

SMART GARDENING SYSTEM



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

SMART GARDENING SYSTEM

HALIM MOHSIN BIN MOHAMMAH FUZIA



This report is submitted in partial fulfillment of the requirements for the Bachelor of [Computer Science (Computer Networking)] with Honours.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[YEAR OF SUBMISSION]

DECLARATION

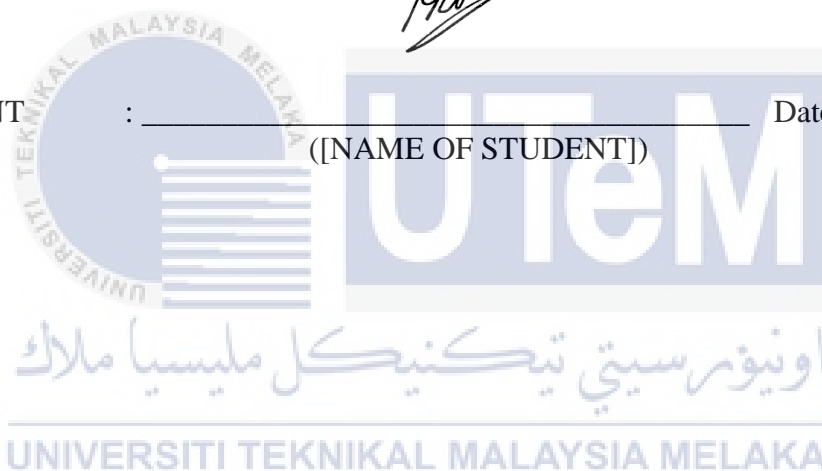
I hereby declare that this project report entitled

[TITLE OF FINAL YEAR PROJECT]

is written by me and is my own effort and that no part has been plagiarized
without citations.



STUDENT : _____ Date : _____
([NAME OF STUDENT])



I hereby declare that I have read this project report and found
this project report is sufficient in term of the scope and quality for the award of
Bachelor of [Computer Science (Software Development)] with Honours.

SUPERVISOR : _____ Date : _____
([NAME OF THE SUPERVISOR])

DEDICATION

This project is dedicated to my beloved family who has consistently been other for me at whatever point I need and gives me a strong support that consistently encompasses me all the time. Moreover, this task likewise committed to my supervisor, Ts. Irda Binti Roslan who has guided me with extraordinary consideration and persuaded me to set a higher objective to produce a better project. To wrap things up, I devoted this project to my friends for the help of any support they have provided to me when I face the problem.



ACKNOWLEDGEMENTS

First of all, all the praise to Allah S.W.T for giving me the opportunities, strength, and patience throughout the course of the project. Without a blessing from Him, I cannot complete this project according to what has been arranged off.

Secondly, my most gratitude thanks to my beloved parents for their nonstop prayers and also give everything to ensure I succeed in this project. Their support made me never give up and have given my best shot to finish the project which has been started.

Next, I would like to give millions thanks to my supervisor Ts. Irda Binti Roslan for guiding me throughout the course of the project to complete this Final Year Project. She always provided me with precious knowledge, encouragement, and advice which brings me to develop better project.

Furthermore, I would like to thanks to all my friends for their time, concern, efforts which always encouraging me during complete this project. With all the help from involves parties, I manage to finish this PSM with all the help, I manage to complete my report and my project

Finally, thanks to Universiti Teknikal Malaysia Melaka (UTeM) for the opportunity given and providing me with a complete resources and information needed.

ABSTRACT

In today's digital environment, people expect automation to make tasks easier, more comfortable, faster, and more efficient. The goal is to transform our current system for supplying water in home gardens, farms, and fields into a Smart Automated System. In this project will describe about Smart Gardening System. This system, there are three sensors have been used which soil moisture sensor, soil temperature sensor and temperature and humidity sensor. When the sensor detect low soil moisture, the water pump will sprinkle the plants automatically. Facilities that provided were user friendly and make easier to users for growth of plant. The main problem that facing by a gardeners tend to consume large amounts of water for plantation as they do not know the sufficient amount of water for watering the plants. This problem will give impact in term of shortage of water resources and over-watering the plant or lack of water for plantation. Gardeners also having difficulties in maintain the quality of the soil which one of main important in maintaining the health of a plant. By having this gardening system a gardener or farmer can monitor the soil humidity. The gardeners also having difficulties in learning the growth and condition of plant. This system is helpful for farmer to learn the condition of the plant and earn more knowledge by monitor the data of the plant. Besides that, this gardening system can be used at any plants because it's a portable device and it provide much more precise data. This gardening system not just offer user to monitor the information from plantation but it also provide opportunity for user to have better understanding on plant growth.

ABSTRAK

Dalam persekitaran digital masa kini, orang mengharapkan automasi membuat tugas lebih mudah, lebih selesa, lebih pantas, dan lebih cekap. Tujuannya adalah untuk mengubah sistem kita sekarang untuk membekalkan air di kebun rumah, ladang, dan ladang menjadi Sistem Automatik Pintar. Dalam projek ini akan menerangkan mengenai Sistem Berkebun Pintar. Sistem ini, ada tiga sensor yang telah digunakan yaitu sensor kelembapan tanah, sensor suhu tanah dan sensor suhu dan kelembapan. Apabila sensor mengesan kelembapan tanah yang rendah, pam air akan menaburkan tanaman secara automatik. Kemudahan yang disediakan adalah mesra pengguna dan memudahkan pengguna untuk pertumbuhan tanaman. Masalah utama yang dihadapi oleh tukang kebun cenderung memakan sejumlah besar air untuk perladangan kerana mereka tidak mengetahui jumlah air yang mencukupi untuk menyiram tanaman. Masalah ini akan memberi kesan dari segi kekurangan sumber air dan penyiraman tanaman yang berlebihan atau kekurangan air untuk perladangan. Tukang kebun juga menghadapi kesukaran untuk menjaga kualiti tanah yang salah satu yang penting dalam menjaga kesihatan tanaman. Dengan adanya sistem berkebun ini, tukang kebun atau petani dapat memantau kelembapan tanah. Tukang kebun juga mengalami kesukaran dalam mempelajari pertumbuhan dan keadaan tanaman. Sistem ini berguna bagi petani untuk mengetahui keadaan tanaman dan memperoleh lebih banyak pengetahuan dengan memantau data tanaman. Selain itu, sistem berkebun ini dapat digunakan di mana-mana tanaman kerana ia adalah alat mudah alih dan menyediakan data yang lebih tepat. Sistem berkebun ini tidak hanya menawarkan pengguna untuk memantau maklumat dari perkebunan tetapi juga memberi peluang kepada pengguna untuk memiliki pemahaman yang lebih baik mengenai pertumbuhan tanaman.

TABLE OF CONTENTS

DECLARATION.....	II
DEDICATION.....	III
ACKNOWLEDGEMENTS.....	IV
ABSTRACT	V
ABSTRAK	VI
LIST OF ABBREVIATIONS	XIV
CHAPTER 1: INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Project Background.....	2
1.3 Problem Statement.....	2
1.4 Project Question.....	3
1.5 Objective	3
1.6 Scope.....	3
1.7 Expected Output.....	4
1.8 Conclusion	4
CHAPTER 2: LITERATURE REVIEW.....	5
2.1 Introduction.....	5
2.2 Related Work	5

2.2.1	Agriculture	5
2.3	Related project	6
2.3.1	Monitoring Moisture of Soil.....	6
2.3.2	Smart Home Agriculture System.....	9
2.3.3	Smart Home Garden Irrigation System Using Raspberry Pi by S.N. Ishak.....	12
2.3.4	Comparison.....	16
2.4	Proposed solution.....	16
2.5	Conclusion	17
CHAPTER 3: ANALYSIS.....		18
3.1	Introduction.....	18
3.2	Prototype Model.....	18
3.2.1	Requirement Analysis Phase	18
3.2.2	Quick Design Phase	18
3.2.3	Build a Prototype Phase.....	19
3.2.4	Initial User Evaluation Phase.....	19
3.2.5	Refining Prototype Phase	19
3.2.6	Implement and Maintenance Phase	19
3.3	Project Milestones.....	20
3.4	Conclusion	22
CHAPTER 4: ANALYSIS AND DESIGN.....		23
4.1	Introduction.....	23
4.2	Problem Analysis	23
4.3	Requirement Analysis.....	23
4.3.1	Data Requirement	24

4.3.2	Functional Requirement.....	25
4.3.3	Software Requirement	25
4.3.4	Hardware Requirement.....	27
4.4	High-Level Design.....	35
4.4.1	System architecture.....	36
4.4.2	Database Design	38
4.5	Conclusion	39
CHAPTER 5: IMPLEMENTATION.....		40
5.1	Introduction.....	40
5.2	Development Environment Setup	40
5.2.1	Hardware Development Setup.....	40
5.2.2	Software Development Setup	41
5.3	Hardware Configuration Management	42
5.3.1	Configuration Environment Setup.....	42
5.3.2	Firebase Setup.....	47
5.3.3	ThingSpeak Setup.....	48
5.4	Implementation Status	50
5.5	Conclusion	50
CHAPTER 6: TESTING		51
6.1	Introduction.....	51
6.2	Test Plan.....	51
6.2.1	Test Environment.....	51
6.2.2	Test Schedule.....	52
6.3	Test Strategy	52

6.3.1	Classes of Tests.....	53
6.4	Test Design	53
6.4.1	Test Description.....	53
6.5	Test Result and Analysis.....	58
6.5.1	NodeMCU Board Test.....	58
6.5.2	OLED Screen Test.....	58
6.5.3	Air Temperature and Humidity Sensor (DHT22) Test.....	59
6.5.4	Soil Moisture Sensor Test.....	59
6.5.5	Soil Temperature Sensor Test.....	60
6.5.6	Water Pump Test	60
6.5.7	Firestore Test.....	61
6.5.8	ThingSpeak Test.....	62
6.6	Conclusion	62
CHAPTER 7: PROJECT CONCLUSION		63
7.1	Introduction.....	63
7.2	Project Summarization.....	63
7.3	Project Contribution.....	64
7.4	Project Limitation	64
7.5	Future Works	64
7.6	Conclusion	65
REFERENCES.....		66
APPENDIX A		67

LIST OF FIGURES

Figure 2.1: Setup for measuring soil moisture of a sample	7
Figure 2.1: Setup for measuring soil moisture of a sample	7
Figure 2.2: Result of dry soil	7
Figure 2.3: Result of medium moisture soil	8
Figure 2.4: Result of wet soil	8
Figure 2.5: Flowchart of System	9
Figure 2.6: Prototype of the system	10
Figure 2.7: Result of sensor value in Blynk	11
Figure 2.8: Results for the Soil moisture.....	11
Figure 2.9: Prototype of Smart Home Garden Irrigation System.....	12
Figure 2.10: Android Apps Interfaces.....	13
Figure 2.11: Email Configuration for Normal Operation.....	14
Figure 2.12: Raspberry Pi terminal for Normal Operation.....	14
Figure 2.13: Email Configuration for Critical Operation.....	15
Figure 2.14: Raspberry Pi terminal for Critical Operation.....	15
Figure 4.1: Data flow for system.....	24
Figure 4.2: Functional Diagram	25
Figure 4.3: Arduino IDE Interface	26
Figure 4.4: Blynk Mobile Application	27
Figure 4.5: NodeMCU ESP8266	28
Figure 4.6: 2 Channel 5V Relay Module.....	29
Figure 4.7: Soil Moisture Sensor.....	30
Figure 4.8: DHT22 Sensor	31
Figure 4.9: DS18B20 Soil Temperature Sensor.....	32
Figure 4.10: Jumper Wire	33
Figure 4.11: Breadboard	34
Figure 4.12: OLED Display Module.....	35

Figure 4.13: Model Design	35
Figure 4.14: System Architecture	37
Figure 4.15: ERD diagram	38
Figure 5.1: NodeMCU Pinout	41
Figure 5.2: System Development	41
Figure 5.3: NodeMCU Code	42
Figure 5.4: OLED Screen Code	43
Figure 5.5: DHT22 Code	44
Figure 5.6: Soil Moisture Code	44
Figure 5.7: Soil Temperature Code	44
Figure 5.8: Water Pump Code	45
Figure 5.9: Blynk Code	45
Figure 5.10: Firebase Code	46
Figure 5.11: ThingSpeak Code	46
Figure 5.12: Firebase website	47
Figure 5.13: Add Project	48
Figure 5.14: New Channel	48
Figure 5.15: Channel Stats	49
Figure 5.16: API Key	49
Figure 6.1: NodeMCU Code	58
Figure 6.2: OLED Screen	59
Figure 6.3: DHT22 Sensor	59
Figure 6.4: Soil Moisture Sensor	60
Figure 6.5: Soil Temperature Sensor	60
Figure 6.6: Water Pump	61
Figure 6.7: Firebase	61
Figure 6.8: ThingSpeak	62

LIST OF TABLES

Table 2.1: Comparison between Three Related Projects	16
Table 3.1: Milestone of Project Activities	20
Table 3.2: Gantt chart of Project Activities.....	21
Table 4.1: Data dictionary	38
Table 5.1: Implementation Status.....	50
Table 6.1: Test Schedule	52
Table 6.2: Test Classes	53
Table 6.3: NodeMCU Test.....	54
Table 6.4: DHT22 Sensor Test	54
Table 6.5: Soil Moisture Sensor Test.....	55
Table 6.6: Soil Temperature Sensor Test.....	55
Table 6.7: Wi-Fi Test	56
Table 6.8: Water Pump Test	56
Table 6.9: Firebase Test.....	57
Table 6.10: ThingSpeak Test.....	57

LIST OF ABBREVIATIONS

FYP - Final Year Project



CHAPTER 1: INTRODUCTION

1.1 Introduction

Smart gardening system is a system that control plantation automatically by computers and electronic devices. The system also attached with soil sensor to monitor the moisture and temperature of the soil, meanwhile the system also contain air sensor to monitor the local air temperature and humidity. The gardening system contain soil sensor to monitor the soil humidity and temperature time to time to maintain the growth of plant. The gardeners also get precise data by monitor the plantation.

This system is designed to resolve the gardener their problems. The first problem that faced by gardeners are they don't know when and how much quantity of water that should watering the plants. Some of farmers consume large amounts of water for plantation which will lead to shortage of water resources. To avoid the over-watering and lack of water for the plantation this gardening system will be helpful. Next problem that faced by farmers is maintain the quality of the soil for maintain the growth of the plants. The soil need to be in good humidity and temperature to maintain the plants growth. The gardening system contain soil sensor to monitor the soil humidity and temperature time to time to maintain the growth of plant. The farmers also get precise data by monitor the plantation.

The objective of this project is to develop smart gardening system using multiple sensors. Second objective is to monitor and analyses humidity and temperature of plant and soil. The last objective is to display data to user about

humidity and temperature of plant and soil. The expected output for this project is to successfully develop the gardening system which using NodeMCU and Blynk for the user to monitor air temperature and humidity and soil temperature and humidity.

1.2 Project Background

Smart gardening system is a system that controlled automatically. Although gardening already have systems and electronic devices, the gardening system with NodeMCU and Blynk contain multiple function which is monitoring and plant watering system using advance sensors, monitor the soil moisture and temperature and monitor local air temperature and humidity. The main problem that facing by a gardeners tend to consume large amounts of water for plantation as they do not know the sufficient amount of water for watering the plants. This problem will give impact in term of shortage of water resources and over-watering the plant or lack of water for plantation. Gardeners also having difficulties in maintain the quality of the soil which one of main important in maintaining the health of a plant. By having this gardening system a gardener or farmer can monitor the soil humidity. The gardeners also having difficulties in learning the growth and condition of plant. This system is helpful for farmer to learn the condition of the plant and earn more knowledge by monitor the data of the plant.

Besides that, this gardening system can be used at any plants because it's a portable device and it provide much more precise data. This gardening system not just offer user to monitor the information from plantation but it also provide opportunity for user to have better understanding on plant growth.

1.3 Problem Statement

The problem statement of this project are:

1. Gardeners tend to consume large amounts of water for plantation as they do not know the sufficient amount of water for watering the plants.
2. Gardeners can't keep an eye on their plants because have no time to monitor in daily time.

3. Gardeners also having difficulties in learning the growth and condition of plant.

1.4 Project Question

1. How much quantity of water needed for watering the plant using Smart Gardening System?
2. How to monitor the plant using Smart Gardening System?
3. How to validate the result of the plant condition using Smart Gardening System?

1.5 Objective

The objective of this project are:

1. To develop a smart gardening system using multiple sensor.
2. To monitor and analyses humidity and temperature of plant and soil.
3. To display data of humidity and temperature of plant and soil to user.

1.6 Scope

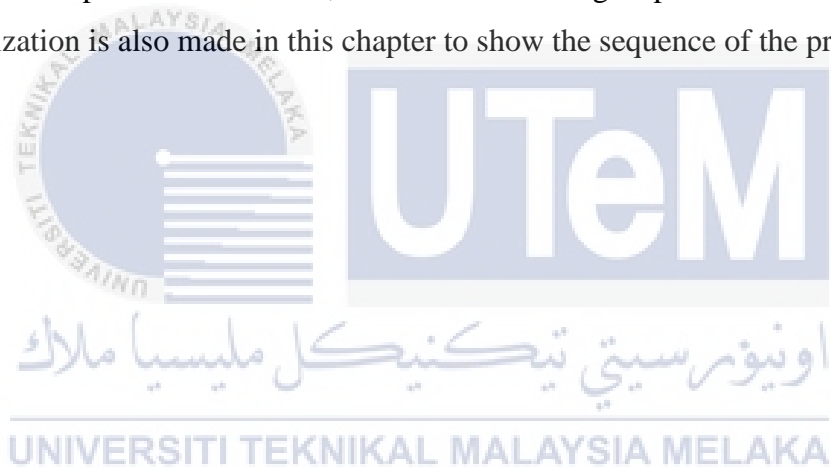
The targeted user for this project is mostly focused on farmers or gardener. The user does not need any basic knowledge about the way to handle the growth of a plant because the user can get information regarding to the condition of the plant by monitoring the plants using the gardening system. The gardening system consists of different type of sensors where user can get information from the sensors based on the plantation according to the information they need which soil sensor use for monitor the soil moisture and temperature and air sensor use for monitor local air temperature and humidity.

1.7 Expected Output

The expected outcome for this project is to successfully develop the smart gardening system using NodeMCU for user to monitor air temperature, air relative humidity, soil temperature and soil moisture. The gardening system also water the plant automatically. This gardening system able to use to any plants.

1.8 Conclusion

For conclusion, the implementation of a smart gardening system using other Internet of Things to make sure user can monitor their plant easily. The problem statement, research objectives and scope within this chapter is for identifying and discuss the problem statement, as well as creating a possible solution. A report organization is also made in this chapter to show the sequence of the project.



CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, it will explain more about the literature review and the existing system for the product. For this project, the observation is made based on the project domain, developing techniques and technology that have been used. The literature review is usually based on several sources such as journals, articles, website, interviews, and technical documents.

2.2 Related Work

2.2.1 Agriculture

Agriculture is the science, art and practice of cultivating plants and livestock. Agriculture was the key development in the rise of sedentary human civilization, whereby farming of domesticated species created food surpluses that enabled people to live in cities. (Wikipedia, 2020)

In recent years, greenhouse technology in agriculture has shifted toward automation, information technology direction with the IOT (Internet of Things) technology rapid development and wide application. Remote monitoring system with internet and wireless communications combined is proposed. At the same time, taking into account the system, information management system is designed. The system data that collected provided for agricultural research facilities.

Agriculture greenhouse production environment measurement and control system is an example of IOT technology application in agriculture. The critical temperature, humidity and soil signals are collected real-time in the agriculture production process, which is transmitted by wireless networks through machine to

machine support platform. It is to gain real-time data of agriculture production environment using SMS (Short Messaging Service), web, WAP (wireless application protocol) pattern, so that the terminal can master the information to guide the production. (Ji-chun Zhao, 2010)

2.3 Related project

2.3.1 Monitoring Moisture of Soil

Project Monitoring Moisture of Soil using Low Cost Homemade Soil Moisture Sensor and Arduino UNO (M. S. Kumar et. al., 2016) presented a method to manufacture soil moisture sensor to estimate moisture content in soil hence by providing information about required water supply for good cultivation. The system will use a low-cost homemade with soil moisture sensor and Arduino UNO. This is a simple Arduino project for a soil moisture sensor that will light up a LED at a certain moisture level. Rapid measurement techniques using electronic sensors which is time domain reflectometers, impedance, capacitance and dielectric sensors offer an alternative to destructive and time consuming gravimetric sampling.

Figure 2.1 shows simple Arduino project for a soil moisture sensor that will light up a LED at a certain moisture level. It uses Arduino Uno microcontroller board. Two wires placed in the soil pot form a variable resistor, whose resistance varies depending on soil moisture. This variable resistor is connected in a voltage divider configuration, and Arduino collects a voltage proportional to resistance between the 2 wires. Insert the 2 probes (wires, pcb) in the dry soil and measure the resistance value and then pour water and measure it again. Same sample of soil was analyzed by varying the volume of water, with values tabulated for various amounts of water. Using this method, we can determine the zone from which a sample originates, as well as the requirements for a successful plantation. To test sensor output values, the circuit consists of one sensor, a resistor, three LEDs for three areas, and an Arduino Uno Board Matlab interface for Arduino.

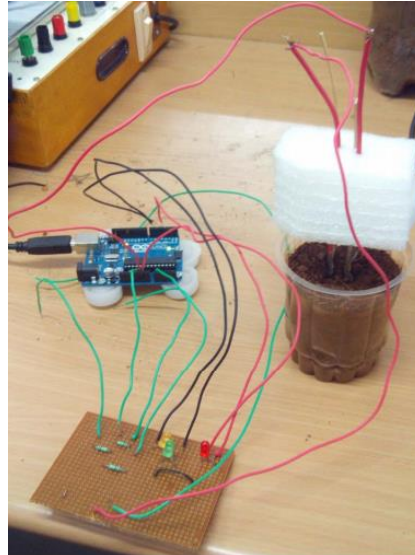


Figure 2.1: Setup for measuring soil moisture of a sample

Figure 2.2, 2.3 and 2.4 shows reading are taken for repeated number of times to check the repeatability of sensor and results are found to be not too deviating from the average value as shown in figures. Accuracy of setup can be increased by implanting more sensors in soil at different locations. Also the variation in soil moisture can be estimated over an area.

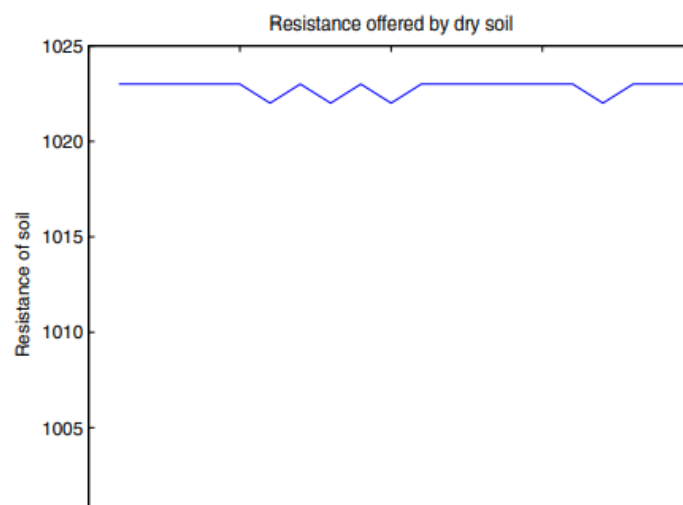


Figure 2.2: Result of dry soil

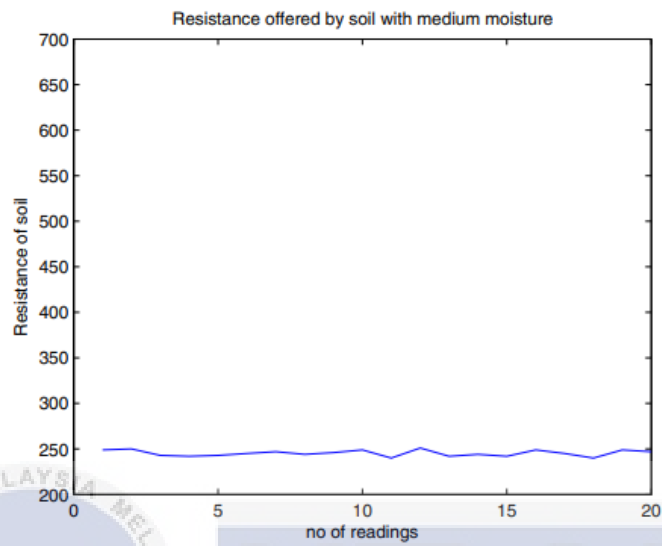


Figure 2.3: Result of medium moisture soil

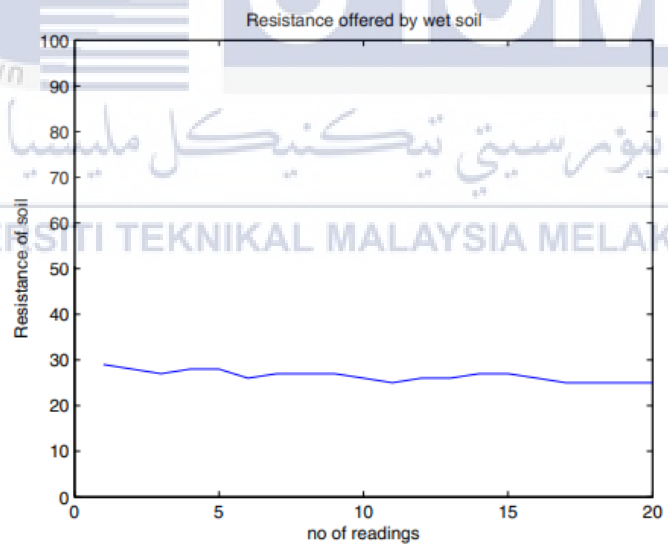


Figure 2.4: Result of wet soil

2.3.2 Smart Home Agriculture System

Based on Smart Home Agriculture System (S. Katangle et. al., 2020) project is used for Agricultural monitoring the environmental parameters including soil moisture, air temperature and humidity (DHT11) has been analyzed and displayed on the OLED screen. Using the water motor as per soil moisture threshold value can control the water flow. Hypertext transfer protocol based ESP8266 Wi-Fi based NodeMCU is routed to the Wi-Fi network which can communicate with cloud storage (Thingspeak) and IoT platform Blynk. The data from the sensors have been gathered, monitored and then sent to the storage cloud and Blynk through Wi-Fi.

Figure 2.5 shows voice command is given to the Google Assistant using mobile to monitor the lights and water Motors used through IFTTT and Blynk app platform. NodeMCU is interfaced with environmental parametric sensors which include moisture (soil) sensor, and water pump as a load. The data received from the sensor is forwarded to the NodeMCU and displayed on the OLED screen and also stored on the cloud storage (ThingSpeak). The water motor is switched on or off as per the soil moisture threshold value is observed. A working prototype for the system is shown in Figure 2.6.

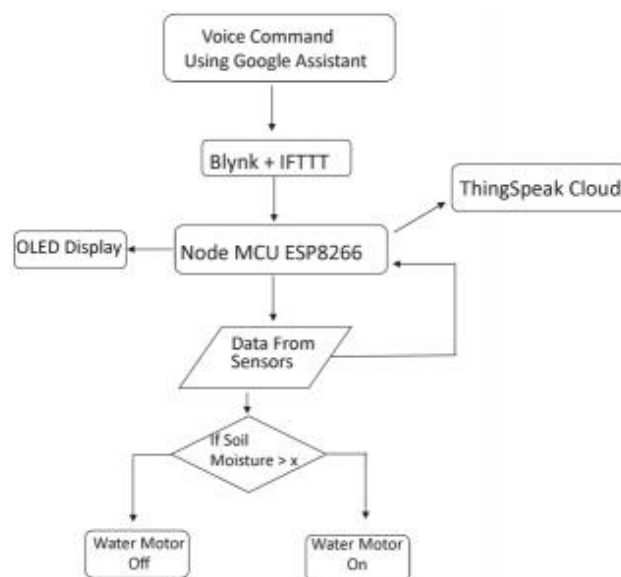


Figure 2.5: Flowchart of System

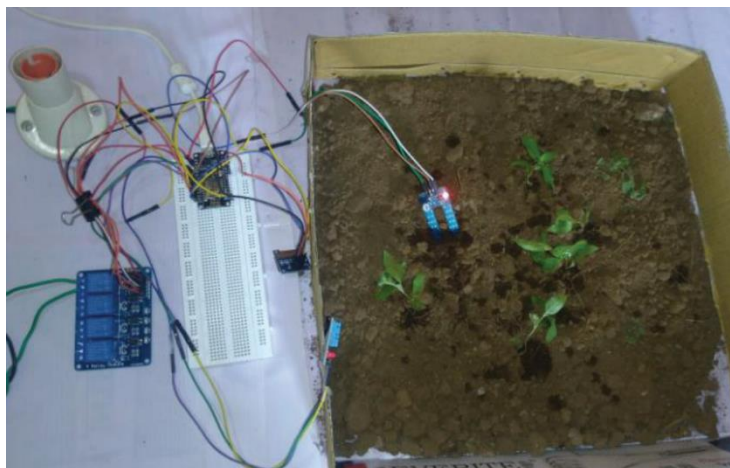


Figure 2.6: Prototype of the system

Figure 2.7 shows result soil moisture and water level which is 38%. Results are observed and able to get the data for environmental parameters including moisture present in the soil moisture readings in ThingSpeak as shown in figure 2.8. The results observed have been stored on the cloud storage ThingSpeak, the data in time-axis format is graphically visualized on IoT platform which allows the user to get informed about the internal conditions of the environment in actual real-time which helps the user to analyze and monitor according to the data.



Figure 2.7: Result of sensor value in Blyn

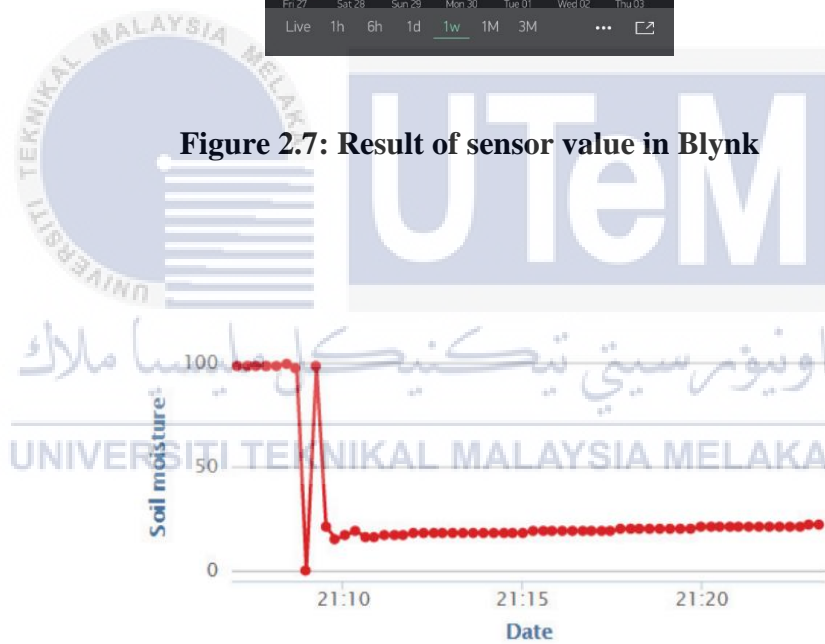


Figure 2.8: Results for the Soil moisture

2.3.3 Smart Home Garden Irrigation System Using Raspberry Pi by S.N. Ishak

Smart Home Garden Irrigation System Using Raspberry Pi (S.N. Ishak et. al., 2017) proposes a design for smart home garden irrigation system that implements system managed to reduce cost, minimize waste water, and reduce physical human interface. In this system, Raspberry Pi is implemented which integrated with multi-sensors such as soil moisture sensors, ultrasonic sensors, and light sensors. This system also managed to measure moisture of the soil and control the solenoid valve according to human's requirements.

This smart home garden irrigation system use Wi-Fi adapter which connected with Raspberry Pi to build an email configuration as a communication link between user and the system via internet. Sensor nodes include light sensor, moisture sensor, and ultrasonic water level sensor are used in measuring and monitoring environment parameters such as light soil moisture, and water level in tank as shown in Figure 2.9. The single relay module is working as switch to control the solenoid valve. Next, LED represents the symbol of activation system and it will turn ON after the user activates the system through their smartphones. Wi-Fi adapter is connected with USB port of Raspberry Pi board to allow the internet connection from router for the email configuration.



Figure 2.9: Prototype of Smart Home Garden Irrigation System

Android Apps had been developed and configured using Java program code. There are three layouts that have been designed as shown in Figure 2.10. First layout shows the user login for the system. The user has to key in their information in database through the second layout before logging in into the system. After the Sign Up and the Login steps completed, user can access the last layout and control the system by clicking the button as displayed.

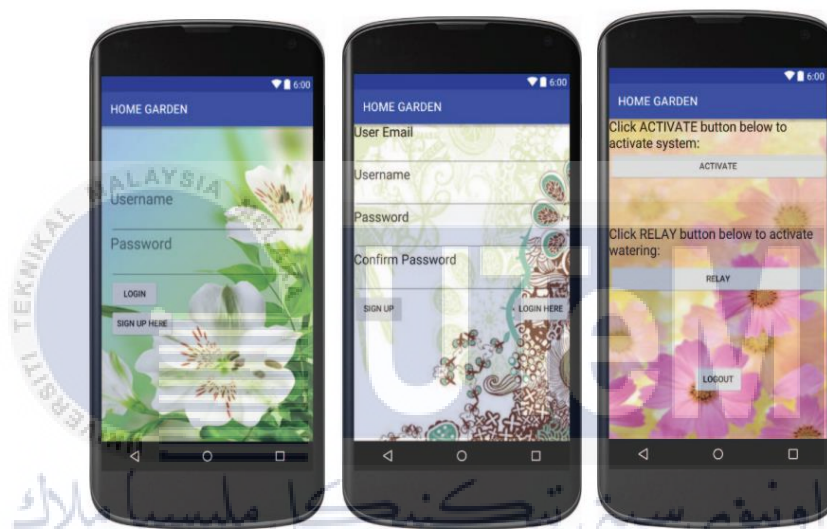


Figure 2.10: Android Apps Interfaces

If the water level is less than 15 cm, the system will email and notify the user as shown in Figure 2.11 to activate the relay button in Android Apps for allowing the water flow out from the solenoid valve. The status of successful of watering system will be displayed on terminal window as illustrated in Figure 2.12. If the reading of water level in tank or ultrasonic measurement is more than 15 cm, an email as shown in Figure 2.13 will be received by the user to inform that the water level that has run out. The system will be automatically stopped if the user did not refill the water tank. The system is detected as in a critical condition which is show in Figure 2.14.

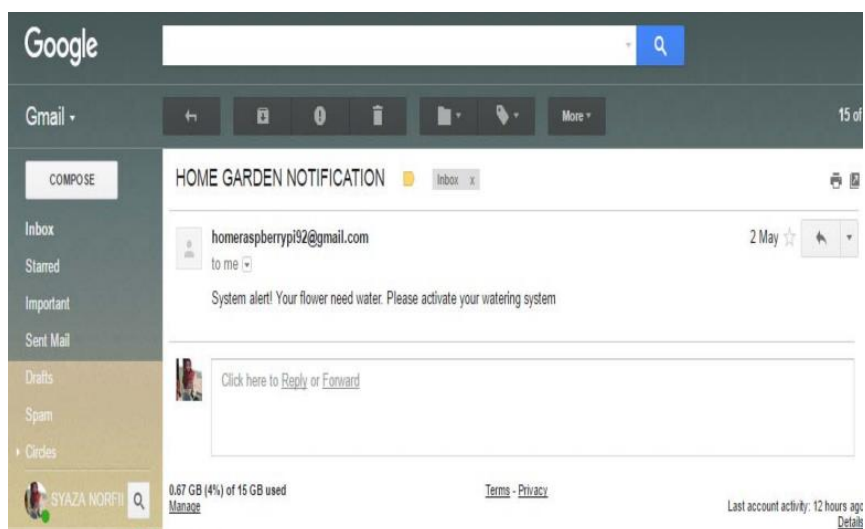


Figure 2.11: Email Configuration for Normal Operation

```

pi@raspberrypi:~$ sudo python3 project.py
LED is OFF
System is Deactive
waiting for activate system from user
LED is ON
System Activated
Light is present! Sunny day was detected.
Distance: 10.489336649576822 cm
Soil need water!
Email sent
Yeayyyy water will flowing out!
Watering successfull
Your flower's soil is just nice
pi@raspberrypi:~$

```

Figure 2.12: Raspberry Pi terminal for Normal Operation

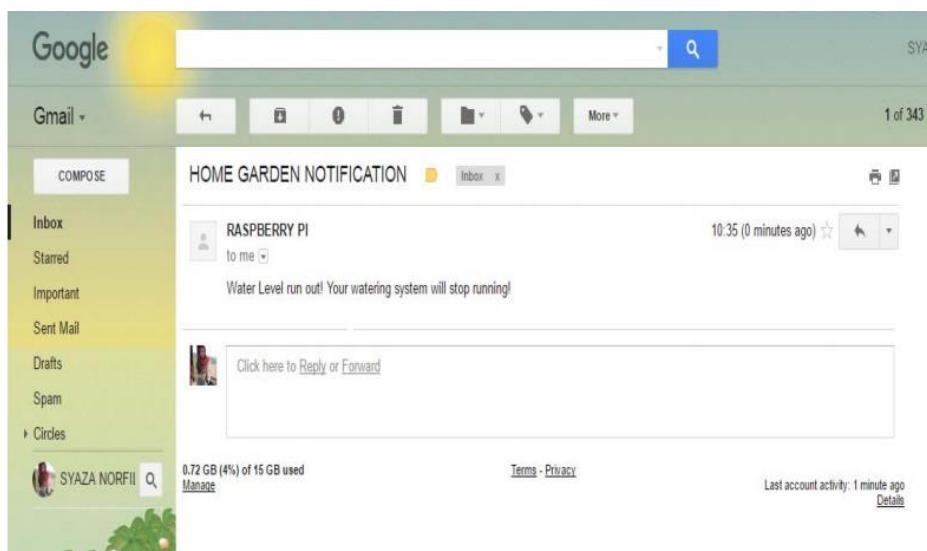


Figure 2.13: Email Configuration for Critical Operation



Figure 2.14: Raspberry Pi terminal for Critical Operation

2.3.4 Comparison

Table 2.1 shows three system which are reference smart system in this project. In this table, comparison have been made between the current system and proposed system.

Table 2.1: Comparison between Three Related Projects

Existing System	Monitoring Moisture of Soil	Smart Home Agriculture System	Smart Home Garden Irrigation System Using Raspberry Pi
Author	M. S. Kumar et. al.	S. Katangle et. al.	S.N. Ishak et. al.
Microcontroller	Arduino UNO	NodeMCU	Raspberry Pi
Sensors	Home-made moisture sensor	Soil Moisture Sensor	Soil moisture sensors, ultrasonic sensors, and light sensors.
Notification	LED	Blynk	Email
Description	Provide the user information about required water supply for good cultivation using moisture sensor.	Monitoring plant and control of irrigation monitored soil moisture.	User can monitor and control the plant growth in home garden area which allowing water to drip slowly to the soil surface and roots of plants.
Strength	Low cost and easy to setup	Easy to control and monitor	Provide good measurement
Weakness	Doesn't have notification in application and hard to monitor	Cannot monitor temperature area	Cannot monitor temperature area

2.4 Proposed solution

Based on the previous research, the comparison have been made in Table 2.1. There are some strength and weakness for each system. For the first project, they only focus on monitoring soil moisture. The second project also focus on monitoring soil moisture but have automatic irrigation function. For the third project, they focus on

automatic irrigation system using multiple sensors. Some functions have chosen from the projects to apply in my project which are the monitoring and plant watering system using advance sensors, monitor the soil moisture and temperature and monitor local air temperature and humidity.

There are several software and hardware that needed to be used in this system. First, for the microcontroller, NodeMCU ESP8266 will be used in this project. This because NodeMCU integrated support for WIFI network and provide low energy consumption. Next, sensors that will used are Temperature and Humidity sensors and soil moisture sensor. For temperature and humidity, DHT22 will be used. The DHT22 is a fundamental, minimal cost advanced temperature and humidity sensor. It also utilizes a capacitive moistness sensor to gauge the encompassing air, and spits out digital signal on the information pin (Prof. M. Sheth, 2019). Soil moisture sensor EK1099 is an inexpensive and easy to use device, which is utilized to monitor soil moistness level.

Based on three project above, they use different of notification platform. So, Blynk application will be used for platform to monitor the plant and send notification to user. This helps to connect the user mobile app with the NodeMCU. It gives effective dashboard by which client can make accurate interfaces utilizing distinctive gadget. The sensor values are sent to the user mobile's Blynk app. All the data sent to the Blynk app is completely secure.

2.5 Conclusion

In conclusion, literature review is an important chapter to build the project concept, it helps to understand the existing features of the system and to get clear picture to implement the system. The research and study will make the progression on doing this project run smoothly and more understanding. Next chapter is chapter 3, analysis which will address more details on requirement analysis consisting of project requirement, software requirement and hardware requirement.

CHAPTER 3: ANALYSIS

3.1 Introduction

The project development is focused in this chapters, proper approach is ensured in this project. This project will be developed by using the Prototype Model. This model is chosen as the methodology of developing this project because simple and easy to understand and use it. This method consists of 6 phases which are requirement analysis, system design, implementation, testing, deployment and maintenance.

3.2 Prototype Model

3.2.1 Requirement Analysis Phase

In this phase, all requirements of this project are analysed and gathered in a specification document and a feasibility analysis is performed to research if these requirements are justifiable. A literature review is inspected to recognize whether a smart system approach on gardener and the outcomes have proven that this approach is functional for them. During the process, the users of the system are interviewed to know what their expectation from the product is.

3.2.2 Quick Design Phase

In this phase, the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

3.2.3 Build a Prototype Phase

During this phase an actual prototype is designed based on the information gathered from previous phase. It is a small working model of the required system. The physical design specifications are turned into a working code. Arduino IDE is cross-platform application for Windows used to write and upload programs to compatible board. With inputs from the system design, the system is developed in small programs which will be integrated and tested in the next phase. In this phase, all code that applied in the system will be fulfill the requirement and specification.

3.2.4 Initial User Evaluation Phase

In this phase, the proposed system is presented to the user for an initial evaluation. It helps to find out the strength and weakness of the working model. Feedback with comment and suggestion are collected from the tester or the user for product assessment and review.

3.2.5 Refining Prototype Phase

If the user is dissatisfied with the existing prototype, it must be refined based on the user's input and ideas. This phase will not be completed until all of the conditions provided by the user have been satisfied. Once the user is pleased with the constructed prototype, a final system based on the authorized final prototype is created.

3.2.6 Implement and Maintenance Phase

The final system is carefully tested and put to production when it is designed based on the final prototype. Routine maintenance is performed on the system to save downtime and prevent large-scale breakdowns. There are some issues or problem that

have been collected from the feedback of users in their environment. So, in this phases provide maintenance and fix those issues, making sure it runs smoothly. Maintenance is done to deliver these changes to ensure the customer environment better.

3.3 Project Milestones

Project Milestone as a reference point that will used to monitor the project's progress and marks the major activity in a project. In order to make sure the flow of this project runs smoothly, the project milestone will be created and well planned to ensure all the activities in the project are able to be completed within the project timeline.

Table 3.1: Milestone of Project Activities

Project Activities	Start	End	Action
Requirement Analysis	Week 1	Week 4	<ul style="list-style-type: none"> - Gathering all the requirement - Make some research about agriculture and smart system - Study 3 related project
Design the system	Week 5	Week 7	<ul style="list-style-type: none"> - Create flowchart and diagram of the system
Build a Prototype	Week 8	Week 10	<ul style="list-style-type: none"> - Apply code of the system through Arduino IDE
Initial User Evaluation	Week 11	Week 12	<ul style="list-style-type: none"> - Evaluate the product to make sure the product suitable, safe to run and meet the requirement and objective

3.4 Conclusion

In conclusion, the methodology of this project is using the Prototype Model. Waterfall model methodology is implemented in this project because this methodology is simple and easy to understand and use it. Next chapter which is chapter 4, will address more detail on requirement analysis consisting of project requirement, software requirement, and hardware requirement.



CHAPTER 4: ANALYSIS AND DESIGN

4.1 Introduction

In the previous chapter, it already explained how the project was conducted. This chapter will define the results of the analysis of the preliminary design and the result of the detailed design. It will also focus on the analysis of the requirements of the project. The block diagram for this project will also be stated to ensure the project can be completed and well designed. The requirements such as the hardware and software needed on this project are includes in this chapter.

4.2 Problem Analysis

This project is a smart system for garden that being programmed in microcontroller (NodeMCU). Since the smart system is a simple and practical tool, the user may monitor their plants anytime and anywhere using their phone. The general flow of the system architecture of this advanced project will be discussed in this portion.

4.3 Requirement Analysis

Requirement analysis is the process of identifying the expectations of users for a product that is to be created or upgraded is known as requirement analysis. It includes

all of the tasks that are carried out in order to determine the demands of various stakeholders. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. (Wikipedia, 2021)

4.3.1 Data Requirement

The data requirement will be shown through data flow in Figure 4.1. To calculate whether the plant need to be watered or not all data will be analyze by the sensors. When the sensor detect the soil moisture is low which is below 30%, the water pump will start sprinkle the water. Then, the water pump will stop watering the plant when soil moisture is back to normal.

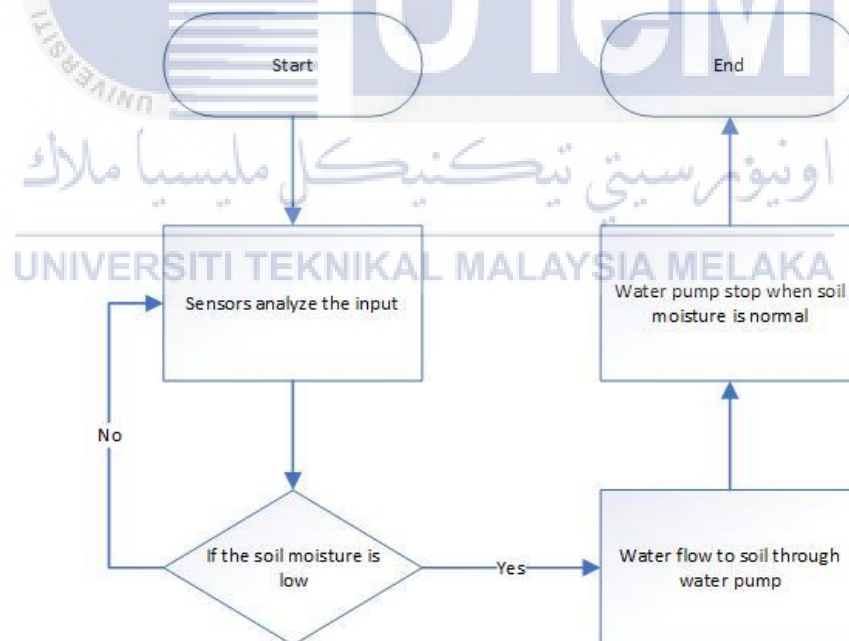


Figure 4.1: Data flow for system

4.3.2 Functional Requirement

Figure 4.2 shows functional diagram of this project and represent mainly about smart gardening controlling part. The smartphone would enable the user to interact with the entire system. The interaction point between the smartphone and microcontroller will be the Wi-Fi module. The Wi-Fi module would convey the information towards the microcontroller. Therefore, the microcontroller would send the commands towards the relay circuit. The relay circuit will be responsible in switching on the smart home appliances or make it works as demanded. There are three sensors connected and will send data to NodeMCU and displayed on the LED screen. After that, the water pump will react on/off when data receive in NodeMCU.

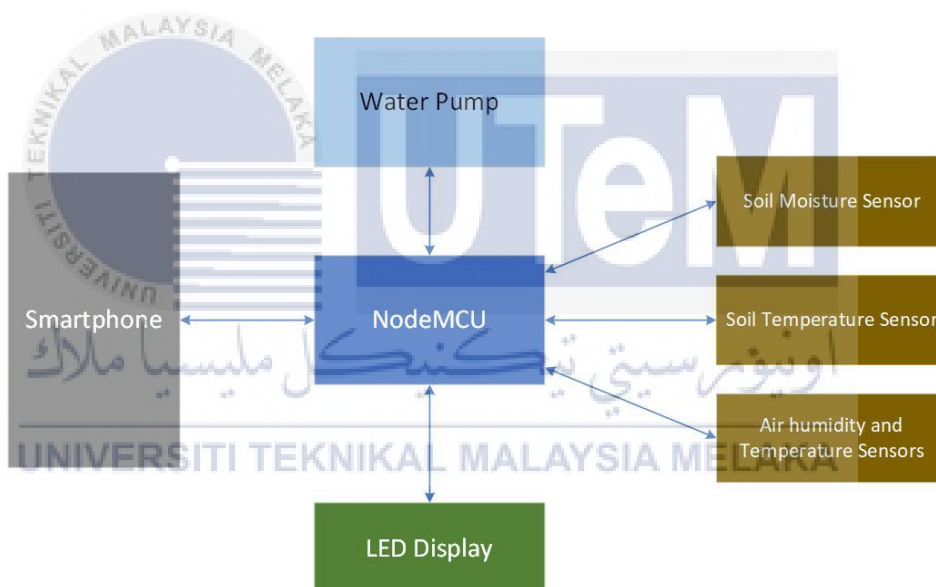


Figure 4.2: Functional Diagram

4.3.3 Software Requirement

4.3.3.1 Arduino IDE

Arduino Integrated Development Environment (IDE) is a cross-platform application can be supported with Windows, Mac OS and Linux. It is used to

write and upload programs into Arduino compatible boards. Arduino IDE uses language C and C++ specially for code structuring. (Wikipedia, 2021)



```

Blink | Arduino 1.8.5

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}

32 Arduino/Genuino Uno on COM1

```

Figure 4.3: Arduino IDE Interface

4.3.3.1 Blynk Application

Blynk is an IoT software that interact between user and system through internet connectivity. It function for display and receive input to user. In this project, it is used to display moisture and temperature of the soil and also display local air temperature and humidity reading that being capture by sensors.

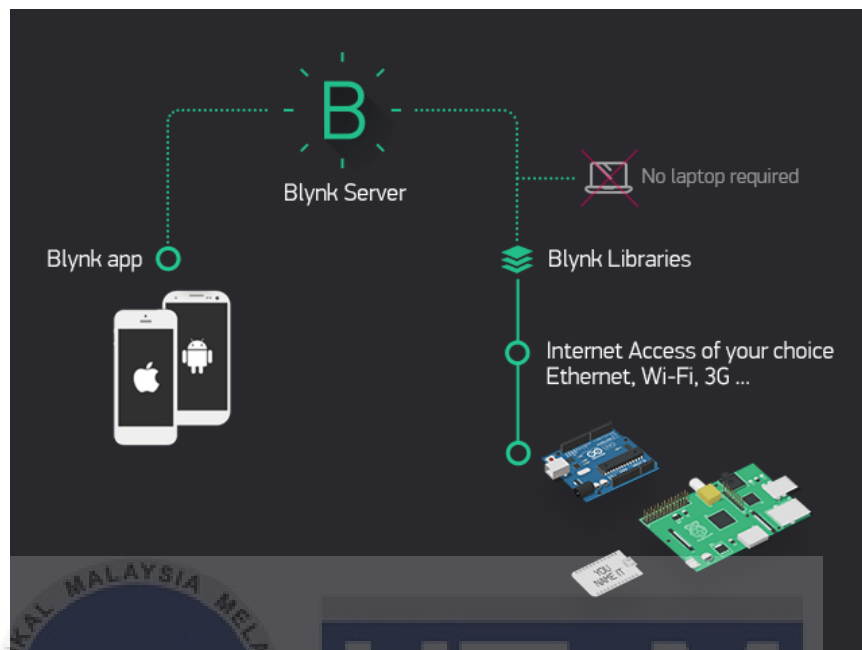


Figure 4.4: Blynk Mobile Application

4.3.4 Hardware Requirement

4.3.4.1 Microcontroller (NodeMCU)

NodeMCU is an open-source IoT platform based firmware and development board. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. NodeMCU ESP8266 development board comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface. (component101, 2020)

Features:

- Support STA/AP/STA+AP 3 working modes
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Flash Memory: 4 MB
- SRAM: 64 KB
- USB connector: Micro USB
- Temperature range: -40°C - 125°C
- Clock Speed: 80 MHz



Figure 4.5: NodeMCU ESP8266

4.3.4.2 Relay Module

A power relay module is an electrical switch operated by an electromagnet which controlling circuits by low-power signal or when several circuits must be controlled by one signal. A separate low-power signal from a microcontroller activates the electromagnet. The electromagnet pulls to open or close an electrical circuit when energized.

Features:

- 5V 2 channel relay interface board
- Able to control high load current, which can reach 250V, 10A or 125V, 15A
- Maximum Current Rating: 10A
- Maximum Voltage Rating: AC 250V / DC 30V

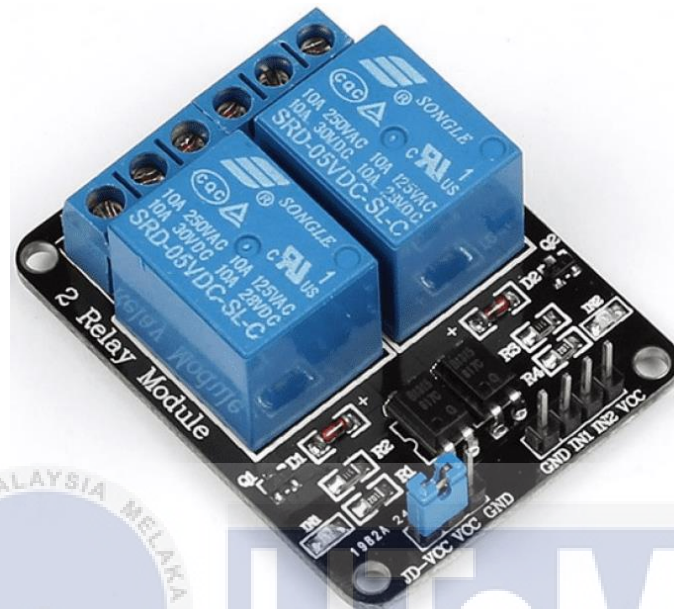


Figure 4.6: 2 Channel 5V Relay Module

4.3.4.3 Soil Moisture Sensor

Soil moisture sensor is sensor that measure the volumetric water content in soil. Because direct gravimetric measurement of free soil moisture necessitates the removal, drying, and weighing of a sample, soil moisture sensors indirectly detect volumetric water content by employing a proxy such as electrical resistance, dielectric constant, or neutron interaction. (Wikipedia, 2021).

Features:

- Operating Voltage: 3.3V to 5V DC
- Operating Current: 15mA
- Output Digital - 0V to 5V, Adjustable trigger level from preset
- Output Analog - 0V to 5V based on infrared radiation from fire flame falling on the sensor

- LEDs indicating output and power
- PCB Size: 3.2cm x 1.4cm
- LM393 based design
- Easy to use with Microcontrollers or even with normal Digital/Analog IC

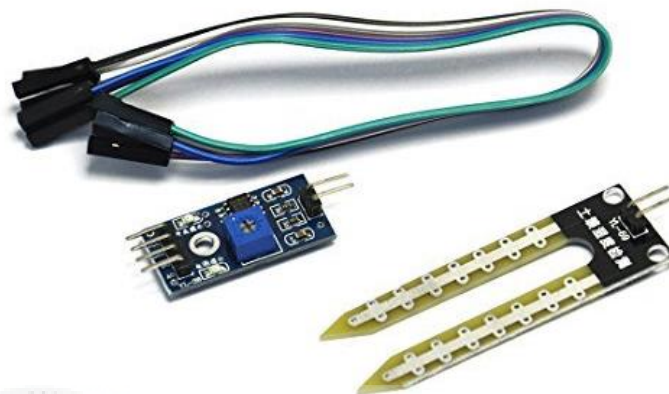


Figure 4.7: Soil Moisture Sensor

4.3.4.4 Air Temperature and Humidity Sensor (DHT22)

DHT22 is a low-cost and basic digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin which no analog input pins needed. It's fairly simple to use, but requires careful timing to grab data.

Features:

- 3 to 5V power and I/O
- Body size 27mm x 59mm x 13.5mm (1.05" x 2.32" x 0.53")
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings $\pm 0.5^\circ\text{C}$ accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- 2.5mA max current use during conversion (while requesting data)

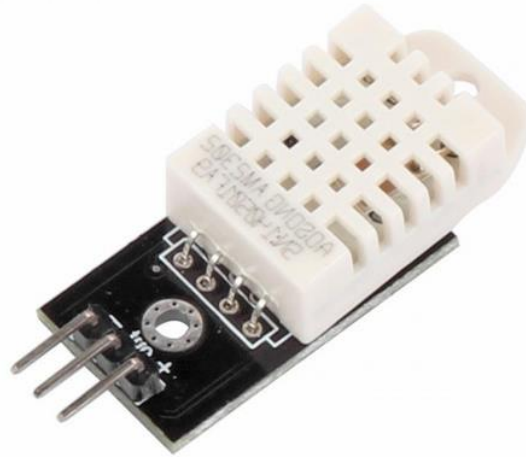


Figure 4.8: DHT22 Sensor

4.3.4.5 Soil Temperature Sensor

In this project, DS18B20 is used for soil temperature sensor. This sensor widely used to measure temperature in hard environments like in chemical solutions, mines or soil. The construction of the sensor is rugged and also can be purchased with a waterproof option making the mounting process easy.

Features:

- Programmable Digital Temperature Sensor
- Communicates using 1-Wire method
- Operating voltage: 3V to 5V
- Temperature Range: -55°C to $+125^{\circ}\text{C}$
- Accuracy: $\pm 0.5^{\circ}\text{C}$
- Output Resolution: 9-bit to 12-bit (programmable)
- Unique 64-bit address enables multiplexing
- Conversion time: 750ms at 12-bit
- Programmable alarm options
- Available as To-92, SOP and even as a waterproof sensor



Figure 4.9: DS18B20 Soil Temperature Sensor

4.3.4.6 Jumper Wire

Jumper wires are a quick and easy way to connect to other devices. Jumper wires come in a difference of colours, lengths, and types. There are three types of jumper wire which are male to male, male to female, and female to female. Every type of jumper wire performs the same function, but it is suitable to the different device with which it is used.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Features:

- Each cable length: about 20 cm /8-inch
- The cables can be separated to form an assembly containing the number of wires you require for your connection and to support non-standard odd-spaced headers
- Including 1X 40-pin male to female jumper wires, 1X 40-pin male to male jumper wires, 1x40-pin female to female jumper wires
- Packing in a color box.
- We have always cared about the customer experience and improve the product function details



Figure 4.10: Jumper Wire

4.3.4.7 Breadboard

A breadboard is a solderless device used for temporary prototype with electronics and test circuit designs. Most electrical components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and connecting them with wires where appropriate. Underneath the breadboard are metal strips that connect the holes on the top of the board.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Features:

- 830 tie points total: 630 tie-point IC-circuit area plus two 100 tie-point distribution strips providing 4 power rails.
- Size: 165mm X 55mm X 8.5mm
- Perfect for shield prototyping and testing
- Accepts wire diameters of 20-29AWG
- Package contents: 3 * 830 Point breadboard

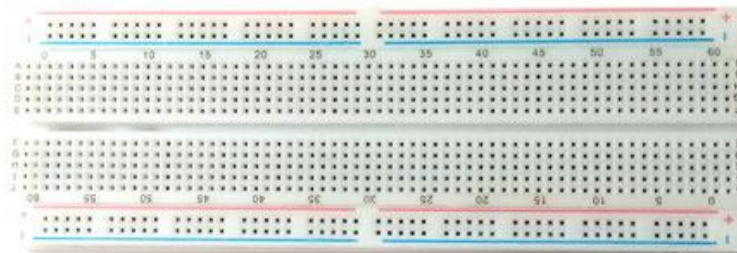


Figure 4.11: Breadboard

4.3.4.8 OLED Display Module

In this project, 0.96 inch blue OLED display module are used as shown in Figure 4.13. it can be interfaced with any microcontroller using SPI/IIC protocols. The resolution is 128x64. OLED or Organic Light-Emitting Diode is a self light-emitting technology composed of a thin, multi-layered organic film placed between an anode and cathode.

Features:

- Resolution: 128 x 64, View angle: > 160°
- Support voltage: 3.3V-5V DC, Power consumption: 0.04W during normal operation, full screen lit 0.08W.
- Embedded Driver IC: SSD1306. Communication: I2C/IIC Interface, only need two I / O ports.
- With high resolution of 128 x 64 pixels, no need backlight, self-illumination, and pixels stand out very well even in a brighter circumstances like full sunlight.



Figure 4.12: OLED Display Module

4.4 High-Level Design

A high-level design document, or HLDD, adds the essential elements to the current project description to represent a coding-ready model. The architecture that would be utilised to construct a system is described in high-level design. Figure 4.13 shows the example of purposed design that which served as a guide for develop the prototype.

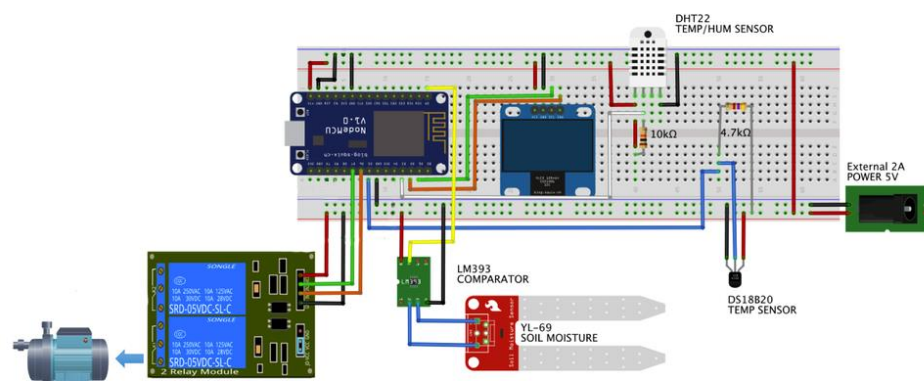


Figure 4.13: Model Design

4.4.1 System architecture

A system architecture is a conceptual model that outlines a system's structure, behaviour, and additional viewpoints. An architecture description is a formal description and representation of a system that is arranged in a way that allows for reasoning about the system's structures and actions. Figure 4.14 shows the system architecture design. The system start when the user open the Blynk application which connect with NodeMCU via Wi-Fi. There are three sensors connected and will send data to NodeMCU. User can monitor the data that have been detected by sensor. When the sensor detect the soil moisture is low, the water pump will start sprinkle the water. Then, the water pump will stop watering the plant when soil moisture is back to normal. The data that have been collected by sensors will store to database



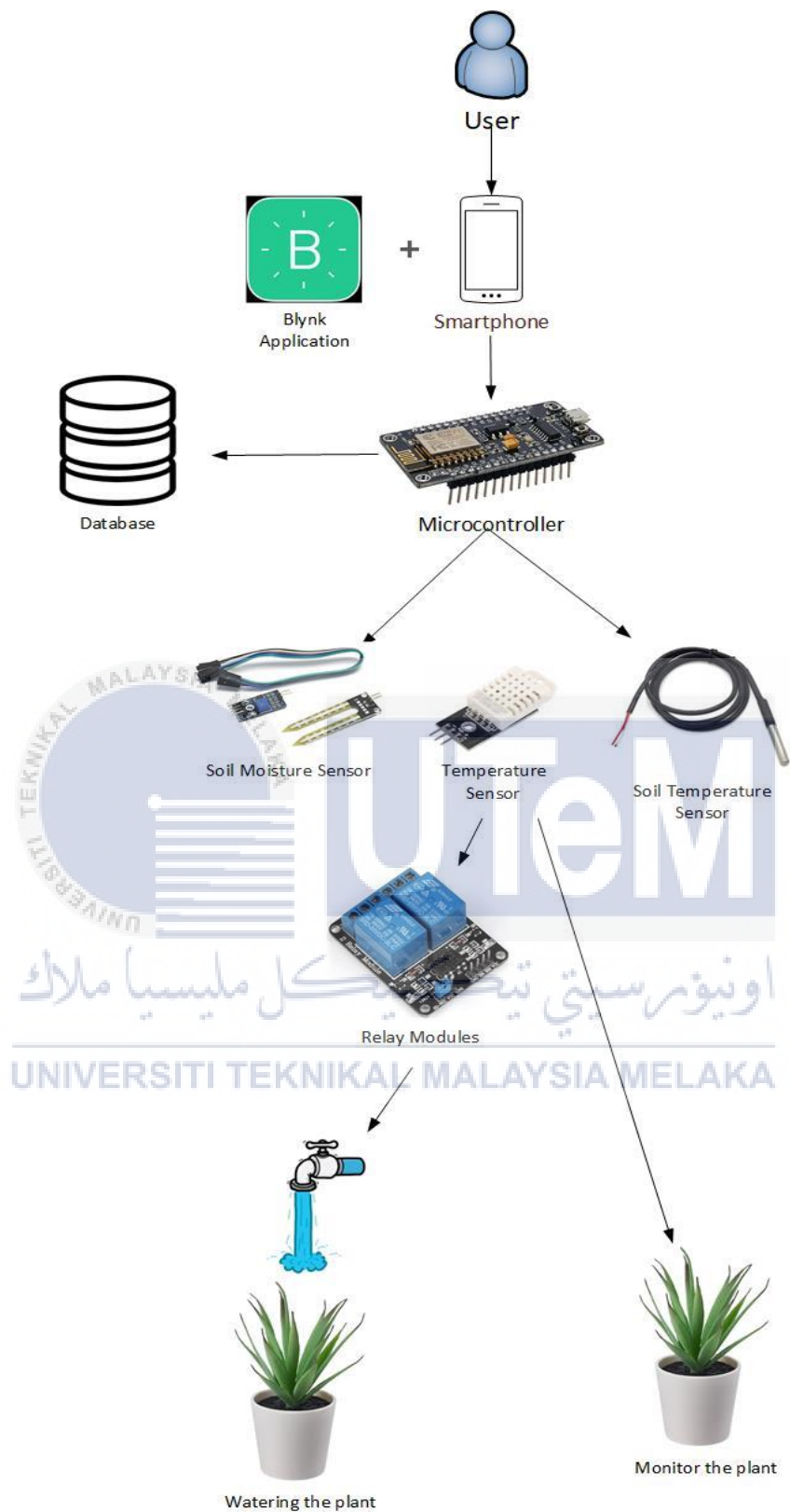


Figure 4.14: System Architecture

4.4.2 Database Design

4.4.2.1 ERD Diagram

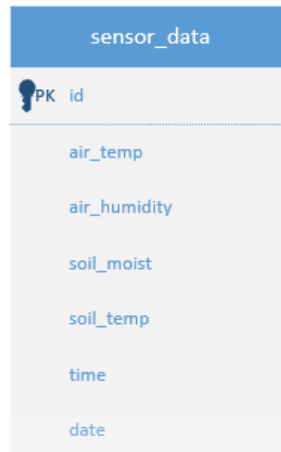


Figure 4.15: ERD diagram

4.4.2.2 Data Dictionary

Table 4.1: Data dictionary

Field Name	Data Type	Field Length	Constraint
id	Int	10	Primary key
air_temp	Varchar	255	Not null
air_humdity	Varchar	255	Not null
soil_moist	Varchar	255	Not null
soil_temp	Varchar	255	Not null
time	Timestamp		Not null
date	Date		Not null

4.5 Conclusion

This chapter is the pre-implementation stage for the implementation and include the flow of the overall system so that to have a better understanding before implement. One of the most important part of project implementation is analysis and design. All software and hardware requirements need to be identified and studied before carry out a project.



CHAPTER 5: IMPLEMENTATION

5.1 Introduction

This chapter discusses the activity involved during implementation phase of developing the system. It also will focus on how hardware and software will incorporate in this system. The monitoring process will ensure that the device is able to operate properly. As a developer and user, system will be review to determine the performance by implementing the appropriate protocol. The system design process will be explained in detail, including software installation and execution phases.

5.2 Development Environment Setup

Hardware and software requirements will include the design of Smart Gardening System device development environment. Every configuration will be listed and explained.

5.2.1 Hardware Development Setup

Hardware that used in this project is already described in previous chapter. Microcontroller that used in this project is NodeMCU ESP8266 model. NodeMCU can receive a hardware control order from Arduino IDE software programme. Data that receive from the sensors will be sent to firebase by NodeMCU. Figure 5.1 shows

indicates the pins of knowledge which can be placed into NodeMCU frame by hardware.

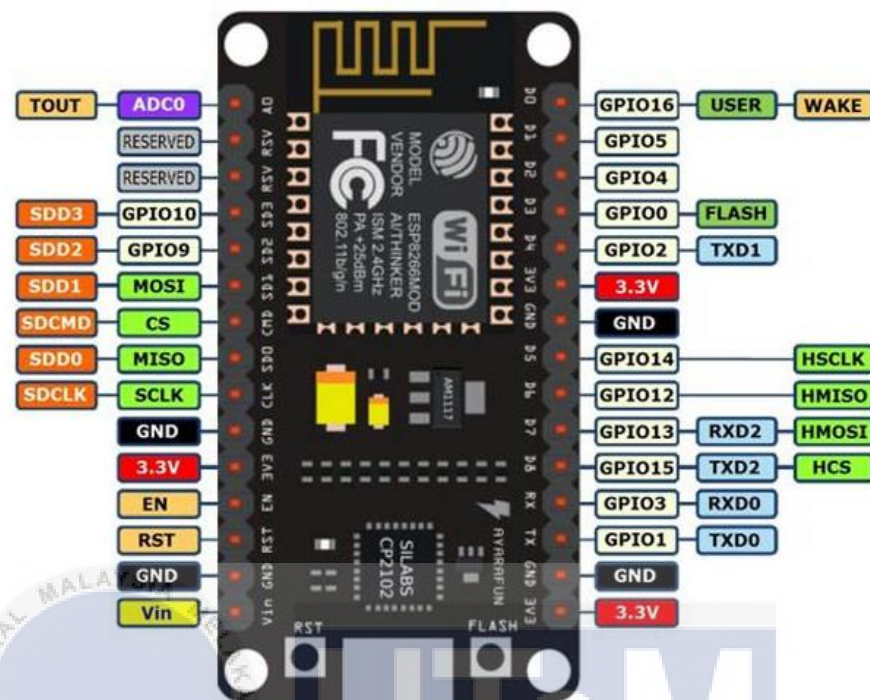


Figure 5.1: NodeMCU Pinout

5.2.2 Software Development Setup

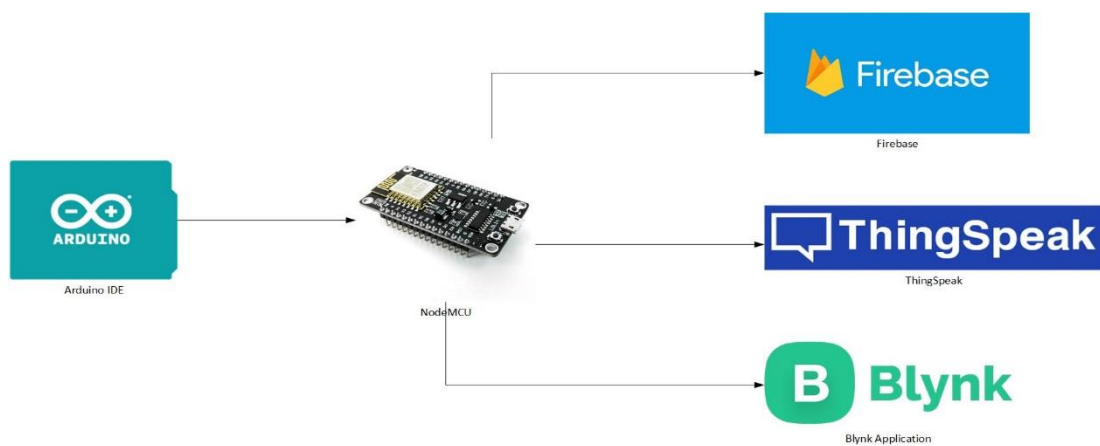


Figure 5.2: System Development

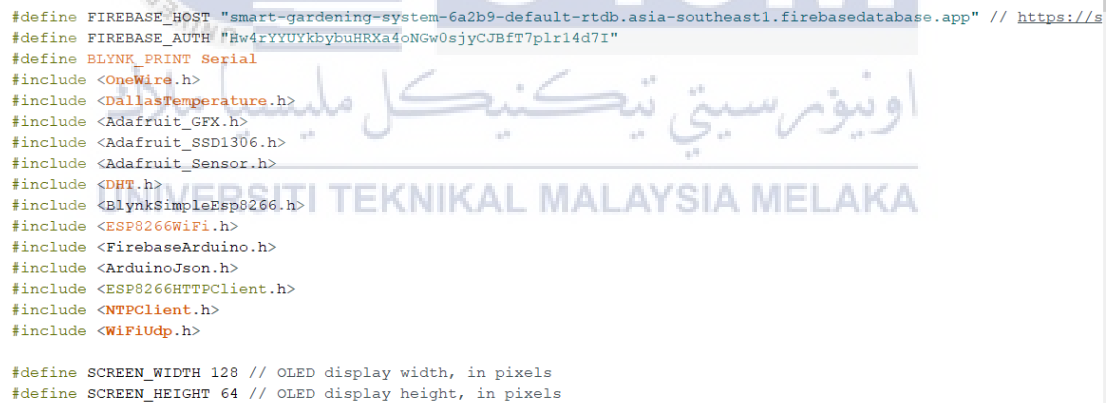
Figure 5.2 shows the system development using NodeMCU board and Arduino IDE. Arduino IDE is application that used for implement codes that connect to NodeMCU which are sensors, Blynk, Firebase and ThingSpeak. Firebase is used to store all data from sensors as database management tool. ThingSpeak is used for visualize and analyze live data from the sensors.

5.3 Hardware Configuration Management

The hardware used in this project are stated in the design phase. Sensors and water pump are attached to NodeMCU board.

5.3.1 Configuration Environment Setup

5.3.1.1 NodeMCU ESP8266 Coding



```
#define FIREBASE_HOST "smart-gardening-system-6a2b9-default-rtdb.asia-southeast1.firebaseio.com" // https://s
#define FIREBASE_AUTH "Hw4rYYUYkbybuHRXa4oNGw0sJyCJBfT7plr14d7I"
#define BLYNK_PRINT Serial
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPTCClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

Figure 5.3: NodeMCU Code

Figure 5.3 shows code that indicates the connection of NodeMCU to the Wi-Fi, Firebase and Blynk application.

5.3.1.2 OLED Screen Display Coding

```

// display temperature
display.setTextSize(1);
display.setCursor(0,0);
display.print("Air Temp: ");
display.setTextSize(1);
display.setCursor(80,0);
display.print(t);
display.print(" ");
display.setTextSize(1);
display.cp437(true);
display.write(167);
display.setTextSize(1);
display.print("C");

// display humidity
display.setTextSize(1);
display.setCursor(0, 10);
display.print("Air Humid: ");
display.setTextSize(1);
display.setCursor(80,10);
display.print(h);
display.print(" %");

//display soil moisture
display.setTextSize(1);
display.setCursor(0, 20);
display.print("Soil moist: ");
display.setTextSize(1);
display.setCursor(80,20);
display.print(soilmoisturepercent);
display.print(" %");

//display soil temperature
display.setTextSize(1);
display.setCursor(0, 30);
display.print("Soil Temp:");
display.setTextSize(1);
display.setCursor(80,30);
display.print(soiltemp);
display.print(" C");

```

Figure 5.4: OLED Screen Code

Figure 5.4 shows code of OLED display which display data from the sensors which are temperature, air humidity, soil moisture and soil temperature.

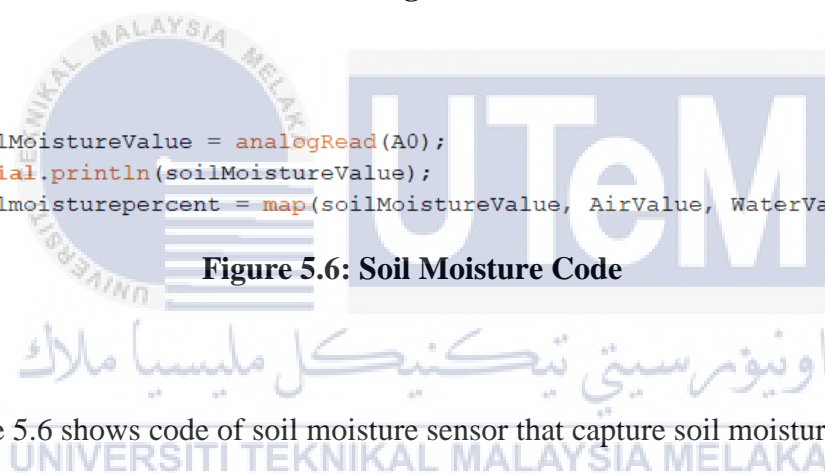
5.3.1.3 DHT22 Sensor Coding

```
//read temperature and humidity
float t = dht.readTemperature();
float h = dht.readHumidity();
sensors.requestTemperatures();
if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
}
```

Figure 5.5: DHT22 Code

Figure 5.5 shows code of DHT22 sensor that capture temperature and air humidity data.

5.3.1.4 Soil Moisture Sensor Coding



```
soilMoistureValue = analogRead(A0);
Serial.println(soilMoistureValue);
soilmoisturepercent = map(soilMoistureValue, AirValue, WaterValue, 0, 100);
```

Figure 5.6: Soil Moisture Code

Figure 5.6 shows code of soil moisture sensor that capture soil moisture data.

5.3.1.5 Soil Temperature Sensor Coding

```
sensors.requestTemperatures();
float soiltemp = sensors.getTempCByIndex(0);
```

Figure 5.7: Soil Temperature Code

Figure 5.7 shows code of soil temperature sensor (DS18B20) that capture soil temperature data.

5.3.1.6 Water Pump Coding

```

/*          WATER PUMP          */
if(soilmoisturepercent < 30) // change this at what level the pump turns on
{
  Serial.println("Nearly dry, Pump turning on");
  digitalWrite(D6,LOW); // Low percent high signal to relay to turn on pump
}
else if(soilmoisturepercent > 85) // max water level should be
{
  Serial.println("Nearly wet, Pump turning off");
  digitalWrite(D6,HIGH); // high percent water high signal to relay to turn off pump
}

```

Figure 5.8: Water Pump Code

Figure 5.8 shows code for activation water pump. Water pump will activate when soil moisture below 30 % and soil moisture exceed 85 %.

5.3.1.7 Blynk Coding

```

/*          Blynk          */
Blynk.virtualWrite(V10, t);
Blynk.virtualWrite(V11, h);
Blynk.virtualWrite(V12, soilmoisturepercent);
Blynk.virtualWrite(V13, soiltemp);
//PUSH Notification

if(soilmoisturepercent < 30){
  Blynk.notify("Alert!!! - Soil Moisture under 30%");
}

```

Figure 5.9: Blynk Code

Figure 5.9 shows code for Blynk application. All data from sensors will display in the application and notification will be appear when soil moisture under 30%.

5.3.1.8 Firebase Coding

```

String strHum = String(h); //--> Convert Humidity value to String data type.
String strTem = String(t); //--> Convert Temperature values to the String data type.
String strSm = String(soilmoisturepercent);
String strSt = String(soiltemp);

//-----Send Humidity data to the Firebase Realtime
String DBaddH = DBnm + "/" + TimeNow + "/" + HD; //--> Creating a Database path
Firebase.setString(DBaddH,strHum); //--> Command or code for sending Humidity data in

// Conditions for handling errors.
if (Firebase.failed()) {
    Serial.print("setting Humidity failed :");
    Serial.println(Firebase.error());
    delay(500);
    return;
}

```

Figure 5.10: Firebase Code

Figure 5.10 shows code for Firebase. All data from sensors will stored in Firebase.

5.3.1.9 ThingSpeak Coding

```

/*
  ThingSpeak
*/
if (client.connect(server,80)) {
  String postStr = apiKey;
  postStr += "&field1=";
  postStr += String(t);
  postStr += "&field2=";
  postStr += String(h);
  postStr += "&field3=";
  postStr += String(soilmoisturepercent);
  postStr += "&field4=";
  postStr += String(soiltemp);
  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);
}

```

Figure 5.11: ThingSpeak Code

Figure 5.11 shows code for ThinkSpeak. All data from sensors will display in form of graph website.

5.3.2 Firebase Setup

For Firebase configuration, user need to sign up Firebase account using Google account. Next, open the Firebase website to create real time database which shows in Figure 5.12.

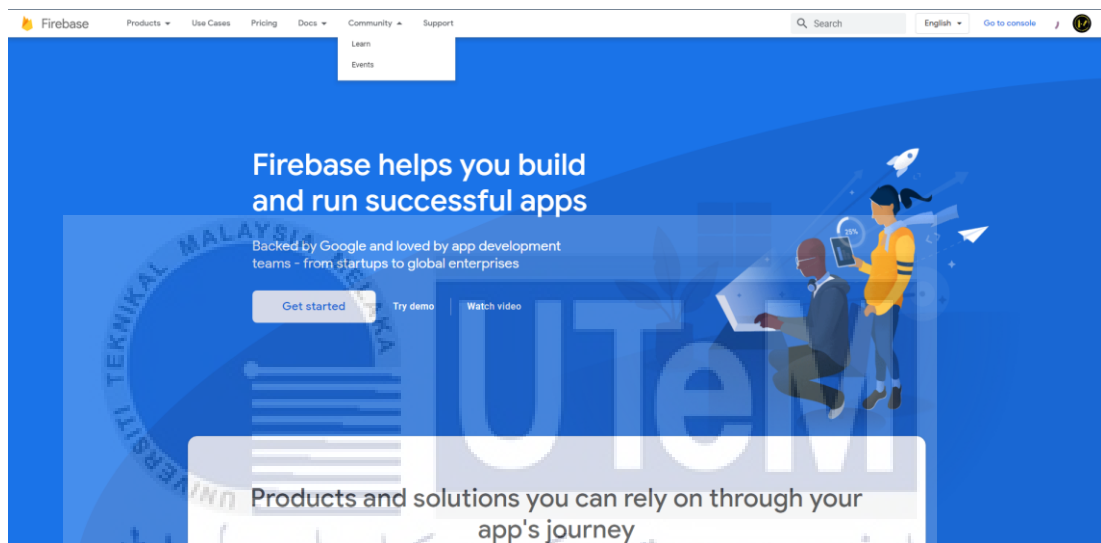


Figure 5.12: Firebase website

After that, add project and name the project as Smart Gardening System which shows in Figure 5.13. Then, start the code in Arduino IDE.

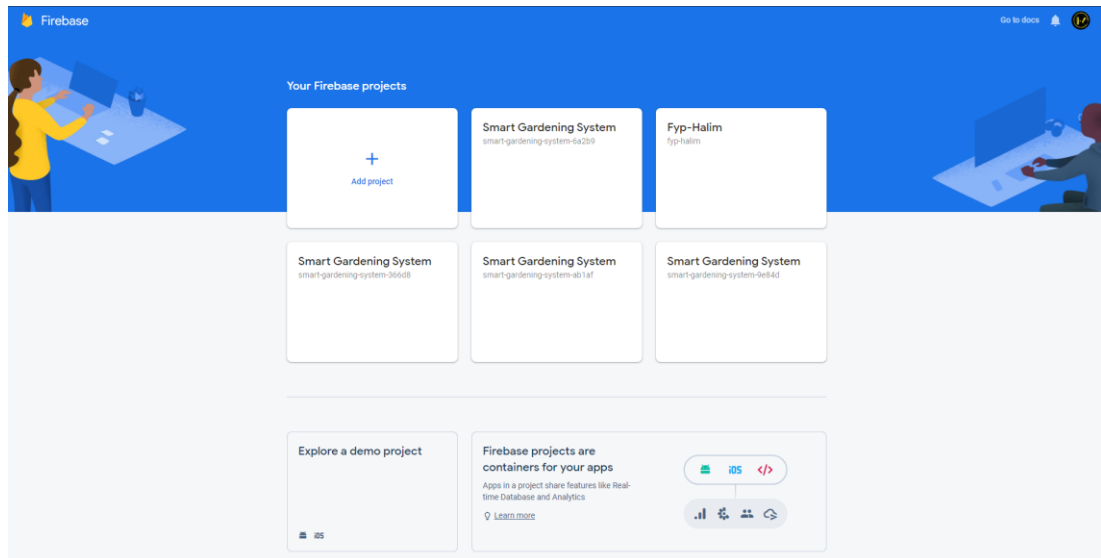


Figure 5.13: Add Project

5.3.3 ThingSpeak Setup

For ThinkSpeak configuration, user need to sign up ThinkSpeak account. Next, create new channel and fill up the information of system which shows in Figure 5.14.

Figure 5.14: New Channel

Then, edit field name as shown in Figure 5.15. Go to API Keys and copy the key which shows in Figure 5.16. After that, apply it in Arduino IDE code.

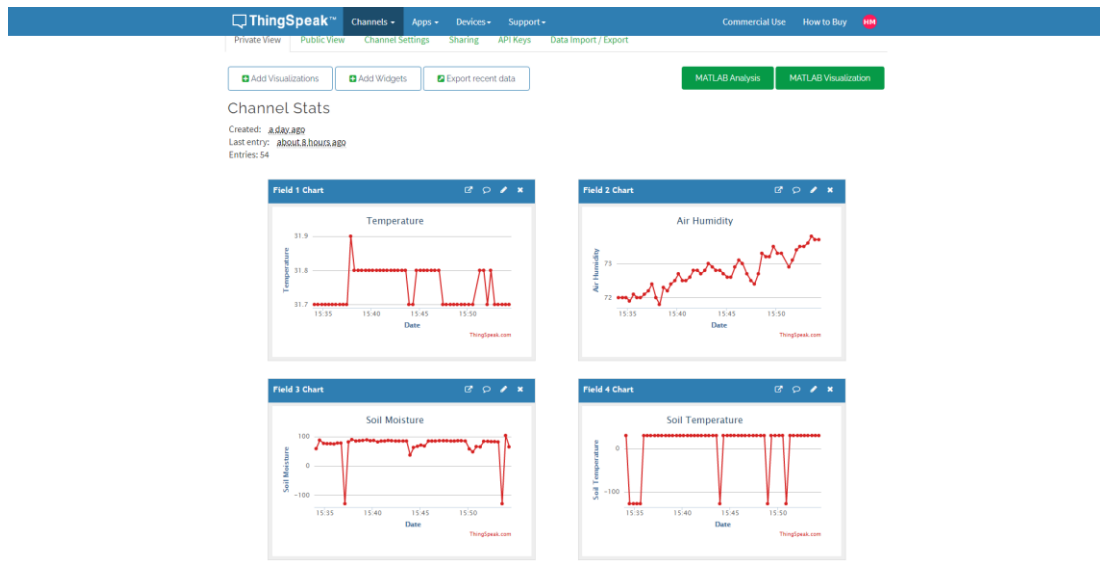


Figure 5.15: Channel Stats

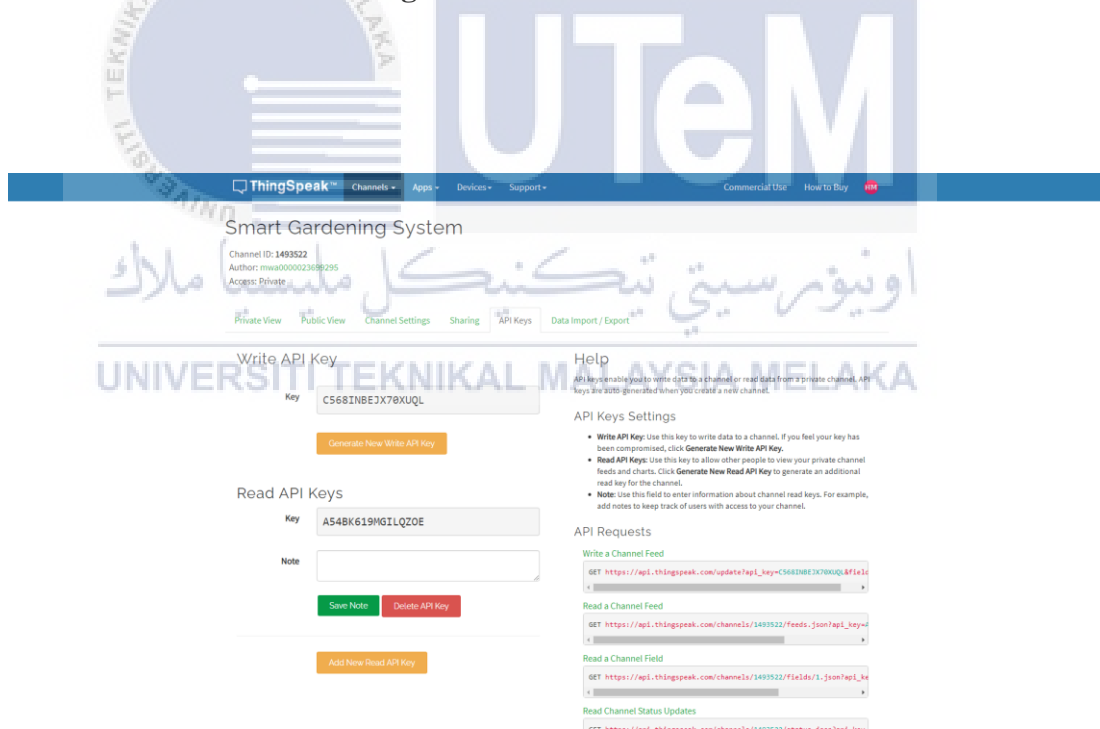


Figure 5.16: API Key

5.4 Implementation Status

Table 5.1: Implementation Status

No.	Status	Description	Duration (days)
1	Assembling hardware	Process to connect all hardware to board which are sensors, OLED screen and water pump.	14 days
2	Building Prototype	This process is to setup the prototype of the project	20 days
3	Upload NodeMCU coding	This process is to ensure the NodeMCU connect with all hardware.	28 days
4	Developing system	All system coding will be applied and arranged in the NodeMCU board through Arduino IDE.	60 days

5.5 Conclusion

In conclusion, for this chapter the prototype has been implemented according to the designs, problem statement and objectives. This chapter is most important part in developing the system which discusses implementation procedure. Finally, testing part will be discussed in further in next chapter.

CHAPTER 6: TESTING

6.1 Introduction

In this chapter, the testing of the prototype is carried out to know whether it is successful or not. All hardware and software need to be tested. If the testing is not being carry out, the prototype cannot be justified to be success and need some feedback from the users.

6.2 Test Plan

Testing plans describe the activities, scope and basic testing for the application. It is important to ensure all the objective achieved and overall, of the application is running smoothly. Any bugs or errors can be detected and fixed it before sending to the customer.

6.2.1 Test Environment

This project will remotely access through laptop by connecting wirelessly to establish the connection for NodeMCU board before carrying out any testing process. The Raspberry Pi will be running with the all hardware and connect to the Wi-Fi connection. This is ensuring the real-time capture image can be test out successfully.

6.2.2 Test Schedule

The testing processes for the system will take period of time. Any bugs or error being found will return be back to the implementation phase to check again. This is the continuation process until the system is error-free and able to execute successfully.

Table 6.1: Test Schedule

Module Name	Duration	Test Start Date	Test Date Completed
Communication between NodeMCU board and hardware.	2 weeks	5 July 2021	18 July 2021
Communication between NodeMCU and Arduino IDE coding.	2 weeks	19 July 2021	1 August 2021
The effectiveness of the sensors detect and sent the data to NodeMCU.	2 weeks	2 August 2021	15 August 2021
Communication between NodeMCU to Firebase and ThingSpeak.	2 weeks	16 August 2021	29 August 2021

6.3 Test Strategy

Test strategy is guiding plan for the software development cycle of this project. It will explain the test method where the phase is a breakthrough in deciding who, what, when and how to test. The project will be tested by project developer.

6.3.1 Classes of Tests

The classes of tests have four types of tests for this project to ensure the algorithm and hardware does not have any errors. The specification type of testing during the testing are software test, hardware test, usability test and functionality test. The classes of tests will be mentioned in the table 6.2.

Table 6.2: Test Classes

Test	Description
Software Test	Any errors or problem in the code will be identified and repaired
Hardware Test	Any incorrect hardware and connections will be determined to ensure that all hardware involved in the project are in good condition
Functionality Test	To ensure that the functionality of the project as planned
Accuracy Test	To ensure that accuracy of the project is good.

6.4 Test Design

Test design discusses about the test case identification, test cases and expected result for each scenario which are designed and documented. The test description discusses integration test and functionality test.

6.4.1 Test Description

Table 6.3: NodeMCU Test

Test Case ID	TC_1
Test Functionality	To test NodeMCU board functionality.
Test Case Summary	To upload coding to the NodeMCU board
Precondition	Established the connection of NodeMCU board
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload
Expected Result	The code is successfully uploaded to the NodeMCU board
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.4: DHT22 Sensor Test

Test Case ID	TC_2
Test Functionality	To test DHT22 sensor functionality
Test Case Summary	To capture surrounding temperature and air humidity data
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload
Expected Result	The data of temperature and air humidity will be display on OLED screen and Blynk.

Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.
---------------	--

Table 6.5: Soil Moisture Sensor Test

Test Case ID	TC_3
Test Functionality	To test soil moisture sensor functionality
Test Case Summary	To capture soil moisture data
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload
Expected Result	The data of soil moisture will be display on OLED screen and Blynk.
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.6: Soil Temperature Sensor Test

Test Case ID	TC_4
Test Functionality	To test soil temperature sensor functionality
Test Case Summary	To capture soil temperature data
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload

Expected Result	The data of soil temperature will be display on OLED screen and Blynk.
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.7: Wi-Fi Test

Test Case ID	TC_5
Test Functionality	To test Wi-Fi connection on board functionality
Test Case Summary	To connect the board to internet which are to Blynk, Firebase and ThingSpeak
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload
Expected Result	The board can make connection which receive and sent the data to server
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.8: Water Pump Test

Test Case ID	TC_6
Test Functionality	To test water pump functionality

Test Case Summary	To make sure water pump flow the water to soil.
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload
Expected Result	The water pump will flow the water when soil moisture low and stop the flow when high soil moisture reading.
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.9: Firebase Test

Test Case ID	TC_7
Test Functionality	To test Firebase connection
Test Case Summary	To make sure all sensors data can store in Firebase real-time database.
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload 4. Check Blynk application
Expected Result	The Firebase will show the real-time database of the sensors
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

Table 6.10: ThingSpeak Test

Test Case ID	TC_8
--------------	------

Test Functionality	To display graph of sensors data in ThingSpeak website
Test Case Summary	To make sure data displayed in form of graph
Execution Steps	<ol style="list-style-type: none"> 1. Run the Arduino IDE 2. Write coding to upload into the NodeMCU board 3. Upload 4. Check ThingSpeak website
Expected Result	User able to view the sensors data in the ThingSpeak website.
Error Message	If the coding is not correct or have an error, the Arduino IDE will show error alert and not able to run successfully.

6.5 Test Result and Analysis

6.5.1 NodeMCU Board Test

NodeMCU board code will be uploaded in Arduino IDE as shown in Figure 6.1.



```

#define FIREBASE_HOST "smart-gardening-system-6a2b9-default-rtdb.asia-southeast1.firebaseio.com" // https://smart-gardening-system-6a2b9-d
#define FIREBASE_AUTH "Hw4rYYUYkbybuHRKa4cNGw0sjyCUBf7plr14d7I"
#define BLYNK_PRINT Serial
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

```

Figure 6.1: NodeMCU Code

6.5.2 OLED Screen Test

OLED screen is connected to NodeMCU board which is to display the sensors data as shown in Figure 6.2.

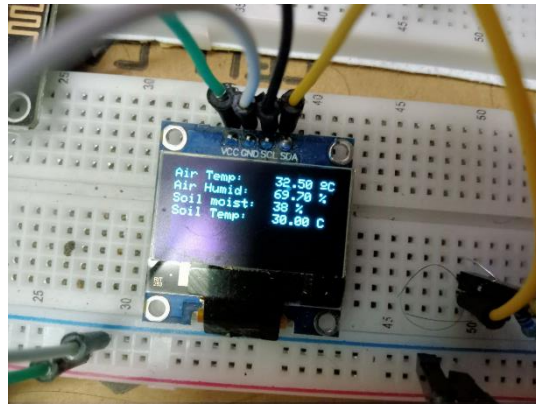


Figure 6.2: OLED Screen

6.5.3 Air Temperature and Humidity Sensor (DHT22) Test

The sensor will receive air temperature and humidity data and display it on OLED screen as output.

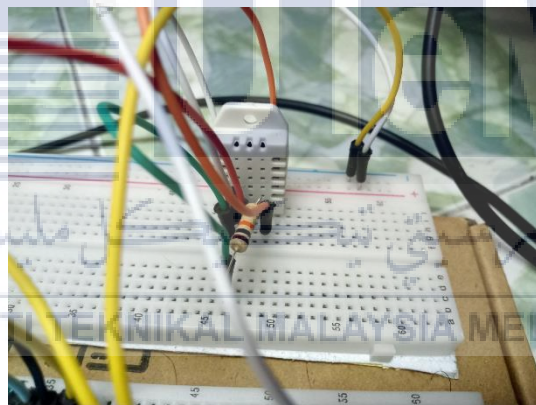


Figure 6.3: DHT22 Sensor

6.5.4 Soil Moisture Sensor Test

The sensor will receive soil moisture data and display it on OLED screen as output.



Figure 6.4: Soil Moisture Sensor

6.5.5 Soil Temperature Sensor Test

The sensor will receive soil temperature data and display it on OLED screen as output.

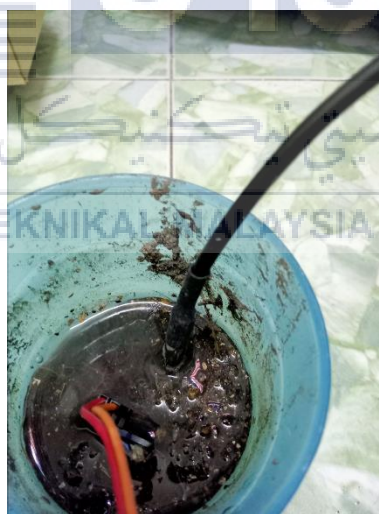


Figure 6.5: Soil Temperature Sensor

6.5.6 Water Pump Test

The water pump will flow the water when soil moisture low and stop the flow when high soil moisture reading.



Figure 6.6: Water Pump

6.5.7 Firebase Test

Firestore will receive all the data from the sensors and store in the realtime database as shown in Figure 6.7.

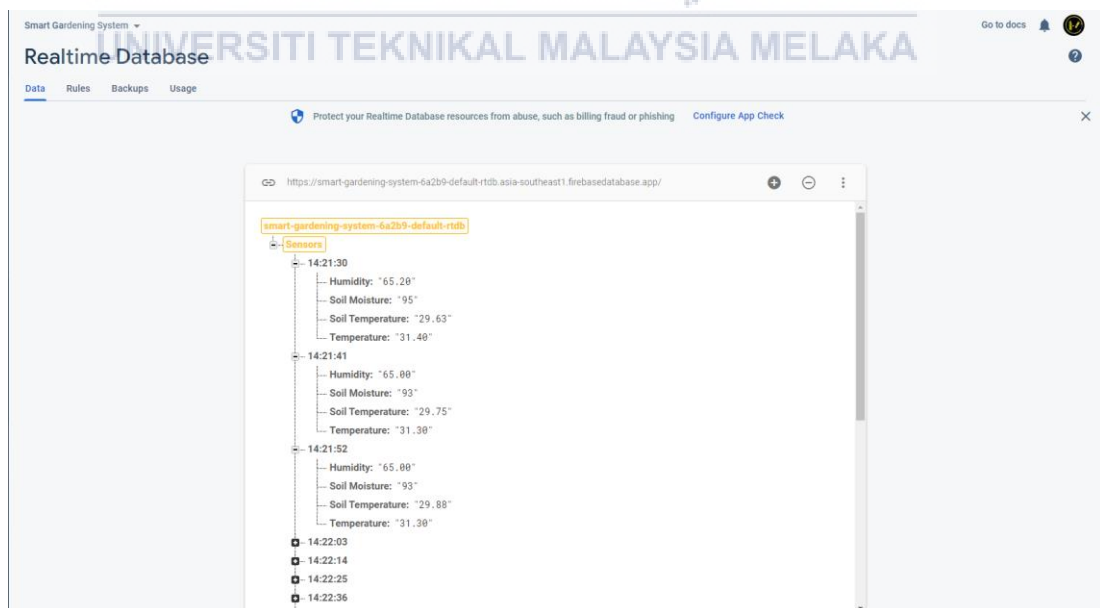


Figure 6.7: Firebase

6.5.8 ThingSpeak Test

ThingSpeak website will display all data from sensors and generate a graph as shown in Figure 6.8.

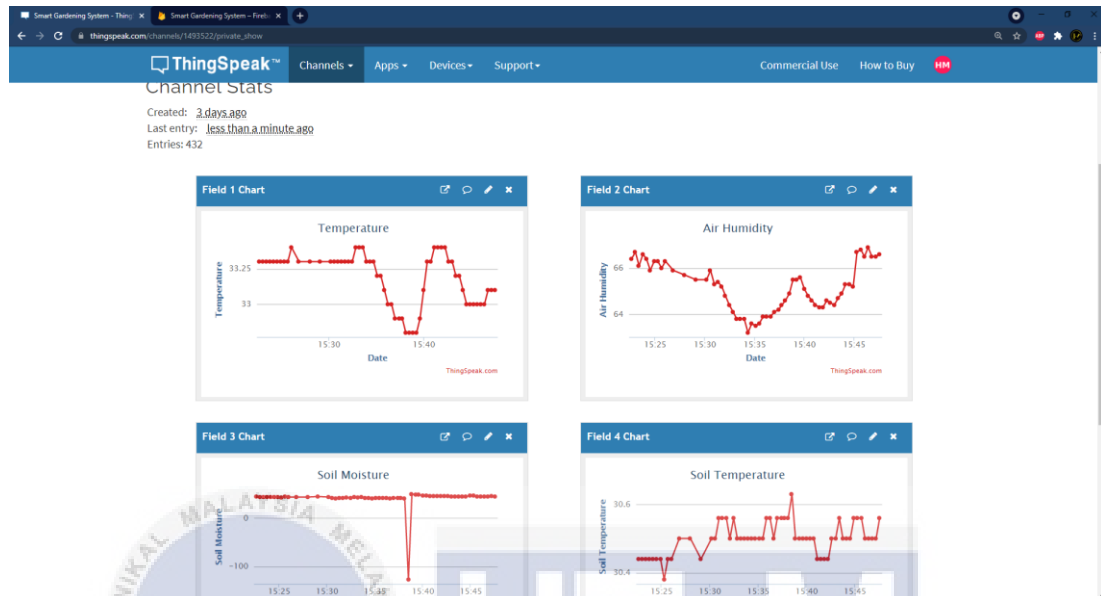


Figure 6.8: ThingSpeak

6.6 Conclusion

In conclusion, for this chapter is crucial and need a lot of observation to make sure the testing is success. However, planning and timing need to be parallel to complete the testing. The prototype that has been developed need to do a lot of testing until the results satisfied developer. All the testing that has been done is to verify the products is a success or fail. If success, the testing and developing of the project is doing a good job and if fail, justification is a must. The next chapter is chapter seven which is project conclusion.

CHAPTER 7: PROJECT CONCLUSION

7.1 Introduction

In this chapter, there will be a conclusion and summarization the overall project from the beginning until the completion of the project. The limitation and future work of the project also will be stated for any further improvement in the future. This will effectively improve the efficiency of the system and make it more efficient also comprehensive.

7.2 Project Summarization

In the beginning, the goal of this project is to develop smart gardening system using multiple sensors. Second objective is to monitor and analyses humidity and temperature of plant and soil. The last objective is to display data to user about humidity and temperature of plant and soil.

The definition of develop smart gardening system using multiple sensors is successful assemble all hardware and find the suitable functions to make this project happens. The prototype works successfully with assembly of hardware and software. This objective also can be link with second objective which is to monitor and analyses humidity and temperature of plant and soil. User can control water flow and monitor their plant using Blynk. User also can analyses them in ThingSpeak website. Furthermore, the third objective is also link to the second objective to complete the system. The data is success displayed to user about humidity and temperature of plant and soil. They can view the data in mobile phone which is Blynk application and view through OLED screen.

The result of the project is success and complete all the objectives. It takes time and need a lot of research as developing new products is not easy as it seems. The strength of the product is it have high accuracy in humidity and temperature of plant and soil and also can send real-time notifications to the user through the application. The weakness of the products is water flow only based on soil moisture reading.

7.3 Project Contribution

The project contribution will be assigned to home gardener or small gardener. This project's initial contribution is the usage of a microcontroller with other components that used to complete this project. All the project contribution that has been mentioned is also for individual purposes and of course for users that interested to use this product.

7.4 Project Limitation

The project limitation during doing this project are time and limited resources. In this project, the prototype unranged which jumpwire not organized well. It might be effect the wiring in breadboard.

7.5 Future Works

For the future works, the suggestion that can be made is by adding new functions and features. For example, a system can activate heater or cooler when temperature is high or low. Furthermore, the current components can be replaced by more advanced or modern components.

7.6 Conclusion

In conclusion, the projects met the objectives that have been stated before and achieved the goals. However, to make this project perfects a lot of hard work and sacrifice is needed. The problem statement that has been stated have been solved. From the chapter until last chapter, there is a lot thing that happened and fortunately all the problems can be solved. Chapter 1 until 4 the documentation part is easy to do as the product is still developing and not have problems yet. When the product has been developed, there is problems occur like how to implement the coding and error when debug the coding. Last but not least, for the future of this product, this product can make more functionality for gardener.



REFERENCES

Matti Satish Kumar, T. R. (2016). Monitoring moisture of soil using low cost homemade Soil Moisture Sensor and Arduino UNO .

Blynk. (2021). Retrieved from <https://blynk.io/>

NodeMCU. (2021). Retrieved from https://www.nodemcu.com/index_en.html

S.N. Ishak, N. M. (2017). Smart Home Garden Irrigation System Using Raspberry Pi.

Shubham Katangle, M. K. (2020). Smart Home Automation-cum Agriculture System

PrahladBhadani, Dr.VVashisht (2019). Soil Moisture, Temperature, and Humidity Measurement Using Arduino, International Conference on Cloud Computing, Data Science & Engineering,

C. Kumar Sahu and P. Behera (2015). A low cost smart irrigation control system.

Yao Chuanan and Yu Yongchang (2010). Implementation of greenhouse monitoring system based on RF transceiver.

S. J. P. Byoungwook Min (2018). A Smart Indoor Gardening System Using IoT Technology.

B. S. A. K. G. Satyam Kumar Sinha (2017). IOT Based Smart Garden Monitoring System.

H.Durani, M.Sheth, M.Vaghasia, S.Kotech (2018). Smart Automated Home Application using IoT with Blynk App.

APPENDIX A

In this appendix A contains complete coding for Smart Gardening System

```

#define FIREBASE_HOST "smart-gardening-system-6a2b9-default-rtdb.asia-
southeast1.firebaseio.com"
#define FIREBASE_AUTH "Hw4rYYUYkbybuHRXa4oNGw0sjyCJBfT7plr14d7I"
#define BLYNK_PRINT Serial
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define DHTPIN 14 // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:

#define DHTTYPE DHT22 // DHT 22 (AM2302)

#define ONE_WIRE_BUS D3

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

DHT dht(DHTPIN, DHTTYPE);

String apiKey = "C568INBEJX70XUQL";

char auth[] = "bJpBMqX8z74vMmnP73bH_HppaetffOsB"; //AUTHENTICATION
KEY PROVIDED BY BLYNK CLOUD
char ssid[] = "Halim@unifi"; //WIFI SSID
char pass[] = "hlmrvp99"; //WIFI PASSWORD

```

```

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

// > UTC +07:00 -> 7 * 60 * 60 = 25200
const long utcOffsetInSeconds = 25200;
//-----

//-----Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);
//-----

//const int DS18B20_PIN = D3;
const int AirValue = 620;
const int WaterValue = 310;
int soilMoistureValue = 0;
int soilmoisturepercent=0;
String DBnm = "Sensors";
String TD = "Temperature";
String HD = "Humidity";
String SM = "Soil Moisture";
String ST = "Soil Temperature";
boolean state = false;

WiFiClient client;

void setup() {
  Serial.begin(115200);
  pinMode(D6,OUTPUT);
  dht.begin();
  sensors.begin();
  Blynk.begin(auth, ssid, pass);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
  timeClient.begin();
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop() {
  delay(5000);
  Blynk.run();

  //read temperature and humidity
  float t = dht.readTemperature();

```

```

float h = dht.readHumidity();

if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
}

soilMoistureValue = analogRead(A0);
Serial.println(soilMoistureValue);
soilmoisturepercent = map(soilMoistureValue, AirValue, WaterValue, 0, 100);

sensors.requestTemperatures();
float soiltemp = sensors.getTempCByIndex(0);

/*          OLED SCREEN          */
display.clearDisplay();

// display temperature
display.setTextSize(1);
display.setCursor(0,0);
display.print("Air Temp: ");
display.setTextSize(1);
display.setCursor(80,0);
display.print(t);
display.print(" ");
display.setTextSize(1);
display.cp437(true);
display.write(167);
display.setTextSize(1);
display.print("C");

// display humidity
display.setTextSize(1);
display.setCursor(0, 10);
display.print("Air Humid: ");
display.setTextSize(1);
display.setCursor(80,10);
display.print(h);
display.print(" %");

//display soil moisture
display.setTextSize(1);
display.setCursor(0, 20);
display.print("Soil moist: ");
display.setTextSize(1);
display.setCursor(80,20);
display.print(soilmoisturepercent);
display.print(" %");

//display soil temperature
display.setTextSize(1);

```

```

display.setCursor(0, 30);
display.print("Soil Temp:");
display.setTextSize(1);
display.setCursor(80,30);
display.print(soiltemp);
display.print(" C");

display.display();

/*          Blynk          */

Blynk.virtualWrite(V10, t);
Blynk.virtualWrite(V11, h);
Blynk.virtualWrite(V12, soilmoisturepercent);
Blynk.virtualWrite(V13, soiltemp);

//PUSH Notification

if(soilmoisturepercent < 30){
  Blynk.notify("Alert!!! - Soil Moisture under 30%");
}

/*          ThingSpeak          */
if (client.connect(server,80)) {
  String postStr = apiKey;
  postStr += "&field1=";
  postStr += String(t);
  postStr += "&field2=";
  postStr += String(h);
  postStr += "&field3=";
  postStr += String(soilmoisturepercent);
  postStr += "&field4=";
  postStr += String(soiltemp);
  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);

}

/*          FIREBASE          */

```

```

//-----Get time from the internet and format the
display.
// Without the conditions below, the time display will be like this: 1:1:1, 1:50:5
// With the conditions below, the time display will be like this: 01:01:01, 01:50:05
timeClient.update();
String hr, mn, sc;
if (timeClient.getHours() < 10) {
    hr = "0" + String(timeClient.getHours());
}
else {
    hr = String(timeClient.getHours());
}

if (timeClient.getMinutes() < 10) {
    mn = "0" + String(timeClient.getMinutes());
}
else {
    mn = String(timeClient.getMinutes());
}

if (timeClient.getSeconds() < 10) {
    sc = "0" + String(timeClient.getSeconds());
}
else {
    sc = String(timeClient.getSeconds());
}

String TimeNow = hr + ":" + mn + ":" + sc;
Serial.print(TimeNow);

String strHum = String(h); //--> Convert Humidity value to String data type.
String strTem = String(t); //--> Convert Temperature values to the String data type.
String strSm = String(soilmoisturepercent);
String strSt = String(soiltemp);

//-----Send Humidity data to the Firebase Realtime
Database.
String DBaddH = DBnm + "/" + TimeNow + "/" + HD; //--> Creating a Database
path
Firebase.setString(DBaddH,strHum); //--> Command or code for sending Humidity
data in the form of a String data type to the Firebase Realtime Database.

// Conditions for handling errors.
if (Firebase.failed()) {
    Serial.print("setting Humidity failed :");
    Serial.println(Firebase.error());
    delay(500);
    return;
}

```

```

}
//-----

//-----Send Temperature data to the Firebase Realtime
Database.
String DBaddT = DBnm + "/" + TimeNow + "/" + TD; //--> Creating a Database
path
Firebase.setString(DBaddT,strTem); //--> Command or code for sending
Temperature data in the form of a String data type to the Firebase Realtime Database.

// Conditions for handling errors.
if (Firebase.failed()) {
    Serial.print("setting Temperature failed :");
    Serial.println(Firebase.error());
    delay(500);
    return;
}
//-----

//-----Send Humidity data to the Firebase Realtime
Database.
String DBaddSm = DBnm + "/" + TimeNow + "/" + SM; //--> Creating a Database
path
Firebase.setString(DBaddSm,strSm); //--> Command or code for sending Humidity
data in the form of a String data type to the Firebase Realtime Database.

// Conditions for handling errors.
if (Firebase.failed()) {
    Serial.print("setting Soil Moisture failed :");
    Serial.println(Firebase.error());
    delay(500);
    return;
}
//-----

//-----Send Humidity data to the Firebase Realtime
Database.
String DBaddSt = DBnm + "/" + TimeNow + "/" + ST; //--> Creating a Database
path
Firebase.setString(DBaddSt,strSt); //--> Command or code for sending Humidity
data in the form of a String data type to the Firebase Realtime Database.

// Conditions for handling errors.
if (Firebase.failed()) {
    Serial.print("setting Soil Temperature failed :");
    Serial.println(Firebase.error());
    delay(500);
    return;
}
//-----

```

```
Serial.println("Setting successful");
Serial.println();
```

```
delay(5000);
```

```

/*                WATER PUMP                */
if(soilmoisturepercent < 30) // change this at what level the pump turns on
{
  Blynk.notify("Nearly dry, Pump turning on");
  digitalWrite(D6,LOW); // Low percent high signal to relay to turn on pump
}
else if(soilmoisturepercent > 85) // max water level should be
{
  Blynk.notify("Nearly wet, Pump turning off");
  digitalWrite(D6,HIGH); // high percent water high signal to relay to turn off pump
}
}

BLYNK_WRITE(V1)
{
  if (state == false) {
    state = true;
    Blynk.notify("You just watered your plant.");
    digitalWrite(D6,LOW);
    delay(1000);
  }
  else {
    state = false;
    digitalWrite(D6,HIGH);
  }
}
}

```