Arduino Based Microcontroller-Driven Indoor Hydroponic Fodder System

A.K.Ranaweera

ITBNM-2110-0046

Faculty of Information Technology

Horizon Campus

Supervisor

Ms. Naduni Jayathilake

November-2024

This project proposal is submitted in partial fulfilment of the requirement of the Degree of Bachelor of Information Technology (Hons) in Network and Mobile Computing of the Horizon Campus.

DECLARATION

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: _____ Date: _____

The supervisor/s should certify the thesis/dissertation with the following declaration.

The above candidate has carried out research-based or product-based project for the thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of the supervisor: _____

Signature of the supervisor: _____ Date: _____

ABSTRACT

The microcontroller-driven indoor hydroponic fodder system is novel solution using advanced technology to transform fodder production into a highly sustainable and effective means of farming. This technological setup utilizes IoT- based sensors along with automation technologies to create an optimized indoor environment for growth of the nutrient-rich hydroponic fodder as the means to address certain agricultural challenges - limited agricultural land, water scarcity, and fluctuating climate conditions.

This system will be implemented to work intelligently and will provide all the sensing and the control needed in real time based on variables such as temperature, humidity, and light intensity. The design system shall be done with many sensors such as DHT11 temperature and humidity sensor, turbidity sensor for the water quality, and LDR for light intensity.

Real-time data is gathered and used by the system through a microcontroller, such as Arduino, to regulate parameters automatically. Other actuators include LED grow lights and water pumps to ensure ideal growth conditions that improve the speed and quality of fodder growth.

The interface of the system consists of a Mobile application, which is user-friendly and hence enables the farmers to monitor and operate the entire setup remotely. It provides updates on real-time data, visualization with historical data, and sends anomaly alerts to enable evidence-based decisions. That keeps the system's access and usability in order without having broad technical knowledge.

The novelty of the project is that it focuses on resource efficiency. Hydroponic farming uses much less water and space compared to conventional agriculture. The system becomes really advantageous in regions with limited access to arable land or water resources, thus managing to maintain fodder production throughout the year independent of external climatic conditions. Automation and integration of IoT further reduce labor and operational costs

The system will solve some of the modern-day burning agriculture problems, food security, and sustainability of the ecosystem, besides scaling up the model to make it adaptable to other crops. This enables him to support livestock small- and medium-scale farmers by offering an affordable and reliable solution that does not cause much ecological damage.

This work epitomizes how technology-driven innovation can turn around traditional practices into sustainable high-yield systems. Hence, this paper bridges the gap between research and real-world application in setting the benchmark for a future of agricultural practices—the microcontroller-driven indoor hydroponic fodder system.

ACKNOWLEDGMENT

I would like to express my heartfelt thanks to Horizon Campus for giving me the dignity and support needed to carry out this project. The academic environment and infrastructure have been instrumental in my journey.

My unshakable and everlasting thanks should go to my project supervisor, Ms. Naduni Jayathilake for her precious and for tremendous guidance, motivation, support, and smart criticism. The direction and execution of this project have greatly benefited from her expertise and insights. She is, very patient and willing to answer my many questions, and this has greatly enhanced my learning.

My heartiest thanks to my parents, too, for their constant support.

Contents

List of Figure

List of tables

List of Abbreviations

Abbreviation	Description
IoT	Internet of Things
ESP8266	Wi-Fi-Enabled Microcontroller
ESP32	Advanced IoT Microcontroller with Wi-Fi and Bluetooth
MQTT	Message Queuing Telemetry Transport
API	Application Programming Interface
LCD	Liquid Crystal Display
TDS	Total Dissolved Solids
pH	Potential of Hydrogen
EC	Electrical Conductivity
CO_2	Carbon Dioxide
LED	Light-Emitting Diode
Wi-Fi	Wireless Fidelity
PWM	Pulse Width Modulation
ADC	Analog-to-Digital Converter
GPIO	General Purpose Input/Output
DHT	Digital Humidity and Temperature Sensor
Firebase	Google’s Real-Time Cloud Database
MQ-2	Gas Sensor for Smoke and CO Detection
PCB	Printed Circuit Board
OTA	Over-the-Air (Firmware Updates)

CHAPTER 1 – INTRODUCTION

1.1 Background of the Study

Such burning issues confronting humanity include climate change, infectious diseases, growing urbanization, and the depletion of natural resource deposits. All these are responsible for a radical change in our global lifestyles. There is quite a fair probability that hydroponic farming can minimize the risks that all these problems pose to our food chain. One of the other most attractive benefits to hydroponic farming involves the use of CEA technology to grow crops under nearly optimum conditions. Another added benefit of indoor hydroponically grown crops is that they can be produced virtually anyplace in the world at any time of the year, independent of weather, availability of cultivable land, or soil quality. Hydroponic farming has potential in offering locally grown, fresh food in areas that have inferior quality soil and suffer from extreme droughts, including sub-Saharan Africa, where finding leafy green vegetables can be quite a task. In hydroponic plant growth, a water-based nutrition solution is used in place of soil, together with an aggregate substrate or growing media like perlite, vermiculite, or coconut coir.

The Hydroponic production systems are utilized by commercial enterprises, hobbyists, and small farmers. Hydroponically, nearly any crop can be grown; however, the most common ones are watercress, celery, tomatoes, peppers, cucumbers, strawberries, and leaf lettuce. In system design, an important consideration is how a crop is sustained in the nutrient solution.

1.2 Research Problem

Feeding a growing population is an unprecedented challenge that the world faces . create an environment that supports sustainable urban life, promoting a state of good health for all those who choose to live in cities.Our current agricultural system is up to a huge task by 2050. We will need to increase food production by about 70% in order to meet the caloric needs of a global population of 9.8 billion people 68% of whom are projected to live in urban areas.

Fodder is a critical input into the livestock feed and plays an important role in the agricultural economy. Livestock farming in general provides food, employment, and income to millions around the world. However, traditional fodder production has some serious drawbacks: it relies heavily on favorable weather, large volumes of water, land availability, and human labor. These constraints render the production of fodder susceptible to various disasters related to environmental changes like droughts, floods, and uncertain weather conditions.

These challenges call for the urgent need for an alternative fodder production approach that is efficient, sustainable, and adaptable to urban environments. One of the promising solutions is the microcontroller-driven indoor hydroponic fodder system. This system combines hydroponics with automation technology in creating an indoor farming environment that can produce high-quality fodder efficiently, regardless of external weather conditions. Farming is slowly being transformed by hydroponics.

Agriculture expands via the capacity to grow indoors. We develop here a small, mechanized grow chamber for growing feed indoors in under a week. The system makes use of grow lights that would imitate sunlight, a temperature-controlled chamber that can sustain a cool airflow atmosphere, monitoring of water and moisture, and grow lights. Next, temperature, moisture, and water sensors monitor interior conditions constantly. In addition, the system monitors the temperature and moisture levels by means of sensors, controlling the water level with a pump motor to maintain ideal temperature and moisture conditions for development. Artificial sunlight inside is turned on/off automatically by

the control based on the settings by the user. A control logic can be developed on an Arduino controller to effectively monitor the entire procedure to ensure it happens recursively without failure. The development of the system also notifies an alarm when the water in the tank runs out completely. Therefore, thisArduino controller-based system provides the most assured automated indoor fodder growing process.

Automation, IoT, and hydroponics bring a complete change in the mode of production with full control over environmental conditions and resources. These are the technologies that allow farmers to optimize water, nutrient, and energy use to achieve better yields with minimal wastage.

Hydroponics means a method of growing plants without using soil, but rather an aqueous solution full of nutrients. It usually uses much less water than the original farming and can be done indoors. Hydroponic systems, combined with microcontroller-based automation, can monitor and control critical parameters such as temperature, moisture, and nutrient levels in real-time. A level of control that ensures ideal conditions for crop growth and consistent quality.

The microcontrollers, like Arduino, are the 'brains' behind automated systems. These may interface with a variety of sensors and actuators, gathering data, making decisions, and setting controls on processes. With the integration of IoT technology, farmers can access and monitor these systems remotely using smartphones or computers.

1.3 Motivation and Significance of the Project

Motivation

The development of this microcontroller-based indoor hydroponic fertilizer system is a demand determined by the pressing needs the modern world faces: food production, resource conservation, and sustainable management. By 2050, the global population is projected to reach 9.8 billion people with 68% urban population; thus, the challenge shifts with added pressure on food demand on conventional agriculture. Traditional ways of farming have been burdened with several drawbacks in terms of high water usage, complete reliance on arable land, variability in climate, and environmental deterioration.

Fodder production, being no exception, is critical in supporting livestock. Globally, livestock farming is an important part of food security, economic stability, and livelihoods. However, traditional fodder production relies heavily on large areas of land, a lot of water, and favorable weather conditions. With the rise in urbanization, climate change, and water shortage, farmers face difficulties in producing good-quality and consistent fodder. These challenges indicate a greater need for innovative solutions that are efficient, scalable, and adaptable to modern constraints.

Hydroponics, being a method of soilless growth wherein plants are grown in nutrient solution-rich water, is a very promising technique. By integrating hydroponics with microcontroller-driven automation, we can optimize fodder production under an indoor environment. This approach reduces the water consumption, maximizes the usage of space, and the produce is consistent irrespective of the external environment conditions. Automation of the system minimizes manual labor and allows real-time monitoring and adjustments, hence workable for farmers of every scale.

Also, IoT technology integrates the potential for remote monitoring and control, hence giving farmers a great deal of flexibility and convenience. The system tackles three important problems of traditional farming: inconsistent yields, waste of resources, and susceptibility to unfavorable changes in climate. With such a sustainable, efficient, and cost-effective solution, this project has the potential to change the face of fodder production and support global food security.

Significant

The importance of this project can be summed up as the great revolution in livestock farming since it will introduce an inexpensive and environmentally friendly way to produce nutrient-rich fodder. Some of the major contributions from this project include:

Resource Efficiency:

The hydroponic system accommodates 70–90% less water compared to a traditional farming methods, thus being extremely apt for regions experiencing water insufficiency. Its compact indoor setup also requires very minimal space, hence an efficient alternative to land-intensive agriculture.

Year-Round Production:

Unlike traditional fodder farming, which relies on seasonal conditions, this system makes it possible to produce throughout the year without interruption, thereby assuring the livestock of a constant feed supply.

Economic Benefits to Farmers:

It will also help in cutting operational costs by automating key processes such as irrigation, lighting, and nutrient distribution. Above all, the ability to grow fodder independent of climatic conditions assures financial stability for farmers.

Technology Adoption in Agriculture:

This will encourage farmers to use technology for increased productivity and proper decision-making through integration with IoT sensors, microcontrollers, and user-friendly interfaces. It is in line with the global movement toward "smart farming" and precision agriculture.

The project will, in the final analysis, not only solve today's most important agricultural challenges but also contribute to be a sustainable and technologically driven future in farming. The microcontroller-driven indoor hydroponic fodded system will change the

face of livestock farming and promote environmental stewardship by increasing resource efficiency, productivity, & accessibility.

1.4 Aim(s) and Objectives of the Project

1.4.1 Aim of the Project

The primary aim of the Microcontroller-Driven Indoor Hydroponic Fodder System is to develop a smart, automated, and efficient hydroponic farming solution that ensures optimal plant growth conditions while minimizing resource wastage. By integrating IoT technology, real-time monitoring, and automation, this system provides farmers with a user-friendly and scalable solution for indoor fodder cultivation.

1.4.2 Objectives

Following are some general objectives one may keep in mind while embarking on a project in hydroponic farming:

1. Sustainable Agriculture:
Resource- and environment-friendly farming

2. Evergreen Product:
To supply the crops throughout the year with no effects of seasonal changes.

3.Water Conservation:
Conserving water by minimizing the loss and optimizing the irrigation techniques

4. Maximize Space Utilization:
Best land-use or space would be utilized by vertical farming and small hydroponic systems.

5.Enhance Crop Quality:

Raise superior crops, enhanced in flavor, nutrition, and appearance

6.Crop Yield increase:

Surpass conventional soil-based farming in output per unit area.

7. Control:

Through control over the growth area, lessen the chances of infection from soil-borne diseases and pests.

8. Educational Outreach:

The company shall always undertake to offer educational programs to the general public, schools, and any interested party seeking knowledge about modern farming methods and sustainable agriculture.

9. Job Creation:

Design and maintain hydroponic farming systems in ways that create jobs within the community.

10. Food Innovation:

For food innovation, experimentation should be encouraged on diversified crops, some of which are less traditional within mainstream agriculture.

1.4.3 Limitations of the Project

Limited Scalability:

The system may not scale for larger enterprises without considerable modification and changes since it was intended for small-scale application. Similarly, though an Arduino platform is versatile, there are memory and computational capacity limits that may restrain the system's complexity and scale.

Sensor Accuracy and Reliability:

Low-cost sensors utilized in Arduino applications are inconsistent in their reliability and accuracy. The inconsistency of the sensors could mean that the system may not maintain optimal growing conditions, which may have effects on plant health and growth.

Maintenance and Calibration

Sensors and actuators should be serviced and calibrated periodically to ensure appropriate operation of the systems. Poor maintenance leads to less than perfect growing conditions due to friction of the mechanical parts and sensor drifting.

Initial Setup Costs

Initially, the investment in the setup involving buying of sensors, microcontrollers, LED grow lights, and other peripherals involved is very high for a small-scale farmer. This can be one of the major deterrents to the adoption, especially in low-income farming communities.

Dependency on Electricity

Most of the operations in the system require electricity for the microcontroller, sensors, lighting, and water pumps. Since there is power instability in most regions, efficiency may be lowered in the workability of the system. Although backup systems, like solar panels, could be considered, they add to upfront costs.

Knowledge Gap in Adoption

Farmers usually have no experience with IoT-based systems and thus may have various problems with its adoption. Training should be provided, along with user-friendly interfaces that would bridge the gap, but this requirement takes time and additional resources.

Water Quality Dependency

This system is heavily relying on the quality of water used. The use of poor water quality, carrying excessive minerals or contaminants, may interfere with plant growth and clog irrigation systems. Advanced water filtration may be needed and perhaps will be more expensive.

1.6 Chapter Outline

The following chapters of this thesis will elaborate in detail and systematically on the research and development journey of the Microcontroller-Driven Indoor Hydroponic Fodder System. Each chapter addresses a specific aspect of the project, starting with the background and current state of hydroponic and automated farming systems, progressing through the design, implementation, and evaluation of the proposed system, and concluding with key findings and future research directions. This project aims to develop a sustainable and automated solution for the production of fodder, which will ensure that the necessities for efficient resource use and productivity increase in modern agriculture are met.

Chapter 2: Literature Review

A critical review of the literature concerning existing technologies of hydroponic farming, automation, and IoT applications in agriculture, and the present practices concerning fodder production, shall be presented. Limitations of traditional methods, benefits of hydroponics, and how automation/microcontrollers can help in making this more efficient and sustainable will form the basis of this chapter.

Chapter 3: Research Methodology

Explain the approach to research, designing process, and data collection procedure applied in the system. They comprise sensors, actuators, microcontroller selections, and strategies for the integration to realize automation and monitoring of the system in real-time.

Chapter 4: System Design

The Microcontroller-Driven Indoor Hydroponic Fodder System: In-depth Technical Design. This chapter will talk about the system architecture; hardware components comprising sensors, actuators, and microcontrollers; software design; block diagram and circuit diagram; strategy of integration. It talks also about manual control features and IoT connectivity.

Chapter 5: Implementation

Implementation of system design, including hardware building and preparation, sensor calibration, and interfacing with a microcontroller via software development: explanation regarding the development of Mobile interfaces for remote monitoring and control, troubles found in the implementation.

Chapter 6: Testing and Performance Evaluation

Characterization characterizes the performance analysis of the system through tests and calibration. The considerations made within this chapter are the accuracy of the sensor readings, efficiency in water and nutrient delivery, reliability in automated control, and user experience. This chapter, therefore, covers data analysis, results, and test run feedback.

Chapter 7: Conclusion and Future Work

The findings from this system regarding fodder production optimization are summarized here. It gives discussions on implications to sustainable agriculture, accrued benefits from automation, and possible improvements. Further suggestions for research are given, including the integration of renewable energy sources into the system and scaling up the system for other crops.

CHAPTER 2 – LITERATURE REVIEW.

2.1 Introduction

The need for efficient and sustainable agricultural practices has recently brought hydroponics into the limelight as one of the potential means of food production. As agricultural practices face increasing pressure, innovative approaches, such as hydroponics, have been considered—a manifestation of the soilless cultivation of crops, particularly in an urban set-up and resource-constraint system. Through this literature review, the present study aims to explore the microcontroller-driven indoor hydroponic fodder system in terms of its design, implementation, and benefits. Hydroponics is a method for growing plants without soil, and its resource use efficiency to produce high-quality fodder in a controlled environment has therefore gained relevance. In this review, an attempt has been made to synthesize findings from various studies to arrive at an in-depth understanding of the topic and to establish a foundation for future research in this area. The review shall focus on the technological evolution of hydroponic systems, the input of microcontrollers, and livestock feeding with agricultural sustainability. It argues for the benefits that accrue in the use of hydroponic systems and the challenges experienced in their actualization. With regard to such considerations, the review establishes a basis for understanding the contribution of microcontrollers toward the optimization of indoor hydroponic systems in fodder production.

Another objective would be to review the progress on, as well as applications of, microcontroller-driven indoor hydroponic fodder systems, specifically considering design factors that address functionality and agricultural productivity. The review is drawn from a variety of recent studies that have emphasized some themes, such as the technological frameworks applied. Through the synthesis of findings from various sources, this review provides insight into the current understanding of microcontroller-driven indoor hydroponic systems and future directions.

2.2 Technological Advancements in Hydroponic Systems

In this regard, recent research in hydroponic systems is focused on the integration of microcontrollers and IoT. Abu Sneineh and Shabaneh [1][7] present an example of a smart hydroponics monitoring system using an ESP32 microcontroller that automates the collection of environmental parameters, including pH, temperature, and nutrient levels. This system improves efficiency in water and nutrient supply and allows access from anywhere through applications like Blynk. The authors have observed that real-time data collection enables the optimization of plants' growth conditions and avoids the wastage of resources.

Ghorbel and Koşum [5] [18]describe the advantages of hydroponic fodder production, specifying that it can be performed on completely closed surfaces, thereby ensuring maximum production with the smallest possible area of land consumption. In their view, hydroponic systems can produce high-quality fodder all year round without being subject to the constraints of conventional agricultural production.

The microcontrollers are introduced to the hydroponic system to automate activities related to nutrient supply and environmental factors. Thus, Susilawati et al. have discussed the design and development of an Arduino microcontroller-based automatic hydroponic plant watering system that enhances the efficiency of the management of its nutrients by its sensors, which measure water and nutrient levels. It not only automates the process of watering but also provides real-time data related to nutrient concentrations to ensure that plants are subject to optimum conditions for growth.

On the other hand, Bhat et al. have suggested a mini set-up of indoor hydroponic fodder production with LED lighting and automated irrigation to get maximum yield with a minimum consumption of water [10][19]. In this set-up, the amalgamation of microcontroller technology with controlled lighting helps in the production of fodder throughout the year independently of seasonal changes in natural light.

2.3. Role of Microcontrollers in Hydroponic Systems

Microcontrollers set the core of automated hydroponic systems in real-time monitoring and environment control. Deshan et al. (2024) present an integrated device called "HypoSense" for monitoring temperature, humidity, pH, and light intensity. The adoption of Arduino-based microcontrollers in this device simply proves that affordable technology has all it takes to realize precision agriculture through the provision of accurate data to growers and reducing manual labor. The authors emphasize that these systems can increase efficiency for hydroponics farming considerably for smaller growers who lack the resources to undertake an expensive commercial setup [14].

Microcontrollers form a very vital aspect of automating and optimizing hydroponic systems. In another contribution to evaporative cool hydroponic chambers, Singh et al. [3][4] make a review using microcontroller technology for the creation of an enabling environment for fodder production. Some authors report improved temperature regulation and humidity control; two very critical factors for better growing conditions of hydroponic crops. This integration of technology increases crop yield while decreasing labor costs for monitoring and managing the same by hand.

Reddy and Harani [2] add that even microcontroller-driven systems are economically viable since they occupy less space and require fewer resources compared to conventional farming. In the process, monitoring environmental conditions in real time offers quite an advantage in areas experiencing undesirable climatic variability and water scarcity.

2.4. Benefits of Hydroponic Fodder for Livestock

Hydroponic fodder has long been documented to be nutritionally beneficial. Thalkar strongly emphasizes that the production of hydroponic fodder reduces water usage by as low as 4.78 liters per kilogram of the produced fodder in comparison to conventional methods, which often require much more. This can be immensely useful in saving water in water-scarce areas. Ghorbel and Koşum [5] further insist that hydroponically grown fodder is rich in protein, vitamins, and minerals, hence good for feeding livestock. With the short

production cycle of hydroponic fodder, the freshness of the feed in the stock is always assured, more so in areas with a limited supply of traditional fodder. The nutrient profiles of hydroponically grown fodder are also superior on many counts.

According to a comparative study by Barwant et al., hydroponic fodder not only increases nutritional value in feed but also improves livestock health and productivity, thus consequently raising milk yield and weight gain in animals [8]. It allows for close control over nutrient delivery, thus allowing tailored feeding regimens for improving the performance of livestock.

Hydroponic systems have many advantages from an environmental point of view, mainly because of resource conservation. Newell et al. [11][17] considered the greenhouse gas emissions related to hydroponic fodder production. It was determined that such a process reduces these emissions compared to conventional fodder cultivation. According to this study, hydroponically cultivated fodder uses fewer water resources and reduces the use of land to a minimum, which is very beneficial in terms of fighting climate change. The authors believe that hydroponics holds an opportunity for a more sustainable agricultural model, particularly in areas where the environment is degraded.

According to Singh et al. [3][16], it was observed that feeding hydroponically produced fodder to livestock improved digestibility and nutrient absorption. Therefore, improving milk production and enhancing general animal health would depend on it. They argue that incorporating hydroponic systems into livestock feeding regimes will greatly enhance both food security and sustainable agricultural practices.

2.5. Challenges and Limitations

This is not to say that there are no challenges in adopting microcontroller-driven hydroponic systems: considerable challenges will include the set-up cost, the technical know-how needed to run them, and so forth. Thalkar reveals that high returns come with

an attachment of huge investment in the technology and infrastructure in advanced hydroponics [6].

Reddy and Harani [2][9] assert that education concerning hydroponic systems should be continuous, and support to the farmers provided on an ongoing basis to effect change. The findings also suggest that optimization studies should further be performed for the development of nutrient solutions and growing conditions for the requirements of other crops/livestock.

However, such reliance on technology introduces the scope for failures or malfunctioning of a system, which can result in extreme risks to crop yields. Thus to maintain and troubleshoot these systems effectively, according to Mardiansyah et al., the users need training, otherwise the potential of hydroponic systems is wasted on underperformance.

2.6. Nutritional Advantages of Hydroponic Fodder

Another critical factor in adopting hydroponically grown fodder is its nutritional value. On the same note, Malhi et al. [12] emphasize that hydroponic fodder is rich in essential nutrients and, therefore, a better feed for the animal inventory. Value Addition to Livestock Feed: This study has shown one can increase digestibility and milk production by 8-13% just by introducing it into animal feeding. In areas where traditional sources of fodder are limited or low in quality, this nutritional benefit is of great importance.

The nutritional makeup of hydroponic fodder could be different depending on light intensities, and nutrient solutions, among others. Bedeke et al. (2024) examine the economic incentives to adopt hydroponic fodder production among pastoralist households in Ethiopia. There is a high improvement in household income through the reliable supply of high-quality feed for livestock production realized when hydroponic fodder is used in the livestock enterprise. These results show the potential of microcontroller-driven systems to enhance agricultural efficiency and contribute to economic stability in farming communities [13].

2.7. Automation and IoT in Hydroponic Farming

Their integration with the Internet of Things allows hydroponic systems to help in better automation and data-driven decision-making. On this topic, Suresh et al. (2024) describe an IoT-enabled hydroponic system fitted with sensors to monitor some critical parameters, like pH, electrical conductivity, and water level, to ensure growing conditions are at their best while preventing any waste of resources. The authors say that IoT integration provides operational efficiency and the facility for remote management of systems to farmers, thereby improving productivity and sustainability.[15]

2.8. Future Directions

Although the future of microcontroller-driven indoor hydroponic systems may seem encouraging, research is underway to enhance the resilience and efficiency of the system. Investigations are going on, especially in innovations such as IoT integration for remote monitoring and control, in order to make the functioning of the system smoother. Apart from that, it may also have reduced operational costs and increased sustainability by using alternative energy sources, such as solar power, within research. The focus of future research should thus be on the development of cost-effective, user-friendly systems that farmers can easily adopt in various agricultural scenarios.

2.9. Conclusion

In conclusion, microcontroller-driven indoor hydroponic fodder systems hold immense promise for advancing agricultural sustainability and food security. The integration of automation and IoT technologies enhances resource management, crop yields, and the nutritional quality of livestock feed, offering a viable solution for fodder production in resource-constrained environments. Despite these benefits, challenges such as microbial risks, high initial investment costs, and the need for technical expertise must be addressed to ensure the widespread adoption of these systems. Future research should focus on optimizing system designs, improving disease management, enhancing user training, and integrating sustainable energy sources. By overcoming these obstacles, the agricultural sector can fully harness the potential of hydroponic systems to meet the increasing demand for sustainable and efficient livestock feed production.

CHAPTER 3– ANALYSIS.

3.1 Introduction

The microcontroller-driven indoor hydroponic fodder system is a technological intervention to overcome the limitation of the traditional method of fodder production. This system relies on automation, microcontrollers, and IoT for better efficiency in indoor conditions for the growth of fodder.

The main idea of the designed system is continuous monitoring of the Hydroponic System situation over the internet. IoT monitored Hydroponic system has 6 sensors. First one is a temperature and humidity Sensor, second is Soil Moisture Sensor, third one is Turbidity Sensor, the fourth one Water Level Sensor and the fifth one is LDR sensor, sixth one is TDS sensor. This project is very useful since the farmer can monitor Hydroponic system parameters just by visiting a Mobile application. And nowadays many IoT apps are also being developed. So now the Farmer can monitor the Hydroponic system through the Internet. The ESP8266 board continuously reads input from these 6 sensors. Then it sends this data to the cloud Then this action of sending data to the cloud is repeated after a particular interval of time.

3.2 Feasibility Study

3.2.1 Technical feasibility

- Language – Arduino, C++, Html, JS, CSS
- Hardware – ESP-8266 board, Arduino Uno board, Soil moisture Sensor, DHT 11 sensor, water leavel sensor , Turbidity Sensor, TDS sensor, LDR sensor,Gas sensor , Water Pump.
- Tool - Uno boards wire
- Software – Arduino IDE, Sublime Text, NotePad++
- Cloud – Thinkspik or Firebase.

3.2.2 Operation feasibility

Operational feasibility means that there will be a very important activity in the implementation process that will make sure the Microcontroller-Driven Indoor Hydroponic Fodder System can be utilized and maintained by the farmers and the agricultural technicians for its sustainability.

Feasibility analysis of the system-that is, how pragmatic for realistic agricultural settings it would be when put to work-meaning ease of use, user skills, maintenance, and resources. This will ensure that, in so doing, the system achieves its objective: to provide a sustainable, effective, reachable means of enhancing fodder production.

Salient among these features of this system is the fact that it is highly user-friendly. Having a more automated process minimizes human intervention, thus allowing this to cater to just about any given number of users, be they technical or non-technical. This system is a pretty straightforward interface with an LCD display for real-time monitoring of temperature, moisture, and nutrient status. It is very easy to use because, with minimal training, farmers can master how it works very easily. The aforementioned system sends automated messages when an issue has taken place, such as the water level, nutrient deficiency, or any problem with the sensor, for added confidence of the user and avoidance of any type of error or omission.

Moreover, the system allows for IoT-enabled applications that may be used for remote monitoring and control. This will provide the user with access to the system for data and its adjustment through their Mobile and computers. To farmers that have several tasks to attend to at different places, remote access of the system facilitates convenience in several ways. The Mobile interface was made very friendly to ensure that even customers less exposed to technology could use this application without much hassle.

3.2.3 Financial feasibility

Table 1-Financial feasibility

Item	Price (RS)
ESP-8266 board	1500
Arduino Uno board	3500
Soil moisture Sensor	400
LDR Sensor	350
DHT 11 Sensor	450
TDS Sensor	3450
Turbidity Sensor	3850
Water level sensor	300
Sola panel	900
Water Pump	1200
Pipes and Socket	7000
Water tank	1700
Jumper Wire	900
LCD display	800
Others	7000
Total	**33,300**

3.2.4 Legal feasibility

Legal feasibility is a very crucial factor in the system's success for its implementation and usage, considering some regions will be under strict agricultural, environmental, and technological regulations.

Another critical dimension of legal feasibility is compliance with the standards and requirements concerning agriculture. It should not run in conflict with the national or local policy concerning indoor farming, hydroponic farming, and automation technology use for agriculture. There will be a need to set up regulations regarding water use, nutrient use, and environmental impact to prove that the system operates within allowable limits.

Other factors include electrical safety and certification. The system will be reliant on electric components, including microcontrollers, sensors, and actuators; it therefore has to be in line with set electrical safety standards that guarantee the safety of the users and reduce malfunctioning. Certifications like CE Compliance or FCC are required and may depend on the location.

Data privacy and security by law, if the incorporation of IoT technology is involved with this system. The operating environment and performance data would be collected and transmitted via this system. Following protection legislation on data, for example, GDPR, requires assurance that user information could securely be stored and transmitted against unauthorized access.

Conclusion Addressing legal feasibility means that the system will be compliant with all relevant laws and standards, protecting users and engendering trust that could lead to wider diffusion of this innovative agricultural solution.

3.3 Fact Finding Techniques

3.3.1 Survey Question

A survey has been designed to understand the needs and opinions of potential users in ensuring that the Microcontroller-Driven Indoor Hydroponic Fodder System is both effective and user-friendly. The questions are intended to establish user needs, problems

faced in traditional fodder production, and expectations from an automated system. Responses will be used to guide system optimization and innovation.

Farmers

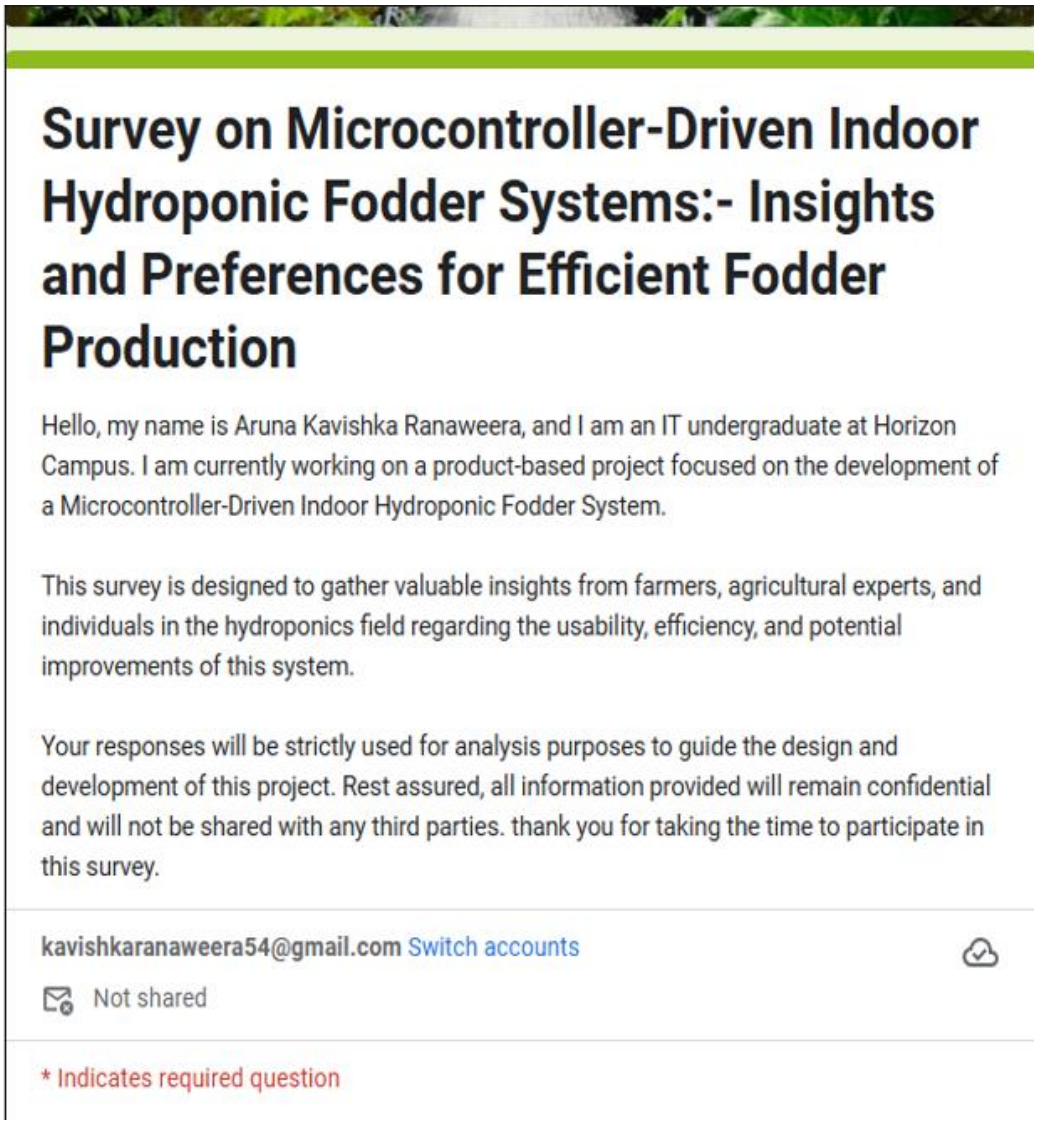

Figure 1- question01

1. What is your age group? (ඔබේ වයස් කාණ්ඩය කුමක්ද?) *

- ◯ Under 18
- ◯ 18-25
- ◯ 26-35
- ◯ 36-45
- ◯ 46-55
- ◯ 56+
- ◯ Other:

2.What is your occupation?(ඔබේ රැකියාව කුමක්ද?)

- ◯ Farmer
- ◯ Livestock Producer
- ◯ Agronomist
- ◯ Engineer
- ◯ Agricultural Technician
- ◯ Researcher/Scientist
- ◯ Student
- ◯ Other:

Figure 2 - question02

3.Do you have experience in farming or agriculture? (ඔබට ගොවිතැන හෝ කෘෂිකර්මාන්තය පිළිබඳ අත්දැකීම් තිබේද?) *

- ◯ Yes
- ◯ No

2. Experience with Hydroponic Systems

4.Have you used a hydroponic system before? (ඔබ මීට පෙර හයිඩ්‍රොපොනික් පද්ධතියක් භාවිතා කර තිබේද?) *

- ◯ Yes
- ◯ No

5.If yes, what type of hydroponic system have you used? (ඔව් නම්, ඔබ භාවිතා කර ඇත්තේ කුමන ආකාරයේ හයිඩ්‍රොපොනික් පද්ධතියක්ද?)

- ◯ Deep Water Culture
- ◯ Nutrient Film Technique (NFT)
- ◯ Ebb and Flow
- ◯ Other:

Figure 3-question03

8.What challenges have you faced with your hydroponic systems? (ඔබේ හයිඩ්‍රොපොනික් පද්ධති සමඟ ඔබ මුහුණ දී ඇති අභියෝග මොනවාද?) *

- Temperature control
- Humidity control
- Mold or fungal growth
- Uneven water distribution
- Nutrient imbalance
- High energy consumption
- Equipment malfunction
- I have no idea this is my first time
- Other:

9.How satisfied are you with the current hydroponic technology? (වත්මන් හයිඩ්‍රොපොනික් තාක්ෂණය ගැන ඔබ කෙකරම් තෘප්තිමත්ද?) *

- (Very dissatisfied)
- (Dissatisfied)
- (Neutral)
- (Satisfied)
- (Very satisfied)

Figure 4 - question04

10.What are the main challenges you face in producing fodder for your livestock? (ඔබේ පශු සම්පත් සඳහා ආහාර නිෂ්පාදනය කිරීමේදී ඔබ මුහුණ දෙන ප්‍රධාන අභියෝග මොනවාද?) *

- Availability of water
- High cost of equipment
- Labor requirements
- Maintaining consistent growth
- I have no idea this is my first time
- Other:

11.How do you currently monitor and control the environment for fodder production? How would you like to observe?(ඔබ දැනට ආහාර නිෂ්පාදනය සඳහා පරිසරය නිරීක්ෂණය සහ පාලනය කරන්නේ කෙසේද?) (ඔබ නිරීක්ෂණය කිරීමට කැමති කෙසේද?) *

- Manually
- Semi-automated
- Fully automated
- No monitoring or control

Figure 5 - question05

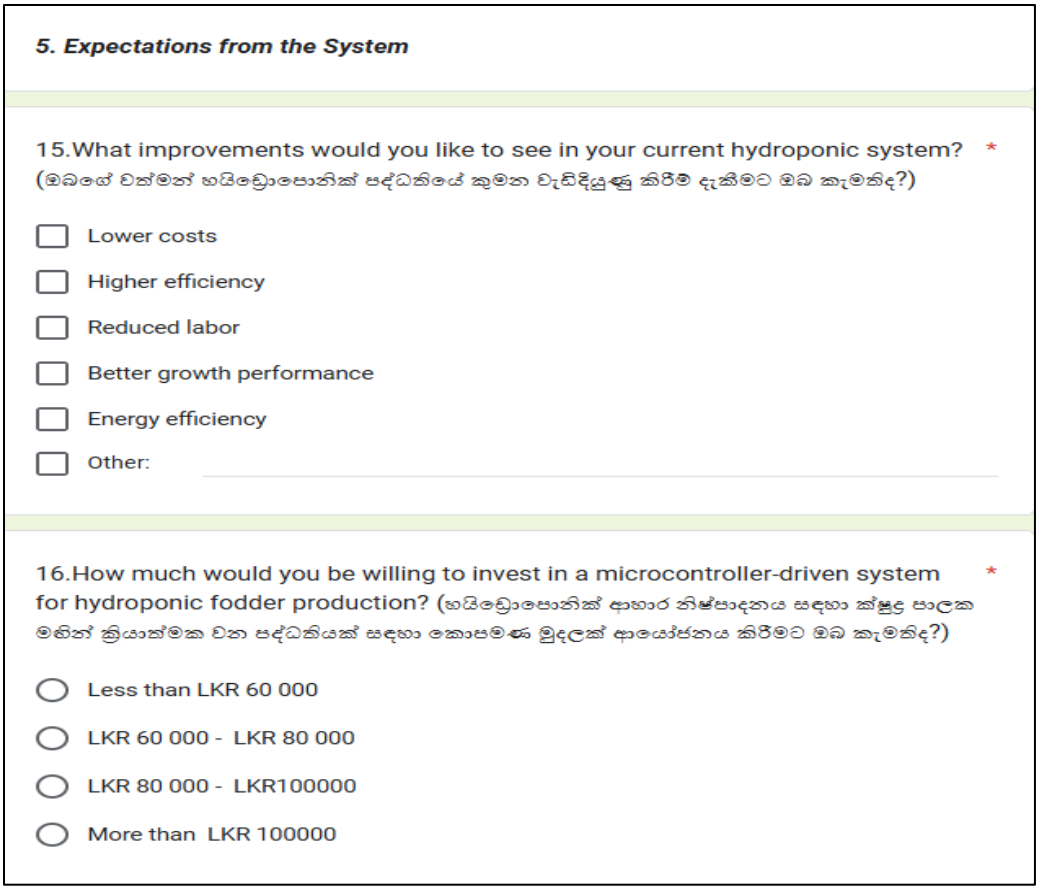

Figure 6 - question06

3.4 Requirement Analysis

Surveys questionnaires for Farmers Analysis

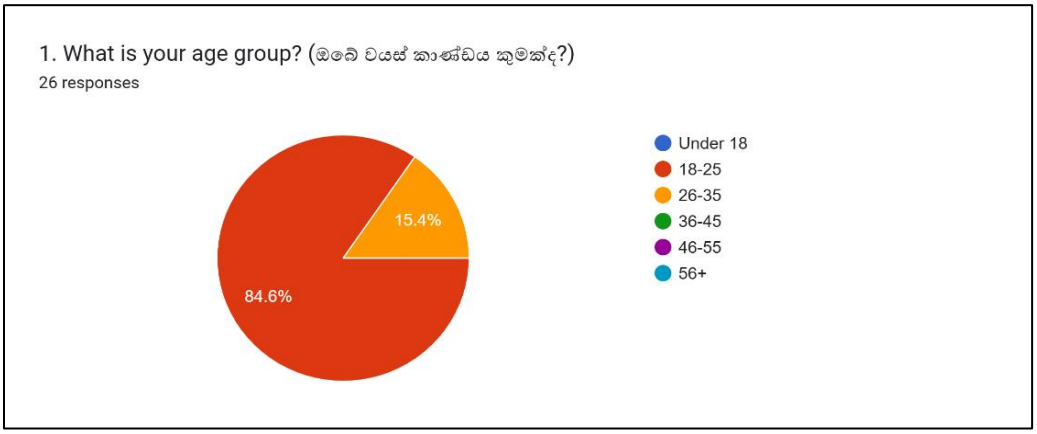

Figure 7-Answer01

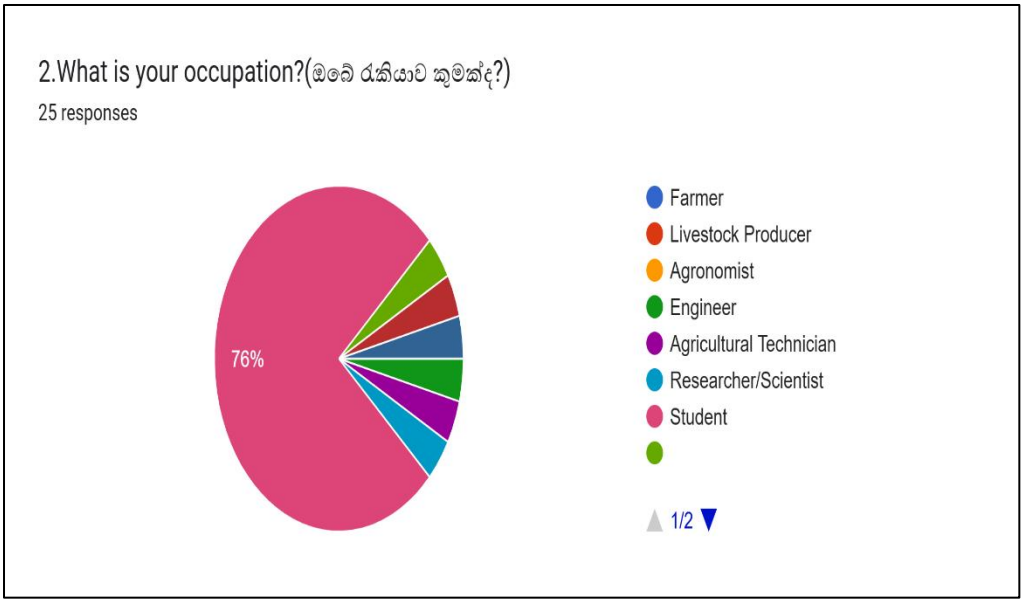

Figure 8-Answer02

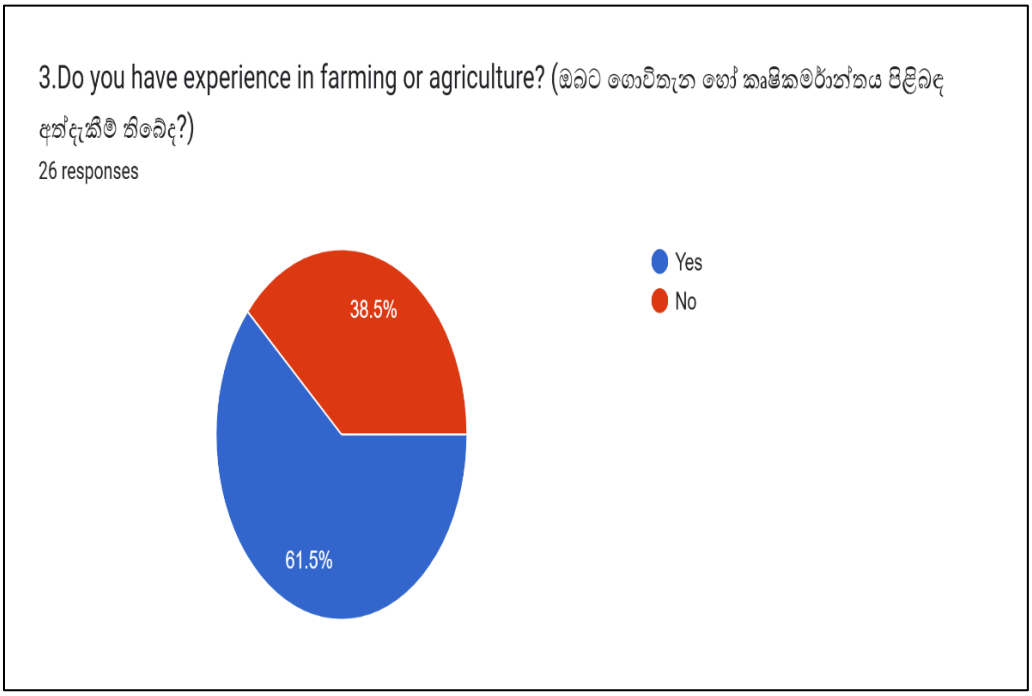

Figure 9-Answer03

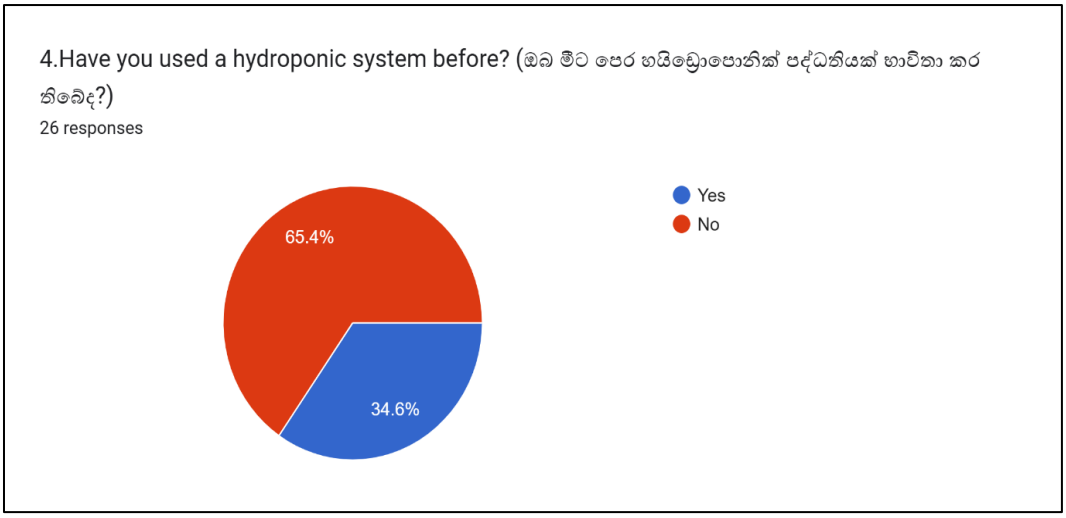

Figure 10-Answer04

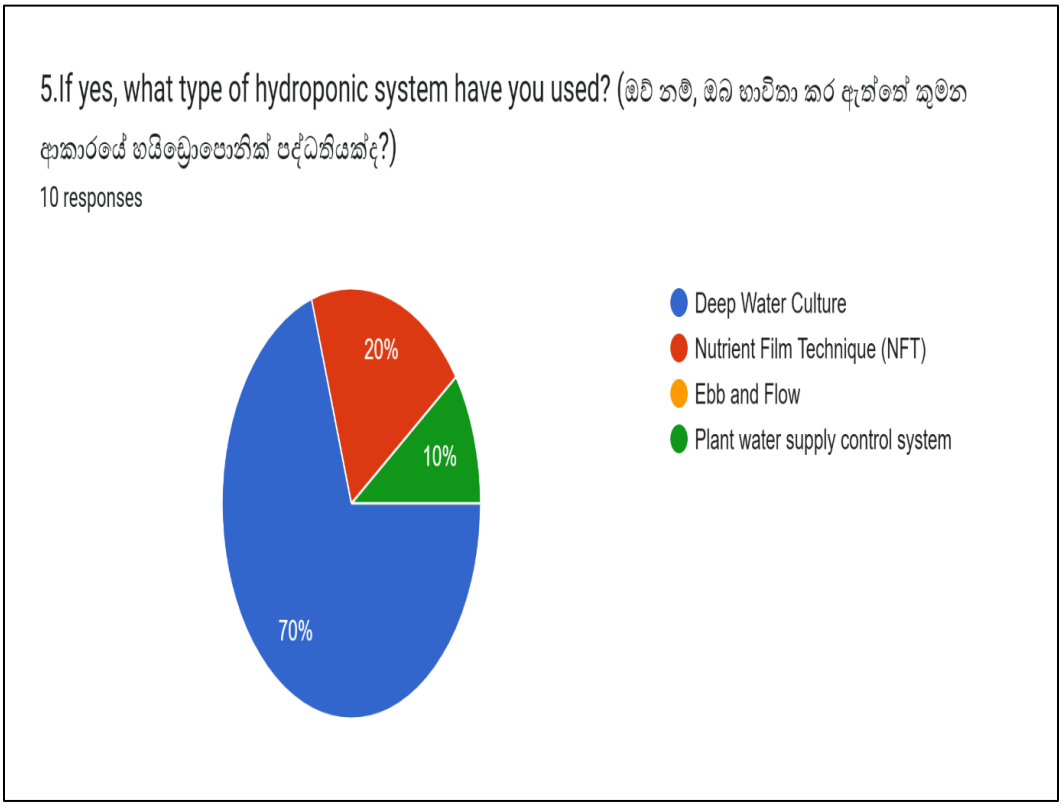

Figure 11-Answer05

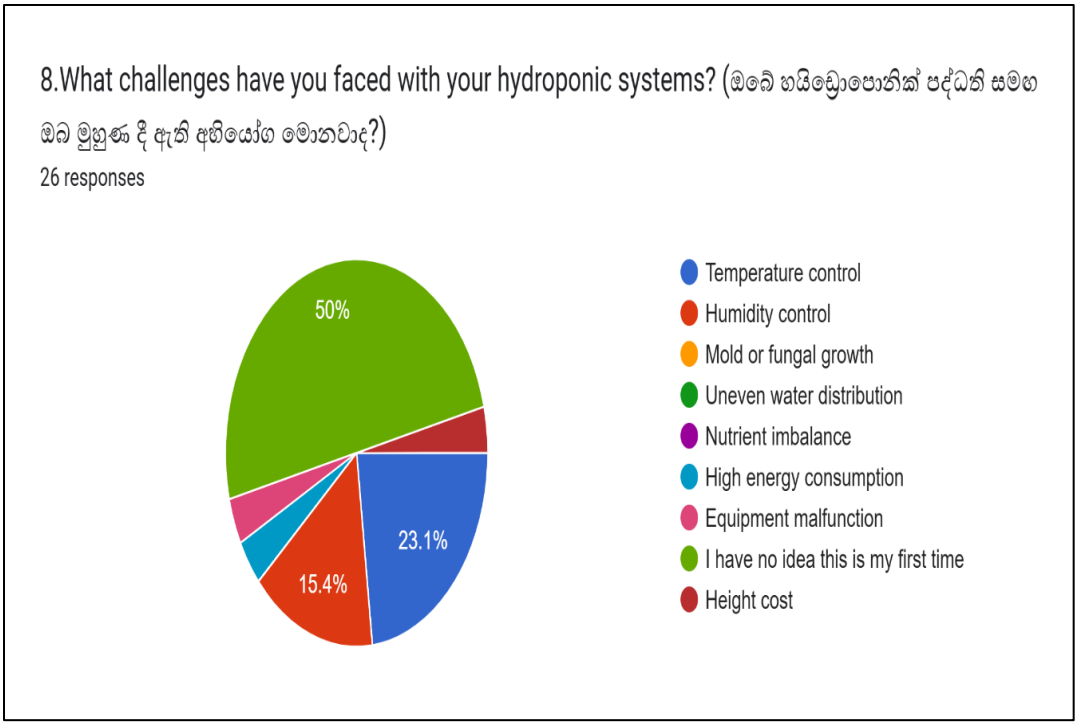

Figure 12-Answer06

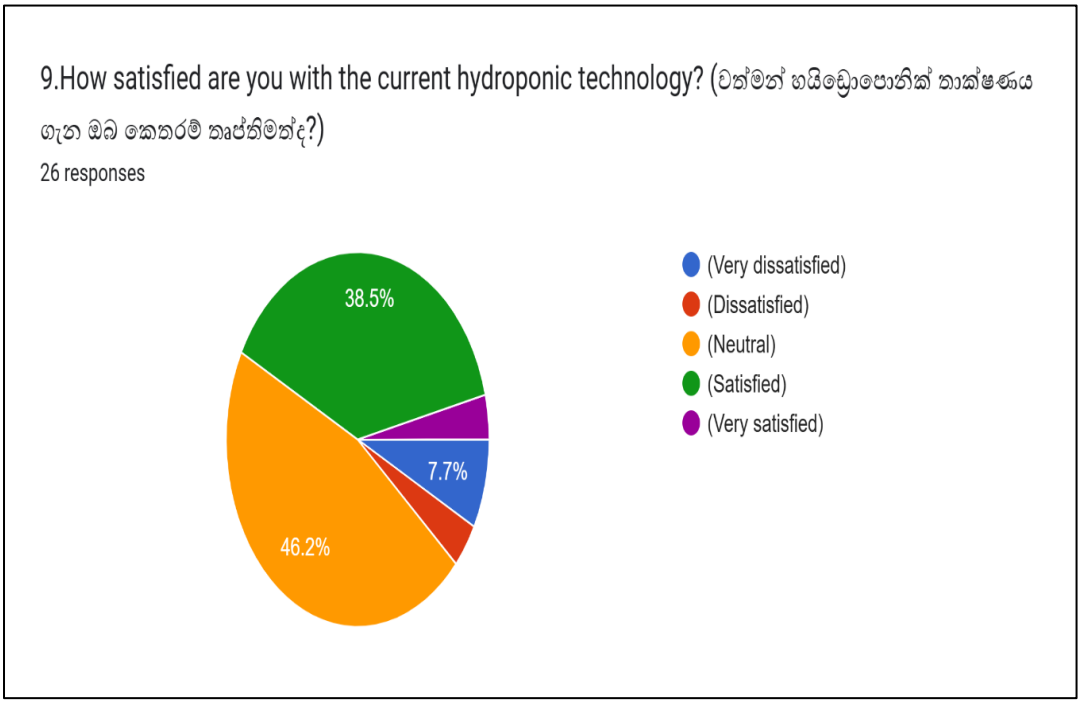

Figure 13-Answer07

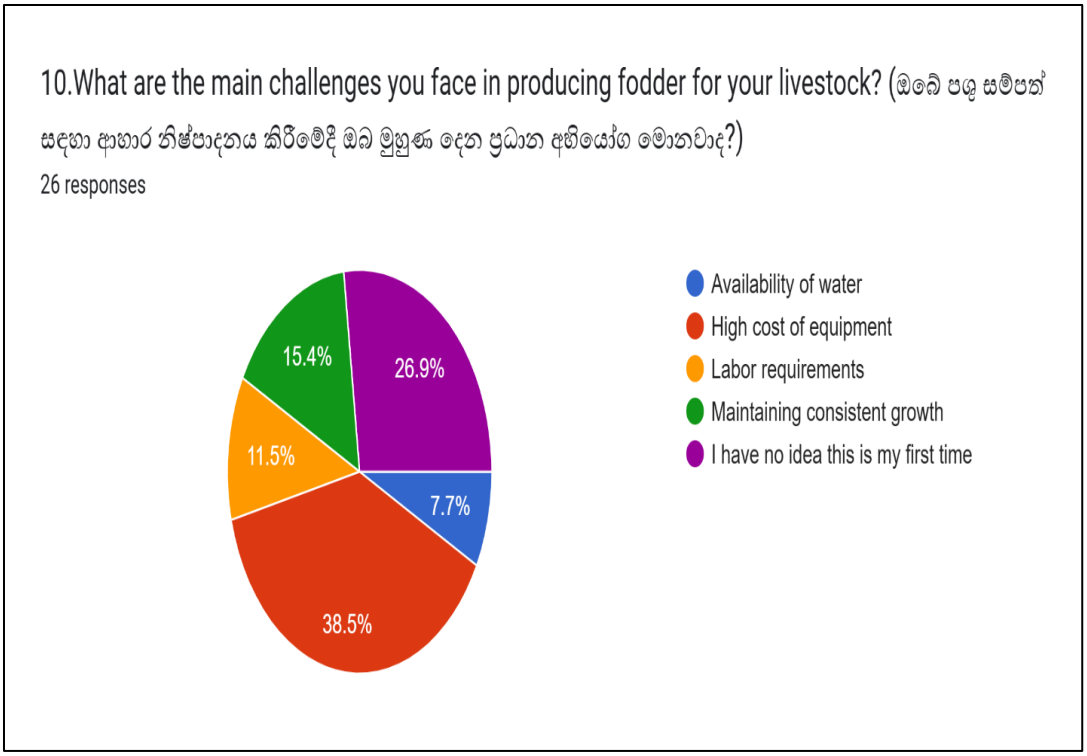

Figure 14-Answer08

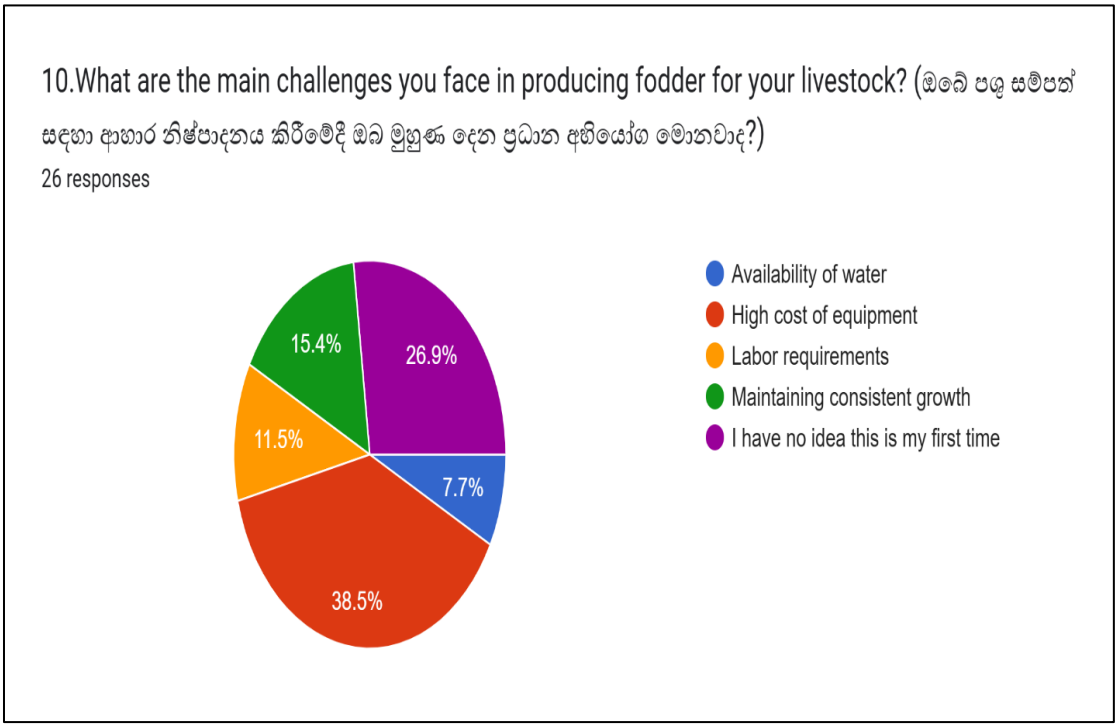

Figure 15-Answer09

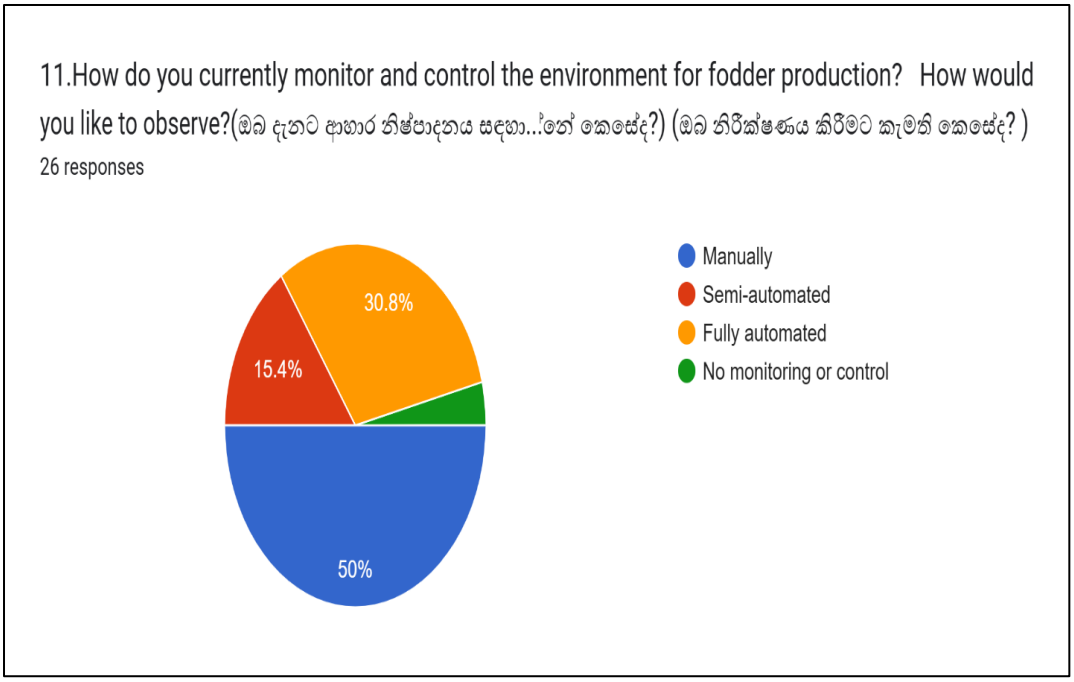

Figure 16-Answer10

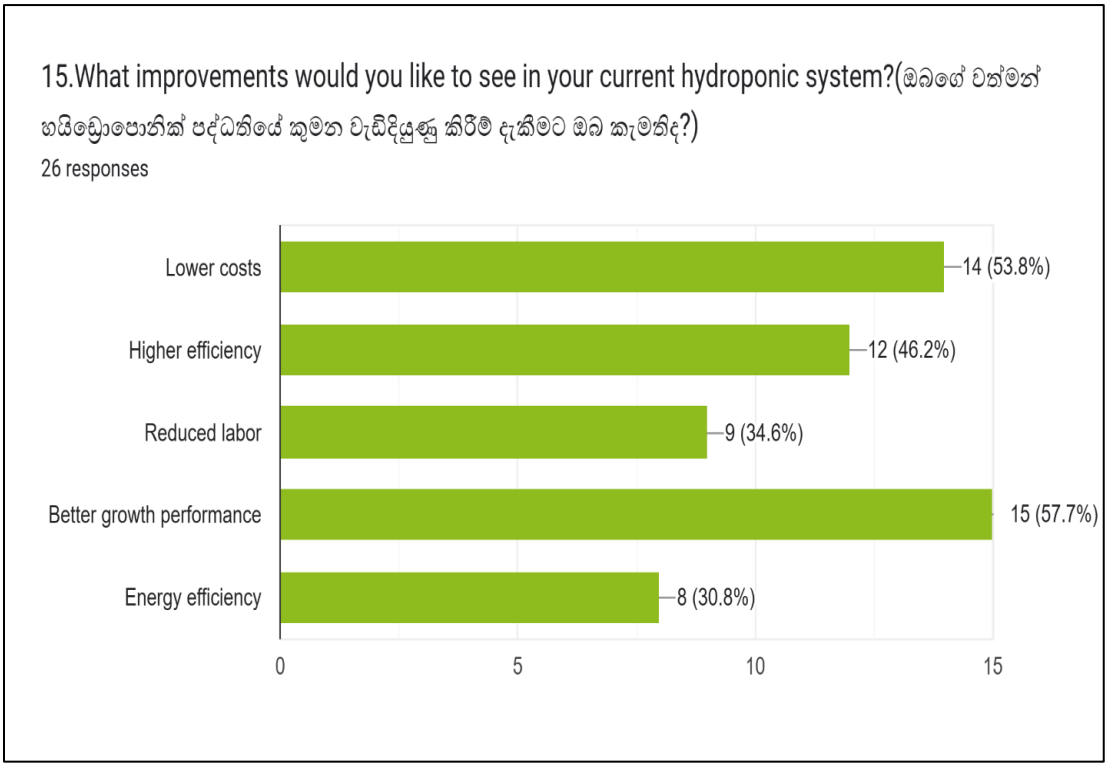

Figure 17-Answer11

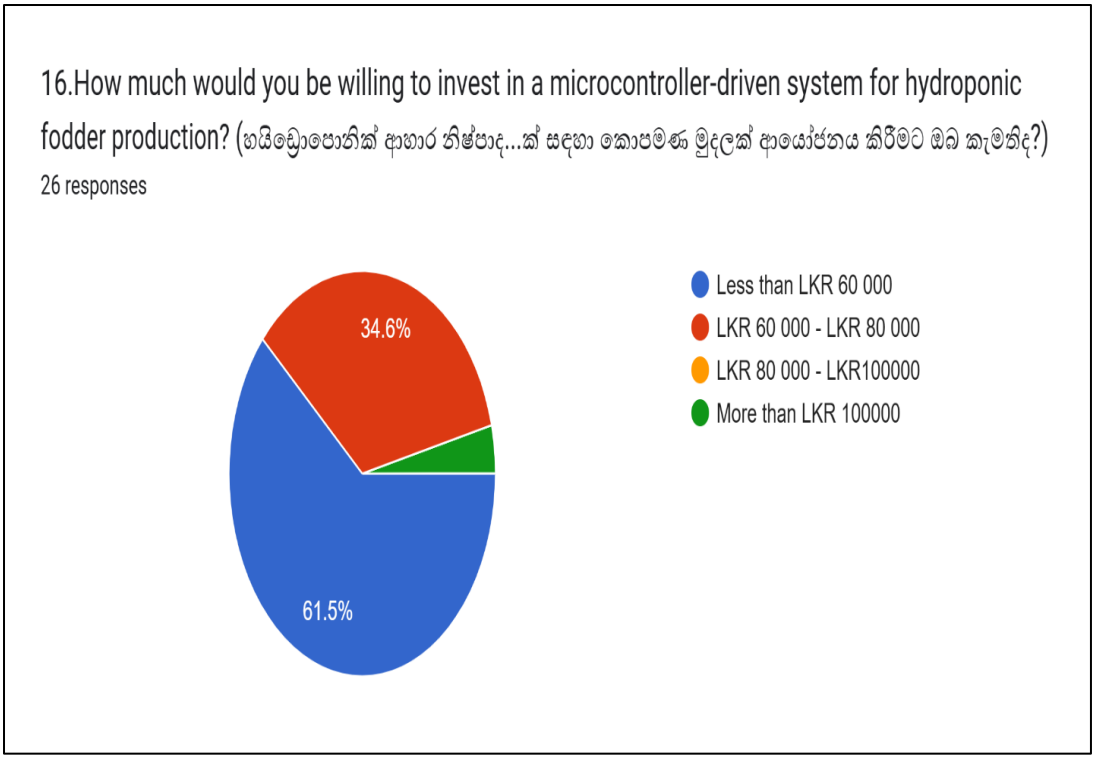

Figure 18-Answer12

3.5 Functional Requirements

Monitoring of Sensor Data:

Be continuously informed about environmental parameters like temperature, humidity, TDS level, and moisture level.

Automatic Control:

Automatically running water flow via pumps and solenoid valves based on sensor readings.

Nutrient Delivery System:

Provide the appropriate quantity of nutrient solution to the hydroponic trays at the right time

Real-Time Feedback:

Display the system parameters such as temperature, humidity, and TDS level and moisture level on the LCD screen.

IoT Connectivity:

Provide remote monitoring and control of the system through any Mobile or Mobile-based application.

Energy Efficiency:

The functioning of various components of the system efficiently decreases energy consumption.

Light Intensity Control:

LED grow lights are controlled to simulate natural sunlight by switching them on/off to maintain a certain light intensity as per predefined requirements.

LED lights must turn on when the level of intensity in light falls below a programmable level.

3.6 Non-Functional Requirements

Reliability:

The system will operate for long periods without interruption, with minimal system downtime.

Scalability:

Assuming this would be extended to more trays or sensors, the system design should support such scaling.

Performance:

Sensor data shall be processed, and reaction time in case of changes shall be less than 1 second.

Environmental conditions should be regularly maintained to achieve optimal fodder growth.

Usability:

Use is made easy via a simple-to-use, intuitive interface for user-system interaction.

The system must be easy to install and set up.

Maintainability:

Design components to be modular and easily replaceable or upgradable when needed. Provide clear error messages and self-diagnosis capabilities whenever failures may occur with the system.

Security:

Perform encryption on IoT communication to avoid unauthorized access. Limit physical access to the hardware components themselves.

3.7 Methodology for the System Development

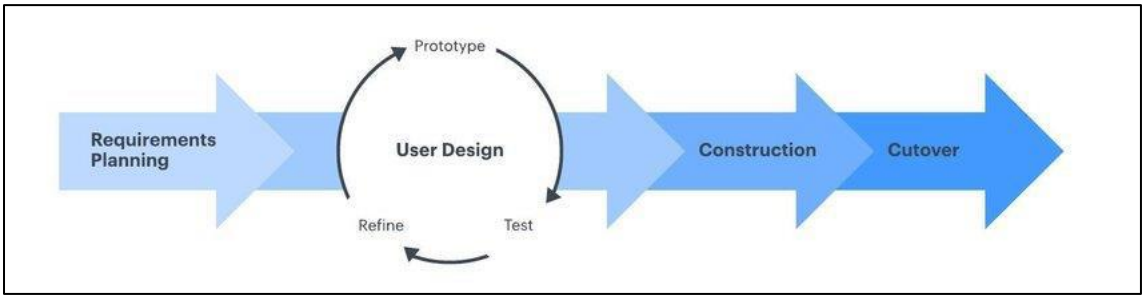

Figure 19-Methodology for system development

3.7.1 Background of the Project

Agriculture is a principal source of food production in the world, with technology consistently changing traditional agricultural practice. One such technology is hydroponics, a practice of growing crops in a water-based, nutrition-supplemented environment, free of soil. In urban areas, such a practice comes in useful, with little spaces and availability of water. Microcontroller-Driven Indoor Hydroponic Fodder System aims at automating and optimizing practice in growing fodder, offering farmers an efficient, environmentally friendly, and cost-effective alternative.

The traditional practice of producing fodder entails a lot of manpower, expansive lands, high water consumption, and constant observation through manpower alone. Climate change, unpredictable weather, and lands degradation are additional factors eroding uniform availability of fodder. By combining microcontrollers, sensors, and automation, such a project aims at countering such factors through a controlled environment for growing fodder indoors, regardless of external factors.

The use of IoT (Internet of Things) technology takes system performance a notch higher, with farmers having access to remotely monitor and manage key factors such as temperature, humidity, water level, and nutrition delivery. With increased demand for environmentally friendly agricultural practice worldwide, such a system helps in minimizing wastage of resources and high-quality production of fodder.

3.7.2 Selected Methodology

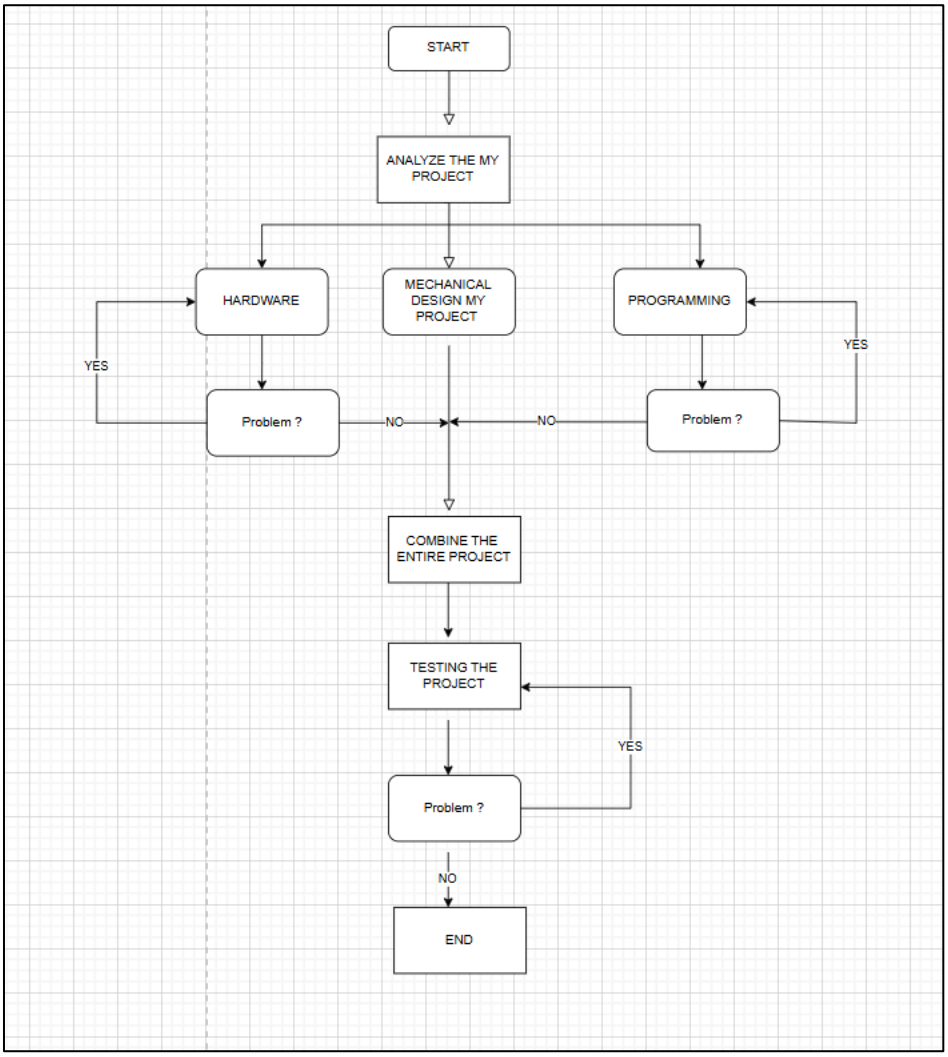

Figure 20-Methodology

1 Requirements Gathering and Analysis

The first stage of development focuses on clearly defining the system requirements through extensive research and analysis. The main objectives identified are:

- Automation of fodder cultivation using a microcontroller-based system.

- Real-time monitoring of environmental parameters such as temperature, humidity, water levels, and nutrient concentration.
- Automated irrigation and nutrient supply to optimize plant growth.
- Remote monitoring via IoT integration to ensure ease of access for users.

2. System Design:

The design includes connecting the necessary elements to an Arduino microcontroller in order to automate the growth process. It will integrate sensors, actuators, and IoT modules to monitor and control key parameters such as TDS levels, temperature sensor DHT11, and moisture levels while enabling manual control when needed.

3. Hardware Development:

Microcontroller:

Model: ESP 8266 or Arduino Uno

Function: This will serve as the central control unit that receives data from sensors and issues commands to various actuators. Its purpose will be to automate irrigation, lighting, and delivery of nutrients.

Connectivity: With a built-in Wi-Fi module for IoT functionality, this should be able to be monitored through an application or Mobile interface.

Temperature Sensor (DHT11 or DS18B20): Provides ambient temperature.

TDS (Total Dissolved Solids) Sensor: Provides the nutrient concentration in water.

Moisture Sensor (Capacitive or Resistive): Detects moisture content in the growing medium-soil or hydroponic trays.

Actuators:

Water Pump: For nutrient solution/water delivery

LED Grow Lights: Full-spectrum LED lights to support photosynthesis.

Cooling Fan or Heater (Optional): To control the temperature.

4. Software Development:

Libraries: DHT11, TDS, and moisture sensors. Since the ESP8266 will be used for remote monitoring, its IoT library-usually Blynk or ThingSpeak-will be employed.

Real-Time Control and Automation:

The system has predefined threshold levels in terms of temperature, humidity, moisture, and TDS levels. The system automatically switches on the pump, lights, or fans if its parameters are out of the range from the ideal ones.

• Arduino IDE (C++, Adriano.)

• Mobile app development platform - Visual Studio code, expo go

• Cloud Service – Firebase

5. Programming:

Here is the simplified Arduino code for the microcontroller-driven indoor hydroponic fodder system. The program below incorporates sensor readings, actuator control, and basic decision logic for the system. The system will monitor temperature, TDS level, and moisture level while activating actuators that include a water pump, LED lighting, and a cooling fan. Available options in the program are also given for manual control

6. Testing and Calibration:

Each sensor, whether temperature, TDS, or moisture, has to be calibrated with known reference values for accurate reading. The TDS sensor is normally calibrated using standard solutions, whereas the moisture sensor is subjected to different soil conditions to

check its response. Different actuators such as pumps and fans are also checked upon to see if their activation depends upon the threshold level set by sensors. The system-wide testing will involve the setup running under controlled conditions to validate the automated responses . Regular calibration ensures long-term accuracy and efficient performance of the system

7. Integration and Optimization:

- Hardware Integration: Sensors (temperature, TDS, and moisture) and actuators-pump, fan, grow lights-must be connected to the microcontroller. The sensors must provide seamless data flow, and the actuators will take commands from the microcontroller.
- Software Integration: The integrated program has to process sensor data, automate the actuators, and interface with the IoT platform for remote monitoring
- IoT and User Interface: Integrate IoT modules that log data in real time using a user-friendly Mobile interface.
- Testing for Efficiency: Perform the end-to-end tests to make sure everything works together, with optimal response times and overall system reliability.

CHAPTER 4 - DESIGN

4.1 Introduction

Designing a robust and efficient Microcontroller-Driven Indoor Hydroponic Fodder System is crucial to ensuring its effectiveness in automating the fodder cultivation process. This chapter focuses on the system design, detailing the hardware and software components, their interactions, and the overall architecture of the system. The design phase is essential to transforming theoretical concepts into a fully functional prototype by defining system architecture, hardware selection, sensor integration, control mechanisms, and user interfaces.

A well-structured design ensures that the system operates optimally by integrating microcontrollers, sensors, actuators, and an IoT-based monitoring platform. The use of Arduino/ESP8266, various sensors (temperature, humidity, water level, and nutrient concentration), and control components like pumps, LED grow lights, and ventilation systems is planned to enhance efficiency. Additionally, software development plays a crucial role in implementing data processing algorithms, automation logic, and user interaction.

This chapter begins with an overview of the design principles and objectives and then explores the system architecture, hardware selection, circuit design, software flow, and interface development. By ensuring a modular, scalable, and efficient design, this system aims to optimize fodder growth conditions, minimize resource wastage, and provide farmers with a user-friendly automated farming solution.

The following sections will present a detailed hardware and software design, along with diagrams such as block diagrams, circuit schematics, and flowcharts, to illustrate the interactions between different system components.

4.2 System Design Process

4.2.1 Use Case Diagram

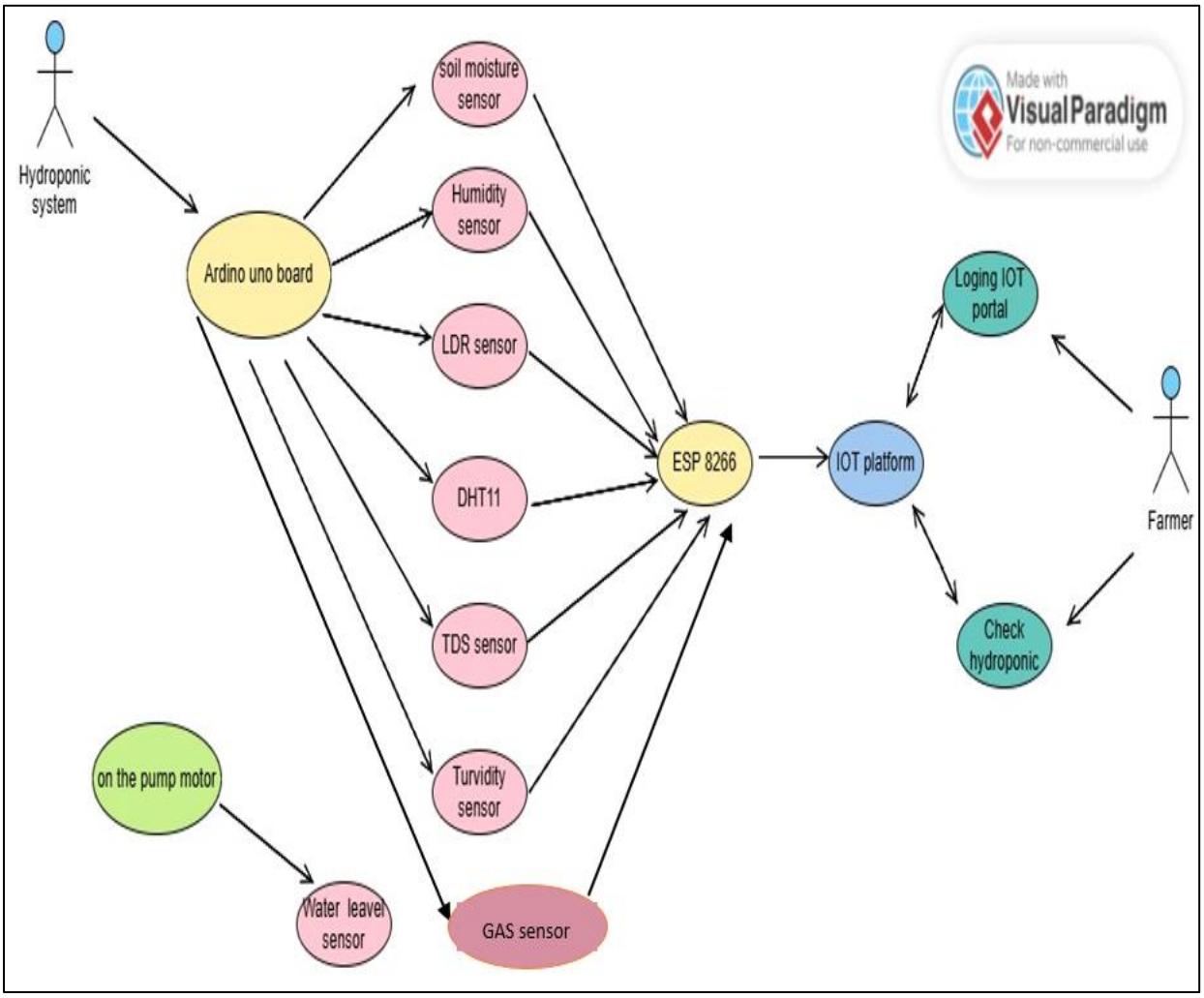

Figure 21-use case diagram

4.2.2 Block Diagram

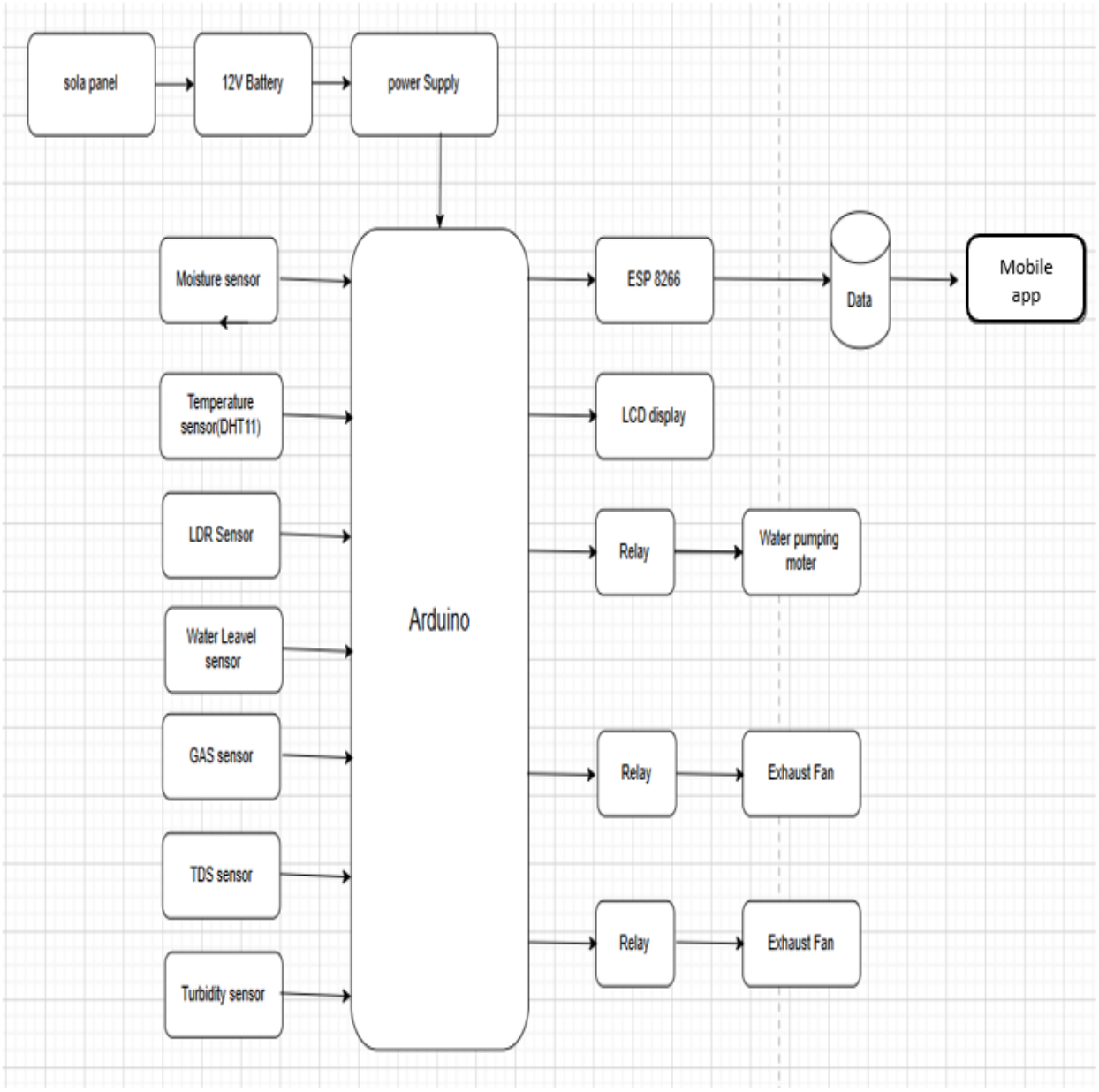

Figure 22-Block Diagram

4.2.3 Sequence Diagram

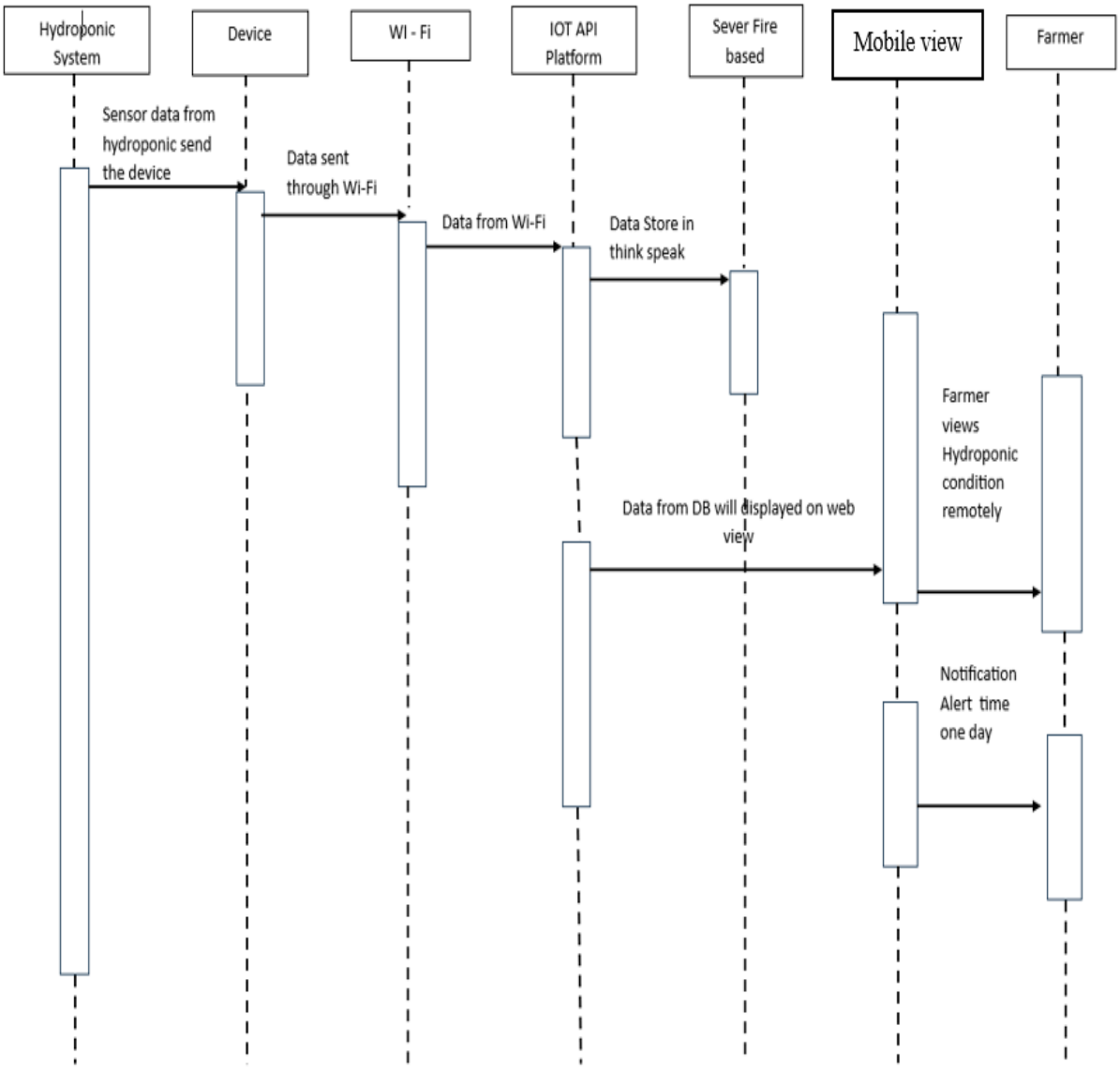

Figure 23-Sequence Diagram

4.2.4 Activity Diagram

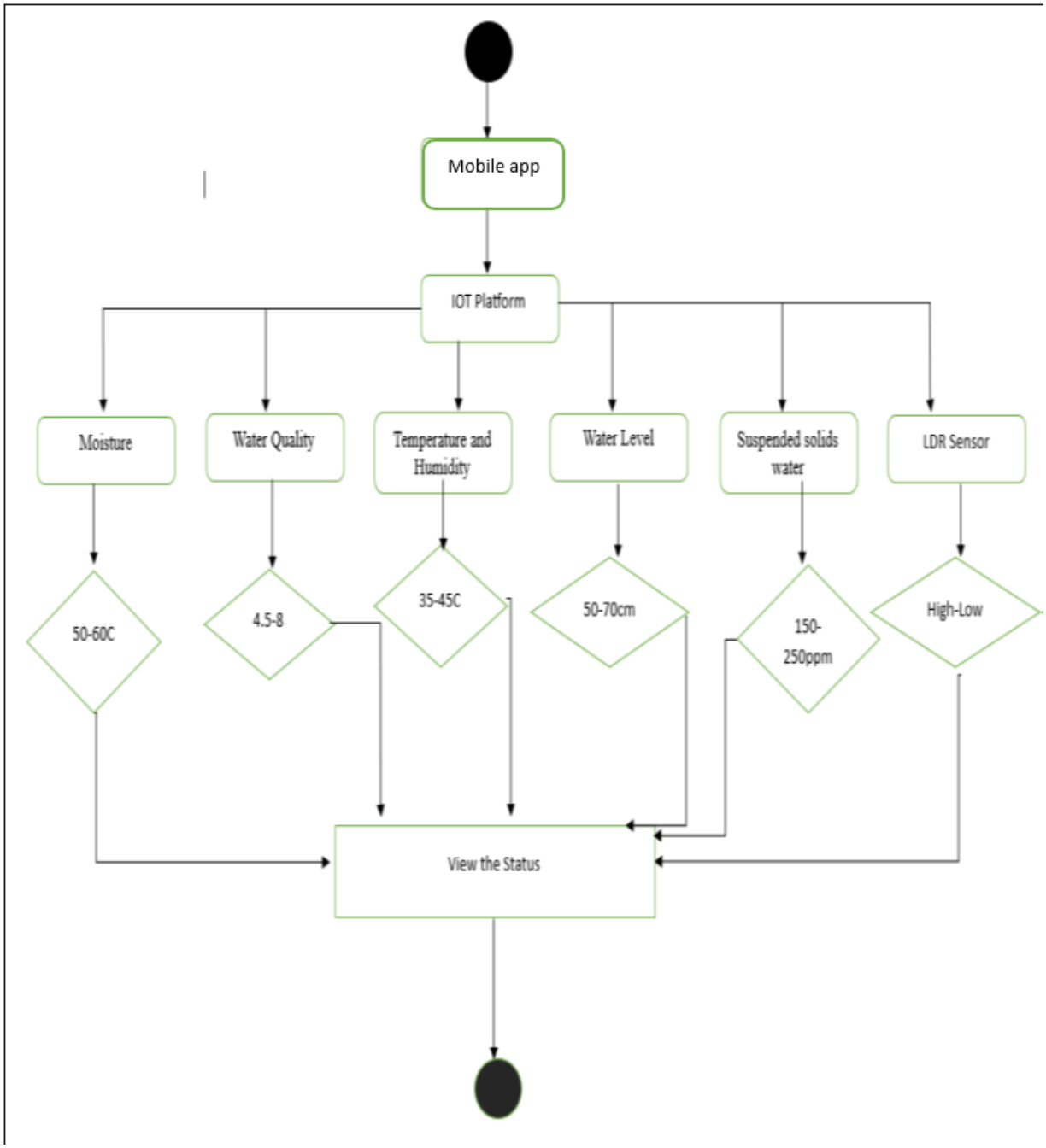

Figure 24-Activity Diagram

4.2.5 Class Diagrams

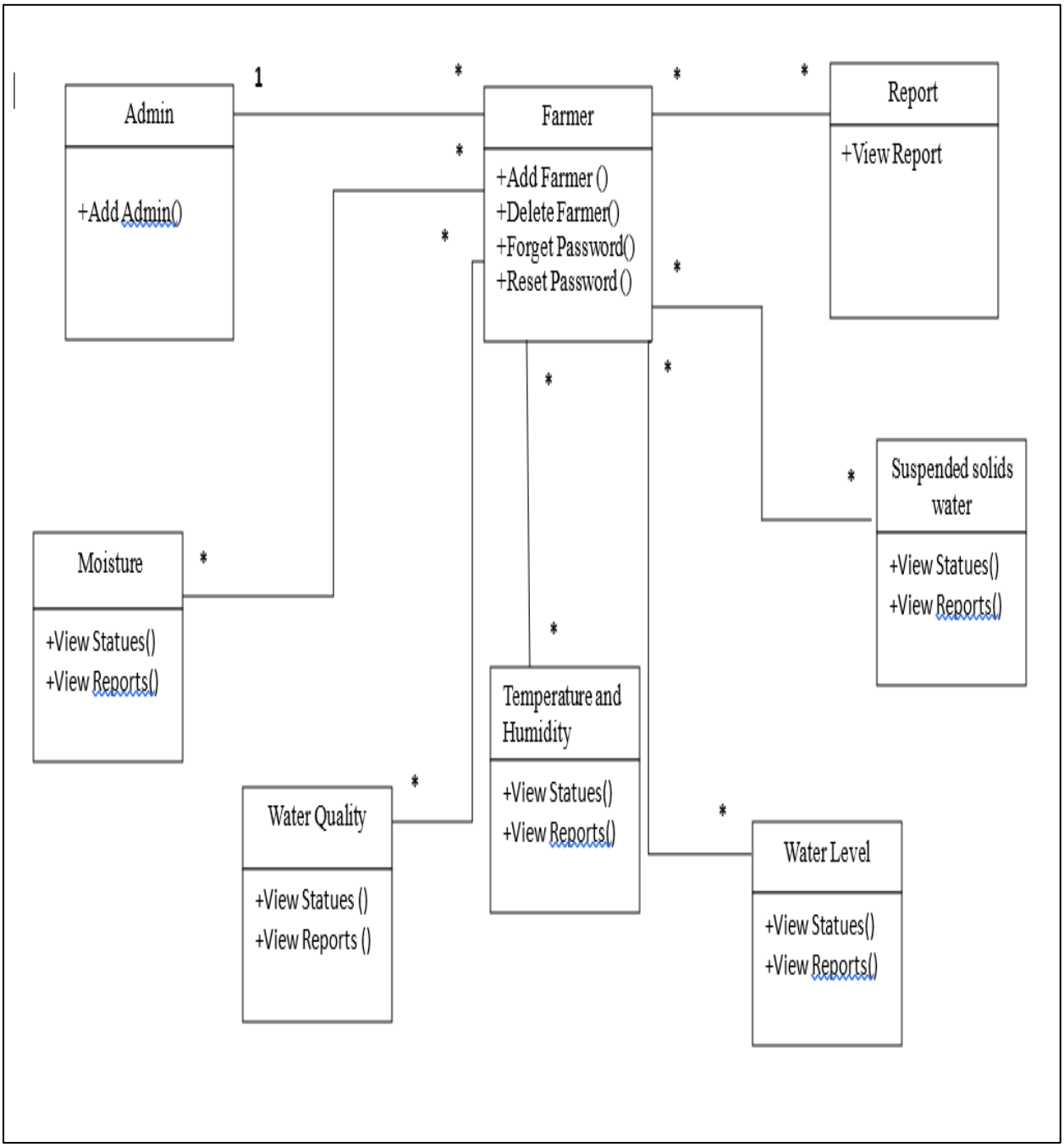

Figure 25-Class Diagrams

4.2.6 Flow chart

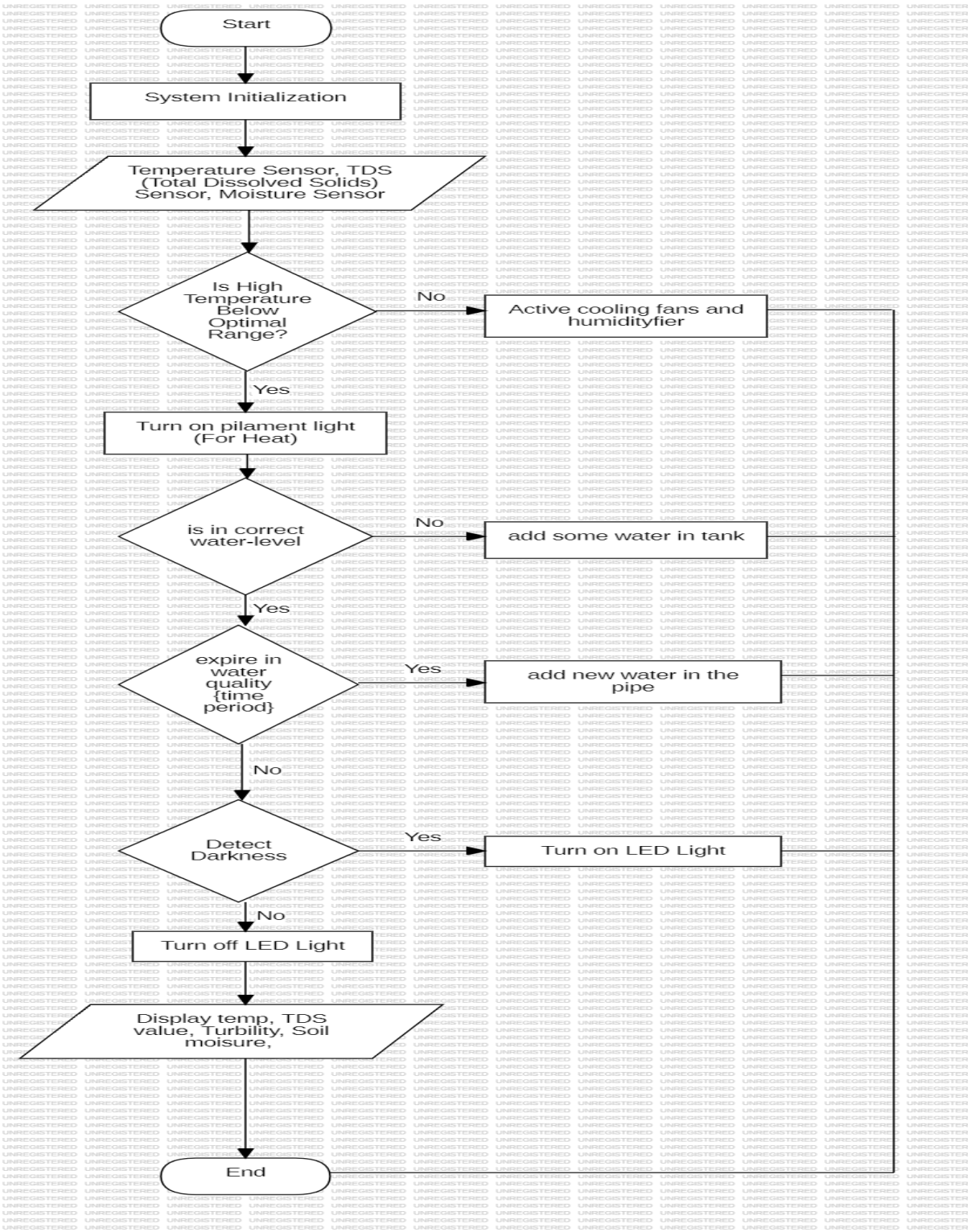

Figure 26-Flow chart

4.2.7 Design Diagram

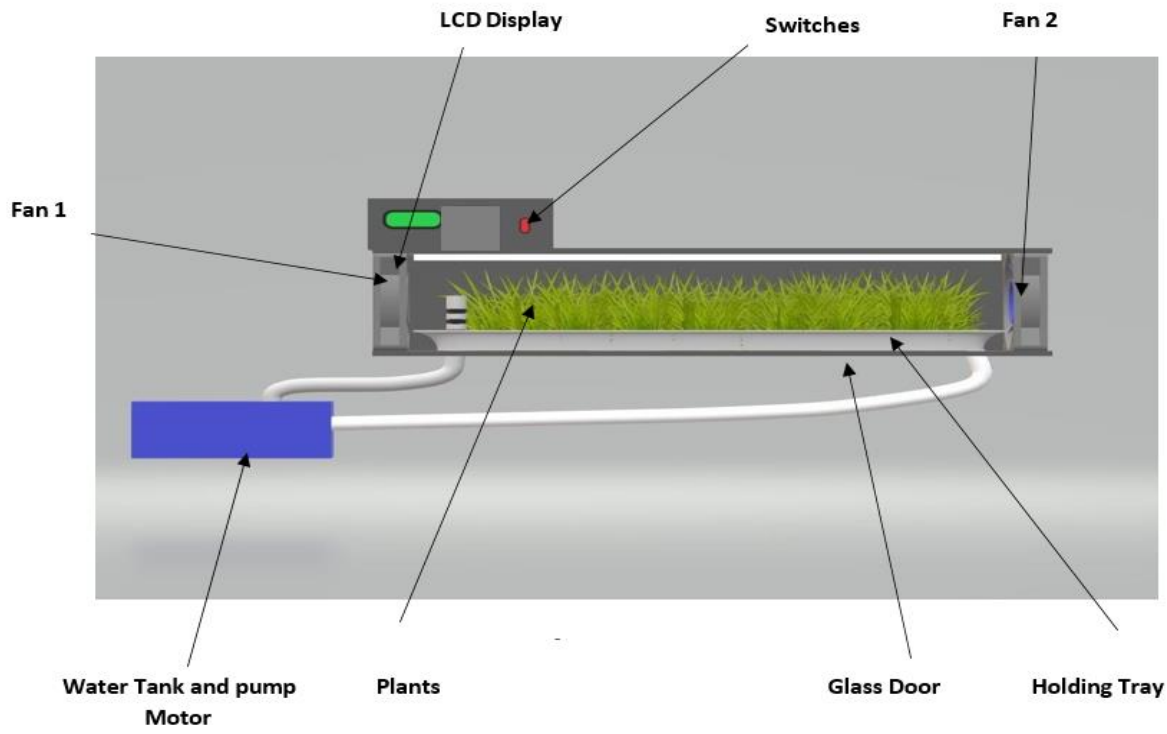

Figure 27-Design Diagram01

Figure 28-Design Diagram02

4.2.8 Circuit Diagram

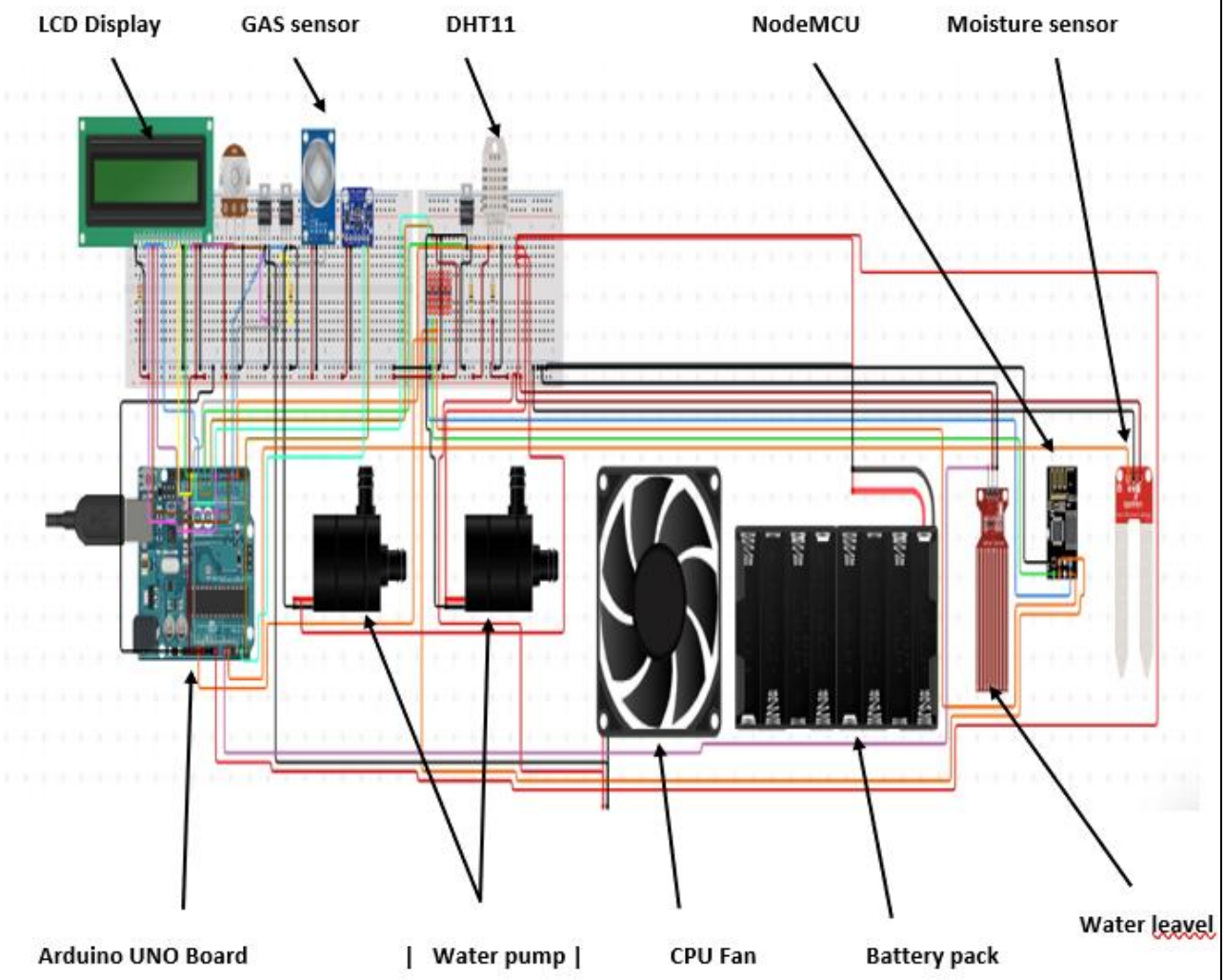

Figure 29-Circuit Diagram

4.3 Interface Design

4.3.1 Mock Screen of the System

- **Temperature and Humidity Sensor.**

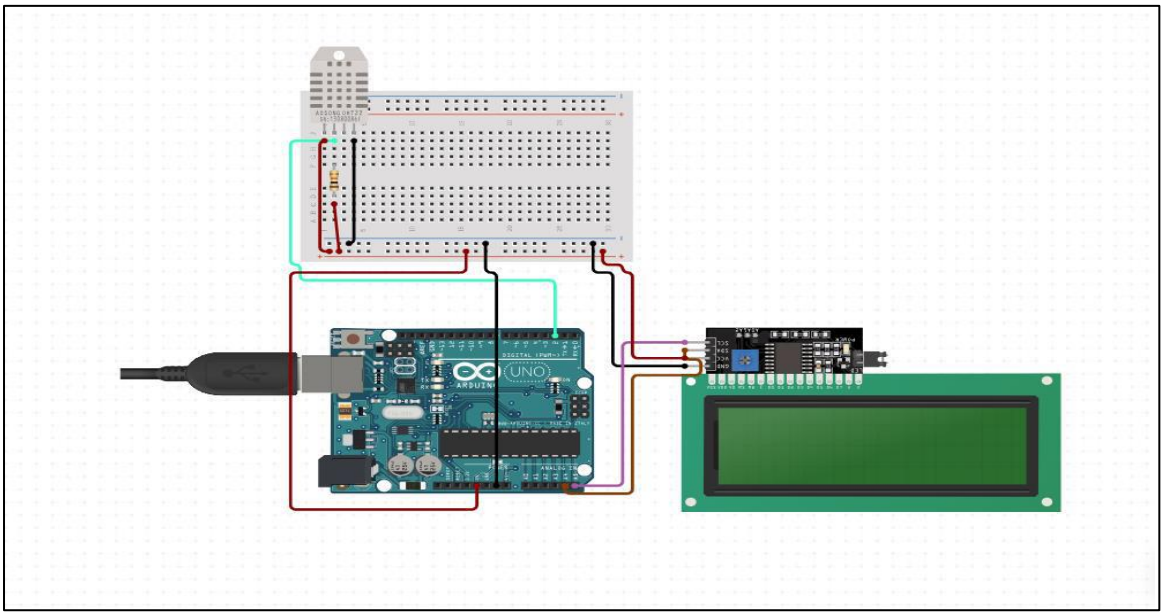

Figure 30-Temperature and Humidity Sensor

- **Soil Moisture Sensor.**

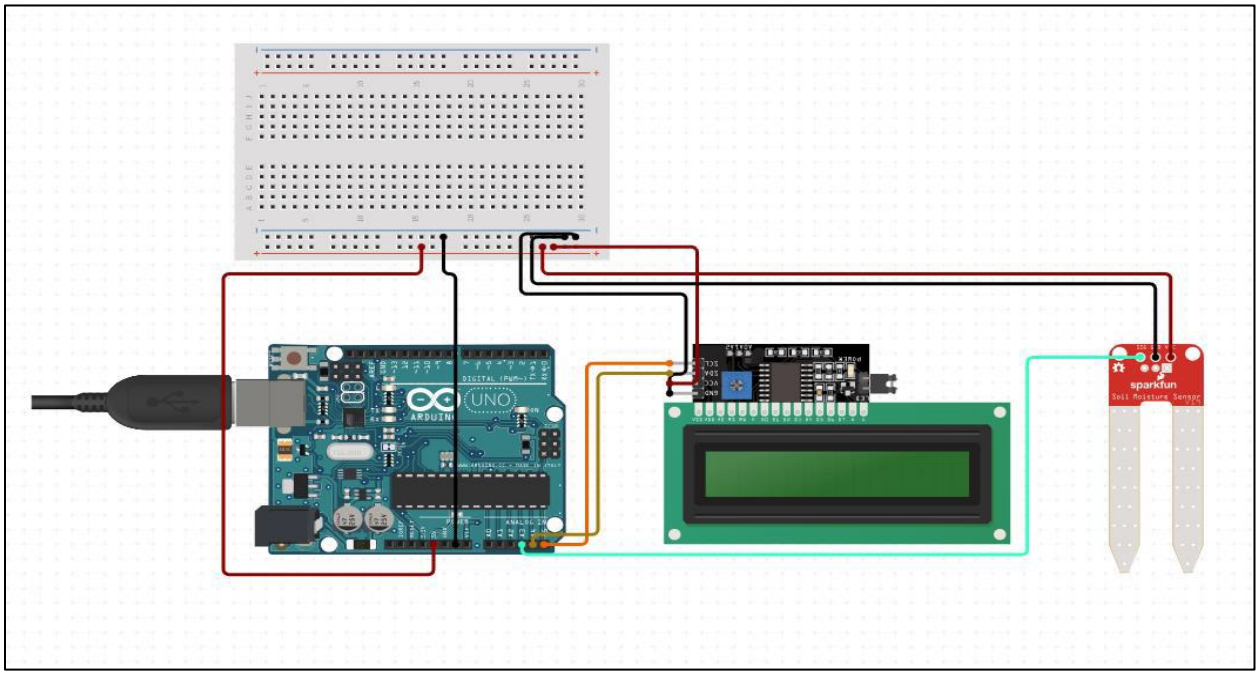

Figure 31-Soil Moisture Sensor

- **LDR Sensor.**

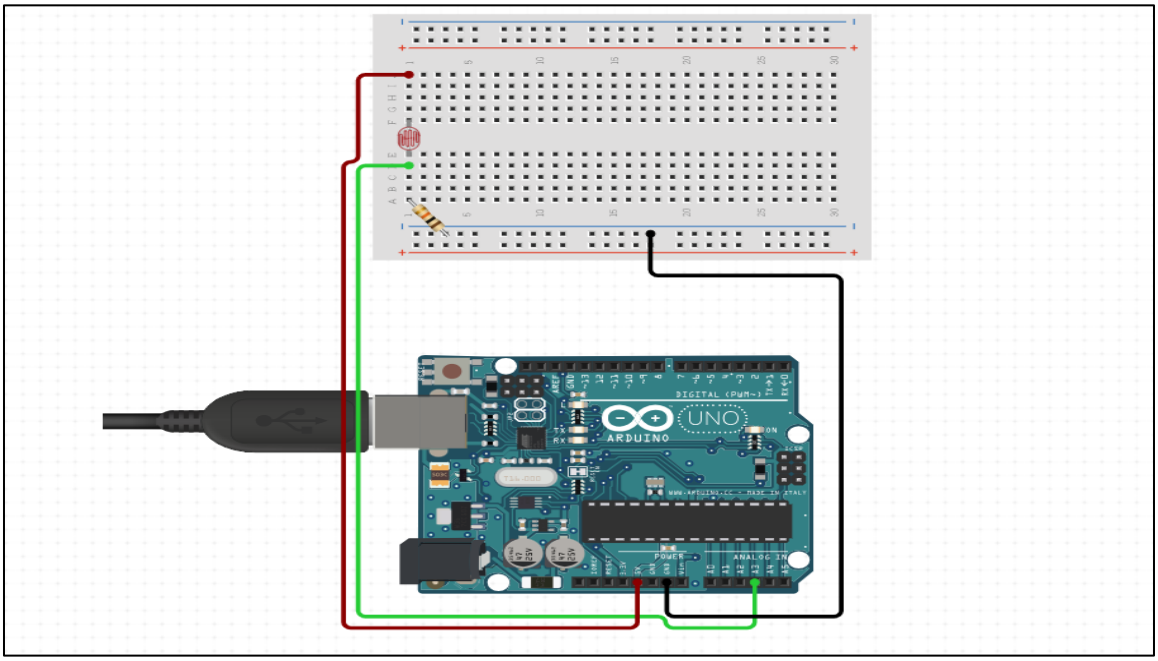

Figure 32-LDR Sensor.

- **TDS Sensor.**

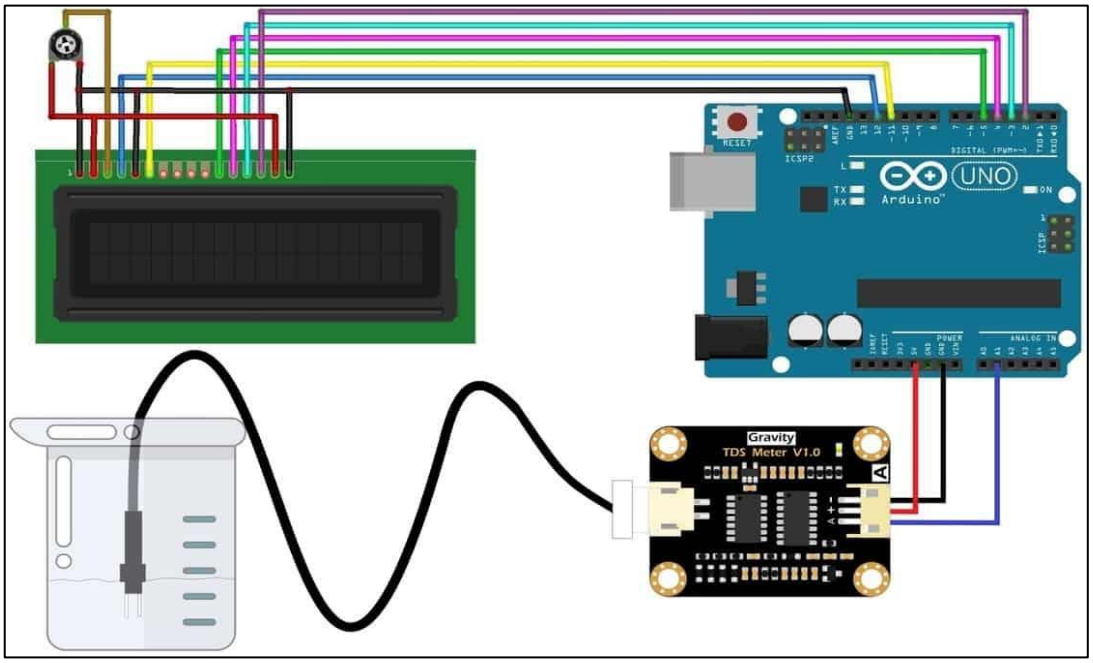

Figure 33-TDS Sensor.

- **Turbidity Sensor.**

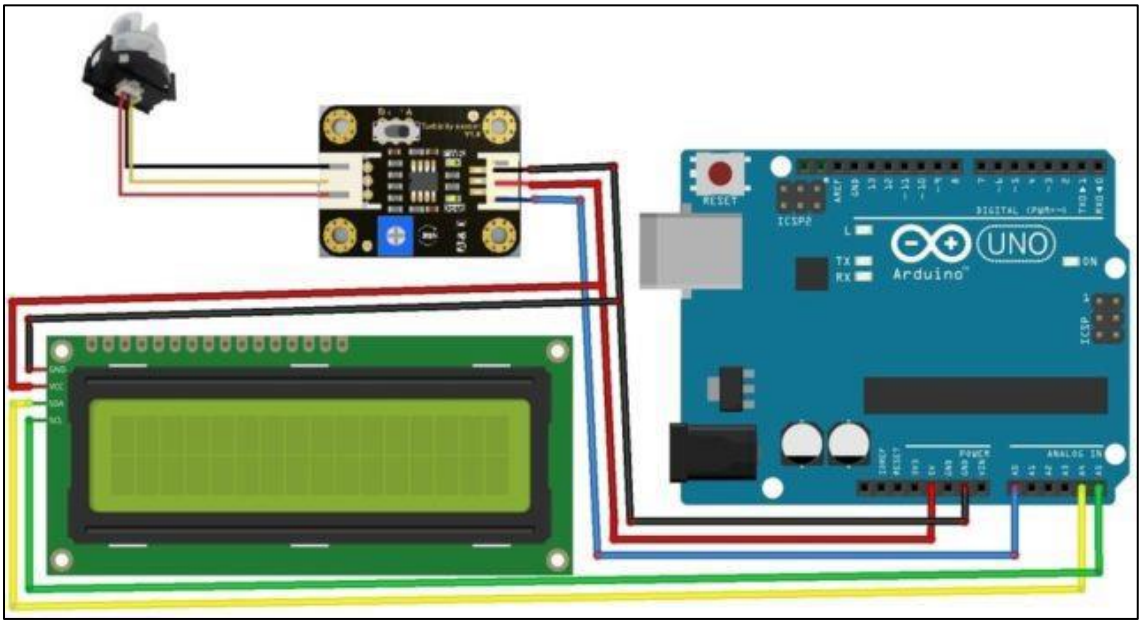

Figure 34-Turbidity Sensor.

- **Water level sensor**

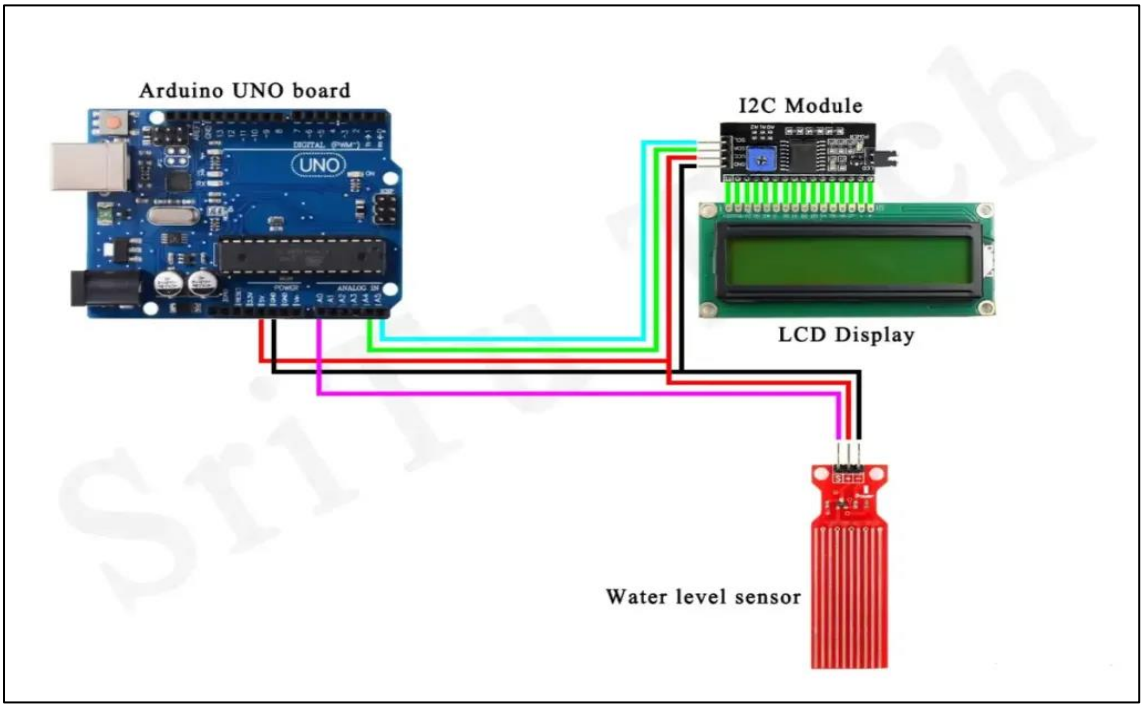

Figure 35-Water level sensor

- **GAS sensor**

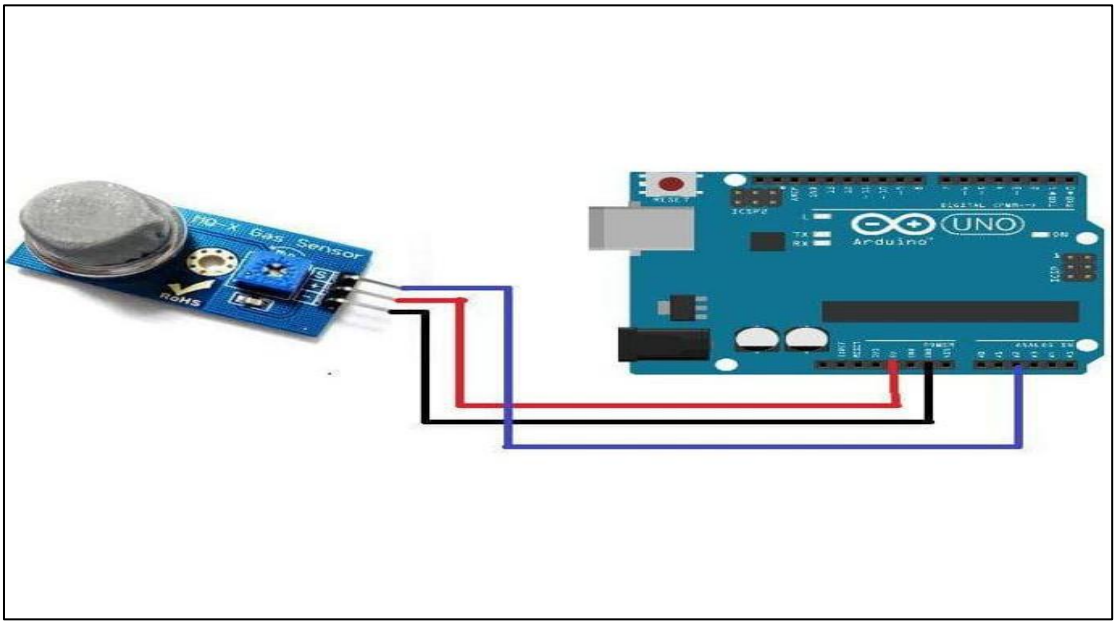

Figure 36-GAS sensor

- **Exhaust Fan and Relays**

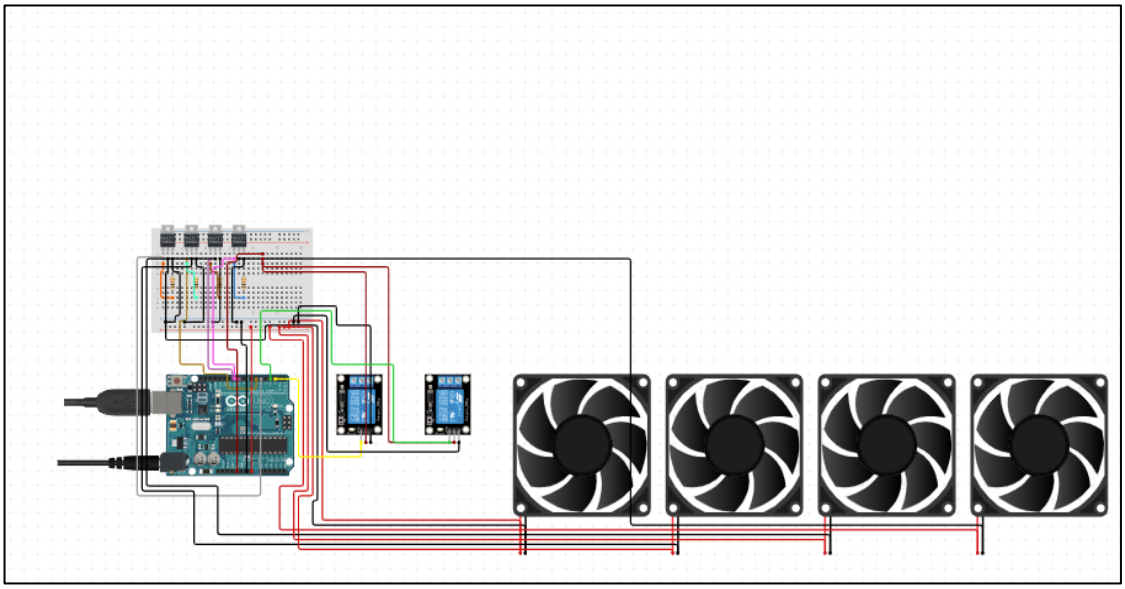

Figure 37-Exhaust Fan and Relays

- **Node MCU Board and DHT11 sensor**

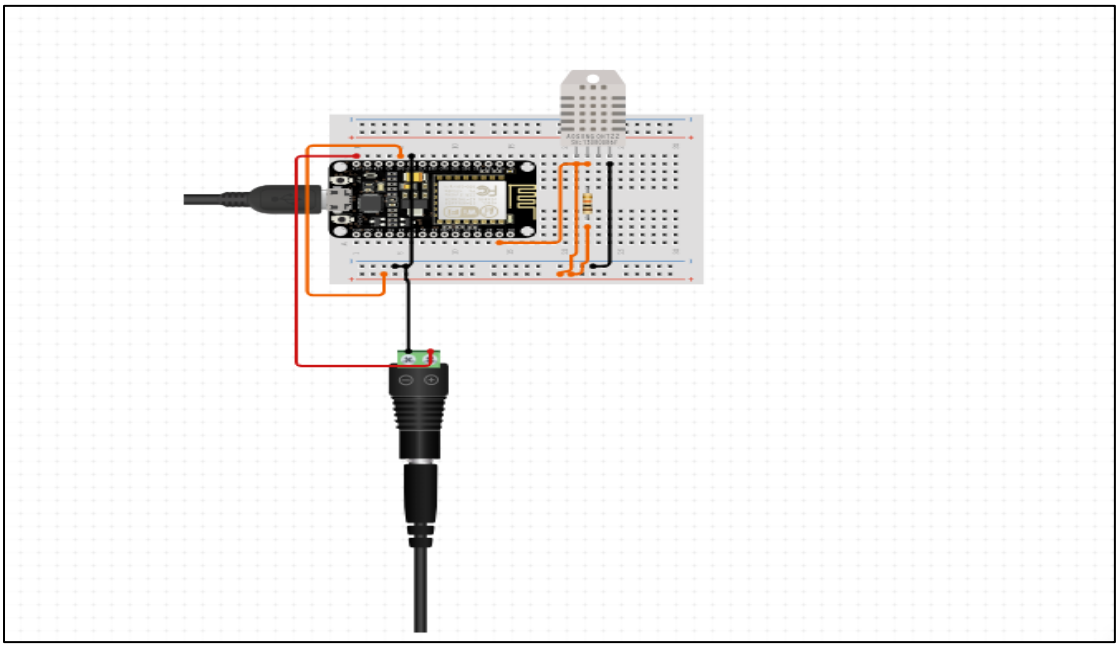

Figure 38-Node MCU Board and DHT11 sensor

- **Node MCU Board and LCD Display**

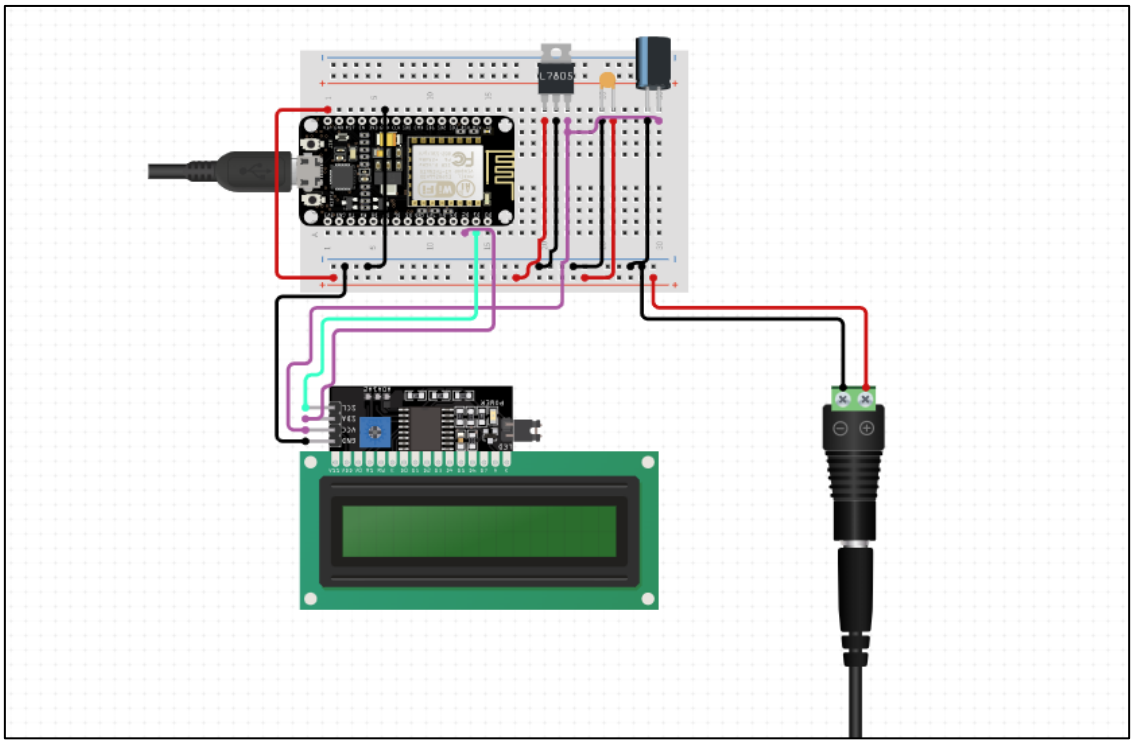

Figure 39-Node MCU Board and LCD Display

4.4 Hardware Components

- **Arduino Uno board**

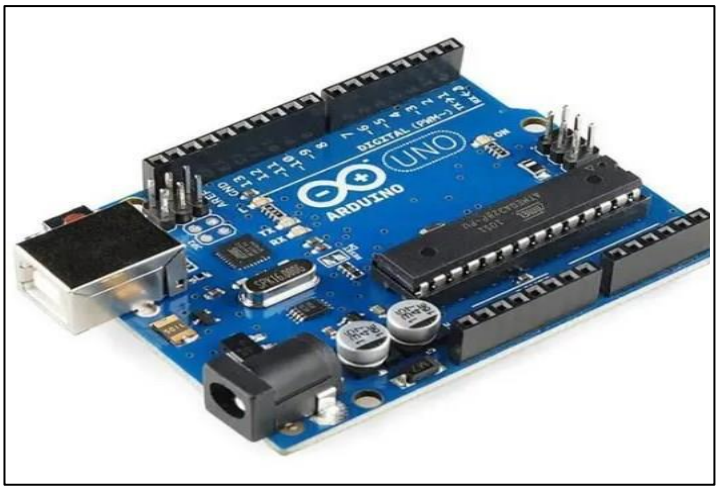

Figure 40-Arduino Uno board

The Arduino Uno is a widely used open-source microcontroller board designed to make electronics and programming more accessible to students, hobbyists, and professionals. It is built around the ATmega328P microcontroller, which acts as the brain of the board, handling input from connected sensors and controlling output devices such as LEDs, motors, or relays. The board operates at 5V and can be powered either through a USB cable(used for both programming and power) or an external adapter via a barrel jack. It features 14 digital I/O pins (six of which provide PWM output) and 6 analog input pins, making it capable of interfacing with both digital and analog components. With 32 KB of flash memory, it stores the uploaded code (called sketches), while 2 KB of SRAM is used for runtime data processing, and 1 KB of EEPROM allows for non-volatile data storage. The board also includes a 16 MHz crystal oscillator to ensure precise timing. A built-in ICSP header enables in-circuit programming, while TX/RX LEDs indicate serial communication, making debugging easier. The Arduino IDE provides a user-friendly environment for writing, compiling, and uploading code through the USB connection. One of the key advantages of the Arduino Uno is its simplicity, combined with extensive community support and a wealth of tutorials, which make it ideal for building projects such as home automation systems, robotic devices, and IoT solutions. This versatility and ease of use have made the Arduino Uno a cornerstone of learning and experimentation in the field of embedded systems.

ESP 8266 Wi-Fi Board (Node MCU Board)

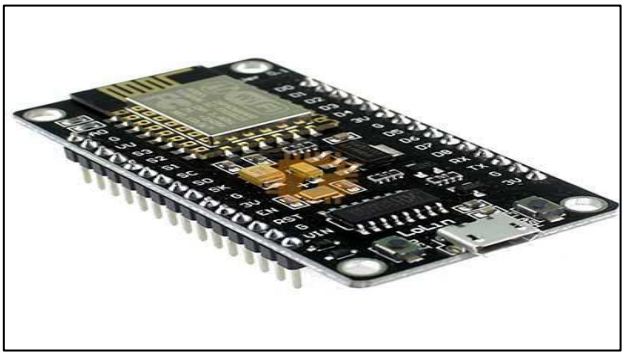

Figure 41-ESP 8266 Wi-Fi Board

The ESP8266 Node MCU is a low-cost Wi-Fi microcontroller module that allows for easy integration of Internet of Things (IoT) applications. It is based on the ESP8266 Wi-Fi SoC (System on Chip) manufactured by Espressif Systems, and comes with an open-source firmware called Node MCU that allows you to write and run Lua scripts or programs. The NodeMCU firmware is open source and allows programming in Lua, but the module can be programmed using the Arduino IDE and MicroPython, which gives it versatility in different programming environments. It features built-in Wi-Fi capabilities that comply with the 802.11 b/g/n standards, making it ideal for wireless communication in IoT projects. The module has several GPIO (General Purpose Input/Output) pins for connecting sensors and actuators. It supports analog input via a single analog pin (A0) and has a variety of communication protocols including UART, SPI, and I2C. This makes interfacing with other peripherals easy. Powered via a USB cable (5V) or a 3.3V pin, NodeMCU is widely used in home automation, sensor monitoring, and various IoT systems due to its flexibility and ease of use.

Specification

- 3.3V operating voltage, internal voltage regulator allows 5V on power input
- USB – micro USB port for power, programming and debugging
- Serial Wi-Fi transmission rate: 110-460800bps
- Available Firmware for Arduino IDE

Soil Moisture Sensor

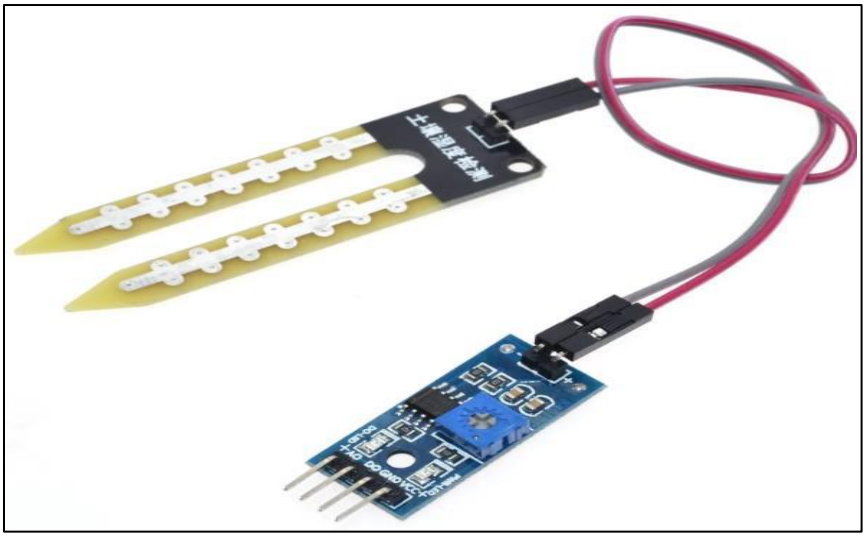

Figure 42-Soil Moisture Sensor

The Arduino soil moisture sensor is an electronic device used to measure the water content in soil, making it essential for automated irrigation, hydroponics, and gardening projects. It typically consists of two main parts: the probe (sensor) and the signal processing module. The probe has two conductive metal prongs that act as electrodes; when inserted into the soil, they measure how easily current flows through it, as water increases the conductivity. The sensor outputs either analog signals, indicating varying moisture levels (e.g., from 0 to 1023 in Arduino's ADC), or digital signals that switch based on a preset threshold. It operates at 3.3V to 5V and connects to the Arduino microcontroller via the analog (A0) or digital (D0) pins. The module's onboard potentiometer allows users to adjust the threshold sensitivity for digital output. When the soil is dry, resistance between the prongs increases, resulting in a lower output voltage, while moist soil reduces resistance, increasing the output. A typical soil moisture sensor setup with Arduino might read moisture levels and activate water pumps automatically to maintain optimal plant growth. However, the sensor can degrade over time due to corrosion from prolonged exposure to water, requiring periodic replacement or upgrades to capacitive sensors for better durability. This device is widely used in smart agriculture, greenhouse automation, and IoT-based gardening systems, offering a low-cost, efficient way to conserve water and ensure healthy crop management by automating irrigation systems based on real-time soil conditions.

Temperature and Humidity Sensor (DHT 11)

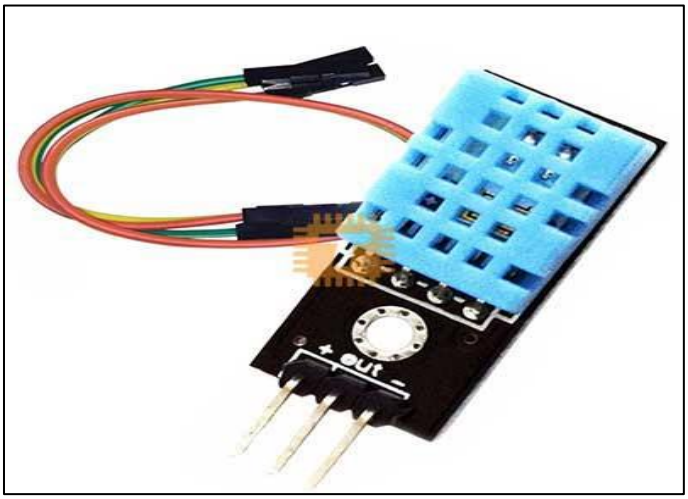

Figure 43-Temperature and Humidity Sensor

The Arduino DHT11 sensor is a low-cost, easy-to-use sensor designed to measure temperature and humidity, making it popular for weather monitoring, HVAC systems, and smart home automation. It consists of a thermistor to measure temperature and a capacitive humidity sensor to detect the moisture content in the air. The sensor is housed in a plastic shell with vent holes, which allows for air exposure while protecting internal components. The DHT11 provides digital output, eliminating the need for complex analog-to-digital conversions. It operates within a temperature range of 0°C to 50°C and a humidity range of 20% to 90% RH. Communication between the DHT11 and Arduino is handled via a single-wire protocol, using just one digital pin to transmit data. The sensor needs 3.3V to 5V for operation and requires a 2-second interval between readings to prevent data inconsistency. A key advantage of the DHT11 is its simple interface, which makes it ideal for beginners, though it has limited precision and a slower response time compared to more advanced sensors like the DHT22. In typical Arduino projects, the sensor's data is read and processed to display real-time temperature and humidity values on LCD screens, send alerts via IoT platforms, or control devices such as fans and humidifiers based on environmental conditions. However, its limited range and accuracy make it more suitable for indoor applications rather than demanding industrial environments. Overall, the DHT11 is a reliable and cost-effective solution for basic environmental monitoring tasks.

Lightning Sensor (LDR)

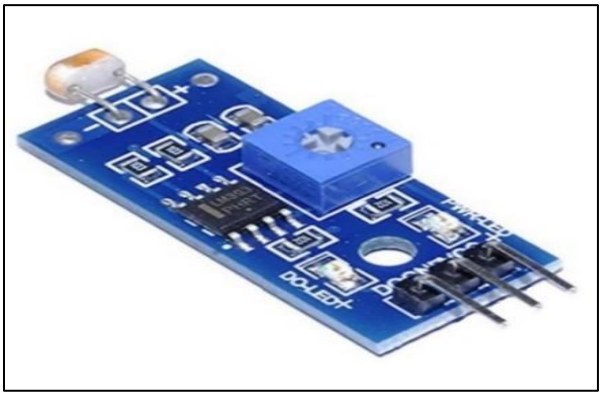

Figure 44-Lightning Sensor

The Arduino LDR (Light Dependent Resistor) sensor is a simple yet effective device used to measure light intensity in various applications, such as smart lighting systems, solar tracking, and ambient light monitoring. An LDR, also known as a photoresistor, is a passive component whose resistance decreases as the intensity of light falling on it increases. It operates on the principle of photoconductivity, meaning the material inside the LDR becomes more conductive when exposed to light. Typically, in darkness or low-light conditions, the LDR exhibits high resistance (in the range of megaohms), while in bright light, the resistance can drop to a few hundred ohms.

In an Arduino setup, the LDR is connected in a voltage divider circuit alongside a fixed resistor, and the changing resistance of the LDR produces a varying voltage, which is fed to one of the analog input pins of the Arduino. The Arduino reads this voltage as an analog value which corresponds to the detected light intensity. By analyzing the sensor data in real-time, the Arduino can trigger actions such as turning on/off lights when the ambient light falls below or rises above a certain threshold. LDRs are commonly used in automatic street lighting systems, where they help turn on streetlights at night and off during the day.

However, LDRs are not very precise compared to advanced light sensors like photodiodes or phototransistors, and they exhibit slower response times. Additionally, their readings can be affected by factors such as temperature and aging, which may slightly degrade performance over time. Despite these limitations, LDRs are widely used in DIY projects and educational setups due to their simplicity, affordability, and ease of integration with microcontrollers like Arduino.

Turbidity Sensor

Figure 45-Turbidity Sensor

The Arduino turbidity sensor is a device used to measure the cloudiness or haziness of a liquid, which indicates the presence of suspended particles such as dirt, algae, or pollutants. It is widely used in applications such as water quality monitoring, hydroponics, aquaculture, and wastewater treatment plants. The most common turbidity sensor for Arduino projects is based on light scattering and absorption principles. It typically consists of an infrared LED (light source) and a photodetector (receiver), arranged such that light passes through the water sample. When the liquid is clear, most of the light passes straight through and is detected by the receiver, resulting in higher output voltage. In contrast, when the liquid contains suspended particles, they scatter or absorb the light, reducing the amount detected, which lowers the output voltage.
The turbidity sensor connects to the analog input pins of the Arduino, where the varying voltage is read and converted into a corresponding turbidity level (measured in NTU – Nephelometric Turbidity Units). These readings help determine the clarity of the liquid, which is critical for ensuring healthy aquatic environments or clean water supplies. The sensor typically operates at 5V and outputs an analog voltage signal proportional to the turbidity. Some models also provide a digital output that triggers when the turbidity exceeds a user-defined threshold, adjusted via an onboard potentiometer. While turbidity sensors are affordable and easy to integrate, they have some limitations, such as sensitivity to temperature, fouling, and bubbles in the water that can affect readings. Routine calibration with standard turbidity solutions is required for maintaining accuracy.

TDS Sensor

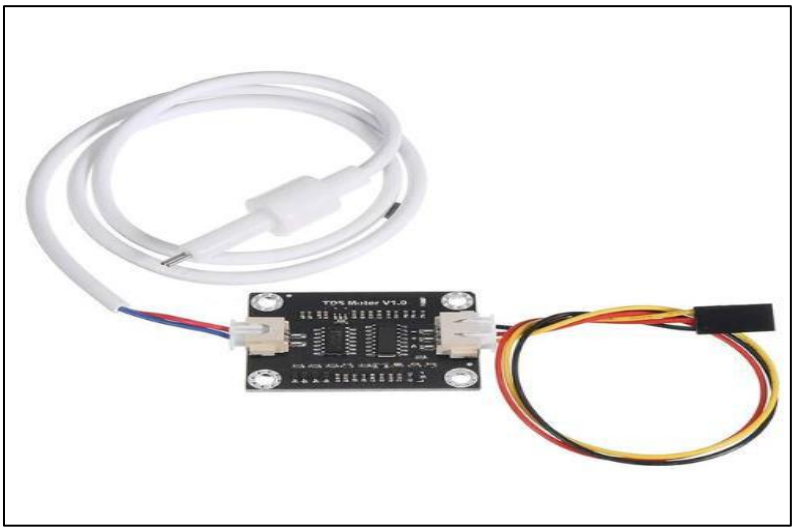

Figure 46-TDS Sensor

The Arduino TDS (Total Dissolved Solids) sensor is used to measure the concentration of dissolved solids in water, which indicates its purity or salinity. TDS sensors are essential in applications like hydroponics, aquaponics, water purification systems, swimming pools, and aquariums, as they monitor the concentration of substances such as salts, minerals, and nutrients. The sensor works by measuring the electrical conductivity (EC) of water since dissolved ions increase conductivity. It contains two or more metal probes that come into direct contact with the water. When voltage is applied across the probes, ions in the water allow a small current to pass through, and the sensor converts the conductivity into a TDS value, typically expressed in ppm (parts per million).

In an Arduino setup, the TDS sensor module outputs an analog voltage signal proportional to the TDS level, which can be read through the analog input pin (e.g., A0) of the Arduino. The microcontroller processes the signal and converts it to a TDS value using a specific calibration factor. Some sensors also provide temperature compensation since water temperature affects conductivity; thus, accurate measurements require either built-in or external temperature sensors. TDS sensors typically operate at 5V and are effective within a range of 0 to 1000 ppm or more, making them suitable for monitoring tap water, hydroponic nutrient solutions, and aquaculture systems.

However, continuous exposure to water can cause scaling or fouling on the probes, requiring regular maintenance and recalibration to ensure accuracy. Additionally, the sensor's precision might degrade in highly polluted or very low-ion solutions.

MQ2 Gas sensor

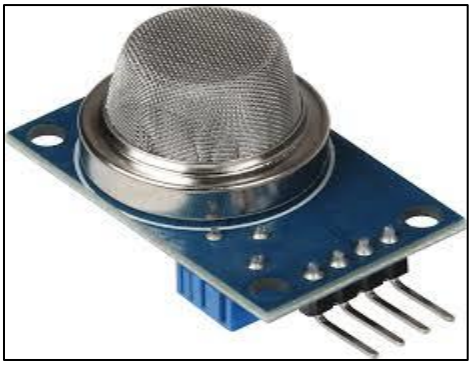

Figure 47-MQ2 Gas sensor

The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as Chemiresistors because sensing is based on the change in resistance of the sensing material when exposed to gasses. The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW. It can detect LPG, Smoke, Alcohol, Propane, Hydrogen, Methane and Carbon Monoxide concentrations ranging from 200 to 10000 ppm. The MQ2 is a heater-driven sensor. It is therefore covered with two layers of fine stainless-steel mesh known as an "anti-explosion network". It ensures that the heater element inside the sensor does not cause an explosion because we are sensing flammable gasses. MQ2 is widely used in applications such as smoke detection, LPG leak alarms, fire protection systems, and air quality monitoring systems.

Specifications and Features

The MQ2 gas sensor operates at a standard voltage of 5V DC and has a power consumption of around 800mW. It can detect a wide range of gases, making it versatile for multiple applications. Some of the key specifications include:

Methane Detection Range: 500 – 10000 ppm

Carbon Monoxide (CO) Detection Range: 20 – 2000 ppm

Operating Temperature: -10°C to 50°C

Humidity Resistance: Below 95% RH

Water level sensor

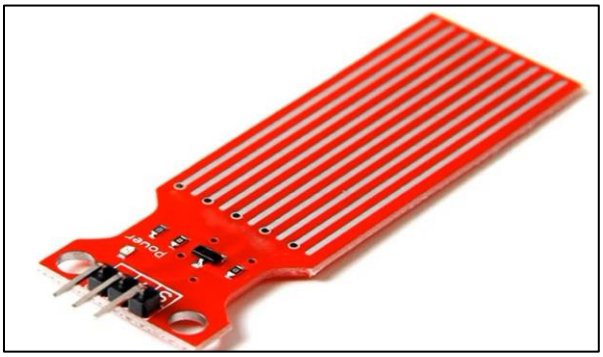

Figure 48-Water level sensor

Water is an important resource in residential and commercial settings, and its use must effectively be monitored for its use in a continuous state. One of the most important tools involved in water management is a water level sensor. Water level sensor is an electronic device with a function of sensing, tracking, and controlling water level in a tank, reservoir, and any holding system for a liquid. Water level sensors conserve water and its accompanying assets from overflows, leakages, and run-outs, allowing effective use of water in a variety of operations including agricultural, processes, residential water tank, and hydroponics.

Water Level Sensor Functionality

Water level sensor function is through presence and absence of water at a level and its conversion to an electrical output. Microcontrollers, programmed programs, and alarm triggers respond in consequence of information processing received. Water level sensors can detect water level in a continuous level or at a level, depending on its configuration.

For instance, float sensors utilize a floating mechanism that moves between two positions with an elevation and drop in water level, toggling a switch when a level is attained. Ultrasonic sensors utilize sound waves for distance calculation between a point in contact with water and its level, estimating its actual level with no contact involved. Capacitive and resistive sensors work through electrical property values changing with contact with a conductive substance

Relay Module

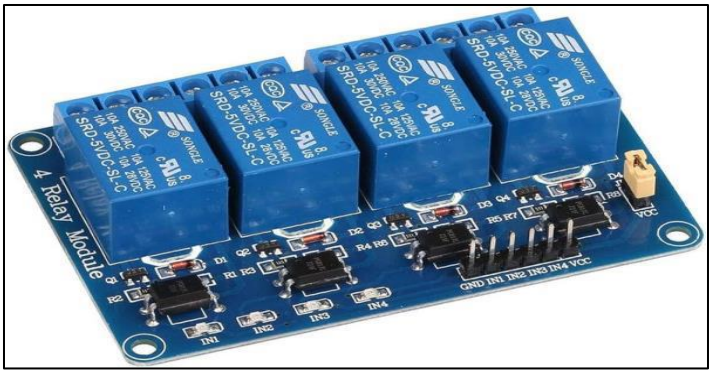

Figure 49-Relay Module

The Arduino relay module is an electronic switching device used to control high-voltage or high-current loads, such as appliances, motors, lights, fans, or pumps, with the low-voltage digital output from an Arduino microcontroller. Relays function as electrically operated switches that allow a small control signal from the Arduino to isolate and manage larger loads safely. A typical 5V relay module consists of an electromagnetic coil, a mechanical switch (normally open and normally closed terminals), and a transistor driver circuit that allows the low-power Arduino pin to trigger the relay. When the Arduino sends a HIGH signal to the relay input pin, it energizes the coil, creating a magnetic field that pulls the switch to the opposite position, enabling or disabling the connected device.

Relay modules often come with optocouplers (optical isolators) that protect the Arduino from high-voltage spikes by electrically isolating the microcontroller from the relay's high voltage side. They typically have 3 input pins (VCC, GND, and IN) for connection to the Arduino, along with screw terminals for the load. Depending on the configuration, the relay can work in either normally open (NO) or normally closed (NC) mode meaning it can allow or block current by default until it receives a signal to switch state. For example, a NO configuration keeps a device off until the relay is activated, while an NC setup keeps it on until the relay switches it off.

Arduino relay modules are available in single-channel or multi-channel (2, 4, 8-channel) versions, depending on the number of devices to control. They are widely used in home automation systems to remotely control appliances or in industrial automation to manage motors and valves. However, due to the mechanical nature of the switch, relays have a limited lifespan and might wear out with frequent use.

5V Pump Motor

Figure 50-5V Pump Motor

An Arduino 5V pump motor is a small DC-powered water pump that is used in automated liquid handling systems such as hydroponics, aquaponics, plant irrigation systems, aquariums, and water circulation projects. This type of pump is designed to operate at a low voltage of 5V, making it compatible with Arduino microcontrollers and suitable for projects that require energy-efficient water flow or small-scale fluid transfer. The motor inside the pump uses electromagnetic coils to rotate an impeller or fan-like blade, which creates suction to draw water in through an inlet and expel it through an outlet. These pumps typically operate in a submersible or inline mode, meaning they can either be fully submerged in water or connected to a pipe system to move liquids externally.

The Arduino controls the 5V pump by sending signals to an NPN transistor, MOSFET, or relay module since the pump’s current draw (often around 150–300mA) can exceed the safe limit of Arduino's GPIO pins. When activated, the pump starts circulating water, making it ideal for watering plants at intervals or replenishing water levels in hydroponic systems based on data from moisture sensors or water level sensors. The pump's compact size and low power consumption allow it to run for long periods without consuming significant energy. However, it may have limitations in pressure and flow rate, typically producing 80–120 liters per hour (L/H), making it best suited for low-flow, small-scale applications.

While these pumps are inexpensive and easy to integrate, they require regular maintenance to prevent clogging, especially if used with non-filtered water. Additionally, prolonged dry running can damage the impeller or motor .

LCD Display

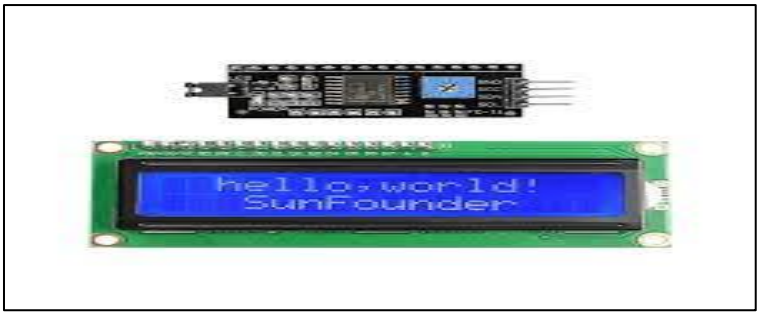

Figure 51-LCD Display

An Arduino LCD (Liquid Crystal Display) is a screen used to display alphanumeric characters, symbols, or simple graphics, making it essential for projects that require visual feedback, such as weather stations, IoT devices, home automation systems, and sensorbased monitoring setups. One of the most common models is the 16x2 LCD, which can display two rows of 16 characters each. It uses a HD44780 driver or a similar controller, making it easy to interface with Arduino microcontrollers. The LCD works by passing electrical signals to the liquid crystal layer, which manipulates light to form characters. This model supports 5x8 pixel matrix characters and can also display custom-designed characters like degree symbols or icons.

To connect a 16x2 LCD to an Arduino, it requires several digital I/O pins typically 6 pins (RS, E, D4, D5, D6, and D7) for data transfer, along with VCC (5V), GND, and an optional contrast pin connected via a potentiometer to adjust screen brightness. However, to reduce the number of required pins, an I2C (Inter-Integrated Circuit) adapter module is often used, which enables communication over just two pins: SDA and SCL. The LCD is powered by 5V, matching the Arduino's voltage, and data can be displayed dynamically in response to sensor inputs, button presses, or time-based events.

Using the Arduino Liquid Crystal library, developers can easily program the LCD to print messages, sensor readings, or instructions, and even clear or update the display in realtime. These LCDs are well-suited for applications where readability and simplicity are more important than complex graphics, making them ideal for low-power, embedded projects. However, they do not support backlight color changes unless modified, and the refresh rate is relatively slow for rapidly changing data.

Exhaust Fan

Figure 52-Exhaust Fan

The exhaust fan plays a crucial role in the Microcontroller-Driven Indoor Hydroponic Fodder System by ensuring proper air circulation, temperature control, and humidity regulation. Since hydroponic fodder systems operate in enclosed environments where moisture, temperature, and air quality directly affect plant growth, an exhaust fan helps to maintain optimal conditions.

Without proper ventilation, these factors could lead to high humidity, fungal growth, and poor air quality, negatively impacting plant health. The exhaust fan helps by:

Controlling Humidity -The fan removes excess moisture from the air, preventing mold and mildew formation on the fodder. This reduces the chances of plant diseases caused by excessive dampness.

Regulating Temperature -High temperatures can slow down fodder growth or damage the plants. The exhaust fan helps to cool the system by expelling warm air and allowing fresh air to enter.

Improving Air Circulation -Proper airflow prevents stagnant air, ensuring a constant supply of oxygen for healthy plant metabolism. This also helps in CO_2 exchange, which is necessary for photosynthesis.

Preventing Unpleasant Odors -Over time, hydroponic systems may develop odors due to decomposing organic matter.The exhaust fan helps in removing these odors and maintaining a fresh and clean environment.

Jumper Wires.

Figure 53-Jumper Wires

Jumper wires are flexible, insulated wires used to make temporary electrical connections between components on a breadboard or between a breadboard and devices like Arduino boards, sensors, modules, and other circuits. These wires are essential in prototyping and circuit development because they allow quick and easy wiring without the need for soldering. Jumper wires typically come in three types: male-to-male, female-to-female, and male-to-female, depending on the type of connectors at each end. Male connectors fit into female headers or breadboard holes, while female connectors fit onto pins of components or modules. This versatility makes them suitable for connecting different types of electronic devices.

Jumper wires are available in various lengths and colors to help organize circuits and easily distinguish between connections. They are often bundled as sets with multiple colors, helping developers follow circuit paths clearly and avoid wiring errors. The conductive core inside the insulation is typically made of copper or tinned copper for low resistance, ensuring efficient electrical transmission. The outer PVC insulation protects the wire from short circuits or accidental contact.In Arduino projects, jumper wires are commonly used to connect sensors, relays, displays, motors, or other modules to the Arduino's GPIO (general-purpose input/output) pins. For example, a male-to-female jumper wire might connect an Arduino’s pin to a sensor’s input, while male-to-male wires are used extensively on breadboards to route signals between components. Since they are designed for temporary connections, jumper wires are perfect for testing and debugging circuits.

CHAPTER 5 – SYSTEM DEVELOPMENT

5.1 Navigation / Module Structure

The ESP8266 is a low-cost, multi-purpose, and powerful Wi-Fi module with capabilities of providing microcontrollers with access to a network and Mobile -related actions. It is increasingly utilized in IoT (Internet of Things) implementations for its TCP/IP stack integration, low consumption, and small form factor.

The module ESP8266 in this work is accompanied with an added ESP32 microcontroller for providing information about sensors to a cloud via a Firebase database. With it, farmers can access Mobile portals remotely for values in a hydroponic system, such as temperature, humidity, and water level.

ESP8266 Microcontroller Initialization.

Temperature and Humidity Sensor.

The DHT-11(also named as AM2302) is a digital-output, relative humidity, and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends a digital signal on the data pin. In this project, this sensor with Arduino UNO The hydroponics humidity and Temperature will be sent on the Cloud.

- Power − 3-5V
- Max Current − 2.5mA
- Humidity − 0-100%, 2-5% accuracy
- Temperature − 40 to 80°C, ±0.5°C accuracy

Operating Conditions:

- Operating Temperature: 0°C to 50°C
- Storage Temperature: -20°C to 60°C
- Operating Humidity: 20% to 90% RH
- Storage Humidity: 0% to 80% RH

Soil Moisture Sensor.

The sensor measures moisture by detecting the electrical conductivity or resistance between two probes. Since water increases the conductivity of soil, the more moisture present, the higher the conductivity (or lower the resistance). The sensor outputs either an analog signal representing the moisture level or a digital signal when a predefined threshold is reached.

Technical Specifications

- Operating Voltage: 3.3V – 5V
- Current Consumption: 20 mA
- Dimensions: Probe length 60mm; Module 30mm x 15mm
- Response Time: 1 second.

Components: -

- Probes:

- Two metal probes inserted into the soil to measure resistance/conductivity.
- Corrosion-resistant versions (e.g., capacitive sensors) are available for longer durability.
- Control Module:
- Converts the signal from the probes into analog and digital outputs.
- Onboard potentiometer for adjusting the threshold of the digital output.
- Pins:
- VCC: Connects to 3.3V or 5V power supply.
- GND: Ground connection.
- A0: Analog output to the Arduino's analog pin.
- D0: Digital output (triggers HIGH or LOW).

Lightning Sensor.

An LDR sensor, also known as a photoresistor or light-dependent resistor, is a light-sensitive component used to detect changes in ambient light intensity. It is made from semiconductor materials that change their resistance based on the amount of light falling on their surface. This makes it ideal for Arduino projects requiring light sensing, such as automatic streetlights, light meters, solar tracking systems, and home automation systems.

Technical Specifications.

- Operating Voltage: 3.3V – 5V (when used with microcontrollers)
- Resistance Range:
- In darkness: 1 MΩ or higher
- In bright light: 1 kΩ – 10 kΩ

- Response Time:
- Rise Time: 20-30 ms (for increasing light)
- Decay Time: 30-50 ms (for decreasing light)
- Operating Temperature: -30°C to +75°C
- Material: Cadmium Sulfide (CdS) or other photoconductive materials.

TDS Sensor.

A TDS sensor measures the concentration of total dissolved solids (TDS) in water, expressed in ppm (parts per million). Dissolved solids include inorganic salts, minerals, and small organic particles that can influence water quality. This sensor plays an important role in monitoring water purity in various applications, such as hydroponics, aquariums, drinking water filtration systems, pools, and agricultural irrigation. When connected to an Arduino or another microcontroller, it enables real-time water quality measurements and automated responses. [9]

Technical Specifications: -

- **Operating Voltage**: 3.3V – 5V
- **Measurement Range**: 0 – 1000 ppm (some sensors extend up to 5000 ppm)
- **Accuracy**: ±10 ppm
- **Signal Output**: Analog voltage (0 to 2.3V range)
- **Response Time**: 500 ms
- **Sensor Material**: Stainless steel probe (corrosion-resistant)
- **Operating Temperature**: 0°C to 80°C
- **Temperature Compensation**: Some sensors support temperature compensation for accuracy.

Turbidity Sensor.

A turbidity sensor measures the cloudiness or haziness of a liquid caused by the presence of suspended particles such as silt, algae, and organic matter. It is widely used in water quality management systems, including aquariums, hydroponics, drinking water plants, wastewater treatment, and environmental monitoring. In Arduino-based projects, turbidity sensors help detect water contamination and ensure optimal water conditions.

Technical Specifications: -

- Operating Voltage: 5V
- Measurement Range: 0 – 1000 NTU
- Output Signal:
- Analog output: 0-4.5V (for continuous monitoring)
- Digital output: HIGH/LOW (for threshold detection)
- Response Time: < 500 ms
- Operating Temperature: 0°C to 80°C
- Cable Length: Typically, 1 meter (for probe-type sensors)
- Material: Sensor casing is often made of plastic or acrylic for water resistance.

Water Level Sensor

A water level sensor is an electronic device used to measure the level of water in a container, tank, reservoir, or hydroponic system. It plays a crucial role in water management, preventing overflow, leaks, or dry conditions. Water level sensors are widely used in automated irrigation systems, hydroponics, wastewater treatment, and industrial water storage.

In Arduino-based projects, water level sensors provide real-time data on water levels, allowing automation of water pumps and alerts for low or high water levels to ensure efficient system operation.

Technical Specifications:

- Operating Voltage: 3.3V – 5V
- Measurement Range: 0 – 100% water level detection
- Output Signal:
- Analog Output: Varies based on water level for continuous monitoring
- Digital Output: HIGH/LOW (for threshold detection)
- Response Time: < 500 ms
- Operating Temperature: 0°C to 85°C
- Cable Length: Typically, 1 meter (for probe-type sensors)
- Material: Corrosion-resistant material (plastic or stainless steel) for durability

Gas Sensor (MQ-2)

A gas sensor is an electronic device used to detect the presence of gases in an environment and measure their concentration. It plays a crucial role in safety systems, environmental monitoring, and industrial applications by identifying potentially hazardous gases such as LPG, methane, carbon monoxide, hydrogen, smoke, and alcohol vapors.

In Arduino-based projects, gas sensors like the MQ-2 help detect gas leaks and air quality issues, triggering alarms or activating ventilation systems when gas levels exceed safe limits.

In the Microcontroller-Driven Indoor Hydroponic Fodder System, the MQ-2 gas sensor monitors the air quality inside the enclosed growing chamber. If harmful gases or excessive CO_2 levels are detected. Additionally, the ESP8266 microcontroller sends alerts to the farmer via the Mobile portal or wMobile app, ensuring a safe and controlled environment for optimal plant growth.

Technical Specifications:

- Operating Voltage: 5V DC
- Gas Detection Range:
- LPG: 200 – 5000 ppm
- Carbon Dioxide - 300 - 400 ppm
- Methane: 500 – 10000 ppm
- Carbon Monoxide (CO): 20 – 2000 ppm
- Analog Output: Provides variable voltage based on gas concentration
- Digital Output: HIGH/LOW signal for threshold detection
- Preheat Time: ~20 seconds
- Response Time: < 10 seconds

5.2 Development Environment.

Connections of Sensors.

5.2.1. Temperature and Humidity Sensor.

Pin Configuration:

- **VCC**: Power supply (3.3V to 5.5V)
- **GND**: Ground
- **Data**: Data pin (connects to Arduino I/O pin, typically with a pull-up resistor of 4.7kΩ to 10kΩ)

```
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Read temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT11 sensor!");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  delay(2000);
}
```

Figure 54-Temperature and Humidity Sensor code

5.2.2. Soil Moisture Sensor.

Pin Configuration:

- VCC from the sensor connects to the 5V pin on the Arduino.
- GND from the sensor connects to GND on the Arduino.
- AO from the sensor connects to A0 on the Arduino (for analog readings).

```
const int sensorPin = A0;
int sensorValue = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(sensorPin);

  Serial.print("Moisture Level: ");
  Serial.println(sensorValue);

  delay(1000);
}
```

Figure 55-Soil Moisture Sensor code

5.2.3. LDR Sensor.

Pin Configuration:

- Connect one leg of the LDR to the 5V pin on the Arduino.
- Connect the other leg of the LDR to A0 (analog pin).
- Connect one leg of the 10kΩ resistor to the same LDR leg connected to A0.
- Connect the other leg of the resistor to GND.

```
const int ldrPin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int ldrValue = analogRead(ldrPin);


  Serial.print("Light Intensity: ");
  Serial.println(ldrValue);

  delay(500);
}
```

Figure 56-LDR Sensor code

5.2.4. TDS Sensor.

Pin Configuration:

- Connect VCC from the TDS sensor module to the 5V pin on the Arduino.
- Connect GND from the module to a GND pin on the Arduino.
- Connect AO from the module to the A0 pin on the Arduino Uno.
- Submerge the TDS probe into the water to be tested.

```
const int tdsPin = A0;
float tdsValue = 0;
float voltage = 0;
const float vRef = 5.0;
const float ecConversionFactor = 500.0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int analogValue = analogRead(tdsPin);
  voltage = analogValue * (vRef / 1023.0);


  tdsValue = (voltage / vRef) * ecConversionFactor;

  Serial.print("TDS Value: ");
  Serial.print(tdsValue);
  Serial.println(" ppm");

  delay(1000);
}
```

Figure 57-TDS Sensor code

5.2.5. Turbidity Sensor.

Pin Configuration:

- VCC from the turbidity sensor connects to the 5V pin on the Arduino.
- GND from the sensor connects to the GND pin on the Arduino.
- AO from the sensor connects to the A0 pin on the Arduino Uno.

```
const int turbidityPin = A0;
float voltage = 0;
float turbidity = 0;
const float vRef = 5.0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(turbidityPin);
  voltage = sensorValue * (vRef / 1023.0);


  turbidity = voltage * 200;

  Serial.print("Voltage: ");
  Serial.print(voltage);
  Serial.print(" V, Turbidity: ");
  Serial.print(turbidity);
  Serial.println(" NTU");

  delay(1000);
}
```

Figure 58-Turbidity Sensor code

5.2.6. Gas Sensor (MQ-2)

Pin Configuration:

- VCC = Connect to the 3.3V or 5V pin on the ESP32.
- GND = Connect to the GND pin on the ESP32.
- AO (Analog Output) = Connect to the A0 (ADC) pin on the ESP32 to measure gas concentration.
- DO (Digital Output) = Connect to a digital GPIO pin on the ESP32 for threshold-based gas detection.

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// WiFi Credentials
#define WIFI_SSID "Your_WiFi_Name"
#define WIFI_PASSWORD "Your_WiFi_Password"

// Firebase Credentials
#define FIREBASE_HOST "https://your-firebase-database.firebaseio.com/"
#define FIREBASE_AUTH "your-firebase-auth-key"

// Initialize Firebase
FirebaseData firebaseData;

// Gas Sensor Pin
#define GAS_SENSOR_PIN A0  // Analog pin connected to MQ-2 or MQ-135

// Threshold for gas detection
int gasThreshold = 400;  // Adjust this based on calibration

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

    // Connect to Wi-Fi
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to WiFi!");

    // Initialize Firebase
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
    Serial.println("Connected to Firebase!");
}

void loop() {
    // Read gas sensor value
    int gasValue = analogRead(GAS_SENSOR_PIN);
    Serial.print("Gas Sensor Value: ");
    Serial.println(gasValue);

    // Upload gas sensor data to Firebase
    Firebase.setInt(firebaseData, "/Hydroponics/GasLevel", gasValue);

    // Check if gas level exceeds the threshold
    if (gasValue > gasThreshold) {
        Serial.println("WARNING: High Gas Levels Detected!");
        Firebase.setString(firebaseData, "/Hydroponics/Alert", "High Gas Detected! Take Action!");
    } else {
        Firebase.setString(firebaseData, "/Hydroponics/Alert", "Gas Levels Normal");
    }

    delay(5000); // Delay before next reading
}
```

Figure 59-Gas Sensor (MQ-2)

5.2.7. Water level sensor

Pin Configuration:

- VCC = Connect to the 3.3V or 5V pin on the ESP32.
- GND = Connect to the GND pin on the ESP32.
- AO (Analog Output) =Connect to the A0 (ADC) pin on the ESP32 to measure water level changes.
- DO (Digital Output) = Connect to a digital GPIO pin on the ESP32 for threshold-based water level detection.

```
#define WATER_LEVEL_PIN D1 // GPIO pin for water level sensor
#define RELAY_PIN D2       // GPIO pin for relay module

void setup() {
  pinMode(WATER_LEVEL_PIN, INPUT); // Water level sensor as input
  pinMode(RELAY_PIN, OUTPUT);      // Relay as output
  digitalWrite(RELAY_PIN, LOW);    // Ensure relay is off initially
  Serial.begin(9600);
}

void loop() {
  int waterLevel = digitalRead(WATER_LEVEL_PIN); // Read water level sensor
  Serial.print("Water Level: ");
  Serial.println(waterLevel);

  if (waterLevel == HIGH) {
    // If water level is HIGH (e.g., tank is full), turn off the pump
    digitalWrite(RELAY_PIN, LOW);
    Serial.println("Pump OFF");
  } else {
    // If water level is LOW (e.g., tank is empty), turn on the pump
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println("Pump ON");
  }

  delay(1000); // Delay for readability
}
```

Figure 60-Water level sensor

5.2.8. DHT11 Sensor Connect to NodeMCU Board (ESP8266)

The DHT11 sensor is a temperature and humidity sensor used in the Microcontroller-Driven Indoor Hydroponic Fodder System to monitor environmental conditions and ensure optimal plant growth. It provides real-time temperature (°C) and humidity (%) data, which helps automate ventilation and water regulation.

Pin Configuration:

- VCC (Power)=Connect to 3.3V or 5V on NodeMCU
- GND (Ground)=Connect to GND on NodeMCU
- DATA (Signal)=Connect to D4 (GPIO2) on NodeMCU

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(115200);
  dht.begin();
  Serial.println("DHT11 Sensor Initialized");
}
void loop() {
  // Reading temperature and humidity
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  // Check if readings are valid
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Print values to the Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%  Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
  delay(2000);
```

Figure 61-DHT11 Sensor Connect to NodeMCU Board (ESP8266)

5.2.9. LCD Display (16*2) connect to NodeMCU Board (ESP8266)

The 16x2 LCD display is used in the Microcontroller-Driven Indoor Hydroponic Fodder System to show real-time sensor readings, such as temperature, humidity, water level, and TDS values. Since the ESP8266 has limited GPIO pins, the LCD is connected using an I2C module to reduce the number of connections.

Pin Configuration:

- I2C module VCC: Connect to the 3.3V pin of the NodeMCU or Separate 5v Power Supply
- I2C module GND: Connect to the GND pin of the NodeMCU or Separate Power Supply GND
- I2C module SDA: Connect to the D2 pin of the NodeMCU
- I2C module SCL: Connect to the D1 pin of the NodeMCU

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_PCF8574 lcd(0x27);


void setup() {
  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Hello, sensor reading!");
}

void loop() {
  lcd.setCursor(0, 1);
  lcd.print("ESP8266 & LCD");
  delay(1000);
}

```

*Figure 62-LCD Display (16*2) connect to NodeMCU Board (ESP8266)*

5.3 Tools Used

5.3.1 Arduino. IDE

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. The Arduino project provides the Arduino integrated development environment (IDE), which is a crossplatform application and can be written in any language. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board or Any other Compatible board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. A program written with the IDE for Arduino is called a sketch.

5.3.2. Firebase Integration in the IoT

In the Microcontroller-Driven Indoor Hydroponic Fodder System, Firebase plays a crucial role as a real-time cloud database that enables data storage, retrieval, and remote monitoring. The system uses IoT components such as temperature sensors, humidity sensors, water level sensors, and TDS sensors to continuously read environmental values and store them in Firebase. By using Firebase, the system allows farmers to access real-time data through a React Native Mobile application, enabling them to monitor and control their hydroponic setup from anywhere.

How Firebase Works in the System =The ESP8266/NodeMCU microcontroller is responsible for reading sensor values and sending them to Firebase using Wi-Fi connectivity. Firebase provides a NoSQL real-time database, meaning that as soon as a sensor detects a change in values, the data is immediately updated in the cloud without delays. This ensures that farmers always have up-to-date information about their hydroponic system. The Firebase Realtime Database follows a tree-structured format where data is stored in JSON format. Each sensor has its own unique path in the database, making it easy to organize and retrieve data efficiently.

5.3.3. VS Code and React Native Development

Visual Studio Code (VS Code) is a lightweight yet powerful code editor that is widely used for JavaScript and React Native development. It provides syntax highlighting, debugging tools, extensions, and integrated terminal support, making it an ideal choice for building Mobile applications. Since you are developing a React Native Mobile application with Expo, VS Code will be the primary environment for writing and managing your code efficiently.

5.4 Deployment Architecture

The deployment architecture of the Microcontroller-Driven Indoor Hydroponic Fodder System defines how different components—hardware, software, cloud services, and Mobile application—interact to enable real-time monitoring, data storage, and automation. The system consists of IoT sensors, ESP8266/NodeMCU microcontroller, Firebase cloud database, and a React Native Mobile application.

1. System Components in Deployment Architecture

- IoT Hardware Layer (On-Site Hydroponic System)
- Sensors: DHT11 (Temperature & Humidity), Water Level Sensor, TDS Sensor, MQ-2 Gas Sensor
- Actuators: Water Pump, Exhaust Fan, LED Grow Lights
- Microcontroller: ESP8266/NodeMCU (Collects sensor data & sends to Firebase)
- Power Source: 5V/12V DC supply
- Communication Layer (IoT to Cloud)
- ESP8266/NodeMCU connects to WiFi and transmits data using HTTP requests or MQTT protocol.

- Data is sent to Firebase in real-time for storage and retrieval.
- Cloud Layer (Firebase Realtime Database & Authentication)
- Firebase stores sensor data in a structured JSON format.
- Provides user authentication to secure access to hydroponic data.
- Enables real-time data synchronization for Mobile and Mobile applications.
- Application Layer (User Interface & Remote Monitoring)
- React Native Mobile App (Expo framework) for real-time monitoring & remote control.
- Mobile Portal (Optional - React.js or Next.js) for system access via browser.
- Users can view data, control actuators, and receive notifications on abnormal conditions.

2. Workflow of the Deployment Architecture

- Sensor Data Collection
- IoT sensors continuously measure temperature, humidity, water level, CO_2 levels, and nutrient concentration.
- Data is processed by ESP8266/NodeMCU.
- Data Transmission to Firebase
- The microcontroller sends data to Firebase Realtime Database via WiFi.
- Data is updated instantly in the cloud.
- Data Processing & Automation
- Firebase stores and manages real-time data.
- Automated triggers can activate actuators (e.g., water pump turns on when the water level is low).

- User Access via Mobile Application
- Farmers can log in to the React Native app to view real-time sensor data.
- Users can remotely control components such as water pumps, fans, and lights from their smartphones.
- Firebase sends alerts if critical thresholds are exceeded (e.g., low water level, high temperature).

3. Benefits of This Deployment Architecture

- Real-Time Monitoring: Farmers can track environmental conditions anytime, anywhere.
- Cloud-Based Automation: The system runs independently with minimal manual intervention.
- Scalability: Can support multiple hydroponic units across different locations.
- Security & Authentication: Firebase ensures secure user access to hydroponic data.
- Remote Control Capabilities: Users can activate pumps, fans, or lights remotely through the Mobile app.

5.4.1 Graphical User Interface

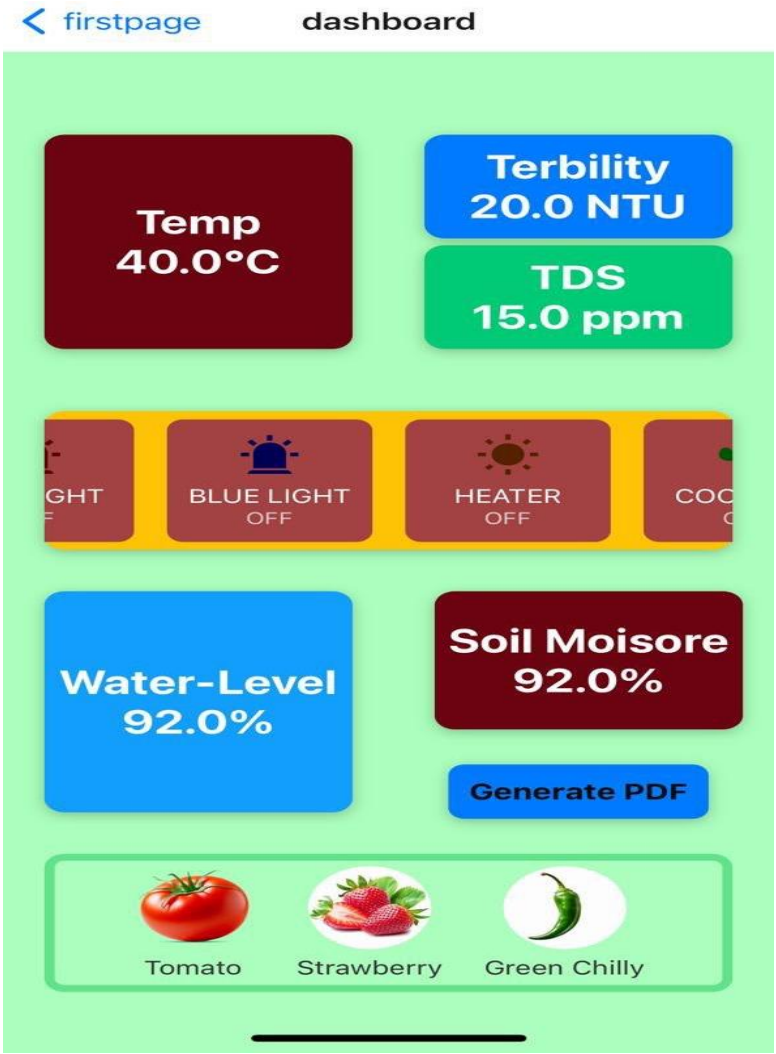

Figure 63-Graphical User Interface

5.4.1.1 System Setup

The configuration of Microcontroller-Driven Indoor Hydroponic Fodder involves integration of an assortment of software and hardware modules for real-time observation, automation, and distant access. Microcontroller ESP8266 is utilized as a processing unit, communicating an assortment of sensors, actuating tools, and cloud platforms for creating perfect growing environments for the plants. The configuration involves a full

automation for a hydroponic farm with real-time observation, smart decision-making, and distant access through a smartphone app for enhanced efficiency and productivity.

Figure 65-System Setup01

Figure 64-System Setup02

5.5 Major Code Segments

DHT11 fan with 12v bulb final Sensor code

```
#include "DHT.h"
#define DHTPIN A1 // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // DHT 11
#define FAN_RELAYPIN 3 // Pin where the fan relay module is connected
#define BULB_RELAYPIN 4 // Pin where the bulb relay module is connected
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(FAN_RELAYPIN, OUTPUT);
pinMode(BULB_RELAYPIN, OUTPUT);
digitalWrite(FAN_RELAYPIN, LOW); // Ensure the fan relay is off initially
digitalWrite(BULB_RELAYPIN, LOW); // Ensure the bulb relay is off initially
}
void loop() {
delay(2000); // Wait a few seconds between measurements
float temp = dht.readTemperature(); // Read temperature as Celsius
// Check if any reads failed and exit early (to try again).
if (isnan(temp)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Serial.print("Temperature: ");
Serial.print(temp);
Serial.println(" *C");
if (temp > 35.0) {
digitalWrite(FAN_RELAYPIN, HIGH); // Turn the fan relay on (activates the fan)
} else {
digitalWrite(FAN_RELAYPIN, LOW); // Turn the fan relay off
}
if (temp < 30.0) {
digitalWrite(BULB_RELAYPIN, HIGH); // Turn the bulb relay on (activates the bulb)
} else {
digitalWrite(BULB_RELAYPIN, LOW); // Turn the bulb relay off
}
}
```

Figure 66-DHT11 fan with 12v bulb final Sensor code

DHT 11 with fan

```
#include "DHT.h"

#define DHTPIN A1      // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // DHT 11
#define RELAYPIN 3    // Pin where the relay module is connected

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(RELAYPIN, OUTPUT);
  digitalWrite(RELAYPIN, LOW); // Ensure the relay is off initially
}

void loop() {
  delay(2000); // Wait a few seconds between measurements

  float temp = dht.readTemperature(); // Read temperature as Celsius

  // Check if any reads failed and exit early (to try again).
  if (isnan(temp)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" *C");

  if (temp > 35.0) {
    digitalWrite(RELAYPIN, HIGH); // Turn the fan on
  } else {
    digitalWrite(RELAYPIN, LOW); // Turn the fan off
  }
}
```

Figure 67-DHT 11 with fan

DHT11 with fan and blub code

```
#include "DHT.h"

#define DHTPIN A1        // Pin where the DHT11 is connected
#define DHTTYPE DHT11    // DHT 11
#define FAN_RELAYPIN 3   // Pin where the fan relay module is connected
#define BULB_RELAYPIN 4  // Pin where the bulb relay module is connected

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(FAN_RELAYPIN, OUTPUT);
  pinMode(BULB_RELAYPIN, OUTPUT);
  digitalWrite(FAN_RELAYPIN, LOW);  // Ensure the fan relay is off initially
  digitalWrite(BULB_RELAYPIN, LOW); // Ensure the bulb relay is off initially
}

void loop() {
  delay(2000);  // Wait a few seconds between measurements

  float temp = dht.readTemperature();  // Read temperature as Celsius

  // Check if any reads failed and exit early (to try again).
  if (isnan(temp)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" *C");

  if (temp > 35.0) {
    digitalWrite(FAN_RELAYPIN, HIGH);  // Turn the fan relay on (activates the fan)
  } else {
    digitalWrite(FAN_RELAYPIN, LOW);  // Turn the fan relay off
  }

  if (temp < 30.0) {
    digitalWrite(BULB_RELAYPIN, HIGH);  // Turn the bulb relay on (activates the bulb)
  } else {
    digitalWrite(BULB_RELAYPIN, LOW);  // Turn the bulb relay off
  }
}
```

Figure 68-DHT11 with fan and blub code

water level sensor and water pump motor connect to uno board

```
#define WATER_LEVEL_PIN D1 // GPIO pin for water level sensor
#define RELAY_PIN D2       // GPIO pin for relay module

void setup() {
  pinMode(WATER_LEVEL_PIN, INPUT); // Water level sensor as input
  pinMode(RELAY_PIN, OUTPUT);      // Relay as output
  digitalWrite(RELAY_PIN, LOW);    // Ensure relay is off initially
  Serial.begin(9600);
}

void loop() {
  int waterLevel = digitalRead(WATER_LEVEL_PIN); // Read water level sensor
  Serial.print("Water Level: ");
  Serial.println(waterLevel);

  if (waterLevel == HIGH) {
    // If water level is HIGH (e.g., tank is full), turn off the pump
    digitalWrite(RELAY_PIN, LOW);
    Serial.println("Pump OFF");
  } else {
    // If water level is LOW (e.g., tank is empty), turn on the pump
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println("Pump ON");
  }

  delay(1000); // Delay for readability
}
```

Figure 69-water level sensor and water pump motor connect to uno board

CHAPTER 6 –TESTING AND EVALUATION

6.1 Introduction

Device Testing is a process of executing the Device with an intent to find any Device bugs. It is used to check whether the Device met its expectations and all the functionalities of the Device is working. The final goal of testing is to check whether the Device is behaving in the way it is supposed to under specified conditions. All aspects of the code are examined to check the quality of the Device. The primary purpose of testing is to detect Sensor failures so that defects may be uncovered and corrected. The test cases are designed in such way that scope of finding the bugs is maximum.

6.2 Testing Procedure

There are various testing levels based on the specificity of the test.

- **Unit testing:** Unit testing refers to tests conducted on a section of code in order to verify the functionality of that piece of code. This is done at the function level.

- **Integration Testing:** Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Its primary purpose is to expose the defects associated with the interfacing of modules.

- **System Testing:** System testing tests a completely integrated system to verify that the system meets its requirements.

- **Acceptance testing:** Acceptance testing tests the readiness of application, satisfying all requirements.

- **Performance testing:** Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device such as response time or millions of instructions per second etc

6.3 Test Plan and Test Cases

A test case is a set of test data, preconditions, expected results, and post-conditions developed for a test scenario to verify compliance against a specific requirement. I have designed and executed a few test cases to check if the project meets the functional requirements.

Test Cases for Water Level sensor.

Table 2-Test Cases for Water Level sensor

Test No	**Test Case Description**	**Steps Carried Out**	**Expected Result**
01	Water Level Sensor	Detecting low water levels in the tank	Detect low water level and send data to the controller
02	Water Pump Activation	If low water level is detected, turn on the pump	Pump is turned on, and water starts flowing
03	Pump Deactivation	If water level reaches the required level, turn off the pump	Pump stops when the required water level is reached
04	Water level Alert -	If low Water level is detected	Display the warning message and Turn on the pump motor

Test Cases for Temperature Sensor

Table 3-Test Cases for Temperature Sensor

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Temperature Measurement	Read temperature data from the DHT11 sensor	The correct temperature value is detected and displayed
02	Humidity Measurement	Read humidity data from the DHT11 sensor	Correct humidity value is detected and displayed
03	High Temperature	Detecting High Temperature	Detect the high temperature, turn on the fan, and turn on the humidity fire. Then Display on the Mobile app
04	Low Temperature	Detecting Low Temperature	Detect the low temperature and turn on the pilement blub. Then Display on the Mobile app

Test Cases for light-detect Sensor

Table 4-Test Cases for light-detect Sensor

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Light Intensity Measurement	The correct light intensity value is detected	The correct light intensity value is detected
02	Automatic Light Control	If it is dark, turn on a connected light source	If it is dark, turn on a connected light source

03	LED light Turn on	If it is dark, turn on a connected LED light	Light turn on automatically
04	LED light Turn off	If it is light, Turn off connected LED light	Light turn off automatically .

Test Cases for TDS Sensors

Table 5-Test Cases for TDS Sensors

Test No	Test Case Description	Steps Carried Out	Expected Result
01	TDS Value Measurement	Read TDS value from the sensor	Display correct TDS value

Test Cases for Turbidity Sensors

Table 6-Test Cases for Turbidity Sensors

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Turbidity Value Measurement	Read turbidity value from the sensor	Display correct turbidity value

Test Cases for Soil Moisture Sensor

Table 7-Test Cases for Soil Moisture Sensor

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Soil Moisture Measurement	Read soil moisture level	Display correct moisture value

Test Cases for CO_2 Gas Sensor

Table 8-Test Cases for CO_2 Gas Sensor

Test No	Test Case Description	Steps Carried Out	Expected Result
01	CO_2 Gas Detection	Read CO_2 concentration from gas sensor	Display correct CO_2 value

Test Cases for LCD Display

Table 9-Test Cases for LCD Display

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Data Show	Simulate CO2 Gas Sensors, TDS Sensors, Turbidity Sensors, Temperature, Soil Moisture Sensors, and System Statuses	The LCD Displays Accurate Readings and Status Messages

Test Cases for Sensor Data on Mobile App

Table 10-Test Cases for Sensor Data on Mobile App

Test No	Test Case Description	Steps Carried Out	Expected Result
01	Show Data on the Mobile app	Display the Data on the Mobile app in Quick Time	Run Data and Relay Information on the Mobile app on Time

6.4 Test Data and Test Results

Test Cases for Water Level sensor.

Table 11-Test Cases for Water Level sensor Status

Test No	Test Case Description	Steps Carried Out	Status
01	Water Level Sensor	Detecting low water levels in the tank	Pass
02	Water Pump Activation	If a low water level is detected, turn on the pump	Pass
03	Pump Deactivation	If the water level reaches the required level, turn off the pump	Pass
04	Water level Alert -	If a low Water level is detected	Pass

Test Cases for Temperature Sensor

Table 12-Test Cases for Temperature Sensor Status

Test No	Test Case Description	Steps Carried Out	Status
01	Temperature Measurement	Read temperature data from the DHT11 sensor	Pass
02	Humidity Measurement	Read humidity data from the DHT11 sensor	Pass
03	High Temperature	Detecting High Temperature	Pass
04	Low Temperature	Detecting Low Temperature	Pass

Test Cases for light-detect Sensor

Table 13-Test Cases for light-detect Sensor Status

Test No	Test Case Description	Steps Carried Out	Status
01	Light Intensity Measurement	The correct light intensity value is detected	Pass
02	Automatic Light Control	If it is dark, turn on a connected light source	Pass
03	LED light Turn on	If it is dark, turn on a connected LED light	Pass
04	LED light Turn off	If it is light, Turn off connected LED light	Pass

Test Cases for TDS Sensors

Table 14-Test Cases for TDS Sensors Status

Test No	Test Case Description	Steps Carried Out	Status
01	TDS Value Measurement	Read TDS value from the sensor	Pass

Test Cases for Turbidity Sensors

Table 15-Test Cases for Turbidity Sensors Status

Test No	Test Case Description	Steps Carried Out	Status
01	Turbidity Value Measurement	Read turbidity value from the sensor	Pass

Test Cases for Soil Moisture Sensor

Table 16-Test Cases for Soil Moisture Sensor Status

Test No	Test Case Description	Steps Carried Out	Status
01	Soil Moisture Measurement	Read soil moisture level	Pass

Test Cases for CO_2 Gas Sensor

Table 17-Test Cases for CO_2 Gas Sensor Status

Test No	Test Case Description	Steps Carried Out	Status
01	CO_2 Gas Detection	Read CO_2 concentration from gas sensor	Pass

Test Cases for LCD Display

Table 18-Test Cases for LCD Display Status

Test No	Test Case Description	Steps Carried Out	Status
01	Data Show	Simulate CO2 Gas Sensors, TDS Sensors, Turbidity Sensors, Temperature, Soil Moisture Sensors, and System Statuses	Pass

Test Cases for Sensor Data on Mobile App

Table 19-Test Cases for Sensor Data on Mobile App Status

Test No	Test Case Description	Steps Carried Out	Status
01	Show Data on the Mobile app	Display the Data on the Mobile app in Quick Time	Pass

CHAPTER 7 - CONCLUSION AND FUTURE WORK

7.1 Conclusion

Rapid global population growth is putting pressure on agricultural resources. especially the land which is becoming increasingly lacking When the urban area expands Cultivated area has decreased. This makes it difficult to produce food using traditional agricultural methods. To address this challenge IoT-based hydroponic systems have emerged as a solution for efficient food production. Especially in environments with limited space.

Automation of an IoT-based hydroponic system using ESP8266.

Introduction to Hydroponics: Hydroponics is a method of growing plants without using soil. It uses an aqueous nutrient solution to provide the essential nutrients that plants need. Plant roots are suspended in a nutrient-rich solution, such as clay pebbles, coconut charcoal, or an inert medium. This allows the plant to better control its environment. This may result in faster growth and higher yields compared to traditional land-based agriculture...

The role of IoT in hydrographic: Integrating the Internet of Things (IoT) into hydrographic systems revolutionizes the way these systems are monitored and controlled. IoT enables remote access and real-time monitoring of environmental parameters. Makes it easy to manage your hydroponic setup. Especially for users who may have limited time or are away from home, ANAM and its use can automate key functions such as water flow, lighting and temperature control. This ensures that plants receive optimum conditions at all times...

ESP8266 Microcontroller: The ESP8266 is a compact yet powerful Wi-Fi-enabled microcontroller, making it an ideal choice for IoT-based hydroponic farming applications. It serves as the central processing unit (CPU) of the Hydroponic Fodder System, enabling real-time data collection, cloud communication, and remote control.

In this project, the ESP8266 is responsible for reading sensor data from various environmental monitoring components such as temperature, humidity, water level, TDS, and gas sensors. It processes this data and transmits it to a cloud-based database (Firebase), where users can remotely access system updates via a React Native Mobile application

Due to its low power consumption, wireless connectivity, and efficient processing capabilities, the ESP8266 makes hydroponic farming more intelligent, automated, and scalable, allowing farmers to monitor and manage their indoor fodder system anytime, anywhere.

Benefits of the IoT-Based Hydroponic System

Space-Efficient Farming:

The IoT-enabled hydroponic system is designed to maximize crop yield in minimal space, making it an excellent option for urban dwellers with limited room for gardening. It allows users to grow fresh produce in apartments, balconies, or small indoor spaces.

High Yield and Faster Growth:

By providing precise control over the growing conditions, the system can significantly increase the growth rate and yield of crops compared to traditional soil farming. Plants receive a balanced supply of nutrients, water, and light, leading to healthier and more productive growth.

Convenience for Busy Lifestyles:

One of the main advantages of this system is that it caters to people who may not have the time to manage a traditional garden. With automation and remote monitoring capabilities, users can easily manage their hydroponic system without constant manual intervention.

Resource Efficiency:

The hydroponic system uses water and nutrients more efficiently than traditional farming, as the closed-loop design recirculates water, reducing wastage. This is particularly important in areas with limited water resources, making it a sustainable option for growing food.

Scalability:

The system can be scaled up by adding more sensors, increasing the number of plants, or expanding the size of the hydroponic setup. The modular nature of the ESP8266 and IoT framework makes it easy to adapt and expand as needed.\

7.2 Future Plan

The proposed system focuses on monitoring and controlling the conditions within a hydroponic system, with plans for future enhancements in both hardware and software. The hardware improvements aim to incorporate sensors for measuring key parameters like pH, electrical conductivity (EC) . A pH meter will determine the acidity or alkalinity of the nutrient solution, an EC meter will assess nutrient concentration , all of which are critical for optimal plant growth. On the software side, the plan includes developing a Mobile site and a Mobile application. The Mobile app will provide detailed, real-time data and reports on greenhouse conditions, making it easier for users to monitor the system remotely. Additionally, the Mobile app is designed to enable agricultural officers to observe the hydroponic system and provide expert guidance. Leveraging the Internet of Things (IoT), the system aims to ensure seamless data collection, transmission, and analysis, creating a comprehensive and user-friendly solution for efficient hydroponic management.

To make the system more intelligent, Artificial Intelligence (AI) and Machine Learning (ML) will be integrated. These technologies will analyze historical sensor data to predict future environmental conditions and automate responses accordingly. This predictive approach will reduce energy usage and improve efficiency. Automated nutrient dosing will also be introduced, where AI algorithms determine the exact amount of water and

nutrients required based on plant growth stages and environmental conditions. This feature will significantly minimize waste and improve crop yields.

In conclusion, the future development of this IoT-driven hydroponic system will focus on improving automation, optimizing resource usage, and enhancing remote access. With advanced sensors, AI-driven automation, cloud connectivity, and sustainable energy solutions, this system has the potential to revolutionize indoor farming, making it more productive, efficient, and accessible to farmers worldwide.

7.3 References

[1] A. Abu Sneineh and A. A. Shabaneh, "Design of a smart hydroponics monitoring system using an ESP32 microcontroller and the Internet of Things," MethodsX, vol. 11, 2023.

[2] P. Bagavan Reddy and M. Harani, "Hydroponics: A sustainable way of green fodder production," Indian Farming, vol. 73, no. 2, pp. 2-5, Feb. 2023.

[3] S. K. Singh et al., "Development and performance evaluation of evaporative cool hydroponic fodder production chamber," Range Management and Agroforestry, vol. 43, no. 1, pp. 132-138, Aug. 2022.

[4] A. Adelowokan et al., "Evaluating the performance of small-scale indoor vertical hydroponics systems for lettuce production," Poster, June 2023.

[5] R. Ghorbel and N. Koşum, "Hydroponic Fodder Production: An Alternative Solution for Feed Scarcity," Proceedings of the 6th International Students Science Congress, Sept. 2022.

[6] M. G. Thalkar, "Hydroponics technology for fodder production," Agriculture and Food Magazine, vol. 1, no. 11, pp. 84-89, Nov. 2019.

[7] Y. Mardiansyah et al., "Application of smart indoor hydroponic technology to support food security," Abdimas: Jurnal Pengabdian Masyarakat Universitas Merdeka Malang, vol. 8, no. 4, pp. 572-582, Nov. 2023.

[8] M. Barwant and K. Barwant, "Comparative Study of Artificial Fodder Production (Hydroponic) and its Benefits," International Journal of Innovative Science and Research Technology, vol. 5, no. 4, pp. 106-111, Apr. 2020.

[9] A. G. Bhat et al., "Development of Small Scale Indoor Hydroponic Fodder Production System," International Journal of Environment and Climate Change, vol. 11, no. 7, pp. 47-51, Sep. 2021.

[10] A. Susilawati et al., "Development of Automatic Hydroponic Plant Watering Based Arduino Microcontroller," Jurnal Proksima, vol. 1, no. 2, pp. 60-65, Oct. 2023.

[11] R. Newell, L. Newman, M. Dickson, B. Vanderkooi, T. Fernback, and C. White, "Hydroponic fodder and greenhouse gas emissions: a potential avenue for climate mitigation strategy and policy development," FACETS, vol. 6, pp. 334-357, 2021.

[12] G. S. Malhi, M. Kaur, K. Sharma, and G. Gupta, "Hydroponics Technology for Green Fodder Production under Resource Deficit Condition," Vigyan Varta, vol. 1, no. 5, pp. 65-68, October 2020.

[13]. Bedeke, T. Melkato, M. Dejene, and T. Fentaw, "Impact of Adoption of Hydroponic Fodder Production on Pastoralist Households' Income in Borena, Ethiopia," Agricultural Economics, vol. 16, no. 1, pp. 363-374, January 2024. DOI: 10.21203/rs.3.rs-4268088/v1.

[14]. L. Hasith Deshan, K. T. Y. Mahima, M. P. C. K. Divyanjalee, Thilini A. Perera, Damitha Sandaruwan, L. N. C. De Silva, R. Pushpannda, C. I. Keppitiyagama, and T. N. K. De Zoysa, "Hyposense: An Integrated Sensor Device for Hydroponics Farm Monitoring," KDU Journal of Multidisciplinary Studies, vol. 6, no. 1, pp. 43-54, July 2024. DOI: 10.4038/kjms.v6i1.108.

[15] B. Mansingh, G. Nanda, S. Kumar, R. Banik, and P. Kalita, "Fodder Hydroponics: A Creative Proceed towards Sustainable Green Fodder Production," Fodder Hydroponics, vol. 2, no. 8, pp. 1-20, July 2023.

[16] H. M. Manju, M. Singh, K. K. Yadav, and S. R. Bhakar, "Development of Solar Operated Hydroponic Fodder Production System," Int. J. Curr. Microbiol. App. Sci., vol. 9, no. 11, pp. 2936-2942, 2020.

[17] S. S. Turakne, S. B. Jondhale, P. M. Vikhe, and M. N. Gore, "Hydroponics Fodder Grow Chamber," Int. J. Sci. Res. Sci. Eng. Technol., vol. 8, no. 3, pp. 383-387, May-June 2021.

[18]. Yang, J. Sun, X. Wang, and B. Zhang, "Light Intensity Affects Growth and Nutrient Value of Hydroponic Barley Fodder," Agronomy, vol. 14, no. 6, pp. 1099, May 2024. DOI: 10.3390/agronomy14061099.

[19]. Suresh, T. Logasundari, V. Shanmukha Sravani, M. Ali, and S. Srinivasan, "IoT Based Automated Indoor Hydroponic Farming System," E3S Web of Conferences, vol. 45, no. 702002, 2024. DOI: 10.1051/e3sconf/202454702002.

APPENDICES OF THE SYSTEM

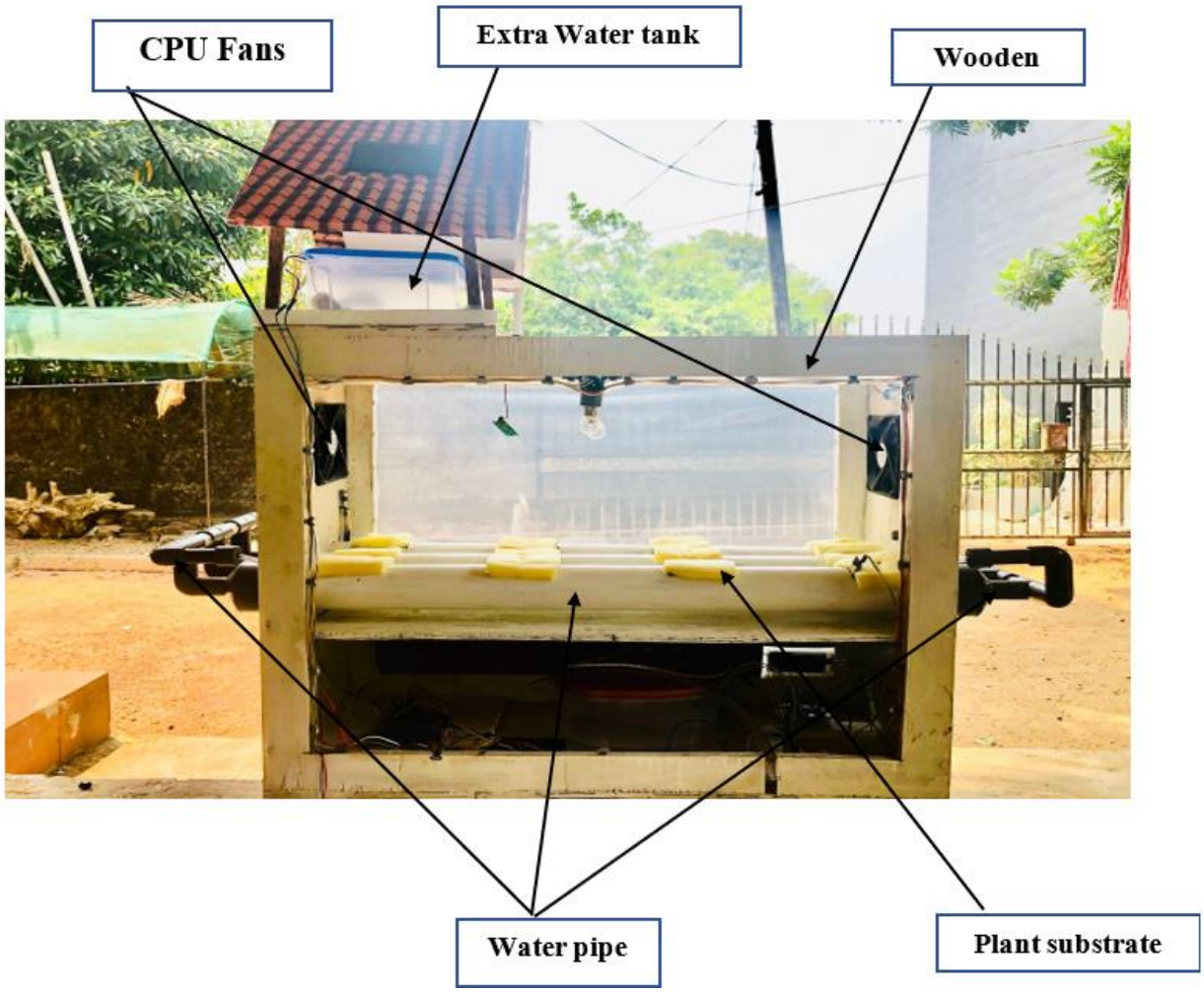

Figure 70-System Photo 1

Wooden Structure: The frame holding the entire system in place, providing stability and support for the other components.

Water Pipe: Channels the water from the tank through the system, distributing it evenly to the plant substrates.

Plant Substrate: Mediums where the plants are placed for growth, holding the roots while absorbing the nutrient solution.

Extra Water tank :The Extra Water Tank stores backup water to ensure a continuous supply for the hydroponic system, preventing water shortages.

CPU Fans: These are used for air circulation, ensuring that the air inside the structure remains well-ventilated and that the temperature and humidity are evenly distributed.

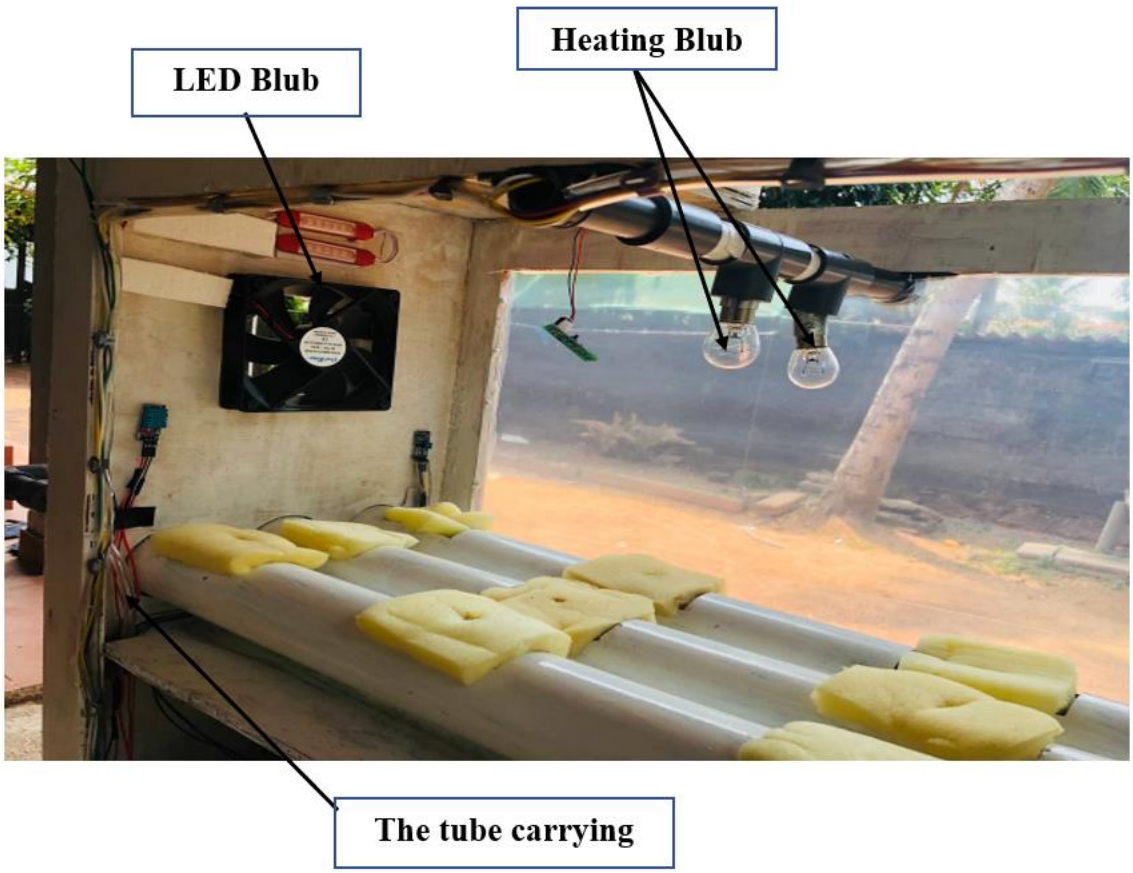

Figure 71-System Photo 2

LED Bulb: These are small light sources, likely used to provide illumination within the structure. Given the presence of multiple LED bulbs and the enclosed space, the bulbs may be used for specific purposes, such as:

- Providing artificial lighting for plant growth (if this is a hydroponic or agricultural setup).
- Simulating environmental conditions in an experiment or controlled

environment. Heating bulb: Provides warmth to the system, helping to maintain a suitable temperature for plant growth, especially in colder environments.

Heating bulb: Provides warmth to the system, helping to maintain a suitable temperature for plant growth, especially in colder environments.

Tube Carrying the Water: This tube transports nutrient-rich water through the system, ensuring a consistent flow to the plant substrates and facilitating the hydroponic growing process.

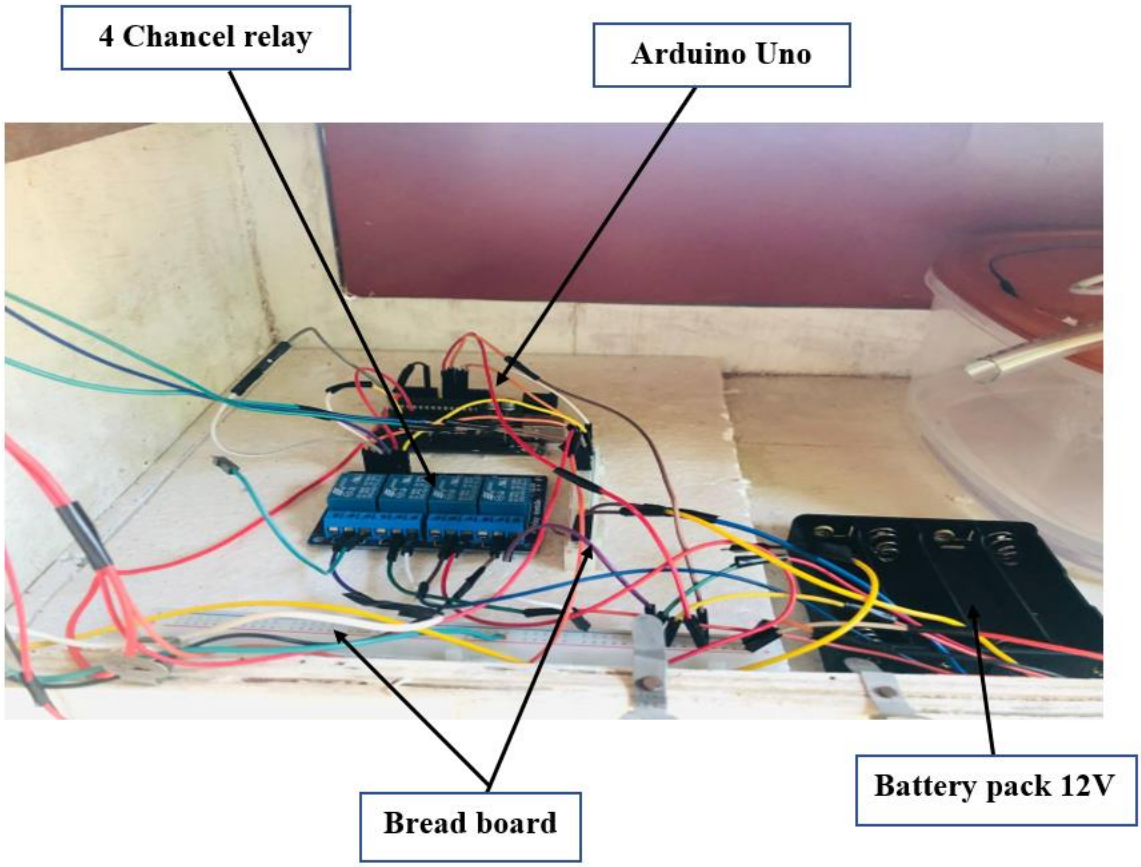

Figure 72-System Photo 3

4-Channel Relay Module: This module acts as a switch to control high-power devices like pumps, fans, or lights. It allows the microcontroller to control multiple components in the system by turning them on or off.

Arduino Uno Board: A microcontroller used for reading data from various sensors and controlling actuators. It serves as the central control unit for managing the system's functions.

Breadboard: Used for prototyping and connecting electronic components without the need for soldering. It helps in organizing the wiring and connections between the microcontrollers, sensors, and relays.

Battery Pack 12V: A set of rechargeable batteries connected in series or parallel to provide a 12-volt power supply for the device

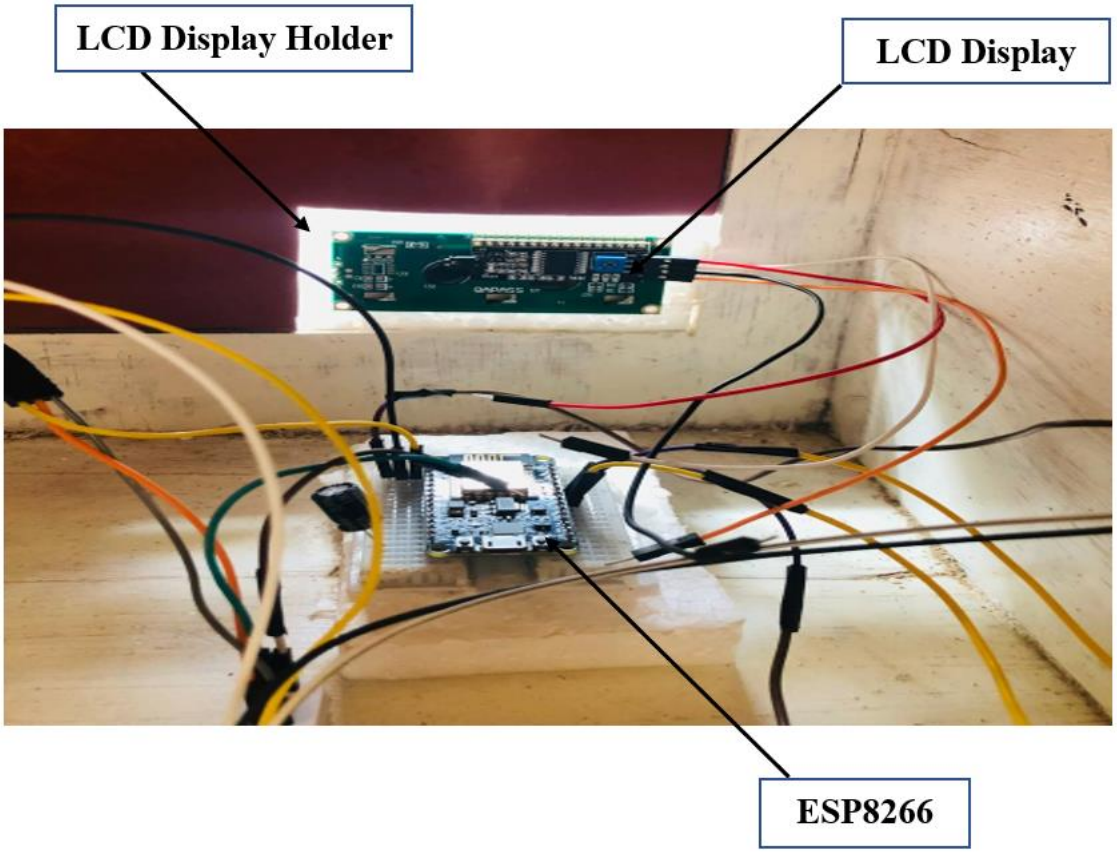

Figure 73-System Photo 4

LCD Display: A screen that shows data or readings. Here, it appears to be displaying a numeric value, possibly indicating a sensor reading (e.g., TDS value, voltage, or another measurement).

LCD Display Holder: A frame or support structure that securely holds the LCD display in place, ensuring visibility and stability during operation.

ESP8266: A compact Wi-Fi-enabled microcontroller, used for wireless communication in the hydroponic system. It enables real-time data transmission to the Firebase cloud, allowing remote monitoring and control via a mobile app.

Figure 74-System Photo 5

solar panel : A renewable energy source that powers the hydroponic system, reducing dependency on electricity. It provides sustainable energy for sensors, water pumps, and microcontrollers, ensuring continuous operation while promoting eco-friendly farming practices.

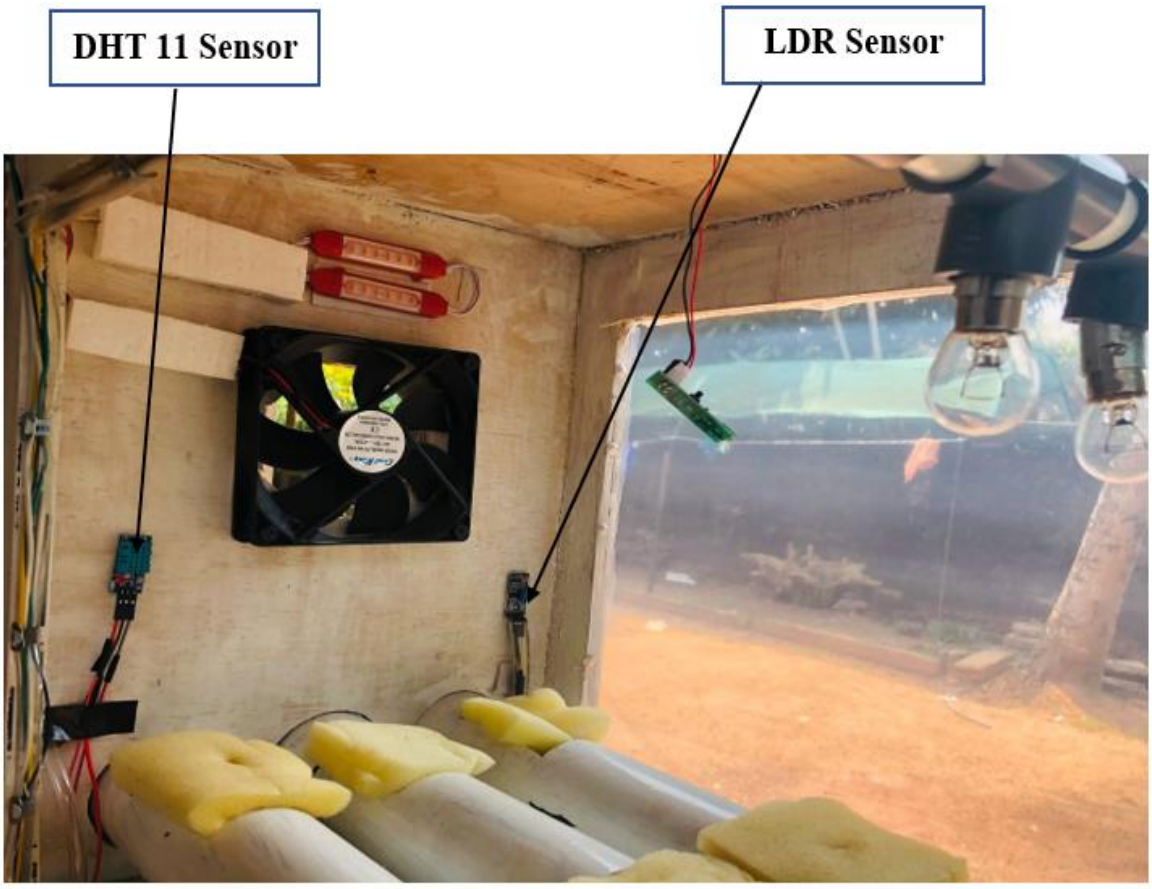

Figure 75-System Photo 6

DHT11 Sensor: The DHT11 sensor is used to measure both temperature and humidity levels. It provides calibrated digital output, making it easy to connect with microcontrollers such as Arduino or Raspberry Pi.In this setup, the DHT11 sensor plays a role in monitoring environmental conditions, ensuring the temperature and humidity are kept within optimal ranges for the intended application.

LDR Sensor (Light Dependent Resistor):

An LDR sensor detects light levels. Its resistance changes based on the amount of light it receives.

In this setup, it may monitor the lighting conditions and adjust the LED lights accordingly to maintain desired illumination.

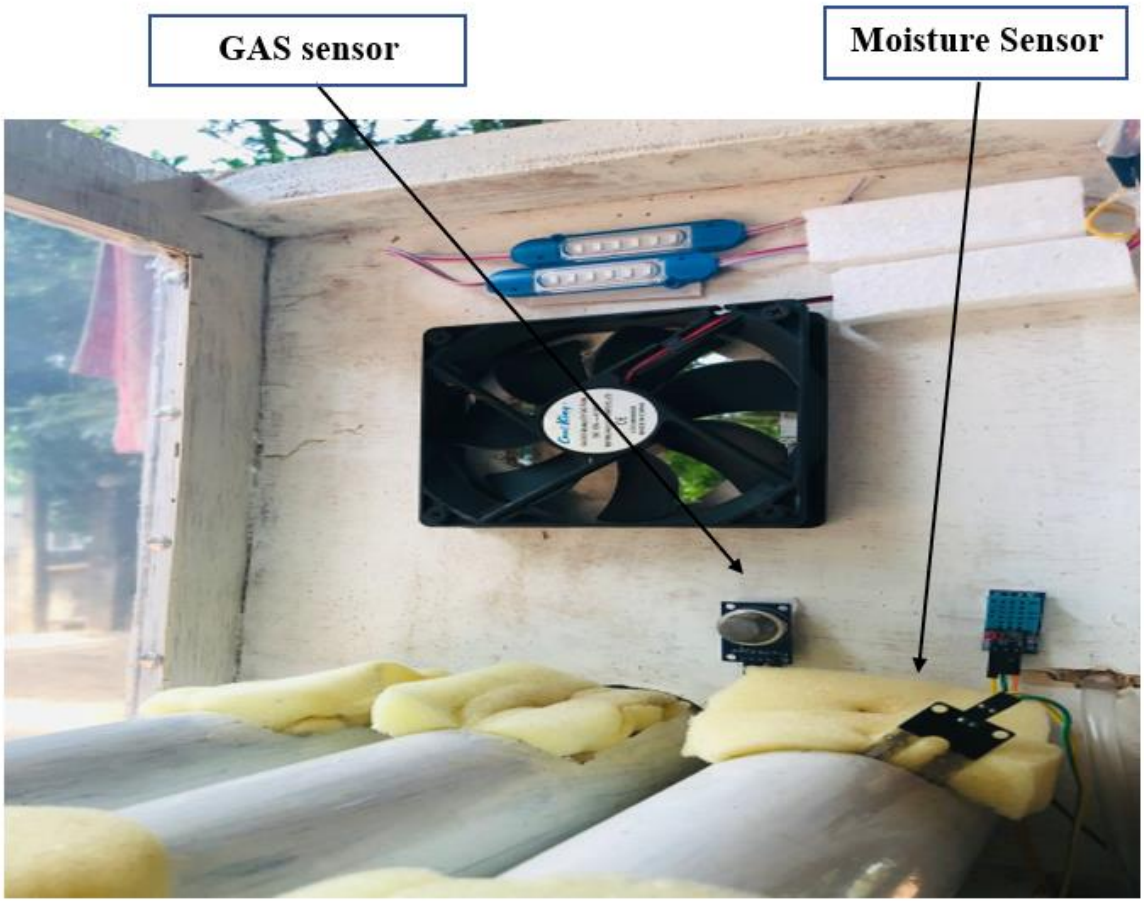

Figure 76-System Photo 7

Moisture Sensor: This sensor is used to measure the moisture content, likely of the surrounding air or soil in the setup. If this is a plant growth or environmental monitoring system, the sensor helps ensure optimal humidity or soil moisture levels by triggering adjustments (e.g., watering or misting).

GAS sensor :A carbon dioxide (CO_2) sensor used to monitor air quality in the hydroponic system. It helps regulate CO_2 levels to optimize plant growth and photosynthesis, ensuring a healthy growing environment while preventing excess CO_2 buildup.

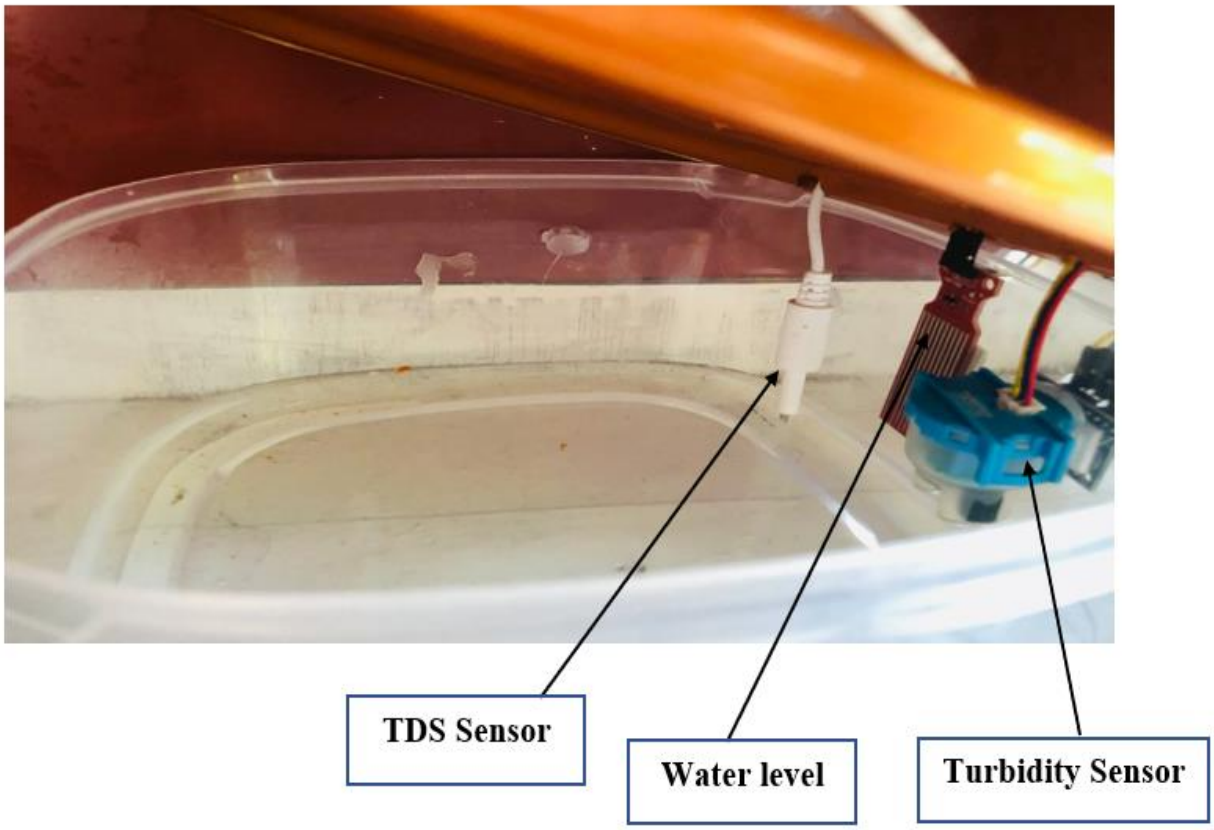

Figure 77-System Photo 8

TDS Sensor: The Total Dissolved Solids (TDS) sensor measures the concentration of dissolved substances in a liquid, typically used to assess water quality. In this setup, it likely monitors the purity or contamination level of the water.

Turbidity Sensor: This sensor detects the cloudiness or haziness of a liquid by measuring the amount of light scattered by suspended particles. It helps determine water clarity, which is essential in water quality analysis.

Water level :A sensor used to monitor water levels in the hydroponic system, ensuring a consistent water supply. It helps prevent water shortages or overflows by triggering the water pump when needed, maintaining optimal plant growth conditions.

Figure 78-System Photo 9

Figure 79-System Photo 10

Figure 80-System Photo 11

Figure 81-System Photo 12

DHT11 fan with 12v bulb final Sensor code

```
#include "DHT.h"
#define DHTPIN A1 // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // DHT 11
#define FAN_RELAYPIN 3 // Pin where the fan relay module is connected
#define BULB_RELAYPIN 4 // Pin where the bulb relay module is connected
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(FAN_RELAYPIN, OUTPUT);
pinMode(BULB_RELAYPIN, OUTPUT);
digitalWrite(FAN_RELAYPIN, LOW); // Ensure the fan relay is off initially
digitalWrite(BULB_RELAYPIN, LOW); // Ensure the bulb relay is off initially
}
void loop() {
delay(2000); // Wait a few seconds between measurements
float temp = dht.readTemperature(); // Read temperature as Celsius
// Check if any reads failed and exit early (to try again).
if (isnan(temp)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Serial.print("Temperature: ");
Serial.print(temp);
Serial.println(" *C");
if (temp > 35.0) {
digitalWrite(FAN_RELAYPIN, HIGH); // Turn the fan relay on (activates the fan)
} else {
digitalWrite(FAN_RELAYPIN, LOW); // Turn the fan relay off
```

```
}
if (temp < 30.0) {
digitalWrite(BULB_RELAYPIN, HIGH); // Turn the bulb relay on (activates the bulb)
} else {
digitalWrite(BULB_RELAYPIN, LOW); // Turn the bulb relay off
}
```

This code uses a DHT11 sensor to monitor the temperature and control two devices: a fan and a bulb, using relays. The DHT11 sensor is connected to pin `A1`, the fan relay is connected to pin `3`, and the bulb relay is connected to pin `4`. In the `setup()` function, serial communication is initialized for displaying temperature readings, and the relays for the fan and bulb are set up as outputs. Both relays are initially turned off to ensure that the fan and bulb start in the off state.

In the `loop()` function, the code reads the temperature every 2 seconds from the DHT11 sensor. If the sensor fails to provide a valid reading, an error message is printed, and the loop restarts. If the reading is successful, the temperature is printed to the Serial Monitor. If the temperature exceeds 35°C, the fan is activated by turning on the fan relay. If the temperature is below 30°C, the bulb is activated by turning on the bulb relay. The system ensures that if the temperature is between 30°C and 35°C, both the fan and the bulb remain off.

DHT11 with fan code

```
#include "DHT.h"
#define DHTPIN A1 // Pin where the DHT11 is connected
#define DHTTYPE DHT11 // DHT 11
#define RELAYPIN 3 // Pin where the relay module is connected
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(RELAYPIN, OUTPUT);
digitalWrite(RELAYPIN, LOW); // Ensure the relay is off initially
}
void loop() {
delay(2000); // Wait a few seconds between measurements
float temp = dht.readTemperature(); // Read temperature as Celsius
// Check if any reads failed and exit early (to try again).
if (isnan(temp)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Serial.print("Temperature: ");
Serial.print(temp);
Serial.println(" *C");
if (temp > 35.0) {
digitalWrite(RELAYPIN, HIGH); // Turn the fan on
} else {
digitalWrite(RELAYPIN, LOW); // Turn the fan off
}
```

This code uses a DHT11 sensor to monitor the temperature and control a device (such as a fan) through a relay. The DHT11 sensor is connected to pin `A1`, and the relay controlling the fan (or other device) is connected to pin `3`.

In the `setup()` function, serial communication is initialized for displaying temperature readings. The DHT sensor is also initialized, and the relay pin is set as an output. The relay is initially turned off by setting `RELAYPIN` to LOW, ensuring that the fan or device starts in the off state.

In the `loop()` function, the system waits 2 seconds before reading the temperature from the DHT11 sensor. If the sensor fails to provide a valid reading, an error message is printed, and the loop restarts. If the temperature reading is successful, it is printed to the Serial Monitor. If the temperature exceeds 35°C, the relay is turned on (activating the fan or device). If the temperature is 35°C or below, the relay is turned off, keeping the fan or device off. The process repeats every 2 seconds, continuously monitoring the temperature and controlling the fan.

LCD display Code

```
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>
#include <OneWire.h>
#include <DallasTemperature.h>
// Pin definitions
#define TURBIDITY_PIN A0 // Only analog pin on ESP8266
#define TDS_PIN A0 // Assuming sharing the analog pin or using an ADC
#define TEMP_SENSOR_PIN 4 // GPIO4 (D2 equivalent)
#define TRIG_PIN 2 // GPIO2 (D4 equivalent)
#define ECHO_PIN 14 // GPIO14 (D5 equivalent)
#define MOISTURE_PIN A0 // Assuming sharing the analog pin or using an ADC
// LCD setup (adjust the address if needed, commonly 0x27 or 0x3F)
LiquidCrystal_PCF8574 lcd(0x27);
// Temperature sensor setup
OneWire oneWire(TEMP_SENSOR_PIN);
DallasTemperature sensors(&oneWire);
void setup() {
// Initialize I2C with GPIO5 (D1) as SDA and GPIO4 (D2) as SCL
Wire.begin(5, 4); // GPIO5 is SDA, GPIO4 is SCL
lcd.begin(16, 2);
lcd.setBacklight(255);
// Initialize temperature sensors
sensors.begin();
}
void loop() {
// 1. Read turbidity sensor value
int turbidityValue = analogRead(TURBIDITY_PIN);
// 2. Simulate TDS sensor value (if analog pin shared)
int tdsValue = analogRead(TDS_PIN);
```

```
// 3. Read temperature value
sensors.requestTemperatures();
float temperature = sensors.getTempCByIndex(0);
// 4. Measure distance using ultrasonic sensor
long duration, distance;
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
duration = pulseIn(ECHO_PIN, HIGH);
distance = (duration / 2) / 29.1;
// 5. Simulate moisture sensor value (if analog pin shared)
int moistureValue = analogRead(MOISTURE_PIN);
// Display values on the LCD in cycles
// Display Temperature
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(temperature);
lcd.print("C");
delay(2000); // Show for 2 seconds
// Display Distance
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Dist: ");
lcd.print(distance);
lcd.print("cm");
delay(2000); // Show for 2 seconds
// Display Turbidity
lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("Turbidity: ");
lcd.print(turbidityValue);
delay(2000); // Show for 2 seconds
// Display TDS
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("TDS: ");
lcd.print(tdsValue);
delay(2000); // Show for 2 seconds
// Display Moisture
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Moisture: ");
lcd.print(moistureValue);
delay(2000); // Show for 2 seconds
}
```

This code reads data from multiple sensors (turbidity, TDS, temperature, moisture) and displays the values on an LCD screen using I2C communication. It uses the following pin definitions:

- TURBIDITY_PIN, TDS_PIN, and MOISTURE_PIN share the analog pin A0, potentially via an ADC multiplexer or other mechanism.
- TEMP_SENSOR_PIN connects a Dallas temperature sensor (DS18B20), while TRIG_PIN and ECHO_PIN are used for an ultrasonic sensor.
- An I2C-based LCD is connected for displaying the sensor readings.

In the setup(), the I2C bus is initialized with specific pins for SDA (GPIO5) and SCL (GPIO4).

Moisture Sensor code

```
#define SoilMoistureSensorPin A0
#define VREF 3.3 // NodeMCU ADC reference voltage
#define SCOUNT 30 // Sampling times
int analogBuffer[SCOUNT]; // Store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0;
float averageVoltage = 0, moistureValue = 0;
void setup() {
Serial.begin(115200);
pinMode(SoilMoistureSensorPin, INPUT);
}
void loop() {
static unsigned long analogSampleTimepoint = millis();
if (millis() - analogSampleTimepoint > 40U) { // Every 40 milliseconds, read the analog value from the soil moisture sensor
analogSampleTimepoint = millis();
analogBuffer[analogBufferIndex] = analogRead(SoilMoistureSensorPin);
analogBufferIndex++;
if (analogBufferIndex == SCOUNT) {
analogBufferIndex = 0;
}
```

```
static unsigned long printTimepoint = millis();
if (millis() - printTimepoint > 800U) {
printTimepoint = millis();
for (int copyIndex = 0; copyIndex < SCOUNT; copyIndex++) {
analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
}
averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 1024.0;
moistureValue = calculateMoisture(averageVoltage);
Serial.print("Moisture Value: ");
Serial.print(moistureValue, 2);
Serial.println(" %");
}
int getMedianNum(int bArray[], int iFilterLen) {
int bTab[iFilterLen];
for (byte i = 0; i < iFilterLen; i++) {
bTab[i] = bArray[i];
}
int i, j, bTemp;
for (j = 0; j < iFilterLen - 1; j++) {
for (i = 0; i < iFilterLen - j - 1; i++) {
if (bTab[i] > bTab[i + 1]) {
bTemp = bTab[i];
```

```
bTab[i] = bTab[i + 1];

bTab[i + 1] = bTemp; }

if ((iFilterLen & 1) > 0) {

bTemp = bTab[(iFilterLen - 1) / 2];

} else {

bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2; }

return bTemp; }

float calculateMoisture(float voltage) {

// This function should convert voltage to moisture percentage

// Moisture = (Voltage * Scale) + Offset

float scale = 100.0 / VREF; // Example scale factor, needs calibration

float offset = 0.0; // Example offset, needs calibration

return (voltage * scale) + offset;
```

This code reads data from a soil moisture sensor connected to the analog pin A0 of a NodeMCU and calculates the soil moisture percentage. It takes multiple sensor readings over time, stores them in a buffer, and processes them by calculating the median value to reduce noise. The code reads from the sensor every 40 milliseconds, stores the values in an array of 30 samples, and every 800 milliseconds, it calculates the median analog value from the stored samples. This median value is converted into voltage using the reference voltage of 3.3V. The `calculateMoisture()` function then converts the voltage into a moisture percentage based on a simple calibration formula, which can be adjusted depending on the sensor's specific characteristics. The resulting moisture value is printed to the Serial Monitor as a percentage, providing real-time feedback on the soil moisture levels. This approach helps to smooth out fluctuations in sensor readings and gives more stable output values.

TDS Sensor Code

```
#include <EEPROM.h>
#define TdsSensorPin A0
#define VREF 3.3 // NodeMCU ADC reference voltage
#define SCOUNT 30 // Sampling times
int analogBuffer[SCOUNT]; // Store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;
void setup() {
Serial.begin(115200);
pinMode(TdsSensorPin, INPUT);
EEPROM.begin(512); // Initialize EEPROM
}
void loop() {
static unsigned long analogSampleTimepoint = millis();
if (millis() - analogSampleTimepoint > 40U) { //every 40 milliseconds, read the analog
value from the TDS sensor
analogSampleTimepoint = millis();
analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin);
analogBufferIndex++;
if (analogBufferIndex == SCOUNT) {
analogBufferIndex = 0;
}
}
static unsigned long printTimepoint = millis();
if (millis() - printTimepoint > 800U) {
printTimepoint = millis();
for (int copyIndex = 0; copyIndex < SCOUNT; copyIndex++) {
analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
```

```
}
averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF /
1024.0;
float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
float compensationVoltage = averageVoltage / compensationCoefficient;
tdsValue = (133.42 * compensationVoltage * compensationVoltage *
compensationVoltage - 255.86 * compensationVoltage * compensationVoltage + 857.39
* compensationVoltage) * 0.5; // Convert the voltage value to the TDS value
Serial.print("TDS Value: ");
Serial.print(tdsValue, 0);
Serial.println(" ppm");
}
}
int getMedianNum(int bArray[], int iFilterLen) {
int bTab[iFilterLen];
for (byte i = 0; i < iFilterLen; i++) {
bTab[i] = bArray[i];
}
int i, j, bTemp;
for (j = 0; j < iFilterLen - 1; j++) {
for (i = 0; i < iFilterLen - j - 1; i++) {
if (bTab[i] > bTab[i + 1]) {
bTemp = bTab[i];
bTab[i] = bTab[i + 1];
bTab[i + 1] = bTemp;
}
}
}
if ((iFilterLen & 1) > 0) {
bTemp = bTab[(iFilterLen - 1) / 2];
} else {
```

```
bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
}
return bTemp;
}
}
}
```

This code measures Total Dissolved Solids (TDS) using a TDS sensor connected to the analog pin `A0` on a NodeMCU. It stores analog readings in an array and calculates a stable TDS value using median filtering and compensation based on temperature. The `analogBuffer` array holds multiple TDS sensor readings over time, which are processed every 40 milliseconds. Every 800 milliseconds, the code calculates the median value from the readings to minimize noise, converts this median to a voltage, and applies temperature compensation to adjust for changes in water temperature. The compensated voltage is then converted to a TDS value using a calibration formula, and the result, in parts per million (ppm), is printed to the Serial Monitor. The `getMedianNum` function sorts the readings to find the median, reducing fluctuations in the final TDS value.

Turbidity Sensor Code

```
#define TurbiditySensorPin A0
#define VREF 3.3 // NodeMCU ADC reference voltage
#define SCOUNT 30 // Sampling times
int analogBuffer[SCOUNT]; // Store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0;
float averageVoltage = 0, turbidityValue = 0;
void setup() {
Serial.begin(115200);
pinMode(TurbiditySensorPin, INPUT);
}
void loop() {
static unsigned long analogSampleTimepoint = millis();
if (millis() - analogSampleTimepoint > 40U) { //every 40 milliseconds, read the analog
value from the turbidity sensor
analogSampleTimepoint = millis();
analogBuffer[analogBufferIndex] = analogRead(TurbiditySensorPin);
analogBufferIndex++;
if (analogBufferIndex == SCOUNT) {
analogBufferIndex = 0;
}
}
static unsigned long printTimepoint = millis();
if (millis() - printTimepoint > 800U) {
printTimepoint = millis();
for (int copyIndex = 0; copyIndex < SCOUNT; copyIndex++) {
analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
}
```

```
averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF /
1024.0;
turbidityValue = calculateTurbidity(averageVoltage); // Convert the voltage value to the
turbidity value
Serial.print("Turbidity Value: ");
Serial.print(turbidityValue, 2);
Serial.println(" NTU");
}
}
float calculateTurbidity(float voltage) {
// This formula depends on your specific sensor and calibration
// Here is an example conversion formula
float turbidity = -1120.4 * voltage * voltage + 5742.3 * voltage - 4352.9;
if (turbidity < 0) {
turbidity = 0;
}
return turbidity;
}
int getMedianNum(int bArray[], int iFilterLen) {
int bTab[iFilterLen];
for (byte i = 0; i < iFilterLen; i++) {
bTab[i] = bArray[i];
}
int i, j, bTemp;
for (j = 0; j < iFilterLen - 1; j++) {
for (i = 0; i < iFilterLen - j - 1; i++) {
if (bTab[i] > bTab[i + 1]) {
bTemp = bTab[i];
bTab[i] = bTab[i + 1];
bTab[i + 1] = bTemp;
}
```

```
}
}
if ((iFilterLen & 1) > 0) {
bTemp = bTab[(iFilterLen - 1) / 2];
} else {
bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
}
return bTemp;
}
```

The provided code is designed for use with an Arduino (or compatible) platform to read and process data from a turbidity sensor connected to analog pin A0. It defines key constants such as the reference voltage (3.3V) and the number of samples (30) to be averaged. In the `setup()` function, it initializes serial communication and configures the turbidity sensor pin as an input. The main loop captures sensor readings every 40 milliseconds, storing them in a circular buffer, and processes the data every 800 milliseconds by copying it to a temporary array. It calculates the median of the readings using the `getMedianNum` function, converts the median voltage to a turbidity value using a specific polynomial formula in `calculateTurbidity`, and prints the turbidity value in NTU (Nephelometric Turbidity Units) to the serial monitor. Overall, this code enables continuous monitoring of water quality by filtering noise from sensor readings and providing a reliable turbidity measurement.

Mobile app Dashboard

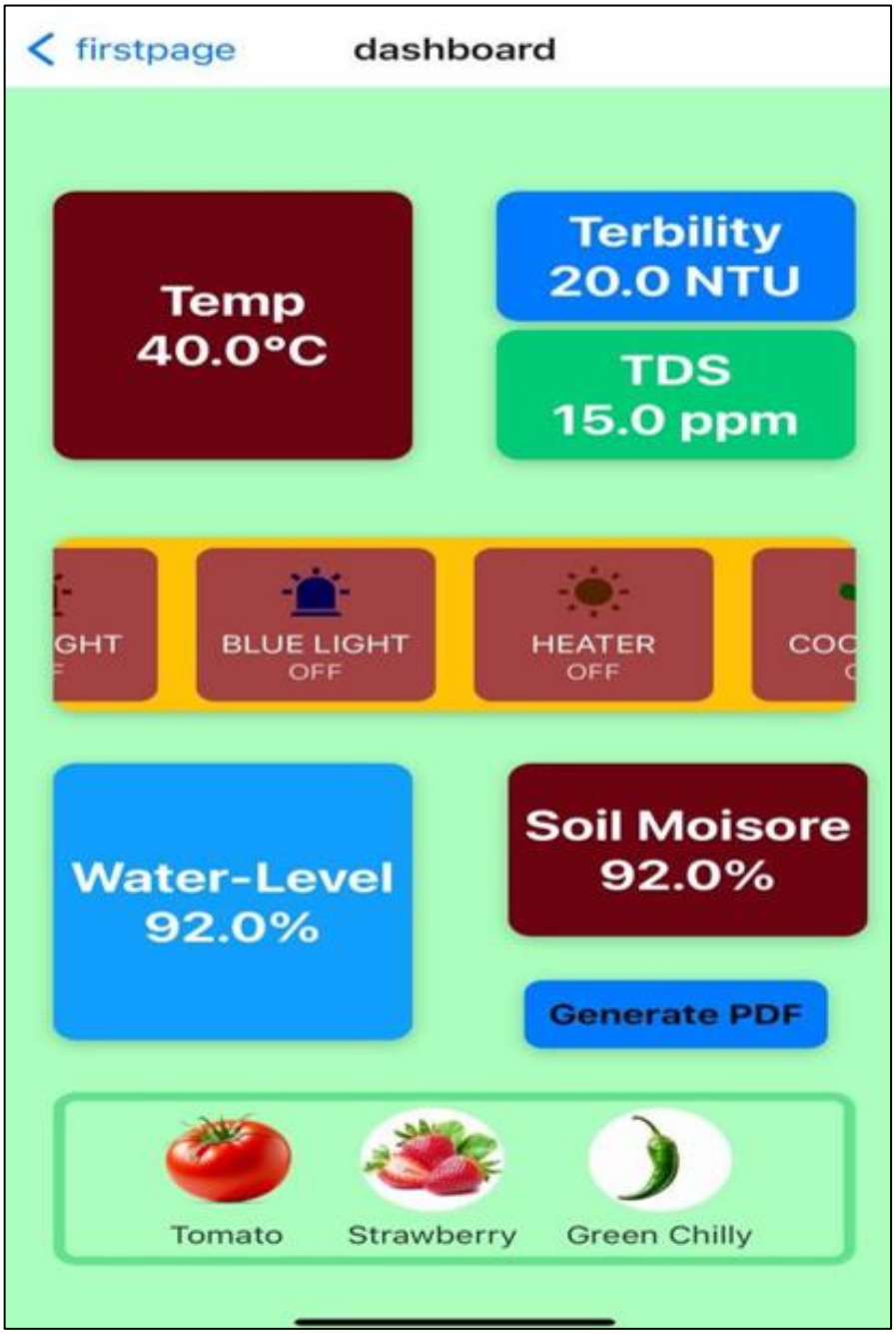

Figure 82-Mobile app Dashboard

Mobile app pdf in Real-Time Data Summary

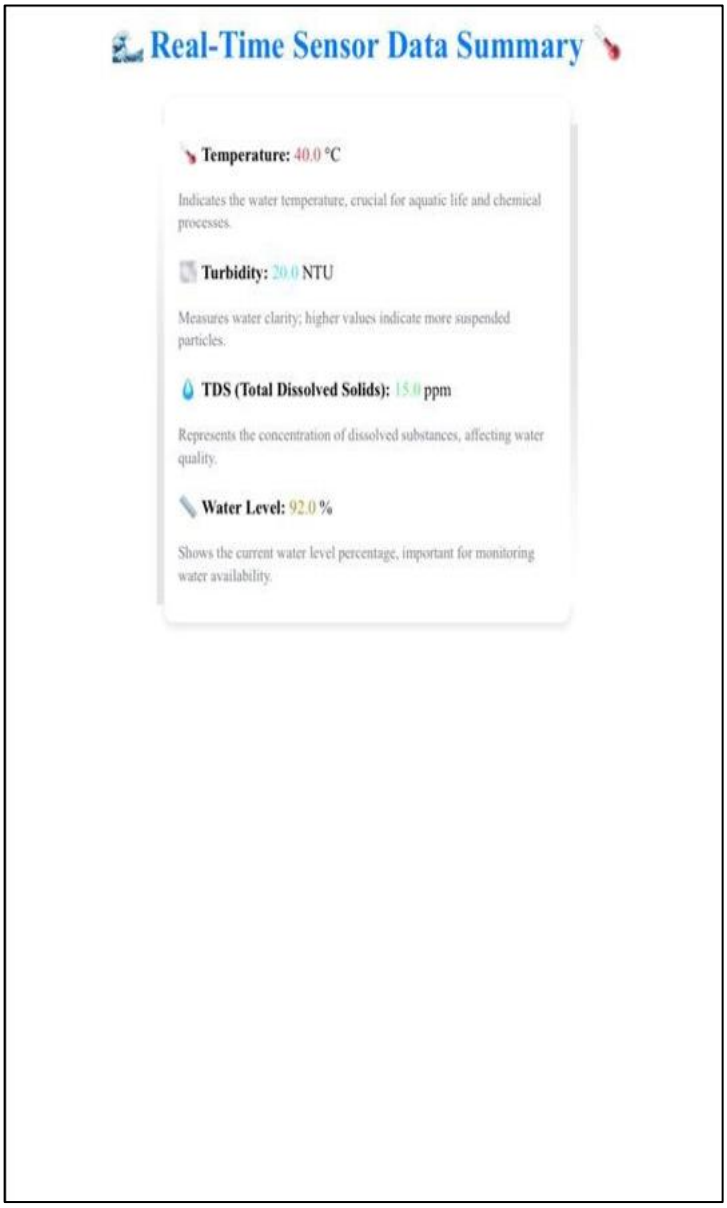

Figure 83-Mobile app pdf in Real-Time Data Summary

The First page in Dash board plant growth information

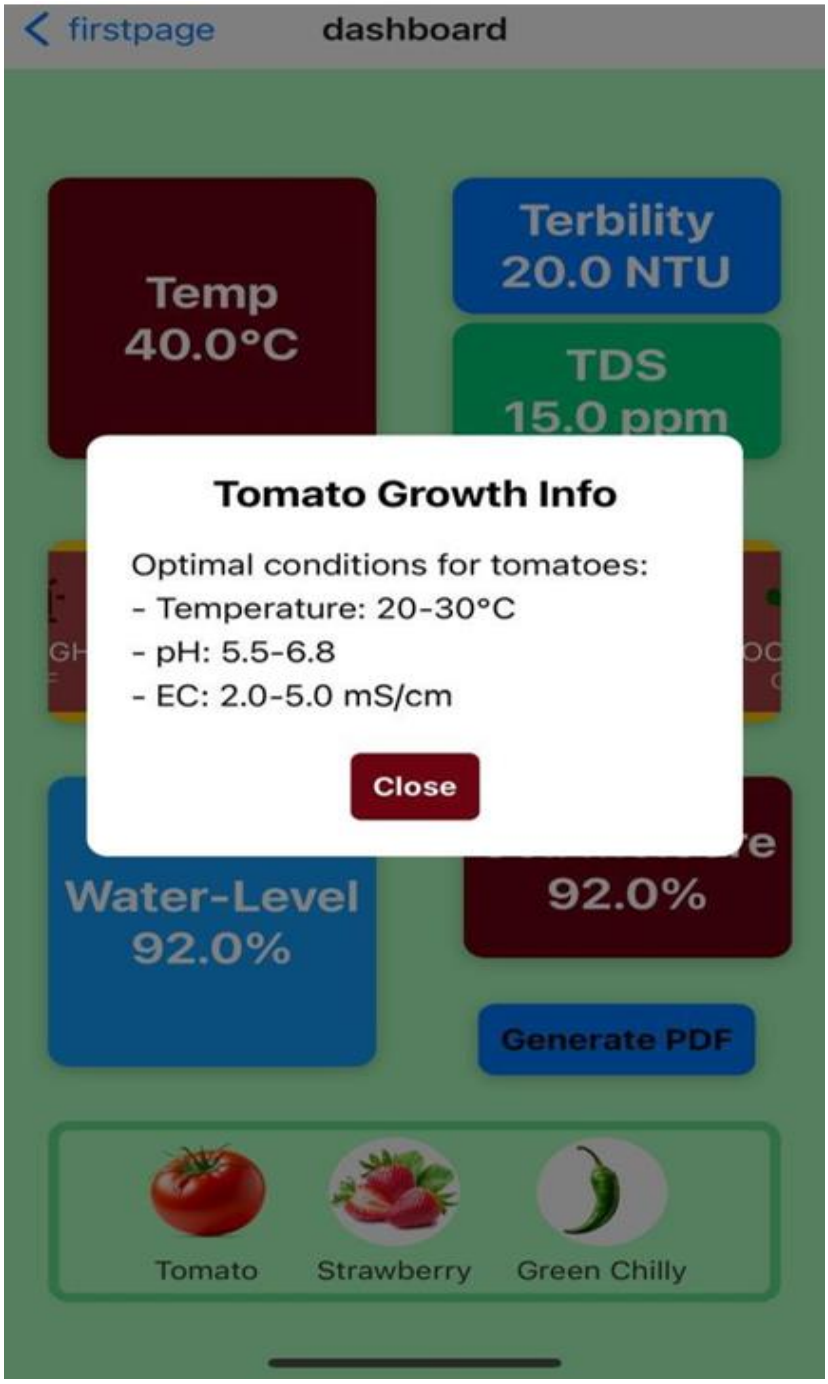

Figure 84-The First page in Dash board plant growth information