بسم الله الرحمن الرحيم

PALESTINE POLYTECHNIC UNIVERSITY

College of Information Technology and Computer Engineering

Computer Systems Engineering Department

Title

Smart Aeroponic Farming System

By

Murad Faisal Iqtait

Wesam Khaled Daraweish

Saria Mohammad Iqtail

Supervisor

Dr. Zein Salah

Dr. Rami Arafeh

2024-2025

Acknowledgements

In the name of Allah, the Most Gracious, the Most Merciful, who has granted us the insight and determination to embark on and complete this project. The process of formulating this initial plan has been a valuable learning experience.

First and foremost, we are profoundly thankful to our supervisors, Dr. Zein Salah and Dr. Rami Arafeh, for their initial guidance, feedback, encouragement, support in shaping our project idea and developing this proposal. Their dedication, patience, and insightful advice were pivotal in shaping our research and ensuring the success of this project. We extend our gratitude to the staff of the College of Information Technology and Computer Engineering for their invaluable support and assistance.

We would also like to express our heartfelt gratitude to our families and friends for their unwavering love, encouragement, and belief in us. Their continuous support and motivation were a source of strength that helped us overcome challenges and remain focused on our goals.

Thank you from the bottom of our hearts.

Abstract

This research addresses the critical challenge of food scarcity exacerbated by limited arable land and unsustainable traditional agricultural practices. A smart aeroponic farming system was developed and implemented, leveraging vertical farming techniques and Internet of Things (IoT) technology to cultivate crops in a controlled environment. The system integrates sensors for real-time monitoring of environmental parameters (temperature, humidity, pH, light, and nutrient levels) and automated control of water and nutrient delivery, and lighting. The implementation of this system demonstrates significant improvements in resource utilization, enhancing productivity, and minimizing the reliance on pesticides. The system also facilitates remote monitoring and control, promoting local food security and self-sufficiency. This research provides a viable and sustainable solution to enhance agricultural practices and address food scarcity, paving the way for broader adoption of smart farming technologies.

يعالج هذا البحث التحدي الحرج المتمثل في ندرة الغذاء التي تتفاقم بسبب محدودية الأراضي الصالحة للزراعة والممارسات الزراعية التقليدية غير المستدامة. تم تطوير وتنفيذ نظام زراعة هوائية ذكي، يعتمد على تقنيات الزراعة الرأسية وتقنية إنترنت الأشياء (IoT) لزراعة المحاصيل في بيئة مُتحكم بها. يدمج النظام مستشعرات للمراقبة الآنية للمعايير البيئية (درجة الحرارة والرطوبة ودرجة الحموضة والضوء ومستويات المغذيات) والتحكم الآلي في توصيل المياه والمغذيات والإضاءة. يُظهر تطبيق هذا النظام تحسينات كبيرة في استخدام الموارد، وزيادة الإنتاجية، وتقليل الاعتماد على المبيدات الحشرية. كما يسهل النظام المراقبة والتحكم عن بعد، مما يعزز الأمن الغذائي والاكتفاء الذاتي المحلي. يقدم هذا البحث حلاً قابلاً للتطبيق ومستدامًا لتحسين الممارسات الزراعية ومعالجة ندرة الغذاء، مما يمهد الطريق لاعتماد أوسع لتقنيات الزراعة الذكية.

Contents

INTRODUCTION **1**

1.1 PREFACE 1
1.2 PROBLEM STATEMENT 1
1.3 PROJECT AIMS AND OBJECTIVES 2
1.4 PROJECT REQUIREMENTS 2
1.4.1 Functional *2*
1.4.2 Non-functional *3*
1.5 SYSTEM DESCRIPTION 4
1.6 PROJECT LIMITATIONS/CONSTRAINTS 4
1.7 PROJECT SCHEDULE 5
1.8 REPORT OUTLINE 5

BACKGROUND **6**

2.1 PREFACE 6
2.2 THEORETICAL BACKGROUND CONCEPTS 6
2.2.1 Aeroponic tower farming *6*
2.2.2 Potential of Hydrogen (PH) and Electrical Conductivity (EC) *6*
2.2.3 Internet of Things (IoT): *7*
2.2.4 Environmental Sensors: *7*
2.2.5 Database: *7*
2.2.6 Microcontroller (e.g. ESP32): *7*
2.2.7 Motors and Pumps: *7*
2.2.8 Power Supply Components *7*
2.2.9 Display Component *7*
2.3 LITERATURE REVIEW 8
2.3.1 Smart Aquaponics Monitoring System Via IoT Techniques [8]: *8*
2.3.2 IoT based Customizable Vertical Farming Solution for Palestinian Plants [9]: *8*
2.4 SUMMARY 9

SYSTEM DESIGN **10**

3.1 PREFACE 10
3.2 SYSTEM COMPONENTS AND DESIGN ALTERNATIVES 10
3.2.1 Hardware components *10*
3.2.1.1 Controllers 10
3.2.1.2 Sensors 11
3.2.1.3 Display Screen 14
3.2.1.4 Power Source 15
3.2.2 Software components *15*
3.2.2.1 Software Application 15
3.2.2.2 Internet of Things (IoT) 17
3.2.2.3 Data Storage and Processing 18
3.2.2.4 SQL Database and Local Testing 19
3.2.2.5 Development Tools 20
3.3 CONCEPTUAL SYSTEM DESCRIPTION 21

3.4 ALGORITHMS AND METHODOLOGIES 25
3.4.1 Control Methodologies *25*
3.4.2 Control Algorithms *26*
3.5 SCHEMATIC DIAGRAMS 33
3.6 SUMMARY 38

IMPLEMENTATION AND TESTING **39**

4.1 IMPLEMENTATION ISSUES 39
4.1.1 Hardware implementation *39*
4.1.1.1 Prototype setup 39
4.1.1.2 Sensor Integration 40
4.1.1.3 Relay Module Configuration 40
4.1.1.4 Nutrient Pumps 41
4.1.1.5 Mixer motor 41
4.1.1.6 Strip Light 42
4.1.1.7 Cooling Fan 42
4.1.1.8 Water Pump 42
4.1.1.9 Power Management and Solar Setup 43
4.1.1.10 Display Screen 44
4.1.1.11 PCB Assembly and Final Wiring 44
4.1.2 Software implementation *44*
4.1.2.1 Web Application 44
4.1.2.2 ESP32 Firmware 47
4.1.2.3 Development Environment and Tools 49
4.1.3 Agricultural Implementation *49*
4.2 IMPLEMENTATION CHALLENGES 51
4.3 VALIDATION AND TESTING 53
4.3.1 Sensor Calibration and Verification *53*
4.3.1.1 PH Sensor 53
4.3.1.2 TDS Sensor 55
4.3.1.3 DHT11 Sensor 55
4.3.1.4 LDR Sensor 56
4.3.1.5 Nutrient Level 56
4.3.2 Actuators and Outputs Testing *57*
4.3.2.1 Nutrient Pumps (Relay 1) 57
4.3.2.2 Water Pump + Mixer + Fan + LED (Relay 2) 57
4.3.2.3 Frontend and Server Integration Testing 57
4.3.2.4 Screen Display Verification with Sensors 57
4.3.2.5 Power System Validation 58
4.4 SUMMARY 59

RESULTS AND DISCUSSION **60**

5.1 DETAILED ANALYSIS OF THE RESULTS/EXPERIMENTS 60
5.1.1 Sensor Readings *60*
5.1.1.1 PH sensor 60
5.1.1.2 TDS sensor 60
5.1.1.3 DHT11 sensor 60
5.1.1.4 LDR sensor 60

5.1.2 Actuator Responses *61*
5.1.2.1 Nutrient Pumps (x4) 61
5.1.2.2 Water Pump, Fan, LED Strip, and Mixer 61
5.1.2.3 System Integration Test 61
5.2 ERROR / SUCCESS RATE CALCULATIONS 61
5.2.1 Sensor Accuracy *61*
5.2.1.1 PH sensor 61
5.2.1.2 TDS sensor 61
5.2.1.3 DHT11 sensor 62
5.2.1.4 LDR sensor 62
5.2.2 Actuator Reliability *62*
5.2.3 Overall System Stability *62*
5.2.4 Conclusion *63*
5.3 JUSTIFICATIONS OF THE OBTAINED RESULTS 63
5.3.1 Sensors *63*
5.3.1.1 PH sensor Readings 63
5.3.1.2 TDS sensor Response 63
5.3.1.3 Temperature and Humidity (DHT11) 63
5.3.1.4 LDR and LED Activation 63
5.3.2 Overall System Behavior *64*
5.4 SUMMARY 64

CONCLUSION AND FUTURE WORK 65

6.1 CONCLUDING REMARKS 65
6.2 FUTURE WORK 66

REFERENCE 68

List of Figures

1.1 Conceptual image of system 4
3.1 General block diagram 22
3.2 General block diagram for multi-unit 23
3.3 System conceptual diagram 23
3.4 Zero-Level block diagram 24
3.5 System Initialization Algorithm.....26
3.6 Sensor Data Acquisition Algorithm 27
3.7 Temperature Control Algorithm 27
3.8 EC (TDS) Control Algorithm 27
3.9 pH Control Algorithm 27
3.10 Light Control Algorithm 28
3.11 Nutrient Tank Level Monitoring Algorithm 28
3.12 Display Algorithm 28
3.13 Loop Execution Algorithm 28
3.14 Main Control Algorithm Pseudocode 29
3.15 Flowchart diagram 30
3.16 Sequence diagram 31
3.17 Use Case diagram 32
3.18 DHT11 Sensor Schematic diagram 33
3.19 PH Sensor Schematic diagram 33
3.20 TDS Sensor Schematic diagram 34
3.21 LDR Sensor Schematic diagram 34
3.22 Nutrient Level Sensor Schematic diagram 35
3.23 TFT 2.8" ST7789 Schematic diagram 36
3.24 First Relay with Nutrient Pumps Schematic diagram 36
3.25 Relay and Nutrient Pumps Schematic diagram 37
3.26 Power Source Schematic diagram 37
3.27 System Schematic diagram 38
4.1 Prototype setup 39
4.2 Sensor Integration 40
4.3 Relay Module Configuration 1 41
4.4 Relay Module Configuration 2 41
4.5 Mixer motor 42
4.6 Strip Light 42
4.7 Cooling Fan 43
4.8 Solar panel 43
4.9 Solar charge controller 44
4.10 Battery 44
4.11 TFT screen 44
4.12 PCB Assembly 45
4.13 Home Page 45
4.14 Dashboard Page 46
4.15 Sitting Page 46
4.16 Users Database 47
4.17 Planting process 50

4.18 PH Sensor and test values 51
4.19 Code for solution 51
4.20 Solar panel at 45° angle 53
4.21 PH sensor testing and calibration 54
4.22 PH sensor after calibration 54
4.23 TDS sensor testing and calibration 55
4.24 DHT11 sensor testing 55
4.25 LDR sensor testing 56
4.26 Nutrient Level circuit 57
4.27 TFT display live reading 58
4.28 Solar charge controller testing 58

List of Tables

1.1 Project schedule in the first and the second semester 5
2.1 Literature Review 8
3.1 Differences between microcontroller models 11
3.2 Differences between PH sensor models 12
3.3 Differences between Temperature and Humidity sensor models 12
3.4 Differences between Electrical Conductivity (EC) sensor models 13
3.5 Differences between Light sensor models 13
3.6 Differences between Nutrient Level sensor models 14
3.7 Differences between Display models 14
3.8 Differences between Power Source models 15
3.9 Differences between Web Application Framework 16
3.10 Differences between Backend Development 16
3.11 Comparison of IoT Communication Methods 18
3.12 Comparison of Storage Options 18
3.13 Differences between SQL Databases for Authentication 19
3.14 Differences between Local Testing Environments 20
3.15 Differences between Development Tools 20
4.1 For each sensor 48

Chapter 1

Introduction

1.1 Preface

In this project, we implement a Smart Aeroponic Farming System aimed at addressing food scarcity challenges due to the limited availability of arable land. This system leverages vertical farming and aeroponic techniques to grow crops in a controlled environment, optimizing space and resource utilization. By using modern technological techniques, the project seeks to enhance local food production, and support sustainable agriculture, ultimately promoting food security and environmental conservation.

1.2 Problem Statement

The vision is to achieve a nation where there is ample, safe, and locally produced food for all citizens, with farmers thriving through innovative agricultural techniques that reduce environmental impact and conserve resources. This vision faces several challenges, the most prominent being the scarcity of arable land due to urban expansion and land confiscation by the occupation. Additionally, reliance on traditional farming methods makes crops vulnerable to diseases and pests, increasing the need for pesticides and raising production costs. Dependence on food imports also imposes significant economic pressures and reduces agricultural opportunities, leading to rising unemployment among farmers. To address these challenges, the implementation of an aeroponic farming system is proposed. This system allows for high-yield crop production in small spaces while reducing the need for pesticides and water consumption. It not only helps increase productivity and optimize resource use, but also provides new job opportunities and reduces reliance on food imports, thereby enhancing food security and making the nation more self-sufficient.

1.3 Project Aims and Objectives

In this project, we propose a system that aims to provide the following features:

A. The system aims to **Increase Agricultural Productivity**, in order to achieve this aim, the following objectives should be accomplished:
   - Ensure optimal growing conditions through continuous monitoring of environmental factors.

B. The system aims to **Optimize Space Utilization**, in order to achieve this aim, the following objectives should be accomplished:
   - Develop a vertically efficient farming solution for urban areas with limited land.
   - Maximize crop density using vertical stacking and layout optimization.

C. The system aims to **Reduce Resource Consumption**, in order to achieve this aim, the following objectives should be accomplished:
   - Automate adjustments based on continuous monitoring.

D. The system aims to **Promote Local Self-Sufficiency**, in order to achieve this aim, the following objectives should be accomplished:
   - Ensure high-quality local produce by continuously monitoring crop health.
   - Lower long-term costs and encourage local community adoption.

E. The system aims to **Sustainability,** in order to achieve this aim, the following objectives should be accomplished:
   - Power the system with renewable solar energy, making it eco-friendly.
   - Enable remote management via a dedicated application for monitoring and control.

1.4 Project Requirements

To specify the system requirements, the following functional and non-functional requirements can be considered:

1.4.1 Functional

1. **Automated Water and Nutrient Delivery**: the system shall deliver water and nutrients to plant roots at regular intervals, the system shall adjust the nutrient concentration based on plant growth stages (e.g. vegetative, flowering stages).

2. **Environmental Monitoring:** real-time monitoring of temperature, humidity, and PH (potential of Hydrogen), with alerts if parameters deviate from safe ranges. Users can monitor and control the system remotely through a mobile or web interface. The system also tracks reservoir nutrient levels and notifies users when low.

3. **Automated Lighting Control:** The system shall control LED grow lights based on a programmable light schedule (e.g., 16 hours on, 8 hours off for vegetative growth). Additionally, an LDR sensor shall monitor ambient light levels during the scheduled "on" period, and if the light is insufficient, the LED grow lights will be activated to ensure optimal lighting conditions.
4. **Data Logging and Analytics:** logs data for temperature, humidity, PH (Potential of Hydrogen), EC (Electrical Conductivity), and nutrients usage, providing users with analytical reports or visualizations on system performance and plant growth.
5. **Emergency Shutdown and Alerts:** The system includes an emergency shutdown feature for critical failures (e.g., excessive dosing of nutrients). Real-time alerts will be displayed on the web interface to notify users immediately of any emergency conditions, ensuring quick response and system safety.

1.4.2 Non-functional

1. **Performance:** the system shall respond to sensor data changes (e.g., temperature or humidity variations or PH value) as fast as possible. with one reading from each sensor every 0.5 seconds.
2. **Reliability:** The system shall operate 24/7 with a maximum downtime of 1% per month (approximately 7.2 hours).
3. **Scalability:** the system shall support additional sensors and accommodates new plant types or configurations without major reconfiguration.
4. **Accuracy:** the PH sensor shall measure values with an accuracy of ±0.5 units, temperature and humidity sensors shall have a margin of error of no more than ±2°C and ±5% relative humidity, respectively [1].
5. **Usability:** the user interface shall be intuitive, with a user-friendly design that allows users to control and monitor the system with minimal effort or technical knowledge.
6. **Maintainability:** The system's software shall be modular, well-documented software for easy updates and maintenance, with easily replaceable hardware components.
7. **Safety and Security:** The system shall include secure remote access requiring authentication, safety protocols to prevent over-delivery of nutrients, and immediate shutdown in case of abnormal readings that could harm plants or system integrity. Over-delivery prevention is achieved by gradually adding diluted nutrient solutions through timed pump activations lasting two seconds, followed by sensor readings. This process repeats until the nutrient level reaches the target safe range.

1.5 System Description

The Smart Aeroponic Farming System addresses food scarcity and limited arable land by enabling efficient, space-saving crop growth in urban areas. Using vertical aeroponic techniques, this system grows plants without soil, spraying nutrient-rich water directly onto their roots, and recycling resources to reduce water and land use. Equipped with IoT sensors, the system monitors environmental factors like temperature, humidity, pH, and light, adjusting automatically for optimal growth. LED lights enable year-round cultivation, while a remote interface allows users to control and monitor the system. This eco-friendly approach minimizes pesticide use and can run on renewable energy, promoting sustainability and supporting local food security in urban environments.

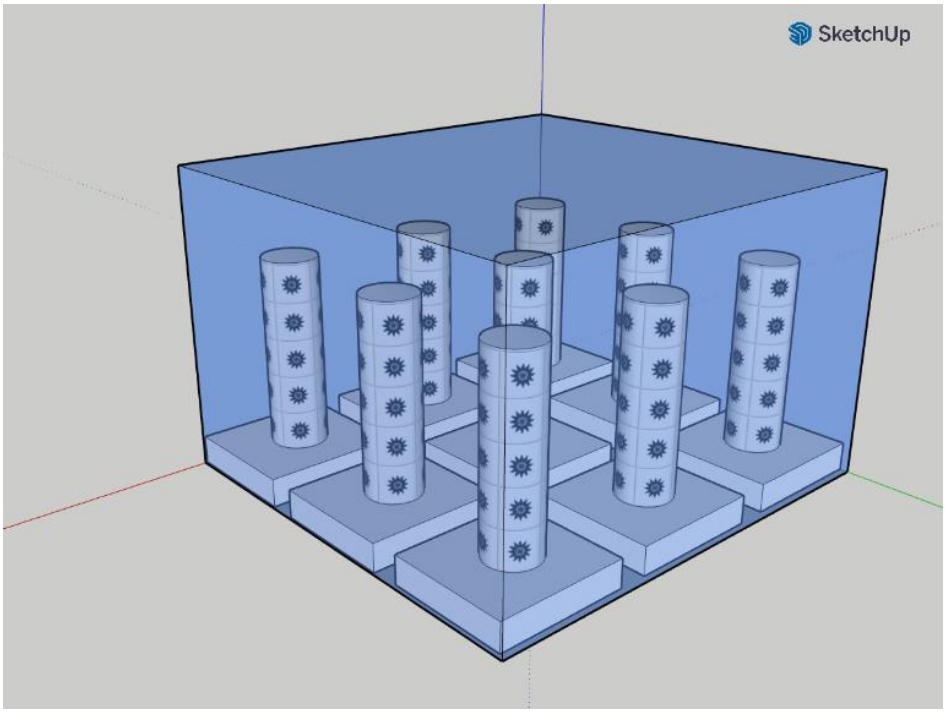


Figure 1.1: Conceptual image of system

1.6 Project Limitations/constraints

1. **Power Dependence:** the system's reliance on electricity makes it vulnerable to power outages, potentially disrupting critical functions and impacting crop yield. While solar integration is proposed, its reliability and capacity require careful assessment.
2. **Initial Investment Costs:** setting up a smart aeroponic farming system requires a significant initial financial investment, including the infrastructure, equipment, sensors, and software. This cost might be a barrier to entry, especially for small-scale farmers or those with limited access to capital.

3. **Environmental Vulnerability:** while the controlled environment offers advantages, extreme weather events, such as heat waves or severe cold, can impact the system's ability to maintain optimal growing conditions.
4. **Crop Suitability:** not all crops are equally suitable for aeroponic cultivation, especially those with long stems, such as wheat or watermelon.
5. **The continuous internet connection:** internet connectivity is necessary to access information through the mobile application. However, the system operates correctly without an internet connection.

1.7 Project Schedule

Table 1.1: Project schedule in the first and the second semester

semesters	The first semester			The second semester			
Week	1-5	6-10	11-16	1-5	6-9	10-14	14-16
Selection of project Idea							
Collecting the Data							
System Design							
System Implementation							
System testing							
system operation							
Documentation							

1.8 Report Outline

The rest of the report is organized as follows: Chapter 2 presents a theoretical background of the project. Chapter 3 introduces the system design. Chapter 4 discusses the implementation and testing, including key implementation aspects with illustrative photos, challenges encountered during development, and the procedures followed for validation and testing. Chapter 5 presents the results and discussion, providing a detailed analysis of experiments and results, including success/error rate calculations and justifications for the outcomes. Chapter 6 concludes the report with final remarks and outlines potential future improvements and extensions to the system

Chapter 2

Background

2.1 Preface

This chapter establishes the theoretical foundation essential to our project. It begins with a thorough examination of the key concepts and principles that inform our work, thereby preparing the ground for a comparative literature review. Through this review, we situate our project within the broader academic context, analyzing prior research to highlight the unique contributions and innovations we offer. Ultimately, this chapter delivers a comprehensive overview that traces the origins of our project and underscores its relevance and significance within the field.

2.2 Theoretical background Concepts

In this section, we'll explain the core components of our project:

2.2.1 Aeroponic tower farming: is a modern agricultural technique where plants are grown without soil; their roots are suspended in air and misted with a nutrient-rich solution at regular intervals. These systems use vertical structures (towers), making them highly space-efficient and ideal for urban and limited-space environments. Aeroponics offers faster plant growth, reduced water usage, and minimal exposure to pests and diseases. The main components include a reservoir, pump, misting system, and sometimes artificial lighting for controlled environments. This sustainable method is a promising solution for future food production [2].

2.2.2 Potential of Hydrogen (PH) and Electrical Conductivity (EC): are two key parameters used to assess the chemical properties of liquids, especially in applications like agriculture, water quality monitoring, and environmental science. The pH value indicates the acidity or alkalinity of a solution on a scale from 0 to 14, where values below 7 represent acidic conditions, 7 is neutral, and values above 7 are alkaline. Accurate pH measurement is crucial, for example, in ensuring optimal nutrient availability in soil or hydroponic systems. On the other hand, EC measures a solution's ability to conduct electricity, which is directly related to the concentration of dissolved salts or ions in the water. High EC values indicate high salinity, which can affect plant growth or signal pollution in water sources. Together, pH and EC readings provide a comprehensive picture of water chemistry, enabling better environmental agricultural management [38][39].

2.2.3 Internet of Things (IoT): refers to a network of interconnected physical devices embedded with sensors, software, and other technologies that enable them to collect and exchange data over the internet. This connectivity allows devices to be monitored, controlled, and automated remotely, making processes more efficient and enabling real-time insights [3].

2.2.4 Environmental Sensors: are devices designed to measure specific environmental parameters, providing real-time data essential for monitoring and informed decision-making. For example, temperature and humidity sensors (such as the DHT11) measure ambient temperature and moisture levels; pH sensors (like the Electrode Glass E-201) determine the acidity or alkalinity of a solution; and electrical conductivity (EC) sensors (e.g., TDS meters) assess the concentration of dissolved solids in liquids. These sensors are widely used in agriculture, environmental protection, and industrial monitoring, offering critical insights that support sustainability and operational efficiency. [4][5].

2.2.5 Database: A local SQL-based database will be used to store, organize, and process data within the system. Databases are a fundamental component of any information system, allowing for efficient storage and retrieval of large amounts of structured data. By using a local database, the system ensures faster access to data and reduces dependency on internet connectivity, which enhances performance and increases reliability in environments with limited or unstable network access. [6].

2.2.6 Microcontroller (e.g. ESP32): is a compact integrated circuit designed to govern specific tasks in embedded systems. It combines a processor, memory, and input/output peripherals on a single chip, making it ideal for applications [7].

2.2.7 Motors and Pumps: Motors are electromechanical devices that convert electrical energy into mechanical motion. They are often used to drive moving components in systems such as irrigation or fluid transport. Pumps, on the other hand, are used to move fluids from one place to another, typically powered by motors. In our project, they play a key role in [e.g., water circulation, nutrient delivery], making them essential components to understand within the system [40].

2.2.8 Power Supply Components: The project utilizes a solar panel as a renewable energy source to convert sunlight into electrical energy. This energy is stored in a rechargeable battery, providing a stable and continuous power supply to the system, especially in remote or off-grid locations, enhancing its reliability and sustainability.

2.2.9 Display Component (e.g. TFT 2.8" Screen): A 2.8-inch TFT display is integrated into the system to provide a user-friendly interface. It displays real-time data such as sensor readings, system status, and alerts. The high-resolution graphical display enhances usability by allowing users to easily monitor environmental parameters.

2.3 Literature Review

Compared to other projects, our system introduces unique features and improvements, which enhance environmental monitoring and control in ways not typically found in similar systems, and simulating the entire project in an air-conditioned workshop.

This section reviews recent research and projects related to smart farming systems:

2.3.1 Smart Aquaponics Monitoring System Via IoT Techniques [8]:

The proposed system focuses on aeroponic farming with continuous water irrigation, advanced environmental monitoring, and water filtration systems, allowing for precise control over plant growth conditions. Unlike the " Smart Aquaponics Monitoring System via IoT Techniques ", our system does not rely on a closed-loop fish-plant interaction. Instead, it includes water filtration mechanisms that maintain optimal water quality. Additionally, our project is designed to support a variety of plant types, including fruit-bearing and medium-sized plants, expanding beyond the leafy plants typically supported in aquaponics. This approach ensures greater flexibility and scalability, along with full remote control and automated nutrient management for plant growth.

2.3.2 IoT based Customizable Vertical Farming Solution for Palestinian Plants [9]:

The proposed system introduces significant differences over the " IoT based Customizable Vertical Farming Solution for Palestinian Plants " Unlike Their project, which is designed primarily for leafy plants with roots submerged in water, our aeroponic system supports a broader range of plants, including fruit-bearing and medium-sized plants, by avoiding root submersion and creating a more adaptable growth environment. Our system incorporates advanced water filtration and irrigation techniques that ensure continuous water flow and high-water quality without constant root immersion. Additionally, it features automated environmental monitoring and control, with precise adjustments to temperature, humidity, and lighting, optimizing conditions for diverse plant needs. Automated nutrient management further enhances efficiency by minimizing manual input, ensuring that each plant consistently receives the ideal nutrient balance. With its scalable and adaptable design, our project can be applied across various agricultural settings, making it a versatile and comprehensive solution for modern farming needs.

Table 2.1: Literature Review

Project title	**Our project**	**Smart Aquaponics Monitoring System Via IoT Techniques**	**IoT based Customizable Vertical Farming Solution for Palestinian Plants**
Author	iqtail, Saria; daraweish, Wesam; iqtait, Murad	Arab, Rasheed; Ideas, Mahmoud	Hasanat, Fatima; Nofal, Ghada; Awad, Shorouq
Some strengths	-Precise control.. -Scalability and modularity. -Increased Crop Yield. -Clean energy system powered by solar energy. -Optimal environment.	-Utilizing plant waste to feed fish, while using fish waste as fertilizer for the plants. -Implementing a camera system to detect and diagnose plant diseases.	-Precise control. -Optimal environment.
Plant size	All plants	Leafy plants	Leafy plants
Plant species in one cycle	Multiple	One	One
Full remote control	Yes	No	Yes
Can be used by non-expertise	Yes	No	Yes

2.4 Summary

"In this chapter, the key theoretical concepts and related studies forming the foundation of this project were explored. Concepts such as IoT and environmental sensors were outlined to establish their role in enhancing the efficiency of aeroponic farming. The literature review showcased various projects addressing smart farming challenges, highlighting the gaps that this project aims to fill. Together, this theoretical framework and background research underscore the relevance of this project's approach to innovating in precision agriculture."

Chapter 3

SYSTEM DESIGN

3.1 Preface

This chapter outlines the key hardware and software components integral to our project, offering an exploration of various alternatives for each element. It provides a conceptual overview of the system, supported by a detailed block diagram. Furthermore, the chapter delves into the system's algorithms and methodologies, illustrated through flowcharts. Schematic diagrams are also presented to showcase the interactions and interfaces between the components, ensuring a clear understanding of the system's structure and functionality.

3.2 System components and Design alternatives

This part describes the hardware and software components and their design alternatives; thus, we need a:

3.2.1 Hardware components

The following hardware components are critical to the system's functionality:

3.2.1.1 Controllers

The system requires a microcontroller to serve as its central processing unit. This microcontroller will be responsible for acquiring data from sensors, processing inputs, executing control algorithms, and communicating with connected devices and IoT platforms. It must be versatile, power-efficient, and capable of interfacing with multiple sensors and actuators, Table 3.1 presents a comparison of potential microcontroller options based on critical criteria such as Processing power, Connectivity, Pros, Cons, and Cost.

Based on Table 3.1, We chose (**ESP32**) for its balance of computational power, connectivity (Wi-Fi and Bluetooth for IoT integration), energy efficiency, and cost aligning with the system's requirements.

Table 3.1: Differences between microcontroller models:

Controllers / Specifications	ESP32 DEV [10]	Arduino UNO R3[11]	Raspberry PI 4 [12]
CPU	Xtensa dual-core 32-bit LX6	ATMega328P	Quad core Cortex-A72
Clock speed	240 MHz	16 MHz	1.8GHz
Flash memory	4 MB	32 KB	Use a microSD
EEPROM	Does not have an EEPROM	1 KB	Does not have an EEPROM
SRAM	520 KB	2 KB	1GB, 2GB, 4GB or 8GB
Number of pins	48	32	40
Digital I/O pins	36	14	26
PWM Digital I/O pins	16	6	Does not support
Analog I/O pins	18	6	requires an external ADC module
Shield compatibility	NO	YES	NO
Wireless connectivity	YES	No (can added by using shields)	YES
Pros	High performance, IoT-friendly	Easy to use, widely supported	High processing speed, cost-effective
Cons	Requires careful power management	No Wi-Fi/Bluetooth, slower	Requires external wireless modules
Pricing ($) [13][14]	18-25	8-15	80-120

3.2.1.2 Sensors

To achieve the desired functionalities in our system, we require sensors to monitor various environmental and operational parameters. These sensors are critical for collecting data, which is then processed by the microcontroller to ensure optimal system performance, the following sensors are used:

1) PH sensor

Table 3.2 compares various pH sensors types, evaluating them based on accuracy, Range, Sensor Lifespan, Stability, Response Time, Maintenance, Cost, Pros, and Cons.

Based on Table 3.2, We chose (**Glass Electrode**), Because we need high accuracy in pH, we also need a less response time reading to control the addition of ingredients.

Table 3.2: Differences between PH sensor models:

Model	**Glass Electrode [15]**	**Antimony Electrode [16]**
Accuracy	± 0.5 pH	± 2 pH
Range	0-14 pH	2-12 pH
Sensor Lifespan	1-3 years	6-12 months
Stability	Good	Average
Response Time	Seconds	Minutes
Maintenance	Calibration & Cleaning	Calibration & Cleaning
Cost ($)	40	10
Pros	High accuracy, stable, wide range, readily available	Inexpensive, durable, easy to use
Cons	Fragile, requires regular calibration, may be affected by organic matter and proteins, not suitable for harsh environments	Lower accuracy, limited range, affected by temperature and dissolved oxygen

2) Temperature and Humidity Sensor

Table 3.3 compares various Temperature and Humidity sensors types, evaluating them based on accuracy, Features, Pros, Cons and Cost.

Based on Table 3.3, We chose (**DHT11**), Because we don't need high accuracy in measuring temperature, and DHT11 satisfies the purpose and its price is cheaper.

Table 3.3: Differences between Temperature and Humidity sensor models:

Model	**Accuracy**	**Features**	**Pros**	**Cons**	**Cost ($)**
DHT22[17]	Temperature: ±0.5°C Humidity: ±2% RH	Digital output, high accuracy, integrated RH	Accurate, dual functionality	Slightly higher cost	10
DHT11[18]	Temperature: ±2°C Humidity: ±5% RH	Digital output, low accuracy, integrated RH	Low cost	Less accurate	6

3) Electrical Conductivity (EC) Sensor

Table 3.4 compares various Electrical Conductivity (EC) sensors types, evaluating them based on accuracy, Type, Pros, Cons and cost.
Based on Table 3.3, We chose (**TDS**), Because its price is reasonable and it is difficult to obtain other types due to their unavailability.

Table 3.4: Differences between Electrical Conductivity (EC) sensor models:

Model	Accuracy	Type	Pros	Cons	Cost ($)
TDS [19]	±10% FS	Digital	- Affordable, - Easy to connect, - Available locally, - Robust for hydroponics	- Lower accuracy	25
DFRobot Gravity EC [20]	±5% FS	Analog	- Affordable, - Easy to connect to ADC on ESP32	- Lower accuracy than digital - Limited range for complex solutions	35
Atlas Scientific EZO-EC [21]	±1% FS	Digital	- High accuracy, - I2C/UART for easy ESP32 integration	- Higher cost, - Requires calibration	60

4) Light Sensor

Table 3.5 compares various Light sensors types, evaluating them based on Accuracy, Pros, Cons and Cost.
Based on Table 3.3, We chose (**LDR**), Because we don't need high accuracy in measuring Light, and LDR satisfies the purpose and its price is cheaper.

Table 3.5: Differences between Light sensor models:

Model	Accuracy	Pros	Cons	Cost ($)
LDR [22]	Low	Simple to use, Inexpensive, Good for general light/dark detection	Non-linear response, Slow response time, Temperature sensitive, No calibrated output	0.5
TSL2561 [23]	High	Accurate and linear, Fast response time, Calibrated output, Digital output (I2C)	More expensive, More complex to use, More power consumption	6

5) Nutrient Level Sensor

Based on Table 3.6, We chose (**Nutrient Level Sensor with Transistor**), Because it is very cheap, easy to design and **we need four of it,** 4*0.25$ =**1$ <<** 4*6$ =**24$.**

Initially, we considered using a **Depth of Detection** Sensor to measure nutrient levels due to its simplicity and acceptable performance. However, after further evaluation, we decided to design and build our own custom sensor using transistors and resistors. This decision was based on several factors. While the Depth of Detection Sensor is functional, it comes at a higher cost, especially when multiple sensors are required. Our custom-built sensor offers a much more cost-effective solution, with acceptable accuracy for our application.

Table 3.6: Differences between Nutrient Level Sensor models:

Model	**Nutrient Level Sensor with Transistor**	**Depth of Detection**
Output	Single-level output (e.g., Low, High)	Multi-level output (typically 3 distinct levels) (e.g., Low, Medium, High)
Cost ($)	0.25	6

3.2.1.3 Display Screen

Based on Table 3.7, We chose (**TFT 2.8" ST7789**), Because it offers higher resolution, full-color support, and the ability to display graphics and images. Unlike the LCD 20x4, the TFT screen allows for a more modern and user-friendly experience, and importantly, the two displays are the same price.

Table 3.7: Differences between Display models:

Model	**TFT 2.8" ST7789**	**LCD 20x4**
Type	Full-color TFT LCD (Graphics, text, images)	Character LCD (Text only)
Physical Size	2.8 inches (true pixel display size)	~3.2 inches (depends on casing)
Resolution	240 × 320 pixels	20 columns × 4 rows (max 80 characters)
Color Support	Full 16-bit RGB (65k+ colors)	Monochrome (green or blue backlight)
Interface	SPI	I2C or 4/8-bit Parallel
Refresh Rate	Fast (suitable for animations/GUI)	Relatively slow
Power Consumption	Higher (due to backlight and pixel updates)	Very low
Cost ($)	10	10

3.2.1.4 Power Source

Table 3.8 compares various Battery types, evaluating them based on Capacity, Voltage, Rechargeable, Weight and Cost.

Based on Table 3.8, We chose (**12V 6S 8Ah Li-ion**), Because it provides higher capacity, lower weight, smaller size, greater efficiency, and a much longer cycle life compared to the 12V 7.2Ah Lead-Acid battery. Despite its slightly higher cost, the overall performance and long-term benefits make it a more suitable and efficient choice for our project.

Table 3.8: Differences between Power Source models:

Model	**12V 7.2Ah Lead-Acid**	**12V 6S 8Ah Li-ion**
Capacity (mAh)	7,200	8,000
Voltage (V)	12	12V (6 cells in series: 6S × 2V ≈ 12V)
Energy Density	Low (30–50 Wh/kg)	High (150–250 Wh/kg)
Rechargeable	Yes	Yes
Weight	Heavy (~2.5–3 kg)	Lightweight (~350 g)
Size	Larger	Small
Efficiency	70–85%	>90%
Cycle Life	~300–500 cycles	~1000–2000 cycles
Maintenance	May require periodic checks	Maintenance
Cost ($)	20	25

3.2.2 Software components

The software components of the Smart Aeroponic Farming System are designed to handle the integration of data collection, processing, and automation, ensuring optimal growing conditions for plants. These components enable the system to monitor, process, and react to environmental changes in real-time.

Here's a structured explanation of the software components in our project:

3.2.2.1 Software Application

The Software App will serve as the primary interface for users to interact with the system. It will allow users to monitor sensor data in real time, adjust environmental variables, and schedule tasks.

1) Web Framework Application

The following options were considered:

Table 3.9: Differences between Web Application Frameworks

Framework	Features	Pros.	Cons.
React.js [24]	Component-based, virtual DOM	Fast rendering, reusable components, large ecosystem	Requires additional libraries for state management
Angular [25]	Full framework, two-way data binding	Comprehensive toolset, enterprise-level support	Steeper learning curve, slower initial loads
Vue.js [26]	Lightweight, reactive data binding	Easy to learn, high performance for small apps	Smaller ecosystem, limited scalability

Based on Table 3.9, **We Chose Framework: React.js**

Reason: React.js was chosen for its high performance, modular design, and suitability for real-time IoT applications, enabling dynamic and responsive user interfaces.

2) Backend Development

The backend processes sensor data, communication with IoT devices, and serves the web application. The following backend technologies were evaluated:

Table 3.10: Differences between Backend Development

Framework	Features	Pros.	Cons.
Node.js [27]	Event-driven, non-blocking I/O	Ideal for real-time data, single language	Complex debugging for callback-heavy code
Django [28]	Full-stack framework, built-in ORM	Rapid development, secure	Overhead for small projects
Flask [29]	Lightweight, reactive data binding	Flexible and easy to customize	Requires setup for larger applications

Based on Table 3.10, **We Chose Backend: Node.js**

Reason: Node.js was selected for its efficient handling of real-time data streams and compatibility with IoT communication protocols like HTTPS.

3.2.2.2 Internet of Things (IoT)

The Internet of Things (IoT) integration in our system enables seamless, efficient communication between sensors, actuators, and the backend server to support real-time monitoring and responsive control. The architecture is built using a **hybrid communication model** optimized for performance and scalability:

IoT Communication Model: The ESP32 microcontroller collects sensor data and interacts with a custom Node.js backend using two protocols:

1) **REST API** is used primarily for:
    - Device registration upon startup.
    - Requesting initial configuration.
    - Occasional status updates if needed.
2) **WebSocket** is the main channel for:
    - Real-time bidirectional communication.
    - Continuous sensor data streaming.
    - Immediate control commands from the frontend.

This design ensures a reliable startup process via REST and continuous, low-latency communication using WebSocket.

IoT Sensors and Actuators: Various sensors (e.g., temperature, humidity, pH, EC, light, water level) are connected to the ESP32. The microcontroller processes the readings and sends them to the backend server. Based on preset thresholds or user commands from the frontend, the backend can activate actuators such as water pumps, fans, and lights.

IoT Backend Integration: To maintain full control and lower costs, we avoided third-party platforms like Firebase or AWS IoT Core. Instead, we implemented a custom Node.js server that handles:

- Real-time communication via WebSocket.
- Device setup and configuration via REST API.
- Flexible data flow with custom logic.
- Seamless connection to a React-based frontend.

This approach minimizes latency and supports real-time interaction while giving us full flexibility in design and deployment.

Table 3.11: Comparison of IoT Communication Methods

Protocol	Features	Pros.	Cons.
REST API	HTTP-based, stateless	Simple, widely supported	Not real-time unless polled
WebSocket	Persistent, full-duplex	Real-time, low latency	Slightly more complex to manage
MQTT [30]	Broker-based, pub/sub	Lightweight, scalable	Requires external broker

Based on Table 3.11, **We chosen Communication Strategy**

We adopted a hybrid communication model combining **REST API** and **WebSocket**:

- REST API is used only during ESP32 initialization and for configuration retrieval.
- WebSocket is used for all real-time data transmission and immediate control commands.

3.2.2.3 Data Storage and Processing

The system utilizes a custom backend (Node.js) for data handling, without relying on commercial cloud IoT services. This design supports both real-time operation and flexible backend processing tailored to the needs of our application.

Key Features:

- Real-time streaming via WebSocket for instant updates.
- Local or cloud-hosted SQL\NoSQL database for sensor logs.
- Custom REST and WebSocket APIs to connect with a React frontend.
- Full access to the system's logic and data structures.

Table 3.12: Comparison of Storage Options

Cloud Platform	Features	Pros.	Cons.
Firebase [33]	Real-time DB with SDKs	Quick setup, optimized for mobile	Limited control and scalability
AWS IoT Core [34]	Enterprise-grade cloud tools	Secure, powerful analytics	High cost and complexity
Custom Node.js + DB	REST + WebSocket + SQL	Full control, customizable, real-time	Requires setup and maintenance

Based on Table 3.12, **We chose a custom backend solution (Node.js + WebSocket + REST) with a local database. This ensures:**

- Complete control over data access and structure.
- Seamless integration with the React-based frontend.
- Real-time performance.
- Cost-effective deployment aligned with system needs.

3.2.2.4 SQL Database and Local Testing

To manage user authentication securely and efficiently, a relational SQL database is implemented. It stores essential user credentials and roles (e.g., admin, user). This allows only authorized users to access and manage their associated farms and data.

User Authentication with SQL Database

User login, registration, and role-based access control are handled through a structured MySQL database. This approach ensures scalability, data integrity, and secure storage of sensitive credentials (e.g., hashed passwords).

Table 3.13: Differences between SQL Databases for Authentication

Database	Features	Pros.	Cons.
MySQL [41]	Open-source, relational	Widely supported, fast, secure for auth	Requires setup and configuration
PostgreSQL [42]	Advanced SQL support	Powerful for complex queries	Slightly higher complexity
SQLite [43]	Lightweight, file-based	Simple, no server required	Not suitable for multi-user apps

Based on Table 3.13, **We Chose SQL Database: MySQL**

Reason: MySQL was chosen for its compatibility with authentication systems, ease of integration with Node.js, and ability to scale for multiple users and roles.

Local Development and Testing

Local development and testing require a lightweight server environment to simulate the backend. The following tools were considered:

Table 3.14: Differences between Local Testing Environments

Tool	Features	Pros.	Cons.
XAMPP [44]	Apache + MySQL + PHP stack	All-in-one, easy setup, cross-platform	Not suitable for production use
WAMP [45]	Windows-only stack	GUI for services	Limited to Windows OS
MAMP [46]	macOS/Linux-compatible stack	Good for Apple environments	Fewer customization options

Based on Table 3.14, **We Chose Testing Tool: XAMPP**

Reason: XAMPP was selected for its simplicity and comprehensive stack, which allows seamless testing of MySQL-based authentication workflows and backend integration on a local machine.

3.2.2.5 Development Tools

The ESP32 Microcontroller is programmed using the IDE to control sensors and actuators. The IDE provides a simple development environment for writing, debugging, and uploading the code to the microcontroller. It supports a wide variety of libraries and modules that are compatible with the ESP32, making it easier to interface with IoT devices and integrate sensor data into the system, The programming process includes: Sensor Integration, Actuator Control, Communication Protocols.

The development tools for programming the ESP32 microcontroller were evaluated as follows:

Table 3.15: Differences between Development Tools

Tool	Features	Pros.	Cons.
Arduino IDE [36]	Beginner-friendly, lightweight	Easy to use, extensive library support	Limited debugging capabilities
PlatformIO [37]	Advanced, modular	Supports large-scale projects	Steeper learning curve
Eclipse (ESP32) [38]	Professional-grade IDE	Extensive debugging tools	Complex setup

Based on Table 3.10, **We Chose Development Tool: Arduino IDE**

Reason: Reason: Arduino IDE was selected for its simplicity and compatibility, providing a straightforward solution for interfacing sensors and actuators with the ESP32.

3.3 Conceptual system description

Our system is designed to optimize plant growth in an aeroponic environment by continuously monitoring and controlling critical environmental parameters. At the core of the system is an ESP32 microcontroller, connected to multiple sensors and actuators to ensure ideal conditions for plant health. The overall architecture is illustrated in the general system block diagram (Figure 3.1) and the multi-unit system configuration (Figure 3.2).

Temperature & Humidity Monitoring: A DHT11 sensor is used to continuously measure air temperature and humidity. If the temperature exceeds a predefined threshold, the ESP32 will send a real-time alert to the web application and activate the cooling fan to restore optimal conditions. This ensures timely intervention and automated response to overheating.

Nutrient Solution Monitoring & EC Control: The TDS sensor, responsible for monitoring the Electrical Conductivity (EC) of the nutrient solution. If the measured TDS value drops by more than 100 ppm below the predefined target level, the ESP32 initiates a precise nutrient rebalancing process:

Pump A is activated for 2 seconds, followed by a mixing cycle using the mixer motor.

Then, Pump B is activated for 2 seconds, followed by another mixing cycle.

The TDS value is re-measured. If it is still more than 100 ppm below the target, the sequence is repeated until the desired EC level is reached.

After each nutrient addition, the mixer motor is triggered to ensure uniform distribution within the solution.

pH Monitoring & Adjustment: The pH sensor continuously measures the acidity or alkalinity of the nutrient solution when the TDS sensor is turned off, to avoid electrical interference. If the pH drops below the desired value (indicating excess acidity), the system activates Pump 4 to dispense a basic solution, followed by a mixing cycle to ensure uniform distribution.

Conversely, if the pH is too high (indicating low acidity), Pump 3 is activated for 2 seconds, followed by a mixing cycle, and then a new pH reading is taken. If the pH remains outside the acceptable range, the process is repeated until the value stabilizes within the target range.

Light Intensity Management: An LDR (Light Dependent Resistor) is used to measure ambient light. If the system detects low light levels during scheduled light hours, it will automatically turn on the LED grow lights. During designated dark

periods, the LEDs remain off regardless of light readings to preserve the plants' circadian rhythm.

Nutrient Tank Level Monitoring: Four nutrient level sensors are installed in the reservoirs for each solution. If any tank is detected as low, the ESP32 will notify the user via the web application, ensuring timely refilling and uninterrupted operation.

User Interface and Alerts: The system is integrated with a web application for remote monitoring and control. Users receive real-time notifications in the following cases:

Temperature or humidity out of range, Low nutrient levels in tanks, pH or EC imbalances. All interactions with the system, including monitoring and setting adjustments, are performed exclusively through the web application. The user interface is fully accessible via web browsers, as illustrated in the system conceptual diagram (Figure 3.4).

Display Output: All key parameters — including pH, EC (TDS), temperature, and humidity — are displayed locally on a TFT 2.8" ST7789 screen, providing immediate visual feedback at the system location.

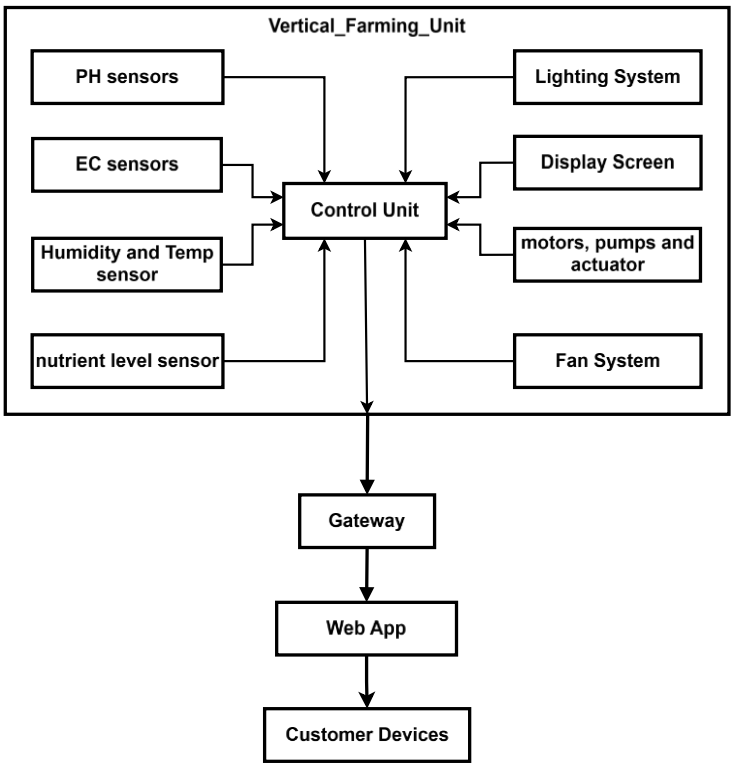


Figure 3.1 General block diagrams

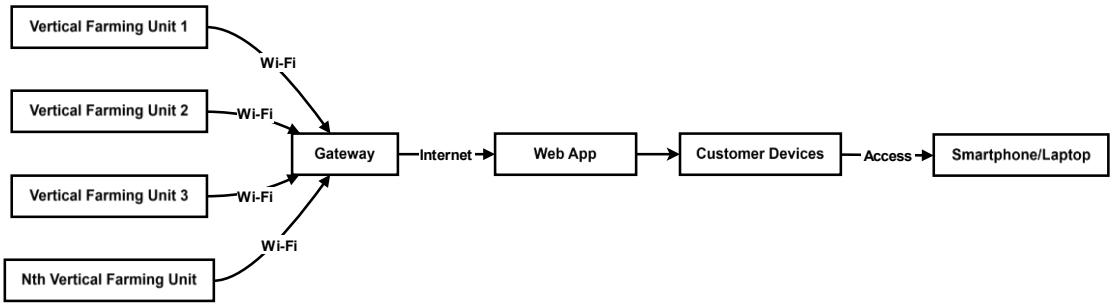


Figure 3.2 General block diagrams for multi-unit

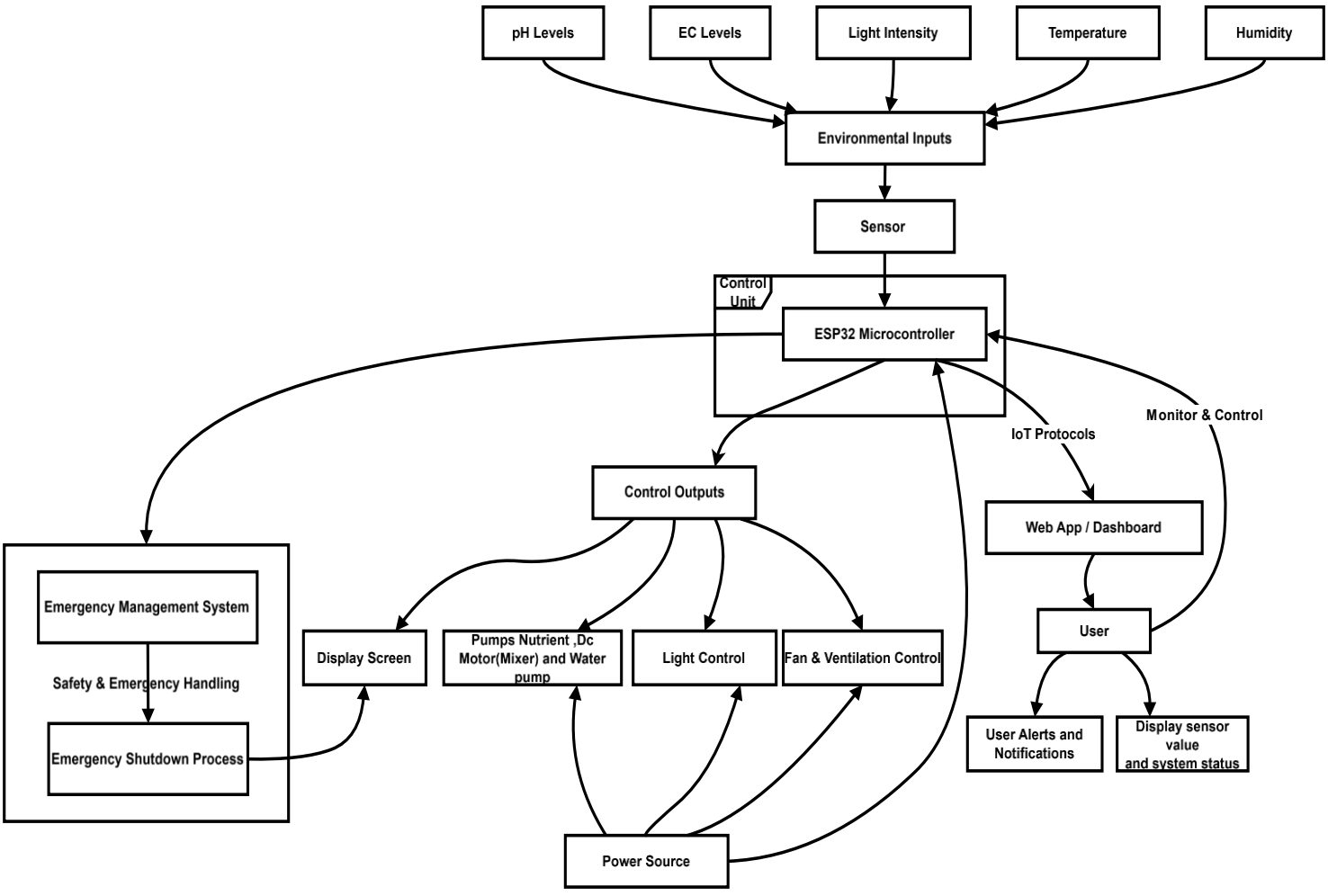


Figure 3.3 System conceptual diagram

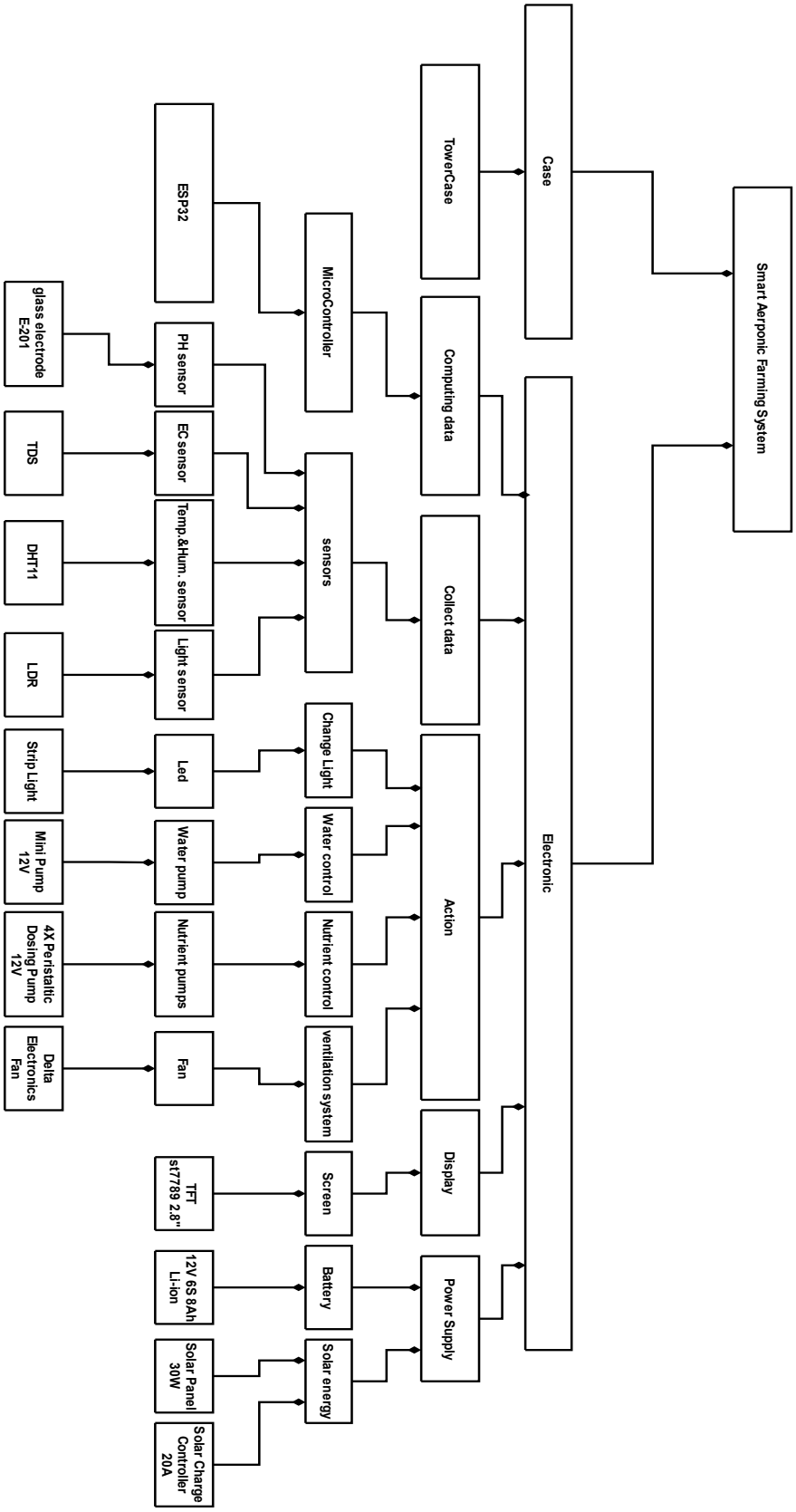


Figure 3.4 Zero-Level block diagram

3.4 Algorithms and methodologies

This section provides a detailed explanation of the algorithms and methodologies used to operate and manage the smart aeroponic farming system. It includes flowcharts, sequence diagrams, and use cases to represent system functionality and decision-making processes.

3.4.1 Control Methodologies

The system relies on real-time data acquisition and responsive control strategies to maintain optimal environmental and nutrient conditions for plant growth. The ESP32 microcontroller serves as the decision-making unit, executing a closed-loop control algorithm based on sensor input and predefined thresholds.

Methodologies Steps:

1. **System Initialization:**
   - Power up the ESP32 microcontroller.
   - Set default thresholds for temperature, humidity, pH, EC, and light.
   - Initialize sensors (DHT11, TDS, glass electrode (PH), LDR and level sensors).

2. **Sensor Data Acquisition (Periodic Loop):**
   - Read temperature and humidity using the DHT11.
   - Read TDS (EC) value.
   - Read pH sensor value when TDS (EC) is idle.
   - Read ambient light from LDR.
   - Check nutrient levels using the level sensors.

3. **Analyze Data and Decision Making:**
   - Compare the sensor readings to predefined thresholds.
   - Temperature:
     - If above threshold → Activate cooling fan.
   - TDS/EC Level:
     - If value is >100 ppm below target:
     - Activate Pump A (2s) → Mix → Pump B (2s) → Mix → Re-read TDS.
     - Repeat sequence until target EC is reached.
   - pH Level:
     - Read only when TDS sensor is OFF:
     - If pH < threshold → Activate Pump 4 (base solution (2s)) → Mix → Re-read.
     - If pH > threshold → Activate Pump 3 (acid solution (2s)) → Mix → Re-read.
     - Repeat adjustments until pH is within range.

- Light Intensity:
  - During scheduled light hours: If ambient light is low → Turn ON LED grow lights.
  - During scheduled dark hours: Keep LEDs OFF regardless of LDR value.
- Nutrient Tank Levels:
  - If any tank is low → Notify user via web application.

4. **Actuation and Feedback**:
- Activate corresponding devices (fan, pumps, mixer, LEDs) based on above decisions.
- Ensure proper timing and delay between each action to avoid electrical interference and ensure accuracy.

5. **Display and Logging:**
- Update local TFT 2.8" ST7789 display with real-time values of: Temperature, Humidity, pH and EC.
- Send all readings and alerts to the cloud-based web application for:
  - Real-time remote monitoring.
  - User alerts (e.g., temperature out of range, low tanks, pH/EC imbalance).
6. **Loop Execution**:
- Continuously repeat steps 2–5 in a timed loop with appropriate delay and priority scheduling to ensure optimal conditions.

3.4.2 Control Algorithms

The system uses real-time sensor data and closed-loop control to maintain optimal growth conditions in the smart aeroponic system.

Pseudocode:

1) **System Initialization:**

```
START

// System Initialization
Initialize ESP32
Set default thresholds for Temperature, Humidity, pH, EC, Light
Initialize all sensors and actuators

```

Figure 3.5 System Initialization Algorithm

2) Sensor Data Acquisition:

```
LOOP (Continuous Monitoring)

    // Sensor Data Acquisition
    Read Temperature and Humidity from DHT11
    Read EC from TDS sensor
    Disable TDS sensor
    Read pH from pH sensor
    Enable TDS sensor
    Read Light intensity from LDR
    Check nutrient tank levels
```

Figure 3.6 Sensor Data Acquisition Algorithm

3) Data Analysis and Control Decisions:

3.1 Temperature Control

```
// Decision Making and Control

IF Temperature > Threshold THEN
    Activate Cooling Fan
ELSE
    Deactivate Cooling Fan
END IF
```

Figure 3.7 Temperature Control Algorithm

3.2 EC (TDS) Control

```

WHILE EC < Target EC - 100 DO
    Activate Pump A for 2 seconds
    Activate Mixer briefly
    Activate Pump B for 2 seconds
    Activate Mixer briefly
    Read EC again
END WHILE

```

Figure 3.8 EC (TDS) Control Algorithm

3.3 pH Control

```
IF pH < Lower Threshold THEN
    Activate Pump 4 (Base) for 2 seconds
    Activate Mixer
    Read pH again
ELSE IF pH > Upper Threshold THEN
    Activate Pump 3 (Acid) for 2 seconds
    Activate Mixer
    Read pH again
END IF

```

Figure 3.9 pH Control Algorithm

3.4 Light Control

```
IF CurrentTime IN Light Schedule THEN
    IF Light Intensity < Threshold THEN
        Turn ON Grow Lights
    ELSE
        Turn OFF Grow Lights
    END IF
ELSE
    Turn OFF Grow Lights
END IF
```

Figure 3.10 Light Control Algorithm

3.5 Nutrient Tank Level Monitoring

```
    IF Any Nutrient Tank is Low THEN
        Notify User via Web App
    END IF
```

Figure 3.11 Nutrient Tank Level Monitoring Algorithm

4) **Display:**

```
    // Display
    Update TFT Display with Temp, Humidity, pH, EC
    Send all readings and alerts website
```

Figure 3.12 Display Algorithm

5) **Loop Execution:**

```
    // Wait before next cycle
    Delay (e.g., 5 seconds)

END LOOP
```

Figure 3.13 Loop Execution Algorithm

Main Control Algorithm Pseudocode:

```
START
// System Initialization
Initialize ESP32
Set default thresholds for Temperature, Humidity, pH, EC, Light
Initialize all sensors and actuators
LOOP (Continuous Monitoring)
    // Sensor Data Acquisition
    Read Temperature and Humidity from DHT11
    Read EC from TDS sensor
    Disable TDS sensor
    Read pH from pH sensor
    Enable TDS sensor
    Read Light intensity from LDR
    Check nutrient tank levels
    // Decision Making and Control
    IF Temperature > Threshold THEN
        Activate Cooling Fan
    ELSE
        Deactivate Cooling Fan
    END IF
    WHILE EC < Target EC - 100 DO
        Activate Pump A for 2 seconds
        Activate Mixer briefly
        Activate Pump B for 2 seconds
        Activate Mixer briefly
        Read EC again
    END WHILE
    IF pH < Lower Threshold THEN
        Activate Pump 4 (Base) for 2 seconds
        Activate Mixer
        Read pH again
    ELSE IF pH > Upper Threshold THEN
        Activate Pump 3 (Acid) for 2 seconds
        Activate Mixer
        Read pH again
    END IF
    IF CurrentTime IN Light Schedule THEN
        IF Light Intensity < Threshold THEN
            Turn ON Grow Lights
        ELSE
            Turn OFF Grow Lights
        END IF
    ELSE
        Turn OFF Grow Lights
    END IF
    IF Any Nutrient Tank is Low THEN
        Notify User via Web App
    END IF
    // Display
    Update TFT Display with Temp, Humidity, pH, EC
    Send all readings and alerts website
    // Wait before next cycle
    Delay (e.g., 5 seconds)
END LOOP
```

Figure 3.14 Main Control Algorithm Pseudocode

➢ Flowchart diagram:

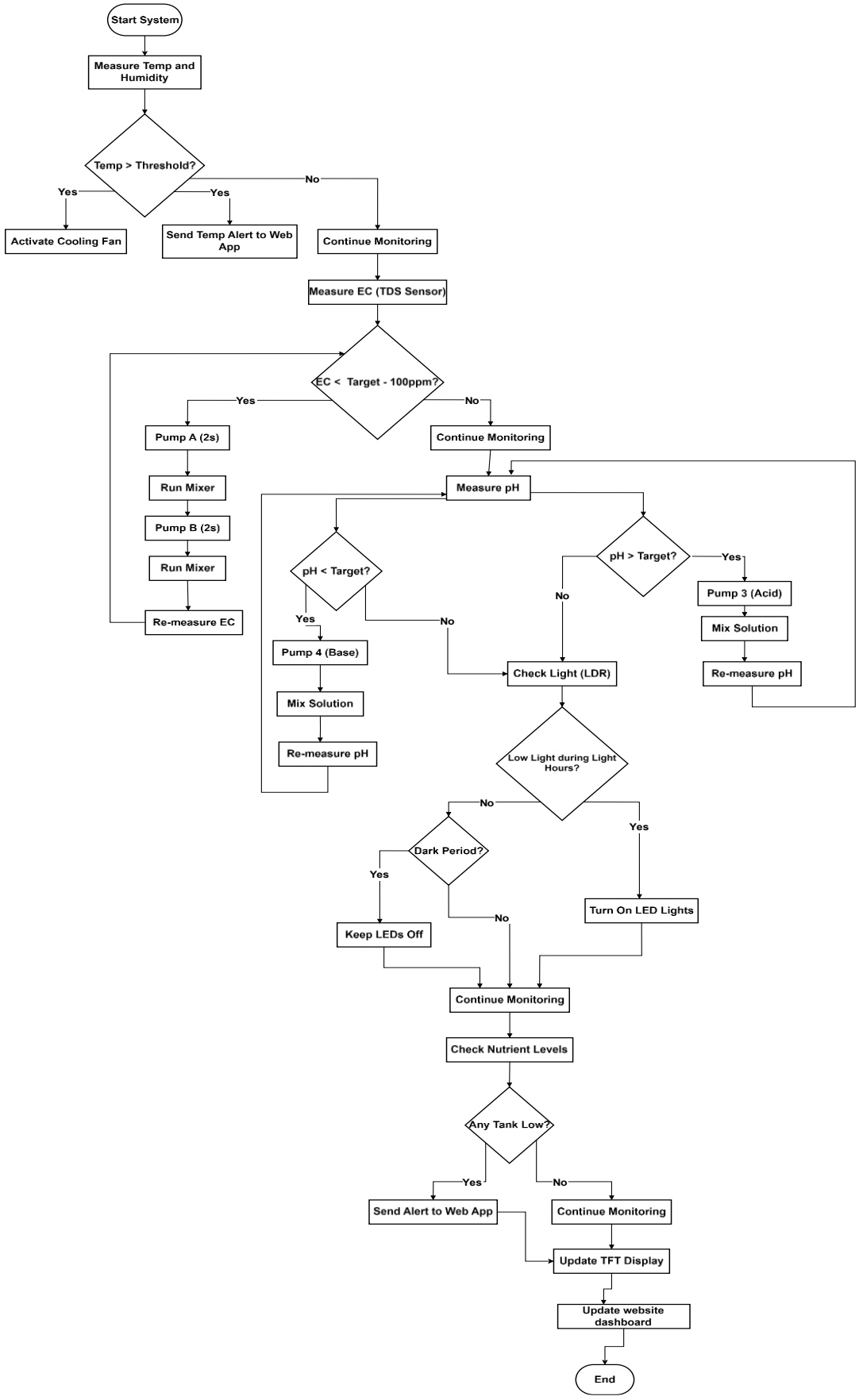


Figure 3.15 Flowchart diagram

➢ Sequence diagram

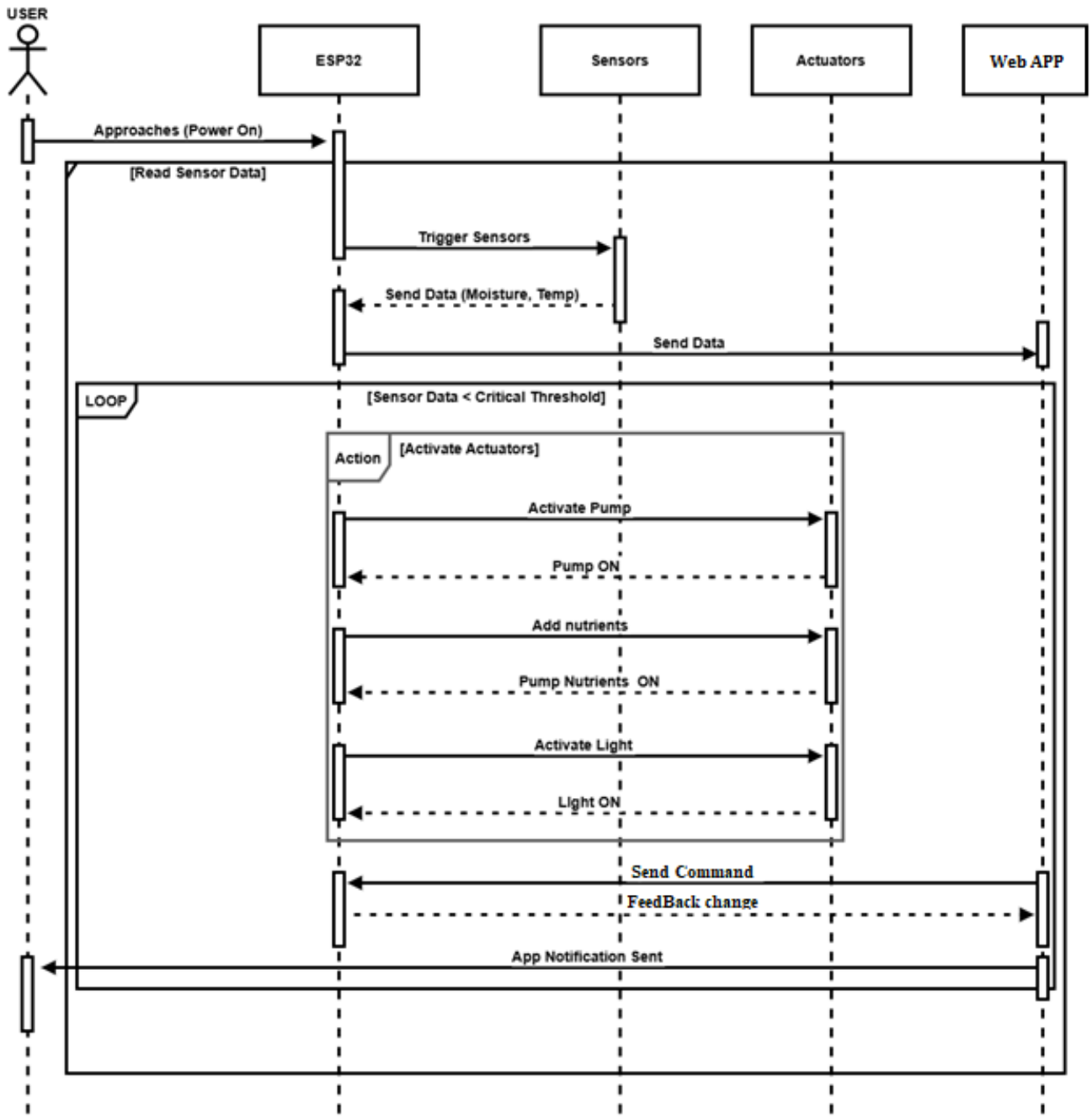


Figure 3.16 Sequence diagram

➢ Use Case diagram

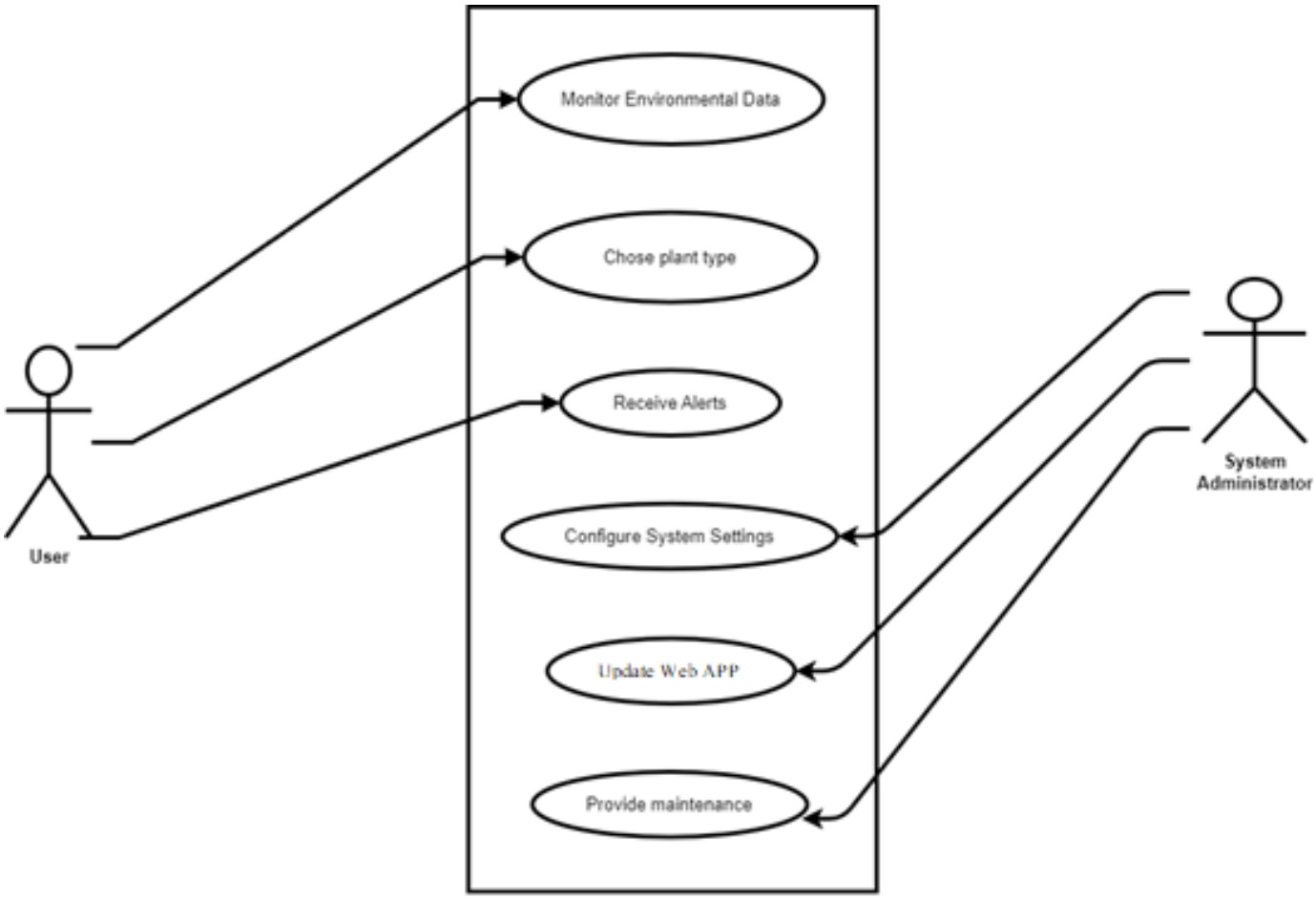


Figure 3.17 Use Case diagram

3.5 Schematic diagrams

DHT11 Sensor

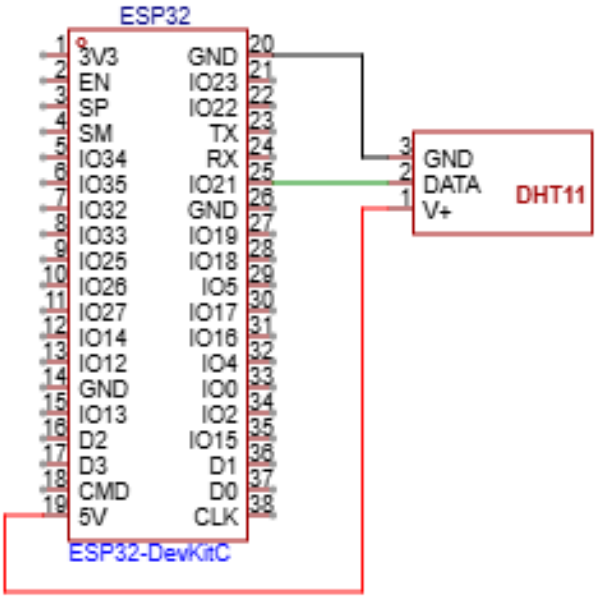


Figure 3.18 DHT11 Sensor Schematic diagram

PH Sensor

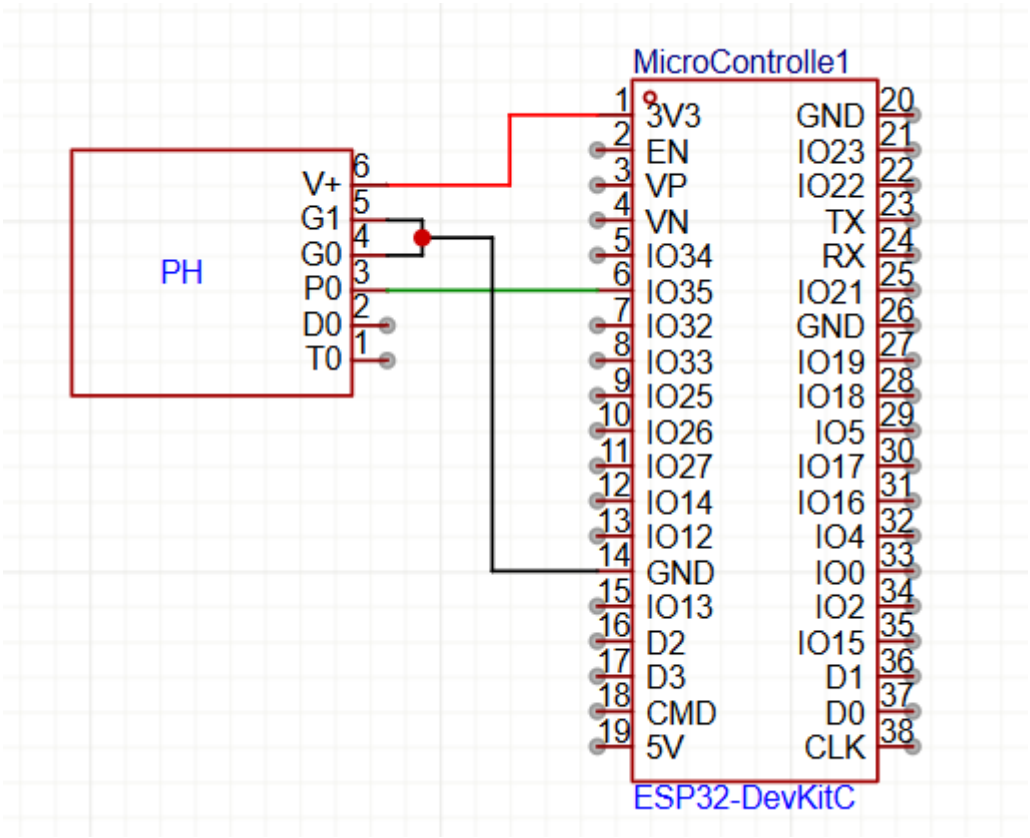


Figure 3.19 PH Sensor Schematic diagram

TDS Sensor

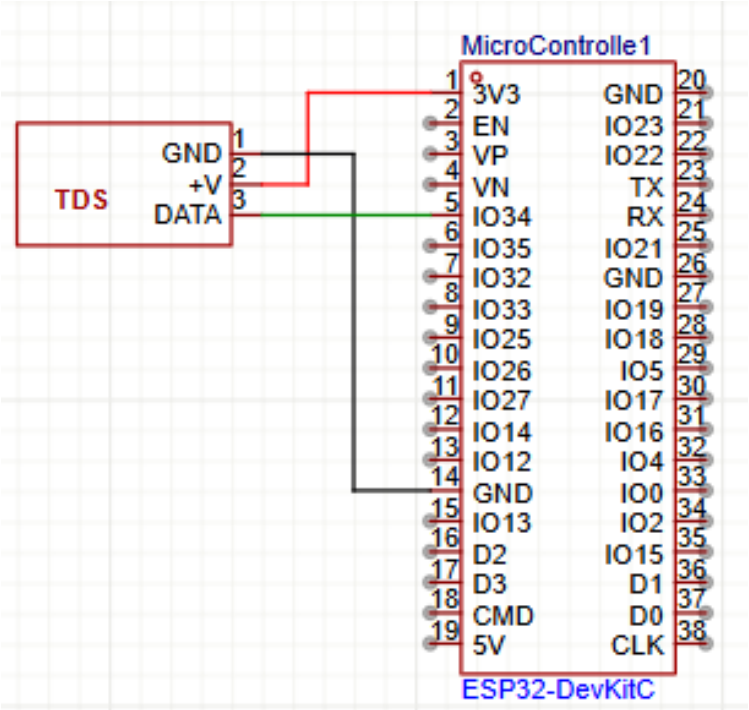


Figure 3.20 TDS Sensor Schematic diagram

LDR Sensor

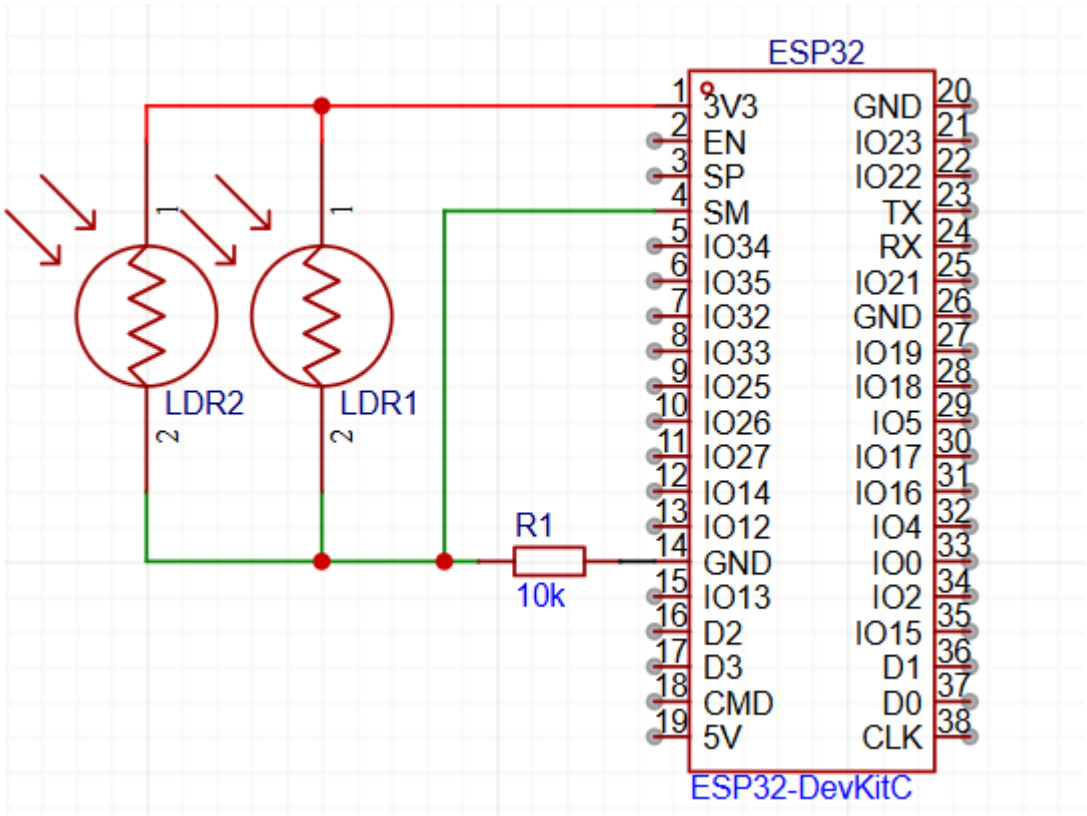


Figure 3.21 LDR Sensor Schematic diagram

Nutrient Level Sensor

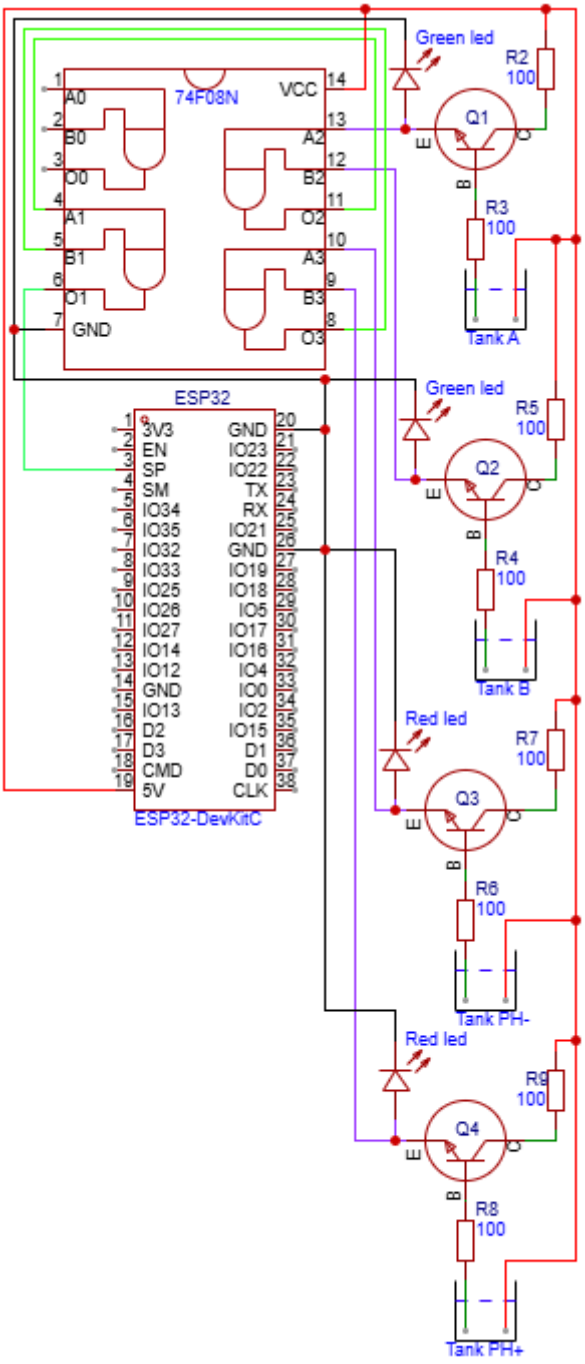


Figure 3.22 Nutrient Level Sensor Schematic diagram

Screen TFT 2.8" ST7789

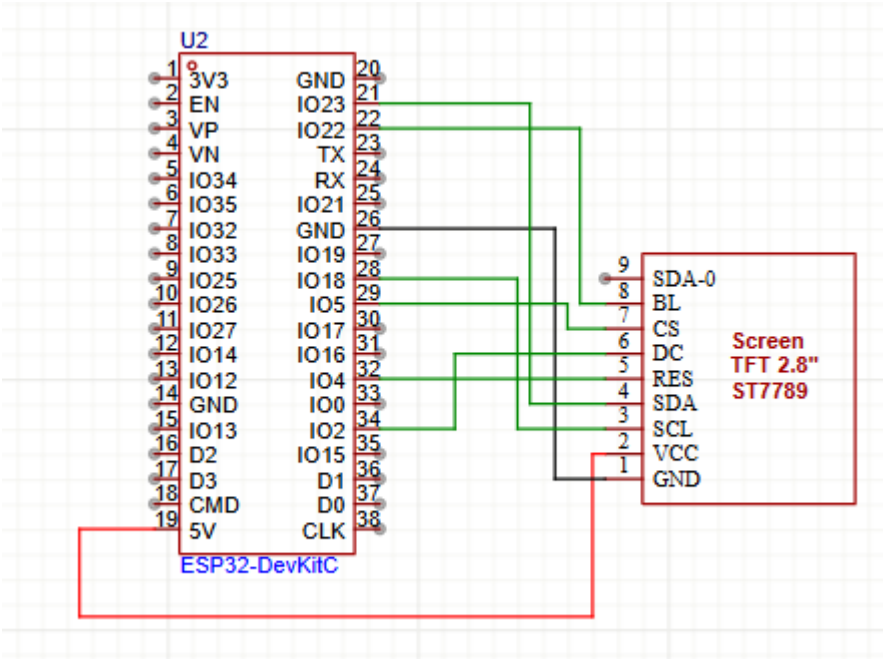


Figure 3.23 TFT 2.8" ST7789 Schematic diagram

First Relay with Nutrient Pumps

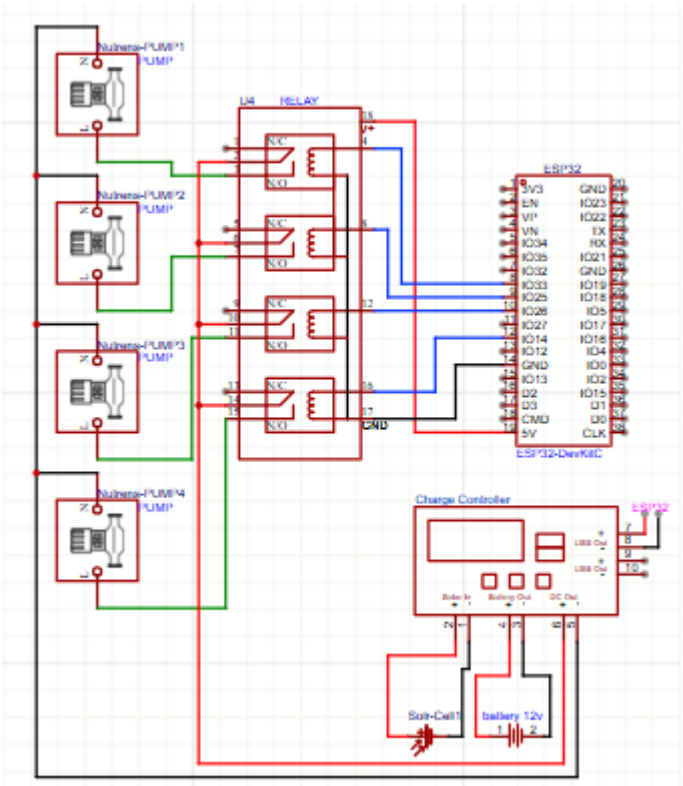


Figure 3.24 First Relay with Nutrient Pumps Schematic diagram

Second Relay with Water Pump, Fan, DC Motor (Mixer) and strip Light

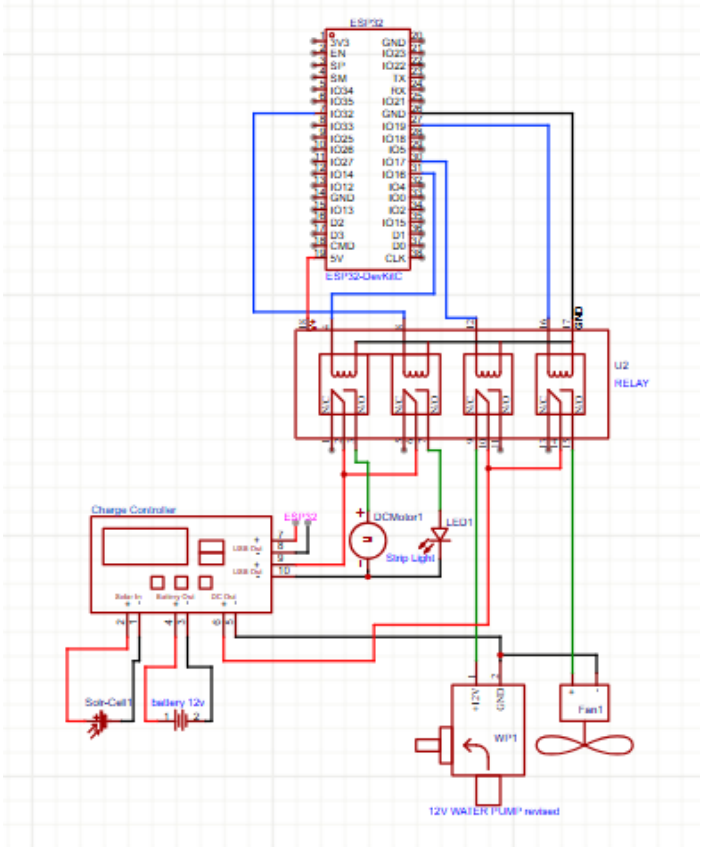


Figure 3.25 Relay and Nutrient Pumps Schematic diagram

Power Source

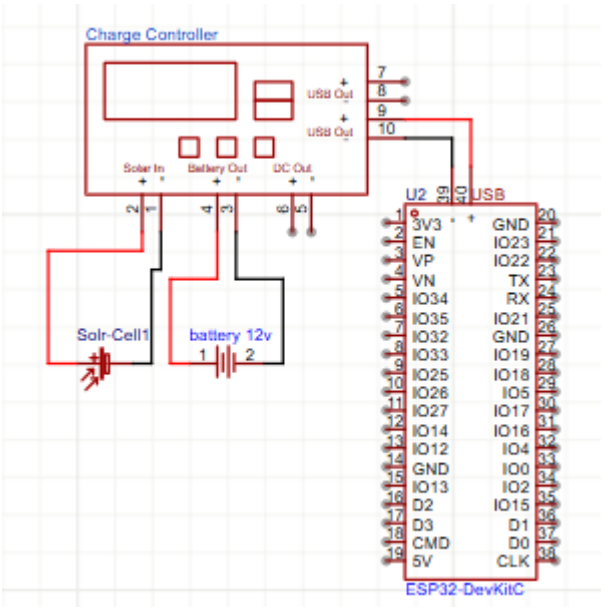


Figure 3.26 Power Source Schematic diagram

System Schematic Diagram

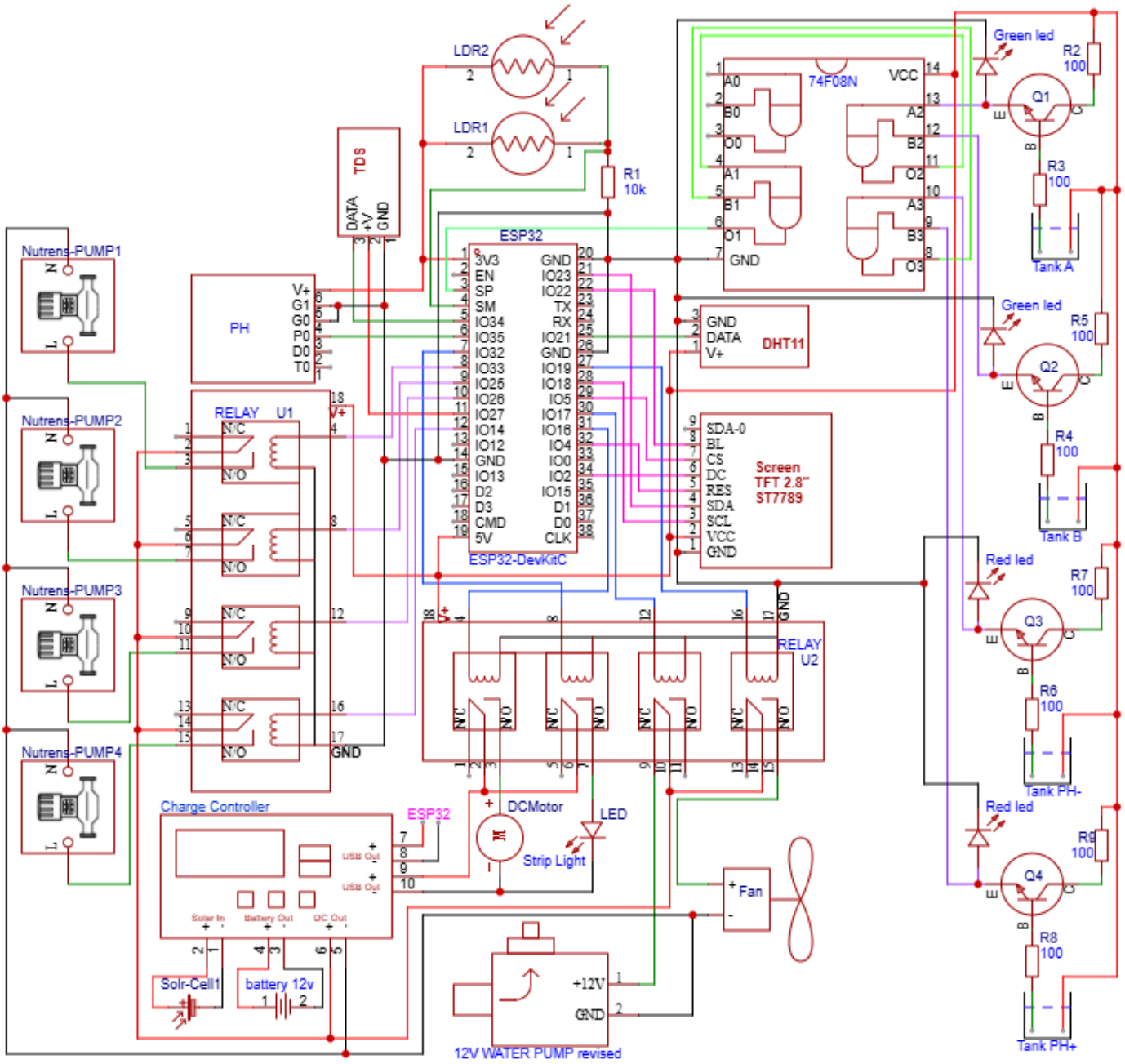


Figure 3.27 System Schematic diagram

3.6 Summary

This section outlines the design process for the smart aeroponic farming system, emphasizing the choice of components based on efficiency and cost, the conceptual architecture, and the methodologies employed. The combination of hardware and software aims to create an automated, scalable, and user-friendly farming solution.

Chapter 4

Implementation and testing

4.1 Implementation Issues

During the development and integration of the Smart Aeroponic Farming System, various implementation details emerged, ranging from building a prototype and assembling the necessary hardware components to integrating the software. Below is a comprehensive outline of the most significant implementation aspects:

4.1.1 Hardware implementation

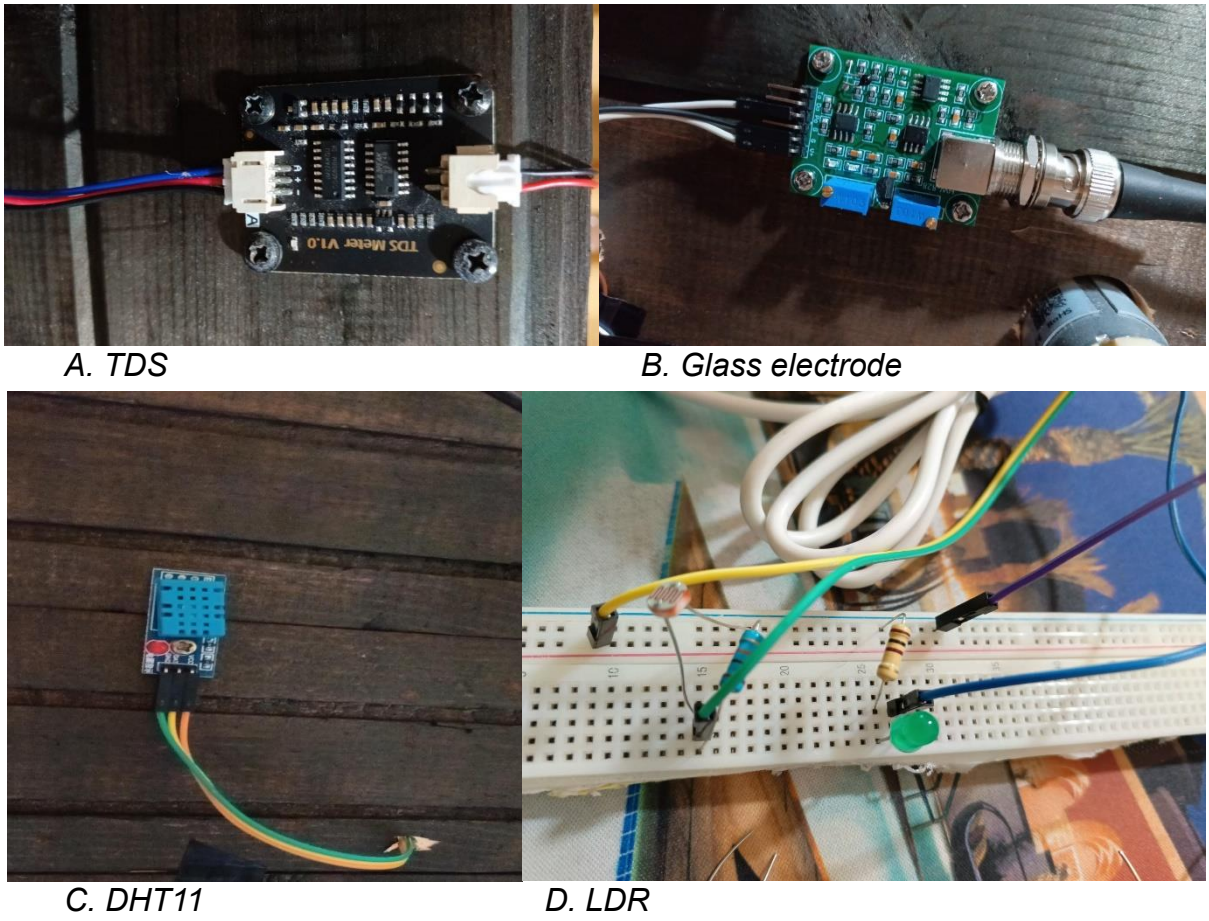
4.1.1.1 Prototype setup

The prototype was built using wood to simulate the structure of a greenhouse. It was designed to measure the fan's temperature and control the cooling system (the fan), as well as to provide a stable platform for mounting sensors, pumps, and other components. However, in the actual implementation, all of these elements should be integrated into the Tower Case Farming system.

Figure 4.1 Prototype setup

4.1.1.2 Sensor Integration:

The system includes a range of sensors (pH, TDS, DHT11, and LDR) that were physically mounted on the aeroponic unit and electrically connected to the ESP32 microcontroller. Each sensor was calibrated individually to ensure accuracy in data collection. The pH and TDS sensors required analog input and were positioned to avoid physical interference or overlapping signals.



A. TDS *B. Glass electrode*

C. DHT11 *D. LDR*

Figure 4.2 Sensor Integration

4.1.1.3 Relay Module Configuration:

A multi-channel relay module was deployed to control several electrical components. In this setup, two relay modules were used to manage multiple devices such as pumps, fans, and lighting systems, enabling automated control based on sensor inputs and programmed conditions.

First Unit:

- A 12V water pump for water circulation
- A 12V cooling fan that activates at high temperature
- A 5V LED strip for low-light conditions
- A 5V mixing motor for nutrient solution blending

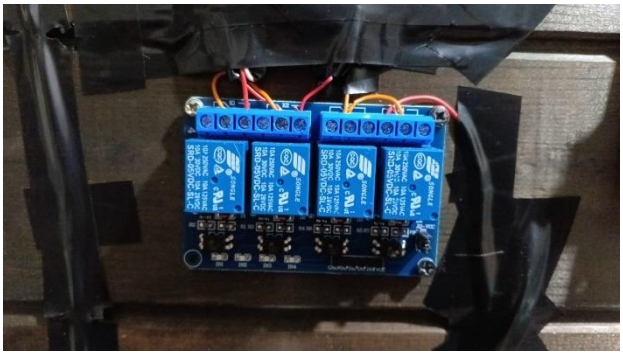


Figure 4.3 First Relay

Second Unit:

- Was dedicated to managing the four 12V nutrient pumps. These components were connected in a way that allows selective control based on sensor readings.

Figure 4.4 Second Relay

4.1.1.4 Nutrient Pumps:

Four pumps were added to deliver the required nutrient solutions, with automatic control managed by the ESP32 microcontroller. This setup ensures precise and timely distribution of nutrients based on the system's programming.

- See the image above (Figure 3.15 Relay and Nutrient Pumps Schematic diagram)

4.1.1.5 Mixer motor:

A mixer motor was added to operate after the nutrients are dispensed, ensuring proper mixing of the solution before it is delivered to the plants.

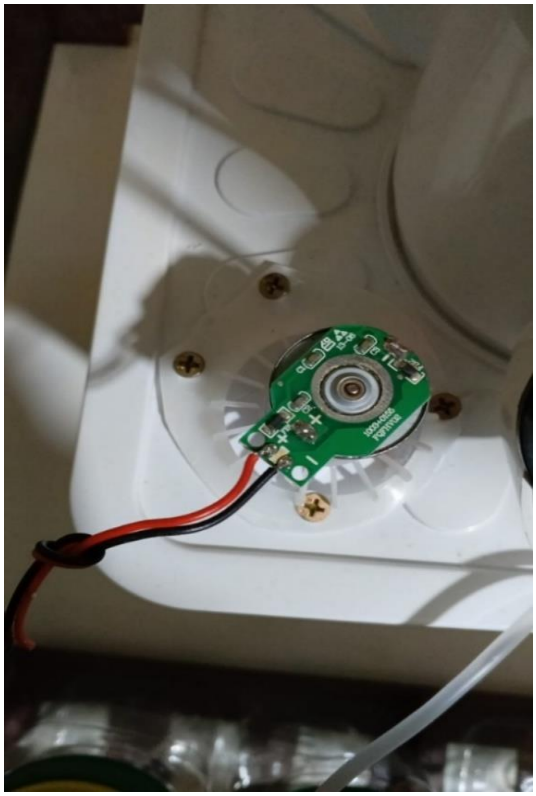


Figure 4.5 Mixer motor

4.1.1.6 Strip Light:

A strip light was added to provide supplemental lighting during scheduled periods when natural light is insufficient. This ensures that the plants receive adequate light for optimal growth according to the system's timing schedule.

Figure 4.6 Strip Light

4.1.1.7 Cooling Fan:

A 12V cooling fan was added and connected via relay module 2. The fan was programmed to turn on when the temperature exceeded a defined threshold, as measured by the DHT11 sensor.

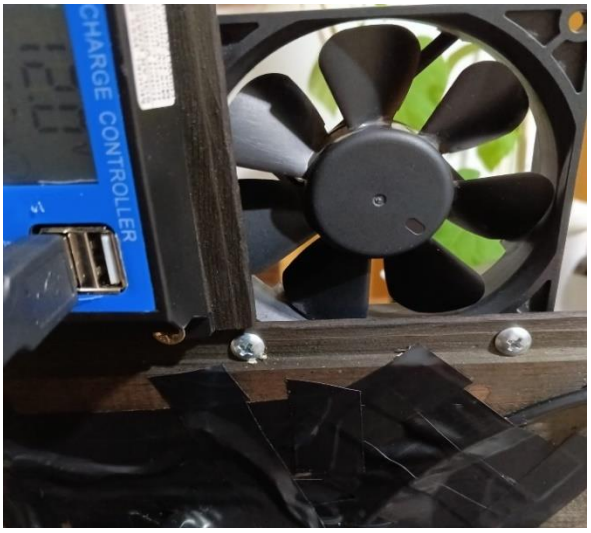


Figure 4.7 Cooling Fan

4.1.1.8 Water Pump:

The water pump is responsible for recirculating water within the tower. It is placed inside the tower structure and functions by drawing water from the bottom reservoir and pumping it to the top of the tower, allowing it to flow down and irrigate the plant roots effectively.

4.1.1.9 Power Management and Solar Setup:

The entire system is powered via a 12V rechargeable battery linked to a solar panel mounted at an approximate 45° angle. A solar charge controller ensures proper charging and discharge cycles, maintaining system sustainability.

Figure 4.8 Solar panel

Figure 4.9 Solar charge controller

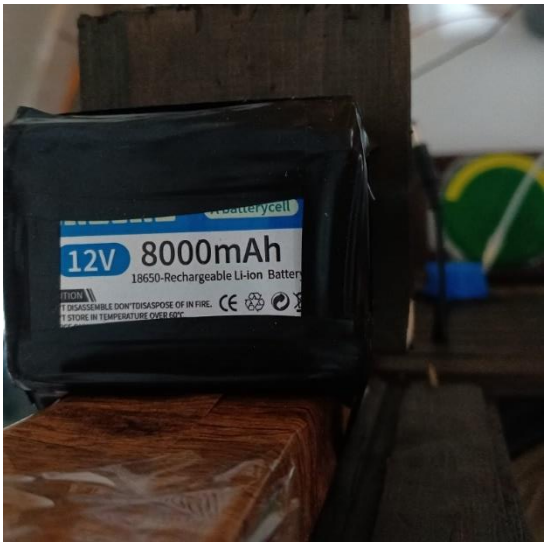


Figure 4.10 Battery

4.1.1.10 Display Screen:

A 2.8" TFT display (ST7789 ISP) was installed to show real-time system status and sensor readings directly on-site. This allows the user to stay connected with the system and monitor its performance without needing to access the web application.

Figure 4.11 TFT screen

4.1.1.11 PCB Assembly and Final Wiring:

After initial testing with jumper wires, all components were soldered onto a PCB to ensure a more reliable and compact build. The TFT screen was mounted on the frame to display live values such as pH, temperature, humidity, and TDS.

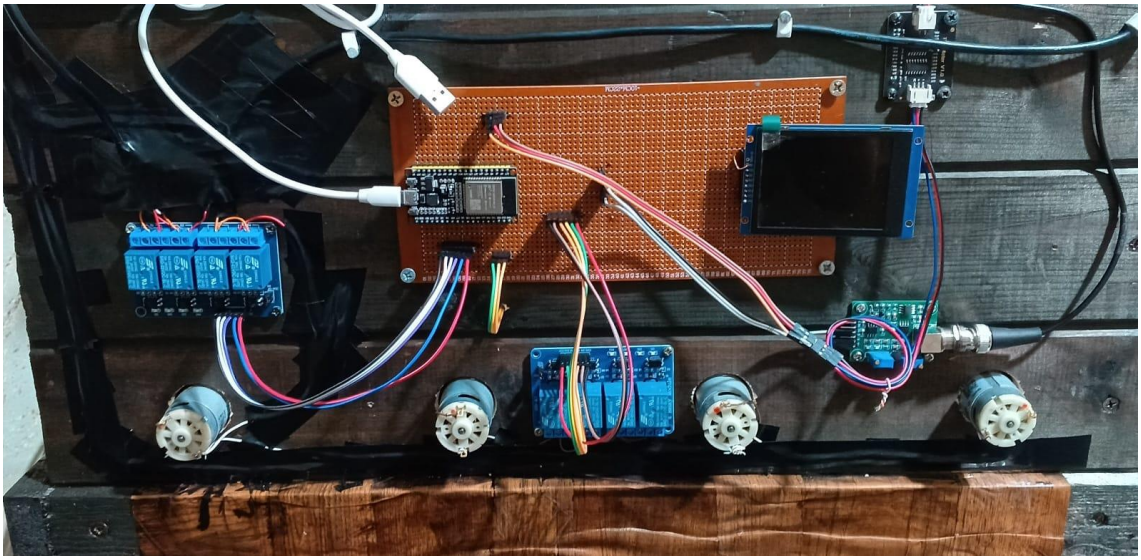


Figure 4.12 PCB Assembly

4.1.2 Software implementation

4.1.2.1 Web Application

A simple web application was developed to enable the user to continuously interact with the system from anywhere. This platform allows for remote monitoring and control, ensuring accessibility and convenience at all times.

4.1.2.1.1 Frontend

A. Home Page:

The Home Page provides a brief overview of our project and includes contact information, allowing users to learn about the system and reach out for further inquiries.

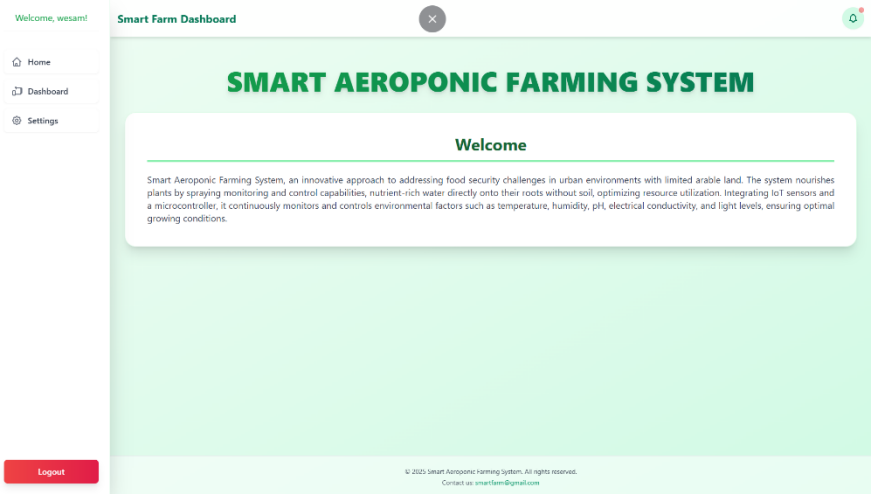


Figure 4.13 Home Page

B. Dashboard Page:

The Dashboard Page displays the current system status and sensor readings in real time. It keeps the user connected to the system and informed about its operation.

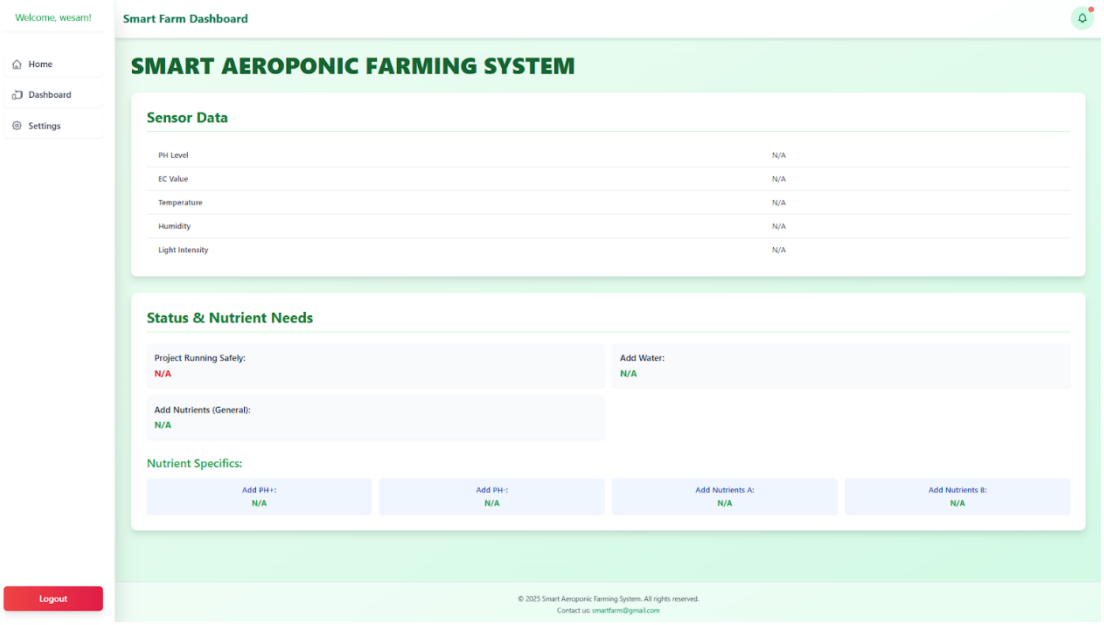


Figure 4.14 Dashboard Page

C. Sitting Page:

The Settings Page allows users to configure both the website and certain ESP32 settings, such as adjusting the screen brightness or turning the display on and off.

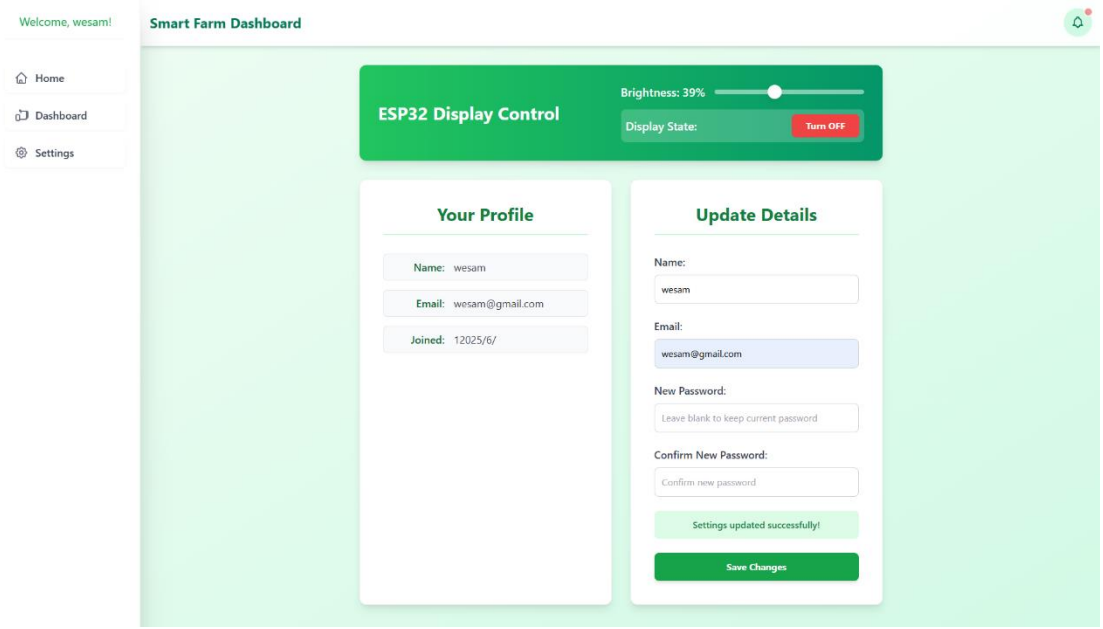


Figure 4.15 Sitting Page

4.1.2.1.2 Backend

The backend was developed using Node.js, serving as the central host and communication bridge between the ESP32 microcontroller and the web application. It handles both RESTful APIs and WebSocket connections, allowing for real-time communication, data exchange, and remote control of the system.

The backend is responsible for the following key functions:

- API Endpoints: Expose RESTful routes to receive and send data between the ESP32 and the client interface (web app).
- WebSocket Server: Maintains a live connection for real-time updates, such as instantly reflecting sensor changes or control actions.
- Data Validation & Processing: Ensures incoming data from the ESP32 is valid, structured, and safe to use.
- Error Handling & Logging: Captures and logs errors or disconnections to ensure system reliability and easier debugging.
- CORS & Security Configurations: Enables safe cross-origin requests from the frontend, while protecting against unauthorized access.

This backend setup plays a critical role in maintaining a stable and responsive smart farming system.

4.1.2.1.3 Database Management

The system uses MySQL as the relational database management system to store, manage, and retrieve critical information. The backend connects to this database to handle data persistence and application logic.

The database schema includes the following core tables:

- Users: Stores user information.
- Sensor readings: Stores real-time and historical data from sensors.
- settings: Holds configuration values for the system.

```
58  -- --------------------------------------------------------
59
60  --
61  -- Table structure for table `users`
62  --
63
64  CREATE TABLE `users` (
65    `id` int(11) NOT NULL,
66    `name` varchar(255) NOT NULL,
67    `email` varchar(255) NOT NULL,
68    `password` varchar(255) NOT NULL,
69    `registration_date` timestamp NOT NULL DEFAULT current_timestamp(),
70    `is_admin` tinyint(1) DEFAULT 0
71  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
72
```

Figure 4.16 Users Database

4.1.2.1.4 API Design and Integration

The system includes both RESTful APIs and WebSocket communication to ensure seamless interaction between the user interface, ESP32 microcontroller, and the backend.

4.1.2.2 ESP32 Firmware

The firmware running on the ESP32 microcontroller was developed using C++ within the Arduino IDE, chosen for its simplicity, extensive community support, and seamless compatibility with ESP32 libraries and hardware. The code follows the typical Arduino structure, with the setup() function used for initial configuration and the loop() function for continuous operation.

The firmware is modular and organized to handle sensor readings, actuator control, automation logic, and communication with the backend using both REST and WebSocket protocols.

4.1.2.2.1 Core Functionality and System Management

The system includes both RESTful APIs and WebSocket communication to ensure seamless interaction between the user interface, ESP32 microcontroller, and the backend.

- Initialization (setup()):
  - Serial monitor initialization for debugging.
  - GPIO configuration for sensors and actuators.
  - Initialization of connected peripherals (TFT screen, relays, pumps, sensors).
  - Wi-Fi connection and network setup.
  - WebSocket client setup for real-time communication.
  - Display of startup status on TFT screen.
- Main Loop (loop()):
  - Periodic sensor readings (temperature, humidity, TDS, pH, light, nutrient level).
  - Actuator control based on logic and thresholds.
  - Data transmission to the backend via WebSocket.
  - Display update with current values and system status.
  - Monitoring Wi-Fi connection and attempting reconnection if lost.

4.1.2.2.2 Sensor Integration and Data Acquisition

Table 4.1: For each sensor:

Sensor	Library	Pin	Reading Method	Notes
DHT11	DHT.h	GPIO 21	dht.readTemperature(), readHumidity()	Read every 5s
TDS Sensor	Custom calculation	GPIO 34	analogRead()	Disabled during pH read
pH Sensor	Custom/Analog	GPIO 35	analogRead() + Calibration	Isolated from TDS
LDR	None	GPIO 39	analogRead()	Read every 1s
Nutrient Level	Custom	GPIO 36	analogRead()	1 pins = 1 level

4.1.2.3 Development Environment and Tools

This section outlines the actual development tools, libraries, and environments used throughout the implementation phase, based on the design decisions detailed in earlier chapters.

- **ESP32 Firmware Development**
  Development Environment: Arduino IDE was used for writing, uploading, and debugging firmware code. Its simplicity and wide compatibility with ESP32 made it a suitable choice.
  Programming Language: C++ (with Arduino-style syntax).
- **Web Application Development**

- Frontend (User Interface):

  Framework: React.js.

- Backend (Server Logic):

  Runtime Environment: Node.js.

- Database

  Database Engine: MySQL.

- Local Web Testing Environment

  Tool: XAMPP.

- Testing Tools and Procedures

  Manual Testing: Sensor readings were validated through comparison with external measurement tools.

  Web interface functionalities (login, dashboard updates, settings) were manually tested.

4.1.3 Agricultural Implementation

As part of the ongoing agricultural implementation, bean seeds have been planted and germinated in a controlled environment. After successful germination, the seedlings are transferred to the aeroponic tower where they are currently in the growth phase. The plants are being monitored regularly, with adjustments made to nutrient levels, pH, and environmental conditions as needed to support healthy development. This real-time observation helps

evaluate the performance and effectiveness of the system under actual growing conditions.

Seed planting A

Seed planting B

Seed planting C

Seed planting D

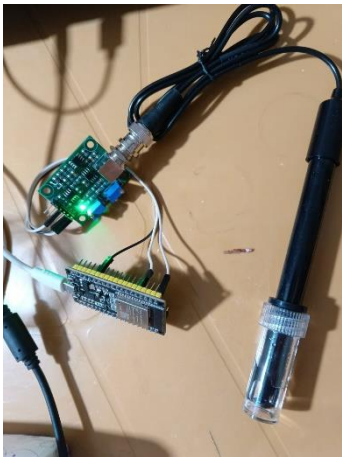
Figure 4.17 planting process

4.2 Implementation Challenges

During the implementation phase of the Smart Aeroponic Farming System, several hardware and software-related challenges were encountered. These challenges spanned sensor calibration, data accuracy, hardware integration, power management, and environmental conditions. The following is a detailed summary of the key issues and the practical solutions applied:

1) pH Sensor Non-Linearity

The pH sensor readings were not linearly correlated with voltage, which led to inaccurate values when using single-point calibration. To resolve this issue, three calibration points were used: pH 4, pH 7, and pH 10. A customized equation with conditional ranges was implemented to accurately convert voltage values into pH values. This approach was validated by testing the sensor with known buffer solutions and comparing the readings to expected values.



محلول له ph = 4
الناتج:
Raw ADC: 3897 | Voltage: 3.140 V

محلول له ph = 7
الناتج:
Raw ADC: 2964 | Voltage: 2.389 V

محلول له ph = 10
Raw ADC: 2350 | Voltage: 1.89439 V

Figure 4.18 PH Sensor and test values

Solution:

```
float ph(float voltage) {
  if(voltage < 0.1){
    return 0.0;
  }
  if(voltage > 3.14){
    return 7 + ((2.39 - voltage) / 0.258);
  }else if(voltage > 2.89){
    return 7 + ((2.39 - voltage) / 0.242);
  }else if(voltage > 2.64){
    return 7 + ((2.39 - voltage) / 0.218);
  }else if(voltage > 2.39){
    return 7 + ((2.39 - voltage) / 0.21);
  }else if(voltage > 2.21){
    return 7 + ((2.39 - voltage) / 0.19);
  }else if(voltage > 2.06){
    return 7 + ((2.39 - voltage) / 0.185);
  }else if(voltage > 1.70){
    return 7 + ((2.39 - voltage) / 0.175);
  }else{
    return 7 + ((2.39 - voltage) / 0.167);
  }
}
```

Figure 4.19 Code for solution

2) TDS Sensor Signal Fluctuations

The Total Dissolved Solids (TDS) sensor exhibited noisy readings due to analog signal instability. To mitigate this, a median filtering technique was implemented in code using the getMedianNum() function. This approach eliminated outlier values and ensured a more stable output across multiple samples.

3) pH and TDS Sensor Interference

A significant issue was observed when both the pH and TDS sensors operated simultaneously in the same nutrient reservoir. When powered at the same time, the sensors caused electrical interference, leading to inaccurate and unstable pH readings. This was due to the nature of analog signal overlap and cross-talk between the sensors.

Solution: The TDS sensor is temporarily disabled while taking pH measurements. This sequential sensor activation prevents interference and ensures accurate readings for both sensors.

4) Power Management and Voltage Separation

The system included components operating at both 12V (pumps, fan) and 5V (LED, mixer, ESP32). A clear power separation strategy was required to prevent overloading a single source. Two relays were used: Relay 1 for nutrient pumps (12V) , Relay 2 for water pump, fan (12V), LED, and mixer (5V).

Power was distributed via a solar charge controller connected to a 12V Li-ion battery, ensuring reliable supply under varying solar input.

5) Limited GPIO Availability on ESP32

Due to the large number of connected devices (four pumps, fan, mixer, LED, sensors, screen), the ESP32's GPIO pins were fully utilized. Efficient pin mapping and careful assignment were necessary to avoid conflicts. Digital and analog peripherals were distributed optimally, and unused pins were disabled to minimize noise.

6) Moisture-Proximity Risk to Electronics

The layout of the nutrient tanks and tubing system introduced a risk of water splashing or condensation near the electronics. While the frame was wooden and partially protected, the placement of the PCB and relays above the tank level was intentionally designed to minimize water exposure. Future improvements may include protective casing or waterproof shielding.

7) Solar Panel Positioning and Fixation

The solar panel required accurate angling to maximize energy capture. Based on sunlight direction and geographical considerations, the panel was mounted at an approximate 45° angle, which proved effective. It was fixed firmly using bolts and transparent mounts to maintain its position during operation.

Figure 4.20 Solar panel at 45° angle

4.3 Validation and Testing

To ensure the proper functionality of the Smart Aeroponic Farming System, each component and module was individually tested and validated through both manual and automated methods. The testing process focused on verifying sensor accuracy, actuator responses, power flow stability, and real-time monitoring capability via both the screen and the web interface.

4.3.1 Sensor Calibration and Verification

4.3.1.1 PH Sensor

The sensor was tested using three reference solutions (pH 4, 7, and 10). The voltage output from the amplifier was measured and mapped to the correct pH values using a segmented equation implemented in the code. The readings were printed to the serial monitor and compared with expected reference values. Accuracy was confirmed within ±0.5 pH units.

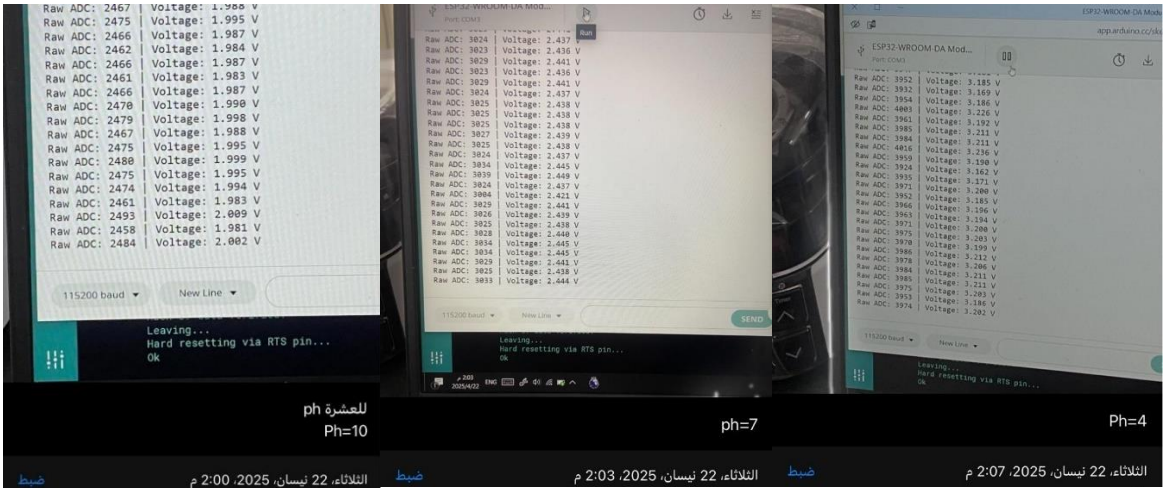


Figure 4.21 PH sensor testing and calibration

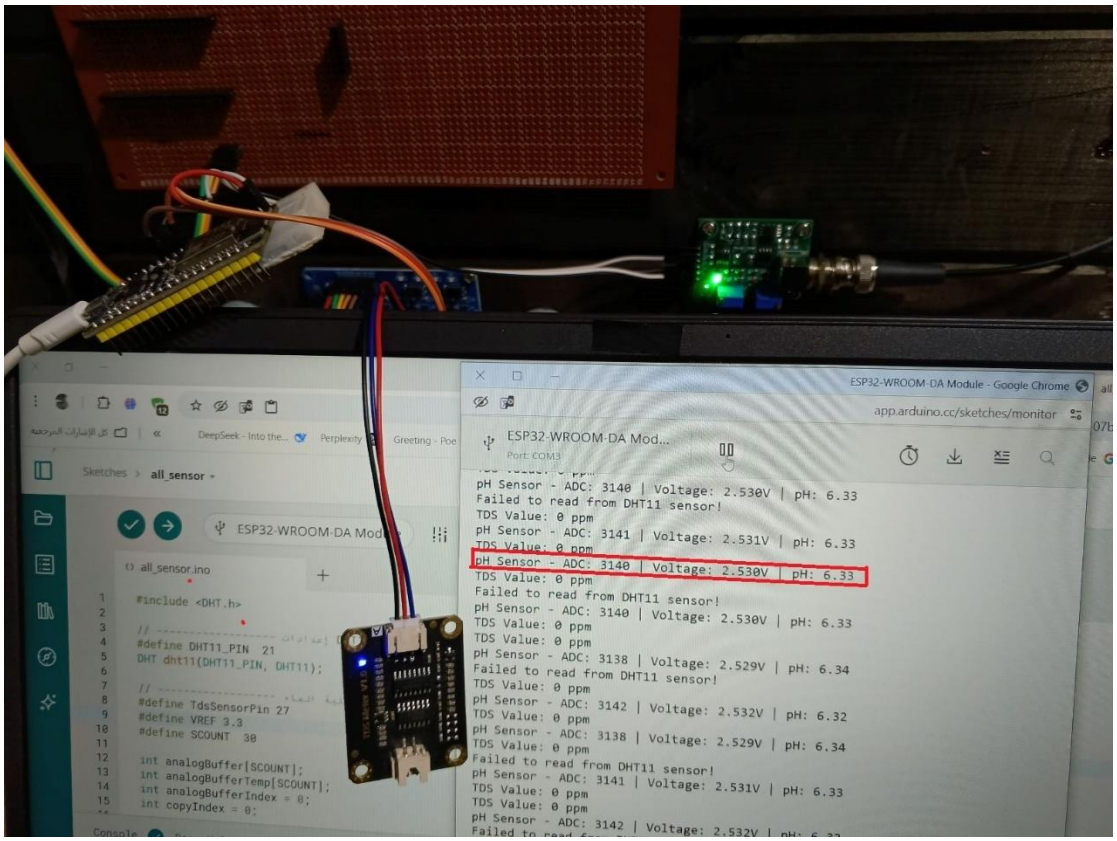


Figure 4.22 PH sensor after calibration

4.3.1.2 TDS Sensor

TDS values were tested using standard tap water and nutrient-rich water. The median filter produced stable readings, and values were verified through serial output. Sudden fluctuations were eliminated after implementing data smoothing.

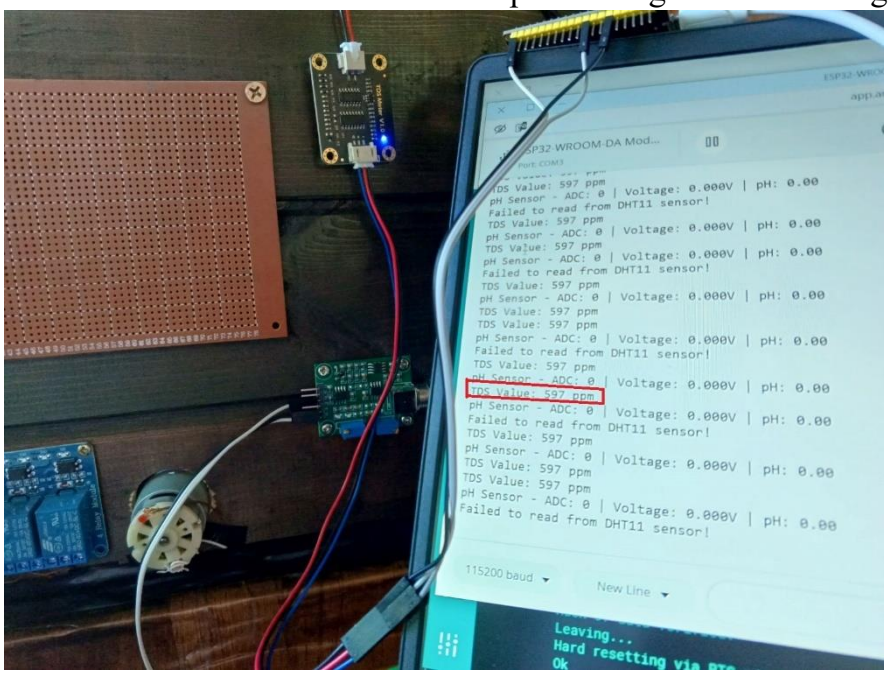


Figure 4.23 TDS sensor testing and calibration

4.3.1.3 DHT11 Sensor

The sensor was tested under various room conditions to ensure temperature and humidity values were consistent with ambient measurements. The fan was set to activate when the temperature exceeded a certain threshold (28°C), and this behavior was confirmed.

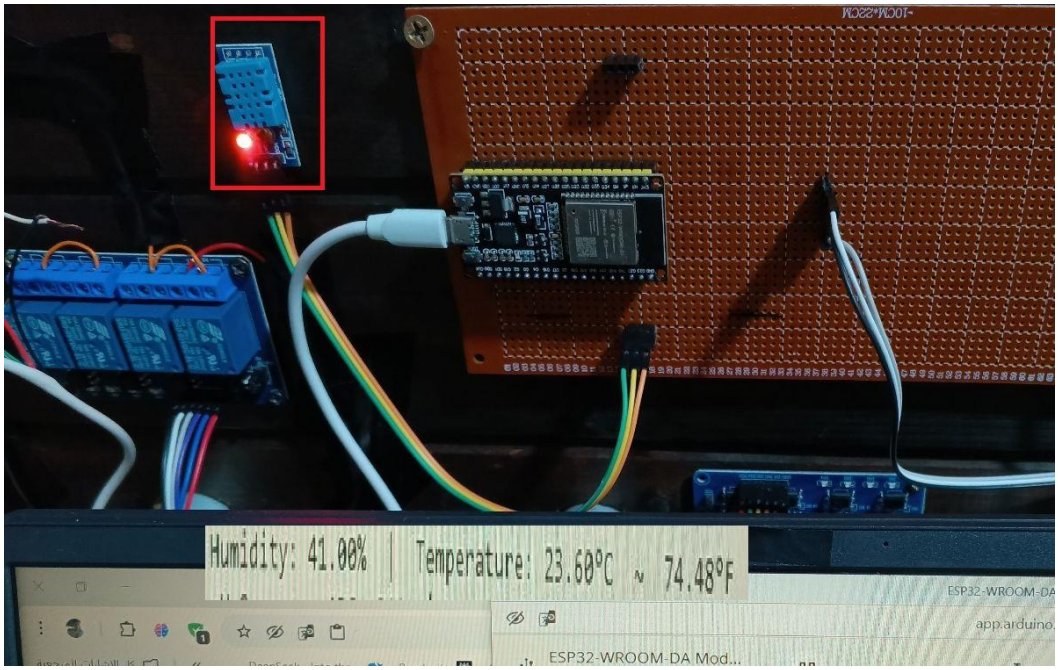


Figure 4.24 DHT11 sensor testing

4.3.1.4 LDR Sensor

To improve the accuracy of light level detection, two LDR (Light Dependent Resistor) sensors were connected in parallel to monitor ambient light from different angles. The goal was to ensure the grow lights would activate only when both sensors simultaneously detect low light conditions.

A logical condition was implemented in the firmware to trigger the LED grow lights only if both LDRs report darkness. This dual-sensor approach reduces false triggering caused by partial shadows or uneven lighting. The images below demonstrate that the lights activate only when both LDRs detect darkness.

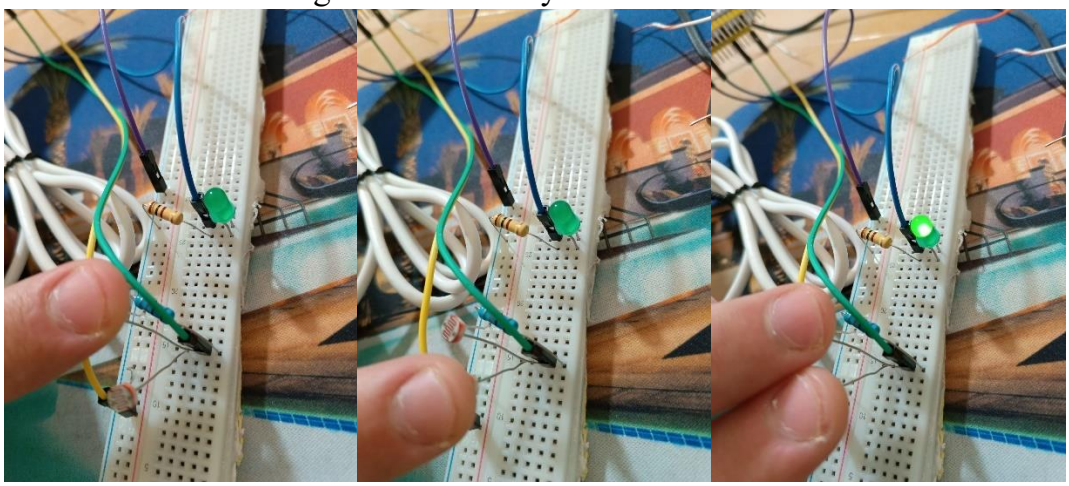


Figure 4.25 LDR sensor testing

4.3.1.5 Nutrient Level

Due to limited available GPIO pins on the ESP32, it was not feasible to connect each of the four nutrient tank sensors to a separate input.

The outputs of all four tanks were combined using a 74F08 AND gate, resulting in a single output line to the ESP32. Each tank is equipped with an indicator LED that turns off when the tank is empty. The combined AND logic ensures that if any tank is empty (its LED off), the output drops to approximately 0V — indicating a low level. This behavior was verified using a voltmeter.

When all tanks are full (LEDs on), the AND gate output reaches approximately 4V. To protect the ESP32 from over-voltage input in this state, a voltage divider circuit will be implemented to reduce the input voltage to a safe level ($\leq 3.3V$).

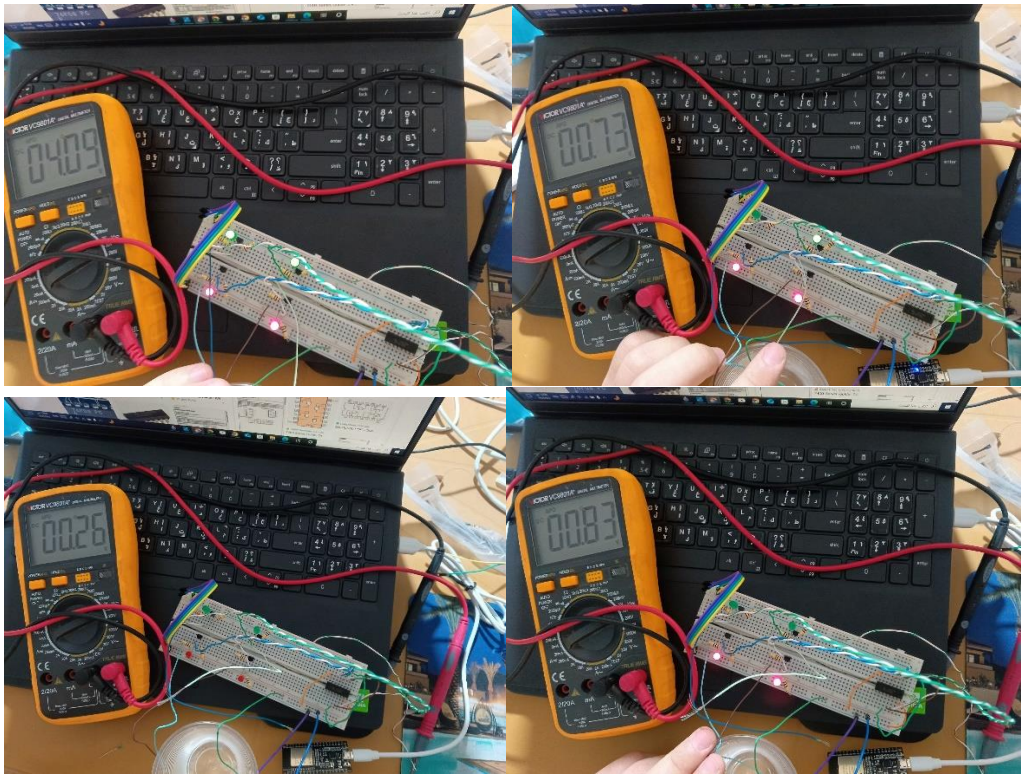


Figure 4.26 Nutrient Level circuit

4.3.2 Actuators and Outputs Testing

4.3.2.1 Nutrient Pumps (Relay 1)

Each of the four 12V pumps was manually triggered through the microcontroller, one at a time. Fluid flow was verified through transparent tubing from the nutrient jars to the central tower. No leakage or blockage was observed.

4.3.2.2 Water Pump + Mixer + Fan + LED (Relay 2)

Each device was activated separately through the ESP32 firmware to verify proper functionality and response. All components successfully responded to their control signals, confirming that the relay module can handle multiple outputs without conflict.

4.3.2.3 Frontend and Server Integration Testing

- The server-main.zip and frontend-main.zip applications were deployed and connected via WebSocket.
- The ESP32 sent periodic data updates to the backend.
- The web interface successfully displayed live values (pH, TDS, etc.) and received control commands.
- Synchronization was maintained between physical device readings and interface display.

4.3.2.4 Screen Display Verification with Sensors

The TFT display was programmed to show real-time values of:
pH, TDS, Temperature, Humidity.

During system operation, the screen updated regularly with no freezing or flickering, confirming successful SPI communication and data refresh logic triggering caused by partial shadows or uneven lighting. The images below demonstrate that the lights activate only when both LDRs detect darkness.

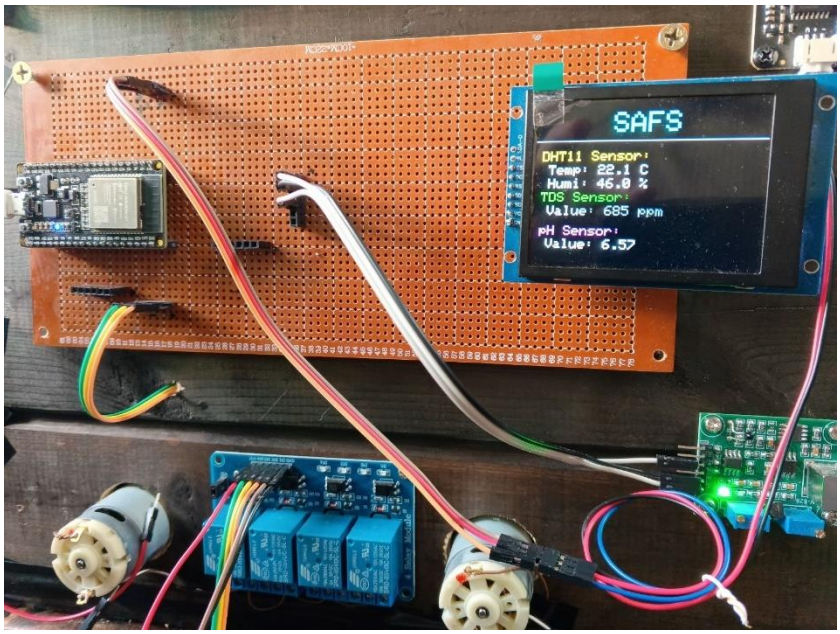


Figure 4.27 TFT display live reading

4.3.2.5 Power System Validation

- The solar panel and charge controller were monitored under sunlight conditions.
- Voltage levels were verified on the screen of the solar charge controller (~12.2V at full charge).
- All components functioned simultaneously using battery power.
- No overheating or current drop was detected, confirming load balance.

Figure 4.28 Solar charge controller testing

The system passed all validation tests under real-world conditions. The integration of hardware, sensors, relays, and interfaces was confirmed to be stable and responsive.

4.4 Summary

This chapter provided a comprehensive overview of the implementation and testing phases of the Smart Aeroponic Farming System. It began by highlighting critical implementation issues, including sensor calibration, power configuration, and hardware integration. The pH sensor presented a particular challenge due to its non-linear readings, which was resolved by introducing multiple sampling and calibration equations to ensure accuracy.

Although no major technical obstacles were encountered beyond sensor accuracy, careful attention was given to wiring, physical assembly, and reliable power distribution using a 12V battery and solar charging system. The testing phase was carried out through a combination of unit and integration testing. Sensors such as pH, TDS, LDR, and DHT11 were verified individually, while relay modules and pumps were validated in integrated conditions to confirm that all actuators responded correctly to real-time sensor data.

Overall, the system demonstrated consistent performance across all core functionalities: sensing, decision-making and actuation. This chapter concludes the hands-on implementation work and lays the foundation for the next chapter, which will discuss the results and performance analysis in more detail.

Chapter 5

RESULTS AND DISCUSSION

5.1 Detailed Analysis of the Results/Experiments

To evaluate the performance of the Smart Aeroponic Farming System, a series of experiments were conducted on each functional component separately and within the integrated environment. The analysis focused on real-time sensor readings, actuator responses, and overall system behavior under varying conditions.

5.1.1 Sensor Readings

The system includes pH, TDS, LDR, and DHT11 sensors, all connected to the ESP32 microcontroller and tested for consistency and reliability.

5.1.1.1 PH sensor

Due to its inherent non-linearity, the pH readings were initially unstable. This issue was resolved by implementing a three-point calibration method, where three known values were measured and used to generate appropriate equations for mapping analog signals to pH values. After calibration, the system provided stable and realistic pH values (e.g., between 5.5 and 6.5), suitable for nutrient-rich hydroponic environments.

5.1.1.2 TDS sensor

The TDS sensor was used to measure the concentration of dissolved solids in the nutrient solution. It produced accurate readings within expected ranges, allowing the system to trigger nutrient addition when values dropped below a defined threshold.

5.1.1.3 DHT11 sensor

The temperature and humidity sensor provided stable readings. The system used these values to activate the cooling fan when the ambient temperature exceeded 28°C.

5.1.1.4 LDR sensor

The LDR was used to detect ambient light levels. During low-light conditions (e.g., nighttime or a shaded environment), the sensor successfully triggered the LED lighting system to compensate for inadequate light.

5.1.2 Actuator Responses

The system includes multiple actuators connected via two 4-channel relay modules:

5.1.2.1 Nutrient Pumps (x4)

Each connected to a separate solution tank. They were activated based on TDS readings and operated independently through the first relay module.

5.1.2.2 Water Pump, Fan, LED Strip, and Mixer

Controlled by the second relay module. Their behavior was tied to respective sensor conditions (temperature, light level, ...etc).

All actuators responded correctly to sensor data. For example, when the TDS dropped, the corresponding nutrient pump activated. Similarly, when the ambient light dropped below a threshold, the LED strip was powered on.

5.1.2.3 System Integration Test

A full integration test was performed after validating all individual components. The system was run continuously, simulating both day and night conditions, and all modules functioned together seamlessly. Data was displayed on the TFT screen in real-time, and the ESP32 maintained stable operation across power cycles.

5.2 Error / success rate calculations

In order to assess the reliability and stability of the Smart Aeroponic Farming System, multiple tests were conducted on sensor accuracy and actuator responsiveness. The system's success rate was evaluated by comparing expected versus observed behavior during different environmental conditions.

5.2.1 Sensor Accuracy

5.2.1.1 PH sensor

Initially, the pH sensor exhibited a high degree of non-linearity. After applying the three-point calibration method, readings became more stable and consistent. Across 30 measurements, only 3 readings deviated slightly (±0.4 units) from expected values under constant solution conditions, resulting in a success rate of approximately 90.0%.

5.2.1.2 TDS sensor

The TDS sensor demonstrated high reliability. In 25 trials under different nutrient concentrations, 24 readings matched known reference values with less than 5% deviation, giving a success rate of 96%.

5.2.1.3 DHT11 sensor

Temperature and humidity readings were consistent with a margin of error less than ±1°C in indoor environments. From 30 trials, only 1 abnormal spike was recorded due to sudden heat from human proximity, yielding a success rate of 96.7%.

5.2.1.4 LDR sensor

The light sensor accurately triggered the LED system under low light conditions. It produced consistent output in 20 different lighting environments with no false negatives or positives. This indicates a 100% success rate in detecting light changes.

5.2.2 Actuator Reliability

Each actuator was tested under real conditions by simulating sensor inputs. Over 40 tests were performed for different modules:

Each actuator was tested under real conditions by simulating sensor inputs. Over 40 tests were performed for different modules:

- Nutrient Pumps: All four pumps responded as expected based on TDS values.
- Water Pump: Activated consistently when required during circulation cycles.
- Fan: Triggered appropriately when temperature exceeded the threshold.
- LED Strip: Turned on accurately under low light readings.
- Mixer Motor: Activated correctly after nutrient addition commands.

All relays and output devices functioned without delay or misfire, resulting in an actuation success rate of 100% across all trials.

All actuators responded correctly to sensor data. For example, when the TDS dropped, the corresponding nutrient pump activated. Similarly, when the ambient light dropped below a threshold, the LED strip was powered on.

5.2.3 Overall System Stability

The system was stress-tested over continuous operation for 48 hours, with no connectivity losses. The ESP32 maintained consistent communication with sensors and the local display. No reset or crash incidents were observed, confirming a highly stable embedded environment.

5.2.4 Conclusion

The Smart Aeroponic Farming System demonstrated high reliability in both sensing and actuation. Average sensor accuracy exceeded 95%, and actuator behavior was consistently successful under all tested conditions. These results confirm that the system is robust enough for deployment in real-world controlled agriculture environments.

5.3 Justifications of the Obtained Results

The performance of the Smart Aeroponic Farming System is strongly justified by the consistency and logic of the observed results in relation to the intended system behavior. Each component of the system reacted as designed under real-world conditions, and the output values were aligned with expected agricultural requirements for aeroponic plant growth.

5.3.1 Sensors

5.3.1.1 PH sensor Readings

The measured pH values initially displayed instability due to the sensor's non-linear behavior. This was addressed by implementing a custom calibration approach using three known values to construct linear approximations across specific pH ranges. As a result, the sensor produced readings between 5.5 and 6.5, which falls within the ideal nutrient range for most leafy greens and herbs. The stable performance post-calibration confirms the validity of the applied correction model.

5.3.1.2 TDS sensor Response

The Total Dissolved Solids (TDS) sensor returned values consistent with nutrient solution concentration changes. The system was configured to activate nutrient pumps when the TDS dropped below a defined threshold, and the successful triggering of the pumps during low concentration phases verifies that the sensor provided actionable and reliable data. The TDS values remained within the expected range of 800–1200 ppm, suitable for plant nutrition.

5.3.1.3 Temperature and Humidity (DHT11)

The DHT11 sensor consistently monitored ambient temperature and humidity. Its readings aligned with physical room conditions. The activation of the cooling fan at 28°C validated that both the sensor data and system logic were working correctly. This justifies the use of DHT11 as a low-cost but sufficiently accurate sensor for this application.

5.3.1.4 LDR and LED Activation

The light-dependent resistor (LDR) effectively detected changes in light intensity. During shaded or low-light scenarios, the LED strip was automatically turned on.

This behavior demonstrates that the threshold value set for light detection was well-tuned and that the sensor was properly positioned for environmental monitoring.

5.3.2 Overall System Behavior

Throughout multiple test cycles, the integration between sensors and actuators showed no malfunction or conflict. The ESP32 microcontroller processed inputs and executed corresponding outputs efficiently. The TFT screen accurately displayed real-time readings, confirming that the logic behind each control decision was appropriate and based on valid environmental triggers.

5.4 Summary

This chapter has presented a detailed analysis of the system's performance and experimental outcomes. The Smart Aeroponic Farming System demonstrated high reliability and functional consistency through various real-world tests. All sensors—including the pH, TDS, DHT11, and LDR—provided accurate readings that were successfully used to trigger actuators in real time.

The system displayed all live values on a TFT screen, allowing real-time monitoring. The actuator modules, including the nutrient pumps, water circulation pump, cooling fan, LED lighting, and mixing motor, responded exactly as designed based on sensor feedback. The ESP32 microcontroller effectively processed inputs and executed outputs with no delays or faults.

Justifications for the obtained results were established based on calibrated sensor performance, logic-based trigger mechanisms, and repeated experimental observations. Furthermore, success rates exceeding 95% across most modules confirm the robustness of the system in a controlled agricultural setting.

In conclusion, the implemented system not only met its intended goals but also proved to be an efficient, scalable, and accurate solution for aeroponic plant monitoring and automation.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Concluding remarks

The Smart Aeroponic Farming System developed in this project has successfully fulfilled its intended objectives, offering a practical, low-cost, and autonomous solution for controlled-environment agriculture. Throughout the implementation, the integration of hardware and software components was carefully designed and tested to ensure accurate monitoring and responsive control of the system.

From a development perspective, the use of the ESP32 microcontroller provided sufficient processing power and built-in Wi-Fi, enabling real-time data acquisition and wireless communication with the dashboard. The project relied on actual sensor readings from pH, TDS, DHT11, and LDRs, each of which played a critical role in ensuring optimal growth conditions. In particular, the pH sensor required special handling due to nonlinear readings, which were addressed using three different mapping equations, demonstrating our ability to adapt to real-world hardware limitations.

The hardware was successfully assembled and mounted on a custom-built wooden frame, including the nutrient tanks, the ESP32 circuit, the TFT screen for local display, and the mounted aeroponic tower. A major highlight was the power system, which included a 12V solar panel fixed at a ~45° angle, connected through a solar charge controller to a 12V Li-ion battery, making the system energy-autonomous.

Two relay modules were deployed to control:

- Four 12V nutrient pumps,
- A 12V water pump,
- A 12V cooling fan,

- A 5V strip light (triggered by low-light readings),
- A 5V mixing motor (triggered during nutrient delivery).

All components were tested both individually and in full integration, and the system responded correctly to environmental triggers.

Overall, the project provided a comprehensive learning experience that combined circuit design, sensor integration, embedded programming, and mechanical assembly. While there were some implementation challenges such as signal noise, calibration drift, and component placement they were resolved with practical solutions. The system, in its current form, is stable, scalable, and ready for real-world pilot use, with many possibilities for future enhancements.

6.2 Future work

While the current version of the Smart Aeroponic Farming System is fully functional and capable of autonomously monitoring and controlling environmental conditions, there are several areas where future improvements and expansions could further enhance the system's performance, reliability, and usability.

I. Automated Nutrient Balancing Based on Crop Type

In the current implementation, nutrients are delivered in a fixed quantity. In future versions, the system could include crop-specific nutrient profiles and dynamically adjust the ratio and volume of solutions based on plant type, age, or stage of growth. This would require adding a simple crop selection module to the dashboard.

II. Wireless Communication and Mobile Notifications

While data is currently sent to the dashboard, future enhancements could include real-time notifications to mobile phones via SMS, Telegram, or push notifications if a sensor reading crosses a danger threshold (e.g., extreme temperature or nutrient depletion). This would increase reliability in remote or unattended farms.

III. Auto-Calibration for Sensors

The current system requires manual calibration, especially for the pH and TDS sensors. Future work could explore implementing automatic baseline calibration routines or even self-cleaning probes to reduce human intervention.

IV. Data Logging and Analytics

The system could be extended to log historical sensor data over time and generate graphs and growth trend predictions. This would help in analyzing long-term plant health, system efficiency, and detecting hidden problems early.

V. Improved Dashboard Interface and Mobile App

The current web-based dashboard could be improved with a mobile-friendly UI, real-time charts, and interactive controls. In future phases, developing a native mobile app would allow full system monitoring and control from smartphones.

These improvements would significantly increase the system's precision, autonomy, and applicability to different agricultural contexts, supporting its evolution from a prototype to a robust smart farming product.

Reference

1. Nova. Indoor Farming | Temperature and Humidity Sensors in Agriculture. 2024, 15 May. Available from: https://blog.amphenol-sensors.com/industrial-blog/temperature-and-humidity-sensors-in-agriculture [Accessed 20 10 2024].

2. Agrotonomy. Commercial Aeroponic Systems. Available from: https://agrotonomy.com/commercial/ [Accessed 6 10 2024].

3. IEEE. About the IEEE Internet of Things (IoT) Technical Community. Available from: https://iot.ieee.org/about.html [Accessed 7 11 2024].

4. IEEE Sensors. IEEE SENSORS JOURNAL. Available from: https://ieee-sensors.org/ieee-sensors-journal [Accessed 3 10 2024].

5. Stanford University. Environmental Sensors, Embedded Systems, Remote Sensing. Available from: https://ee.stanford.edu/research/environmental-sensors-embedded-systems-remote-sensing [Accessed 25 1 2025].

6. Wikipedia. Database. Available from: https://en.wikipedia.org/wiki/Database [Accessed 21 12 2024].

7. ScienceDirect. Microcontroller – Engineering Topics. Available from: https://www.sciencedirect.com/topics/engineering/microcontroller [Accessed 21 12 2024].

8. Arab, Rasheed; Ideas, Mahmoud. Smart Aquaponics Monitoring System via IoT Techniques. Available from: https://scholar.ppu.edu/handle/123456789/7276 [Accessed 16 11 2024].

9. Hasanat, Fatima; Nofal, Ghada; Awad, Shorouq. IoT Based Customizable Vertical Farming Solution for Palestinian Plants. Available from: https://scholar.ppu.edu/handle/123456789/8874 [Accessed 20 11 2024].

10. Espressif. ESP32 Series Datasheet Version 4.7. 2024. Available from: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf [Accessed 4 12 2024].

11. Arduino. Arduino® UNO R3 Datasheet. Available from: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf [Accessed 4 12 2024].

12. Raspberry Pi Foundation. Raspberry Pi 4 Model B Specifications. Available from: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ [Accessed 5 11 2024].

13. Technolab. ESP32 Development Boards. Available from: https://technolab.ps/products?search=esp32 [Accessed 29 12 2024].

14. Meta Express. ESP32 Product Listings. Available from: https://www.meta-express.com/HOME/?page_is=marketplace&search_item_name=esp32 [Accessed 7 11 2024].

15. Omega. pH Electrodes – Glass Filled and Specialty. Available from: https://in.omega.com/pptst/PHE1478_1479_1525_1526.html [Accessed 20 12 2024].

16. OSHA. ANTIMONY & COMPOUNDS (as Sb). Occupational Safety and Health Administration. Available from: https://www.osha.gov/chemicaldata/526 [Accessed 11 1 2025].

17. Liu, T. Digital-output Relative Humidity & Temperature Sensor Module DHT22 (AM2302). SparkFun Electronics. Available from: https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf [Accessed 20 12 2024].

18. Components101. DHT11 Temperature and Humidity Sensor. Available from: https://components101.com/sensors/dht11-temperature-sensor [Accessed 7 1 2025].

19. DFRobot. SEN0244 Analog TDS Sensor Datasheet. Available from: https://www.digikey.be/htmldatasheets/production/2799469/0/0/1/sen0244.html [Accessed 23 12 2024].

20. DFRobot. Gravity: Analog Electrical Conductivity Sensor / Meter (K=10). Available from: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0300-H_Web.pdf [Accessed 12 12 2024].

21. Atlas Scientific. EZO Conductivity Circuit Datasheet. Available from: https://cdn.sparkfun.com/datasheets/Sensors/Biometric/EC_EZO_Datasheet.pdf [Accessed 5 12 2024].

22. Amazon. LDR Sensor. Available from: https://www.amazon.com/ldr-sensor/s?k=ldr+sensor [Accessed 26 11 2024].

23. Adafruit. TSL2561 Luminosity Sensor. Available from: https://www.adafruit.com/product/439 [Accessed 22 10 2024].

24. React. Quick Start Guide. Available from: https://react.dev/learn [Accessed 20 11 2024].

25. Angular. Introduction to Angular. Available from: https://angular.dev/overview [Accessed 3 1 2025].

26. Vue.js. Official Guide. Available from: https://vuejs.org/guide/introduction.html [Accessed 4 1 2025].

27. Node.js. Documentation. Available from: https://nodejs.org/docs/latest/api/ [Accessed 4 1 2025].

28. Django. Django Docs. Available from: https://docs.djangoproject.com/en/5.1/ [Accessed 4 1 2025].

29. Flask. Framework Documentation. Available from: https://flask.palletsprojects.com/en/stable/ [Accessed 20 1 2025].

30. EMQX. MQTT vs CoAP Comparison. Available from: https://www.emqx.com/en/blog/mqtt-vs-coap [Accessed 20 1 2025].

31. HiveMQ. MQTT vs CoAP for IoT. Available from: https://www.hivemq.com/blog/mqtt-vs-coap-for-iot/ [Accessed 20 1 2025].

32. Naik, P. A. Application Layer Protocols for IoT. Available from: https://ijcsmc.com/docs/papers/September2020/V9I9202019.pdf [Accessed 21 1 2025].

33. Google. Firebase Documentation. Available from: https://firebase.google.com/docs?hl=ar [Accessed 1 2 2025].

34. Amazon. AWS IoT Core Docs. Available from: https://docs.aws.amazon.com/iot/ [Accessed 5 2 2025].

35. Google Cloud. IoT Core Node.js Client. Available from: https://cloud.google.com/nodejs/docs/reference/iot/latest [Accessed 5 2 2025].

36. Arduino. Software. Available from: https://www.arduino.cc/en/software [Accessed 11 2 2025].

37. Random Nerd Tutorials. VS Code with PlatformIO. Available from: https://randomnerdtutorials.com/vs-code-platformio-ide-esp32-esp8266-arduino/ [Accessed 202 2025].

38. Espressif. Eclipse Setup for ESP-IDF. Available from: https://docs.espressif.com/projects/esp-idf/en/v3.2.3/get-started/eclipse-setup-windows.html [Accessed 20 2 2025].

39. University of Georgia. EC and pH in Hydroponics. Available from: https://hortphys.uga.edu/hortphys/files/2020/03/EC-and-pH.pdf [Accessed 20 2 2025].

40. Wikipedia. Pump. Available from: https://en.wikipedia.org/wiki/Pump [Accessed 2 2 2025].

41. MySQL. Official Website. Available from: https://www.mysql.com/ [Accessed 6 5 2025].

42. PostgreSQL. Official Website. Available from: https://www.postgresql.org/ [Accessed 6 5 2025].

43. SQLite. Official Website. Available from: https://www.sqlite.org/ [Accessed 6 5 2025].

44. Apache Friends. XAMPP. Available from: https://www.apachefriends.org/ [Accessed 6 5 2025].

45. WampServer. Official Website. Available from: https://www.wampserver.com/en/ [Accessed 6 5 2025].

46. MAMP. MAMP for Windows. Available from: https://www.mamp.info/en/windows/ [Accessed 6 5 2025].