



Faculty of Computer Science and Information Technology

***IOT-BASED INDOOR PLANT CARE***

Rofinna Ellya Embang Anak Umar @ Richard

Bachelor of Computer Science with Honours (Network Computing)

2025

# **IOT-BASED INDOOR PLANT CARE**

**ROFINNA ELLYA EMBANG ANAK UMAR @ RICHARD**

This project is submitted in partial fulfillment of the requirements for the degree of  
Bachelor of Computer Science and Information Technology  
(Network Computing)

Faculty of Computer Science and Information Technology  
UNIVERSITI MALAYSIA SARAWAK  
2025

# **PENJAGAAN TUMBUHAN DALAMAN BERASASKAN IOT**

**ROFINNA ELLYA EMBANG ANAK UMAR @ RICHARD**

Projek ini merupakan salah satu keperluan untuk Ijazah Sarjana Muda Sains  
Komputer dan Teknologi Maklumat  
(Pengkomputeran Rangkaian)

Fakulti Sains Komputer dan Teknologi Maklumat  
UNIVERSITI MALAYSIA SARAWAK  
2025

## DECLARATION

I, Rofinna Ellya Embang Anak Umar @ Richard with matric number of 81056, with this declare that this project **“IoT-based Indoor Plant Care”** submitted by me in partial fulfilment of the requirements for the Bachelor of Computer Science with Honours (Software Engineering) is a record of an original work carried out by me under the guidance of **Madam Seleviawati bt Tarmizi**. I further declare that the work reported in this project has not been previously or concurrently submitted for any other degree in UNIMAS or another institution. I declare that I have not copied or plagiarized from any other sources except quotations and citations which have been duly acknowledged.



.....

ROFINNA ELLYA EMBANG ANAK UMAR @ RICHARD (81056)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

## **ACKNOWLEDGEMENT**

I would like to express my gratitude to my supervisor, Madam Seleviawati bt Tarmizi, for her guidance throughout my Final Year Project 1 (FYP1). Her feedback, guidance, and unwavering support truly inspired me to successfully complete my FYP1. Her insight and feedback have been invaluable, and I truly appreciate the effort she put into helping me grow and succeed. I must also acknowledge Professor Dr. Wang Yin Chai, who served as my Final Year Project coordinator, for his invaluable advice and guidance in the successful completion of my project.

I would convey my sincere gratitude to all individuals who have supported me throughout my journey, especially my family, whose constant love and support have been a source of strength. Their belief in me, especially when I am at my lowest, has motivated me to keep on pushing forward.

Lastly, I would like to extend my heartfelt thanks to my close friends for your encouragement and company, as well as for staying together even at our lowest. The way we are facing this journey together is a rewarding experience for me.

## **ABSTRACT**

The goal of this project is to develop an Internet of Things (IoT)-based system that automates plant care in order to address the issue of indoor plant care for people with hectic schedules. Even while indoor plants are well-known for their aesthetic and environmental advantages, they need constant care, particularly regarding lighting and watering. However, a lot of people lack the requisite expertise or find it difficult to find the time to care for plants. Often, existing solutions are either complex or not designed for indoor use. The proposed method uses fuzzy logic to automate watering based on real-time data from temperature and soil moisture sensors in order to overcome these difficulties and guarantee that plants get the proper amount of water without running the risk of being overwatered or underwatered. Additionally, users can customise watering for individual plants based on their specific requirements using a predetermined profile-based setting. This method gives plant owners a dependable and effective way to keep their plants in ideal conditions while lowering the need for physical intervention. The system guarantees continuous care with features like automated changes and real-time monitoring, making it simple for customers to maintain the health of their plants despite a hectic schedule.

## ABSTRAK

Matlamat projek ini adalah untuk membangunkan sistem berasaskan *Internet of Things* (IoT) yang mengautomatikkan penjagaan tumbuhan bagi menangani isu penjagaan tumbuhan dalam rumah untuk individu yang sibuk. Walaupun tumbuhan dalam rumah terkenal dengan kelebihan estetik dan persekitarannya, mereka memerlukan penjagaan yang berterusan, terutamanya dalam hal pencahayaan dan penyiraman. Namun, ramai orang kekurangan kepakaran yang diperlukan atau sukar untuk mencari masa bagi menjaga tumbuhan. Penyelesaian yang ada sering kali rumit atau tidak direka untuk kegunaan dalam rumah. Kaedah yang dicadangkan menggunakan *fuzzy logic* untuk mengautomatikkan penyiraman berdasarkan data masa nyata daripada sensor suhu dan kelembapan tanah bagi mengatasi masalah ini dan memastikan tumbuhan menerima jumlah air yang tepat tanpa risiko disiram berlebihan atau kekeringan. Selain itu, pengguna boleh menyesuaikan jadual penyiraman untuk tumbuhan individu berdasarkan keperluan khusus mereka melalui tetapan berasaskan profil. Kaedah ini memberikan pemilik tumbuhan cara yang boleh dipercayai dan berkesan untuk memastikan tumbuhan mereka berada dalam keadaan ideal sambil mengurangkan keperluan campur tangan fizikal. Sistem ini memastikan penjagaan yang berterusan dengan ciri seperti perubahan automatik dan pemantauan masa nyata, menjadikannya mudah bagi pengguna untuk menjaga kesihatan tumbuhan mereka walaupun dengan jadual yang sibuk.

## TABLE OF CONTENT

|                                                                                            |            |
|--------------------------------------------------------------------------------------------|------------|
| <b>DECLARATION .....</b>                                                                   | <b>i</b>   |
| <b>ACKNOWLEDGEMENT .....</b>                                                               | <b>iii</b> |
| <b>ABSTRACT .....</b>                                                                      | <b>iv</b>  |
| <b>ABSTRAK.....</b>                                                                        | <b>v</b>   |
| <b>LIST OF FIGURES .....</b>                                                               | <b>ix</b>  |
| <b>LIST OF TABLES .....</b>                                                                | <b>xi</b>  |
| <b>LIST OF ABBREVIATIONS.....</b>                                                          | <b>xii</b> |
| <b>CHAPTER 1: INTRODUCTION.....</b>                                                        | <b>1</b>   |
| 1.1 Introduction.....                                                                      | 1          |
| 1.2 Problem Statement .....                                                                | 3          |
| 1.3 Objectives .....                                                                       | 5          |
| 1.4 Methodologies.....                                                                     | 5          |
| 1.5 Scope.....                                                                             | 7          |
| 1.6 Significant of Project.....                                                            | 8          |
| 1.7 Project Schedule.....                                                                  | 9          |
| 1.8 Expected Outcomes .....                                                                | 9          |
| <b>CHAPTER 2: LITERATURE REVIEW .....</b>                                                  | <b>11</b>  |
| 2.1 Introduction.....                                                                      | 11         |
| 2.2 Review on Existing system .....                                                        | 11         |
| 2.2.1 IoT-based Smart Gardening System .....                                               | 12         |
| 2.2.2 P4L (Plants for Life), Internet of Things (IoT)-based Indoor Plant Care System ..... | 13         |
| 2.2.3 IoT-based Automatic Plant Sprinkler and Light Regulation .....                       | 15         |
| 2.3 Table of Comparison between Existing System .....                                      | 16         |
| 2.4 Tools and Technology Used.....                                                         | 18         |
| 2.4.1 Hardware .....                                                                       | 18         |
| 2.4.2 Software .....                                                                       | 21         |
| 2.5 Findings and Discussion .....                                                          | 24         |
| 2.6 Summary.....                                                                           | 26         |
| <b>CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN .....</b>                                    | <b>27</b>  |
| 3.1 Introduction.....                                                                      | 27         |
| 3.2 Agile Methodologies .....                                                              | 27         |
| 3.3 Planning .....                                                                         | 28         |

|                                                              |            |
|--------------------------------------------------------------|------------|
| 3.3.1 Questionnaire .....                                    | 28         |
| 3.3.2 Software Requirements .....                            | 34         |
| 3.3.3 Hardware Requirements .....                            | 35         |
| 3.4 Design.....                                              | 38         |
| 3.4.1 Use Case .....                                         | 39         |
| 3.4.2 Activity Diagram .....                                 | 42         |
| 3.4.3 Layout of Proposed System .....                        | 43         |
| 3.5 Development.....                                         | 46         |
| 3.6 Testing.....                                             | 46         |
| 3.7 Deployment .....                                         | 47         |
| 3.8 Review .....                                             | 47         |
| 3.9 Summary .....                                            | 48         |
| <b>CHAPTER 4: IMPLEMENTATION .....</b>                       | <b>49</b>  |
| 4.1 Introduction.....                                        | 49         |
| 4.1.1 System architecture.....                               | 49         |
| 4.2 System Configuration and Setup.....                      | 50         |
| 4.2.1 Arduino IoT Cloud and Remote App Configuration .....   | 50         |
| 4.3 System Features and Operations .....                     | 62         |
| 4.4 Workflow of Proposed System.....                         | 67         |
| 4.5 Coding of Proposed Algorithm .....                       | 71         |
| 4.5.1 Overall Indoor Plant Condition.....                    | 71         |
| 4.5.2 Fuzzy Logic for Automated Watering System .....        | 74         |
| 4.5.3 Automated Watering Setting Based on Plant Profile..... | 77         |
| 4.6 Summary .....                                            | 81         |
| <b>CHAPTER 5: TESTING .....</b>                              | <b>83</b>  |
| 5.1 Introduction.....                                        | 83         |
| 5.2 Functional Testing .....                                 | 83         |
| 5.2.1 Unit Testing.....                                      | 84         |
| 5.2.2 Integration Test .....                                 | 89         |
| 5.3 Non-Functional Testing .....                             | 91         |
| 5.4 Analysis on User Acceptance Survey.....                  | 96         |
| 5.5 Discussion on Results .....                              | 103        |
| 5.6 Summary .....                                            | 104        |
| <b>CHAPTER 6: CONCLUSION AND FUTURE WORKS .....</b>          | <b>105</b> |
| 6.1 Introduction.....                                        | 105        |

|                                             |     |
|---------------------------------------------|-----|
| 6.2 Objectives and Achievements .....       | 105 |
| 6.3 Project Limitation and Constraint ..... | 107 |
| 6.4 Future Works .....                      | 110 |
| 6.5 Conclusion .....                        | 112 |
| <b>REFERENCES</b> .....                     | 113 |
| <b>APPENDICES</b> .....                     | 116 |
| Appendix A .....                            | 116 |
| Appendix B .....                            | 118 |
| Appendix C .....                            | 121 |
| Appendix D .....                            | 125 |

## LIST OF FIGURES

|                                                                                                                          |    |
|--------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1.1 Agile Methodology (Asana, 2024) .....                                                                         | 5  |
| Figure 1.2 Gantt Chart of Project .....                                                                                  | 9  |
|                                                                                                                          |    |
| Figure 2.1 Smart Gardening System Architecture Design (Choudhari et al., 2023).....                                      | 13 |
| Figure 2.2 The P4L system architecture (Ulloa et al., 2023) .....                                                        | 14 |
| Figure 2.3 IoT-based Automatic Plant Sprinkler and Light Regulation System<br>Architecture (Setiawan et al., 2023) ..... | 16 |
| Figure 2.4 NodeMCU ESP8266.....                                                                                          | 18 |
| Figure 2.5 Arduino Uno R3 .....                                                                                          | 19 |
| Figure 2.6 ESP32.....                                                                                                    | 19 |
| Figure 2.7 Example mobile application and web dashboard of user interface of Blynk<br>(Choudhari et al., 2023).....      | 21 |
| Figure 2.8 Example of Arduino IDE Interface.....                                                                         | 22 |
|                                                                                                                          |    |
| Figure 3.1 The first question for Section A.....                                                                         | 28 |
| Figure 3.2 The second question for Section A.....                                                                        | 29 |
| Figure 3.3 The third question for Section A .....                                                                        | 29 |
| Figure 3.4 The first question for Section B.....                                                                         | 30 |
| Figure 3.5 The second question for Section B.....                                                                        | 30 |
| Figure 3.6 The third question for Section B .....                                                                        | 30 |
| Figure 3.7 The fourth question for Section B .....                                                                       | 31 |
| Figure 3.8 The fifth question for Section B.....                                                                         | 31 |
| Figure 3.9 The first question for Section C.....                                                                         | 32 |
| Figure 3.10 The second question for Section C.....                                                                       | 32 |
| Figure 3.11 The third question for Section C .....                                                                       | 33 |
| Figure 3.12 The fourth question for Section C.....                                                                       | 33 |
| Figure 3.13 Arduino IDE.....                                                                                             | 34 |
| Figure 3.14 Blynk.....                                                                                                   | 34 |
| Figure 3.15 ESP32 38-pin .....                                                                                           | 35 |
| Figure 3.16 Breadboard.....                                                                                              | 35 |
| Figure 3.17 Soil Moisture Sensor.....                                                                                    | 36 |
| Figure 3.18 DHT22 Module.....                                                                                            | 36 |
| Figure 3.19 5V Relay Module.....                                                                                         | 36 |
| Figure 3.20 Water Pump and hose.....                                                                                     | 37 |
| Figure 3.21 Jumper Wires.....                                                                                            | 37 |
| Figure 3.22 LED light small DC model.....                                                                                | 37 |
| Figure 3.23 LDR Module .....                                                                                             | 38 |
| Figure 3.24 . IoT-based Indoor Plant Care System Use Case .....                                                          | 39 |
| Figure 3.25 IoT-based Indoor Plant Care System Activity Diagram .....                                                    | 42 |
| Figure 3.26 Full layout of proposed system .....                                                                         | 43 |
| Figure 3.27 Hardware circuit and software connection of proposed system.....                                             | 44 |
| Figure 3.1 Homepage of mobile application.....                                                                           | 44 |
| Figure 3.29 Homepage of plant type.....                                                                                  | 45 |
|                                                                                                                          |    |
| Figure 4.1 IoT-based Indoor Plant Care Prototype.....                                                                    | 49 |
| Figure 4.2 Main page of Arduino IoT Cloud.....                                                                           | 50 |

|                                                                             |     |
|-----------------------------------------------------------------------------|-----|
| Figure 4.3 Arduino IoT Cloud main dashboard interface.....                  | 51  |
| Figure 4.4 “Things” interface.....                                          | 51  |
| Figure 4.5 Setup interface.....                                             | 52  |
| Figure 4.6 Associate Device Interface.....                                  | 53  |
| Figure 4.7 Select Third part Device.....                                    | 53  |
| Figure 4.8 Choose Suitable Model.....                                       | 54  |
| Figure 4.9 “Device ID” and “Secret Key” file.....                           | 55  |
| Figure 4.10 Name the device.....                                            | 55  |
| Figure 4.11 On right panel, select “Configure” on network section.....      | 56  |
| Figure 4.12 Network configuration details.....                              | 56  |
| Figure 4.13 Arduino Cloud Agent download.....                               | 57  |
| Figure 4.14 Arduino IoT Cloud mobile application.....                       | 60  |
| Figure 4.15 Main interface of the application Arduino IoT Cloud Remote..... | 61  |
| Figure 4.16 User interface of the IoT-based Indoor Plant Care.....          | 62  |
| Figure 4.17 Flowchart of workflow of Proposed System.....                   | 66  |
| Figure 4.18 Full code block updatePlantCondition().....                     | 70  |
| Figure 4.19 Ideal Conditions Based on Plant Type (1).....                   | 71  |
| Figure 4.20 Ideal Conditions Based on Plant Type (2).....                   | 71  |
| Figure 4.21 Ideal Conditions Based on Plant Type (3).....                   | 72  |
| Figure 4.22 Penalty Calculation (1).....                                    | 72  |
| Figure 4.23 Penalty Calculation (2).....                                    | 72  |
| Figure 4.24 The score clamped between 0% and 100%.....                      | 72  |
| Figure 4.25 The complete code block for the applyFuzzyLogic() function..... | 73  |
| Figure 4.26 Code from applyFuzzyLogic() (1).....                            | 74  |
| Figure 4.27 Code from applyFuzzyLogic() (2).....                            | 75  |
| Figure 4.28 Code from applyFuzzyLogic() (3).....                            | 75  |
| Figure 4.29 Code from applyFuzzyLogic() (4).....                            | 75  |
| Figure 4.30 Code from applyFuzzyLogic() (5).....                            | 76  |
| Figure 4.31 The complete code block for the handleWatering() function.....  | 77  |
| Figure 4.32 Code from handleWatering() (1).....                             | 78  |
| Figure 4.33 Code from handleWatering() (2).....                             | 79  |
| Figure 4.34 Code from handleWatering() (3).....                             | 79  |
| Figure 4.35 Code from handleWatering() (4).....                             | 79  |
| Figure 4.37 Code from handleWatering() (5).....                             | 79  |
| Figure 5.1 The first question for Section A.....                            | 95  |
| Figure 5.2 The second question for Section A.....                           | 95  |
| Figure 5.3 The first question for Section B.....                            | 96  |
| Figure 5.4 The second question for Section B.....                           | 96  |
| Figure 5.5 The third question for Section B.....                            | 97  |
| Figure 5.6 The first question for Section C.....                            | 97  |
| Figure 5.7 The second question for Section C.....                           | 98  |
| Figure 5.8 The third question for Section C.....                            | 98  |
| Figure 5.9 The first question for Section D.....                            | 99  |
| Figure 5.10 The second question for Section D.....                          | 99  |
| Figure 5.11 The first question for Section E.....                           | 100 |
| Figure 5.12 The second question for Section E.....                          | 101 |

## LIST OF TABLES

|                                                                           |     |
|---------------------------------------------------------------------------|-----|
| Table 2.1 Table of Reviewed System Comparison .....                       | 16  |
| Table 3.1 Overview of the software requirements .....                     | 34  |
| Table 3.2 Overview of the hardware requirements .....                     | 35  |
| Table 3.3 IoT-based Indoor Plant Care System Use Case Details .....       | 41  |
| Table 4.1 Features of IoT-based Indoor Plant Care Mobile Application..... | 63  |
| Table 5.1 Unit test case for sensor reading.....                          | 84  |
| Table 5.2 Unit test case for real-time sensor data display.....           | 84  |
| Table 5.3 Unit test case for watering control system.....                 | 85  |
| Table 5.4 Unit test case for plant health evaluation system.....          | 87  |
| Table 5.5 Unit test case for automatic grow light control.....            | 87  |
| Table 5.6 Unit test case for Overall Integration.....                     | 88  |
| Table 5.7 Unit test case for Performance Monitoring Module.....           | 91  |
| Table 5.8 Unit test case for System Monitoring Module.....                | 92  |
| Table 5.9 Unit test case for Mobile app.....                              | 94  |
| Table 6.1 Details of Objectives and Achievement.....                      | 104 |

## LIST OF ABBREVIATIONS

|                       |                                    |
|-----------------------|------------------------------------|
| <b>IoT</b>            | Internet of Things                 |
| <b>CPU</b>            | Central Processing Unit            |
| <b>UI</b>             | User Interface                     |
| <b>LDR</b>            | Light-Dependent Resistor           |
| <b>NH<sub>4</sub></b> | Ammonium                           |
| <b>CO<sub>2</sub></b> | Carbon Dioxide                     |
| <b>IDE</b>            | Integrated Development Environment |
| <b>PWM</b>            | Pulse Width Modulation             |
| <b>GPIO</b>           | General-Purpose Input/Output       |
| <b>DC</b>             | Direct current                     |
| <b>Wi-Fi</b>          | Wireless Fidelity                  |
| <b>AI</b>             | Artificial Intelligence            |

## CHAPTER 1: INTRODUCTION

### 1.1 Introduction

The interior landscape has gained widespread recognition after the construction of the Ford Foundation Centre in New York, USA, and the Hyatt Regency in Atlanta, USA (Chatakul et al.,2022). Nowadays, the popularity of indoor plants is surging since they offer an opportunity to create more urban green spaces (Chatakul et al.,2022). Many people maintain indoor plants in their homes primarily for aesthetic purposes, and some even believe they can enhance the quality of the air, where it helps to absorb pollutants and releasing oxygen, making them an appealing addition to modern interiors. In addition, having indoor plant also a mood booster as it able to increase work productivity, as well as able to reduce the temperature of room temperature (Shamsuri, 2015). However, for plants to stay healthy, they need to be in their optimum condition and consistent care, such as providing them adequate light, water, etc. Unfortunately, due to their busy lives and hectic schedules, many individuals lack the time to properly care for their plants (Choudhari et al., 2023). In addition, some individuals may lack the necessary knowledge of plant maintenance where various plant species required specific needs. This eventually resulting in improper watering routing, insufficient light or unsuitable environment temperature. Poor plant care practices can lead to wilting and eventual death of the plant.

However, Internet-of-Things (IoT) can provide a solution to this issue (Ulloa et al., 2023). The IoT-based Indoor Plant Care system is a smart indoor gardening system that optimizes indoor plant care and reduces human intervention. The system integrates a microcontroller ESP32 with a soil moisture, temperature sensor and light sensors. By monitoring soil moisture and temperature, it helps to ensure the plant grows in an ideal environment; some plants are sensitive to extreme temperatures. This helps prevent stress on plants, which can lead to poor growth. In addition, by controlling both an

automated watering system and an artificial lighting setup, the system delivers water and light precisely when needed, creating an optimal environment regardless of time constraints or lighting limitations.

Moreover, Wi-Fi connectivity will transmit real-time, which allows the user to remotely monitor their indoor plant health. This project aims to minimise human intervention when caring for indoor plants; it is useful for busy people since they can monitor their plants remotely and without expertise. Furthermore, one of the key features of this system is an automated fuzzy logic-based watering system. ESP32 will control the amount of water needed by the plant based on the soil moisture and temperature sensor readings to keep the plant at its optimal moisture level and avoid underwatering and overwatering.

The automation of the system, which smartly adapts to the unique requirements of the plants, is its most alluring feature. An automated fuzzy logic-based watering system is incorporated into the system to guarantee that the plants receive the proper amount of water without running the risk of being overwatered or underwatered. Instead of true or false, fuzzy logic variables can be any real number between 0 and 1, which makes it a type of many-valued logic (GeeksforGeeks, 2018). The ESP32 microcontroller employs fuzzy logic to make small adjustments in response to changing environmental circumstances, including soil moisture, temperature, and ambient light. For example, the system will briefly turn on the watering pump if the soil is somewhat dry and the temperature is cool. The system will lengthen the watering period, though, if the soil is too dry and the temperature is high. This fuzzy logic method provides a more accurate watering schedule than conventional systems that rely on strict thresholds. Additionally, profile-based watering system is also included in the system, which enables users to customise watering schedules for various plant species. The system may modify watering schedules based on the individual needs of each plant by linking particular moisture and temperature profiles

to various plants. This ensures that each plant gets the ideal amount of water based on its particular demands. Healthy plant growth is ensured by this individualised method, which improves the system's accuracy and efficiency.

Furthermore, to guarantee that the plant gets enough light, even in low light levels, the system has an automated artificial lighting component in addition to watering. In order to encourage photosynthesis, this feature simulates natural day-night cycles by turning on when the light intensity drops below a predetermined threshold. Regardless of the weather outside, the system ensure that plants receive optimal light to support healthy growth.

In summary, the ESP32's Wi-Fi connectivity powers the system's ability to perform real-time monitoring by transmitting information to a mobile app, including temperature, lighting status, and soil moisture. This enables customers to make wise judgements by remotely monitoring the health of their plants. Users who might need to relocate their plants because of adverse climatic circumstances, including extremely high or low temperatures, will find this function particularly helpful. Fuzzy logic, automatic lighting, watering, and real-time monitoring together provide a smart and dependable indoor gardening solution that guarantees plants grow in ideal conditions without requiring regular human interaction.

## **1.2 Problem Statement**

Due to the beauty and benefits of indoor plants, many individuals have begun to keep them in their homes. Taking care of indoor plants can be a relaxing and rewarding hobby, but it can be a different experience for those with busy lifestyles. Having an indoor plant requires consistency, especially water and light. A busy lifestyle can make it difficult to find the right time to care for a plant, as work-life balance and other commitments can be challenging.

Without a consistent routine, an individual may not be aware of basic plant care and may fail to notice early signs of pests, diseases, or nutrient deficiencies. As a result of knowledge gaps, inadequate or irregular watering schedules combined with less-than-ideal climatic conditions, namely temperature, eventually cause plants to wilt or die.

Furthermore, although there are some existing digital solutions, they are often complex or designed for larger outdoor gardens rather than indoor environments. This creates a gap for a straightforward, affordable system that can provide timely and consistent indoor plant care with minimal user intervention. An excellent approach is the use of fuzzy logic in an IoT-based indoor plant care system. Fuzzy logic can implement adaptive decisions depending on the real-time measurements of soil moisture, temperature, and light activity so that optimal plant health can be achieved. This allows the system to modify lighting and watering as necessary. This smart and flexible system can also provide plant owners with a convenient but also effective system to take care of plants by reducing the number of manual adjustments that need to be made and preventing the risk of overwatering, underwatering, or not having enough light.

The system includes a profile-based configurable watering system to increase its effectiveness further. This feature enables users to customise watering schedules for various plants according to their requirements for temperature and moisture. This proposed system ensures that it gives every plant a profile, which adjusts to give the optimal amount of water in relation to the needs of a particular plant. Fuzzy logic and profile-based irrigation reduce the possibility of overwatering, underwatering, or insufficient sunlight while guaranteeing that plants receive precise care. When combined, these characteristics provide plant owners a smart, flexible, and user-friendly way to manage indoor plant care with little physical intervention.

### 1.3 Objectives

The objective of this project is to develop a mobile application of IoT-based Indoor Plant Care. The system will:

1. To monitor environmental factors such as soil moisture, temperature levels, and light intensity using sensors to gather real-time data for plant health.
2. To implement fuzzy logic in the watering system to analyse soil moisture levels and temperature, making adaptive decisions to ensure plants receive the right amount of water based on real-time conditions.
3. To automate the watering system based on fuzzy logic-driven analysis, adjusting watering duration and frequency including profile-based configuration watering system based on different plant's needs.
4. To automate lighting based on light sensor.

### 1.4 Methodologies

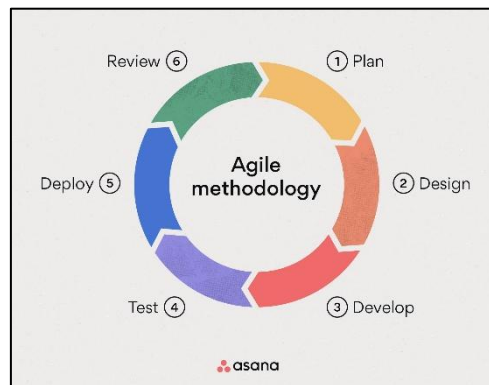


*Figure 1.1 Agile Methodology (Asana, 2024)*

For this project, agile methodologies are used. Agile methodology allows flexibility and focus on the user's needs. If any issues arise, it allows easy adjustment since it is an iterative approach. Using agile methodologies allows for the division of tasks into smaller, more manageable tasks.

## **I. Planning**

The planning phase involves identifying the project scope and requirements. The project goals are clearly outlined in this phase by identifying what tools and data sources will be used. As an example, this project will develop a system that monitors the soil moisture and temperature of indoor plants and automatically waters and lights them.

## **II. Designing**

During the design phase, a system architecture for the IoT-based plant care system will be designed, taking into account the connections between all components and the system's workflow. The user interface (UI) will be designed on how all the data and control buttons will be displayed. Moreover, the component layout of the system will be properly planned to ensure accessibility and efficient use of space.

## **III. Development**

Throughout the development phase, we implement all features in small, manageable increments and perform regular integration and testing in order to prevent difficult complications and guarantee compatibility.

## **IV. Test**

At the testing stage, every component is tested separately in order to prove the accuracy and precision. The whole system will then be tested to test compatibility among all the components with the ability to give correct data.

## **V. Deployment**

Before full deployment, the deploy phase sets up the entire model in controlled testing environments to assess its functionality and performance. During this phase, any issues with connectivity, sensors, and other issues are identified and addressed.

## **VI. Review (Maintenance)**

In the review phase, the entire system undergoes a comprehensive evaluation to identify areas for improvement based on testing. Therefore, we will make any necessary adjustments for continuous system improvement after evaluating the system under real-world conditions.

### **1.5 Scope**

The IoT-based Indoor Plant Care project aims to provide an automated plant care system for busy indoor plant owners who have limited knowledge about indoor plant necessities. The primary objective of this project is to create automation and monitoring of the health of a plant. Features of this system are soil moisture monitoring, soil temperature, automated watering and lighting, and online real-time data updates through a mobile application. We anticipated the completion of this project by July 2025. This system was designed for indoor environments such as offices and indoor gardens, among others. The system will operate within Wi-Fi-enabled spaces to ensure reliable connectivity with the mobile app for real-time updates. This innovation aims to provide a low-maintenance plant care solution by reducing human intervention with indoor plants and enabling them to meet their optimal needs. Humans can also monitor their plants remotely. The ESP32 microcontroller gathers data from the sensor and triggers automated watering or lighting when the sensor reading falls below a predetermined threshold. As the system communicates via Wi-Fi, it displays real-time data on plant health via a mobile app.

## **1.6 Significant of Project**

This project helps to simplify and enhance indoor plant care by automating crucial plant needs, such as watering and lighting. This reduces the risk of improper watering and insufficient lighting, ensuring the plant is maintained in good condition with minimal effort. Despite their hectic schedules, people can monitor their plant health remotely through a mobile application. Furthermore, this practice is sustainable as it utilizes resources efficiently, only activating the automated watering system when necessary. This system does not only assist individuals who are interested in plants but also demonstrates the growing potential of Internet of Things (IoT) technology in smart home devices that may encourage sustainability and ease of access.

## 1.7 Project Schedule

The project time schedule is used as an outline and as the reminder that the IoT-based Indoor Plant Care project is accomplished on time and that the deadlines are fulfilled. It includes a list of tasks, deadlines, and milestones. To guarantee timely and well-organised project execution, each task was given a definite start and end date. Planning, design, development, testing, deployment, and review are the six phases of this timeline, which follows the Agile methodology. Figure 1.2 displays the quick Gantt chart for the project, while Appendix A provides a more comprehensive timetable. Given the project's tight timeline, each phase can start before all the tasks from the previous phase are finished, or they can be executed concurrently to guarantee that time utilisation is properly optimised.

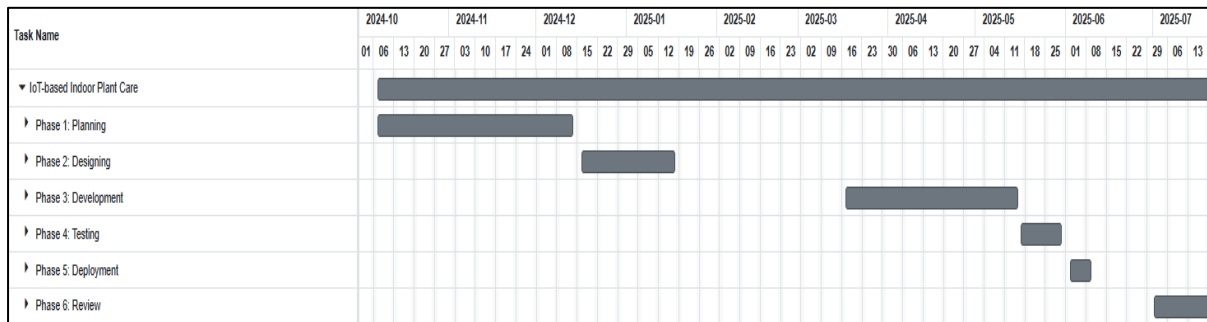


*Figure 1.2 Brief Gantt Chart of Project*

## 1.8 Expected Outcomes

The outcome of this project will be a fully automated plant care system that has an ability to monitor and maintain the best condition of the indoor plants. This project involved excellent improvements in several features of a mobile application that can monitor soil moisture, temperature, and a plant's current well-being. The user can supervise their general plant's health at any moment regardless of being further from the actual location. Furthermore, this project offers users convenience, which allows them to control the system by triggering the automated watering mechanism whenever the moisture or

temperature level drops under a specific point. In addition, this system also offers ensuring the indoor plant to receive sufficient light even in low light conditions.

## **CHAPTER 2: LITERATURE REVIEW**

### **2.1 Introduction**

It will be useful to review the literature that is related to the topic of developing an indoor plant care system based on the IoT before proceeding with the work. This chapter examines the prior relevant studies, systems, and technologies. It is possible to learn about the benefits, limitations, and relevance of previous research and comparable systems to the proposed solution by reviewing these sources. Additionally, the research shows how important the project's unique features are and helps find where current solutions fall short.

This chapter comprises multiple parts. The analysis of comparable existing systems, which examine their designs, features, and efficacy, highlights the available options for plant care. While a comparison table summarises the aspects of these systems to identify opportunities for improvement, the findings and discussions section delves deeper into the outcomes produced from these studies. We also provide a thorough overview of the components used in the development of the IoT-based Indoor Plant Care by discussing the tools and technology that support this project. Summaries conclude each chapter, synthesizing the key points and establishing the foundation for the project's future stages.

### **2.2 Review on Existing system**

An analysis of the current system entails an examination of the existing technologies, approaches, or solutions that have already been implemented to serve as comparisons to solve similar issues or achieve related goals. It helps in identifying the strengths and weaknesses of the existing system, and therefore it provides a better overview of what has been achieved and what must be achieved. Through their study, useful information can be obtained on how they work, their design, performance and drawbacks. Another benefit of this review is to prevent the repetition of the previous errors, adjust the strategies that

were already tested, and get new ideas to improve or create a more convenient and productive solution. It acts as a base in developing a proposed system that can address the weakness of the current methods and make use of successful components in them. Three existing systems are under analysis and comparison in this literature review.

### **2.2.1 IoT-based Smart Gardening System**

Choudhari et al. (2023) note that smart gardening systems are created based on the application of the Internet of Things (IoT) as a way to offer support in the automation and remote control and control in real-time to make the process of plant monitoring more comfortable and improved. The system includes a central processing unit (CPU), known as the NodeMCU, which is a versatile and cheap component with built-in Wi-Fi technology. The water pump and fertilizer dispenser are two examples of actuators that this controller helps control; it also helps communicate data to the Blynk platform so users may access it.

To keep the plant in its optimal state, the system continuously measures the critical parameter. The soil moisture sensor monitors the level of moisture in the soil and this is monitored in order to regulate the amount of water and prevent over or under watering the plants. On the other hand, a DHT11 sensor tracks environmental conditions for plants by measuring both humidity and temperature. To make sure the plant gets enough light to thrive, a light-dependent resistor (LDR) detects the intensity of the light. The sensors are actively gathering data, sending it to the cloud via NodeMCU, enabling the system to respond appropriately. Additionally, this system leverages the Blynk platform, an IoT platform. The Blynk platform provides an intuitive UI through its web dashboard and mobile app. This platform allows customers to remotely operate and monitor various system functions, including irrigation, lighting, and fertilizer distribution in real time.

The Internet of Things (IoT) smart gardening system provides the option to switch between an automatic and a manual mode, according to Choudhari et al. (2023). The sensor detects the level of moisture in soil and controls the water pump when the automated mode is enabled. In order to keep the soil around plants at the ideal moisture level, the pump will turn on automatically once the level drops below a certain threshold. In manual mode, the settings of the system can be accessed via the mobile application or online dashboard. Additionally, this system includes a remote-controlled automated fertiliser dispenser. This way, the plant will always get the nutrition it needs, without any human interference.

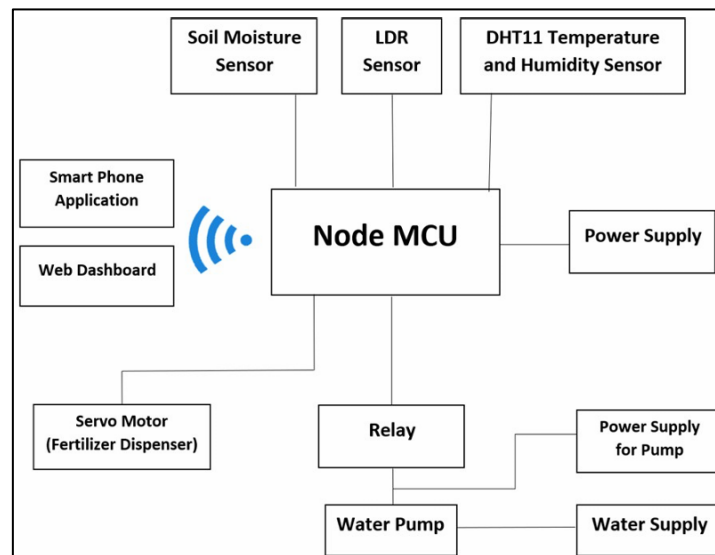


*Figure 2.1* Smart Gardening System Architecture Design (Choudhari et al., 2023)

### 2.2.2 P4L (Plants for Life), Internet of Things (IoT)-based Indoor Plant Care System

P4L (Plants for Life) is an innovative system that uses the internet of things to automate indoor plant care while enhancing the air quality indoors. This system integrates with sensors, actuators, and microcontrollers, such as the Arduino Uno R3. This system measures several significant parameters, such as humidity, temperature, light intensity, air quality, and environmental conditions (Ulloa et al., 2023). Soil moisture sensors tell

you when to water your plants based on their current moisture levels, while environmental temperature and humidity sensors assist you in cultivating the ideal environment for plant growth. Ulloa et al. (2023) said that air quality sensors may detect pollutants such as CO<sub>2</sub>, NH<sub>4</sub>, alcohol, and toluene, which can impact both plant and human health. Light intensity sensors, on the other hand, measure the amount of light reaching plants to ensure adequate facilitation of photosynthesis. The online application and mobile application of the software have a user-friendly interface to ensure the user can access real-time data, study history and change system settings. Through Wi-Fi, the P4L device may talk to these apps, which can exchange data in real time using WebSockets and do additional tasks using RESTful web services. The system automatically triggers watering, adjusts lighting, and alerts users when any monitored metric falls below a specific threshold, all pertaining to plant health and air quality. Improving indoor air quality and making plant care easier, this software and hardware combo lets customers easily monitor and manage their houseplants from afar.

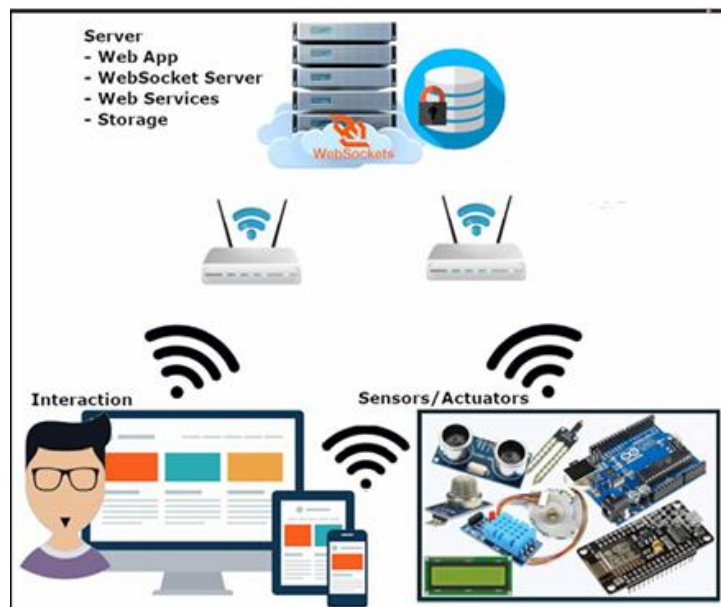


*Figure 2.2* The P4L system architecture (Ulloa et al., 2023)

### **2.2.3 IoT-based Automatic Plant Sprinkler and Light Regulation**

One such system that aims to automate plant maintenance using IoT technologies is the Automatic Plant Sprinkler and Light Regulation system (Setiawan et al., 2023). The system consists of a set of hardware and software allowing monitoring and controlling the care of the plant appropriately. The hardware in the system includes an Arduino Uno R3 microcontroller that reads the signals via different sensors, such as the AM 2302 temperature and humidity sensor, LDR photoresistor to sense light, and soil hygrometer to measure moisture content of the soil. A water pump is used for irrigation, while an ultraviolet LED is used for additional lighting. With the assistance of a NodeMCU ESP8266 module, the Android app Blynk is able to monitor the data in real-time, as it helps wireless connection. The use of sensor data to collect information enables the system to operate independently. The Arduino monitors and controls soil moisture, activating the water pump to water the plants when it falls below 60% (Setiawan et al., 2023). In a similar vein, when light levels drop too low, the UV LED will on to make sure photosynthesis gets the light it needs. By transmitting all data to the NodeMCU, the Blynk app lets users keep tabs on system performance remotely.

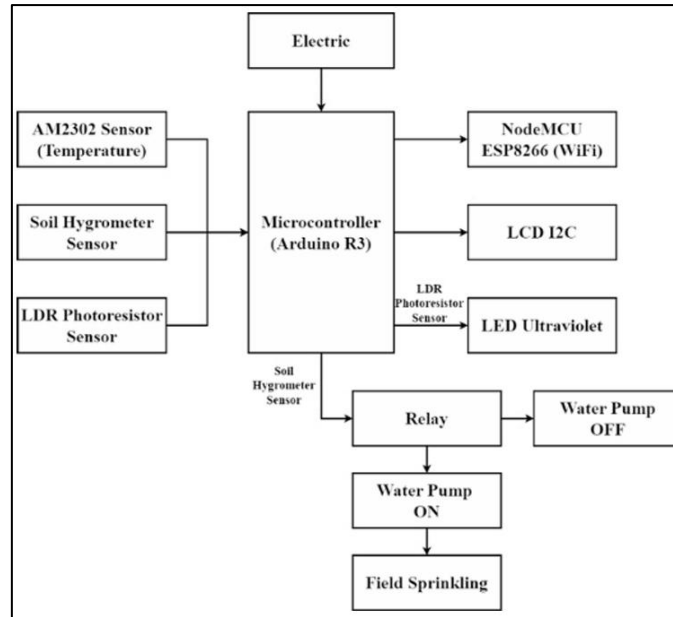


Figure 2.3 IoT-based Automatic Plant Sprinkler and Light Regulation System Architecture (Setiawan et al., 2023)

## 2.3 Table of Comparison between Existing System

Table 2.1 below shows the reviewed system comparison between proposed system , IoT-based Indoor Plant Care and Table 2.2 shows summary of reviewed system comparison.

Table 2.1 Table of Reviewed System Comparison

| Application<br>Features   | Similar Existing System          |                       |                                                          | Proposed System                       |
|---------------------------|----------------------------------|-----------------------|----------------------------------------------------------|---------------------------------------|
|                           | IoT-based Smart Gardening System | P4L (Plants for Life) | IoT-based Automatic Plant Sprinkler and Light Regulation | Iot-based Indoor Plant Care           |
| Microcontroller           | NodeMCU                          | Arduino Uno R3        | Arduino Uno R3, NodeMCU ESP8266                          | ESP32                                 |
| Software Used             | Blynk, Arduino IDE               | Arduino IDE           | Blynk                                                    | Arduino IoT Cloud, Arduino IoT Remote |
| Monitor plant remotely    | ✓                                | ✓                     | ✓                                                        | ✓                                     |
| Automatic watering system | ✓                                | ✓                     | ✓                                                        | ✓                                     |
| Automatic lighting system | X                                | ✓                     | ✓                                                        | ✓                                     |

|                                                                     |   |   |   |   |
|---------------------------------------------------------------------|---|---|---|---|
| <b>Application &amp; display</b>                                    | ✓ | ✓ | ✓ | ✓ |
| <b>Internet Connection</b>                                          | ✓ | ✓ | ✓ | ✓ |
| <b>Profile-based configuration watering system</b>                  | X | X | X | ✓ |
| <b>Incorporating with Fuzzy logic for automated watering system</b> | X | X | X | ✓ |

## 2.4 Tools and Technology Used

### 2.4.1 Hardware

#### 2.4.1.1 NodeMCU ESP8266

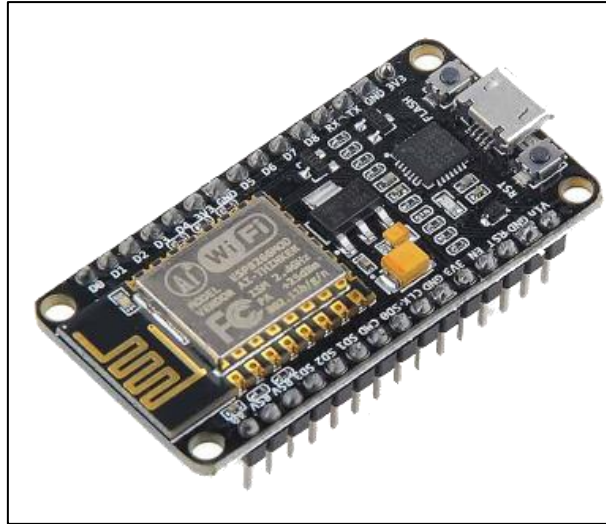


Figure 2.4 NodeMCU ESP8266

The ESP8266, an affordable System-on-a-Chip (SoC), serves as the foundation for NodeMCU, an open-source software and hardware development environment (*NodeMCU ESP8266 Specifications, Overview and Setting Up*, n.d.). An ESP8266 WiFi transceiver, according to a 2018 Random Nerd Tutorial review, is ideal for use in home automation and Internet of Things projects. The NodeMCU ESP8266, a low-powered microcontroller, has the ability to enter a deep sleep state to reduce its energy consumption (Getting Started with ESP8266 WiFi Transceiver (Review) | Random Nerd Tutorial, 2018)). Programming NodeMCU ESP8266 is a breeze, even for newcomers, thanks to the Arduino IDE, Lua, or MicroPython. This microcontroller is loved by both experts and hobbyists for its affordable price and practical features. However, it has its drawbacks, where it has a smaller number of general-purpose input/output (GPIO) pins. Despite its drawbacks, it is a great option for learning and experimenting with IoT projects due to its affordability and ease of usage. Figure 2.4 shows NodeMCU ESP8266.

### 2.4.1.2 Arduino Uno R3

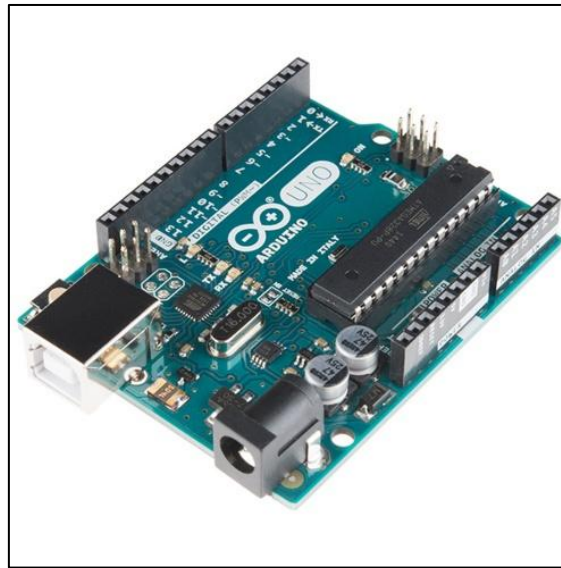


*Figure 2.5* Arduino Uno R3

The Arduino Uno R3 is one of the popular microcontroller boards that is ideal for developing electronic projects. It uses a USB port to communicate as well as program it, has six analogue input pins, 14 digital I/O pins, 6 of which are PWM capable, and runs on the ATmega328P microcontroller. It has a 5V operation voltage, hence seamless integration of sensors, actuators, or communication modules (Arduino, 2024). The Arduino Uno R3 is simple to use and cheap, which makes it attractive to beginner developers and professionals as well as education initiatives for Internet of Things (IoT) projects and prototypes. The Arduino IDE and its extensive library of pre-written code enable rapid prototyping. The Arduino Uno R3 are shown in Figure 2.5.

### 2.4.1.3 ESP32

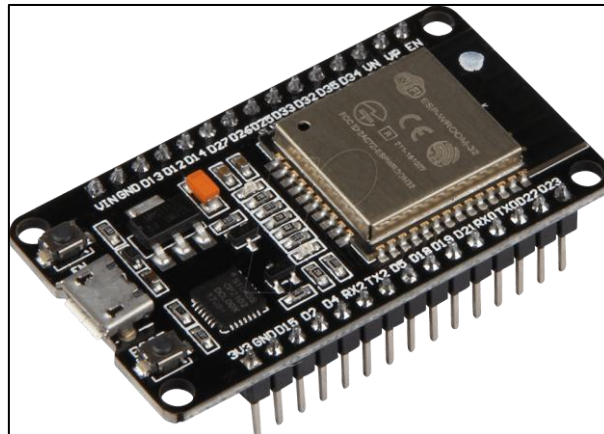


*Figure 2.6* ESP32

ESP32 is a dual-core microcontroller (MCU) made by Expressif Systems, combined with Wi-Fi and Bluetooth (Teja, 2021) and extensively applied in Internet of Things (IoT) projects and embedded systems, as it is known for being a low-cost and low-powered microcontroller. It has built-in Wi-Fi and Bluetooth, 2 cores processor up to 240 MHz, which is suitable to be used in wireless communication and remote control. The ESP32 has a broad selection of input/output interfaces, such as GPIO, ADC, DAC, PWM, I2C, SPI, UART. These interfaces allow the controller to connect to several sensors, actuators and so on. The ESP32 is also known to be power efficient, which is why it is suited to build battery-powered devices, like wearables, audio, baby monitors, and smart watches; the ESP32 should be very easy to use (Teja, 2021). The ESP32 flexibility, ease of use (when compared to other development environments such as Arduino IDE and MicroPython) and broad community support have made the ESP32 a favourite among developers, students and hobbyists developing smart home systems, automation, data logging and many other exciting applications. Figure 2.6 shows ESP32.

## 2.4.2 Software

### 2.4.2.1 Blynk

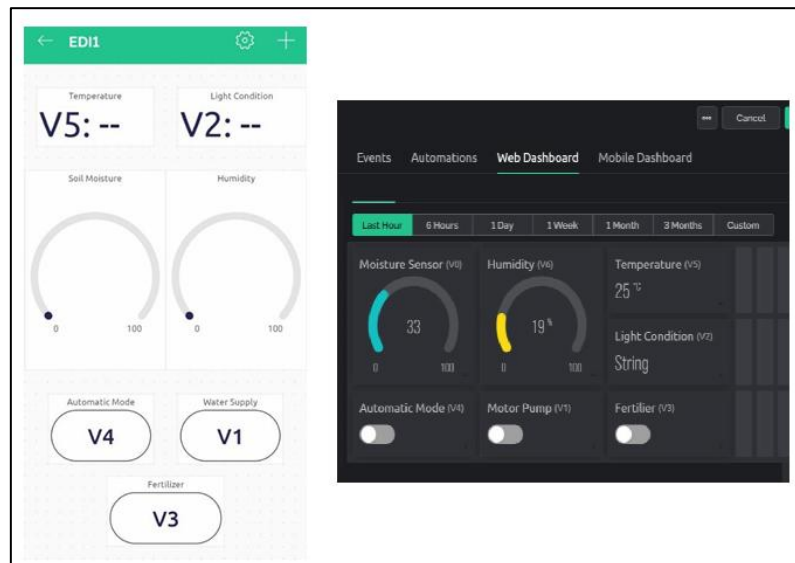


*Figure 2.6* Example mobile application and web dashboard of user interface of Blynk

(Choudhari et al., 2023)

Blynk is an adaptable platform that makes it simple to build Internet of Things (IoT) apps by bringing together software, hardware, and cloud services in a straightforward manner (Blynk, 2015). It offers dependable cloud connectivity through Blynk Cloud or private servers, has a user-friendly drag-and-drop interface for creating personalized online or mobile dashboards, and is compatible with a broad variety of devices, including Raspberry Pis, Arduinos, and ESP32s (Blynk, 2015). It makes managing and monitoring devices easier using pre-made widgets like sliders, buttons, and graphs. Blynk is ideal in most IoT projects because it also provides the opportunity to automate, set up schedules, and even send notifications in real-time. Fast prototyping, compatibility with multiple platforms, scalability for both small and big deployments, and beginner-friendliness are some of its benefits. However, it also has some drawbacks, such as its reliance on the internet for most functions, the lack of features in the free plan, the cost of more sophisticated features, its limited customization compared to custom-built solutions, and

the potential impact of cloud latency on real-time apps. Blynk is still a useful tool for people who are interested in or work in the Internet of Things (IoT), even with these restrictions. Figure 2.6 shows the example of mobile application and web dashboard of user interface of Blynk

#### 2.4.2.2 Arduino IDE

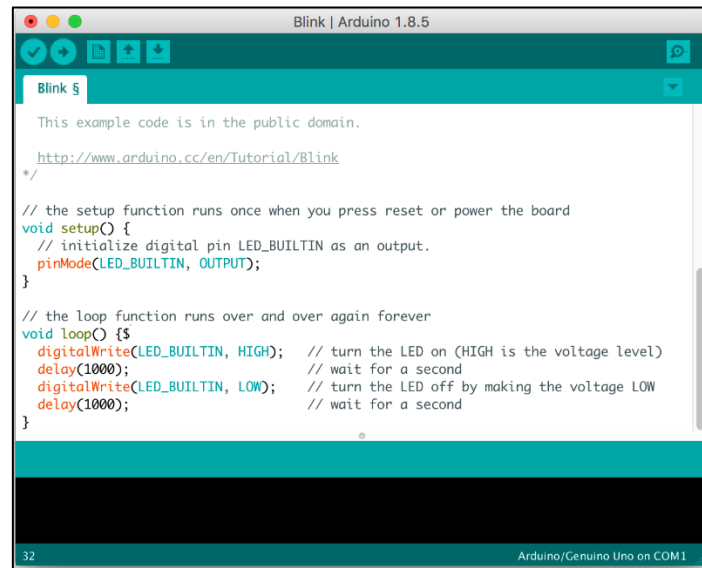


*Figure 2.7* Example of Arduino IDE Interface

An open-source software platform, the Arduino IDE makes it simple to write, compile, and upload code to microcontroller boards made by Arduino (Andprof, 2024). The language it supports is the Arduino programming language, which is based on C/C++ and has a simple interface. That is why it is ideal for the beginners. An intuitive code editor with syntax highlighting and auto-indentation is one of the features. Other useful tools include a built-in serial monitor that makes it easy to debug and monitor in real-time, extensive library support for sensors and actuators, and simple one-click compilation and uploading. Its advantages are that it is well suited to rapid prototyping, as it is easy to work with, compatible with a large number of Arduino boards, and there exists a huge amount of community resources. Unfortunately, the Arduino IDE struggles with sophisticated or large-scale projects, has limited debugging capabilities, and lacks advanced features such

as code restructuring in its minimalist editor. In spite of these caveats, the Arduino IDE is still a fantastic pick for teachers, students, and anyone just starting out in the world of embedded systems and electronics. The example of Arduino IDE interface is shown in Figure 2.7.

#### 2.4.2.3 Arduino IoT Cloud and Remote



Figure 2.8 Example of Arduino IoT Cloud interface

Arduino IoT Cloud is an internet-based platform developed by Arduino. It provides users with the opportunity to simply build, observe and work on projects related to the Internet of Things (IoT) (*Getting Started with Arduino Cloud*, 2024). Its straightforward compatibility with Arduino boards that are compatible with it, including the ESP32, MKR, and Nano series (*Getting Started with Arduino Cloud*, 2024), allows users to add devices to the cloud to monitor data in real-time and control them remotely. The users can also generate dashboards enabling them to visualise sensor information, operate actuators, and receive alerts or messages through the cloud platform. The Arduino IoT Cloud also has its own mobile application called Arduino IoT Remote. This software enables the customers to have the freedom to view their dashboards wherever they are through the use of a mobile gadget like a smartphone or a tablet. This software provides a wireless interface between the user and their gadget of choice, making it a very good

candidate for applications involving plant monitoring, smart home systems, and other remote-control projects. The union of the Arduino IoT Remote app and the Arduino IoT Cloud provides an extensive and conveniently available solution to building interconnected systems, offering a simple interface and cloud-hosted tools.

## **2.5 Findings and Discussion**

The existing systems that have been studied have several characteristics that are designed to improve plant care by means of automation, remote control, and real-time monitoring. One example is the study of Choudhari et al. (2023), who show how to incorporate sensors for soil moisture, light intensity, and surrounding conditions using Wi-Fi-enabled NodeMCU microcontrollers.... In a similar vein, Ulloa et al. (2023) highlight how the P4L system improves indoor air quality and plant health with its extensive monitoring capabilities. Setiawan et al. (2023) demonstrate the feasibility of automating irrigation and lighting using a combination of an Arduino Uno R3 and a NodeMCU ESP8266. Collectively, these studies show how useful low-cost technologies are for making IoT-based solutions accessible and user-friendly, such as NodeMCU, Arduino-compatible hardware, and platforms like Blynk.

By eliminating the need for human intervention and optimizing environmental conditions, the reviewed systems effectively automate vital plant care operations, such as watering, lighting, and fertilization. In addition, they use Internet of Things (IoT) systems for remote monitoring, which helps to save water and energy by turning components on and off only when needed. Because they use open-source software and inexpensive, widely available hardware, these systems also have intuitive user interfaces. Even with all these positive aspects, there are still some drawbacks. Interruptions in the network may negatively affect remote functionality due to its reliance on constant internet access. Similarly, reliance on a power source can be a problem during blackouts, and the need for technical knowledge could prevent non-technical users from accessing

certain features. Some systems also lack the ability to customize species-specific plant care and use inexpensive sensors, which can reduce accuracy and durability.

The goal of the proposed Internet of Things (IoT) indoor plant care system is to address these shortcomings by enhancing current solutions. The system leverages an ESP32 microcontroller to overcome the hardware limitations of previous systems. The ability to automatically water plants in response to adjustments in the surrounding environment or a decrease in sensor readings below a predetermined threshold is one of the key advantages of this system. In addition, a database of plant-specific variables allows for customized care, ensuring optimal growth for a diverse array of species that are primarily grown indoors. Furthermore, this system has its limits, most notably the requirement for a constant Wi-Fi connection in order to conduct real-time monitoring. This system requires routine servicing, such as cleaning up the sensor and refilling the water supply. Despite this drawback, the system offers a fresh and practical solution, meets the needs of those who love houseplants, and improves efficiency and convenience without sacrificing plant health.

Overall, there have been major developments in IoT-based plant care, especially with regard to automation and remote control, according to the analysis of current systems. Problems, including reliance on the internet, power consumption, and lack of personalization, suggest there is room for development. The proposed system strives to address these issues by designing it to be efficient, user-friendly, and robust enough to maintain indoor plants. It aims to meet the needs of indoor gardening enthusiasts with improved usability and dependability through the intelligent integration of advanced Internet of Things (IoT) technology.

## 2.6 Summary

In summary, the literature review lays out the pros and cons of prior IoT-based plant care systems, which forms the basis of the suggested solution, an IoT-based indoor plant care system. Using IoT platforms, existing systems may automate critical activities like watering and lighting, enabling remote monitoring. To further save costs, they make use of Arduino-compatible components and low-cost technology like NodeMCU. Nevertheless, it's important to consider a number of drawbacks, such as the fact that they depend on reliable internet and electricity, that they require technical knowledge, that inexpensive sensors aren't always accurate, and that they are not tailor-made to meet the needs of plants.

In order to address these issues, the proposed system incorporates modern hardware such as the ESP32 microprocessor, permits operation even when without power source, provides a database of plant-specific characteristics, and has durable sensors and backup power alternatives. A simplified user interface is also given priority to make it more accessible. By fixing the problems with current systems and making better use of their strengths, the proposed system hopes to give a trustworthy, efficient, and convenient solution for indoor plant care. The varied requirements of gardeners are its target audience.

## **CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN**

### **3.1 Introduction**

The IoT-based indoor plant care system utilised the IoT technology to alleviate indoor plant care among busy individuals or those with limited knowledge about plant care, as it involves the use of ESP32 microcontrollers and sensors like soil moisture, temperature, and light sensors. It has an automated and a manual watering system, as well as a light system. It was also connected to a mobile application that facilitated real-time monitoring. By addressing issues like uneven care and inexperience, this cost-effective solution promotes healthier indoor plants while guaranteeing optimal plant health with little user intervention.

This chapter focuses on the requirement analysis, system design, implementation, and evaluation of the proposed system. Section 3.1 delves into the methodology, while section 3.2 delves into the planning phase to gain a deeper understanding of user requirements for mobile application UI design, survey results, and the system's software and hardware needs. Furthermore, in section 3.3, there is a detailed explanation of the project design of IoT-based Indoor Plant Care, while sections 3.4 and 3.5 will discuss the development and test phases, section 3.6 will discuss the deployment phase, section 3.7 will discuss the system review, and section 3.8 will be about the summary of this chapter.

### **3.2 Agile Methodologies**

For this project Agile methodology is being used. According to Geeksforgeeks (2023), the agile methodology is an adaptable and iterative approach to software development and project management. With the strong emphasis on teamwork, flexibility, and constant development, it helps to achieve superior results. Agile breaks big projects into small and manageable pieces referred to as 'sprints' or 'iterations'. This method guarantees that the end result successfully achieves its specified objectives while encouraging adaptability to

adjust to alterations and sustain consistent advancement. Prioritising the project's goals, accepting change, and producing results quickly are all important aspects.

### 3.3 Planning

The planning phase lays the foundation for the project, defining its objectives, scope, and deliverables. This phase ensures an organised approach to execution by defining tasks, determining requirements, and establishing schedules. Strategies for effective completion are developed by evaluating potential risks and limits. This stage offers a precise road map for effectively accomplishing the project's objectives. This section will delve into the detailed analysis of the questionnaire input, software, and hardware requirements.

#### 3.3.1 Questionnaire

The aim of the survey is to gather insight on user preferences, behaviours, and expectations related to the UI design and functionality of the IoT-based Indoor Plant Care mobile application. Sections A, B, and C comprise this survey. Section A primarily focuses on the respondent's background and their usage and experience of the system, while Section B concentrates on understanding user preferences, and Section C concentrates on key features and aesthetic preferences. To see the results of the survey and the result of the complete questionnaire, it is listed below.

#### Section A: User System Usage and Experience

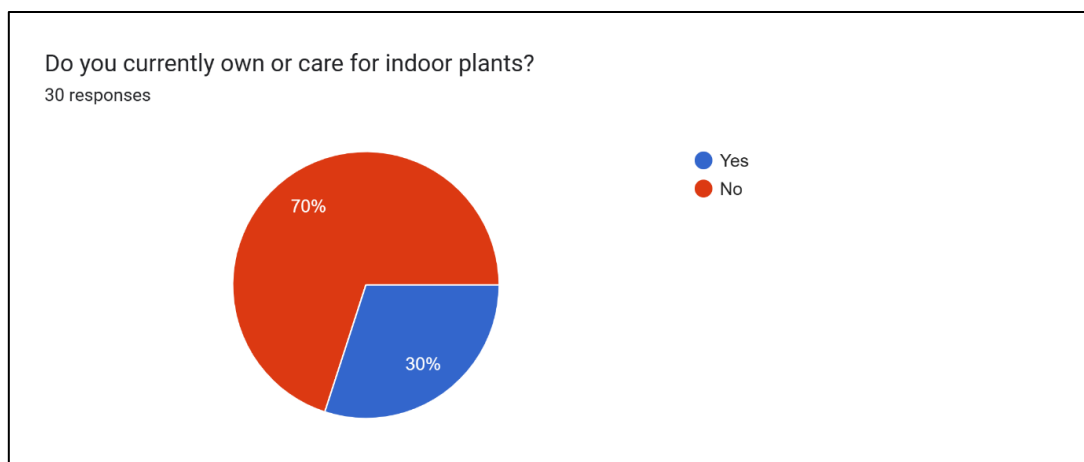


*Figure 3.2* The first question for Section A

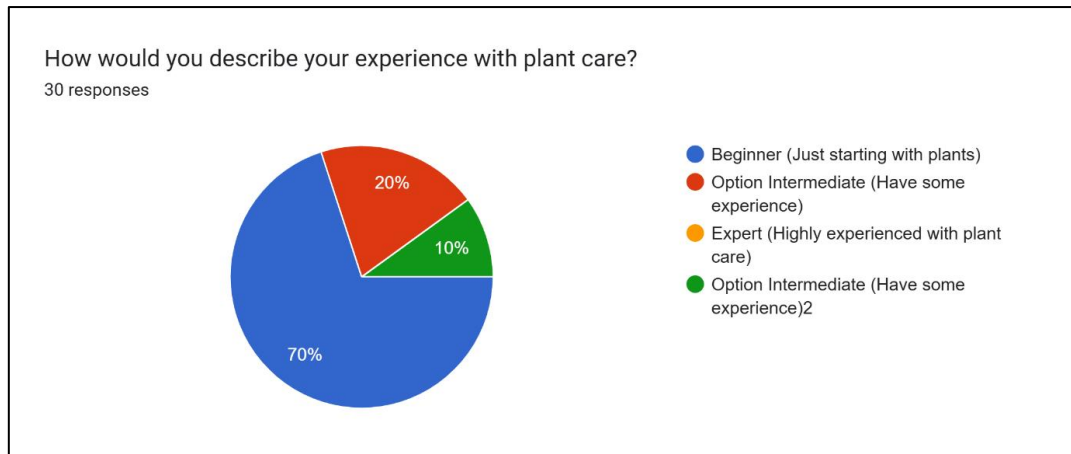


*Figure 3.3* The second question for Section A

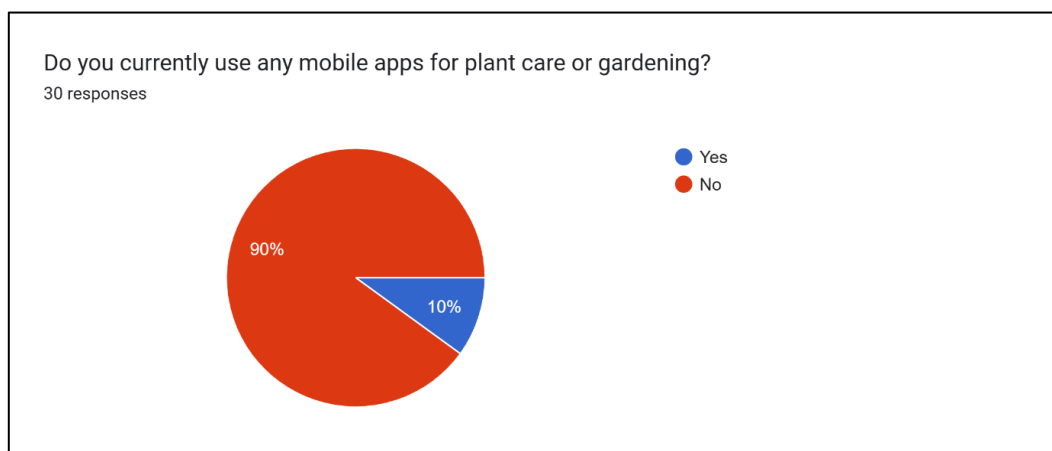


*Figure 3.4* The third question for Section A

Section A focuses on the respondent's background and their experience having indoor plants and using the existing mobile application for plant care and gardening. In these sections, we can analyze the respondent's familiarity with taking care of plants. In figure 3.1, 70% of the respondents did not own an indoor plant, and only 30% owned it. However, 30% of the respondents have some experience in taking care of plants, while the majority, 70%, have little to no experience in plant care, as illustrated in figure 3.2. Furthermore, in figure 3.3, only 10% of the respondents are currently using existing mobile applications for plant care or gardening.

## Section B: Understanding User Preferences

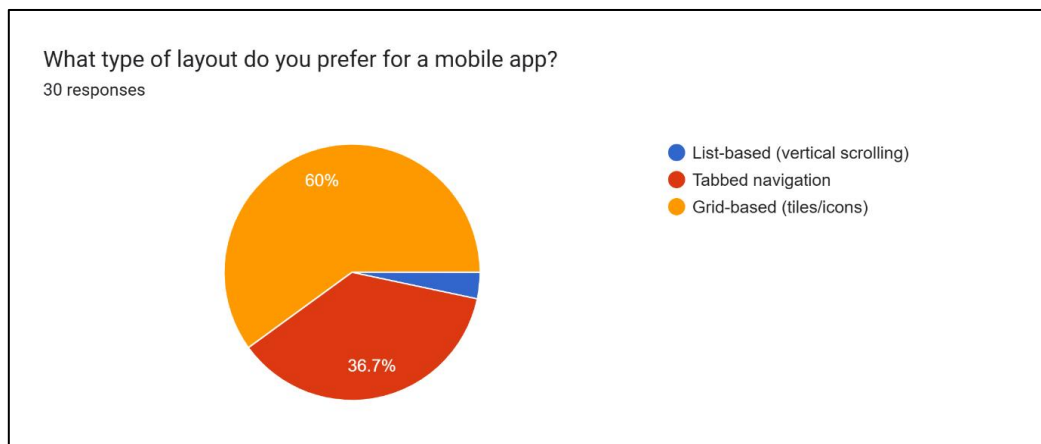


*Figure 3.5* The first question for Section B

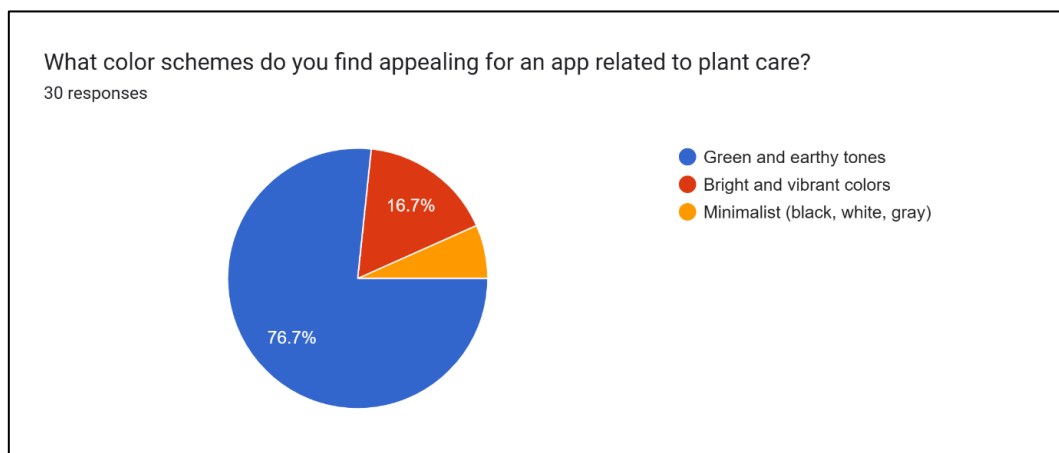


*Figure 3.6* The second question for Section B

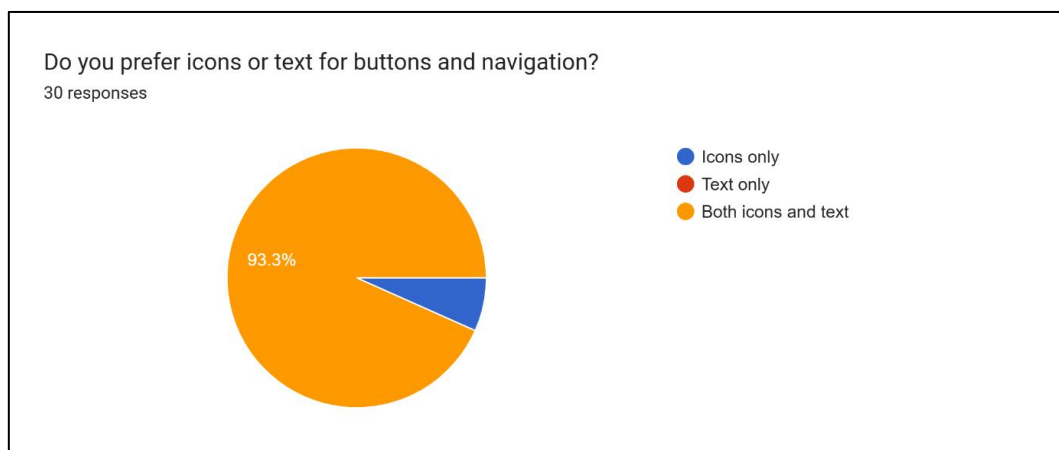


*Figure 3.7* The third question for Section B

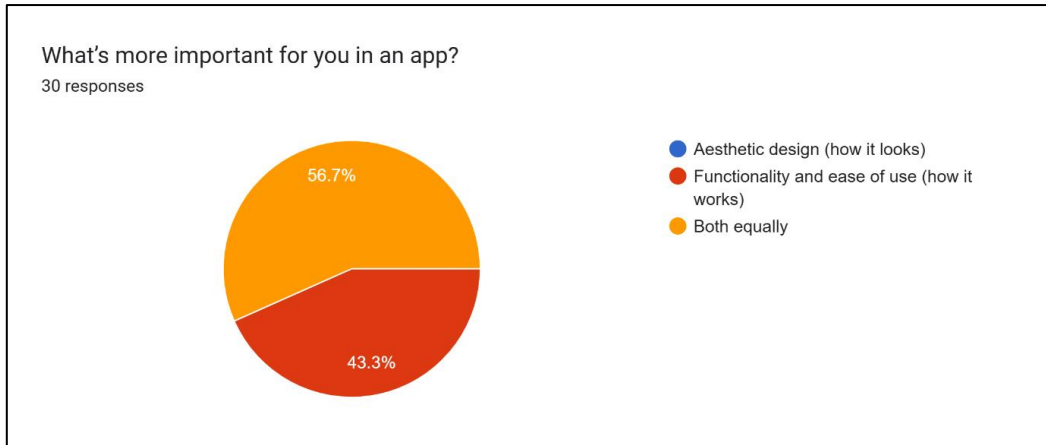


*Figure 3.8* The fourth question for Section B

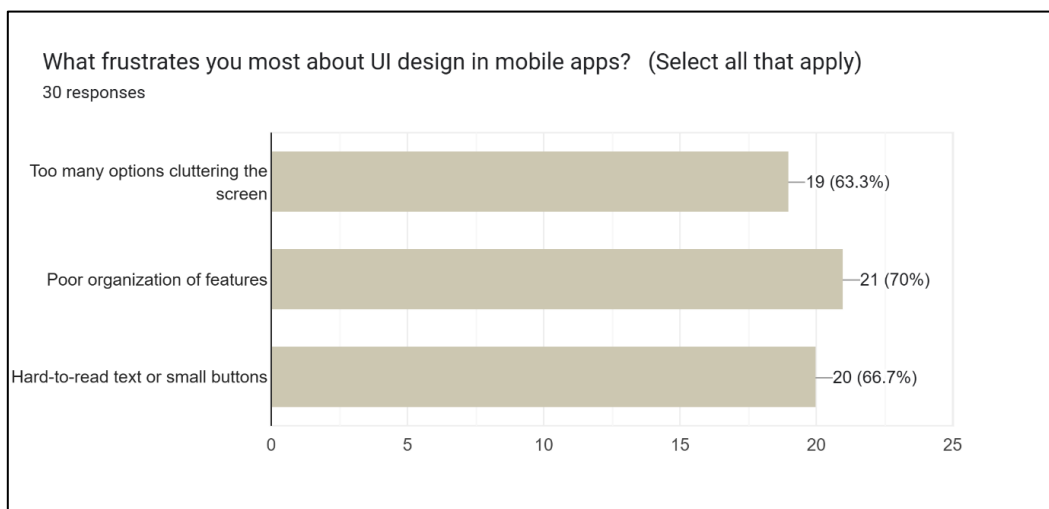


*Figure 3.9* The fifth question for Section B

In Section B, the focus is on gaining a deeper understanding of user preferences, particularly in relation to the UI design of the mobile application. Based on Figure 3.4, most of the respondents, 60% of them, prefer to have a grid-based layout, which means most of the users are more comfortable using a grid layout, which for them, it is easy to navigate through the app. When it comes to the colour scheme of a plant care mobile app, the majority of respondents opted for green and earthy tones, with 76.7% stating, as illustrated in Figure 3.5, that this colour scheme is more appropriate for plant care mobile applications. Based on Figure 3.6, 93.3% of the respondents preferred both icons and text for buttons and navigations, indicating that incorporating both elements enhances the user-friendliness and reliability of the application. Furthermore, as shown in Figure 3.7,

56.7% of the respondents concur that the app's aesthetic design, functionality, and ease of use are crucial factors that can enhance user engagement by making the app both enjoyable and helpful to use. Lastly, in Figure 3.8, most of the respondents stated that poor organization of features and hard-to-read text or small buttons made them frustrated about the UI design of mobile applications.

### Section C: Key Features and Aesthetic Preferences

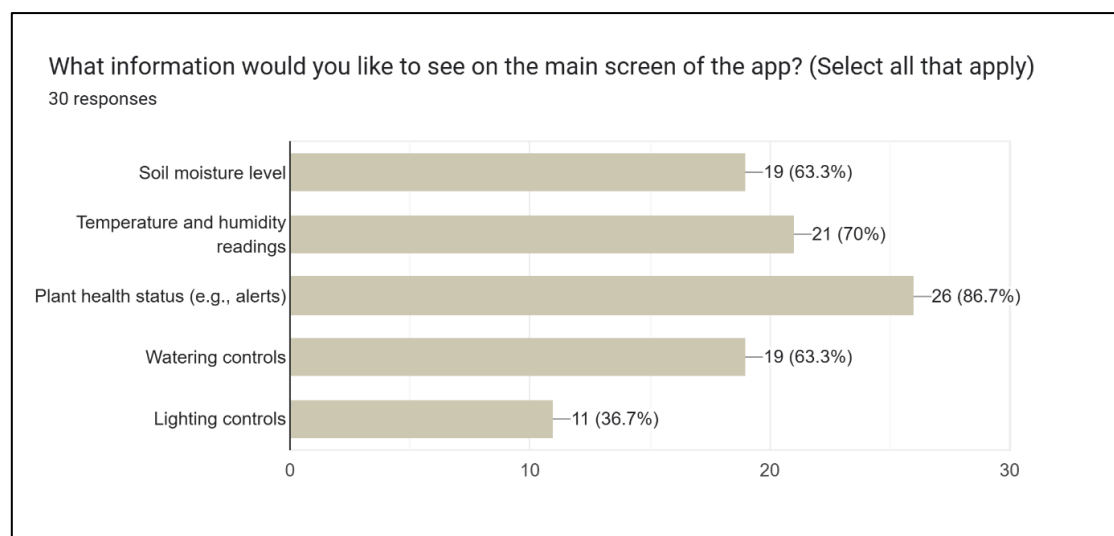


*Figure 3.10* The first question for Section C

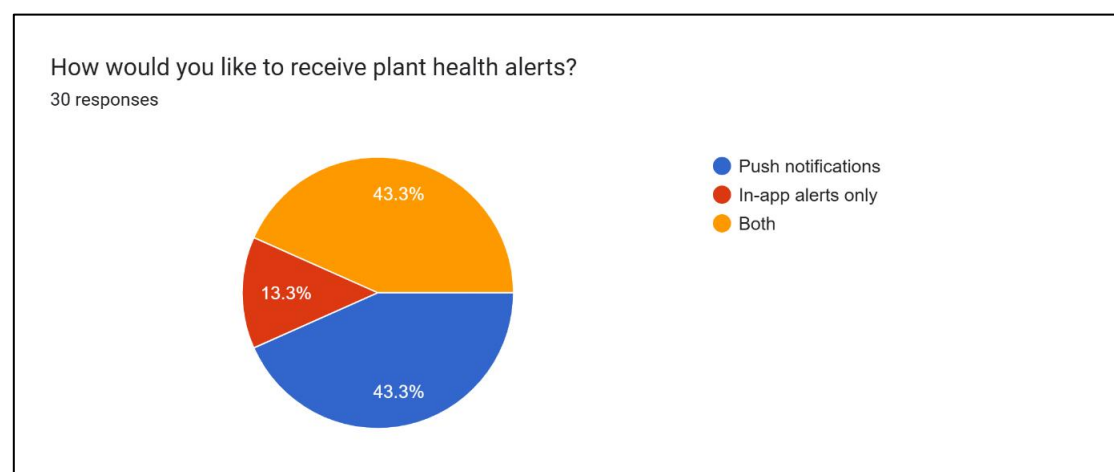


*Figure 3.11* The second question for Section C

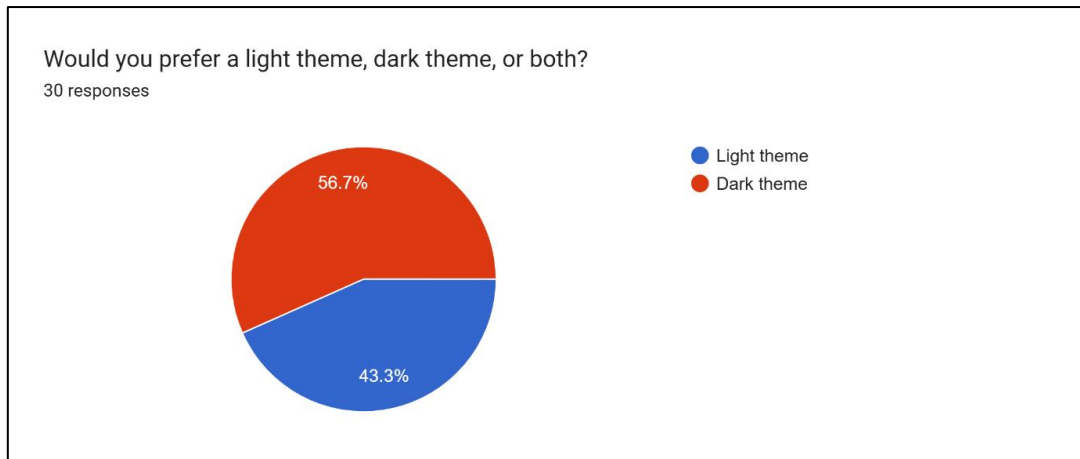


*Figure 3.12* The third question for Section C

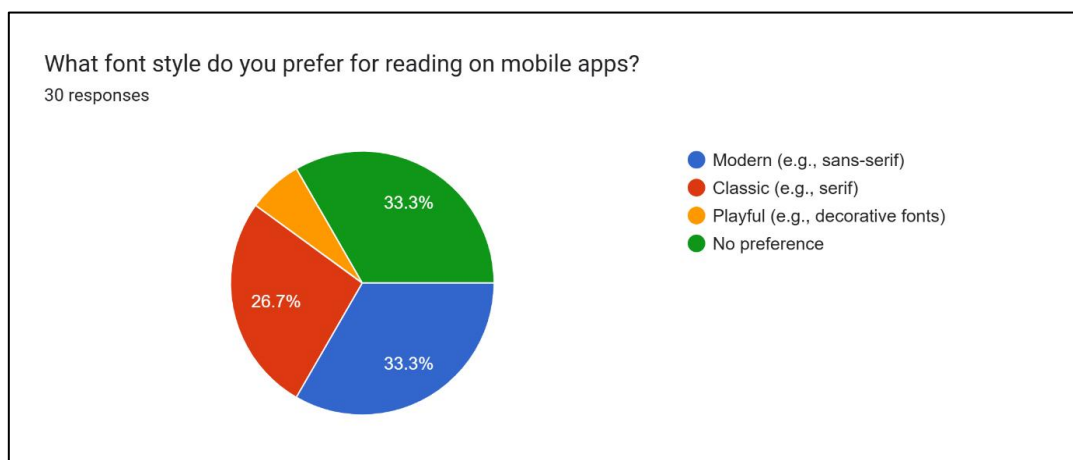


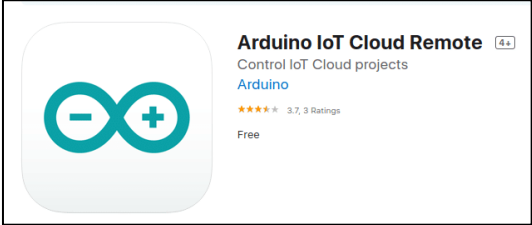
*Figure 3.13* The fourth question for Section C

In this section, it focused more on key features and aesthetic preferences. Based on Figure 3.9, 86.7% of the respondents prefer to have plant health status on the main screen of the mobile application. In addition, Figure 3.10 shows that 43.3% of respondents prefer push notifications, and another 43.3% prefer both push notifications and in-app alerts for receiving plant health alerts. In Figure 3.11, 56.7% of respondents prefer dark themes as they promote eye comfort. Finally, when it comes to font preference, Figure 3.12 shows that 33.3% of respondents have no preference for any particular font style.

### 3.3.2 Software Requirements

The software used throughout the process of development of the proposed system is summarised as in Table 3.1 below.

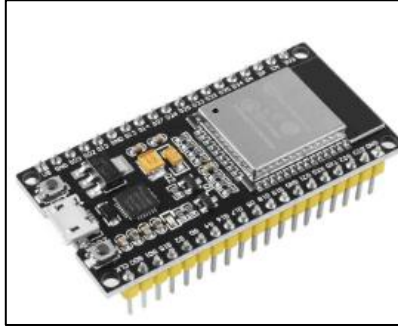
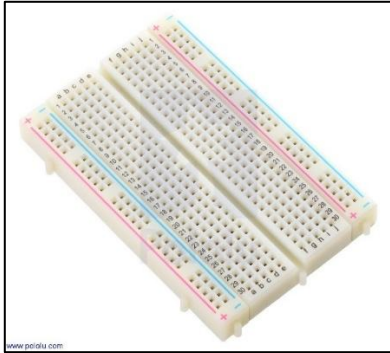
*Table 3.1* Overview of the software requirements

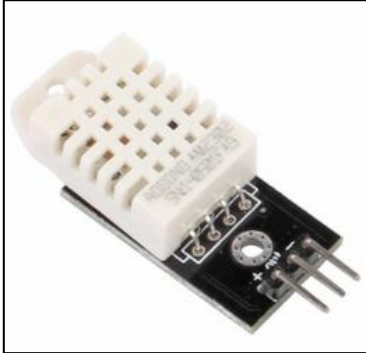
| No. | Software                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | <b>Arduino IoT Remote</b> | <p>Arduino IoT Remote is an application developed for mobile devices that allows the user to watch and manage Arduino IoT Cloud devices remotely. The sleek dashboard presents real-time data and allows manual control, making it suitable for controlling IoT systems in any location..</p>  <p><i>Figure 3.14</i> Arduino IoT Remote</p>                                                                                                                                                                                                                                                    |
| 2   | <b>Arduino IoT Cloud</b>  | <p>Arduino IoT Cloud is a web interface that enables a user to develop, control, and monitor their projects or devices with ease. It can be monitored in real-time via a web dashboard of its mobile application called Arduino IoT Remote. The variables can be defined by using this platform and then linked to actual physical sensors and actuators. Such variables will automatically synchronise on the cloud and the device, which can allow the streaming visualisation of data and its control without the need for complex codes.</p>  <p><i>Figure 3.15</i> Arduino IoT Cloud</p> |

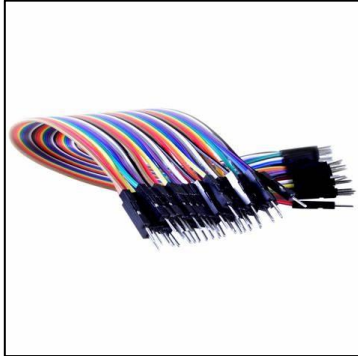
### 3.3.3 Hardware Requirements

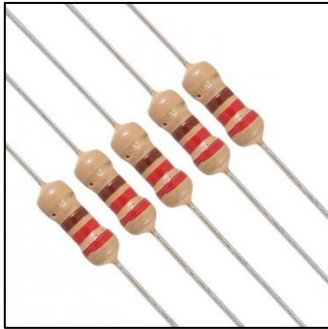
The hardware used throughout the process of development of the proposed system is summarised as in Table 3.2 below.

*Table 3.2 Overview of the hardware requirements*

| No. | Hardware                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | <b>ESP32</b>                | <p>The ESP32 is a strong and adaptable microcontroller with Bluetooth and Wi-Fi integrated into it. It is in charge of analysing sensor data, managing the water pump, LED light and relay, and interacting with the mobile app. Since a large number of sensors and other devices must be integrated into this system, two ESP32 is used for this project.</p>  <p><i>Figure 3.16 ESP32</i></p> |
| 2   | <b>Breadboard</b>           | <p>As it enables connections between components without the need for soldering, a breadboard is an excellent tool for prototyping. Connecting the components rapidly is helpful for initial testing and debugging.</p>  <p><i>Figure 3.17 Breadboard</i></p>                                                                                                                                    |
| 3   | <b>Soil Moisture Sensor</b> | <p>The system will be able to detect when the plants need to be watered because it will have a soil moisture sensor, which will determine the amount of water present in the soil. A capacitive soil moisture sensor will not form any corrosion later due to the fact that it protects its electrodes against elements, and no electrical flow can pass between them (ESP32 - Soil Moisture Sensor, n.d.).</p>                                                                     |

|   |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                |
|---|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
|   |                            |  <p>A black PCB sensor with a yellow, red, and blue cable. The text on the PCB reads 'Capacitive Soil Moisture Sensor v1.2' and 'www.duino.lk'.</p>                                                                                                                                                                                                                               | <p><i>Figure 3.18 Soil Moisture Sensor</i></p> |
| 4 | <b>DHT22 Module</b>        | <p>A digital sensor that measures temperature is the DHT22 module. It has a resistor built in. It is crucial for monitoring the local environmental conditions surrounding the plant and ensuring they stay at optimal levels.</p>  <p>A white plastic sensor housing mounted on a black PCB with three pins. The PCB has '48834U' printed on it.</p>                            | <p><i>Figure 3.19 DHT22 Module</i></p>         |
| 5 | <b>5V Relay Module</b>     | <p>Using a low-voltage microcontroller like an ESP32, the single Relay Module controls high-voltage electrical devices. For this system, both the water pump requires a relay module.</p>  <p>A blue relay module with a blue terminal block and three pins. The text on the relay includes 'SONGLE', '10A 250VAC 18A 125VAC', '10A 30VDC 18A 28VDC', and 'SRD-05VDC-SL-C'.</p> | <p><i>Figure 3.20 5V Relay Module</i></p>      |
| 6 | <b>Water Pump and hose</b> | <p>The water pump will directly water the plants when the amount of moisture in the soil is below a certain specified figure. The ESP32 will use the sensor readings to activate the relay, which turns the pump on or off.</p>                                                                                                                                                                                                                                     |                                                |

|   |                     |                                                                                                                                                                                                                                                                                                                                                                  |
|---|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |                     |  <p><i>Figure 3.21 Water Pump and hose</i></p>                                                                                                                                                                                                                                 |
| 7 | <b>Jumper wires</b> | <p>Jumper wires will be used to complete the connection between the parts, including connecting the ESP32 to the breadboard, sensors, and actuators.</p>  <p><i>Figure 3.22 Jumper Wires</i></p>                                                                               |
| 8 | <b>LED light</b>    | <p>The LED light is used for simulating sunlight, providing artificial light for indoor plants when natural light is insufficient. The LED light will automatically light up when the LDR detects the room in low light conditions.</p>  <p><i>Figure 3.23 LED light</i></p> |
| 9 | <b>LDR Module</b>   | <p>The LDR sensor measures the surrounding light intensity. When the light is faint, it will assist the system in determining whether the room is dark and will cause the LED light to switch on automatically.</p>                                                                                                                                              |

|    |                                         |                                                                                                                                                                                                                                                                                                                    |
|----|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    |                                         |  <p><i>Figure 3.24 LDR Module</i></p>                                                                                                                                                                                            |
| 10 | <b>Resistor 220 <math>\Omega</math></b> | <p>A resistor is needed for an LED light, as it controls the current flowing through the LED. Thus, it gives a safe current level for the LED, as the LED is very sensitive to current.</p>  <p><i>Figure 3.24 Resistor</i></p> |

### 3.4 Design

The design phase involves creating detailed plans and visuals to guide the development of the proposed system. This section will describe the use cases, detailing how the user interacts with the system to achieve specific goals. With the use of an activity diagram, how the system's processes and decisions are structured and prioritised can be seen, which can greatly improve the understanding of how the system works. This section will also offer a thorough overview of the system's architecture, encompassing the hardware and software components, their interactions, and the design of the user interface, which is derived from the previous survey and establishes the system's overall structure.

### 3.4.1 Use Case

According to Figma (2023), a use case describes the steps that users take while interacting with a system or product in order to accomplish a specific goal. System activities, user interactions, and the conditions under which system actions occur are all identified. This section explains the system's behaviour in various situations to ensure it meets all functional requirements. The project aims to automate plant care, enhance user convenience, ensure effective system operation, and design each use case to align with these objectives. Below Figure 3.24 shows the use case of IoT-based Indoor Plant Care System.

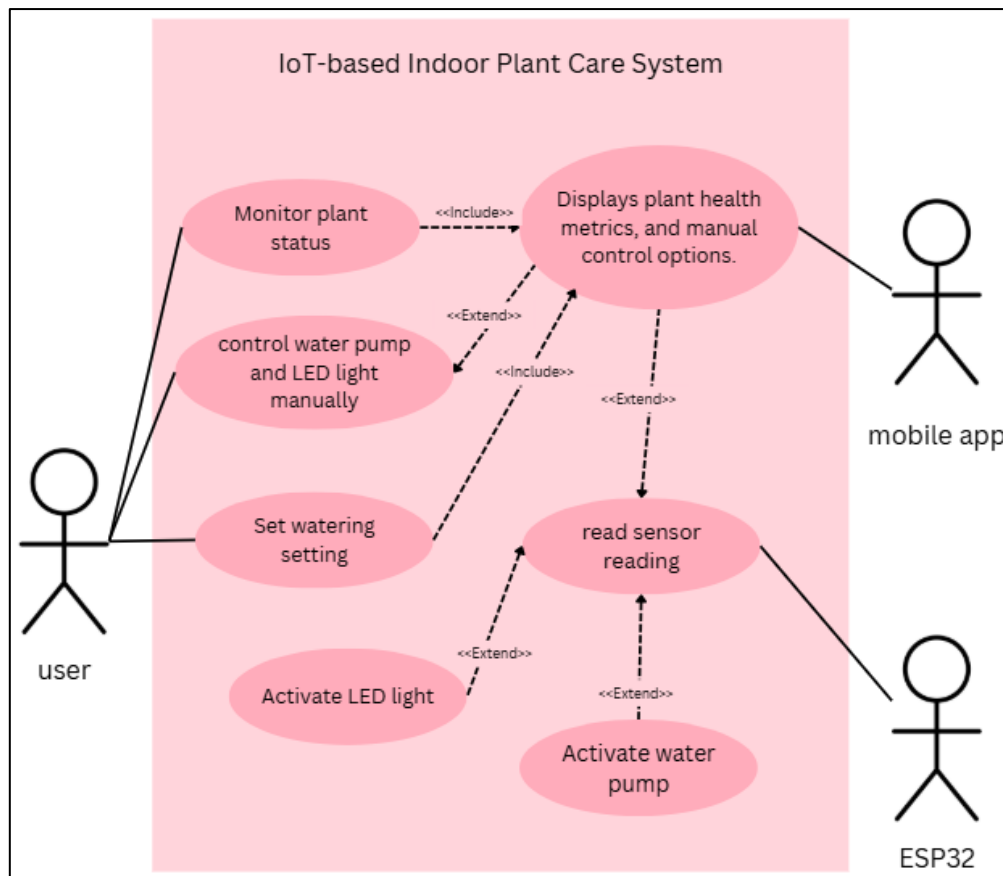


*Figure 3.25* IoT-based Indoor Plant Care System Use Case

The system is based on the use of sensors to monitor the temperature, the level of light, and soil moisture using an ESP32 microprocessor, which will ensure plants get optimal growing conditions. Its system gives real-time information with the help of a

mobile application and does not require any manual maintenance, as it has automated functions, including lights and watering. Plant profile-based configuration makes the system functions flexible and easy to operate by adjusting to different plant needs. This project aims to reduce the burden of indoor plant care and also promote healthy growth in plants. The details of the use case are available in Table 3.3.

Table 3.3 IoT-based Indoor Plant Care System Use Case Details

| No. | Use case                                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----|-----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | Monitor plant status                                      | <p>Actors Involved: User, Mobile App</p> <ul style="list-style-type: none"> <li>The mobile app allows the user to monitor the condition of their plants, such as soil moisture, temperature, and overall health. The user can examine real-time sensor information in this use case to determine the plants' health.</li> </ul>                                                                              |
| 2   | Control Water Pump and LED Light Manually                 | <p>Actors Involved: User, Mobile App, ESP32</p> <ul style="list-style-type: none"> <li>If the user selects the manual setting, they can use the mobile app to manually control the LED light and water pump. They may like to turn on the LED light or water the plants, for example, without depending on automation. To carry out these tasks, the ESP32 receives commands from the mobile app.</li> </ul> |
| 3   | Set Watering Setting                                      | <p>Actors Involved: User, Mobile App</p> <ul style="list-style-type: none"> <li>Selecting the automated watering setting allows the user to customize irrigation preferences according to prefix options via the mobile app. With this setting, the ESP32 uses the sensor readings to decide when to turn on the water pump.</li> </ul>                                                                      |
| 4   | Activate LED Light                                        | <p>Actors Involved: User, ESP32</p> <ul style="list-style-type: none"> <li>The LED light can be manually turned on using a mobile app, or the ESP32 can turn it on automatically when it detects low light levels. This guarantees that there is enough light for photosynthesis in the plants.</li> </ul>                                                                                                   |
| 5   | Displays Plant Health Metrics, and Manual Control Options | <p>Actors Involved: Mobile App</p> <ul style="list-style-type: none"> <li>The mobile app displays notifications, manual control options, and plant health parameters. For a smooth experience, it combines user commands with system data.</li> </ul>                                                                                                                                                        |
| 6   | Read Sensor Reading                                       | <p>Actors Involved: ESP32</p> <ul style="list-style-type: none"> <li>The ESP32 collects data from sensors like soil moisture, temperature, and light levels. These readings are essential for determining the plant's status and triggering automated responses or notifying the user.</li> </ul>                                                                                                            |
| 6   | Activate Water Pump                                       | <p>Actors Involved: ESP32</p> <ul style="list-style-type: none"> <li>When sensor readings show that the soil moisture is below the predetermined threshold, the ESP32 turns on the water pump. This guarantees that the plants receive automatic watering according to the predetermined parameters.</li> </ul>                                                                                              |

### 3.4.2 Activity Diagram

The activity diagram graphically represents the system's workflow and procedures, highlighting the process steps and decision-making points. It clearly sets out the stages for finishing jobs from the user's input to the system's answer. The figure highlights the control flow by demonstrating how the various components interact and make decisions. By outlining these procedures, the diagram guarantees that the logic of the system is understandable and straightforward; hence, it highlights opportunities for workflow enhancement or inefficiencies. Figure 3.26 below displays the activity diagram of the IoT-based Indoor Plant Care System.

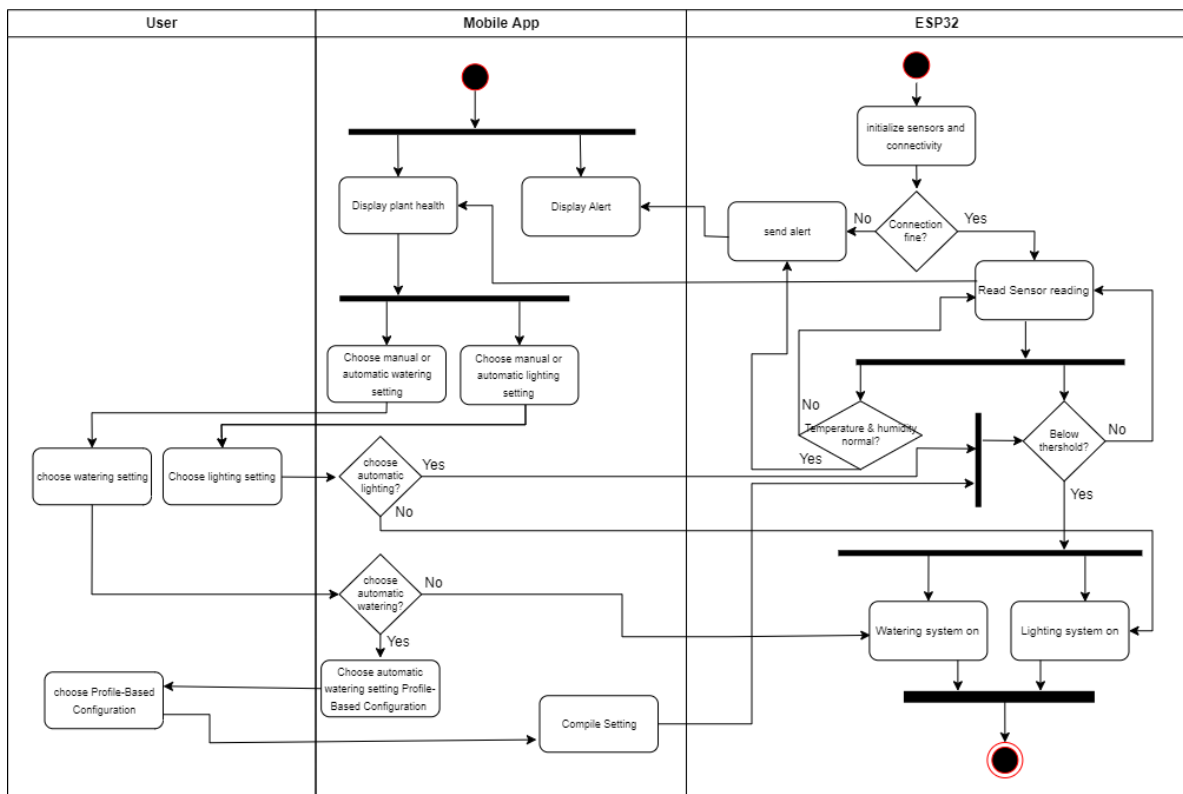


Figure 3.26 IoT-based Indoor Plant Care System Activity Diagram

Figure 3.25, above, shows the activity diagram of the proposed system. This activity diagram is divided into 3 swimlanes, which are user, mobile app, and ESP32. It starts with ESP32 initializing the sensor. The ESP32 reads sensor data, including temperature, humidity, soil moisture, and light, if the connections are fine. In addition to reading the sensor, the ESP32 also transmitted real-time sensor data for display in the mobile app.

Moreover, the user can configure settings for watering and lighting through the app. The options include the manual setting, where the user has direct control over the watering or lighting system, and the automatic setting, where the system automatically manages watering or lighting based on sensor data. If the user chooses the automatic watering setting, they can choose a profile-based configuration that allows customization for specific plant requirements. After user input, the mobile app compiles the settings and sends them to the ESP32 for execution. Based on the settings, the ESP32 activates the watering system if the soil moisture falls below the threshold and activates the lighting system if the lighting condition is dark. The system continues its operations based on the settings until terminated.

### 3.4.3 Layout of Proposed System

This section covers the system architecture, which includes the details of hardware circuit connections and the UI design of the mobile application. These designs were based on the majority opinion of the surveys. Below, Figure 3.26 shows the full layout of the proposed system.

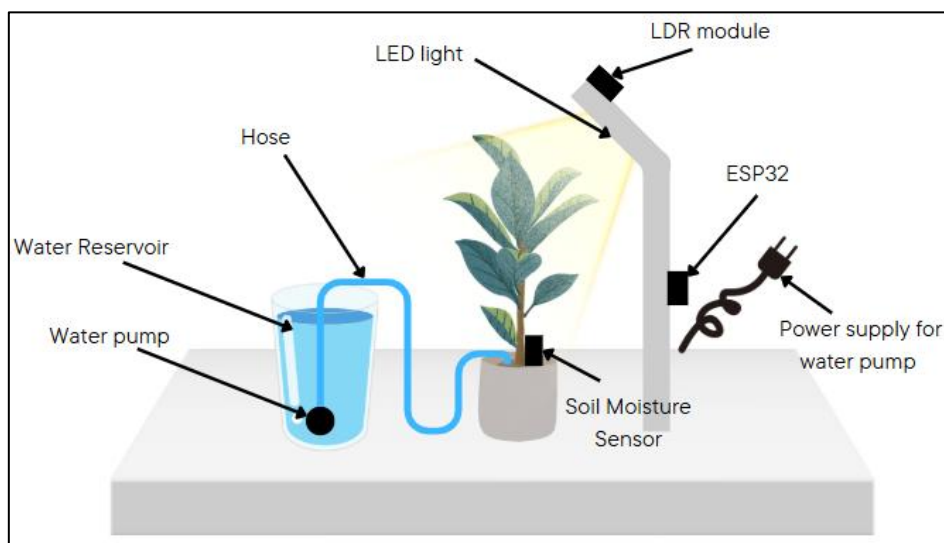


*Figure 3.27* Full layout of proposed system

### 3.4.3.1 System Architecture

Figure 3.27 below illustrates the hardware circuit configuration of the proposed system. We will connect the ESP32 to a laptop to provide power. In this system, two ESP32s will be used to ensure all the sensors or components receive enough power supply. The first ESP32 will be connected with the DHT22 module, soil moisture sensor, and relay module, while the second ESP32 will be connected with an LED, LDR module, and resistor, and both wirelessly connect to Arduino IoT Cloud via Wi-Fi. The water pump requires an external power supply, as their power requirements exceed the capabilities of the ESP32, and they are connected to the relay module, which functions as an electronically operated switch that allows a low-power device such as the ESP32 to operate a high-powered device safely and efficiently.

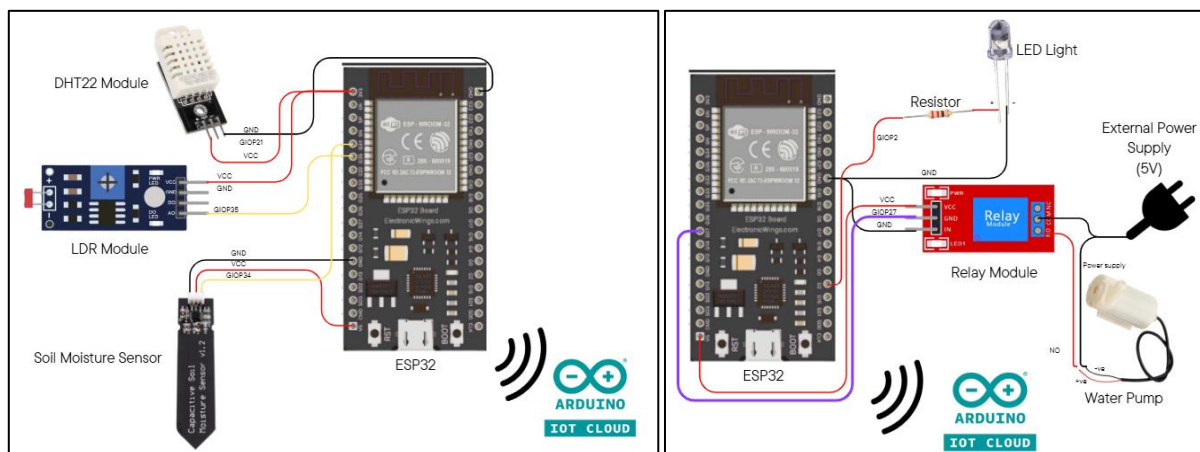


*Figure 3.28* Hardware circuit and software connection of proposed system

### 3.4.3.1 UI Design

The user interface (UI) is made simple, clean and easy to use. It shows real-time information like soil moisture, temperature, and light intensity where one can monitor the status of his or her plants. As much as the system is set to run in automatic mode to control watering by sensor readings, the user can then manually initiate watering at their convenience. The system employs visual indicators and intuitive controls to facilitate effective user engagement in both web and mobile environments. Figure 3.28 and Figure 3.29 below show UI design for proposed system mobile applications.

#### Mobile Applications UI design

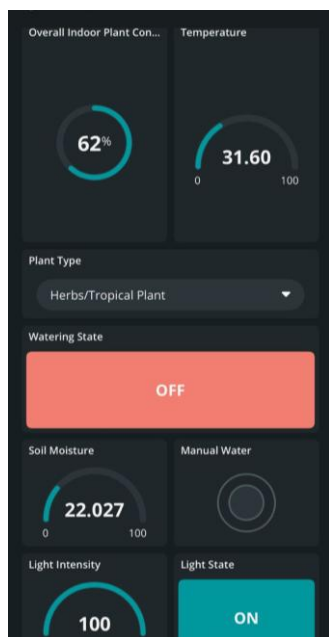


*Figure 3.25* Homepage of mobile application

The homepage offers a clear overview of the current plant status through its simple, clean, and user-friendly design. On the homepage, there are indicators of overall plant condition, temperature, plant type options, watering state, soil moisture indicator, manual watering button, light intensity indicator and LED light state. The user selects the plant type and active watering manually with a dedicated button. Additionally, the light state indicates if the LED light is currently on or off, and the same goes for the watering state; it indicates if the pump is on or off. The design also focuses on the readability and ease of engagement, so the plant tracking can be of use even to users with very little technical knowledge.



When the user clicks the Plant Type option, this will come out, where there are three types of plants with three different settings, which are “Cactus/Succulent”, “Herbs/Tropical Plant”, and “Ferns/Flowering Plants”.

*Figure 3.29 Homepage of plant type*

### 3.5 Development

The development phase focuses on building and implementing the system based on the detailed plans created during the design phase. In this project, the ESP32 microcontroller will be programmed to process sensor data and initiate actions like watering, among other essential IoT gardening system functions. It also entails creating the mobile application interface and integrating hardware elements, such as linking sensors, actuators, and the Wi-Fi module. During this stage, a variety of tools and platforms are used to make if the system is operational and meets project requirements. Debugging and continuous testing are frequently done concurrently to find and fix problems early in the process. The specifics of the development process will shortly be described in Final Year Project 2.

### 3.6 Testing

An essential stage in confirming the system's performance, dependability, and functioning is testing. Final Year Project 2 will be the site of all testing. This involves verifying that every portion of the IoT-base Indoor Plant Care system, such as the water pump, soil

moisture sensor, and mobile app, ensures they function as intended, both separately and in conjunction with the rest of the system. Functional testing ensures the system works as planned for functions like data monitoring, automated irrigation, and notifications. The stress tests were carried out to evaluate the system's behaviour in various environmental circumstances, such as shifting soil moisture levels or varying Wi-Fi connectivity. To guarantee a flawless user experience, this step also entails finding and fixing errors or inefficiencies.

### **3.7 Deployment**

In Final Year Project 2, the completely constructed and tested system will be set up in its intended operational context as part of the deployment phase. Installing the ESP32 system with its linked sensors and actuators in the plant care area, setting up the mobile application for real-time monitoring and control, and guaranteeing dependable Wi-Fi access are all part of this project. To guarantee smooth functioning, we evaluate automation functions like lighting and watering and properly calibrate sensors during this step. In order to make future usage or maintenance easier, deployment also entails writing down user manuals, setup instructions, and troubleshooting advice.

### **3.8 Review**

To assess the project's overall performance and success, the review phase is crucial. This entails evaluating how effectively the IoT gardening assistant achieves its goals, which include preserving optimal plant care conditions and offering a user-friendly interface. Sensor readings and irrigation history are examples of system data that can be studied to find patterns or areas that need improvement. To further improve the system, end-user feedback or deployment observations are gathered. The knowledge gained at this stage is recorded to direct subsequent initiatives and guarantee ongoing enhancements to the system's architecture and operation.

### 3.9 Summary

The IoT-based Indoor Plant Care project automates indoor plant care by combining temperature and moisture sensors with an ESP32 microcontroller system. Plant care tasks like watering and lighting adjustments are automated by the system, which also provides real-time data via a smartphone app. The project adheres to the Agile methodology, which prioritises frequent feedback and iterative development to guarantee a high-quality result.

Every stage of the project was meticulously carried out, starting with thorough planning and design that established use cases, flow diagrams, and system requirements. Coding the essential features, integrating the hardware and sensors, and designing the user interface for mobile access were the main goals of the development phase. To guarantee the dependable functioning of every system component and to detect any problems prior to deployment, extensive testing was carried out.

The system was deployed in its intended location, set up for optimum performance, and tested in real-world scenarios during the deployment phase. The project's overall success was evaluated during the final review phase, which also collected input for possible enhancements and recorded lessons acquired for upcoming initiatives. With the ability to remotely monitor and maintain optimal conditions, this Internet of Things-based gardening assistant provides consumers with a useful, automated solution for indoor plant care.

## **CHAPTER 4: IMPLEMENTATION**

### **4.1 Introduction**

In this chapter, the implementation phase for IoT-based indoor plant care, derived from the requirement analysis and design in Chapter 3, will be underlined using the software and hardware described in the previous chapter. In the development phase, it was broken into several sprints, and each sprint was targeting specific features like sensor integration, automated watering, lighting control, and mobile app connection via Arduino IoT Cloud. It is aimed at designing a working prototype that can allow indoor plant care with minimum human interference and achieve all the goals. Further, under consideration of the fuzzy logic approach to automated watering, system development, connection of circuits and proposed system logic will be considered in detail.

#### **4.1.1 System architecture**

The IoT-based indoor plant care system has been successfully created and installed. The automatic watering system will be triggered by a soil sensor where the fuzzy logic will analyse the soil moisture and temperature, making adaptive decisions to ensure plants receive the optimal amount of water based on real-time conditions. It also analyses the profile-based configuration settings. Figure 4.1 shows the complete proposed system prototype.

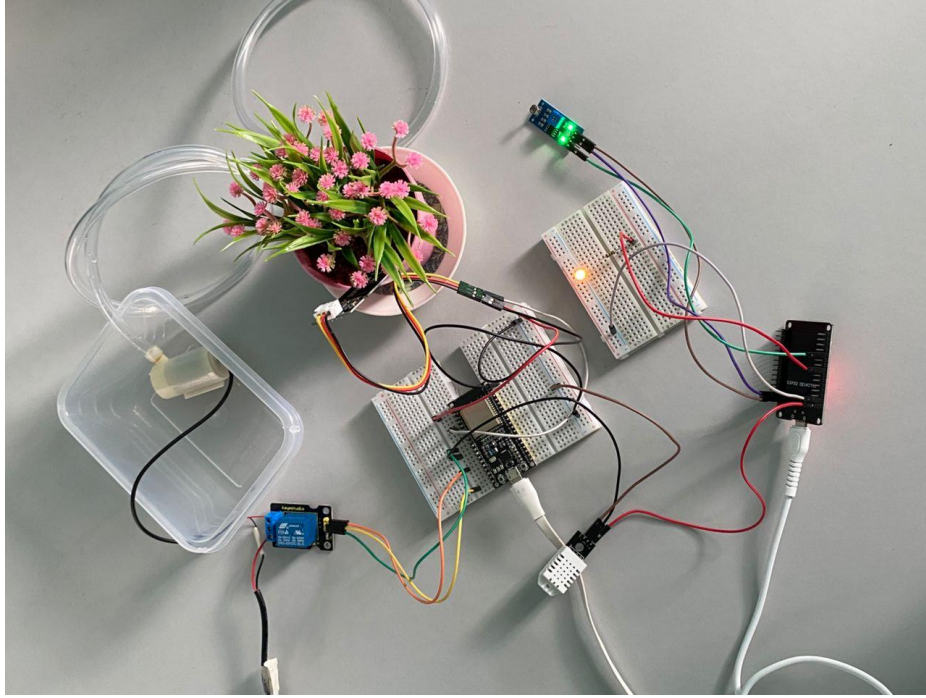


*Figure 4.1* IoT-based Indoor Plant Care Prototype

## **4.2 System Configuration and Setup**

In this part, essential steps for configuration and setup of hardware and software components of the IoT-based indoor plant care will be covered. There are few crucial setups to ensure the system works well. The system components that are used in constructing this proposed system must be properly configured and installed in order for all features of them to operate smoothly.

### **4.2.1 Arduino IoT Cloud and Remote App Configuration**

Arduino IoT Cloud enables ease of communication between the microcontroller ESP32 and the cloud server with the capability to exchange data and perform task automation in real time, based on sensor readings. It also provides the support structure for remote monitoring. The establishment of Arduino IoT Cloud and the remote app is necessary to make sure this system works because it provides a productive and convenient mechanism for controlling plant care automation with minimum effort.

#### 4.2.1.1 Configure Arduino IoT Cloud

The Arduino IoT Cloud can be found at this link: <https://cloud.arduino.cc/>. Before being able to access the Arduino IoT Cloud, an account should be registered. To register, click on the button “Get Started for Free” as shown in Figure 4.2 below.

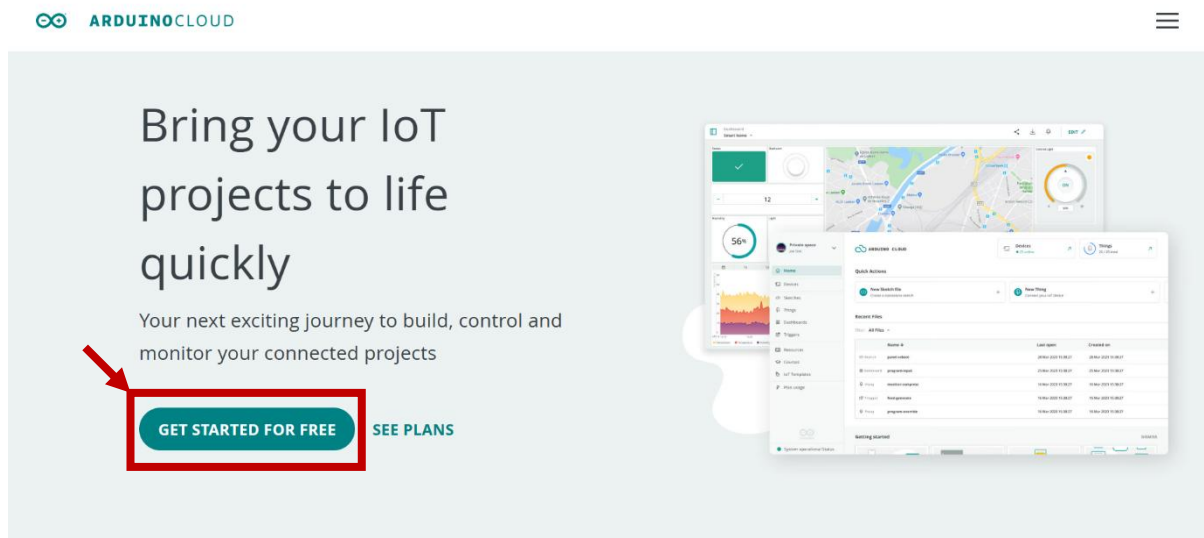


*Figure 4.2 Main page of Arduino IoT Cloud*

*Source: <https://cloud.arduino.cc/>.*

Once an account is successfully registered on the Arduino IoT Cloud platform, a page as shown in Figure 4.3 below will show, where the main dashboard interface will be displayed.

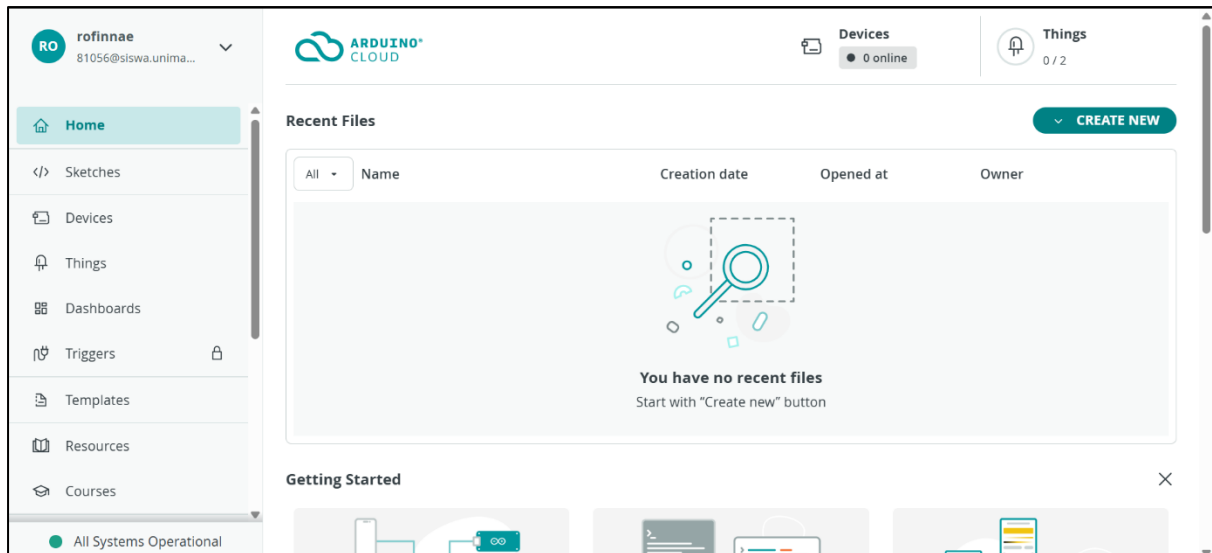


Figure 4.3 Arduino IoT Cloud main dashboard interface

To develop an IoT-based system, go to the left-hand sidebar and choose the “Things” bar. In this part, “Things” were controlled, in which all the devices are connected to them and referred to as “Things”. While on the “Things” interface, click the "Create Thing" button for system configuration as per Figure 4.4.

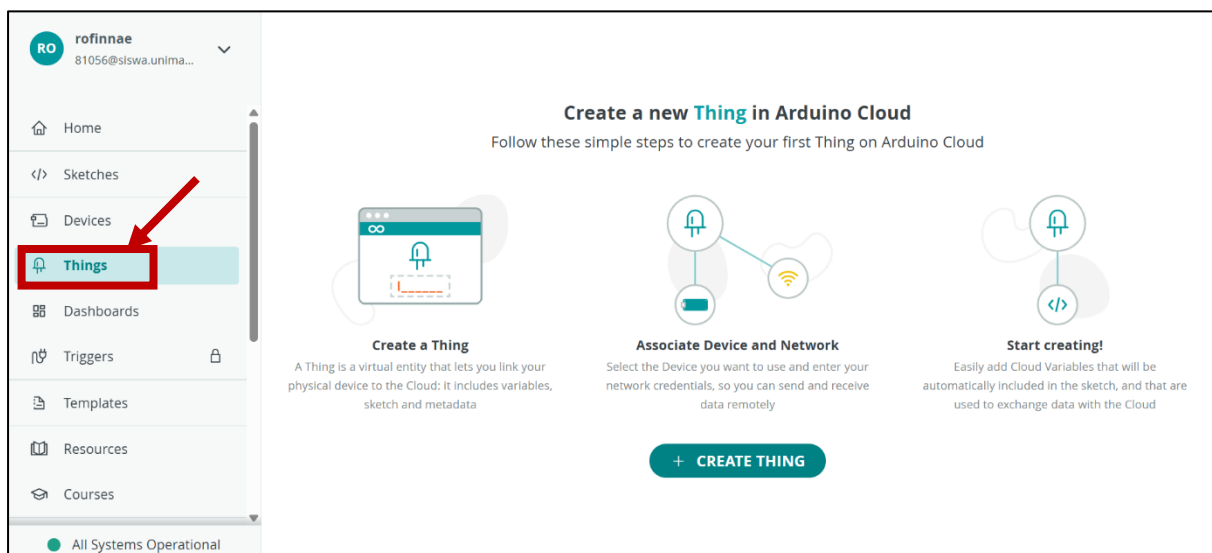


Figure 4.4 “Things” interface

This initiates the setup process where the user can establish a name for the Thing, associate a certain device, for instance, IoT-based Indoor Plant Care, and set the cloud

variables that will be utilised in monitoring and controlling the data, as is demonstrated in Figure 4.5.

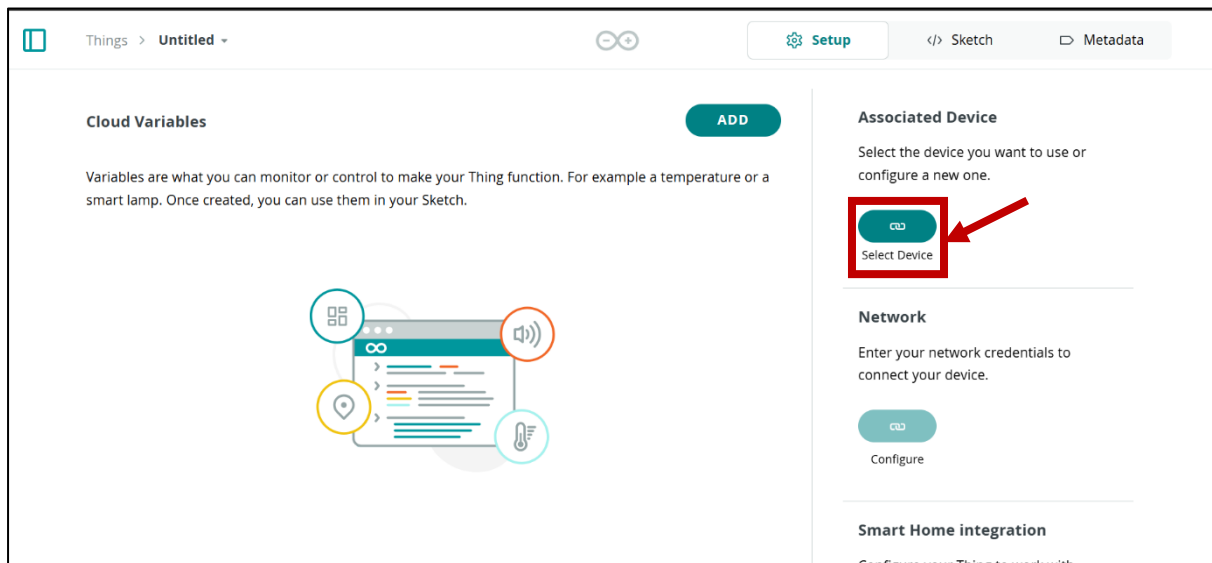


*Figure 4.5 Setup interface*

#### 4.2.1.2 Configure ESP32 Microcontroller

Next, to configure the device that is a microcontroller ESP32, go to the right sidebar and click on “Select Device” in the “Associated Device” section. This step lets the user select an existing device, which belonged to the previous entries, or register a new device. The device implies a microcontroller, as represented in Figure 4.5. To set up a new device, select “Set Up New Device “ as in Figure 4.6.

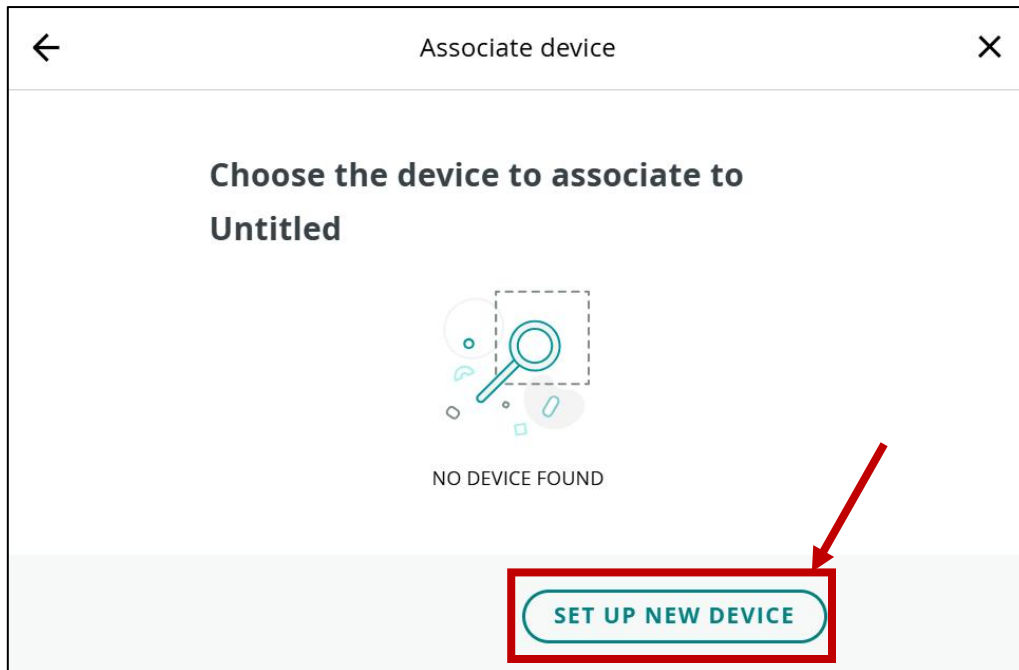


Figure 4.6 Associate Device Interface

As the microcontroller ESP32 will be used, on device setup select “Third Party Device” as in Figure 4.7.

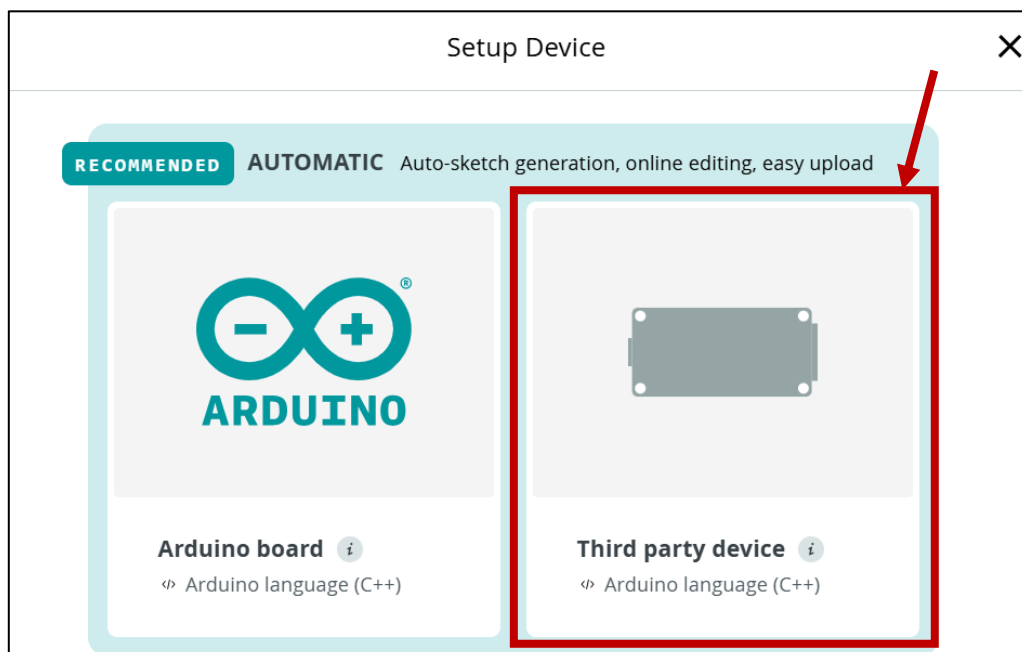
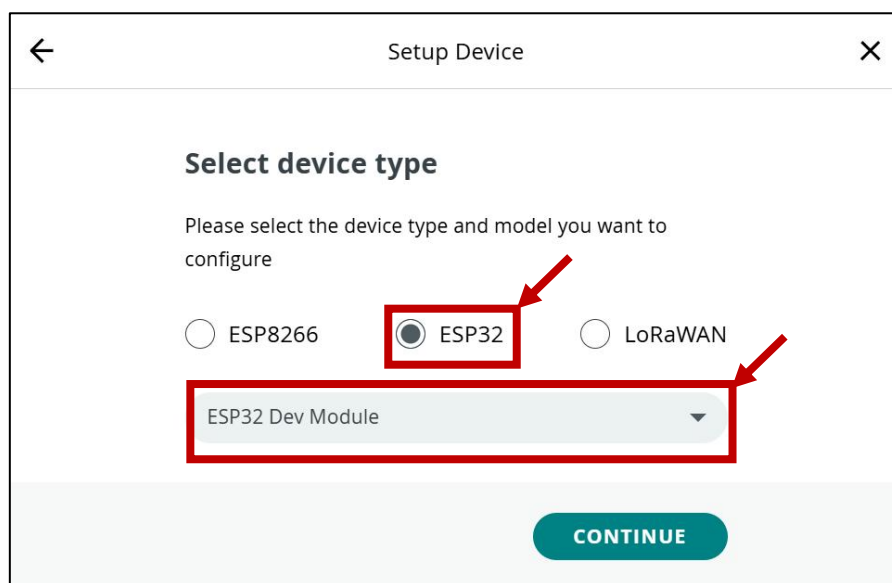


Figure 4.7 Select Third part Device

Second, choose ESP32 for a board family and pick the specific model “ESP32 Dev Module” from the given available models as shown below in figure 4.8. The aim is to make sure that Arduino IoT Cloud will properly identify the microcontroller and be capable of communicating with other microcontrollers. It is very essential to choose the best board type so that the platform can configure the relevant libraries and settings required for uploading sketches, managing the connectivity, and syncing data between the device and the cloud.



The screenshot shows a mobile application interface titled "Setup Device". Under the heading "Select device type", there is a prompt: "Please select the device type and model you want to configure". Three radio button options are listed: "ESP8266", "ESP32", and "LoRaWAN". The "ESP32" option is selected, indicated by a filled circle and a red box with an arrow pointing to it. Below these options is a dropdown menu that currently displays "ESP32 Dev Module", which is also highlighted with a red box and an arrow. At the bottom right of the screen is a green button labeled "CONTINUE".

Figure 4.8 Choose Suitable Model

After the device type and model are chosen correctly, make sure to save the “Device ID” and “Secret Key” provided for the registered device or board, as it will be important for device network configuration as illustrated in Figure 4.9. If “Device ID” and “Secret Key” are lost, there is no way to retrieve them again other than registering a new device.

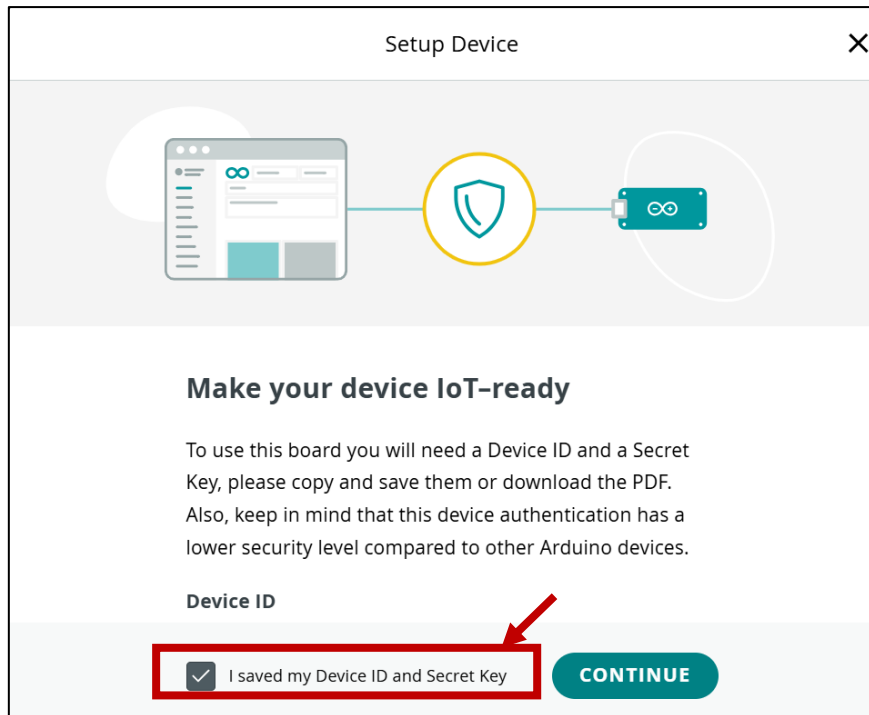


*Figure 4.9* “Device ID” and “Secret Key” file

Once “Device ID” and “Secret Key” have been confirmed saved, make sure to name the device so it can be easier to find, as in Figure 4.10.

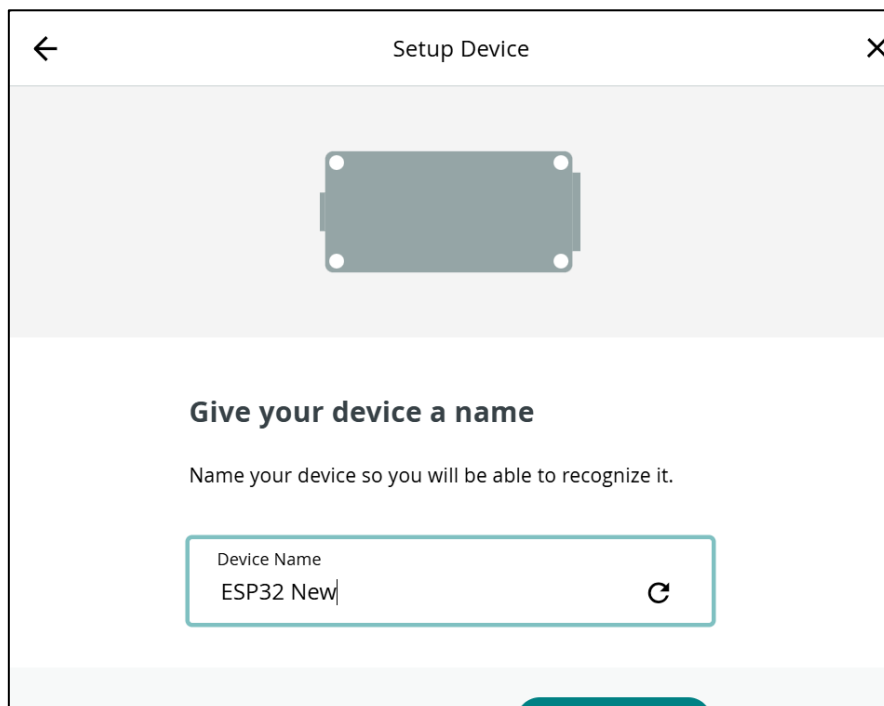


*Figure 4.10* Name the device

### 4.2.1.3 Device Network Configuration

For device network configuration, navigate to the right panel and choose “Configure” in the Network section as shown below in Figure 4.11. This step will configure the device network.

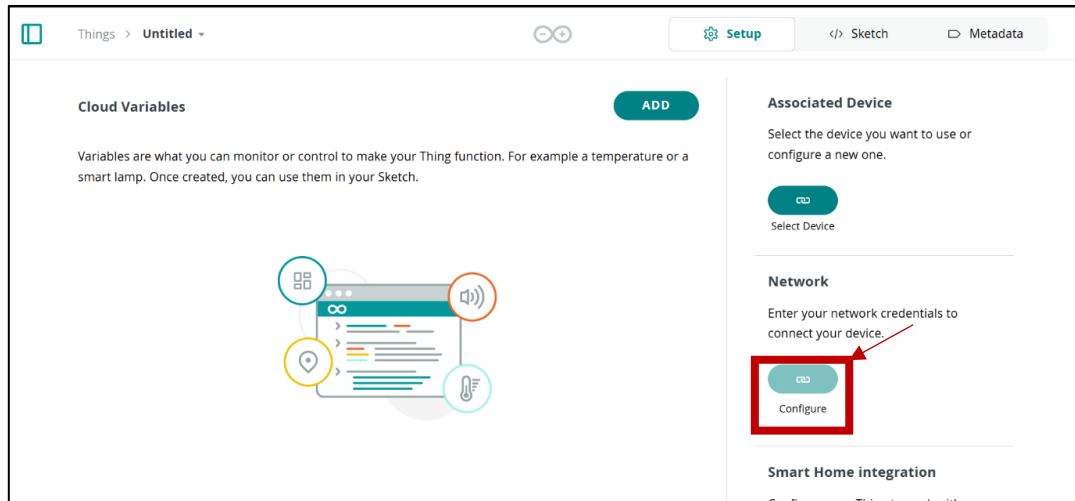


Figure 4.11 On right panel, select “Configure” on network section

During network configuration, make sure to key in the correct Wi-Fi name and password and the secret key provided earlier. Figure 4.12 below shows the credentials that needed for network configuration.

The screenshot shows a 'Configure network' dialog box. It has a title bar with a close button (X). The main text says 'Enter your network credentials to allow your device to connect to the Cloud.' Below this, there are three input fields: 'Wi-Fi Name \*', 'Password' (with an eye icon for toggling visibility), and 'Secret Key \*' (also with an eye icon). At the bottom right, there is a 'SAVE' button.

Figure 4.12 Network configuration details

#### 4.2.1.4 Installation and Configuration of Arduino Cloud Agent

The installation of Arduino Cloud Agent is important, as it will act as a tool that facilitates the communication between Arduino IoT Cloud and connected microcontrollers. This Arduino Cloud Agent allows the web platform to recognise the device when uploading sketches or starting a connection. It can be downloaded from this link: <https://cloud.arduino.cc/download-agent/>, as illustrated in Figure 4.13.

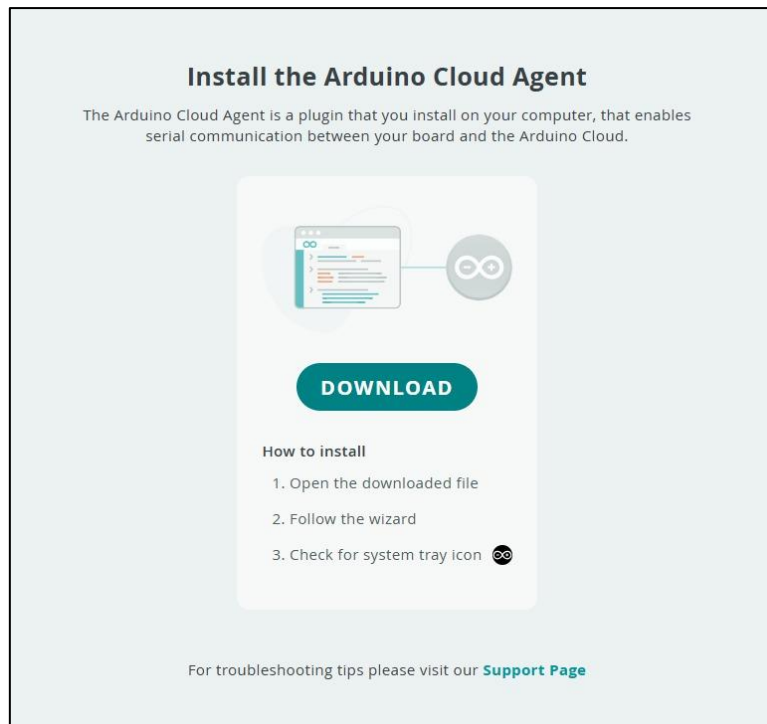
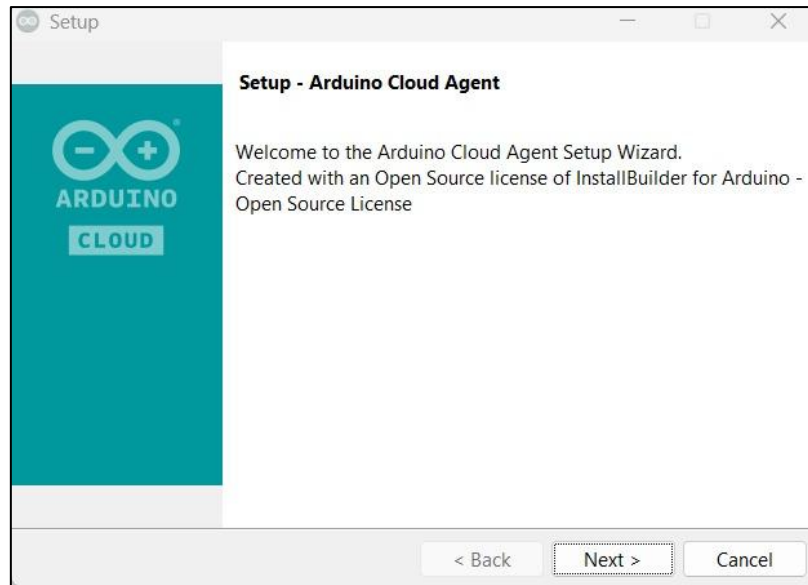


*Figure 4.13* Arduino Cloud Agent download

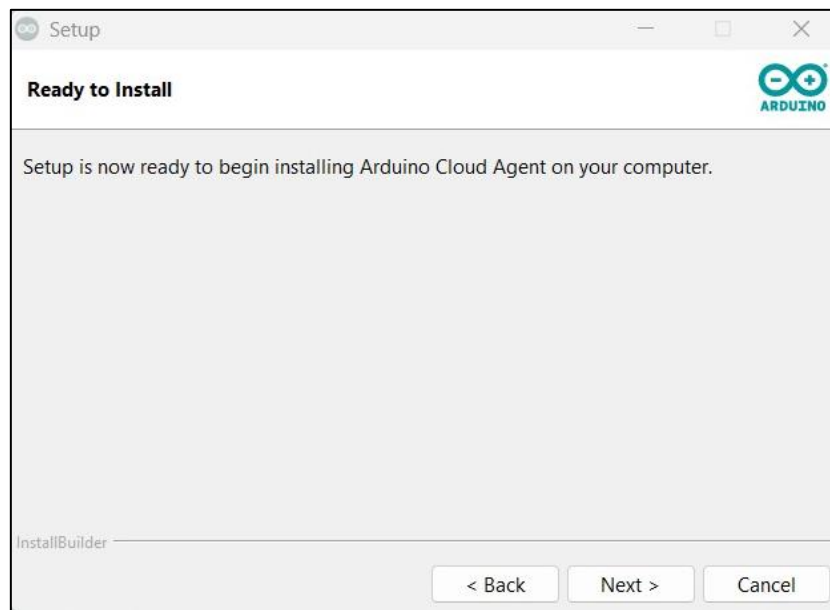
After the download is complete, make sure to launch the file and grant the necessary permissions so that it can be placed on the desktop.

#### Step to configure Arduino Cloud Agent

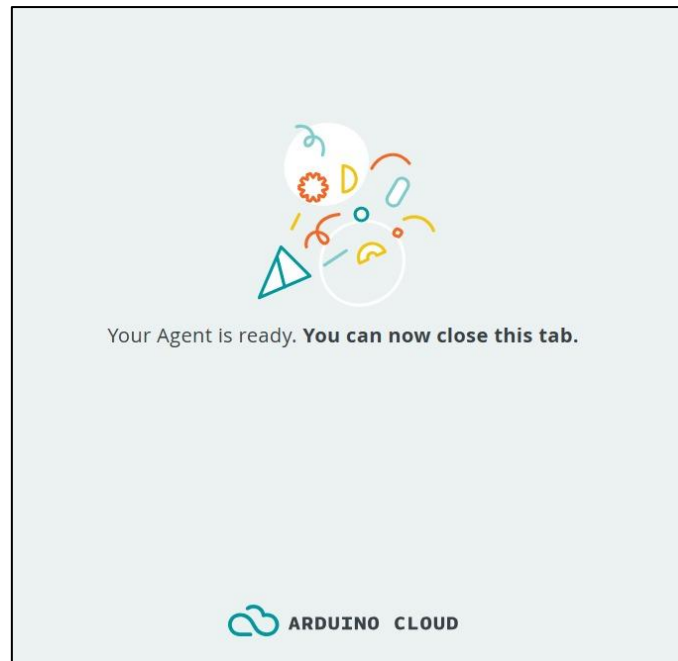
1. Launch Arduino Cloud Agent and allow the permission. Once launched, click next and save it in the proper file.



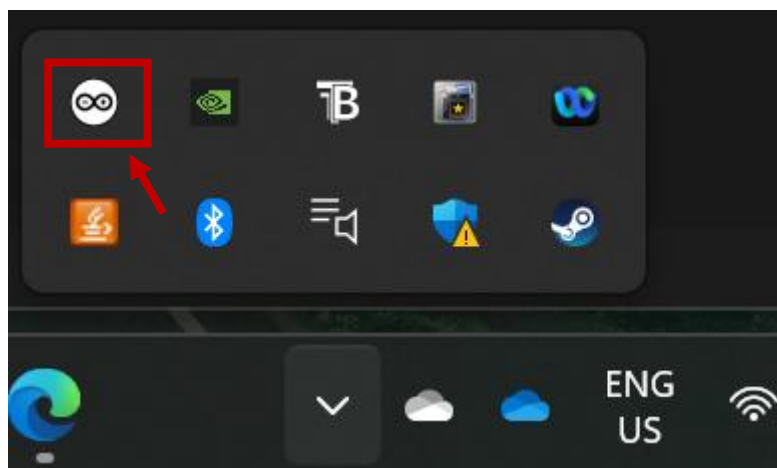
2. Click next to proceed with installation



3. Once the figure as below shows up on the website, it means that Arduino Cloud Agent is successfully installed.



4. To check if the Arduino Cloud Agent is successfully installed and currently running in the background, go to the desktop navigation bar and go to the System Tray overflow menu. If the Arduino Cloud Agent icon is present, it shows it is currently running in the background.



#### 4.2.1.5 Installation and Configuration of Arduino IoT Remote

The Arduino IoT Remote is one such application that permits the users to monitor as well as control the IoT devices connected to the Arduino IoT Cloud from anywhere at any time. With this app, users will be able to work with live data, get notifications, and control system functions right from the smartphone. This application is easily locatable on the

app store for the iOS users and the Google Play Store for the Android gadgets. Just type “Arduino IoT Cloud Remote” in your preferred search engine and download and install this application as displayed in Figure 4.14.

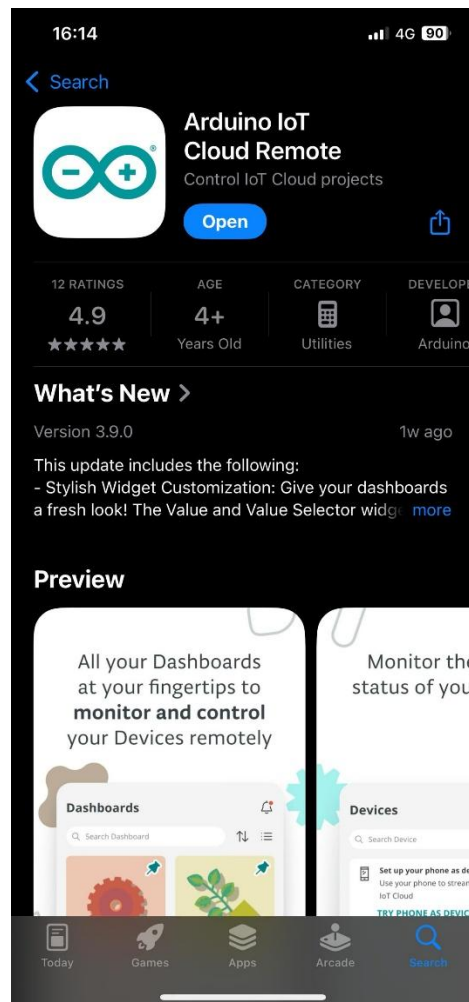


*Figure 4.14* Arduino IoT Cloud mobile application

Then when the downloading and installation process is over, open the application from the mobile device. When the app is launched, an account password and username are needed to log in with the same account as the Arduino IoT Cloud platform. This is a very important step to synchronise mobile applications with the existing cloud-based IoT projects. The app will automatically fetch all the information that is associated with the project as well as the dashboards that are connected to that account for easy remote

monitoring and control. Below, Figure 4.15 is the example of how the main interface of the application is presented.

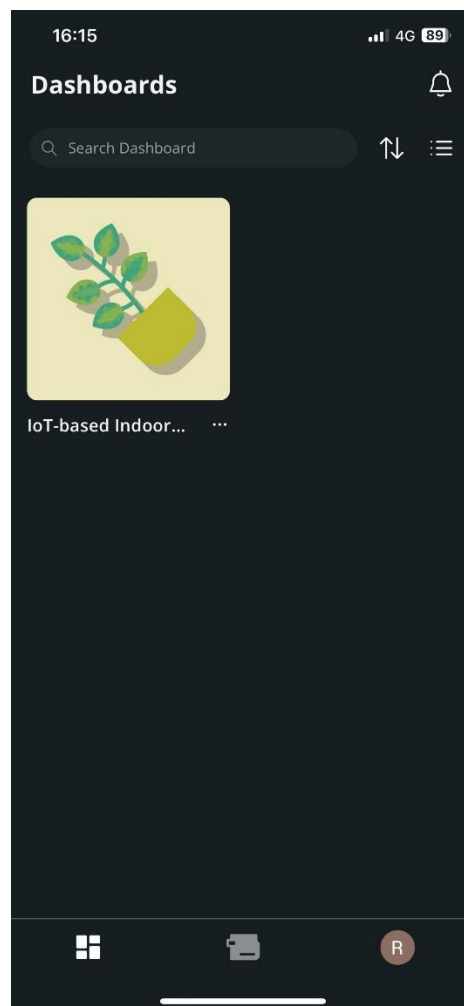


*Figure 4.15* Main interface of the application Arduino IoT Cloud Remote

### 4.3 System Features and Operations

The system is designed with a focus to automate vital watering and lighting systems for indoor plants for the best care. The system that was proposed also included the basic activities such as watering, lighting, controlling and environmental monitoring and gave real-time data to the user. Each feature makes it possible to reduce the elements of manual intervention, promote the plant health, and enhance the level of convenience of the user. This part will touch on the micro details of every feature. The user interface of the system below in Figure 4.16 is given below.

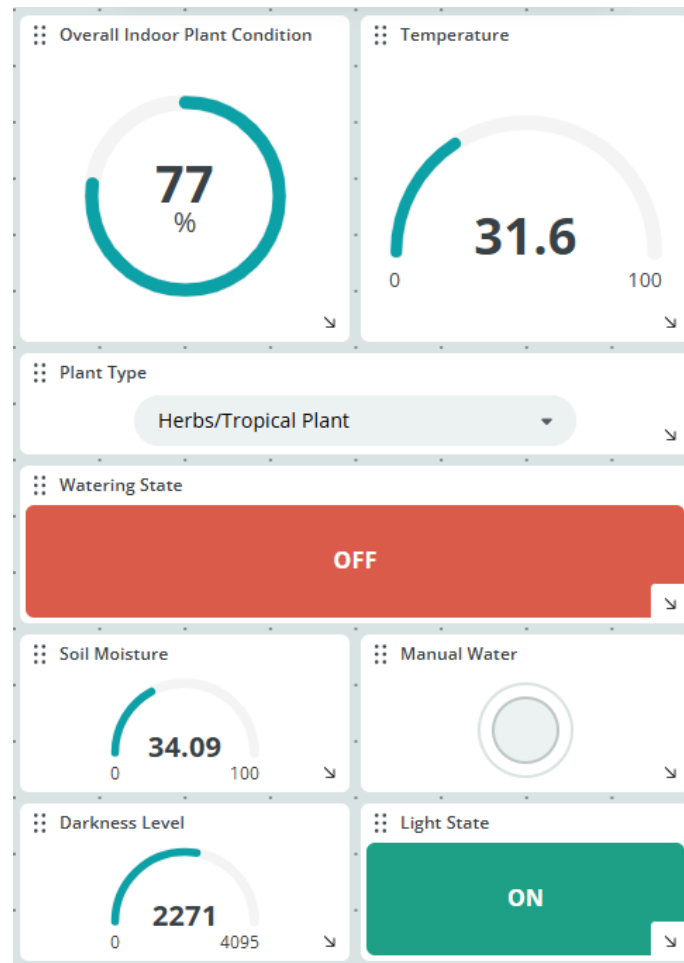
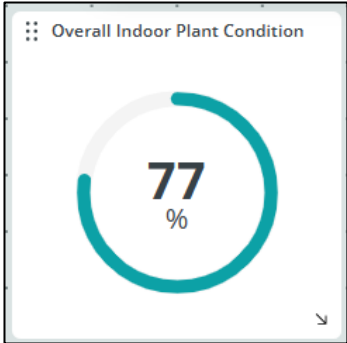
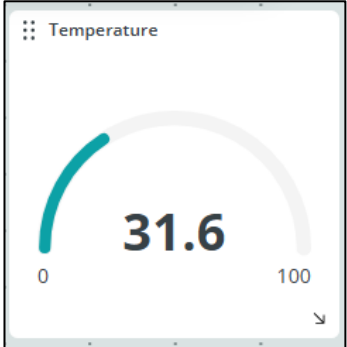
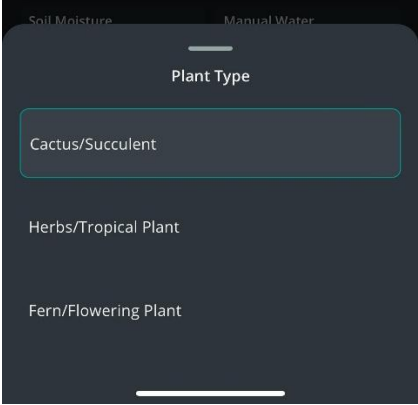
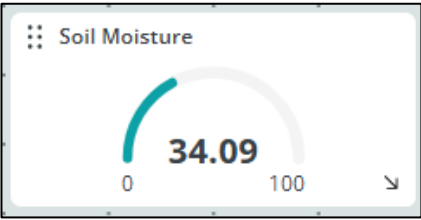


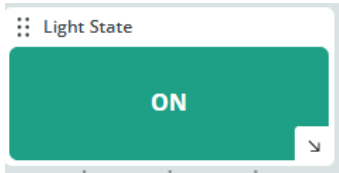
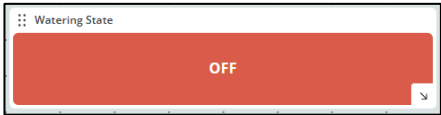
*Figure 4.16* User interface of the IoT-based Indoor Plant Care

The following Table 4.1 is therefore provided for a more detailed run-through on the functionality presented by the Arduino IoT Remote mobile application. All the points from the list are required to provide the opportunity for the users to monitor and control the system of providing care to an indoor plant via the means of a smartphone or a tablet. The following is the table presenting key features and what they are and how they are useful in the aspect of system usability and convenience and real-time plant state monitoring. With the help of these capabilities, the users can still provide the best care to their plants even if they are not themselves there.

Table 4.1 Features of IoT-based Indoor Plant Care Mobile Application

| No. | Features                                                                                                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | <p>Overall Indoor Plant Condition</p>         | <p>Descriptive overall indoor plant conditions will be employed such that the users would tell the overall plant conditions by measuring environmental aspects of the plant, such as soil moisture and surrounding temperature, to ensure that the plant is healthy. The consistent gathering and analysis of information provided by sensors will allow the system to track any undesirable shifts in the environment. Such moves allow the users to take immediate action if needed, particularly if the surrounding temperature is too high. In the end, this aspect really did play a prominent role in ensuring healthy plant growth and allows avoiding harm caused by environmental negligence .</p>                                                                                                                                                                                                                                                                                                                                                                                |
| 2   | <p>Surrounding Temperature</p>               | <p>The plant's ambient temperature is monitored by the use of the surrounding temperature feature. The surrounding temperature should be monitored to make sure it remains in healthy levels, particularly in indoor plants. However, when the temperature in the surroundings is greater than the temperature threshold, which is 35 °C, to prevent the plant from the heat stress. The 35°C is meticulously selected on the basis of 3 plant types throughout the system because it is an average where the plants are cacti or succulents, herbs or tropical plants, and ferns or flowering plants and set as the maximum limit. The tolerance level of a cactus or succulent towards high temperature is up to 49°C (Coral, 2025), whereas herbs or tropical plants can bear up to 30°C (Manzur, n.d.)and ferns or flowering plants can sustain up to 27°C (<i>Ideal Temperature for Plant Growth: Ultimate Guide [Solved]</i>, 2024). The threshold level of 35°C covers most of the plant species without unnecessarily issuing heat stress warnings for the heat-tolerant ones.</p> |
| 3   | <p>Automated watering system initiator</p>  | <p>By choosing a certain type of plant from the "Plant Type" interface, the automatic watering system is adjusted to the unique watering requirements of each planted plant. All the plant types have individual needs. When a user selects a plant type, the system applies the appropriate thresholds for the soil moisture and the logic of watering it</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|   |                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |                                         | <p>automatically. This function guarantees that the plant will get the right amount of water free from the danger of overwatering and underwatering. This aspect not only makes plant maintenance easier for users who do not possess much knowledge of gardening but also encourages the plant to grow healthier as water is applied according to the requirement of the chosen plant profile.</p>                                                                                                                                                                                                                                                                                                                                                                                            |
| 4 | <p><b>Soil Moisture Reading</b></p>     | <p>This feature indicates the amount of water content in soil that reads in real time via a soil moisture sensor that is embedded in the ground of the plant. The sensor continues to detect moisture contents and relay the information to the ESP32 microcontroller, which serves the role of processing and data transfer. The ESP32, with its internal Wi-Fi connectivity, transmits the soil moisture reading to the Arduino IoT Cloud, where the values are displayed in the user display.</p>                                                                                                                                                                                                                                                                                           |
| 5 | <p><b>Manual watering button</b></p>  | <p>This is the Manual Water button, and with it, users are able to manually operate the watering system regardless of the soil moisture readings. This feature is more flexible and allows users to water their plants depending on their observation or specific needs that may not be in the automatic system. For instance, the manual option may be used by the users after applying fertiliser to the soil, on excessively hot days, and when testing system functionality.</p>                                                                                                                                                                                                                                                                                                           |
| 6 | <p><b>Darkness Level reading</b></p>  | <p>The Darkness Level Reading Display feature offers continuous monitoring of the light intensity around so as to ensure proper lighting needed for the healthy growth of indoor plants. The larger the value, the darker the room are. The system continuously determines the intensity of surrounding light in the plant's environment with the use of a Light Dependent Resistor (LDR) sensor. The readings are then sent to the ESP32 microcontroller, which in turn sends the readings to the Arduino IoT Cloud platform through Wi-Fi. The Arduino IoT Remote mobile application, as well as the web interface, acts to show the measured light intensity on the user dashboard in real-time. This provides the users with an opportunity to monitor whether the plant gets adequate</p> |

|   |                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |                                                                                                           | <p>natural or artificial light during the day. Furthermore, this feature is combined with the automated grow light control of the system. When the light level dips beneath a threshold, for example, on cloudy days or at night, the system can automatically turn the LED grow lights ON to increase light and create the appropriate conditions for photosynthesis. This is especially applicable in indoor plant arrangements that are exposed to little sun. By giving continuous feedback and automatic light regulation, this feature guarantees that the plants are not exposed to the long periods of darkness or excessive illumination, which results in the harmonic development of the plants.</p> |
| 7 | <p>LED light state</p>  | <p>The LED light state feature shows the current state of the LED grow light inside the system, where it can be observed whether the LED grow light is ON or OFF at a particular point. This feature gives the users real-time reports on the state of the lighting condition that the plant is receiving and helps to ensure the system is running as intended.</p>                                                                                                                                                                                                                                                                                                                                            |
| 8 | <p>Watering State</p>  | <p>The watering state shows whether the water pump is working or not. Once activated, it powers the water pump to water the plant according to readings of the moisture sensors and set limits. This helps ensure that the soil is in the best condition to produce a healthy plantation.</p>                                                                                                                                                                                                                                                                                                                                                                                                                   |

#### 4.4 Workflow of Proposed System

This part presents the operational flow of an IoT-based indoor plant care system with the description of what interactions between system components are used to automate observing and taking care of plants.

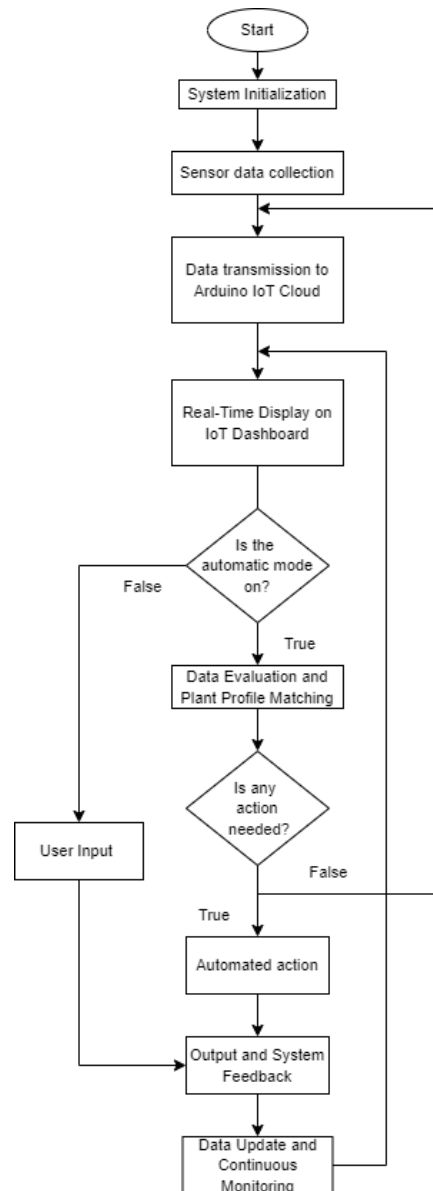


*Figure 4.17* Flowchart of workflow of Proposed System

The workflow shown above, Figure 4.17, explains how data is obtained from the environment, processed by the microcontroller, and used to cause necessary actions or

notifications to take place so as to maintain indoor plants in ideal conditions with minimal human intervention.

The operation of the proposed IoT-based indoor plant care system is based on the process of constant monitoring, evaluation, and control of the major environmental conditions for the well-being of indoor plants. The system is designed upon the ESP32 microcontroller, and a collection of sensors are used in integrating them, and they are connected to the Arduino IoT Cloud platform. The system aims to automate activities such as watering and lighting a plant as per the sensor data in real time. The step-by-step workflow is shown in the following.

### **Step 1: System Initialization**

The ESP32 microcontroller engages Wi-Fi and connects to the Arduino IoT Cloud platform once it is powered up. During this initialisation phase, all the sensors, like the soil moisture, temperature, and light sensors, are configured and activated. This will make the system convenient for collecting and transferring data effectively.

### **Step 2: Sensor Data Collection**

The ESP32 begins to collect a reading from the sensors at set intervals after initialisation. These sensors monitor the surroundings of the plant. The light sensor, LDR, is used to determine the level of light intensity, the temperature sensor is used to record the surrounding temperature, and the soil moisture sensor is to read the level of moisture within the soil. These readings are important when working out the plant's requirements.

### **Step 3: Data Transmission to Arduino IoT Cloud**

The ESP32 collects the sensor data and then makes use of a secure Wi-Fi connection to send it to the Arduino IoT Cloud platform. It is in this phase that it is made possible to

remotely view the real-time condition of the plants from any location using a mobile device.

#### **Step 4: Real-Time Display on IoT Dashboard**

The data transmitted is viewed on the Arduino IoT Cloud Dashboard, which is a remote app, so that the users can have a visual interface to monitor the state of their indoor plants. Temperature, soil moisture level, and light intensity, among other parameters, are displayed in real-time, which means the user can always know the current condition.

#### **Step 5: Mode Evaluation (Automatic or Manual)**

The system determines whether or not the automatic mode is active. If the automatic mode is disabled, then the system waits for manual input from the user through the IoT Cloud interface to perform any actions, for example, on or off LED lights manually. If the automatic mode is activated, the system goes ahead to analyse the sensor data and determine whether any actions are called for without involving human intervention.

#### **Step 6: Data Evaluation and Plant Profile Matching**

In the automatic mode, ESP32 compares the current findings with predefined thresholds that are based on the plant profile selected, including the cactus, tropical, and flowering plants. For instance, the acceptance of lower moisture levels will be varied where the cactus profile is concerned as compared to the tropical profile. These profiles are used by the system to come up with sound decisions according to the needs of each category of plants. For lighting, in case the level of light is below the threshold, then the system will automate the grow light to make sure that the plant receives enough light.

#### **Step 7: Decision-Making Logic**

After the evaluation of the data, the system then tests whether some of the conditions are out of the acceptable range. If all the readings are within the limits, the system updates

the cloud and proceeds with monitoring. But in case a condition is beyond or below the threshold, such as the soil being too dry or the environment too dark, the system detects that some action is needed.

### **Step 8: Automated action**

When an action is considered necessary, the ESP32 automatically activates the appropriate hardware. For example:

- In case soil moisture is low, the relay triggers the water pump, hence irrigating the plant.
- If there is insufficient light in the surroundings, then the grow light is switched.

Such actions assist in keeping the optimal growing environment of the plant without constant monitoring by the users.

### **Step 9: Output and System Feedback**

After taking the required action, the system provides feedback by changing the status at the Arduino IoT Dashboard. For instance, it can show the LED grow light turned on. This feature grants the user clear visibility in terms of what the system is doing in real time.

### **Step 10: Data Update and Continuous Monitoring**

After each action or status check, the ESP32 posts to the cloud the updated variables of the system. This makes the process of monitoring continuous and always in work with the up-to-date data. The cycle continues where the system will be able to effectively achieve the optimal health of plants at all times through the real-time mechanisation and tracking.

## 4.5 Coding of Proposed Algorithm

A complete coding the system are displayed in Appendix C.

### 4.5.1 Overall Indoor Plant Condition

One of the most important aspects of the IoT based indoor plant monitoring care system is the ability to determine the overall condition of health of the plant in real time. This function is taken care of by the `updatePlantCondition()` function, which outputs a numerical score that provides an indication of how closely the prevailing environmental conditions are to the plant's ideal growth requirements. Figure 4.18 below shows the full code block for `updatePlantCondition()`.

```
void updatePlantCondition() {
    float score = 100.0;
    float idealTemp = 0.0;
    float idealSoil = 0.0;

    if (plantType == 0) { // Cactus
        idealTemp = 30.0;
        idealSoil = 25.0;
    } else if (plantType == 1) { // Tropical
        idealTemp = 27.0;
        idealSoil = 50.0;
    } else if (plantType == 2) { // Flowering
        idealTemp = 24.0;
        idealSoil = 65.0;
    } else {
        idealTemp = 25.0;
        idealSoil = 45.0;
    }

    float tempPenalty = abs(temp - idealTemp) * 2.0;
    float soilPenalty = abs(soilmoisture - idealSoil);
    float totalPenalty = tempPenalty + soilPenalty;

    score -= totalPenalty;
    if (score < 0) score = 0;
    if (score > 100) score = 100;

    plantCondition = score;

    Serial.print("Plant Type: ");
    if (plantType == 0) Serial.print("Cactus | ");
    else if (plantType == 1) Serial.print("Tropical | ");
    else if (plantType == 2) Serial.print("Flowering | ");
    else Serial.print("Unknown | ");

    Serial.print("Temp: ");
    Serial.print(temp);
    Serial.print(" °C | Soil: ");
    Serial.print(soilmoisture);
    Serial.print(" % => Condition: ");
    Serial.print(plantCondition);
    Serial.println(" %");
}
```

Figure 4.18 Full code block `updatePlantCondition()`

The objective of the `updatePlantCondition()` function is to present a simplified percentage criteria for health score, which is 0-100%, that will reflect the suitability or unsuitability of the ambient conditions on which the plants are placed. This score is conspicuously featured on the Arduino IoT Cloud dashboard and mobile application and can easily be used by users to measure plant health and take necessary action.

The function starts off assigning a perfect plant health score of 100%. It then generates penalties for the difference between real-time sensor readings and the set ideal values for temperature, soil moisture and light intensity. These penalties are deduced from the base score, and the final score is clamped between the range of 0% and 100%.

### Ideal Conditions Based on Plant Type

The ideal levels are basically determined depending on general requirements within indoor plants but could be tweaked or modified depending on types of plants. The code show ideal temperatures and soil moisture based on plant type:

```
if (plantType == 0) { // Cactus
  idealTemp = 30.0;
  idealSoil = 25.0;
```

*Figure 4.19* Ideal Conditions Based on Plant Type (1)

- If the plant type is **Cactus**, it prefers hot and dry conditions.

```
} else if (plantType == 1) { // Tropical
  idealTemp = 27.0;
  idealSoil = 50.0;
```

*Figure 4.20* Ideal Conditions Based on Plant Type (2)

- If **Tropical**, it prefers warm and moderately moist soil.

```

} else if (plantType == 2) { // Flowering
    idealTemp = 24.0;
    idealSoil = 65.0;
}

```

*Figure 4.21 Ideal Conditions Based on Plant Type (3)*

- If **Flowering**, it prefers cooler and wetter conditions.

## Penalty Calculation

Every environmental parameter has its penalty formula to express how great the deviation from the ideal may be, and the total result in each penalty is then summed and deducted from the initial 100%. Below shows the code for penalty calculations:

```

float tempPenalty = abs(temp - idealTemp) * 2.0;
float soilPenalty = abs(soilmoisture - idealSoil);

```

*Figure 4.22 Penalty Calculation (1)*

If the current temp or soilmoisture deviate from the ideal:

- tempPenalty is more sensitive, penalty multiplied by 2.
- soilPenalty is direct difference.

```

float totalPenalty = tempPenalty + soilPenalty;

```

*Figure 4.23 Penalty Calculation (2)*

- Total impact on the condition score.

## Score Clamping and Output

To guarantee the availability of consistent data, the obtained score is clamped between 0% and 100% of the presented value in Figure 4.24:

```

// Clamp to 0-100%
if (score < 0) score = 0;
if (score > 100) score = 100;

```

*Figure 4.24 The score clamped between 0% and 100%*

This final score is saved in the variable `plantCondition` and is considered the main indicator of the plant's present wellbeing.

## Display and Feedback

The score can be accessed either through Serial Monitor in the case of debugging or through the dashboard or app of the IoT Cloud for user interaction. Such a percentage condition feature makes it simpler for users to check multiple environmental parameters using a single, transparent indicator.

The `updatePlantCondition()` function is a combined health checker that makes it easy for the environment to be detected from complex conditions in a simplified score. As it constantly compares sensor input with ideal conditions for growth, it enables users to continue to provide the best care for their indoor plants and initiate corrective measures where necessary.

### 4.5.2 Fuzzy Logic for Automated Watering System

To guarantee the plant gets adequate water supply when needed, the system makes use of a simplified fuzzy logic approach still implemented via the `applyFuzzyLogic()` function. This approach enables the system to function on a variety of inputs of watering with soft thresholds as compared to the binary forms of rules. It draws an analogy to human decision-making. Figure 4.25 below shows the complete code block for the `applyFuzzyLogic()` function.

```
float applyFuzzyLogic(float temp, float soilmoisture, int plantType) {  
    float base = 0;  
    if (plantType == 0) base = 0.5;           // Cactus  
    else if (plantType == 1) base = 1.0;      // Tropical  
    else if (plantType == 2) base = 1.5;      // Flowering  
    //else base = 1.0;                         // Default  
  
    float tempFactor = (temp > 30) ? 0.5 : 0.2;  
    float moistureFactor = (soilmoisture < 40) ? 1.0 : 0.5;  
  
    return base * tempFactor * moistureFactor;  
}
```

*Figure 4.25* The complete code block for the `applyFuzzyLogic()` function

The main goal of the `applyFuzzyLogic()` function is to find out a watering intensity level which is presented through the float value. This value indicates for how long the water pump should be active to effectively manage the duration of watering, based on three main parameters, which include plant type, the current temperature and the current soil moistness. Combining all three variables through fuzzy logic principles, the function takes an adaptable and responsive form of an irrigation strategy that adapts to the real-world diversity of the environment.

## Input Parameters

The inputs to the function are as follows:

```
float applyFuzzyLogic(float temp, float soilmoisture, int plantType) {
```

*Figure 4.26* Code from `applyFuzzyLogic()` (1)

- temp: Current temperature reading (°C)
- soilmoisture: Current soil moisture level (0–100%)
- plantType: Integer representing the plant category:
  - 0 = Cactus
  - 1 = Tropical
  - 2 = Flowering

## Step-by-Step Logic

### 1. Base Watering Need by Plant Type

There is a specific baseline water demand for each plant type characteristic, and this value represents the starting point for further alterations:

```

if (plantType == 0) base = 0.5;      // Cactus
else if (plantType == 1) base = 1.0; // Tropical
else if (plantType == 2) base = 1.5; // Flowering

```

*Figure 4.27* Code from applyFuzzyLogic() (2)

- **Cactus:** Requires minimal water (base = 0.5)
- **Tropical Plants:** Require moderate watering (base = 1.0)
- **Flowering Plants:** Require frequent watering (base = 1.5)

## 2. Temperature Factor

A multiplier is used depending on the ambient temperature:

```

float tempFactor = (temp > 30) ? 0.5 : 0.2;

```

*Figure 4.28* Code from applyFuzzyLogic() (3)

- If the temperature is more than 30°C, a higher factor, 0.5, is used due to increased evaporation and plant stress.
- Otherwise, a lower factor (0.2) is used, denoting less need of water.

## 3. Soil Moisture Factor

The function too takes into consideration how dry or damp the soil is:

```

float moistureFactor = (soilmoisture < 40) ? 1.0 : 0.5;

```

*Figure 4.29* Code from applyFuzzyLogic() (4)

- If the soil is dry, which is less than 40%, the intensity of watering stays high at 1.0.
- When the soil contains 40% or more moisture, the watering regime reduces to 0.5.

#### 4. Final Output Calculation

The final watering level is calculated as:

```
return base * tempFactor * moistureFactor;
```

*Figure 4.30* Code from applyFuzzyLogic() (5)

This is a product that integrates the influence of plant type, temperature, and soil quality into a unified product. Later on, this float value is multiplied by a time constant, for example 1000 ms, for determining the watering duration.

##### **Example:**

- Plant Type: Flowering (2)
- Temperature: 32°C
- Soil Moisture: 35%

- ```
base = 1.5  
tempFactor = 0.5  
moistureFactor = 1.0  
wateringLevel = 1.5 * 0.5 * 1.0 = 0.75
```

The apply The FuzzyLogic() function smartly modifies the watering time based on the interplay of plant varieties, heat levels and soil conditions. It comes with flexible and natural watering behaviour, without excessively wasting water in taking care of plants. This fuzzy logic model turns the system smarter and more responsive to various environment and plant requirements.

##### **4.5.3 Automated Watering Setting Based on Plant Profile**

The watering control is one of the components of the indoor plant care system that plays an important role. This function is part of the major ESP32 program and is in charge of

managing manual and automatic watering operations. It intelligently monitors the water pump relay activation depending on the evaluation of the sensor data and user configurations. The code block of the `handleWatering()` function is demonstrated in full below in Figure 4.31.

```
void handleWatering(){
    if (manualWater){
        wateringDuration=2000;
        isWatering = true;
        wateringStartTime = millis();
        manualWater = false;
        wateringState = true;
    }
    else if (autoWater && millis() - lastWaterTime > wateringCooldown && !isWatering){
        float threshold = getMoistureThreshold(plantType);
        wateringLevel = applyFuzzyLogic(temp, soilmoisture, plantType);

        if (soilmoisture < threshold && wateringLevel > 0) {
            wateringDuration = (int)(wateringLevel * 1000);
            isWatering = true;
            wateringStartTime = millis();
            lastWaterTime = millis();
            wateringState = true;
        }
    }
}
```

```
        Serial.println("Watering completed.");
    }
}
if (isWatering) {
    digitalWrite(RELAY_WATER_PIN, HIGH);
    if (millis() - wateringStartTime >= wateringDuration) {
        digitalWrite(RELAY_WATER_PIN, LOW);
        isWatering = false;
        wateringState = false;
        Serial.println("Watering completed.");
    }
} else {
    digitalWrite(RELAY_WATER_PIN, LOW);
}
}
```

*Figure 4.31* The complete code block for the `handleWatering()` function

The `handleWatering()` function is designed to make sure watering is only done when it is needed, as well as to reduce water use and maximise plant growth. It has two modes of operation:

- Manual mode: in which the user can access the pump directly using the Arduino IoT Cloud dashboard or mobile application.
- Automatic mode: in which the system will automatically decide when and for how long to water based on fuzzy logic rules and sensor output.

## Manual Mode

When the user acts on the manual watering button (`manualWater`), the system launches watering for a fixed amount of 2 seconds. This feature lets users test or water quickly in a regulated manner without having to use automated calculations.

```
if (manualWater){
  wateringDuration=2000;
  isWatering = true;
  wateringStartTime = millis();
  manualWater = false;
  wateringState = true;
}
```

*Figure 4.32* Code from `handleWatering()` (1)

The above code, Figure 4.32, allows for setting a preset watering time, triggers the beginning of watering, and resets the manual command to avoid repeated execution. The variable `wateringState` will display the state of watering.

## Automatic Watering Logic

Automatic watering is more sophisticated. The function itself looks at a couple of conditions and then starts the pump.

- The `autoWater` mode is enabled.
- The time that has passed since the last watering is more than some cooldown threshold.
- The system is not in a watering session at this point.

```
else if (autoWater && millis() - lastWaterTime > wateringCooldown && !isWatering) {
```

*Figure 4.33* Code from handleWatering() (2)

When these conditions are met, the system accesses a plant-specific soil moisture threshold by the following:

```
float threshold = getMoistureThreshold(plantType);
```

*Figure 4.34* Code from handleWatering() (3)

This threshold is important, as there are various plant types, cactus, tropical or flowering plants, that require different amounts of water. Next, the system employs fuzzy logic to determine the amount of water that the plant needs in terms of the current temperature, soil moisture and plant type:

```
wateringLevel = applyFuzzyLogic(temp, soilmoisture, plantType);
```

*Figure 4.35* Code from handleWatering() (4)

If the soil moisture in question is lower than the accepted threshold level and the output of the fuzzy logic is positive, then the pump is switched on for a pre-calculated period:

```
if (soilmoisture < threshold && wateringLevel > 0) {  
    wateringDuration = (int)(wateringLevel * 1000);  
    isWatering = true;  
    wateringStartTime = millis();  
    lastWaterTime = millis();  
}
```

*Figure 4.37* Code from handleWatering() (5)

This enables dispensing of water only in cases of need and in an acceptable amount, relative to the existing situation and plant species. A new part of the code that comes into play once watering starts is controlling the water pump relay. It switches the relay on

when `isWatering` is true and switches it off automatically after the `wateringDuration` time. It prevents overwatering and energy-efficient operations.

The `handleWatering ()` function provides the core logic for caring for a plant. It has capabilities for user-initiated and smart automatic watering services. The integration of fuzzy logic adds flexibility and enhances precision by incorporating environmental aspects and the plant's specific needs. This implementation is in line with the objective of the system: to keep or maintain healthy plant conditions with low manual effort.

#### **4.6 Summary**

This chapter presents an extensive description of the corresponding phase of the implementation of the proposed system. In this phase, there was practical implementation of hardware and software aspects that were designed and planned earlier on. The set-up of the hardware consisted of the integration of the sensors and actuators that included temperature sensors, soil moisture sensors, light-dependent resistors (LDR), LED lights and water pumps, all connected to a central microcontroller unit. These elements were thoroughly configured so that the proper data collection and optimal functioning of the system could be guaranteed.

On the software side, the logic to carry out sensor data processing and take appropriate action was created and implemented using a microcontroller programming environment. Codes of such plant care operations supported both automatic and manual control and data-driven decisions on the basis of fuzzy logic and condition-based actions. The system also made use of cloud connectivity, which was achieved through a platform upon which the same users can monitor conditions of plants, see readings from sensors and remotely control certain functions through a mobile application or web dashboard.

Additionally, the implementation accounted for the communications between components so that sensor readings could be used to define real-time actions such as grow

lights coming on or starting the watering routines. User interface elements were created to give concrete feedback and clear control possibilities and improve the overall user experience. In conclusion, this chapter summarises a successful integration of physical elements, embedded programming, and IoT technologies to provide a practical, intelligent system capable of achieving the purposes of efficient plant care and remote approach.

## **CHAPTER 5: TESTING**

### **5.1 Introduction**

During this testing phase it performs an important function as confirmation of the usability, reliability and performance of the hardware and software components of the system. The testing ensures the system works and intended under real-world conditions. All important aspects of the system such as the accuracy of sensor data, automated watering management, lighting response, evaluation of plant state, and monitoring over the cloud are analysed and address any problems that may have been present in the system, where both functionality testing and non-functionality testing are carried out. The outcomes of these tests help verify that the system can effectively monitor environmental parameters and respond appropriately to maintain optimal conditions for plant growth. This chapter outlines the testing procedures, test cases, results, and any necessary adjustments made during the refinement of the system.

### **5.2 Functional Testing**

The purpose of functional testing is to ensure every feature of the IoT-based Indoor Plant Care system follows the expected guidelines. It permits the system to fulfil its functions when it deals with sensors, actuators, and a mobile app. It checks to confirm that everything, like measuring temperature and soil, auto-irrigation, lighting control, and cloud display, is working as it should under different and challenging input values.

The purpose of functional testing is to prove that the system functions according to its plan when used on either the dashboard or mobile app. It covers checking how accurate the sensors are, how the system answers to human instructions, how it warns users, and if it properly performs automated routines for each plant in the best environment. Every function is tested by itself using different test cases, such as real-time sensor values, pressing buttons, and using conditions, to ensure the system gives the correct results.

With functional testing, any issues that can be detected and fixed resulting from differences between the system's actions and its expected functions, which guarantees high usability and user satisfaction.

### **5.2.1 Unit Testing**

To perform unit testing, each part of the IoT indoor plant care system is tested individually to check that it is working as intended. A unit can mean any small part of the program, like those calculating watering time, plant condition, or checking sensor values.

Unit testing mainly ensures that certain sections of code are correct and properly functioning without affecting other parts of the application. For example, the `applyFuzzyLogic()` function is tested by itself so that its calculations for plant water are confirmed using temperature, moisture level, and plant variety. The `updatePlantCondition()` function is also tested to check if it gives the proper plant health score for certain environmental conditions.

Normally, unit testing takes place as the code is being made to spot errors quickly and make the code dependable. Tests typically exchange test data in coded or virtual environments to verify the function's behaviour in both typical and unusual scenarios. By unit testing, it is confirmed that the essential system parts are free of errors before they are used in the entire application.

The test case of the proposed system is shown in Table 5.1 to Table 5.6.

#### **5.2.1.1 Test Cases for the Sensor Module**

Table 5.1 shows the test case for the unit testing for sensor reading.

Table 5.1 Unit test case for sensor reading

|               |                                                                                                                             |                                    |                              |                                         |               |        |
|---------------|-----------------------------------------------------------------------------------------------------------------------------|------------------------------------|------------------------------|-----------------------------------------|---------------|--------|
| ID            | 1                                                                                                                           | Module Name                        | Sensor Reading               |                                         |               |        |
| Objective     | To test the hardware functionality                                                                                          |                                    |                              |                                         |               |        |
| Name          | Rofinna Ellya Embang<br>Anak Umar @ Richard                                                                                 |                                    | Date                         | 25 <sup>th</sup> May 2025               | Status        | Pass   |
| No            | Prerequisites                                                                                                               |                                    |                              |                                         |               |        |
| 1             | The dashboard or the Serial Monitor simultaneously updates the values displayed by the sensors.                             |                                    |                              |                                         |               |        |
| 2             | A sensor is expected to give a response, not null or 0, in most situations in the test.                                     |                                    |                              |                                         |               |        |
| 3             | The system works well, and there is no noticeable delay or slowness.                                                        |                                    |                              |                                         |               |        |
| 4             | Results from sensors are matched accurately with the test input observations (made by hand or in a controlled environment). |                                    |                              |                                         |               |        |
| Test Scenario |                                                                                                                             | To test temperature sensor reading |                              |                                         |               |        |
| ID            | Test Case                                                                                                                   |                                    | Input Data                   | Expected Result                         | Actual Result | Status |
| 1             | Verify accuracy of temperature sensor                                                                                       |                                    | Simulated temperature = 28°C | Dashboard shows 28°C and matches sensor | As Expected   | Pass   |

|                      |                                         |                          |                                        |             |      |
|----------------------|-----------------------------------------|--------------------------|----------------------------------------|-------------|------|
| <b>Test Scenario</b> | To test soil moisture reading           |                          |                                        |             |      |
| 2                    | Verify accuracy of soil moisture sensor | Simulated moisture = 35% | Dashboard shows 35% and matches sensor | As Expected | Pass |

|                      |                                 |                        |                                      |             |      |
|----------------------|---------------------------------|------------------------|--------------------------------------|-------------|------|
| <b>Test Scenario</b> | To test light level reading     |                        |                                      |             |      |
| 3                    | Verify accuracy of light sensor | Simulated light = 1078 | Dashboard shows 1078 light intensity | As Expected | Pass |

### 5.2.1.1 Test Cases for the Mobile Application

Table 5.2 shows the test case for real-time sensor data display.

Table 5.2 Unit test case for real-time sensor data display

|               |                                                                                         |                                       |                                  |                           |        |      |
|---------------|-----------------------------------------------------------------------------------------|---------------------------------------|----------------------------------|---------------------------|--------|------|
| ID            | 2                                                                                       | Module Name                           | Real-Time Sensor Synchronization |                           |        |      |
| Objective     | To test the real-time sensor data display                                               |                                       |                                  |                           |        |      |
| Name          | Rofinna Ellya Embang<br>Anak Umar @ Richard                                             |                                       | Date                             | 25 <sup>th</sup> May 2025 | Status | Pass |
| No            | Prerequisites                                                                           |                                       |                                  |                           |        |      |
| 1             | Sensor data updates every 3 seconds and is reflected in the app/dashboard in real time. |                                       |                                  |                           |        |      |
| 2             | Pump turns on for 2000 ms when the button is pressed.                                   |                                       |                                  |                           |        |      |
| 3             | System applies correct thresholds after plant type is selected.                         |                                       |                                  |                           |        |      |
| 4             | Email alert sent when moisture is below threshold.                                      |                                       |                                  |                           |        |      |
| 5             | Light turns on immediately when the button is tapped.                                   |                                       |                                  |                           |        |      |
| Test Scenario |                                                                                         | To test real-time sensor data display |                                  |                           |        |      |

| ID | Test Case                              | Input Data                                | Expected Result                       | Actual Result | Status |
|----|----------------------------------------|-------------------------------------------|---------------------------------------|---------------|--------|
| 1  | Verify app syncs with live sensor data | Sensor sends updated data every 3 seconds | Dashboard and app update in real-time | As Expected   | Pass   |

| Test Scenario |                                     | To test manual watering activation |                            |             |      |
|---------------|-------------------------------------|------------------------------------|----------------------------|-------------|------|
| 2             | Tap "Manual Watering" button on app | User taps button                   | Pump activates for 2000 ms | As Expected | Pass |

| Test Scenario |                                | To test plant type selection    |                                                    |             |      |
|---------------|--------------------------------|---------------------------------|----------------------------------------------------|-------------|------|
| 3             | Select different plant profile | User selects "Tropical Plant"   | System sets thresholds for tropical plants         | As Expected | Pass |
| 4             | Select different plant profile | User selects "Flowering Plant"  | System sets thresholds for flowering plants        | As Expected | Pass |
| 5             | Select different plant profile | User selects "Cactus/Succulent" | System sets thresholds for Cactus/Succulent plants | As Expected | Pass |

| Test Scenario |                                   | To test lighting activation |                             |             |      |
|---------------|-----------------------------------|-----------------------------|-----------------------------|-------------|------|
| 7             | Verify LED activation towards LDR | The room get dark           | LED light will be activated | As Expected | Pass |

### 5.2.1.1 Test Cases for the Watering Control System

Table 5.3 shows the test case for the unit testing for watering control system.

*Table 5.3 Unit test case for watering control system*

|               |                                                                                             |                                              |                 |                           |             |
|---------------|---------------------------------------------------------------------------------------------|----------------------------------------------|-----------------|---------------------------|-------------|
| ID            | 3                                                                                           | Module Name Automate watering control        |                 |                           |             |
| Objective     |                                                                                             | To test the automate watering control system |                 |                           |             |
| Name          | Rofinna Ellya Embang<br>Anak Umar @ Richard                                                 |                                              | Date            | 25 <sup>th</sup> May 2025 | Status Pass |
| No            | Prerequisites                                                                               |                                              |                 |                           |             |
| 1             | The system executes fuzzy logic correctly to calculate watering duration.                   |                                              |                 |                           |             |
| 2             | The water pump activates only when necessary.                                               |                                              |                 |                           |             |
| 3             | Watering duration corresponds to environmental conditions (not too long/short).             |                                              |                 |                           |             |
| 4             | Watering is skipped if conditions are already optimal (moist soil or suitable temperature). |                                              |                 |                           |             |
| 5             | Sensors (temperature, soil moisture) provide accurate readings.                             |                                              |                 |                           |             |
| Test Scenario |                                                                                             | To test automated watering when dry          |                 |                           |             |
| ID            | Test Case                                                                                   | Input Data                                   | Expected Result | Actual Result             | Status      |

|   |                                                                        |                                                                                                                                     |                                                                                                                                                                               |             |      |
|---|------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|------|
| 1 | Fuzzy logic watering for dry soil and hot temperature (Cactus)         | <ul style="list-style-type: none"> <li>• Temp = 35°C</li> <li>• Soil Moisture = 25%</li> <li>• Plant Type = Cactus (0)</li> </ul>   | Pump runs for approximately, 250 ms (Pump runs briefly)                                                                                                                       | As Expected | Pass |
| 2 | Fuzzy logic watering for Moist Soil, Normal Temperature (Tropical)     | <ul style="list-style-type: none"> <li>• Temp = 28°C</li> <li>• Soil Moisture = 50%</li> <li>• Plant Type = Tropical (1)</li> </ul> | Watering skipped due to sufficient moisture                                                                                                                                   | As Expected | Pass |
| 3 | Fuzzy logic watering for Moderate Conditions (Flowering Plant)         | <ul style="list-style-type: none"> <li>• Temp = 30°C</li> <li>• Soil Moisture = 38%</li> <li>• Plant Type = Flowering</li> </ul>    | <ul style="list-style-type: none"> <li>• Pump runs for approximately, 300 ms (Short watering cycle)</li> <li>• Normal operation under moderate dryness</li> </ul>             | As Expected | Pass |
| 4 | Fuzzy logic watering for Cold Temperature, Dry Soil (Tropical Plant)   | <ul style="list-style-type: none"> <li>• Temp = 22°C</li> <li>• Soil Moisture = 28%</li> <li>• Plant Type = Tropical</li> </ul>     | <ul style="list-style-type: none"> <li>• Pump runs for approximately, 200 ms (Minimal watering despite dryness)</li> <li>• Reduced watering due to low temp</li> </ul>        | As Expected | Pass |
| 5 | Fuzzy logic watering for Hot Temperature, Moist Soil (Flowering Plant) | <ul style="list-style-type: none"> <li>• Temp = 34°C</li> <li>• Soil Moisture = 55%</li> <li>• Plant Type = Flowering</li> </ul>    | <ul style="list-style-type: none"> <li>• Pump runs for approximately, 375 ms (Short duration, limited watering)</li> <li>• Avoid unnecessary watering despite heat</li> </ul> | As Expected | Pass |

### 5.2.1.1 Test Cases for the Plant Health Evaluation

Table 5.4 shows the test case for the unit testing for plant health evaluation system.

*Table 5.4 Unit test case for plant health evaluation system*

|               |                                                               |                                                     |                                                     |                           |        |      |
|---------------|---------------------------------------------------------------|-----------------------------------------------------|-----------------------------------------------------|---------------------------|--------|------|
| ID            | 4                                                             | Module Name                                         | Plant Health Evaluation System                      |                           |        |      |
| Objective     |                                                               | To test the plant health evaluation system          |                                                     |                           |        |      |
| Name          |                                                               | Rofinna Ellya Embang<br>Anak Umar @ Richard         | Date                                                | 25 <sup>th</sup> May 2025 | Status | Pass |
| No            | Prerequisites                                                 |                                                     |                                                     |                           |        |      |
| 1             | Sensor data is read correctly (temperature, moisture, light). |                                                     |                                                     |                           |        |      |
| 2             | Health score is calculated using the formula.                 |                                                     |                                                     |                           |        |      |
| 3             | Score stays between 0% and 100%.                              |                                                     |                                                     |                           |        |      |
| 4             | Score is shown on the app and Serial Monitor.                 |                                                     |                                                     |                           |        |      |
| Test Scenario |                                                               | To test health score calculation                    |                                                     |                           |        |      |
| ID            | Test Case                                                     | Input Data                                          | Expected Result                                     | Actual Result             | Status |      |
| 1             | Evaluate condition based on 3 parameters                      | Type = Tropical<br>Temp = 30.5°C,<br>Moisture = 40% | Score = 83%,<br>displayed on app and Serial Monitor | As Expected               | Pass   |      |

|                      |                                  |                                |                              |                      |               |
|----------------------|----------------------------------|--------------------------------|------------------------------|----------------------|---------------|
| <b>Test Scenario</b> | To test clamping of health score |                                |                              |                      |               |
| <b>ID</b>            | <b>Test Case</b>                 | <b>Input Data</b>              | <b>Expected Result</b>       | <b>Actual Result</b> | <b>Status</b> |
| 2                    | Clamp score to 0–100% range      | Temp = 45°C,<br>Moisture = 10% | Final score is clamped to 0% | As Expected          | Pass          |

### 5.2.1.1 Test Cases for the Automatic Light Activation

Table 5.5 shows the test case for the unit testing for automatic grow light control.

*Table 5.5 Unit test case for automatic grow light control*

|               |                                                                              |                                             |                              |                           |               |        |
|---------------|------------------------------------------------------------------------------|---------------------------------------------|------------------------------|---------------------------|---------------|--------|
| ID            | 5                                                                            | Module Name                                 | Automatic Grow Light Control |                           |               |        |
| Objective     |                                                                              | To test the automatic grow light control    |                              |                           |               |        |
| Name          |                                                                              | Rofinna Ellya Embang<br>Anak Umar @ Richard | Date                         | 25 <sup>th</sup> May 2025 | Status        | Pass   |
| No            | Prerequisites                                                                |                                             |                              |                           |               |        |
| 1             | Light sensor provides correct real-time readings.                            |                                             |                              |                           |               |        |
| 2             | System checks if auto light feature is enabled.                              |                                             |                              |                           |               |        |
| 3             | If light is below threshold and auto mode is ON → grow light turns ON.       |                                             |                              |                           |               |        |
| 4             | If light is sufficient or auto mode is OFF → grow light stays OFF.           |                                             |                              |                           |               |        |
| 5             | Changes are reflected in the growLightState variable and physical LED state. |                                             |                              |                           |               |        |
| Test Scenario |                                                                              | To test light level                         |                              |                           |               |        |
| ID            | Test Case                                                                    |                                             | Input Data                   | Expected Result           | Actual Result | Status |
|               | Light level drops below                                                      |                                             | Light Level = 4022           | • System detects darkness |               |        |

|   |                                |                   |                                                                                                                                                                                               |             |      |
|---|--------------------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|------|
| 1 | threshold at night             |                   | <ul style="list-style-type: none"> <li>• Grow light state (growLightState) set to ON</li> <li>• LED is physically switched ON</li> </ul>                                                      | As Expected | Pass |
| 2 | Light Level is Sufficient      | Light Level = 800 | <ul style="list-style-type: none"> <li>• System detects sufficient natural light</li> <li>• Grow light state (growLightState) remains OFF</li> <li>• Grow light is not activated</li> </ul>   | As Expected | Pass |
| 3 | Auto Light Feature is Disabled | Light Level = 765 | <ul style="list-style-type: none"> <li>• Automation is skipped</li> <li>• growLightState does not change automatically</li> <li>• Grow light remains OFF regardless of light level</li> </ul> | As Expected | Pass |

### 5.2.2 Integration Test

Table 5.6 shows the test case for the unit testing for overall intergration.

*Table 5.6 Unit test case for Overall Integration*

|               |                                                                                                                                                           |                                             |                                                                                                                        |                                                                                                                   |               |        |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|---------------|--------|
| ID            | 6                                                                                                                                                         | Module Name                                 | Overall Integration                                                                                                    |                                                                                                                   |               |        |
| Objective     |                                                                                                                                                           | To test the integration of the system       |                                                                                                                        |                                                                                                                   |               |        |
| Name          |                                                                                                                                                           | Rofinna Ellya Embang<br>Anak Umar @ Richard | Date                                                                                                                   | 25 <sup>th</sup> May 2025                                                                                         | Status        | Pass   |
| No            | Prerequisites                                                                                                                                             |                                             |                                                                                                                        |                                                                                                                   |               |        |
| 1             | ESP32 is connected to Wi-Fi and synced with Arduino IoT Cloud.                                                                                            |                                             |                                                                                                                        |                                                                                                                   |               |        |
| 2             | Mobile app and dashboard reflect real-time cloud variable updates.                                                                                        |                                             |                                                                                                                        |                                                                                                                   |               |        |
| 3             | All commands sent from app/cloud are successfully received and executed by ESP32.                                                                         |                                             |                                                                                                                        |                                                                                                                   |               |        |
| 4             | Sensor readings and system variables (e.g., soilMoisture, temp, lightLevel, manualWater, etc.) are correctly updated in both directions (device ↔ cloud). |                                             |                                                                                                                        |                                                                                                                   |               |        |
| Test Scenario |                                                                                                                                                           | To test Sensor-to-Cloud Communication       |                                                                                                                        |                                                                                                                   |               |        |
| ID            | Test Case                                                                                                                                                 |                                             | Input Data                                                                                                             | Expected Result                                                                                                   | Actual Result | Status |
| 1             | Verify that ESP32 pushes real-time data to Arduino IoT Cloud                                                                                              |                                             | <ul style="list-style-type: none"><li>• Soil Moisture: 45%</li><li>• Temperature: 26°C</li><li>• Light: 1007</li></ul> | <ul style="list-style-type: none"><li>• soilMoisture = 45</li><li>• temp = 26</li><li>• lightLevel = 60</li></ul> | As Expected   | Pass   |

| Test Scenario |                                                          | To test Cloud-to-Hardware Trigger            |                                                                                                                                                                                        |               |        |
|---------------|----------------------------------------------------------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| ID            | Test Case                                                | Input Data                                   | Expected Result                                                                                                                                                                        | Actual Result | Status |
| 2             | Manual watering command from app triggers the water pump | User sets manualWater = true from mobile app | <ul style="list-style-type: none"> <li>• ESP32 detects command</li> <li>• Relay activates pump</li> <li>• Pump runs for defined time</li> <li>• manualWater resets to false</li> </ul> | As Expected   | Pass   |

| Test Scenario |                                                   | To test Auto-Watering Flow End-to-End                                                                                                           |                                                                                                                                                                                                              |               |        |
|---------------|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| ID            | Test Case                                         | Input Data                                                                                                                                      | Expected Result                                                                                                                                                                                              | Actual Result | Status |
| 3             | Soil moisture is low and auto watering is enabled | <ul style="list-style-type: none"> <li>• soilMoisture = 25</li> <li>• autoWater = true</li> <li>• plantType = 2</li> <li>• temp = 33</li> </ul> | <ul style="list-style-type: none"> <li>• applyFuzzyLogic() runs on ESP32</li> <li>• Calculates watering duration</li> <li>• Relay activates pump</li> <li>• wateringLevel is updated in IoT Cloud</li> </ul> | As Expected   | Pass   |

| Test Scenario |                    | To test Auto Light Activation                                         |                                                                                                                                                                        |               |        |
|---------------|--------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| ID            | Test Case          | Input Data                                                            | Expected Result                                                                                                                                                        | Actual Result | Status |
| 4             | Light level is low | <ul style="list-style-type: none"> <li>• lightLevel = 3009</li> </ul> | <ul style="list-style-type: none"> <li>• ESP32 turns ON grow light</li> <li>• growLightState = true in Cloud</li> <li>• Status is updated on the mobile app</li> </ul> | As Expected   | Pass   |

| Test Scenario |                                                | Plant Condition Score Evaluation & Display                                                 |                                                                                                                                                                                 |               |        |
|---------------|------------------------------------------------|--------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------|
| ID            | Test Case                                      | Input Data                                                                                 | Expected Result                                                                                                                                                                 | Actual Result | Status |
| 5             | Health score updates based on environment data | <ul style="list-style-type: none"> <li>• temp = 29</li> <li>• soilMoisture = 40</li> </ul> | <ul style="list-style-type: none"> <li>• updatePlantCondition() runs</li> <li>• Score calculated (e.g., 95%)</li> <li>• plantCondition variable updates in IoT Cloud</li> </ul> | As Expected   | Pass   |

|  |  |  |                                                                              |  |  |
|--|--|--|------------------------------------------------------------------------------|--|--|
|  |  |  | <ul style="list-style-type: none"> <li>• Score is shown in app UI</li> </ul> |  |  |
|--|--|--|------------------------------------------------------------------------------|--|--|

### 5.3 Non-Functional Testing

Non-functional testing aims at testing the performance of the IoT-based indoor plant care system as opposed to what the system does. Non-functional testing, in contrast to functional testing that verifies distinct features or actions, evaluates performances, dependability, usability, and the behaviour of the system when subjected to different situations.

In the current project, the non-functional testing will make sure that the system does not only do its work properly but also corresponds to the quality requirements that influence the user experience and the stability of the system under work. To give some examples, performance testing verifies the responsiveness of the system in cases when the values provided by the sensors change, whereas reliability testing ensures that the system is capable of stable operation over a long time without a malfunction. Usability testing is used to identify whether the mobile application interface is intuitive and friendly to the user, particularly when presenting the plant condition or controlling the devices.

These tests play an important role to ensure that the system may be functionally beastly in a real-life setting. They assist in revealing the problems in the system response time, taken up by the system, unresponsiveness or hardware/software restrictions that might influence the efficiency of the plant care system.

With the incorporation of non-functional testing, the overall stability, usability and future reliability of the system are greatly enhanced, making it provide a reliable and easy-to-use solution to indoor plant monitoring and automation.

### 5.3.1 Performance Testing

The performance testing measures how responsive, quick, and reliable the IoT-based indoor plant care system can be in normal and high-load scenarios. It is concerned with the measurement of the sensor data update rate on the mobile app, the latency of the user command response (manual watering), the latency of automated control (auto-watering), and the capability of the system to sustain constant data broadcast and synchronisation with the Arduino IoT Cloud. This testing will make sure that the system provides real-time feedback and is responsive without delays or failure. Table 5.7 presents the test case used for unit testing related to performance monitoring.

Table 5.7 Unit test case for Performance Monitoring Module

|               |                                                                               |                                          |                                                                                                                                |                                                               |               |        |
|---------------|-------------------------------------------------------------------------------|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|---------------|--------|
| ID            | 7                                                                             | Module Name                              | Performance Monitoring Module                                                                                                  |                                                               |               |        |
| Objective     | To test the performance of the system                                         |                                          |                                                                                                                                |                                                               |               |        |
| Name          | Rofinna Ellya Embang<br>Anak Umar @ Richard                                   |                                          | Date                                                                                                                           | 25 <sup>th</sup> May 2025                                     | Status        | Pass   |
| No            | Prerequisites                                                                 |                                          |                                                                                                                                |                                                               |               |        |
| 1             | ESP32 device is powered on and connected to Wi-Fi.                            |                                          |                                                                                                                                |                                                               |               |        |
| 2             | All sensors (soil moisture, temperature, light) are connected and calibrated. |                                          |                                                                                                                                |                                                               |               |        |
| 3             | Arduino IoT Cloud dashboard is active and synced with the ESP32.              |                                          |                                                                                                                                |                                                               |               |        |
| 4             | Mobile app is open and connected to the same IoT Cloud account.               |                                          |                                                                                                                                |                                                               |               |        |
| 5             | AutoWater and growLight features (if tested) are enabled in settings.         |                                          |                                                                                                                                |                                                               |               |        |
| Test Scenario |                                                                               | System responsiveness during data update |                                                                                                                                |                                                               |               |        |
| ID            | Test Case                                                                     |                                          | Input Data                                                                                                                     | Expected Result                                               | Actual Result | Status |
| 1             | Verify sensor data is displayed in real-time on the mobile app                |                                          | <ul style="list-style-type: none"><li>• Soil moisture = 30%</li><li>• Temperature = 28°C</li><li>• Light level = 689</li></ul> | Sensor values update on the mobile dashboard within 3 seconds | As Expected   | Pass   |

|                      |                                                                      |                              |                                   |                      |               |  |
|----------------------|----------------------------------------------------------------------|------------------------------|-----------------------------------|----------------------|---------------|--|
| <b>Test Scenario</b> | Response time for manual watering command                            |                              |                                   |                      |               |  |
| <b>ID</b>            | <b>Test Case</b>                                                     | <b>Input Data</b>            | <b>Expected Result</b>            | <b>Actual Result</b> | <b>Status</b> |  |
| 2                    | Check delay between pressing manual water button and pump activation | Tap "Manual Watering" in app | Pump activates within 1–2 seconds | As Expected          | Pass          |  |

| Test Scenario |                                                                  | Load handling during multiple updates                   |                                                           |               |        |
|---------------|------------------------------------------------------------------|---------------------------------------------------------|-----------------------------------------------------------|---------------|--------|
| ID            | Test Case                                                        | Input Data                                              | Expected Result                                           | Actual Result | Status |
| 3             | Check performance when multiple sensors send data simultaneously | Soil, temp, and light sensors send data every 2 seconds | All sensor data updates smoothly without app lag or crash | As Expected   | Pass   |

| Test Scenario |                                                      | Auto-watering trigger latency                                                                                                |                                                                 |               |        |
|---------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|---------------|--------|
| ID            | Test Case                                            | Input Data                                                                                                                   | Expected Result                                                 | Actual Result | Status |
| 4             | Measure time between dry reading and pump activation | <ul style="list-style-type: none"> <li>• Soil moisture = 22%</li> <li>• AutoWater = true</li> <li>• plantType = 1</li> </ul> | Pump activates within 2 seconds after dry condition is detected | As Expected   | Pass   |

| Test Scenario |                                                | Cloud synchronization speed             |                                                             |               |        |
|---------------|------------------------------------------------|-----------------------------------------|-------------------------------------------------------------|---------------|--------|
| ID            | Test Case                                      | Input Data                              | Expected Result                                             | Actual Result | Status |
| 5             | Measure sync delay between ESP32 and IoT Cloud | Update any variable (e.g., temp = 32°C) | Variable reflects in IoT Cloud dashboard within 1–5 seconds | As Expected   | Pass   |

### 5.3.2 Reliability Testing

Table 5.8 shows the test case for the unit testing for system monitoring.

*Table 5.8 Unit test case for System Monitoring Module*

|           |                                                                                        |                                             |                          |                           |        |      |
|-----------|----------------------------------------------------------------------------------------|---------------------------------------------|--------------------------|---------------------------|--------|------|
| ID        | 8                                                                                      | Module Name                                 | System Monitoring Module |                           |        |      |
| Objective |                                                                                        | To test the reliability of the system       |                          |                           |        |      |
| Name      |                                                                                        | Rofinna Ellya Embang<br>Anak Umar @ Richard | Date                     | 25 <sup>th</sup> May 2025 | Status | Pass |
| No        | Prerequisites                                                                          |                                             |                          |                           |        |      |
| 1         | Ensure ESP32 is powered continuously and running the final, error-free firmware.       |                                             |                          |                           |        |      |
| 2         | Confirm all sensors (soil moisture, temperature, light) are calibrated and functional. |                                             |                          |                           |        |      |
| 3         | Verify a stable Wi-Fi connection is available for the entire test duration.            |                                             |                          |                           |        |      |
| 4         | Place the system in a normal indoor environment with minimal disturbances.             |                                             |                          |                           |        |      |
| 5         | Confirm manual buttons for watering and lighting are working as expected.              |                                             |                          |                           |        |      |
| 6         | Ensure the system can reconnect automatically after temporary Wi-Fi disconnection.     |                                             |                          |                           |        |      |

| Test Scenario |                                                                          | System stability over continuous operation                   |                                                                                  |               |        |
|---------------|--------------------------------------------------------------------------|--------------------------------------------------------------|----------------------------------------------------------------------------------|---------------|--------|
| No.           | Test Case                                                                | Input Data                                                   | Expected Result                                                                  | Actual Result | Status |
| 1             | Check if system operates continuously for 24 days without reset or crash | Continuous environmental monitoring (normal room conditions) | System remains responsive and functional for 24 hours without errors or restarts | As Expected   | Pass   |

| Test Scenario |                                             | Sensor accuracy over long runtime                  |                                                                    |               |        |
|---------------|---------------------------------------------|----------------------------------------------------|--------------------------------------------------------------------|---------------|--------|
| No.           | Test Case                                   | Input Data                                         | Expected Result                                                    | Actual Result | Status |
| 2             | Verify consistent sensor data over 24 hours | Soil, temperature, and light readings at intervals | Sensor values stay within expected range with minimal fluctuations | As Expected   | Pass   |

| Test Scenario |                                                             | Cloud data sync reliability                             |                                                                             |               |        |
|---------------|-------------------------------------------------------------|---------------------------------------------------------|-----------------------------------------------------------------------------|---------------|--------|
| No.           | Test Case                                                   | Input Data                                              | Expected Result                                                             | Actual Result | Status |
| 3             | Confirm data continues syncing to IoT Cloud during long use | ESP32 sends updates to cloud every 3 seconds for 24 hrs | Data updates appear consistently on cloud dashboard without missing entries | As Expected   | Pass   |

| Test Scenario |                                                 | Manual control reliability               |                                                           |               |        |
|---------------|-------------------------------------------------|------------------------------------------|-----------------------------------------------------------|---------------|--------|
| No.           | Test Case                                       | Input Data                               | Expected Result                                           | Actual Result | Status |
| 4             | Validate repeated use of manual watering button | Manual button pressed 20 times for water | Each press activates pump correctly without skip or delay | As Expected   | Pass   |

| Test Scenario |                                                          | Recovery from Wi-Fi disconnection                 |                                                                      |               |        |
|---------------|----------------------------------------------------------|---------------------------------------------------|----------------------------------------------------------------------|---------------|--------|
| No.           | Test Case                                                | Input Data                                        | Expected Result                                                      | Actual Result | Status |
| 5             | Check if system resumes normal function after Wi-Fi loss | Wi-Fi is disconnected for 2 minutes then restored | ESP32 reconnects automatically and resumes cloud sync and operations | As Expected   | Pass   |

| Test Scenario |                                                            | Auto-watering reliability over time                            |                                                                      |               |        |
|---------------|------------------------------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------------|---------------|--------|
| No.           | Test Case                                                  | Input Data                                                     | Expected Result                                                      | Actual Result | Status |
| 6             | Check auto-watering triggers consistently when soil is dry | Soil moisture drops below threshold multiple times in 48 hours | Pump activates accurately every time based on fuzzy logic thresholds | As Expected   | Pass   |

### 5.3.3 Usability Testing

Reliability testing measures the system's stability and constancy under long-term conditions. It guarantees that the IoT-based indoor plant care system may run sustained, for example, 24 hours, without crashing, memory leaking, or a decrease in performance. It is also confirmed in this test that the sensor readings are consistent, the data is continuously synchronised with the IoT Cloud, and the system can resume normal operation after some interruptions, for instance, a temporary loss of Wi-Fi connectivity. The aim is to ensure that the system is reliable in the real-world situation. Table 5.9 shows the test case for the unit testing for the mobile app.

*Table 5.9 Unit test case for Mobile app*

|               |                                                                          |                                          |                                |                                                                                             |               |        |
|---------------|--------------------------------------------------------------------------|------------------------------------------|--------------------------------|---------------------------------------------------------------------------------------------|---------------|--------|
| ID            | 9                                                                        | Module Name                              | Mobile app                     |                                                                                             |               |        |
| Objective     |                                                                          | To test the automatic grow light control |                                |                                                                                             |               |        |
| Name          | Rofinna Ellya Embang<br>Anak Umar @ Richard                              |                                          | Date                           | 25 <sup>th</sup> May 2025                                                                   | Status        | Pass   |
| No            | Prerequisites                                                            |                                          |                                |                                                                                             |               |        |
| 1             | The ESP32 device must be online.                                         |                                          |                                |                                                                                             |               |        |
| 2             | The plant profile and auto watering feature must be set up in the cloud. |                                          |                                |                                                                                             |               |        |
| Test Scenario |                                                                          | User interface intuitiveness and control |                                |                                                                                             |               |        |
| No.           | Test Case                                                                |                                          | Input Data                     | Expected Result                                                                             | Actual Result | Status |
| 1             | Verify that users can enable/disable manual watering from the mobile app |                                          | Pressed manual watering button | System toggles auto watering and provides clear visual confirmation (e.g., status = ON/OFF) | As Expected   | Pass   |

## 5.4 Analysis on User Acceptance Survey

The goal of the survey is to provide insight on whether the proposed system has met its goals as well as feedback on the experience of the overall system. This questionnaire is composed of five parts, namely A, B, C, D, and E. The first part mainly dealt with general satisfaction, whereas section B was concerned with system usability. In Section C that focuses on the usefulness of the system, Section D will be dedicated to the system reliability, and Section E will be used to collect information on the user feedback. The overall set of questions of the survey is found in Appendix D. The list of the results of the survey and all the questions presented are included below.

### Section A: General Satisfaction

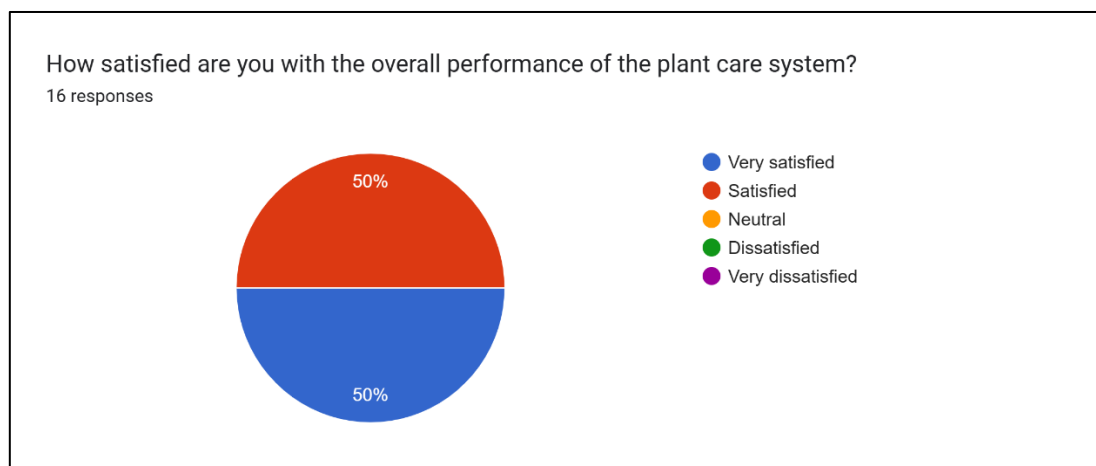


*Figure 5.1* The first question for Section A

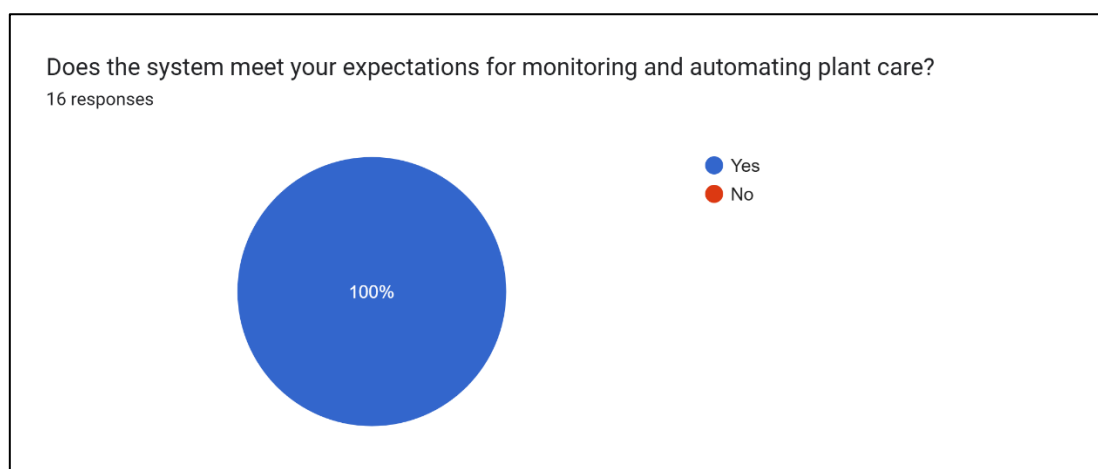


*Figure 5.2* The second question for Section A

In section A, the overall general satisfaction with the system was emphasised by the respondent. In figure 5.1, we find that 100% of the respondents in this section are satisfied with the overall performance of the system of plant care. Figure 5.2 also reveals that all the respondents agree the system is doing just as is expected in terms of its monitoring and automating the plant care system.

## Section B: Usability

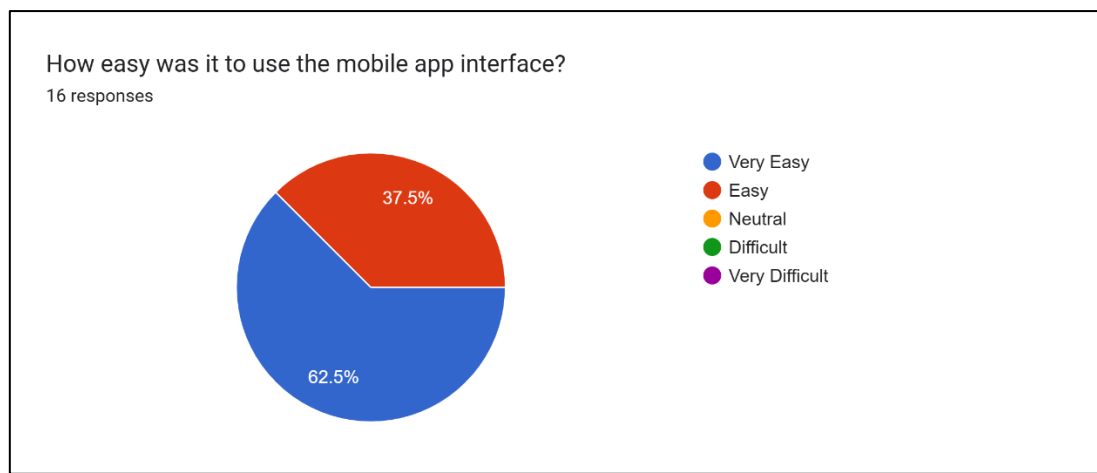


*Figure 5.3 The first question for Section B*

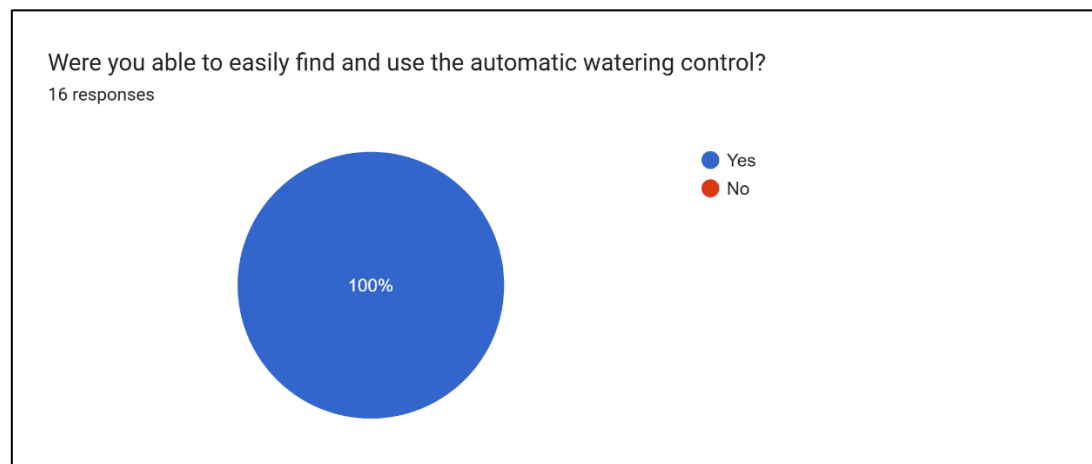


*Figure 5.4 The second question for Section B*

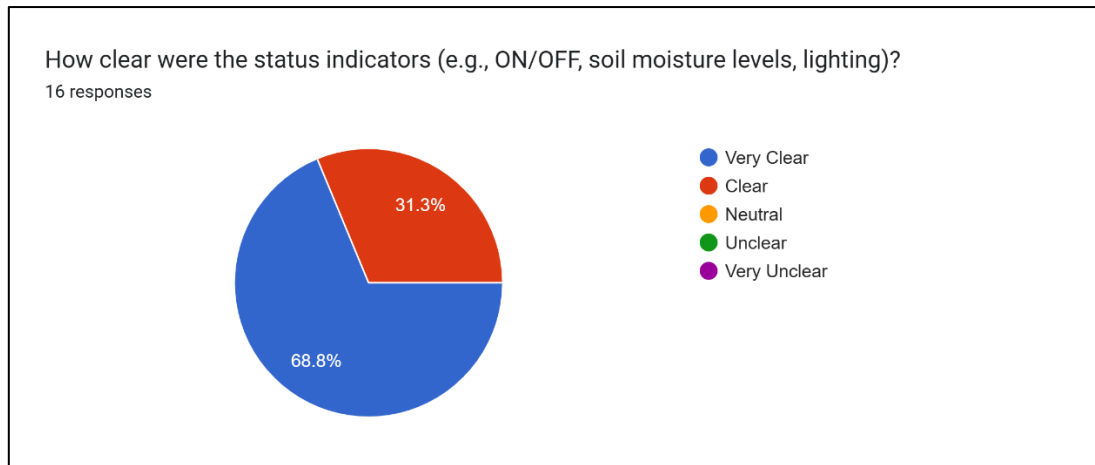


*Figure 5.5* The third question for Section B

Section B aims to obtain information primarily about the usability of the system. In Figure 5.3, every respondent has confirmed that the mobile application is convenient and user-friendly. All the respondents agree that automatic watering control is easy to find, according to the opinion expressed in Figure 5.4. In Figure 5.5, the respondents backed the claim that the status indicator is clearly visible by all the respondents.

### Section C: Usefulness of Features

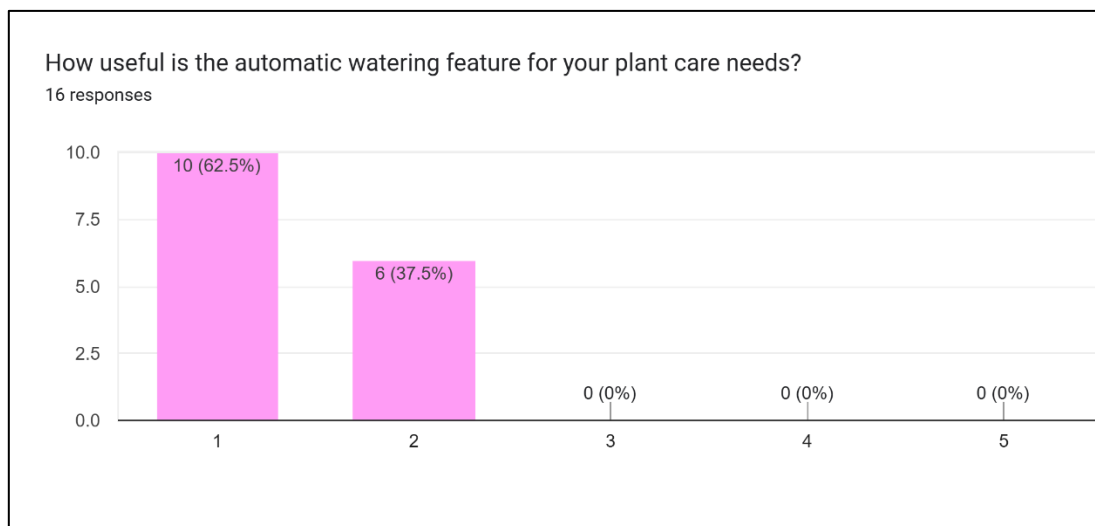


*Figure 5.6* The first question for Section C

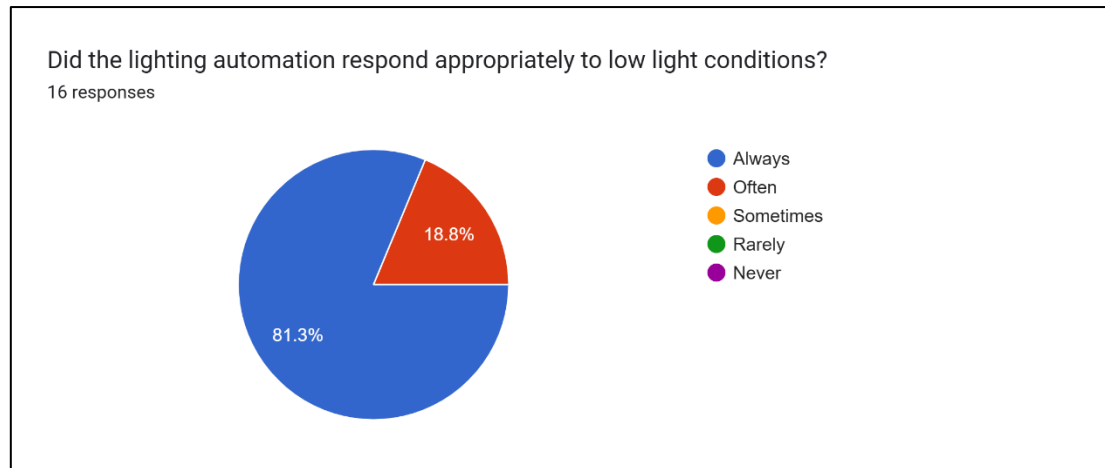


*Figure 5.7* The second question for Section C

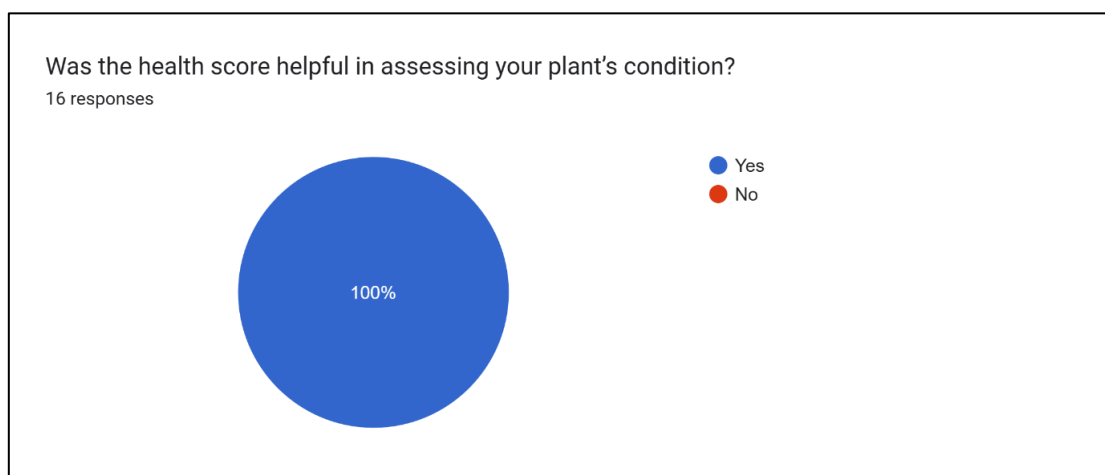


*Figure 5.8* The third question for Section C

This section will discuss the usefulness of the available features. The respondents will be expected to assess the current system features, such as automation of watering and lighting, sensor monitoring, etc., to figure out whether they are effective enough to assist in plant maintenance. Figure 5.6 displays that in all the respondents, they all discovered that the automatic watering is beneficial in the maintenance of plant care. 81.3% of the respondents concur that they tried the lighting automation, and it lights up the plant when there is not enough light, as given in Figure 5.7. Finally, in this section, every respondent responds positively when asked about the usefulness of the plant health score to measure their plant condition.

## Section D: Reliability

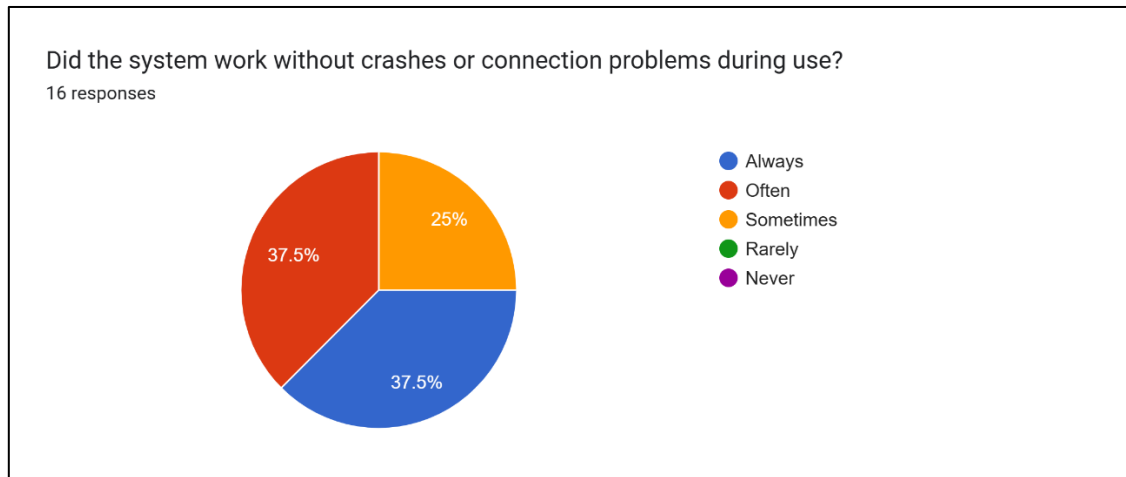


*Figure 5.9 The first question for Section D*

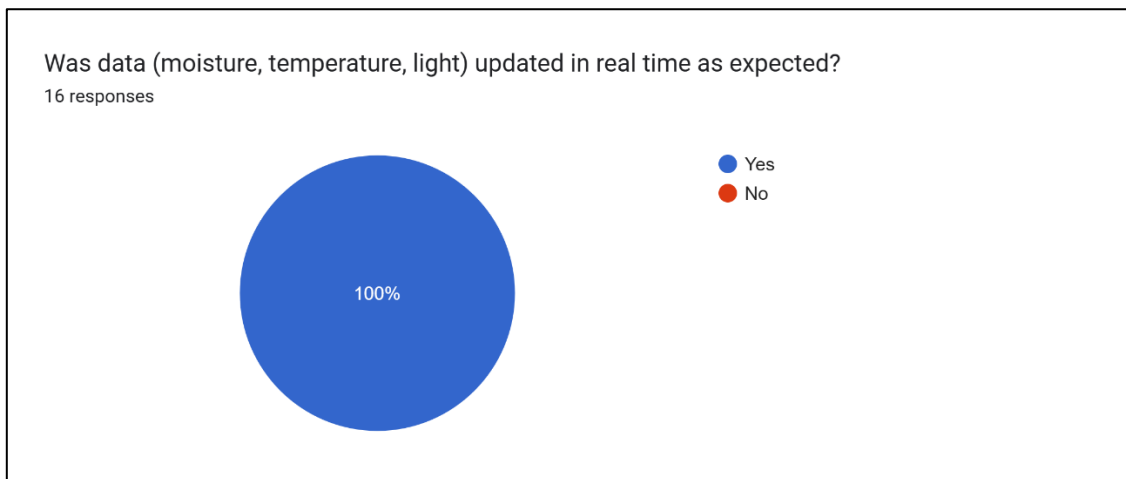


*Figure 5.10 The second question for Section D*

Here, the stability and reliability of the IoT-based system of taking care of plants indoors are assessed. The assessment focusses on the system's effectiveness, ability to fulfil its functions, and absence of errors or unexpected system failures. In Figure 5.9, 75 per cent of the respondents revealed that they did not have problems or crashes with the system or connection challenges, but 25 per cent of the respondents reported that they had these issues at times. In Figure 5.10, all the respondents concur that the data is expected to be updated in real-time.

## Section E: Feedback

What features did you like the most?

16 responses

setup straightforward

The automatic watering system is really convenient and works well with different plant types.

The live data feed is reliable

I like the overall indoor plant condition indicator

The grow light turns on automatically when it's too dark

The fuzzy logic watering system is impressive it waters just enough for each plant

I like that it automatically stops watering after a set time to avoid overwatering

I love the plant condition score. It's helpful to see a quick health rating

The live sensor data updates every few seconds

The system is very responsive

It's great that it supports multiple plant profiles like cactus, tropical, and flowering

I appreciate how the ESP32 handles multiple tasks smoothly without crashing

The LED grow light control based on brightness levels is a thoughtful feature.

The mobile app is easy to use and lets me monitor everything from my phone.

The system is beginner friendly

I liked that it used both temperature and moisture data to make watering decisions.

*Figure 5.11 The first question for Section E*

This section collects open-ended questions about the interests of users in the overall experience with the system. It will prompt the participants to discuss what they liked best and what could be improved so that it provides beneficial advice to make more improvements in the future. The responses provided by the respondents, most of whom exclusively appreciate the automatic function, appear in Figure 5.11. This feature was noted by many users as being especially helpful because it makes plant care easier since there is no need to manually monitor and water the plants. The users often referred to the time and convenience of the process as part of the automation, and therefore the automation played a great part in the overall positive experience of using it.

What improvements would you suggest for the system?
16 responses

Add a feature to control watering duration manually in the app

It would be better if I could add custom plant profiles instead of just the 3 defaults

Add a feature to simulate sunrise/sunset lighting to promote healthier growth

Make the watering threshold adjustable directly from the mobile app.

It would be nice to control multiple plant systems from one app dashboard

Let users receive push notifications on their phone

Consider adding voice control integration with Alexa or Google Assistant

An option to log historical data and view trends over time would be useful.

The system could benefit from a moisture status icon or color indicator for quicker visual feedback

Include humidity sensors to monitor air moisture levels as well.

Sometimes the app is slow to load

Make the watering threshold adjustable directly from the mobile app

The interface could use a more modern design

It would help if there were a battery backup in case of power outages

Include a water level sensor to alert when the water source is running low

Allow users to name their plants, it's easier to identify them that way

*Figure 5.12 The second question for Section E*

In Figure 5.12, there is a majority of people who mentioned that there was some improvement that could be done in future. A majority of the respondents also proposed some changes that might be adopted in the further development of the system. Among these recommendations are an increased number of available plant profiles, a smoother connection between the mobile app and a phone, and additional features such as the inclusion of a humidity sensor or a battery for precautionary measures. The guidelines obtained by the inquiry show that the needs of users constitute their interest in the further development of the system that would be more advanced and capable of more diverse applications in terms of plant care.

## 5.5 Discussion on Results

The testing for the IoT-based indoor plant care system confirms that it has achieved its prospective objectives. The most essential functions, such as automatic watering, light regulation, real-time monitoring, and the possibility of remotely accessing the device through the Arduino IoT Cloud, worked as expected and allowed users to have a comfortable means of taking care of their indoor plants. The system could adjust to variability in the environmental factors, like soil moisture and daylight levels, efficiently proving that it can be used to sustain the plant growth conditions, which remain appropriate over time with minimal human intervention.

During the actual testing period, technical and user-based tests have portrayed a positive performance. Technically, the sensors and actuators were working regularly, and the connection between the ESP32 microcontroller and the Arduino IoT Cloud was stable and working. Performance testing also demonstrated that the system responded quickly to the sensor data, hence the on-time execution of the watering and lighting processes.

The result of user acceptance testing indicated a positive opinion of the participants, with many of them having mentioned that the automatic features were the best thing about the system. The user interface of the mobile app, with which remote monitoring and control of the system is possible, was also commended to be convenient. Such outcomes indicate that it is offering practical value to the system and would likely feel such an aspect is useful even to users who might not be constantly available to ensure their plants are catered to on their own.

Although the testing was successful, it also revealed certain areas which could still be improved. A few of its users encountered minor inaccuracies in the sensor readings, and there is a need to improve calibration or filtration of data. Also, it was proposed to extend the functionality of the system, namely, to ensure the ability to monitor humidity,

introduce additional customisable plant profiles, and make the mobile interface more convenient to work with. They are changes that are made to make the system work well even in varied circumstances and to suit the needs of more people.

Overall, the IoT-based indoor plant care system proves to be a successful prototype. It shows how the power of smart gardening technology could make the work of plant life simple. Nevertheless, the enhancement of features and additional sophistication will be necessary to raise the accuracy and scalability of the system and the degree of user satisfaction. These upgrades could transform the system into a more sophisticated and useful device for indoor gardening in the modern world.

## **5.6 Summary**

The system development lifecycle includes testing as an essential element in the process. IoT-based indoor plant care has been evaluated through a variety of tests that have been carried out. Functional and non-functional tests are the two types that have been adopted in the process of carrying out the evaluation. Functional tests are designed to guarantee that all the system's components accomplish their intended functions. In contrast to non-functional tests, which are designed to evaluate the system's performance, reliability, usability, and user acceptability testing (UAT). During this test, the overall performance of the system, as well as its dependability, usability, and user satisfaction, are evaluated.

## CHAPTER 6: CONCLUSION AND FUTURE WORKS

### 6.1 Introduction

This chapter will portray the overall conclusion of the IoT-based indoor plant care system and discuss the future improvement and development that can be done to improve the system. Moreover, the initial objective in Chapter 1 will be compared with the outcome to ensure that all the proposed objectives are achieved. Besides, system limitations will be discussed in this chapter and about future works that can be applied to the system.

### 6.2 Objectives and Achievements

Specific objectives that have been set in the planning phase in Chapter 1 have been successfully achieved. Although the objectives were successfully achieved, there is still room for improvement or refinement in the system. The primary goal is to optimise indoor plant care and reduce human intervention, especially to help those with busy lifestyles or those who have little knowledge of plant care, by focusing on sensor readings as well as fuzzy logic integration for automated watering.

With the integration of ESP32 microcontrollers, sensors, and Arduino IoT Cloud, the system aims to address the issues of indoor plant care with minimal human intervention to optimise the plant growth. The following table summarises the objectives and their corresponding achievements:

*Table 6.1* Details of Objectives and Achievement

| Objectives                                                                                                                                               | Achievement                                                                                                                                                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| To monitor environmental factors such as soil moisture, temperature levels, and light intensity using sensors to gather real-time data for plant health. | <b>Soil Sensor:</b> Implemented and integrated successfully with ESP32 and Arduino IoT Cloud<br><b>DHT22 (Temperature Sensor):</b> Implemented and integrated successfully with ESP32 and Arduino IoT Cloud<br><b>LDR Sensor:</b> Implemented and integrated successfully with ESP32 and Arduino IoT Cloud |

|                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                              | <p><b>Real-time data:</b> Implemented successfully and display accurate data on Arduino IoT Cloud</p> <p><b>Challenges:</b> Testing becomes quite challenging when the Internet connection is unstable. To ensure the real-time data is accurate, make sure the Internet connection is stable.</p>                                           |
| To implement fuzzy logic in the watering system to analyse soil moisture levels and temperature, making adaptive decisions to ensure plants receive the right amount of water based on real-time conditions. | <p><b>Fuzzy Logic:</b> Implemented successfully using a mathematical formula to analyse the amount of water needed for a particular plant based on soil moisture and surrounding temperature.</p> <p><b>Challenges:</b> While the logic works well, further optimisation is needed to accurately analyse water needed for indoor plants.</p> |
| To automate the watering system based on fuzzy logic-driven analysis, adjusting watering duration and frequency including profile-based configuration watering system based on different plant's needs.      | <p><b>Automate the watering system based on fuzzy logic-driven analysis:</b> Implemented successfully based on plant profile-based configuration.</p> <p><b>Plant profile-based configuration watering system:</b> Implemented successfully for automated watering system.</p>                                                               |
| To automate lighting based on light sensor.                                                                                                                                                                  | <p><b>Automated lighting:</b> Implemented successfully based on LDR sensor reading.</p>                                                                                                                                                                                                                                                      |

This system also includes several additional features which are mainly on mobile applications. The table below shows the details of the following additional features:

*Table 6.2 Additional features of the Proposed System*

| Objective                                       | Description                                                                                                   | Achievement                                                                                                                        |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Ability to view the lighting and watering state | Allow the user to view the state of the current system, especially the on/off state of watering and lighting. | Successfully integrated on mobile application.                                                                                     |
| Ability to control watering manually            | Allow the user to have control with watering, especially if they prefer to control it manually.               | Successfully integrated into the mobile application, making the user interact with the system, thus not limiting the user control. |

## **Discussion on Achievements**

The IoT-based Indoor Plant Care system successfully achieves its primary goals by incorporating ESP32 microcontrollers, sensors, and Arduino IoT Cloud, as well as automated lighting and fuzzy logic for automated watering. Creating this system in hope of tackling the problems that encounter individuals who have hectic lifestyles and do not have time to take care of their indoor plants and to those who love indoor plants but have limited knowledge of indoor plant care. Furthermore, the additional features significantly enhance the system and mobile app functionality, making plant care more efficient and user-friendly.

However, further optimisation is needed for fuzzy logic and also internet connectivity, as real-time data will inaccurately display if the internet connection is unstable, to improve the system's accuracy and reliability. Overall, all of the objectives were successfully met, and the system achieved its intended functionality. The proposed system is robust and user-friendly due to the combination of real-time monitoring and intelligent decision-making with fuzzy logic, automation, and user-controlled control.

### **6.3 Project Limitation and Constraint**

Despite achieving all the intended objectives, this system has several significant limitations that need to be addressed before it can be considered for real-world deployment. These limitations include technical, hardware, and software issues.

#### **1. Internet Dependency**

ESP32 and Arduino IoT Cloud need a constant Internet connection to ensure data transmission and system responsiveness. Inability to have a stable Internet connection makes the ESP32 fail to send data to the cloud. As a result, real-time data displayed on the user interface would either fail to update, display inaccurate data, or disconnect

entirely. This reduces the reliability of the monitoring system and may lead to faulty decisions in plant care, particularly when the system is unable to respond automatically to water and lighting due to inaccurate sensor input. This dependency can actually have a huge impact on the total performance of the system in practical use, particularly on rural or indoor sites that do not broadly cover with Wi-Fi.

## 2. Limited Processing Power

The ESP32 microcontroller is efficient and suitable for IoT applications. However, it has limited processing and memory capabilities. It becomes a challenge when it needs to manage multiple tasks such as reading sensor data, maintaining stable cloud connections, reacting to user input, etc. These limitations constrained the implementation of multitasking or more advanced algorithms, as it would exceed the ESP32 capabilities.

## 3. Sensor Accuracy

As the current sensor used is a low-cost sensor, it did not guarantee an accurate reading, as it is only sufficient for basic monitoring. These low-cost sensors are prone to inaccuracy and inconsistency in sensor readings over time. These inconsistencies and inaccuracies may lead to incorrect watering or lighting decisions, especially since this system required real-time data to activate automation.

## 4. Data Latency

Data latency is a time difference between sensor readings, data transmissions, and the display of a cloud or application interface. The speed of data transmission from ESP32 to the Arduino IoT Cloud and the frequency of the app's ability to update the data will define the real-time performance of the system. In areas with slow and unstable internet services, the user might notice delayed data as a result of increased latency. The system

may become less responsive when prompt action, such as automated watering, is required due to such delay.

## 5. Power Supply Dependence

The ESP32 microcontroller needs a continuous and reliable power supply in order for it to function well. In the current system architecture, the ESP32 microcontroller is connected to the laptop via USB cable to receive continuous and sufficient power supply for it to operate. If a power interruption happens, it will shut down the entire system and eventually disturb the real-time monitoring process.

## 6. Fuzzy Logic Precision

The fuzzy logic implemented in the system is a manually set rule and formula to analyse the appropriate soil moisture and temperature based on plant profile-based settings. Although it is successful, it still needs refinement, as it may lack precision and adaptability to handle various plant types or environmental conditions. Manual fuzzy rules refinement is difficult as the number of types of plants or environmental situations grows. The data-driven optimisation is important, as otherwise, the system can either overwater or submerge the plant when the sensor readings are close to the threshold bounds of the fuzzy sets.

## 6.4 Future Works

As the development of the IoT-based indoor plant care system continues to develop, a few areas for the future need to be identified to enhance system usability, reliability and user experience. Enhance this area, not only making this system powerful but also able to expand the system usability and reliability for real-world scenarios.

### 1. Improve Network Reliability

One of the limitations of IoT-based indoor plant care is continuous Internet dependency. Future works will focus on optimising network reliability, which includes:

- Implement an offline function so that the system can operate locally without dependencies on the Internet. This can enable the system to continuously operate when Wi-Fi is unavailable and automatically sync data when reconnected.
- Include a backup connectivity option like Bluetooth or LoRa when the Wi-Fi connection is weak or unavailable.

### 2. Upgrade hardware

Enhancing the hardware of the system is also crucial for system success. This future work includes:

- Upgrading to a more powerful microcontroller, such as Raspberry Pi, to support more complex operations like AI-based or machine learning
- Optimising the existing code to reduce memory usage and CPU load by simplifying logic

### 3. Use Industrial-Grade Sensors

Low-cost sensors are useful for basic monitoring. However, it often suffers from inaccurate sensor readings and a shorter lifespan. Upgrading the sensor is also crucial, as it is able

to improve the sensor's reading or data accuracy. The following shows how sensor reliability and precision in future works can be done:

- Use industrial-grade sensors like SHT31 temperature sensors, which are more accurate than DHT22 and BH1750 light sensors, for more accurate light intensity reading.
- Use redundant sensors, especially for soil moisture sensors. This basically uses two soil moisture sensors in one indoor plant pot and computes the average reading, which also can improve system reliability.

#### 4. Use Smarter Data Handling Techniques

Currently, the system experiences delays in sensor updates or command responses due to data latency. To improve this, future works will involve:

- Implementing local data caching, where the reading of ESP32 can be stored locally when offline, such as using an SD card. Once reconnected to the internet, the system can push the cached data to the cloud.

#### 5. Upgrade Power Supply Options

To ensure the IoT-based indoor plant care operates continuously, a few solutions can be done. This includes:

- Integrate a backup battery system where the ESP32 can seamlessly switch to battery during power outages.
- Integrating solar power with battery storage enables off-grid operations, such as those in remote setups. This alternative promotes energy efficiency and environmental sustainability.

## 6. Integrate Machine Learning

The existing fuzzy logic system is using pre-determined sets of rules that make decisions on watering manually established. Although among the successful in the case of the general condition, it lacks versatility. The future works include:

- Integrate the system with machine learning where the train models like TinyML can learn about the scheme of plant watering and relations with the environment. Such models are capable of modifying watering behaviours dynamically, relying upon historical data and user feedback.

### 6.5 Conclusion

The IoT-based indoor plant care shows success in achieving the promised objectives of monitoring environmental factors, implementing fuzzy logic in the automated watering system, plant profile-based configuration for watering automation and automated lighting. By focusing on enhancing technical, hardware, and software limitations, the system can become more robust, reliable, and user-friendly. This enhancement not only benefits system performance but also helps indoor plant enthusiasts in taking care of their plants with minimal human intervention.

## REFERENCES

- Andprof. (2024). *What is Arduino software (IDE), and how use it? - AndProf*. AndProf.  
<https://andprof.com/tutorials/what-is-arduino-software-ide-and-how-use-it/>
- Arduino. (2024). *UNO R3 | Arduino Documentation*. Docs.arduino.cc.  
<https://docs.arduino.cc/hardware/uno-rev3/>
- Asana. (2024, February 2). *What Is Agile Methodology? (A Beginner's Guide)* • Asana.  
Asana. <https://asana.com/id/resources/agile-methodology>
- Blynk. (2015). *Blynk*. Blynk.io. <https://blynk.io/>
- Chatakul, P., & Janpathompong, S. (2022). Interior plants: Trends, species, and their benefits. *Building and Environment*, 222(1), 109325.  
<https://doi.org/10.1016/j.buildenv.2022.109325>
- Choudhari, G. R., Dagale, P. A., Dashetwar, I. S., Desai, R. R., & Marathe, A. A. (2023). IoT-based Smart Gardening System. *Journal of Physics. Conference Series*, 2601(1), 012006–012006. <https://doi.org/10.1088/1742-6596/2601/1/012006>
- Coral (2025). What Temperature is Too Hot for Succulents? A Guide to Optimal Temperature Conditions - louisheplantgeek.com. *louisheplantgeek.com*.  
<https://louisheplantgeek.com/what-temperature-is-too-hot-for-succulents/>
- ESP32 - Soil Moisture Sensor. (n.d.). ESP32 Tutorial. <https://esp32io.com/tutorials/esp32-soil-moisture-sensor>
- Figma. (2023). *What Is a Use Case? How To Write One, Examples, + Template* | Figma.  
Figma. <https://www.figma.com/resource-library/what-is-a-use-case/>
- GeeksforGeeks. (2018). *Fuzzy Logic | Introduction*. GeeksforGeeks.  
<https://www.geeksforgeeks.org/fuzzy-logic-introduction/>

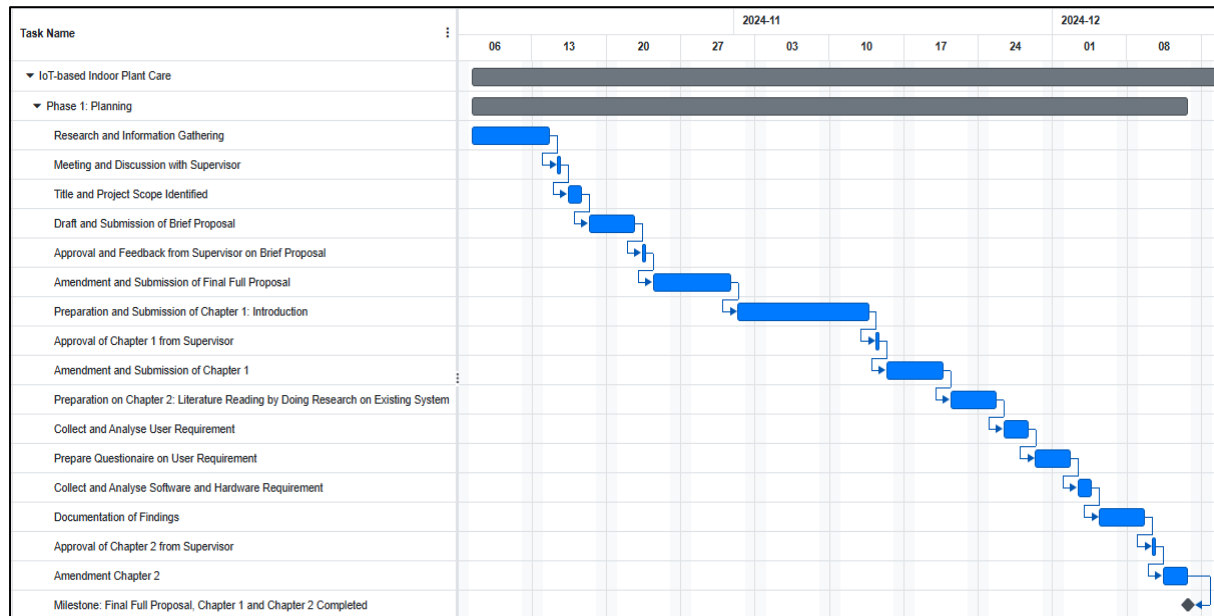
- Geeksforgeeks. (2023). *What is Agile Methodology?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-agile-methodology/>
- Getting Started with Arduino Cloud.* (2024). Arduino.cc. <https://docs.arduino.cc/arduino-cloud/guides/overview/>
- Getting Started with ESP8266 WiFi Transceiver (Review) | Random Nerd Tutorials.* (2018). Random Nerd Tutorial. <https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/>
- Ideal Temperature for Plant Growth: Ultimate Guide [Solved].* (2024). Farmtechx.com. <https://farmtechx.com/blog/ideal-temperature-for-plant-growth-guide>
- Manzur, D. (n.d.). *Temperature range for house plants: A Complete guide.* <https://idealindoor.com/temperature-range-for-house-plants/>
- NodeMCU ESP8266 Specifications, Overview and Setting Up.* (n.d.). Make-It.ca. <https://www.make-it.ca/nodemcu-details-specifications/>
- Setiawan, Y. D., Hartanto, W., Lukas, E. E., Don, N., Kurniawan, S., & Siswanto, B. (2023). Smart Plant Watering and Lighting System to Enhance Plant Growth Using Internet of Things. *Procedia Computer Science*, 227, 966–972. <https://doi.org/10.1016/j.procs.2023.10.604>
- Shamsuri, M. M. S., Leman, A. M. (2015) Indoor Plants as a Filtration of Indoor Air Pollution: A Review. *Journal of Occupational Safety and Health*, 12(2), 41-58.
- Teja, R. (2021, February 17). *Introduction to ESP32 | Specifications, ESP32 DevKit Board, Layout,.* Electronics Hub. <https://www.electronicshub.org/getting-started-with-esp32/>

Ulloa, G. G., García, A. M., Lindao, V. T., Mecías, V. Z., Domínguez, C. R., & Hornos, M. J. (2023). Internet of Things (IoT)-based indoor plant care system. *Journal of Ambient Intelligence and Smart Environments*, 15(1), 47–62.  
<https://doi.org/10.3233/AIS-220483>

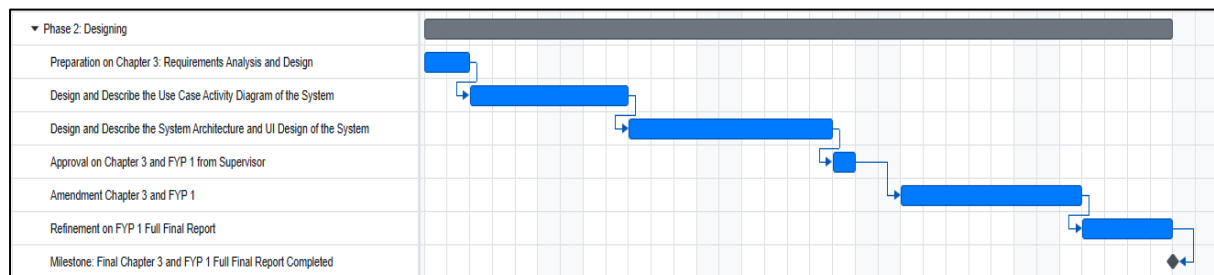
## APPENDICES

### Appendix A Project Schedule (Gantt Chart)

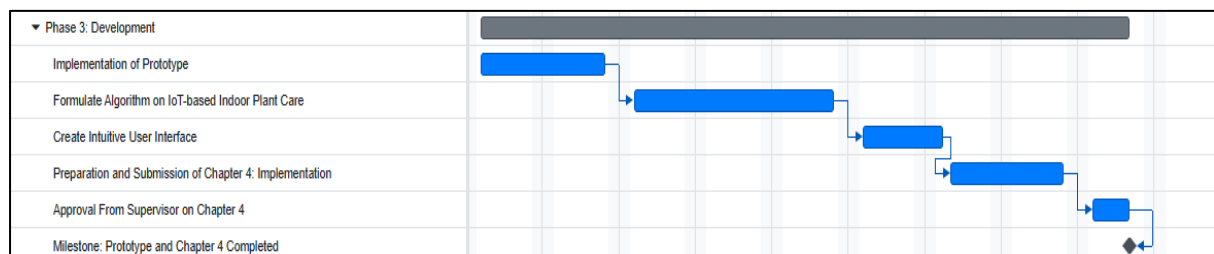
#### Phase 1: Planning



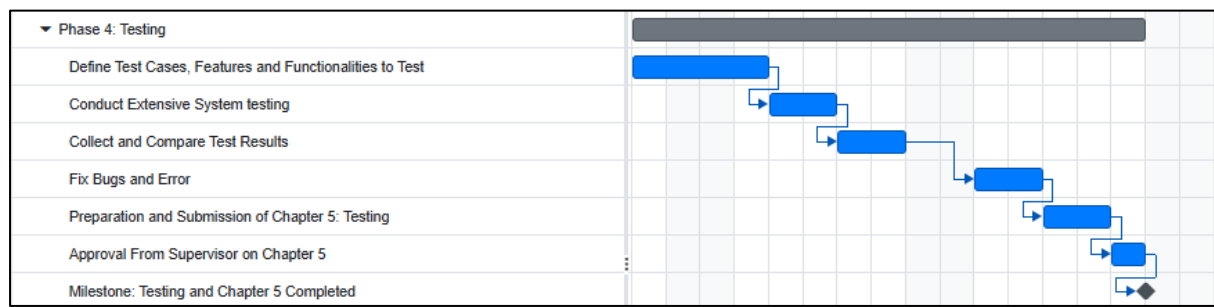
#### Phase 2: Designing



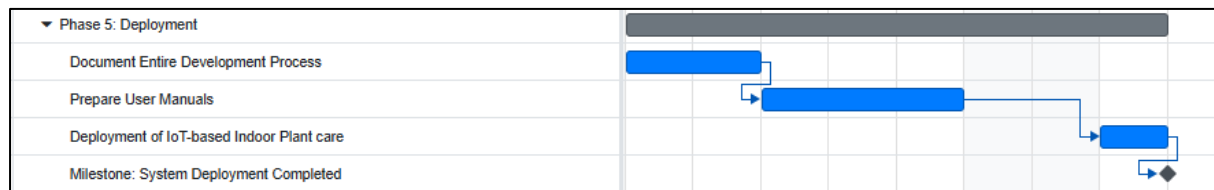
#### Phase 3: Development



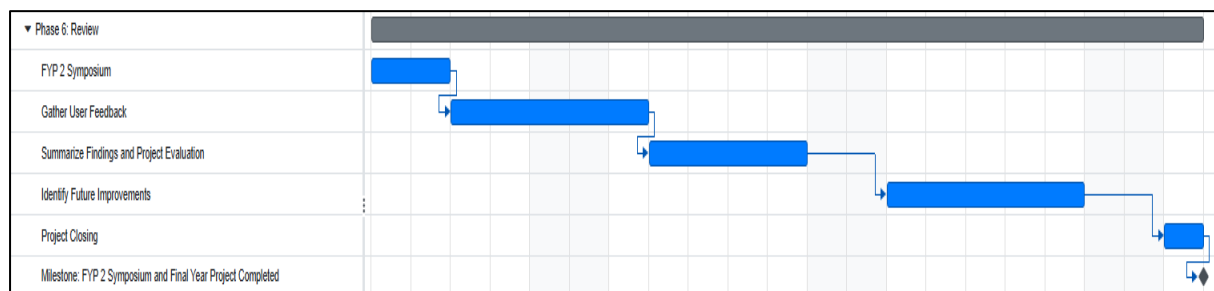
## Phase 4: Testing



## Phase 5: Deployment



## Phase 6: Review



## Appendix B

### Sample Questionnaire



### Final Year Project: IoT-based Indoor Plant Care

Hello! 🌱

I'm Rofinna Ellya Embang Anak Umar @ Richard (81056), final year student from Bachelor of Computer Science (Network Computing) from Universiti Malaysia Sarawak (UNIMAS). I am developing an

IoT-based Indoor Plant Care mobile application that helps plant enthusiasts monitor and care for their indoor plants efficiently.

This survey aims to understand your preferences and expectations for the app's design, layout, and features. It will take approximately **5-10 minutes** to complete. Your responses will remain anonymous and will only be used for this project.

By participating, you will help shape an app tailored to make plant care easy, enjoyable, and accessible. Let's grow together! 🌱

\* Indicates required question

Age \*

☐ Under 18

☐ 18-24

☐ 25-34

☐ 35 and above

What is your gender? \*

☐ Male

☐ Female

☐ Prefer not to disclose

What is your occupation? \*

☐ Student

☐ Professional (e.g., office worker, engineer)

☐ Unemployed

☐ Retired

☐ Other

Do you currently own or care for indoor plants? \*

☐ Yes

☐ No

How would you describe your experience with plant care? \*

☐ Beginner (Just starting with plants)

☐ Option Intermediate (Have some experience)

☐ Expert (Highly experienced with plant care)

Do you currently use any mobile apps for plant care or gardening? \*

☐ Yes

☐ No

Understanding User Preferences

This section gathers insights into your preferences for layout, navigation, and overall usability to help design an app that aligns with your needs.

What type of layout do you prefer for a mobile app? \*



☐ List-based (vertical scrolling)



☐ Tabbed navigation



☐ Grid-based (tiles/icons)

What color schemes do you find appealing for an app related to plant care? \*



☐ Green and earthy tones



☐ Bright and vibrant colors



☐ Minimalist (black, white, gray)

☐ Other: \_\_\_\_\_

Do you prefer icons or text for buttons and navigation? \*

☐ Icons only

☐ Text only

☐ Both icons and text

What's more important for you in an app? \*

☐ Aesthetic design (how it looks)

☐ Functionality and ease of use (how it works)

☐ Both equally

What frustrates you most about UI design in mobile apps? (Select all that apply) \*

☐ Too many options cluttering the screen

☐ Poor organization of features

☐ Hard-to-read text or small buttons

☐ Other: \_\_\_\_\_

### Key Features and Aesthetic Preferences

This section explores the features and visual elements you value most, ensuring the app is both functional and visually appealing.

What information would you like to see on the main screen of the app? (Select all <sup>\*</sup> that apply)

- ☐ Soil moisture level
- ☐ Temperature and humidity readings
- ☐ Plant health status (e.g., alerts)
- ☐ Watering controls
- ☐ Lighting controls

How would you like to receive plant health alerts? <sup>\*</sup>

- ☐ Push notifications
- ☐ In-app alerts only
- ☐ Both

Would you prefer a light theme, dark theme, or both? <sup>\*</sup>

- ☐ Light theme
- ☐ Dark theme

What font style do you prefer for reading on mobile apps? <sup>\*</sup>

- ☐ Modern (e.g., sans-serif)
- ☐ Classic (e.g., serif)
- ☐ Playful (e.g., decorative fonts)
- ☐ No preference

## Appendix C

### Full Coding IoT-based Indoor Plant Care

#### Main code:

```
#include <WiFi.h>
#include "thingProperties.h"
#include <DHT.h>

// sensor & pin
#define DHTPIN 27          // DHT22 connected to GPIO 27
#define DHTTYPE DHT22
#define SOIL_PIN 34
#define RELAY_WATER_PIN 25

DHT dht(DHTPIN, DHTTYPE);

unsigned long lastWaterTime = 0;
const unsigned long wateringCooldown = 5 * 60 * 1000;
bool isWatering = false;
unsigned long wateringStartTime = 0;
int wateringDuration = 0;

void setup() {
  Serial.begin(115200);
  delay(3000);

  initProperties(); // Initialize Cloud variables
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  pinMode(SOIL_PIN, INPUT);
  pinMode(RELAY_WATER_PIN, OUTPUT);
  digitalWrite(RELAY_WATER_PIN, LOW);

  dht.begin();

  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  delay(3000);
  ArduinoCloud.update();
  readSensors();
  handleAlerts();
  handleWatering();
  updatePlantCondition();
}

void readSensors() {
  // --- Soil Moisture ---
  soilmoisture = (1.0 - analogRead(SOIL_PIN) / 4095.0) * 100.0;

  // --- Temperature ---
  float tempReading = dht.readTemperature();
  if (!isnan(tempReading)) {
    temp = tempReading;
    Serial.print("Temp: ");
    Serial.print(temp);
    Serial.print(" °C | ");
  } else {
    Serial.println("Failed to read temperature from DHT22.");
  }

  Serial.print("Soil: ");
  Serial.print(soilmoisture);
  Serial.println(" %");
}

void updatePlantCondition() {
```

```

float score = 100.0;
float idealTemp = 0.0;
float idealSoil = 0.0;

if (plantType == 0) { // Cactus
    idealTemp = 30.0;
    idealSoil = 25.0;
} else if (plantType == 1) { // Tropical
    idealTemp = 27.0;
    idealSoil = 50.0;
} else if (plantType == 2) { // Flowering
    idealTemp = 24.0;
    idealSoil = 65.0;
} else {
    idealTemp = 25.0;
    idealSoil = 45.0;
}

float tempPenalty = abs(temp - idealTemp) * 2.0;
float soilPenalty = abs(soilmoisture - idealSoil);
float totalPenalty = tempPenalty + soilPenalty;

score -= totalPenalty;
if (score < 0) score = 0;
if (score > 100) score = 100;

plantCondition = score;

Serial.print("Plant Type: ");
if (plantType == 0) Serial.print("Cactus | ");
else if (plantType == 1) Serial.print("Tropical | ");
else if (plantType == 2) Serial.print("Flowering | ");
else Serial.print("Unknown | ");

Serial.print("Temp: ");
Serial.print(temp);
Serial.print(" °C | Soil: ");
Serial.print(soilmoisture);
Serial.print(" % => Condition: ");
Serial.print(plantCondition);
Serial.println(" %");
}

void handleAlerts() {
    alertTemp = temp > 35;
    alertLight = lightLevel < 30;
    alertSoil = soilmoisture < getMoistureThreshold(plantType);

    alertMessage = "";

    if (alertTemp) alertMessage += "Temperature too high! ";
    if (alertSoil) alertMessage += "Soil is too dry!";
    if (!alertTemp && !alertLight && !alertSoil) alertMessage = "Plant is doing fine.";
}

void handleWatering(){
    if (manualWater){
        wateringDuration=2000;
        isWatering = true;
        wateringStartTime = millis();
        manualWater = false;
        wateringState = true;
    }
    else if (autoWater && millis() - lastWaterTime > wateringCooldown && !isWatering){
        float threshold = getMoistureThreshold(plantType);
        wateringLevel = applyFuzzyLogic(temp, soilmoisture, plantType);
    }
}

```

```

        if (soilmoisture < threshold && wateringLevel > 0) {
            wateringDuration = (int)(wateringLevel * 1000);
            isWatering = true;
            wateringStartTime = millis();
            lastWaterTime = millis();
            wateringState = true;
            Serial.println("Watering completed.");
        }
    }
    if (isWatering) {
        digitalWrite(RELAY_WATER_PIN, HIGH);
        if (millis() - wateringStartTime >= wateringDuration) {
            digitalWrite(RELAY_WATER_PIN, LOW);
            isWatering = false;
            wateringState = false;
            Serial.println("Watering completed.");
        }
    } else {
        digitalWrite(RELAY_WATER_PIN, LOW);
    }
}

float applyFuzzyLogic(float temp, float soilmoisture, int plantType) {
    float base = 0;
    if (plantType == 0) base = 0.5;
    else if (plantType == 1) base = 1.0;
    else if (plantType == 2) base = 1.5;

    float tempFactor = (temp > 30) ? 0.5 : 0.2;
    float moistureFactor = (soilmoisture < 40) ? 1.0 : 0.5;

    return base * tempFactor * moistureFactor;
}

float getMoistureThreshold(int plantType) {
    if (plantType == 0) return 20.0;
    if (plantType == 1) return 40.0;
    if (plantType == 2) return 60.0;
    return 40.0;
}

// === CALLBACKS ===

void onManualWaterChange() {
}

void onAlertMessageChange() { Serial.println("Alert message updated: " + alertMessage); }
void onLightLevelChange() {}
void onSoilmoistureChange() {}
void onTempChange() {}
void onWateringLevelChange() {}
void onPlantTypeChange() {}
void onAlertLightChange() {}
void onAlertSoilChange() {}
void onAlertTempChange() {}
void onAutoWaterChange() {}
void onGrowLightAutoChange() {}
void onPlantConditionChange() {}
void onManualGrowLightChange() {
    Serial.print("Manual Grow Light changed to: ");
    Serial.println(manualGrowLight);
}

void onWateringStateChange() {
    // Optional: log cloud change of state
    Serial.print("Watering state changed from Cloud: ");
    Serial.println(wateringState);
}

```

## LDR and LED code:

```
#include "thingProperties.h"

#define LDR_PIN 35      // Connect LDR to analog pin 35
#define LED_PIN 25      // Connect LED to digital pin 25

void setup() {
  Serial.begin(115200);
  delay(1500);

  pinMode(LDR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
}

void loop() {
  ArduinoCloud.update();

  // Read LDR sensor
  int rawValue = analogRead(LDR_PIN);
  lightLevel = rawValue;

  // Print the raw value to Serial Monitor
  Serial.print("LDR Value: ");
  Serial.println(rawValue);

  // Simple threshold to control LED
  if (rawValue > 1000) { // Dark environment
    digitalWrite(LED_PIN, HIGH);
    growLightstate = true;
    Serial.println("LED ON");
  } else { // Bright environment
    digitalWrite(LED_PIN, LOW);

    growLightstate = false;
    Serial.println("LED OFF");
  }

  delay(1000);
}

// Add empty callback definitions to fix compile error
void onLightLevelChange() {}
void onTempChange() {}
void onGrowLightAutoChange() {}
void onGrowLightstateChange() {}
void onManualGrowLightChange() {}
```

## Appendix D

### Sample Questionnaire of User Acceptance

### User Acceptance Survey for IoT-based Indoor Plant Care

Hello! 🌱

I'm Rofinna Ellya Embang Anak Umar @ Richard (81056), final year student from Bachelor of Computer Science (Network Computing) from Universiti Malaysia Sarawak (UNIMAS).

This survey is designed to gather feedback from users regarding the usability, functionality, and overall experience of the IoT-based Indoor Plant Care system. The goal is to assess how well the system meets user expectations in terms of ease of use, feature accessibility, and satisfaction with automated plant care features such as watering, lighting, and monitoring. Responses will help improve system design, interface, and reliability for future use.

rofinnaellya@gmail.com [Switch accounts](#)

🔒 Not shared

\* Indicates required question

Gender \*

☐ Female

☐ Male

Age \*

Your answer

#### Section A: General Satisfaction

How satisfied are you with the overall performance of the plant care system? \*

☐ Very satisfied

☐ Satisfied

☐ Neutral

☐ Dissatisfied

☐ Very dissatisfied

Does the system meet your expectations for monitoring and automating plant care? \*

☐ Yes

☐ No

#### Section B: Usability

How easy was it to use the mobile app interface? \*

- ☐ Very Easy  
☐ Easy  
☐ Neutral  
☐ Difficult  
☐ Very Difficult

Were you able to easily find and use the automatic watering control? \*

- ☐ Yes  
☐ No

How clear were the status indicators (e.g., ON/OFF, soil moisture levels, lighting)? \*

- ☐ Very Clear  
☐ Clear  
☐ Neutral  
☐ Unclear  
☐ Very Unclear

#### Section C: Usefulness of Features

How useful is the automatic watering feature for your plant care needs? \*

- Very Useful      1      2      3      4      5      Not Applicable
- ☐      ☐      ☐      ☐      ☐

Did the lighting automation respond appropriately to low light conditions? \*

- ☐ Always  
☐ Often  
☐ Sometimes  
☐ Rarely  
☐ Never

Was the health score helpful in assessing your plant's condition? \*

- ☐ Yes  
☐ No

#### Section D: Reliability

Did the system work without crashes or connection problems during use? \*

- ☐ Always  
☐ Often  
☐ Sometimes  
☐ Rarely  
☐ Never

Was data (moisture, temperature, light) updated in real time as expected? \*

- ☐ Yes  
☐ No

#### Section E: Feedback

What features did you like the most? \*

Your answer \_\_\_\_\_

What improvements would you suggest for the system? \*

Your answer \_\_\_\_\_