THE COPPERBELT UNIVERSITY
SCHOOL OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING P.O.
Box: 21692, Kitwe, Zambia.

DESIGN AND FABRICATION OF AN IOT MONITORING AND CONTROLLING SYSTEM FOR GREENHOUSE APPLICATION.

A PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF ENGINEERING DEGREE IN MECHATRONICS ENGINEERING

Submitted by

PRAISE MWANZA

Under the Guidance of

MR. CHIKOPA E. SOKOTELA

AUGUST 2024

THE COPPERBELT UNIVERSITY
SCHOOL OF ENGINEERING

P.O. Box: 21692, Kitwe, Zambia.

DEPARTMENT
OF
MECHANICAL ENGINEERING

APPROVAL

This is to certify that the project titled Design and Fabrication of an IOT Monitoring and Controlling System for Greenhouse Application was carried out by Praise Mwanza has been read and approved for meeting part of the requirements and regulations governing the MC 500 Mechatronics Individual Design Project course in Bachelor of Mechatronics Engineering at the Copperbelt University, Kitwe, Zambia during the academic year 2023-2024.

Name of Supervisor	Sign	Date
MR CHIKOPA E. SOKOTELA	…….	…….
Name of Coordinator	Sign	Date
MR E. BWEMBYA	…….	…….

DECLARATION

I hereby declare that I carried out the work reported in this report in the Department of Mechanical Engineering, The Copperbelt University, under the supervision of Mr. Sokotela. I solemnly declare that to the best of my knowledge no part of this report has been submitted in a previous application for the award of a bachelor's degree. All the sources of knowledge used have been duly acknowledged.

STUDENTS:

PRAISE MWANZA

Sign  Date 5th September 2024

DEDICATION

I dedicate this project to my mother for her unwavering support, my village people, including my grandmother, as I plan to build a greenhouse system on their farm, and the robotics club that provided me with machines and equipment. Ultimately, I dedicate this project to myself for the perseverance and hard work invested in this endeavour.

ACKNOWLEDGEMENT

First and foremost, I would like to thank Almighty God for His grace, which has guided me throughout my academic journey and made it possible to reach this milestone. I extend heartfelt thanks to my mother for her constant prayers and unwavering support, which have been a source of strength for me.

I also express my deep gratitude to my younger brother, Brian Mwanza, whose inquisitive nature and persistent questions played a vital role in shaping my research and enhancing my thought process.

My sincerest appreciation goes to my supervisor, Mr. C.E. Sokotela, for his invaluable guidance, mentorship, and dedication. His support was instrumental in the success of this research, and I am immensely grateful for his contribution.

Additionally, I would like to thank the lecturers from the Mechanical and Electrical Departments at Copperbelt University. Their knowledge and teachings were key to equipping me with the skills necessary for this project.

I am also indebted to my friends from Uniplexity AI and the Robotics Club, whose encouragement and camaraderie provided me with the motivation to push through challenges.

Lastly, I extend my gratitude to the Higher Education Loans and Scholarship Board (HELSB) for their financial support, which was instrumental in the successful completion of this work.

.

LIST OF TABLES

TABLE 1 DATA COLLECTED FROM SENSOR REAL TIME READINGS LII
TABLE 2 TABLE OF FORMULARS LXXII
TABLE 3 DATASHEET LXXV
TABLE 4 BUDGET LXXX

LIST OF FIGURES

FIGURE 1 TECHNOLOGIES, PROTOCOLS, AND APPLICATIONS OF INTERNET OF THINGS IN GREENHOUSE FARMING. XVI
FIGURE 2 IOT GENERAL ARCHITECTURE FOR GREENHOUSE APPLICATIONS. XVI
FIGURE 3 OVERALL SYSTEM ARCHITECTURE OF CLOUD-BASED IOT GREENHOUSE XVII
FIGURE 4 TEMPERATURE DOMAIN, IMAGE FROM XVIII
FIGURE 5 ANN MODEL XIX
FIGURE 6 MPC DIAGRAM XX
FIGURE 7 PID DIAGRAM. XXI
FIGURE 8 METHODOLOGY FLOW CHART. XXV
FIGURE 9 FLOW OF THE SYSTEM XXVII
FIGURE 10 DHT11 SENSOR. XXVIII
FIGURE 11 INTERNAL DHT PLATES. XXVIII
FIGURE 12 DHT CHARACTERISTICS CURVE XXIX
FIGURE 13 COMMUNICATION PROCESS. XXIX
FIGURE 14 SOIL MOISTURE SENSOR XXX
FIGURE 15 BH1750 SENSOR XXX
FIGURE 16 HUMIDIFIER FOR THE MINI GREENHOUSE XXXIV
FIGURE 17 ESP 32 XXXV
FIGURE 18 UNCOMPENSATED STEP RESPONSE, IMAGE FROM MATLAB. XXXVIII
FIGURE 19 ROOT LOCUS OF UNCOMPENSATED SYSTEM, IMAGE FROM MATLAB. XXXVIII
FIGURE 20 POLE-ZERO MAP OF UNCOMPENSATED SYSTEM, IMAGE FROM MATLAB. XXXIX
FIGURE 21 NYQUIST PLOT OF UNCOMPENSATED SYSTEM, IMAGE FROM MATLAB. XL
FIGURE 22 BODE PLOT OF UNCOMPENSATED SYSTEM, IMAGE FROM MATLAB. XL
FIGURE 23 COMPENSATED STEP RESPONSE, IMAGE FROM MATLAB. XLI
FIGURE 24 ROOT LOCUS OF COMPENSATED SYSTEM, IMAGE FROM MATLAB. XLII
FIGURE 25 POLE-ZERO MAP OF COMPENSATED SYSTEM, IMAGE FROM MATLAB. XLII
FIGURE 26 NYQUIST PLOT COMPENSATED SYTEM, IMAGE FROM MATLAB. XLIII
FIGURE 27 BODE PLOT OF COMPENSATED SYSTEM. XLIII
FIGURE 28 TRACKING RECTANGULAR TRAJECTORY, IMAGE FROM MATLAB. XLIV
FIGURE 29 CHAT BOT FLOW CHART. XLV
FIGURE 30 GREENHOUSE CAD MODEL, IMAGE FROM SOLIDWORKS. LI
FIGURE 31 MINI GREENHOUSE PROTOTYPE, CAPTURED IMAGE. LI
FIGURE 32 GREENHOUSE AUTOMATION SYSTEM, IMAGE FROM PROTEUS. LII
FIGURE 33 DEPICTION OF SIMULATED SENSOR DATA, IMAGE FROM MATLAB. LIII
FIGURE 34 SIMULATION OF A DHT, IMAGE FROM PROTEUS. LIV
FIGURE 35 ANALOGUE ANALYSIS OF THE DHT SENSOR, IMAGE FROM PROTEUS. LV
FIGURE 36 BEFORE CONTROL ACTUATOR WAS ACTIVATED, IMAGE FROM THE GREENHOUSE UI APP. LVII
FIGURE 37 TEMPERATURE GRAPH AFTER CONTROL ACTUATOR WAS ACTIVATED, IMAGE FROM THE GREENHOUSE UI APP. LVII
FIGURE 38 GRAPH SHOWING HOW PID AFFECTS FAN OUTPUT SPEED IMAGE FROM THE GREENHOUSE UI APP. LVIII
FIGURE 39 LIGHT INTENSITY OVER TIME BEFORE SYSTEM WAS ACTIVATED, IMAGE FROM THE UI APP. LIX

FIGURE 40 SOIL MOISTURE OVER TIME, IMAGE FROM THE UI APP. LX
FIGURE 41 HUMIDITY OVER TIME, IMAGE FROM THE UI APP. LXI
FIGURE 42 USE CASES OF PREDICTIVE ANALYTICS WITH ML, AI GENERATED IMAGE BY AI. LXIV
FIGURE 43 COMPUTER VISION APPLICATION IN AGRICULTURE, IMAGE BY ENCORD. LXV
FIGURE 44 SOIL MONITORING WITH IOT - SMART AGRICULTURE, IMAGE BY MANX TECHNOLOGY GROUP. LXV
FIGURE 45 EDGE AI-BASED ARCHITECTURE FOR EARLY HEALTH PREDICTION, IMAGE BY COLLEGE OF INFORMATION TECHNOLOGY. LXVI
FIGURE 46 CAD DRAWING FOR THE GREENHOUSE, IMAGE FROM SOLIDWORKS. LXX
FIGURE 47 USER INTERFACE FOR THE GREENHOUSE MONITORING SYSTEM, IMAGE FROM THE UI APP. LXX
FIGURE 48 AI CHATBOT LXXI

LIST OF ACRONYMS/ABBREVIATIONS

1. A2D - Analog to Digital Converter 35
2. AC - Alternating Current 10
3. ADC - Analog-to-Digital Converter 43
4. API - Application Programming Interface 74
5. BH1750 - Light intensity sensor model (BH1750) 23
6. BJT - Bipolar Junction Transistor 78
7. BLE - Bluetooth Low Energy 54
8. CAM - Computer-Aided Manufacturing 71
9. CAD - Computer-Aided Design 61
10. DAC - Digital-to-Analog Converter 45
11. DC - Direct Current 8
12. D2A - Digital to Analog Converter 18
13. DHT - Digital Humidity and Temperature sensor 5
14. DFT - Design for Testability 41
15. EEPROM - Electrically Erasable Programmable Read-Only Memory 51
16. ESP - Espressif Systems 63
17. FTDI - Future Technology Devices International 56
18. GH – Greenhouse 3
19. GPIO - General-Purpose Input/Output 91
20. GND – Ground 78
21. GUI - Graphical User Interface 47
22. HTTP - Hypertext Transfer Protocol 90
23. I2C - Inter-Integrated Circuit 47
24. IDE - Integrated Development Environment 63
25. IFTTT - If This Then That (automation platform) 63
26. IoT - Internet of Things 2
27. JSON - JavaScript Object Notation 87
28. LAN - Local Area Network 67
29. LCD - Liquid Crystal Display 12
30. LED - Light Emitting Diode 56
31. L298N - Motor Driver Module 52
32. MCU - Microcontroller Unit 4
33. MQTT - Message Queuing Telemetry Transport 78
34. MOSFET - Metal-Oxide-Semiconductor Field-Effect Transistor 61
35. NTP - Network Time Protocol 40
36. OTA - Over-the-Air (firmware updates) 72
37. PCB - Printed Circuit Board 91

38. PID - Proportional-Integral-Derivative (control system)……77
39. PMW - Power Management Unit……33
40. PSU - Power Supply Unit……..100
41. PWM - Pulse Width Modulation……...7
42. PWM - Pulse Width Modulation……86
43. RAM - Random Access Memory……84
44. RFID - Radio Frequency Identification……...39
45. RMS - Root Mean Square……87
46. ROM - Read-Only Memory……96
47. RTC - Real-Time Clock……...41
48. RTOS - Real-Time Operating System……88
49. SMD - Surface-Mount Device……81
50. SIM - Subscriber Identity Module……..30
51. SOIC - Small Outline Integrated Circuit……79
52. SPI - Serial Peripheral Interface……..35
53. SSR - Solid State Relay……...36
54. TFT - Thin-Film Transistor (display technology)……80
55. UART - Universal Asynchronous Receiver-Transmitter……19
56. UART - Universal Asynchronous Receiver-Transmitter……31
57. UPS - Uninterruptible Power Supply……..35
58. USB - Universal Serial Bus……73
59. VCC - Voltage at the Common Collector……11
60. VFD - Variable Frequency Drive……21
61. WAN - Wide Area Network……26
62. Wi-Fi - Wireless Fidelity……69
63. WLAN - Wireless Local Area Network……4
64. WSN - Wireless Sensor Network……55

Table of Contents

DECLARATION iii
DEDICATION iv
ACKNOWLEDGEMENT iv
LIST OF TABLES v
LIST OF FIGURES v
LIST OF ACRONYMS/ABBREVIATIONS vi
ABSTRACT xi
1. CHAPTER ONE xii
1.1 INTRODUCTION xii
1.2 BACKGROUND xii
1.3 MOTIVATION xiii
1.4 OBJECTIVES xiii
1.5 PROJECT SCOPE xiii
2. CHAPTER TWO xv
2.1 LITERATURE REVIEW xv
2.2 CASE STUDY OF MONITORING AND CONTROLLING TECHNOLOGIES xv
2.3.1 TECHNIQUES USED TO OPTIMIZE MONITORING AND CONTROL IN THE SMART GREENHOUSE xvii
2.3 GENERAL LITERATURE REVIEW OF AUTOMATED IOT GREENHOUSE SYSTEM xxii
3. CHAPTER 3 xxv
3.1 METHODOLOGY xxv
3.1.1 METHODOLOGICAL APPROACH xxv
3.1.2 SYSTEM REQUIREMENTS xxvi
3.1.3 TECHNOLOGY AND COMPONENT SELECTION REVIEW xxvii
3.2 SYSTEM DESIGN xxxv
3.2.1 SOFTWARE DESIGN xxxvi
3.2.1 HARDWARE xlv
4. CHAPTER 4 xlix
4.1 MANUFACTURE AND ASSEMBLY METHODS xlix
4.2 MANUFACTURING METHODS xlix

4.3 ASSEMBLY METHODS l
4.3.1 CAD MODEL l
4.3.2 PROTOTYPE li
4.4 CIRCUIT DESIGN lii
4.5 MODELLING AND SIMULATION lii
4.5.1 SENSOR SIMULATIONS lii
4.5.1 CIRCUIT SIMULATIONS liv
5. CHAPTER 5 lvi
5.1 SYSTEM TESTING lvi
5.2 CONDUCTED TESTS lvi
5.2.1 EXPERIMENT 1: TEMPERATURE REGULATION lvi
5.2.2 EXPERIMENT 2: LIGHT INTENSITY ADJUSTMENT lix
5.2.3 EXPERIMENT 3: SOIL MOISTURE REGULATION lix
5.2.4 EXPERIMENT 4: HUMIDITY REGULATION lxi
6. CHAPTER 6 lxii
6.1 CONCLUSIONS lxii
6.2 RECOMMENDATIONS lxii
6.2.1 OPTIMAL PLACEMENT OF SENSORS FOR ACCURATE ENVIRONMENTAL MONITORING lxii
6.2.2 REDUNDANCY AND BACKUP FOR CRITICAL SENSORS lxiii
6.2.3 ENHANCED DATA PROCESSING AND INTEGRATION lxiii
6.2.4 INCREASED USE OF AUTOMATION FOR IMPROVED EFFICIENCY lxiii
6.3 PROPOSED FUTURE WORKS lxiii
6.3.1 INTEGRATION OF AI FOR PREDICTIVE ANALYTICS. lxiv
6.3.2 UTILIZATION OF COMPUTER VISION FOR CROP HEALTH MONITORING lxiv
6.3.3 INCORPORATING PH MONITORING FOR SOIL AND WATER QUALITY MANAGEMENT lxv
6.3.4 DEVELOPMENT OF AN AI-POWERED DISEASE DETECTION SYSTEM lxvi
References lxvii
APPENDICES lxx
APPENDIX 1: MODEL MODELLINIG lxx

1. GREENHOUSE MODELLING lxx
2. USER INTERFACE DESIGN lxx
3. AI CHAT BOT lxxi
APPENDIX 2: TABLE OF FORMULARS lxxii
APPENDIX 3: COMPONENTS DATASHEET lxxv
APPENDIX 4: BUDGET lxxx
APPENDIX 5: CODE USED lxxxiii
1 PID SYSTEM DESIGN IN MATLAB (Matlab code) lxxxiii
2. SENSOR SIGNAL SIMULATION MATLAB CODE (Matlab code) lxxxv
3. MONITORING AND CONTROLLING FOR THE ESP32(Code in C) lxxxvii
4. AI CHATBOT AND USER INTERFACE (Python code) xciv

ABSTRACT

In conventional farming, farmers must frequently visit the farmland to estimate various environmental parameters like temperature, humidity, light intensity, and soil moisture to ensure timely harvests in optimal soil conditions. Although this traditional technique has been in use for many years, it is inconsistent and often leads to suboptimal productivity due to the imprecise assessment of environmental parameters. Greenhouse farming, however, allows crops to be grown in controlled ecosystems where environmental factors are adjusted to suit specific crop types, including those that cannot thrive in harsh conditions. This work aims to develop an automated IoT greenhouse monitoring and control system that integrates multiple sensors, such as a temperature sensor, humidity sensor, light intensity sensor and soil moisture sensor, to capture essential environmental parameters. The system utilizes the ESP32 development board to store and process data while providing Wi-Fi functionality. The light intensity, temperature and humidity sensor, and soil moisture sensor control the lighting, fan and heater activation, and pump triggering, respectively, whenever environmental parameters fall below threshold values. Additionally, the system employs proportional integral derivative control for precise regulation of these parameters. Leveraging the ESP32's WiFi capability, the Internet of Things (IoT) is used to process data, and deliver information to a user's web application for monitoring and controlling greenhouse conditions. Furthermore, critical values trigger notifications sent to users' mobile phones for offline updates, and an AI Chabot is included in the system to assist farmers with real-time guidance and support. This system enables the farming of crops such as vanilla, saffron, and orchids, which cannot be grown in harsh environments like those found in Zambia, by creating optimal growing conditions within the automated greenhouse.

Keywords: IoT, Greenhouse, Web Application, Control System, Smart Agriculture, PID Control, Mobile Notifications, AI Chabot.

1. CHAPTER ONE

1.1 INTRODUCTION

Agriculture is one of the most critical sectors, providing a living for a great number of people and contributing to climate change [1]. It is a primary field that sustains a country's economy and has been the lifeblood of all living things because it is the most important source of food and raw materials [2]. Agriculture constituted 17.5% of Zambia's overall Gross Domestic Product (GDP) in 2020[3]. Over 70% of Zambians work in agriculture, primarily at a subsistence level. Despite the significance of copper exports, the agricultural sector remains the foundation of the Zambian economy since it is the primary source of subsistence for most Zambians.

However, Zambia's agricultural industry faces various obstacles that hamper its productivity, including poor land tenure, insufficient irrigated farming, climate change, and land degradation [4]. Other challenges include low technology, high production costs, inadequate input distribution, restricted finance, substantial post-harvest losses, and limited market access [5]. These issues significantly affect the efficiency and productivity of the sector [6]. The focus of this work will be on the challenges experienced due to climate change and its irregularities.

To address these challenges, greenhouses offer a viable solution by providing the necessary conditions for plants to flourish through the monitoring and control of environmental conditions within the structure [7]. Monitoring involves the collection of data, which has evolved with the advancement of technology into a system that collects data and generates actionable information [7]. This monitoring process is carried out using sensors, which measure changes in the environment and convert them into electrical signals that can be read and analysed.

1.2 BACKGROUND

Technological advancements have created means by which the monitoring of conditions can be done more effectively and remotely, leveraging the Internet of Things (IoT)[8]. The IoT has been incorporated in many sectors, such as healthcare, home automation, smart cities, and industry, and it is appropriate for integration into the agricultural sector, which is the backbone of many economies, including Zambia's [9]. The IoT is a collection of interconnected devices that exchange data without the need for human-machine interaction. These devices, integrated with electronics, sensors, and network connectivity, gather and exchange data to create a highly efficient monitoring and control system [10].

Farming on any scale is challenging, as different plants and crops have varying requirements for soil water content, temperature, fertilizers, light intensity, etc. [11]Farmers often struggle to provide the required quantities adequately, either underproviding or overproviding. An automated greenhouse using IoT technology can make farming less strenuous by providing better control of environmental parameter fluctuations, leading to increased crop yield [12].

This project's significance includes eliminating time spent on monotonous tasks, channelling time to more critical tasks, improving crop yield by ensuring plants are available in and out of season, and managing resources more effectively and efficiently [13]. It also enhances the research process in agriculture by making data provision readily available on the cloud [14]. Given the changing climatic conditions and the limitations of traditional greenhouse monitoring systems that require significant human intervention, there is a need for an automated greenhouse [15]. This project aims to address these needs by designing a greenhouse monitoring and control system that leverages the capabilities of the microcontroller platform [16].

The proposed system offers several advantages over traditional manual greenhouse management. It provides real-time monitoring capabilities, allowing farmers to remotely access and monitor the greenhouse's environmental data through a user-friendly interface [17]. This system enhances resource management, reduces labour requirements, and improves overall crop yield and quality [18]. The automation capabilities allow for timely interventions, ensuring optimal growing conditions and increasing the efficiency of the farming process [19].

1.3 MOTIVATION

1.4

Traditional farming in Zambia is labour-intensive, time-consuming, and often imprecise, leading to suboptimal growing conditions and lower crop yields. Even with greenhouse farming, manual interventions are still common, resulting in inefficiencies and potential crop losses. High-value crops like vanilla, saffron, and orchids, which require specific conditions, cannot be grown effectively with current practices.

The primary problem is the inefficiency and inaccuracy of traditional greenhouse management. There is a need for an automated system that can monitor and control environmental parameters, provide real-time data, and allow remote management. Leveraging IoT and microcontroller technology like the ESP32, this project aims to develop a solution to improve the efficiency, productivity, and profitability of greenhouse farming in Zambia.

Using IoT technology, this project aims to replicate the native tropical environmental conditions required for the cultivation of high-value foreign crops such as vanilla, orchids, and saffron. By creating an automated greenhouse system that can precisely monitor and control temperature, humidity, light, and soil moisture, the project will enable the local production of these crops in Zambia.

This technological advancement will reduce the reliance on imports, thereby improving the economic stability and sustainability of the agricultural sector. Furthermore, it will allow farmers to diversify their crop production, increase their income, and contribute to the overall food security and economic growth of the region. The ability to produce such crops locally will also foster innovation and technological adoption in agriculture, promoting more efficient and sustainable farming practices.

1.4 OBJECTIVES

1. To design a robust architecture for the IoT-based greenhouse monitoring and controlling system.
2. To select and integrate appropriate sensors and actuators to measure environmental parameters and control greenhouse components.
3. To design a user-friendly user interface for remote monitoring, control, and management of the greenhouse system.

1.5 PROJECT SCOPE

i. System Design:

   Develop the architecture, components, and communication protocols for the IoT monitoring and controlling system.

ii. Hardware Requirements:

   Procure and integrate sensors, actuators, microcontrollers, and communication modules.

iii. Software Requirements:

Integrate control algorithms, IoT gateway software, and develop a user-friendly interface.

iv. Integration and Testing:

Assemble and test the prototype in a mini greenhouse.

2. CHAPTER TWO

2.1 LITERATURE REVIEW

The literature review aims to provide a comprehensive understanding of existing research on IoT-based systems for greenhouse management [20]. This section introduces the scope of the review, highlighting the importance of IoT technologies in enhancing greenhouse operations [21]. It sets the stage for discussing historical developments, technological advancements, and current trends.

2.2 CASE STUDY OF MONITORING AND CONTROLLING TECHNOLOGIES.

Greenhouse monitoring and control technologies are crucial in modern agriculture for ensuring optimal growing conditions and enhancing productivity [22]. This section reviews various studies that highlight the integration of IoT and other technologies in greenhouse management, focusing on the frameworks, control strategies, and algorithms used [23].

2.2.1 IOT-BASED SMART GREENHOUSE FRAMEWORKS

1. IoT Based Smart Greenhouse Framework and Control Strategies for Sustainable Agriculture.

   This study presents an IoT network framework designed to enhance sustainability in greenhouse environments through efficient resource allocation. It explores various IoT-driven applications, involving diverse sensors and communication protocols, and evaluates greenhouse sensor technologies and standards [24]. The research addresses challenges in smart greenhouse farming and security concerns, suggesting potential future work. Technologies, network architecture, topology, and platforms underlying IoT-enabled greenhouses are detailed, developing taxonomies classifying IoT-leveraged greenhouse management solutions and potential cyber threats [25].

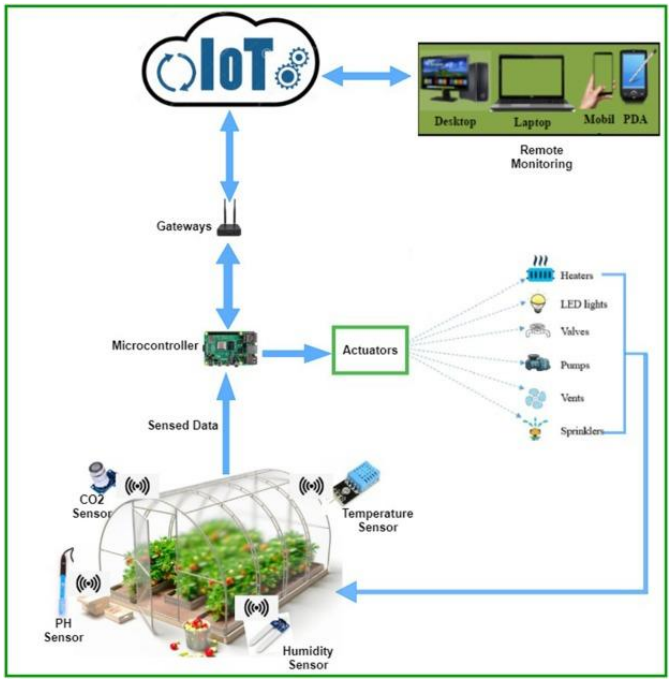

Figure 1 Technologies, Protocols, and applications of Internet of Things in greenhouse Farming [16].

2. Internet of Things Approaches for Monitoring and Control of Smart Greenhouses in Industry 4.0.

This paper discusses a system that merges power electronics and power systems using GSM techniques, Arduino microcontrollers, and various sensors to capture real-time data within greenhouse environments [26]. The system processes live data on crop conditions and transmits it to users' mobile devices, allowing direct access to monitor and regulate temperature, watering schedules, and lighting conditions [27]. This system offers cost-effective and efficient monitoring and control, minimizing the need for constant human intervention [28].

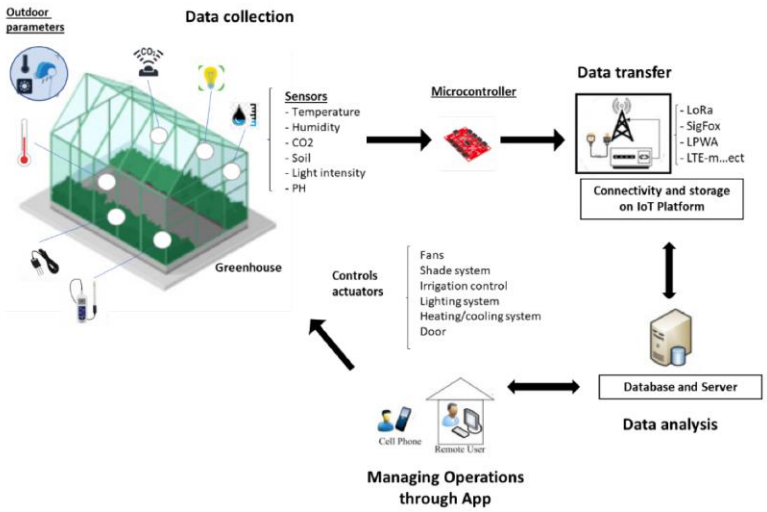

Figure 2 IOT General Architecture for greenhouse applications.

3. A Cloud-Based IoT Platform for Precision Control of Soilless Greenhouse Cultivation

This study focuses on designing an intelligent IoT-based system for controlling and monitoring greenhouse temperature, highlighting its importance in precision farming and sustainability [29].

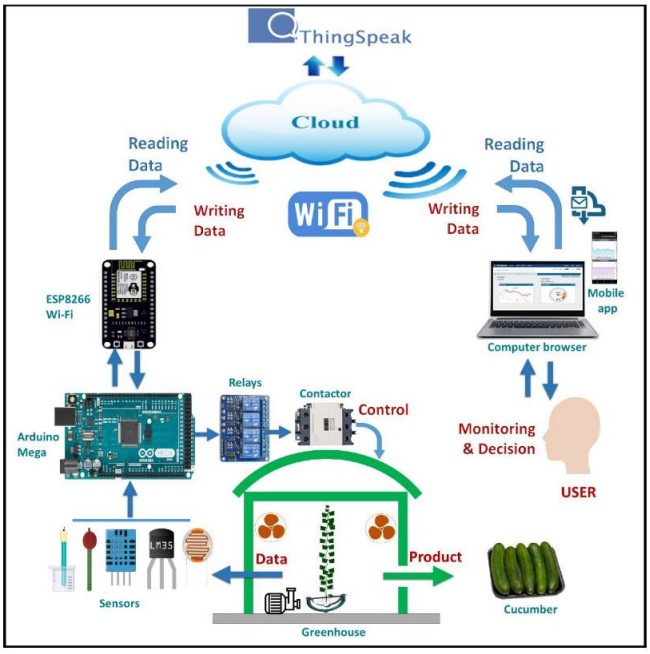

Figure 3 Overall system architecture of cloud-based IoT greenhouse

4. A New IoT-Based Platform for Greenhouse Crop Production.

   The study introduces a new IoT-based platform aimed at enhancing greenhouse crop production, providing insights into its implementation and benefits for modern agriculture [30].

Design and Deployment

1. Design and Deployment of a Practical IoT-Based Monitoring System for Protected Cultivations

   This study discusses the design and deployment of a practical IoT-based monitoring system, focusing on its application in protected cultivations and its effectiveness in enhancing agricultural practices [31].

2. Smart Sustainable Greenhouses Utilizing Microcontroller and IoT in the GCC Countries This paper explores smart sustainable greenhouses using microcontrollers and IoT, specifically in the GCC countries, including energy requirements and economic analysis for a concept model in Qatar [32].

3. An IoT-Based Hierarchical Control Method for Greenhouse Seedling Production

   The study proposes a hierarchical control method for greenhouse seedling production using IoT, emphasizing the method's effectiveness in improving seedling quality and production efficiency [33].

2.3.1 TECHNIQUES USED TO OPTIMIZE MONITORING AND CONTROL IN THE SMART GREENHOUSE.

Several techniques can be used to optimize control and monitoring. In this first part, the main challenging methods are introduced: fuzzy logic, ANN, MPC, and PID. In the second part of this section, a briefly discussion is given other recent control and monitoring methods applied to smart greenhouses [34].

Fuzzy Logic

The fuzzy control system is a mathematical system that analyses analog input values in terms of logical variables. Fuzzy Logic Control (FLC) techniques decompose a complex system into several subsystems based on human experts' knowledge about the system. Many processes controlled by human operators cannot be automated using conventional control techniques because their performance is often lower than that of the operators [35]. One reason for this is that linear controllers, which are commonly used in control systems, are not suitable for nonlinear plants [36]. Fuzzy sets are used to define the meaning of qualitative values of the controller inputs, and fuzzy logic can capture the continuous nature of human decision processes and is therefore an improvement over methods based on binary logic [37]. The mathematical approach related to this concept can be written as:

$$P_i : \text{if } x_1 \text{ is } C_{i1} \ldots \text{and } x_N \text{ is } C_{iN} \text{ then } u \text{ is } D_i \quad \forall i = 1, 2, \ldots, K \quad \text{.....1}$$

The fuzzy sets are determined based on deep human expertise and knowledge of the thermal behaviour of the greenhouse system (GHS) as well as of the usual operating ranges of the GHS [38].

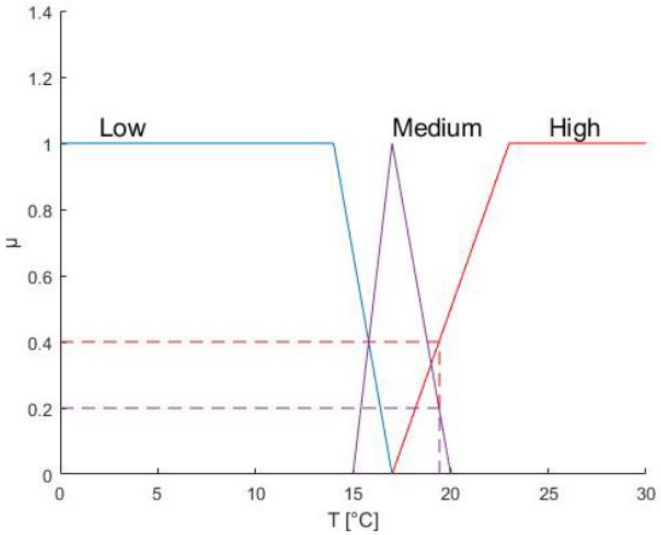

Figure 4 Temperature domain, image from

Temperature domain divided into three fuzzy sets. The dash lines help to take in information about the μ value in the three fuzzy sets at a specific temperature.

Artificial Neural Networks (ANN)

ANNs consist of many simple processors linked by weighted connections that can acquire knowledge from the environment through a learning process and store the knowledge in its connections [40].

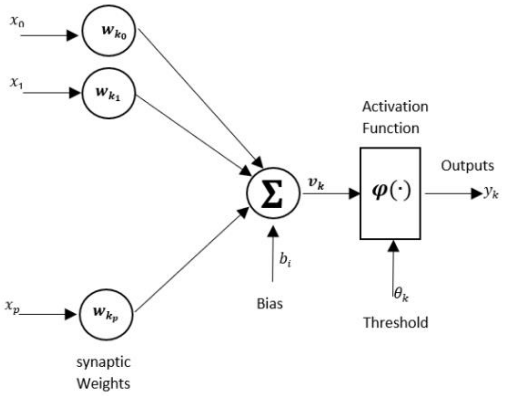

Figure 5 ANN model

Mathematical model of ANN.

A functional model of neurons must consider three basic components:

1. Synapses of the neuron are modelled as weights whose values represent the strength of the connection. Positive weight values represent excitatory connections, and negative values represent inhibitory connections [41].
2. A summing function sums all inputs modified by their weights.
3. An activation function controls the amplitude of the output of the neuron.

The mathematical formulation according to the model in the previous figure is:

$$y_k = \varphi(v_k) = \varphi\left(\sum_{i=1}^{p} w_{ki}x_i + b_i\right) \quad \text{.....2}$$

where b_i is the bias, y_k is the output, w_{ki} is the weight between neuron k of the previous layer and neuron i of the current layer, and ϕ is the activation function, which is a differentiable and nonlinear function.

ANNs have been implemented in various greenhouse applications, from microclimate forecasting to energy consumption-specific tasks, including CO2 control [42].

Model Predictive Control (MPC)

The MPC approach is outlined in Figure below.

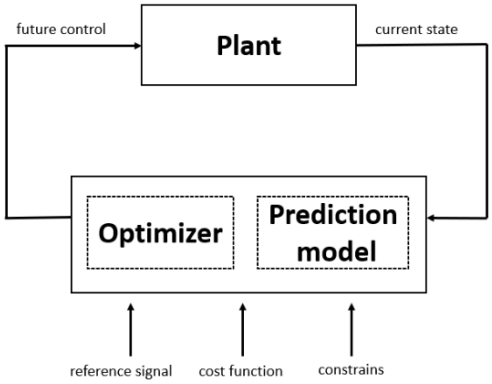

Figure 6 MPC diagram

MPC schema

The plant represents the true system—in our case, the greenhouse. The prediction model is usually a simplified model, generally linear, describing the dynamics of the state variables of the plant. Such a prediction model usually gives good performance in the first instances of simulation, while it becomes completely unreliable after a while [43]. The optimizer is usually defined as a cost function to be minimized and is often subject to constraints. Mathematical programming, ANN, or other solving techniques can be used here. However, the solution should be computed in a relatively fast time compared to the system dynamics, which are usually less than the duration of a sample time in the case of discrete-time systems. MPC implements a rolling horizon approach to control the system. At each step, only the first value of the optimal control sequence is applied, as it is continuously recomputed taking into account new measures of the output of the system [44]. The following formula shows a typical mathematical formulation with quadratic cost for a constrained optimal control:

$$\min_u \quad x'Qx + u'Ru \qquad \text{.....3}$$

$$\text{subject to} \quad u_{\min} \le u_k \le u_{\max} \quad \forall k = 0, 1, \ldots, N-1 \qquad \text{.....4}$$

$$x_{\min} \le x_k \le x_{\max} \quad \forall k = 1, \ldots, N \qquad \text{.....5}$$

where

- x is the prediction state variable;
- u is the prediction control variable;
- Q is the semi-definite positive symmetric matrix;
- R is the definite positive symmetric matrix.

PID Control Algorithms

PID control algorithms are widely used in greenhouse systems to maintain desired environmental conditions by adjusting heating, cooling, ventilation, and irrigation systems in response to sensor data [45]. A PID controller applies a feedback correction that continuously calculates an error value as the difference between a reference value r(t) and the measured variable y(t). The control function in a continuous time system can be written as:

$$u(t) = K_p e(t) + K_i \int_0^T e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad \text{.....6}$$

where $e(t) = y(t) - r(t)$, and K_p, K_i, and K_d are respectively the proportional, integrative, and derivative gains. The control parameters must be adjusted to improve system performance. Stability is a base requirement, but different values of the gains may lead to different settling times, overshooting, etc.

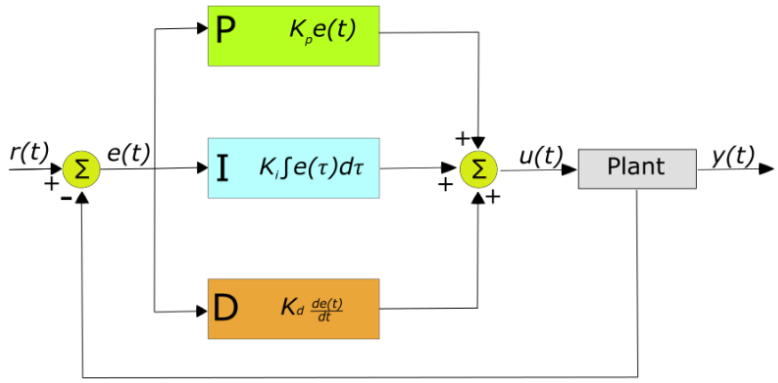

Figure 7 PID diagram.

PID controller design

Other Relevant Methodological and Technological Aspects

Robust Control

Robust control is an approach to controller design related to uncertainty [46]. The goal of robust methods is to ensure stability in the presence of limited modelling errors [47]. The system is often optimized and controlled to minimize the maximum worst performance to guarantee a certain quality of service. Some real applications use this approach [48].

Distributed Control

Distributed control is an automatic control methodology where the controllers are in different systems and a central supervisory is not present. Due to the easy scaling architecture, it adds or removes subsystems and sensors. This approach can be used for example among different neighbouring greenhouses wishing to cooperate, sharing resources like water and energy, or even at the level of a single greenhouse [49].

Image Analysis

Image analysis is commonly used to monitor crops in horticulture. Due to the difficulty in monitoring the growth of tomato cultivation in plant clusters, an automated clip-type IoT has been designed to address this issue.

Kalman Filtering

Filtering can be a classic way to cope with noise in sensor data. A filtering model based on the ANN algorithm to update the error covariance in the Kalman filter has been proposed. The experimental results prove that the proposed method improves performance compared to traditional techniques [50].

Finite Difference Method

The finite difference method is used for managing heat transfer in greenhouse walls. This method was built to evaluate the periodic variation of environmental conditions, with a forward and backward approach to simulate the system efficiently.

2.3 GENERAL LITERATURE REVIEW OF AUTOMATED IOT GREENHOUSE SYSTEM

IoT-based greenhouse monitoring and controlling system comprises several interconnected components working together to ensure optimal environmental conditions for plant growth [31]:

Literature Survey

A. Sensors

1 Soil Moisture Sensor: Measures the moisture content of the soil, providing crucial data for irrigation management.

2 DHT11 Sensor: Monitors temperature and humidity levels within the greenhouse, ensuring a conducive climate for plant growth [23].

3 Piezoelectric Sensors: Convert mechanical vibrations caused by wind, plant movement, or human activity into electrical energy for renewable power generation.

4 Flame IR Sensor: Detects potential fire hazards, safeguarding the greenhouse from risks.

B. Microcontroller and Communication Module

1 Arduino Uno: Acts as the central processing unit, collecting data from sensors, executing control algorithms, and managing the overall system operation.

2 ESP-01 Wi-Fi Module: Facilitates wireless communication, enabling data transmission to remote servers for monitoring and control purposes [34].

C. Actuators

1 Buzzer: Provides audible alerts to greenhouse operators in case of critical events, such as fire detection.

2 Cooling Fan: Regulates temperature inside the greenhouse by dissipating excess heat, maintaining optimal growing conditions for plants.

3 Pumps: Manage irrigation by controlling water flow to ensure plants receive the right amount of water.

4 Humidifiers: Increase humidity levels within the greenhouse to maintain an ideal environment for plant growth [11].

5 Heaters: Provide warmth to maintain optimal temperature conditions during cooler periods.

6 Lights: Supplement natural light to promote photosynthesis and support plant growth during periods of low natural light [15].

7 LCD Display: Offers real-time visualization of environmental parameters and system status, providing essential feedback to operators.

D. Power Supply

1 Rechargeable Battery: Stores electrical energy generated from the solar panel and piezoelectric sensors, ensuring uninterrupted operation of the monitoring and control system, even during periods of low solar irradiance.

2 Solar Panel: Converts solar energy into electrical power, serving as the primary source of renewable energy for charging the rechargeable battery and powering the system components.

System Operation

The operation of the IoT-based greenhouse monitoring and controlling system is a multi-step process designed to ensure precise management of environmental parameters for optimal plant growth:

A. Data Acquisition

The system continuously collects data from various sensors deployed throughout the greenhouse. The soil moisture sensor measures soil moisture content, providing critical information for irrigation management. Simultaneously, the DHT11 sensor monitors temperature and humidity levels within the greenhouse to maintain a conducive climate for plant growth. Additionally, the flame IR sensor detects potential fire hazards, while the piezoelectric sensors capture mechanical vibrations, converting them into electrical energy for renewable power generation [14].

B. Control Logic

Upon receiving sensor data, the Arduino Uno microcontroller executes sophisticated control algorithms to regulate environmental parameters within the greenhouse. Based on the collected data, the control logic activates actuators, such as the cooling fan, to manage temperature levels, ensuring they remain within predefined thresholds conducive to plant growth. Moreover, the system adjusts irrigation schedules based on soil moisture levels, ensuring plants receive adequate water without overwatering or under watering. In the event of fire detection by the flame IR sensor, the system triggers alarms through the buzzer to alert greenhouse operators promptly [37].

C. Data Transmission

The processed data is transmitted to a designated cloud platform or server via the ESP-01 WiFi module. This wireless communication enables real-time monitoring and analysis of greenhouse conditions remotely. By leveraging cloud-based IoT platforms, users can access the collected data from anywhere, at any time, using web-based interfaces or mobile applications.

V. Renewable Energy Integration

Renewable energy integration within the IoT-based greenhouse monitoring and controlling system plays a pivotal role in enhancing sustainability, reducing operational costs, and mitigating environmental impact:

A. Solar Panel Integration

Solar panels are strategically mounted on the greenhouse roof or adjacent structures to maximize exposure to sunlight. These photovoltaic modules convert solar radiation into electrical energy through the photovoltaic effect. The generated electricity powers various components of the monitoring and control

system, including sensors, microcontrollers, communication modules, actuators, and display interfaces. By harnessing solar energy, the system reduces dependency on conventional grid electricity, lowering operational costs and carbon emissions [13].

B. Piezoelectric Sensor Utilization

Piezoelectric sensors are strategically placed within the greenhouse environment to capture mechanical vibrations induced by natural phenomena such as wind, plant movement, or human activity. These sensors convert mechanical energy into electrical energy, which is stored in the system's rechargeable battery [51]. The harvested energy from piezoelectric sensors serves as an additional renewable energy source, supplementing the power generated by solar panels. This dual-source approach ensures continuous and reliable power supply, particularly during periods of low solar irradiance or insufficient sunlight .

Remote Monitoring and Control

The remote monitoring and control aspect of the IoT-based greenhouse system provides users with real-time access to critical environmental parameters, enabling informed decision-making and proactive intervention when necessary:

A. ThingSpeak Integration

The system utilizes ThingSpeak as the cloud-based IoT platform for data storage, visualization, and remote access. ThingSpeak offers a user-friendly interface that allows greenhouse operators to monitor various parameters, such as temperature, humidity, soil moisture, and fire hazards, from any internet-enabled device. By accessing the ThingSpeak channel associated with the greenhouse system, users can visualize sensor data in the form of interactive graphs, charts, and widgets, providing insights into current conditions and historical trends [52].

B. Control Mechanism

The implementation of temperature threshold control involves defining acceptable temperature ranges within the greenhouse. When temperature readings surpass the predefined upper threshold, indicating overheating, the system automatically triggers a control action to switch on the cooling fan. This proactive approach helps maintain optimal temperature levels within the greenhouse, mitigating the risk of heat stress and promoting plant growth [53].

3. CHAPTER THREE

3.1 METHODOLOGY

3.1.1 METHODOLOGICAL APPROACH

The initial phase of my approach involved a thorough literature review. This step is crucial for understanding the existing knowledge, identifying gaps, and gathering insights from previous studies related to my project. This foundational step ensures that my project is grounded in well-established research and leverages existing advancements.

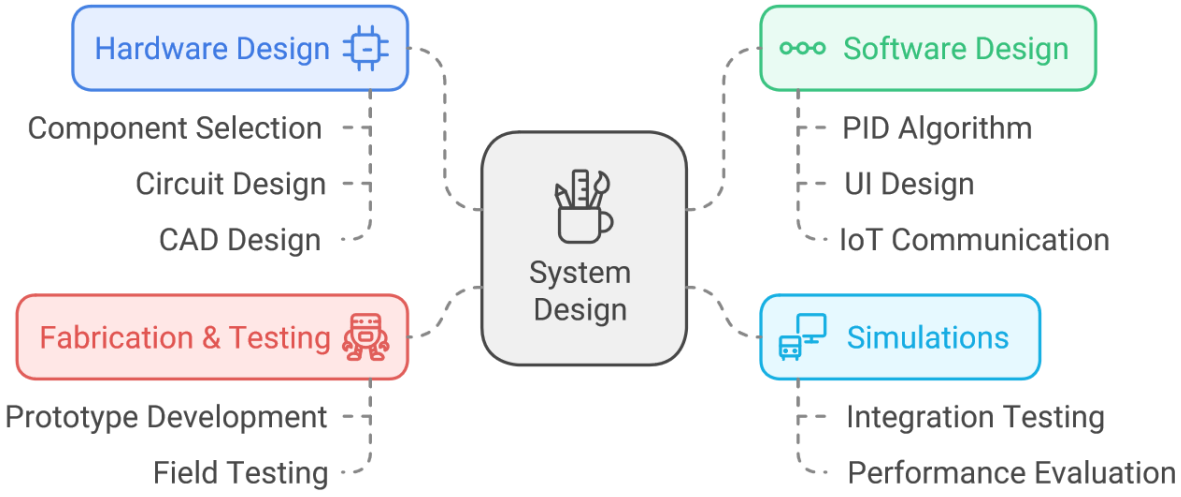

Figure 8 Methodology flow chart.

System Design

Following the literature review, a transition into the system design phase, which is divided into two main components: hardware design and software design.

(A) Hardware Design

1. Component Selection: This sub-phase involves selecting the appropriate components required for my system. It is essential to choose components that meet the specifications and requirements derived from the literature review and project objectives.
2. Circuit Design: In this step, I designed the circuit that integrates the selected components. This involves creating schematics and ensuring that the connections are correct and efficient.
3. CAD Design: Unit Box & Mini Greenhouse: This step involves using computer-aided design (CAD) tools to create the physical layout of the unit box and mini greenhouse. The design must accommodate all the hardware components and facilitate easy assembly and maintenance.

(B) Software Design

1. PID Algorithm: I design and implement a Proportional-Integral-Derivative (PID) algorithm for controlling the system. This algorithm helps in maintaining the desired set points by minimizing the error between the set point and the measured value.
2. UI Design: This step involves designing the user interface for the system. The UI should be user-friendly and allow easy interaction with the system, including monitoring and control functionalities.
3. IoT Communication: This involves designing the communication protocols and mechanisms to enable the system to connect to the Internet of Things (IoT). This allows for remote monitoring and control, as well as data logging and analysis.

Simulations

After completing the hardware and software design, I move on to the simulation phase. This step is crucial for testing the system in a virtual environment to identify and address any issues before physical implementation. Simulations help in verifying the functionality and performance of both the hardware and software components.

Fabrication & Testing

The final phase is the fabrication and testing of the system. This involves building the physical prototype based on the designs and testing it to ensure that it meets the project requirements and performs as expected. Any issues identified during testing are addressed through iterative improvements.

Iterative Process

My methodological approach was iterative, with feedback loops between the simulations and the design phases. This iterative process allows for continuous refinement and improvement of the system, ensuring a robust and reliable final product.

3.1.2 SYSTEM REQUIREMENTS

The problem I am trying to solve is the inefficiency and inaccuracy of traditional greenhouse management in Zambia. This project, titled "Design and Fabrication of an IoT Monitoring and Controlling System for Greenhouse Application," aims to develop a robust IoT-based system for monitoring and controlling environmental parameters in greenhouses to optimize growing conditions. By leveraging sensors, actuators, PID control, and a user-friendly interface for remote management, this system will enhance the efficiency, productivity, and profitability of greenhouse farming.

i. Robust System Architecture
    - Design a scalable, modular architecture.
    - Ensure reliable operation in greenhouse conditions.
ii. Sensor Integration
    - Select and integrate sensors for temperature, humidity, soil moisture, and light intensity.
    - Use high-accuracy sensors (e.g., DHT22, BH1750).
iii. Actuator Control
    - Integrate actuators for heaters, fans, humidifiers, lights, and irrigation pumps.
    - Use relays or motor drivers (e.g., L298N for pump control).
iv. PID Control Implementation
    - Implement PID control to maintain optimal environmental conditions.
v. User Interface Design
    - Develop a user-friendly web application.

- Display real-time data.

vi. Data Logging and Real-Time Reporting
- Implement data logging and real-time reporting of environmental parameters.

vii. Remote Management and IoT Communication
- Enable remote access via Wi-Fi (ESP32) and GSM (SIM800L).
- Send notifications for out-of-range conditions and actuator status changes.

viii. Power Management and Stabilization
- Use capacitors (e.g., 0.1µF) and pull-up resistors (e.g., 10kΩ) for power stability and signal quality.

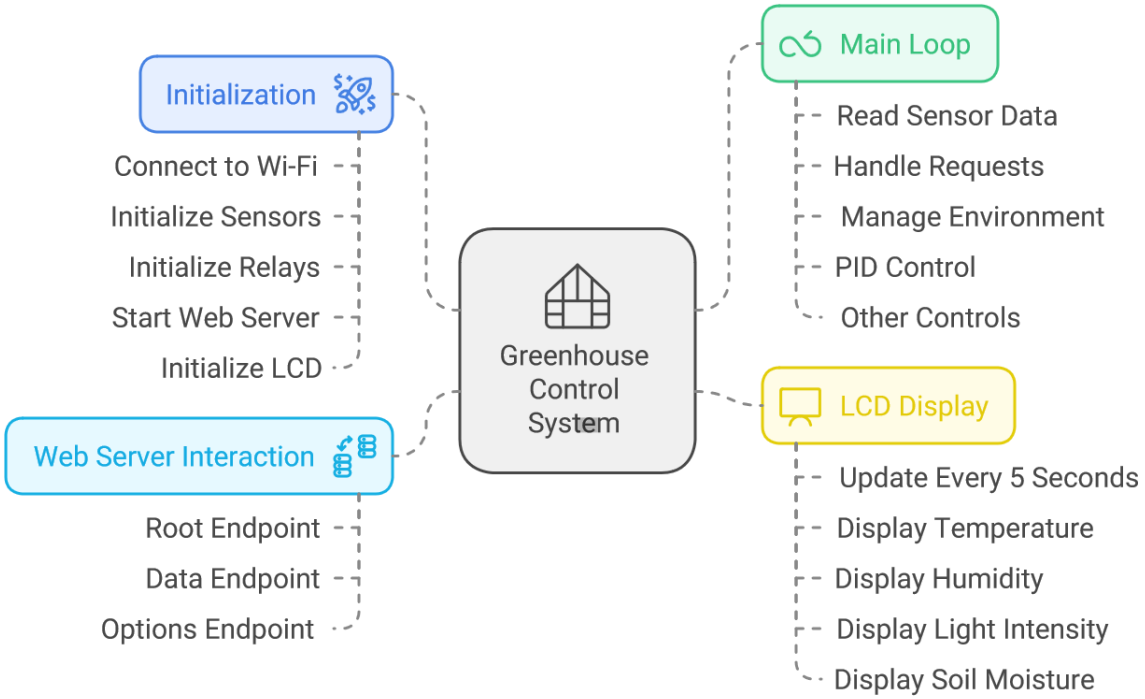

Figure 9 Flow of the system

3.1.3 TECHNOLOGY AND COMPONENT SELECTION REVIEW

This technology review covers the essential components and technologies selected for the design and fabrication of an IoT-based greenhouse monitoring and controlling system. The main components include sensors, actuators, motor drivers, voltage sensors, relays, LCD displays, and microcontrollers. Each component is reviewed based on its suitability for the project requirements.

3.1.3.1 SENSORS

DHT Sensor

DHT stands for Digital Humidity and Temperature. The DHT sensor is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi to measure humidity and temperature instantaneously.

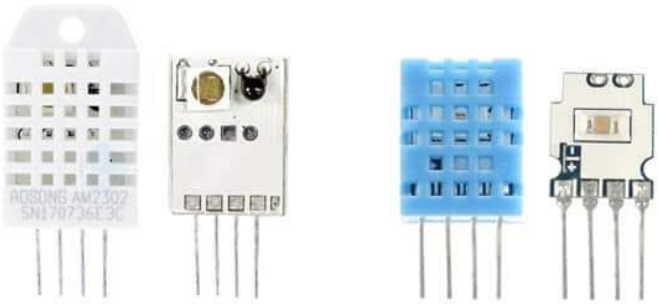

Figure 10 DHT11 sensor.

We have 3 versions of the DHT sensors they are: DHT11 DHT21 and DHT22, here I will only compare the DHT11 and DHT22 since the DHT 21 and DHT22 have similar characteristics.

The DHT22 is bit more expensive as it has better specifications like it is more precise, more accurate and works in a bigger range of temperature & humidity. Its temperature measuring range is from -40°C to +125°C with +-0.5 degrees accuracy, while the DHT11 temperature range is from 0°C to 50°C with +-2 degrees accuracy. Also the DHT22 sensor has better humidity measuring range, from 0 to 100% with 2-5% accuracy, while the DHT11 humidity range is from 20 to 80% with 5% accuracy.

There are few things where DHT11 sensor can be a better choice than DHT22 sensor. As it is less expensive, smaller in size and has higher sampling rate. The sampling rate of the DHT11 is 1Hz i.e. one reading every second, while the sampling rate of DHT22 is 0.5Hz i.e. one reading for every two seconds

DHT sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels

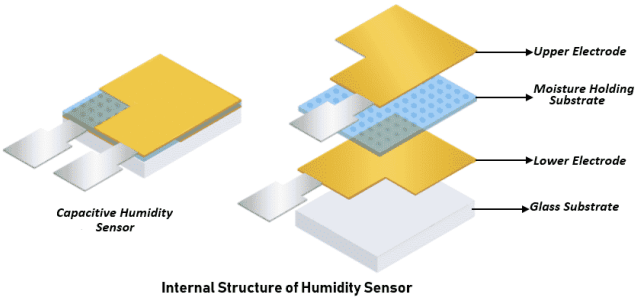

Figure 11 Internal DHT plates.

For measuring temperature this sensor uses a NTC thermistor, the term "NTC" means "Negative Temperature Coefficient", which means that the resistance decreases with increase of the temperature as shown in the graph below. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

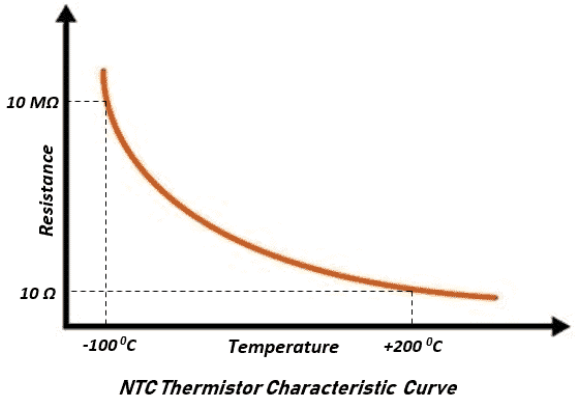

Figure 12 DHT Characteristics curve

Overall Communication Process

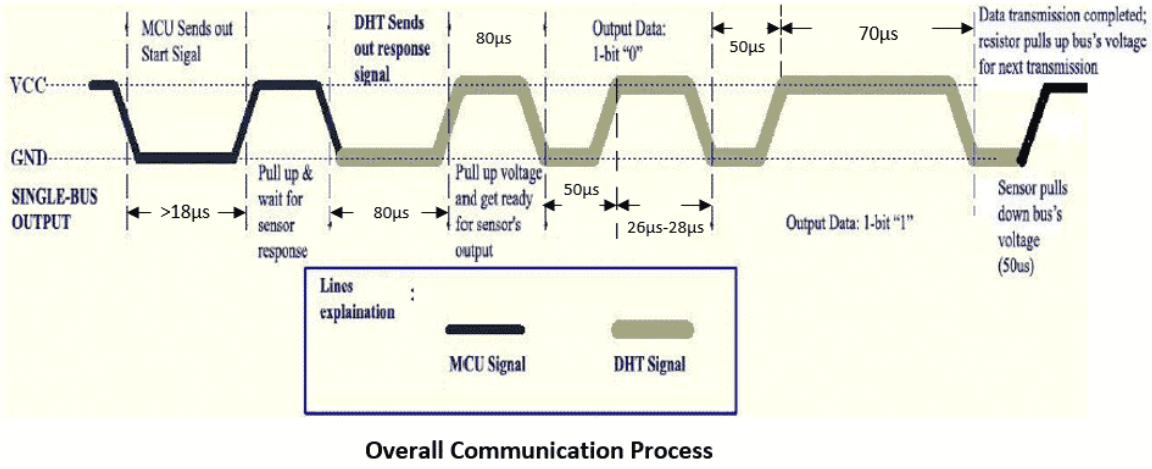

Figure 13 Communication process.

Soil Moisture Sensor

Soil moisture sensors are critical components for monitoring the water content in soil, essential for effective irrigation management in greenhouses. These sensors come in two main types: capacitive and resistive. Both types are simple and inexpensive, providing real-time data on soil moisture levels. The analog output from these sensors can be easily read by the analogue-to-digital converter (ADC) pins on microcontrollers. By continuously monitoring soil moisture, these sensors enable the system to automate irrigation, ensuring that plants receive the optimal amount of water without human intervention. This automation helps in maintaining consistent soil moisture levels, preventing both overwatering and under watering, which can harm plant health and reduce crop yield.

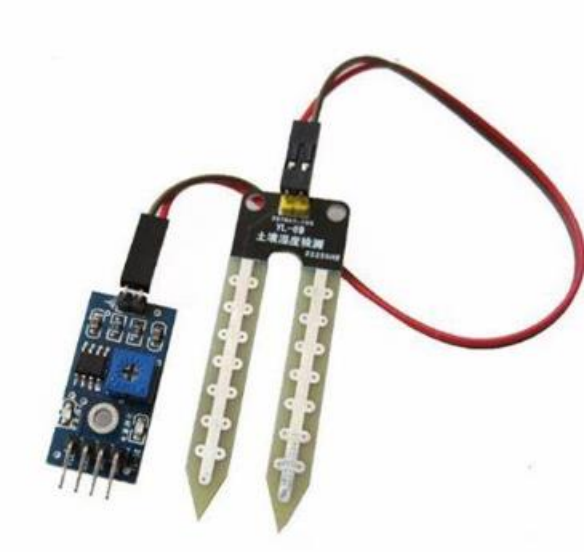

Figure 14 Soil moisture sensor

BH1750 Light Intensity Sensor

The BH1750 is a digital light intensity sensor that measures ambient light in lux, making it ideal for applications where precise light monitoring is required, such as in greenhouses. This sensor features a high resolution and a wide measurement range, from 1 to 65535 lux, which covers typical light conditions found in greenhouses. It communicates via the I2C interface, allowing easy integration with microcontrollers that support I2C communication. The BH1750's low power consumption and high accuracy make it suitable for continuous monitoring of light levels, ensuring that plants receive the appropriate amount of light necessary for optimal growth. This sensor's data can be used to control artificial lighting systems, maintaining the required light conditions automatically.

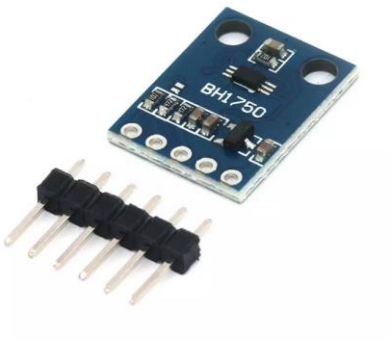

Figure 15 BH1750 sensor

3.1.3.2 ACTUATORS

Water Pump

Water pumps are essential for automated irrigation systems in greenhouses. They ensure that plants receive the correct amount of water based on real-time soil moisture data. Various types of pumps are available, including submersible and surface pumps, each suited to different irrigation needs. The pump selected for this project operates on DC power ranging from 6V to 9V, with a current draw of 2A at 6V. It has a flow rate of 7 liters per minute (L/min) or 110 gallons per hour (GPH) at its maximum 12V rating, and 4.5 L/min or 65 GPH at 6V. The intake and outlet ports have an inner diameter of 8.6mm and an outer diameter of 12mm, with a cutoff height of 1.5 meters (5 feet) or 2.16 psi, meaning it cannot self-prime and must be fed by gravity.

This pump features a brush motor and is not submersible. Its overall size is 30mm x 78.3mm, and it is designed to operate continuously at 12V for no more than 15 minutes due to heat concerns. Additionally, the pump should not be allowed to run dry, as it must be fed with water either by prime or gravity.

Typically controlled via relays or motor drivers, these pumps receive signals from the microcontroller based on soil moisture readings. By automating the irrigation process, the water pump helps maintain consistent soil moisture levels, promoting healthy plant growth and optimizing water usage. This automation reduces the need for manual watering, saving time and labor while ensuring that plants are adequately hydrated.

Two-Blade Cooling Fan

In the design and development of the green mini greenhouse, an effective cooling system is paramount to maintaining optimal growing conditions. For this purpose, a two-blade cooling fan was selected and integrated into the system. This section details the rationale behind the selection of the two-blade cooling fan, its expected effectiveness, the cooling volume, and the relevant equations used to determine its performance.

The two-blade cooling fan was chosen based on several critical criteria. First, two-blade fans are known for their ability to move air efficiently with lower power consumption. This characteristic is vital for the sustainability of the mini greenhouse, which aims to minimize energy usage while maintaining effective cooling. Second, the compact size of the two-blade fan allows it to fit easily within the confined space of the mini greenhouse without obstructing plant growth or other components. Finally, two-blade fans generally offer a balance between performance and cost, making them an economical choice for this application.

The effectiveness of the cooling fan is measured by its ability to lower the temperature within the mini greenhouse to the desired range. The performance is evaluated based on the fan's airflow rate (Q), which is the volume of air it can move per unit time, and its impact on the ambient temperature. The airflow rate Q (measured in cubic feet per minute, CFM) is given by:

$$Q = A \times v \quad \text{.....7}$$

where A is the cross-sectional area of the fan blades (in square feet), and v is the air velocity (in feet per minute). For a two-blade fan with blade length L and blade width W:

$$A = 2 \times (L \times W) \quad \text{..... 8}$$

Substituting A into the equation for Q:

$$Q = 2 \times (L \times W) \times v \quad \text{..... 9}$$

This equation allows us to calculate the volume of air moved by the fan, which is crucial for determining its cooling capacity.

The cooling volume, or the volume of air circulated by the fan, directly impacts the temperature regulation within the mini greenhouse. By ensuring a steady flow of air, the fan helps maintain uniform temperature and humidity levels, preventing hotspots and promoting healthy plant growth. The cooling effect can be approximated using the heat transfer equation:

Heating System Design: Electric Heater

Alongside cooling, maintaining adequate warmth during colder periods is essential for the green mini greenhouse. To achieve this, an electric heater was selected for the heating system. This section outlines the reasons for choosing the electric heater, its effectiveness, and the calculations used to ensure it meets the heating requirements.

The electric heater was selected based on its reliability, ease of control, and efficiency. Electric heaters provide consistent and controllable heat, which is critical for maintaining a stable temperature within the mini greenhouse. They can be easily integrated with the existing control systems to automate temperature regulation. Additionally, electric heaters are relatively compact, allowing them to fit within the limited space of the greenhouse without causing obstruction.

The effectiveness of the electric heater is evaluated by its ability to raise and maintain the internal temperature of the mini greenhouse to the desired levels. The heater's power rating (P), measured in watts, determines its heating capacity. Designing an electric heater for a mini greenhouse involves calculating the heat loss and then determining the heater's power requirement to maintain the desired temperature.

1. Determine the Heat Loss (Q)

- The heat loss through the greenhouse walls, roof, and floor must be calculated. The heat loss is primarily through conduction, and it can be estimated using the formula.

$$Q = \sum_{i=1}^{n} U_i \cdot A_i \cdot \Delta T \quad \text{..... 10}$$

Where:

- Q = Heat loss (in Watts, W)
- U_i = Overall heat transfer coefficient for the ith component (in W/m²·°C)
- A_i = Surface area of the ith component (in m²)
- ΔT = Temperature difference between inside and outside (in °C)

2. Calculate the Required Heater Power (P)

- Once the total heat loss Q is known, the heater power can be sized. The required heater power must match or exceed the heat loss to maintain the desired internal temperature.

$$P = Q \times \text{Safety Factor} \quad \text{..... 11}$$

Where:

- P = Required heater power (in Watts, W)
- The **Safety Factor** is typically between 1.1 to 1.5 to account for unexpected heat losses or inefficiencies.

3. Mini greenhouse Calculations

- Assume a mini greenhouse with the following dimensions:
  - Floor Area (A) = 2 m²
  - Wall Area (A) = 6 m²
  - Roof Area (A) = 2 m²
  - U values (assuming standard glass):
    - Wall (U) = 5.7 W/m²·°C
    - Roof (U) = 5.7 W/m²·°C
    - Floor (U) = 0.5 W/m²·°C
  - Temperature difference (ΔT) = 20°C

- **Heat Loss Calculation:**

$$Q_{\text{walls}} = U_{\text{walls}} \times A_{\text{walls}} \times \Delta T = 5.7 \times 6 \times 20 = 684 \text{ W} \quad \text{..... 12}$$

$$Q_{\text{roof}} = U_{\text{roof}} \times A_{\text{roof}} \times \Delta T = 5.7 \times 2 \times 20 = 228 \text{ W} \quad \text{..... 13}$$

$$Q_{\text{floor}} = U_{\text{floor}} \times A_{\text{floor}} \times \Delta T = 0.5 \times 2 \times 20 = 20 \text{ W} \quad \text{..... 14}$$

$$Q_{\text{total}} = Q_{\text{walls}} + Q_{\text{roof}} + Q_{\text{floor}} = 684 + 228 + 20 = 932 \text{ W} \quad \text{.....15}$$

- **Heater Power Requirement:**

$$P = 932 \times 1.2 \text{ (Safety Factor)} = 1118.4 \text{ W} \quad \text{.....16}$$

  - A heater with at least 1.12 kW power is required.

Considerations

- Insulation: Better insulation reduces heat loss and lowers heater power requirements.
- Ventilation: Account for any necessary ventilation which could increase heat loss.
- Humidity: Consider the impact of humidity control on heating needs.

Figure 16 Humidifier for the mini greenhouse

Humidifier Atomization Plate Circuit Board for Home

Integrating a humidifier is essential for maintaining optimal humidity levels for plant growth. One can achieve this by using an ultrasonic or mist humidifier controlled via a relay module connected to the ESP32 microcontroller. The DHT22 sensor, which is already used for humidity monitoring, will continuously measure the humidity levels inside the greenhouse. If the humidity drops below 30%, the ESP32 will activate the relay,

turning on the humidifier to increase the humidity. Once the humidity reaches the desired level, the humidifier will be turned off.

3.1.3.2 MICROCONTROLLERS

Microcontrollers are compact integrated circuits designed to perform specific control tasks in embedded systems. They serve as the central processing unit of many devices, executing instructions, processing data, and interfacing with various sensors and actuators. Key features of microcontrollers include a central processing unit (CPU) for executing instructions, memory for storing programs and temporary data, input/output ports for interaction with external devices, timers and counters for precise event tracking, and communication interfaces like UART, SPI, and I2C for connecting with other devices.

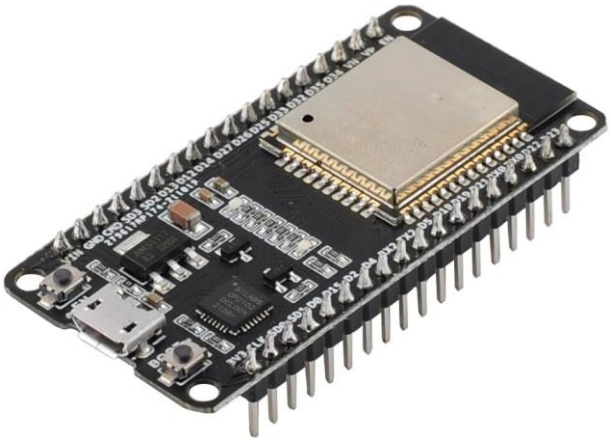

Figure 17 ESP 32

Among the popular microcontrollers are Arduino, known for its ease of use and extensive community support; PIC, favored in industrial applications for its robustness; AVR, recognized for its versatility and performance; and STM32, which offers high performance and a wide range of peripherals. The ESP32, developed by Espressif Systems, stands out for its powerful dual-core CPU, integrated Wi-Fi and Bluetooth capabilities, and versatile I/O options. Its low power consumption and support for various power-saving modes make it ideal for battery-operated applications. The ESP32's combination of high performance, connectivity, and flexibility makes it a preferred choice for modern embedded projects, including smart home systems, industrial automation, and Internet of Things (IoT) applications.

3.1.3.3 COMMUNICATION PROTOCALS

In the realm of IoT applications involving the ESP32, numerous communication protocols can be employed to facilitate device interaction and data exchange. The utilization of Wi-Fi allows the ESP32 to establish connections with wireless networks, thereby enabling internet access and network communication capabilities crucial for cloud-based applications and remote control functionalities. The Controller Area Network (CAN) delivers reliable communication in noisy environments, frequently employed in automotive and industrial settings by interfacing with CAN transceivers. In the realm of web-based communication, HTTP/HTTPS protocols facilitate interactions with web servers and API requests, playing a crucial role in achieving cloud integration. The ESP32's comprehensive support for these diverse protocols positions it as a versatile option for a broad spectrum of IoT projects.

3.2 SYSTEM DESIGN

This section is divided in three parts Software and algorithm, electronics and mechanical

3.2.1 SOFTWARE DESIGN

The design of software for the Internet of Things (IoT) system necessitates the development of a resilient and effective codebase capable of managing diverse responsibilities such as acquiring sensor data, facilitating communication, implementing control logic, and overseeing user interface operations. This software is commonly organized into multiple essential modules:

3.2.1.1 CONTROL ALGORITHM

In this project PID control is implement to the temperature control of the fan in the greenhouse. Below is the design of the system.

Step-by-Step Approach for the design system.

i. Develop a Mathematical Model.
   - Identify the controlled variable: Temperature inside the greenhouse.
   - Identify the manipulated variable: Speed or power of the fan.
   - Identify disturbance variables: External temperature, sunlight intensity.
   - Create a model that relates the temperature inside the greenhouse to the fan speed, considering disturbances.

ii. Visualize the Uncompensated System Response.
   - Simulate the step response of the system without any controller.

iii. Develop Plots for the Uncompensated System.
   - Use MATLAB to generate root locus, pole-zero, Nyquist, and Bode plots for the system.

iv. Design a PID Compensator.
   - Use root locus technique and MATLAB to design a PID controller.
   - Ensure that the system's settling time decreases four-fold, with zero overshoot and zero steady-state error.

v. Visualize the Compensated System Response.
   - Simulate the step response of the system with the PID controller.

vi. Develop Plots for the Compensated System.
   - Generate root locus, pole-zero, Nyquist, and Bode plots for the compensated system.

vii. Develop a Discrete Version of the PID Controller.
   - Convert the designed PID controller into a discrete version.

viii. Implement the Controller on Arduino (greenhouse)
   - Prototype the digital controller on an Arduino

1. Develop a Mathematical Model

For the greenhouse motor fan control system, the key variables are:

- Controlled Variable (CV): Temperature inside the greenhouse.
- Manipulated Variable (MV): Speed of the fan (controlled by the motor).
- Disturbance Variables (DV): External temperature, humidity, solar radiation.

Mathematical Model:

Assuming a first-order linear system for simplicity, the dynamics of the temperature control can be represented as:

$$\frac{dT(t)}{dt} + aT(t) = b \cdot u(t) + d(t) \quad \text{..... 17}$$

where:

- $T(t)$ is the greenhouse temperature.
- $u(t)$ is the control input (fan speed).
- $d(t)$ is the disturbance (external temperature effects).
- a and b are constants representing system parameters.

2. Visualize the Uncompensated Angular Displacement Step Response

The response of the system to a unit step input can be found using the inverse Laplace transform.

For a unit step input $F(s) = \frac{1}{s}$:

$$T(s) = G(s) \cdot \frac{1}{s} = \frac{K}{\tau s+1} \cdot \frac{1}{s} = \frac{K}{s(\tau s+1)}$$

Using partial fraction decomposition:

$$T(s) = \frac{A}{s} + \frac{B}{\tau s+1}$$

Solving for A and B:

$$K = A(\tau s + 1) + Bs$$

$$A = \frac{K}{\tau}, \quad B = -\frac{K}{\tau}$$

So,

$$T(s) = \frac{K}{\tau s} - \frac{K}{\tau} \cdot \frac{1}{\tau s+1}$$

Taking the inverse Laplace transform:

$$T(t) = \frac{K}{\tau}(1 - e^{-\frac{t}{\tau}}) \quad \text{.....18}$$

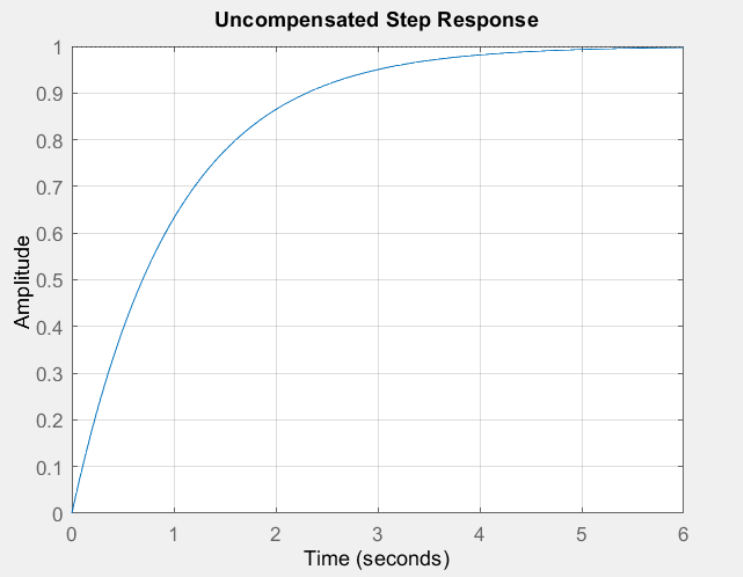

Figure 18 Uncompensated step response, image from Matlab.

3. Develop Plots for the Uncompensated System

a. Root Locus Plot:

The root locus plot shows how the poles of the system change as a parameter (usually gain) varies. For G(s):

$$G(s) = \frac{K}{\tau s+1} \quad \text{..... 19}$$

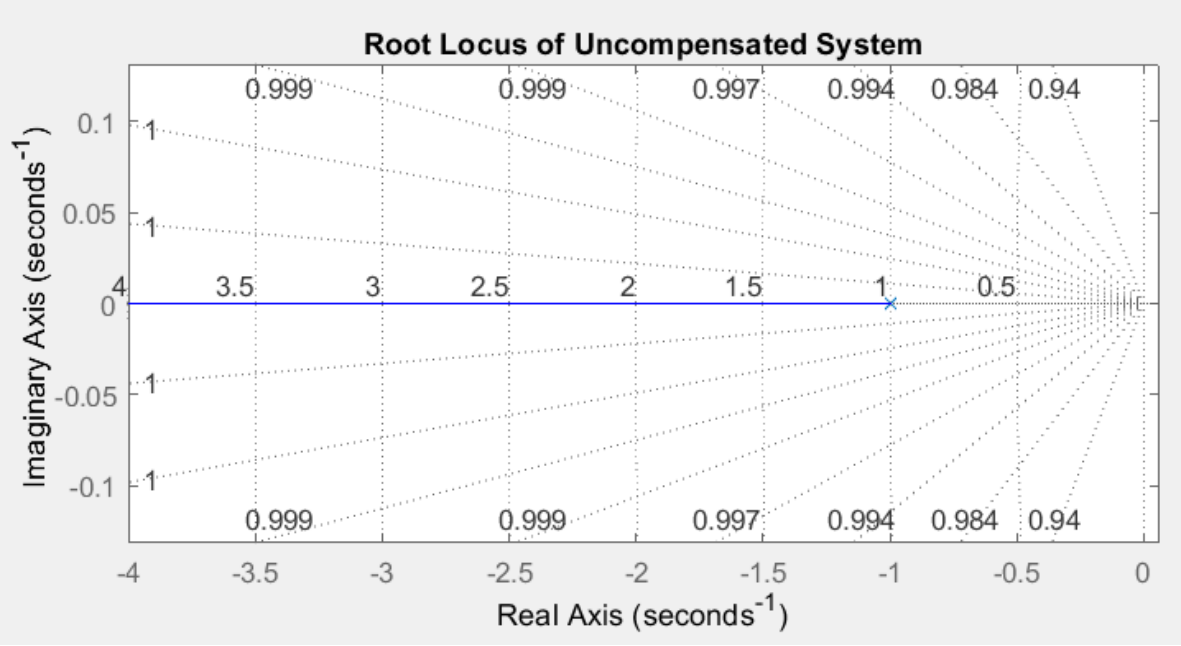

Figure 19 Root locus of uncompensated system, image from Matlab.

b. Pole-Zero Map:
The pole-zero map shows the locations of the system's poles and zeros.
For G(s):

$$\text{Pole: } s = -\frac{1}{\tau} \quad \text{.....} \quad 20$$

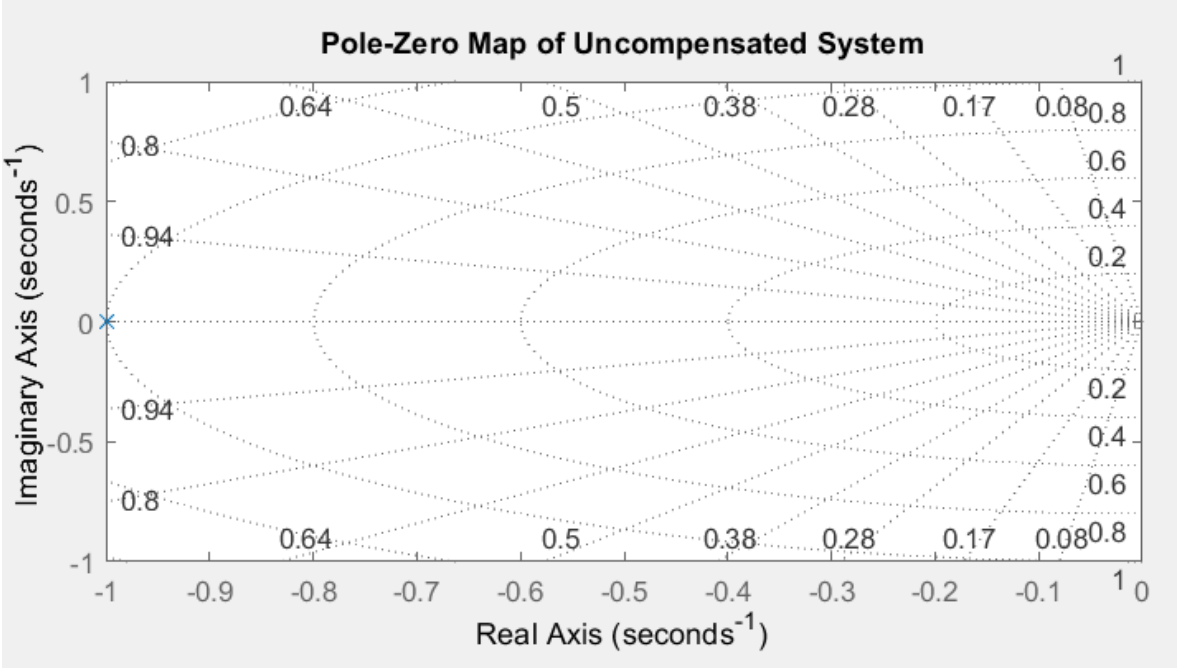

Figure 20 Pole-Zero map of uncompensated system, image from Matlab.

c. Nyquist Plot:

The Nyquist plot shows the frequency response of the open-loop system:

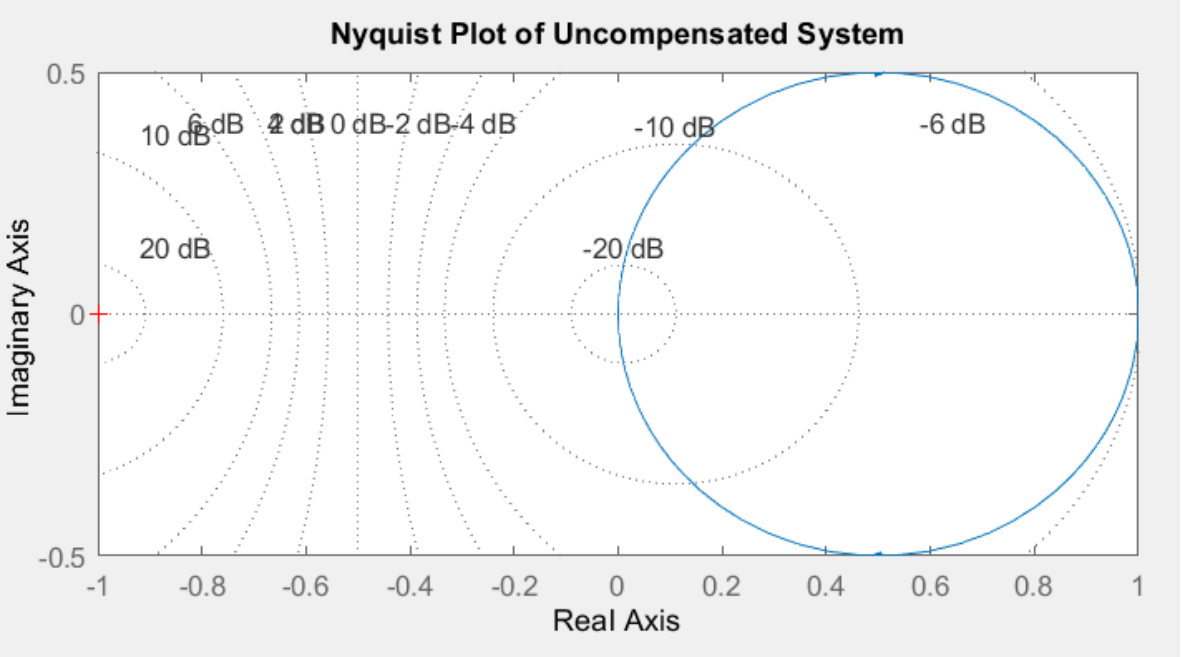

Figure 21 Nyquist plot of uncompensated system, image from Matlab.

d. Bode Plot:
The Bode plot shows the magnitude and phase of the system's frequency response:

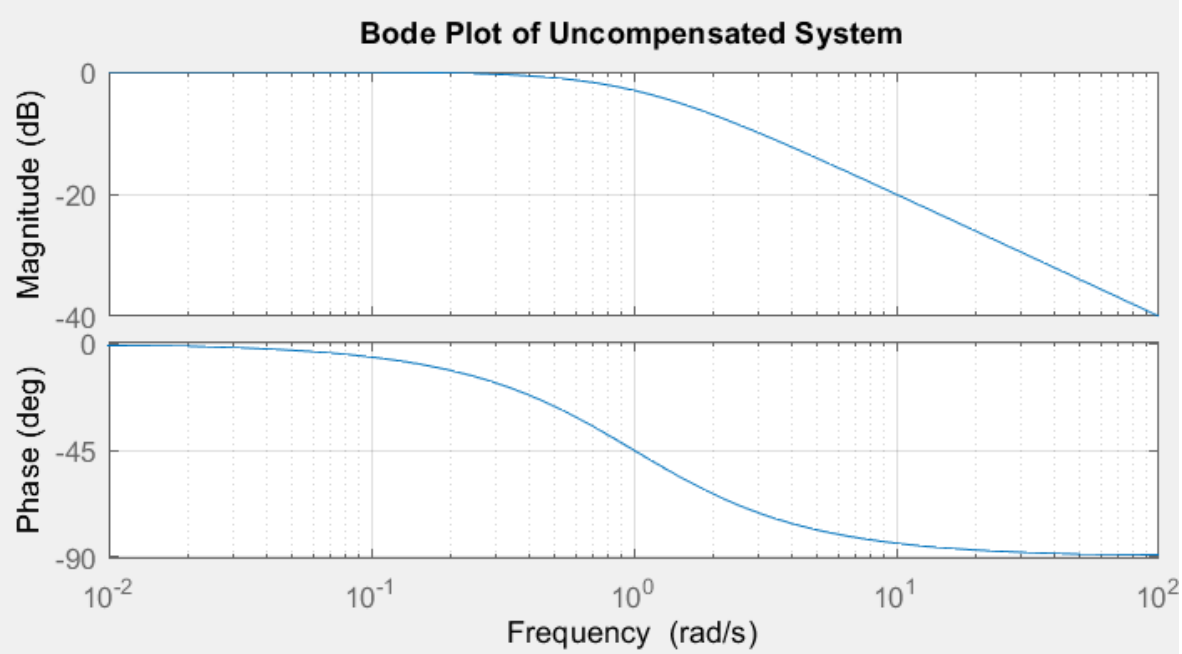

Figure 22 Bode plot of uncompensated system, image from Matlab.

4. Design a PID Compensator

The PID controller has the following transfer function in the Laplace domain:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \quad \text{..... 22}$$

To design a PID controller, we need to find appropriate values for *Kp, Ki,* and *Kd* such that the system meets the performance criteria: settling time decreases four-fold, zero overshoot, and zero steady-state error.
Let's assume some initial values for the PID gains:
Kp=1, Ki=1, Kd=1.

The closed-loop transfer function with the PID controller is:

$$T_{comp}(s) = \frac{C(s)G(s)}{1+C(s)G(s)} = \frac{\left(K_p+\frac{K_i}{s}+K_d s\right)\frac{K}{\tau s+1}}{1+\left(K_p+\frac{K_i}{s}+K_d s\right)\frac{K}{\tau s+1}} \quad \text{..... 23}$$

5. Visualize the Compensated Step Response

The step response of the compensated system can be plotted using MATLAB code provided in the previous step.

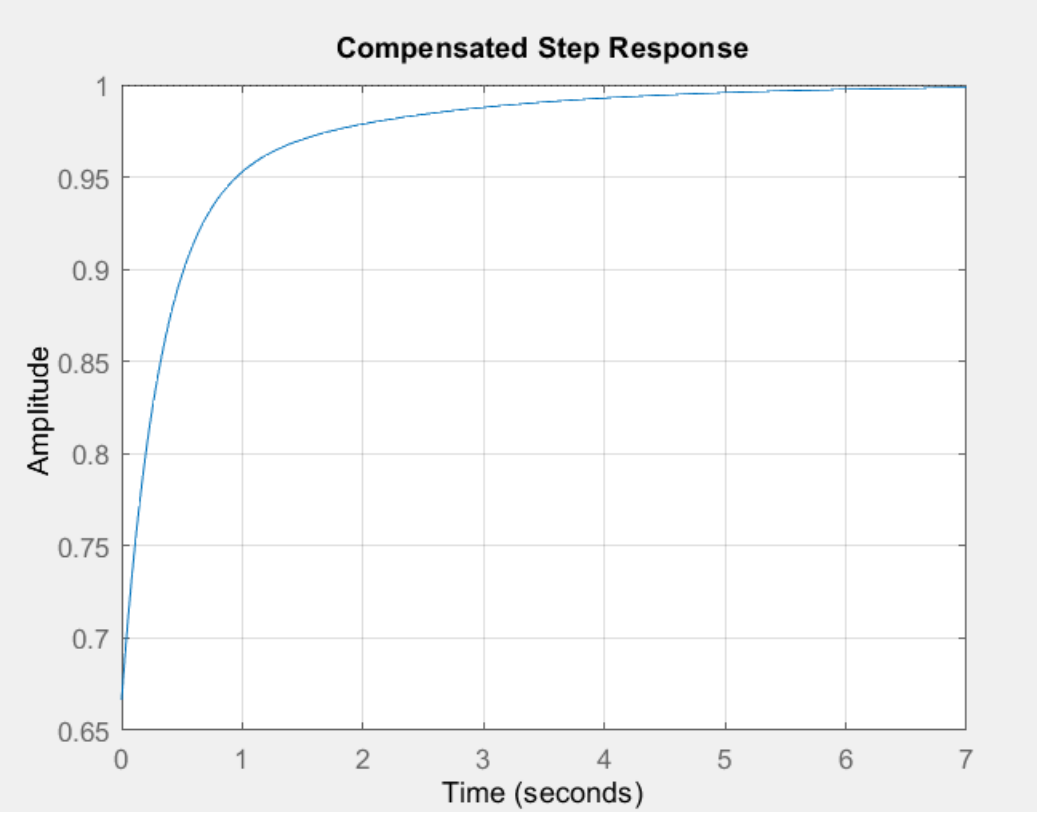

Figure 23 Compensated step response, image from Matlab.

6. Develop Plots for the Compensated System

a. Root Locus Plot:

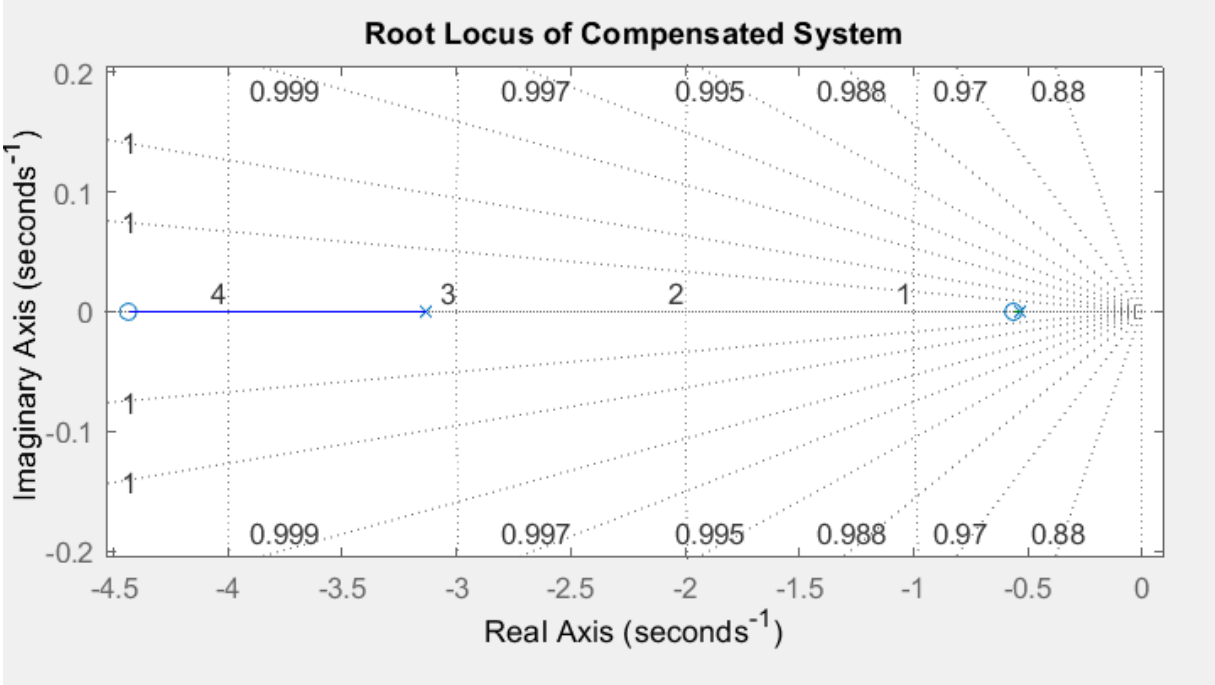

Figure 24 Root locus of compensated system, image from Matlab.

b. Pole-Zero Map:

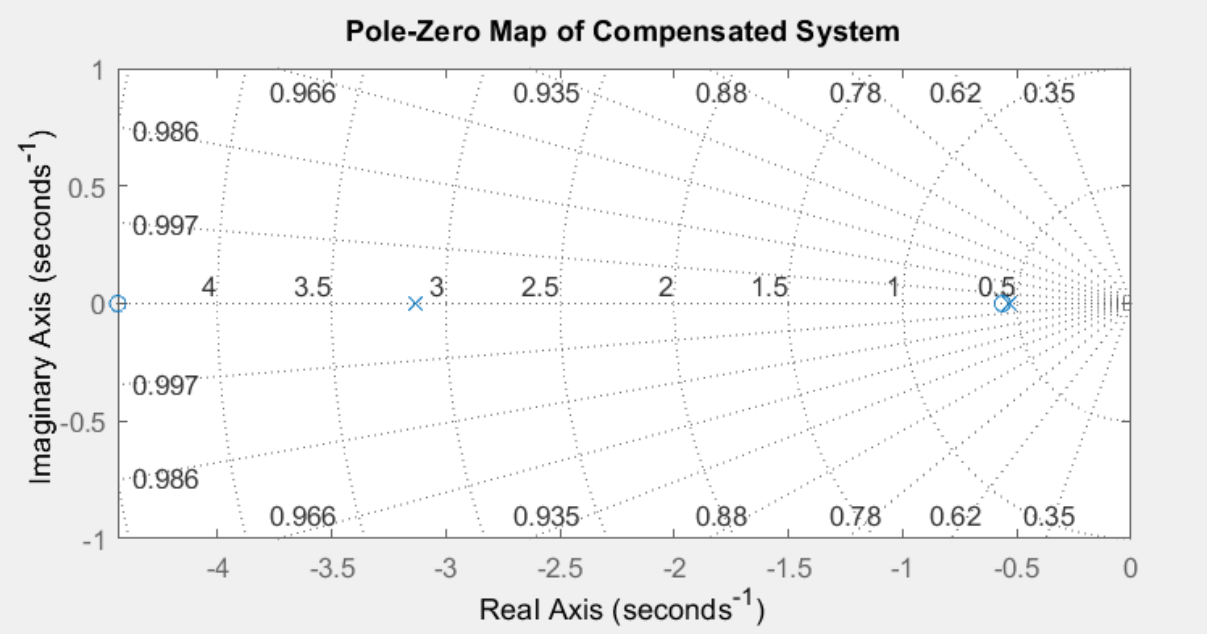

Figure 25 Pole-Zero map of compensated system, image from Matlab.

c. Nyquist Plot:

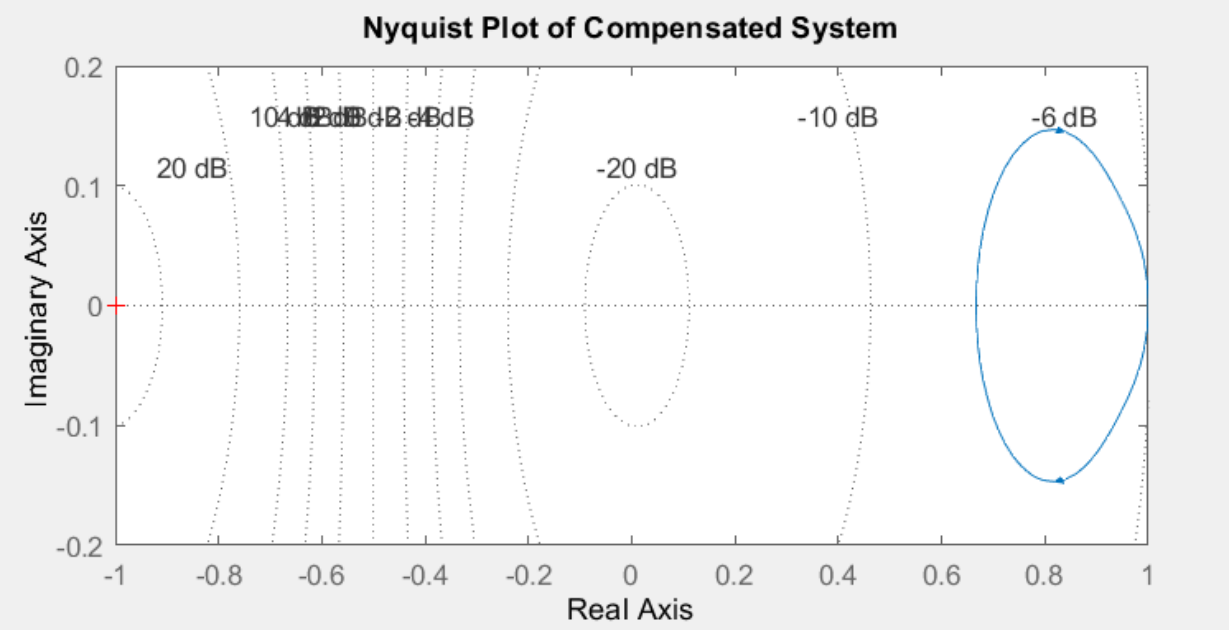

Figure 26 Nyquist plot compensated sytem, image from Matlab.

d. Bode Plot:

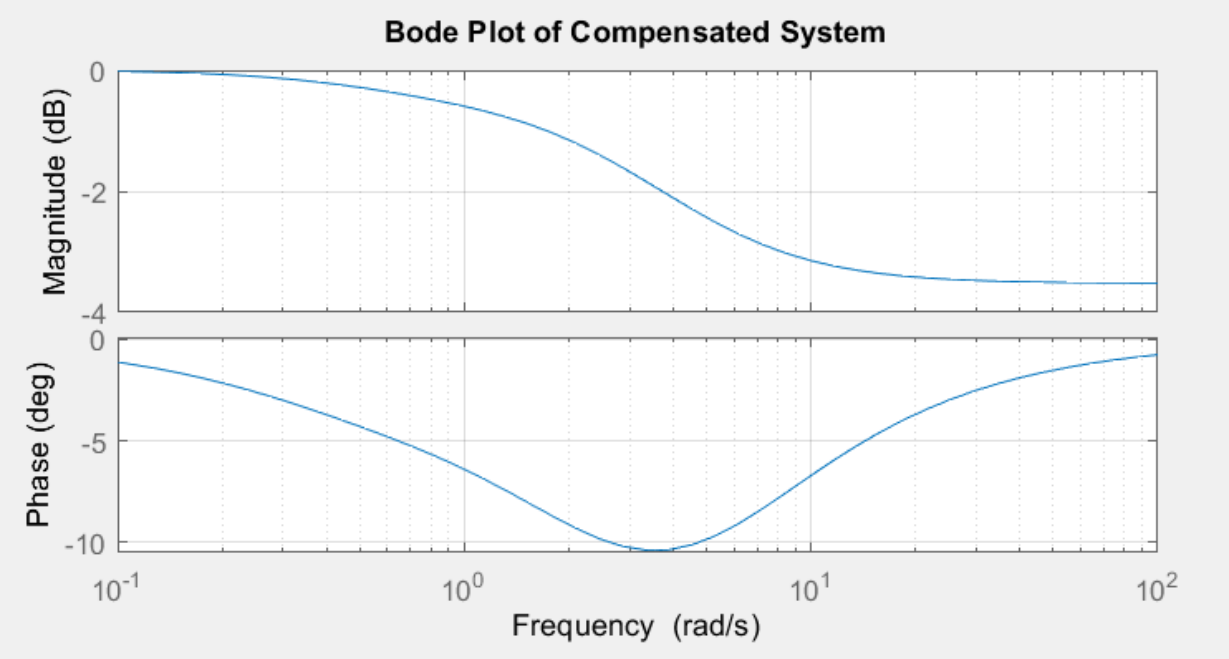

Figure 27 Bode plot of compensated system.

7. Develop a Discrete Version of the PID Controller

To develop a discrete version of the PID controller, we use the Tustin (bilinear) transformation with a sampling time Ts:

$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

Using the Tustin transformation:

$$s = \frac{2}{T_s}\frac{z-1}{z+1}$$

The discrete PID controller becomes:

$$C(z) = K_p + K_i \frac{T_s}{2}\frac{1+z^{-1}}{1-z^{-1}} + K_d \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}} \quad \text{..... 24}$$

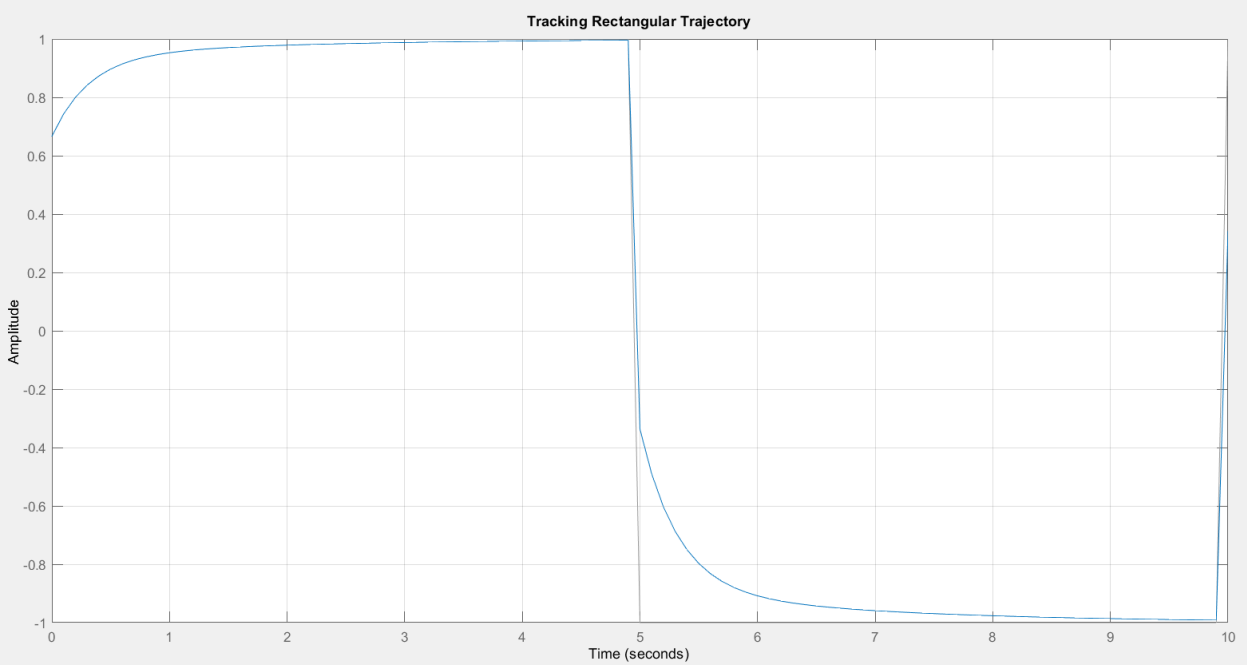

Figure 28 Tracking rectangular trajectory, image from Matlab.

8. Implement the Controller on Arduino

$$u[n] = u[n-1] + K_p(e[n] - e[n-1]) + K_i T_s e[n] + K_d \frac{1}{T_s}(e[n] - 2e[n-1] + e[n-2]) \quad \text{..... 25}$$

Where:

- $u[n]$ is the control output at the current step.
- $e[n]$ is the error at the current step.

3.2.1.2 USER INTERFACE

The design of the user interface for the Internet of Things (IoT)-based greenhouse system aimed to deliver a comprehensive and interactive user experience for the supervision and monitoring of greenhouse conditions. It consisted of a two-part layout accessible via Streamlit, comprising of "Greenhouse Monitoring" and "Greenhouse Assistant."

Within the "Greenhouse Monitoring" segment, users had the ability to observe real-time data concerning temperature, humidity, light intensity, and soil moisture, as well as manage different environmental variables. The interface retrieved information from the ESP32 microcontroller, exhibiting current metrics and illustrating historical patterns through interactive graphs. Users could adjust parameters for temperature, humidity, light, and soil moisture through sliders, with updates being directly transmitted to the ESP32. The real-time updates and visual representations facilitated users in monitoring conditions and making informed modifications to enhance the greenhouse setting.

The "Greenhouse Assistant" division featured an integrated chatbot intended to offer operational guidance and assistance to farmers. This chatbot, accessible within the Streamlit interface, provided suggestions and responses to queries related to greenhouse management. It aided users in comprehending the system operations, resolving problems, and receiving suggestions for refining greenhouse supervision. Through the inclusion of the chatbot, the system improved user experience and delivered valuable support, simplifying the operation and optimization of the greenhouse environment for farmers efficiently.

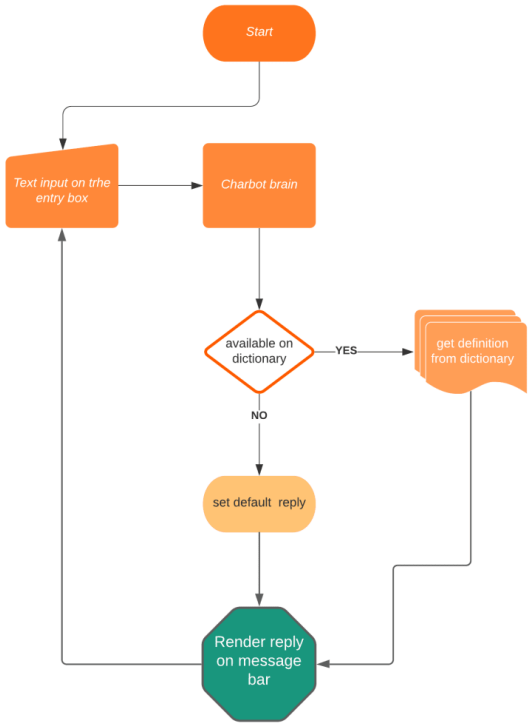

Figure 29 Chat bot flow chart.

3.2.1 HARDWARE

Circuit design

1. Power Supply Calculations

Given:

- **Input Power**: 260V AC
- **Output Power**: 12V DC

Total Power Requirement

First, let's calculate the total power requirement for all components.

Fans (12V, 0.5A each)

- **Power per Fan**:

$$P_{fan} = V_{fan} \times I_{fan} = 12V \times 0.5A = 6W \quad \text{.....} \quad 26$$

- **Total Power for Fans**:

$$P_{total\_fans} = 2 \times P_{fan} = 2 \times 6W = 12W$$

Pump (12V, 1A)

- **Power**:

$$P_{pump} = V_{pump} \times I_{pump} = 12V \times 1A = 12W$$

Humidifier (5V, 0.5A)

- **Power**:

$$P_{humidifier} = V_{humidifier} \times I_{humidifier} = 5V \times 0.5A = 2.5W$$

LED Strip (5V, 0.5A)

- **Power**:

$$P_{LED} = V_{LED} \times I_{LED} = 5V \times 0.5A = 2.5W$$

Power Conversion from 12V to 5V

Using a DC-DC step-down (buck) converter with an efficiency of 85%, we calculate the power drawn from the 12V supply for the 5V devices.

- **Total 5V Power**:

$$P_{total5V} = P_{humidifier} + P_{LED} = 2.5W + 2.5W = 5W$$

- **Power Drawn from 12V Supply** (considering efficiency):

$$P_{drawn} = \frac{P_{total5V}}{\text{Efficiency}} = \frac{5W}{0.85} \approx 5.88W$$

Total Power Requirement from 12V Supply

- **Total Power**:

$$P_{total} = P_{total\_fans} + P_{pump} + P_{drawn} = 12W + 12W + 5.88W \approx 29.88W$$

- **Total Current from 12V Supply**:

$$I_{total} = \frac{P_{total}}{12V} \approx \frac{29.88W}{12V} \approx 2.49A$$

2. **DHT11 Temperature and Humidity Sensor**

Temperature Calculation

The DHT11 provides a direct digital output for temperature in °C.

- **Temperature Range**: 0°C to 50°C
- **Accuracy**: ±2°C

Example: If the DHT11 output reads 25°C, this is the actual temperature.

Humidity Calculation

The DHT11 also provides relative humidity as a direct digital output.

- **Humidity Range**: 20% to 90% RH
- **Accuracy**: ±5% RH

Example: If the DHT11 output reads 60% RH, this is the actual relative humidity.

3. **BH1750 Light Intensity Sensor**

Illuminance Calculation

The BH1750 outputs a direct digital value in lux, which represents the light intensity.

- **Light Intensity Range**: 1 - 65535 lux
- **Resolution**: 1 lux

Example: If the BH1750 reads 200 lux, this is the ambient light intensity.

4. **Soil Moisture Sensor**

Voltage to Moisture Conversion

The soil moisture sensor provides an analog voltage corresponding to soil moisture.

- **Analog Voltage Range**: 0V (dry) to 5V (wet)
- **ADC Resolution**: Let's assume a 10-bit ADC with a 5V reference.

- **Moisture Percentage**:

$$\text{Soil Moisture (\%)} = \left(\frac{\text{ADC Value}}{1023} \right) \times 100\%$$

Example: If the ADC value is 512:

$$\text{Soil Moisture (\%)} = \left(\frac{512}{1023} \right) \times 100\% \approx 50\%$$

5. **L298N Motor Driver for Fans**

Power Dissipation in L298N

The L298N driver will dissipate power due to the voltage drop across the internal transistors.

- **Voltage Drop per Channel**: Approx. 1.8V
- **Power Dissipation per Channel**:

$$P_{dissipation\_per\_channel} = I_{motor} \times V_{drop} = 1A \times 1.8V = 1.8W$$

- **Total Power Dissipation**:

$$P_{dissipation\_total} = 2 \times P_{dissipation\_per\_channel} = 2 \times 1.8W = 3.6W$$

Ensure the L298N is adequately heat-sinked to handle this dissipation.

Summary of Calculations:

1. **Total Power Supply Requirement**: The 12V DC power supply should provide at least 3A (considering a safety margin).
2. **Sensor Calculations**: Direct digital readings from DHT11 and BH1750, with voltage to moisture percentage conversion for the soil moisture sensor.
3. **L298N Power Dissipation**: Adequate cooling is necessary for the L298N motor driver due to power dissipation.

4. CHAPTER FOUR

4.1 MANUFACTURE AND ASSEMBLY METHODS

The greenhouse structure was manufactured using durable materials like aluminium or steel for the frame, with transparent acrylic or polycarbonate sheets for the walls and roof to provide insulation and light transmission. The components were precisely cut and shaped using a CNC machine or laser cutter, followed by assembly using nuts, bolts, and corner brackets. The panels were attached with either adhesive or screws, and weather stripping and silicone sealant were applied along the joints to ensure the greenhouse was airtight and watertight.

For the circuit, the process began with designing the PCB using software like Eagle or KiCad to create a compact layout that accommodated all necessary components, such as sensors, microcontrollers, and relays. The design was then sent to a PCB manufacturer for production or fabricated using a CNC milling machine for prototyping. Components like resistors, capacitors, transistors, relays, and sensors were sourced from reliable suppliers, and soldering was performed to attach these components to the PCB, ensuring that the solder joints were clean and secure to prevent short circuits. Wiring was organized and routed using cable ties and terminal blocks to maintain a clean and functional layout, with color-coded wires for easy identification.

During assembly, the PCB was mounted inside the greenhouse using standoffs, positioned to avoid interference with other components and to allow easy access for maintenance. A wiring harness was created to bundle cables that connected various sensors and actuators to the control board, improving safety and reducing clutter. Sensors like temperature, humidity, and soil moisture were placed in optimal positions inside the greenhouse, with actuators such as fans, pumps, and lights securely mounted to the frame. The circuit was connected to a regulated power supply that provided the necessary voltage and current, ensuring stability and sufficient load capacity.

Testing and calibration were crucial steps, starting with circuit testing through continuity checks to confirm correct connections, followed by powering up the circuit to verify functionality. Calibration of sensors and actuators ensured they operated within the desired parameters, with set points in the software adjusted to match the specific greenhouse environment requirements. Troubleshooting tools like multimeters and oscilloscopes were used if issues arose.

4.2 MANUFACTURING METHODS

Component Assembly: The first step in the manufacturing process involves assembling the various components of the greenhouse automation system. This includes mounting the ESP32 microcontroller, L298N motor driver, sensors, and other electronic components onto a suitable base or enclosure. The components are securely attached using screws, adhesive, or mounting brackets to ensure stability. The wiring is carefully routed and connected according to the circuit diagrams to ensure proper electrical connections. Careful attention is given to soldering joints and connectors to avoid issues with signal integrity and power delivery.

Enclosure Fabrication: The next step involves fabricating the physical enclosure for the greenhouse control system. This can be achieved using various materials such as acrylic sheets, metal enclosures, or custom 3D-printed parts. The design of the enclosure should ensure adequate ventilation for heat dissipation and protection from environmental factors. The enclosure is cut, shaped, and assembled using techniques such as laser cutting, CNC machining, or 3D printing, depending on the material used. Openings are made for connectors, displays, and ventilation as required. The enclosure is then tested to ensure it fits all components securely and provides the necessary access for operation and maintenance.

System Integration and Testing: Once the components and enclosure are assembled, the next phase involves integrating the system and performing thorough testing. This includes checking the functionality of each component, verifying that the sensors and actuators operate correctly, and ensuring that the software and hardware interfaces work as intended. Testing is conducted to confirm that the system responds appropriately to various environmental conditions and that the control algorithms perform as expected. Any issues identified during testing are addressed through adjustments or modifications to ensure the system operates reliably in a greenhouse environment.

Quality Control and Calibration: After successful integration and testing, the final step is quality control and calibration. This involves inspecting the entire system for any defects or inconsistencies and calibrating sensors and actuators to ensure accurate readings and control. Calibration procedures are followed to fine-tune the system's performance and ensure that it meets the required specifications. The final product is then documented, and user manuals or operational guides are prepared to assist with installation and use.

4.3 ASSEMBLY METHODS

The assembly of the greenhouse automation system involves several key methods to ensure that all components are integrated and function correctly. The process begins with **component placement and secure mounting**, where electronic parts such as the ESP32 microcontroller, sensors, and motor drivers are positioned on a suitable base or enclosure. Components are mounted using screws, adhesive, or custom brackets, ensuring stability and minimizing movement. Proper routing of wiring is essential to avoid tangling and interference, with connections made according to detailed circuit diagrams.

Soldering and Connector Assembly follow, where electrical connections are established through soldering wires to connectors or directly to the circuit boards. This step requires precision to ensure strong, reliable joints and to avoid short circuits or connectivity issues. For components like the L298N motor driver, careful attention is given to the alignment and secure soldering of pins.

Next, the **enclosure fabrication** involves cutting and shaping the housing for the system. Depending on the material—such as acrylic, metal, or 3D-printed plastic—the enclosure is prepared using techniques like laser cutting, CNC machining, or 3D printing. Openings are created for connectors, displays, and ventilation to accommodate the assembled components.

Finally, **integration and testing** are performed to ensure that all parts work together seamlessly. This includes connecting the assembled components to power and performing functional tests to verify operation. Any necessary adjustments or modifications are made to address issues identified during testing. This systematic assembly approach ensures that the greenhouse automation system is robust, reliable, and ready for deployment.

4.3.1 CAD MODEL

The CAD model represents a well-structured mini greenhouse system with a focus on automation and efficient space utilization. The greenhouse features a semi-cylindrical roof, optimized for light management and insulation, with a transparent structure likely made of plastic. The design includes a central growing area with planting beds, and a control unit housing electronics. The model shows a symmetrical layout that would help ensure even distribution of light, air, and water, promoting optimal growing conditions.

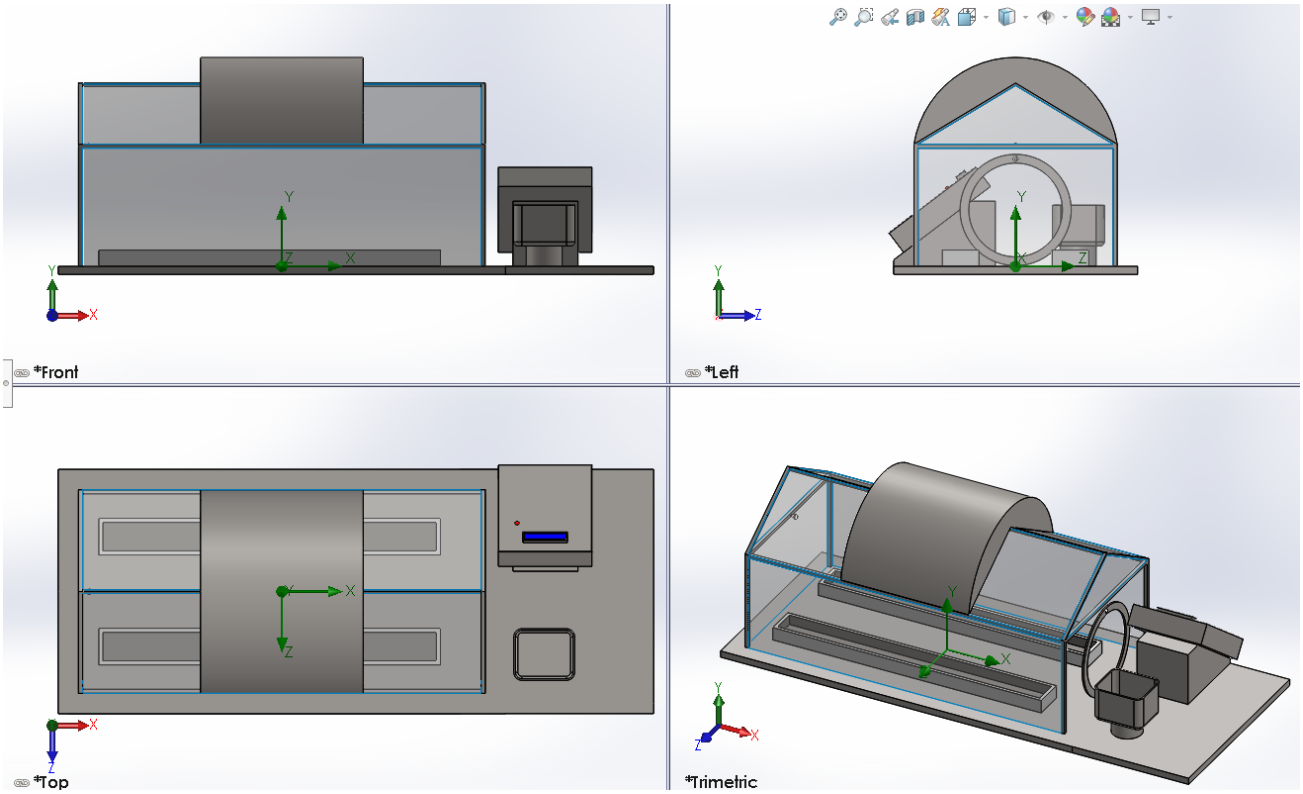

Figure 30 Greenhouse cad model, image from solidworks.

4.3.2 PROTOTYPE

Figure 31 Mini greenhouse prototype, captured image.

4.4 CIRCUIT DESIGN

The circuit simulation layout illustrates a greenhouse automation system that uses an ESP32 microcontroller to monitor and control environmental conditions. Sensors, including a DHT22 for temperature and humidity and a soil moisture sensor, feed data to the ESP32, which processes this information to manage actuators like fans, heaters, and pumps via relays. The system includes a power supply circuit to ensure all components receive the correct voltage and current, and LED indicators provide visual feedback on the status of the actuators. This setup is designed to maintain optimal conditions within the greenhouse through automated control.

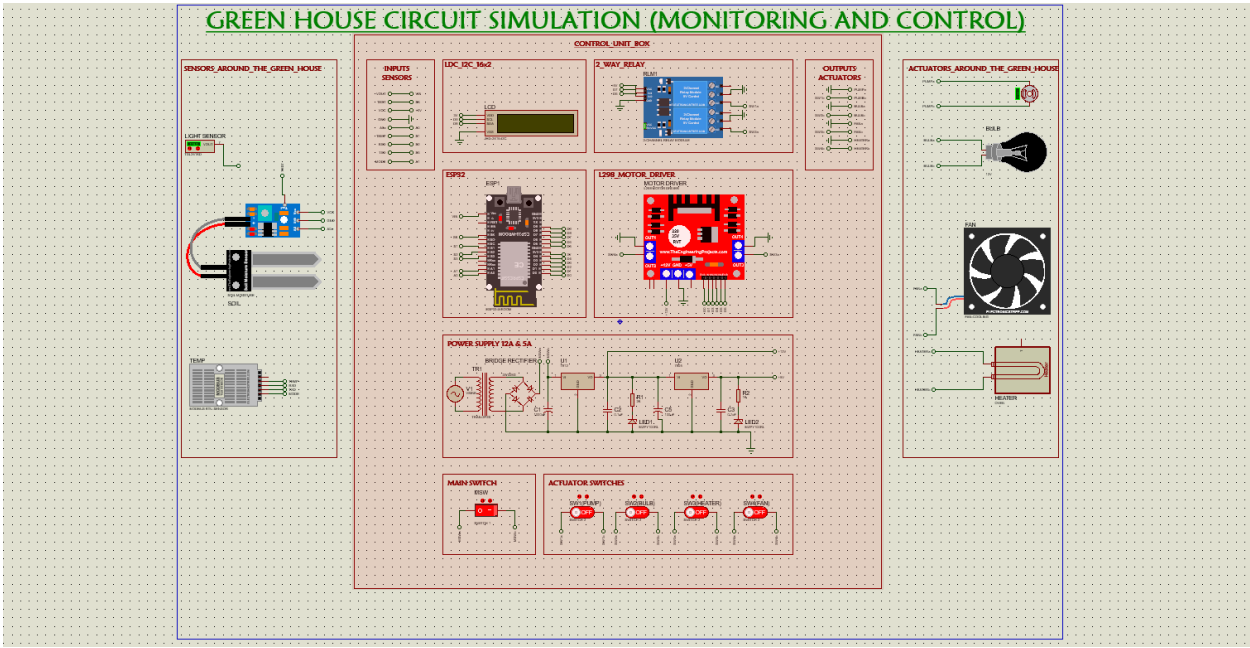

Figure 32 Greenhouse automation system, image from proteus.

4.5 MODELLING AND SIMULATION

4.5.1 SENSOR SIMULATIONS

Table 1 Data collected from sensor real time readings

Parameter	Graph Color	Unit	Range	Duration (s)
Simulated Temperature	Red	°C	15 to 35	0 to 60
Simulated Humidity	Blue	%	35 to 65	0 to 60
Simulated Light Intensity	Green	lux	200 to 800	0 to 60
Simulated Soil Moisture (analog)	Magenta	Analog value	500 to 1500	0 to 60

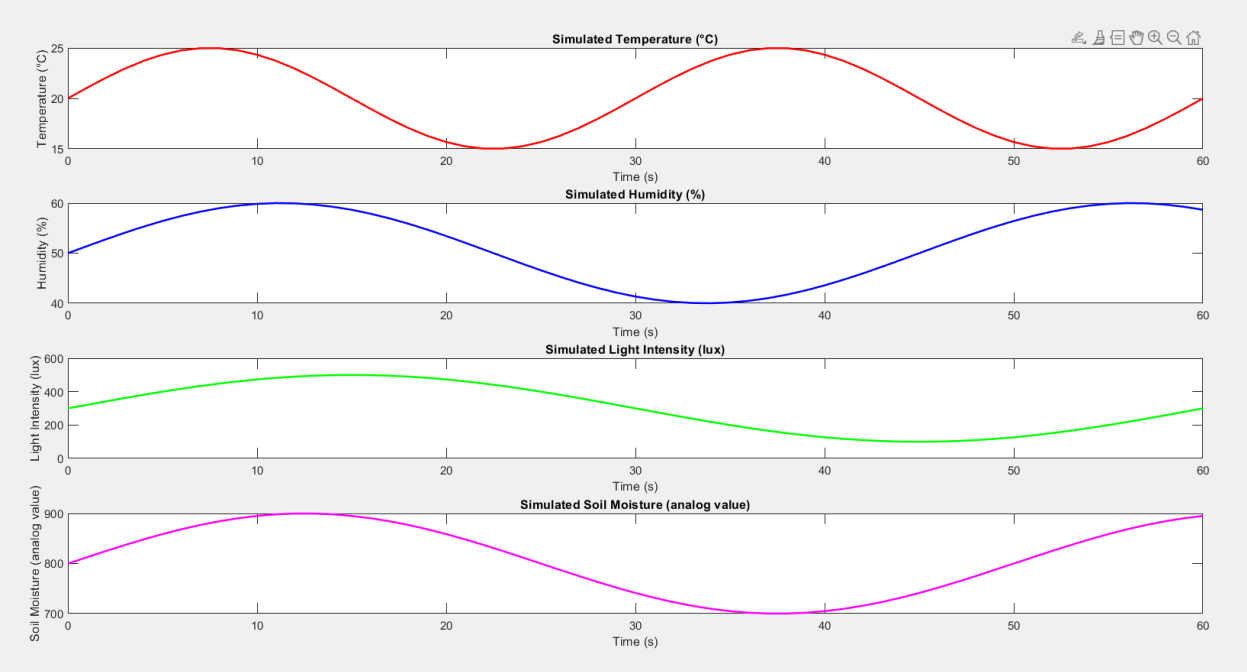

Figure 33 Depiction of simulated sensor data, image from Matlab.

The illustration in Figure 4.5.1 depicts simulated sensor data for key environmental parameters within a greenhouse, encompassing temperature, humidity, light intensity, and soil moisture. These simulations play a vital role in the advancement and fine-tuning of a greenhouse monitoring and control setup. Here, I elaborate on the significance of calibrating and simulating these sensor readings:

Significance of Calibration

Precision and Accuracy: The calibration of sensors is imperative to guarantee the accuracy and dependability of the data they furnish. Erroneous sensor data could lead to inappropriate control measures, ultimately impacting the growth of plants. For example, an un-calibrated temperature sensor might provide inaccurate readings, causing the system to either overheat or undercool the greenhouse environment.

Consistency in Data Collection: Calibration ensures that sensors offer consistent data over time, which is crucial for upholding stable environmental conditions within the greenhouse. Consistent humidity and soil moisture measurements, for instance, are essential for triggering the appropriate irrigation and ventilation responses.

Compensation for Sensor Drift: With time, sensors may encounter drift, wherein their readings gradually diverge from the true values. Regular calibration aids in rectifying this drift and upholding sensor precision.

Significance of Simulation

Validation and Testing of the System: Simulating sensor data permits the testing and validation of the greenhouse control system under diverse conditions without jeopardizing actual plants. This proves particularly beneficial during the developmental phase when comprehending and enhancing system behavior is essential. For instance, simulating temperature fluctuations aids in evaluating the response of heating or cooling systems and ensuring they maintain the desired temperature range.

Training and Optimization: Simulation offers a secure setting for training operators and refining control algorithms. It facilitates experimentation with various control tactics and observation of their impact on the simulated environment. Enhancing how the system responds to alterations in light intensity or soil moisture can result in improved resource management and enhanced plant well-being.

Early Detection of System Concerns: Simulation aids in the identification of potential issues within the system, such as delays in sensor response or incorrect actuator triggers, before they pose risks in a real-world setting. If a simulated soil moisture sensor displays erratic behavior, it can be rectified prior to deploying the system in an operational environment.

The calibration and simulation of sensor data represent fundamental stages in the development, evaluation, and implementation of an IoT-based greenhouse monitoring and control system. Calibration guarantees the system's operation with precision and dependability, while simulation offers a controlled arena for testing, validating, and enhancing the system's efficiency. Together, these processes boost the overall effectiveness of the greenhouse system, resulting in enhanced plant growth, resource effectiveness, and system dependability.

4.5.1 CIRCUIT SIMULATIONS

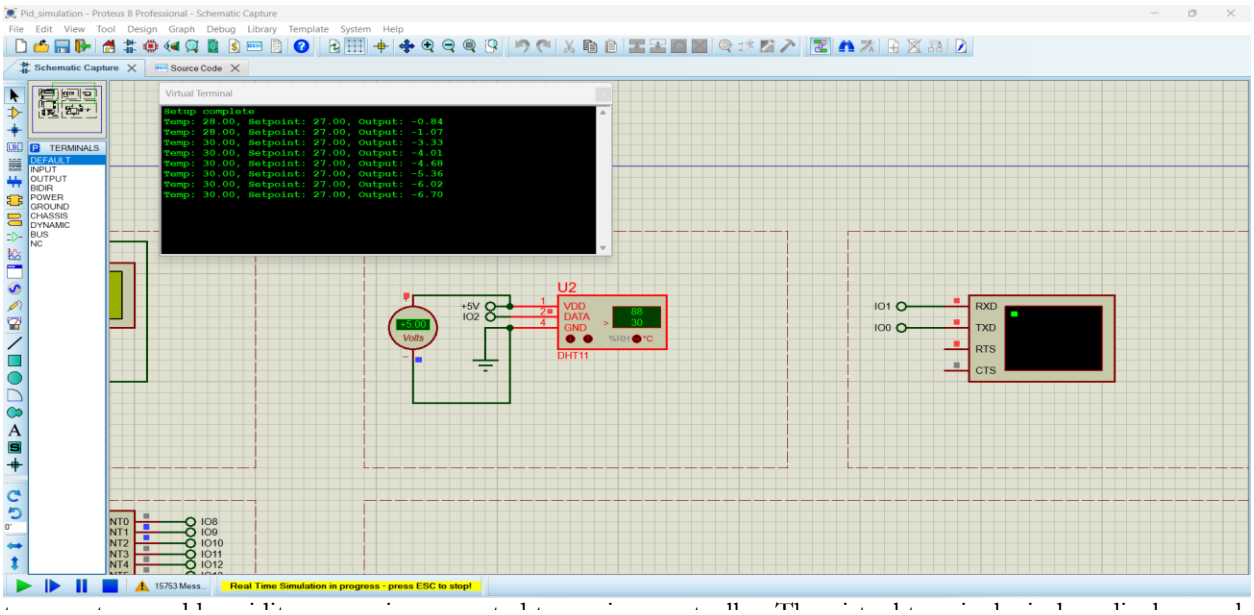

Figure 34 Simulation of a DHT, image from proteus.

DHT sensor (U2) connected to the microcontroller via digital pins. The microcontroller processes the sensor's data and sends it to the virtual terminal, where the values are displayed, confirming that the sensor and circuit are functioning as expected. The circuit seems to be part of a broader system, as indicated by additional components and connections in the schematic.

Figure 35 Analogue analysis of the DHT sensor, image from proteus.

5. CHAPTER FIVE

5.1 SYSTEM TESTING

System testing is a critical phase in ensuring that the greenhouse automation system operates as intended under various conditions. This phase involves validating the functionality of each component, including the ESP32 microcontroller, sensors, actuators, and control algorithms. Testing begins with verifying individual components, such as checking sensor accuracy and actuator response. For instance, the temperature and humidity sensors are tested to ensure they provide accurate readings within the specified ranges, while the motor drivers and relays are assessed for proper control of the fan, heater, and humidifier. Each component's performance is evaluated against its expected behaviour to confirm that it meets the design requirements.

Following component validation, integration testing is performed to ensure that all system parts work cohesively. This involves running the complete greenhouse automation system and simulating various environmental conditions to test the system's response. The system's ability to control temperature, humidity, and soil moisture. Additionally, the Streamlit UI dashboard is tested to confirm it provides accurate real-time monitoring and control capabilities. Any discrepancies or issues identified during this phase are documented and addressed to refine the system's reliability and performance. Comprehensive testing ensures that the greenhouse automation system operates efficiently and effectively in real-world scenarios.

5.2 CONDUCTED TESTS

5.2.1 EXPERIMENT 1: TEMPERATURE REGULATION

In the first experiment concerning temperature regulation, a Proportional-Integral-Derivative (PID) controller is employed to uphold a specified target temperature within a defined range. The methodology of achieving temperature regulation with the PID controller and the functions of the heater and fan are elucidated in this context.

Fundamentals of PID Controller

A PID controller is a feedback control mechanism extensively utilized in industrial settings to regulate processes. It adapts the control inputs to minimize the deviation between the desired set point (target temperature) and the actual temperature. The PID controller operates based on three fundamental constituents:

Proportional (P): This element adjusts the control output proportionately to the existing error (discrepancy between the target and actual temperatures), with a larger error leading to a more substantial control action. Integral (I): This constituent deals with the accumulation of prior errors by integrating the error over time, aiding in eliminating any residual steady-state error not addressed solely by the proportional term. Derivative (D): This component forecasts future errors by assessing the error's rate of change to enhance stability and dampen the response by reacting to temperature variations.

Execution of Temperature Regulation

Temperature Measurement: The current temperature of the environment or system is gauged using a temperature sensor such as a thermocouple in this case the DHT11. Error Calculation: By deducting the measured temperature from the target temperature, the PID controller computes the error. Application of PID Algorithm: The controller processes the error through the PID algorithm to determine the requisite adjustment, calculating the proportional, integral, and derivative terms to formulate a control signal. Actuator Control:

Heater: In instances where the temperature is below the target, the PID controller elevates the heater's output to introduce thermal energy into the system, thereby elevating the temperature towards the target. Fan: If the temperature surpasses the target or cooling is necessary, the PID controller augments the fan's output to dissipate heat from the system, consequently lowering the temperature.

Illustrative Scenario

Here the target temperature is 23°C, the PID controller acquires the current temperature from the sensor, resulting in an error of +5°C when the current temperature is 28°C.

Proportional Component: The controller adjusts the fan output in proportion to the +5°C error. Integral Component: Persistent deviations below the target temperature lead to the accumulation of error over time by the integral term, further enhancing the fan output to rectify the prolonged deviation. Derivative Component: In cases of rapid temperature escalation, the derivative term aids in modifying the fan output to avert overshooting the target temperature.

In situations where the temperature decreases below 25°C, the PID controller curtails the fan output while boosting the heater's output to facilitate system heating.

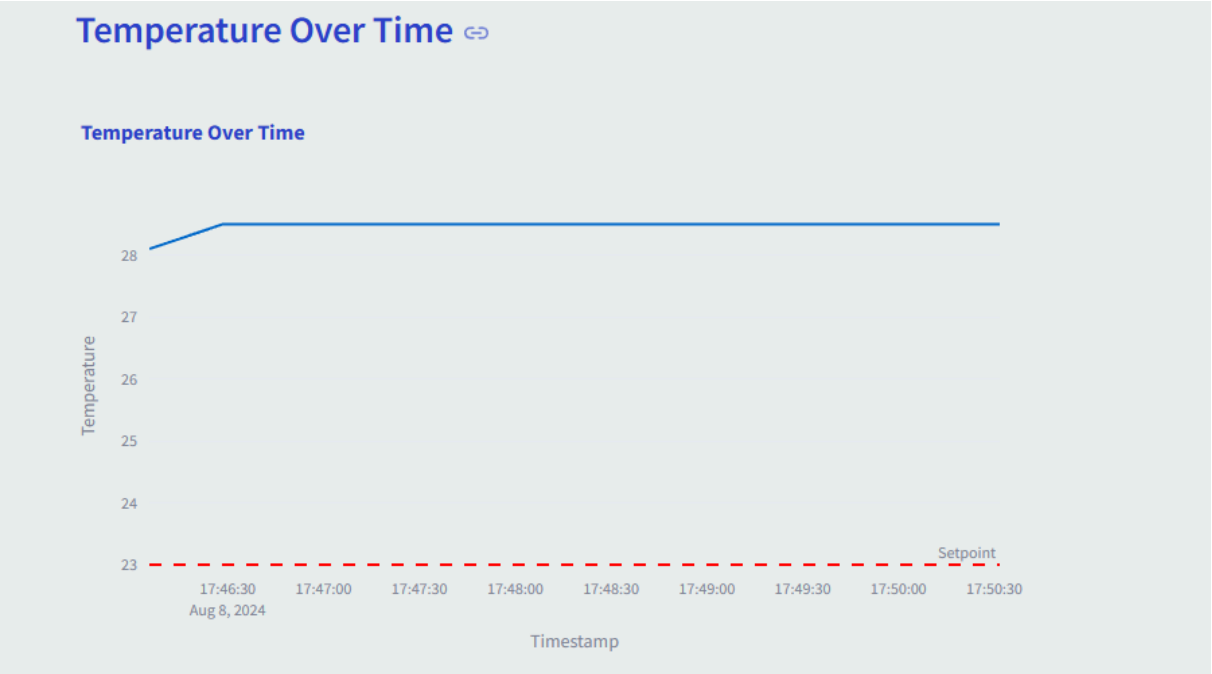

Figure 36 Before control actuator was activated, image from the greenhouse UI app.

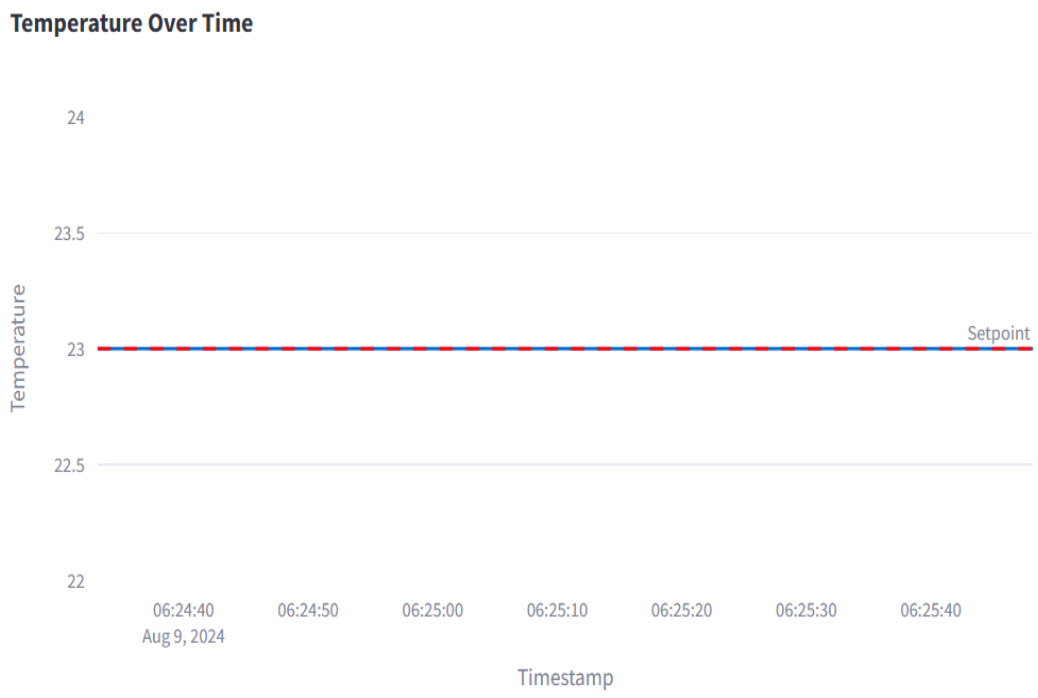

Figure 37 Temperature graph after control actuator was activated, image from the greenhouse UI app.

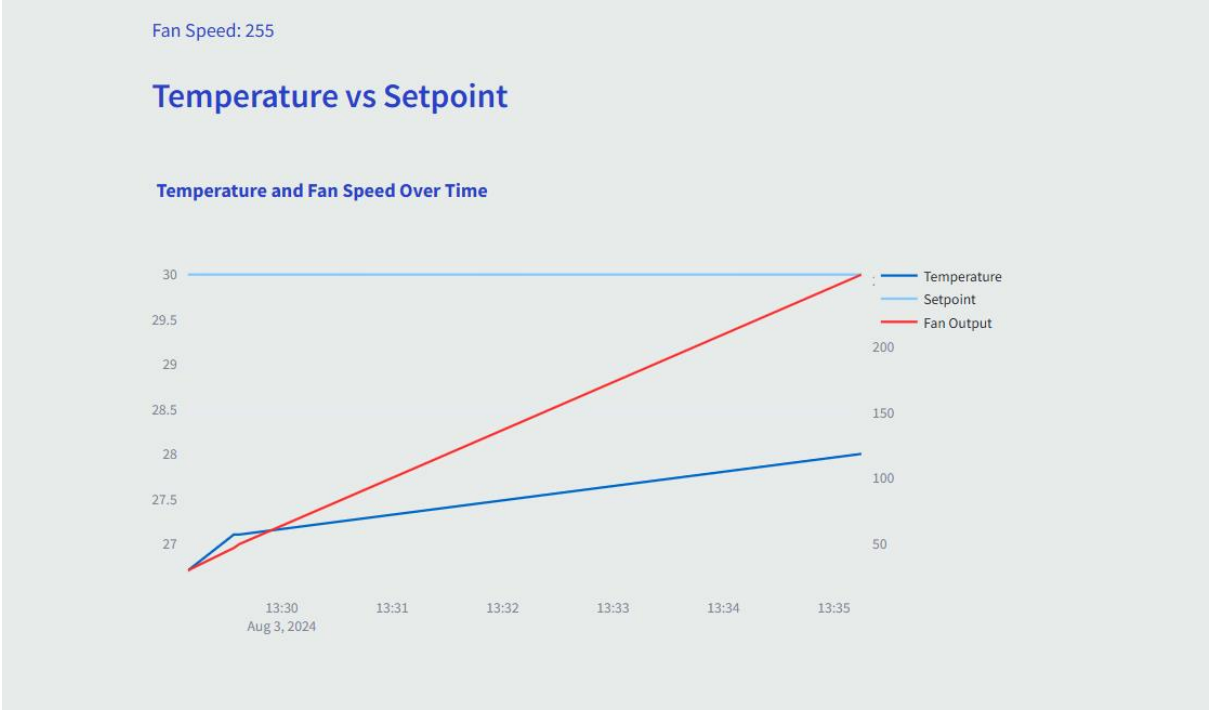

Figure 38 Graph showing how PID affects fan output speed image from the greenhouse UI app.

5.2.2 EXPERIMENT 2: LIGHT INTENSITY ADJUSTMENT

Threshold control is a basic technique where the activation of the light source depends on whether the measured light intensity exceeds or falls below a specific threshold.
Application of the Theory to Adjustment of Light Intensity
When adjusting light intensity with the BH1750 sensor and on/off mechanism, the procedure is akin to the following:
Establishing the Reference Point: Select a preferred light intensity reference point, for instance, 5000 lux. Establishing upper and lower thresholds around this reference point, such as 1000 lux and 10000 lux.
Monitoring Light Intensity: Consistently monitor the light intensity detected by the BH1750 sensor.
On/Off Mechanism Logic: In case the light intensity drops under the lower threshold (1000 lux), activate the light source. If the light intensity exceeds the upper threshold (10000 lux), deactivate the light source.
Projected Graph for Light Intensity Adjustment
The blue line indicates the recorded light intensity changes over time. The red dashed lines symbolize the threshold for the light intensity.
Once the system is on the light intensity fluctuate near the reference point, generating a saw tooth pattern as the light source switches on and off to sustain the intensity within the acceptable spectrum.

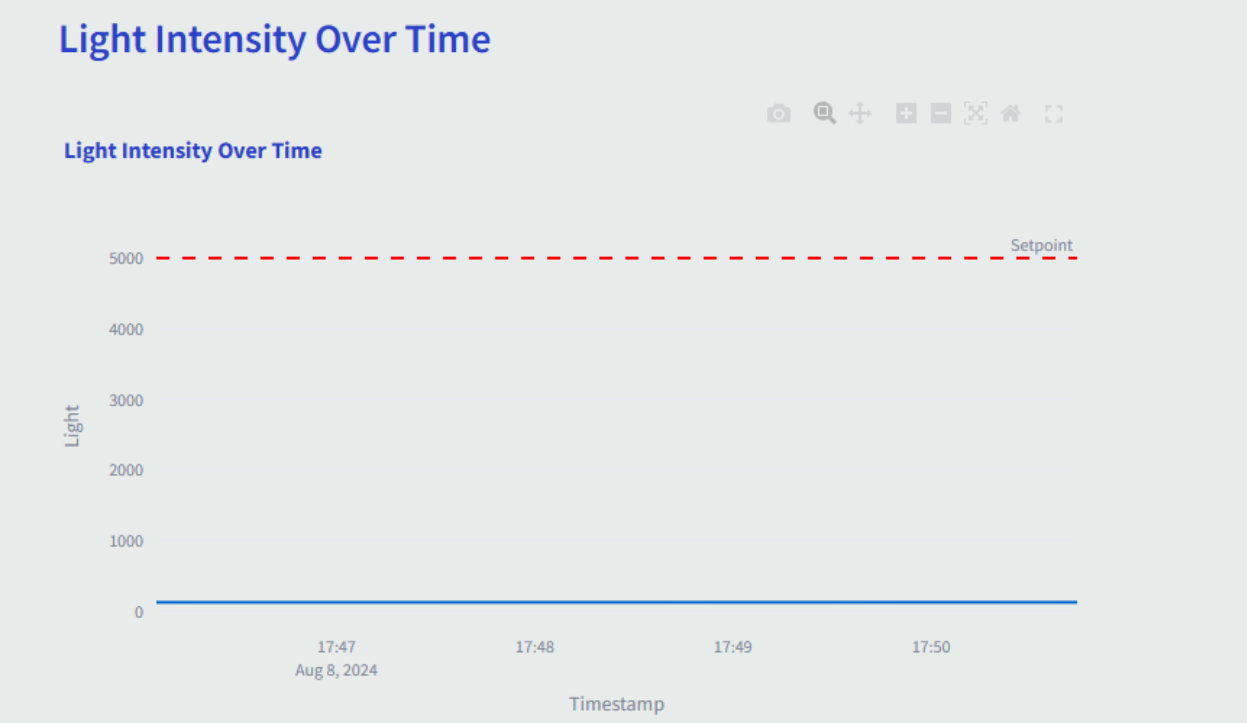

Figure 39 Light intensity over time before system was activated, image from the UI app.

5.2.3 EXPERIMENT 3: SOIL MOISTURE REGULATION

Analysing the Graph of Soil Moisture

The representation of the soil moisture graph is as follows:

The temporal dimension is depicted on the x-axis. The vertical axis illustrates the soil moisture content (presumably in units specific to the sensor). The plotted blue line reflects the recorded soil moisture levels across time. A red dashed line denotes the predetermined threshold or set point for soil moisture. Utilization of On/Off Control Mechanism for Regulating Soil Moisture
The application of on/off control in managing soil moisture entails activating or deactivating the irrigation system (e.g., a water pump) based on whether the soil moisture content surpasses or falls below a specified threshold.
Procedures in Implementing On/Off Control for Soil Moisture Regulation
Establishing the Set Point:
Select a preferred soil moisture set point, for instance, 1400 units.
Monitoring Soil Moisture Levels:
Regularly monitor the soil moisture content using the moisture sensor.
Application of On/Off Control Strategy:
In this case the soil moisture level descends below the lower threshold, activate the irrigation system. Conversely, if the soil moisture content rises above the upper threshold, deactivate the irrigation system. Anticipated Patterns in the Graph Blue Line (Recorded Soil Moisture): Variations in this line will be observed as the irrigation system toggles on and off. Activation of the irrigation system when the soil moisture dips below the lower threshold results in an increase in moisture content. Conversely, deactivation of the system when the moisture surpasses the upper threshold leads to a decrease in moisture content. Red Dashed Line (Set Point): This line signifies the target soil moisture level.

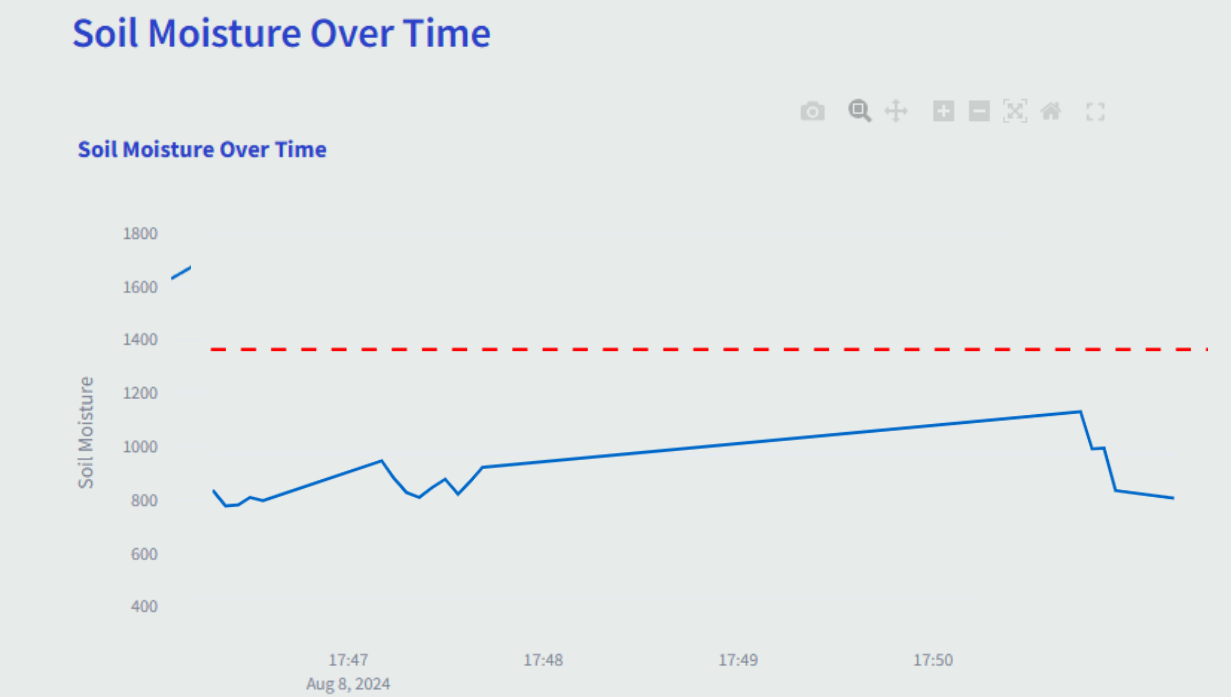

Figure 40 Soil Moisture over time, image from the UI app.

5.2.4 EXPERIMENT 4: HUMIDITY REGULATION

In the realm of On/Off Control, activation of the humidifier occurs when the humidity descends below a specific threshold value, for instance, 55%. Following this, the humidifier remains operational until the humidity attains or surpasses the designated set point of around 60%, prompting its deactivation. In contrast, Hysteresis Control aims to mitigate frequent switching by ensuring that the humidifier remains inactive until the humidity dips slightly below the threshold, approximately 55%, and remains active until the humidity slightly surpasses the set point, typically 60%.

System Response:

Throughout the operation of the humidifier, there is a gradual escalation in humidity levels. This phenomenon is visually depicted in the graph by the gradual ascent of the blue line. It is probable that the system functions in repetitive patterns, wherein the humidifier switches on during instances of low humidity and off upon achieving the desired humidity level. Nevertheless, the graph's indication of minimal humidity change suggests that the humidifier is either in its nascent stage of operation or is encountering limitations in its efficacy due to prevailing environmental conditions.

Steady State:

The ultimate objective is to attain a state of equilibrium where the humidity consistently revolves around the predefined set point with minimal oscillation. Nonetheless, challenges may arise in situations where achieving the 60% humidity target proves arduous due to factors such as heightened ventilation or low external humidity levels. Under such circumstances, the system will persist in its efforts to elevate the humidity until it either attains the set point or stabilizes in close proximity to it.

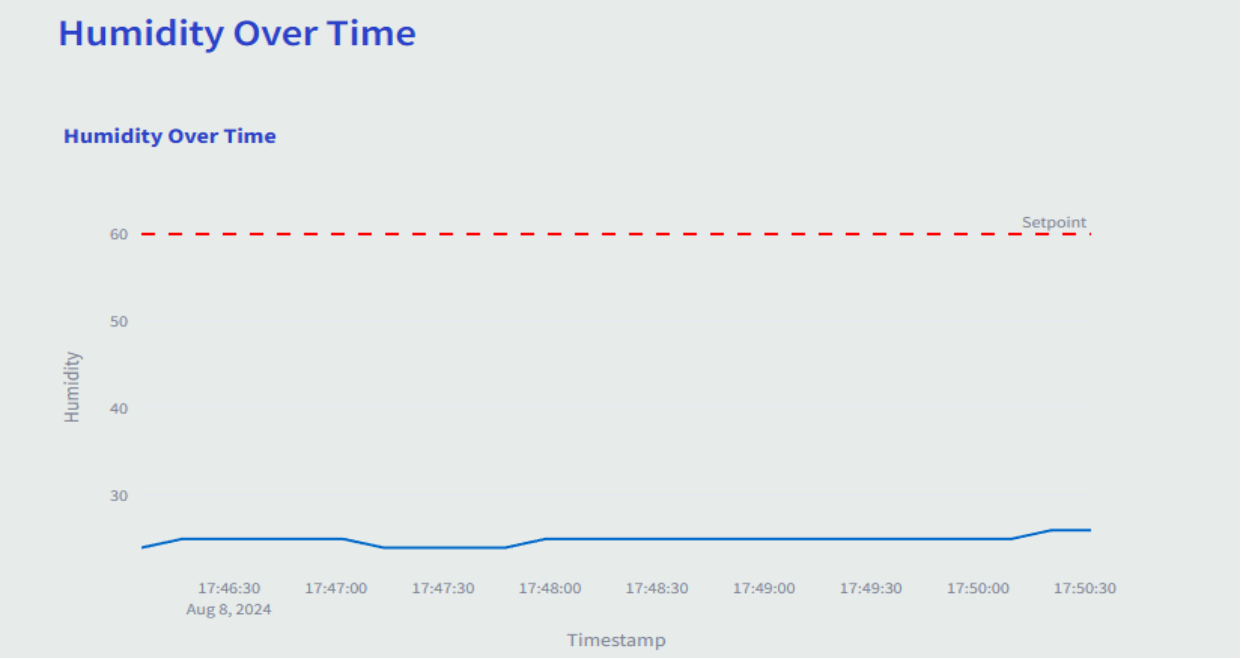

Figure 41 Humidity over time, image from the UI app.

6. CHAPTER 6

6.1 CONCLUSIONS

In conclusion, the creation of a greenhouse monitoring and control system utilizing the Internet of Things (IoT) signifies a significant advancement in improving agricultural practices through technological advancements. By employing a variety of sensors, microcontrollers, communication modules, actuators, and sustainable energy sources, the system offers a holistic approach to improving plant growth conditions while minimizing energy usage and environmental impact. This knowledge enables operators to make informed choices and take prompt actions to ensure optimal plant health and productivity.

The use of cloud-based platforms for remote monitoring and control enables individuals to access real-time data and manage operations from any location, thereby enhancing operational efficiency and enabling proactive management of greenhouse activities, ultimately leading to increased crop yields and profitability.

The Internet of Things (IoT) emerges as a captivating technology with immense potential for greenhouse agriculture, providing diverse technical solutions for object recognition, data gathering, storage, processing, and transmission in physical settings and the convergence of physical and virtual domains. IoT facilitates the shift from conventional greenhouses to intelligent systems, including smart greenhouses equipped to monitor internal conditions and communicate with farmers for automated decision-making to safeguard crops and enhance production. Smart greenhouses supported by IoT incorporate a wide array of devices integrated with sensors and actuators that communicate and interact through the internet, fostering deeper integration between the physical and digital domains.

By adjusting the proportional, integral, and derivative parameters, the system can minimize overshoot, diminish steady-state error, and enhance response times of actuators like humidifiers, fans, and irrigation systems. This precise regulation guarantees that the greenhouse environment remains within the optimal range for plant growth, thus optimizing crop yield and quality while reducing resource wastage.

6.2 RECOMMENDATIONS

Based on the results and findings of this project, the following recommendations are made to enhance the performance, accuracy, and reliability of the greenhouse automation system:

6.2.1 OPTIMAL PLACEMENT OF SENSORS FOR ACCURATE ENVIRONMENTAL MONITORING

Multiple Temperature Sensors: Instead of relying on a single temperature sensor to monitor the entire greenhouse, it is recommended to place multiple temperature sensors at different locations and heights within the greenhouse. This will ensure more accurate and representative temperature readings, as temperature can vary significantly across different areas, especially in larger greenhouses.

Strategic Placement of Humidity Sensors: Humidity levels can also vary depending on the location within the greenhouse. Placing humidity sensors in different zones, including near the plants, at the center of the greenhouse, and near the exhaust fans, will provide a more comprehensive understanding of the humidity distribution, leading to better control and management.

Soil Moisture Sensor Deployment: For better soil moisture management, multiple soil moisture sensors should be deployed at various depths and locations in the soil. This will help in obtaining a more accurate assessment of the soil moisture levels across the entire growing area, preventing over- or under-watering of the plants.

6.2.2 REDUNDANCY AND BACKUP FOR CRITICAL SENSORS

Redundant Sensors: To increase the reliability of the system, especially for critical parameters like temperature, humidity, and soil moisture, it is recommended to include redundant sensors. This ensures that if one sensor fails, the system can continue to operate effectively using data from the backup sensor, reducing the risk of system failure.

Regular Calibration: Ensure that all sensors are regularly calibrated to maintain accuracy. Sensor drift over time can lead to inaccurate readings, which could negatively impact the greenhouse environment. Implementing a routine calibration schedule will help maintain optimal sensor performance.

6.2.3 ENHANCED DATA PROCESSING AND INTEGRATION

Sensor Data Fusion: To achieve more precise control, integrate data from multiple sensors using data fusion techniques. By combining data from temperature, humidity, and soil moisture sensors, the system can make more informed decisions, improving the accuracy of environmental control actions.

Real-Time Data Monitoring: Implement real-time data monitoring and alert systems that notify operators immediately if any sensor readings fall outside of the desired range. This will enable quicker response times to correct any environmental issues, thereby protecting the plants.

6.2.4 INCREASED USE OF AUTOMATION FOR IMPROVED EFFICIENCY

Automated Climate Zones: Consider dividing the greenhouse into separate climate zones, each with its own set of sensors and controls. This would allow for more targeted environmental control, catering to the specific needs of different plant types or growth stages within the same greenhouse.

Automated Irrigation Based on Soil Moisture: Rather than using a fixed schedule for irrigation, automate the watering system based on real-time soil moisture data. This approach ensures that plants receive the right amount of water when needed, promoting healthier growth and reducing water waste.

These recommendations are aimed at optimizing the greenhouse automation system, improving the accuracy and reliability of environmental monitoring, and ensuring the best possible growing conditions for the plants. By implementing these recommendations, the greenhouse can achieve higher efficiency, better crop yields, and more sustainable operation.

6.3 PROPOSED FUTURE WORKS

To further enhance the capabilities of an IoT-based greenhouse monitoring and control system, several things can be made to incorporate advanced technologies such as Artificial Intelligence (AI), computer vision, and pH monitoring for disease detection. These steps aim to optimize greenhouse management, improve crop yield, and reduce the environmental impact of agricultural practices.

6.3.1 INTEGRATION OF AI FOR PREDICTIVE ANALYTICS.

Predictive Analytics: AI can be leveraged to analyze historical and real-time data from various sensors, allowing for predictive modeling of environmental conditions. For instance, machine learning algorithms can predict future temperature, humidity, or soil moisture levels based on current trends, enabling the system to make proactive adjustments to maintain optimal conditions.

Autonomous Control: AI-driven algorithms can automate decision-making processes, reducing the need for human intervention. By analyzing sensor data and considering factors such as weather forecasts, crop growth stages, and resource availability, AI can autonomously adjust greenhouse conditions, optimize resource usage, and respond to potential threats such as pest infestations or diseases.

Figure 42 Use cases of predictive analytics with ML, AI generated image by AI.

6.3.2 UTILIZATION OF COMPUTER VISION FOR CROP HEALTH MONITORING

Disease Detection: Integrating computer vision technology into the greenhouse system can enhance the detection of plant diseases at an early stage. High-resolution cameras and image processing algorithms can monitor the leaves, stems, and fruits of plants, identifying visual symptoms of diseases, pests, or nutrient deficiencies.

Growth Monitoring: Computer vision can also be used to track the growth and development of crops over time. By analyzing images, the system can measure parameters such as leaf area, plant height, and fruit size, providing valuable insights into the growth rate and health of the plants. This information can be used to optimize irrigation, fertilization, and other management practices.

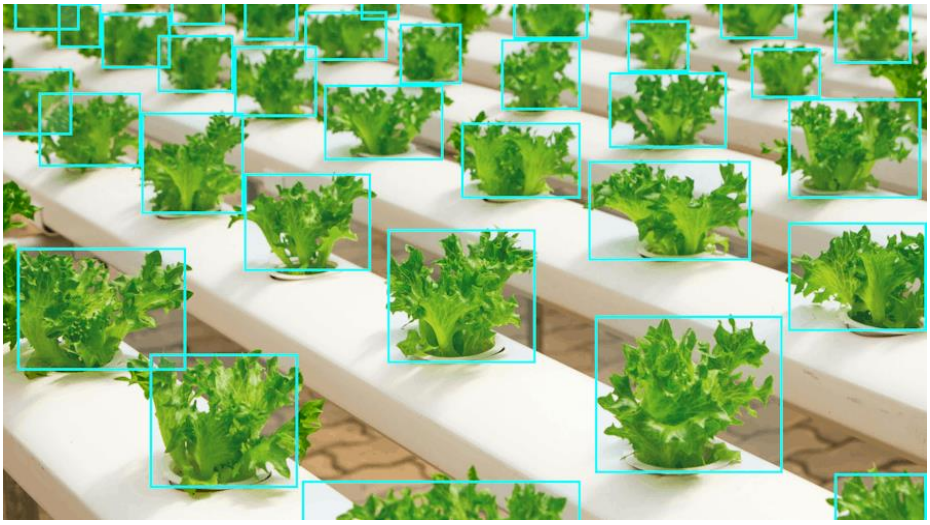

Figure 43 Computer Vision Application in Agriculture, image by Encord.

6.3.3 INCORPORATING PH MONITORING FOR SOIL AND WATER QUALITY MANAGEMENT

Soil pH Monitoring: Continuous monitoring of soil pH is crucial for maintaining the health of plants, as pH levels directly affect nutrient availability. Sensors can be integrated into the system to measure soil pH in real-time, and AI algorithms can recommend or automatically adjust the application of lime or sulfur to balance the pH levels.

Water Quality Control: pH sensors can also be used to monitor the pH of irrigation water. Maintaining the correct pH in water is essential to prevent damage to plants and ensure the efficient uptake of nutrients. The system can automatically adjust the pH of the water by adding appropriate chemicals, ensuring optimal conditions for plant growth.

Figure 44 Soil Monitoring with IoT - Smart Agriculture, image by Manx Technology Group.

6.3.4 DEVELOPMENT OF AN AI-POWERED DISEASE DETECTION SYSTEM

Multi-Sensor Integration: Combine data from pH sensors, computer vision, and environmental sensors (e.g., temperature, humidity) to create a comprehensive disease detection system. AI algorithms can analyze this data to detect anomalies and identify potential disease outbreaks before they become widespread.

Real-Time Alerts and Recommendations: When the system detects signs of disease, it can send real-time alerts to greenhouse operators via a cloud-based platform. The AI system can also provide recommendations for treatment, such as the application of specific pesticides or changes in environmental conditions to mitigate the spread of the disease.

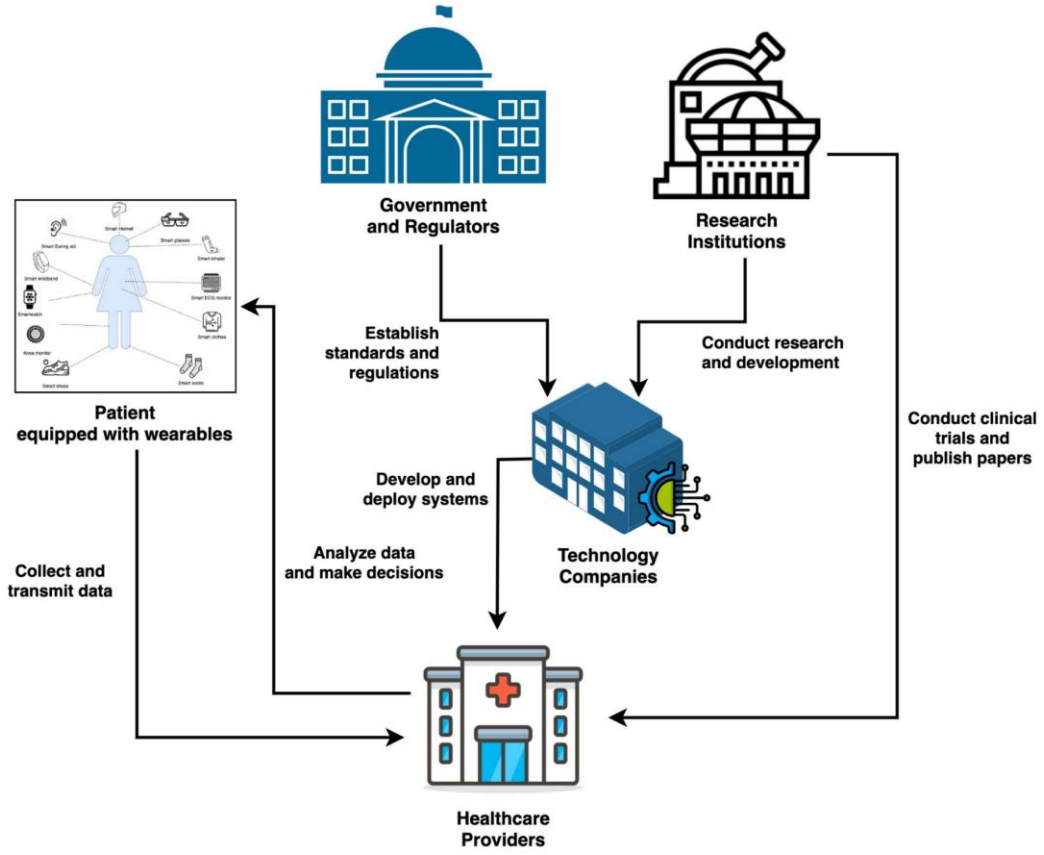

Figure 45 Edge AI-based architecture for early health prediction, image by College of Information Technology.

References

[1] P. Keshavrao Mudholkar *et al.*, "Design Engineering Environmental Monitoring System Based on Internet of Things Design Engineering Environmental Monitoring System Based on Internet of Things Design Engineering," 2021, [Online]. Available: https://www.researchgate.net/publication/372562620

[2] B. P. Mulenga, "Zambia Agriculture Status Report." [Online]. Available: https://www.researchgate.net/publication/363136795

[3] "AgricultureStatusReport2020".

[4] H. Ngoma, B. P. Mulenga, and M. Subakanya, "Climate-Smart Agriculture, Cropland Expansion, and Deforestation in Zambia: Linkages, Processes, and Drivers." [Online]. Available: https://www.researchgate.net/publication/338335441

[5] "Agri_Status_Report_019_12_12_019-C".

[6] B. P. Mulenga, "Climate Change and Agriculture in Zambia: Impacts, Adaptation and Mitigation options Hambulo Ngoma International Maize and Wheat Improvement Center", doi: 10.13140/RG.2.1.3738.0240.

[7] R. Rayhana, G. Xiao, and Z. Liu, "Internet of Things Empowered Smart Greenhouse Farming," *IEEE Journal of Radio Frequency Identification*, vol. 4, no. 3, pp. 195–211, Sep. 2020, doi: 10.1109/JRFID.2020.2984391.

[8] "Green_House_Drawing".

[9] H. Ibrahim *et al.*, "A layered IoT architecture for greenhouse monitoring and remote control," *SN Appl Sci*, vol. 1, no. 3, Mar. 2019, doi: 10.1007/s42452-019-0227-8.

[10] A. Simo, S. Dzitac, G. E. Badea, and D. Meianu, "Smart Agriculture: IoT-based Greenhouse Monitoring System," *International Journal of Computers, Communications and Control*, vol. 17, no. 6, 2022, doi: 10.15837/ijccc.2022.6.5039.

[11] J. Yang, A. Sharma, and R. Kumar, "IoT-based framework for smart agriculture," *International Journal of Agricultural and Environmental Information Systems*, vol. 12, no. 2, pp. 1–14, Apr. 2021, doi: 10.4018/IJAEIS.20210401.oa1.

[12] C. Bersani, C. Ruggiero, R. Sacile, A. Soussi, and E. Zero, "Internet of Things Approaches for Monitoring and Control of Smart Greenhouses in Industry 4.0," May 01, 2022, *MDPI*. doi: 10.3390/en15103834.

[13] A. Sagheer, M. Mohammed, K. Riad, and M. Alhajhoj, "A cloud-based IoT platform for precision control of soilless greenhouse cultivation," *Sensors (Switzerland)*, vol. 21, no. 1, pp. 1–29, Jan. 2021, doi: 10.3390/s21010223.

[14] S. D. Nath, M. Shahadat Hossain, I. A. Chowdhury, S. Tasneem, M. Hasan, and R. Chakma, "Design and Implementation of an IoT Based Greenhouse Monitoring and Controlling System," *Journal of Computer Science and Technology Studies*, doi: 10.32996/jcsts.

[15] S. D. Nath, M. Shahadat Hossain, I. A. Chowdhury, S. Tasneem, M. Hasan, and R. Chakma, "Design and Implementation of an IoT Based Greenhouse Monitoring and Controlling System," *Journal of Computer Science and Technology Studies*, doi: 10.32996/jcsts.

[16] T. G. Alexandru and C. Pupaza, "The development of PID temperature controllers based on FEM thermal analysis," *MATEC Web of Conferences*, vol. 342, p. 06008, 2021, doi: 10.1051/matecconf/202134206008.

[17] A. Abdulkadir, M. A. Gadam, and M. N. Danladi, "Design and Implementation of An Arduino-Based Greenhouse Monitoring System Using IoT," 2022. [Online]. Available: https://www.researchgate.net/publication/361495279

[18] R. Rayhana, G. G. Xiao, and Z. Liu, "Printed Sensor Technologies for Monitoring Applications in Smart Farming: A Review," 2021, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/TIM.2021.3112234.

[19] J. Yang, A. Sharma, and R. Kumar, "IoT-based framework for smart agriculture," *International Journal of Agricultural and Environmental Information Systems*, vol. 12, no. 2, pp. 1–14, Apr. 2021, doi: 10.4018/IJAEIS.20210401.oa1.

[20] "Programmable Logic Controllers." "A Research Paper on Internet of Things based upon Smart Homes with Security Risk Assessment using OCTAVE Allegro."
[21] I. Fuat and M. Can, "Wireless Sensor Networks." "Advances and Challenges in the IOT Routing Protocols: A Comprehensive Review," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp.
[22] Katsuhiko. Ogata, *Modern control engineering*. Prentice-Hall, 2010.
[23] A. Sharma, S. Sharma, and D. Gupta, "A Review of Sensors and Their Application in Internet of Things (IOT)," *Int J Comput Appl*, vol. 174, no. 24, pp. 27–34, Mar. 2021, doi: 10.5120/ijca2021921148.
[24] "iotsensorpov-sensortypes".
[25] J. Salazar and S. Silvestre, *INTERNET OF THINGS*. [Online]. Available: http://www.techpedia.eu
[26] A. Bilal Zia and K. Chauhan, "A Research Paper on Internet of Things based upon Smart Homes with Security Risk Assessment using OCTAVE Allegro." [Online]. Available: www.ijert.org
[27] A. Sharma and S. Sharma, "Advances and Challenges in the IOT Routing Protocols: A Comprehensive Review," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 9350–9371, 2020, [Online]. Available: https://www.researchgate.net/publication/353511365
[28] S.-Iong. Ao, Oscar. Castillo, Craig. Douglas, D. Dagan. Feng, and Alexander. Korsunsky, *International MultiConference of Engineers and Computer Scientists : IMECS 2016 : 16-18 March, 2016, the Royal Garden Hotel, Kowloon, Hong Kong*. Newswood Limited, International Association of Engineers, 2016.
[29] A. Sharma and S. Sharma, "Advances and Challenges in the IOT Routing Protocols: A Comprehensive Review," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 9350–9371, 2020, [Online]. Available: https://www.researchgate.net/publication/353511365
[30] A. Sharma, S. Sharma, and D. Gupta, "A Review of Sensors and Their Application in Internet of Things (IOT)," *Int J Comput Appl*, vol. 174, no. 24, pp. 27–34, Mar. 2021, doi: 10.5120/ijca2021921148.
[31] E. Badidi, "Edge AI for Early Detection of Chronic Diseases and the Spread of Infectious Diseases: Opportunities, Challenges, and Future Directions," Nov. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/fi15110370.
[32] D. Sehrawat and N. S. Gill, "Smart sensors: Analysis of different types of IoT sensors," in *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, Institute of Electrical and Electronics Engineers Inc., Apr. 2019, pp. 523–528. doi: 10.1109/ICOEI.2019.8862778.
[33] "What the Internet of Things (IoT) Needs to Become a Reality."
[34] P. K. Tripathy, A. K. Tripathy, A. Agarwal, and S. P. Mohanty, "MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture," *IEEE Consumer Electronics Magazine*, vol. 10, no. 4, pp. 57–62, Jul. 2021, doi: 10.1109/MCE.2021.3055930.
[35] Katsuhiko. Ogata, *Modern control engineering*. Prentice-Hall, 2010.
[36] M. F. , K. B. . GOLNARAGHI, *Automatic control systems*. Wiley-Blackwell, 3009.
[37] V. Sundara Mahalingam, "SIMULATION OF TEMPERATURE CONTROLLED FAN USING MATLAB," JETIR, 2018. [Online]. Available: www.jetir.org932
[38] "Apago PDF Enhancer SOLUTION MANUAL 1-17 Solutions to Problems," 2011. [Online]. Available: http://www.wiley.com/go/permissions.
[39] R. K. Kodali, V. Jain, and S. Karagwal, "IoT based smart greenhouse," in *IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Apr. 2017. doi: 10.1109/R10-HTC.2016.7906846.
[40] N. S. Nise, "Antenna Azimuth P< Control System ion."
[41] "Power Electronics, Drives, and Advanced Applications."
[42] M. H. Rashid, "THIRD EDITION."
[43] M. H. . Rashid, *Power electronics : devices, circuits, and applications*. Pearson, 2014.
[44] V. Kumar, R. Kumar Behera, D. Joshi, and R. Bansal, "Power Electronics, Drives, and Advanced Applications."
[45] Katsuhiko. Ogata, *Modern control engineering*. Prentice-Hall, 2010.

[46] S. D. Nath, M. Shahadat Hossain, I. A. Chowdhury, S. Tasneem, M. Hasan, and R. Chakma, "Design and Implementation of an IoT Based Greenhouse Monitoring and Controlling System," *Journal of Computer Science and Technology Studies*, doi: 10.32996/jcsts.
[47] R. K. Kodali, V. Jain, and S. Karagwal, "IoT based smart greenhouse," in *IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016*.
[48] V. Muthu Lakshmi, P. Joshi, and A. Professor, "Issue 3 www.jetir.org(ISSN-2349-5162)," JETIR, 2024. [Online]. Available: www.jetir.orgf11
[49] "Modern Control Systems ThirTeenTh ediTion."
[50] N. S. . Nise, *Control systems engineering*. Wiley, 2015.
[51] K. Sohraby, D. Minoli, and T. Znati, "WIRELESS SENSOR NETWORKS Technology, Protocols, and Applications."
[52] "Bolton - Mechatronics_ Electronic Control Systems in Mechanical and Electrical Engineering 6th Edition c2015 txtbk".

APPENDICES

APPENDIX 1: MODEL MODELLINIG

1. GREENHOUSE MODELLING

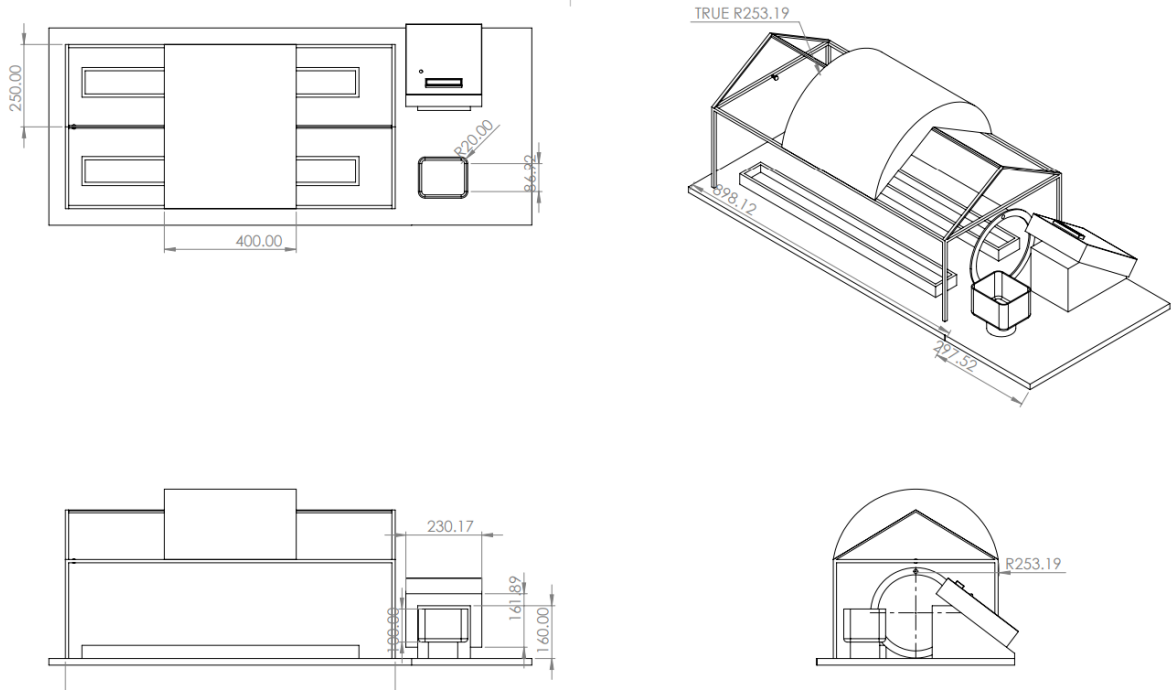

Figure 46 CAD drawing for the greenhouse, image from Solidworks

2. USER INTERFACE DESIGN

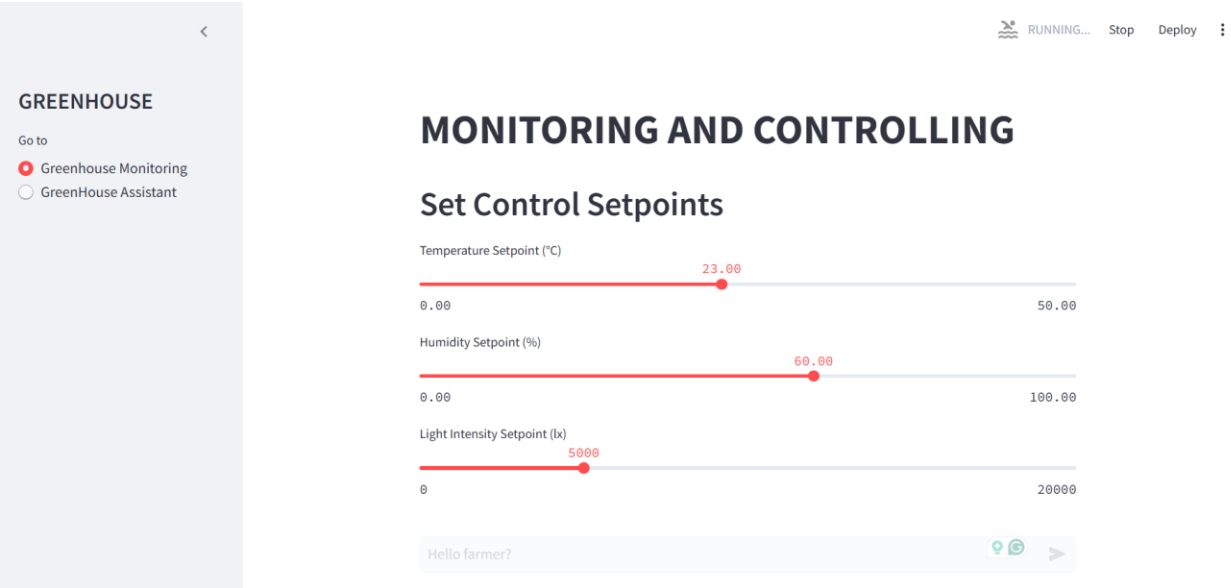

Figure 47 User interface for the greenhouse monitoring system, image from the UI app

3. AI CHAT BOT

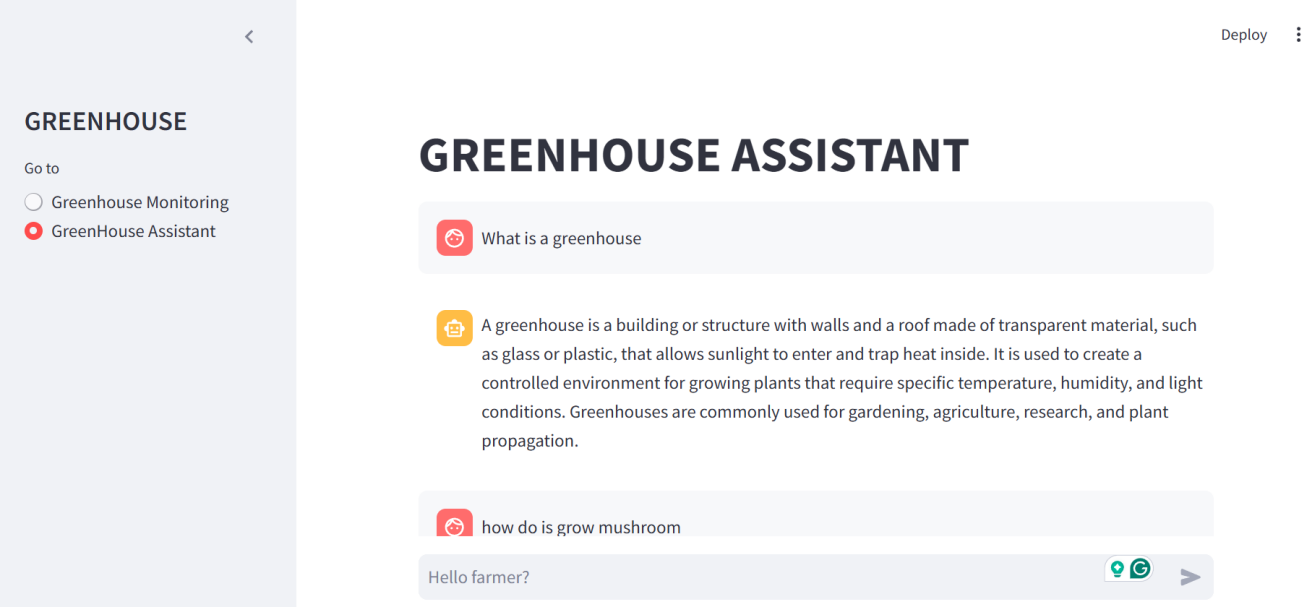

Figure 48 AI Chatbot

APPENDIX 2: TABLE OF FORMULARS

Table 2 Table of formulars

Category	Formula	Description	Variables
1. Power Calculations	$P = V \times I$	Calculates the electrical power consumed by a device.	P = Power (Watts), V = Voltage (Volts), I = Current (Amperes)
	$I = \frac{P}{V}$	Determines the current drawn by a device based on power and voltage.	I = Current (Amperes), P = Power (Watts), V = Voltage (Volts)
	$P_{total} = \sum(P_{device})$	Calculates the total power requirement for all devices in the system.	P_{total} = Total Power (Watts), P_{device} = Power of each device (Watts)
2. Heater Sizing	$Q = m \times c \times \Delta T$	Determines the energy required to heat a space to a desired temperature.	Q = Energy (Joules), m = Mass of air (kg), c = Specific heat capacity (J/kg°C), ΔT = Temperature change (°C)
	$P_{heater} = \frac{Q}{t}$	Determines the power rating of the heater required to achieve the temperature rise in a specific time.	P_{heater} = Heater Power (Watts), Q = Energy (Joules), t = Time (seconds)

3. Pump Sizing	$P_{pump} = \frac{\rho \times g \times h \times Q}{\eta}$	Calculates the power requirement for a pump based on flow rate and head.	P_{pump} = Pump Power (Watts), ρ = Density of water (kg/m^3), g = Gravity (9.81 m/s^2), h = Head (meters), Q = Flow rate (m^3/s), η = Efficiency of pump
	$Q = \frac{V_{water}}{t}$	Determines the flow rate needed to achieve a certain water volume over time.	Q = Flow Rate (m^3/s), V_{water} = Volume of water (m^3), t = Time (seconds)
4. LED Lighting	$E = \frac{P_{light}}{A}$	Calculates the illuminance (lux) provided by an LED strip over a given area.	E = Illuminance (lux), P_{light} = Light Power (Watts), A = Area (m^2)
	$P_{LED} = L \times P_{LED/m}$	Determines the total power consumption of an LED strip.	P_{LED} = Total LED Power (Watts), L = Length of LED strip (meters), $P_{LED/m}$ = Power per meter of LED strip (Watts/m)
5. Motor Sizing	$P_{motor} = \frac{T \times \omega}{\eta}$	Determines the power requirement for a DC motor based on torque and angular velocity.	P_{motor} = Motor Power (Watts), T = Torque (Nm), ω = Angular velocity (rad/s), η = Efficiency
	$T = F \times r$	Calculates the torque required based on force and radius.	T = Torque (Nm), F = Force (N), r = Radius (m)
	$\omega = \frac{2 \times \pi \times N}{60}$	Converts rotational speed from RPM to rad/s.	ω = Angular velocity (rad/s), N = Rotational speed (RPM)
6. Sensor Calibration	$V_{out} = \frac{R_2}{R_1 + R_2} \times V_{in}$	Voltage divider formula for calibrating sensors.	V_{out} = Output Voltage (V), R_1, R_2 = Resistors (Ω), V_{in} = Input Voltage (V)
	$R = \frac{V_{supply} \times V_{out}}{V_{supply} - V_{out}}$	Determines the resistance of a sensor based on output voltage and supply voltage.	R = Resistance (Ω), V_{supply} = Supply Voltage (V), V_{out} = Output Voltage (V)
7. PID Control	$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$	PID controller formula for calculating control output.	$u(t)$ = Control Output, K_p = Proportional Gain, K_i = Integral Gain, K_d = Derivative Gain, $e(t)$ = Error Signal

	$e(t) = r(t) - y(t)$	Calculates the error signal in PID control based on reference and actual output.	$e(t)$ = Error Signal, $r(t)$ = Reference Signal, $y(t)$ = Actual Output
	$T_s \approx \frac{4}{\zeta \omega_n}$	Estimation of settling time for a second-order system in PID control.	T_s = Settling Time (seconds), ζ = Damping Ratio, ω_n = Natural Frequency (rad/s)
	$K_c = \frac{K_u}{1.2}, T_i = \frac{T_u}{2}, T_d = \frac{T_u}{8}$	Ziegler-Nichols tuning rules for PID parameters.	K_c = Controller Gain, T_i = Integral Time, T_d = Derivative Time, K_u = Ultimate Gain, T_u = Ultimate Period
8. System Response	$y(t) = K \times (1 - e^{-\frac{t}{\tau}})$	First-order system step response.	$y(t)$ = System Output, K = System Gain, τ = Time Constant, t = Time (seconds)
9. Light Sensing	$I = \frac{V_{out} \times 1000}{R_L \times G}$	Photodiode current calculation for light sensing.	I = Current (mA), V_{out} = Output Voltage (V), R_L = Load Resistance (Ω), G = Photodiode Gain
10. Humidity and Temperature	$RH = \frac{P_{H_2O}}{P_{H_2O,sat}} \times 100$	Relative Humidity calculation based on partial pressures.	RH = Relative Humidity (%), P_{H_2O} = Partial Pressure of Water Vapor, $P_{H_2O,sat}$ = Saturation Pressure
	$T_C = \frac{T_F - 32}{1.8}$	Converts temperature from Fahrenheit to Celsius.	T_C = Temperature (°C), T_F = Temperature (°F)
11. Soil Moisture	$V_{out} = V_{max} \times \frac{M_{soil} - M_{min}}{M_{max} - M_{min}}$	Soil moisture sensor output voltage calculation based on moisture content.	V_{out} = Output Voltage (V), V_{max} = Max Output Voltage (V), M_{soil} = Soil Moisture Level, M_{min}, M_{max} = Min/Max Moisture Levels

APPENDIX 3: COMPONENTS DATASHEET

Component	Description	Specifications	Manufacturer/Part Number	Additional Information

Table 3 Datasheet

ESP32	A powerful microcontroller with built-in Wi-Fi and Bluetooth, widely used for IoT applications. Supports multiple GPIOs, ADCs, DACs, and PWM.	240 MHz dual-core Xtensa® LX6 microprocessor, 520 KB SRAM, Wi-Fi 802.11 b/g/n, Bluetooth 4.2, 34 GPIO pins, 2.2V-3.6V input	Espressif/ESP32-WROOM-32	Can be programmed using Arduino IDE or MicroPython. Ideal for IoT-based greenhouse automation projects due to its connectivity features.
L298 Motor Driver	A dual H-Bridge motor driver that allows control of two DC motors or a stepper motor. Suitable for controlling the speed and direction of motors in automation systems.	Operating Voltage: 5V-35V, Max Output Current: 2A per channel, Logic Voltage: 5V, Control: TTL logic levels, Enable/Disable	STMicroelectronics/L298N	Features built-in diodes for back EMF protection. Typically used for motor control in robotics and automation projects.
Switch	A simple on/off switch used to manually control the power or operational states of electronic circuits.	Type: SPST (Single Pole Single Throw), Current Rating: 3A, Voltage Rating: 250V AC	Various	Available in different types such as toggle, push-button, or slide switches, depending on the specific requirement of the project.
LED	A light-emitting diode that can be used as an indicator or for low-power lighting applications.	Forward Voltage: 2.0-2.2V, Forward Current: 20mA, Color: Red/Green/Blue/Yellow	Various	LEDs are energy-efficient and available in various colors and sizes. Can be used to indicate the status of the system or for basic illumination.

Relay 12V	An electromechanical relay used for switching high voltage or high current loads with low voltage signals from a microcontroller.	Coil Voltage: 12V DC, Contact Rating: 10A 250VAC/10A 30VDC, Contact Type: SPDT (Single Pole Double Throw)	Songle/SRD-12VDC-SL-C	Commonly used for switching devices like heaters, pumps, or lights in greenhouse automation. Provides isolation between control and load circuits.
Small Humidifier	A compact, Arduino-compatible humidifier for maintaining adequate humidity levels in a greenhouse.	Operating Voltage: 5V DC, Power Consumption: 2W, Humidity Output: 50ml/h	Various	Can be controlled using a relay or transistor from the ESP32. Helps in regulating humidity levels, which is crucial for plant health.
Strip of LEDs	A flexible LED strip used for lighting or providing supplemental light to plants in a greenhouse.	Operating Voltage: 12V DC, Power: 4.8W per meter, LED Type: SMD 3528, Number of LEDs: 60 LEDs/m	Various	Can be cut to size and arranged in different patterns. Useful for providing additional light to plants or for aesthetic purposes.
Heater	A heating element used to maintain the required temperature in the greenhouse environment.	Power: 100W, Operating Voltage: 220V AC, Type: PTC (Positive Temperature Coefficient)	Various	Controlled via relay for automatic temperature regulation. Helps maintain optimal temperature conditions for plant growth.
Connecting Wires	Electrical wires used for connecting various electronic components. Essential for setting up circuits on	Diameter: 1mm (0.001m), Length: 4m, Material: Copper, Insulation: PVC	Various	Available in different colors for easy identification of connections. Typically used for low-power, low-current applications.

	breadboards or connecting sensors and actuators.			
LCD 16x2	A liquid crystal display module used for displaying data such as sensor readings, system status, or user instructions.	16 characters x 2 lines, Operating Voltage: 5V, Interface: Parallel (16-pin), Backlight: LED	Hitachi/HD44780	Can be interfaced with the ESP32 using an I2C adapter for reducing the number of GPIO pins used.
Breadboard	A solderless prototyping board for constructing and testing electronic circuits.	Size: 830 tie points, Power Rails: 2, Material: ABS plastic, Dimensions: 165mm x 55mm x 9mm	Various	Allows for quick and easy changes to circuits without the need for soldering. Essential for prototyping and testing before final implementation.
ESP32 Shield	An expansion board designed to simplify the connection of peripherals to the ESP32 microcontroller.	Compatible with ESP32, Breakout pins for GPIOs, Power Supply: 3.3V/5V, Additional Features: I2C, UART, SPI connectors	Various	Provides easy access to the pins of the ESP32 and includes connectors for external modules like sensors, displays, or relays.
2 Blade Propeller	A propeller attached to a DC motor used for air circulation or ventilation within the greenhouse.	Diameter: 100mm, Material: Plastic, Mounting Hole Diameter: 3mm	Various	Ensures proper air circulation to maintain even temperature and humidity levels. Essential for preventing mold and ensuring healthy plant growth.
DC Motor	A small motor used to drive mechanical components such as propellers or pumps.	Operating Voltage: 6-12V, Speed: 3000 RPM, Current: 0.5A	Various	Commonly used in automation systems for driving fans, pumps, or other mechanical

				parts. Can be controlled via PWM for speed regulation.
Diodes	Semiconductor devices that allow current to flow in one direction, used for rectification or protection against reverse voltage.	Forward Voltage: 0.7V, Forward Current: 1A, Type: Silicon, Package: DO-41	1N4001	Protects circuits from reverse polarity or used in power supplies for converting AC to DC. Essential for safeguarding components.
Resistors	Passive components used to limit current, divide voltage, or pull up/pull down signals in circuits.	Resistance: 1kΩ, Power Rating: 0.25W, Tolerance: ±5%, Material: Carbon Film, Package: Axial	Various	Available in different resistance values. Fundamental component in almost all electronic circuits.
Capacitors	Components used for filtering, decoupling, or energy storage in electronic circuits.	Capacitance: 100µF, Voltage Rating: 25V, Type: Electrolytic, Package: Radial	Various	Used for smoothing out voltage fluctuations in power supplies or for timing applications in RC circuits.
Power Cable	A cable used for delivering AC power from an outlet to the system.	Length: 1.5m, Type: 3-core (Live, Neutral, Earth), Rated Voltage: 220V AC, Material: Copper	Various	Essential for powering the control system and connected devices. Typically includes a plug on one end and stripped wires on the other.
Power Cable Socket	A socket used for connecting the power cable to the power supply or device.	Type: IEC320 C13, Rated Voltage: 250V AC, Rated Current: 10A, Material: Thermoplastic	Various	Provides a secure and standardized connection for the power cable. Widely used in power supply units and electrical appliances.

Pump	A small water pump used for irrigation or watering plants in the greenhouse.	Operating Voltage: 12V DC, Max Flow Rate: 240L/h, Max Head: 2m, Current: 0.5A	Various	Can be controlled via relay or MOSFET for automated watering based on soil moisture levels. Important for maintaining proper irrigation.
Tubes	Tubing used for distributing water from the pump to different parts of the greenhouse.	Inner Diameter: 4mm, Outer Diameter: 6mm, Length: 2m, Material: PVC	Various	Flexible and durable, suitable for low-pressure water distribution in irrigation systems.
Jumper Cables	Cables used for making temporary or semi-permanent connections on a breadboard or between modules.	Type: Male-to-Male, Length: 20cm, Insulation: PVC, Conductor: Copper	Various	Color-coded for easy identification, essential for prototyping and testing circuits on breadboards.
BH1750 Sensor	A digital light intensity sensor used to measure ambient light levels, which can be used to adjust lighting in the greenhouse.	Measurement Range: 0-65535 lux, Interface: I2C, Operating Voltage: 3.3-5V, Resolution: 1 lux	ROHM/BH1750FVI	Provides precise light measurement, useful for optimizing plant growth by controlling artificial lighting based on natural light availability.
DHT11 Sensor	A basic temperature and humidity sensor used for monitoring environmental conditions inside the greenhouse.	Temperature Range: 0-50°C, Humidity Range: 20-90% RH, Accuracy: ±2°C (Temp), ±5% (Humidity), Interface: Digital, 1-wire	Aosong Electronics/DHT11	Commonly used in DIY projects, provides basic environmental data for automation systems. Inexpensive and easy to use with microcontrollers.
Moisture Sensor	A sensor used to detect soil	Operating Voltage: 3.3-5V, Output: Analog (0-	Various	Can be used with an ADC or

	moisture levels, critical for automated irrigation systems in the greenhouse.	1023) or Digital (HIGH/LOW), Probes: Copper, Interface: 2-pin/3-pin		digital input of the ESP32. Helps in automating watering based on soil dryness, improving plant health.
Voltage Sensor	A sensor used to monitor the voltage levels in the system, ensuring safe operation of connected devices.	Input Voltage: 0-25V, Output Voltage: 0-5V, Interface: Analog, Ratio: 5:1, Accuracy: ±1%	Various	Converts high input voltage to a lower output voltage suitable for microcontrollers, useful for battery monitoring or power supply checks.

Additional Notes:

- **Compatibility:** All components are selected for compatibility with the ESP32 microcontroller, ensuring ease of integration and reliable operation.
- **Environmental Suitability:** Components like the heater, humidifier, and sensors are chosen to operate effectively in a greenhouse environment, which may have varying temperature and humidity levels.
- **Safety Considerations:** Components such as relays and power sockets are selected with appropriate ratings to handle the expected loads, ensuring the safety and longevity of the system.
- **Flexibility:** Components like breadboards and jumper cables allow for easy modification and testing of the system before final implementation.

APPENDIX 4: BUDGET

Table 4 Budget

Component	Description	Quantity	Unit Cost (ZMW)	Total Cost (ZMW)
Control System Components				
ESP32	Microcontroller with Wi-Fi and Bluetooth	1	450	450
L298 Motor Driver	Dual H-Bridge motor driver	1	220	220
Switch	Basic on/off switch	2	10	20
LED	Light-emitting diode	1	5	5
Relay 12V	Electromechanical switch	4	150	600

Small Humidifier (Arduino Compatible)	For maintaining humidity	1	200	200
Strip of LEDs	Flexible LED strip for lighting	1	100	100
Heater	Heating element for temperature control	1	300	300
0.001m Diameter Connecting Wires	Electrical wires	4 meters	5	20
LCD 16x2	Display module	1	80	80
Breadboard	Solderless board for prototyping	1	120	120
ESP32 Shield	Shield for ESP32 microcontroller	1	130	130
2 Blade Propeller	Propeller for air circulation	1	30	30
DC Motor	Motor for driving the propeller	1	100	100
Diodes	Semiconductor devices	10	1	10
Resistors	Passive components for current control	10	1	10
Capacitors	For filtering and stabilizing the circuit	10	2	20
Power Cable	Cable for supplying power	1	90	90
Power Cable Socket	Socket for power cable connection	1	10	10
Pump	Water pump for irrigation	1	150	150
Tubes	Tubing for water distribution	2 meters	10	20
Jumper Cables	For temporary connections	2 pieces	2.5	5
BH1750 Sensor	Digital light sensor	1	180	180
DHT11 Sensor	Temperature and humidity sensor	1	50	50
Moisture Sensor	Soil moisture monitoring	1	80	80
Voltage Sensor	For monitoring voltage levels	1	30	30
Subtotal				**K3,030.00**
Mini Greenhouse Components				
15x15 Square Tube	Structural tube for greenhouse frame	1	200	200
Transparent Plastic	Covering material for the greenhouse	2 meters^2	50	100
Glue Sticks	Adhesive sticks for assembly	10 pieces	2	20
Glue Gun	Tool for applying glue sticks	1	50	50
Double-Sided Tape	For securing plastic sheeting	1 roll	30	30
Masking Tape	For temporary fastening or labeling	1 roll	10	10
Subtotal				**410**
Total Budget				**k3, 440.00**

This table provides a detailed breakdown of the components required for the mini greenhouse and control system, along with their descriptions, quantities, unit costs, and total costs in Zambian Kwacha (ZMW).

APPENDIX 5: CODE USED

1 PID SYSTEM DESIGN IN MATLAB (Matlab code)

```
%% 1. Develop a Mathematical Model
% Parameters
a = 1; % system parameter
b = 1; % system parameter

% Transfer function G(s)
s = tf('s');
G = b / (s + a);

%% 2. Visualize the Uncompensated Angular Displacement Step Response
figure;
step(G);
title('Uncompensated Step Response');
grid on;

%% 3. Develop the Root Locus, Pole-Zero, Nyquist, and Bode Plots of the Uncompensated System
figure;
subplot(2,2,1);
rlocus(G);
title('Root Locus of Uncompensated System');
grid on;

subplot(2,2,2);
pzmap(G);
title('Pole-Zero Map of Uncompensated System');
grid on;

subplot(2,2,3);
nyquist(G);
title('Nyquist Plot of Uncompensated System');

subplot(2,2,4);
bode(G);
title('Bode Plot of Uncompensated System');
grid on;

%% 4. Design a PID Compensator
% Desired performance
% Settling time reduction by four-fold
desired_ts = 1; % example value
zeta = 1; % zero percentage overshoot
Kp = 10; % example value, adjust as needed
Ki = 5;  % example value, adjust as needed
Kd = 2;  % example value, adjust as needed

```

```
46
47  % PID controller transfer function
48  C = Kp + Ki/s + Kd*s;
49
50  % Compensated system transfer function
51  T_cl = feedback(C*G, 1);
52
53  %% 5. Visualize the Compensated Angular Displacement Step Response
54  figure;
55  step(T_cl);
56  title('Compensated Step Response');
57  grid on;
58
59  %% 6. Develop the Root Locus, Pole-Zero, Nyquist, and Bode Plots of the Compensated System
60  figure;
63  title('Root Locus of Compensated System');
64  grid on;
65
66  subplot(2,2,2);
67  pzmap(T_cl);
68  title('Pole-Zero Map of Compensated System');
69  grid on;
70
71  subplot(2,2,3);
72  nyquist(T_cl);
73  title('Nyquist Plot of Compensated System');
74  grid on;
75
76  subplot(2,2,4);
77  bode(T_cl);
78  title('Bode Plot of Compensated System');
79  grid on;
80
81  %% 7. Develop a Discrete Version of the PID Controller
82  Ts = 0.1; % Sampling time
83  C_discrete = c2d(C, Ts, 'tustin');
84
85  %% 8. Implement a Prototype on an Arduino (Pseudocode Example)
86  % This step involves writing the corresponding Arduino code to implement the PID controller.
87  % Pseudocode for Arduino:
88  % double Kp = 10;
89  % double Ki = 5;
90  % double Kd = 2;
91  % double Ts = 0.1;
92  % double integral = 0;
93  % double previous_error = 0;
94  %
95  % void loop() {
96  %   double setpoint = ...; // desired temperature
97  %   double measured_value = ...; // read from sensor
98  %   double error = setpoint - measured_value;
99  %   integral += error * Ts;
100 %   double derivative = (error - previous_error) / Ts;
101 %   double output = Kp*error + Ki*integral + Kd*derivative;
102 %   // apply output to actuator (e.g., control fan speed)
103 %   previous_error = error;
104 %   delay(Ts * 1000);
105 % }
106
```

```
%% 9. Evaluate the Controller's Efficacy
% Simulate tracking of reference trajectories (rectangular and circular)

% Rectangular trajectory
t = 0:Ts:10; % time vector
T_ref_rect = square(2*pi*0.1*t); % rectangular trajectory

figure;
lsim(T_cl, T_ref_rect, t);
title('Tracking Rectangular Trajectory');
grid on;

% Circular trajectory (approximated by sine wave)
T_ref_circ = sin(2*pi*0.1*t); % circular trajectory

figure;
lsim(T_cl, T_ref_circ, t);
title('Tracking Circular Trajectory');
grid on;
```

2. SENSOR SIGNAL SIMULATION MATLAB CODE (Matlab code)

```
% MATLAB Script to Simulate DHT Sensor, BH1750 Sensor, and Soil Moisture Sensor

% Simulation time
timeStep = 1; % Time step in seconds
totalTime = 60; % Total simulation time in seconds

% Pre-allocate arrays for storing sensor data
timeArray = 0:timeStep:totalTime;
temperatureArray = zeros(1, length(timeArray));
humidityArray = zeros(1, length(timeArray));
lightIntensityArray = zeros(1, length(timeArray));
soilMoistureArray = zeros(1, length(timeArray));

% Simulate the sensor data
for i = 1:length(timeArray)
    % DHT Sensor Simulation (Temperature and Humidity)
    temperatureArray(i) = 20 + 5 * sin(2 * pi * (1/30) * timeArray(i)); % Simulate a sinusoidal temperature variation
    humidityArray(i) = 50 + 10 * sin(2 * pi * (1/45) * timeArray(i));   % Simulate a sinusoidal humidity variation

    % BH1750 Sensor Simulation (Light Intensity)
    lightIntensityArray(i) = 300 + 200 * sin(2 * pi * (1/60) * timeArray(i)); % Simulate a sinusoidal light intensity variation

    % Soil Moisture Sensor Simulation
    soilMoistureArray(i) = 800 + 100 * sin(2 * pi * (1/50) * timeArray(i));   % Simulate a sinusoidal soil moisture variation
end
```

```
26
27  % Plot the simulated data
28  figure;
29  subplot(4,1,1);
30  plot(timeArray, temperatureArray, 'r', 'LineWidth', 1.5);
31  title('Simulated Temperature (°C)');
32  xlabel('Time (s)');
33  ylabel('Temperature (°C)');
34
35  subplot(4,1,2);
36  plot(timeArray, humidityArray, 'b', 'LineWidth', 1.5);
37  title('Simulated Humidity (%)');
38  xlabel('Time (s)');
39  ylabel('Humidity (%)');
40
41  subplot(4,1,3);
42  plot(timeArray, lightIntensityArray, 'g', 'LineWidth', 1.5);
43  title('Simulated Light Intensity (lux)');
44  xlabel('Time (s)');
45  ylabel('Light Intensity (lux)');
46
47  subplot(4,1,4);
48  plot(timeArray, soilMoistureArray, 'm', 'LineWidth', 1.5);
49  title('Simulated Soil Moisture (analog value)');
50  xlabel('Time (s)');
51  ylabel('Soil Moisture (analog value)');
52
52
53  % Display the final readings
54  fprintf('Final Simulated Readings:\n');
55  fprintf('Temperature: %.2f °C\n', temperatureArray(end));
56  fprintf('Humidity: %.2f %%\n', humidityArray(end));
57  fprintf('Light Intensity: %.2f lux\n', lightIntensityArray(end));
58  fprintf('Soil Moisture: %.2f (analog value)\n', soilMoistureArray(end));
59
```

3. MONITORING AND CONTROLLING FOR THE ESP32 (Code in C)

```
#include <WiFi.h>
#include <WebServer.h>
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>

// WiFi credentials
const char* ssid = "--";
const char* password = "--";

// Create a web server on port 80
WebServer server(80);

// I2C pins for LCD
#define LCD_SDA_PIN 21
#define LCD_SCL_PIN 22

// I2C pins for BH1750
#define BH1750_SDA_PIN 19
#define BH1750_SCL_PIN 18

// DHT sensor setup
#define DHTTYPE DHT11
#define DHTPIN 4
DHT dht(DHTPIN, DHTTYPE);

// Relay pin for temperature control
#define HEATER_RELAY_PIN 23
```

```
31
32  // Relay pin for humidity control
33  #define HUMIDIFIER_RELAY_PIN 17
34
35  // Light sensor and relay control
36  TwoWire I2C_BH1750 = TwoWire(1); // Use I2C port 1
37  BH1750 lightMeter;
38
39  // Define light relay pin
40  #define LIGHT_RELAY_PIN 5
41
42  // Soil moisture sensor and pump control
43  const int SOIL_MOISTURE_PIN = 35;
44  #define PUMP_RELAY_PIN 2
45  int soilSensorValue = 0;
46
47  // Fan control pins
48  #define FAN_IN1_PIN 12
49  #define FAN_IN2_PIN 14
50  #define FAN_SPEED_PIN 13
51
52  // Voltage sensor pin
53  const int VOLTAGE_SENSOR_PIN = 34;
54
55  // LCD object
56  hd44780_I2Cexp lcd;
57  const int LCD_COLUMNS = 16;
58  const int LCD_ROWS = 2;
59
60  // Variables for cycling display
61  unsigned long previousMillis = 0;
62  const long interval = 5000; // 5 seconds interval
63  int currentMode = 0;
64
65  // Arrays to store previous readings
66  #define NUM_READINGS 10
67  float temperatureReadings[NUM_READINGS];
68  float humidityReadings[NUM_READINGS];
69  int currentReadingIndex = 0;
70
71  // Dynamic setpoints received from Streamlit
72  float tempSetPoint = 23.0;
73  float humiditySetPoint = 60.0;
74  float lightSetpoint = 5000.0;
75  int soilSetpoint = 300;
76
```

```
// PID variables
double dt, last_time_heater, last_time_fan;
double integral_heater, previous_heater, output_heater = 0;
double integral_fan, previous_fan, output_fan = 0;
double kp_heater = 0.8, ki_heater = 0.20, kd_heater = 0.001;
double kp_fan = 0.8, ki_fan = 0.20, kd_fan = 0.001;

void setup() {
  Serial.begin(115200);
  Serial.println("Setup start");

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(2000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
```

```
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());

  // Initialize DHT sensor
  dht.begin();
  Serial.println("DHT sensor initialized");

  // Initialize relay pin for temperature control
  pinMode(HEATER_RELAY_PIN, OUTPUT);
  digitalWrite(HEATER_RELAY_PIN, LOW);
  Serial.println("Heater relay initialized");

  // Initialize relay pin for humidity control
  pinMode(HUMIDIFIER_RELAY_PIN, OUTPUT);
  digitalWrite(HUMIDIFIER_RELAY_PIN, LOW);
  Serial.println("Humidifier relay initialized");

  // Initialize BH1750 light sensor on I2C port 1
  I2C_BH1750.begin(BH1750_SDA_PIN, BH1750_SCL_PIN);
  lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE, 0x23, &I2C_BH1750);
  Serial.println("Light sensor initialized");

  // Initialize relay pin for light control
  pinMode(LIGHT_RELAY_PIN, OUTPUT);
  digitalWrite(LIGHT_RELAY_PIN, HIGH);
  Serial.println("Light relay initialized");

  // Initialize relay pin for pump control
  pinMode(PUMP_RELAY_PIN, OUTPUT);
  digitalWrite(PUMP_RELAY_PIN, LOW);
  Serial.println("Pump relay initialized");

  // Initialize motor driver pins for fan control
  pinMode(FAN_IN1_PIN, OUTPUT);
```

```
  pinMode(FAN_IN2_PIN, OUTPUT);
  pinMode(FAN_SPEED_PIN, OUTPUT);
  Serial.println("Fan control initialized");

  // Initialize the LCD on default I2C port
  Wire.begin(LCD_SDA_PIN, LCD_SCL_PIN);
  if (lcd.begin(LCD_COLUMNS, LCD_ROWS)) {
    lcd.print("LCD Initialized");
  } else {
    Serial.println("Failed to initialize LCD");
  }

  Serial.println("LCD initialization checked");

  // Setup web server routes
  server.on("/", handleRoot);
  server.on("/data", handleData); // New route for sending sensor data
  server.on("/setpoints", handleSetpoints); // New route for receiving setpoints
  server.begin();
  Serial.println("Server started");
  Serial.println("Setup complete");

  // Initialize previous readings arrays with default values
  for (int i = 0; i < NUM_READINGS; i++) {
    temperatureReadings[i] = 0.0;
    humidityReadings[i] = 0.0;
  }

  // Initialize PID timers
  last_time_heater = millis();
  last_time_fan = millis();
}

void loop() {
  Serial.println("Loop start");
  readAndControlSensors();
  server.handleClient();
  cycleLCD();
  delay(1000); // Add a delay to avoid flooding the serial monitor
}

void readAndControlSensors() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    // If the sensor fails, use the last valid readings
    temperature = temperatureReadings[currentReadingIndex];
    humidity = humidityReadings[currentReadingIndex];
    Serial.println("Failed to read from DHT sensor, using last valid readings");
```

```
} else {
  // Store the new readings in the array
  temperatureReadings[currentReadingIndex] = temperature;
  humidityReadings[currentReadingIndex] = humidity;
  currentReadingIndex = (currentReadingIndex + 1) % NUM_READINGS;
}

float lux = lightMeter.readLightLevel();
soilSensorValue = analogRead(SOIL_MOISTURE_PIN);
int voltageSensorValue = analogRead(VOLTAGE_SENSOR_PIN);
float voltage = (voltageSensorValue / 1023.0) * 5.0;

Serial.print("Temperature: ");
Serial.println(temperature);
Serial.print("Humidity: ");
Serial.println(humidity);
Serial.print("Light Intensity: ");
Serial.println(lux);
Serial.print("Soil Moisture: ");
Serial.println(soilSensorValue);
Serial.print("Voltage: ");
Serial.println(voltage);

// Temperature control with PID
double now_heater = millis();
dt = (now_heater - last_time_heater) / 1000.0;
last_time_heater = now_heater;

double error_heater = tempSetPoint - temperature;
output_heater = pid(error_heater, kp_heater, ki_heater, kd_heater, dt, integral_heater, previous_heater);

if (output_heater > 0) {
  digitalWrite(HEATER_RELAY_PIN, HIGH);
} else {
  digitalWrite(HEATER_RELAY_PIN, LOW);
}

// Fan control with PID
double now_fan = millis();
dt = (now_fan - last_time_fan) / 1000.0;
last_time_fan = now_fan;

double error_fan = temperature - tempSetPoint;
output_fan = pid(error_fan, kp_fan, ki_fan, kd_fan, dt, integral_fan, previous_fan);

if (error_fan > 0) {
  digitalWrite(FAN_IN1_PIN, HIGH);
  digitalWrite(FAN_IN2_PIN, LOW); // Assuming you only need one direction
  analogWrite(FAN_SPEED_PIN, output_fan); // Adjust fan speed with PID output
```

```
  } else {
    digitalWrite(FAN_IN1_PIN, LOW);
    digitalWrite(FAN_IN2_PIN, LOW); // Stop fan
    analogWrite(FAN_SPEED_PIN, 0); // Turn off fan
  }

  // Humidity control
  if (humidity < humiditySetPoint) {
    digitalWrite(HUMIDIFIER_RELAY_PIN, HIGH);
  } else {
    digitalWrite(HUMIDIFIER_RELAY_PIN, LOW);
  }

  // Light control
  if (lux < lightSetpoint) {
    digitalWrite(LIGHT_RELAY_PIN, LOW); // Turn on light
  } else {
    digitalWrite(LIGHT_RELAY_PIN, HIGH); // Turn off light
  }

  // Soil moisture control
  if (soilSensorValue < soilSetpoint) {
    digitalWrite(PUMP_RELAY_PIN, HIGH);
  } else {
    digitalWrite(PUMP_RELAY_PIN, LOW);
  }
}

double pid(double error, double kp, double ki, double kd, double dt, double &integral, double &previous) {
  integral += error * dt;
  double derivative = (error - previous) / dt;
  double output = kp * error + ki * integral + kd * derivative;
  previous = error;
  return output;
}

void handleRoot() {
  server.send(200, "text/html", "<h1>Welcome to the Greenhouse Control System</h1>");
}

void handleData() {
  String json = "{";
  json += "\"temperature\": " + String(dht.readTemperature()) + ",";
  json += "\"humidity\": " + String(dht.readHumidity()) + ",";
  json += "\"light\": " + String(lightMeter.readLightLevel()) + ",";
  json += "\"soil_moisture\": " + String(soilSensorValue) + ",";
  json += "\"voltage\": " + String(analogRead(VOLTAGE_SENSOR_PIN)) + ",";
  json += "\"fan_speed\": " + String(analogRead(FAN_SPEED_PIN));
  json += "}";
  server.send(200, "application/json", json);
}
```

```
295
296 void cycleLCD() {
297   unsigned long currentMillis = millis();
298   if (currentMillis - previousMillis >= interval) {
299     previousMillis = currentMillis;
300     lcd.clear();
301
302     switch (currentMode) {
303       case 0:
304         lcd.print("Temp: ");
305         lcd.print(temperatureReadings[(currentReadingIndex + NUM_READINGS - 1) % NUM_READINGS]);
306         lcd.print("C");
307         currentMode = 1;
308         break;
309       case 1:
310         lcd.print("Hum: ");
311         lcd.print(humidityReadings[(currentReadingIndex + NUM_READINGS - 1) % NUM_READINGS]);
312         lcd.print("%");
313         currentMode = 2;
314         break;
315       case 2:
316         lcd.print("Light: ");
317         lcd.print(lightMeter.readLightLevel());
318         lcd.print("lx");
319         currentMode = 3;
320         break;
321       case 3:
322         lcd.print("Soil: ");
323         lcd.print(soilSensorValue);
324         currentMode = 0;
325         break;
326     }
327   }
328 }
```

4. AI CHATBOT AND USER INTERFACE (Python code)
Screen One

```python
import streamlit as st
import requests
import time
import pandas as pd
import plotly.express as px
from chatbot import greenhouse_assistant

# IP address of the ESP32
ESP32_IP = "10.90.10.208"

# Navigation menu
st.sidebar.title("GREENHOUSE")
page = st.sidebar.radio("Go to", ["Greenhouse Monitoring", "GreenHouse Assistant"])

if page == "Greenhouse Monitoring":
    st.title("MONITORING AND CONTROLLING")

    def fetch_data():
        try:
            response = requests.get(f"http://{ESP32_IP}/data")
            if response.status_code == 200:
                data = response.text.split(',')
                expected_length = 10
                if len(data) < expected_length:
                    st.error(f"Incomplete data received: {data}")
                    return None

                # Adding a fallback for missing elements
                while len(data) < expected_length:
                    data.append('0')  # Append '0' or a default value to missing elements

                return {
                    "temperature": float(data[0]),
                    "humidity": float(data[1]),
                    "light": float(data[2]),
                    "soil_moisture": int(data[3]),
                    "voltage": float(data[4]),
                    "heater_status": data[5] == '1',
```

```
                    "fan_status": data[6] == '1',
                    "humidifier_status": data[7] == '1',
                    "pump_status": data[8] == '1',
                    "light_status": data[9] == '1'
                }
            else:
                st.error("Failed to fetch data from ESP32")
        except Exception as e:
            st.error(f"Error: {e}")
        return None

    def send_setpoints(temp, humidity, light, soil):
        try:
            response = requests.get(f"http://{ESP32_IP}/setpoints", params={
                "tempSetPoint": temp,
              "humiditySetPoint": humidity,
              "lightSetpoint": light,
              "soilSetpoint": soil
            })
            if response.status_code == 200:
                st.success("Setpoints updated successfully")
            else:
                st.error("Failed to update setpoints")
        except Exception as e:
            st.error(f"Error: {e}")

    def update_dataframe(df, data):
        new_row = {
            "Timestamp": pd.Timestamp.now(),
            "Temperature": data["temperature"],
            "Humidity": data["humidity"],
            "Light": data["light"],
            "Soil Moisture": data["soil_moisture"]
        }
        return pd.concat([df, pd.DataFrame([new_row])], ignore_index=True)

    def plot_data(df, temp_setpoint, humidity_setpoint, light_setpoint, soil_setpoint):
        # Update plots with actual data
        with temp_chart:
```

```
        st.subheader("Temperature Over Time")
        temp_fig = px.line(df, x="Timestamp", y="Temperature", title="Temperature Over Time")
        temp_fig.add_hline(y=temp_setpoint, line_dash="dash", line_color="red", annotation_text="Setpoint")
        st.plotly_chart(temp_fig, use_container_width=True)

    with humidity_chart:
        st.subheader("Humidity Over Time")
        humidity_fig = px.line(df, x="Timestamp", y="Humidity", title="Humidity Over Time")
        humidity_fig.add_hline(y=humidity_setpoint, line_dash="dash", line_color="red", annotation_text="Setpoint")
        st.plotly_chart(humidity_fig, use_container_width=True)

    with light_chart:
        st.subheader("Light Intensity Over Time")
        light_fig = px.line(df, x="Timestamp", y="Light", title="Light Intensity Over Time")
        light_fig.add_hline(y=light_setpoint, line_dash="dash", line_color="red", annotation_text="Setpoint")
        st.plotly_chart(light_fig, use_container_width=True)

    with soil_chart:
        st.subheader("Soil Moisture Over Time")
        soil_fig = px.line(df, x="Timestamp", y="Soil Moisture", title="Soil Moisture Over Time")
        soil_fig.add_hline(y=soil_setpoint, line_dash="dash", line_color="red", annotation_text="Setpoint")
        st.plotly_chart(soil_fig, use_container_width=True)

# Setpoints
st.header("Set Control Setpoints")
temp = st.slider("Temperature Setpoint (°C)", min_value=0.0, max_value=50.0, value=23.0)
humidity = st.slider("Humidity Setpoint (%)", min_value=0.0, max_value=100.0, value=60.0)
light = st.slider("Light Intensity Setpoint (lx)", min_value=0, max_value=20000, value=5000)
soil = st.slider("Soil Moisture Setpoint", min_value=0, max_value=2000, value=300)

if st.button("Update Setpoints"):
    send_setpoints(temp, humidity, light, soil)

# Real-Time Data
st.header("Real-Time Data")

# Initialize placeholders for metrics
temperature_placeholder = st.empty()
humidity_placeholder = st.empty()
light_placeholder = st.empty()
soil_moisture_placeholder = st.empty()
voltage_placeholder = st.empty()

# Initialize placeholders for statuses
status_placeholder = st.empty()

# Initialize placeholders for charts
temp_chart = st.empty()
humidity_chart = st.empty()
light_chart = st.empty()
soil_chart = st.empty()
```

```
160             data_df = update_dataframe(data_df, data)
161
162             # Update metrics with actual data
163             with temperature_placeholder:
164                 st.metric(label="Temperature (°C)", value=data["temperature"])
165             with humidity_placeholder:
166                 st.metric(label="Humidity (%)", value=data["humidity"])
167             with light_placeholder:
168                 st.metric(label="Light Intensity (lx)", value=data["light"])
169             with soil_moisture_placeholder:
170                 st.metric(label="Soil Moisture", value=data["soil_moisture"])
171             with voltage_placeholder:
172                 st.metric(label="Voltage (V)", value=data["voltage"])
173             with status_placeholder:
174                 st.text(f"Heater Status: {'ON' if data['heater_status'] else 'OFF'}")
175                 st.text(f"Fan Status: {'ON' if data['fan_status'] else 'OFF'}")
176                 st.text(f"Humidifier Status: {'ON' if data['humidifier_status'] else 'OFF'}")
177                 st.text(f"Pump Status: {'ON' if data['pump_status'] else 'OFF'}")
178                 st.text(f"Light Status: {'ON' if data['light_status'] else 'OFF'}")
179
180             # Update the charts with actual data
181             plot_data(data_df, temp, humidity, light, soil)
182
183         time.sleep(10)
184
185 elif page == "GreenHouse Assistant":
186     st.title("GREENHOUSE ASSISTANT")
187
188     # Directly set the API key here
189     api_key = "API_KEY"
190
191     # Call the chatbot from the separate file
192     greenhouse_assistant(api_key)
193
```

AI Chat bot: Screen Two

```python
# chatbot.py
from openai import OpenAI
import streamlit as st

def greenhouse_assistant(api_key):
    client = OpenAI(api_key=api_key)

    if "openai_model" not in st.session_state:
        st.session_state["openai_model"] = "gpt-3.5-turbo"

    if "messages" not in st.session_state:
        st.session_state.messages = []

    for message in st.session_state.messages:
        with st.chat_message(message["role"]):
            st.markdown(message["content"])

    if prompt := st.chat_input("Hello farmer?"):
        st.session_state.messages.append({"role": "user", "content": prompt})
        with st.chat_message("user"):
            st.markdown(prompt)

        with st.chat_message("assistant"):
            stream = client.chat.completions.create(
                model=st.session_state["openai_model"],
                messages=[
                    {"role": m["role"], "content": m["content"]}
                    for m in st.session_state.messages
                ],
                stream=True,
            )
            response = st.write_stream(stream)
        st.session_state.messages.append({"role": "assistant", "content": response})

```