UNIVERSITY OF GHANA LEGON

ACCRA GHANA

DEPARTMENT OF COMPUTER ENGINEERING

SCHOOL OF ENGINEERING SCIENCES

FINAL YEAR PROJECT REPORT

ON

SMART DRIP IRRIGATION SYSTEM FOR TOMATO FARMING USING WIRELESS SENSOR NETWORK

PROJECT REPORT SUBMITTED IN PARTIAL FULLFILMENT OF THE REQUIREMENTS FOR THE BACHELOR OF SCIENCE DEGREE IN COMPUTER ENGINEERING

GYATENG EMMANUEL OKYERE – 10660873

SHERIF SULEMAN - 10681755

SUPERVISOR: MRS. GIFTY OSEI

24TH SEPTEMBER, 2021

SMART DRIP IRRIGATION SYSTEM FOR TOMATO FARMING USING WIRELESS SENSOR NETWORK

By

Gyateng Emmanuel Okyere (10660873)

And

Sherif Suleman (10681755)

Submitted to the Department of Computer Engineering in

Partial Fulfilment of the Requirements for the Degree of

Bachelor of Science in Computer Engineering

University of Ghana

24th September, 2021

Name of Student: Gyateng Emmanuel Okyere

Signature of Student: ____

Name of Student: Sherif Suleman

Signature of Student: ____

Name of Supervisor: Mrs. Gifty Osei

Signature of Supervisor: ____

Name of Head of Department: Dr. Godfrey A. Mills

Signature of Head of Department: _______________________________________

Declaration of Originality

Department of Computer Engineering

I certify that this thesis is my own work except where indicated by referencing. I also certify that I have read and understood the guidelines on plagiarism in the Computer Engineering undergraduate thesis handbook including the University of Ghana's policy on plagiarism. I have acknowledged in the text of this thesis all sources used. I have also fully referenced including page numbers all the relevant texts, figures, data, and tables quoted from books, journals, articles, Internet websites, and works of other people. I have not used the services of any professional person or agency to produce this document. I have also not presented the work of any student present or past from other academic or research Institutions. I understand that any false claim in respect of this thesis document will result in disciplinary action in accordance with the regulations of the University of Ghana.

Please complete the information below by Hand and in BLOCK LETTERS.

Student Name: ....

Index Number: ....

Student Signature: .... Date.......................................

Student Name: ....

Index Number: ....

Student Signature: .... Date.......................................

Name of Supervisor(s): ....

Supervisor's Signature: .... Date.......................................

Abstract

Smart Drip Irrigation System for Tomato Farming Using Wireless Sensor Network

Irrigation systems are becoming more important owing to the increase in human population, global warming, and food demand. This project designs and implement a drip irrigation system for tomato farming and reduce water wastage during irrigation and increase tomato yields in all seasons. An internet of things- based smart drip irrigation system using a soil moisture sensor and actuators, is designed to make water drip directly onto the roots of tomato plants when moisture level goes beyond the set threshold. Engineering system development lifecycle and waterfall model design methodologies have been deployed in the development paradigm. Using Arduino integration development environment Proteus design suite, a working prototype of this project is designed. A Wi-Fi module was used a node to handle computations and process sensor data. It also allows remote control and view of the system using an android app. With the program embedded on the Node MCU, DC pump can be controlled as ON- State and OFF- State depending on the soil moisture level when dispensing water unto the roots of tomato plants. This Smart drip irrigation system for deployment on tomato farms, will help prevent under irrigation and over irrigation also, help in the reduction of labor and labor cost.

Acknowledgement

We would like to first thank the Almighty God. Our second gratitude goes to our supervisor Mrs. Gifty Osei for the time she sacrificed to lead to the success of this project. We thank our family members for their support financially.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1- INTRODUCTION

1.0. Background

Farming serves as the livelihood for most of the Ghanaian population [1]. One of the products from these farmers is tomatoes. Tomato is one of the farm products that is highly demanded by most Ghanaians [2]. Farmers are not able to produce enough tomatoes to meet the demand [3]. Generally, tomato farmers in Ghana tend to use old farming practices which usually result in under irrigation or over irrigation. Most tomato farms in the country also do not have large source of water for irrigating their plants. The farmers usually depend on small wells in their farms and also rain water for the irrigation of their farms. In dry seasons, farmers do not have enough water to irrigate their tomato plants which leads to low production of tomatoes. Most tomato farmers in Ghana plant in only one season [4]. Tomato traders mostly import fresh tomatoes from neighboring countries, particularly Burkina Faso [5]. This high importation can be attributed to farmers not being able to provide enough tomatoes to meet the demand.

Considering all these problems tomato farmers go through to irrigate their farms, we designed and implemented a smart IOT based drip irrigation system using wireless sensor network architecture. Most smart irrigation systems uses the sprinkle irrigation system, this system of irrigation uses a lot of water and also supplies water to parts that do not require water for the development of the plant. In our case where water source is less, we have decided to use a system that produces the water directly to the root system of the plant which is where water is needed for the development of the plant in a way to use the available water efficiently. This system is the drip irrigation system and it also increases yields. This thesis describes the processes we went through, the tool we used and the knowledge based on which we designed the system.

1.1. Problem Definition

According to an article titled “The case of Tomato in Ghana” published by International Food policy research institute on April 23, 2010, over the years, the production of tomatoes in Ghana appears to be falling gradually [6].

One of the main reasons for this fall is because most of the farmers still use traditional systems of irrigation for their plants.

These systems usually leads to either over irrigation or under irrigation and thus make it less efficient which reduces the quality and quantity of the crop yields.

The source of fresh water in most tomato farming areas in the country are usually rain and small well, this therefore raises the need for a system of irrigation which is not only efficient but also reduce waste of water during irrigation.

1.2. Motivation

In Ghana, the largest occupation is farming, most of these farmers regardless of their hard work still do not have enough money to live comfortably because of low income due to low yields. There is a need to help boost the income of these farmers by putting in place measures to help increase yields.

Also, almost all the foods Ghanaians enjoy require tomato to prepare, this makes people go the extra mile to import tomatoes from elsewhere to satisfy our high demand for the crop, and this also means a lot of money which could have gone to our local farmers will rather go elsewhere.

1.3. Project Objectives

The objectives of this project are:

1. Design and implement an Internet of Things based system that will be used to automate irrigation.
2. Design a drip irrigation system.
3. Design an android application that will be used to monitor the system and schedule for irrigation.

1.4. Relevance of Project

The project relevance is bulleted as below

- Water wastage is eradicated since water is dripped directly unto the root.
- Reduces labor and labor cost since the system is automated.
- Increases the yield of tomatoes.
- Tomato farming can be done in both seasons since less water is required.
- Prevent over-irrigation and under irrigation since the soil moisture level is constantly monitored before dispensing water into the soil.

1.5. Outline of Thesis

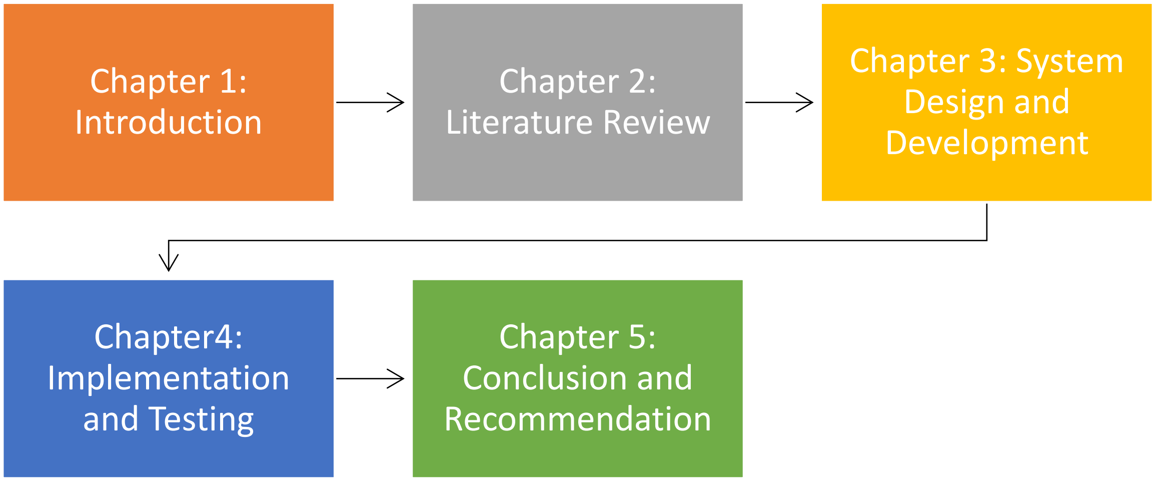

Chapter 1: Introduction
Chapter 2: Literature Review
Chapter 3: System Design and Development
Chapter4: Implementation and Testing
Chapter 5: Conclusion and Recommendation

FIGURE 1.1: OUTLINE OF THESIS

Chapter 1 of the thesis provides background information on smart drip irrigation. It justify the use of our smart irrigation system as a replacement to the current systems of irrigation which has been proven not efficient and reliable.

Chapter 2 focuses on the review of other works done previously to justify this project. We also compare and contrast reviewed works to know their strength and shortcomings.

Chapter 3 describes the design and development of the proposed system. It also outlines the requirement, specification and their analysis necessary for the design and development of the proposed system.

Chapter 4 has a detailed description of how the smart drip irrigation system was implemented and tested. It also shows how we were able to control and monitor the system using our mobile application.

Chapter 5 is the concluding part of the thesis, it highlights the major system development efforts and possible recommendation for further works to enhance the efficiency of the system.

CHAPTER 2- LITERATURE REVIEW

2.0. Introduction

The high demand for foods has called for the need to improve food production technology. This has forced many projects by researchers to contribute to the modernization of irrigation systems used in crop production. Many researchers and scholars have made significant efforts to design better systems for irrigation of farms. This chapter examines previous works on smart irrigation systems and also looks at the strengths and limitations of these works.

2.1. Review of Existing Works

1. Intelligent Internet of things- Based Automation Irrigation system

S. e. a. Sankaranarayanan et al [7], designed and implemented an IoT based automation irrigation system that uses both soil moisture sensor and temperature sensor. These sensors are connected to an Arduino microcontroller. The sensed values are transmitted serially to an edge level processor called the raspberry Pi3 which has a machine-learning algorithm called the KNN (k nearest neighbor) classification. The prediction of the soil condition is based on the moisture and temperature level. The algorithm classifies the sensor values based on the closest training examples. This is a type of instance based learning or lazy learning where the function is approximated locally. Computation is deferred until classification is done with. With the classification technique they used, little or prior knowledge about distribution of data is required. In order to make an intelligent analysis and prediction of the condition of the farms field, they considered dry, little dry, little wet and wet soil for training of the data set. The analyzed data along with the field irrigated are uploaded to a cloud server. The data in the cloud server can be accessed by the farmers mobile. In addition to the data, a graph data sheet of moisture versus temperature is also sent to the cloud server. The predicted condition is used for send control signal through serial communication to the Arduino to control a water pump for watering the field accordingly. The final aspect of their project is recording the soil moisture and temperature level and prediction with date and time in the cloud server for farmers to access from their mobile.

For their system, the method used in classifying the soil was intelligent but farmers were not given the flexibility to control the system. Also, for the farmer to understand the graphs generated by the system, he or she has to have prior knowledge about distribution of data.

2. Internet of things- based smart irrigation system

S. Rawal [8] , proposed and designed an IoT based smart irrigation system that employs the use of two soil moisture sensors along with a comparator that compares the values from the two soil moisture sensors. The two soil moisture sensors are connected to an Arduino (ATMEGA328P), which checks if the value from the sensor is below the specified threshold value, if it is below the threshold, the system will start the irrigation process and if not it will not start the process. The readings from the sensor were taken over a period of one hour to observe the rate at which moisture content in the soil is reducing when the sprinklers are off. Sensor values are transmitted through the GSM modem (GSM-GPRS SIM900A) which is a modem attached to the Arduino and gives the system its Internet of things capabilities. The data is sent to a cloud server and a web application displays the current water sprinkler status i.e. on or off and a button that redirects the farmer to the cloud server (Thingspeak.com) to view graphically, the values from the sensor.

The method of irrigation used by this system is sprinkling. This method of irrigation waste water across farms. Also, the system checking for moisture level every one hour makes it inefficient. Due to diverse weather conditions, farmers should have been given the privilege to control the system.

3. Internet of things- Based Smart Irrigation Monitoring & Controlling System in Agriculture

Md. M. Islam et al [9], considered to design and implement a system within which the farmer first has to register with his or her email. The system uses four sensors, a soil moisture sensor, water level sensor, temperature and humidity sensor and a pH sensor. The sensors are connected to a raspberry pi3 for processing. A high and low threshold value for irrigation was specified, if the soil moisture sensor value is below a low threshold value, the system automatically starts irrigation and if it is above the value, it does not, the system also stops irrigation if the sensor value rises above the high threshold value. The condition and time is then sent to the user via email. The water level sensor is used to check the amount of water pumped for irrigation in order to stop irrigation at the appropriate time. When irrigation is stopped, the time and condition will again be sent to the user. Both the pH and the temperature and humidity sensors are to update the user on the condition of the soil. These information together with the Time, is sent to the farmer via email. All the data from the sensors are stored on a cloud server. The farmer also registers an IFTT (If This Then That) app, which predicts rainfall on the next day so that they can turn the pumps off since there would be rain. The IFTT is connected to the user’s smartphone.

With this system, once it gets the weather condition for the next day and it predict that it will rain, the pump are turned off on that day the IFTT predicted that it will rain. Whether it rains or not, the pumps will be off. The farmer is not given the privilege to turn on or off the pumps.

4. Smart Drip Irrigation System using Internet of things

T. Gaikwad et al [10], implemented a system where soil type and form is considered in the system. Using two way power i.e. Solar and AC supply, the system always has power to perform its operations. The proposed system made up of a soil moisture sensor, temperature sensor, humidity sensor and a rain drop sensor. All these sensors are connected to a raspberry pi 3. The values from the sensors are authenticated and processed by the raspberry pi by comparing throughput values from the web server. The raspberry pi communicates with the web server through an internet module. This system prioritizes the values from the rain drop sensor to make decisions without checking the soil moisture level.

For this system, weather predictions and decisions were not made. The pumps are turned off immediately when rain is picked up by the rain drop sensor. The drip irrigation method used by this system is applaudable.

5. Smart Irrigation System using Internet of things approach

Dr. S. J. Muneeswaril [11] , considered an automatic irrigation system using the Arduino microcontroller with moisture sensor and water flow management. The humidity sensor unit consists of an Arduino board, Wi-Fi unit, humidity sensor (DHT 11A) and water flow control mechanism. Along with moisture sensor, the temperature sensor output was also be taken into consideration while irrigating the farm. If the moisture content of soil is very low and the temperature is very high then, the system irrigates the plants. The time for which irrigation will be carried out is different for different temperature range. The reason being that, if the temperature is very high then the evaporation rate is also very high and hence the system has to irrigate for a longer period of time in order to attain the proper moisture level in the soil. Hence for different temperature range and moisture content level in the soil the farm will be irrigated for different time interval. The data taken from the humidity sensor will be sent to a data monitoring system by Arduino boards over a wireless network using Wi-Fi. At the monitoring system, the humidity levels are monitored and any decrease in humidity below a limit, will be reported as requirement for water and signal is raised to the entire humidity sensor unit to open the water flow management. Furthermore, humidity level in agricultural field can be checked any time through the web portal.

For this system, the moisture level of the soil was not taken into consideration. Also farmers were not given the flexibility to control the system from the web portal created for the system. The system does not update the farmer on the current weather condition.

6. Automatic watering Device for Tomato using soil moisture sensor

B. I. Agustinus et al [12], implemented a system that used a real time clock to check if the current time is within 7:00 am to 17: 00 pm, the system will work only if it is within this period. The system then checks for the soil moisture level with a soil moisture sensor if the current time is within the mentioned period. If the soil moisture sensor value is below 70% the irrigation will automatically start until it the soil moisture sensor value is greater than 70%. An LCD screen is attached to the system to display the current situation of the system and moisture sensor value.

The system does not store the data from the sensors. The farmer can only see if the pump is on or off but not the data from the sensor. No control ability is given to the farmer over the system. Also, the time bound of the system limits the efficiency of the system implemented.

7. Design of an Automated Irrigation System: Student Paper Competition

Marie et al. [13], reported that the sensed data, regarding the quantity of water content level in the soil that was obtained from a moisture sensor can significantly improve the agricultural yield. This would motivate researchers to perform extensive studies in various fields of interest that are related to the irrigation systems. A significant advancement in measuring the development of plant growth is the minimization of the costs of irrigation management, particularly for a mechanized watering system. This approach would significantly reduce water wastage and lessen the amount of human power required for the continuous monitoring of the source of water supply in an irrigation system. The use of a feedback-based system would efficiently help track and manage resources, compared with open-loop systems that are at the expense of higher complexities. Thus, soil moisture is difficult to correctly measure, and the target moisture content levels are difficult to sustain.

The limitations of this proposed irrigation system is that, cloud abilities were not included and farmers cannot view any data graphically from the proposed system.

8. Wireless Smart Irrigation System

M. Senapati et al [14], implemented a wireless smart irrigation system. This system is micro control based and can be operated from remote location through wireless transmissions. Soil moisture sensor, temperature sensor, humidity sensor and rain sensor was used in this system. These sensors are connected to an Arduino Uno which processes data from these sensors and make decisions on whether to start irrigation or stop irrigation. Decision making is controlled by the farmer using the microcontroller. The data received from the sensors are sent to sever database using wireless transmissions. Irrigation is automated when the moisture and temperature of the farm is reduced. The farmer is notified periodically with information regarding the field through the farmer's mobile.

The flaw of this system is that, even though farmers are given the privilege to control the system from remote locations over the internet, scheduling for irrigation is not implemented.

2.2. Proposed Work

This project aims to design and implement an IoT based smart drip irrigation system using wireless sensor network architecture for tomato farmers. This system comprises of a microcontroller which act as a processor and at the same time can transmit data to the cloud. The function of the proposed system include the following;

1. The farmer is given the privilege to schedule for irrigation using the android application.
2. The system will measure soil moisture using soil moisture sensor, approximately 1 sensor per hectare will be deployed.
3. The system starts or stops irrigation automatically using the sensor values from the soil moisture sensor.
4. The system, using cloud services, stores its data. This data can be viewed graphically by the farmer using a smartphone.

2.3. Scope of the Project

The thesis focuses on the development of an internet of things based smart irrigation system with drip irrigation being the adopted irrigation system. In the development of the system, a soil moisture sensor is used to pick up soil moisture levels from the soil, a Node MCU interfaced with a water pump and a Wi-Fi module will be programmed to process data from each sensor in the network. Based on the data received from the sensor, the water pump will be turned on or off for irrigation to take place and data will be sent to a cloud platform. We will also design an android application on which users can see the state of the system. The users can also schedule irrigation at their preferred time using the mobile application. This project will take 8 months to complete. Table 2.1 below shows the timeline of the project.

TABLE 2.1: TIMELINE OF THE PROJECT

Activity/Month	Jan	Feb	March	April	May	June	July	August	Sept
Literature Review	🟧								
Documentation	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
Research	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Node MCU Programming	🟩	🟩	🟩	🟩	🟩				
Android application Development				🟦🟥	🟦🟨	🟦🟥	🟦🟨		
Interfacing the Node MCU						🟪🟦	🟪⬛		
Testing				🟥	🟥	🟥	🟥	🟥	🟥

CHAPTER 3- SYSTEM DESIGN AND DEVELOPMENT

3.0. **Introduction**

Since the IoT drip irrigation system could be deployed on large tomato farms, our smart drip irrigation system was designed using the wireless Network Architecture. In this architecture, there are sensors, sensor nodes and a microcontroller. Our system have only soil moisture sensors and nodes. Our nodes can perform computation and also handle wireless communication to our cloud.

3.1. System Overview and Functions

Using wireless sensor network architecture, our smart drip irrigation system has soil moisture sensors to collect soil moisture data. These soil moisture data from the sensors are transmitted to the node. Each soil moisture sensor is connected to a single node. These soil moisture sensors are capacitive sensors and do not corrode easily. Data sent to the node, in our architecture, is processed and sent to the cloud for the tomato farmer to view the data the node is processing. Node MCU Esp8266 is used as the node in the system. A program uploaded to the Node MCU helps it to control the system efficiently.

The Node MUC processing the real time data from the soil moisture sensor also checks that is the soil moisture in the soil is sufficient or inefficient. There is set threshold for which when the soil moisture exceed, the node MCU sets it to normal. The set threshold is when the soil moisture is below 17%. When the soil moisture comes below this threshold, the Node MCU controls the DC Water pump to pump water through the drip pipe to dip directly onto the roots of tomatoes. The DC water pump would stop pumping water when soil moisture is sufficient.

In our mobile interface for the system, the farmer can schedule for irrigation. When the scheduled time is up, a notification is sent to the farmer to start the irrigation. At the time the irrigation is set and the soil moisture in the soil is above 80%, irrigation would not start and alert the farmer that the soil moisture is sufficient. Weather widget in the application allows the farmer to know if it going to rain or going to be sunny. This would help the farmer make good decision on when to schedule for irrigation.

3.2. Requirements, Analysis and Specifications

This project seeks to implement an automated smart irrigation system using the drip irrigation system. In this era of climate changes where one cannot, based on past experiences, predict what could happen during a particular season. We as a country that largely practice farming based on seasons need to employ a new system of farming that do not largely depend on the increasingly unpredictable weather. To achieve this aim, there are certain functionalities and requirement that must be met. These functionalities are grouped into functional and non-functional requirement and are discussed below.

3.2.1. Functional Requirement

The functional requirement for both the system and the mobile application are;

1. The system should automatically check for the soil moisture level and process the data to inform its decisions as to start irrigation or not.
2. The system should allow farmers to view the current state of the system remotely.
3. The system should allow the farmer the ability to control the system remotely.
4. The mobile application should register and authenticate farmers using firebase authentication services.
5. The mobile application should allow farmers to schedule for irrigation at their own preference time.
6. The mobile application should use the weather to make decision when scheduled irrigation time is due.

3.2.2. Non-functional Requirement

The non-functional requirement for both the system and mobile application are as follows;

1. The system should have a short response time.
2. The mobile application should be user friendly.
3. The system should be reliable and always available.

3.3. System Design and Development Process

In the design of our system, we first connect the soil moisture sensor to an analog pin on our node MCU, on the node MCU, we will have an Arduino code on it which will process the data from the soil moisture sensor. In that same code, we also send the sensor data to a thingspeak channel and also connects the system to a firebase real time database to handle user requests from our application for wireless control of the system.

3.3.1. System Architecture

Figure 3.1, shows the system architecture of our system. From figure 2, each soil moisture sensor is connected to one node MCU ESP8266. The node MCU is the processor in this architecture, it is connected to a cloud server wirelessly through the internet. The farmer using the application is also connected to the cloud server through the internet, requests made by the farmer is sent to the cloud server. The Node MCU then picks the farmers request and process it to control irrigation on the farm. Also, when the soil moisture level is at a level which can lead to plant stress, the system will automatically start irrigation and if the level of the moisture is above the needed level, the system will stop irrigation and it will do so even if the farmer tries to trigger the irrigation from our application.

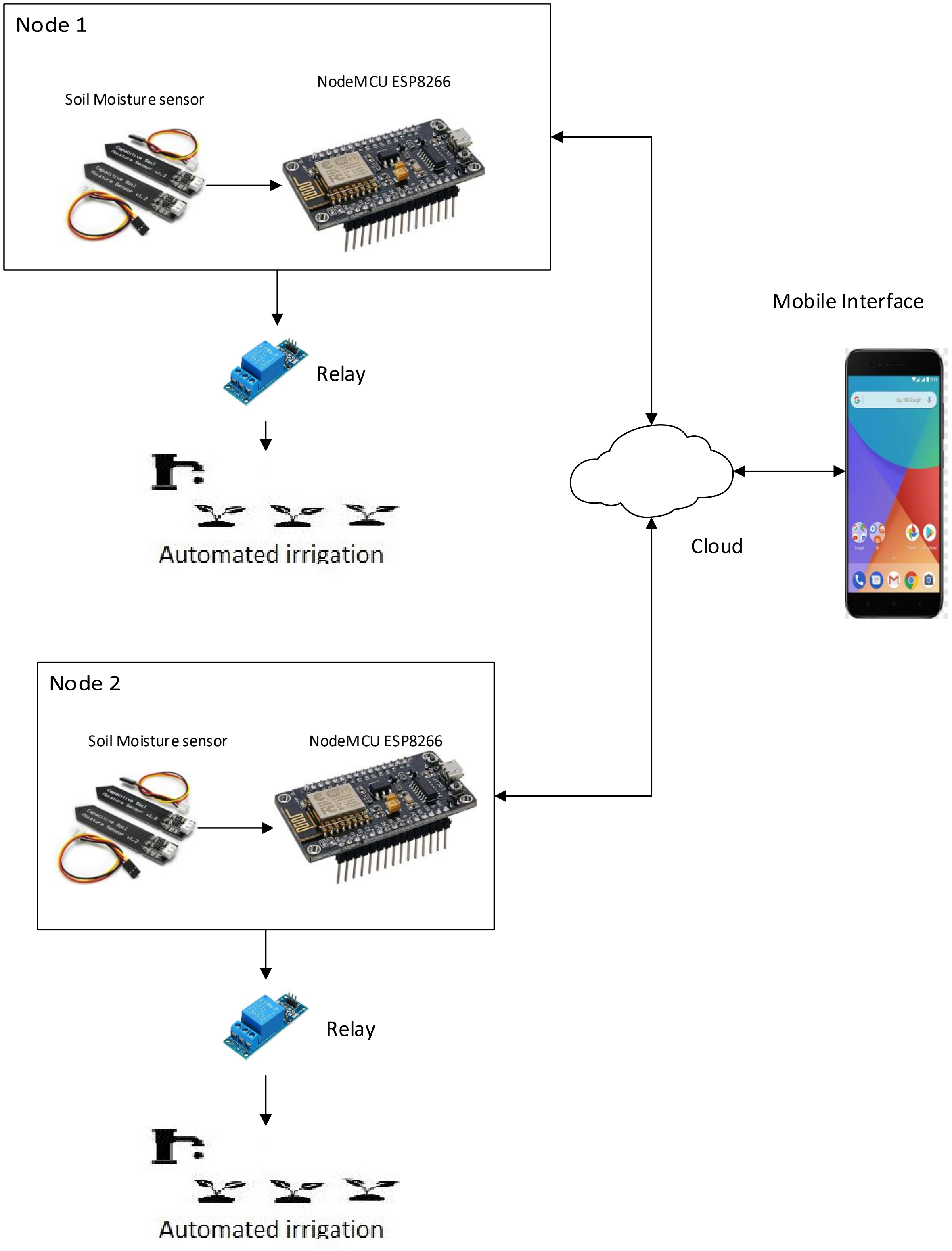


FIGURE 2.1: SYSTEM ARCHITECTURE

3.3.2. System Flow Chart

From **Figure 3.2** below**,** when the system is connected to power source or is started, it first checks for the soil moisture level using its soil moisture sensor. The analog data picked by the soil moisture sensor is sent to the node MCU for processing. The node MCU checks if the data sent from the soil moisture sensor is less than or greater than 49% with values being greater depicting dry soil. If the value is greater than 49%, the system will turn on the water pumps to start irrigation till the moisture reading is less than 33%. When soil moisture reading is less than 33%, pumps are turned off. After every 10 seconds, the soil moisture sensor data value is pushed to the cloud for our mobile application to get access to the data. We then visualize all the data sent to the cloud platform in our application for the farmer to get a graphical representation of the soil condition.

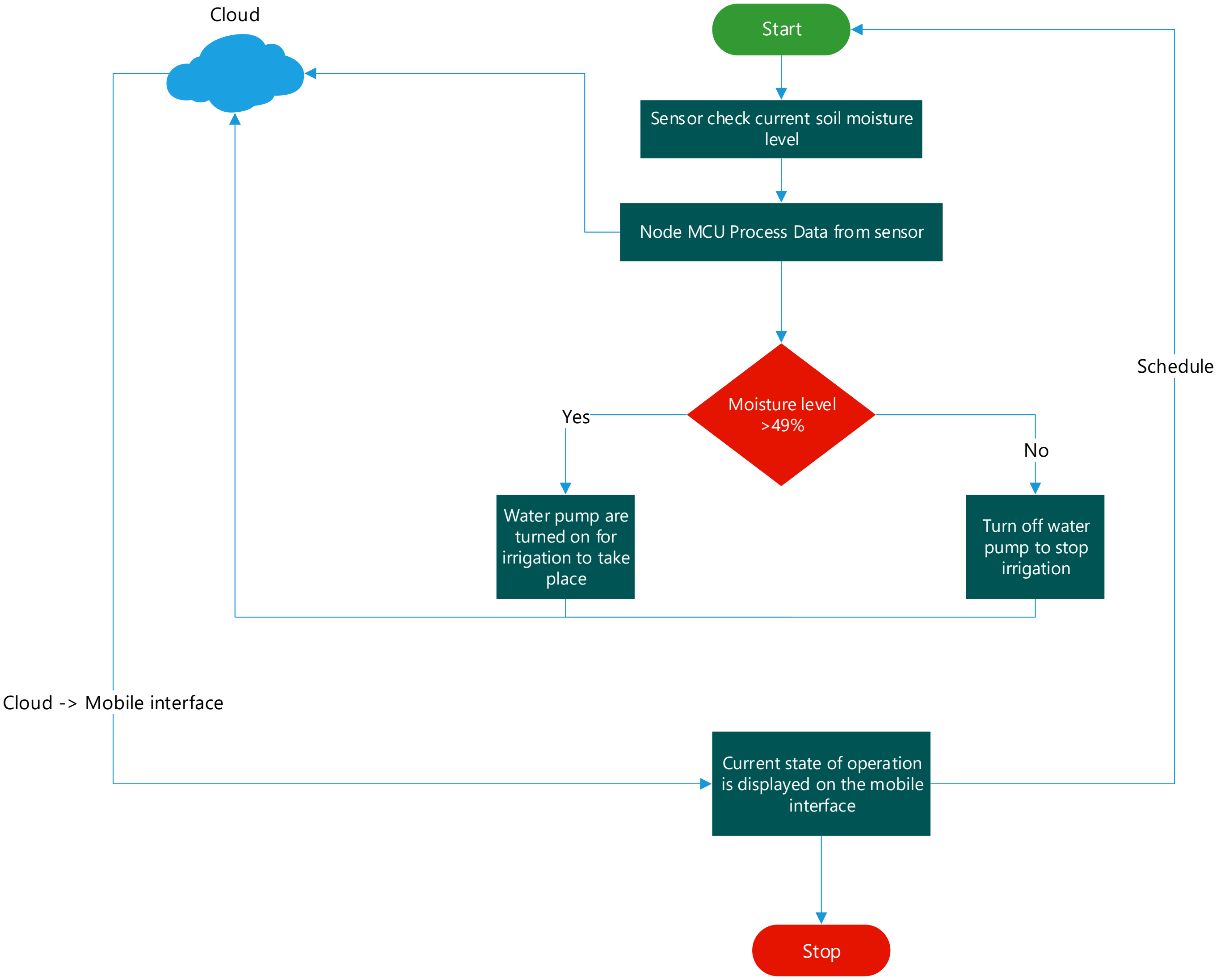

Cloud
Start
Sensor check current soil moisture level
Node MCU Process Data from sensor
Schedule
Moisture level >49%
Yes
No
Water pump are turned on for irrigation to take place
Turn off water pump to stop irrigation
Cloud -> Mobile interface
Current state of operation is displayed on the mobile interface
Stop

Figure 3.2: System Flow Chart

3.3.3. Flow Chart For Scheduling

When the time scheduled by the farmer for irrigation to start using the application is up, a weather API (Application Programming Interface) integrated in our application first checks if there is a chance of rain or it is actually raining, if so, irrigation will not start and the farmer will be alerted, if the possibility of rain is low, the signal is sent to the system. When the system receives this signal, it first check the current soil moisture level. If the current soil moisture level is above 33%, irrigation start till the moisture sensor reading gets below 33%. If the current soil moisture is below 33%, irrigation does not start.

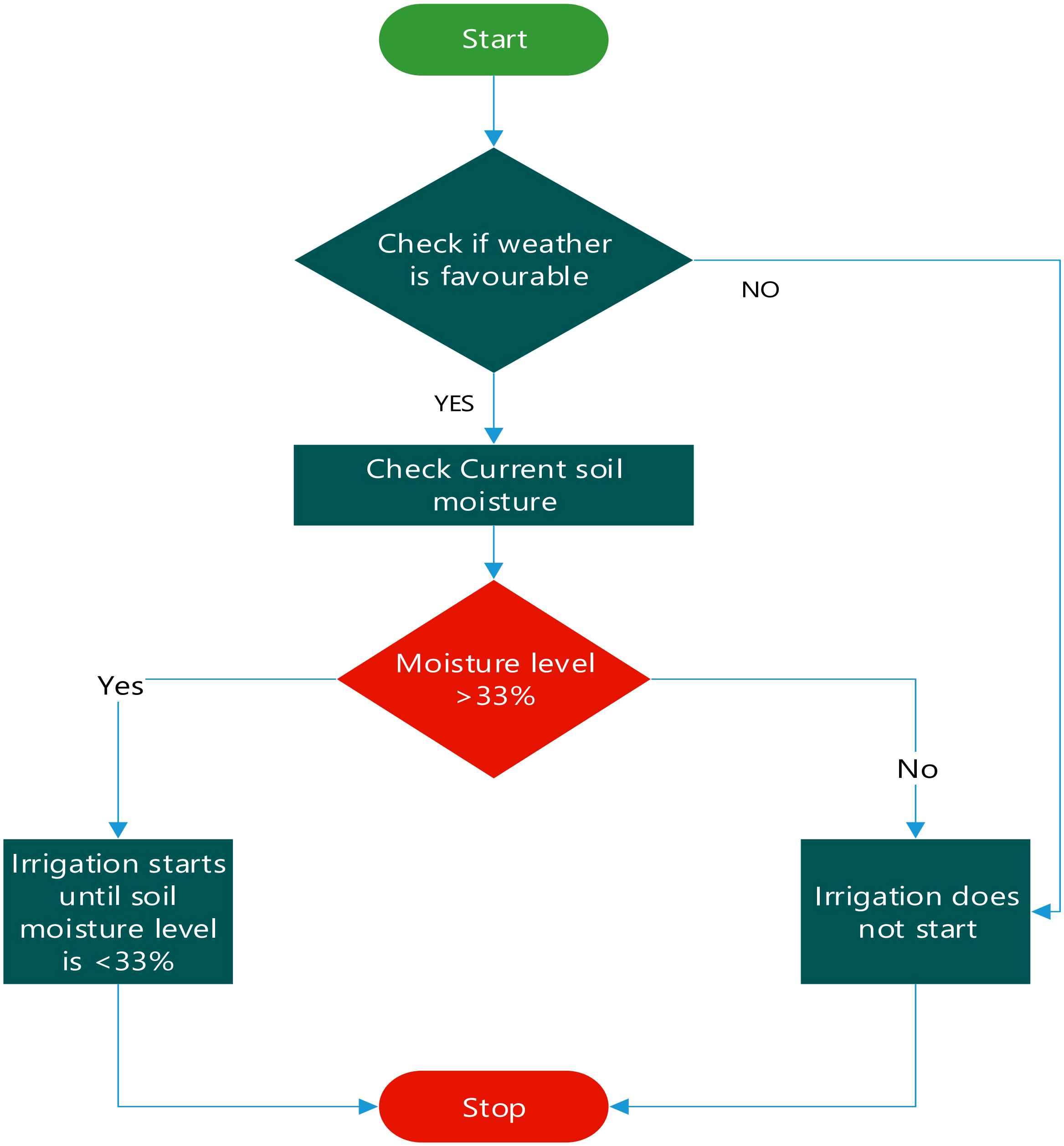

Start
Check if weather is favourable
NO
YES
Check Current soil moisture
Moisture level >33%
Yes
No
Irrigation starts until soil moisture level is <33%
Irrigation does not start
Stop

Figure 3.3: Flow chart for scheduling

3.3.4. Application Use Case Diagram

Figure 5 shows the activities that can be performed by the farmer using the application**.**

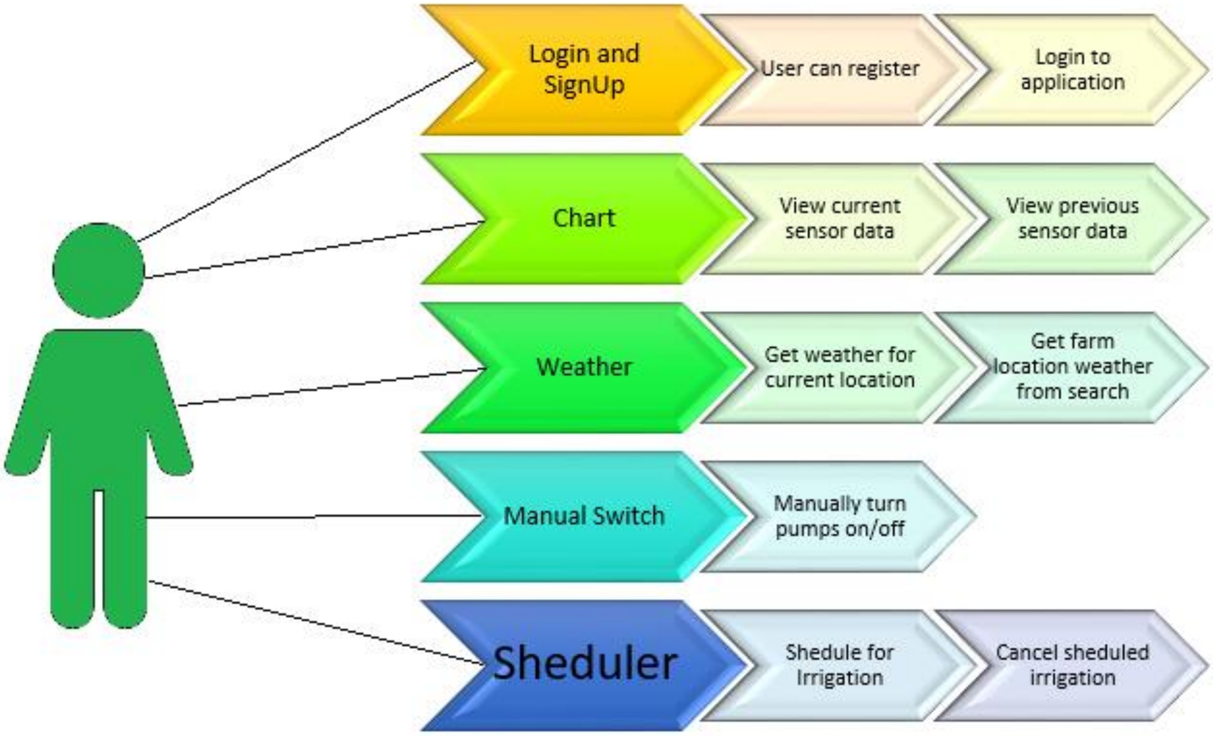
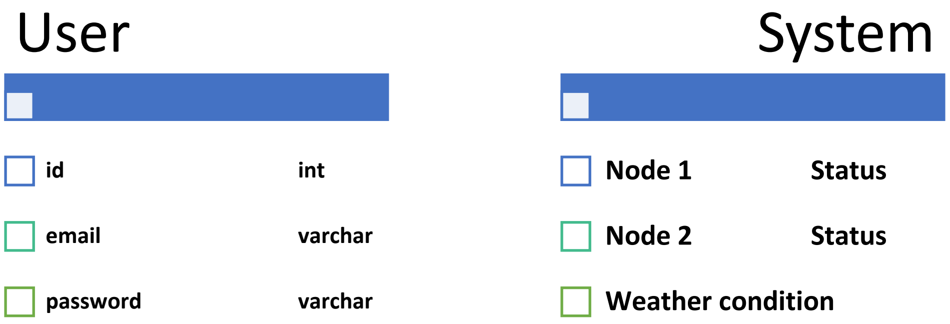

Login and SignUp
User can register
Login to application
Chart
View current sensor data
View previous sensor data
Weather
Get weather for current location
Get farm location weather from search
Manual Switch
Manually turn pumps on/off
Sheduler
Shedule for Irrigation
Cancel sheduled irrigation

Figure 3.4: Application user case diagram

3.3.5. Database Design

The database was implemented using a firebase real time database which is a NoSQL cloud hosted database. For the system data, using firebase, the system can be triggered. To achieve this, the status of the pumps will have to be stored as “on” or off”. Storing the weather condition helps in the decision making when schedule time for irrigation is up.



User	
□ id	int
□ email	varchar
□ password	varchar

System	
□ Node 1	Status
□ Node 2	Status
□ Weather condition	

3.3.6. Cloud Storage

Thingspeak, a Matlab cloud storage service provider for IoT, was chosen for this project. Each sensor value captures by the system, is stored in JSON format and synchronized in real time to each user. Also, a firebase real time database was also used for the wireless control of the system.

3.4. System Modelling and Simulation

In the simulation of this system, we used Proteus design suite. This design suite did not come with some of the components needed for the simulation of the smart drip irrigation system. We used other available components that can represent the actual component needed for the simulation.

We used a potentiometer to represent a variations in the soil moisture, Arduino UNO to represent Node MCU, motor to represent the dc motor pumping water to the tomato plant. Using a virtual terminal, we were able to view the state of the system. Varying the potentiometer between 100% and 0% helps vary the soil moisture data the capacitive sensor is collecting and sending to the system. A sketch uploaded to the Arduino UNO was programed using Arduino IDE. **Figure 3.5** shows the system simulation.

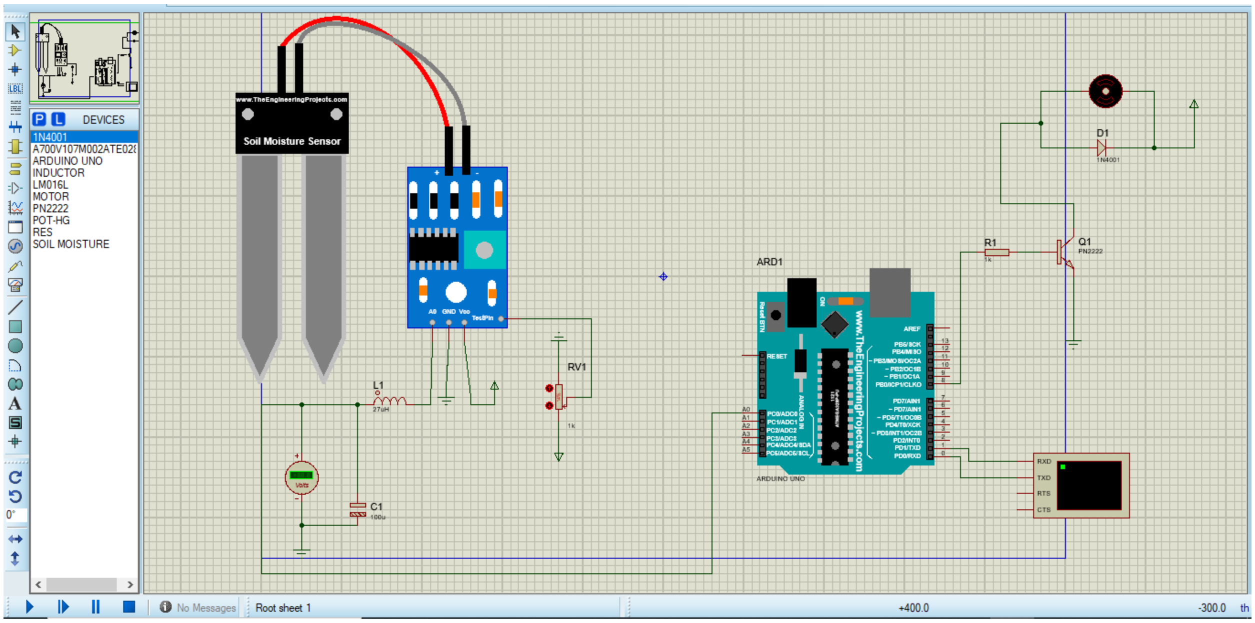
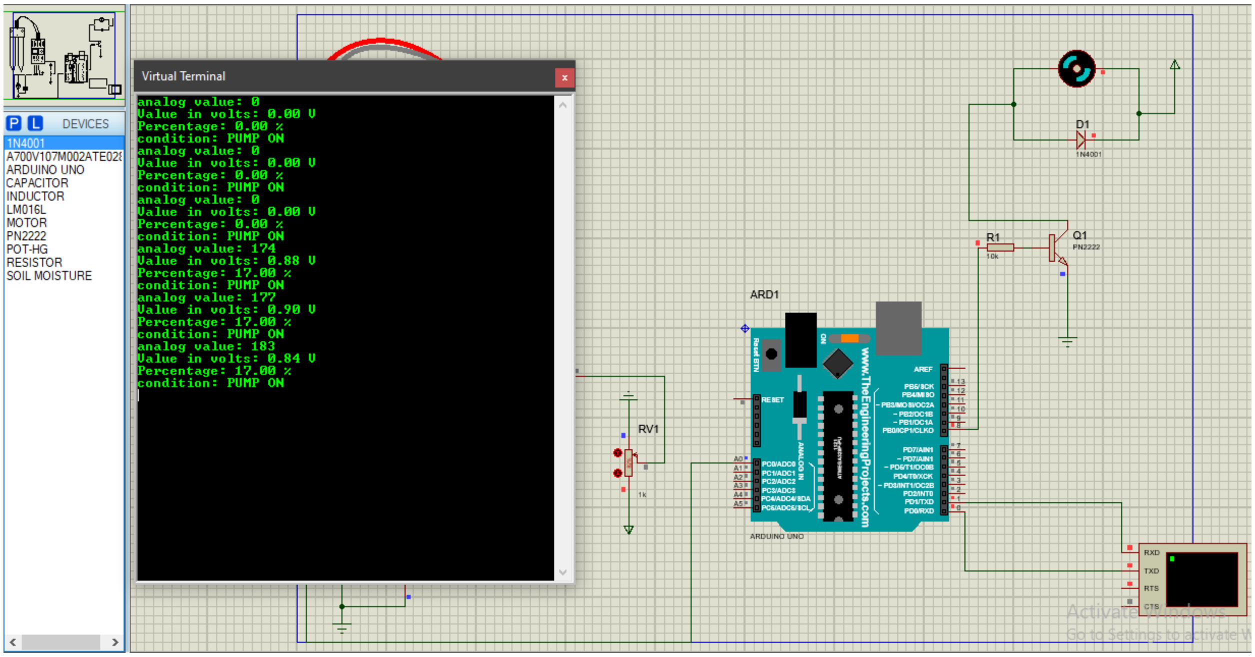


FIGURE 3.5: SMART DRIP IRRIGATION SYSTEM MODEL SIMULATION



```
analog value: 0
Value in volts: 0.00 V
Percentage: 0.00 %
condition: PUMP ON
analog value: 0
Value in volts: 0.00 V
Percentage: 0.00 %
condition: PUMP ON
analog value: 0
Value in volts: 0.00 V
Percentage: 0.00 %
condition: PUMP ON
analog value: 174
Value in volts: 0.88 V
Percentage: 17.00 %
condition: PUMP ON
analog value: 177
Value in volts: 0.90 V
Percentage: 17.00 %
condition: PUMP ON
analog value: 183
Value in volts: 0.84 V
Percentage: 17.00 %
condition: PUMP ON
```

FIGURE 3.6: RUNNING SMART DRIP IRRIGATION SYSTEM MODEL SIMULATION

Figure 3.6, shows the virtual terminal of our running simulation. In the simulation, we variated the potentiometer and our simulation was responding to those variations. When the potentiometer is variated to 100%, meaning the soil has enough moisture, the motor stop running. When the potentiometer is variated to below 17%, the motor rotates, indicating the start of irrigation.

3.3. Development Tools and Material Requirements

3.3.1. Material Requirement

3.3.1.1. Node MCU ESP8266 Wi-Fi module.

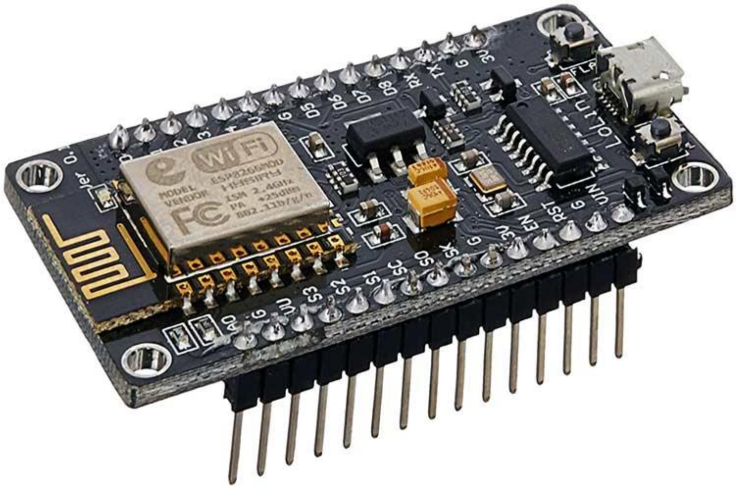


FIGURE 3.7: NODE MCU ESP8266 WI-FI MODULE

Node MCU ESP8266 Wi-Fi module is a microcontroller that has a Wi-Fi module integrated on it. It is used as the node for our system. The node MCU is made up of Tensilica 32-bit RISC CPU Xtensa LX106 microcontroller. The table below shows the specifications of the Node MCU ESP8266.

TABLE 3.1: SPECIFICATION FOR NODE MCU ESP8266

Specification	Node MCU ESP8266
Microcontroller	Tensilica 32-bit RISC CPU Xtensa LX106
Operating Voltage	3.3V
Input Voltage	7-12V
Digital I/O pins	16
Analog input Pins	1 (A0)
Flash Memory	4MB
SRAM	64KB
Clock Speed	80MHz

3.3.1.2. Capacitive Soil moisture Sensor

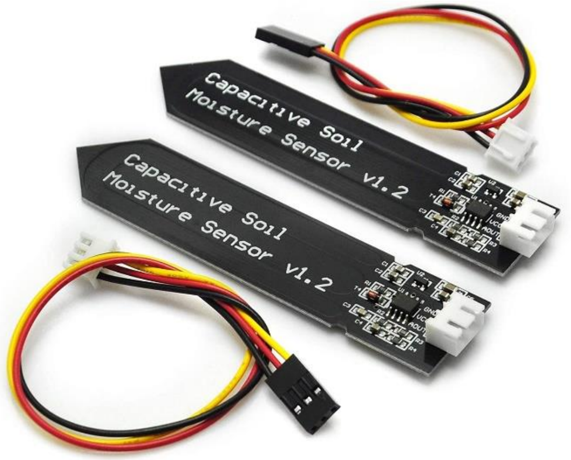


FIGURE 3.8: CAPACITIVE SOIL MOISTURE SENSOR

Capacitive soil moisture sensor is a device that is use to pick real time analog data from a soil. The sensor sends data in bits (0-1023). It uses change in capacitance to determine the moisture level in a soil. The part of the sensor inserted in the soil is coated with solder which prevents it from corroding while in the soil. With this sensor, the higher the sensor value, the dryer the soil and vice versa. Table 3.2 below shows the specifications of the sensor.

TABLE 1.2: SPECIFICATION OF CAPACITIVE SOIL MOISTURE SENSOR

Specification	Capacitive Soil Moisture Sensor
Operating Voltage	3.3 – 5.5 VDC
Output Voltage	0 – 3.0 VDC
Interface	PH2.54 – 3P
Version	2.0
Operating Current	5mA

3.3.1.3. Relay Module

FIGURE 3.9: RELAY MODULE

The relay module is an electrically operated switch that can be turned on or off deciding to let current flow through or not. Table 3.3 below shows the specifications of the relay module.

TABLE 3.3: SPECIFICATION FOR THE RELAY MODULE

Specification	Relay module
Supply voltage	3.75V to 6V
Quiescent current	2mA
Current when the relay is active	70mA
Relay maximum contact voltage	250VAC or 30VDC
Relay maximum current	10A

3.3.1.4. 12Volts DC submersible water pump

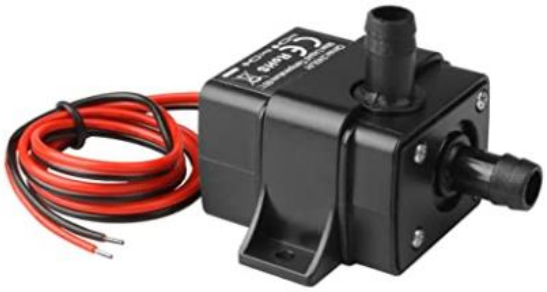


FIGURE 3.10: 12V DC SUBMERSIBLE WATER PUMP

DC submersible water pump is an electronic device that is used to pump water from a water source given a command. Table 3.4 below shows the specification of the DC water pump.

TABLE 3.4: SPECIFICATION OF SUBMERSIBLE DC WATER PUMP

Specification	12V DC submersible water pump
Operating voltage	12VDC-24VDC
Maximum discharge flow	3.2 GPM
Pump type	Submersible pump

3.3.1.5. Micro Drippers

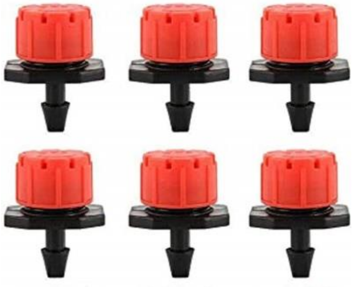


FIGURE 3.11: MICRO DRIPPERS

Micro drippers connected to drip lines are used to implement drip irrigation. Table 3.5 below shows the specification of the micro dripper.

TABLE 3.5: SPECIFICATION FOR MICRO DRIPPERS

Specification	Micro Dripper
Flow rate	1.75 liters/hour
Sealing Pressure	1.7 PSI

3.3.1.6. Other Components Used

i. Breadboard

FIGURE 3.12: BREAD BOARD

Bread board was used to connect electronic the moisture sensors and node MCU.

ii. Jumper Wires

Jumper wires were used to connect all the component of the system.

FIGURE 3.13: JUMPER WIRES

3.3.2. DEVELOPMENT TOOLS

i. Arduino IDE

Arduino IDE is an environment that is used to develop and upload code to Node MCU ESP8266 microcontroller. This code is used to control the processes of the water pump and relay. The code uploaded on the Node MCU ESP8266 was written in C++ programming language.

ii. Android Studio

Android studio is an integrated development environment that is used to develop mobile applications. Applications developed with this IDE is for android operating system. Java programming language was used to code the mobile application. Android studio 4.0 was used.

CHAPTER 4 –IMPLEMENTATION AND TESTING

4.0. Introduction

This section details how our system was implemented and also tested to verify that our objectives for the project was met.

4.1. System Implementation Process

4.1.1. THE GENERAL SYSTEM IMPLEMENTATION

i. An internet of things based smart irrigation system would be designed using a node MCU esp8266, capacitive soil moisture sensor, a relay and a 12v water pump.

ii. The system is built based on the wireless sensor architecture where there exist multiple nodes. For our project, we built two nodes.

iii. Each node consist of a node MCU esp8266, a capacitive soil moisture sensor and a water pump.

iv. The soil moisture sensor detects the level of water in the soil and sends it to the node MCU esp8266.

v. The node MCU esp8266 which is a combination of a microcontroller and a Wi-Fi module processes the data received from the sensor.

vi. The node MCU esp8266 then forwards the processed data to a thing speak channel for storage and also sends turns the water pump on or off based on the value of the processed data.

vii. The node MCU esp8266 also sends the condition of the system (on or off) to a firebase real time database.

viii. An android application built is used to monitor the system, schedule for irrigation at any place and also control the system anytime.

ix. The firebase database is also used to store the data of users when they register on the application.

4.1.2. HARDWARE SUBSYSTEM IMPLEMENTATION

i. Before our system was physically built, we first implemented a Proteus simulation of the project to see how feasible it will be.
ii. We went on to purchase the various components and also test the individual components to check if it was working properly.
iii. A C++ code was written using the Arduino ide to implement the logic of how the system will operate.
iv. The code was uploaded unto the node MCU esp8266. The logic of the code was same for each microcontroller but the thingspeak and firebase fields were different for each microcontroller.

v. The capacitive soil moisture sensor were connected to the node MCU esp8266 on a bread board using jumper wires.

vi. In order to connect the 12V water pump to the node MCU, we had to connect a relay to the node MCU, and then connected the water pump to the relay. This was done so because the output voltage from the digital pin of the node MCU is not enough to drive the water pump.

4.1.3. Software subsystem implementation

i. The software was developed using the android studio IDE using the java programming language.

ii. In the software, a register and login page was designed to allow users to register to use the application and also login into the application using the details they provided while registering the application. This is done to make sure anyone without the required details cannot access the data of a user.

iii. For our main page we designed a sidebar navigation menu with six modules. The modules are home, weather, manual switch, schedule, about and logout.

iv. For the home page, a graph which is plotted using the soil moisture data collected from the thingspeak channel is displayed to inform the user on the soil moisture levels.

v. For the weather page, user is allowed to enter location of farm and with the help a weather API we used, the current weather condition is shown.

vi. For the manual switch page, the user can manually turn the system on or off using the application.

vii. For the schedule page, the user can select a specific time off preference to start irrigation. When the time for irrigation is up and it is raining or the likelihood of rain is high, irrigation will not start and the farmer is also notified with a reason.

viii. For the about page, we display the details of our application and how it functions.

ix. For the logout page, users when they click here are taken back to login page.

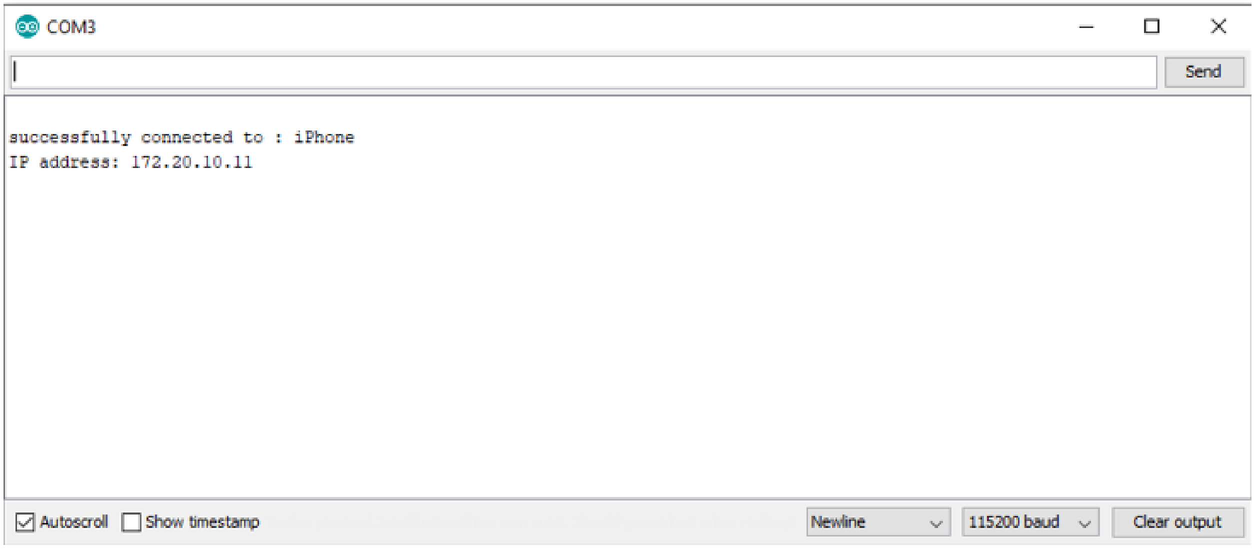
x. For storing of data, we used the firebase database.

4.2. Testing and Results

4.2.1. HARDWARE SUBSYSTEM

4.2.1.1. **Connecting node MCU to the internet**

From figure 4.1 below, you will realize that the node MCU has connected to a specified Wi-Fi network.

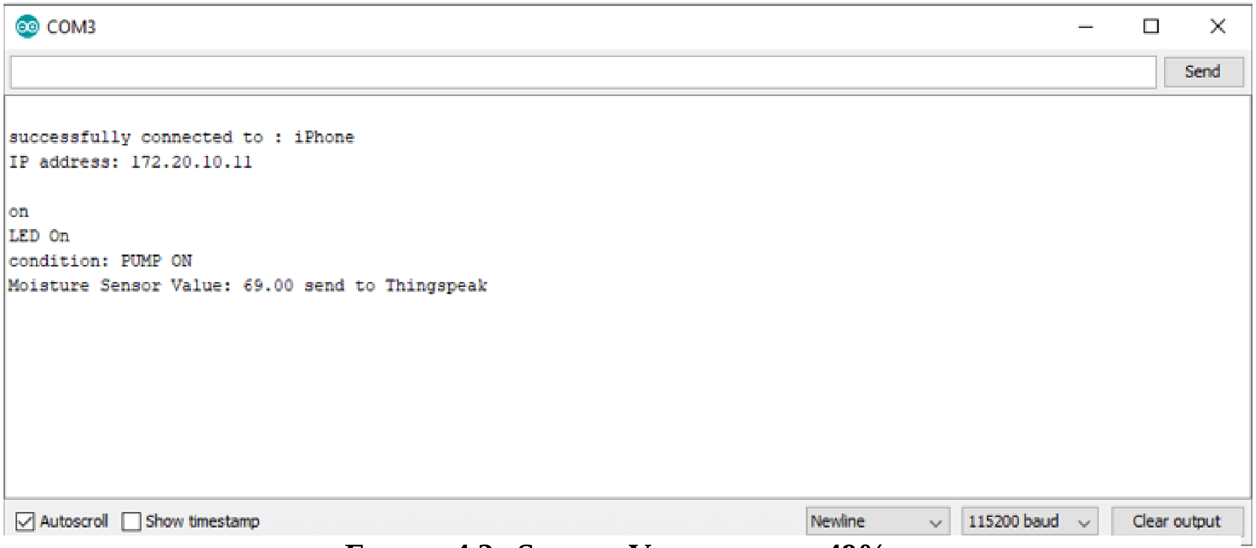


```
successfully connected to : iPhone
IP address: 172.20.10.11
```

FIGURE 4.1: CONNECTING NODE MCU TO THE INTERNET

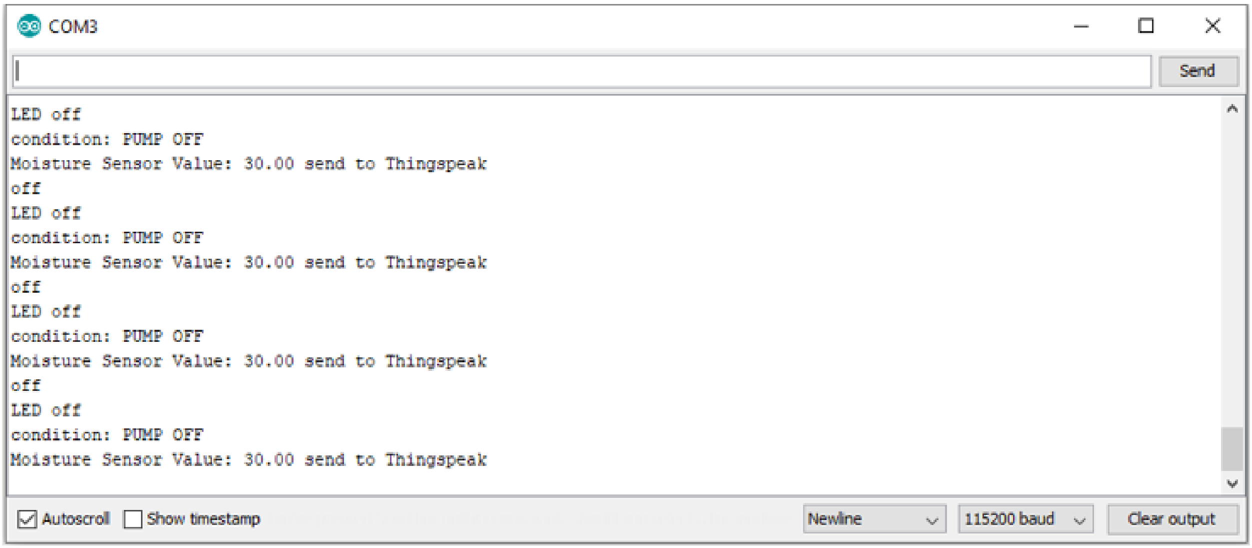
4.2.1.2. **Controlling Water Pump**

From figure 4.2 below, you will realize that the value after the IP address of the connected network is on, this represents the data read from our firebase real time database. The next shows the current condition of the system and it stays on because the actual sensor value is 69 which based on the operation of our system represents a very dry soil so the pump stays on.



```
successfully connected to : iPhone
IP address: 172.20.10.11

on
LED On
condition: PUMP ON
Moisture Sensor Value: 69.00 send to Thingspeak
```

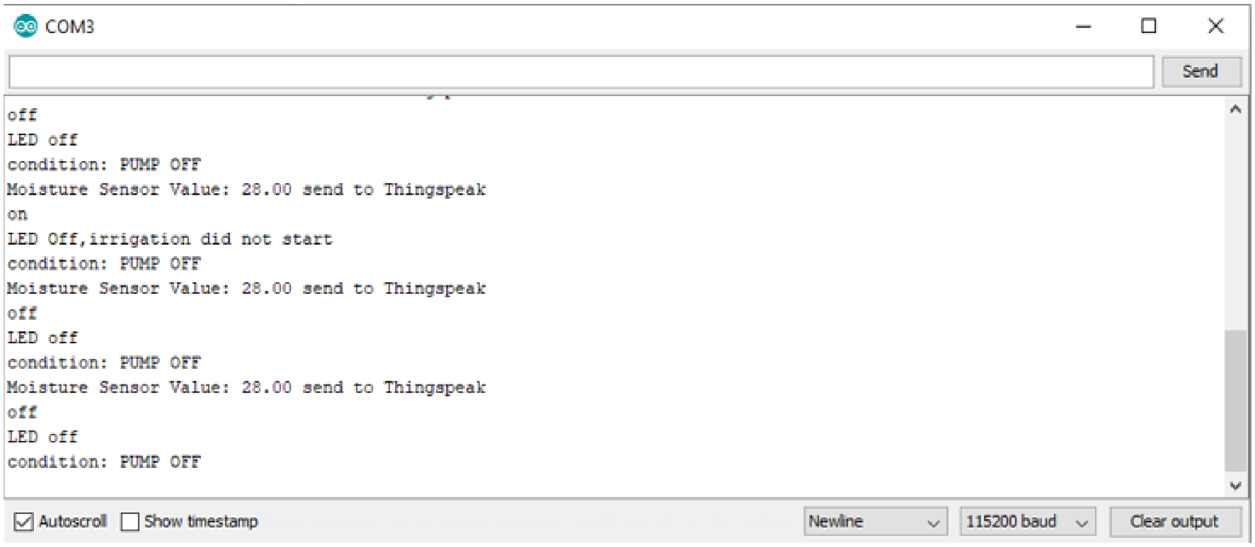
FIGURE 4.2: SENSOR VALUE ABOVE 49%



```
LED off
condition: PUMP OFF
Moisture Sensor Value: 30.00 send to Thingspeak
off
LED off
condition: PUMP OFF
Moisture Sensor Value: 30.00 send to Thingspeak
off
LED off
condition: PUMP OFF
Moisture Sensor Value: 30.00 send to Thingspeak
off
LED off
condition: PUMP OFF
Moisture Sensor Value: 30.00 send to Thingspeak
```

FIGURE 4.3: MOISTURE LEVEL BELOW 33%

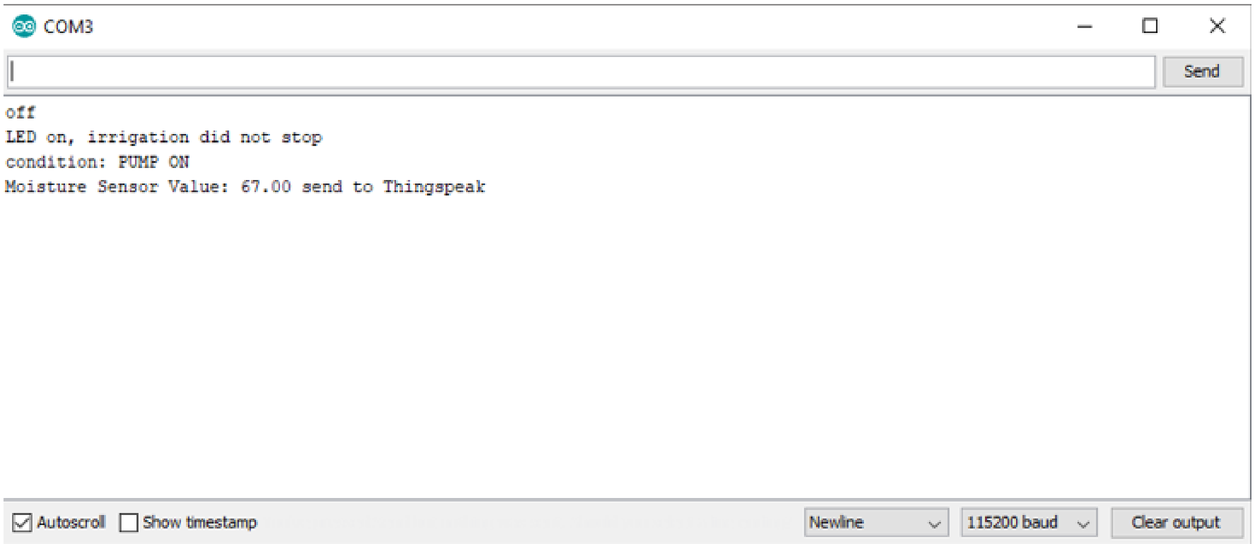
In figure 4.4 below, you will realize that on the 5th line, the system read on from the cloud database which depicts a user trying to turn on the system remotely at an instance or a scheduled time for irrigation is up, but on the 6th line you will realize the system did not turn on but stayed on. This is because we programmed the system to check if the soil moisture sensor is not below 33% whenever there is a command to turn on for irrigation.



```
off
LED off
condition: PUMP OFF
Moisture Sensor Value: 28.00 send to Thingspeak
on
LED Off,irrigation did not start
condition: PUMP OFF
Moisture Sensor Value: 28.00 send to Thingspeak
off
LED off
condition: PUMP OFF
Moisture Sensor Value: 28.00 send to Thingspeak
off
LED off
condition: PUMP OFF
```

FIGURE 4.4: TURNING ON IRRIGATION WHILE SENSOR VALUE IS<30%

From figure 4.5 below, on the first line you can see that the first line reads off which is a command from the real time database to turn off the system but it did not succeed as seen from subsequent lines due to the soil moisture level being above 49%.



```
off
LED on, irrigation did not stop
condition: PUMP ON
Moisture Sensor Value: 67.00 send to Thingspeak
```

FIGURE 4.5: TURNING OFF SYSTEM WHILE SENSOR VALUE IS>49

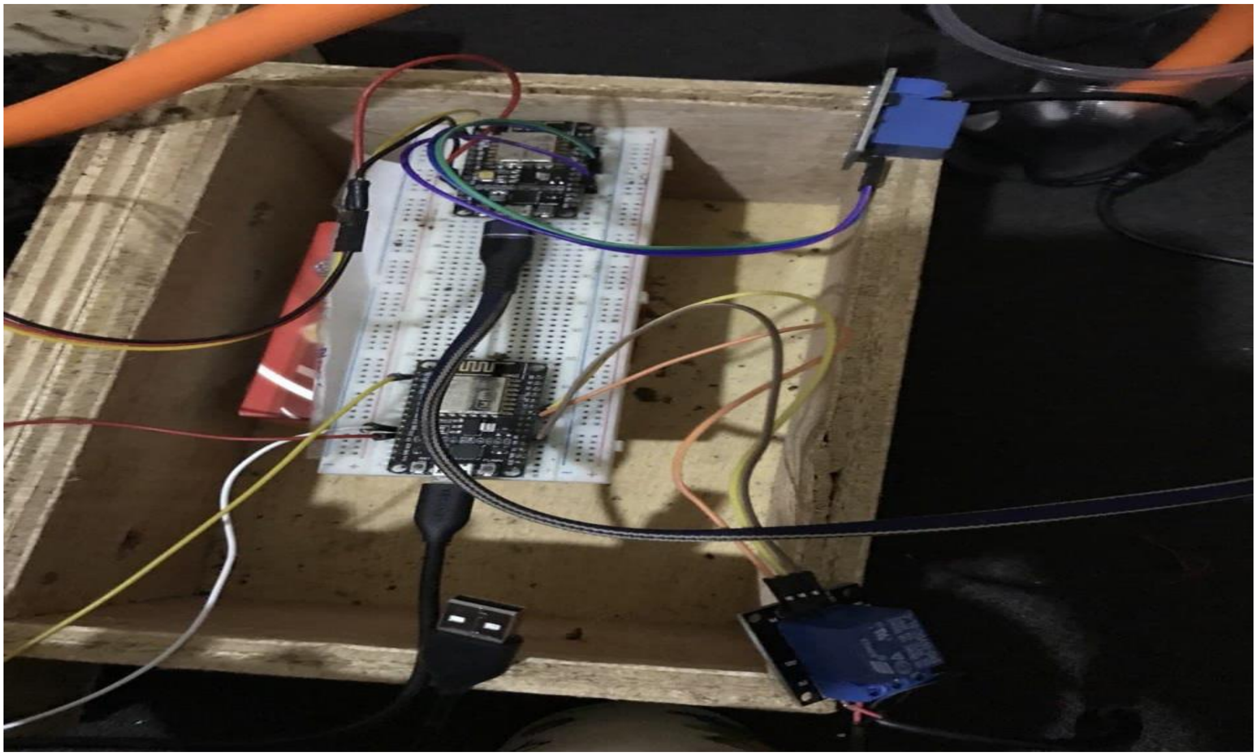


FIGURE 4.6: CONFIGURATION OF THE HARDWARE COMPONENTS

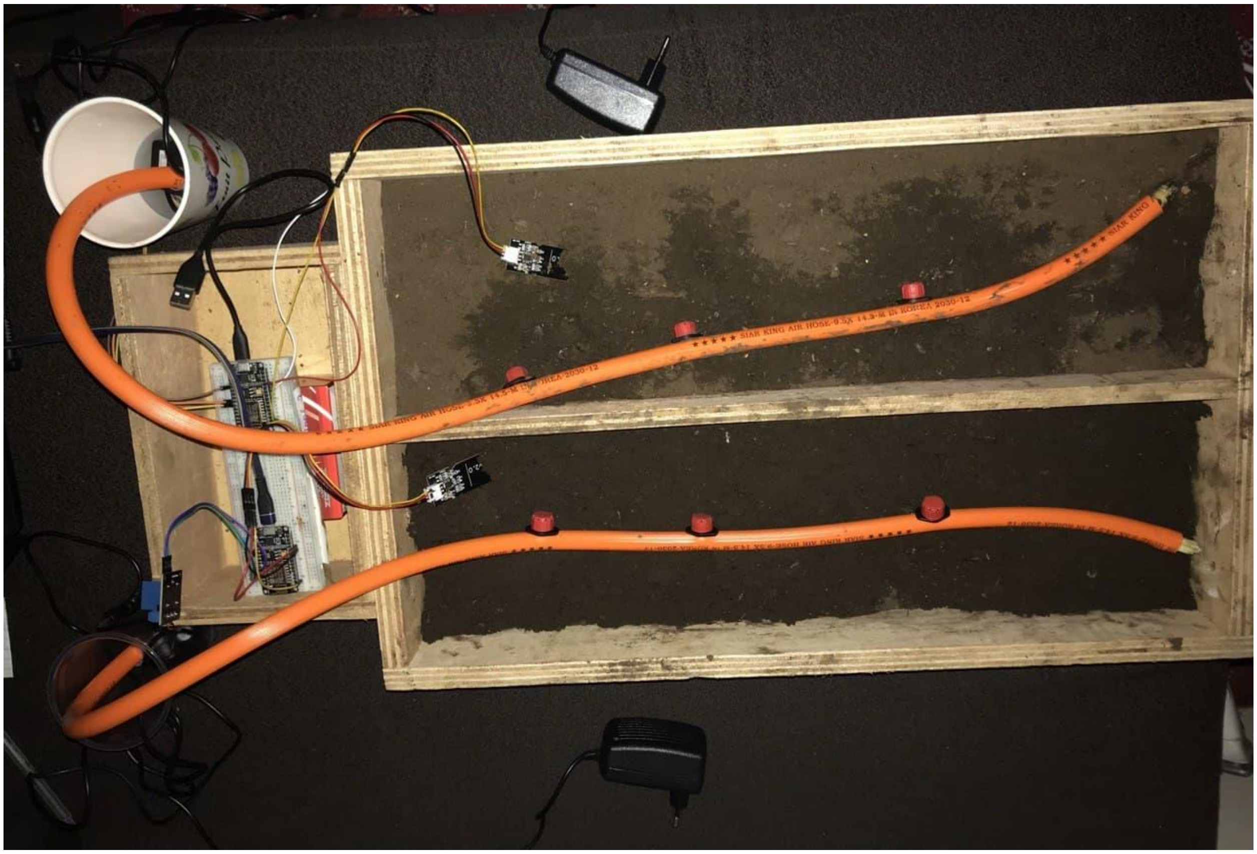
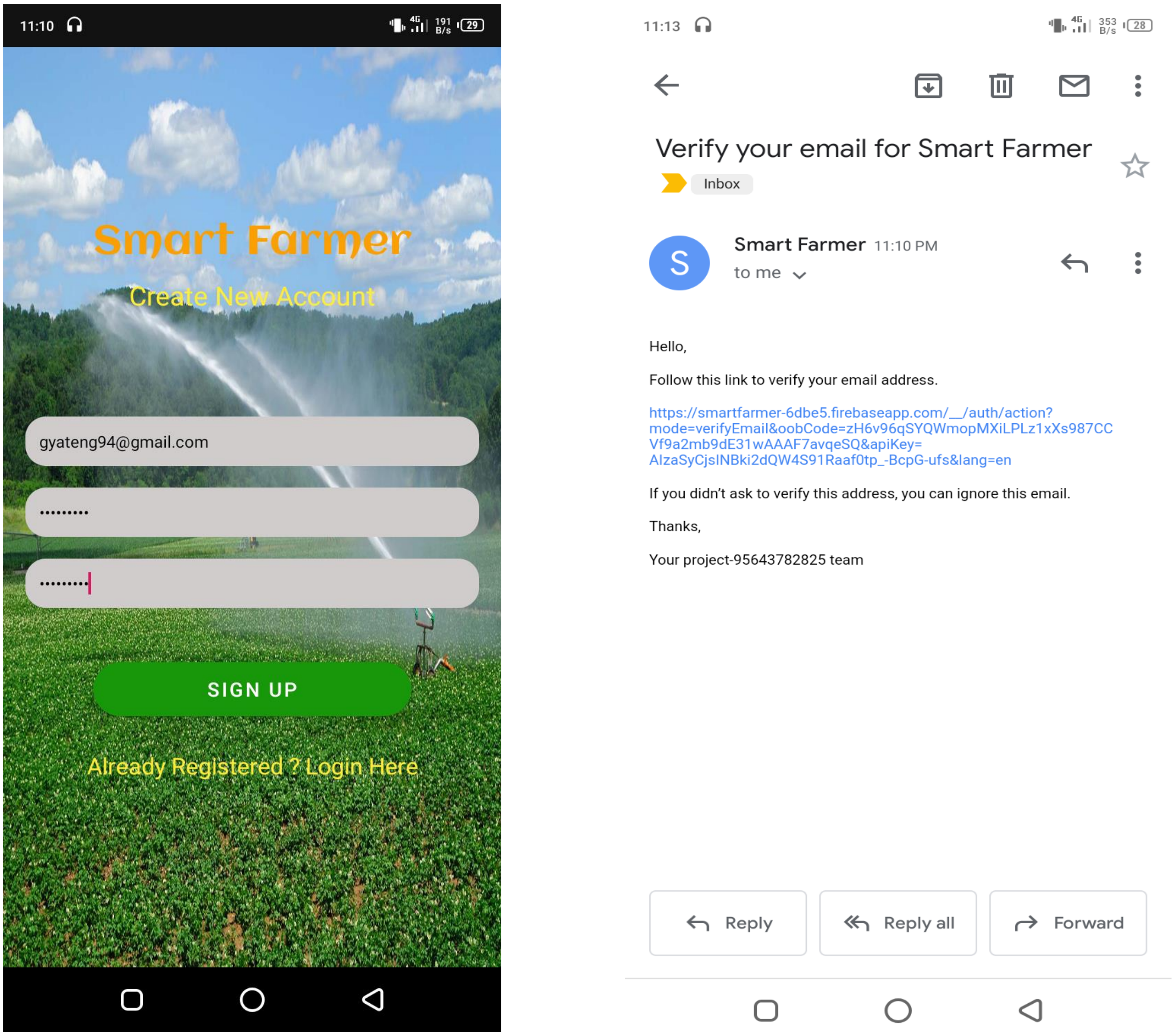


FIGURE 4.7: PROTOTYPE OF HOW THE SYSTEM WILL BE DEPLOYED ON A FARM

4.2.2. Software Subsystem

4.2.2.1. Registering to use our application

Here, you are allowed to register to use the application. If you already have an account, a button is provided for you to go to the login page to log in to your account. The requirements for registering are an email and a password with users being required to reenter their passwords to confirm their first password. Upon registering, you are required to verify your email through a mail sent to the email account you provided while registering. Below are images of both the instances already discussed in this section.



Verify your email for Smart Farmer

Smart Farmer 11:10 PM
to me

Hello,

Follow this link to verify your email address.

https://smartfarmer-6dbe5.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=zH6v96qSYQWmopMXiLPLz1xXs987CCVf9a2mb9dE31wAAAF7avqeSQ&apiKey=AIzaSyCjsINBki2dQW4S91Raaf0tp_-BcpG-ufs&lang=en

If you didn't ask to verify this address, you can ignore this email.

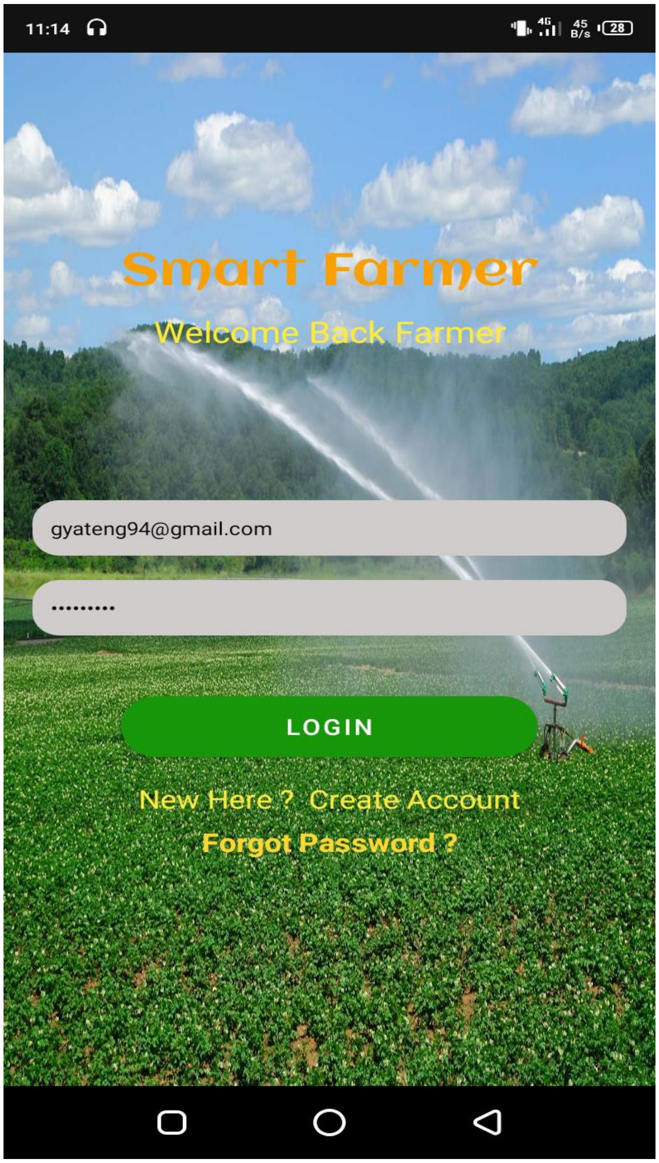
Thanks,

Your project-95643782825 team

Figure 4.8: Sign up page and Email verification sent to users

4.2.2.2. Login

After registering as a user, you will be required to login to use the application, below is an image of the login interface of our application.



gyateng94@gmail.com

•••••••••

FIGURE 4.9: USER ENTERING DETAIL TO LOGIN TO THE APPLICATION

4.2.2.3. Home page

Home page displays a line chart based on the soil moisture level of each node. This graph is interactive i.e. farmers can zoom in or out of the chart.

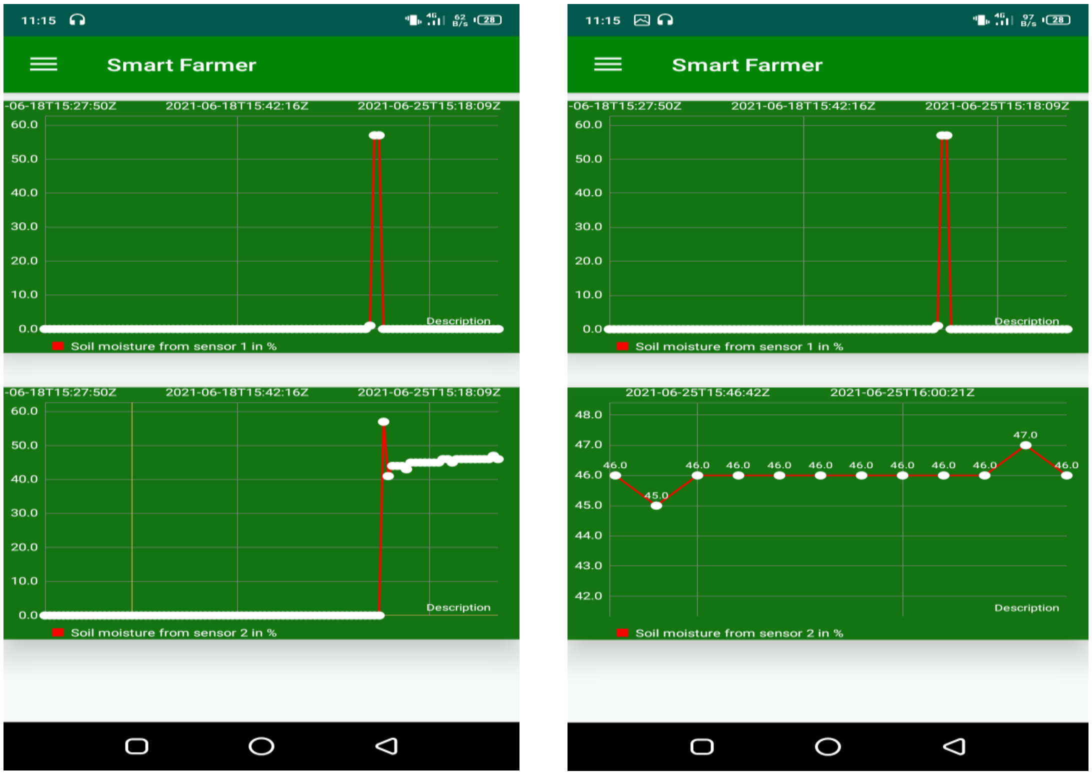


FIGURE 4.10: LINE GRAPH REPRESENTING SOIL MOISTURE DATA FROM SYSTEM

4.2.2.4. Side bar navigation showing various fragment that a user can navigate to.

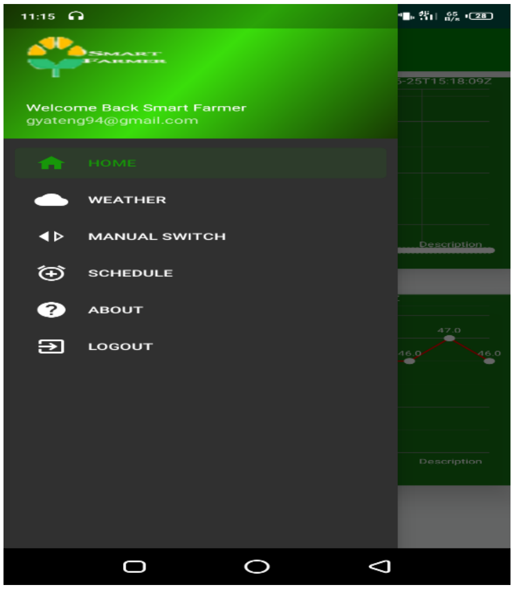
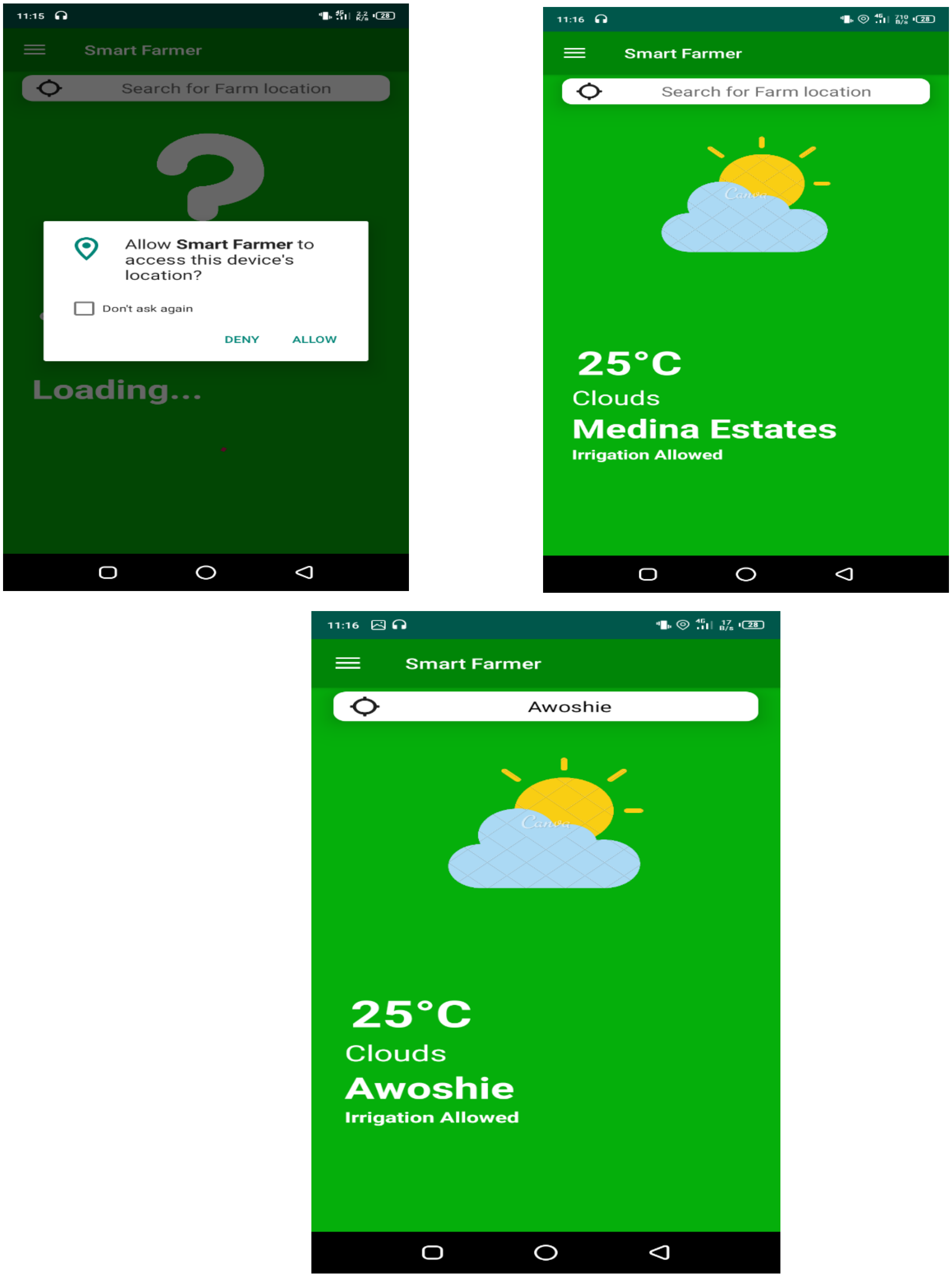


FIGURE 4.11: NAVIGATION DRAWER FOR USER TO SELECT WHAT TO DO

4.2.2.5. Weather page

This page first requests for permission to use the phone's location and also allows the user to search for locations of the farm in case they are not at the farm location. The current weather condition of the provided location is displayed afterwards.



11:15

Smart Farmer

Search for Farm location

Allow **Smart Farmer** to access this device's location?

□ Don't ask again

DENY ALLOW

Loading...

11:16

Smart Farmer

Search for Farm location

25°C

Clouds

Medina Estates

Irrigation Allowed

11:16

Smart Farmer

Awoshie

25°C

Clouds

Awoshie

Irrigation Allowed

FIGURE 4.12: USER'S LOCATION REQUESTED AND SEARCHING FOR FARM LOCATIONS

4.2.2.6. Manual switch

Here, the farmer can start and stop irrigation at a specific node instantly with the click of a button. When these buttons are clicked, values in the firebase real time database changes and the change signal is sent to the node MCU.

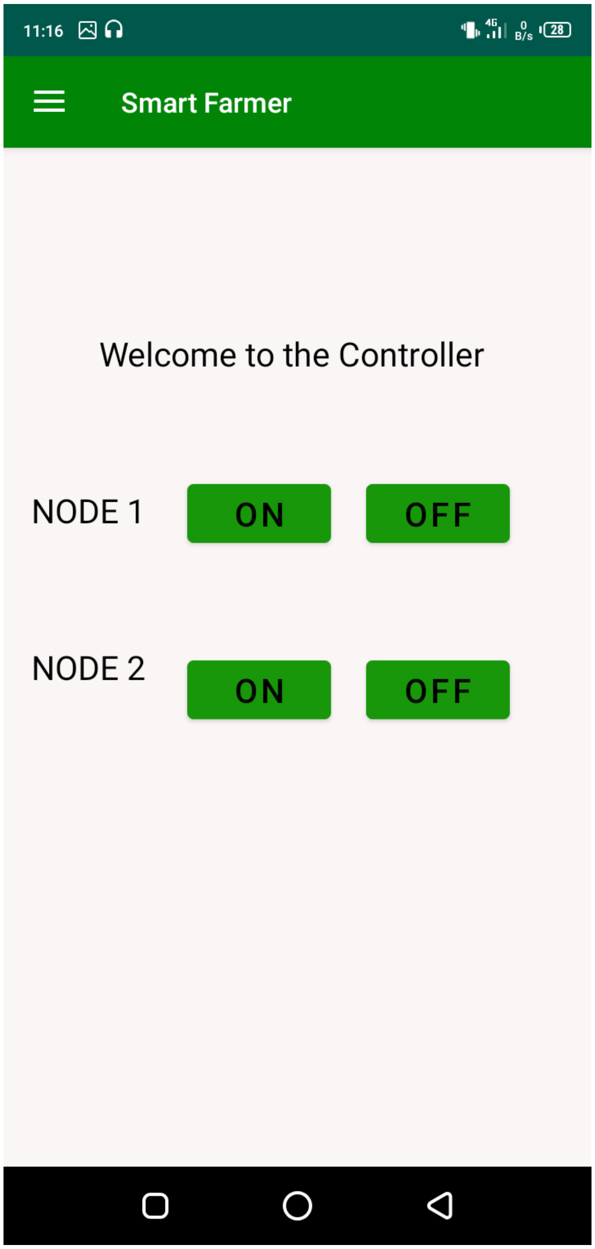
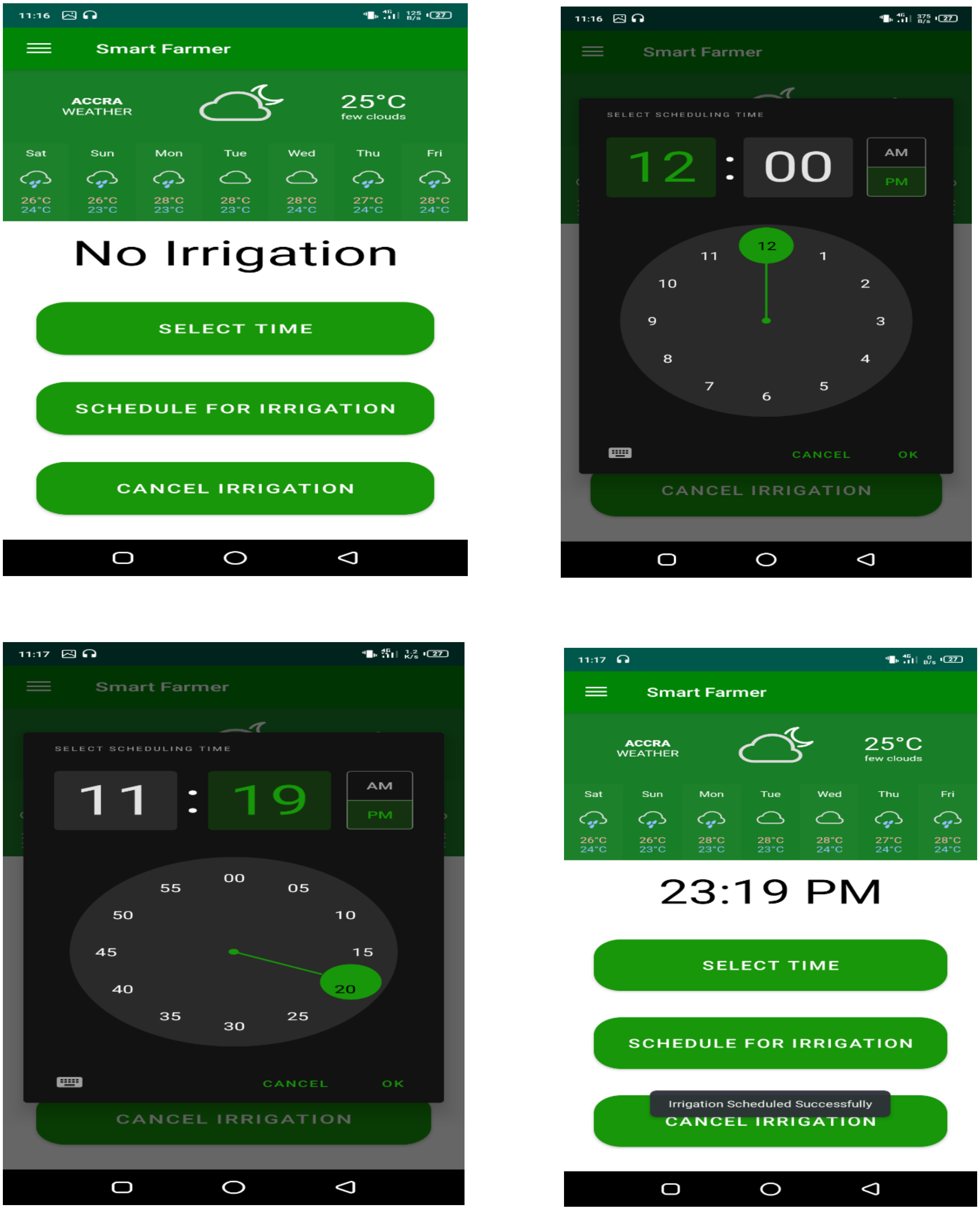


FIGURE 4.13: USER MANUALLY SWITCHING PUMPS ON/OFF

4.2.2.7. Schedule page

Here, the user can schedule for irrigation to occur at a time of choice. The farmer is notified once the irrigation time is up. When weather conditions are not favorable, irrigation does not start.

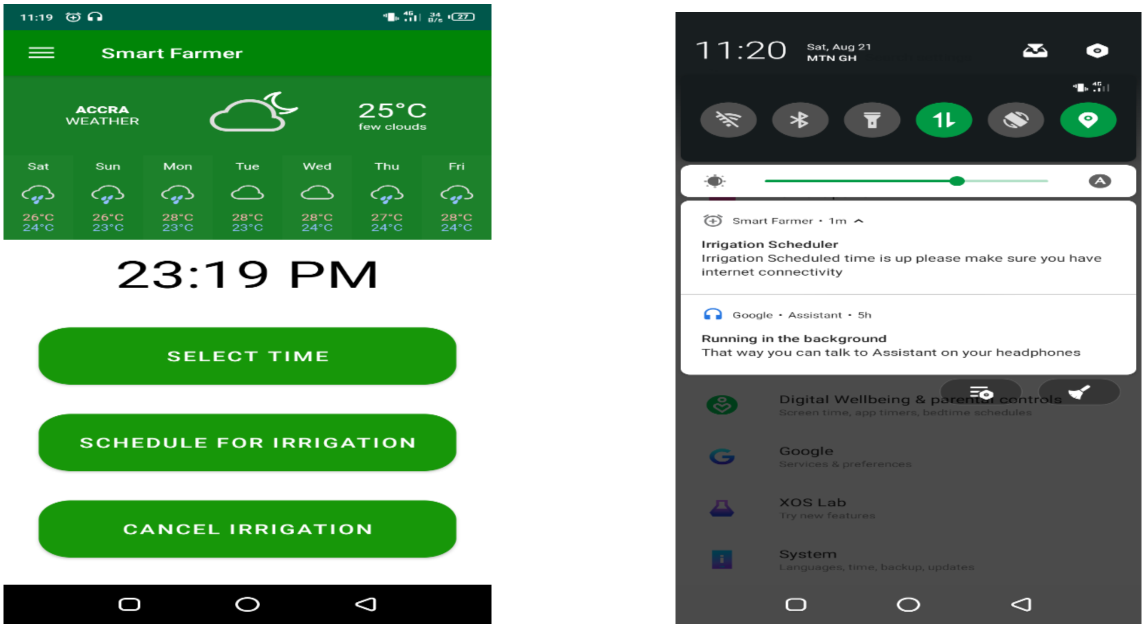


No Irrigation

23:19 PM

Irrigation Scheduled Successfully

FIGURE 4.14: USER SCHEDULING FOR IRRIGATION



Smart Farmer • 1m

Irrigation Scheduler

Irrigation Scheduled time is up please make sure you have internet connectivity

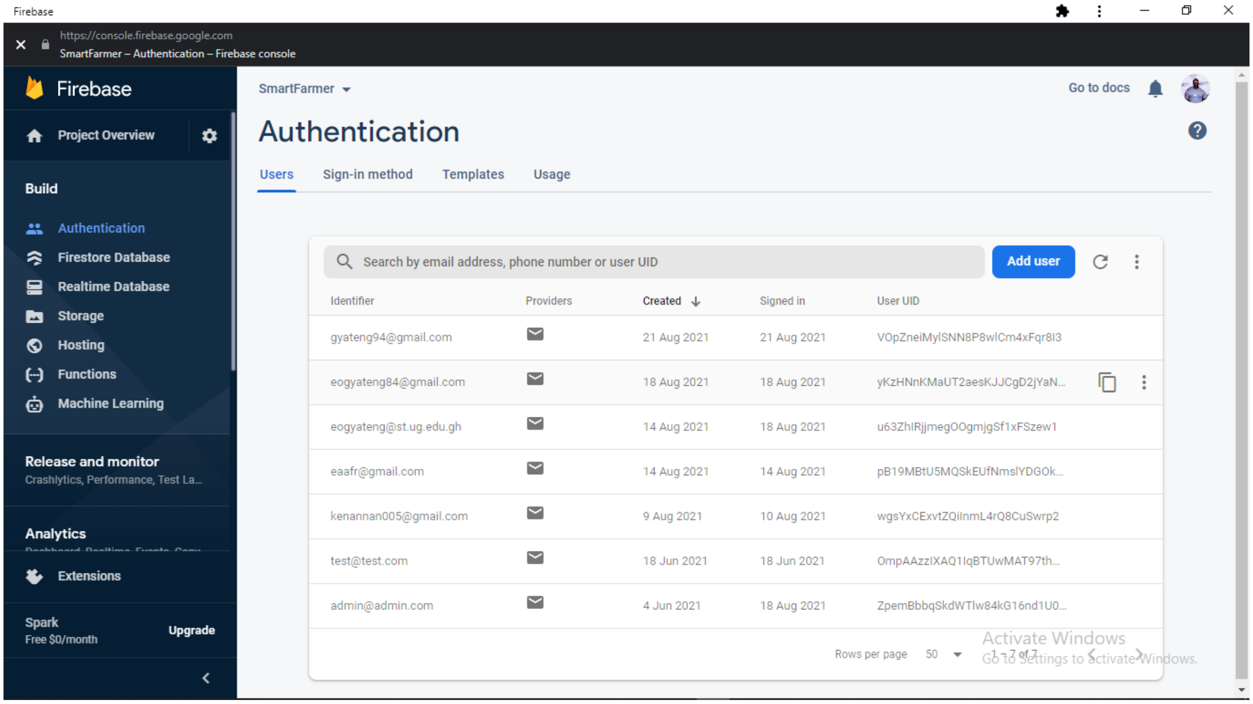
Google • Assistant • 5h

Running in the background

That way you can talk to Assistant on your headphones

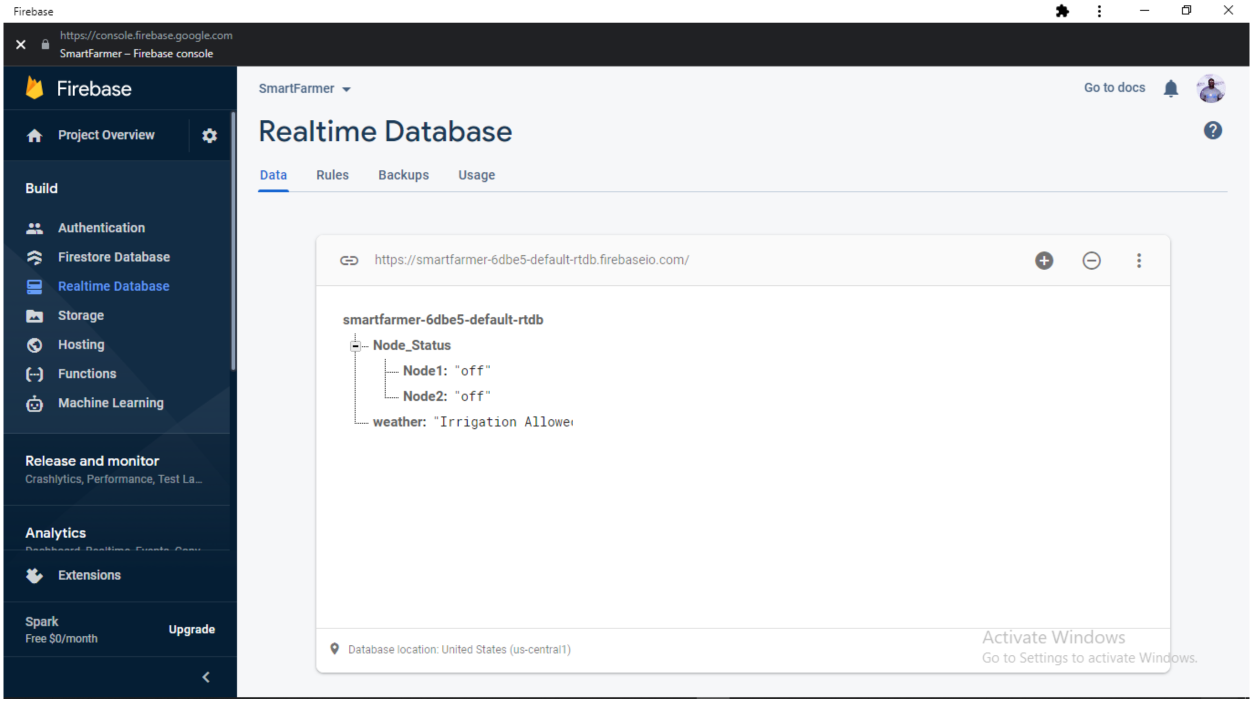
FIGURE 4.15: SCHEDULING TIME UP NOTIFICATION

4.2.2.8. Cloud storage and database



Identifier	Providers	Created	Signed in	User UID
gyateng94@gmail.com	✉	21 Aug 2021	21 Aug 2021	VOpZneiMylSNN8P8wlCm4xFqr8I3
eogyateng84@gmail.com	✉	18 Aug 2021	18 Aug 2021	yKzHNnKMaUT2aesKJJCgD2jYaN...
eogyateng@st.ug.edu.gh	✉	14 Aug 2021	18 Aug 2021	u63ZhIRjjmegOOgmjgSf1xFSzew1
eaafr@gmail.com	✉	14 Aug 2021	14 Aug 2021	pB19MBtU5MQSkEUfNmslYDGOk...
kenannan005@gmail.com	✉	9 Aug 2021	10 Aug 2021	wgsYxCExvtZQiInmL4rQ8CuSwrp2
test@test.com	✉	18 Jun 2021	18 Jun 2021	OmpAAzzIXAQ1IqBTUwMAT97th...
admin@admin.com	✉	4 Jun 2021	18 Aug 2021	ZpemBbbqSkdWTlw84kG16nd1U0...

FIGURE 4.16: USERS AUTHENTICATED BY FIREBASE USING THE APP

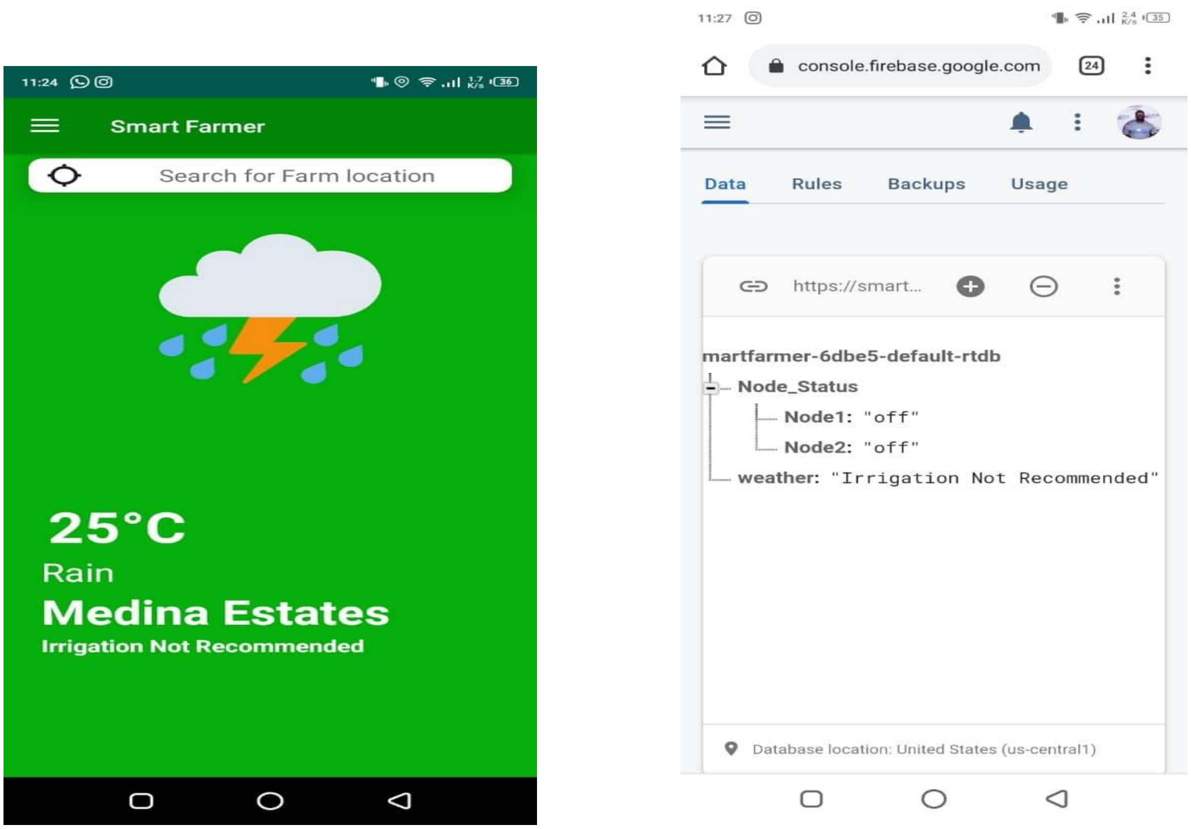


Firebase
https://console.firebase.google.com
SmartFarmer – Firebase console
Firebase
Project Overview
Build
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
SmartFarmer
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://smartfarmer-6dbe5-default-rtdb.firebaseio.com/
Database location: United States (us-central1)
Activate Windows
Go to Settings to activate Windows.

```
smartfarmer-6dbe5-default-rtdb
  Node_Status
    Node1: "off"
    Node2: "off"
  weather: "Irrigation Allowe
```

FIGURE 4.17: REAL TIME DATABASE SHOWING THE OF THE SYSTEM

4.2.2.9. Scheduling whilst raining



25°C

Rain

Medina Estates

Irrigation Not Recommended

```
martfarmer-6dbe5-default-rtdb
⊟ Node_Status
    Node1: "off"
    Node2: "off"
  weather: "Irrigation Not Recommended"
```

Database location: United States (us-central1)

FIGURE 4.18: NODES OFF BECAUSE WEATHER CONDITIONS ARE NOT FAVORABLE FOR IRRIGATION

4.3. Discussion of Results and Analysis

When you connect the system to a power source, the first thing it does is to try to connect to a specified Wi-Fi network in the code on the controller as seen in figure 4.1. The system then goes on to check the data from the firebase database. It then checks the soil moisture data from the sensors and if the sensor value is greater than 49%, it means the soil is dry and not good for the tomato plant so the system will start irrigation as seen in figure 4.2. If the value from the sensor is below 33%, the system will automatically stop the irrigation as seen in figure 4.3. When a scheduled time for irrigation is up or the user tries to turn on the system wirelessly using our application and the data from the soil is below 33%, irrigation will not start as seen in figure 4.4, but if the value is greater than 33%, the command will be effective. If the user tries to wirelessly turn off the system while the value from the sensor is greater than 49%, the command will not be effective and the system will stay on as seen in figure 4.5 but if the value is less than 49%, the system will stop when the command is received. Figure 4.6 shows an arrangement of the hardware components on a bread board. Figure 4.7 shows how the system could be deployed in farm.

For our android application, figures 4.8 and 4.9 show how our system registers a user with requirements being their email and also a password. The user after registering is also made to confirm their email through a link sent to the provided email, the user can now sign in to use the application using the email with which they used to register and also the password they provided.

When you successfully log in to our system, you first see a graph which is generated using the sensor values sent to thingspeak from the node MCU as seen in figure 4.10. Figure 4.11 shows a navigation drawer where user can select a fragment to navigate to. Figure 4.12 shows content of the second fragment, here a user has to first allow the application to access their phones location and after that, they can enter their farm's location in case they are not on the farm.

In figure 4.13, the fragment where a user can wirelessly start or stop irrigation instantly using buttons provided is shown. A user can also schedule for irrigation at a preferred time of their own, this is shown in figure 4.14. In both the instant control and scheduling of irrigation, the application with the help of an API will check if it is raining or the chances of raining is high, if so, the system will not start irrigation and the user will be alerted. The system alerts the user when the time scheduled for irrigation is due as shown if figure 4.15. Figure 4.16 and 4.17 shows a firebase database where user details are stored and the real time database where commands are stored for the node MCU to pick up and operate based on it. Figure 4.18 shows and instance where irrigation was scheduled and on the due time it was raining, you can see that instead of sending an on command for the user, the application sent and off command rather to make sure the system does not start or will go off in case it is already on.

4.4. Performance Evaluation and Limitation of the System

4.4.1. Performance Evaluation

The table below shows a comparison between our system and other existing ones based on our main objectives.

Table 4.1: Performance Evaluation

SYSTEMS	AUTOMATED IRRIGATION	SCHEDULING OF IRRIGATION	INSTANTANEOUS CONTROL OF IRRGATION	MONITORING OF SYSTEM	USE OF DRIP IRRIGATION
Our system	Yes	Yes	Yes	Yes	Yes
Existing system [1]	Yes	No	No	Yes	No
Existing system [2]	Yes	No	No	Yes	No
Existing system [3]	Yes	No	No	Yes	No
Existing system [4]	Yes	No	No	Yes	Yes
Existing system [5]	Yes	No	No	Yes	No
Existing system [6]	Yes	No	No	Yes	No
Existing system [8]	Yes	No	Yes	Yes	No

4.4.1.1 Performance Metric Evaluation of Hardware

TABLE 4.2: PERFORMANCE METRIC EVALUATION OF HARDWARE

SYSTEM	TESTING	EXPECTATION	OBSERVATION
Soil moisture sensor	The sensor was tested using a very dry soil, a wet soil, water and air to determine the various values to make a range for automatic irrigation of system	Detect the sensor values for all the instances	We realized that with this sensor, data represented how dry the soil is and then, higher values meant the soil is dry and smaller values also shows that the soil is wet
final hardware	Check how the system responds to dry and wet soil and also what happens when a command comes from the application.	System should turn on when soil is very dry and also turn off when soil is wet. Also system should stay off when command comes for it to turn on while soil is wet and go on when soil is considered not wet, also, it should stay on when command comes in for it to go off while very dry and also off when soil is not dry	System turns on when soil gets dry and also turns off when soil is wet. When there is a command to turn on the system, it goes on when the soil is not considered wet and stays off if soil is wet, when the command is to turn off the system, it goes off when soil is not considered dry but will stay on when dry. We realized that when theinternet connection to the node MCU was poor, the system had a problem with turning on or off immediately when command is sent from the application but it works fine when the internet is restored.
Android application	Check if the graph shows and it is a true representation of the data from the sensor, turn on and off the hardware system at a	Application should have a graph representing data from the sensor, schedule for irrigation and instant	We realized that, the graph produced was an actual representation of the data from the sensor. Also, when the button

	moment using the application and also schedule for irrigation for anytime.	control of irrigation should work only when there is no rain or immediate possibility of rain	to start irrigation was clicked, the system followed through only when there was no rain, also, one could schedule for irrigation at any time and irrigation will start when there is no rain or possible rain when the scheduled time is up.

4.4.2. Limitation Of System

The main limitation of the system is that, it requires internet for its operation and will not operate without it. In places where the internet connection is poor, the system will work very poorly and will not respond well to commands from the application.

CHAPTER 5 –CONCLUSION AND RECOMMENDATION

5.0. **Introduction**

Smart drip irrigation system for tomato farming will change the face of the tomato industry in Ghana by solving the problem of less efficient and less effective way of irrigation. Therefore increasing the yield of tomatoes in the country and reduce water wastage across tomato farms.

5.1. **Conclusion**

The project objectives were achieved when the system was tested. The results from the testing we favorable. The system will allow tomato farmers to monitor and schedule irrigation when they see fit using an android application, while automating irrigation.

5.2. **Recommendations**

We recommend that all future exploration of this project be geared towards making the life of the farmer easier. Future systems can implemented for other plants other than tomato plants.

5.3. **Observation**

Internet connectivity is needed to utilize the full capacity of the drip irrigation system. Without internet connectivity, little can be done with the system. Users would have to enable their location on their mobile device or search for location of farm to enable the system make informed decision.

REFERENCES

[1] S. Yamba, "Smallholder Farmers' Livelihood Security Options amidst Climate Variability and Change in Rural Ghana," *Scientifica,* vol. 2017, p. 10, 2017.

[2] Mofa-IFPRI, "Ghana's Tomato Market," *MoFA-IFPRI Market Brief,* vol. I, no. 2, p. 4, 2020.

[3] B. T. Anang, "Production constraints and measures to enhance the competitiveness of the tomato industry in Wenchi Municipal District of Ghana.," *American Journal of Experimental Agriculture,* vol. 3, no. 4, pp. 824-838, 2013.

[4] J. V. Asselt, I. Masias and S. Kolavalli, "Competitiveness of the Ghanaian Vegetable Sector," International Food Policy Research Institute, Accra, 2018.

[5] H. M. Bortey and A. S. Osuman, "Analysing The Constraints Faced By The Small Holder Tomato Growers In Ghana," *Intetrnational Journal of Agricultural Extension,* vol. 4, no. 2, pp. 111-117, 2016.

[6] E. J. Robinson and S. L. Kalavalli, "The case of Tomato in Ghana: Productivity," *GSSP Working Paper,* vol. 19, p. 25, 2010.

[7] S. e. a. Sankaranarayanan, "Intelligent IoT Based Automation Irrigation System," *International Journal of applied Engineering Research,* vol. 12, no. 18, 2017.

[8] S. Rawal, "IoT based smart irrigation system," *International Journal of Computer Application,* vol. 159, no. 8, 2017.

[9] Md. M. Islam et al, "IoT Based Smart irrigation monitoring & Controlling system in Agriculture," *International Journal of recent Technology and Engineering (IJRTE),* vol. 8, no. 6, 2020.

[10] T. Gaikwad et al, "Smart Drip Irrigation System using IoT," *International Journal of Engineering and Technology (IRJET),* vol. 5, no. 10, 2018.

[11] Dr. S. J. Muneeswaril, "Smart Irrigation System using IoT approach," *International Journal of Engineering Research & Science (IJOER) ,* vol. 3, no. 3, 2017.

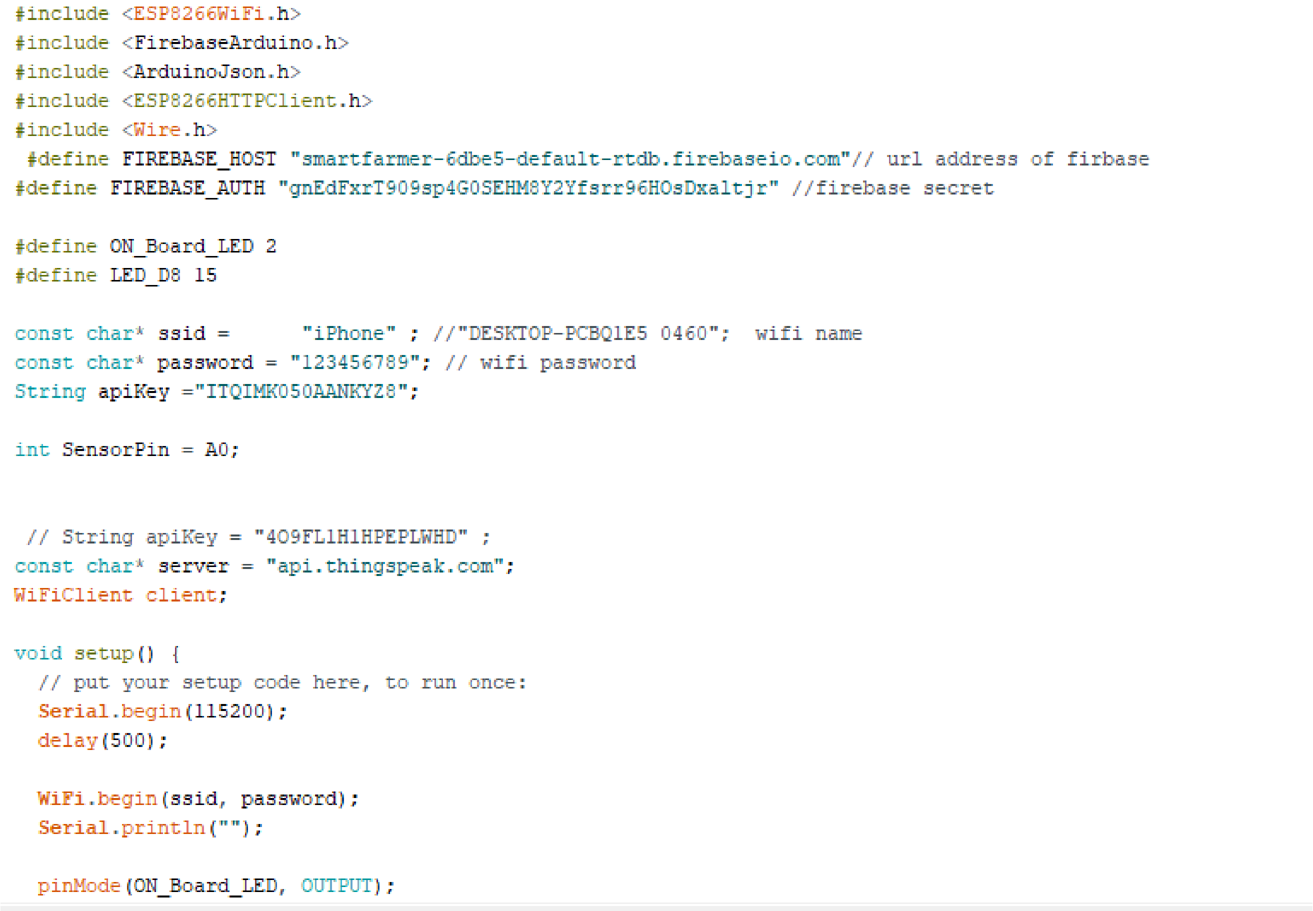
[12] B. I. Agustinus et al, "Automatic watering Device for Toamato using soil moisture sensor," *The first International Conference and exhibition on science and Technology (ICEST),* 2018.

[13] M. F.L., "Design of an automated Irrigation system: Student Paper," McGill University, Montreal, QC, Canada, 2005.

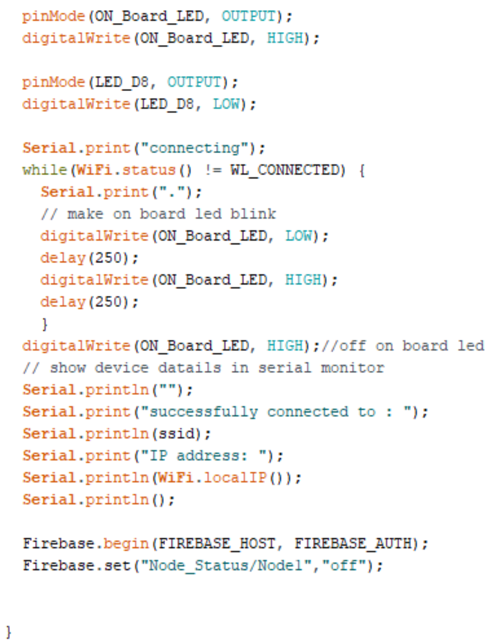
[14] M. Senapati, "Wireless Smart Irrigation System," *International Journal of Engineering Research & Technology (IJERT),* vol. 8, no. 1, 2020.

APPENDICES

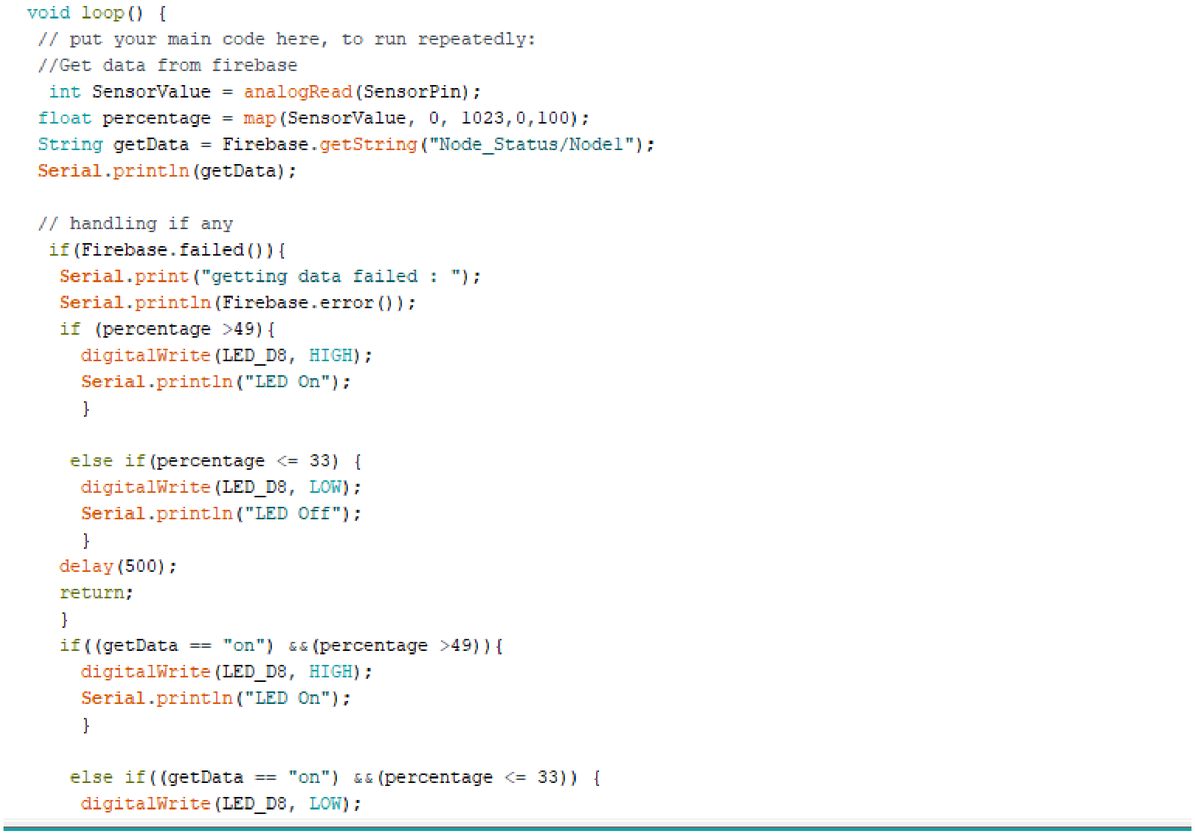
Appendix A: Code Snippet for the Hardware System



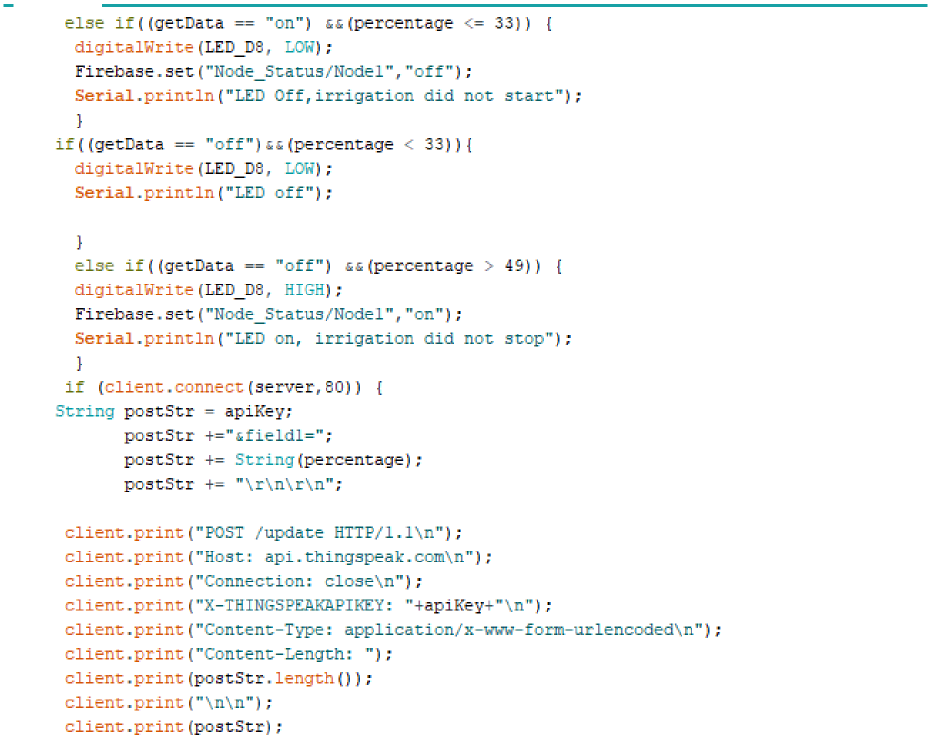
```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <Wire.h>
 #define FIREBASE_HOST "smartfarmer-6dbe5-default-rtdb.firebaseio.com"// url address of firbase
#define FIREBASE_AUTH "gnEdFxrT909sp4G0SEHM8Y2Yfsrr96HOsDxaltjr" //firebase secret

#define ON_Board_LED 2
#define LED_D8 15

const char* ssid =      "iPhone" ; //"DESKTOP-PCBQ1E5 0460";  wifi name
const char* password = "123456789"; // wifi password
String apiKey ="ITQIMK050AANKYZ8";

int SensorPin = A0;


 // String apiKey = "4O9FL1H1HPEPLWHD" ;
const char* server = "api.thingspeak.com";
WiFiClient client;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  delay(500);

  WiFi.begin(ssid, password);
  Serial.println("");

  pinMode(ON_Board_LED, OUTPUT);
```



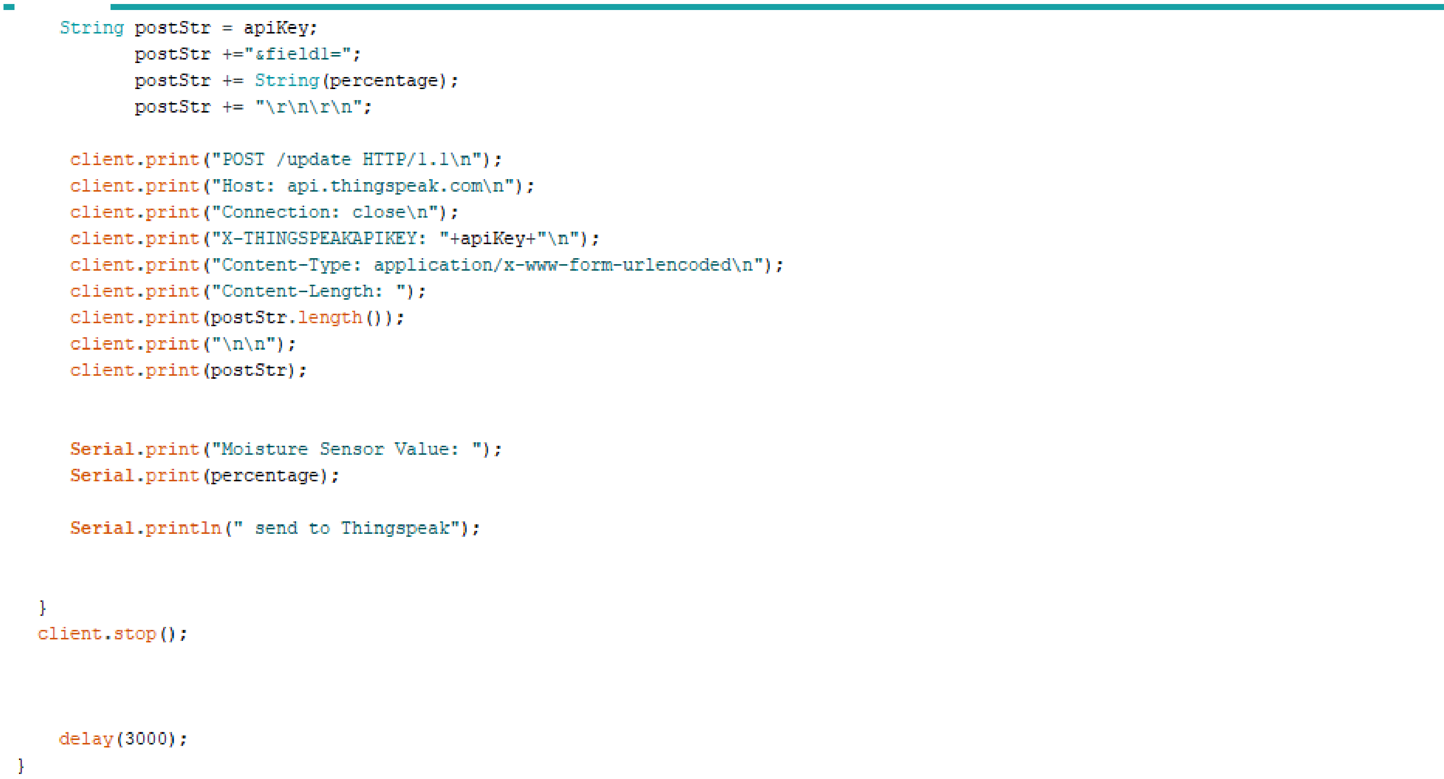
```
  pinMode(ON_Board_LED, OUTPUT);
  digitalWrite(ON_Board_LED, HIGH);

  pinMode(LED_D8, OUTPUT);
  digitalWrite(LED_D8, LOW);

  Serial.print("connecting");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    // make on board led blink
    digitalWrite(ON_Board_LED, LOW);
    delay(250);
    digitalWrite(ON_Board_LED, HIGH);
    delay(250);
    }
  digitalWrite(ON_Board_LED, HIGH);//off on board led
  // show device datails in serial monitor
  Serial.println("");
  Serial.print("successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.set("Node_Status/Node1","off");


}
```



```
void loop() {
 // put your main code here, to run repeatedly:
 //Get data from firebase
  int SensorValue = analogRead(SensorPin);
 float percentage = map(SensorValue, 0, 1023,0,100);
 String getData = Firebase.getString("Node_Status/Node1");
 Serial.println(getData);

 // handling if any
  if(Firebase.failed()){
   Serial.print("getting data failed : ");
   Serial.println(Firebase.error());
   if (percentage >49){
     digitalWrite(LED_D8, HIGH);
     Serial.println("LED On");
     }

    else if(percentage <= 33) {
     digitalWrite(LED_D8, LOW);
     Serial.println("LED Off");
     }
   delay(500);
   return;
   }
   if((getData == "on") &&(percentage >49)){
     digitalWrite(LED_D8, HIGH);
     Serial.println("LED On");
     }

    else if((getData == "on") &&(percentage <= 33)) {
     digitalWrite(LED_D8, LOW);
```



```
  else if((getData == "on") &&(percentage <= 33)) {
   digitalWrite(LED_D8, LOW);
   Firebase.set("Node_Status/Node1","off");
   Serial.println("LED Off,irrigation did not start");
   }
 if((getData == "off")&&(percentage < 33)){
   digitalWrite(LED_D8, LOW);
   Serial.println("LED off");

   }
   else if((getData == "off") &&(percentage > 49)) {
   digitalWrite(LED_D8, HIGH);
   Firebase.set("Node_Status/Node1","on");
   Serial.println("LED on, irrigation did not stop");
   }
  if (client.connect(server,80)) {
 String postStr = apiKey;
        postStr +="&field1=";
        postStr += String(percentage);
        postStr += "\r\n\r\n";

  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n");
  client.print(postStr);
```



```
  String postStr = apiKey;
         postStr +="&field1=";
         postStr += String(percentage);
         postStr += "\r\n\r\n";

   client.print("POST /update HTTP/1.1\n");
   client.print("Host: api.thingspeak.com\n");
   client.print("Connection: close\n");
   client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
   client.print("Content-Type: application/x-www-form-urlencoded\n");
   client.print("Content-Length: ");
   client.print(postStr.length());
   client.print("\n\n");
   client.print(postStr);


   Serial.print("Moisture Sensor Value: ");
   Serial.print(percentage);

   Serial.println(" send to Thingspeak");


 }
 client.stop();



   delay(3000);
}
```