

AUTOMATIC MUSHROOM HOUSE
MONITORING SYSTEM (MHMS) USING
INTERNET OF THINGS (IoT)

MUHAMMAD SYAFIK IKMAL BIN
NAZARUDDIN
CA20019

BACHELOR OF COMPUTER SCIENCE
(COMPUTER SYSTEM AND
NETWORKING)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MUHAMMAD SYAFIK IKMAL BIN NAZARUDDIN

Date of Birth

Title : AUTOMATIC MUSHROOM HOUSE MONITORING SYSTEM
(MHMS) USING INTERNET OF THINGS (IoT)

Academic Session : 2022/2023

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

New IC/Passport Number
Date:02 DECEMBER 2022

(Supervisor's Signature)

TS DR NOR SYAHIDATUL NADIAH BINTI
ISMAIL

Name of Supervisor
Date:26 JULY 2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Doctor of Philosophy/ Master of Engineering/ Master of Science in

(Supervisor's Signature)

Full Name : TS DR NOR SYAHIDATUL NADIAH BINTI ISMAIL

Position : SENIOR LECTURE

Date :

(Co-supervisor's Signature)

Full Name :

Position :

Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

(Student's Signature)

Full Name : MUHAMMAD SYAFIK IKMAL BIN NAZARUDDIN

ID Number : CA20019

Date : 02 DECEMBER 2022

AOTOMATIC MUSHROOM HOUSE MONITORING SYSTEM (MHMS) USING
INTERNET OF THINGS (IoT)

MUHAMMAD SYAFIK IKAL BIN NAZARUDDIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of Bachelor of Computer Science (Computer System and
Networking) with honor

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JUNE 2023

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Dr Nor Syahidatul Nadiah Binti Ismail, for her guidance and support in completing my project. The supervision and suggestions pointed me in the right direction in terms of idea generation. It also allowed me to broaden my knowledge and experience while working on my final year project. Next, I'd like to thank my parents and friends for their assistance in coming up with project ideas. Furthermore, my friends contribute to knowledge sharing, which greatly assists me in completing the proposal.

Finally, I'd like to express my appreciation to Dr. Danakorn, the project coordinator, for coordinating my project and providing me with guidance and encouragement to complete the proposal. Furthermore, the proposal writing guidance allows me to understand the proper format for writing the proposal.

ABSTRAK

Industri cendawan Malaysia sedang mengalami lonjakan dalam permintaan domestik dan antarabangsa, mewujudkan peluang pengembangan yang ketara. Kerajaan Malaysia menyokong penanaman cendawan sebagai komoditi bernilai tinggi, mengiktiraf potensinya. Peningkatan kawasan penanaman cendawan daripada 78 hektar pada 2010 kepada 340 hektar pada 2020 mencerminkan peningkatan permintaan. Untuk memanfaatkan peluang ini, sistem pemantauan rumah cendawan automatik (MHMS) berasaskan Internet of Things (IoT) dicadangkan. Projek ini menggabungkan perkakasan, seperti penderia DHT22, mikropengawal Node MCU esp8266, kipas dan pam air, untuk mengawal suhu dan kelembapan dalam rumah cendawan. Geganti dengan saluran dwi mengawal selia kipas dan pam air. Proses pembangunan perisian melibatkan pembinaan aplikasi mudah alih pintar yang memantau suhu, kelembapan dan operasi pam ekzos dan air. Dengan penderia dan aplikasi telefon pintar untuk pemantauan masa nyata, penanam boleh mengoptimumkan keadaan pertumbuhan dan meningkatkan proses penanaman. Sistem pemantauan alam sekitar berasaskan IoT memenuhi keperluan khusus industri cendawan Malaysia dengan memantau faktor iklim kritikal dalam rumah cendawan. Metodologi pembangunan tangkas digunakan kerana demonstrasi keupayaan pantas dan kecekapan sumber. Dengan mengawal selia suhu dan kelembapan, sistem berharap dapat mengekalkan kualiti tanaman dan meningkatkan kecekapan. Industri cendawan di Malaysia mempunyai potensi pertumbuhan yang ketara, yang boleh dieksploitasi dengan sokongan kerajaan dan pelaksanaan teknologi inovatif.

ABSTRACT

Malaysia's mushroom industry is experiencing a surge in both domestic and international demand, creating significant expansion opportunities. The Malaysian government supports mushroom cultivation as a high-value commodity, recognizing its potential. The rise in mushroom cultivation areas from 78 hectares in 2010 to 340 hectares in 2020 reflects the increasing demand. To capitalize on this opportunity, an Internet of Things (IoT)-based automatic mushroom house monitoring system (MHMS) is proposed. This project incorporates hardware, such as the DHT22 sensor, Node MCU esp8266 microcontroller, fan, and water pump, to regulate the temperature and humidity within the mushroom house. A relay with dual channels regulates the fan and water pump. The software development process involves the construction of smart mobile applications that monitor the temperature, humidity, and operation of the exhaust and water pumps. With sensors and smartphone applications for real-time monitoring, growers can optimize growing conditions and enhance the cultivation process. The IoT-based environmental monitoring system meets the specific requirements of Malaysia's mushroom industry by monitoring critical climatic factors within mushroom houses. The agile development methodology is utilized because of its rapid capability demonstration and resource efficiency. By regulating temperature and humidity, the system hopes to preserve crop quality and boost efficiency. The mushroom industry in Malaysia has significant growth potential, which can be exploited with government support and the implementation of innovative technology.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	ii
ABSTRAK	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF SYMBOLS	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Scope	3
1.5 Thesis Organization	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Existing Work	5
2.1.1 Automatic Climate Control for Mushroom Cultivation Using IoT approach. (System 1)	5
2.1.2 Green technology monitoring system for oyster mushroom. (System 2)	9

2.1.3	Mushroom House Monitoring System Using Internet of Things (IoT), (System 3)	11
2.2	Analysis of comparison on existing system	15
2.3	Relevance of Comparison with project tittle	18
2.4	Summary	18
CHAPTER 3		19
METHODOLOGY		19
3.1	Introduction	19
3.2	Project Methodology	19
3.2.1	Agile Model	19
3.2.2	Iteration Model	20
3.2.3	Waterfall Model	20
3.2.4	Model for Proposed Project	21
3.3	Project Requirement	23
3.3.1	Functional Requirement	23
3.3.2	Non-Functional	23
3.3.3	Limitation	23
3.3.4	Constrain	23
3.3.5	User Requirement	24
3.4	Propose Design	25
3.4.1	Architecture Design	26
3.4.2	Flowchart	27
3.4.3	Context diagram	30
3.4.4	Use case diagram.	30
3.4.5	Activity diagram	32

3.5	Data Design	34
3.5.1	ERD	34
3.5.2	Data base dictionary	34
3.6	Hardware and software	35
3.6.1	Software	35
3.6.2	Hardware	37
3.7	Proof of Initial Concept	42
3.7.1	Smart Mobile Application.	42
3.7.2	Hardware setup	46
3.8	Testing /Validation Plan	47
3.9	Potential use of Proposed Solution	48
CHAPTER 4 RESULT AND DISCUSSION		50
4.1	SMART MOBILE APPLICATION OF AUTOMATIC MUSHROOM HOUSE MONITORING SYSTEM (MHMS)	50
4.1.1	Login Interface	50
4.1.2	Register Interface	51
4.1.3	Main interface	51
4.1.4	Real time monitoring interface	52
4.1.5	Graph Report	53
4.2	Implementation	55
4.2.1	Authentication	55
4.2.2	Real Time Database	56
4.2.3	Smart mobile application source code	58
4.2.4	Hardware implementation	63
4.3	Testing and Discussion	69

4.3.1	Automatic mushroom house monitoring system (MHMS) authentication	69
4.3.2	Esp8066 connection to real time database	69
4.4	Conclusion	71
CHAPTER 5 CONCLUSION		72
5.1	Constraints	72
5.1.1	Hardware Constraints	72
5.1.2	Time Constraints	72
5.1.3	Cost Constraints	73
5.1.4	Technical Constraints	73
5.2	Future Works	73
REFERENCE		74
APPENDIX A GANTT CHART		75
APPENDIX B		76
User Acceptance test (UAT)		76

LIST OF TABLES

Table 2.1 Comparison of Methods among Past Research Papers.	15
Table 3.1 Threshold value for temperature and humidity	25
Table 3.2 Register Data Dictionary	34
Table 3.3 login Data Dictionary	35
Table 3.4 Real time Data Dictionary	35

LIST OF FIGURES

Figure 1.1 Flow for PSM 1	4
Figure 1.2 Flow for PSM 2	4
Figure 2.1 System Schematic Diagram	6
Figure 2.2 System Operation Flowchart	8
Figure 2.3 System web interface	9
Figure 2.4 solar PV Power Supply Block Diagram	10
Figure 2.5 The block diagram for the oyster mushroom monitoring system using green technology.	10
Figure 2.6 Display of the Blynk apps	11
Figure 2.7 Concept diagram of the Mushroom House monitoring	13
Figure 2.8 Flowchart of the System	14
Figure 2.9 Dashboard show latest update real data monitoring.	15
Figure 3.1 Agile model phase	21
Figure 3.2 illustration of the purpose design	25
Figure 3.3 hardware connection of purpose design	26
Figure 3.4 Architecture Design	27
Figure 3.5 hardware Flowchart	28
Figure 3.6 System flowchart	29
Figure 3.7 System context Diagram	30
Figure 3.8 system Context Diagram	31
Figure 3.9 Hardware activity diagram	32
Figure 3.10 System Activity Diagram	33
Figure 3.11 ERD diagram	34
Figure 3.12 Arduino IDE software	36
Figure 3.13 Android Studio Dolphin 2021.3.1	37
Figure 3.14 Firebase Logo	37
Figure 3.15 NodeMCU 8266	38
Figure 3.16 DHT22 sensor	39
Figure 3.17 Micro Submersible Water Pump	40
Figure 3.18 DC Cooling Exhaust Fan	41
Figure 3.19 One channel relay	41
Figure 3.20 Login page	42
Figure 3.21 Register Page	43

Figure 3.22 Main page	44
Figure 3.23 Real Time Monitoring Page	45
Figure 3.24 Graph /Report page	46
Figure 3.25 Hardware setup of Proof and initial concept	47
Figure 4.1 login interface	50
Figure 4.2 registers interface.	51
Figure 4.3 main interface	52
Figure 4.4 Realtime interface	53
Figure 4.5 Graph interface for humidity	54
Figure 4.6 Graph interface for Temperature	54
Figure 4.7 Authentication database	55
Figure 4.8 Authentication Database code in Android Studio	55
Figure 4.9 Realtime Database for temperature, humidity, fan, and water pump status	56
Figure 4.10 Code in Arduino IDE to gain the temperature and humidity value.	57
Figure 4.11 Rules for power on and off Water Pump	57
Figure 4.12 Source Code for Login Page	58
Figure 4.13 Source Code for Login Page	59
Figure 4.14 Source Code for Login Page	59
Figure 4.15 Source Code for Register Page	60
Figure 4.16 Source Code for Register Page	60
Figure 4.17 Source Code for Register Page	61
Figure 4.18 Source Code for Homepage	61
Figure 4.19 Source Code for Real Time page	62
Figure 4.20 Source Code for Real Time page	62
Figure 4.21 circuit diagram	63
Figure 4.22 prototype diagram	64
Figure 4.23 prototype diagram (node mcu esp8266 placement)	64
Figure 4.24 prototype diagram (DHT22 Sensor placement)	65
Figure 4.25 prototype diagram (Water Pump placement)	65
Figure 4.26 Setup data base code	66
Figure 4.27 Setup for Wi-Fi connection	66
Figure 4.28 Setup for firebase connection code	67
Figure 4.29 Setup for read temperature and humidity code.	67
Figure 4.30 Setup for power on/off fan code	68

Figure 4.31 Setup for power on/off water pump code	68
Figure 4.32 Authentication After User created.	69
Figure 4.33 Code Uploading successful in the Node mcu ESP8266.	70
Figure 4.34 The ESP8266 successfully connected to the internet.	70
Figure 4.35 The ESP8266 Humidity, temperature, fan and water pump status reading successfully	70
Figure 4.36The Humidity, temperature, fan, and water pump status reading successfully stored in realtime database	70

LIST OF SYMBOLS

°C	degree Celsius
%	percentage

LIST OF ABBREVIATIONS

Iot	Internet of things
APKs	(Android Package Kits)
RH	Relative Humidity
Node MCU	Node Micro Controller Unit
Co2	Carbon dioxide
V	Voltage
AC	alternating current
DC	direct current
LCD	Liquid Crystal Display
PV	Photovoltaic
Amp	Ampere
volts	Voltage
Wi-Fi	wireless fidelity- high fidelity
IDE	integrated development environment
SDLC	Software Development Life Cycle

CHAPTER 1

INTRODUCTION

1.1 Introduction

Future climate change and climate variability, including an increase in extreme weather events, will especially affect the agriculture industry. The land and water environment will alter as a result of changes in temperature and precipitation, which will have an impact on agricultural (Anwar et al., 2013). Due to improvements in agricultural system technology, the field of agriculture has recently gained appeal. Due to the country's growing population and the development of new uses for mushrooms in both medicine and cuisine, the mushroom market in Malaysia had experienced significant demand.

Furthermore, cultivated mushrooms are one of Malaysia's most valuable crops. The Malaysian government classifies mushrooms as one of seven commercially grown high value crops. Raw materials are intensively cultivated with government support. Mushroom cultivation area will be expanded from 78 hectares in 2010 to 340 hectares in 2020. Demand for mushrooms in both local and global markets is high and growing (Haimid et al., 2013). Mushroom is taken into consideration to be a brand new and small enterprise in Malaysia. This enterprise, however, is developing as one of the reasserts of meals and new supply of wealth for farmers. Mushroom is a fleshy, spore bearing fruiting frame of a fungus, generally produced aboveground on soil or on its meals supply (Haimid et al., 2013)

Meanwhile, the Internet of Things (IoT), smartphones tools, and sensors are a technology that, in addition, allow farmers to recognize the precise of their field, consisting of soil temperature, the quantity of water required, climate conditions, and lots more(Obaideen et al., 2022). The Internet of Things-based design scheme for an environmental monitoring system was presented to maintain and control the growth process and monitor mushroom development in real-time online. This design scheme is based on the primary climatic factors that affect mushroom growth. Additionally, the proper sensor and sensor placement was made within the mushroom house following the climatic conditions there.

1.2 Problem Statement

Oyster mushrooms in Malaysia still use old methods in the agricultural sector such as Workers have to irrigate the area around the mushroom house manually every several hours, especially to control the environment that is suitable for the farming of the mushrooms to grow in and to produce good-sized oyster mushrooms. Thus, the method of farming grey oyster mushrooms involved several specifications that needed to be adhered to, where the climate conditions in Malaysia are constantly changing, causing abnormal and slow growth of mushrooms. Some problems have been identified as follows:

Operators may neglect to check and regulate the temperature, and eventually it is not done consistently in the mushroom house. Last but not least, it is impossible to create mushrooms with the desired qualities, such as those that are lighter and smaller in size. Oyster mushrooms need moderate temperature and humidity to grow well. If the temperature inside the mushroom house changes, it will affect the growth of the mushrooms themselves. So, this system, which can control and monitor the temperature inside the mushroom house using IoT concept, can be controlled via a smartphone application.

Manual labour was expended to water the mushroom house, lowering the temperature and managing the humidity. When watering manually, water will indirectly be wasted. The temperature and humidity are some of the important parameters need to manage in a mushroom farm and these data are collected through wireless sensors. But in this scenario the humidity and temperature will be the major parameter that will be

observe. The implementation of this smart mushroom system have optimized the usage of resources such as water and fertilizer and also maximized the quality and productivity of the mushroom.

1.3 Objective

The aims of this project are to develop a functional IoT system increase and control the humidity and temperature. There are three objectives in this project which are:

- i. To study the existing system on mushroom cultivation system.
- ii. To develop the automatic mushroom house monitoring using internet of things that can control the humidity and temperature.
- iii. To evaluate the functionality of the purpose system.

1.4 Scope

The scope of the project is:

- User Scope
 1. Mushroom house entrepreneur.
 2. Individuals who desire to build a mushroom house in their home.
- System Scope
 1. Cover the humidity and temperature in mushroom house.
 2. Observe the mushroom house's temperature and humidity using smart mobile application.

1.5 Thesis Organization

This thesis is divided into five chapters. PSM 1 is covered in chapters one through three, while PSM 2 is covered in chapters four and five. The first chapter discusses the introduction, problem statements, project objectives, scope, and methodology. While Chapter 2 contains a review of the literature, Chapter 3 explains the methodology used in this project. The discussion and outcomes of the projects will then be discussed in Chapter 4. Finally, the conclusion and project outcome will be summarised in Chapter 5.

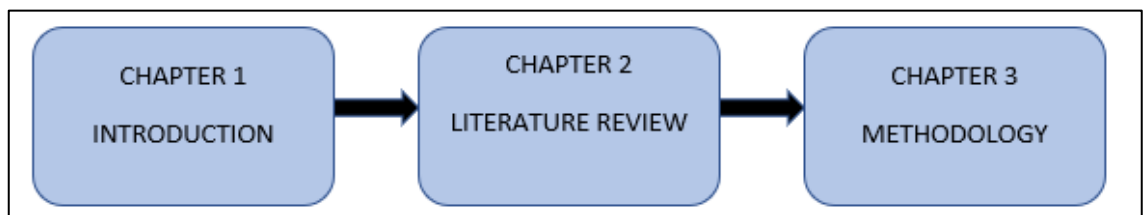


Figure 1.1 Flow for PSM 1

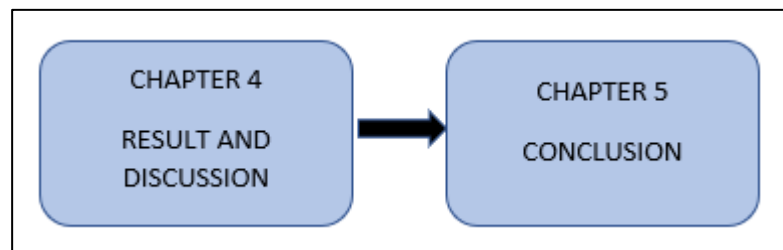


Figure 1.2 Flow for PSM 2

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Work

Oyster mushroom farming can provide significant earnings due to the product's high demand, both locally and internationally. The cultivation of grey oyster mushrooms is one of the contributors to the revenue of Malaysian farmers as well as to the development of Malaysia's agricultural industry.

The design scheme of the environment monitoring system based on the Internet of things was put forward on Climatic conditions and the primary climatic conditions that influence the growth of mushroom were put forward in order to maintain and control the growth process and monitor mushroom growth in real time. The primary climatic factors influencing the growth of mushrooms were included in the design scheme for the environment monitoring system.

2.1.1 Automatic Climate Control for Mushroom Cultivation Using IoT approach. (System 1)

The majority of farmers only rely on conventional agricultural methods, according to this source by (Mohd Ariffin et al., 2020). Modern agricultural technology has proven to be superior to conventional methods for increasing the productivity and efficiency of their farms. Internet of Things (IoT) is typically associated with modern agriculture and gives farmers the ability to remotely and in real-time monitor the health of their farms. Therefore, the goal of this effort is to create an automated climate control system with the lowest possible cost that can manage the farm's condition and maximize its resource use.

This system was developed using a limited number of components, such as a mushroom home, an internet of things control box, and a web client interface. The walls

of the mushroom home are constructed out of fine mesh netting to allow for unrestricted airflow and to keep dogs from getting inside. A water pump is also included in the mushroom house's amenities. The outside netting and the floor are sprayed with water using a water pump so that the humidity and temperature may be controlled without accidentally contaminating the mushroom. This is done to ensure that the mushroom does not become contaminated.

The temperature control system, which has a number of parts, serves as the system's brain. The node MCU ESP8266 provides the system's processing power. Temperature and humidity readings from the DHT22 sensor were used as system input. The DHT-22 is a reliable sensor. A 5V two-channel relay is used by the system to regulate the water pump and exhaust fan, both of which were powered by AC current. when an environmental regulatory corrective action is carried out by the system algorithm. Moreover the LCD could display the system status, uptime, temperature, and humidity value at the moment. Figure 2.1 illustrates the connection between the hardware.

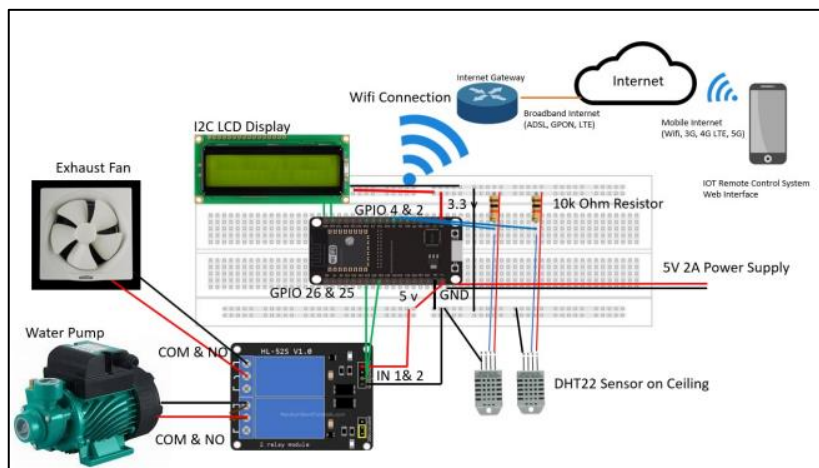


Figure 2.1 System Schematic Diagram

The system ran in a loop until it was switched off. Because the DHT22 could only deliver a value after a certain amount of time had passed, the system first waited for 5 seconds before getting a reading from the temperature and humidity sensor. From many sensors, the system estimated the average temperature and humidity readings. The system will run from one of the three sub-programs based on the average values.

Program 1: The mechanism activated the water pump and exhaust fan via relay for 20 seconds to cool down if the average temperature surpassed 27 °C. Program 1 counter was increased to reflect the number of times it was run before the program ended.

Program 2: When program 1 was not operating and the humidity in the mushroom house above 80%, the system activated the exhaust fan for 20 seconds using a relay. Program 2 counter was increased to reflect how many times it was run before the program ended.

Program 3: If programs 1 and 2 were not operating and the humidity in the mushroom house environment was below 60%, the system activated the water pump through a relay for 20 seconds. To keep track of how many times Program 3 was run, a counter was increased. The figure 2.2 illustrates the system flowchart.

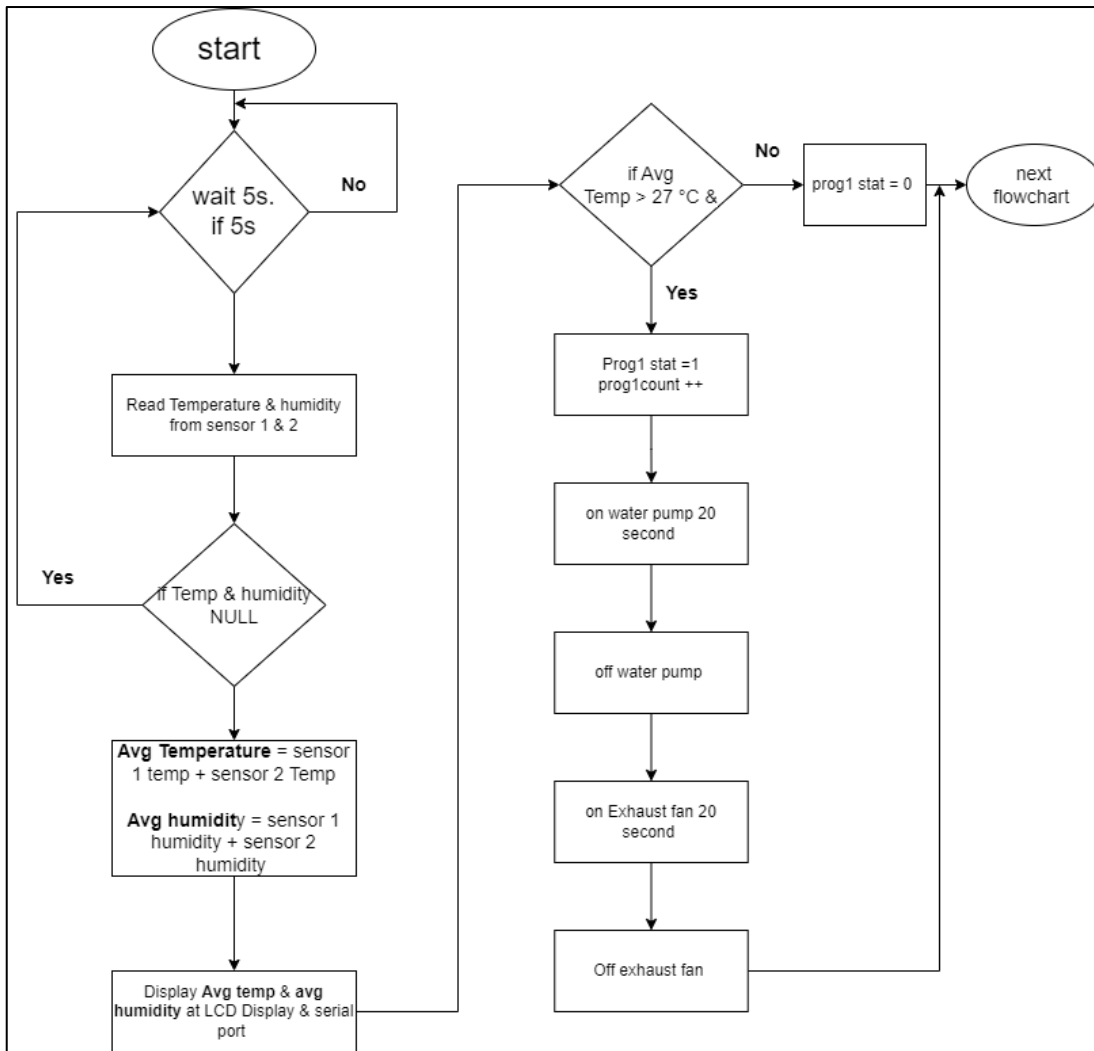


Figure 2.2 System Operation Flowchart

This system uses a web-based application to monitor and collect the parameter values for the mushroom house. Any mobile device with a web browser can access the user interface that the web client offers. Users can only access the system via the web interface, which shows humidity, mushroom house temperature, and system uptime statistics in real-time. Only the IP address of the user can access the system. Figure 2.3 shows the web interface for the system.

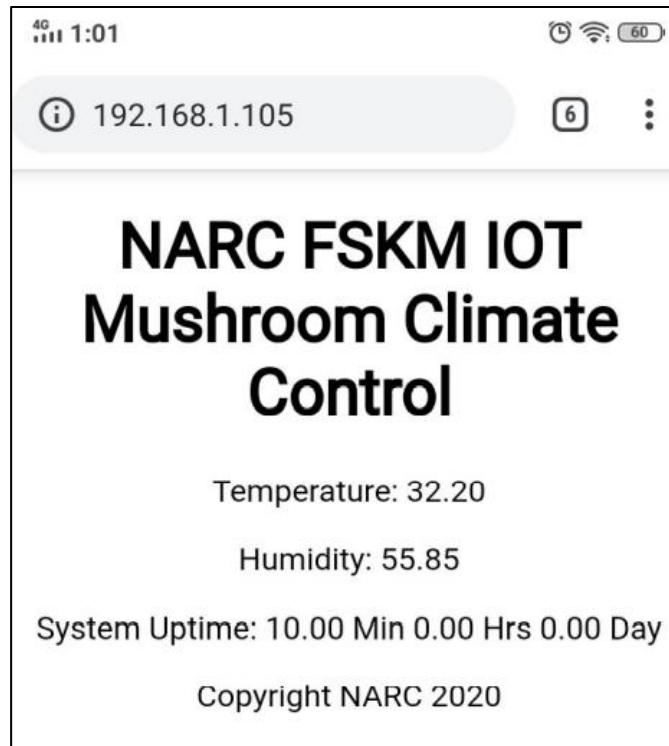


Figure 2.3 System web interface

2.1.2 Green technology monitoring system for oyster mushroom. (System 2)

This system was proposed by Siti Sri Noraini Ta'ahal, Alias Masek, Yusmarwati Yusof (Sri et al., 2022). The goal of this system is to create systems that use the least amount of power while also spending the least amount of money on overall system development. This monitoring system is designed to control the ambient temperature of an oyster mushroom home and design a mushroom house power supply system utilizing photovoltaic (PV) solar energy. The main brain of this system is the Node MCU ESP8266 as a microcontroller, and all the data regarding humidity, temperature, and water tank level is kept in the Blynk application. It also incorporates a DHT22 sensor to measure humidity and ambient temperature.

Based on the figure 2.4 of the Block Diagram of solar PV Power supply, The 25W 6V solar panel attached to the 10A solar charge controller is where the solar PV power source is first connected. Following that, the battery with a capacity of 12 volts and 7.0-amp hours is linked to the solar charge controller before being directly connected to the inverter to supply the system with alternating current (AC).

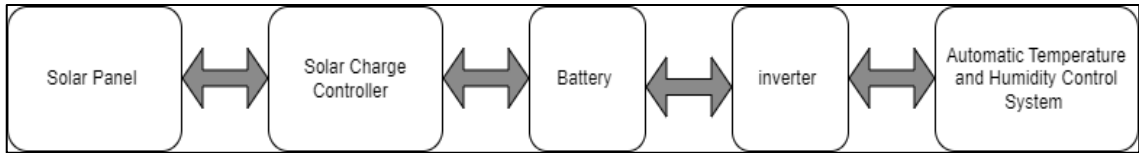


Figure 2.4 solar PV Power Supply Block Diagram

A 12VDC power supply is provided to the water pump control circuit, while another power supply is decreased to 7V by utilizing a relay for delivery to the ESP8266 Node MCU. The solar energy source is directly into the Node MCU ESP8266. The water pump will then turn on automatically in three (3) seconds if the DHT22 sensor detects a temperature above 30 C or a humidity of less than 70%, as seen on the Blynk app's display. The Blynk programs will use the received data to show it on the mobile device. Figure 2.5 show the block diagram for the oyster mushroom monitoring system using green technology.

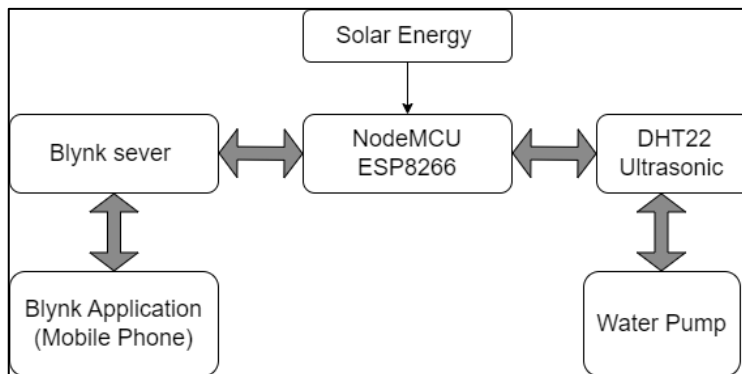


Figure 2.5 The block diagram for the oyster mushroom monitoring system using green technology.

The ultrasonic sensor, which is placed on the tank lid and should be pointing down to measure the distance between the water and the cover, and the DHT22 sensor, which was mounted on the mushroom rack to measure the temperature and humidity around the chunks of oyster mushrooms, make up the system's total component positions. The ultrasonic sensor, which is mounted on the tank cover and is facing down so that it is in opposition to the water, determines the location of the system's overall component. The oyster mushroom chunk's body serves as the location for the water channel, which is

secured to the iron of the mushroom rack. The solar panels are situated in locations that get a lot of sunlight in the meantime.

based on the system design that was created and used in the mushroom house to create the product prototype. The Blynk application will display all the data, including temperature and humidity, on the phone, as shown in Figure 2.6.

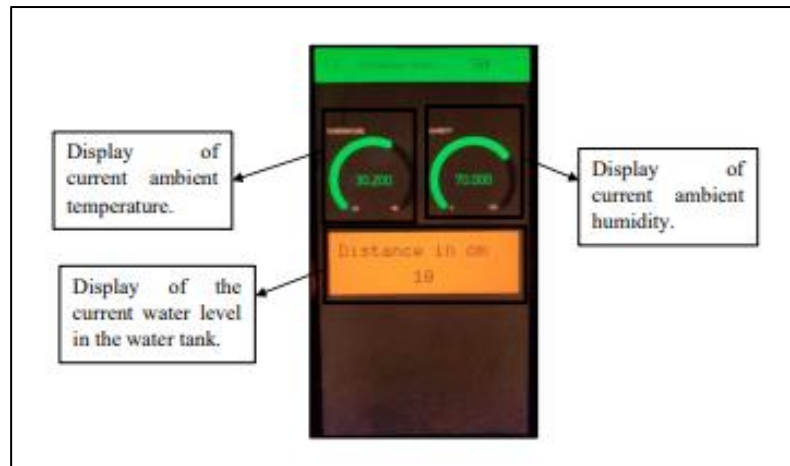


Figure 2.6 Display of the Blynk apps

In conclusion, this system just the uses the NodeMCU ESP8266 as microcontroller to regulate the system's smoothness. DHT22 and ultrasonic sensors are the two types of sensors that are employed. Ultrasonic sensors are applied to measure the water level in the tank, while DHT22 sensors are used to measure temperature and humidity. In order to maintain the mushroom house's humidity and temperature, this method just needs water. Additionally, this system is one of the energy-efficient green systems because it completely relies on solar power.

2.1.3 Mushroom House Monitoring System Using Internet of Things (IoT), (System 3)

This system was propose by Muhammad Hafizuddin Mohd Hishamuddin , Aizan Ubin (Hafizuddin et al., 2022), The goal of this system is assist people understand how the Internet of Things (IoT) supports humans in all aspects of agriculture, including real-time

data updating, monitoring, control, and environment control. The majority of agricultural plants rely on conventional farming and a lot of labour-intensive work. In order to solve this issue, this project involves the development of a monitoring kit for mushroom houses as well as automatic temperature and humidity control.

The Node MCU ESP32, Arduino Uno, and DHT 11 make up the hardware system. This component is able to recognize changes in the mushroom house's humidity and temperature. The Node MCU ESP 32 functions as a Wi-Fi to upload data to Thingspeak, which serves as an IoT platform. To further control humidity and maintain a low temperature, the water level system is used.

The DHT11 is connected to the Arduino Uno, and it must be configured in order for it to detect temperature and humidity. The project's IoT capability is enabled by the NODE MCU ESP 32. It establishes a connection with the nearest WIFI in order to allow the transmission of any incoming data to the desired destination. The system's conceptual diagram is shown in Figure 2.7.

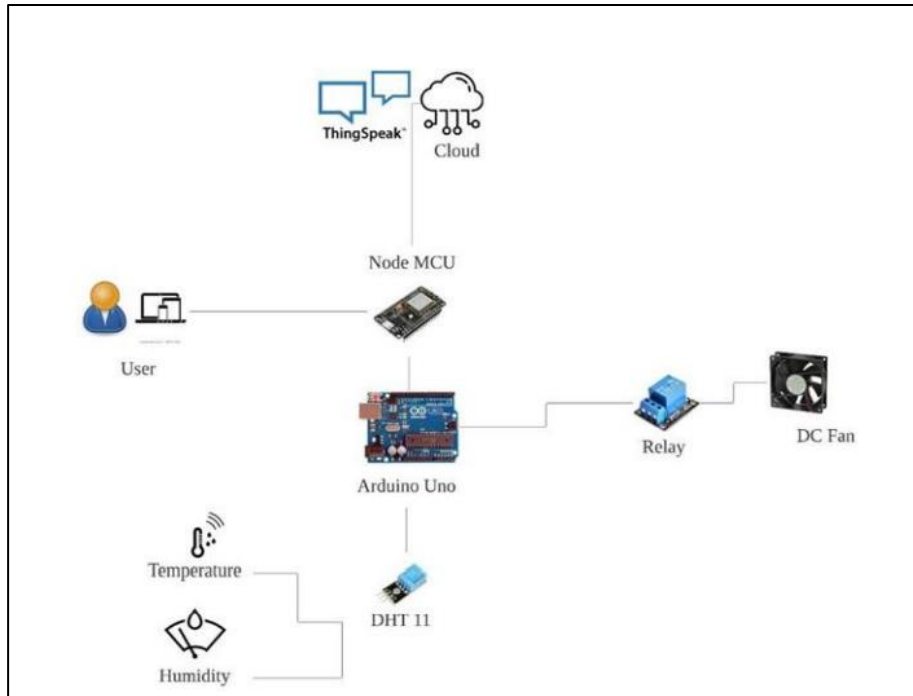


Figure 2.7 Concept diagram of the Mushroom House monitoring

The process operates as sensors required to monitor and to automate surrounding. The microcontroller will both do the process and send the sensor data and upload the performance status to the server. Whenever the temperature exceeds its limit, the fan will turn on until the it regulated back to its temperature. This condition also applied to humidity changes, whenever it goes lower than normal, the dc fan is expected to operate until it adjusted. In order to fully understand this system's operation, a flowchart is presented in Figure 2.8.

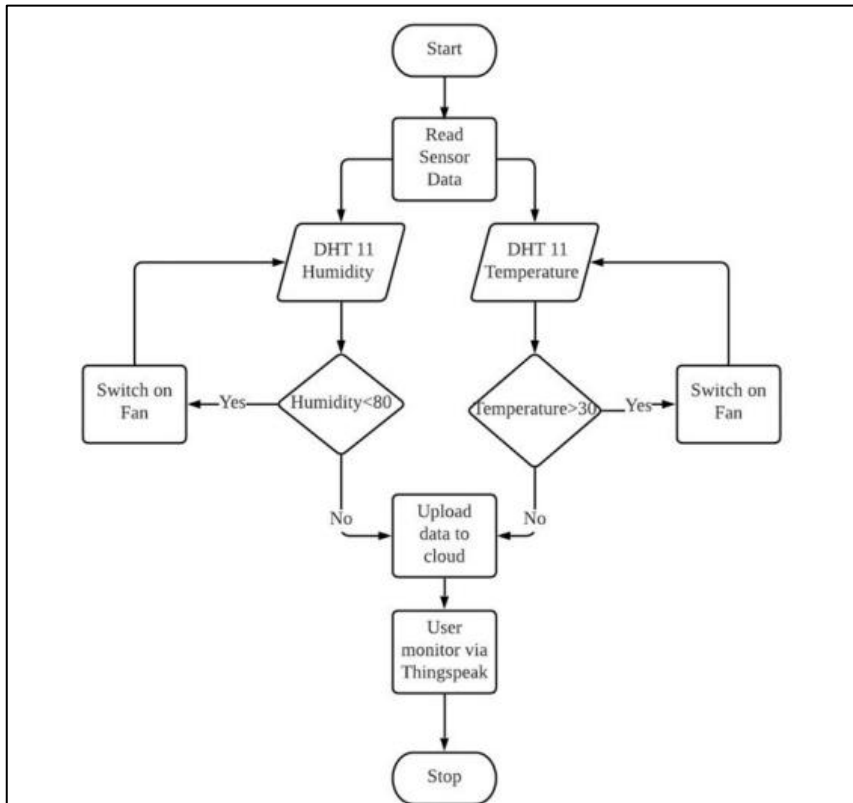


Figure 2.8 Flowchart of the System

The product prototype was developed using the concept diagram for the mushroom house monitoring system, which was developed and implemented in the mushroom house. As seen in Figure 2.9, the Thingspeak will show all the data, including the temperature and humidity, on the phone.

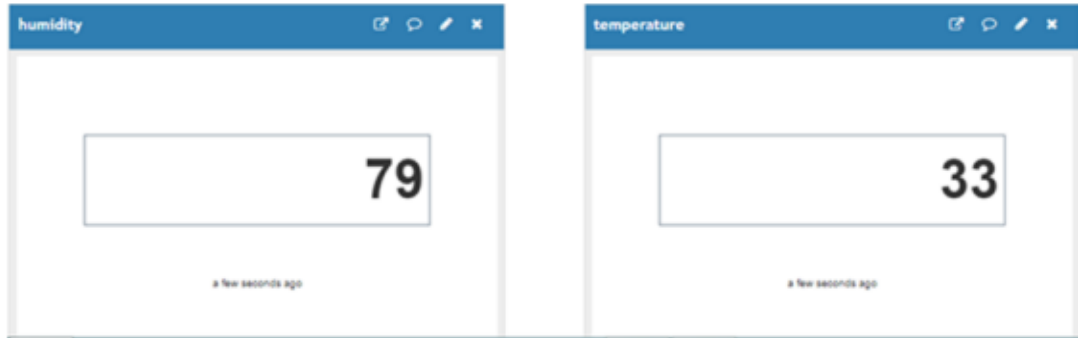


Figure 2.9 Dashboard show latest update real data monitoring.

In conclusion, this system utilizes the Arduino Uno and the NodeMCU 32, two different types of microcontrollers. The sensor and DC fan are controlled by Arduino Uno, while the data transfer from the sensor to the cloud database and display on the Thingspeak dashboard are controlled by NodeMCU 32. The mushroom house's humidity is only managed by this system's single DC fan. The only location where water is used is underneath the mushroom rack.

2.2 Analysis of comparison on existing system

Table 2.1 Comparison of Methods among Past Research Papers.

Specification	System 1	System 2	System 3	Purpose system
Sensor	DHT 22 (Temperature and humidity)	DHT 22 (Temperature and humidity)	DHT 22 (Temperature and humidity)	DHT 22 (Temperature and humidity))
Hardware (microcontroller)	Node MCU ESP 8266	Node MCU ESP 8266	Node MCU ESP 32 and Arduino uno	Node MCU ESP 8266
Hardware (control)	Fan and water pump	Water pump	Fan	Fan and water pump

humidity and temperature)				
Software	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE and android studio
Application	Web base	Blynk	Thingspeak	Smart mobile application
Wi-Fi communication	Yes	Yes	Yes	Yes
Cloud base	Yes	Yes	Yes	Yes
Advantages	<ul style="list-style-type: none"> • Using a fan and water as two mechanisms for controlling the humidity and temperature. • After executing the software three times, the system stopped for two hours, therefore the cost was small. • Use the LCD screen to make the monitoring procedure easier. 	<ul style="list-style-type: none"> • Green technology because the system is solar powered. • Applying a blank application to reduce costs and make it user-friendly 	<ul style="list-style-type: none"> • enabling many users to keep an eye on the humidity and temperature. • Simple system flow and design. 	

Disadvantages	<ul style="list-style-type: none"> • The water pump and fan are both costly pieces of equipment. 	<ul style="list-style-type: none"> • solar panels and batteries make the first implementation pricey. • Complex to implement 	<ul style="list-style-type: none"> • It is difficult to manage the temperature and humidity when only the fan is used. • When collecting data on temperature and humidity, the sensor DHT11 is not exact value 	
---------------	---	--	--	--

In summary, the most of the system's components work to control and maintain the humidity and temperature inside the mushroom house. DHT22 is the command sensor used. It has been applied to systems 1 and 2. Because it is more accurate when measuring the temperature and humidity level in the mushroom house, the DHT22 sensor is used. The dht11 sensor is not appropriate because, as mentioned in system 2, it does not capture accurate temperature and humidity. The other sensor that has been used to determine the water level in a water tank is an ultrasonic sensor. Only System 2 uses this sensor.

All existing systems are designed to automatically stabilize the mushroom house's temperature and humidity when the system's sensors detect that the temperature and humidity are not suitable for mushroom growth in the mushroom house. The node MCU 8266 microcontroller, which is used by systems 1 and 2, is the same. Both an Arduino Uno and a node MCU 32 are used in System 3.

Through the web and applications, all systems may monitor temperature and humidity in real time. The web base is a tool that System 1 utilizes to display the temperature and humidity that were read from the sensor. While systems 1 and 2 monitor temperature and humidity using blynk and Thinkspeak. All of these systems, however, require the closest Wi-Fi connection to always send the most recent updates to the web and applications in order to allow users to obtain temperature and humidity data. To make

it simpler for users to check the data, the temperature and humidity readings are kept in their respective cloud databases.

2.3 Relevance of Comparison with project title

The suggested system will use a DHT 22 sensor to measure the temperature and humidity in order to compare it to the three currently in use systems. Additionally, if the humidity and temperature fall below a certain threshold, it will communicate data to the user and notify them via an application that uses the database. This alerting feature can inform the user when the temperature has to be raised or lowered in order to achieve the perfect condition for a mushroom house, which eliminates the need for frequent manual check-ups.

The ESP8266 board is intended to be utilized for data interchange with the database over Wi-Fi in order to interact with user apps that were suggested for this new system. As a reference to all the current systems, the Arduino IDE software will be the appropriate software to program the ESP8266 board. The software called Android Studio was used to create the mobile application. The suggested system will be built on the idea of the Internet of Things (IoT), making it a cloud-based system that is totally dependent on the internet for communication networks.

2.4 Summary

In conclusion, smart monitoring for mushroom houses is best employed for effective mushroom establishment in the mushroom house since mushrooms require the proper temperature and humidity. As a result, it is necessary to minimize the traditional approach because it results in a lot of human mistakes. Additionally, IoT technology is integrated into mushroom house monitoring so that consumers may view real-time temperature and humidity through simply smart applications and the web.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In the following chapter, we will discuss the project methodology, which will provide an outline of the process of project development. The approach for the development of systems that leads the researcher step-by-step through the process of constructing, planning, and developing the system. The iterative model, the incremental model, and the waterfall model are the three available approaches that are most suited for the development of this system.

3.2 Project Methodology

A methodology is important in system development because it gives a thorough framework for the researcher to build a system, reducing errors and delays during system implementation. The software development life cycle (SDLC) is applied in this methodology. Software development is carried out using the System Development Life Cycle, or SDLC. The SDLC idea consists of a number of steps, starting with planning, moving on to analysis, design, implementation, and finishing with system maintenance. There are numerous models that can be used depending on the project, including the agile model, iterative model, and waterfall model.

3.2.1 Agile Model

Agile project management techniques divide work into smaller iterations or pieces without directly including long-term planning. Plans for the quantity, length, and scope of each iteration are spelled out in detail in advance. The project's requirements and scope are established at the start of the development phase (*Agile Model (Software Engineering) - Javatpoint, n.d.*). Moreover, Agile is an adaptive strategy in which there is no specific planning and merely a clear understanding of future work in terms of what features must be produced. Development is feature-driven, and the team constantly adjusts to shifting product requirements(*SDLC - Agile Model, n.d.*).There is a testing step

for each iteration. Regression testing can be implemented each time new code or functions are deployed. A team goes through the entire software development life cycle at each iteration, including planning, requirements analysis, design, coding, and testing, before showing the client a functional result (*Agile Model (Software Engineering)* - Javatpoint, n.d.).

3.2.2 Iteration Model

The iterative model was created in response to issues with the waterfall model, which created a need for software development models that could produce results more quickly, with less information needed, and with better flexibility (Nugroho et al., 2017). Each iteration includes its own testing phase. Regression testing can be implemented each time new code or functions are deployed. A team goes through the entire software development life cycle at each iteration, including planning, requirements analysis, design, coding, and testing, before showing the client a functional end result (Nugroho et al., 2017). But This iterative model's drawback is that it can only be used for large, complex software development projects. This is due to the difficulty of subdividing a tiny software system into smaller, more manageable units.

3.2.3 Waterfall Model

The first Process Model to be introduced was the Waterfall Model. The term "linear-sequential life cycle model" is also used to describe it. It is incredibly easy to use and comprehend. There is no overlap between phases in a waterfall model; each step must be finished before the subsequent phase can start (*SDLC - Waterfall Model*, n.d.). The waterfall model of the SDLC is frequently used. This method divides the entire software development process into different SDLC phases. The results of one phase serve as the input for the following one in this SDLC paradigm. The issue of this development is that little time is given for reflection or modification. It is quite challenging to go back and fix something that wasn't carefully thought out or documented during the concept stage once an application has entered the testing phase.

3.2.4 Model for Proposed Project

Following thorough evaluation, it was determined that the agile model was the optimum paradigm for creating this system. This is due to the speedy development and demonstration of capabilities. The number of resources needed is small. Delivers early, imperfect functioning solutions that are suitable for fixed or shifting requirements. In this model, the phases include requirements, design, development, testing, and deployment. Figure 3.1 show the phase in the agile model.

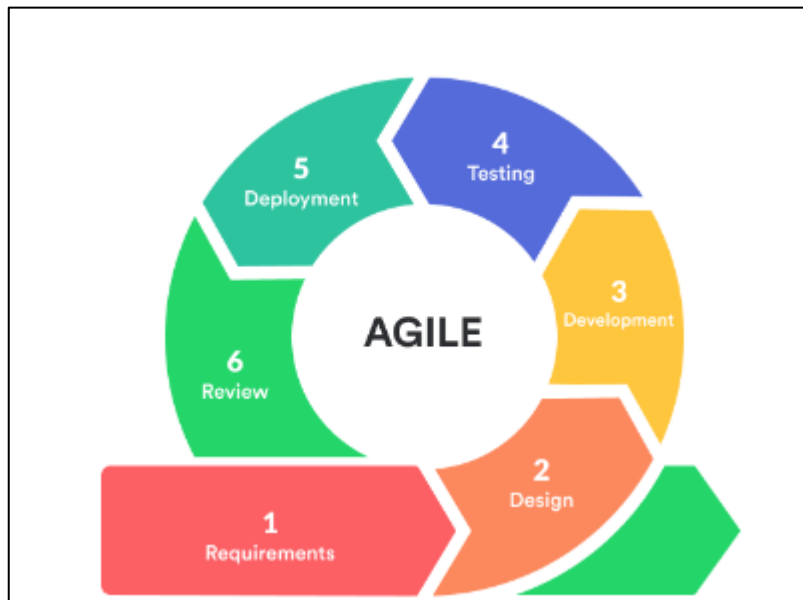


Figure 3.1 Agile model phase

3.2.4.1 Requirement Phase

During the requirement phase, the proposed project comprises automatic water drops and a fan function when sensing variations in temperature and humidity in the mushroom house. Additionally, it will display real-time data to smart application users. Additionally, if the temperature rises beyond 30 degrees Celsius and the humidity falls below 70%, the sensor will also detect humidity and warn you to any change. Users just track temperature and humidity, and they will be informed via the application, therefore there are no further design requirements. The user will find it simpler to frequently water the area surrounding the mushroom home as a result.

3.2.4.2 Design Phase

Work began in the design phase, which includes establishing the system's models. Some of the tasks that were finished were system design and smart application interface design. To illustrate how the system interacts with the end user, approaches including flow charts, use case diagrams, and context diagrams are used in system design. Data flow from the ESP 8266 microcontroller to the system, as well as data flow from the system to the user or vice versa, are shown in the context diagram.

3.2.4.3 Development Phase

The Arduino IDE will be used to programme hardware like the nodeMCU ESP 8266 during the development process. Additionally, it will be based on the current flow system

3.2.4.4 Testing Phase

During the testing phase, the system was tested against the functional requirements that were stated. When detecting changes in temperature and humidity, these parameters are measured using a DHT22 sensor, the system is tested for water droplet dispensing and fan function. afterwards be seen on the smart application.

3.2.4.5 Deployment Phase

The system is now in the deployment phase, when it has finished development and is prepared for delivery to end customers. Although the system is ready for usage, a prototype system will be tested in this project.

3.2.4.6 Review Phase

During the review phase, all test results are analysed to ensure that the system is operating at optimal levels as required by this project. But if any faults or issues are discovered during testing, they will be assessed in this stage, and the development process will then move back to the requirements stage. The project developer will keep doing this until they can complete all the requirements and deliver the desired outcomes.

3.3 Project Requirement

3.3.1 Functional Requirement

1. Humidity and temperature control: User able to control the humidity and temperature of the mushroom house using water pump and fan to get the ideal environment for mushroom.
2. Sensor detection: The DHT 22 sensor in this system required to detect any changes due to humidity and temperature inside mushroom house.
3. Display information: The user can view the temperature and humidity readings that have been transmitted from the device's sensor monitoring via the database via the application.

3.3.2 Non-Functional

1. Availability

When the temperature and humidity are lower than the coding requirements, the smart mushroom home is obliged to dispense water. The applications must constantly be available to the user in order to operate and monitor the temperature and humidity in the mushroom house. A continuous internet connection is also necessary to connect to the ESP 8266's Wi-Fi module and exchange data with smart application.

2. Usability

The smart mushroom house must be functioning at optimum level to detect the temperature and humidity. it also functioning the water pump and fan when it is connected to the power supply by the user

3.3.3 Limitation

This device relies on a power supply and an internet connection to offer real-time temperature and humidity readings in the mushroom house.

3.3.4 Constrain

1. The microcontroller is easily exposed to water in the moist environment of the mushroom house, which leads to a short circuit.

2. This system is entirely reliant on electricity, and if there is a power outage, the system will be able to stabilise the temperature and humidity in the mushroom house.

3.3.5 User Requirement

1. The user must interact with the smart application to obtain real-time humidity and temperature data.
2. The user is able to review past data as needed in order to maintain the environment of the mushroom house.

3.4 Propose Design

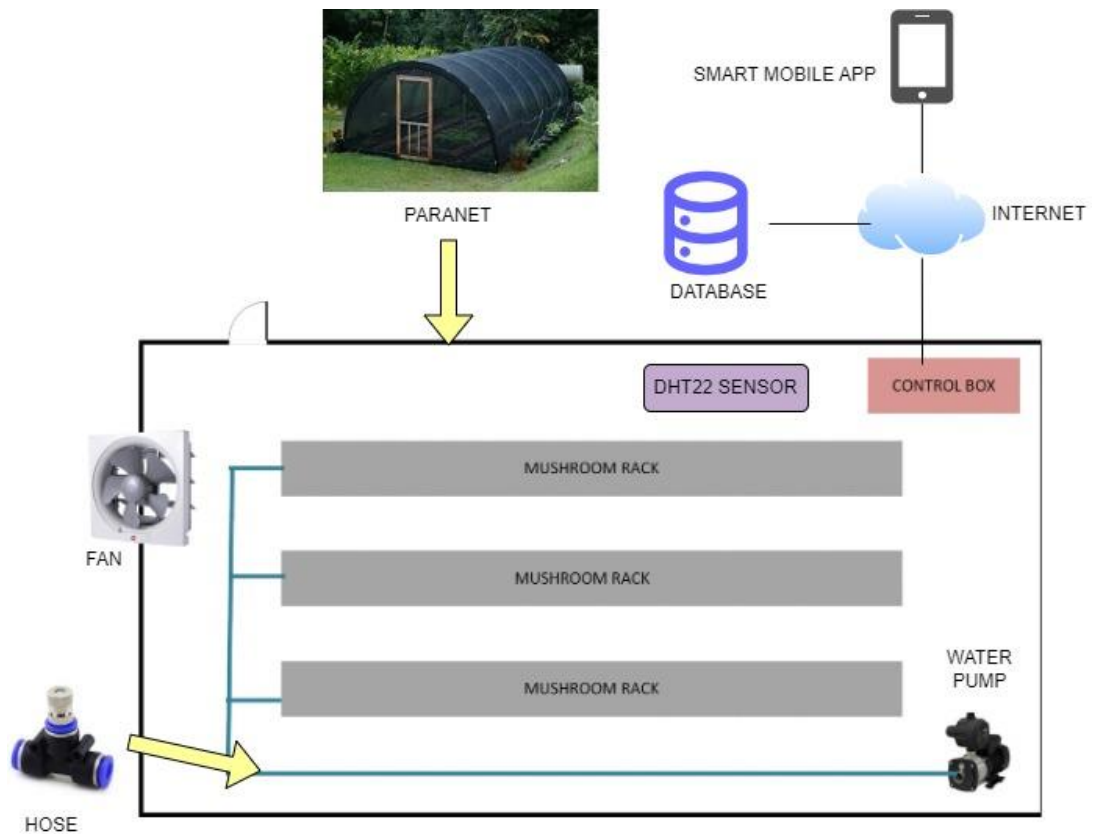


Figure 3.2 illustration of the purpose design

The mushroom house requires the ideal temperature and humidity that are set based on the purpose of the design. The dht22 sensor will detect the temperature and humidity in the environment. If the temperature falls below a certain threshold, the water pump will start. The fan will activate when the humidity is at its peak. The best and worst temperature and humidity values are shown in Table 3.1.

Table 3.1 Threshold value for temperature and humidity

	Temperature	Humidity
Best	20°C - 30 °C	70 % - 80 %
Worst	<20 °C , > 30°C	<70% , > 80%

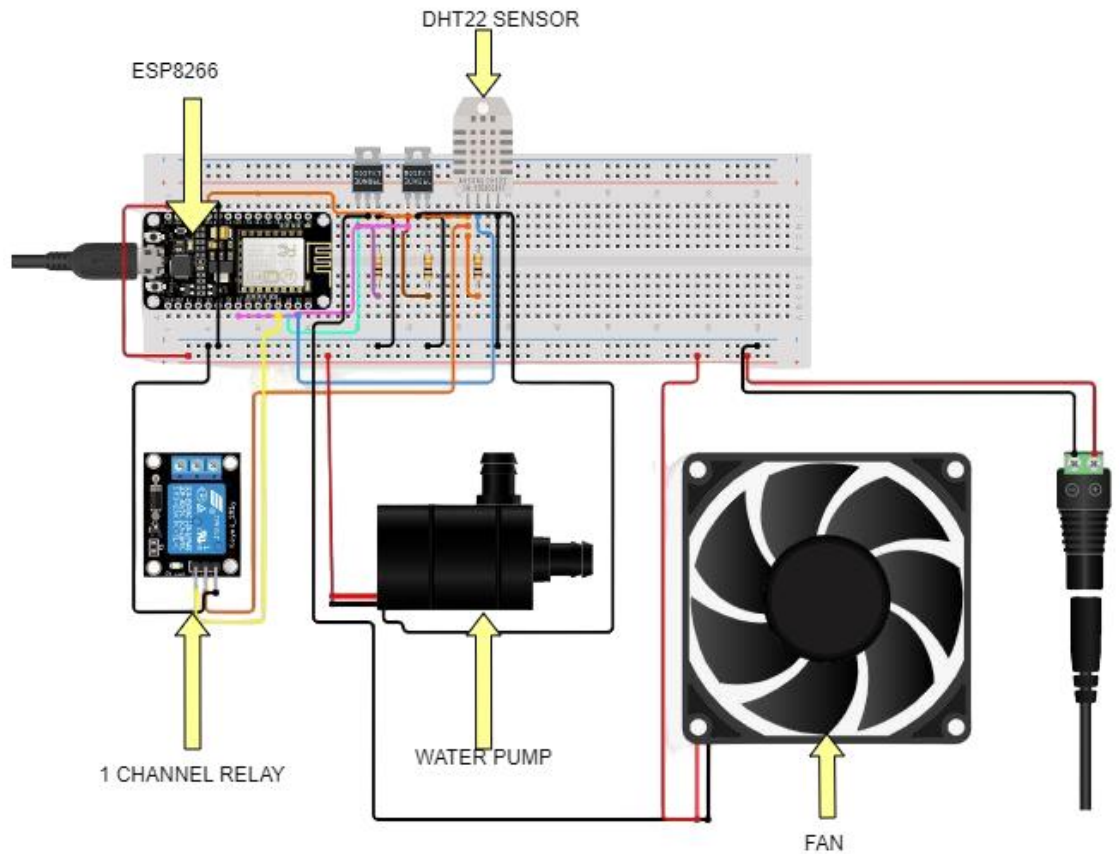


Figure 3.3 hardware connection of purpose design

3.4.1 Architecture Design

In this proposed system, the microcontroller used is the ESP8266. The system uses value data from a DHT22 sensor to automatically regulate temperature and humidity. The ESP8266 microcontroller and all of the sensors are directly linked. The Wi-Fi module, which is also on the ESP8266 board, is connected to the ESP8266. The data can be sent to the database thanks to the Wi-Fi module. The user and the smart application will receive data or alarm notifications from the database. The user will be able to keep an eye on the temperature and humidity. Figure 3.2 displays the finished sensor design in detail.

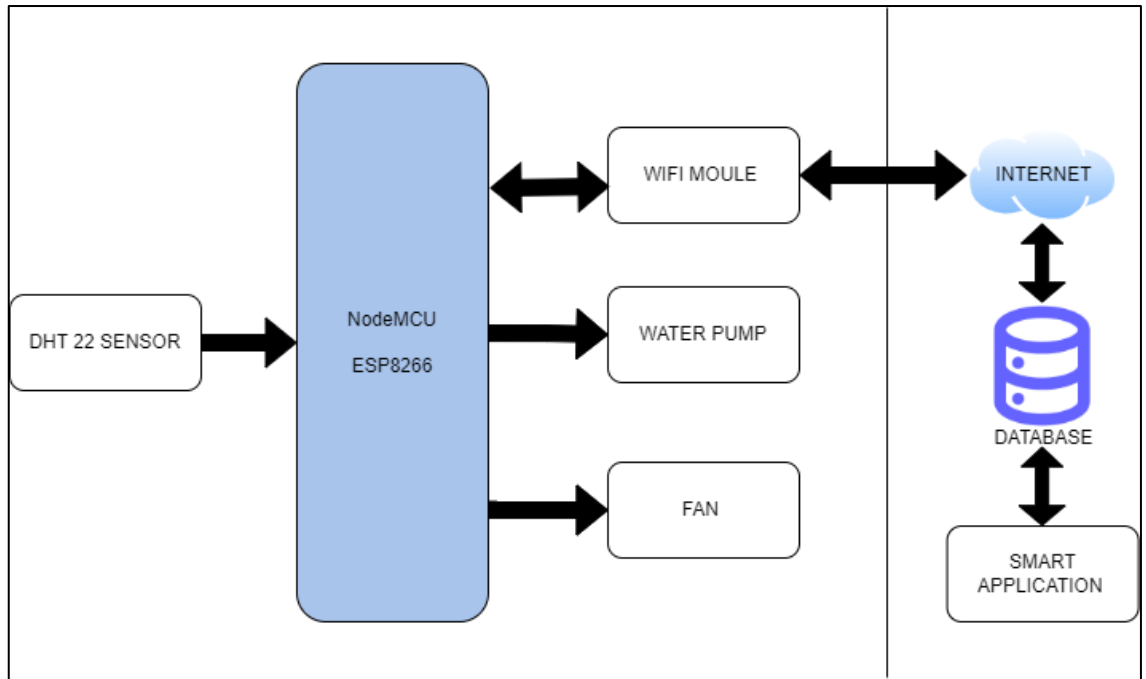


Figure 3.4 Architecture Design

3.4.2 Flowchart

The suggested system is an automatic mushroom house monitoring system that would provide real-time data to the user application. If the temperature is over 30 degrees Celsius and the humidity is above 80%, it will also turn on the fan and water pump. To establish the connection between the microcontroller and the sensors, the system must first be turned on. The DHT-22 sensors will then be turned on in a subsequent stage to collect the temperature and humidity readings. The microcontroller will continue to receive data from the sensors.

The ESP8266 microcontroller will receive the signal from the DHT22 sensor as part of the automated cleaning process, and the programming language code attached to the microcontroller will analyse the data to determine whether the temperature and humidity have reached their maximum values. After 30 seconds of operation, the fan and water pump will turn off, and the sensor will once more receive data. The operation will be repeated if the value still reaches the maximum value.

Finally, the ESP8266 microcontroller will receive the signal once it has been analysed as part of mushroom home monitoring, and information will then be sent and

stored in the database via the Wi-Fi module. The user will further analyse the data supplied to the database to create a report on the typical humidity and temperature of the mushroom house by week or month. Figure 3.3 show the flowchart of the hardware and figure 3.4 show the flowchart of the system.

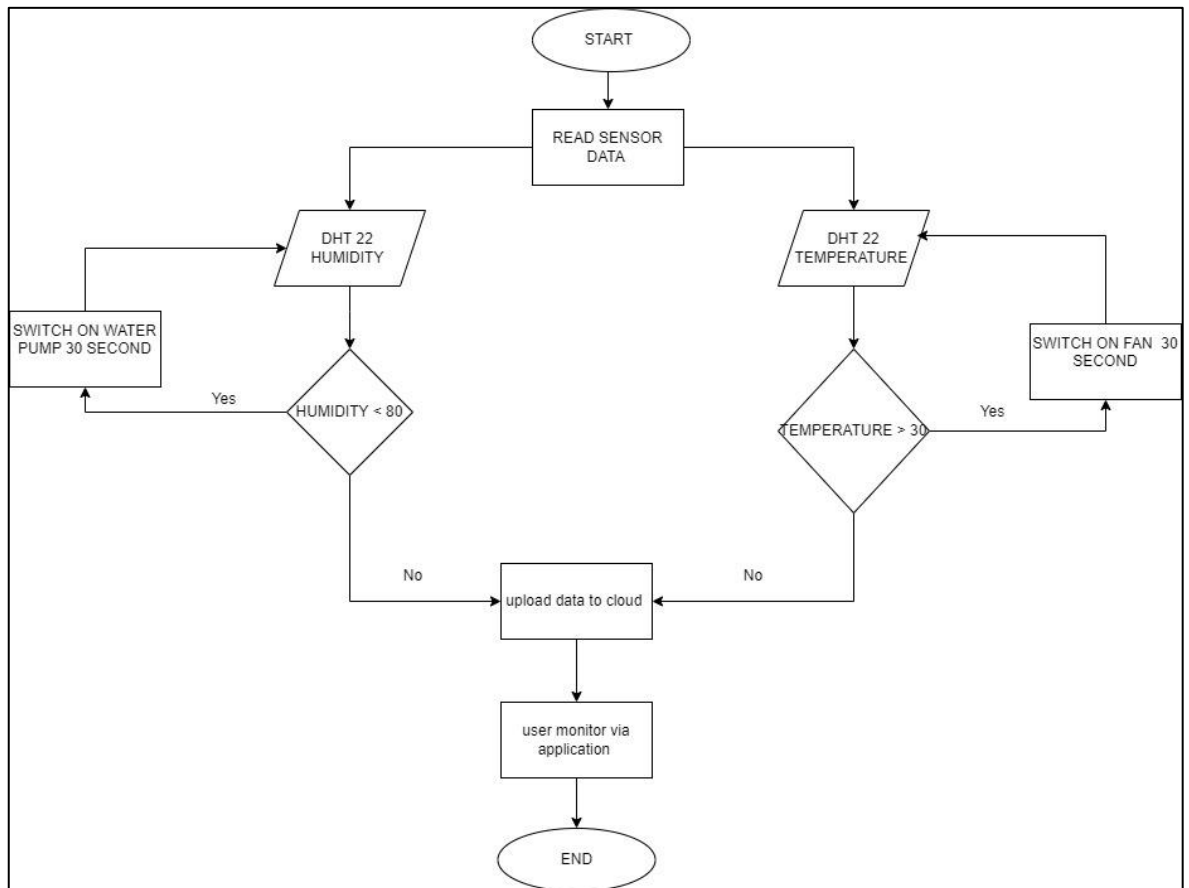


Figure 3.5 hardware Flowchart

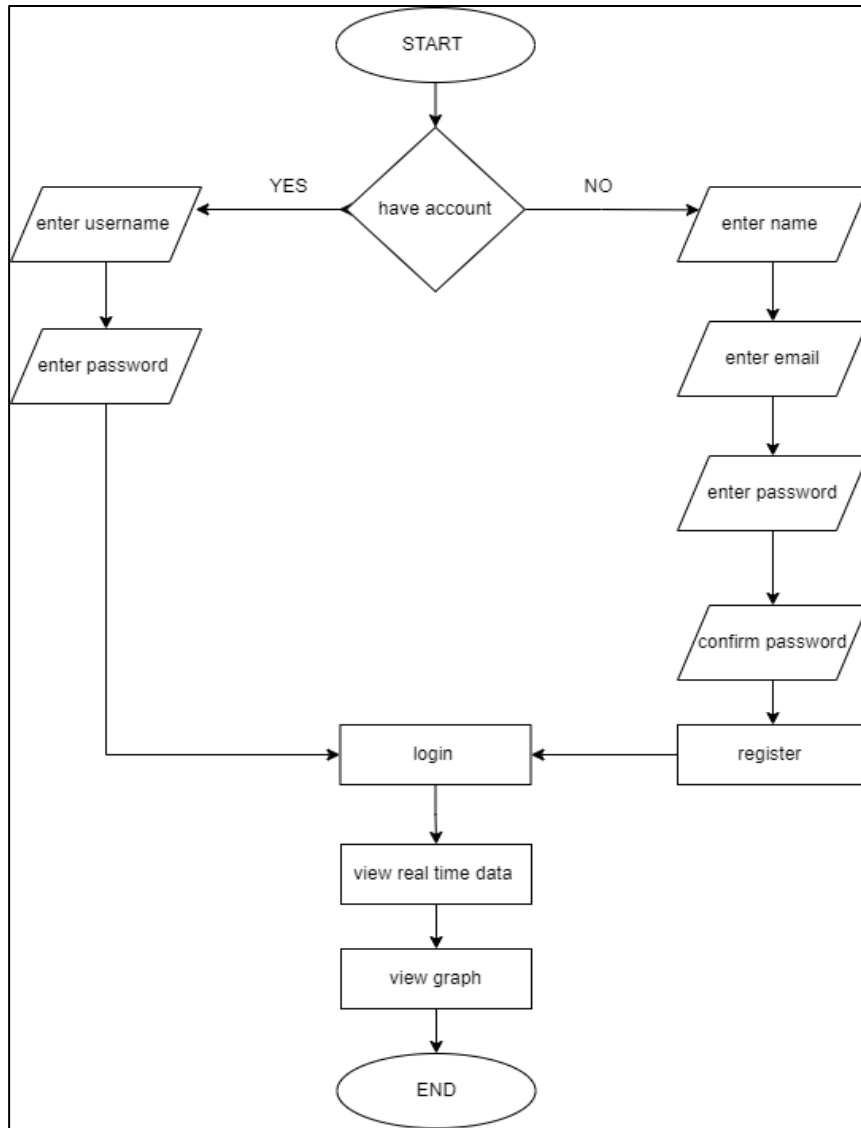


Figure 3.6 System flowchart

3.4.3 Context diagram

The context diagram displays the system's context and limits, as well as the system's relationship with external entities by depicting the information flow between the two. The context diagram for the suggested system is shown in Figure 3.5. ESP8266 microcontroller and the user are two examples of external entities in this system. While sending the action to the system to be processed and logging into the user account, the user will receive the humidity and temperature data of the mushroom house in real-time. The ESP8266 will receive the action, process and communicate the activated fan and water pump to the system, as well as send the value-added data it receives from the sensor.

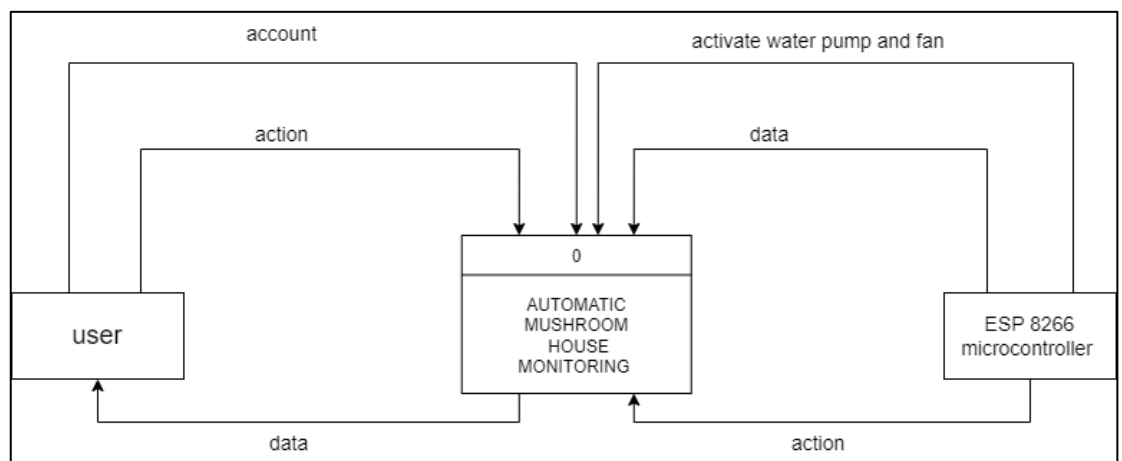


Figure 3.7 System context Diagram

3.4.4 Use case diagram.

Figure 3.6 depicts the relationships between the system, actors, and the use case in the use case diagram, which is a visual representation of the system's behaviour. The activities that the actors could carry out through the system are described in the use case. The user will be able to sign in to this system, access real-time data, and view a graph or report. Both the user and the developer will be controlling the error; the user will contact the developer if any error happens in the system, and the developer will offer the fix for the error. The developer handles the software updates for the smart application.

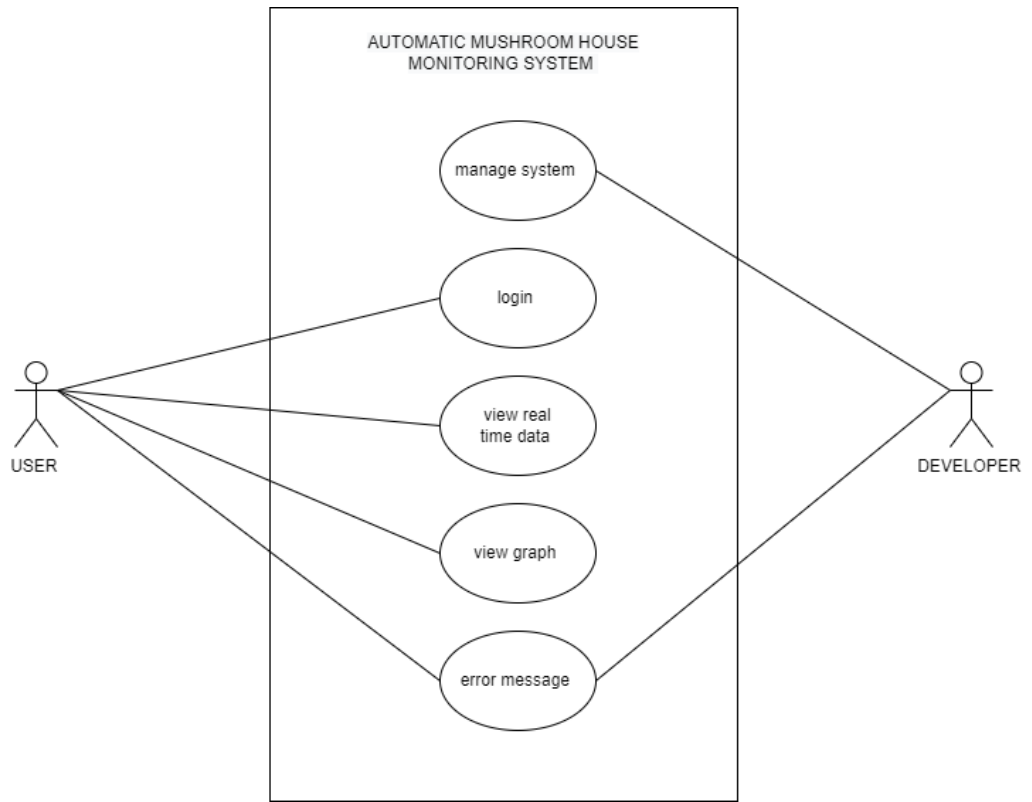


Figure 3.8 system Context Diagram

3.4.5 Activity diagram

Figure 3.7 depicts the hardware activity diagram part of the automatic monitoring process, the ESP8266 microcontroller will receive the signal from the DHT22 sensor. The microcontroller's connected programming language code will then analyse the signal to determine whether the temperature and humidity have reached their maximum levels. The fan and water pump will stop running after 30 seconds, at which point the sensor will start receiving data once more. If the value still reaches the maximum value, the action will be repeated.

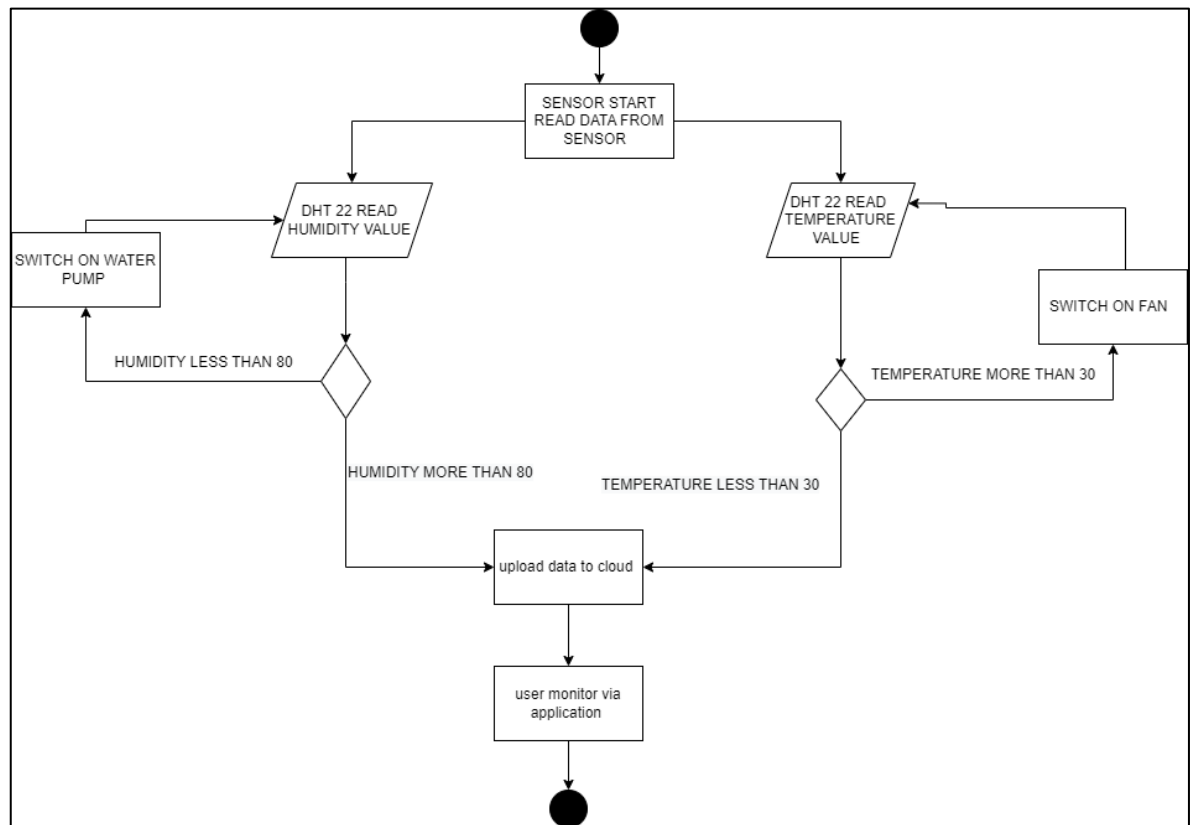


Figure 3.9 Hardware activity diagram

The system's activity diagram is shown in Figure 3.8. Users who want to use this system should take advantage of this activity. To view current temperature and humidity readings, the user must first log into the system. However, if the user doesn't already have an account, they can register first before using the application. view a graph or report in addition to seeing real-time data.

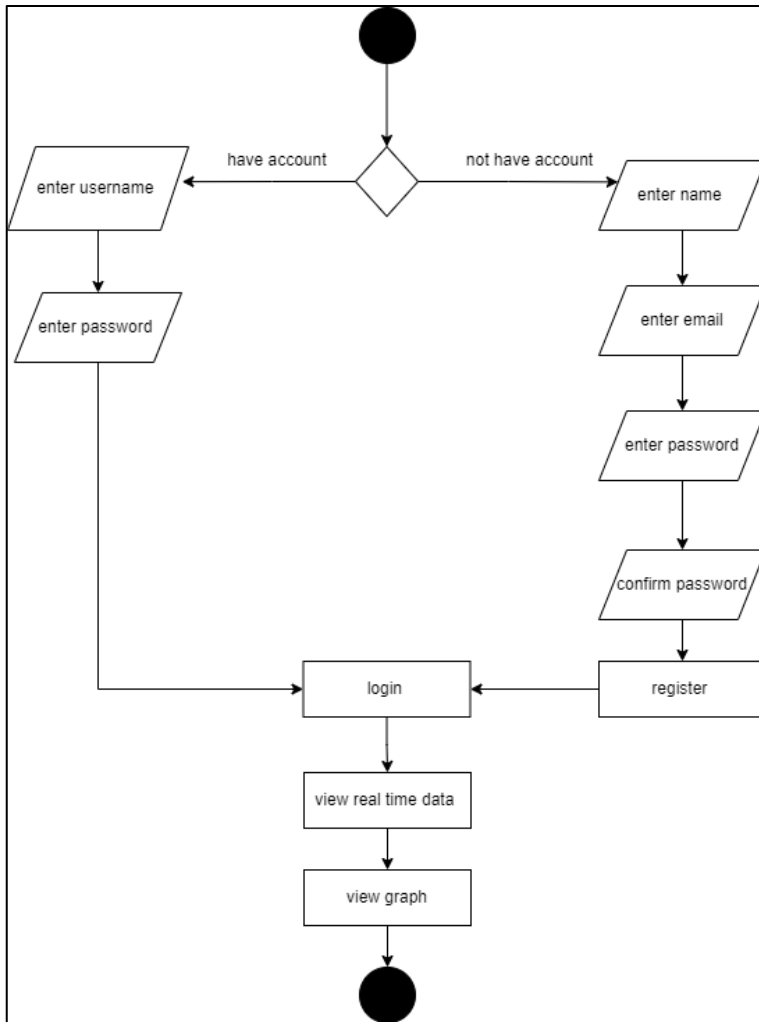


Figure 3.10 System Activity Diagram

3.5 Data Design

3.5.1 ERD

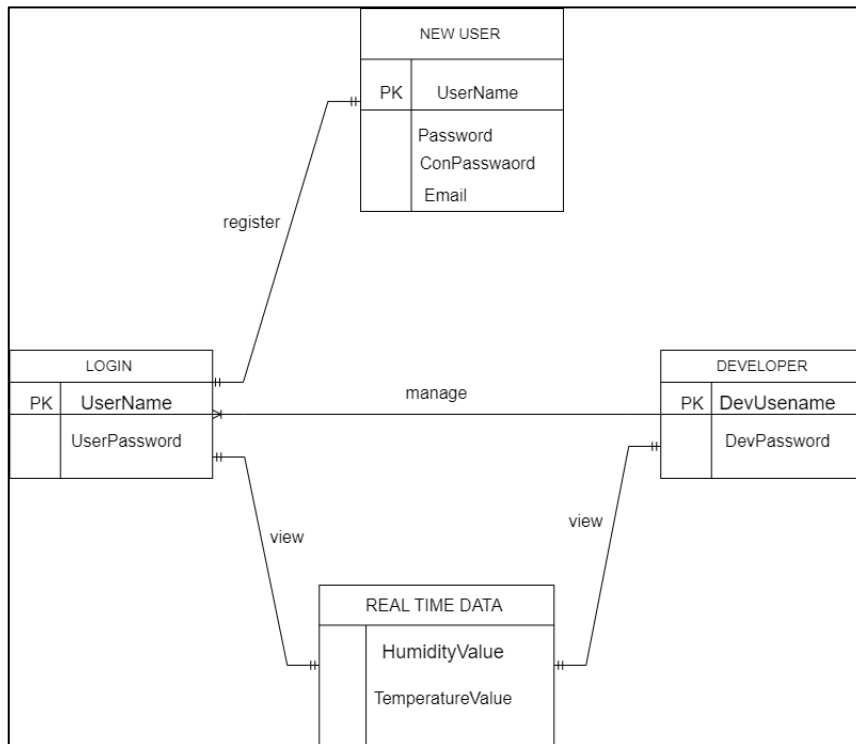


Figure 3.11 ERD diagram

3.5.2 Data base dictionary

3.5.2.1 Data dictionary for register new user.

Table 3.2 Register Data Dictionary

Attribute	Data type / size	Description	Example
Password	Varchar (30)	Password for the user account	Abc123
ConPassword	Varchar (30)	Retype password for confirmation	Abc123
Email	Email (50)	Email of the user	User123@gmail.com

3.5.2.2 Data dictionary for login

Table 3.3 login Data Dictionary

Attribute	Data type / size	Description	Example
Email	Email (50)	Email of the user	User123@gmail.com
Password	Varchar (30)	Password for the user account	Abc123

3.5.2.3 Data dictionary for real time data

Table 3.4 Real time Data Dictionary

Attribute	Data type / size	Description	Example
HumidityValue	Float (10)	Value of the humidity in mushroom house	70
TemperatureValue	Flat (10)	Value of the temperature in mushroom house	32

3.6 Hardware and software

3.6.1 Software

3.6.1.1 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software application that allows users to write, upload, and execute code on a microcontroller using an Arduino. The IDE supports the C and C++ programming languages and offers a simple and user-

friendly interface for creating Internet of Things (IoT) project. In addition to these microcontrollers, the Arduino IDE supports a wide range of third-party boards and microcontrollers that are compatible with the Arduino platform, such as the ESP8266 and ESP32, which are commonly used in Wi-Fi and IoT projects. Figure 3.12 above is the logo of the Arduino IDE software.



Figure 3.12 Arduino IDE software

3.6.1.2 Android Studio

The Android Studio IDE is based on the IntelliJ IDEA platform and supports the Java, Kotlin, and C++ programming languages, as well as the Android NDK (Native Development Kit). It also works with a wide variety of Android devices, such as smartphones, tablets, Android TV, and Android wear. Furthermore, Android Studio includes a set of tools for publishing apps to the Google Play Store, such as the ability to sign and export APKs (Android Package Kits) and generate App Bundles, a new publishing format that allows developers to deliver smaller, more optimised apps to users. The Android Studio is a powerful and feature-rich development environment that enables developers to create high-quality and feature-rich Android applications easily and efficiently. Figure 3.13 above is the latest version of the software, Android Studio Dolphin 2021.3.1, was released in December 2021.



Figure 3.13 Android Studio Dolphin 2021.3.1

3.6.1.3 Firebase

Firebase, a Google-supported app development framework, allows developers to create iOS, Android, and Web apps. Firebase offers tools for tracking statistics, analysing reports, troubleshooting app problems, and executing marketing and product experiments. Firebase Authentication makes it easier for programmers to create secure authentication systems while also improving user sign-in and onboarding. Because of the cloud-hosted NoSQL Firebase Realtime Database, data may be saved and updated in real time between users. Because the data is linked in instantly across all clients, it is available even when an app is offline. Figure 42 shows the firebase logo.



Figure 3.14 Firebase Logo

3.6.2 Hardware

3.6.2.1 NodeMCU 8266

The ESP8266 microcontroller is at the heart of the NodeMCU 8266 development board. The ESP8266 is a low-cost Wi-Fi microcontroller with an integrated TCP/IP

protocol stack for connecting devices to Wi-Fi networks and the internet. The NodeMCU 8266 development board is intended to make prototyping and development of IoT (Internet of Things) projects simple. It includes a USB-to-Serial converter that allows it to be easily connected to a computer for programming and debugging. The board has a variety of input and output pins for connecting sensors, actuators, and other devices.

Furthermore, the NodeMCU 8266 development board is a versatile and powerful tool for creating IoT projects. It is the most widely used microcontroller for IoT projects because it is small and inexpensive. It's an excellent choice for hobbyists, students, and professionals interested in developing their own IoT projects. Figure 3.14 above show the NodeMCU 8266 microcontroller.

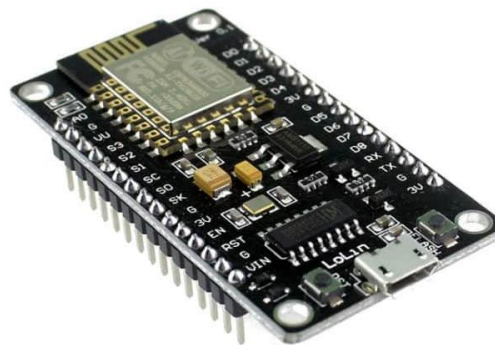


Figure 3.15 NodeMCU 8266

3.6.2.2 DHT 22 Sensor

The DHT22 sensor is a digital humidity and temperature sensor that measures the surrounding air using a capacitive humidity sensor and a thermistor. It has a relative humidity (RH) range of 0-100% and a temperature range of -40 to 80°C. It uses a single-wire digital interface to communicate with a microcontroller or single-board computer, making it simple to integrate into a wide range of projects. Figure 3.15 above is DHT22 sensor.



Figure 3.16 DHT22 sensor

3.6.2.3 Micro Submersible Water Pump

A micro submersible water pump is a small electric pump that is submerged in water. These pumps are typically powered by direct current motors and can be controlled by a switch, a microcontroller, or a computer. They are available in a variety of sizes and designs, with varying flow rates and head pressures. Some micro submersible water pumps are compact and portable, making them suitable for use in a variety of situations. Figure 3.16 above is micro submersible water pump.



Figure 3.17 Micro Submersible Water Pump

3.6.2.4 DC Cooling Exhaust Fan

A direct current (DC) cooling exhaust fan is a type of fan that uses direct current (DC) electricity to cool electronic devices and systems by exhausting hot air from the enclosure. DC cooling exhaust fans are typically smaller in size than their AC counterparts, and they operate more efficiently and have a longer life span. They can be controlled by a switch, a microcontroller, or a computer, among other things. They are available in a variety of sizes and designs, with varying flow rates and static pressure. Some fans are designed to be small and portable, making them useful in a variety of situations. Figure 3.17 above is DC) cooling exhaust fan.



Figure 3.18 DC Cooling Exhaust Fan

3.6.2.5 One channel relay

A relay is a type of electrical switch that can be controlled by an external voltage. A single channel relay has a single set of contacts that can be opened or closed by passing a voltage through the relay coil. Typically, the contacts are used to control the flow of electrical current in a circuit, allowing a low power control circuit to switch a higher power load. The relay is commonly found in control circuits.



Figure 3.19 One channel relay

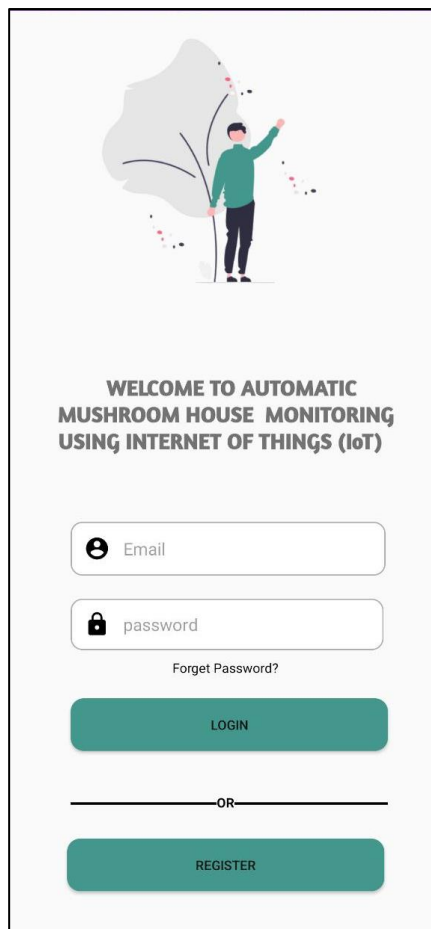
3.7 Proof of Initial Concept

3.7.1 Smart Mobile Application.

Android Studio was used to create the smart application for monitoring the mushroom house. Interfaces are classified into several types, including the login page, the register page, the main page, the real-time monitoring page, and the graph page.

3.7.1.1 Login

The user will first see this interface before seeing the value of real-time temperature and humidity data, as shown in the figure 3.19 below. The user must sign in with their email address and password. A user can register for the first time by clicking on the register button, which will take them to the registration page.

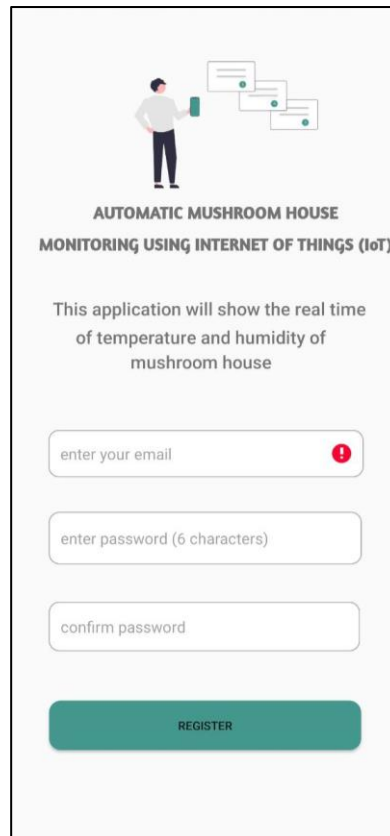


The image shows a mobile application login screen. At the top, there is an illustration of a person in a green shirt holding a large leaf with small red and white dots. Below the illustration, the text reads: "WELCOME TO AUTOMATIC MUSHROOM HOUSE MONITORING USING INTERNET OF THINGS (IoT)". There are two input fields: "Email" with an envelope icon and "password" with a lock icon. Below the password field is a link that says "Forget Password?". There are two large teal buttons: "LOGIN" and "REGISTER". A horizontal line with "OR" in the center separates the two buttons.

Figure 3.20 Login page

3.7.1.2 Register

A first-time user must first register their email address before they can access the application. As shown in the figure 3.20 below, the user must enter their username, email address, and password. After they complete all the fields, they can click on the register button, and it will bring them back to the login page.



The screenshot shows a mobile application interface for "AUTOMATIC MUSHROOM HOUSE MONITORING USING INTERNET OF THINGS (IoT)". At the top, there is an illustration of a person holding a smartphone, with several floating boxes representing data or notifications. Below the illustration, the text reads "AUTOMATIC MUSHROOM HOUSE MONITORING USING INTERNET OF THINGS (IoT)". Underneath, a descriptive sentence states: "This application will show the real time of temperature and humidity of mushroom house". The registration form consists of three input fields: "enter your email" (with a red exclamation mark icon on the right), "enter password (6 characters)", and "confirm password". At the bottom of the form is a prominent green button labeled "REGISTER".

Figure 3.21 Register Page

3.7.1.3 Main page

After completing the login process, the application will redirect the user to the application's main page. The user will see some of the application's explanation and purpose as figure 3.21 below. On this page, there is only one button, and the user must click it to view the current humidity and temperature levels in the mushroom house.

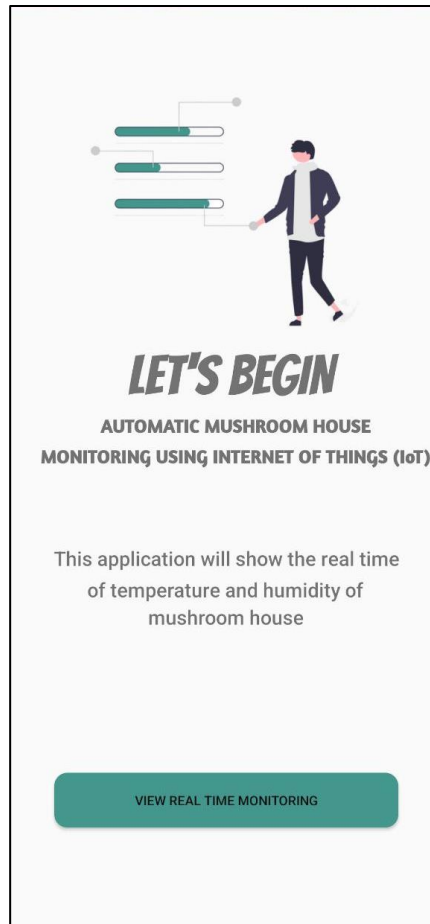


Figure 3.22 Main page

3.7.1.4 Real Time Monitoring Page

On this page, users can view real-time data on the humidity and temperature of the mushroom house. The user can also monitor the status of the water pump and fan and turn them on and off. User also able to switch on the water pump and fan manually. On the bottom of the page as figure 3.22, users can click on the "View Graph" button to view the history of the temperature and humidity of the mushroom house.

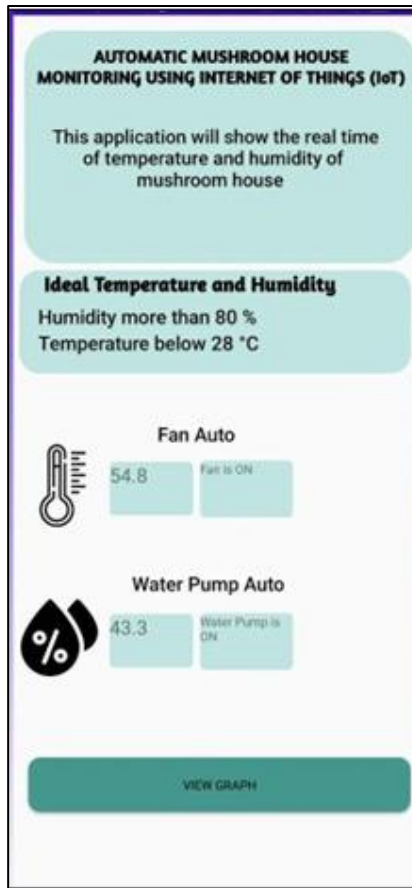


Figure 3.23 Real Time Monitoring Page

3.7.1.5 Graph Page

The user will see a report of the mushroom house's temperature and humidity on this page. It will be easier to see the average temperature and humidity by hour using this graph.

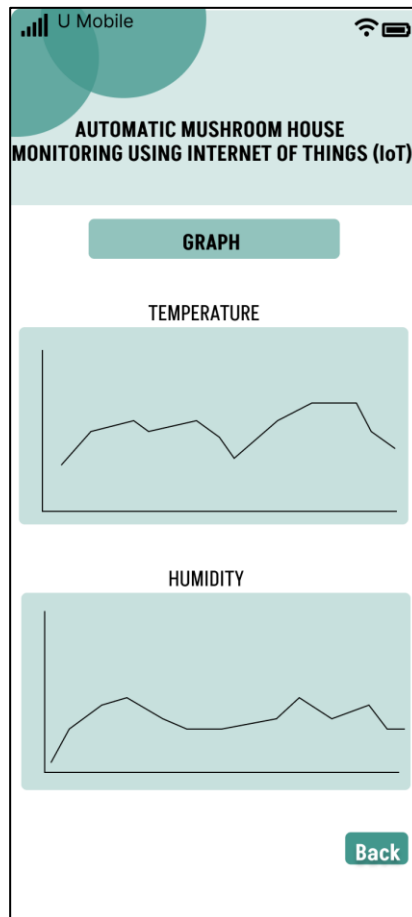


Figure 3.24 Graph /Report page

3.7.2 Hardware setup

All the hardware was assembled in accordance with the purpose described in this chapter 3.4, and all hardware decisions were used to complete the design. The dht22 sensor will detect the ambient temperature and humidity. The water pump will be activated if the temperature falls below a certain threshold. When the humidity is at its highest, the fan will turn on. Table 3.1 shows the best and worst temperature and humidity values.

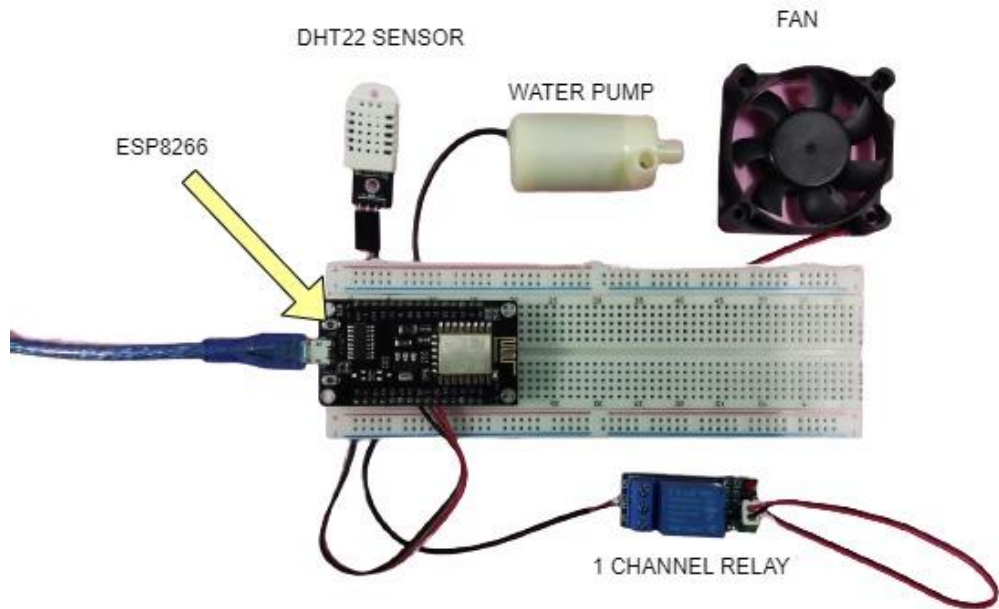


Figure 3.25 Hardware setup of Proof and initial concept

3.8 Testing /Validation Plan

Testing occurs only after the development process is complete. Both functional and non-functional needs will be considered when assessing the system's performance.

User Acceptance Testing (UAT)					
No	Module	Activities	Pass	Fail	Comment
1	Login	user login			
		user register			
		Able to get in into real time data interface			
2	real time data	value of temperature			
		value of humidity			
		notes of ideal humidity and temperature			
3	hardware functioning	water pum			
		fan			
		water hose			
4	navigation	login interface to register			
		register to login			
		login ito real time data			
		real time data to report			
		report to real time data			
5	report	graph of humidity			
		graph of temperature			

3.9 Potential use of Proposed Solution

The potential application of mushroom house monitoring using Internet of Things (IoT) technology is to optimise mushroom growing conditions and increase the efficiency and yield of mushroom cultivation. Some keyways in which IoT can help with this include:

1. Real-time monitoring: Internet of Things sensors can be used to continuously monitor environmental factors such as temperature and humidity, providing real-time data that can be used to adjust growing conditions as needed.

2. IoT devices can be used to automate the control of various systems in the mushroom house, such as heating and cooling. This enables the growing conditions to be automatically optimised, eliminating the need for manual intervention.
3. Savings on labour: By automating control systems, the need for manual labour can be reduced, resulting in cost savings. Furthermore, by optimising the growing conditions, the yield of mushrooms can be increased, resulting in long-term cost savings.
4. Remote monitoring: The mushroom house can be monitored remotely using IoT, allowing growers to keep an eye on the conditions inside the house from anywhere. This can be beneficial for growers who are unable to physically attend the mushroom house.

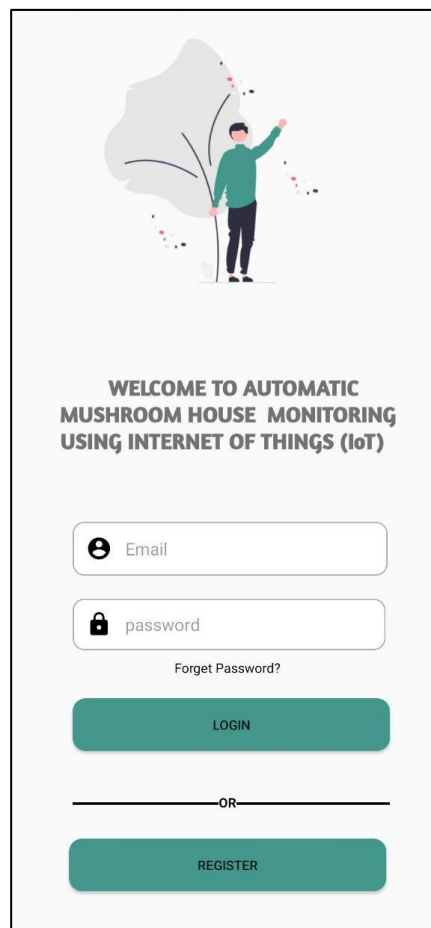
CHAPTER 4

RESULT AND DISCUSSION

4.1 SMART MOBILE APPLICATION OF AUTOMATIC MUSHROOM HOUSE MONITORING SYSTEM (MHMS)

4.1.1 Login Interface

The user will first see this interface before seeing the value of real-time temperature and humidity data, as shown in the figure 4.2 below. The user must sign in with their email address and password. A user can register for the first time by clicking on the register button, which will take them to the registration page.



WELCOME TO AUTOMATIC
MUSHROOM HOUSE MONITORING
USING INTERNET OF THINGS (IoT)

Email

password

Forgot Password?

LOGIN

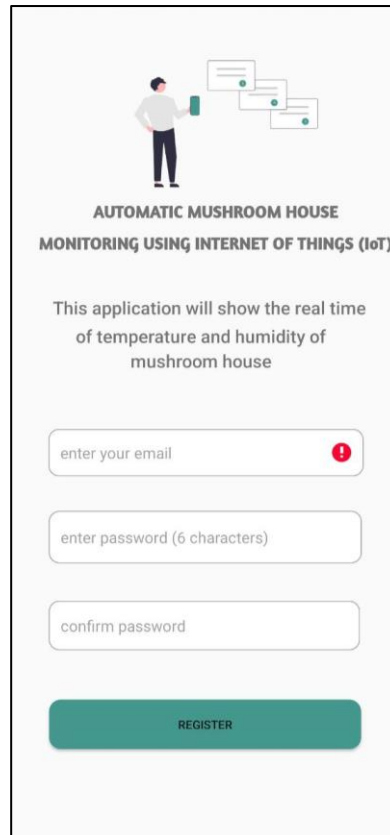
OR

REGISTER

Figure 4.1 login interface

4.1.2 Register Interface

A first-time user must first register their email address before they can access the application. As shown in figure 4.3 below, the user must enter their username, email address, and password. After they complete all the fields, they can click on the register button, and it will bring them back to the login page.



The screenshot displays a registration form for an application titled "AUTOMATIC MUSHROOM HOUSE MONITORING USING INTERNET OF THINGS (IoT)". At the top, there is an illustration of a person holding a smartphone, with several floating document icons. Below the illustration, the application title is centered. A descriptive text states: "This application will show the real time of temperature and humidity of mushroom house". The form consists of three input fields: "enter your email" (with a red exclamation mark icon on the right), "enter password (6 characters)", and "confirm password". At the bottom of the form is a prominent green button labeled "REGISTER".

Figure 4.2 registers interface.

4.1.3 Main interface

After completing the login process, the application will redirect the user to the application's main page. The user will see some of the application's explanation and purpose as figure 4.4 below. On this page, there is only one button, and the user must click it to view the current humidity and temperature levels in the mushroom house.

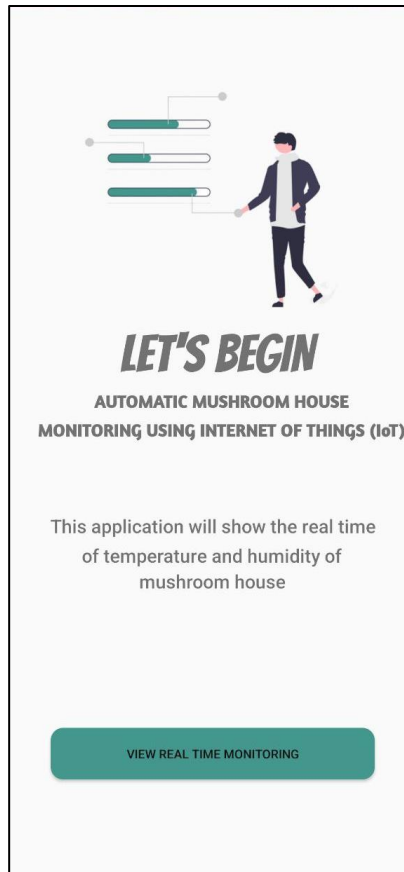


Figure 4.3 main interface

4.1.4 Real time monitoring interface

On this page, users can view real-time data on the humidity and temperature of the mushroom house. The user can also monitor the status of the water pump and fan and turn them on and off. User also able to switch on the water pump and fan manually. On the bottom of the page as figure 4.5, users can click on the "View Graph" button to view the history of the temperature and humidity of the mushroom house.

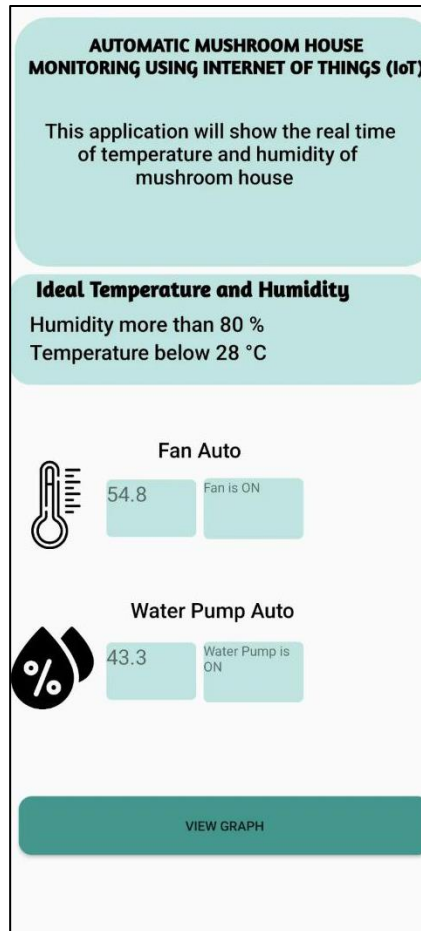


Figure 4.4 Realtime interface

4.1.5 Graph Report

Based on the figures 4.5 and 4.6 show the page that users can view graph for the real-time data on the humidity and temperature of the mushroom house.

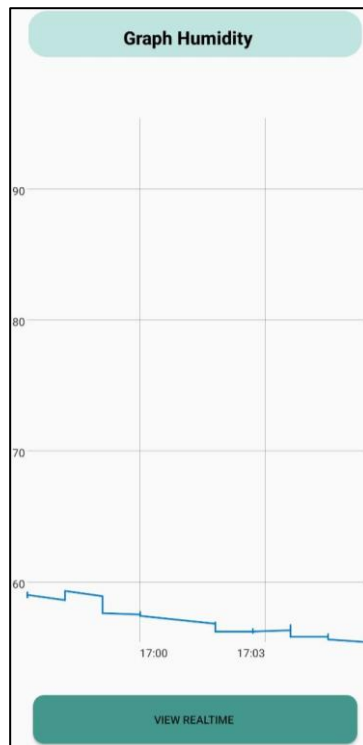


Figure 4.5 Graph interface for humidity

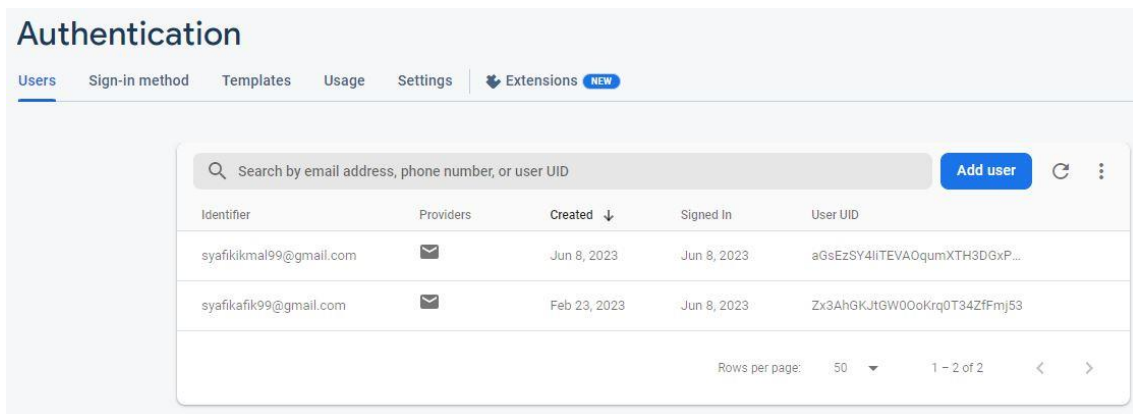


Figure 4.6 Graph interface for Temperature

4.2 Implementation

4.2.1 Authentication

This project uses the Firebase authentication service. Firebase Authentication offers backend services, simple SDKs, and pre-built UI libraries to authenticate users within your app. It supports authentication using passwords, phone numbers, and well-known federated identity providers such as Google, Facebook, and Twitter, among others. But for this project, only an email address and password are required to log in and register for the smart mobile application. Figure 4.7 demonstrates a user who has registered to use the smart mobile application and Figure 4.8 represents the authentication code for connecting android studio to firebase.



The screenshot shows the 'Authentication' page in the Firebase console. It features a search bar at the top with the text 'Search by email address, phone number, or user UID' and an 'Add user' button. Below the search bar is a table with the following columns: Identifier, Providers, Created, Signed In, and User UID. The table contains two rows of user data.

Identifier	Providers	Created	Signed In	User UID
syafikikmal99@gmail.com	📧	Jun 8, 2023	Jun 8, 2023	aGsEzSY4iITEVAOqumXTH3DGxP...
syafikafik99@gmail.com	📧	Feb 23, 2023	Jun 8, 2023	Zx3AhGKJtGW00oKrqQT34ZfFm]53

At the bottom of the table, there is a 'Rows per page' dropdown set to '50' and a pagination indicator '1 - 2 of 2'.

Figure 4.7 Authentication database



The screenshot shows the MainActivity.java code in Android Studio. The code is as follows:

```
1 package com.example.myapplicationtest3;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     EditText inputEmail, inputPassword;
8     Button button;
9     String emailPattern = "[a-zA-Z0-9_\\.]+@[a-z]+\\.+[a-z]+";
10    ProgressDialog progressDialog;
11
12    FirebaseAuth mAuth;
13    FirebaseUser mUser;
```

Figure 4.8 Authentication Database code in Android Studio

4.2.2 Real Time Database

Firestore was chosen to serve as the database for this Automatic Mushroom House Monitoring System (MHMS). It has been determined that the temperature and humidity levels inside the mushroom house can be obtained from the real-time data base. Additionally, it displays whether the fan and water pump are on or off in their respective states. Based on the figure 4.8 is the Realtime data base for MHMS and figure 4.9 is the code for gain the temperature and humidity value.

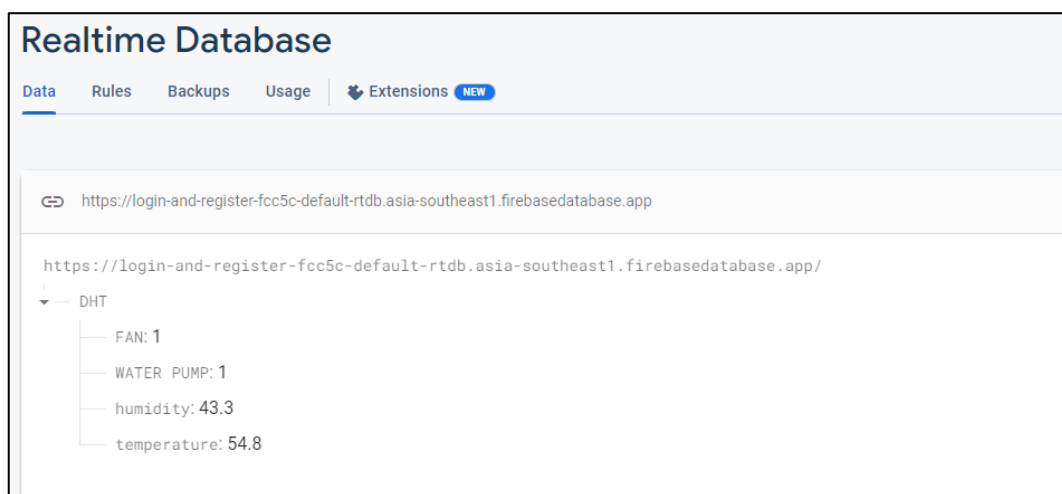


Figure 4.9 Realtime Database for temperature, humidity, fan, and water pump status

```

CODE_1.ino CODE_1.ino
103
104 // Write an Float number on the database path test/float
105 if (Firebase.RTDB.setFloat(&fbdo, "DHT/temperature", t)) {
106     // Serial.println("PASSED");
107     Serial.print("Temperature to database: ");
108     Serial.print(t);
109     Serial.println("°C");
110     Serial.println(".....");
111
112 } else {
113     Serial.println("FAILED");
114     Serial.println("REASON: " + fbdo.errorReason());
115 }
116
117 if (!isnan(t)) {
118     Serial.print("Temperature: ");
119     Serial.print(t);
120     Serial.println(" °C");
121
122
123     if (t > 36) {
124         digitalWrite(FAN_RELAY_PIN, HIGH);
125         Serial.println("Fan is ON");
126         Firebase.RTDB.setFloat(&fbdo, "DHT/FAN", true);
127     }
128     else {
129         digitalWrite(FAN_RELAY_PIN, LOW);
130         Serial.println("Fan is OFF");
131         Firebase.RTDB.setFloat(&fbdo, "DHT/FAN", false);
132     }
133 }

```

Figure 4.10 Code in Arduino IDE to gain the temperature and humidity value.

```

135
136 if (!isnan(h)) {
137     Serial.print("Humidity: ");
138     Serial.print(h);
139     Serial.println("%");
140
141     if (h < 60) {
142         digitalWrite(PUMP_RELAY_PIN, HIGH);
143         Serial.println("Water pump is ON");
144         Firebase.RTDB.setFloat(&fbdo, "DHT/WATER PUMP", true);
145         Serial.println("////////////////////////////////////////");
146     } else {
147         digitalWrite(PUMP_RELAY_PIN, LOW);
148         Serial.println("Water pump is OFF");
149         Firebase.RTDB.setFloat(&fbdo, "DHT/WATER PUMP", false);
150         Serial.println("////////////////////////////////////////");
151     }
152 }
153 }
154 }
155 }
156 }
157 }
158 }

```

Figure 4.11 Rules for power on and off Water Pump

4.2.3 Smart mobile application source code

Android Studio was used as a development environment to create the code for the automatic mushroom house monitoring system (MHMS) application, which was the software used to build this system. Android Studio offers first-rate Kotlin support. It includes utilities for converting Java-based code to Kotlin. Figures 4.11 through 4.19 depict the source code for the intelligent mobile application.

```
1 package com.example.myapplicationtest3;
2
3 import ...
20
21 public class MainActivity extends AppCompatActivity {
22
23     EditText inputEmail, inputPassword;
24     Button button;
25     String emailPattern = "[a-zA-Z0-9, _-]+@[a-z]+\\.+[a-z]+";
26     ProgressDialog progressDialog;
27
28     FirebaseAuth mAuth;
29     FirebaseUser mUser;
30
31     Button btn1;
32
33     @Override
34     protected void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_main);
37
38         btn1 = findViewById(R.id.button2);
39
40         btn1.setOnClickListener(new View.OnClickListener() {
41             @Override
42             public void onClick(View view) {
43                 Intent intent = new Intent(getApplicationContext(), register.class);
44                 startActivity(intent);
45             }
46         });
47     }
48 }
```

Figure 4.12 Source Code for Login Page

```

48     inputEmail = findViewById(R.id.editTextTextPersonName);
49     inputPassword = findViewById(R.id.editTextTextPassword);
50     button = findViewById(R.id.button);
51     progressDialog = new ProgressDialog( context: this);
52     mAuth = FirebaseAuth.getInstance();
53     mUser = mAuth.getCurrentUser();
54
55     button.setOnClickListener(new View.OnClickListener() {
56         @Override
57         public void onClick(View view) { performLogin(); }
60     });
61 }
62
63 private void performLogin() {
64
65     String email = inputEmail.getText().toString();
66     String password = inputPassword.getText().toString();
67
68
69     if (!email.matches(emailPattern)) {
70         inputEmail.setError("Enter Correct Email");
71     } else if (password.isEmpty() || password.length() < 6) {
72
73     } else {
74         progressDialog.setMessage("Please Wait For Login...");
75         progressDialog.setTitle("Login");
76         progressDialog.setCanceledOnTouchOutside(false);
77         progressDialog.show();

```

Figure 4.13 Source Code for Login Page

```

78     mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
79
80         @Override
81         public void onComplete(@NonNull Task<AuthResult> task) {
82
83             if(task.isSuccessful())
84             {
85                 progressDialog.dismiss();
86                 sendUserToNextActivity();
87                 Toast.makeText( context: MainActivity.this, text: "Login Successful", Toast.LENGTH_SHORT).show();
88             }else
89             {
90                 progressDialog.dismiss();
91                 Toast.makeText( context: MainActivity.this, text: "Login Fail"+task.getException(), Toast.LENGTH_SH
92             }
93         }
94     });
95 }
96
97
98 }
99 private void sendUserToNextActivity() {
100
101     Intent intent=new Intent( packageContext: MainActivity.this, welcome_interface.class);
102     intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
103     startActivity(intent);
104 }
105 }

```

Figure 4.14 Source Code for Login Page

```

1      package com.example.myapplicationtest3;
2
3      import ...
19
20     public class register extends AppCompatActivity {
21
22         EditText inputEmail,inputPassword,inputCConformPassword;
23         Button button;
24         String emailPattern = "[a-zA-Z0-9,-_]+@[a-z]+\\.+[a-z]+";
25         ProgressDialog progressDialog;
26         FirebaseAuth mAuth;
27         FirebaseUser mUser;
28
29         @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_register);
33
34         inputEmail=findViewById(R.id.editTextTextEmailAddress);
35         inputPassword=findViewById(R.id.editTextTextPassword);
36         inputCConformPassword=findViewById(R.id.editTextTextPassword2);
37         button=findViewById(R.id.button);
38         progressDialog=new ProgressDialog( context: this);
39         mAuth=FirebaseAuth.getInstance();
40         mUser=mAuth.getCurrentUser();
41

```

Figure 4.15 Source Code for Register Page

```

42         button.setOnClickListener(new View.OnClickListener() {
43             @Override
44             public void onClick(View view) { PerformAuth(); }
45         });
46     }
47
48     private void PerformAuth() {
49
50         String email=inputEmail.getText().toString();
51         String password=inputPassword.getText().toString();
52         String confirmPassword=inputCConformPassword.getText().toString();
53
54         if (!email.matches(emailPattern))
55         {
56             inputEmail.setError("Enter Correct Email");
57         }else if (password.isEmpty() || password.length()<6)
58         {
59             inputPassword.setError("Enter Proper Password");
60         }else if (!password.equals(confirmPassword))
61         {
62             inputCConformPassword.setError("Password Not Match");
63         }else
64         {
65
66         }
67
68     }

```

Figure 4.16 Source Code for Register Page

```

68     {
69         progressDialog.setMessage("Please Wait For Registration...");
70         progressDialog.setTitle("Registration");
71         progressDialog.setCancelable(false);
72         progressDialog.show();
73     }
74     mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>
75     {
76         @Override
77         public void onComplete(@NonNull Task<AuthResult> task) {
78             if(task.isSuccessful())
79             {
80                 progressDialog.dismiss();
81                 sendUserToNextActivity();
82                 Toast.makeText(context,register.this, text, "Registration Successful", Toast.LENGTH_SHORT).show();
83             }else
84             {
85                 progressDialog.dismiss();
86                 Toast.makeText(context,register.this, text, ""+task.getException(), Toast.LENGTH_SHORT).show();
87             }
88         }
89     });
90 }
91 private void sendUserToNextActivity() {
92
93     Intent intent=new Intent( context,register.this,MainActivity.class);
94     intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
95     startActivity(intent);

```

Figure 4.17 Source Code for Register Page

```

1  package com.example.myapplicationtest3;
2
3  import ...
9
10 public class welcome_interface extends AppCompatActivity {
11
12     Button btn3;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_welcome_interface);
18
19         btn3 = findViewById(R.id.button3);
20
21         btn3.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View view) {
24                 Intent intent = new Intent(getBaseContext(),Real_time.class);
25                 startActivity(intent);
26             }
27         });
28     }
29 }

```

Figure 4.18 Source Code for Homepage


```

3 import ...
0
1 public class Real_time extends AppCompatActivity {
2
3     TextView temperature;
4     TextView humidity;
5     TextView fanStatus;
6     TextView waterPumpStatus;
7     DatabaseReference reff;
8
9     Switch fanSwitch;
10    Switch waterPumpSwitch;
11
12
13    @Override
14    protected void onCreate(@Nullable Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_real_time);
17
18        temperature = (TextView) findViewById(R.id.temp1);
19        humidity = (TextView) findViewById(R.id.textView17);
20
21        fanStatus = findViewById(R.id.textView16);
22        waterPumpStatus = findViewById(R.id.textView18);
23
24        fanSwitch = findViewById(R.id.switch1);
25        waterPumpSwitch = findViewById(R.id.switch2);

```

Figure 4.19 Source Code for Real Time page

```

47 reff = FirebaseDatabase.getInstance("https://login-and-register-fcc5c-default-rtdb.asia-southea
48 reff.addValueEventListener(new ValueEventListener() {
49     @Override
50     public void onDataChange(@NonNull DataSnapshot snapshot) {
51
52         String status = snapshot.child("temperature").getValue().toString();
53         String statuss = snapshot.child("humidity").getValue().toString();
54         String fanvalue = snapshot.child("FAN").getValue().toString();
55         String watervalue = snapshot.child("WATER PUMP").getValue().toString();
56
57         temperature.setText(status);
58         humidity.setText(statuss);
59
60         if (fanvalue.equals("1")) {
61             fanStatus.setText("Fan is ON");
62         } else {
63             fanStatus.setText("Fan is OFF");
64         }
65     }
66
67     if (watervalue.equals("1")) {
68         waterPumpStatus.setText("Water Pump is ON");
69     } else {
70         waterPumpStatus.setText("Water Pump is OFF");
71     }
72 }

```

Figure 4.20 Source Code for Real Time page

4.2.4 Hardware implementation

A key component of this system's hardware is the microcontroller. It serves as a controller for an integrated circuit designed to perform a specific function. This system employs the Node MCU esp8266 microcontroller board because its 34 digital input/output pins and other features make it optimal for sensor connection. The sensor type is DHT22. This microcontroller's Wi-Fi module is used to export and import sensor data to and from the Firebase database. After connecting the sensors to the Arduino microcontroller board, hardware components such as a breadboard, male-to-male jumper wires, and male-to-female jumper wires are utilised. First, the equipment required configuration. Male-to-male jumper cables were used to connect the breadboard terminals. The sensors and components are then connected to the breadboard so they can interface with the microcontroller. The code was uploaded to the node MCU esp8266 using a USB connection to the computer, and the serial monitor was utilised for debugging. Figures 4.20 until 4.24 depict the prototype of automatic monitoring system for mushroom houses (MHMS).

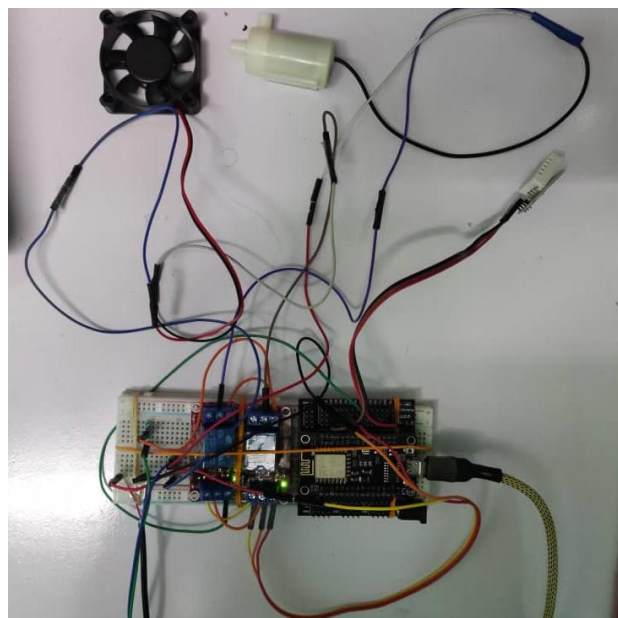


Figure 4.21 circuit diagram



Figure 4.22 prototype diagram



Figure 4.23 prototype diagram (node mcu esp8266 placement)



Figure 4.24 prototype diagram (DHT22 Sensor placement)



Figure 4.25 prototype diagram (Water Pump placement)

4.2.4.1 Hardware source code

The source code setup for each hardware is displayed in Figures 4.24 to 4.29.

```
1 #include <ESP8266WiFi.h>
2 #include <Firebase_ESP_Client.h>
3 #include "DHT.h"
4
5 //Provide the token generation process info.
6 #include "addons/TokenHelper.h"
7 //Provide the RTDB payload printing info and other helper functions.
8 #include "addons/RTDBHelper.h"
9
10 // Insert your network credentials
11 #define WIFI_SSID "POCO F3"
12 #define WIFI_PASSWORD "syafik99"
13
14 // Insert Firebase project API Key
15 #define API_KEY "AIzaSyCuRg3AKU10zk0TY_tjf_06oWkWyOmL_w"
16 // Insert RTDB URLdefine the RTDB URL */
17 #define DATABASE_URL "login-and-register-fcc5c-default-rtdb.asia-southeast1.firebaseio.app/"
18
19 //Define Firebase Data object
20 FirebaseData fbdo;
21 FirebaseAuth auth;
22 FirebaseConfig config;
23 bool signupOK = false;
24
25 #define DHTPIN D2 // Digital pin connected to the DHT sensor
26 #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
27
28 #define FAN_RELAY_PIN D5 // Arduino digital pin to control the fan relay
29 #define PUMP_RELAY_PIN D6 // Arduino digital pin to control the water pump relay
```

Figure 4.26 Setup data base code

```
31 #define DATABASE_PATH "/DHT"
32 String fireStatus = "";
33
34 // Initialize DHT sensor.
35 DHT dht(DHTPIN, DHTTYPE);
36
37 void setup()
38
39 {
40   Serial.begin(9600);
41   pinMode(DHTPIN, INPUT);
42   pinMode(FAN_RELAY_PIN, OUTPUT);
43   pinMode(PUMP_RELAY_PIN, OUTPUT);
44   dht.begin();
45
46   Serial.begin(115200);
47   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
48   Serial.print("Connecting to Wi-Fi");
49   while (WiFi.status() != WL_CONNECTED) {
50     Serial.print(".");
51     delay(300);
52   }
53   Serial.println();
54   Serial.print("Connected with IP: ");
55   Serial.println(WiFi.localIP());
56   Serial.println();
```

Figure 4.27 Setup for Wi-Fi connection


```

58  /* Assign the api key (required) */
59  config.api_key = API_KEY;
60
61  /* Assign the RTDB URL (required) */
62  config.database_url = DATABASE_URL;
63
64
65
66  /* Sign up */
67  if (Firebase.signUp(&config, &auth, "", "")) {
68      Serial.println("ok");
69      signupOK = true;
70  } else {
71      Serial.printf("%s\n", config.signer.signupError.message.c_str());
72  }
73
74  /* Assign the callback function for the long running token generation task */
75  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h
76
77  Firebase.begin(&config, &auth);
78  Firebase.reconnectWiFi(true);
79  }
80

```

Figure 4.28 Setup for firebase connection code

```

81  void loop() {
82      delay(5000);
83
84      float h = dht.readHumidity();
85      float t = dht.readTemperature();
86      // float temperature = dht.readTemperature();
87      // float humidity = dht.readHumidity();
88
89      if (Firebase.ready() && signupOK) {
90          if (Firebase.RTDB.setFloat(&fbdo, "DHT/humidity", h)) {
91              // Serial.println("PASSED");
92              Serial.print("Humidity to database:");
93              Serial.print(h);
94              Serial.println("%");
95          } else {
96              Serial.println("FAILED");
97              Serial.println("REASON: " + fbdo.errorReason());
98          }
99      }
100

```

Figure 4.29 Setup for read temperature and humidity code.

```

104 // Write an Float number on the database path test/float
105 if (Firebase.RTDB.setFloat(&fbdo, "DHT/temperature", t)) {
106     // Serial.println("PASSED");
107     Serial.print("Temperature to database: ");
108     Serial.print(t);
109     Serial.println("°C");
110     Serial.println(".....");
111
112 } else {
113     Serial.println("FAILED");
114     Serial.println("REASON: " + fbdo.errorReason());
115 }
116
117 if (!isnan(t)) {
118     Serial.print("Temperature: ");
119     Serial.print(t);
120     Serial.println(" °C");
121
122
123     if (t > 36 ) {
124         digitalWrite(FAN_RELAY_PIN, HIGH);
125         Serial.println("Fan is ON");
126         Firebase.RTDB.setFloat(&fbdo, "DHT/FAN", true);
127     }
128     else {
129         digitalWrite(FAN_RELAY_PIN, LOW);
130         Serial.println("Fan is OFF");
131         Firebase.RTDB.setFloat(&fbdo, "DHT/FAN", false);
132     }

```

Figure 4.30 Setup for power on/off fan code

```

136 if (!isnan(h)) {
137     Serial.print("Humidity: ");
138     Serial.print(h);
139     Serial.println(" %");
140
141     if (h < 60) {
142         digitalWrite(PUMP_RELAY_PIN, HIGH);
143         Serial.println("Water pump is ON");
144         Firebase.RTDB.setFloat(&fbdo, "DHT/WATER PUMP", true);
145         Serial.println("////////////////////////////////////////");
146
147     } else {
148         digitalWrite(PUMP_RELAY_PIN, LOW);
149         Serial.println("Water pump is OFF");
150         Firebase.RTDB.setFloat(&fbdo, "DHT/WATER PUMP", false);
151         Serial.println("////////////////////////////////////////");
152
153     }

```

Figure 4.31 Setup for power on/off water pump code

4.3 Testing and Discussion

The hardware and software are both tested and discussed to ensure that the authentication function, database connection, sensors, and smart mobile applications are in line with the system's goals. The hardware prototype, the intelligent mobile application, and an internet connection are required materials and instruments for testing. For exhaustive testing, the user acceptance test was performed on the system.

4.3.1 Automatic mushroom house monitoring system (MHMS) authentication

After a user successfully created an account via a mobile application, the authentication database is updated with the account information. Alternative-specific information was added to the database in real-time. After establishing an effective connection to the Firebase database, the data could be stored in databases. Figure 4.30 depicts the authentication generated by the user.

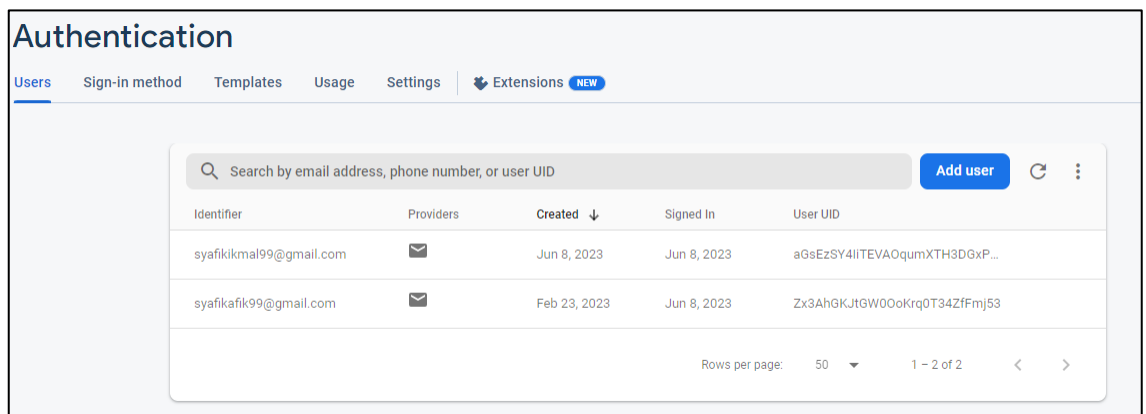


Figure 4.32 Authentication After User created.

4.3.2 Esp8066 connection to real time database

As shown in Figure 4.31, The hardware is operational after being uploaded via the Arduino IDE. In Figure 4.32, the node MCU esp8266 is connected to the internet and a real-time database, allowing for updated data regarding the status of the temperature, humidity, ventilation, and water pump. As depicted in Figure 4.33, the value has been effectively read and stored in the real-time database.


```
Writing at 0x00058000... (92 %)
Writing at 0x0005c000... (96 %)
Writing at 0x00060000... (100 %)
Wrote 551472 bytes (399578 compressed) at 0x00000000 in 35.2 seconds (effective 125.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 4.33 Code Uploading successful in the Node mcu ESP8266.

```
Output Serial Monitor x
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM7')
.....
Connected with IP: 192.168.69.20
ok
```

Figure 4.34 The ESP8266 successfully connected to the internet.

```
Temperature: 30.90 °C
Fan is OFF
Humidity: 74.60 %
Water pump is OFF
```

Figure 4.35 The ESP8266 Humidity, temperature, fan and water pump status reading successfully

```
Realtime Database
Data Rules Backups Usage Extensions NEW
https://login-and-register-fcc5c-default-rtdb.asia-southeast1.firebaseio.com/
https://login-and-register-fcc5c-default-rtdb.asia-southeast1.firebaseio.com/
└─ DHT
  └─ FAN: 1
  └─ WATER PUMP: 1
  └─ humidity: 43.3
  └─ temperature: 54.8
```

Figure 4.36 The Humidity, temperature, fan, and water pump status reading successfully stored in realtime database

4.4 Conclusion

The hardware for this system was initially assembled on a breadboard using the Node MCU esp8266, and then the development of the intelligent mobile application commenced. The connection to the Firebase database was established after the hardware and mobile application were configured using simple compose code to execute both. Rather than soldering, jumper cables were used to connect the sensors and connection to the Node MCU esp8266. After the combined sensor code was uploaded to the Node MC esp8266 using the Arduino IDE, the mobile application was configured and deployed. Using the Firebase database, the connection between the Node MC esp8266 and mobile application was readily established. Once the data is stored in the Firebase real-time database, the machine will turn the water pump and fan on and off based on the temperature and humidity inside the mushroom house. The smart application will also display the status of all database output.

CHAPTER 5

CONCLUSION

5.1 Constraints

These constraints, including those related to hardware, resources, time, and technology, were encountered during the deployment of the system.

5.1.1 Hardware Constraints

Installation of the Automatic Mushroom House Monitoring System (MHMS) was extremely difficult due to the fragile hardware, some of which was broken during handling. Due to a voltage supply imbalance that occurred during programming, the Node MCU 8266 microcontroller was shorted out and could no longer be used to write programmes. The unstable internet connection is also responsible for the Node MCU 8266's malfunction and inability to transmit humidity and temperature readings to the real-time database. The DHT22 is also unable to obtain accurate temperature and humidity readings; consequently, the programme cannot be executed accurately. Since the voltage supply was less than what was required by the fan and water pump, the circuit required voltage directly from the power supply. Each requires its own power source for optimal operation. Even if there were hardware limitations, all hardware could be replaced, and the voltage supply could be rectified before the submission deadline for the project.

5.1.2 Time Constraints

Despite the fact that the system was completed before the deadline, there are numerous enhancements that could have been made. Since the mobile application was developed with the bare minimum of features, additional features may be introduced in the future; these are discussed in the sections that follow. Since the time constraint was applied to the system's execution for enhancement, the enhancement was added to the discussion of the system's future work.

5.1.3 Cost Constraints

Students were unable to afford the necessary hardware for this system, and the need to make repeated purchases of components like microcontrollers and cables in order to maintain functionality resulted in a rise in the overall cost of finishing the system.

5.1.4 Technical Constraints

New talents in mobile applications and the Internet of Things were required to construct this system. It has become more difficult to conduct research, acquire the Arduino and Android Studio programming languages, and configure the Node MCU 8266 with sensors due to a lack of knowledge. Numerous failures occurred before the system's implementation in code was completed, and achieving this objective required extensive research and comprehension of the Automatic Mushroom House Monitoring System (MHMS) concept and implementation procedure.

5.2 Future Works

The implementation of the future work was essential for improving the system's robustness and functionality. A few functions were reduced in this system to increase efficiency. These features can be included in later development. In this forthcoming attempt, system improvement will take precedence. Although the fan and water pump can be turned on and off automatically based on the threshold values that have been specified, there are still a few improvements that might be made to the system's functionality in the future.

The smart application can be used to connect multiple systems to a single user so that they can all be seen concurrently by the smart application. This is possible since the smart application was created to be used by a single system at a time. This makes it simple for the user to organise diverse systems. The system's functionality could also be enhanced by adding additional users to a single system under the initial user, so that additional users could manage the system in case the initial user was unable to perform system maintenance.

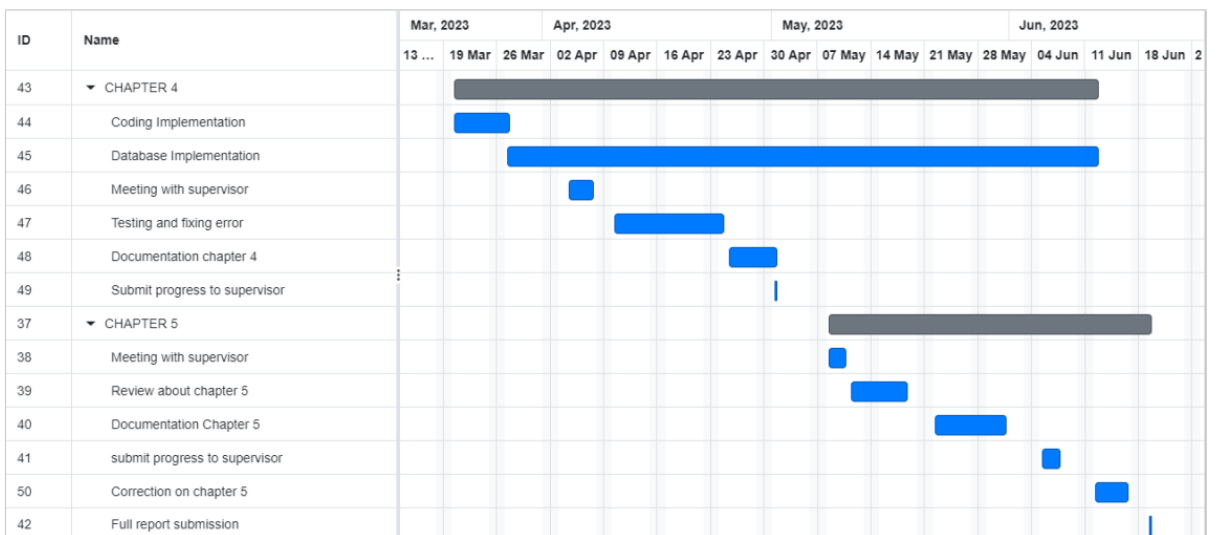
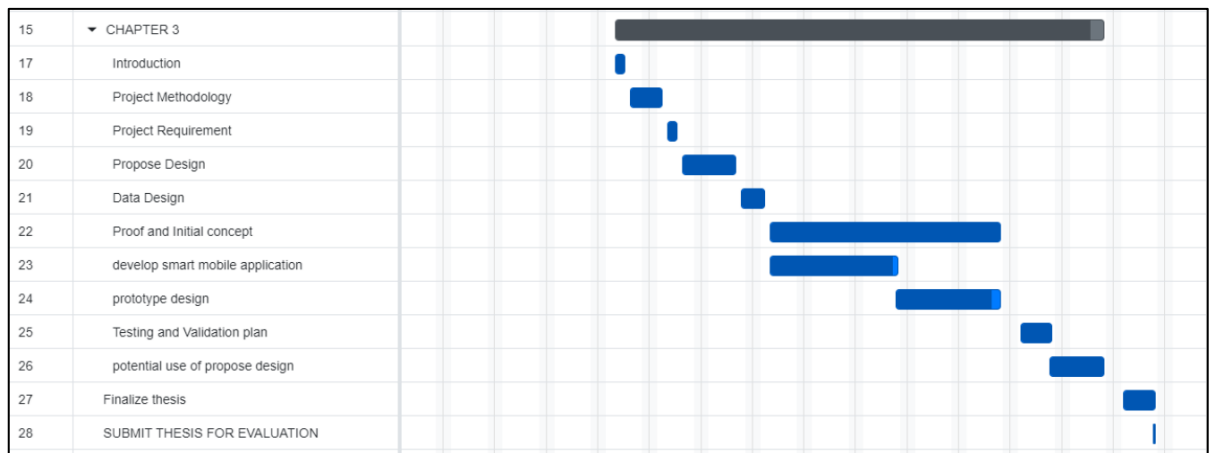
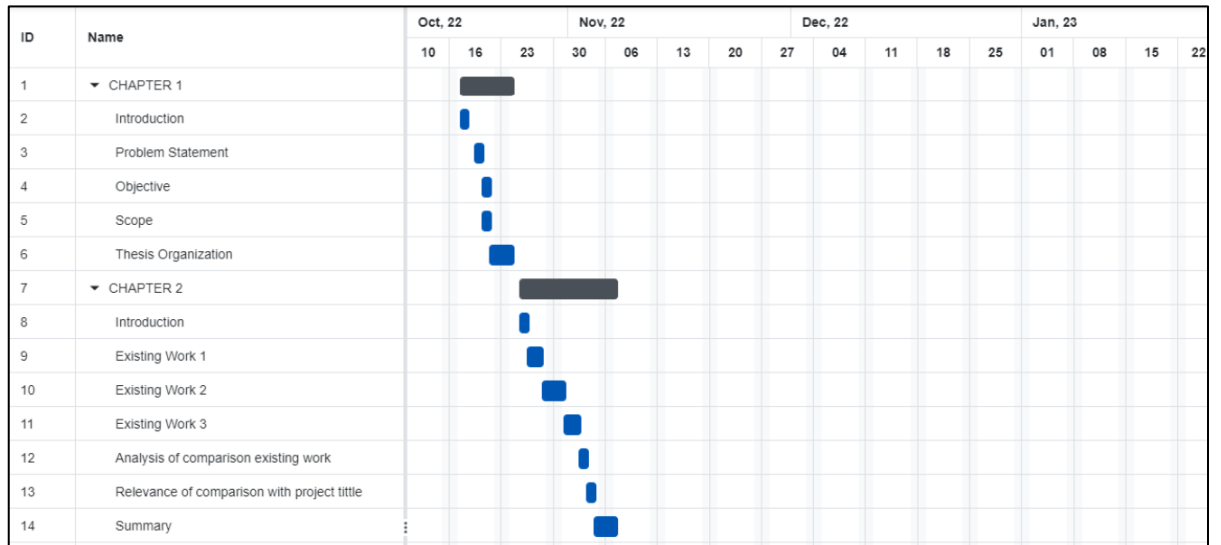
Additionally, a manual power on/off button can be added to the smart application to serve as a backup in the event that the water pump and fan cannot be turned on and off automatically.

Reference

- Agile Model (Software Engineering) - javatpoint.* (n.d.). Retrieved December 2, 2022, from <https://www.javatpoint.com/software-engineering-agile-model>
- Anwar, M. R., Liu, D. L., Macadam, I., & Kelly, G. (2013). Adapting agriculture to climate change: A review. *Theoretical and Applied Climatology*, 113(1–2), 225–245. <https://doi.org/10.1007/s00704-012-0780-1>
- Hafizuddin, M., Hishamuddin, M., & Ubin, A. (2022). Mushroom House Monitoring System Using Internet of Things (IoT). *Evolution in Electrical and Electronic Engineering*, 3(1), 545–553. <https://doi.org/10.30880/eeee.2022.03.01.063>
- Haimid, M. T., Rahim, H., & Dardak, R. A. (2013). Understanding the mushroom industry and its marketing strategies for fresh produce in Malaysia (Memahami industri cendawan dan strategi pemasaran untuk produk segar di Malaysia). In *Economic and Technology Management Review* (Vol. 8).
- Mohd Ariffin, M. A., Ramli, M. I., Mohd Amin, M. N., Ismail, M., Zainol, Z., Ahmad, N. D., & Jamil, N. (2020). Automatic climate control for mushroom cultivation using IoT approach. *2020 IEEE 10th International Conference on System Engineering and Technology, ICSET 2020 - Proceedings*, 123–128. <https://doi.org/10.1109/ICSET51301.2020.9265383>
- Nugroho, S., Hadi, S., & Hakim, L. (2017). Comparative Analysis of Software Development Methods between Parallel, V-Shaped and Iterative. *International Journal of Computer Applications*, 169(11), 7–11. <https://doi.org/10.5120/ijca2017914605>
- Obaideen, K., Yousef, B. A. A., AlMallahi, M. N., Tan, Y. C., Mahmoud, M., Jaber, H., & Ramadan, M. (2022). An overview of smart irrigation systems using IoT. *Energy Nexus*, 7, 100124. <https://doi.org/10.1016/j.nexus.2022.100124>
- SDLC - Agile Model.* (n.d.). Retrieved December 2, 2022, from https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm
- SDLC - Waterfall Model.* (n.d.). Retrieved December 2, 2022, from https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
- Sri, S., Ta'ahal, N., Masek, A., & Yusof, Y. (2022). *Green Technology Monitoring System for Oyster Mushrooms*. 2(2), 12–021. <https://doi.org/10.30880/ritvet.2022.02.02.002>

APPENDIX A

GANTT CHART



APPENDIX B

User Acceptance test (UAT)

the user acceptability test for the automatic mushroom house monitoring system (MHMS) using internet of things (IoT) is covered in this section. To make sure the system is adequate and meets the criteria, testing is done on each module. the system is thoroughly tested and qualified for system implementation, according to the permission of this testing. the flaws that were found are fixed.

User Acceptance Testing (UAT)			Verified by : <i>FARDAUSAUFLAN</i>		
No	Module	Activities	Pass	Fail	Comment
1	Login	user login	✓		
		user register	✓		
		Able to get in into real time data interface	✓		
2	real time data	value of temperature	✓		
		value of humidity	✓		
		notes of ideal humidy and temperature	✓		
3	hardware functioning	water pum	✓		
		fan	✓		
		water hose	✓		
4	navigation	login interface to register	✓		
		register to login	✓		
		login lto real time data	✓		
		real time data to report	✓		
		report to real time data	✓		
5	report	graph of humidity	✓		
		graph of temperature	✓		

	Name	Date
Verified by: <hr/> Developer	MUHAMMAD SYAFIK IKMAL BIN NAZARUDDIN	12/06/2023
Approved by: <hr/> Supervisor	Dr. Nor Syahidatul Nadiah Binti Ismail	