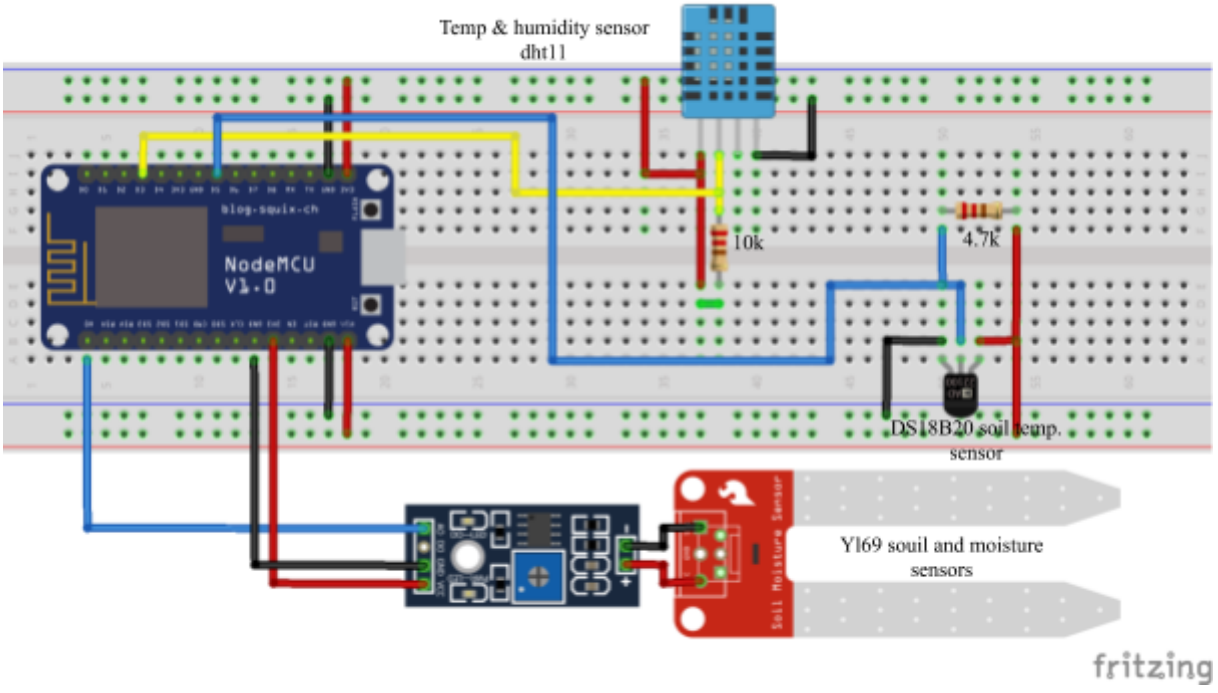

Figure 6.11 connected sensor part to 9v battery

2. Connect the controlling & monitoring circuit to the 9v battery for turning on the NodeMcu-2 to collect database server send to the nodemcu-2 for controlling the motor and lamp.

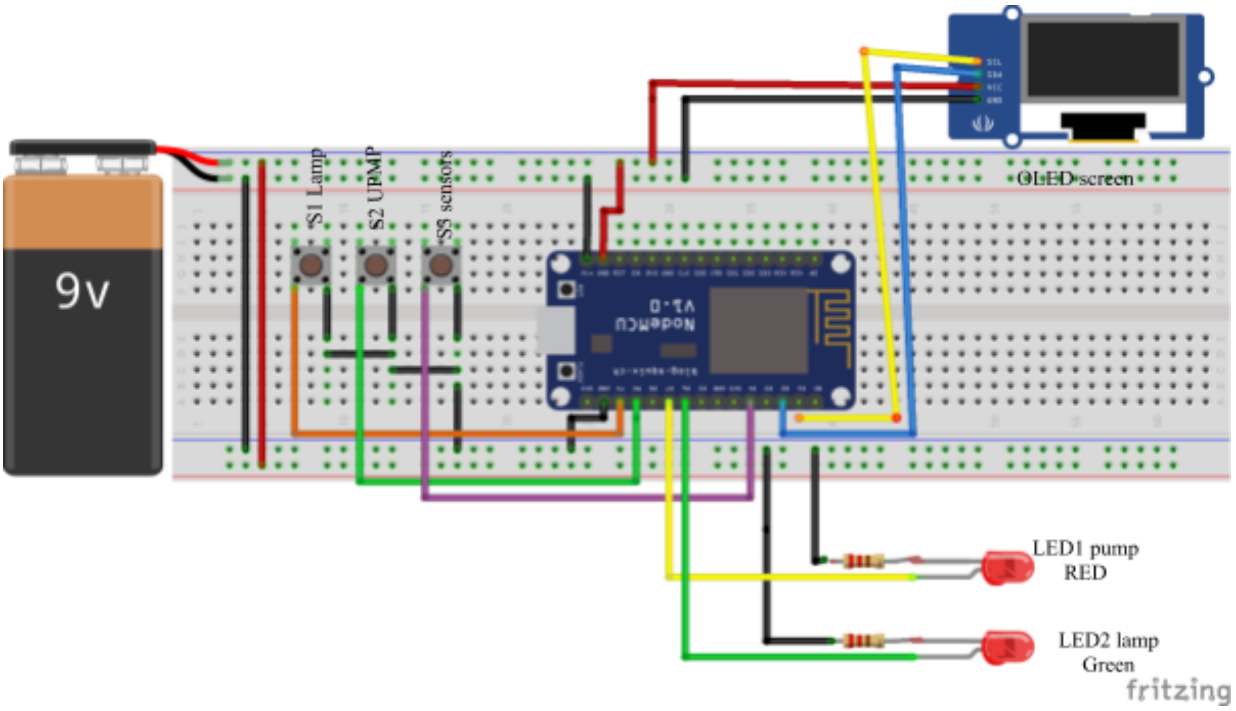

Figure 6.12 connected controlling & monitoring ckt to 9v battery

Step 3. Visualization of data

1. The sensors data will be visualize on the grafana dashboard that are generated by the grafana website by making account online.

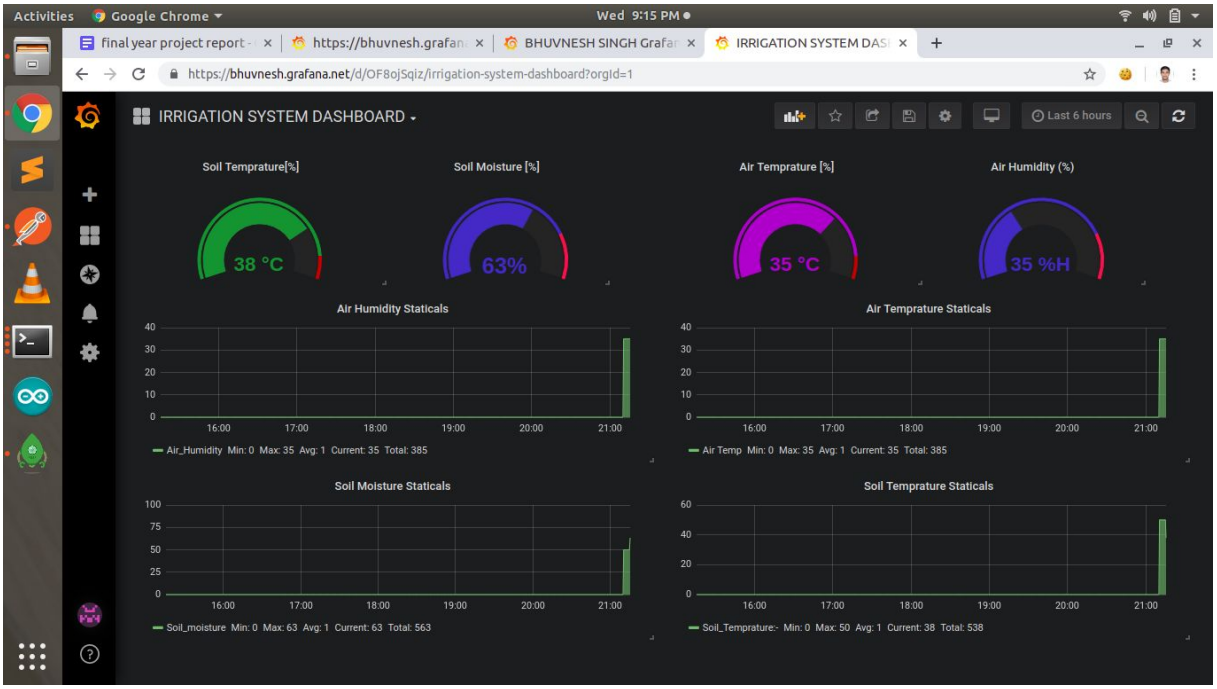

Figure 6.13 visualization of data on grafana dashboard

2. The Sensors Data can also be visualize on the mongo 3t software. It is directly connected with the mongodb database.

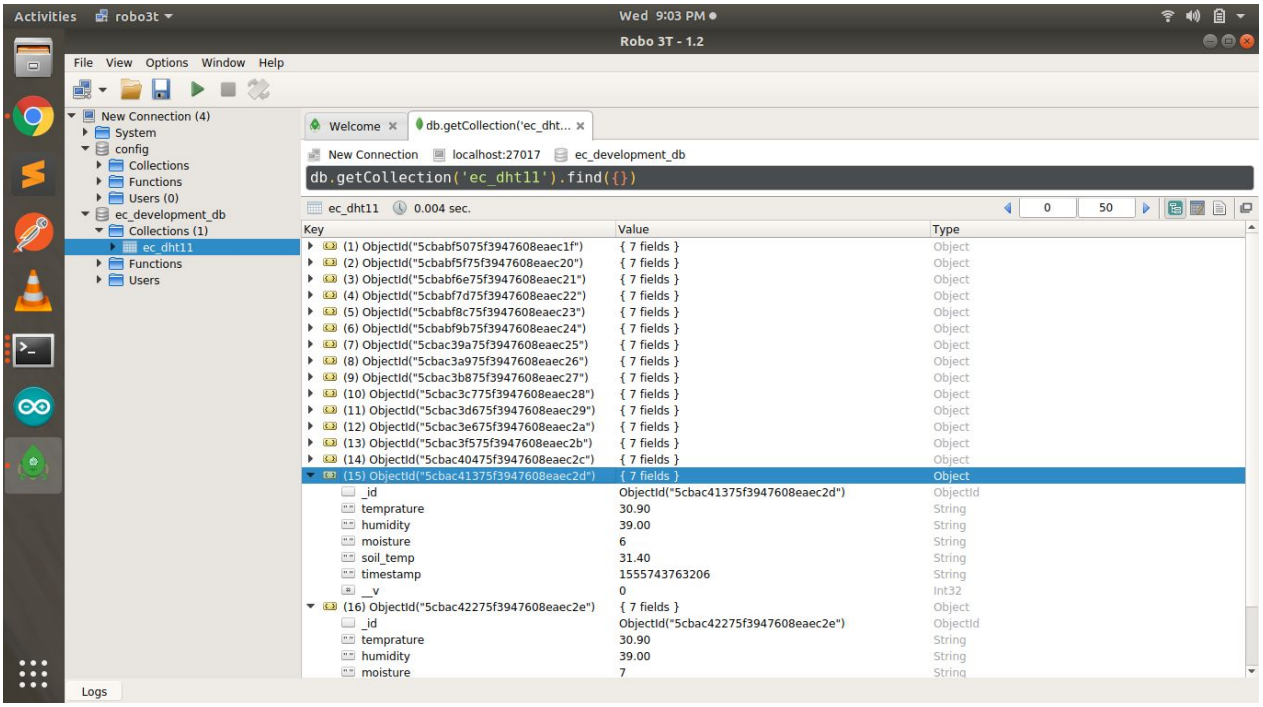

Figure 6.14 visualization of data on Mongo3t software

6.3 MOBILE VIEW USING BLYNK ANDROID APP

Step 1. Open the server and power to circuit

1. The step first will be same for the mobile view using blynk app and also step second to connect the circuit to the power supply also similar to the web view.

Step 2. Prepare the dashboard on blynk app

It is really very easy to build IoT projects using BLYNK. The first you need is to have the BLINK App installed on your phone its Library on the Arduino IDE. If you do not have them yet, please follow the below steps:

1. Download BLYNK app for Apple Iphone or Google Android.
2. Install BLYNK Library for Arduino. Note that you will download the zip file (There are 5 files there that you must manually install in your Arduino Library).
3. Once the Arduino IDE is reloaded, you should be OK to start using BLINK on your IoT project.

Now, let's open our app at the SmartPhone:

1. Open Blynk app.Tap on "Create New Project" screen.
2. Give a name for your project (For example "smart irrigation system").
3. Select the appropriated Hardware Model: "NodeMCU".

4. Take note from Authorization Token (you can email it to you to ease copy & paste on your code).
5. Press "OK". A Blank screen with dots will appear.
6. Tap the Screen to open the "Widget Box".

Revisiting the general specification at introduction, we can summarize that our app we be needed for:

1. Read all Sensors and verify the actuators status.
2. Take remote actions, "turning on/off" Pump and Lamp.
3. Sending messages when the System is "off-line" and/or an actuator is ON.
4. Record the general sensors data.

In order to organize things, let's split the above "tasks" in 3 tabs:

- **SENSORS**
- **ACTUATORS / CONTROL**
- **GRAPHICS**

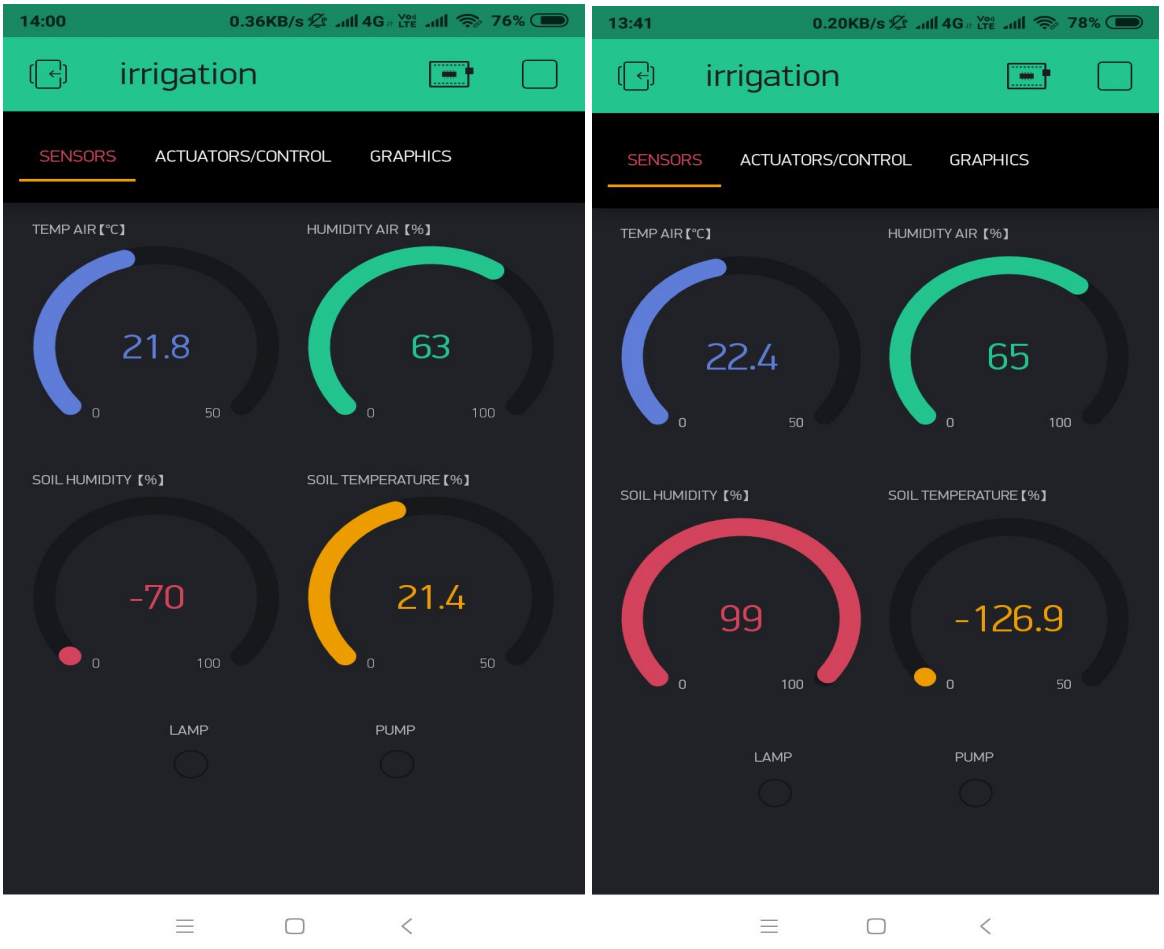

Figure 6.15 visualization of blynk app with sensors data

"Tabs" will be the first Widget to be installed. Enter on it and define the above "Tab names". Next, go to each Tab and install the Widgets as described below:

SENSORS

- Gauge: "Temp Air [oC]" Blue; input: V10 0 to 50; frequency: 5 sec
- Gauge: "Humidity Air [%]" Green; input: V11 0 to 100; frequency: 5 sec
- Gauge: "Soil Humidity [%]" Red; input: V12 0 to 100; frequency: 5 sec
- Gauge: "Soil Temperature[oC]" Yellow; input: V13 -10 to 50; frequency: 5 sec
- LED: "PUMP" Red; V0
- LED: "LAMP" Green; V1

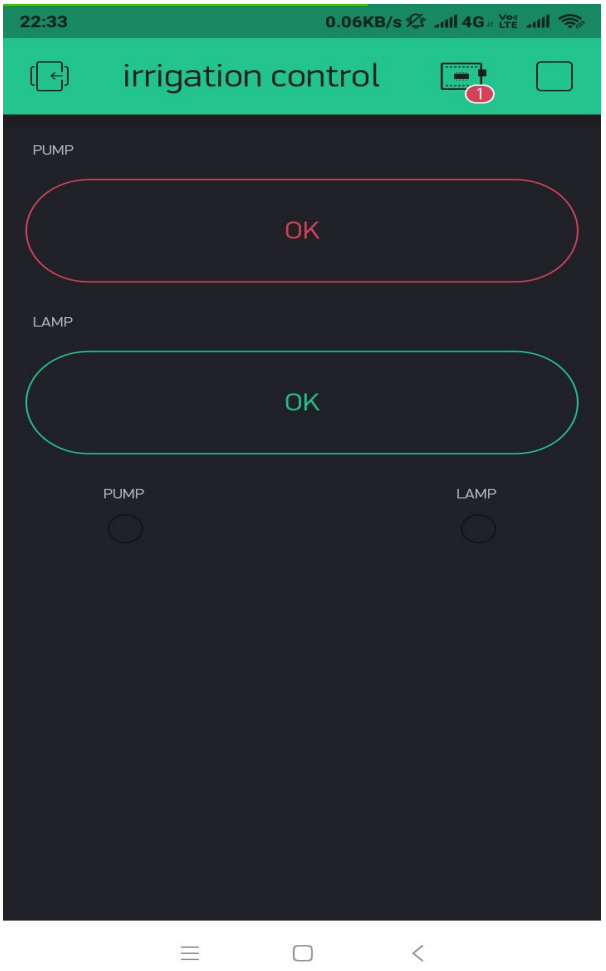

Figure 6.16 visualization of blynk app with controlling section

ACTUATORS / CONTROL

- Button: "PUMP" Red; output: V3 0 to 1; mode: Switch; label: on: ACT, off: OK
- Button: "LAMP" Green; output: V4 0 to 1; mode: Switch; label: on: ACT, off: OK
- LED: "PUMP" Red; V0
- LED: "LAMP" Green; V6
- Notifications: Notify When Hardware goes offline: ON

6.4 ACTUAL HARDWARE VIEW

The actual hardware of the project is shown as figure below:-

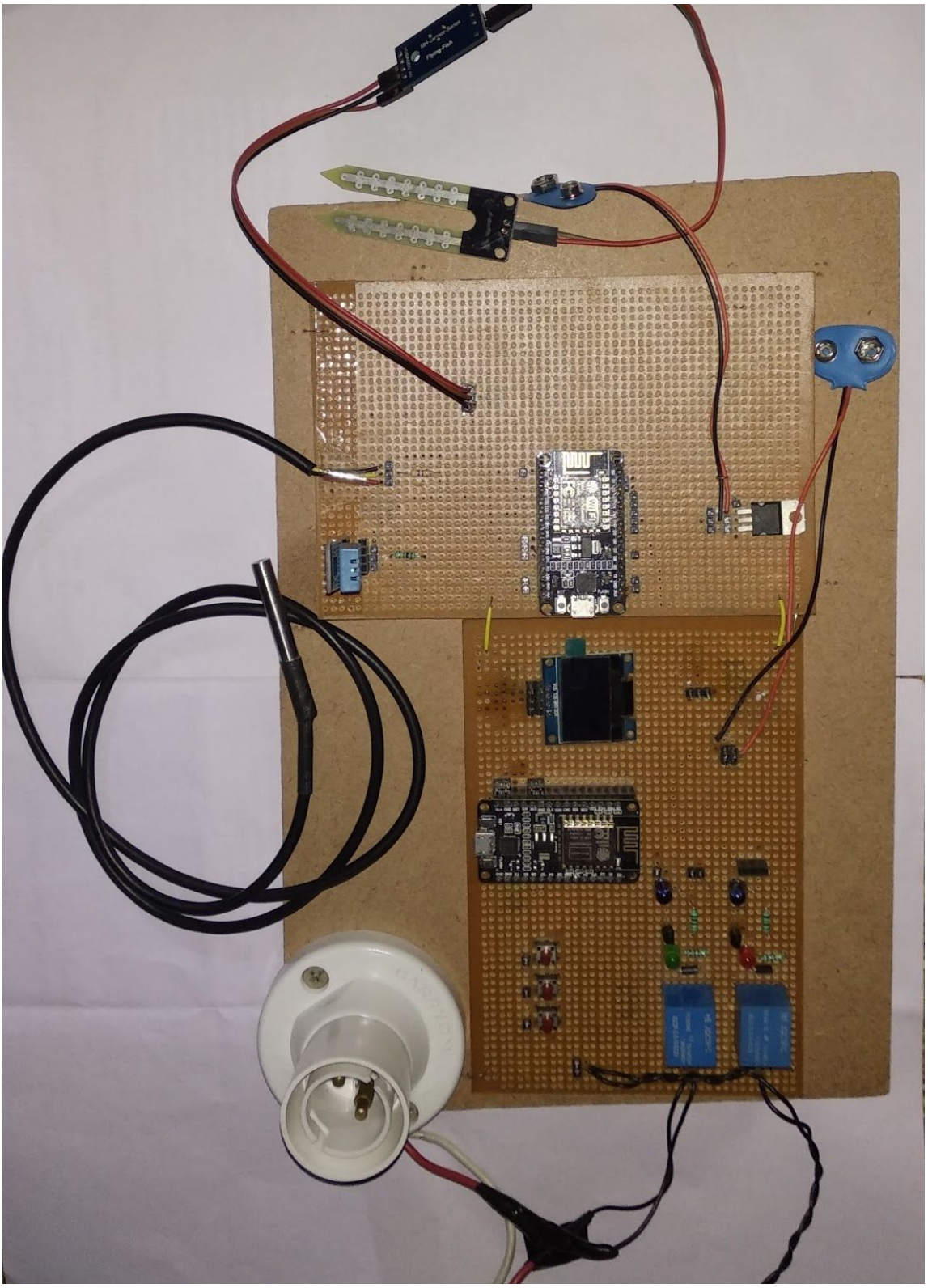

Figure 6.17 visualization of actual hardware project

APPENDIX I

CODE FOR ESP8266-1

```
/* Automatic Control Parameters Definition */
#define DRY_SOIL      35
#define WET_SOIL      70
#define COLD_TEMP     12
#define HOT_TEMP      22
#define TIME_PUMP_ON  15
#define TIME_LAMP_ON  15

/* TIMER */
#define READ_BUTTONS_TM   1L  // definitions in seconds
#define READ_SOIL_TEMP_TM 2L
#define READ_SOIL_HUM_TM  10L
#define READ_AIR_DATA_TM  2L
#define SEND_UP_DATA_TM   10L
#define AUTO_CTRL_TM      60L

/* DHT22*/
#define DHTPIN D3
#define DHTTYPE DHT11
/* Soil Moister */
#define soilMoisterPin A0
#define soilMoisterVcc D8
int soilMoister = 0;
/* DS18B20 Temperature Sensor */
#define ONE_WIRE_BUS 14 // DS18B20 on NodeMCU pin D5 corresponds to GPIO 014 on
Arduino
float soilTemp;
/**************************************************************************
**************************************************
 *  Smart Irrigation System using NodeMCU ESP-12
```

```
 *  DHT connected to NodeMCU pin D3 (Ambient Temperature and Relative Humidity)
 *  Soil Moisture Sensor connected to A0
 *  Sensor data sent to Blynk app
 *  Developed by Bhuvnesh Singh 23 Feb 2019
*************************************************************************
**************************************************/

#include "stationDefines.h"       // Project definitions
#include "stationCredentials.h"
/* ESP & Blynk */
#define BLYNK_PRINT Serial    // Comment this out to disable prints and save space
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>
#include <ArduinoJson.h>
WidgetLED PUMPs(V0);  // Echo signal to Sensors Tab at Blynk App
WidgetLED PUMPa(V5); // Echo signal to Actuators Tab at Blynk App
WidgetLED LAMPs(V1);  // Echo signal to Sensors Tab at Blynk App
WidgetLED LAMPa(V6); // Echo signal to Actuators Tab at Blynk App
/* TIMER */
#include <SimpleTimer.h>
SimpleTimer timer;
ESP8266WebServer server;
/* DHT22*/
#include <DHT.h>
DHT dht(DHTPIN, DHTTYPE);
float airHum = 0;
float airTemp = 0;
const char* host = "http://192.168.43.217";
const char port = 4111;
const int   watchdog = 60000; //  Frequency of sending data to Domoticz
unsigned long previousMillis = millis();
HTTPClient http;
```

```
/* DS18B20 Temperature Sensor */
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
/*****************************************************
 * setup started
 *****************************************************/
void setup()
{
  Serial.begin(115200);
  delay(1000);
  pinMode(DHTPIN, INPUT);
  pinMode(soilMoisterPin, INPUT);
  pinMode(ONE_WIRE_BUS, INPUT);
  Serial.println("Irrigation System By Er. Bhuvnesh Singh");
  Serial.println(".... Starting Setup");
  Serial.println(" ");
  Serial.println("Connecting Wifi...");

  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.print(WiFi.localIP());
  Blynk.begin(auth, ssid, pass);
  dht.begin();
}
void loop()
{
```

```
  timer.run(); // Initiates SimpleTimer
  Blynk.run();
 /*************************************************
  * Get DHT data
  *************************************************/
  float tempIni = airTemp;
  float humIni = airHum;
  airTemp = dht.readTemperature();
  airHum = dht.readHumidity();
  Serial.print("air Temperature(in Percentage) = ");
  Serial.print(airTemp);
  Serial.println("%");
  delay(1000);
  Serial.print("air humidity(in Percentage) = ");
  Serial.print(airHum);
  Serial.println("%");
  delay(1000);
  if (isnan(airHum) || isnan(airTemp))   // Check if any reads failed and exit early (to try
again).
  {
    Serial.println("Failed to read from DHT sensor!");
    airTemp = tempIni;
    airHum = humIni;
    return;
  }
 /*************************************************
  * Get Soil Moister Sensor data
  *************************************************/
 soilMoister = 0;
  digitalWrite (soilMoisterVcc, HIGH);
  delay (500);
  soilMoister = ( 100.00 - ( (analogRead(soilMoisterPin)/1023.00) * 100.00 ) );
  Serial.print("Soil Moisture(in Percentage) = ");
  Serial.print(soilMoister);
```

```
 Serial.println("%");
 delay(1000);
/*************************************************
 * Get SoilTemp sensor data
 *************************************************/
 DS18B20.requestTemperatures();
 soilTemp = DS18B20.getTempCByIndex(0);
 int newData = ((soilTemp + 0.05) * 10); //fix soilTemp value to 1 decimal place.
 soilTemp = (newData / 10.0);
 Serial.print("Soil Temperature(in Percentage) = ");
 Serial.print(soilTemp);
 Serial.println("%");
 Serial.print("connecting to ");
 Serial.println(host);
 Serial.print("Requesting URL: ");
 String url = "http://192.168.43.217:4111/api/temp?";
       url += "temp=";
       url += String(airTemp);
       url += "&hum=";
       url += String(airHum);
       url += "&moist=";
       url += String(soilMoister);
       url += "&soil_temp=";
       url += String(soilTemp);
 Serial.println(url);
 delay(500);
 if((WiFi.status() == WL_CONNECTED)) {
   delay(100);
 http.begin(url);
 int httpCode = http.GET();
   if (httpCode > 0) {
       String payload = http.getString();
       Serial.println(httpCode);
       Serial.println(payload);
```

```
    }
    else {
      Serial.print("Error on HTTP Request..... ");
    }
  Serial.println("closing connection");
  http.end();
  }

  delay(1000);

  Blynk.virtualWrite(V12, soilMoister); // virtual pin V12
  Blynk.virtualWrite(V10, airTemp); // virtual pin V10
  Blynk.virtualWrite(V11, airHum); // virtual pin V11
  Blynk.virtualWrite(V13, soilTemp); //virtual pin V13
}
```

APPENDIX II

CODE FOR ESP8266-2

```
/**************************************************************************
**************************************************
 *  Sensor Data on local OLED Display
 *  Local Command via buttons
 *  OLED Display is off as default. Press Sensor Button to update and display data
 *  Introduced the function "waitButtonPress (int waitTime)", to break initial loop
 *  Automatic Local Control
 *  Display automatic set-up parameters at Start-up
 *  Sensor data sent to Blynk app
 *  Control commands received from Blynk app
 *  Automatic and Remote Control irrigation system - Logic LOW -  Developed by Bhuvnesh
Singh 23 Feb 2019
**************************************************************************
**************************************************/
#define Developer "Er. BHUVNESH SINGH"
#include "stationDefines.h"       // Project definitions
#include "stationCredentials.h"

/* ESP & Blynk */
#define BLYNK_PRINT Serial    // Comment this out to disable prints and save space
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
WidgetLED PUMPs(V0);  // Echo signal to Sensors Tab at Blynk App
WidgetLED PUMPa(V5); // Echo signal to Actuators Tab at Blynk App
WidgetLED LAMPs(V1);  // Echo signal to Sensors Tab at Blynk App
WidgetLED LAMPa(V6); // Echo signal to Actuators Tab at Blynk App

/* Automatic Control Parameters Definition */
#define DRY_SOIL      66
#define WET_SOIL      85
```

```
#define COLD_TEMP     12
#define HOT_TEMP      22
#define TIME_PUMP_ON  15
#define TIME_LAMP_ON  15

/* TIMER */
#define READ_BUTTONS_TM   1L  // definitions in seconds
#define READ_SOIL_TEMP_TM 2L
#define READ_SOIL_HUM_TM  10L
#define READ_AIR_DATA_TM  2L
#define SEND_UP_DATA_TM   10L
#define AUTO_CTRL_TM      60L

/* OLED */
boolean turnOffOLED = 1;
#define SHOW_SET_UP   30


/* Soil Moister */
#define soilMoisterPin A0
#define soilMoisterVcc D8
int soilMoister = 0;

/* DS18B20 Temperature Sensor */
#define ONE_WIRE_BUS 14 // DS18B20 on NodeMCU pin D5 corresponds to GPIO 014 on
Arduino
int soilTemp;

/* Relays */
#define PUMP_PIN D6             //PUMP (Red LED)
#define LAMP_PIN D7             //LAMP (Green LED)
boolean pumpStatus = 0;
boolean lampStatus = 0;
```

```
/* Buttons */
#define PUMP_ON_BUTTON D9        //push-button PUMP (Red)
#define LAMP_ON_BUTTON D10       //push-button LAMP (Green)
#define SENSORS_READ_BUTTON D4   //push-button SENSOR (yellow)

/* TIMER */
#include <SimpleTimer.h>
SimpleTimer timer;

/* OLED */
#include <ACROBOTIC_SSD1306.h> // library for OLED: SCL ==> D1; SDA ==> D2
#include <SPI.h>
#include <Wire.h>

int airHum = 0;
int airTemp = 0;

/* DS18B20 Temperature Sensor */
#include <OneWire.h>
#include <DallasTemperature.h>
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);

  int t;
  int h;
  int temp;
  int hum;

 BLYNK_WRITE(V10) // V10 is the number of Virtual Pin
 {
 int t = param.asInt();
 }

 BLYNK_WRITE(V11) // V11 is the number of Virtual Pin
```

```
  {
  int h = param.asInt();
  }

  BLYNK_WRITE(V13) // V13 is the number of Virtual Pin
  {
  int temp = param.asInt();
  }

  BLYNK_WRITE(V12) // V12 is the number of Virtual Pin
  {
  int hum = param.asInt();
  }

void getDhtData(void)
{
  int airTemp = t;
  int airHum = h;
}

void getSoilMoisterData()
{
  int soilMoister = hum;
}

void getSoilTempData()
{
  int soilTemp = temp;
}

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

```
  Serial.println("Irrigation System By Er. Bhuvnesh Singh");
  Serial.println(".... Starting Setup");
  Serial.println(" ");

  pinMode(PUMP_PIN, OUTPUT);
  pinMode(LAMP_PIN, OUTPUT);
  pinMode(PUMP_ON_BUTTON, INPUT_PULLUP);
  pinMode(LAMP_ON_BUTTON, INPUT_PULLUP);
  pinMode(SENSORS_READ_BUTTON, INPUT_PULLUP);

  Blynk.begin(auth, ssid, pass);
  oledStart();

  PUMPs.off();
  LAMPs.off();
  PUMPa.off();
  LAMPa.off();
  digitalWrite(PUMP_PIN, HIGH); // To be used with Relay module (inverted logic: normally
HIGH)
  digitalWrite(LAMP_PIN, HIGH); // To be used with Relay module (inverted logic: normally
HIGH)

  timer.setInterval(1000L, getDhtData);
  timer.setInterval(1000L, getSoilMoisterData);
  timer.setInterval(1000L, getSoilTempData);

  waitButtonPress (SHOW_SET_UP); // Wait for Sensor Button to be pressed
  oled.clearDisplay();
  startTimers();
  displayData();
}

void loop()
{
```

```
  timer.run(); // Initiates SimpleTimer
  Blynk.run();
}
/**********************************************************************
* Read remote commands
**********************************************************************/
BLYNK_WRITE(3) // Pump remote control
{
  int i=param.asInt();
  if (i==1)
  {
    pumpStatus = !pumpStatus;
    aplyCmd();
  }
}
BLYNK_WRITE(4) // Lamp remote control
{
  int i=param.asInt();
  if (i==1)
  {
    lampStatus = !lampStatus;
    aplyCmd();
  }
}
/**********************************************************************
* Read local commands (Pump and Lamp buttons are normally "HIGH"):
**********************************************************************/
void readLocalCmd()
{
  boolean digiValue = debounce(PUMP_ON_BUTTON);
  if (!digiValue)
  {
    pumpStatus = !pumpStatus;
    aplyCmd();
```

```
  }
  digiValue = debounce(LAMP_ON_BUTTON);
  if (!digiValue)
  {
    lampStatus = !lampStatus;
    aplyCmd();
  }
  digiValue = debounce(SENSORS_READ_BUTTON);
  if (!digiValue)
  {
    turnOffOLED = !turnOffOLED;
    if (!turnOffOLED)
    {
      oled.setTextXY(0,0); oled.putString("UPDATING SENSORS");
      getDhtData();
      getSoilMoisterData();
      getSoilTempData();
      oledStart();
      displayData();
    }else oled.clearDisplay();
  }
}
/*****************************************************
* Receive Commands and act on RELAYs - Logic LOW
*****************************************************/
void aplyCmd()
{
  if (pumpStatus == 1)
  {
    Blynk.notify("irrigation: Warning ==>> Pump ON");
    digitalWrite(PUMP_PIN, LOW); // To be used with Relay module (inverted logic: activate
with LOW)
    if (!turnOffOLED) displayData();
    PUMPs.on();
```

```
   PUMPa.on();
  }
  else
     {
       digitalWrite(PUMP_PIN, HIGH); // To be used with Relay module (inverted logic:
normally HIGH)
       if (!turnOffOLED) displayData();
       PUMPs.off();
       PUMPa.off();
     }
  if (lampStatus == 1)
  {
    Blynk.notify("irrigation: Warning ==>> Lamp ON");
    digitalWrite(LAMP_PIN, LOW); // To be used with Relay module (inverted logic: activate
with LOW)
    if (!turnOffOLED) displayData();
    LAMPs.on();
    LAMPa.on();
  }
  else
     {
       digitalWrite(LAMP_PIN, HIGH); // To be used with Relay module (inverted logic:
normally HIGH)
       if (!turnOffOLED) displayData();
       LAMPs.off();
       LAMPa.off();
     }
}
/*************************************************
* Automatically Control the Plantation based on sensors reading
*************************************************/
void autoControlPlantation(void)
{
  if (soilMoister < DRY_SOIL)
```

```
  {
    turnPumpOn();
  }
  if (airTemp < COLD_TEMP)
  {
    turnLampOn();
  }
}
/*************************************************
* Turn Pump On for a certain amount of time
*************************************************/
void turnPumpOn()
{
  pumpStatus = 1;
  aplyCmd();
  delay (TIME_PUMP_ON*1000);
  pumpStatus = 0;
  aplyCmd();
}
/*************************************************
* Turn Lamp On for a certain amount of time
*************************************************/
void turnLampOn()
{
  lampStatus = 1;
  aplyCmd();
  delay (TIME_LAMP_ON*1000);
  lampStatus = 0;
  aplyCmd();
}
```

REFERENCE

1. Dr. Narayan G. Hegde, “Water Scarcity and Security in India”, BAIF Development Research Foundation, Pune.
2. Marvin T. Batte, “Changing computer use in agriculture: evidence from Ohio”, Computers and Electronics in Agriculture, Elsevier science publishers, vol. 47, 1–13,2005.
3. Karan Kansara, Vishal Zaveri, Shreyans Shah, Sandip Delwadkar, and Kaushal Jani “Sensor based Automated Irrigation System with IOT: A Technical Review”,(IJCSIT) International Journal of Computer Science and Information Technologies.
4. REVIEW PAPER BASED ON AUTOMATIC IRRIGATION SYSTEM BASED ON RF MODULE, by Ms. Deweshvree Rane PG Scholar - VLSI, Sevagram, Wardha, India. Published by IJAICT Volume 1, Issue 9, January 2015 .
5. Pavithra D.S, M.S.Srinath GSM based Automatic Irrigation Control System for Efficient Use of Resources and Crop Planning by Using an Android Mobile.
6. SENSOR BASED AUTOMATED IRRIGATION SYSTEM WITH IoT: A TECHNICAL REVIEW by Karan Kanasura, Vishal Zaveri, Babu Madhav Institute of Technology, Uka Tasadia University, Bardoli, Gujarat, India : ISSN:0975-9646.
7. Gayathri Natarajan and Dr. L. Ashok Kumar, Implementation of IoT Based Smart Village for the Rural Development, International Journal of Mechanical Engineering and Technology 8(8), 2017, pp. 1212–1222.
8. Marvin T. Batte, “Changing computer use in agriculture: evidence from Ohio”, Computers and Electronics in Agriculture, Elsevier science publishers, vol. 47, 1–13, 2005.
9. Sumeet. S. Bedekar, Manoj. A. Mechkul, and Sonali. R. Deshpande “IoT based Automated Irrigation System”, IJSRD - International Journal for Scientific Research & Development| Vol. 3, Issue 04, 2015 | ISSN (online): 2321-0613.