

**IOT SMART AGRICULTURE AND
SMART IRRIGATION SYSTEM**

TAY YONG TANG

**A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Engineering (Honours) Electronic Engineering**

**Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman**

May 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Universiti Tunku Abdul Rahman or other institutions.

Signature :



Name : TAY YONG TANG

ID No. : 18AGB04717

Date : 12 MAY 2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“IOT SMART AGRICULTURE AND SMART IRRIGATION SYSTEM”** was prepared by **TAY YONG TANG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Dr. LEE YU JEN

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2022, Tay Yong Tang. All right reserved.

Specially dedicated to
my beloved mother, father, brothers and sister.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to everyone for their contributions to the successful completion of my project titled IoT smart agriculture and smart irrigation systems. I would like to express my special thanks to my project supervisor, Dr. Lee Yu Jen, for his invaluable advice, guidance, and enormous patience throughout the development of the project.

Next, I would also like to take this opportunity to express my gratitude to my loving parents, who have helped and given me encouragement when I am facing difficulties during the development of the project. In addition, I would like to thank my classmate, Lee Kar Tien, for the information exchange on the project.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Signature :

A handwritten signature in black ink, appearing to be 'TAY YONG TANG', written in a stylized, cursive script.

Name : TAY YONG TANG

IOT SMART AGRICULTURE AND SMART IRRIGATION SYSTEM

ABSTRACT

Indoor plantation is popular among every household to keep people enjoying living in a green space, keep their house fresh, and most like having gorgeous plants around. However, people nowadays have hectic schedules. It is difficult for them to pay more attention to their plants at home while they are working. In this project, an IoT smart agriculture and smart irrigation system is developed. This system is able to measure air temperature and air humidity by using the DHT22 sensor; soil moisture in two different areas by using two capacitive soil moisture sensors; and the water level in the water tank by using an ultrasonic sensor. The Node MCU ESP32 DEVKIT V1 DOIT development board is used in this project to interface with the Blynk server. The conditions of the plants are sent to the cloud and displayed on the smartphone via the Blynk IoT application. Besides that, the conditions can also be displayed physically by using the OLED display. Then, this system includes a smart irrigation system that is operated with the stepper motor, the water pump, and the automations that can be activated in the Blynk IoT app. When the soil moisture is fall below 30 %, a notification is sent to the smartphone, and the irrigation system is activated. In addition, another automation is applied to detect the water level of the water tank and send a notification when the water level is below 15 %. After that, the reliability of this system is analysed, and the overall performance of the display system, irrigation system, and automation achieved above 90 % accuracy. The energy saving efficiency analysis shows that the smart irrigation system is able to save 80.65 % of water when the plants are placed in a cold environment and 67.74 % of water when the plants are placed in a room temperature environment. Lastly, the improvement of this project is suggested by implementing the camera and artificial intelligence in this system.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	vi
ABSTRACT	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xii
LIST OF FIGURES	xiv
LIST OF CODING LISTING	xviii
LIST OF SYMBOLS / ABBREVIATIONS	xix
LIST OF APPENDICES	xxi

CHAPTER

1	INTRODUCTION	1
	1.1 Project Background	1
	1.2 Problem Statements	4
	1.3 Aims and Objectives	4
	1.4 Chapter Outline	5
	1.5 Summary	5
2	LITERATURE REVIEW	7
	2.1 Introduction	7
	2.2 Commercial Products in The Market	7
	2.2.1 Libelium	8
	2.2.2 CropX	9

2.2.3	PRECISIONHAWK	11
2.3	Non-Commercial Projects	11
2.3.1	IoT Automatic Plant Watering System	12
2.3.2	IoT Smart Agriculture & Automatic Irrigation System with ESP8266	13
2.3.3	Avengers Plant Monitoring Device	14
2.4	Overview of Development board	15
2.4.1	Introduction to Arduino	16
2.4.2	Introduction to Node MCU	18
2.4.3	Introduction to Raspberry Pi	20
2.5	Hardware for Soil Moisture Detection	21
2.5.1	Capacitive Soil Moisture Sensor	22
2.5.2	Conductivity Sensor	23
2.5.3	Soil Moisture & Temperature & EC Sensor	24
2.6	Hardware for Humidity and Temperature Detection	25
2.6.1	DHT Sensors	25
2.6.2	DS18B20 Temperature sensor	26
2.6.3	Barometric Sensors	27
2.7	IoT Platform	28
2.7.1	Google Cloud Platform	28
2.7.2	AWS IoT Core	29
2.7.3	Blynk	30
2.8	Summary	31
3	METHODOLOGY	32
3.1	Introduction	32
3.2	Development Board Selection	33
3.3	IoT Monitoring System	34
3.3.1	IoT Platform Selection	34
3.3.2	Soil Moisture Sensor Selection	36
3.3.3	Humidity and Temperature Sensor Selection	38
3.3.4	Ultrasonic Sensor	38
3.4	Irrigation System	39

3.4.1	Irrigation System Model	40
3.4.2	Water Pump	41
3.4.3	Stepper Motor	42
3.5	System Working Principle	43
3.5.1	Block Diagram of the Complete System	44
3.5.2	Operating Flow of Physical Display System	45
3.5.3	Operating Flow of IoT Monitoring System	47
3.5.4	Operating Flow of Irrigation System	49
3.5.5	Operating Flow of Smart Irrigation System	51
3.5.6	Operating Flow of the Complete System	52
3.5.7	Pin Allocation	54
3.6	Cost Estimation	57
3.7	Project Management	59
3.8	Summary	60
4	RESULTS AND DISCUSSIONS	62
4.1	Introduction	62
4.2	Preliminary Result	62
4.3	Hardware Implementation	64
4.3.1	Physical Display System	65
4.3.2	IoT Monitoring System	66
4.3.3	Irrigation System	67
4.3.4	Prototype	68
4.3.5	Prototype Board and Printed Circuit Board	68
4.4	Software Implementation	70
4.4.1	User Interface	70
4.4.2	Automations	72
4.5	System Analysis	74
4.5.1	Humidity and Temperature Detection Ranges Analysis	74
4.5.2	Soil Moisture Detection Range Analysis	75
4.5.3	Display System Analysis	76
4.5.4	Irrigation System Analysis	78

4.5.5	Automations Analysis	80
4.6	Efficiency on Water Saving Analysis	82
4.6.1	Case Study 1: Cold Environment	82
4.6.2	Case Study 2: Room Temperature Environment	83
4.6.3	Total Water Saving	84
4.7	Cost Analysis	85
4.8	Summary	88
5	CONCLUSION AND RECOMMENDATIONS	89
5.1	Conclusion	89
5.2	Limitations	90
5.3	Recommendations for Improvement	91
	RERERENCES	92
	APPENDICES	97

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	Comparison between Arduino UNO and Arduino UNO Wi-Fi (Arduino.cc.,2018)	17
2.2	Comparison between Node MCU ESP32 and Node MCU ESP8266 (Ashwak, 2021)	19
2.3	The Raspberry Pi 2 Model B and The Raspberry Pi 3 Model B+ (www.pololu.com, n.d.)	21
2.4	DHT11 and DHT22 Specifications (Random Nerd Tutorials, 2019).	26
2.5	DS18B20 Temperature Sensor Specifications (Maxim Integrated Products, 2019)	27
2.6	BME280 Sensor Specifications (BME280 -Data sheet, 2018)	28
3.1	Virtual Pins Allocation in Blynk	35
3.2	Pin Definition of NodeMCU ESP32	55
3.3	Pins Allocation of NodeMCU ESP32 Board	56
3.4	Cost Estimation of This Project	57
3.5	Gantt Chart of FYP 1	59
3.6	Gantt Chart of Short Semester	59
3.7	Gantt Chart of FYP 2	60
4.1	Readings of the DHT22 Sensor	74
4.2	Readings of the Capacitive Soil Moisture Sensor	75
4.3	Performance of OLED display and Virtual Display	77

4.4	Performances of Physical LEDs and Virtual LEDs	77
4.5	Accuracy of the Display System	78
4.6	Performances of the Irrigation System	79
4.7	Accuracy of the Irrigation System	80
4.8	Performances of the Automations	81
4.9	Accuracy of the Automations	82
4.10	Water Usage and Saving	85
4.11	Cost Analysis of the Entire Project	86

LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	Sketch Model of the Irrigation System	2
2.1	Smart Agriculture Xtreme (Libelium, n.d.)	8
2.2	Smart Agriculture PRO (Libelium, n.d.)	9
2.3	Dashboard on Mobile App of CropX Technology (CropX, n.d.)	10
2.4	Dashboard on Desktop App of CropX Technology (CropX, n.d.)	10
2.5	Drone with Sensors (PRECISIONHAWK, n.d.)	11
2.6	Watering System Model (Orion, n.d.)	12
2.7	Dashboard of the Blynk IoT Mobile App (Orion, n.d.)	13
2.8	IoT Smart Agriculture & Automatic Irrigation System with ESP8266 (Parajuli, 2022)	14
2.9	Dashboard of the Blynk IoT Mobile App (Parajuli, 2022)	14
2.10	Avengers Plant Monitoring Device (Vishalsoniindia, n.d.)	15
2.6	Raspberry Pi (Monk, 2016)	20
2.7	Capacitive Soil Moisture Sensor (Shawn, 2010)	22
2.8	Hardware Schematic of the Capacitive Soil Moisture Sensor (Alam, 2019)	22
2.9	Resistive Soil Moisture Sensor (Shawn, 2010)	23

2.10	Plot of the Electrical Resistance of a Soil Sample (Antonio, 2021)	24
2.11	Soil Moisture, Temperature and Electrical Conductivity (EC) Sensor (Antonio, 2021)	24
2.12	Logo of Google Cloud Platform (Google, 2019)	29
2.13	Logo of AWS IoT Core Platform (Amazon Web Services, Inc., n.d.)	30
2.14	Logo of Blynk Platform (Blynk.io, 2015)	30
3.1	ESP32 Function Block Diagram (Esp32.net, 2016)	33
3.2	Blynk IoT App	35
3.3	Defining the Input and Output	35
3.4	Accuracy of Capacitive Soil Moisture Sensors (SMEC300 and SM100) and Resistive Soil Moisture Sensor (YL69 and YL100) (Adla et al., 2020)	37
3.5	Overview of Capacitive Soil Sensor (wiki.seeedstudio.com, n.d.)	37
3.6	DHT 22 Sensor Module	38
3.7	HC-SR04P Ultrasonic Sensor (Cytron Technologies Malaysia, n.d.)	39
3.8	HC-SR04 Ultrasonic Sensor (Cytron Technologies Malaysia, n.d.)	39
3.9	Sketch Model of Irrigation System	40
3.10	3D Sketch of Belt Holder	40
3.11	3D Sketch of Motor Holder	40
3.12	3D Sketch of Pulley Holder	41
3.13	3D Sketch of Linear Slider	41
3.14	R385 DC12 V Diaphragm Water Pump (Cytron Technologies Malaysia, n.d.)	42
3.15	L298N Motor Driver (Cytron Technologies Malaysia, n.d.)	42

3.16	12 V 28BJ-48 Stepper Motor (Cytron Technologies Malaysia, n.d.)	42
3.17	ULN2003 Driver Board (Cytron Technologies Malaysia, n.d.)	43
3.18	Block Diagram of the Complete System	45
3.19	Flowchart of Physical Display System	47
3.20	Flowchart of IoT Monitoring System	49
3.21	Flowchart of Irrigation System	50
3.22	Flowchart of Smart Irrigation System	52
3.23	Flowchart of the Complete System	53
3.24	Pin Configuration of NodeMCU ESP32 Board (Ashwak, 2021)	54
4.1	Hardware Connection on the Breadboard	63
4.2	Contents of the OLED Display	63
4.3	User Interface on the Blynk IoT App	64
4.4	Hardware of the Physical Display System	65
4.5	Operation of the Status Indicator LEDs	66
4.6	Hardware of the IoT Monitoring System	66
4.7	Hardware of the Irrigation System	67
4.8	Position of the Limiter Switch	67
4.9	Prototype of the Smart Agriculture and Smart Irrigation System	68
4.10	Prototyped Board	69
4.11	Printed Circuit Board	69
4.12	Web Dashboard on the Blynk Concole	71
4.13	Operation of the Virtual LEDs	71
4.14	Mobile Dashboard on the Blynk IoT App	72

4.15	Automation Page on the Blynk IoT App	73
4.16	Notifications of the Automation	73
4.17	Marking of the Target Distance	79
4.18	Soil Moisture Percentage in Area B from 26 February to 3 March	83
4.19	Soil Moisture Percentage in Area B from 6 April to 9 April	83

LIST OF CODING LISTING

CODE LISTING	TITLE	PAGE
1	Code to Set Connection Between ESP32 Development Board	70

LIST OF SYMBOLS / ABBREVIATIONS

%	Percentage
°C	Degree Celsius
A	Ampere
gf	Gram Force
Hz	Hertz
L	Litre
m	Meter
min	Minute
Pa	Pascal
s	Second
V	Voltage
V _{CC}	Voltage common collector
ADC	analogue to digital converter
AI	artificial intelligence
APP	application
AWS	amazon web service
BLE	bluetooth low energy
DAC	digital to analogue converter
DC	direct current
DHT	digital humidity and temperature
GPIO	general purpose input output
HDMI	high-definition multimedia interface
HSPI	high-speed parallel interface
HTTP	hypertext transfer protocol
IDE	integrated development environment
LED	light-emitting diode

I	input
I/O	input and output
IoT	internet of things
I2C	inter-integrated circuit
I2S	inter-IC sound
MCU	micro-controller unit
MQTT	message queuing telemetry transport
OLED	organic light-emitting diode
OTP	one-time password
P	power
PCB	printed circuit board
PVC	polyvinyl chloride
PWM	pulse width modulation
RAM	random access memory
RTOS	real-time operating system
SBC	single board computer
SCL	serial clock line
SD	secure digital
SDA	serial data line
SDK	software development kit
SOC	system on chip
SPI	serial peripheral interface
TV	television
UART	universal asynchronous receiver/transmitter
ULP	ultra low power
USB	universal serial bus
Wi-Fi	wireless fidelity

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Coding for the IoT Smart Agriculture and Smart Irrigation System	97
B	Printed Circuit Board Design	148

CHAPTER 1

INTRODUCTION

1.1 Project Background

Indoor plantation is popular among every household to keep people enjoying living in a green space, and most like having gorgeous plants around. Some people also aim to keep their house fresh with the indoor plantation, as the green plants are able to cycle the air during the day. In addition, some people also take up indoor planting as a hobby. However, people nowadays have hectic schedules. It is difficult for them to pay more attention to their plants at home while they are working.

There is a smart irrigation system on the market that is able to water the plants automatically, but the conditions of the plants, such as the moisture of the soil, the humidity of the air, and the temperature around the plants, are unable to be detected (Nermin et al., 2017). This raises the issue that people may not water their plants with the right amount of water. Thus, a new solution may be needed to solve this problem. In the years from 2010 to 2011, the Internet of Things (IoT) had been gaining huge popularity as people started to debate that Google was trying to index the physical world by using IoT (Lasse and L.K, 2014). IoT is an automation system that is able to perform networking, sensing, or artificial intelligence to interface a computer system with a device (tutorialspoint, 2016). So, the decision to design an IoT-based plant monitoring system with a smart irrigation system has been made in this project.

In the study of previous projects or products on the market, most of the IoT systems and the smart irrigation systems are separated. Most products on the market

to monitor plants are intended for use in a large field, such as for crops, and are therefore unsuitable for a small-scale plantation, such as an indoor plantation. CropX, for example, has a cloud-based decision support tool that provides users with appropriate plantation decisions and plans based on continuous monitoring of soil and crop conditions (Cropx, n.d.). So, this project plans to develop an IoT system that is able to detect air temperature, air humidity around the plants, and soil moisture, as referred to by the products on the market, but on a smaller scale that is suitable for indoor plantations.

Besides that, for the smart irrigation system, most of the products or projects are designed in such a way that the water is spread out from the ground. This may be unsuitable for indoor plantations as the water may spread around the house, which may lead to accidents. To address this issue, this project intends to create a model that can be watered from the top of the plant and has a fixed watering range, as shown in Figure 1.1. The irrigation system is able to water the plants from the top equally along the fixed area with a bar that is able to move in the X direction, where the water tube is attached to the bar. The design concept of this irrigation system model is inspired by the project done by Orion Maker (Orion, n.d.).

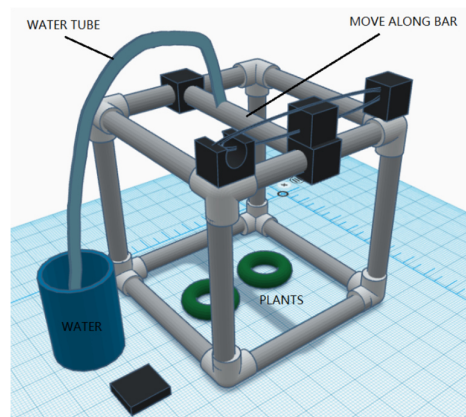


Figure 1.1: Sketch Model of the Irrigation System

Next, in order to connect to the cloud, the microcontroller or hardware control device must be able to connect to the internet. After comparing the various microcontrollers on the market, this project considers using the Node MCU ESP32 as

the control device that has internet connectivity functionality and is able to connect to the IoT cloud via the internet. There are a lot of functions available in Blynk, which is the IoT cloud platform that is used in this project. The language used to develop the connectivity between the development board and the cloud is user friendly. The condition of the plants, such as soil moisture, air humidity, and air temperature, can be stored in the IoT cloud for three months, providing users with the ability to monitor the plants easily. In this project, the dashboard is built using the Blynk IoT app, which is available on smartphones and computers. The design of the input and output to collect data from the plants and display it on the smartphone or computer is done in this project using the Blynk IoT app on the smartphone and the Blynk console on the computer.

In addition, the air humidity and the air temperature are collected using the DHT22 sensor module, and the soil moisture in two different areas is collected using the capacitive soil moisture sensor. Some LEDs to indicate the stepper motor that moves the moving bar and the status of the water pump that pumps the water to the moving bar are included in this project. At the same time, the statuses are also displayed on the Blynk IoT app. An OLED display is also added in this project to display the air temperature, air humidity, and soil moisture physically, even though they have been shown on the Blynk IoT app, to allow users to monitor the plants physically when needed. A button is also included to give an instant watering instruction either physically or remotely in the Blynk app, so that watering can be performed instantly when needed.

Moreover, a 12 V priming diaphragm pump spray motor is used to pump the water to the moving bar with small holes to water the plants, and it is controlled by using the L298N motor driver board. The L298N is able to control the power of the water pump, and this board also contains a heat sink, which is able to prevent it from getting heated easily as it drives a 12 V motor. A 12 V stepper motor is used in this project to move the bar along the X direction and is controlled by the ULN2003 driver board. Last but not least, this system is powered by using 12 V, as the motors used are 12 V. A 12 V power adapter is used as the power supply. Since the power supply is 12 V, this project has to use an LM2596 stepdown power module to limit the voltage supply to the Node MCU ESP32, which needs only 5 V.

1.2 Problem Statements

Ideally, people having indoor plants will be able to take care of their plants well as they can pay more attention to their plants around their home. However, they have a busy lifestyle in reality, which causes them to be unable to look after their plants, even if the plants are just around them, as they need to rush to their workplace and stay there for the whole day. This leads to the death of plants because of a lack of monitoring and watering. Research has shown that applying a smart irrigation system will enable us to water the plants automatically every day to keep them alive. Even so, a problem that still has not been solved completely, is the unknown condition of the plants. Thus, the idea that IoT can be applied to this project to collect data from the plants and send it to the IoT cloud for feedback to the users has emerged.

This project is going to develop an IoT smart agriculture and smart irrigation system. This system is able to measure air temperature, air humidity, and soil moisture. The condition of the plants will be sent to the cloud, and then feedback will be sent to the users via their phones. The users will be able to react to and measure the condition of their plants from anywhere in the world. Then, this system includes a smart irrigation system that automatically waters the plants without the user's engagement.

1.3 Aims and Objectives

The objectives of this project are shown as follow:

- i) To design a system that is able to detect and measure the soil moisture, air temperature and air humidity.
- ii) To develop a device that is able to interact with IoT cloud via internet so that monitoring of plant conditions can be done from an isolated place.
- iii) To achieve an automatic watering system.
- iv) To customize an app showing the condition of the plants.

1.4 Chapter Outline

The introductory chapter introduces the background of the project, including its idea, overview, problem statement, and objectives. Next, Chapter 2 highlights the relevant literature. The literature reviewed includes commercial and non-commercial products related to this project, development boards, soil moisture detection hardware, humidity and temperature detection hardware, and the IoT platform. After that, Chapter 3 justifies the research methodology by explaining the selection of components to be applied to the project, introducing the working principle of this system, and describing cost estimation and project management. Then, Chapter 4 shows and discusses the system development results for hardware and software. Besides that, several analyses are also performed and recorded to evaluate the reliability of the system in Chapter 4. The analyses are system analyses, energy saving efficiency analysis, and cost analysis. Lastly, Chapter 5 concludes this project. The limitations and recommendations for improvement of the system are also included in Chapter 5.

1.5 Summary

This chapter has mentioned that indoor plantations are popular among every household nowadays as a hobby or way to create a green space at home. However, due to some reasons, the owner of the plant is unable to take care of it, leading to its death. This is the problem that needs to be solved in this project. Besides that, this chapter has also mentioned that the IoT is applied to this system to perform networking and sensing to interface computer systems and devices.

After that, the idea of designing an IoT-based plant monitoring system with a smart irrigation system has been discussed in this chapter. This system is able to detect the air humidity, air temperature, and soil moisture of plants and send these conditions to the IoT cloud, which then show up on the owner's app on the phone. This chapter also briefly introduces the components that are used in this system. In addition, this chapter has listed several objectives to frame this project. Lastly, the outline of every chapter has been included in this chapter to provide the roadmap of this report.

The next chapter, Chapter 2, will discuss the relevant literature, review research on commercial and non-commercial products from others, and provide information on every choice of component used in this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, literature reviews are conducted to gain an understanding of existing research that relates to this project. Literature reviews are important to provide knowledge and concepts for this project. This project is able to gain a better understanding of how findings are presented and discussed. Thus, this project has reviewed the commercial products on the market, the non-commercial project, the development board, the moisture detection hardware, the humidity and temperature detection hardware, and the IoT platform. Three products or projects are included for every category that is stated above. The details, pros, and cons of each item reviewed are discussed and highlighted in this chapter.

2.2 Commercial Products in The Market

A commercial product corresponds to a product created by humans and developed from project results that is capable of being sold, registered, relocated, or otherwise discharged, whether for payment or not. In short, consumers need to pay to use the products. Normally, the product cost is high but supported by higher technology. This section lists some products from different companies. The advantages and disadvantages of each product are discussed.

2.2.1 Libelium

Libelium is a company that offers a complete IoT technology (Libelium, n.d.). This company offers smart agriculture products such as Plug & Sense. This product consists of two models, which are Smart Agriculture Xtreme that costs around RM 60000 as shown in Figure 2.1 and Smart Agriculture PRO that costs around RM 5000 as shown in Figure 2.2. This product is designed with more than 20 high-end sensors to monitor the plants and built with an antenna to transfer information to their own Libelium Cloud. It allows the user to monitor a lot of parameters, including the condition of plants, soil monitoring, plant growth analysis, humidity, temperature, atmospheric monitoring, and weather observation.

The advantages of this product are that it is designed with a lot of sensors that are able to collect different environmental data in order to give better results in monitoring the plants; the high-end sensors provide high-accuracy data for monitoring the plants; automation of data collection; real-time information for daily monitoring anywhere in the world; weather observation function; and it is able to be applied in a very large-sized plant field. The disadvantages of this product are that it is designed without an automatic irrigation system that is able to water the plants automatically, and it is also not suitable for a small-scale plantation or a home plantation, as some features may seem to be extra and unnecessary. Besides that, this product is also expensive for household use, which is why it's built with all kinds of high-tech sensors.



Figure 2.1: Smart Agriculture Xtreme (Libelium, n.d.)



Figure 2.2: Smart Agriculture PRO (Libelium, n.d.)

2.2.2 CropX

CropX is a cloud-based decision support tool company that offers integrated hardware and software systems with a suite of decision and planning tools based on continuous monitoring of soil and crop conditions (CropX, n.d.). These planning and reporting tools allow users to record and schedule farm activities at the same time that they monitor the crop's health and growth.

The advantages of this product include: providing a dashboard on a mobile as shown in Figure 2.3 or desktop app as shown in Figure 2.4 that is able to give advice to the farmers after analysing the data from the soil sensor satellites and field properties; obtaining precise weather information including air temperature, humidity, wind speed, min and max temperatures, and more; having a deep depth moisture sensor that accurately captures the of the soil's water content; and being able to collect images of the crops from several satellites and renew them every two to three days on average to observe the crop conditions. The disadvantages of this product are that it is only suitable for very large-scale plantations, and the cost of owning this tool is also high.



Figure 2.3: Dashboard on Mobile App of CropX Technology (CropX, n.d.)

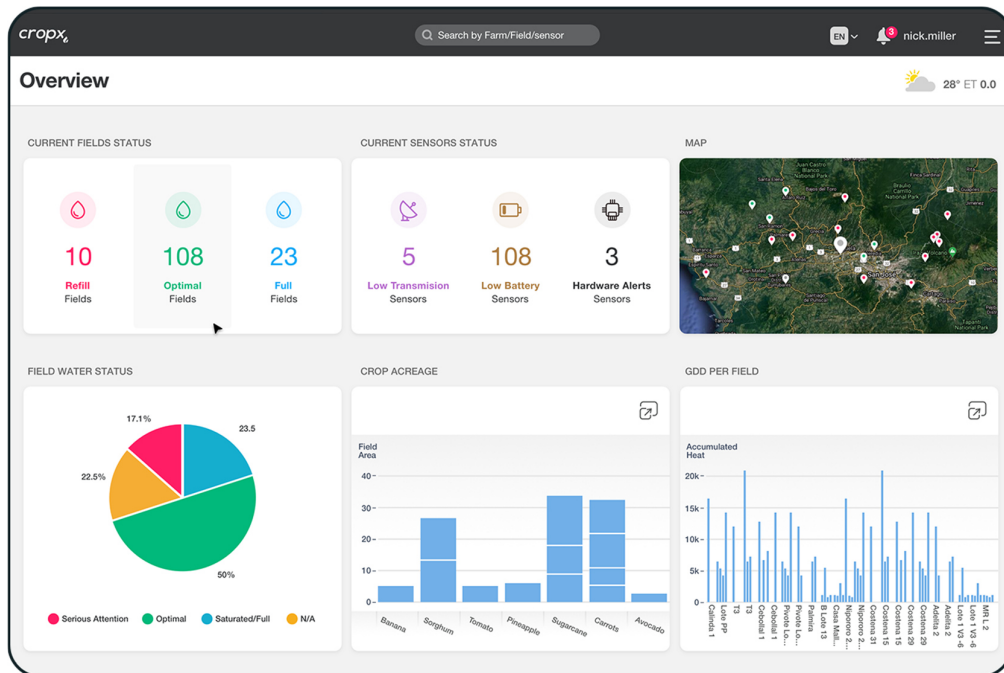


Figure 2.4: Dashboard on Desktop App of CropX Technology (CropX, n.d.)

2.2.3 PRECISIONHAWK

PrecisionHawk is a company that has a drone-based agriculture solution that includes drones, sensors, and its own web-based portal designed for users. Drones will be deployed by an array of sensors and collect crop data as shown in Figure 2.5. This data will be sent to a web-based portal for analysis and feedback to users on the action to take on their crop (PrecisionHawk, n.d.).

The advantages of this product are that it comes with a visual sensor to collect the livestock data and a multispectral sensor that enables the user to capture radiometric thermal data in order to get visibility into the crop health and have a clear view from the top to monitor the plant. However, the disadvantages of this product are that it is only suitable during the daytime with no rainy days, it is unable to truly detect the soil condition of the plant, and it is unable to monitor the plant while in an isolated place.



Figure 2.5: Drone with Sensors (PRECISIONHAWK, n.d.)

2.3 Non-Commercial Projects

A non-commercial project is one undertaken by an individual with no primary profit motive. A non-commercial project is created as a hobby by an individual. Most of the non-commercial projects are cheaper than commercial projects because they have no profile. At the same time, the technology involved in the project will also be lower than in commercial projects to save costs. Most of the non-commercial projects are

suitable for house use only. This section includes the related, non-commercial projects that are carried out by students or professionals. The pros and cons of these projects are discussed.

2.3.1 IoT Automatic Plant Watering System

A watering system model was constructed using a total of 13 pieces of PVC pipe; 12 pieces were built in the rectangular shape as the system's body, and 1 piece was used at the top that can move along and provide watering action as shown in Figure 2.6. This system uses the NodeMCU ESP8266 as the microcontroller to interact with the Blynk IoT to perform actions on the system. The dashboard in the mobile app is shown in Figure 2.7. The actions include a timer set by the user to water the plants and instant watering (Orion, n.d.).

The system's advantages include the ability to automate the watering action on the plant during a set time, the ability for the user to control the watering action on the plant using the phone while in an isolated location, the display of the system's movement status, and the watering of the plant is done equally from the top, preventing water flooding in a specific area only. However, this system is unable to let the user monitor the condition of the plants. The controlling action also cannot be done if the internet is disconnected.

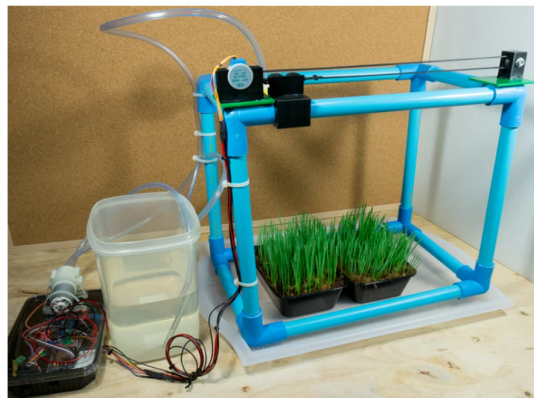


Figure 2.6: Watering System Model (Orion, n.d.)

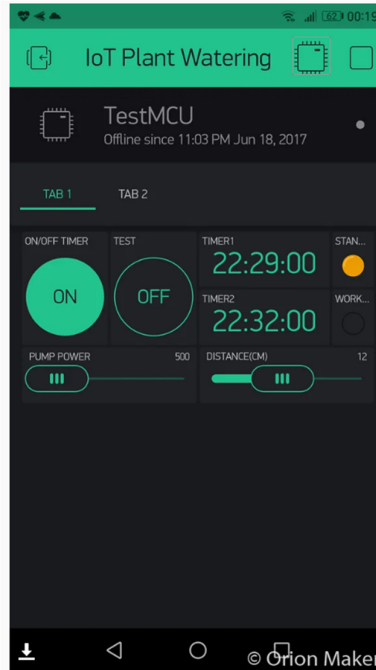


Figure 2.7: Dashboard of the Blynk IoT Mobile App (Orion, n.d.)

2.3.2 IoT Smart Agriculture & Automatic Irrigation System with ESP8266

The project is using the Node MCU ESP8266 to create an IoT-based system to monitor the plants as shown in Figure 2.8. This system is also connected with several sensors to detect the soil moisture level, air temperature and humidity, soil temperature, motion activity and rain status. There is also an automatic irrigation system that controls the water pump to water the plant when soil moisture reaches a certain threshold. The system will feedback the data to the user via the Blynk IoT platform (Parajuli, 2022). The dashboard of Blynk IoT mobile app showing the data of the plants is shown in Figure 2.9.

The advantage of the system built is that it automates the watering action on the plant when soil moisture reaches a certain threshold. This system is also able to display the status of the plant, so that users are able to monitor it using their phones while in an isolated place. The disadvantage of this system is that users are unable to

control the watering system manually when needed. The watering action of straight tubes may result in uneven watering or flooding of the plant.

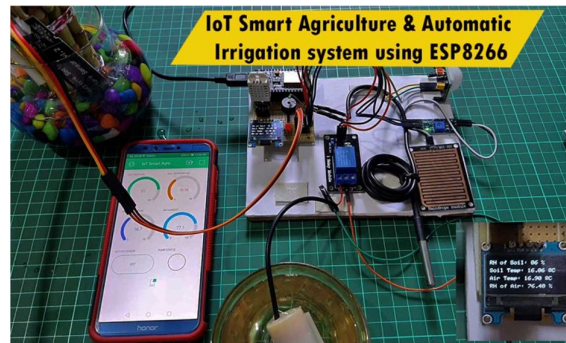


Figure 2.8: IoT Smart Agriculture & Automatic Irrigation System with ESP8266 (Parajuli, 2022)

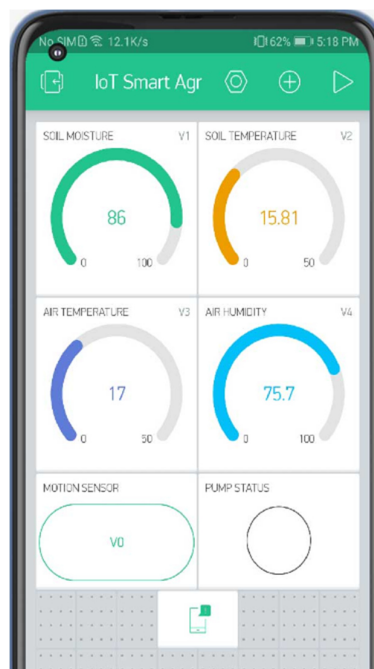


Figure 2.9: Dashboard of the Blynk IoT Mobile App (Parajuli, 2022)

2.3.3 Avengers Plant Monitoring Device

The project is done by using an Arduino Uno with a wireless fidelity (Wi-Fi) module board. The board used is the Tuya CBU board, which was developed by Tuya Smart.

Tuya Smart also has their own IoT cloud to provide the interfacing of data with the phone. This project includes two resistance sensors that are able to measure soil moisture from two plants and also uses Tuya sensors for measuring temperature and humidity (Vishalsoniindia, n.d.). Figure 2.10 shows the plant monitoring system with the data display on the smartphone.

The advantages the system provides are: very cheap in terms of cost to build the system; able to display the conditions of the plant on the phone, so that the user is able to monitor it using the phone while in an isolated place; a customised dashboard can be done on the computer and interacted with the phone using the IoT; and a larger area of soil moisture data can be collected through the use of two soil moisture sensors. However, this system uses a fixed IoT device, which means all technologies used must come from the company. The system is also without an automatic watering system that is able to water the plant easily.



Figure 2.10: Avengers Plant Monitoring Device (Vishalsoniindia, n.d.)

2.4 Overview of Development board

The development board is a printed circuit board that incorporates the target microcontroller as well as some hardware. It helps this project program the processor onboard effectively as the hardware of the development board has been pretested by the manufacturers. Thus, this project is able to develop and test the project efficiently. In this section, various types of development boards are introduced. They include the

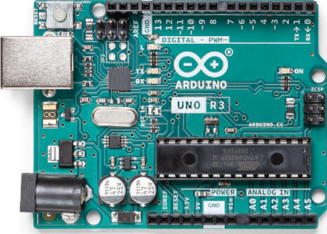
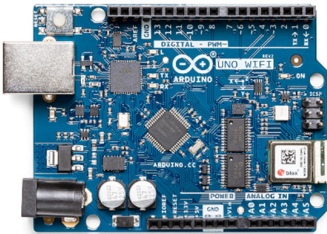
Node Microcontroller Unit (Node MCU), the Raspberry Pi, and the Arduino. The basic introduction of these three common development boards is discussed in this section.

2.4.1 Introduction to Arduino

Arduino is a user-friendly hardware and software-based open-source electronic platform. The Arduino board acts like a mini-computer that is able to interact between the computer and external hardware. A user is able to send some instructions to the microcontroller in the Arduino board to carry out some action in a simple programming language, such as C++ or C. For instance, the Arduino board is able to read the input from a sensor and convert it into an output, such as activating a motor or turning on an light-emitting diode (LED) (Arduino.cc., 2018).

The benefits of using Arduino include its low cost, cross-platform compatibility, simple programming environment, open source, and extensibility in both software and hardware. Arduino is inexpensive because the lowest price of an Arduino board can go up to RM 25 only, such as the Arduino Uno. Arduino is able to work on different operating systems such as Windows and Linux, making it cross-platform. Arduino also comes with its own integrated development environment (IDE), which can be programmed in C or C++ and provides a straightforward programming environment. However, there are some disadvantages to using Arduino in IoT projects. They do not include internet connectivity or the expensive Arduino Wi-Fi board. Arduino boards such as Arduino Uno do not have built-in support for wireless networks and Wi-Fi modules may need to be added in order to get a Wi-Fi connection (Yuan, 2017). The Arduino UNO Wi-Fi board is seen to be an expensive board as the price is up to RM 259. The comparison between the Arduino UNO board and the Arduino UNO Wi-Fi board is shown in Table 2.1.

**Table 2.1: Comparison between Arduino UNO and Arduino UNO Wi-Fi
(Arduino.cc.,2018)**

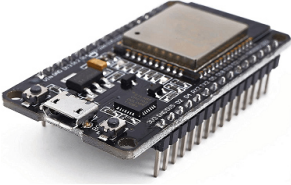
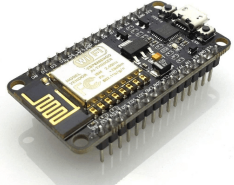
Board	Arduino UNO	Arduino UNO Wi-Fi
Technical Specification		
Microcontroller	ATmega328P	ATmega4809
Flash Memory	32 kB	48 kB
Internet Connectivity	No	Yes
PWM Digital I/O Pins	6	5
Analogue Input Pins	6	
Clock Frequency	16 MHz	
Operating Voltage	5 V	
Analogue Input Pins	6	
Size	68.6 mm × 53.4 mm	
Price	RM 25	RM 259

2.4.2 Introduction to Node MCU

Node Microcontroller Unit (Node MCU) is an open source LUA-based firmware and hardware development environment that is built for the ESP8266 Wi-Fi system on chip (SOC) from Espressif System (Yuan, 2017). Then the Node MCU ESP8266 DEVKIT board is invented to make the Node MCU easier to use. This board incorporates the ESP8266 chip on a standard circuit board. The board is built with a micro universal serial bus (USB) port, a reset button, LED lights, a Wi-Fi antenna, and General Purpose Input Output (GPIO) pins that can be plugged into a bread board. The Node MCU ESP8266 DEVKIT board can be easy to flash like the Arduino by connecting it to a laptop using a micro USB cable. This board can also be programmed by using C++ or C in the Arduino IDE. Later, Espressif System created another advanced version of the Node MCU ESP8266, which is the ESP32. This board is created to replace the lack of security of the ESP8266 board (Ashwak, 2021).

The benefits of using the Node MCU ESP8266 module are that it provides high performance and powerful onboard processing at a low cost, resulting in a functional Wi-Fi. The price of a Node MCU ESP8266 is around RM 15. For less than RM 20, it can monitor and control devices from an isolated place. At the same time, it can also set up a network for itself, allowing other devices to connect to it, which increases the versatility of the ESP8266. Besides that, the Node MCU ESP8266 devkit board has 17 multiplexing GPIO pins. These pins can be assigned to all sorts of peripheral duties, such as analogue to digital converter (ADC) channels, universal asynchronous receiver/transmitter (UART) interface, pulse width modulation (PWM) outputs, serial peripheral interface (SPI), inter-integrated circuit (I2C), and inter-IC sound (I2S) interface. Next, the Node MCU ESP32 is an upgrade of the ESP8266 with an ultra-low power co-processor. The price of a Node MCU ESP32 is around RM 30. It has 34 GPIO pins with an Xtensa dual-core processor running at 160 MHz, working with more complicated projects and providing a faster processing rate. The most important function of ESP32 is that it provides hi-tech security. Additionally, ESP32's built-in temperature sensor allows users to access and detect temperature values remotely without extra external hardware, but the value taken may be affected by the circuit temperature itself. The comparison between Node MCU ESP32 and Node MCU ESP8266 is shown in Table 2.2.

Table 2.2: Comparison between Node MCU ESP32 and Node MCU ESP8266
(Ashwak, 2021)

Board	Node MCU ESP32	Node MCU ESP8266
Technical Specification		
Microcontroller	Single or dual core 32-bit LX6 Xtensa	Single core 32-bit L106 Xtensa
Co-Processor	Ultra Low Power (ULP)	NO
Clock Frequency	160 MHz or 240 MHz	80 MHz
Flash Memory	512 B	4 MB
Internet Connectivity	Yes	Yes
Bluetooth	Bluetooth Low Energy (BLE)	No
Security	Security Boot flash encryption. one-time password (OTP) 1024-bit	No
Power Consumption	10 uA deep sensor	20 uA
Temperature Sensor	Yes	No
Touch Sensor	10	No
Total GPIO	39	17
Total SPI	4	2
Operating Voltage	3 V to 3.6 V	
Size	25.4 mm × 48.3 mm × 3.0 mm	49.0 mm × 24.5 mm × 13.0 mm
Price	RM 30	RM 15

2.4.3 Introduction to Raspberry Pi

As shown in Figure 2.6, the Raspberry Pi is a low-cost single-board computer (SBC) that runs the Linux operating system and can be input with a keyboard or mouse using USB and output to a computer monitor or television (TV) using high-definition multimedia interface (HDMI) (Monk, 2016). The Raspberry Pi can be programmed using any type of language, such as C, C++, Java, or Python. Many tasks can be done by having a Raspberry Pi. For example, document editing, browsing the internet, playing games, and using it as a media centre to play videos (Singh et al., 2021). At a price of around RM 150 for the Raspberry Pi 2 model B and RM 156 to get the Raspberry Pi 3 model B+, Table 2.3 shows two models of the Raspberry Pi.

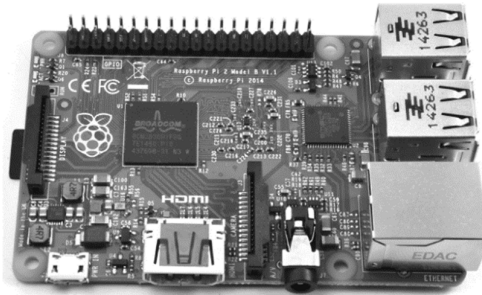


Figure 2.6: Raspberry Pi (Monk, 2016)

The benefits of using the Raspberry Pi for IoT projects are that it supports a wide range of peripherals because it has 26 GPIO pins that can interface with a variety of hardware. The Raspberry Pi also supports almost all the peripherals available on the Arduino. Then, due to the popularity of this board, the Raspberry Pi market resources are vast, providing a large community to support the project. Next, the Raspberry Pi comes with a 1.6 GHz processor, which means this board has a faster processing rate, leading to good performance. The disadvantage of using the Raspberry Pi is that it lacks internal storage, necessitating the use of a micro secure digital (SD) card as internal storage. This will greatly increase the boot time and read or write speed of the Raspberry Pi, as the SD card is slow in speed. Furthermore, because the IoT monitoring system will be running continuously, the Raspberry Pi may overheat. The powerful

processor and multiple features mean high power is needed and it is easy to cause the board to heat up. Heat sinks or cooling fans may need to be added to the Raspberry Pi, but they come at an extra cost.

**Table 2.3: The Raspberry Pi 2 Model B and The Raspberry Pi 3 Model B+
(www.pololu.com, n.d.)**

Board	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B+
Technical Specification		
CPU	BCM2836	BCM2837B0
CPU Speed	900 MHz	1.4 GHz
CPU Cores	4	
HDMI	Yes	
SD Socket	micro SD	
USB Port	4	
Internet Connectivity	No	2.4 GHz 5 GHz 802.11b/g/n/ac
Bluetooth	No	4.2
Size	85.1 mm × 55.88 mm × 20.32 mm	
Price	RM 150	RM 156

2.5 Hardware for Soil Moisture Detection

For plants and agriculture, it is important to have the right amount of water, so as not to over or under water the plants. Since there is a limitation on the human vision system, the moisture level of the soil cannot be detected using naked eyes. Soil moisture sensors play an important role in measuring the volumetric water content of soil. This section includes three types of soil moisture sensors that can be applied to the monitoring system.

2.5.1 Capacitive Soil Moisture Sensor

Figure 2.7 shows the capacitive soil moisture sensor that utilises the dielectric contrast between water and soil to detect the moisture value of the soil (Joshua, 2020). Theoretically, dry soil has a permittivity value of between 2 and 6, while water has a value of roughly 80. So, this sensor analysis determines the permittivity value in order to predict the volumetric water content in the soil. Figure 2.8 shows the hardware schematic of the capacitive soil moisture sensor. There is a built-in 555 timer chip with a frequency oscillator circuit, which supplies a constant square wave to the sensor (Alam, 2019). The higher the soil moisture, the greater the capacitance of the sensor.

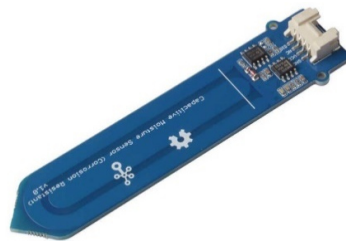


Figure 2.7: Capacitive Soil Moisture Sensor (Shawn, 2010)

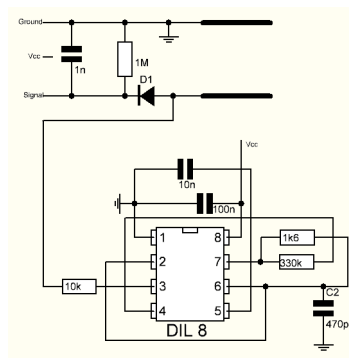


Figure 2.8: Hardware Schematic of the Capacitive Soil Moisture Sensor (Alam, 2019)

One of the advantages of using capacitive soil moisture sensors is that they give a long life service to a project. This is because the capacitive soil moisture sensor is made of corrosion-resistant material and there is no direct exposure of metal

electrodes. Besides that, the accuracy of capacitive soil moisture sensors is also high, as proved in the paper written by Joshua (Joshua, 2020). In Joshua's paper, it is mentioned that there is only a 6 % error in readings from the soil moisture sensor, indicating this sensor gives a good detection of the water content in the soil.

2.5.2 Conductivity Sensor

The conductivity sensor, which is also called a resistive soil moisture sensor, as shown in Figure 2.9, makes use of the relationship between electrical resistance and water content in soil (Shawn, 2010). When the voltage is applied to the two probes, current will be produced and sent from one to another. The change in current will be measured to represent the presence of water. The soil will act as the resistance for the two probes (Antonio, 2021). When the water content in the soil is low, electrical conductivity will be low as in the absence of water, the resistance will be high, leading to low current passing through the probes.

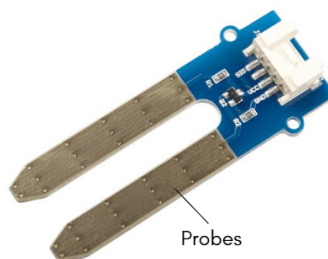


Figure 2.9: Resistive Soil Moisture Sensor (Shawn, 2010)

One reason to get a resistive soil moisture sensor is that it is very cheap in terms of cost, which can be obtained by using RM 5 only. However, the corrosion of this sensor is high as it is frequently used. Then, the sensitivity of this sensor is also low as the conductivity of the two probes may be affected by the concentration of salt in the soil. This is proved by the experiment by Antonio, in which the change in voltage is small upon watering, as shown in Figure 2.10, from 2.11 V to 1.84 V.

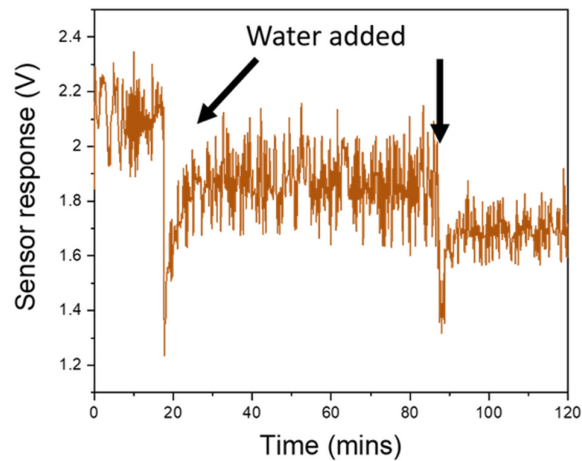


Figure 2.10: Plot of the Electrical Resistance of a Soil Sample (Antonio, 2021)

2.5.3 Soil Moisture & Temperature & EC Sensor

As shown in Figure 2.11, the soil moisture, temperature, and electrical conductivity (EC) sensor is a device that can measure soil moisture and electrical conductivity. This device contains three stainless steel probes that are able to prevent corrosion. They are inserted into the soil surface or profile to measure soil moisture and temperature quickly. The moisture sensor measures the dielectric constant of the soil in order to measure the volume of the soil moisture content. The temperature sensor uses a precision platinum resistance element. The product's built-in drift calibration and temperature compensation circuits can be adapted to most applications. (Antonio, 2021).



Figure 2.11: Soil Moisture, Temperature and Electrical Conductivity (EC) Sensor (Antonio, 2021)

The advantage of using this device is that it is able to detect an extra condition of the soil, which is the temperature of the soil. This allows the monitoring of plants to become more detailed. Secondly, this device is also corrosion free due to the use of stainless steel material as the probes. However, the price of this device is quite high to implement in a project, which costs around RM 400.

2.6 Hardware for Humidity and Temperature Detection

Humidity and temperature are important aspects to take into account in plantations to make sure the plants can grow healthily. So, hardware to detect the humidity and temperature around the plants is needed in order to take quick action in bad conditions. This section will include three types of humidity and temperature sensors that can be applied to the monitoring system. The pros and cons of these sensors will be discussed.

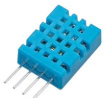

2.6.1 DHT Sensors

DHT sensors are the digital sensors that perform measurements of the temperature and relative humidity. These sensors are made with a capacitive humidity sensor and a thermistor. These sensors are digital because they give a digital signal output for temperature and humidity by using a chip. This gives an analogue to digital conversion. There are two versions of DHT sensors: DHT11 and DHT22. Both have the same pinout but are different in characteristics (Random Nerd Tutorials, 2019).

The advantages of using DHT sensors are that they are low cost, easy to interface with a microcontroller via a digital signal, and they are able to detect both the temperature and humidity of the air. However, these sensors are very basic and slow. Version DHT11 has the advantages of ultra-low cost (RM 4.50 in the market) and faster sampling rate, yet it has low accuracy and range of measurement. In contrast, DHT22 sensors have better accuracy and a longer range of measurement. In other

words, the price of DHT22 will be higher than DHT11, which is RM 15.50. The specifications of both DHT11 and DHT22 are shown in Table 2.4.

Table 2.4: DHT11 and DHT22 Specifications (Random Nerd Tutorials, 2019).


DHT Version	DHT11	DHT22
Specification		
Power Supply Range	3.0 V to 5.5 V	3.0 V to 6 V
Temperature Detection Range	0 °C to 50 °C	-40 °C to 80 °C
Humidity Detection Range	20 % to 90 %	0 % to 100 %
Accuracy	± 2.0 °C	± 0.5 °C
Sampling Rate	1 second	2 seconds
Size	15.5 mm × 12mm × 5.5 mm	15.1 mm × 25 mm × 7.7 mm
Price	RM 4.50	RM 15.50

2.6.2 DS18B20 Temperature sensor

The DS18B20 is a programmable-resolution digital sensor that requires only one data line to interface with a central microcontroller. This sensor provides a resolution of 9-bit to 12-bit temperature measurement and has an alarm function (Maxim Integrated Products, 2019). It also has a unique 64-bit serial code, allowing multiple DS18B20 sensors to connect to one microcontroller with only one data wire. In addition, it can derive power directly from the data line without an external power supply.

The advantages of using DS18B20 temperature sensors are low cost, saving on the usage of microcontroller pins, and adjustable accuracy. However, this sensor only detects or measures the temperature value. The specifications of the DS18B20 temperature sensor are included in Table 2.5.

Table 2.5: DS18B20 Temperature Sensor Specifications (Maxim Integrated Products, 2019)


Sensor	DS18B20 
Specification	
Power Supply Range	3.0 V to 5.5 V
Temperature Detection Range	-55 °C to 125 °C
Accuracy	± 0.5 °C
Price	RM 4.30

2.6.3 Barometric Sensors

A barometric sensor is a sensor that is able to measure the atmospheric pressure. There are two versions of the barometer sensor: BME280 and BMP180. BME280 is a new version of BMP180 that is built with both a temperature and humidity sensor, while BMP180 is only equipped with a temperature sensor (BME280-Data sheet, 2018). Thus, the BME280 is a barometric sensor that is able to measure air temperature, humidity, and pressure. BME280 is able to detect the temperature range from -40 °C to 85 °C, while BMP180 just measures from 0 °C to 65 °C. In addition, the BME280 also provides SPI and I2C interfaces.

The advantages of using BME280 are that this sensor is able to achieve high performance in obtaining the air humidity and pressure values, has a wider temperature measurement range, and has more functionalities. However, the price of a BME280 sensor module is higher, at around RM 40. In addition, the BME280 sensor module self-heats a little bit, which may cause the temperature reading to vary by 1 °C to 2 °C. The specifications of the BME280 sensor are included in Table 2.6.

Table 2.6: BME280 Sensor Specifications (BME280 -Data sheet, 2018)

Sensor	BME280
Specification	
Power Supply Range	3.3 V to 5.0 V
Temperature Detection Range	-40 °C to 85 °C
Accuracy	± 0.5 °C
Price	RM 38.50

2.7 IoT Platform

The IoT platform is served as a middleman to connect hardware and software by using the internet. The IoT cloud platform can simultaneously handle massive data volumes from devices, applications, and sensors and take action to provide a real-time reaction. Three different IoT platform options that can be used with the monitoring system are covered in this section. The advantages and disadvantages of various platforms are examined.

2.7.1 Google Cloud Platform

In 2011, Google has created the Google Cloud Platform, a middleware. This platform makes it simple for consumers to access the cloud and other computing services (Saran, 2018). This platform is also included a tonne of cloud functionalities that helps the customers perform their tasks, like cloud storage, data analytics, and machine learning. The Google Cloud platform also consists of a variety of components, including Google Cloud Dataflow, Google Cloud Endpoints, Google Cloud Storage, and others. These components are beneficial to users in a variety of ways (Google, 2019).

Utilizing the Google Cloud Platform provides its benefits due to its high performance and extensive functionality. The web responds more quickly on this platform and loads pages more quickly. Additionally, it functions well with the hardware setups, which results in a better cloud hosting experience. Additionally, with the aid of the Google cloud servers, consumers have access to their data from anywhere in the world. Users must, however, pay a high subscription fee to use the platform. Figure 2.12 shows the logo of Google Cloud Platform.



Figure 2.12: Logo of Google Cloud Platform (Google, 2019)

2.7.2 AWS IoT Core

AWS IoT Core is a managed cloud service that enables customers to connect their devices to the cloud and communicate with other devices and cloud apps even when they are not online. The AWS IoT Device software development kit (SDK), which facilitates the connection between a user's devices and allows for message exchange with the AWS IoT Core, is one of the main components of the AWS IoT Core (Amazon Web Services, Inc., n.d.). message queuing telemetry transport (MQTT), lightweight communication protocols, and hypertext transfer protocol (HTTP) are all supported. Additionally, with mutual authentication and end-to-end encryption, AWS IoT Core offers a safe platform for transferring data to AWS endpoints and other devices.

The benefits of adopting this platform include its ability to process massive volumes of messages or data, the ability to follow and communicate with apps even when they are not connected, and the ability to securely access devices. However, access to this site requires payment. Pricing is determined based on connectivity,

messaging, and shadow device usage. Figure 2.13 shows the logo of AWS IoT Core Platform.



Figure 2.13: Logo of AWS IoT Core Platform (Amazon Web Services, Inc., n.d.)

2.7.3 Blynk

A group of software tools called Blynk requires users to deploy linked electrical devices, prototype them, and manage them remotely. Blynk's major goal is to make the process of developing mobile applications as simple as possible (Blynk.io, 2015). By creating a no-code Android app and a web application to analyse real-time and historical data flowing from devices, users may connect their gear with the cloud. At the same time, customers have the option of controlling them remotely from a remote location and receiving alerts or notifications when certain criteria are reached. The application is also prepared for end customers, allowing anyone who has bought the product to download it, connect it to their device, and use it right away.

Blynk has the benefits of being free for personal use and prototyping, and the Blynk app is an app editor that can customise an app to share the project with friends and family, who can also use the functionality. In addition, Blynk is a user-friendly platform with simpler visualisation and connectivity. Blynk, however, cannot communicate with devices while they are offline, and it also does not allow the export of significant amounts of data. Figure 2.14 shows the logo of Blynk Platform.



Figure 2.14: Logo of Blynk Platform (Blynk.io, 2015)

2.8 Summary

In this chapter, three commercial products have been reviewed from three different companies: Libelium, CropX, and PrecisionHawk. Firstly, Libelium provides complete IoT technology, including its own Libelium Cloud. Secondly, CropX provides a cloud-based decision support tool to help customers schedule farm activities and monitor their crops' health and growth. Thirdly, PrecisionHawk provides drone-based agriculture solutions with its own-design drones, visual sensors, and web-based portal.

Next, the non-commercial projects that has been reviewed in this chapter are the IoT automatic plant watering system, the IoT smart agriculture & automatic irrigation system with ESP8266, and the Avengers Plant Monitoring Device. Firstly, the IoT automatic plant watering system project has built a watering system that is able to control the watering action using a smartphone anytime, anywhere. Secondly, the IoT smart agriculture & automatic irrigation system with ESP8266 project has built a system that monitors the condition of the plants and performs irrigation automatically. Thirdly, the Avengers Plant Monitoring Device project developed a system that can detect the soil moisture of the plants and display it to the user via smartphone.

Besides that, the development boards, soil moisture detection hardware, humidity and temperature detection hardware, and IoT platforms have been investigated in this chapter. Firstly, the development boards are the Arduino, Node MCU, and Raspberry Pi. Secondly, the soil moisture detection hardware are the capacitive soil moisture sensor, conductivity sensor, and soil moisture, temperature and electrical conductivity (EC) sensor. Thirdly, the reviewed humidity and temperature detection hardware are the DHT sensor, the DS18B20 temperature sensor, and the barometric sensor. Fourthly, the IoT platforms reviewed in this chapter are Google Cloud Platform, AWS IoT Core, and Blynk.

In the next chapter, Chapter 3, the methodology of this project will be discussed. The most suitable hardware and software will be decided. The system working principal also will also be discussed to connect the hardware and software. Lastly, the management of this project will also be included in the next chapter.

CHAPTER 3

METHODOLOGY

3.1 Introduction

There are six sections in this chapter to discuss the principles of this project. The first section is the selection of the development board. This section discusses the most suitable development board reviewed in Chapter 2 to be applied to this project and how it operates in this project. The next section is on the IoT monitoring system, which includes IoT platform selection, soil moisture sensor selection, humidity and temperature sensor selection, and other hardware related to the IoT monitoring system. Their connection and the operation of the hardware are included in this section. The third section is the irrigation system. This section includes the hardware applied to the irrigation system to perform watering actions. In addition, discussion on the system working principle is included in the fourth section to provide the system architecture design and the connection between hardware, so that a clear illustration of system design is given. The operating flow of every part of the entire system is discussed. Furthermore, the cost estimation is also done in this chapter to set a budget for this project. The last section is the management of the project. The Gantt charts are included in this section to demonstrate the project milestones.

3.2 Development Board Selection

The Node MCU ESP32 DEVKIT V1 DOIT development board is used for this project among many other boards analysed because it has the best pricing among the available alternatives for the ESP32. Although ESP8266 costs less than ESP32 and was used in the majority of the projects examined, its 17 GPIO pins are seen as insufficient to construct a system with IoT monitoring and smart irrigation. Therefore, by investing a few extra ringgits to purchase an ESP32 with 30 GPIO pins, the problem of a lack of GPIO pins may be resolved, increasing the capability of the system.

A dual-core 32-bit processor is also included with the ESP32, as can be seen in the function block diagram in Figure 3.1. The dual-core processor in this project allows it to synchronise two tasks: monitoring and watering, to perform multitasking more effectively, despite the fact that FreeRTOS is capable of doing so. The ESP32 comes equipped with extra random access memory (RAM) and Flash memory, an ADC, and a variety of other peripherals, as was mentioned in Chapter 2.4.2. To complete this project, these capabilities are more than adequate. Furthermore, this board's biggest feature is its inclusion of Wi-Fi and Bluetooth, which makes it stand out from similarly priced boards like Arduino. Consequently, this project makes it simple to manage the device from an isolated place at a low price in order to achieve the low-cost IoT monitoring.

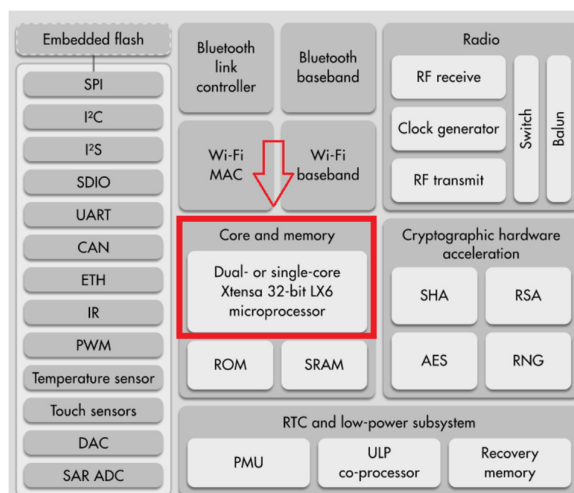


Figure 3.1: ESP32 Function Block Diagram (Esp32.net, 2016)

3.3 IoT Monitoring System

The IoT monitoring system is added to overcome one of the issues faced in the non-commercial project, which is that the condition of the plants is unknown. The low cost and small scale IoT monitoring system is applied to this project, which is suitable for indoor plantation as compared to the commercial projects reviewed in Chapter 2.2. This IoT monitoring system contains parts such as an IoT platform, soil moisture sensor, air temperature and humidity sensor.

3.3.1 IoT Platform Selection

Blynk is the IoT platform that is being used in this project, enabling prototype and remote control of the linked devices. Blynk supports a wide range of devices, including the Arduino, ESP32, ESP8266, Raspberry Pi, and others. The benefit of adopting Blynk is that it is a free IoT platform for personal use, and the system may be shared with up to five other individuals. In addition, Blynk offers a no-code required app, which allows this project to customise an app to accomplish the goal, which is also the motivation behind utilising this platform.

The Blynk IoT app, which can be downloaded using a smartphone, is offered by Blynk and is shown in Figure 3.2. With the help of this app, the project can modify a no-code Android app that analyses both recent and older data from connected devices. In addition to creating mobile dashboards, Blynk also offers web dashboards for customising and interacting with connected devices. This project is defined in Blynk by the input and output using the virtual pin as shown in Figure 3.3 via a computer or mobile device. Blynk offers 256 virtual pins and supports double, integer, and string as its three data types. This project can include more additional functionalities in the system thanks to the large number of virtual pins.

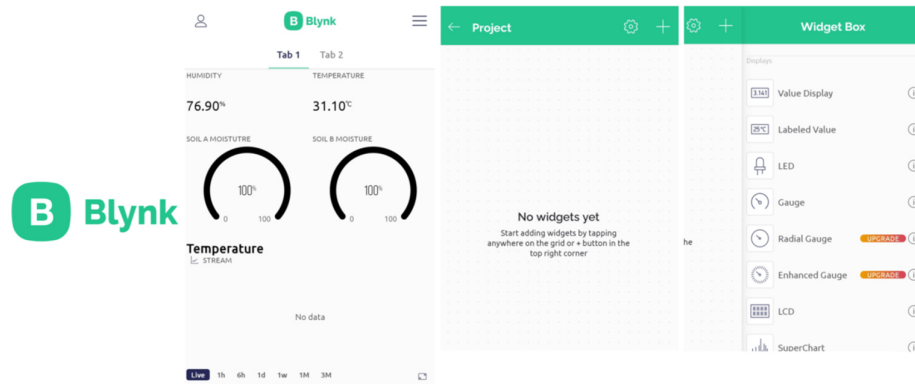


Figure 3.2: Blynk IoT App

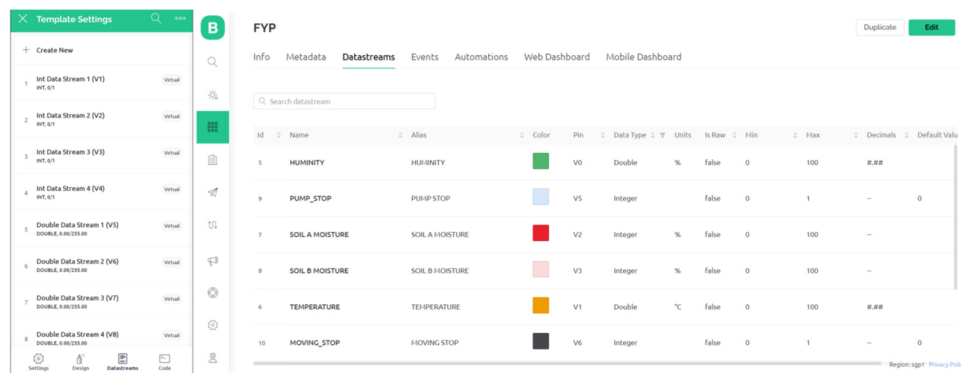


Figure 3.3: Defining the Input and Output

In addition, Blynk also includes the automation function, with which this project can set some events or actions to take when any threshold values are reached. Notification is also sent to report the condition of the plants, and automatic irrigation can be done without engagement. Another interesting function of Blynk is Blynk.Air. The firmware can be sent to the connected device over the air from Blynk to update the device without programming the device directly using a cable. The allocation of virtual pins is shown in Table 3.1.

Table 3.1: Virtual Pins Allocation in Blynk

Virtual Pin	Name	Description
V0	HUMIDITY	Humidity data input from the DHT sensor
V1	TEMPERATURE	Temperature data input from the DHT sensor

V2	SOIL A MOISTURE	Soil moisture in area A data input from the capacitive soil moisture sensor
V3	SOIL B MOISTURE	Soil moisture in area B data input from the capacitive soil moisture sensor
V4	INSTANT WATERING	Instant watering signal output to perform watering action manually
V7	PUMP_START	Pump status (start) input from the connected device
V8	MOVING_START	Stepper motor moving status (start) input from the connected device
V9	DISTANCE	Output signal to the stepper motor to control the total moving distance (0 to 25 cm)
V10	PUMP POWER	Output PWM signal to the water pump to control the strength of water pump
V12	WATER LEVEL PERCENTAGE	Water level data input from the ultrasonic sensor.

3.3.2 Soil Moisture Sensor Selection

This project uses capacitive soil moisture sensors to detect the condition of the soil. As opposed to resistive sensors, capacitive soil moisture sensors do not directly expose the electrode metal to the air, which considerably reduces electrode degradation and gives a long life to this project. This is significant because the system will be in operation for a long time. The soil moisture value needs to be able to be measured often as part of this project. In addition, the capacitive soil moisture sensors outperformed the resistive sensors in terms of sensitivity and accuracy. The experiment done by Adla et al. showed that the capacitive soil moisture sensors are more accurate than the resistive soil moisture sensors (Adla et al., 2020). Figure 3.4 shows the accuracy of capacitive soil moisture sensors (SMEC300 and SM100) and resistive soil moisture sensors (YL69 and YL100), in 4 different soils. The closer the bubble is to

its origin, the more accurate the sensor is. As a result, capacitive sensors can provide a more precise assessment of the soil's state for effective plant monitoring.

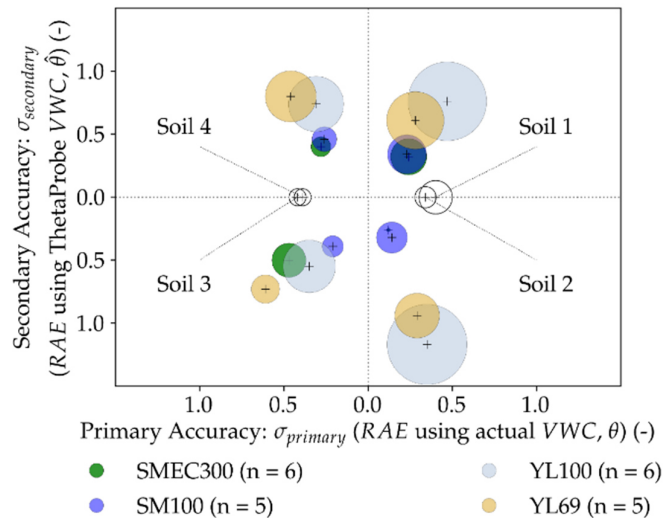


Figure 3.4: Accuracy of Capacitive Soil Moisture Sensors (SMEC300 and SM100) and Resistive Soil Moisture Sensor (YL69 and YL100) (Adla et al., 2020)

In this project, a total of two capacitive soil moisture sensors are used to detect two different areas of soil moisture value. In Figure 3.5, the structural and pin-out of the capacitive soil sensor is shown. It is also included with a rail-to-rail output operation amplifier, LMV358ID, to drive high capacitive-load with low voltage operation needed. A grove cable is connected to the sensor and then connected to the NodeMCU ESP32 board.

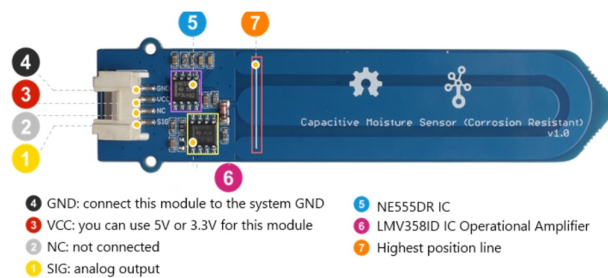


Figure 3.5: Overview of Capacitive Soil Sensor (wiki.seeedstudio.com, n.d.)

3.3.3 Humidity and Temperature Sensor Selection

The DHT22 sensor is used to measure the temperature and humidity of the air. The DHT sensor, which is the most appropriate sensor to utilise in this project, is a basic and inexpensive digital temperature and humidity sensor, as discussed in Chapter 2. The cost of a single DHT22 sensor module is RM 15.50. Then, as illustrated in Figure 3.6, this project can simply connect the leftmost pin to ground, the centre pin to VCC (which ranges from 3 V to 5 V), and the rightmost pin to the ESP32's data input pin. The Arduino IDE has the Adafruit DHT sensor library installed to programme the ESP32 board and manage the DHT22 sensor.

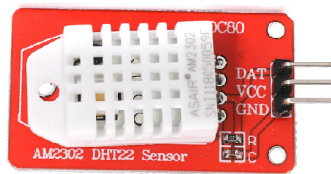


Figure 3.6: DHT 22 Sensor Module

3.3.4 Ultrasonic Sensor

The ultrasonic sensor is added to this project as an extra feature to detect the water level of the tank. This project uses the model HC-SR04P as shown in Figure 3.7, which is compatible with the model HC-SR04 ultrasonic sonar distance sensor as shown in Figure 3.8. The ultrasonic sensor uses an ultrasonic transmitter and ultrasonic receiver to measure the distance between two points. The basic working principle of this sensor is to use the trigger pin to create a high-level signal for at least 10 μ s and then the module sends eight 40 kHz signals and detects the available pulse signal back. The amount of time taken to travel from and back to the sensor is used to measure the distance. The range of detection is about 2 cm to 450 cm with a 5 V power supply and 2 cm to 400 cm with a 3 V power supply (ElecFreaks, 2011). The only difference between model HC-SR04P and model HC-SR04 is that model HC-SR04P supports a 3 V to 5 V power supply, while model HC-SR04 only supports 5 V. Thus, the reason for choosing model HC-SR04P in this project is that node MCU ESP32 is a 3.3 V

system, which means an extra level shifter needs to be added when using model HC-SR04.



Figure 3.7: HC-SR04P Ultrasonic Sensor (Cytron Technologies Malaysia, n.d.)

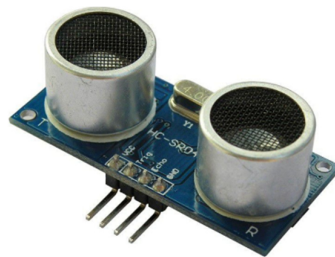


Figure 3.8: HC-SR04 Ultrasonic Sensor (Cytron Technologies Malaysia, n.d.)

3.4 Irrigation System

The irrigation system is added to overcome one of the issues encountered in the non-commercial project, which is that the watering action is insufficient. A low-cost, small-scale irrigation system is applied to this project, which is suitable for indoor plantations. To address the issue of water spreading around the house, this project intends to create a model that can be watered from the top of the plant and has a fixed watering range. This idea comes from one of the non-commercial projects reviewed in Chapter 2.3.1. A model of this irrigation system is developed. This watering system contains parts such as the water pump and stepper motor.

3.4.1 Irrigation System Model

The irrigation system model is constructed using a total of 13 pieces of PVC pipe: 12 pieces are built into the rectangular shape of the system's body, and one piece is used at the top that can move along and provide watering action, as shown in Figure 3.9. There are 8 pieces of 3-way corner elbow PVC fitting connectors that are used to combine the 12 pieces of PVC pipe. The irrigation system is able to water the plants from the top equally along the fixed area with a bar that is able to move in the X direction, where the water tube is attached to the bar. In order to move the bar, there are five pieces of 3D-printed holders applied in the system. Figures 3.10 to 3.13 show the 3D-printed holder: a belt holder, a motor holder, a pulley holder, and two linear sliders that hold the watering bar.

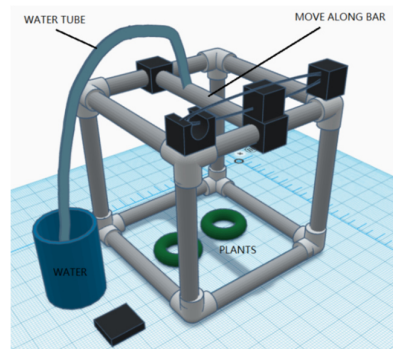


Figure 3.9: Sketch Model of Irrigation System

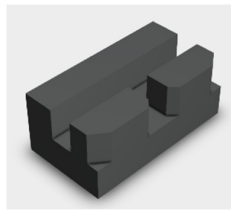


Figure 3.10: 3D Sketch of Belt Holder

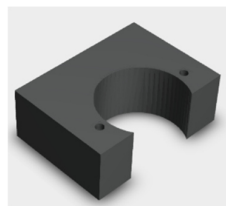


Figure 3.11: 3D Sketch of Motor Holder

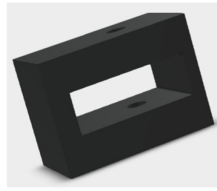


Figure 3.12: 3D Sketch of Pulley Holder

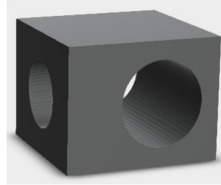


Figure 3.13: 3D Sketch of Linear Slider

3.4.2 Water Pump

The R385 DC12 V diaphragm water pump, as shown in Figure 3.14, is applied in this project to pump the water from the water tank to the moving bar. It operates at a direct current (DC) voltage of 12 V, providing water pressure of 0.3 MPa at the inlet. This pump can pump water at a rate of 1.6 L/min, which is appropriate for this project because it must pump water from a lower to a higher position (Cytron Technologies Malaysia, n.d.). Since there are only power pins allocated on the water pump, a driver is added to provide the signal sent from the ESP32 board and control the input power to turn on or off the water pump. The driver is an L298N motor driver that supports 7 V to 30 V. L298N is a high-voltage and high-current dual full-bridge driver that is able to receive a digital signal and drive the motor (STMicroelectronics, n.d.). This project uses the module of the L298N motor driver as shown in Figure 3.15, which consists of a 5 V or 12 V input voltage terminal, 2 output terminals, a heatsink mounted with the L298N for better heat dissipation, and header pins for a digital input signal.

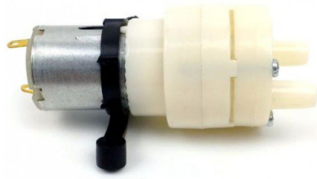


Figure 3.14: R385 DC12 V Diaphragm Water Pump (Cytron Technologies Malaysia, n.d.)

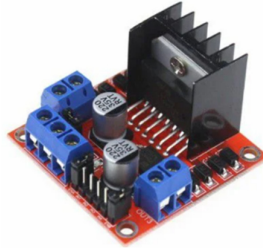


Figure 3.15: L298N Motor Driver (Cytron Technologies Malaysia, n.d.)

3.4.3 Stepper Motor

The 12 V 28BJ-48 stepper motor, as shown in Figure 3.16, is used with a bearing and belt to move the PVC pipe and perform the watering action in this project. This stepper motor is operated at 12 V and is able to provide a pull-in torque of 300 gf/cm. In addition, this is a 4-phase stepper motor that needs 4 digital inputs to be controlled. Thus, the ULN2003 driver board, as shown in Figure 3.17, is used to control the stepper motor. The ULN2003 is a high-voltage and high-current Darlington array, containing seven open collector Darlington pairs with common emitters (STMicroelectronics, n.d.).

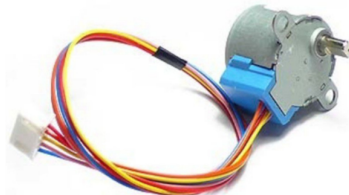


Figure 3.16: 12 V 28BJ-48 Stepper Motor (Cytron Technologies Malaysia, n.d.)

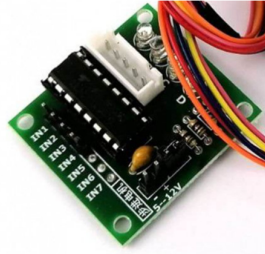


Figure 3.17: ULN2003 Driver Board (Cytron Technologies Malaysia, n.d.)

After that, the steps of the motor required to pull the moving bar by one centimetre is calculated using Equation 3.1 below.

$$\text{Total steps} = \text{Target distance} \times \left[\frac{\text{Steps of one full revolution}}{(\text{Belt pitch} \times \text{Pulley teeth numbers})} \right] \quad (3.1)$$

The stepper motor is operated in half step in this project. Therefore, 4096 steps are needed to complete one full revolution. Then, the pitch of the belt is 2 as used for the GT2 pulley. In addition, the teeth numbers of the pulley that is used in this project are 20. By inserting the value into Equation 3.1, this project able is to get the total steps that the stepper motor is required to pull the moving bar to the target distance. The outcome of Equation 3.1 is in millimetres. Thus, it needs to multiply by ten to get the distance in centimetre. Equation 3.2 shows the final formula to get the total steps value that is used to control the stepper motor.

$$\text{Total steps} = \text{Target distance} \times \left[\frac{4096}{(2 \times 20)} \right] \times 10 \quad (3.2)$$

3.5 System Working Principle

The system architecture design and the flow of the system are discussed in this section to give a clear illustration of the system design. The block diagram is drawn to show the system architecture design. In addition, the pin allocation on the Node MCU ESP32 board is included in this section to make sure the order is sent correctly to the interfaced hardware.

3.5.1 Block Diagram of the Complete System

A block diagram is included in this subsection, as shown in Figure 3.18, giving a visual representation of the system design so that future technical issues can be managed effectively. A 12 V power supply is provided to the system, powering the water pump motor driver, stepper motor driver, and ESP32 board. A step-down power module is added to regulate the 12 V supply voltage to 5 V to make sure the voltage supply to the ESP32 board is maintained at 5 V. The idea of this complete system is that the NodeMCU ESP32 acts as the middleware to interface with Blynk Cloud and the hardware to collect data and give instructions.

The NodeMCU ESP32 is connected to the Blynk server via the internet, storing data in the Blynk Cloud, and displaying the condition of the plants to the application on the computer and smartphone. The conditions of the plants, such as soil moisture, air temperature, and air humidity, are collected using a humidity and temperature sensor and two capacitive soil sensors, connected to the ESP32 board. An ultrasonic sensor is also added to detect the water level in the tank. After that, a push button is added to perform instant watering physically, and a limiter switch is added to control the movement of the watering bar. Moreover, the condition is also displayed on the OLED display by the ESP32 board, and four LEDs are added to indicate the status of the water pump and stepper motor. In addition, a digital signal is sent from ESP32 to the water pump motor driver and stepper motor driver to carry out the watering.

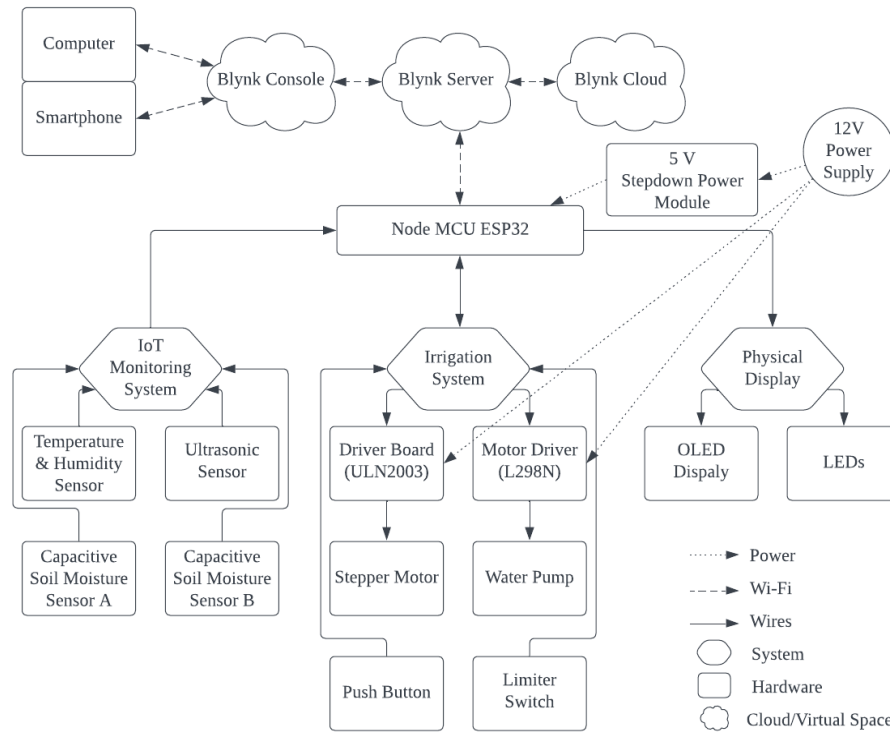


Figure 3.18: Block Diagram of the Complete System

3.5.2 Operating Flow of Physical Display System

The flowchart of the physical display system is shown in Figure 3.19. This is a predefined process to display the condition of the plants physically by using OLED display; and the status of the stepper motor and the water pump by using LEDs. Firstly, air temperature, air humidity, soil moisture area A, and soil moisture area B are detected by the sensors and sent to the Node MCU ESP32 board. At the same time, the ESP32 board stores the data and converts the soil moisture value in areas A and B into a percentage. Then, the air temperature and air humidity data is displayed on the OLED first. The air temperature is displayed in degrees Celsius ($^{\circ}\text{C}$), and the air humidity is displayed in percentage (%). A four-second delay is added before displaying the soil moisture.

After that, the soil moisture in area A is displayed next. The display style is classified into three stages based on the soil moisture. The first stage is when the soil

moisture percentage value is between 0 % and 30 %. At the top of the OLED display, the message "Plant A needs water" is displayed, followed by a crying animated emoji in the middle of the display and the soil moisture percentage value (%) next to the emoji. The second stage is when the soil moisture percentage value is between 31 % and 70 %. At the top of the OLED display, the message "Plant A looks good" is displayed, followed by a neutral animated emoji in the middle of the display and the soil moisture percentage value (%) next to the emoji. The third stage is when the soil moisture percentage value is between 71 % and 100 %. At the top of the OLED display, the message "Plant A very fresh" is displayed, followed by a happy animated emoji in the middle of the display and the soil moisture percentage (%) next to the emoji. A five-second delay is added before displaying the soil moisture value of area B. The soil moisture in area B is displayed next and has the same display style as the soil moisture in area A. The only difference is that the message displayed changes from A to B. For example, "Plant A looks good" changes to "Plant B looks good". A five-second delay is added again.

Meanwhile, there are four LEDs (two red and two green) that are used in this system to indicate the status of the water pump and the stepper motor. When the water pump is activated to pump the water from the water tank to the moving bar, the green LED that indicates the status of the water pump is turned on, and the red led that indicates the status of the water pump is turned off. When the water pump is deactivated, the green LED that indicates the status of the water pump is turned off, and the red LED that indicates the status of the water pump is turned on. This is same for stepper motor status: when the stepper motor is activated to move the moving bar, a green LED that indicates stepper motor status is turned on, and a red LED that indicates stepper motor status is turned off. When the stepper motor is deactivated, the green LED that indicates stepper motor status is turned off, and the red LED that indicates stepper motor status is turned on. The loop system is repeated to display the condition of the plants physically.

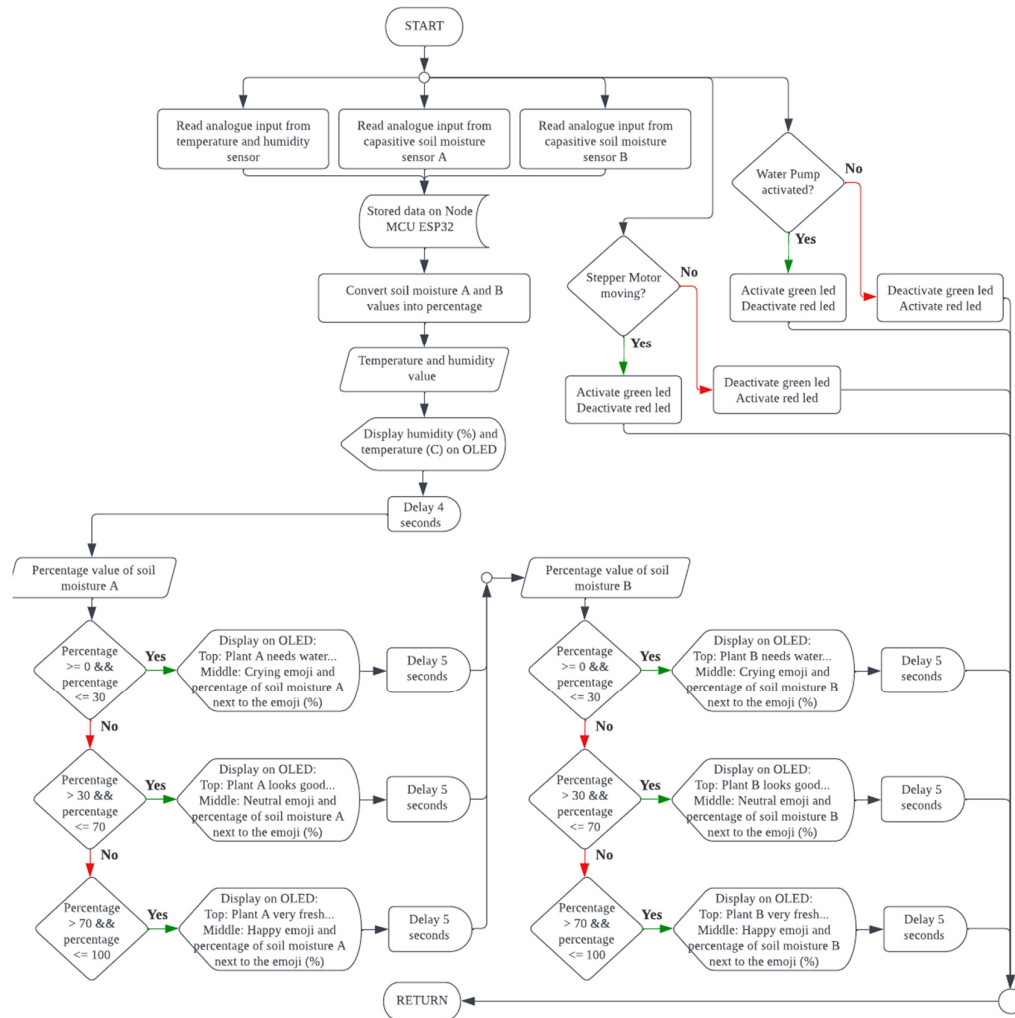


Figure 3.19: Flowchart of Physical Display System

3.5.3 Operating Flow of IoT Monitoring System

The flowchart of the IoT monitoring system is shown in Figure 3.20. This is a predefined process to display the condition of the plants, the status of the water pump, the status of the stepper motor, and the water level of the water tank. Firstly, the connection to the Blynk server is checked. When the connection is made, variables like air temperature, air humidity, soil moisture in area A and area B, and water level are read by the sensors. The Node MCU ESP32 board receives the data by using another core different from the physical display system. The data is sent to the Blynk

server and stored in the Blynk Cloud. Next, the data is sent to the Blynk console on the computer and the Blynk IoT app on the smartphone.

After that, the data are displayed on the Blynk IoT app to let this project monitor the condition of the plants. The data for air temperature, air humidity, and soil moisture in areas A and B are also tabulated into a graph and recorded in the second tab of the Blynk app. The data of air temperature and air humidity are displayed every 30 minutes, while the data of soil moisture in area A and soil moisture in area B are displayed every 50 minutes. For the data of the water level, there is an automation that can be activated. When the automation is activated, the percentage of water level is checked by the system. If the percentage of the water level is less than 15 %, a notification is sent to the smartphone.

Meanwhile, there are two virtual LEDs used on the Blynk IoT app to indicate the status of the water pump and the stepper motor. When the water pump is activated to pump the water from the water tank to the moving bar, the virtual green LED that indicates the status of the water pump is turned on, and it is turned off when the water pump is deactivated. This is the same for stepper motor status: when the stepper motor is activated to move the moving bar, the virtual green LED that indicates stepper motor status is turned on, and it is turned off when the stepper motor is deactivated.

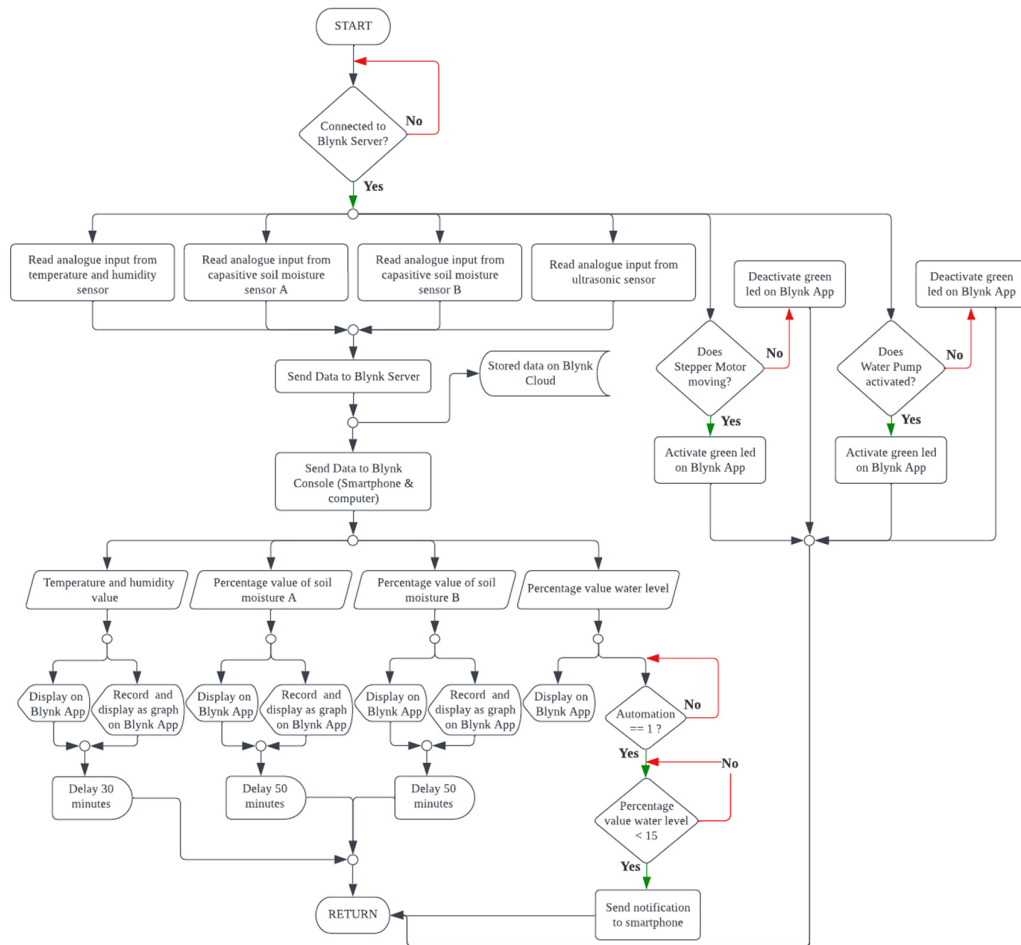


Figure 3.20: Flowchart of IoT Monitoring System

3.5.4 Operating Flow of Irrigation System

The flowchart of the irrigation system is shown in Figure 3.21. This is a predefined process to perform irrigation by controlling the stepper motor and the water pump. Firstly, the stepper motor and the water pump are activated when this predefined process is called. The stepper motor is accelerated to 200 steps per second and then moved at a constant speed of 450 steps per second. The positive sign is used to control the rotation direction of the stepper motor. So, the stepper motor is rotated clockwise to pull the moving bar from left to right. With the distance value and water pump power value set on the Blynk IoT app, the moving bar is moved to the target distance, and water is pumped from the water tank to the moving bar.

After reaching the target distance, the water pump is deactivated. Then, the stepper motor is accelerated to 200 steps per second and then moved at a constant speed of -450 steps per second. The negative sign is used to control the rotation direction of the stepper motor to rotate anticlockwise and pull the moving bar from right to left. After that, this system checks for the limiter switch signal continuously. When the limiter switch signal is equal to 1, which means the moving bar has returned to the starting point, the stepper motor is deactivated, and the moving bar is stopped.

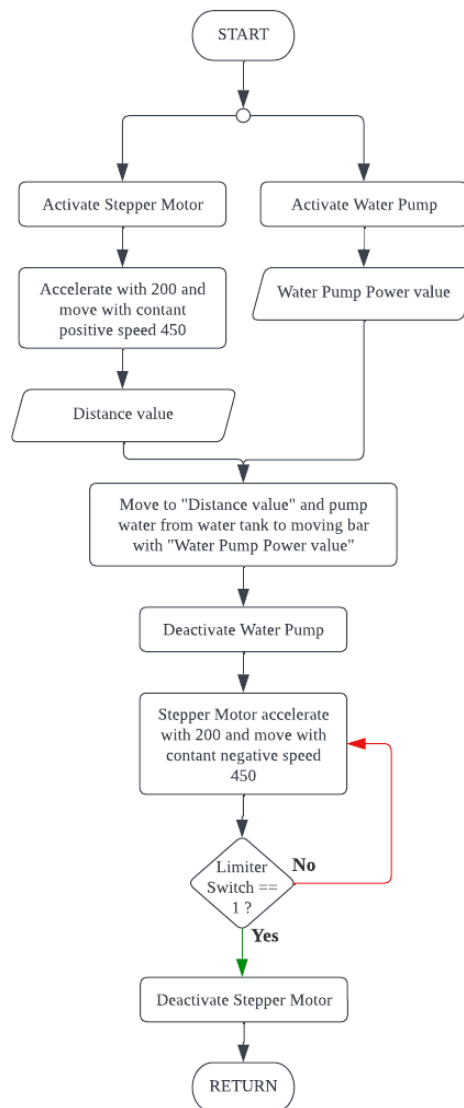


Figure 3.21: Flowchart of Irrigation System

3.5.5 Operating Flow of Smart Irrigation System

The flowchart of the smart irrigation system is shown in Figure 3.22. This is a predefined process to perform watering actions automatically when the automation is activated on the Blynk IoT app. In the Blynk IoT app, there is an automation page where the automation can be activated. This project can choose to activate automation for soil moisture in areas A, B, or both. Thus, this system determines which automation is activated at the beginning.

After that, the soil moisture percentage value in the specific area is checked. When the soil moisture is lower than 30 %, the stepper motor and the water pump are activated. The stepper motor is accelerated to 200 steps per second and then moves at a constant speed of 450 steps per second. The positive sign is used to control the rotation direction of the stepper motor. So, the stepper motor is rotated clockwise to pull the moving bar from left to right. With the distance value and water pump power value set on the Blynk IoT app, the moving bar is moved to the target distance, and water is pumped from the water tank to the moving bar.

After reaching the target distance, the water pump is deactivated. Then, the stepper motor is accelerated to 200 steps per second and then moves at a negative constant speed of 450 steps per second. The negative sign is used to control the rotation direction of the stepper motor to rotate anticlockwise and pull the moving bar from right to left. After that, this system checks for the limiter switch signal continuously. When the limiter switch signal is equal to 1, which means the moving bar has returned to the starting point, the stepper motor is deactivated, and the moving bar is stopped.

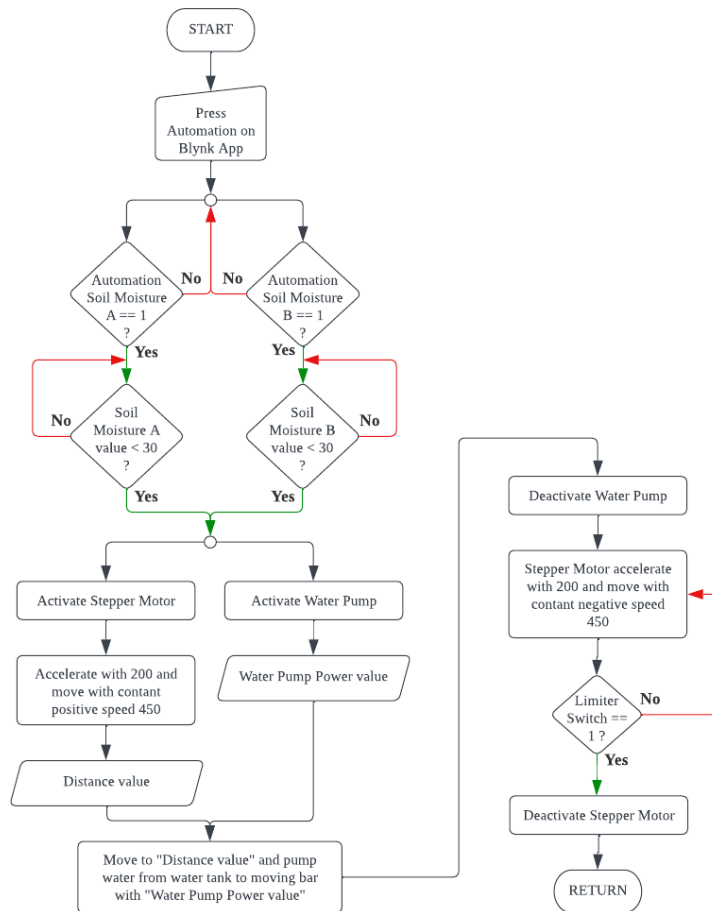


Figure 3.22: Flowchart of Smart Irrigation System

3.5.6 Operating Flow of the Complete System

The flowchart of the whole system in this project is shown in Figure 3.23. At the beginning, the connection of the Node MCU ESP32 development board to the Wi-Fi is detected. When the connection is successful, the system proceeds to check the status of the limiter switch. When the limiter switch gives a signal of 0, the moving bar is not at the starting position. So, the stepper motor is activated to move the moving bar from right to left to the starting position at a negative constant speed of 450 steps per second. When the limiter switch gives a signal of 1, the moving bar has reached its starting position, and the stepper motor is deactivated. After that, the signal from the push button is checked continuously by this system. When the signal from the push button

equals 1, this system enters the predefined process of the irrigation system to perform irrigation on the plants.

At the same time, this system also enters into the predefined processes of the IoT monitoring system and physical display system to perform specific tasks. On the Blynk app, this project is able to set the moving distance value of the moving bar, the water pump power, activate the predefined process of the smart irrigation system, and perform an instant watering action. The values of moving distance and water pump power are used in every watering action. After that, this system checks the signal for instant watering continuously. When the signal is equal to 1, this system enters the predefined process of irrigation system.

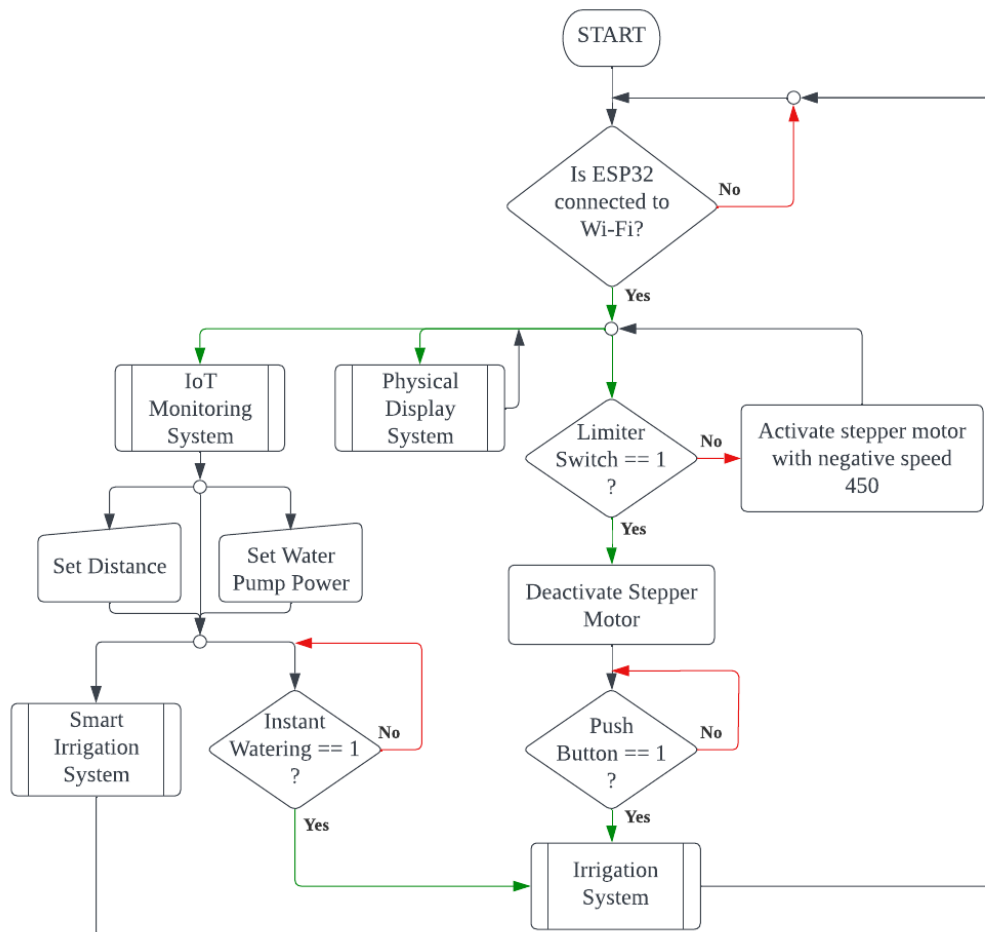


Figure 3.23: Flowchart of the Complete System

3.5.7 Pin Allocation

Before starting to programme the NodeMCU ESP32 board, the allocation of the pins to interface with external hardware such as sensors, LEDs, buttons, and other components is important, so that the order is sent to the correct pin of the hardware. The pin configuration of the NodeMCU ESP32 is shown in Figure 3.24.

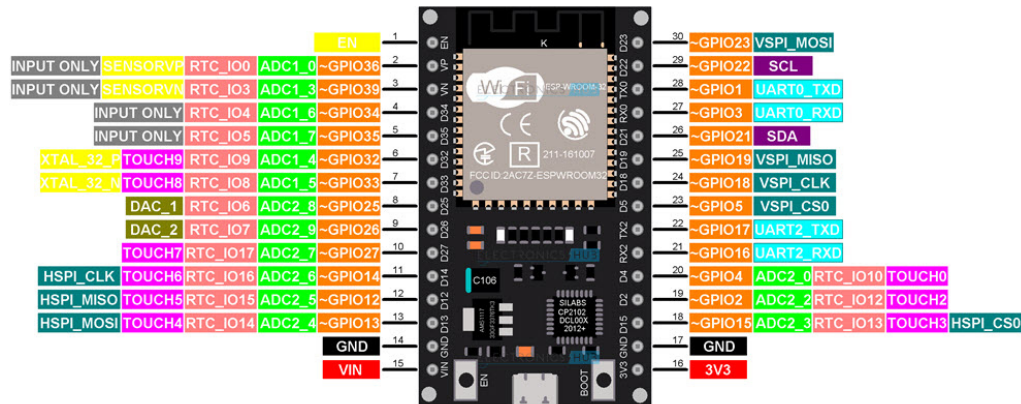


Figure 3.24: Pin Configuration of NodeMCU ESP32 Board (Ashwak, 2021)

The NodeMCU ESP32 has 30 pins in total, the majority of which are multipurpose. The pins are two power pins, two ground pins, one enable pin, 25 GPIO pins that can be assigned for various functions, 12 ADC channels, two digital-to-analogue converter (DAC) channels, UART pins that provide two UART interface functions, eight SPI pins that provide two SPI buses of high-speed parallel interface (HSPI) and VSPI; serial data line (SDA) pin and serial clock line (SCL) pin that provides a single I2C bus, and nine capacitive touch-sensing GPIO pins. The summary of the pins is included in Table 3.2. After figuring out the available pins on the ESP32 board, the tasks are allocated to the pins needed in order to control the external hardware. The pin allocations of the NodeMCU ESP32 board are included in Table 3.3.

Table 3.2: Pin Definition of NodeMCU ESP32

Pin No.	Name	Type	Function
1	EN	I	Module-enable signal (active high)
2	SENSORVP	I	GPIO36, ADC
3	SENSORVN	I	GPIO39, ADC
4	IO34	I	GPIO34, ADC
5	IO35	I	GPIO35, ADC
6	IO32	I/O	GPIO32, ADC, TOUCH, XTAL
7	IO33	I/O	GPIO33, ADC, TOUCH, XTAL
8	IO25	I/O	GPIO25, ADC, DAC
9	IO26	I/O	GPIO26, ADC, DAC
10	IO27	I/O	GPIO27, ADC, TOUCH
11	IO14	I/O	GPIO14, ADC, TOUCH, HSPI_CLK
12	IO12	I/O	GPIO12, ADC, TOUCH, HSPI_MISO
13	IO13	I/O	GPIO13, ADC, TOUCH, HSPI_MOSI
14	GND	P	Ground
15	VIN	P	Input voltage
16	3V3	P	Power supply
17	GND	P	Ground
18	IO15	I/O	GPIO15, ADC, TOUCH, HSPI_CS0
19	IO2	I/O	GPIO2, ADC, TOUCH
20	IO4	I/O	GPIO4, ADC, TOUCH
21	IO16	I/O	GPIO16, UART_RXD
22	IO17	I/O	GPIO17, UART_TXD
23	IO5	I/O	GPIO5, VSPI_CS0
24	IO18	I/O	GPIO18, VSPI_CLK
25	IO19	I/O	GPIO19, VSPI_MISO
26	IO21	I/O	GPIO21, SDA
27	RXD0	I/O	GPIO3, UART_RXD
28	TXD0	I/O	GPIO1, UART_TXD
29	IO22	I/O	GPIO22, SCL
30	IO23	I/O	GPIO23, VSPI_MOSI

Table 3.3: Pins Allocation of NodeMCU ESP32 Board

Pin No.	Name	State	Task Allocated
1	EN	-	-
2	SENSORVP	Input	Limiter switch signal
3	SENSORVN	Input	Instant watering button signal
4	IO34	Input	Analogue input from capacitive soil moisture sensor A
5	IO35	Input	Analogue input from capacitive soil moisture sensor B
6	IO32	Output	Digital output signal to turn on green LED when stepper motor is moving and turn off when stepper motor is stopping
7	IO33	Output	Digital output signal to turn on red LED when stepper motor is stopping and turn off when stepper motor is moving
8	IO25	-	-
9	IO26	Input	Analogue input from echo pin of ultrasonic sensor to receive the pulse signal.
10	IO27	Output	Digital output signal to trigger pin of ultrasonic sensor to trigger it to send pulse signal
11	IO14	Output	Digital output signal to turn on/off the water pump
12	IO12	Output	Digital output signal to turn on green LED when water pump is on and turn off when water pump is off
13	IO13	Output	Digital output signal to turn on red LED when water pump is off and turn off when water pump is on
14	GND	Power	Ground
15	VIN	Power	Direct supply with regulated 5V input voltage
16	3V3	Power	Supply regulated 3.3 V voltage to sensors, display and more

17	GND	Power	Ground
18	IO15	-	-
19	IO2	-	-
20	IO4	Input	Analogue input from DHT22 sensor
21	IO16	-	-
22	IO17	Output	Digital signal to pin 4 of motor driver board (IN4)
23	IO5	Output	Digital signal to pin 3 of motor driver board (IN3)
24	IO18	Output	Digital signal to pin 2 of motor driver board (IN2)
25	IO19	Output	Digital signal to pin 1 of motor driver board (IN1)
26	IO21	Output	Serial data line to OLED display
27	RXD0	-	-
28	TXD0	-	-
29	IO22	Output	Serial clock line to OLED display
30	IO23	-	-

3.6 Cost Estimation

The cost estimation is provided in this section to give a budget to this project, so that this project is built with sufficient funds. The cost of this entire project is estimated to be around RM 327. The details of the cost estimation are shown in Table 3.4.

Table 3.4: Cost Estimation of This Project

A. Main Part		
	Item	Cost (RM)
1.	Development Board	30.00
Total Estimated Cost of Main Part: RM 30.00		
B. IoT Monitoring System		

	Item	Cost (RM)
1.	Humidity and Temperature Sensor	20.00
2.	Capacitive Soil Moisture Sensor (Corrosion Resistance)	60.00
3.	Ultrasonic Sensor used for water level detection	10.00
Total Estimated Cost of IoT Monitoring System: RM 90.00		
C. Physical Display or Control System		
	Item	Cost (RM)
1.	OLED	15.00
2.	Push Button	2.00
Total Estimated Cost of Physical Display or Control System: RM 17.00		
D. Irrigation System		
	Item	Cost (RM)
1.	Belt	10.00
2.	Pulley	5.00
3.	Water Pump with its Driver Board	20.00
4.	Stepper Motor with its Driver Board	15.00
Total Estimated Cost of Irrigation System: RM 50.00		
E. Watering Model		
	Item	Cost (RM)
1.	Platform to hold the Stepper Motor	10.00
2.	PVC Pipe	50.00
3.	Corner Holder	40.00
4.	Water Tube	5.00
Total Estimated Cost of Watering Model: RM 105.00		
F. Power and Other Components		
	Item	Cost (RM)
1.	Power Supply Adapter	10.00
2.	Wires	5.00
3.	Others	50.00
Total Estimated Cost of Power and Other Components: RM 65.00		
Total Estimated Cost: RM 327.00		

Testing and troubleshooting watering system																	
Reporting																	

Table 3.7: Gantt Chart of FYP 2

Task \ Week	Week														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Literature review	█	█	█	█	█										
Program of watering system	█	█	█	█											
Building model of watering system	█	█	█												
Improvement on complete system	█	█	█	█	█	█	█	█	█	█					
Discussion with supervisor	█	█	█	█	█	█	█	█	█	█					
Reporting	█	█	█	█	█	█	█	█	█	█	█				
Presentation preparation											█	█			
Poster design												█			
FYP 2 final report submission												█			
Poster submission													█		
FYP 2 oral presentation														█	█

3.8 Summary

In this chapter, the selection of the development board that is applied to the system has been discussed first. The development board is the Node MCU ESP32 DEVKIT V1 DOIT, which has the best cost and flexibility of function. Next, the IoT monitoring system is discussed in the second section, which selects Blynk IoT as the IoT platform, the capacitive soil moisture sensor to detect the soil moisture, the DHT22 sensor to detect the air humidity and air temperature, and the model HC-SR04P ultrasonic sensor to detect the water level in the water tank.

After that, the third section is the irrigation system. This section discussed the model of the irrigation system. The sketch of the model is shown in this section to give a better illustration of the project. The parts to build the irrigation system model are

also included. The R385 DC12 V diaphragm water pump and 12 V 28BJ-48 stepper motor with the driver board are used in this project to complete the irrigation system.

In addition, the fourth section is the discussion on the system's working principle. The block diagram has been included in this section to connect every component of the entire system. In this section, the flowcharts are also drawn to show the operating flow of every predefined process and the complete system. The pin allocation is also included in this section. Furthermore, the cost estimation has been done in the following section to set a budget for this project. The last section is the management of the project. The Gantt charts are included in this section to demonstrate the project milestones.

In the next chapter, Chapter 4, the results and discussion of this project will be discussed. The hardware and software implementation will be indicated in the next chapter. Next, the system and cost of the project will also be analysed.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, six sections are discussed to show the outcomes of this project. Firstly, the preliminary result is shown at the beginning of this chapter. In the preliminary results section, the tasks that are carried out in the early phase of the project are discussed, which help to smooth the process of this project. Next, the hardware implementation is discussed in the second section. This section shows all the hardware that is used to build the physical display system, the IoT monitoring system, and the irrigation system. Besides that, the prototype of the watering model, the prototype board, and the printed circuit board are also included in this section. After that, the software implementation of this project is discussed in the third section. This section is about the user interface and automation that are applied in this system. Furthermore, the fourth section is the system analysis. The system is operated with a number of tests. The overall accuracy of the system is also included in this section. Moreover, the fifth section analyses the energy-saving efficiency of this system. Lastly, a cost analysis is carried out to calculate the cost of the entire project.

4.2 Preliminary Result

The preliminary task has been carried out in the early phase of this project to expedite the outcomes. The IoT monitoring system, physical display, and control system have

been built on a breadboard during this phase. These systems are built to test the connectivity between the IoT platform and the development board to achieve control action. Thus, troubleshooting and modifications can be performed in the future to make this project more effective. This project is able to purchase some main hardware and apply it to the breadboard.

Figure 4.1 shows the connection of the hardware on the breadboard. In Figure 4.1, the LEDs indicate the status of the water pump and stepper motor, the OLED displays the condition of the plants, the DHT22 sensor measures the air humidity and air temperature, the push button gives a watering order, and the development board is connected to the IoT cloud. Figure 4.2 shows the air humidity, air temperature, and soil moisture displayed on the OLED display. The temperature and humidity of the air, animated emoji, and percentage of soil moisture are shown on the OLED display. The user interface for the Blynk IoT app is shown in Figure 4.3. The conditions of the plant are displayed on the Blynk IoT app. In the first tab of the Blynk IoT app, the air humidity and air temperature are tabulated into a graph, from which this project can observe the change in the values. The status of the water pump and stepper motor are displayed on the next tab, along with a button to give the irrigation order.

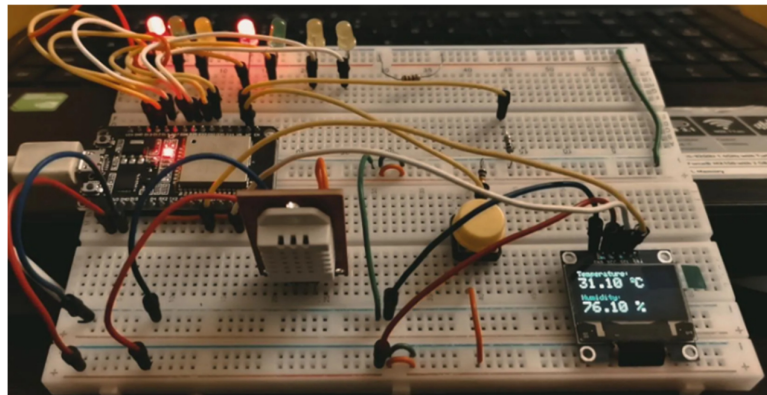


Figure 4.1: Hardware Connection on the Breadboard

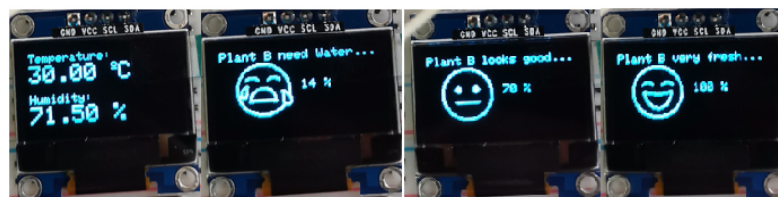


Figure 4.2: Contents of the OLED Display

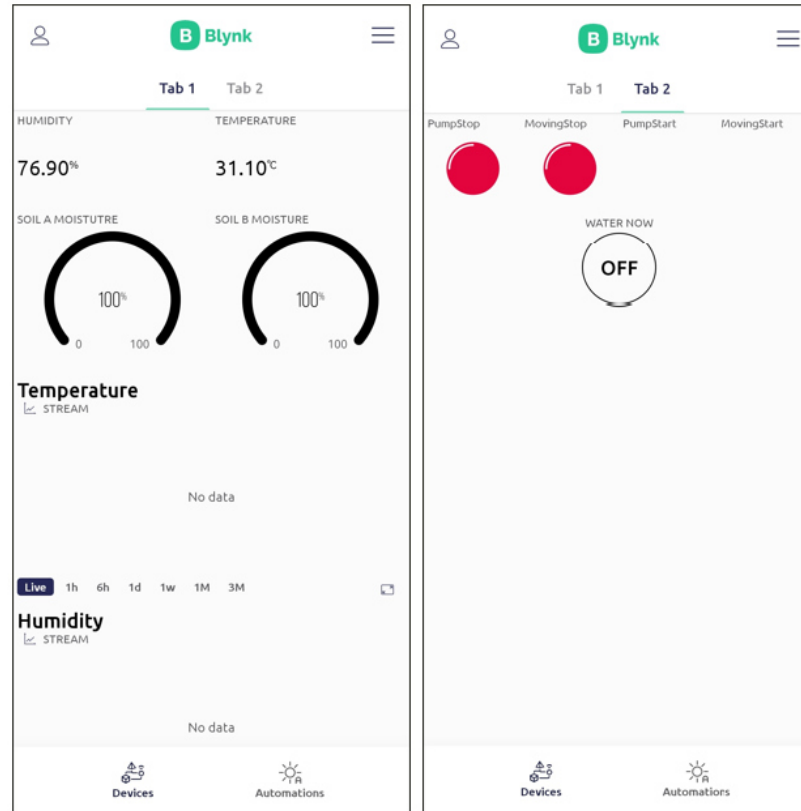


Figure 4.3: User Interface on the Blynk IoT App

4.3 Hardware Implementation

In this section, how the connection of every piece of hardware is built to offer functionality to the system accordingly is discussed. Every piece of hardware is connected together by referring to the system architecture design in Chapter 3.5.1 and the block diagram in Figure 3.18. In addition, the pin allocation in Table 3.3 is used as the preference to connect every piece of external hardware with the development board. Besides that, the prototype of the watering model, the prototype board, and the printed circuit board are also included in this section.

4.3.1 Physical Display System

Figure 4.4 shows the physical display system that is constructed based on the block diagram in Figure 3.18. The hardware in this system is an OLED display and four LEDs. The OLED display shows the condition of the plants with the same contents that have been completed in the preliminary task, as shown in Figure 4.2. Next, the four LEDs are classified into two combinations, one red and one green. The combination of the red and green LEDs on the left indicates the status of the water pump, while the combination of the red and green LEDs on the right indicates the status of the stepper motor.

Figure 4.5 shows the operation of the LEDs in three different situations. The first scenario occurs when the moving bar moves to the target position while the watering action continues. Both green LEDs of the LED combination are lit up. Next, in the second situation, the moving bar moves back to its original position after the watering action. So, the left LED combination is changed from green to red, indicating the moving bar is moved without the watering action. The third situation is when the moving bar stays stationary at the origin. Both red LEDs of the LED combination are lit up.

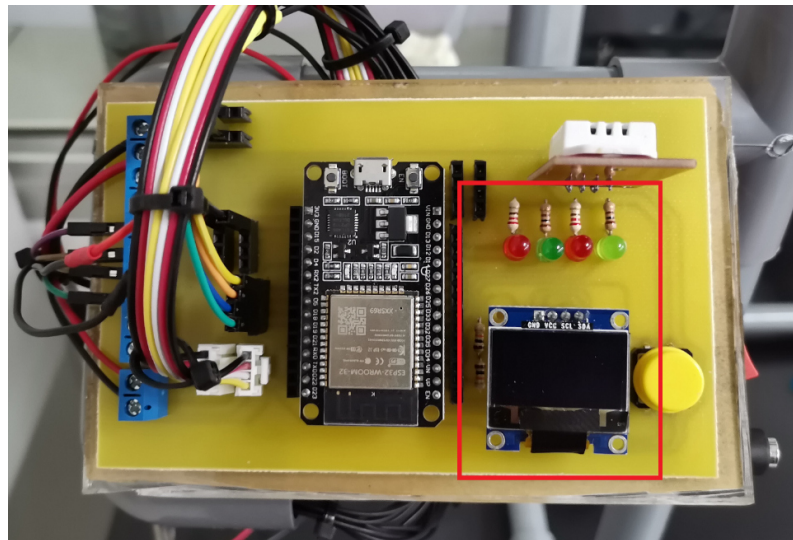


Figure 4.4: Hardware of the Physical Display System

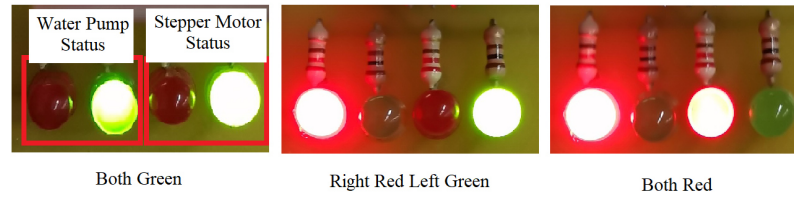


Figure 4.5: Operation of the Status Indicator LEDs

4.3.2 IoT Monitoring System

Figure 4.6 shows the IoT monitoring system that is constructed based on the block diagram in Figure 3.18. The hardware components of this system are a DHT22 humidity and temperature sensor, two capacitive soil moisture sensors, and an ultrasonic sensor. Firstly, the DHT22 sensor is installed on the PCB to measure the air humidity and air temperature in a higher position. Then, the ultrasonic sensor is installed on the top of the water tank to detect the water level. Lastly, two capacitive soil moisture sensors are placed inside the soil of the plant.

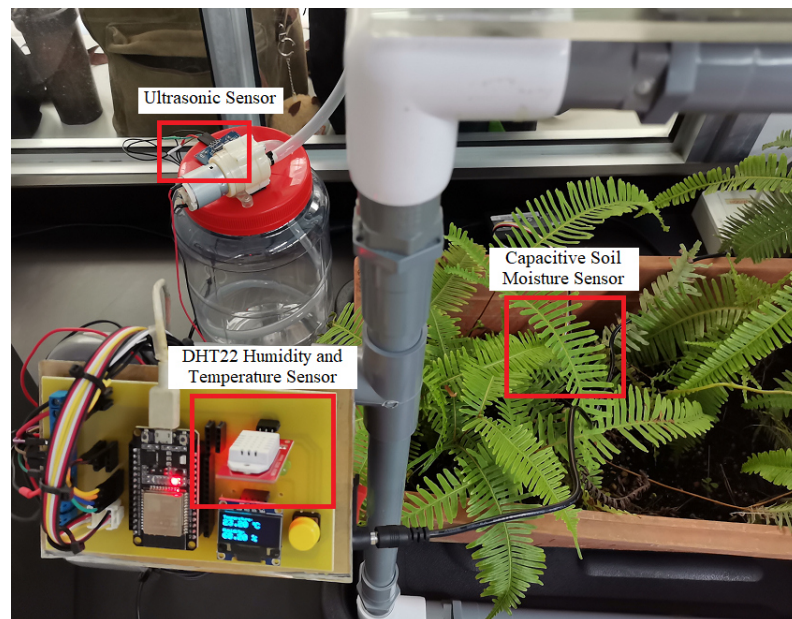


Figure 4.6: Hardware of the IoT Monitoring System

4.3.3 Irrigation System

Figure 4.7 shows the irrigation system that is constructed based on the block diagram in Figure 3.18. The hardware components of this system are a stepper motor with its driver board, a water pump with its driver board, a water tube, a push button, and a limit switch. The stepper motor is able to pull the moving bar with the help of a belt and the pulleys. The water is transferred through the water tube from the water tank to the moving bar by using the water pump. Next, the push button and the limit switch are operated as signal providers to activate and deactivate the system. Figure 4.8 shows the limiter switch which has been attached under the platform of the stepper motor. When the moving bar holder touches the limiter switch, the system is deactivated.



Figure 4.7: Hardware of the Irrigation System



Figure 4.8: Position of the Limiter Switch

4.3.4 Prototype

Figure 4.9 shows the prototype of this project. This prototype is constructed to showcase the operation of the complete system in this project. The dimensions of the prototype are 65 cm ' 55 cm ' 35 cm in XYZ representation. As shown in Figure 4.9, the control system is located on the right side, above the water tank. In addition, there are two platforms that have been installed on top of the prototype to support the stepper motor and pulley holders. The moving bar is moved by pulling the belt in the middle. Besides that, the plant is placed in the middle of the prototype.



Figure 4.9: Prototype of the Smart Agriculture and Smart Irrigation System

4.3.5 Prototype Board and Printed Circuit Board

Figure 4.10 shows the prototype board, and Figure 4.11 shows the printed circuit board (PCB) of the complete system. Both the prototype board and PCB are created based on the board design as shown in Appendix B. Firstly, this project is planned to implement the prototype board only with an enclosure to separate the display system from the development board. However, the Wi-Fi connection is affected after applying the system to the prototype board, as the development board is placed inside the enclosure which the data interface speed with the IoT cloud is slow. Besides that, the

cable usage on the prototype board is minimal as compared to the breadboard, but the connection error occurred due to multiple connection points in a pin.

Thus, the decision to make a PCB is made to avoid any connection errors due to soldering skill, breadboard, or wires. Only a PCB is printed and applied in this project, on which the development board and display system are combined. Besides that, the development board is also exposed to the air to minimise the time delay of data transmission. The blue components on the left are the terminal blocks used to connect with the power and signal wires. Next, the female headers in the middle are signal pins for the stepper motor, and the grove terminals at the bottom are used as power and signal pins for the soil moisture sensor. The physical and control system is on the right of the development board. Lastly, the implementation of the PCB gives the circuit a tidy appearance and has better wire management for future maintenance.

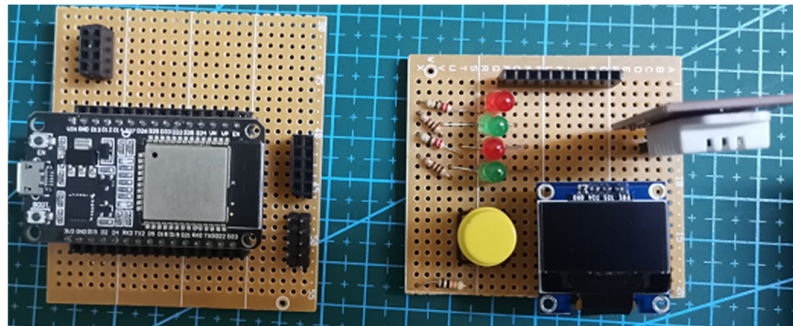


Figure 4.10: Prototyped Board

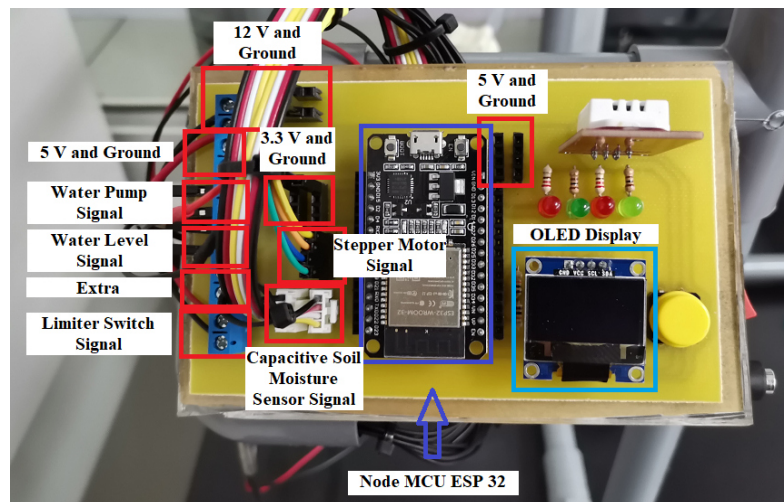


Figure 4.11: Printed Circuit Board

4.4 Software Implementation

The C++ programming language is used in this project to build the system. Besides that, the Arduino IDE is used as the platform to develop the software code of the system. The Arduino IDE provides the library for both the Blynk and ESP32 development board. By using the Arduino IDE, this project is able to programme the hardware and build the connection between the ESP32 development board and the Blynk. Code Listing 4.1 shows the code that is used to set the connection between the ESP32 development board and the Blynk. The complete system programming code is shown in Appendix A.

Code Listing 1: Code to Set Connection Between the Blynk and the ESP32 Development Board

```
#define BLYNK_TEMPLATE_ID "TMPL7hEMVYkv"
#define BLYNK_DEVICE_NAME "FYP"
char auth[] = "2kibXNYB0toNBcrq8NxbYRLRWK5-oAUu";
//Authentication code sent by Blynk
char ssid[] = "taytang";           //WiFi SSID
char pass[] = "1234567890";       //WiFi Password
```

4.4.1 User Interface

Blynk provides a web dashboard via the Blynk console on the computer and a mobile dashboard via the Blynk IoT app on the smartphone to build the user interface manually, as mentioned in Chapter 3.3.1. The developer mode on the Blynk console and Blynk IoT app allows this project to include possible widgets. The widgets are the button, the value displays, the gauges, the LEDs, the sliders, and the super charts. The widgets are linked to the data streams, which are the virtual pins allocated in Table 3.1. Figure 4.12 shows the web dashboard design on the Blynk console, and Figure 4.14 shows two tabs of the mobile dashboard on the Blynk IoT app. There are minor changes that have been made on the mobile dashboard, as shown in Figure 4.3, due to the addition of features and the changes in Blynk's terms on February 2023 where free users are limited to the usage of ten data streams.

On the first tab of the mobile dashboard, the air humidity, air temperature, soil moisture in areas A and B, and water level are displayed using value displays and gauges. After that, the four virtual LEDs used to display the status of the water pump and the stepper motor shown in Figure 4.3 are reduced to two and moved to the first tab. When the water pump or the stepper motor is activated, the LED is turned green, as shown in Figure 4.13. Next, the super charts shown in Figure 4.3 are moved to the second tab. The second tab of the mobile dashboard displays the super charts only. Four super charts are included to visualise live and historical data on the humidity, the temperature, and the soil moisture in areas A and B.

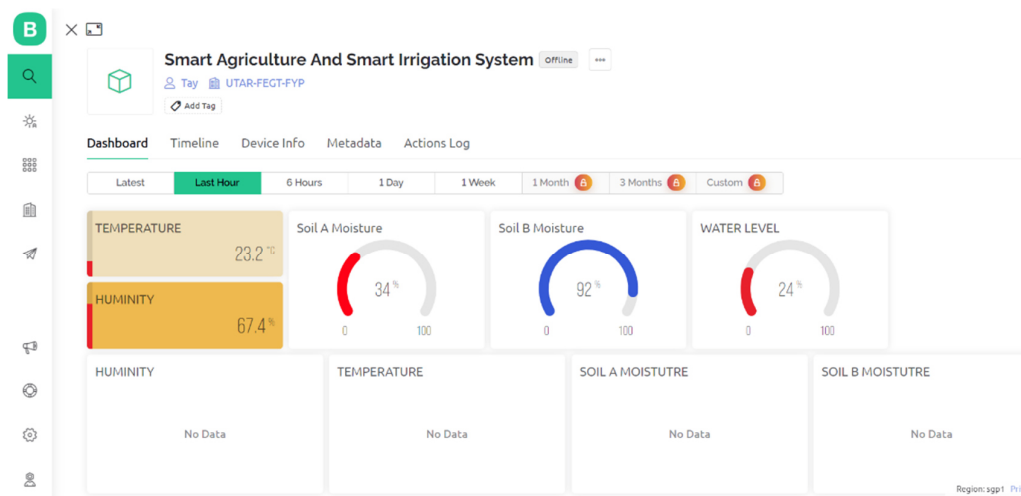


Figure 4.12: Web Dashboard on the Blynk Console



Figure 4.13: Operation of the Virtual LEDs

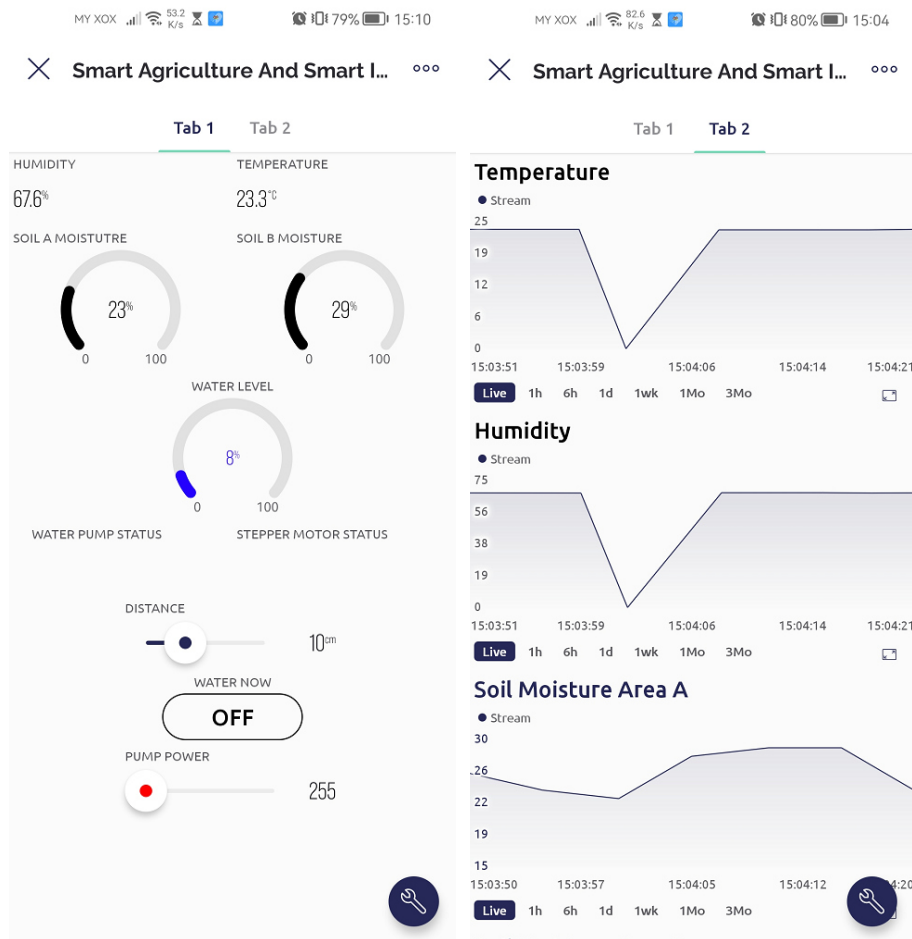


Figure 4.14: Mobile Dashboard on the Blynk IoT App

4.4.2 Automations

The automation is achieved by utilising Blynk. Figure 4.15 shows the automation page on the Blynk IoT app, which can be activated. Three automations are added to this system. The first two automations are named "Soil A" and "Soil B". Both of these automations detect the soil moisture percentage in areas A and B; four actions are carried out automatically when the soil moisture falls below 30 %. Firstly, when the soil moisture is detected as being lower than 30 %, notification is sent to the smartphone. Then, the flow of the smart irrigation system is operated as mentioned in Chapter 3.5.5. The notification that is sent to the smartphone is shown in Figure 4.16. Besides that, the date and time of the latest activation of the automation are also shown.

The "Water Level" automation detects the water level percentage and sends the notification to the smartphone when the water level percentage is below 15 %.

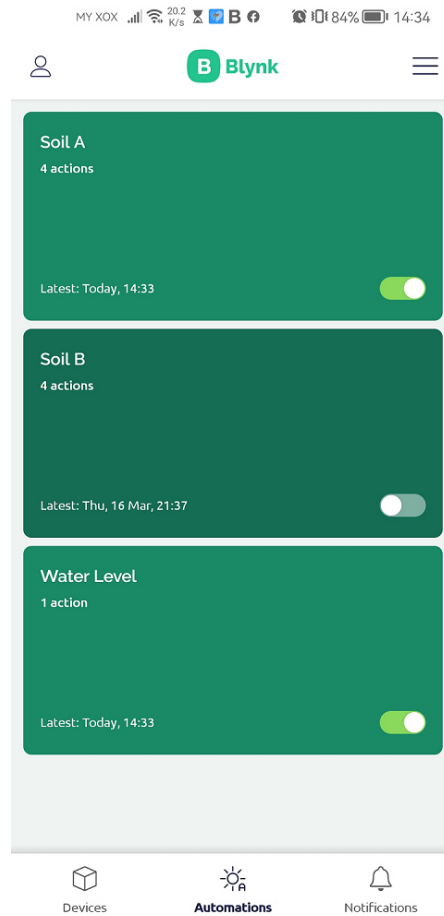


Figure 4.15: Automation Page on the Blynk IoT App

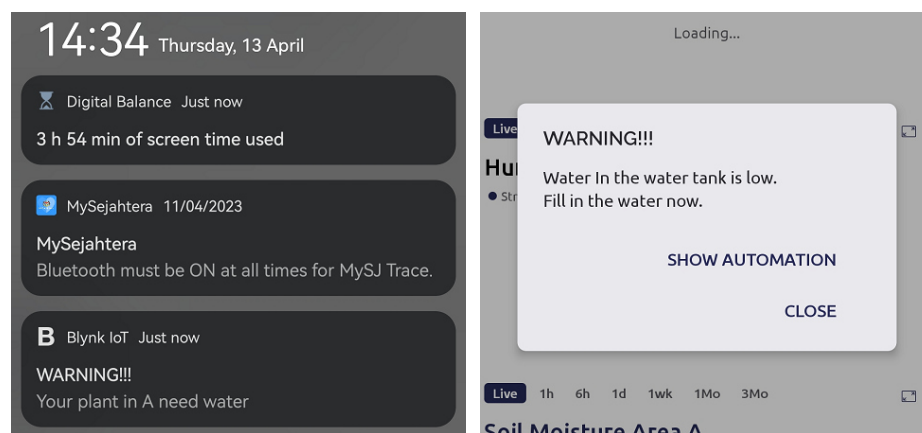


Figure 4.16: Notifications of the Automation

4.5 System Analysis

In this subsection, various analyses are performed and recorded. The analyses performed on the system are humidity and temperature detection range analysis, soil moisture detection range analysis, display system analysis, control system analysis, smart irrigation system analysis, and Wi-Fi analysis. The purpose of these analyses is to evaluate and verify the performance of the system. Besides that, the system analysis is to test the solidity of the system by using the accuracy formula, which is the number of successful tests divided by the total number of samples taken. In addition, the energy-saving efficiency of the smart irrigation system is analysed to estimate the amount of energy saved. Lastly, the cost of this project is analysed to check whether it is over the budget mentioned in Chapter 3.6.

4.5.1 Humidity and Temperature Detection Ranges Analysis

The humidity and temperature detection range analysis has taken place in an air-conditioned environment with a temperature of 24 °C. Overall, the temperature detected is 23.5 °C, which is within the accuracy range of ± 0.5 °C as mentioned in Chapter 2.6.1. After that, a hot and wet towel is placed near the DHT22 sensor with a distance of 5 cm, 10 cm, 15 cm, 20 cm, 30 cm, and 40 cm. The towel is renewed every time a test is conducted to increase the accuracy of the data that is taken. This analysis shows that this system is able to detect the humidity and temperature around 20 cm. This is because the humidity and temperature readings are almost the same as the readings taken without the hot and wet towel while the towel reached 30 cm. The readings of the DHT22 sensor are shown in Table 4.1.

Table 4.1: Readings of the DHT22 Sensor

Readings Without the Hot and Wet Towel is Applied	
Humidity (%)	Temperature (°C)
67.9	23.5
Readings With the Hot and Wet Towel is Applied	

Distance of the Hot and Wet Towel (cm)	Readings	
	Humidity (%)	Temperature (°C)
5	70.6	25.0
10	69.5	24.5
15	68.5	24.3
20	68.0	24.0
30	67.7	23.9
40	67.5	23.4

4.5.2 Soil Moisture Detection Range Analysis

The area of the soil moisture that can be detected by the capacitive soil moisture sensor is tested in this analysis. This analysis is performed by pouring 100 ml of water onto the soil around the sensor from 20 cm to 5 cm, respectively. The results of this analysis show that the effective soil moisture detection range is around 10 cm from the sensor. Thus, the plants can be planted in this range to get the best soil condition detection in this system.

Table 4.2: Readings of the Capacitive Soil Moisture Sensor

Soil Moisture Percentage Without the Water is Added (%)	
25	
Soil Moisture Percentage When the Water is Added	
Distance of the Water Added (cm)	Soil Moisture Percentage (%)
5	47
10	40
15	31
20	25

4.5.3 Display System Analysis

The display system analysis is done on the physical display and the virtual display on the Blynk IoT app. As mentioned in Chapter 3.5.2, the physical display system is operated with an OLED display and four LEDs. Then, the OLED display displays the humidity and temperature for 4 seconds and continues with the display of soil moisture percentage in areas A and B for 5 seconds each. Thus, the 14 seconds of display are classified as a cycle. In this analysis, the contents displayed on the OLED display are checked for 15 cycles. Besides that, the virtual display is analysed by using the super charts widgets, as the data is tabulated in the charts. The historical data for the humidity, the temperature, and the soil moisture is taken 15 times continuously over a time range.

From the analysis, the OLED display is unable to display the humidity and temperature in the 6th cycle, while the soil moisture percentage in areas A and B is displayed successfully throughout the test. The potential reason for the error in the 6th cycle to display the humidity and temperature is the noise that violated the interface between the development board and the sensor. Next, the data taken from the super chart on the Blynk IoT app proved that the humidity, temperature, and soil moisture in areas A and B were successfully displayed throughout the test. In addition, the water level display is observed at the same time directly from the display as there is no super chart to tabulate the data. Table 4.3 shows the analysis results for the physical display and the virtual display. After that, the analysis is continued by testing the four physical LEDs and the two virtual LEDs on the Blynk IoT app that display the status of the stepper motor and the water pump. The operations of the physical LEDs and virtual LEDs are shown in Figures 4.5 and 4.14. The analysis is done 15 times with the irrigation system activated. The physical LEDs performed correctly throughout the test. In contrast, the virtual LEDs fail to show the status of the stepper motor and the water pump at the 5th test when the watering action is performed. The reason is that the mobile Wi-Fi signal strength was weak during the 5th test. Then, the problem is solved by using a strong mobile data plan while using the Blynk IoT app. The result of the analysis is recorded in Table 4.4. Lastly, the overall accuracy of the display system is tabulated in Table 4.5.

Table 4.5: Accuracy of the Display System

Conditions of the Surrounding and the Plants		
Conditions	Accuracy	
	Physical Display	Virtual Display
Humidity and Temperature	93.33 %	100 %
Soil Moisture Percentage in Area A	100 %	100 %
Soil Moisture Percentage in Area B	100 %	100 %
Water Level	-	100 %
Status of Stepper Motor and Water Pump		
Conditions	Accuracy	
	Physical LEDs	Virtual LEDs
No action	100 %	100 %
Watering action is performed	100 %	93.33 %
Moving bar moving back to starting position	100 %	93.33 %

4.5.4 Irrigation System Analysis

In this subsection, the operation of the irrigation system as mentioned in Chapter 3.5.4 is tested. The physical push button and the virtual push button are pressed 15 times. Then, the irrigation system is observed to verify that the operations are carried out successfully. The observations included the activation of the stepper motor, the activation of the water pump, the movement of the moving bar, the ability of the moving bar to reach the target distance, and the ability of the moving bar to stop when touching the limit switch. Next, there are three sets of results that are analysed using the different target distances of 10 cm, 20 cm, and 30 cm. Thus, three lines with 10 cm each are marked on the moving bar path, as shown in Figure 4.17. Lastly, all the operations of the irrigation system have taken place successfully throughout the analysis. The results of the system analysis are shown in Table 4.6, and the overall accuracy of the irrigation system operation is shown in Table 4.7.

O N A L	T U A L	Moving back and stop at starting point (stepper motor activate; water pump deactivate)	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
		Target Distance: 30 cm																
		No. Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
P U S H B U T T O N	P H Y S I C A L	Moving and reaching target distance (stepper motor and water pump activate)	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
		Moving back and stop at starting point (stepper motor activate; water pump deactivate)	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
	V I R T U A L	Moving and reaching target distance (stepper motor and water pump activate)	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
		Moving back and stop at starting point (stepper motor activate; water pump deactivate)	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/

Table 4.7: Accuracy of the Irrigation System

Operations	Accuracy	
	Physical Button	Virtual Button
Moving and reaching target distance (stepper motor and water pump activate)	100 %	100 %
Moving back and stop at starting point (stepper motor activate; water pump deactivate)	100 %	100 %

4.5.5 Automations Analysis

The two automations that have been mentioned in Chapter 4.4.2 are used to make the irrigation system smart, and the other automation is used for water level detection. The operation of the smart irrigation system is described in Chapter 3.5.5. Thus, the analysis in this subsection is undertaken to verify the ability of the automations to

Table 4.9: Accuracy of the Automations

Name of Automations	Accuracy
“Soil A”	93.33 %
“Soil B”	93.33 %
“Water Level”	100 %

4.6 Efficiency on Water Saving Analysis

In this section, the efficiency of water savings is investigated for the smart irrigation system. This analysis aims to estimate the amount of water is saved when the smart irrigation system is activated based on the soil moisture percentage. During the analysis, the soil moisture in area B is measured continuously, and the change in the soil moisture percentage is observed from the super chart on the Blynk IoT app. The first case is studied to determine the times needed for the soil moisture in area B that is in a cold environment to drop below 30 % and the second case is studied to determine the times needed for the soil moisture in area B that is in a room temperature environment to drop below 30 %. Then, the amount of water that is saved in both cases is also mentioned.

4.6.1 Case Study 1: Cold Environment

In this case study, this project with the plants is allocated to a cold environment where the average air temperature is 24 °C. To begin, the soil moisture of the plants is increased to 80%. Then, the plants are left in the environment, and the change in soil moisture is measured by the IoT monitoring system. After some days, the change in soil moisture of the plants is observed from the super chart via the Blynk IoT app. Figure 4.18 shows that the soil moisture is at 80 % as the water is added to the soil on February 26 and drops below 30 % on March 3. So, the soil moisture drops below 30 % in 5 days when the plants are placed in the cold environment. The historical data is displayed by the super chart on the Blynk IoT app is shown in Figure 4.18.

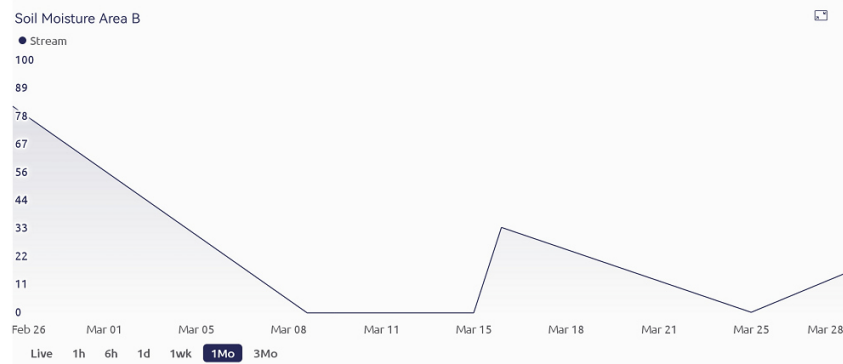


Figure 4.18: Soil Moisture Percentage in Area B from 26 February to 3 March

4.6.2 Case Study 2: Room Temperature Environment

In this case study, this project with the plants is allocated to the hostel, where the average air temperature is 32 °C. Firstly, the soil moisture of the plants is increased by 80 %. Then, the plants are left in the environment, and the change in soil moisture is measured by the IoT monitoring system. After some days, the change of the soil moisture of the plants is observed from the super chart via the Blynk IoT app. Figure 4.19 shows that the soil moisture is 80 % as the water is added to the soil on April 6, and the soil moisture is below 30 % on April 9. So, the soil moisture drops below 30 % in 3 days when the plants are placed in the room-temperature environment. The historical data displayed by the super chart on the Blynk IoT app is shown in Figure 4.19.

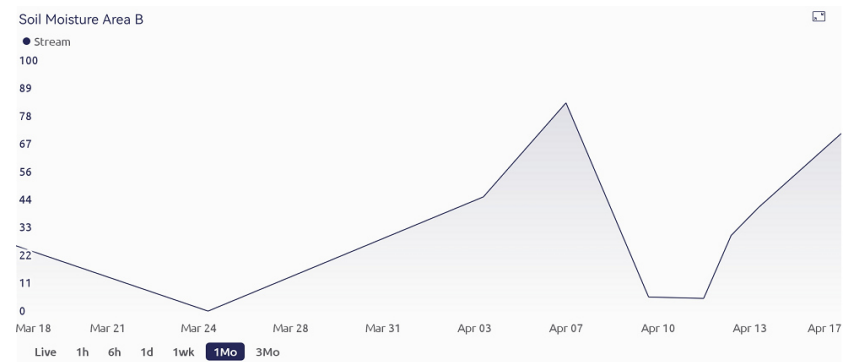


Figure 4.19: Soil Moisture Percentage in Area B from 6 April to 9 April

4.6.3 Total Water Saving

Before both cases are studied, the amount of water pumped to the plants during each irrigation is measured. Therefore, 400 mL of water is pumped to the plants with the lowest power of the water pump, and this amount of water is able to increase the soil moisture percentage from 30 % to 80 %. Assuming that irrigation is carried out every morning manually without the smart irrigation system, the amount of water needed in a month will be 12.4 L (400 mL \times 31 days).

Next, the smart irrigation system is activated automatically when the soil moisture percentage drops below 30 %. When the plants are allocated to the cold environment, this system is estimated to activate six times per month based on the time estimation for the soil moisture percentage to drop below the threshold as mentioned in Chapter 4.6.1. Therefore, 2.4 L (400mL \times 6 times) of water are needed for the plants watering, and 10 L of water are saved by using the smart irrigation system as compared to watering the plants daily for a month. In contrast, this system is activated 10 times per month based on the time estimation as mentioned in Chapter 4.6.2 when the plants are placed in the room-temperature environment. Thus, 4 L (400mL \times 10 times) of water is needed for the plants watering, and 8.4 L of water is saved by using the smart irrigation system as compared to watering the plants daily in a month.

Lastly, the smart irrigation system saves 80.65 % of water when the plants are placed in a cold environment, while 67.74 % of water is saved when the plants are placed in a room-temperature environment. Table 4.10 shows the usage, savings amount, and percentage savings of water by the smart irrigation system in cold and room temperature environments.

Percentage of water saving

$$= \frac{\textit{Water saved by the smart irrigation system in a month}}{\textit{Water needed without the smart irrigation system in a month}} \times 100 \%$$

Table 4.10: Water Usage and Saving

Without the Smart Irrigation System		
Water usage (monthly)	12.4 L	
With the Smart Irrigation System		
	Environment	
	Cold	Room Temperature
Water usage (monthly)	2.4 L	4 L
Water saving (monthly)	10 L	8.4 L
Percentage of water saving (monthly)	80.65 %	67.74 %

4.7 Cost Analysis

The cost analysis is provided in this section to give a cost overview of this project. The cost of this entire project is around RM 286.60. This analysis shows that this project is within the budget, and the fund usage is below RM 327 as estimated in Chapter 3.6. The details of the cost analysis are shown in Table 4.11.

The cost is dynamic based on the watering model. This is because the watering model can be built based on the needs. So, Equation 4.1 below is used to determine the cost based on the size of the watering model.

$$\text{Total cost} = \text{RM } 276 + \left\{ \left[\frac{(4X+4Y+4Z)-320\text{cm}}{30.48 \text{ ft}} \right] \times \text{RM}1 \right\} \quad (4.1)$$

Where

X = length of the watering model, cm

Y = height of the watering model, cm

Z = width of the watering model, cm

Table 4.11: Cost Analysis of the Entire Project

A. Main Part					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	NODEMCU ESP32	Development Board	29.00	1	29.00
Total Cost of Main Part: RM 29.00					
B. IoT Monitoring System					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	DHT22	Humidity and Temperature Sensor	15.50	1	15.50
2.	Grove Capacitive Soil Moisture Sensor	Capacitive Soil Moisture Sensor (Corrosion Resistance)	29.80	2	59.60
3.	3 V to 5.5 V SR04P Ultrasonic Ranging Module	Ultrasonic Sensor used for water level detection	4.90	1	4.90
4.	Blynk IoT	IoT Platform	-	-	-
Total Cost of IoT Monitoring System: RM 80.00					
C. Physical Display or Control System					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	OLED	0.91" 128 × 32 I2C IIC OLED LCD LED Blue Graphic Display Module	14.90	1	14.90
2.	LEDs	Red and Green (3 mm)	0.10	4	0.40
3.	Resistors	100 Ω, 200 Ω, 10k Ω	0.05	6	0.30
4.	Push Button	12 mm × 12 mm × 1 mm	1.60	1	1.60
Total Cost of Physical Display or Control System: RM 17.20					
D. Irrigation System					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	GT2 Belt 6 mm	Belt used to pull moving bar	10.00	1 Meter	10.00
2.	GT2 Pulley 20 Teeth 5 mm Bore	Pulley attached to Stepper Motor	4.60	1	4.60
3.	GT2 Pulley Toothless Idler 5mm Bore	Combination with GT2 Pulley 20 Teeth 5 mm Bore and Belt to pull the moving bar	4.50	1	4.50

4.	R385 DC 12 V Diaphragm Water Pump	Water Pump	10.90	1	10.90
5.	L298N 2 A DC Motor Driver	Water Pump Driver Board	5.40	1	5.40
6.	12 V 28BYJ-48 Stepper Motor with ULN2003	Stepper Motor with its Driver Board	8.80	1	8.80
Total Cost of Irrigation System: RM 44.20					
E. Watering Model					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	Acrylic Sheet	Platform to hold the Stepper Motor	15.00	1	15.00
2.	PVC Pipe	1/2"	1.00	10 Feet	10.00
3.	PVC 3 Way Elbow	3/4"	3.00	8	24.00
4.	PVC PT Socket	1/2"	0.80	24	19.20
5.	PVC V Socket	1/2"	0.50	24	12.00
6.	PVC Tube 5/16"	Tube for water supply	1.00	3 Meter	3.00
Total Cost of Watering Model: RM 83.20					
F. Power and Other Components					
	Item Name	Description	Price/Unit (RM)	Unit	Total Cost (RM)
1.	12 V 2 A AC to DC Power Supply Adapter	Power Supply Adapter	5.40	1	5.40
2.	LM2596 DC to DC Adjustable Step Down Regulator	Step Down Converter used to convert 12 V to 5 V	11.20	1	11.20
3.	Rocker Switch Small 2 Pins Red	Power Switch	1.60	1	1.60
4.	KAR301-2 Way	Terminal Block	0.70	6	4.20
5.	Straight Female Header	1 × 40 ways	1.20	4	4.80
6.	Grove 4-pin Straight Through Hole Socket	Grove Connector for Capacitive Soil Moisture Sensor	0.40	2	0.80
7.	Some Jumper Wire		2.50	2	5.00
Total Cost of Power and Other Components: RM 33.00					
Total Cost: RM 286.60					

4.8 Summary

In this chapter, the preliminary results have been discussed, including the fact that the IoT monitoring system and OLED display are built in the early phases of the project. After that, the hardware implementation results have also been considered in this chapter. This section talks about the combination of several components to operate as a system. The systems are a physical display system, an IoT monitoring system, and an irrigation system. Next, the software implementation has been clarified, which includes the user interface design and the automations. The dashboards of the Blynk console on the computer and the dashboard of the Blynk IoT app on the smartphone are designed and shown in this section. Then, three automations: "Soil A", "Soil B", and "Water Level" have been incorporated to make this system smart.

Besides that, the humidity and temperature detection ranges, the soil moisture detection range, the display system, the irrigation system, and automations have all been analysed in this chapter. Firstly, the effective detection ranges of the humidity and temperature are around 20 cm. Secondly, the capacitive soil moisture sensor is able to measure the soil moisture around 10 cm from the sensor. Thirdly, the physical and virtual displays have achieved 93.33 % accuracy in displaying the humidity and temperature and 100 % accuracy in displaying the soil moisture, status of the stepper motor, and status of the water pump. Fourthly, the irrigation system has achieved 100% accuracy on both reaching the target distance with the stepper motor and water pump activated and moving back to the starting position with the stepper motor and water pump deactivated when physical push buttons are pressed. Lastly, the automations analysis shows that the "Soil A" automation and the "Soil B" automation have achieved 93.33 % accuracy, while the "Water Level" automation has achieved 100 % accuracy. In addition, the efficiency analysis of water savings shows that the smart irrigation system saves 80.65 % of water when the plants are placed in a cold environment and 67.74 % of water when the plants are placed in a room-temperature environment. Furthermore, the cost of this project is RM 286.60. In the next chapter, Chapter 5 will conclude this project. The limitations and recommendations to improve this project will also be included in the next chapter.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the objectives of this project, listed in Chapter 1.4, have been successfully achieved. This IoT smart agriculture and smart irrigation system is able to monitor the condition of the plants from an isolated place by using the Blynk IoT app on the smartphone or the Blynk console on the computer. Besides that, the automatic watering action is also achieved when the soil moisture falls below a certain threshold. Then, the PCB board is also designed and printed in this project to connect the development board with all the external hardware. This system is divided into an IoT monitoring system, a display system, an irrigation system, and the automations.

Firstly, the IoT monitoring system has carried out measurements of the conditions by using different types of sensors, and the data are recorded by the ESP32 development board and sent to the Blynk server that is connected to the Blynk Cloud. The conditions of the plants are stored in the Blynk Cloud and displayed on the smartphone via the Blynk IoT app. Then, there are also four super charts that are utilised to show the live or historical data of the plants' conditions, so that the monitoring of the plants is more effective. Secondly, the display system is adopted to display the condition of the plants physically by using the OLED display and to display the status of the stepper motor and the water pump by using LEDs. This subsystem is coordinated with the IoT monitoring system, and the data are also displayed virtually on the smartphone. Thirdly, the stepper motor and the water pump work together as the irrigation system. A prototype of the watering model is built to showcase the

operation of the irrigation system. The irrigation system can be activated manually by using the physical push button or virtually by using the button in the Blynk IoT app. Before pressing the button, the target distance and the power of the water pump are set. Fourthly, automation is applied to this system to make it smart. The automations work together with the IoT monitoring system and the irrigation system. When the soil moisture of the plants falls below 30 %, the irrigation system is activated automatically to water the plant. At the same time, a notification is sent to the smartphone via the Blynk IoT app. Then, one of the automations measures the water level, and a notification is sent when the water level drops below 15 %.

Lastly, the analyses carried out in Chapter 4 have shown that the overall performance of this system is good and the accuracy of the test is above 90 %. This system is also efficient in terms of saving water, as the smart irrigation system saves 80.65 % of water when the plants are placed in a cold environment and 67.74 % of water when the plants are placed in a room temperature environment.

5.2 Limitations

The IoT smart agriculture and smart irrigation system is performing well if the same plants are growing in two different areas, as the same amount of water will be pumped in both areas. For instance, a small flower is planted in area A, while a tomato is planted in area B. The water demands of both plants are different, so the amount of water needed for the tomato will be higher than for the small flower. A total of 500 mL of water will be pumped to the plants in both areas. The needs of the plants are unable to be fulfilled at the same time.

Besides that, the watering action is forced to activate when moving to the target distance. For example, when the automation detects that soil moisture in area B is below the threshold, the irrigation system is activated to water the plants in area B. However, irrigation is also done to the plants in area A, which are still fresh, as the moving bar is moved along area A to reach area B.

In addition, the change in Blynk's term has limited the data streams for the free user to utilise the IoT platform. This action has reduced the functionality of this project as only 10 data points are able to be applied to this system. Lastly, the plants are not visible in this system. The size of the plants cannot be detected in this system, which may cause the system to fail as the plants get taller because it may block the moving bar from moving along and watering the plants. So, the plants still need to be visible physically in the house.

5.3 Recommendations for Improvement

In this section, some recommendations for future project improvement are suggested to enhance this project's market value. Firstly, the watering action mode can be adjusted to water the plants in the target position with a different amount of water. The target position and the amount of water can be set by using the smartphone. Besides that, a camera is also suggested to be implemented in this system to make the plant visible on the smartphone for better monitoring. The camera can be installed on the moving bar, and the plants can be recorded during the irrigation.

In addition, it is suggested that artificial intelligence (AI) be applied to this system. The AI can work together with the camera that provides vision to the system to analyse the plants. With the image processing, the AI is able to recognise the growth of the plants based on their colour or size and give suitable advice to the owner to plan the plantation. Thus, a more effective monitoring system can be developed as it is able to predict the growth of the plants.

REFERENCES

- Alam, 2019. *Interface Capacitive Soil Moisture Sensor v1.2 with Arduino*. [online] How To Electronics. Available at: <<https://how2electronics.com/interface-capacitive-soil-moisture-sensor-arduino/>> [Accessed 6 August 2022].
- Adla, S., Rai, N.K., Karumanchi, S.H., Tripathi, S., Disse, M. and Pande, S. (2020). Laboratory Calibration and Performance Evaluation of Low-Cost Capacitive and Very Low-Cost Resistive Soil Moisture Sensors. *Sensors*, [online] 20(2), p.363. Available at: <<https://www.mdpi.com/1424-8220/20/2/363#sec3dot1dot2-sensors-20-00363>> [Accessed 3 March 2023].
- Amazon Web Services, Inc., n.d. *AWS IoT Core Features - Amazon Web Services*. [online] Available at: <<https://aws.amazon.com/iot-core/features/?nc=sn&loc=3>> [Accessed 31 July 2022].
- Antonio, R., 2021. *Comparing soil moisture sensors for smart irrigation systems*. [online] Arduino Project Hub. Available at: <https://create.arduino.cc/projecthub/antonio-ruiz/comparing-soil-moisture-sensors-for-smart-irrigation-systems-caa7aa?ref=similar&ref_id=415286&offset=4> [Accessed 6 August 2022].
- Arduino, 2018. *Arduino - Introduction*. [online] Arduino.cc. Available at: <<https://www.arduino.cc/en/guide/introduction>> [Accessed 23 July 2022].
- Ashwak, 2021. *ESP32 vs ESP8266 - Which One To Choose?* [online] Electronics Hub. Available at: <<https://www.electronicshub.org/esp32-vs-esp8266/>> [Accessed 23 July 2022].

Blynk.io., 2015. *Blynk*. [online] Available at: <<https://blynk.io/>> [Accessed 31 July 2022].

BME280 -Data sheet., 2018. [online] Available at: <https://cdn.sparkfun.com/assets/e/7/3/b/1/BME280_Datasheet.pdf> [Accessed 30 July 2022].

CropX., n.d. *CropX Technologies*. [online] Available at: <<https://cropx.com/>> [Accessed 16 July 2022].

Cytron Technologies Malaysia. (n.d.). *12V 28BYJ-48 Stepper Motor + ULN2003 Driver Board*. [online] Available at: <https://my.cytron.io/p-12v-28byj-48-stepper-motor-plus-uln2003-driver-board> [Accessed 30 March 2023].

Cytron Technologies Malaysia. (n.d.). *2Amp 7V-30V L298N Motor Driver / Stepper Driver (2 Channels)*. [online] Available at: <https://my.cytron.io/p-2amp-7v-30v-l298n-motor-driver-stepper-driver-2-channels> [Accessed 30 March 2023].

Cytron Technologies Malaysia. (n.d.). *3V-5.5V SR04P Ultrasonic Ranging Module*. [online] Available at: <https://my.cytron.io/c-sensor/c-ultrasonic-sensor/p-3v-5.5v-ultrasonic-ranging-module> [Accessed 26 March 2023].

Cytron Technologies Malaysia. (n.d.). *5VDC HC-SR04 Ultrasonic Sensor*. [online] Available at: <<https://my.cytron.io/c-sensor/c-ultrasonic-sensor/p-5v-hc-sr04-ultrasonic-sensor>> [Accessed 26 March 2023].

Cytron Technologies Malaysia. (n.d.). *R385 DC12V Diaphragm Water Pump*. [online] Available at: <https://my.cytron.io/p-r385-dc12v-diaphragm-water-pump> [Accessed 30 March 2023].

ElecFreaks (2011). *Ultrasonic Ranging Module HC -SR04*. [online] Available at: <<https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>>.

- Esp32.net., 2016. *The Internet of Things with ESP32*. [online] Available at: <<http://esp32.net/>> [Accessed 30 July 2022].
- Geraldi, J. and Lechter, T., 2012. Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today. *International Journal of Managing Projects in Business*.
- Google, 2019. *Cloud Computing Services | Google Cloud*. [online] Google Cloud. Available at: <<https://cloud.google.com/>> [Accessed 31 July 2022].
- Joshua, H., 2020. (PDF) *Capacitive Soil Moisture Sensor Theory, Calibration, and Testing*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/342751186_Capacitive_Soil_Moisture_Sensor_Theory_Calibration_and_Testing> [Accessed 6 August 2022].
- Libelium., n.d. *Smart Agriculture IoT Solution» How IoT Works in the market?* [online] Available at: <<https://www.libelium.com/iot-solutions/smart-agriculture/>> [Accessed 16 July 2022].
- Lueth, K.L., 2014. *Why it is called Internet of Things: Definition, history, disambiguation*. [online] Iot-analytics.com. Available at: <<https://iot-analytics.com/internet-of-things-definition/>> [Accessed 21 August 2022].
- Maxim Integrated Products, 2019. *DS18B20*. [online] Available at: <<https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>> [Accessed 30 July 2022].
- Nermin Đuzić and Dalibor Đumić (2017). *Automatic Plant Watering System via Soil Moisture Sensing by means of Suitable Electronics and its...* [online] ResearchGate. Available at: <https://www.researchgate.net/publication/319130612_Automatic_Plant_Watering_System_via_Soil_Moisture_Sensing_by_means_of_Suitable_Electronics_and_its_Applications_for_Anthropological_and_Medical_Purposes> [Accessed 16 July 2022].

Orion, n.d. *IoT Automatic Plant Watering System*. [online] Instructables. Available at: <https://www.instructables.com/IoT-Automatic-Plant-Watering-System/> [Accessed 17 July 2022].

Parajuli, A., 2022. *IoT Smart Agriculture & Automatic Irrigation System with ESP8266*. [online] The IOT Projects. Available at: <https://theiotprojects.com/iot-smart-agriculture-automatic-irrigation-system-with-esp8266/> [Accessed 17 July 2022].

PrecisionHawk., n.d. *Agriculture: Drone Mapping and Analytics*. [online] Available at: <https://www.precisionhawk.com/agriculture> [Accessed 16 July 2022].

Random Nerd Tutorials., 2019. *DHT11 vs DHT22 vs LM35 vs DS18B20 vs BME280 vs BMP180 | Random Nerd Tutorials*. [online] Available at: <https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/> [Accessed 30 July 2022].

Saran, G., 2018. *Introduction To Google Cloud Platform - Whizlabs Blog*. [online] Available at: <https://www.whizlabs.com/blog/google-cloud-platform/#:~:text=Google%20cloud%20platform%20is%20a> [Accessed 31 July 2022].

Shawn, 2010. *Soil Moisture Sensor – How to choose and use with Arduino*. [online] Seeedstudio.com. Available at: <https://www.seeedstudio.com/blog/2020/01/10/what-is-soil-moisture-sensor-and-simple-arduino-tutorial-to-get-started/> [Accessed 6 August 2022].

STMicroelectronics. (n.d.). *L298 - STMicroelectronics*. [online] Available at: https://www.st.com/en/motor-drivers/l298.html#st_all-features_sec-nav-tab [Accessed 30 March 2023].

STMicroelectronics. (n.d.). *ULN2003 - STMicroelectronics*. [online] Available at: <https://www.st.com/en/interfaces-and-transceivers/uln2003.html> [Accessed 30 March 2023].

tutorialspoint, 2016. *About the Tutorial*. [online] Available at: https://www.tutorialspoint.com/internet_of_things/internet_of_things_tutorial.pdf [Accessed 21 August 2022].

Vishalsoniindia, n.d. *Avengers Plant Monitoring Device | Arduino IoT Projects | IoT Projects*. [online] Instructables. Available at: <https://www.instructables.com/Avengers-Plant-Monitoring-Device-Arduino-IoT-Proje/> [Accessed 17 July 2022].

wiki.seeedstudio.com., n.d. *Grove - Capacitive Moisture Sensor (Corrosion-Resistant) - Seeed Wiki*. [online] Available at: https://wiki.seeedstudio.com/Grove-Capacitive_Moisture_Sensor-Corrosion-Resistant/#typical-applications [Accessed 30 July 2022].

Yuan, M., 2017. *Getting to know NodeMCU and its DEVKIT board*. [online] IBM Developer. Available at: <https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/> [Accessed 23 July 2022].

APPENDICES

APPENDIX A: Coding for the IoT Smart Agriculture and Smart Irrigation System

```

#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL7hEMVYkv"
#define BLYNK_DEVICE_NAME "FYP"

#include <Wire.h>
#include <DHT.h>
#include <AccelStepper.h>
#include <PWMOutESP32.h>
#include <BlynkSimpleEsp32.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//Setting Up Connection Between Blynk and ESP32
char auth[] = "2kibXNYB0toNBcrq8NxbYRLRWK5-oAUu";
//Authentication code sent by Blynk
char ssid[] = "taytang";           //WiFi SSID
char pass[] = "1234567890";       //WiFi Password

//DHT22 Pin
#define DHTPIN 4    // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//Capacitive Sensor Pins

```

```
#define soilA_Pin 34
#define soilB_Pin 35

//Water Level Pins
#define trigPin 27
#define echoPin 26

//Water Pump Pin
#define waterPump 14

//Hardware Input Pins
#define limitSwitch 36 //Limit Switch Pin
#define watering 39 //PushButton Pin

//LED Indicators Pins
#define pumpOnLED 12
#define pumpOffLED 13
#define workingLED 32
#define restingLED 33

// ULN2003 Motor Driver Pins
#define IN1 19
#define IN2 18
#define IN3 5
#define IN4 17
#define MotorInterfaceType 8 //HALF4WIRE (half stepper with 4 motor pin
usage)

//OLED display size declaration
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
```

```
//define sound speed in cm/uS
#define SOUND_SPEED 0.034

//Range Declaration Of Soil Moisture
const int AirValue = 2440;
const int WaterValue = 1540;

//Range Declaration of water level
const int WLMax = 25;
const int WLMin = 0;

//Hardware Data
int soilMoistureValueA = 0;
int soilmoisturepercentA = 0;
int soilMoistureValueB = 0;
int soilmoisturepercentB = 0;
int limitSwitchState = 0 ;
int waterButtonState = 0;
float duration;
float waterL;
int waterlevelperct;

//Data From Blynk App
int distance = 0;
int totalSteps = 0;
int instantWatering = 0;
int pumpPower = 0;
int readStop = 0;
PWMOutESP32 pwm;
//Virtual LEDs Pins on BLYNK
WidgetLED pump_start(V7);
WidgetLED moving_start(V8);
```

```

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
BlynkTimer timer;
DHT dht(DHTPIN, DHTTYPE);
AccelStepper stepper (MotorInterfaceType, IN1, IN3, IN2, IN4);

//Emoji Declaration
//CRYING/*****/
const unsigned char PROGMEM frame05 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x03, 0x00, 0x01, 0x80, 0x70, 0x0C, 0x1F, 0xE0, 0x07, 0xF8, 0x30, 0x0C,
0x3D, 0x78, 0x1F, 0x7C, 0x38, 0x18, 0x30, 0x18, 0x18, 0x0C, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x07, 0xF0, 0x00, 0x0C, 0x30, 0x00, 0x3F, 0xFC, 0x00,
0x0C, 0x30, 0x00, 0x78, 0x1E, 0x00, 0x0C, 0x30, 0x00, 0xE0, 0x07, 0x00, 0x0C,
0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30, 0x03, 0x80, 0x01, 0xC0, 0x0C, 0x30,
0x03, 0x00, 0x00, 0xC0, 0x0C, 0x18, 0x07, 0x00, 0x00, 0xE0, 0x1C, 0x18, 0x06,
0x00, 0x00, 0x60, 0x18, 0x18, 0x06, 0x00, 0x00, 0x60, 0x18, 0x0C, 0x07, 0xFF,
0xFF, 0xE0, 0x30, 0x0C, 0x0F, 0xFF, 0xFF, 0xF0, 0x30, 0x0E, 0x02, 0x00, 0x00,
0x20, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};
const unsigned char PROGMEM frame25 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x0F, 0xC0, 0x03, 0xF0, 0xE0,
0x0E, 0x1F, 0xF0, 0x0F, 0xF8, 0x70, 0x0C, 0x38, 0x38, 0x1C, 0x1C, 0x30, 0x0C,
0x20, 0x18, 0x18, 0x04, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x18, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x30, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x28, 0x00, 0x00,
0x02, 0x0C, 0x30, 0x30, 0x00, 0x00, 0x07, 0x8C, 0x30, 0x00, 0x07, 0xE0, 0x07,
0xCC, 0x30, 0x00, 0x3F, 0xFC, 0x03, 0xEC, 0x30, 0x00, 0x78, 0x1E, 0x03, 0x3C,
0x30, 0x00, 0xE0, 0x07, 0x03, 0x3C, 0x30, 0x01, 0xC0, 0x03, 0x83, 0x1C, 0x30,
0x03, 0x80, 0x01, 0xC3, 0x1C, 0x18, 0x03, 0x00, 0x00, 0xC3, 0x0C, 0x18, 0x07,
0x00, 0x00, 0xC3, 0x0C, 0x18, 0x06, 0x00, 0x00, 0x63, 0x0C, 0x0C, 0x06, 0x00,
0x00, 0x63, 0x9C, 0x0C, 0x06, 0xAD, 0xDA, 0xF1, 0xF8, 0x0E, 0x0F, 0xFF,
0xFF, 0xE0, 0xF0, 0x07, 0x06, 0xAA, 0xAA, 0xE0, 0xE0, 0x03, 0x00, 0x00, 0x00,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM frame35 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x38, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x02, 0x80, 0x01, 0x40, 0xC0, 0x07, 0x1F, 0xE0, 0x07, 0xF0, 0xE0,
0x0E, 0x3E, 0xF8, 0x1F, 0x7C, 0x70, 0x0C, 0x30, 0x38, 0x18, 0x0C, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x10, 0x00,

```

```

0x00, 0x00, 0x0C, 0x30, 0x38, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x70, 0x00, 0x00,
0x00, 0x0C, 0x30, 0xD0, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x90, 0x02, 0x80, 0x03,
0x8C, 0x31, 0x90, 0x1F, 0xF8, 0x03, 0xCC, 0x30, 0xB0, 0x3D, 0x7E, 0x03, 0xFC,
0x30, 0xE0, 0xF0, 0x0F, 0x03, 0x7C, 0x30, 0x01, 0xC0, 0x03, 0x83, 0x1C, 0x30,
0x01, 0x80, 0x01, 0x83, 0x1C, 0x18, 0x03, 0x80, 0x00, 0xC3, 0x8C, 0x18, 0x03,
0x00, 0x00, 0xC1, 0x0C, 0x18, 0x06, 0x00, 0x00, 0x63, 0x8C, 0x0C, 0x06, 0x00,
0x00, 0x61, 0x8C, 0x0C, 0x06, 0x00, 0x00, 0x61, 0x8C, 0x0E, 0x0F, 0xFF, 0xFF,
0xF1, 0xFC, 0x07, 0x07, 0xFF, 0xFF, 0xE0, 0xF8, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x1C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x07, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame45 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x0F, 0xC0, 0x03, 0xF0, 0x70, 0x0C, 0x3F, 0xF0, 0x0F, 0xFC, 0x30,
0x0C, 0x38, 0x38, 0x1C, 0x1C, 0x30, 0x18, 0x20, 0x08, 0x18, 0x04, 0x18, 0x18,
0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00,
0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x70, 0x00,
0x00, 0x00, 0x0C, 0x30, 0xF0, 0x02, 0xC0, 0x00, 0x0C, 0x33, 0xF0, 0x1F, 0xF8,
0x00, 0x0C, 0x33, 0xE0, 0x7E, 0xBE, 0x00, 0x0C, 0x37, 0x60, 0xF0, 0x0F, 0x03,
0xCC, 0x3E, 0x61, 0xC0, 0x03, 0x83, 0xEC, 0x3E, 0x61, 0x80, 0x01, 0x81, 0xFC,
0x3C, 0x63, 0x00, 0x00, 0xC1, 0xBC, 0x1C, 0x63, 0x00, 0x00, 0xC1, 0x9C, 0x1F,
0xE6, 0x00, 0x00, 0x61, 0x8C, 0x1F, 0xC6, 0x00, 0x00, 0x61, 0x8E, 0x0D, 0x06,
0x5F, 0xFA, 0x61, 0x86, 0x0C, 0x0F, 0xFF, 0xFF, 0xF1, 0x86, 0x0E, 0x07, 0xD0,
0x05, 0xE1, 0x86, 0x07, 0x00, 0x00, 0x00, 0x01, 0xCE, 0x03, 0x00, 0x00, 0x00,
0x00, 0xFC, 0x01, 0x80, 0x00, 0x00, 0x01, 0xF8, 0x01, 0xC0, 0x00, 0x00, 0x03,

```

```

0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

const unsigned char PROGMEM frame55 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00,
0x07, 0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C,
0x00, 0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00,
0x01, 0xC0, 0x03, 0x02, 0x80, 0x01, 0x40, 0xC0, 0x07, 0x1F, 0xE0, 0x07, 0xF8,
0xE0, 0x0E, 0x3E, 0xF8, 0x1F, 0x7C, 0x70, 0x0C, 0x30, 0x38, 0x18, 0x0C, 0x30,
0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18,
0x00, 0x00, 0x00, 0x08, 0x18, 0x18, 0x00, 0x00, 0x00, 0x08, 0x18, 0x30, 0x00,
0x00, 0x00, 0x02, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x07, 0xF0, 0x00, 0x0C, 0x30, 0x00, 0x3F, 0xFC,
0x00, 0x0C, 0x30, 0x00, 0x78, 0x1E, 0x00, 0x0C, 0x31, 0xC0, 0xE0, 0x07,
0x00, 0x0C, 0x37, 0xC1, 0xC0, 0x03, 0x80, 0x0C, 0x3F, 0xC3, 0x80, 0x01, 0xC0,
0x0C, 0x3D, 0x83, 0x00, 0x00, 0xC0, 0x0C, 0x19, 0x87, 0x00, 0x00, 0xE0, 0x1C,
0x39, 0x86, 0x00, 0x00, 0x60, 0x18, 0x31, 0x86, 0x00, 0x00, 0x60, 0x18, 0x31,
0x8F, 0x55, 0x55, 0x60, 0x30, 0x33, 0x87, 0xFF, 0xFF, 0xF0, 0x30, 0x3F, 0x06,
0xDB, 0x6D, 0xA0, 0x60, 0x1F, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00,
0x00, 0x00, 0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00,
0x00, 0x03, 0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00,
0x1E, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0,
0x00, 0x00, 0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00,
0x00, 0x00, 0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00};

const unsigned char PROGMEM frame65 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,

```



```

0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x0F, 0xC0, 0x03, 0xF0, 0x70, 0x0C, 0x3F, 0xF0, 0x0F, 0xFC, 0x30, 0x0C,
0x38, 0x38, 0x1C, 0x1C, 0x30, 0x18, 0x20, 0x08, 0x18, 0x04, 0x18, 0x18, 0x00,
0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x0E, 0x18, 0x30, 0x00, 0x00,
0x00, 0x0F, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x0D, 0x8C, 0x30, 0x00, 0x00, 0x00,
0x04, 0xCC, 0x30, 0x00, 0x00, 0x00, 0x04, 0xCC, 0x30, 0x00, 0x07, 0xE0, 0x04,
0x4C, 0x30, 0x00, 0x3F, 0xFC, 0x04, 0x4C, 0x30, 0x00, 0x78, 0x1E, 0x06, 0xCC,
0x30, 0x00, 0xE0, 0x07, 0x03, 0x8C, 0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30,
0x03, 0x80, 0x01, 0xC0, 0x0C, 0x18, 0x03, 0x00, 0x00, 0xC0, 0x1C, 0x18, 0x07,
0x00, 0x00, 0xE0, 0x18, 0x18, 0x06, 0x00, 0x00, 0x60, 0x18, 0x0C, 0x06, 0x00,
0x00, 0x60, 0x30, 0x0C, 0x07, 0xFF, 0xFF, 0xE0, 0x30, 0x0E, 0x0F, 0xFF, 0xFF,
0xF0, 0x60, 0x07, 0x05, 0x00, 0x00, 0xA0, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame75 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x0F, 0xC0, 0x03, 0xF0, 0x70, 0x0C, 0x3F, 0xF0, 0x0F, 0xFC, 0x30, 0x0C,
0x38, 0x38, 0x1C, 0x1C, 0x30, 0x18, 0x20, 0x08, 0x18, 0x04, 0x18, 0x18, 0x00,
0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x07, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x07, 0x8C, 0x30, 0x00, 0x00, 0x00, 0x07, 0xCC, 0x30, 0x00, 0x07, 0xE0, 0x06,
0xEC, 0x30, 0x00, 0x3F, 0xFC, 0x06, 0x3C, 0x30, 0x00, 0x78, 0x1E, 0x07, 0x3C,

```

```

0x30, 0x00, 0xE0, 0x07, 0x02, 0x1C, 0x30, 0x01, 0xC0, 0x03, 0x87, 0x1C, 0x30,
0x03, 0x80, 0x01, 0xC3, 0x1C, 0x18, 0x03, 0x00, 0x00, 0xC3, 0x1C, 0x18, 0x07,
0x00, 0x00, 0xE3, 0x18, 0x18, 0x06, 0x00, 0x00, 0x63, 0xF8, 0x0C, 0x06, 0x00,
0x00, 0x61, 0xF0, 0x0C, 0x07, 0xFF, 0xFF, 0xE0, 0x70, 0x0E, 0x0F, 0xFF, 0xFF,
0xF0, 0x60, 0x07, 0x05, 0x00, 0x00, 0xA0, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame85 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x07, 0xC0, 0x03, 0xE0, 0xE0,
0x0E, 0x1F, 0xF0, 0x0F, 0xF8, 0x70, 0x0C, 0x3C, 0x78, 0x1E, 0x1C, 0x30, 0x0C,
0x30, 0x18, 0x18, 0x0C, 0x38, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x38, 0x10, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x38, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x78, 0x00, 0x00,
0x00, 0x0C, 0x30, 0xD0, 0x07, 0xF0, 0x00, 0x0C, 0x30, 0x90, 0x3F, 0xFC, 0x00,
0x0C, 0x30, 0x90, 0x78, 0x1E, 0x03, 0x8C, 0x30, 0xF0, 0xE0, 0x07, 0x03, 0xEC,
0x30, 0xE1, 0xC0, 0x03, 0x83, 0xFC, 0x30, 0x03, 0x80, 0x01, 0xC3, 0xBC, 0x30,
0x03, 0x00, 0x00, 0xC1, 0x9C, 0x18, 0x07, 0x00, 0x00, 0xE1, 0x8C, 0x18, 0x06,
0x00, 0x00, 0x61, 0x8C, 0x18, 0x06, 0x00, 0x00, 0x61, 0x8C, 0x0C, 0x07, 0x7F,
0xFE, 0xE1, 0x86, 0x0C, 0x0F, 0xFF, 0xFF, 0xF1, 0x86, 0x0E, 0x02, 0xA0, 0x02,
0xA1, 0x8E, 0x07, 0x00, 0x00, 0x00, 0x01, 0xFC, 0x03, 0x00, 0x00, 0x00, 0x00,
0xF8, 0x01, 0x80, 0x00, 0x00, 0x01, 0xC0, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,

```

```
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM frame95 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x0F, 0xE0, 0x07, 0xF0, 0x70, 0x0C, 0x3F, 0xF0, 0x0F, 0xFC, 0x30, 0x0C,
0x38, 0x38, 0x1C, 0x1C, 0x38, 0x18, 0x20, 0x08, 0x10, 0x04, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x30, 0x00, 0x00, 0x00, 0x0C, 0x30, 0xF8, 0x00, 0x00,
0x00, 0x0C, 0x31, 0xF0, 0x07, 0xF0, 0x00, 0x0C, 0x33, 0xF0, 0x3F, 0xFC, 0x00,
0x0C, 0x33, 0x30, 0x78, 0x1E, 0x00, 0x0C, 0x36, 0x30, 0xE0, 0x07, 0x03, 0xCC,
0x36, 0x71, 0xC0, 0x03, 0x83, 0xEC, 0x36, 0x33, 0x80, 0x01, 0xC1, 0xFC, 0x3E,
0x73, 0x00, 0x00, 0xC1, 0xBC, 0x1F, 0xE7, 0x00, 0x00, 0xE1, 0x9C, 0x1F, 0xE6,
0x00, 0x00, 0x61, 0x8C, 0x19, 0x06, 0x00, 0x00, 0x61, 0x8E, 0x0C, 0x07, 0xFF,
0xFF, 0xE1, 0x86, 0x0C, 0x0F, 0xFF, 0xFF, 0xF1, 0x86, 0x0E, 0x05, 0x00, 0x00,
0xA1, 0x86, 0x07, 0x00, 0x00, 0x00, 0x01, 0xCE, 0x03, 0x00, 0x00, 0x00, 0x00,
0xFC, 0x01, 0x80, 0x00, 0x00, 0x01, 0xF8, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
//Happy/*****
```

```
const unsigned char PROGMEM frame012 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
```

```

0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x07, 0x80, 0x03, 0xC0, 0x30, 0x18, 0x1F, 0xE0, 0x0F, 0xF0, 0x18, 0x18, 0x3C,
0x78, 0x1E, 0x3C, 0x18, 0x18, 0x30, 0x18, 0x18, 0x0C, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x1E, 0x00, 0x00, 0xF0,
0x0C, 0x30, 0x1F, 0xF8, 0x2F, 0xF8, 0x0C, 0x30, 0x1D, 0xFF, 0xFF, 0xB8, 0x0C,
0x30, 0x18, 0x17, 0xF0, 0x30, 0x0C, 0x30, 0x0C, 0x00, 0x00, 0x30, 0x0C, 0x30,
0x0C, 0x00, 0x00, 0x30, 0x0C, 0x18, 0x0E, 0x00, 0x00, 0x60, 0x1C, 0x18, 0x06,
0x00, 0x00, 0x60, 0x18, 0x18, 0x07, 0x00, 0x00, 0xE0, 0x18, 0x0C, 0x03, 0x00,
0x01, 0xC0, 0x30, 0x0C, 0x01, 0x80, 0x03, 0x80, 0x30, 0x0E, 0x01, 0xC0, 0x07,
0x00, 0x60, 0x07, 0x00, 0xF0, 0x0E, 0x00, 0xE0, 0x03, 0x00, 0x7E, 0x3C, 0x00,
0xC0, 0x01, 0x80, 0x1F, 0xF8, 0x01, 0x80, 0x01, 0xC0, 0x03, 0xC0, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame22 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x0F, 0xC0, 0x07, 0xE0, 0xE0,
0x0E, 0x1F, 0xF0, 0x1F, 0xF0, 0x70, 0x0C, 0x38, 0x30, 0x18, 0x38, 0x30, 0x0C,
0x30, 0x38, 0x38, 0x18, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x0F, 0xFF, 0xFF,
0xE0, 0x0C, 0x30, 0x0F, 0xFF, 0xFF, 0xE0, 0x0C, 0x30, 0x0C, 0x00, 0x00, 0x60,
0x0C, 0x30, 0x0C, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x0C, 0x00, 0x00, 0x60, 0x0C,
0x30, 0x0C, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x0E, 0x00, 0x00, 0x60, 0x0C, 0x30,

```

```

0x06, 0x00, 0x00, 0xC0, 0x0C, 0x18, 0x06, 0x00, 0x00, 0xC0, 0x1C, 0x18, 0x07,
0x00, 0x01, 0xC0, 0x18, 0x18, 0x03, 0x00, 0x01, 0x80, 0x18, 0x0C, 0x03, 0x80,
0x03, 0x80, 0x30, 0x0C, 0x01, 0x80, 0x03, 0x00, 0x30, 0x0E, 0x00, 0xC0, 0x06,
0x00, 0x60, 0x07, 0x00, 0xE0, 0x0E, 0x00, 0xE0, 0x03, 0x00, 0x78, 0x3C, 0x00,
0xC0, 0x01, 0x80, 0x1F, 0xF0, 0x01, 0x80, 0x01, 0xC0, 0x07, 0xC0, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame42 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x1F, 0xC0, 0x0F, 0xE0, 0x30, 0x0C,
0x3F, 0xE0, 0x1F, 0xF8, 0x30, 0x18, 0x70, 0x70, 0x38, 0x38, 0x18, 0x18, 0x40,
0x10, 0x20, 0x08, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x38, 0x00, 0x00,
0x70, 0x0C, 0x30, 0x3F, 0x80, 0x07, 0xF0, 0x0C, 0x30, 0x3F, 0xFF, 0xFF, 0xF0,
0x0C, 0x30, 0x30, 0x7F, 0xFC, 0x30, 0x0C, 0x30, 0x38, 0x00, 0x00, 0x70, 0x0C,
0x30, 0x18, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x1C, 0x00, 0x00, 0xE0, 0x0C, 0x30,
0x0C, 0x00, 0x00, 0xC0, 0x0C, 0x18, 0x06, 0x00, 0x01, 0xC0, 0x1C, 0x18, 0x07,
0x00, 0x03, 0x80, 0x18, 0x18, 0x03, 0x80, 0x07, 0x00, 0x18, 0x0C, 0x01, 0xE0,
0x1E, 0x00, 0x30, 0x0C, 0x00, 0xF8, 0x7C, 0x00, 0x30, 0x0E, 0x00, 0x3F, 0xF0,
0x00, 0x60, 0x07, 0x00, 0x0F, 0xC0, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,

```

```

0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

const unsigned char PROGMEM frame52 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x1F, 0xC0, 0x0F, 0xE0, 0x30, 0x0C,
0x3F, 0xE0, 0x1F, 0xF8, 0x30, 0x18, 0x70, 0x70, 0x38, 0x38, 0x18, 0x18, 0x40,
0x10, 0x20, 0x08, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x38, 0x00, 0x00,
0x70, 0x0C, 0x30, 0x3F, 0x80, 0x03, 0xF0, 0x0C, 0x30, 0x3F, 0xFF, 0xFF, 0xF0,
0x0C, 0x30, 0x30, 0xFF, 0xFC, 0x30, 0x0C, 0x30, 0x18, 0x00, 0x00, 0x70, 0x0C,
0x30, 0x18, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x1C, 0x00, 0x00, 0xE0, 0x0C, 0x30,
0x0E, 0x00, 0x00, 0xC0, 0x0C, 0x18, 0x06, 0x00, 0x01, 0xC0, 0x1C, 0x18, 0x07,
0x00, 0x03, 0x80, 0x18, 0x18, 0x03, 0x80, 0x07, 0x00, 0x18, 0x0C, 0x01, 0xE0,
0x1E, 0x00, 0x30, 0x0C, 0x00, 0x7C, 0xFC, 0x00, 0x30, 0x0E, 0x00, 0x3F, 0xF0,
0x00, 0x60, 0x07, 0x00, 0x07, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

const unsigned char PROGMEM frame72 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC3, 0xC0, 0x03, 0xC3, 0x80, 0x03, 0x8F, 0xF0, 0x0F, 0xF1,
0xC0, 0x03, 0x1C, 0x78, 0x0E, 0x38, 0xC0, 0x07, 0x18, 0x18, 0x18, 0x18, 0xE0,

```

```

0x0E, 0x10, 0x18, 0x18, 0x08, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x07, 0xFF,
0xFF, 0xF0, 0x0C, 0x30, 0x0F, 0xFF, 0xFF, 0xE0, 0x0C, 0x30, 0x0C, 0x00, 0x00,
0x30, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x60,
0x0C, 0x30, 0x06, 0x00, 0x00, 0x60, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x60, 0x0C,
0x30, 0x06, 0x00, 0x00, 0xE0, 0x0C, 0x30, 0x03, 0x00, 0x00, 0xC0, 0x0C, 0x30,
0x03, 0x00, 0x00, 0xC0, 0x0C, 0x18, 0x03, 0x80, 0x01, 0xC0, 0x1C, 0x18, 0x01,
0x80, 0x01, 0x80, 0x18, 0x18, 0x01, 0xC0, 0x03, 0x80, 0x18, 0x0C, 0x00, 0xE0,
0x07, 0x00, 0x30, 0x0C, 0x00, 0x70, 0x0E, 0x00, 0x30, 0x0E, 0x00, 0x3C, 0x3C,
0x00, 0x60, 0x07, 0x00, 0x1F, 0xF8, 0x00, 0xE0, 0x03, 0x00, 0x07, 0xE0, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame82 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x03, 0xE0, 0x03, 0xF0, 0xC0, 0x06, 0x0F, 0xF0, 0x07, 0xF8, 0xE0,
0x0E, 0x1C, 0x38, 0x0E, 0x1C, 0x70, 0x0C, 0x18, 0x1C, 0x1C, 0x0C, 0x30,
0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18,
0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00,
0x00, 0x00, 0x10, 0x0C, 0x30, 0x0F, 0xFE, 0xBF, 0xF8, 0x0C, 0x30, 0x0F, 0xFF,
0xFF, 0xF8, 0x0C, 0x30, 0x0C, 0x02, 0xA0, 0x18, 0x0C, 0x30, 0x06, 0x00, 0x00,
0x18, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x30, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x30,
0x0C, 0x30, 0x07, 0x00, 0x00, 0x30, 0x0C, 0x30, 0x03, 0x00, 0x00, 0x60, 0x0C,
0x30, 0x03, 0x80, 0x00, 0x60, 0x0C, 0x18, 0x01, 0x80, 0x00, 0xE0, 0x1C, 0x18,
0x01, 0xC0, 0x01, 0xC0, 0x18, 0x18, 0x00, 0xE0, 0x01, 0x80, 0x18, 0x0C, 0x00,

```

```

0x70, 0x07, 0x00, 0x30, 0x0C, 0x00, 0x38, 0x0F, 0x00, 0x30, 0x0E, 0x00, 0x1F,
0x7C, 0x00, 0x60, 0x07, 0x00, 0x07, 0xF8, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x80,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM frame92 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x07, 0xF0, 0x03, 0xF8, 0x30, 0x0C,
0x1F, 0xF8, 0x07, 0xFC, 0x30, 0x18, 0x1C, 0x1C, 0x0E, 0x0E, 0x18, 0x18, 0x10,
0x0C, 0x08, 0x02, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x0E, 0x00, 0x00,
0x1C, 0x0C, 0x30, 0x0F, 0xE0, 0x00, 0xFC, 0x0C, 0x30, 0x0F, 0xFF, 0xFF,
0xFC, 0x0C, 0x30, 0x0C, 0x3F, 0xFF, 0x0C, 0x0C, 0x30, 0x0E, 0x00, 0x00, 0x1C,
0x0C, 0x30, 0x06, 0x00, 0x00, 0x18, 0x0C, 0x30, 0x07, 0x00, 0x00, 0x38, 0x0C,
0x30, 0x03, 0x00, 0x00, 0x70, 0x0C, 0x18, 0x01, 0x80, 0x00, 0x60, 0x1C, 0x18,
0x01, 0xC0, 0x00, 0xE0, 0x18, 0x18, 0x00, 0xE0, 0x01, 0xC0, 0x18, 0x0C, 0x00,
0x78, 0x07, 0x80, 0x30, 0x0C, 0x00, 0x1E, 0x3E, 0x00, 0x30, 0x0E, 0x00, 0x0F,
0xFC, 0x00, 0x60, 0x07, 0x00, 0x01, 0xC0, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM frame102 [] = {

```



```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x07, 0xF0, 0x03, 0xF8, 0x30, 0x0C,
0x1F, 0xF8, 0x07, 0xFC, 0x30, 0x18, 0x1C, 0x1C, 0x0E, 0x0E, 0x18, 0x18, 0x10,
0x04, 0x08, 0x02, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x0E, 0x00, 0x00,
0x1C, 0x0C, 0x30, 0x0F, 0xE0, 0x01, 0xFC, 0x0C, 0x30, 0x0F, 0xFF, 0xFF,
0xFC, 0x0C, 0x30, 0x0C, 0x1F, 0xFF, 0x0C, 0x0C, 0x30, 0x0E, 0x00, 0x00, 0x1C,
0x0C, 0x30, 0x06, 0x00, 0x00, 0x18, 0x0C, 0x30, 0x07, 0x00, 0x00, 0x38, 0x0C,
0x30, 0x03, 0x00, 0x00, 0x30, 0x0C, 0x18, 0x03, 0x80, 0x00, 0x70, 0x1C, 0x18,
0x01, 0xC0, 0x00, 0xE0, 0x18, 0x18, 0x00, 0xE0, 0x01, 0xC0, 0x18, 0x0C, 0x00,
0x70, 0x07, 0x80, 0x30, 0x0C, 0x00, 0x3E, 0x1F, 0x00, 0x30, 0x0E, 0x00, 0x0F,
0xFC, 0x00, 0x60, 0x07, 0x00, 0x03, 0xE0, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM frame122 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x07, 0xE0, 0x03, 0xF0, 0xE0,
0x0E, 0x0F, 0xF8, 0x0F, 0xF8, 0x70, 0x0C, 0x1C, 0x18, 0x0C, 0x1C, 0x30, 0x0C,
0x18, 0x1C, 0x1C, 0x0C, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00,

```

```

0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x07, 0xFF, 0xFF,
0xF0, 0x0C, 0x30, 0x07, 0xFF, 0xFF, 0xF0, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x30,
0x0C, 0x30, 0x06, 0x00, 0x00, 0x30, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x30, 0x0C,
0x30, 0x06, 0x00, 0x00, 0x30, 0x0C, 0x30, 0x06, 0x00, 0x00, 0x30, 0x0C, 0x30,
0x03, 0x00, 0x00, 0x60, 0x0C, 0x18, 0x03, 0x00, 0x00, 0x60, 0x1C, 0x18, 0x03,
0x80, 0x00, 0xE0, 0x18, 0x18, 0x01, 0x80, 0x00, 0xC0, 0x18, 0x0C, 0x01, 0xC0,
0x01, 0xC0, 0x30, 0x0C, 0x00, 0xC0, 0x01, 0x80, 0x30, 0x0E, 0x00, 0x60, 0x03,
0x00, 0x60, 0x07, 0x00, 0x70, 0x07, 0x00, 0xE0, 0x03, 0x00, 0x3C, 0x1E, 0x00,
0xC0, 0x01, 0x80, 0x0F, 0xF8, 0x01, 0x80, 0x01, 0xC0, 0x03, 0xE0, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame132 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x06, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x07, 0xC0, 0x03, 0xE0, 0x30, 0x18, 0x0F, 0xF0, 0x07, 0xF8, 0x18, 0x18, 0x1C,
0x78, 0x1E, 0x3C, 0x18, 0x18, 0x30, 0x18, 0x18, 0x0C, 0x18, 0x30, 0x00, 0x00,
0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x0E, 0x00, 0x00, 0x78,
0x0C, 0x30, 0x1F, 0xF4, 0x0F, 0xF8, 0x0C, 0x30, 0x1D, 0xFF, 0xFF, 0xB8, 0x0C,
0x30, 0x0C, 0x07, 0xF0, 0x18, 0x0C, 0x30, 0x0C, 0x00, 0x00, 0x30, 0x0C, 0x30,
0x0C, 0x00, 0x00, 0x30, 0x0C, 0x18, 0x06, 0x00, 0x00, 0x70, 0x1C, 0x18, 0x06,
0x00, 0x00, 0x60, 0x18, 0x18, 0x03, 0x00, 0x00, 0xE0, 0x18, 0x0C, 0x03, 0x80,
0x01, 0xC0, 0x30, 0x0C, 0x01, 0xC0, 0x01, 0x80, 0x30, 0x0E, 0x00, 0xE0, 0x03,
0x80, 0x60, 0x07, 0x00, 0x70, 0x0F, 0x00, 0xE0, 0x03, 0x00, 0x3E, 0x3C, 0x00,
0xC0, 0x01, 0x80, 0x1F, 0xF8, 0x01, 0x80, 0x01, 0xC0, 0x01, 0xC0, 0x03, 0x80,

```

```
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};
```

```
//Neutral/*****
```

```
const unsigned char PROGMEM frame03 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x01,
0x80, 0x03, 0x80, 0x18, 0x18, 0x03, 0xE0, 0x07, 0xC0, 0x18, 0x30, 0x03, 0xE0,
0x07, 0xC0, 0x0C, 0x30, 0x07, 0xE0, 0x07, 0xE0, 0x0C, 0x30, 0x03, 0xE0, 0x07,
0xC0, 0x0C, 0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};
```

```
const unsigned char PROGMEM frame013 [] = {
```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00,
0x00, 0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00,
0x01, 0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00,
0xE0, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30,
0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18,
0x01, 0x80, 0x03, 0x80, 0x18, 0x18, 0x03, 0xE0, 0x07, 0xC0, 0x18, 0x30, 0x03,
0xE0, 0x07, 0xC0, 0x0C, 0x30, 0x07, 0xE0, 0x07, 0xE0, 0x0C, 0x30, 0x03, 0xE0,
0x07, 0xC0, 0x0C, 0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18,
0x07, 0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00,
0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00,
0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};

```

```

const unsigned char PROGMEM frame23 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x80, 0x01, 0x00, 0x18, 0x18, 0x03, 0xE0, 0x07, 0xC0, 0x18, 0x30, 0x07, 0xE0,

```

```

0x07, 0xE0, 0x0C, 0x30, 0x07, 0xE0, 0x07, 0xE0, 0x0C, 0x30, 0x03, 0xE0, 0x07,
0xC0, 0x0C, 0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame33 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x03, 0xC0, 0x07, 0xC0, 0x18, 0x30, 0x07, 0xF0,
0x07, 0xE0, 0x0C, 0x30, 0x07, 0xE0, 0x0F, 0xE0, 0x0C, 0x30, 0x03, 0xE0, 0x07,
0xC0, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,

```

```
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};
```

```
const unsigned char PROGMEM frame43 [] = {
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30,
0x0C, 0x00, 0x00, 0x00, 0x00, 0x38, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18,
0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x80, 0x01, 0x00, 0x18, 0x30, 0x07,
0xF0, 0x0F, 0xE0, 0x0C, 0x30, 0x07, 0xF0, 0x0F, 0xE0, 0x0C, 0x30, 0x03, 0xC0,
0x03, 0x80, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x38, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18,
0x07, 0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00,
0x00, 0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00,
0x00, 0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00,
0x00, 0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03,
0x80, 0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00,
0x03, 0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x1F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
```

```
const unsigned char PROGMEM frame53 [] = {
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
```

```

0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x38, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x30, 0x07, 0xE0,
0x07, 0xE0, 0x0C, 0x30, 0x07, 0xF0, 0x0F, 0xE0, 0x0C, 0x30, 0x00, 0x00, 0x00,
0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x38, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame63 [] = {

```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x38, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x80, 0x01, 0x00, 0x18, 0x30, 0x07, 0xF0,
0x0F, 0xE0, 0x0C, 0x30, 0x07, 0xF0, 0x0F, 0xE0, 0x0C, 0x30, 0x03, 0xC0, 0x03,
0x80, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,

```

```

0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x38, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

const unsigned char PROGMEM frame73 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,
0x00, 0x00, 0x00, 0x18, 0x18, 0x03, 0xC0, 0x07, 0xC0, 0x18, 0x30, 0x07, 0xF0,
0x07, 0xE0, 0x0C, 0x30, 0x07, 0xE0, 0x0F, 0xE0, 0x0C, 0x30, 0x03, 0xE0, 0x07,
0xC0, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,

```



```

0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

const unsigned char PROGMEM frame83 [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x07,
0xE0, 0x07, 0xE0, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x3C, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00, 0xE0, 0x00, 0x00,
0x07, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x00, 0x00, 0x01,
0xC0, 0x03, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0,
0x0E, 0x00, 0x00, 0x00, 0x00, 0x70, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0C,
0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x01,
0x80, 0x03, 0x80, 0x18, 0x18, 0x03, 0xE0, 0x07, 0xC0, 0x18, 0x30, 0x03, 0xE0,
0x07, 0xC0, 0x0C, 0x30, 0x07, 0xE0, 0x07, 0xE0, 0x0C, 0x30, 0x03, 0xE0, 0x07,
0xC0, 0x0C, 0x30, 0x01, 0xC0, 0x03, 0x80, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00,
0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C,
0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x30,
0x00, 0x00, 0x00, 0x00, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x1C, 0x18, 0x07,
0xFF, 0xFF, 0xE0, 0x18, 0x18, 0x03, 0xFF, 0xFF, 0xC0, 0x18, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x30, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x30, 0x0E, 0x00, 0x00, 0x00,
0x00, 0x60, 0x07, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x03, 0x00, 0x00, 0x00, 0x00,
0xC0, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x01, 0xC0, 0x00, 0x00, 0x03, 0x80,
0x00, 0xE0, 0x00, 0x00, 0x07, 0x00, 0x00, 0x78, 0x00, 0x00, 0x1E, 0x00, 0x00,
0x3C, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x0F, 0x00, 0x00, 0xF0, 0x00, 0x00, 0x03,
0xE0, 0x0F, 0xC0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

```

```

/*****

```

```

void RunBlynk(void* parameter){
  for(;;){
    Serial.print("BlynkRun is running on: ");
    Serial.println(xPortGetCoreID());
    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}

```

```
Blynk.run();
timer.run();
}
}
/*****/
void HumiTemp(void* parameter){
  for(;;){
    float humidityValue = dht.readHumidity();
    float temperatureValue = dht.readTemperature();
    Blynk.virtualWrite(V0, humidityValue);
    Blynk.virtualWrite(V1, temperatureValue);
    vTaskDelay(3000 / portTICK_PERIOD_MS);
  }
}
/*****/
void SoilA(void* parameter){
  for(;;){
    float soilMoistureValueA = analogRead(soilA_Pin);
    soilmoisturepercentA = map(soilMoistureValueA, AirValue, WaterValue, 0,
100);
    Blynk.virtualWrite(V2, soilmoisturepercentA);
    vTaskDelay(5000 / portTICK_PERIOD_MS);
  }
}
/*****/
void SoilB(void* parameter){
  for(;;){
    float soilMoistureValueB = analogRead(soilB_Pin);
    soilmoisturepercentB = map(soilMoistureValueB, AirValue, WaterValue, 0,
100);
    Blynk.virtualWrite(V3, soilmoisturepercentB);
    vTaskDelay(5000 / portTICK_PERIOD_MS);
  }
}
```

```

}
/*****/
void WaterLvl(void* parameter){
  for(;;){
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculate the distance
    waterL = duration * SOUND_SPEED/2;
    waterlevelperct = map(waterL, WLMax, WLMin, 0, 100);
    Blynk.virtualWrite(V12, waterlevelperct);
    vTaskDelay(3000 / portTICK_PERIOD_MS);
  }
}
/*****/
BLYNK_WRITE(V4){
  instantWatering = param.asInt(); }
BLYNK_WRITE(V9){
  distance = param.asInt(); }
BLYNK_WRITE(V10){
  pumpPower = param.asInt(); }
/*****/
void SysCtrl(void* parameter){
  for(;;){
    distanceCal(distance);
    goHome();
    waterButtonState = digitalRead(watering);

```

```

if((instantWatering == 1) || ( waterButtonState == 1)){
    goWork(totalSteps);
    goHome();
}
}
}

/*****/

// Calculation of steps require to pull the load by 1 cm (Steps/cm)
// Total steps = distance set * (Steps of one full revolution / (belt pitch [GT2 = 2]
x Teeth numbers of pulley [20]))
// The outcome of this equation will in mm. Thus x10 to get cm
void distanceCal(int distance){
    totalSteps = distance * (4096 / (2 * 20)) * 10;
}

/*****/

void goWork(int totalSteps){
    stepper.setCurrentPosition(0);
    stepper.enableOutputs();
    stepper.moveTo(totalSteps);
    workLEDs();
    pwm.analogWrite(waterPump, pumpPower);
    while(stepper.distanceToGo() != 0){
        stepper.setSpeed(450);
        stepper.runSpeed();
    }
    stepper.setSpeed(0);
}

/*****/

void goHome(){
    stepper.setCurrentPosition(0);
    limitSwitchState = digitalRead(limitSwitch);
}

```

```
pwm.analogWrite(waterPump, 0);
if(limitSwitchState==0){
    goBackLeds();
}
else{
    restLEDs();
}
stepper.enableOutputs();
while(limitSwitchState !=1){
    stepper.setSpeed(-450);
    stepper.runSpeed();
    limitSwitchState = digitalRead(limitSwitch);
}
stepper.setSpeed(0);
stepper.disableOutputs();
limitSwitchState = digitalRead(limitSwitch);
}
/*****/
void workLEDs(){
    moving_start.on();
    pump_start.on();
    delay(10);
    digitalWrite(pumpOnLED,HIGH);
    digitalWrite(workingLED,HIGH);
    digitalWrite(pumpOffLED,LOW);
    digitalWrite(restingLED,LOW);
}
/*****/
void goBackLeds(){
    moving_start.on();
    pump_start.off();
    delay(10);
    digitalWrite(pumpOnLED,LOW);
```

```
digitalWrite(workingLED,HIGH);
digitalWrite(pumpOffLED,HIGH);
digitalWrite(restingLED,LOW);
}
/*****/
void restLEDs(){
  moving_start.off();
  pump_start.off();
  digitalWrite(pumpOnLED,LOW);
  digitalWrite(workingLED,LOW);
  digitalWrite(pumpOffLED,HIGH);
  digitalWrite(restingLED,HIGH);
}
/*****/
void setup(){
  // Debug console
  Serial.begin(115200);
  delay(10);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  int wifi_ctr = 0;
  while (WiFi.status() != WL_CONNECTED){
    delay(1000);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Blynk.begin(auth, ssid, pass);
  dht.begin();

  //Pump
  pinMode(waterPump, OUTPUT);
  //Water Level
```

```

pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
//Limit Switch
pinMode(limitSwitch, INPUT);
//Buttons
pinMode(watering, INPUT);
//LEDs
pinMode(pumpOnLED, OUTPUT);
pinMode(pumpOffLED, OUTPUT);
pinMode(workingLED, OUTPUT);
pinMode(restingLED, OUTPUT);

//Define the maximum steps per second
stepper.setMaxSpeed(1000);
stepper.setAcceleration(200);

//initialize with the I2C addr 0x3C (128x64)
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.clearDisplay();
display.setTextColor(WHITE);

xTaskCreatePinnedToCore(RunBlynk, "BlynkRun",2048,NULL,1,NULL,0);
xTaskCreatePinnedToCore(HumiTemp, "DHT",4096,NULL,1,NULL,0);
xTaskCreatePinnedToCore(SoilA, "SoilMoistureA",4096,NULL,1,NULL,0);
xTaskCreatePinnedToCore(SoilB, "SoilMoistureB",4096,NULL,1,NULL,0);
xTaskCreatePinnedToCore(WaterLvl, "WaterLevel",4096,NULL,1,NULL,0);
xTaskCreatePinnedToCore(SysCtrl, "SystemControl",10000,NULL,0,NULL,0);
xTaskCreatePinnedToCore(OLED, "OLED_Display",10000,NULL,1,NULL,1);
}
/*****/
void loop(){

}

```

```
/******  
void OLED(void*parameter){  
  for(;;){  
    float humidityValue = dht.readHumidity();  
    float temperatureValue = dht.readTemperature();  
    //Display Temperature On OLED  
    display.clearDisplay();  
    display.setTextColor(WHITE);  
    display.setTextSize(1);  
    display.setCursor(0, 0);  
    display.print("Temperature: ");  
    display.setTextSize(2);  
    display.setCursor(0, 10);  
    display.print(temperatureValue);  
    display.print(" ");  
    display.setTextSize(1);  
    display.cp437(true);  
    display.write(167);  
    display.setTextSize(2);  
    display.print("C");  
  
    //Display Humidity On OLED  
    display.setTextSize(1);  
    display.setCursor(0, 35);  
    display.print("Humidity: ");  
    display.setTextSize(2);  
    display.setCursor(0, 45);  
    display.print(humidityValue);  
    display.print(" %");  
    display.display();  
    vTaskDelay(5000 / portTICK_PERIOD_MS);  
    soilA_emoji();  
  }  
}
```



```

vTaskDelay(5000 / portTICK_PERIOD_MS);
soilB_emoji();
vTaskDelay(5000 / portTICK_PERIOD_MS);
}
}
/*****/
void soilA_emoji(){
float soilMoistureValueA = analogRead(soilA_Pin);
int percentA = map(soilMoistureValueA, AirValue, WaterValue, 0, 100);
if (percentA < 0){
soilmoisturepercentA = 0;
}
else if (percentA > 100){
soilmoisturepercentA = 100;
}
else if (percentA >= 0 && percentA <= 100){
soilmoisturepercentA = percentA;
}
if (soilmoisturepercentA >= 0 && soilmoisturepercentA <= 30){
//Crying
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame05, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);

```

```
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame25, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame35, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame45, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
```

```
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame55, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame65, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame75, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
```

```
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame85, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant A need Water...");
display.drawBitmap(10, 8, frame95, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(70);
}
else if (soilmoisturepercentA > 30 && soilmoisturepercentA <= 70){
  //Neutral
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print ("Plant A looks good...");
  display.drawBitmap(10, 8, frame03, 48, 48, WHITE);
  display.setCursor(65, 25);
  display.print(soilmoisturepercentA);
  display.println(" %");
  display.display();
  delay(60);
}
```

```
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame013, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame23, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame33, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);
```

```
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame43, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame53, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame63, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);
```

```
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame73, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A looks good...");
display.drawBitmap(10, 8, frame83, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(50);
}
else if (soilmoisturepercentA > 70 && soilmoisturepercentA <= 100){
  //Happy
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print ("Plant A very fresh...");
  display.drawBitmap(10, 8, frame012, 48, 48, WHITE);
  display.setCursor(65, 25);
  display.print(soilmoisturepercentA);
  display.println(" %");
  display.display();
```

```
delay(40);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame22, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(40);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame42, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(40);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame52, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
```



```
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame72, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame82, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame92, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
```

```
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame102, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame122, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant A very fresh...");
display.drawBitmap(10, 8, frame132, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentA);
display.println(" %");
display.display();
```

```
    delay(30);
  }
}

/*****/
void soilB_emoji(){
  float soilMoistureValueB = analogRead(soilB_Pin);
  int percentB = map(soilMoistureValueB, AirValue, WaterValue, 0, 100);
  if (percentB < 0) {
    soilmoisturepercentB = 0;
  }
  else if (percentB > 100){
    soilmoisturepercentB = 100;
  }
  else if (percentB >= 0 && percentB <= 100){
    soilmoisturepercentB = percentB;
  }
  if (soilmoisturepercentB >= 0 && soilmoisturepercentB <= 30){
    //Crying
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print("Plant B need Water...");
    display.drawBitmap(10, 8, frame05, 48, 48, WHITE);
    display.setCursor(65, 25);
    display.print(soilmoisturepercentB);
    display.println(" %");
    display.display();
    delay(30);

    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 0);
```

```
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame25, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame35, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame45, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
```

```
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame55, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame65, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame75, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
```

```
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame85, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print("Plant B need Water...");
display.drawBitmap(10, 8, frame95, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(70);
}
else if (soilmoisturepercentB > 30 && soilmoisturepercentB <= 70){
  //Neutral
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print ("Plant B looks good...");
  display.drawBitmap(10, 8, frame03, 48, 48, WHITE);
  display.setCursor(65, 25);
  display.print(soilmoisturepercentB);
  display.println(" %");
  display.display();
  delay(60);

  display.clearDisplay();
```

```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame013, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame23, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame33, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
```

```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame43, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame53, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame63, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
```



```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame73, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(60);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B looks good...");
display.drawBitmap(10, 8, frame83, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(50);
}
else if (soilmoisturepercentB > 70 && soilmoisturepercentB <= 100){
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame012, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(40);
display.clearDisplay();
```

```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame22, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(40);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame42, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(40);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame52, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
```

```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame72, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame82, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame92, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

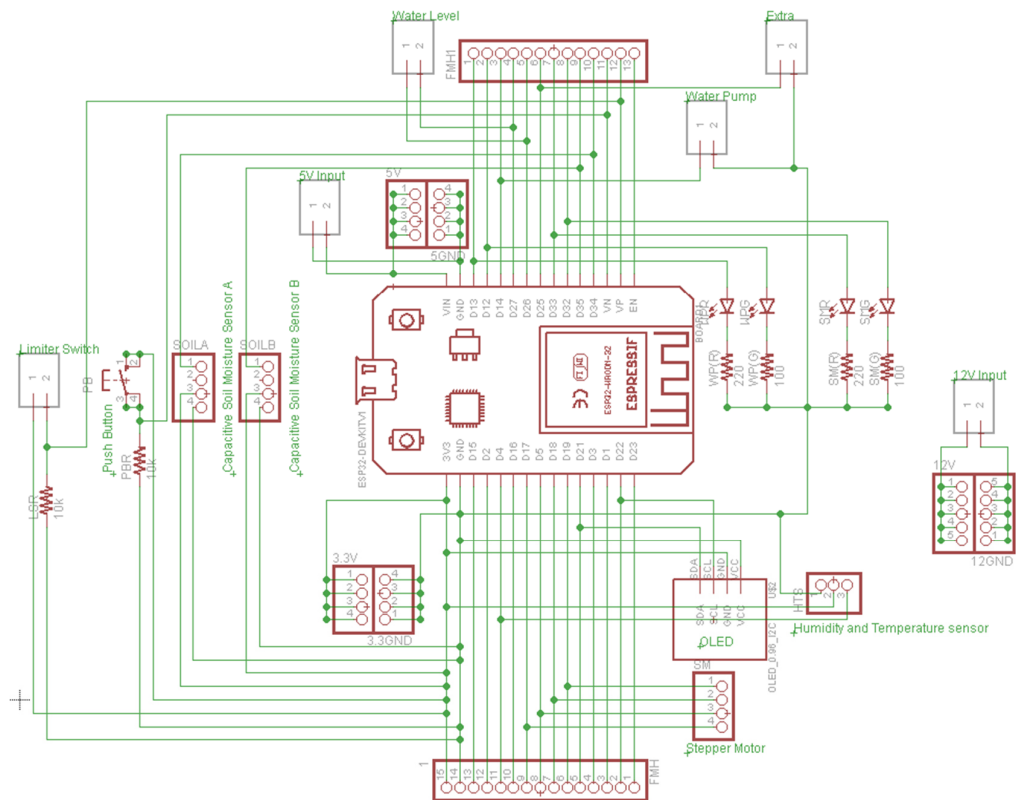
display.clearDisplay();
```

```
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame102, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

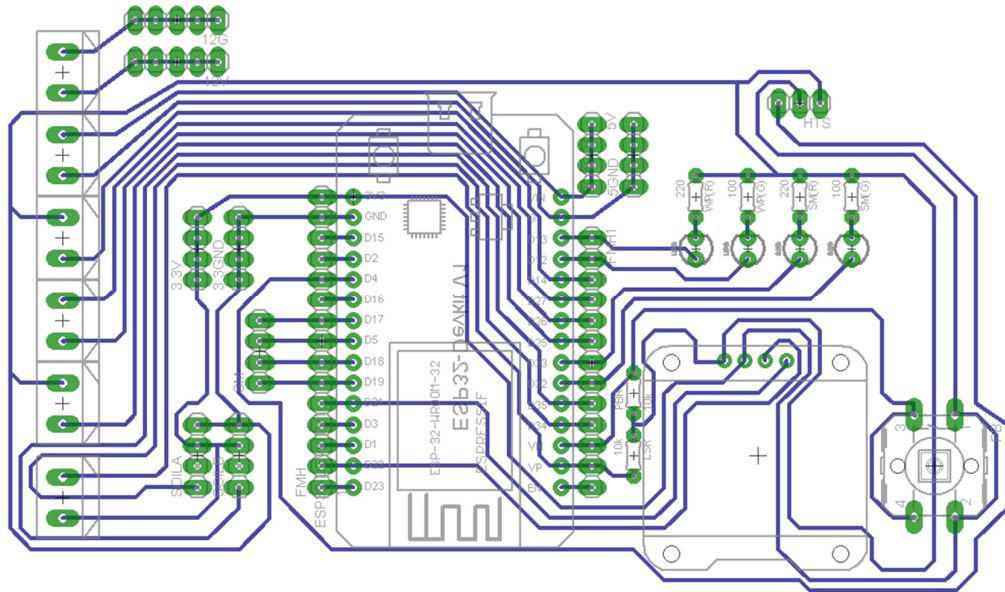
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame122, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);

display.clearDisplay();
display.setTextSize(1);
display.setCursor(0, 0);
display.print ("Plant B very fresh...");
display.drawBitmap(10, 8, frame132, 48, 48, WHITE);
display.setCursor(65, 25);
display.print(soilmoisturepercentB);
display.println(" %");
display.display();
delay(30);
}
}
```

APPENDIX B: Printed Circuit Board Design



Schematic Diagram of the Smart Agriculture and Smart Irrigation System



Board design of the Smart Agriculture and Smart Irrigation System