

**IoT based Hydroponic Deep-Water Culture (DWC) Monitoring System and Plant  
Condition Prediction using CNN**

by

Chang Kwong Ming  
18001074

Dissertation submitted in partial fulfilment of the requirements for the  
Bachelor of Engineering (Hons)  
(Computer Engineering)

JAN 2023

Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

# **CERTIFICATIONS**

## **CERTIFICATION OF APPROVAL**

### **IoT based Hydroponic Deep-Water Culture (DWC) Monitoring System and Plant Condition Prediction using CNN**

by

**Chang Kwong Ming**

**18001074**

A project dissertation submitted to the  
Computer Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfilment of the requirement for the  
**BACHELOR OF ENGINEERING (Hons)**  
**(COMPUTER ENGINEERING)**

Approved by,

---

(Pn. Khairul Nisak bt. Md. Hasan)

**UNIVERSITI TEKNOLOGI PETRONAS**

**TRONOH, PERAK**

**JAN 2023**

## CERTIFICATION OF ORIGINALITY

This is to certify that I am not responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

CHANG KWONG MING

## **ABSTRACT**

The amount of land and space available for conventional farming will continuously shrink as the world moves toward urbanization, and as population density rises, there will be a greater demand for food. Particularly during the Covid-19 crisis, which persists to this day and causes most people to spend their time alone in their homes. Hydroponics is mentioned as a possible solution to the problems. A method of urban agriculture known as hydroponics uses water as the primary growing medium rather than soil to produce large quantities of high-quality plants and vegetables. Despite the advantages that hydroponics offers, there are still some disadvantages. The disadvantages of hydroponics include its high level of technicality and methodical approach, which necessitates research before use. The reason is that individuals frequently struggle to establish the proper ratio of elements to be used, such as how much water to use, how acidic the nutrient solution should be, what kind of water to use, the temperature of the environment, etc. Therefore, integrating Internet of Things (IoT) sensors on the hydroponic system to help assess the pH level, nutritional content, temperature, and humidity will be one of the project's key priorities to address the issues. As determining the health of hydroponic plant growth is one of the obstacles that people must solve, a machine learning-based system for forecasting the plant health condition and plant growth rate will also be included in this project. The data gathered from the sensors will also be stored on a cloud platform, where it will be later shown for analysis. In addition, the findings of the machine learning will be compared to the data from the sensors to determine whether there is any correlation. In conclusion, the project's desired outcome is the development of a comprehensive hydroponic system utilizing machine learning and IoT.

## **ACKNOWLEDGEMENTS**

My sincere appreciation goes out to Pn. Khairul Nisak bt. Md. Hasan, who oversaw my Final Year Project (FYP), for her important advice, assistance, and encouragement. Her knowledge and suggestions were crucial in determining the focus and content of my FYP.

I also like to thank University Technology PETRONAS (UTP) for giving me the chance to work on this project as a requirement for my degree. My research has been made possible by the university's resources, facilities, and academic community, which has also helped me to succeed academically.

Finally, I would want to express my gratitude for my family and friends, who have supported me and kept me motivated throughout my academic career. I would not have been able to accomplish my academic and personal objectives without their love and support. Thank you once more to my supervisor and the university for all of their help and advice during my FYP.

## Table of Contents

<b>CERTIFICATIONS.....</b>	<b>II</b>
<b>ABSTRACT.....</b>	<b>IV</b>
<b>AKNOWLEDGEMENTS .....</b>	<b>V</b>
<b>CHAPTER 1 .....</b>	<b>1</b>
<b>INTRODUCTION.....</b>	<b>1</b>
<b>1.1 Background of Study .....</b>	<b>1</b>
<b>1.2 Problem Statements .....</b>	<b>3</b>
<b>1.3 Objectives.....</b>	<b>4</b>
<b>1.4 Scope of Study .....</b>	<b>4</b>
<b>CHAPTER 2 .....</b>	<b>5</b>
<b>LITERATURE REVIEW .....</b>	<b>5</b>
<b>2.1 What is Hydroponic? .....</b>	<b>5</b>
<b>2.1.1 Factor for Choosing DWC Plants.....</b>	<b>8</b>
<b>2.2 Relation of IoT with Hydroponic.....</b>	<b>11</b>
<b>2.3 Relation of Machine Learning with Hydroponic .....</b>	<b>14</b>
<b>2.3.1 Convolutional Neural Network (CNN) Architecture .....</b>	<b>16</b>
<b>Convolutional Layer .....</b>	<b>16</b>
<b>Pooling Layer .....</b>	<b>17</b>
<b>Fully Connected Layer .....</b>	<b>18</b>
<b>Dropout Layer .....</b>	<b>18</b>
<b>CHAPTER 3 .....</b>	<b>19</b>
<b>METHODOLOGY .....</b>	<b>19</b>
<b>3.1 Project Implementation Flowchart .....</b>	<b>19</b>
<b>3.1.1 Stage 1 Flowchart.....</b>	<b>19</b>
<b>3.1.2 Stage 2 Flowchart.....</b>	<b>20</b>
<b>3.2 Phases of Project Implementation .....</b>	<b>21</b>
<b>3.3 Phase 1: Project Specification/Criteria .....</b>	<b>22</b>
<b>3.3.1 DWC Hydroponic System Material Selection .....</b>	<b>22</b>
<b>3.3.2 DWC Hydroponic System Sensors and Components Selection .....</b>	<b>23</b>
<b>Raspberry Pi.....</b>	<b>23</b>
<b>Analogue-to-Digital Converter (ADS1115).....</b>	<b>24</b>
<b>Temperature &amp; Humidity Sensor.....</b>	<b>24</b>
<b>DIY MORE pH Sensor Kit .....</b>	<b>25</b>

EC Sensor (TDS Meter).....	26
Water Temperature Sensor (DS18B20) .....	26
3.3.3 Machine Learning Algorithm Selection .....	27
3.4 Phase 2: System Development and Design.....	29
3.4.1 DWC Hydroponic System Setup .....	29
3.4.2 IoT System Setup .....	30
3.4.3 Deep Learning System Setup .....	32
Four Stages of Plant Health Prediction.....	33
3.5 Phase 3: System Testing and Assessing.....	35
3.6 Gantt Chart .....	36
CHAPTER 4.....	38
RESULTS AND DISCUSSION .....	38
4.1 Preliminary Results of the IoT Sensors (Arduino) .....	38
4.2 Results of the IoT Sensors (Raspberry Pi).....	40
4.3 Results of the CNN Model.....	44
4.4 Correlation between the sensors data and Machine Learning .....	49
CHAPTER 5.....	54
CONCLUSION AND RECOMMENDATIONS.....	54
REFERENCES.....	56
APPENDIX.....	59
APPENDIX I: Pseudo Code on the Preliminary Results (Using Arduino IDE).....	59
APPENDIX II: Pseudo Code on the Final Results (Using Thonny IDE in Raspberry Pi).....	60
APPENDIX III: Pseudo Code on the Lettuce Health Prediction Model (Using CNN Model).61	
APPENDIX IV: Pseudo Code on Simple Linear Regression to Predict pH and EC Value .....	62
APPENDIX V: Pseudo Code on Segmentation and Size Measurement of Lettuce .....	63
APPENDIX VI: Pseudo Code on Using Captured Image to Predict the Lettuce Health Condition .....	64

## List of Figures

Figure 1.1: Internet of Things .....	2
Figure 2.1: The 6 Basic Types of Hydroponic Systems .....	5
Figure 2.3: Basic CNN Architecture.....	15
Figure 2.3.1a: Convolution Layer Process.....	16
Figure 2.3.1b: Pooling Layer Process .....	17
Figure 2.3.1c: Fully Connected Layer Process .....	17
Figure 2.3.1d: Dropout Layer Process .....	18
Figure 3.1.1: Project Implementation Flowchart (Stage 1).....	19
Figure 3.1.2: Project Implementation Flowchart (Stage 2).....	20
Figure 3.2: 3 Phases of Project Implementation .....	21
Figure 3.3.2a: Raspberry Pi Model 3 Microcontroller.....	23
Figure 3.3.2b: Analogue-to-Digital Converter (ADS1115).....	24
Figure 3.3.2c: Temperature & Humidity Sensor (DHT11).....	24
Figure 3.3.2d: DIY MORE pH Sensor Kit .....	25
Figure 3.3.2e: EC Sensor (TDS Meter V1.0).....	26
Figure 3.3.2f: Waterproof Water Temperature Sensor (DSB18B20) .....	26
Figure 3.4.1a: DWC Hydroponic System Setup.....	29
Figure 3.4.1b: DWC Hydroponic System Setup.....	29
Figure 3.4.2a: Block Diagram of IoT System Setup.....	30
Figure 3.4.2b: Transmission Process of Raspberry Pi to Thingsboard Cloud .....	31
Figure 3.4.3a: Block Diagram of Deep Learning System Setup.....	32
Figure 3.4.3b: Flow Diagram of Plant Health Prediction .....	33
Figure 3.6.1: FYP 1 Gantt Chart.....	36
Figure 3.6.2: FYP 2 Gantt Chart.....	37
Figure 4.1a: Preliminary Sensor's data using Arduino .....	38
Figure 4.1b: Circuitry of the IoT Sensors with Arduino (Video link to the draft setup of the IoT Sensors - Prototype.mp4) .....	38
Figure 4.1c: Cup of drinking water (Act as the hydroponic System) .....	39
Figure 4.2a: Sensors' Data Output Using Raspberry Pi .....	40
Figure 4.2b: Things Board Dashboard.....	41
Figure 4.2c: Timeseries Line Chart of pH Value.....	42
Figure 4.2d: Timeseries Line Chart of EC Value .....	42
Figure 4.2e: Timeseries Line Chart of Water Temperature .....	43
Figure 4.3a: Training and Validation Accuracy Chart.....	44
Figure 4.3b: Training and Validation Loss Chart .....	44
Figure 4.3c: Confusion Matrix of the Model .....	46
Figure 4.4a: Example Output of the Linear Regression Model .....	51
Figure 4.4b: Example Output of the Segmented Images .....	52



## List of Tables

<i>Table 2.1: Descriptions of the 6 basic hydroponic systems</i> .....	6
<i>Table 2.1.1: 5 Factors of Choosing DWC Plants</i> .....	8
<i>Table 2.1.2: Plants Appropriate for DWC System</i> .....	10
<i>Table 2.2: Comparison of Past Works (IoT Implementation)</i> .....	11
<i>Table 2.3: Comparison of Past Works (Machine Learning Implementation)</i> .....	14
<i>Table 3.3.1: Breakdown of Hydroponic System Items</i> .....	22
<i>Table 3.3.2: Breakdown Cost of the Sensors and Components</i> .....	27
<i>Table 4.4a: Lettuce Growing Progress</i> .....	49
<i>Table 4.4b: Example Result of the CNN Model Prediction</i> .....	53

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background of Study**

Up to 2015, 74% of Malaysians lived in urban regions; by 2025, that percentage is expected to progressively rise because of economic circumstances, a lack of available land, and the relocation of rural populations to urban areas [4]. The migration of rural residents into urban areas has increased the population density there, which has resulted in competition for food sources, nutrition, and security. Natural disasters, such as the COVID-19 that has spread over the world, can potentially endanger the food supply [3]. This virus has existed since the year 2019 came to an end. The Movement Control Order (MCO), a law that restricts public movement, was imposed by the Malaysian government during the pandemic to stop the virus' spread. This regulation has made it more difficult for agricultural products to reach urban areas, which has caused a short-term scarcity in the food supply.

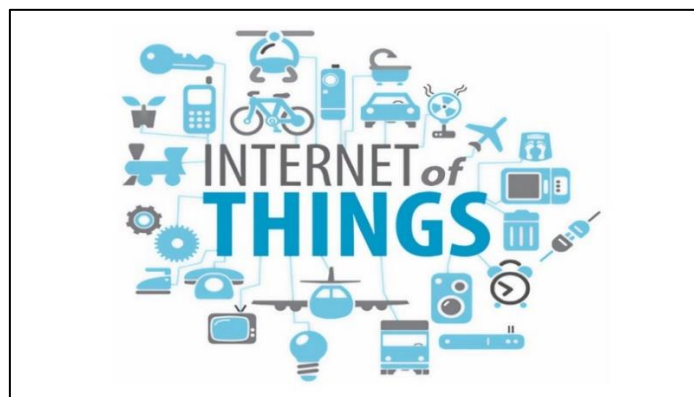
Malaysians are now more conscious of the value of food production because of this circumstance. This is demonstrated by the high rise in urban agricultural involvement from 18,687 in 2019 to 40,219 in 2020 supply [3]. Hydroponics is thus one of the most widely known urban agriculture methods. What exactly is hydroponics? To put it simply, hydroponics is a method of planting that doesn't use soil and instead relies primarily on water. The hydroponic plants' usage of nutrient-rich water helps to support the growth of the plants. Because this method is handled in a more methodical way than traditional farming, hydroponic plants typically have a higher rate of growth, better in quality, and stronger yields [5]. Urban dwellers have used this technique to create fresh, healthful food for several years [5].

The practice of hydroponics can be done in various ways. The Deep-Water Culture hydroponics system will be the main emphasis of this project because it is one of the easiest hydroponics systems for beginners. Without the need for expensive hydroponics equipment, it is quite simple to put together a deep-water culture system, and it works well with any sort of plant. The best feature of the water culture system is that the plant roots are positioned directly into the nutrient system, allowing the plants to absorb the nutrients with ease. The water culture

method of plant cultivation promotes rapid growth due to the direct access to nutrients and oxygen.

Regardless of the benefits hydroponics has for the ecosystem, there are some constraints that prevent it from being widely used in local farming. Based on a study by Rabiul Islam and Chamhuri Siwar from University Kebangsaan Malaysia's Institute for Environment and Development [2], it can be concluded that Malaysian urban farmers receive inadequate support and there is a lack of adequate land use planning for the sector. Moreover, the lack of an integrated development approach is restricting urban agriculture in Malaysia. The study also identified several major issues, including perceptions of limited space, limited resources, and education [2].

Due to the sharp rise in the world's food demand, there is an increasing need to normalize urban agriculture methods like hydroponics. To address this need, the objective of the project is to incorporate an IoT-based hydroponics monitoring system. The IoT is a network of physical objects, also known as "things," that are equipped with sensors, software, and other technologies for the purpose of connecting and exchanging data with other equipment and systems online. Additionally, IoT sensors can be used to gather real-time data from the hydroponic system, such as the water temperature, air temperature and humidity, pH value, etc. The user can then use the collected data for analysis and decision-making, which results in a more effective and efficient approach to maintaining the hydroponic system.



*Figure 1.1: Internet of Things*

## 1.2 Problem Statements

Like every other nation, Malaysia is progressing toward total urbanisation; as a result, there are more high-rise buildings and **more people living in condominiums** than ever before. Furthermore, there will be a **greater demand for food**, and more individuals will eventually begin to farm at home. Condominiums have less room than landed homes, which have more. Because **large hydroponic systems** require a **lot of space** and have **substantial maintenance costs**, they are not appropriate. Therefore, a hydroponic system that is more compact and affordable is more suited for those who live in urban areas.

Moreover, as was already said, hydroponic farming is a water-based technique that substitutes water for soil during plant growth. As a result, **determining the proper proportion of ingredients** to be used is a common **challenge** that affects **most people**. Because **not everyone has a green thumb** and the skills to incorporate an **automated system to monitor the plants**, for example, people frequently use too much or too little ingredients in their plants. In addition, city residents are constantly preoccupied with their jobs and lack free time to care for and keep an eye on the plants.

Additionally, most of the present smart hydroponic monitoring systems also have the **limitation of simply monitoring the plants' external conditions**, such as the water temperature, pH level, EC level, and so on. Since implementing a **machine learning algorithm to detect** and monitor the health **condition of the plants is not** a simple process to **undertake**, detection of the plants' condition is typically not included in the monitoring system. In contrast to our **human** naked eyes, which **produce varying findings** as we grow tired, **machines** have a better processing capability and do not experience fatigue, therefore they are better able to **accurately assess the health of plants**.

### **1.3 Objectives**

An IoT-based Hydroponic Deep-Water Culture (DWC) Monitoring System for indoor gardening and Plant Condition Prediction using CNN are what this project aims to create. Consequently, the following objectives must be fulfilled to achieve the goal:

- i. To create and build a cost-efficient, indoor-suitable prototype hydroponic monitoring system.
- ii. To integrate the Internet of Things sensors on the system to measure the water's nutritional content, temperature, and air humidity & temperature.
- iii. To integrate a machine learning-based system to predict the health and growth rate of the hydroponic plants.
- iv. To have a cloud database to house the sensors data from IoT sensors and machine learning-based system, then display the data on an interactive dashboard.

### **1.4 Scope of Study**

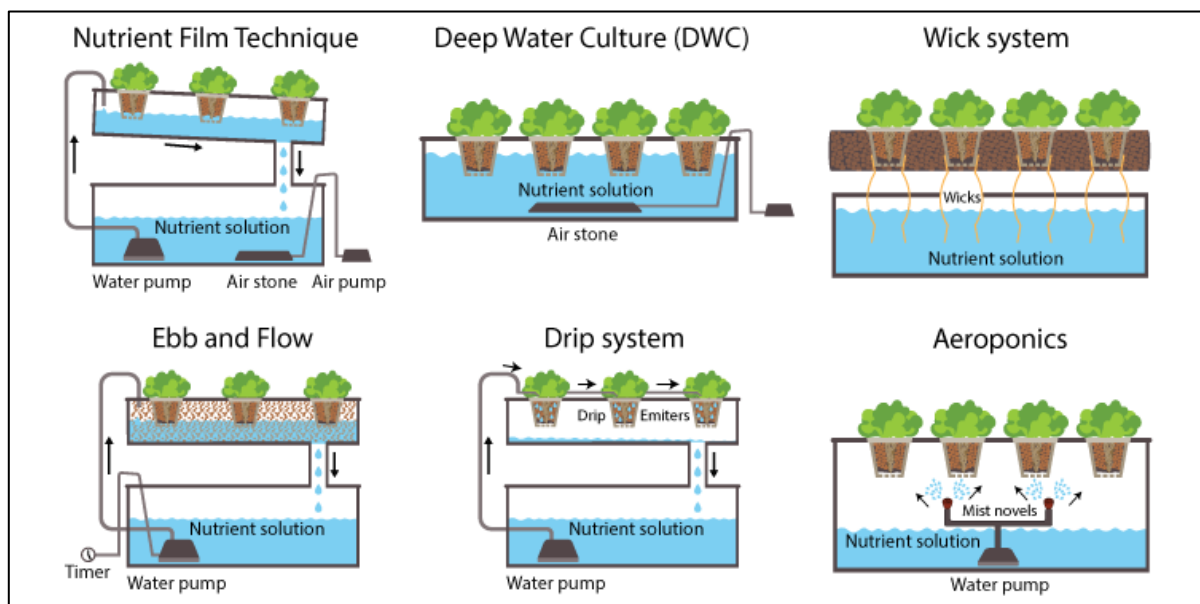
The goal of this research is to use the Internet of Things to monitor the deep-water culture hydroponics system, making it simpler for beginners and those who are interested in the system but don't have a lot of free time to manage it. The hydroponics system will incorporate several different types of sensors, including humidity and temperature sensors, water temperature sensors, EC sensors, and pH value detectors. Additionally, employing image processing to keep an eye on the health of the lettuce plants. Moreover, this project will use the Raspberry Pi microcontroller board as a central processing unit to gather data from the hydroponics system and send the data to a cloud database. Finally, create an interactive dashboard to visualize the database's real-time data for analysis.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 What is Hydroponic?

Hydroponic plants are grown without soil, using an inert growing medium and nutrient-rich solutions, oxygen, and water. This method facilitates rapid development, higher yields, and higher-quality products. There are six basic types of hydroponic systems, namely the Wick System, Deep Water Culture System (DWC), Ebb and Flow System, Drip, Nutrient Film Technique (NFT), and Aeroponic. While there are many variations of these fundamental types of systems, all hydroponic techniques are modifications of these six. Figure 2.1 portrays the six main types of hydroponic systems, and Table 2.1 has descriptions of each type.



*Figure 2.1: The 6 Basic Types of Hydroponic Systems*

*Table 2.1: Descriptions of the 6 basic hydroponic systems*

Hydroponic Systems	Descriptions
Nutrient Film Technique (NFT)	<ul style="list-style-type: none"> <li>• It is a hydroponic active system, which is the preferred method for commercial growers.</li> <li>• It is suitable for crops with short harvest cycles.</li> <li>• Since the nutrient solution is continuously pumped from the reservoir to the grow tray, there is no need for a pump timer.</li> <li>• The grow tray is constructed with enough slope to allow the solution to trickle down rather than flood it.</li> <li>• Nutrients are recycled back into the reservoir.</li> <li>• The most scalable hydroponic systems.</li> </ul>
<b>Deep Water Culture (DWC)</b> <b>(Selected for this project)</b>	<ul style="list-style-type: none"> <li>• The simplest hydroponic active system.</li> <li>• The plant roots are suspended in the nutrient solution and receive oxygen directly through an air stone and air pump.</li> <li>• To assist keep plants secure, they are put in net pots with grow medium.</li> <li>• The root of the plants will always remain submerged.</li> <li>• Systems can be implemented in both small and large scale.</li> </ul>
Wick	<ul style="list-style-type: none"> <li>• The only passive system out of all the hydroponic systems. (Require no electricity)</li> <li>• Through a wick in the growing medium, nutrient solution is pulled up to plant roots.</li> <li>• Function well with miniature houseplants and herbs.</li> <li>• Smaller in scale.</li> </ul>
Ebb and Flow	<ul style="list-style-type: none"> <li>• It is a hydroponic active system.</li> <li>• Temporarily floods a grow tray with nutrient solution using a submerged pump, then drains the solution back into the reservoir.</li> </ul>

	<ul style="list-style-type: none"> <li>• The timer is set to check on the plants multiple times every day, depending on factors such as the size and type of the plants, the temperature, humidity, and the type of growing medium being used.</li> <li>• Each plant's solution volume cannot be adjusted.</li> <li>• Effective for cultivating the same plant species in each container.</li> <li>• Beneficial for cuttings, young plants, and seedlings.</li> </ul>
Drip	<ul style="list-style-type: none"> <li>• It is a hydroponic active system.</li> <li>• A timer is used to control the submerged pump.</li> <li>• Reservoir and grow tray are separated.</li> <li>• Each plant in the grow tray is attach with a drip line.</li> <li>• Each plant's base receives direct nutrient solution drips.</li> </ul>
Aeroponics	<ul style="list-style-type: none"> <li>• It is a hydroponic active system.</li> <li>• The most cutting-edge hydroponic system available.</li> <li>• Suspended plants have roots hanging below them.</li> <li>• A second, higher pressure pump mists nutrient solution over the roots after nutrient solution has been pushed from the reservoir.</li> <li>• Advanced timing is necessary; more frequent misting is necessary because each spray offers less than other systems.</li> <li>• The more the nutrient solution moved, the more oxygen it contained, enabling plants to develop more quickly.</li> </ul>

Despite the benefits of each hydroponic system, the Deep-Water Culture (DWC) hydroponic system will be the main emphasis of this project. To achieve the goals of this project, which is to construct an indoor hydroponic system, the DWC is one of the best systems for a small-scale hydroponic system. The DWC system is also the simplest active hydroponic system there is. Contrary to many hydroponic systems, the DWC system is incredibly affordable and simple to put together. The DWC system's equipment requirements are easily met by purchasing it from


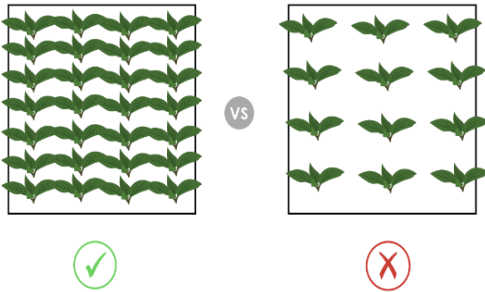





any local gardening store or online retailer. The only maintenance needed once the DWC system is installed is to ensure that the air pump is supplying oxygen to the air stone and to replace the nutritional solution.

### 2.1.1 Factor for Choosing DWC Plants

Finding the right kind of plant to cultivate is by no means an easy undertaking. Therefore, it's crucial to comprehend the requirements for the plants that are optimal for DWC rather than simply following recommendations at face value. The five factors for selecting the best plants for the DWC hydroponic system are listed in Table 2.1.1 below.

*Table 2.1.1: 5 Factors of Choosing DWC Plants*

Factors	Images	Descriptions
Weight		<ul style="list-style-type: none"> <li>• Rafts typically last for a long time and are reasonably priced, but they can only bear so much weight.</li> <li>• Small, light plants work best for DWC.</li> <li>• Top-heavy plants are more likely to tip over or break at the stems.</li> </ul>
Footprint		<ul style="list-style-type: none"> <li>• Since DWC systems are often too heavy to stack, they operate on a single horizontal plane.</li> <li>• Volume to expanding area ratio is 1:1.</li> <li>• Enables denser growing for restricted area.</li> <li>• Efficiently populate the horizontal plane.</li> </ul>

		<ul style="list-style-type: none"> <li>• Make sure that the plant site spacing on your rafts corresponds to the size of your plant.</li> </ul>
Water-friendly		<ul style="list-style-type: none"> <li>• DWC systems don't work well for plants and herbs that require "dry feet".</li> <li>• Thirsty plants.</li> <li>• Choose seed types that adore water.</li> </ul>
Yield		<ul style="list-style-type: none"> <li>• Make sure the plant will make money.</li> <li>• Make sure you have a venue to sell the harvest</li> </ul>
Breeding		<ul style="list-style-type: none"> <li>• Hybrid plants are more feasible in DWC because of their durability and vitality.</li> <li>• Hybrids that are better known and have undergone more research offer a more reliable return on investment.</li> </ul>

The lettuce plant has been chosen as the plant for the DWC hydroponic system in this project. The reason the lettuce plant was chosen was that it met the requirements outlined in table 2.1.1. Furthermore, regardless of the farming methods employed, lettuce plants—including DWC—are regarded for being among the easiest plants to grow. Any growers new to DWC are urged to think about lettuce as a crop for their farm because of its quick growth cycle and high market demand. There are numerous other plant species besides lettuce plants that are appropriate for DWC. Some of the plants from the example that are appropriate for DWC are shown in Table 2.1.2.

*Table 2.1.2: Plants Appropriate for DWC System*

Plants	Images	Grow Time
<b>Lettuce</b> (Selected for this project)		<ul style="list-style-type: none"> <li>• 5 – 6 weeks from seed</li> </ul>
Okra		<ul style="list-style-type: none"> <li>• 7 – 9 weeks from seed</li> </ul>
Kale		<ul style="list-style-type: none"> <li>• 5 – 6 weeks from seed</li> </ul>
Collard Greens		<ul style="list-style-type: none"> <li>• 7 – 8 weeks from seed</li> </ul>
Sorrel		<ul style="list-style-type: none"> <li>• 4 – 6 weeks from seed</li> </ul>
Bok Choy		<ul style="list-style-type: none"> <li>• 4 – 5 weeks from seed</li> </ul>
Chard		<ul style="list-style-type: none"> <li>• 4 – 5 weeks from seed</li> </ul>

## 2.2 Relation of IoT with Hydroponic

Even though the hydroponic method offers several benefits and conveniences, many farmers continue to use traditional farming techniques in the face of the COVID-19 outbreak [6]. This is since the hydroponic approach requires more systematic control and precision than the conventional agricultural method. To advance agriculture, we need a technology-based hydroponic system that can increase production, optimize yields, provide better products, and make it simpler for farmers to manage and care for their plants.

Every element of life, including agriculture, adapts by utilizing the most up-to-date technologies that can offer convenience. This hydroponic system can be connected to the internet of things to make it easier for farmers to perform their responsibilities during the COVID-19 pandemic [6]. The benefits of using the IoT in hydroponic farming activities include the ability to monitor and manage processes as well as take care of plants whenever and wherever you want.

This section of the report will examine a few real-world hydroponic systems that have IoT integration. The varieties of microcontrollers that have been employed and the kinds of sensors that have been incorporated into their hydroponic system are the factors that will be analyzed. Show the list of the components that have been utilized in some prior studies on integrating IoT in hydroponic systems in Table 2.2 below.

*Table 2.2: Comparison of Past Works (IoT Implementation)*

No	Sources	Microcontroller	Sensors
1	[7]	NodeMCU	<ul style="list-style-type: none"><li>• Humidity &amp; Temperature Sensor</li><li>• pH Sensor</li><li>• Ultrasonic Sensor</li><li>• Soil Moisture Sensor</li></ul>
2	[5]	NodeMCU	<ul style="list-style-type: none"><li>• Humidity &amp; Temperature Sensor</li><li>• pH Sensor</li><li>• Ultrasonic Sensor</li></ul>

			<ul style="list-style-type: none"> <li>• Water Temperature Sensor</li> </ul>
3	[8]	NodeMCU	<ul style="list-style-type: none"> <li>• Humidity &amp; Temperature Sensor</li> <li>• pH Sensor</li> <li>• EC Sensor</li> </ul>
4	[9]	Arduino and Raspberry pi	<ul style="list-style-type: none"> <li>• Humidity &amp; Temperature Sensor</li> <li>• pH Sensor</li> <li>• Water Temperature Sensor</li> </ul>
5	[10]	Raspberry pi	<ul style="list-style-type: none"> <li>• Humidity &amp; Temperature Sensor</li> <li>• pH Sensor</li> <li>• EC Sensor</li> <li>• Water Level Sensor</li> <li>• Water Temperature Sensor</li> </ul>
6	[11]	NodeMCU	<ul style="list-style-type: none"> <li>• Humidity &amp; Temperature Sensor</li> <li>• pH Sensor</li> <li>• Water Temperature Sensor</li> </ul>
7	[12]	Arduino	<ul style="list-style-type: none"> <li>• Humidity &amp; Temperature Sensor</li> <li>• pH Sensor</li> <li>• LDR Sensor</li> <li>• IR Sensor</li> </ul>
8	[13]	Arduino	<ul style="list-style-type: none"> <li>• Humidity and Temperature Sensor</li> <li>• pH Sensor</li> <li>• EC Sensor</li> </ul>

			• LDR Sensor
--	--	--	--------------

To aid in the growth of plants in the hydroponic system, a variety of sensors are used, as shown in Table 2.2. All past models include three types of sensors, namely pH, humidity, and temperature sensors. The pH sensor is primarily used to monitor the acidity level of nutrient solutions, which varies depending on the types of plants being grown. Meanwhile, the humidity and temperature sensors help keep track of the temperature and water vapor concentration in the environment surrounding the plants, as these factors can affect their growth rates.

Additionally, in the 1, 2, and 5 systems ultrasonic and water level sensors were utilized to monitor the tray's water level and check for leaks in the grow tray. A sensor known as the EC sensor was used in systems 3,5, and 8 to assess the purity and cleanliness of the water used. Since the EC and pH level of the nutrient solution may fluctuate depending on the water temperature, the water temperature sensor that was discovered in the 2,4,5 and 6 systems was employed to measure the water temperature.

In addition to the above-described sensors, some systems also use LDR sensors, IR sensors, and soil moisture sensors. Light intensity is measured using an LDR sensor, and infrared radiation is detected using an IR sensor. As for measuring soil moisture, sensors are employed to measure plant moisture.

Moreover, through the Table 2.2 we can observe that there are 3 types of microcontrollers Arduino, Raspberry Pi and NodeMCU has been used in the past systems for Integration of IoT into hydroponic system. Despite the types of microcontrollers has been used each of them serve the same purpose of controlling and monitoring of the sensors integrated on the hydroponic system.

### 2.3 Relation of Machine Learning with Hydroponic

IoT technology can automate hydroponics, however managing the hydroponics system also requires intelligence. As a result, artificial intelligence subfield machine learning is useful in this case. Controlling and monitoring the health and development of plants grown in hydroponic systems has been made easier with the use of machine learning. In hydroponic farming, there is steadily less risk of crop failure because no soil is used. Therefore, the risk that plants would catch a disease or display abnormal growth can also be decreased by using machine learning in this system to spot variations in growth.

How does machine learning operate then? Machine learning assists in giving computers the ability to carry out tasks independently after being trained for a certain purpose. First and foremost, a machine must first think and learn like a human person to think like a human mind. The human mind makes decisions for the future based on the events and information from the past that it has been exposed to.

In this part of this report, we'll take a closer look at a few actual hydroponic systems that are using machine learning. Moreover, analyze the various Machine Learning approaches and algorithms that were used in the hydroponic system. The list of prior work that has integrated machine learning into their hydroponic system, along with the justification for doing so, is presented in Table 2.3 below.

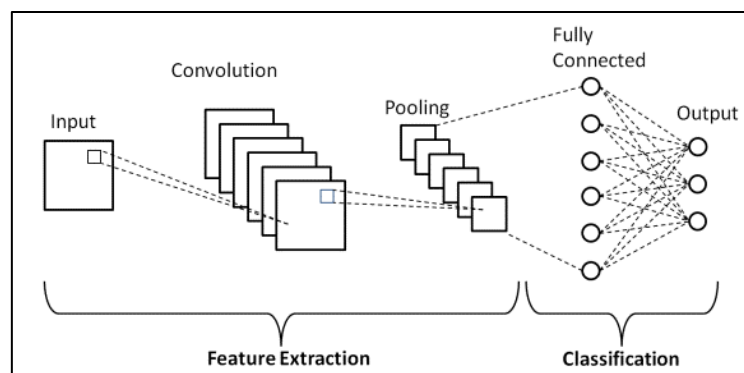
*Table 2.3: Comparison of Past Works (Machine Learning Implementation)*

No	Sources	Technique used	Purposes
1	[15]	Bayesian Network	To perform predictive analysis on the results of the hydroponic system using the sensors data collected.
2	[16]	Artificial Neural Network (ANN)	To forecast the results of the hydroponic system in terms of potential hydrogen and electrical conductivity.

3	[17]	Bayesian Network	Utilizing the information gathered from the sensors to perform a prediction analysis on the hydroponic system's output.
---	------	------------------	---

Three of the systems listed in the table above have a significant flaw, namely that they only focus on using machine learning algorithms like the Bayesian Network and ANN to predict the outcomes of hydroponic systems and what the best course of action is based on a pool of previously collected sensor data, instead of using the live or latest set of data. Therefore, the aim of this project will be developing a hydroponic system with improved intelligence and accuracy by using both IoT and Machine Learning to achieve the project's goals and advance the current hydroponic systems.

To undertake this project, a real-time image of a hydroponic plant will first be acquired using a camera. The plant images will then be sent into a machine learning algorithm to forecast the plant state of the hydroponic system, in this case, a lettuce hydroponic system. The model for detecting plant diseases in this project will be a convolutional neural network (CNN), a deep learning technique. Why CNN and what is deep learning? The principal applications of the machine learning subfield known as "Deep Learning" include the classification of images, object recognition, and natural language processing. One of the most used algorithms for detecting plant diseases right now is CNN. In addition, it performs target identification and image recognition more accurately and generally than conventional approaches such as K-Nearest Neighbors, Random Forest, ANN, etc. [19].



*Figure 2.3: Basic CNN Architecture*



### 2.3.1 Convolutional Neural Network (CNN) Architecture

As depicted in Figure 2.3, the CNN architecture consists of three main layers: convolutional, pooling, and fully connected layers. CNN Architecture is the result of the three layers stacking together. The following will describe the kind of function that each of the three layers does inside the CNN Architecture.

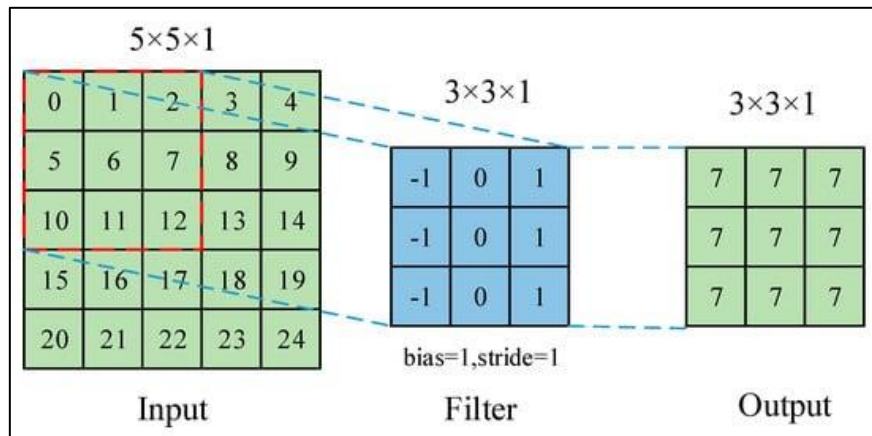


Figure 2.3.1a: Convolution Layer Process

#### Convolutional Layer

In the CNN architecture, the first layer is the convolutional layer. This layer contains various features extracted from the source image. A mathematical convolution is performed between the input image and a filter with a specific  $M \times M$  matrix size [20]. The filter is moved across the input image, and the dot product between the filter and the areas of the image that correspond to the filter size is computed [20].

The resulting output, also known as the feature map, provides information on the image, such as its corners and edges [20]. Subsequent layers are then given access to this feature map to learn new features from the input image. The convolutional layers in CNNs retain the spatial relationship between the pixels [20].

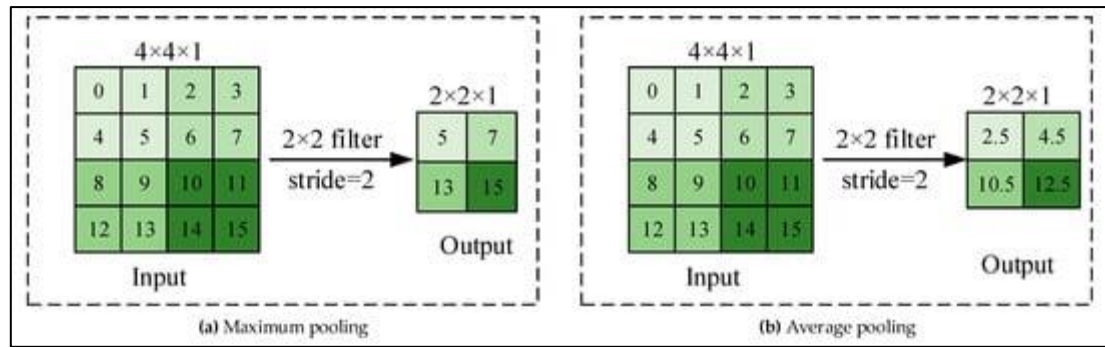


Figure 2.3.1b: Pooling Layer Process

## Pooling Layer

Following the convolutional layer, the next layer in the CNN architecture is the pooling layer. The main purpose of this layer is to reduce the size of the convolved feature map, thereby reducing computing costs [20]. This is accomplished by decreasing the connections between layers and working independently on each feature map [20]. There are various types of pooling methods depending on the technique used. Max pooling is primarily driven by the feature map, while average pooling computes the average of the elements in a picture part of a certain size. Sum pooling calculates the cumulative sums of the components inside the given section.

In essence, this layer summarizes and generalizes the features that the convolution layer collected, assisting the networks in independently identifying the features. Consequently, the network's overall computations were lowered [20].

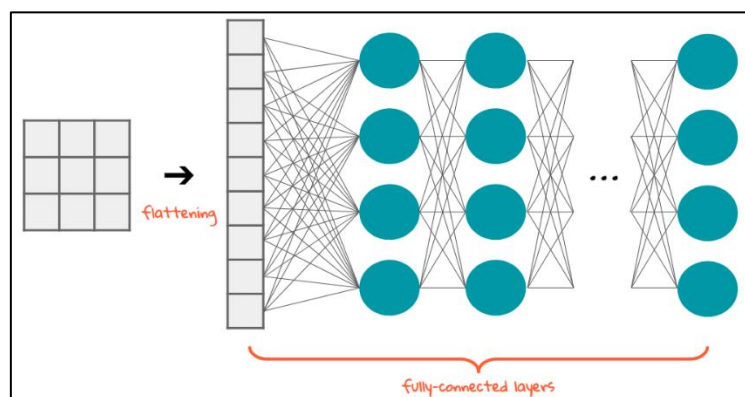


Figure 2.3.1c: Fully Connected Layer Process

## Fully Connected Layer

The FC layer, also known as the fully connected layer, serves as a link between two layers by connecting the neurons and adding weights and biases to them [20]. Usually, these layers are positioned prior to the output layer and form the last few layers of the CNN architecture.

After passing through the preceding layers, the input image is flattened before being transmitted to the FC layer. The flattened vector then undergoes several more FC layers where typical mathematical operations are performed. Additionally, this is where the classification process occurs. Furthermore, as more layers are added, the performance of the system improves [20]. This layer of the CNN architecture reduces the need for human supervision [20].

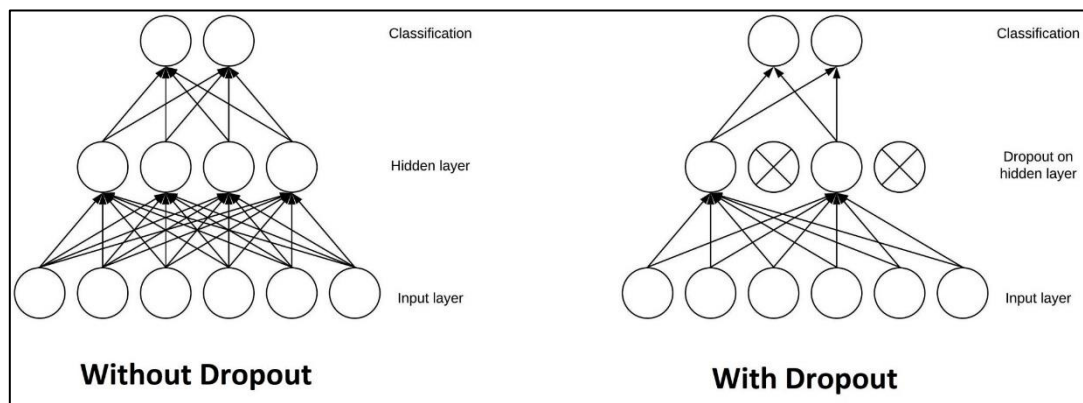


Figure 2.3.1d: Dropout Layer Process

## Dropout Layer

In addition to the other three levels, there is also a fourth layer called the dropout layer. When all features are connected to the FC layer, the training dataset may become overfit, which may negatively affect the model's performance [20].

To tackle the issue, the dropout layer is utilized, which shrinks the size of the model by eliminating a few neurons from the neural network during training. Dropout improves the model's performance since it lessens overfitting by streamlining the network [20].

## CHAPTER 3

### METHODOLOGY

#### 3.1 Project Implementation Flowchart

##### 3.1.1 Stage 1 Flowchart

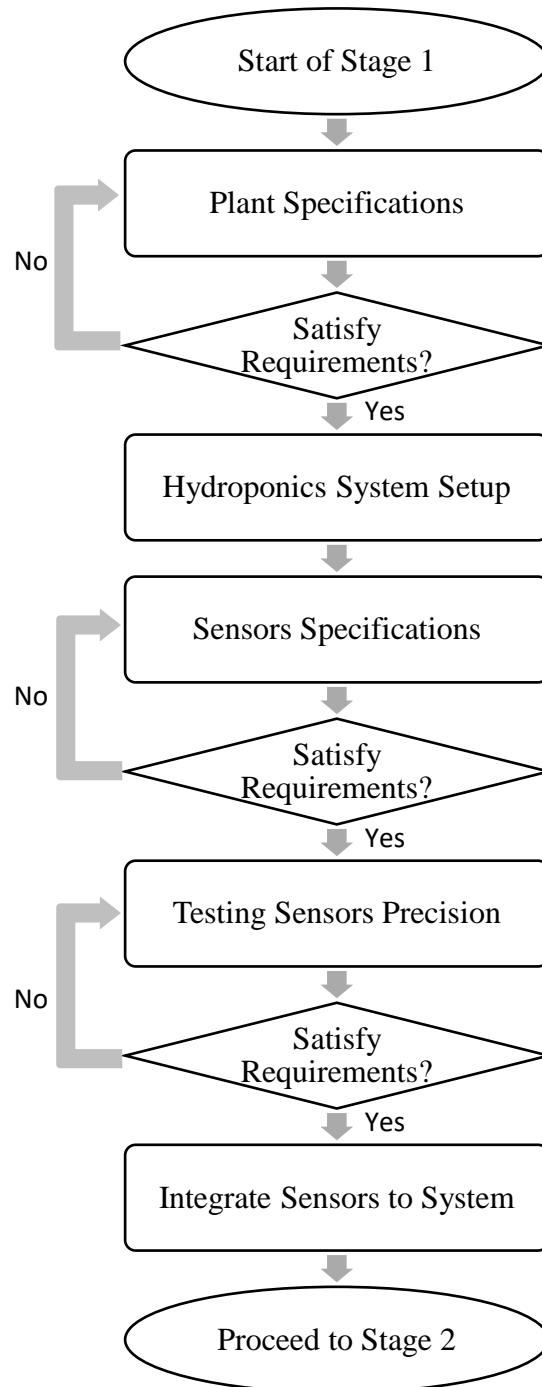


Figure 3.1.1: Project Implementation Flowchart (Stage 1)

### 3.1.2 Stage 2 Flowchart

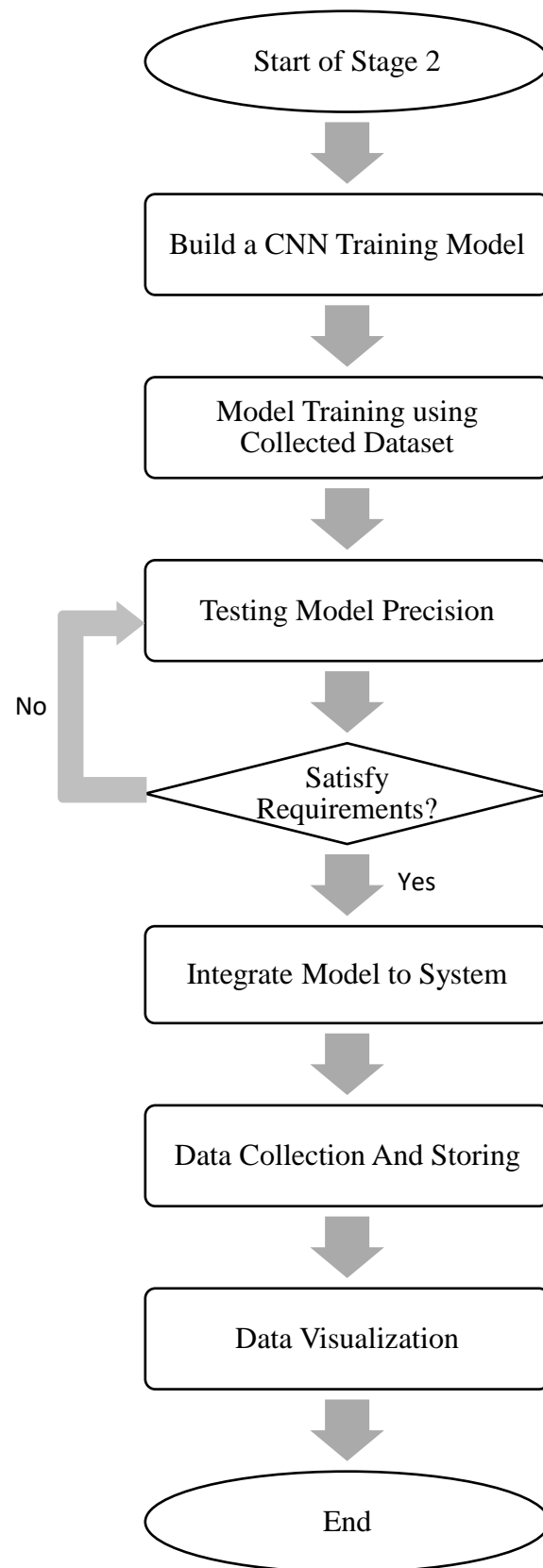
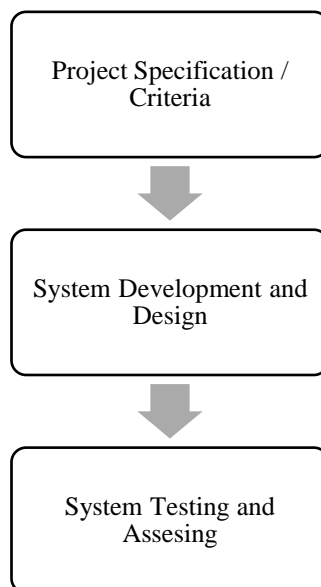


Figure 3.1.2: Project Implementation Flowchart (Stage 2)

### 3.2 Phases of Project Implementation

To ensure a successful implementation of the project, I've categorized my tasks into three primary phases, as illustrated in Figure 3.2. Establishing the project criteria is the initial step. Its goal is to determine the components that the project needs. Therefore, the integration of an IoT Deep-Water Culture Hydroponic Monitoring System will be the focus of this project's research. The second phase, which is the system development and design comes after the first phase. The project system prototype will be built at this phase. Finally, we reach the third and most crucial phase. To make sure the outcomes are accurate, testing and system data analysis are done in this last phase.



*Figure 3.2: 3 Phases of Project Implementation*

### 3.3 Phase 1: Project Specification/Criteria

#### 3.3.1 DWC Hydroponic System Material Selection

The size and cost of the materials are the two key factors to be taken into consideration to fulfil the project's goals, which are to construct an indoor, cost-effective hydroponic system. Nowadays, the e-commerce sector makes it possible to buy just about everything (e.g., Shopee and Lazada). Therefore, a complete bundle of a cheap and affordable hydroponic set can easily be purchased through the internet instead of travelling to a store to choose and compare the goods one by one. Because they are not included in the bundle, the water pump and growth LED light must be purchased separately. Furthermore, I purchased rockwool as a replacement for the complimentary sponges that came with the bundle. Why are the extra three components needed? It is done to enhance the plant's growth. The items breakdown of the hydroponic system that I purchased is shown in table 3.3.1 below.

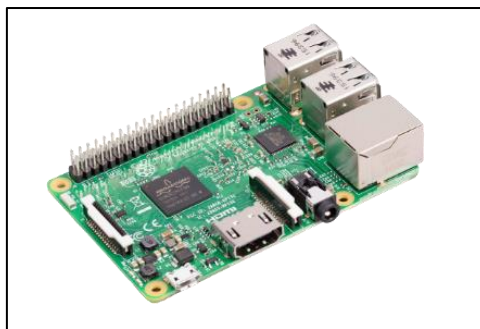
*Table 3.3.1: Breakdown of Hydroponic System Items*

Item	Quantity	Price (RM)
Tray	1	Bundle total is 14.70 (seeds I get from my uncle)
Tray Cover	1	
Net pot	6	
AB Fertilizer	1	
Rockwool	10	4.79
Growing LED Light	1	9.57
USB Water Pump	1	12.38
Total (RM)	RM 41.44	

### 3.3.2 DWC Hydroponic System Sensors and Components Selection

In addition to picking the materials for the DWC System, we now need to choose the sensors and components that will help us complete this project's goal of establishing an IoT-based DWC hydroponic monitoring system. Therefore, the following sensors and components are being chosen to be put into the system after considering what are the characteristics of leafy plants and characteristics of a DWC hydroponic system, as well as conducting research on some previous comparable projects.

#### Raspberry Pi



*Figure 3.3.2a: Raspberry Pi Model 3 Microcontroller*

The 26 GPIO Pins of the Raspberry Pi are incredibly helpful for embedded projects and connecting electronics. These pins are quite helpful for understanding component interaction. Due to the huge amount of GPIO pins provided, numerous digital sensors can be combined. It works with practically all the Arduino's available peripherals.

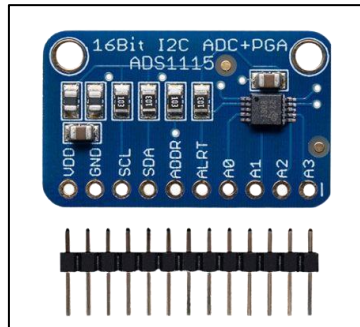
It supports numerous sensors at once, as was discussed in the part above about how many GPIO pins it has. This makes this board suitable for the project since the system will incorporate several analogue sensors.

Furthermore, the Raspberry Pi runs on a Linux environment, allowing us to code in nearly any language, including C, C++, C#, Java, Python, etc., in contrast to microcontrollers like Arduino, which only support C/C++. Python will be used in this project since it provides higher readability for code and enables users to type concepts in less lines of code.



Additionally, the 1.6 GHz processor in the Raspberry Pi allows for a quicker processing speed, which translates to better performance when compared to other microcontrollers like the Arduino.

### **Analogue-to-Digital Converter (ADS1115)**

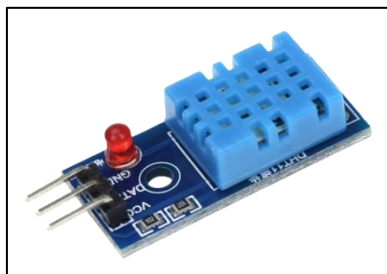


*Figure 3.3.2b: Analogue-to-Digital Converter (ADS1115)*

Instead of the Raspberry Pi's digital outputs of 1 and 0, we need to get the real reading (analogue) from the sensors for this project. As a result, the above component ADS1115 is necessary to accomplish this because, unlike Arduino, Raspberry Pi lacks integrated analogue pins for reading analogue sensor results.

The ADS1115 was chosen because it is a powerful, accurate analogue-to-digital converter with low power consumption that can be adjusted to the Raspberry Pi's 3.3V or 5V power supply. Additionally, it has 4 analogue input terminals, which are necessary for this project because it uses 4 analogue sensors. Moreover, because ADS1115 is high temperature tolerant, it may run for a longer period.

### **Temperature & Humidity Sensor**



*Figure 3.3.2c: Temperature & Humidity Sensor (DHT11)*

The DHT11 is necessary for this project to track the temperature and humidity of the air around the hydroponic plants and determine how the environment's exterior conditions affect the plants' ability to grow.

This sensor was chosen since it has an accuracy of  $\pm 1^{\circ}\text{C}$  and  $\pm 1\%$  and can measure temperature from  $0^{\circ}\text{C}$  to  $50^{\circ}\text{C}$  and humidity from 20% to 90%. The range that the sensor provides is more than sufficient to be incorporated into this project because the average indoor air temperature and humidity are roughly  $\pm 28^{\circ}\text{C}$  and  $\pm 66\%$ , respectively.

### **DIY MORE pH Sensor Kit**

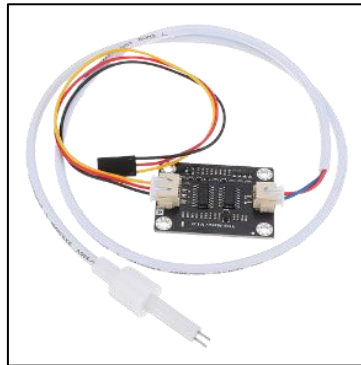


*Figure 3.3.2d: DIY MORE pH Sensor Kit*

One of the most important sensors for this project is the pH sensor. Since we will be dealing with a hydroponic system, it is crucial to understand the pH value of the nutrient solution because different plants have varied pH levels that are optimal for their growth.

The DIY MORE pH Sensor Kit was chosen for this project because, given the cost of the components, it offers a high precision output and uses a laboratory-grade probe to measure pH levels. The DIY MORE pH sensor interface circuit board also makes it simple to calibrate the sensor to fit project requirements.

### **EC Sensor (TDS Meter)**



*Figure 3.3.2e: EC Sensor (TDS Meter V1.0)*

The TDS reading obtained from an EC sensor provides information on the number of soluble solids that have dissolved in one liter of water, measured in milligrams. Generally, a higher TDS value indicates that more solids are present in the water, resulting in lower water purity. Because understanding the purity of the water is so critical for hydroponic systems in particular, this sensor is also one of the most important pieces that this project needs.

Since lettuce plants, which will be used in this project, require water with a TDS of between 568 and 840, the range of the TDS measurement on this EC sensor is between 0 and 1000 ppm, which meets the project's requirements. The TDS probe is additionally waterproof and can be submerged in water for an extended amount of time.

### **Water Temperature Sensor (DS18B20)**



*Figure 3.3.2f: Waterproof Water Temperature Sensor (DS18B20)*

The water temperature sensor is just as crucial a sensor as the pH and EC sensors, in part because changes in water temperature can have an impact on the water's pH and TDS levels,

which can affect plant development. So, using the water temperature sensor, we can keep an eye on and regulate the water's temperature.

This sensor was selected due to its wide temperature range of  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$  and precision of  $\pm 0.5^{\circ}\text{C}$ . In addition, the sensor's probe is waterproof, just like the probes for EC and pH sensors.

The cost breakdown of the used components and sensors is shown in the table 3.3.2 below:

*Table 3.3.2: Breakdown Cost of the Sensors and Components*

Item	Quantity	Price (RM)
Raspberry Pi	1	Borrow from UTP (free)
ADC	1	23.39
pH level Sensor	1	53.78
Water Temperature Sensor	1	16.89
DHT11	1	4.80
EC Sensor	1	21.89
Total (RM)	RM 120.75	

### 3.3.3 Machine Learning Algorithm Selection

Machine learning will also be included into the system to guarantee that it is more resilient and adaptable. The Convolutional Neural Network (CNN), a Deep Learning model, will be employed in this project, as mentioned in chapter 2 of this report. This is because it can forecast things like a leaf's health status with a better degree of accuracy than the other Deep Learning models, which makes it appropriate for this project, which involves cultivating leafy plants like lettuce. The reasons CNN is effective for forecasting the health of leafy plants are shown in the list below:

#### **Image classification:**

CNN are perfect for identifying patterns in plant leaves and categorizing them as good or sick as they are developed for image classification jobs.

**Feature extraction:**

CNN can extract intricate details from photos, including textures, forms, and patterns. This makes it simpler to spot the small variations in a plant's leaf's appearance that point to a change in health.

**Transfer learning:**

Large datasets may be used to train CNN, and the skills acquired for one job can be used to others that are comparable. This implies that a CNN may be improved to categories the health of a particular kind of leafy plant after being trained on a huge collection of plant photos.

**Robustness to noise:**

In the instance of predicting plant health, where the pictures may not always be of great quality, CNN are resilient to noise and fluctuations in image quality.

**Automated prediction:**

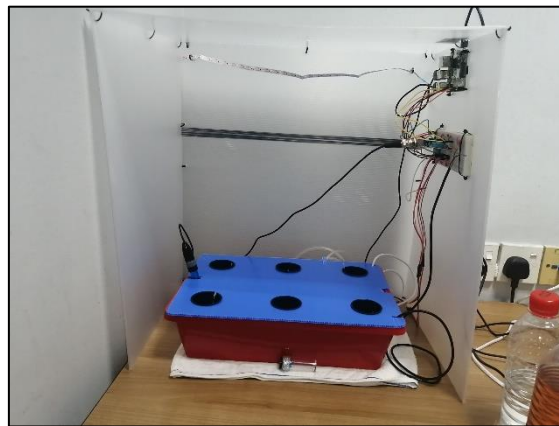
The process of forecasting plant health can be automated, making it quicker and more precise than manual techniques.

In conclusion, because CNN can categories pictures, extract complicated characteristics, transfer learning, manage noise, and automate the prediction process, they are effective at forecasting the health of leafy plants.

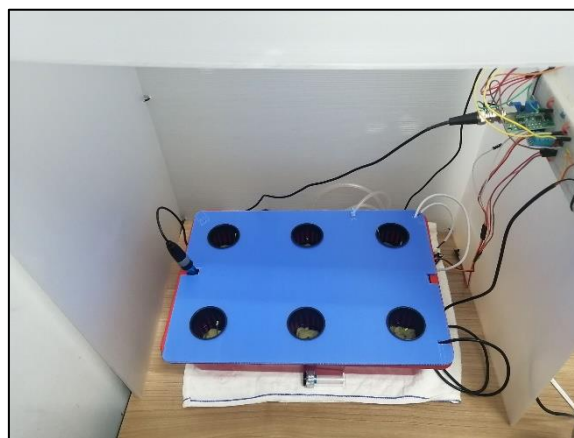
## 3.4 Phase 2: System Development and Design

### 3.4.1 DWC Hydroponic System Setup

The configuration of the DWC hydroponic system is shown in Figures 3.4.1. The IoT sensors and components are mounted on the hydroponic system using a frame made of corrugated plastic sheet. The block diagram in figure 3.4.2 below serves as the setup guide for the sensors and other components. Using cable ties, the raspberry pi and breadboard that are needed to manage and support the sensors are mounted on the right side of the frame. Cable ties are also used to secure the frame. Regarding the hydroponic system, the tray cover, constructed of the same material as the frame, has six holes intended to hold the net pot that will be used to sow the seeds. By squeezing through a tiny gap on the tray lid, the sensors and water pump were all placed within the tray. The LED lights are then hung at the frame's rear.



*Figure 3.4.1a: DWC Hydroponic System Setup*



*Figure 3.4.1b: DWC Hydroponic System Setup*

### 3.4.2 IoT System Setup

Below figure 3.4.2(1) is a simplified block diagram showing how the IoT system is set up for this project. **Three essential elements** make up the IoT System Setup. The **Raspberry Pi** microcontroller, the brain of the system, is by far the most important part because its primary function is to troubleshoot the data gathered from the sensors. The integration of all **IoT sensors** is the second step in tracking system internal and external environmental factors. The information from the system will then be gathered, saved, and used for visualization in the **Thingsboard Cloud**.

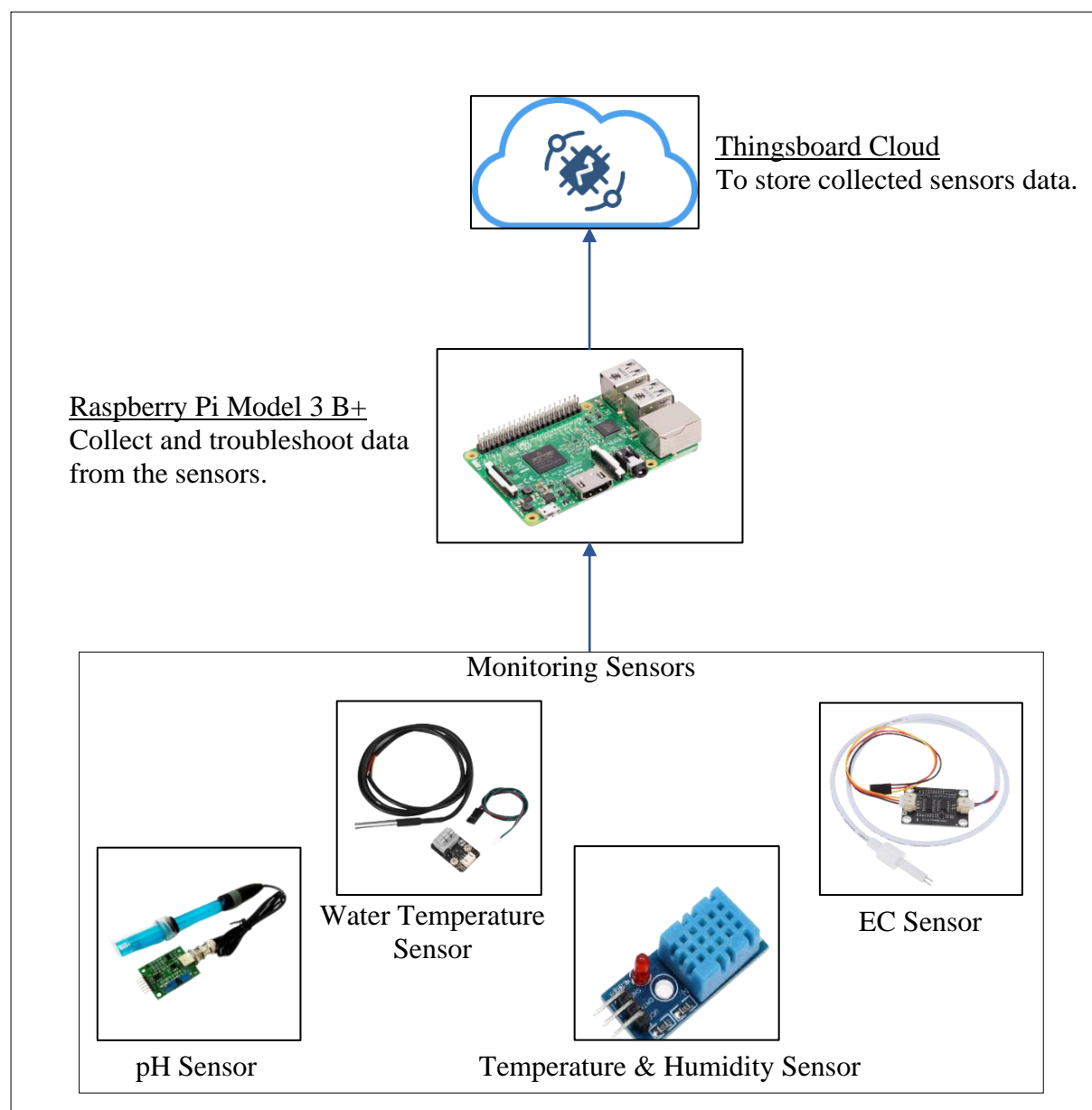
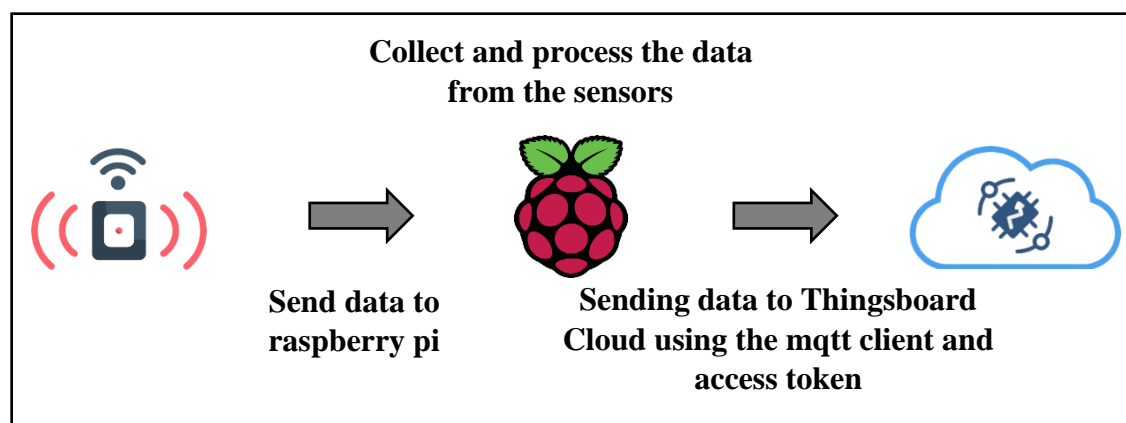


Figure 3.4.2a: Block Diagram of IoT System Setup

Since Raspberry Pi is and how the sensors work has been explained. The Thingsboard Cloud and how it's used in this project has not been elaborated. Firstly, ThingsBoard is an open-source IoT platform for data gathering, processing, visualization, and device management. ThingsBoard Cloud is a version of ThingsBoard that runs in the cloud, allowing customers to host their ThingsBoard instance there instead of on-site infrastructure and upkeep. For monitoring IoT devices and processing their data, it offers a scalable, secure, and highly available platform. Data visualization, real-time data streaming, a rule engine, and device administration are just a few of the services available through the ThingsBoard Cloud.

To transfer the Raspberry Pi's sensor data to the Thingsboard Cloud. Simply sign up for a free Thingsboard Cloud account and create a new device group. Next, simply obtain the device group's access token, which will be used to authenticate the Raspberry Pi when it sends data to the Thingsboard Cloud. Then, simply create a piece of code to gather data from the sensors and send it to the Thingsboard Cloud using the mqtt client libraries. The simple transmitting process flow diagram is shown in the following figure 3.4.2(2).



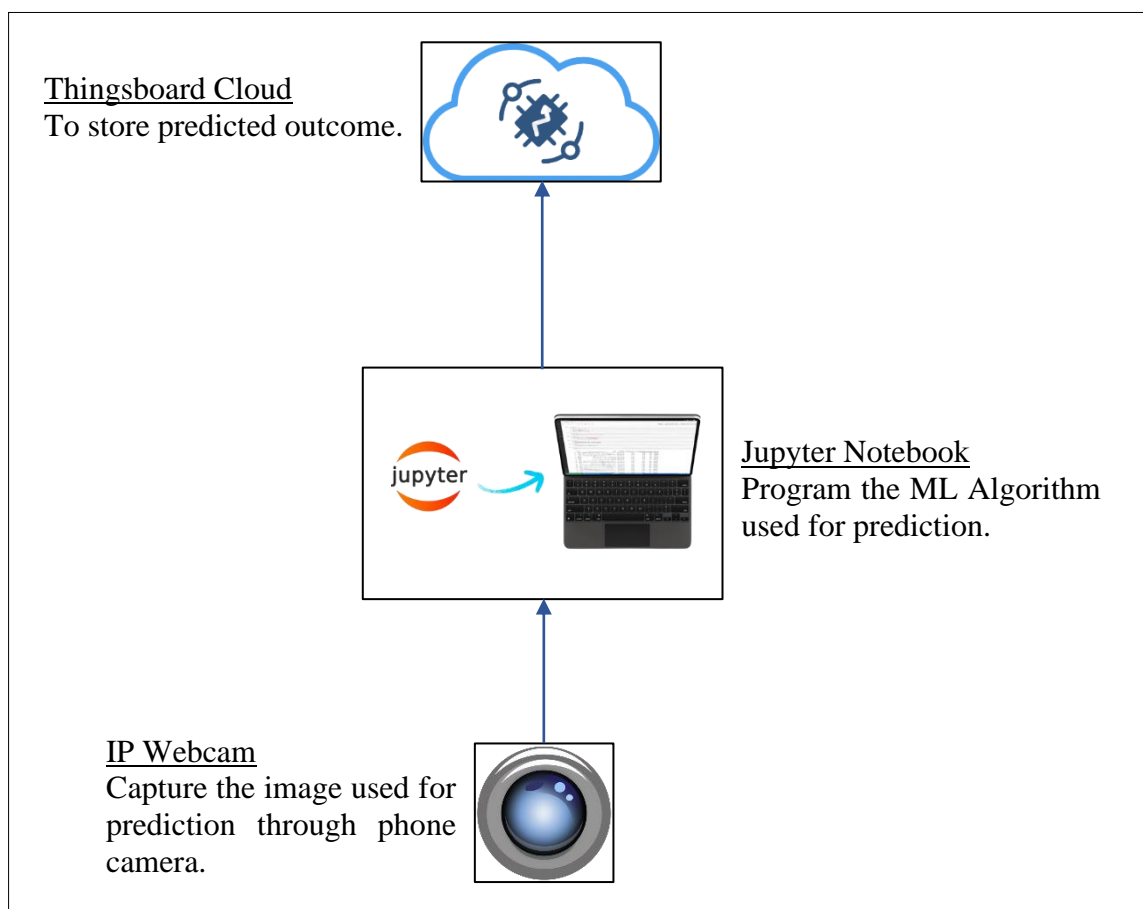
*Figure 3.4.2b: Transmission Process of Raspberry Pi to Thingsboard Cloud*



### 3.4.3 Deep Learning System Setup

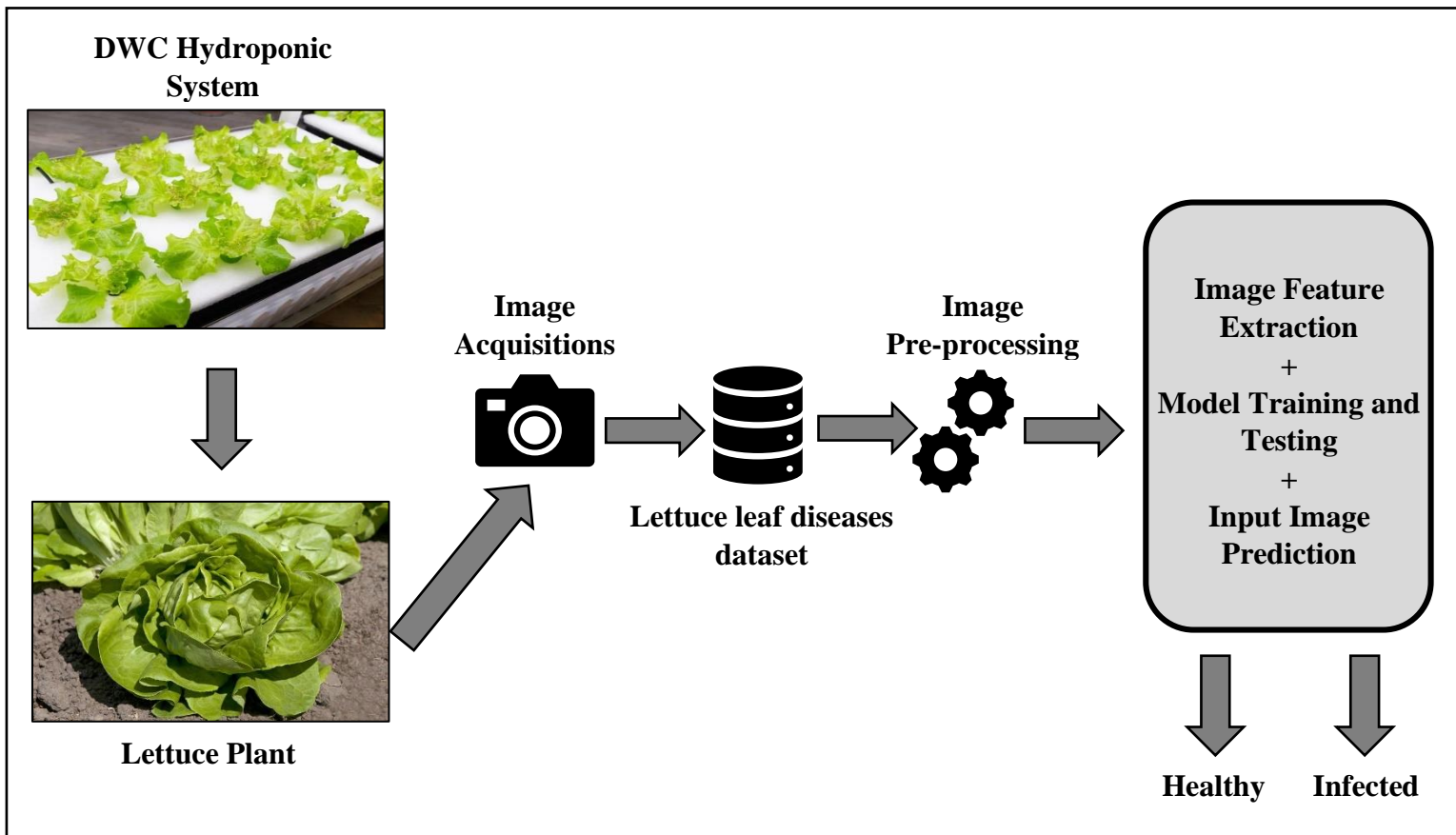
The below figure 3.4.3(1) is a simplified block diagram of how the project's machine learning system is set up. Three essential elements make up the machine learning system setup. The Jupyter Notebook application is the primary part since it gives users a place to programme and train the deep learning model. The second is the IP Webcam app, which allows us to utilize our smartphone cameras to capture a live image of the system's plant which will then later be used to feed into the trained deep learning model for health prediction. Same as the sensors the outcomes of the prediction made by the deep learning model will then be gathered and stored in the Thingsboard Cloud, where they will be afterwards used for visualization.

It was originally planned to utilize a Raspberry Pi for designing a deep learning model. However, it was discovered that the model was not powerful enough to handle the processing of the deep learning algorithm. Consequently, the decision was made to run and train the deep learning model on the personal laptop.



*Figure 3.4.3a: Block Diagram of Deep Learning System Setup*

The flow diagram for the project's chosen method of plant disease prediction is shown in the below figure 3.4.3(2). The suggested approach consists of four main stages: **Image Acquisitions**, **Image Features Extraction**, **Model Training and Testing** and **Input Image Prediction**.



*Figure 3.4.3b: Flow Diagram of Plant Health Prediction*

## Four Stages of Plant Health Prediction

### Image Acquisitions

At this stage, numerous images of lettuce plants are being collected, which encompass the lettuce condition. The datasets for the lettuce plants were obtained from GitHub, a platform that allows users to host and share their projects, code, and other programmed resources for their project. Additionally, some images were procured from Google Images. The datasets have been divided into two classifications, healthy and infected, comprising 100 images each, resulting in a total of 200 images. Equally distributing the images into both categories ensures that the trained model does not show bias towards any category.

### **Image Features Extraction**

All the characteristics of the lettuce images being gathered will now be retrieved from the images in this stage. The CNN model will be utilized for this project to extract features, which include: (1) Edges and lines, (2) Shapes and patterns, (3) Textures and gradients, (4) Objects and their positions, (5) Higher-level concepts, such as context and scene understanding. The image features that were retrieved will be used for training and testing purposes later.

### **Model Training and Testing**

In this stage, the characteristics that were retrieved during the extraction phase will be divided into two parts training and testing. The datasets will be split into 30% for testing and 70% for training. As this is a very common practice done by anyone doing machine learning. It is believed that larger train data sets can result in greater results prediction accuracy.

### **Input Image Prediction**

In this final stage, the prediction of the health status of the lettuce plant will be derived from the images gathered from the datasets, features extracted of the images, model training and testing of the retrieved features from the images using the proposed model.

### **3.5 Phase 3: System Testing and Assessing**

The final and most crucial phase involves testing and evaluating the accuracy of both the IoT sensors and the trained deep learning model to ensure the reliability of the outcomes. The following procedure will be used to test and evaluate both the IoT sensors and the deep learning model:

#### **Testing and Assessing of IoT Sensors**

Calibration and testing are not required for all sensors. However, the EC sensor and pH level sensor require calibration since they provide digital output that must be converted to analogue output using a formula. To develop an accurate formula for the sensors, an EC meter and a pH level meter were purchased. The results from these meters were compared to those obtained from the sensors, and the formula was adjusted until a close match was achieved.

#### **Testing and Assessing of Deep Learning Model**

To evaluate the precision of the CNN model, a series of procedures were followed. Firstly, the data was divided into training and validation (testing) sets, with the former comprising 70-80% of the data and the latter making up the remaining 20-30%. The model was then trained using the training data, fitting it to the data and updating the model parameters. Next, the model's performance was assessed using the validation results, comparing the model's predictions with the actual labels in the validation data. To calculate accuracy metrics, the number of accurate predictions was divided by the total number of predictions, with accuracy being the most used accuracy metric for classification tasks. Other metrics, such as precision, recall, and F1-score, may also be employed. Finally, to identify the best-performing model, the process was repeated using different model architectures or setups.

### 3.6 Gantt Chart

The FYP 1 and FYP 2 project's milestone is depicted in a Gantt chart in Figure 3.6.1 and Figure 3.6.2 respectively. The Gant Chart's main objective is to ensure that the project can be finished in accordance with the timeline.

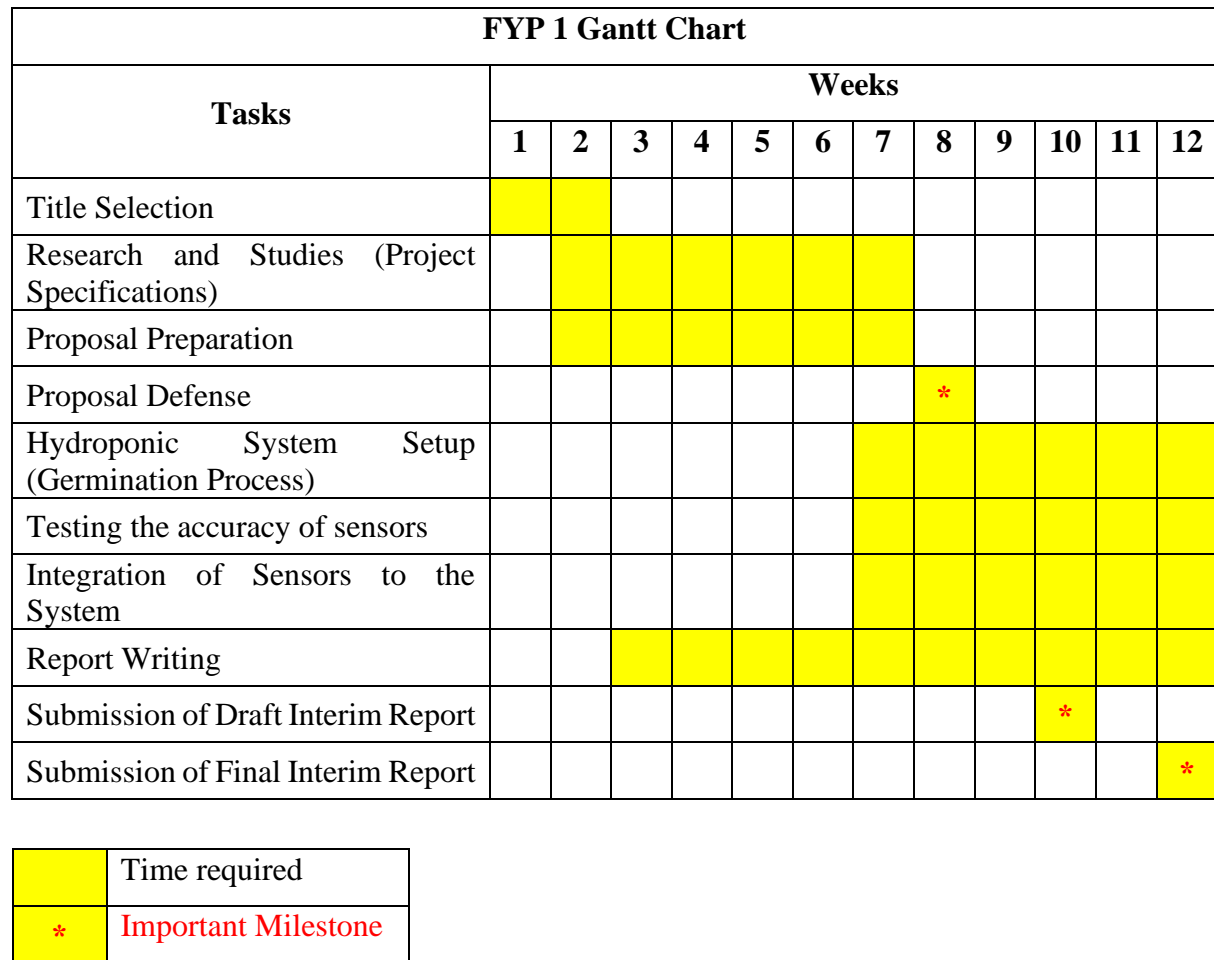


Figure 3.6.1: FYP 1 Gantt Chart

FYP 2 Gantt Chart													
Tasks	Weeks												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Research more on Machine Learning algorithm for plant health classification													
Selection of the classification algorithm to be used													
Building and training the selected algorithm model													
Testing the precision of the algorithm													
Integrate the tested algorithm to the system													
Data collection and send to a cloud database													
Visualize of the data on web-based dashboard													
FYP 2 Dissertation Preparation													
FYP 2 Dissertation Submission												*	
FYP 2 Viva Presentation													*

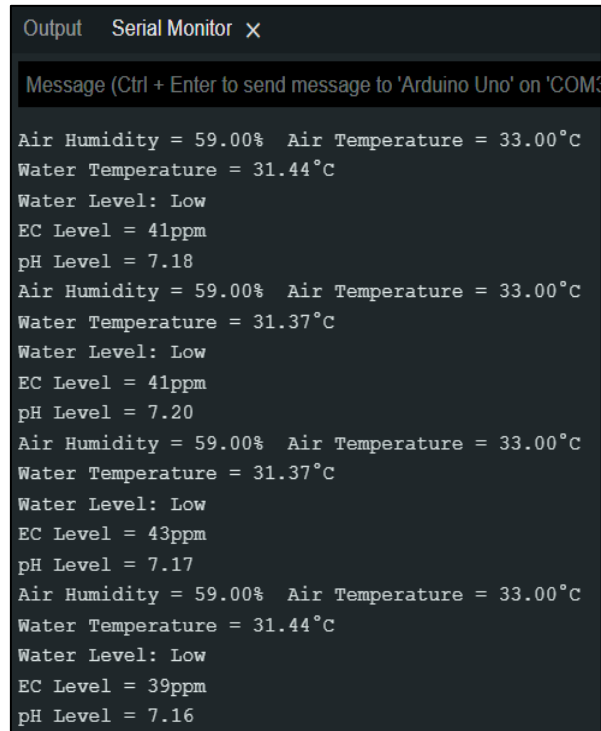
	Time required
*	Important Milestone

Figure 3.6.2: FYP 2 Gantt Chart

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Preliminary Results of the IoT Sensors (Arduino)

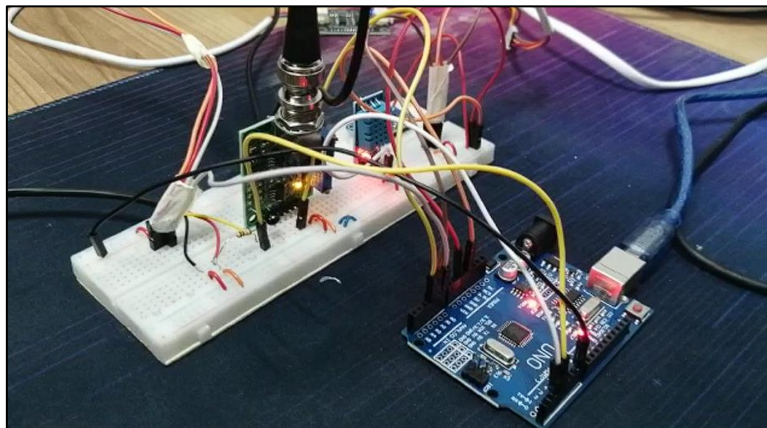


The screenshot shows a Serial Monitor window with the following text:

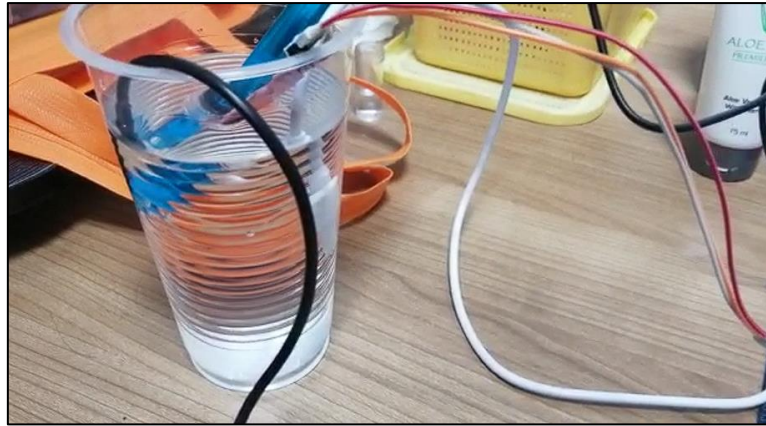
```
Output  Serial Monitor X
Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM3')

Air Humidity = 59.00%  Air Temperature = 33.00°C
Water Temperature = 31.44°C
Water Level: Low
EC Level = 41ppm
pH Level = 7.18
Air Humidity = 59.00%  Air Temperature = 33.00°C
Water Temperature = 31.37°C
Water Level: Low
EC Level = 41ppm
pH Level = 7.20
Air Humidity = 59.00%  Air Temperature = 33.00°C
Water Temperature = 31.37°C
Water Level: Low
EC Level = 43ppm
pH Level = 7.17
Air Humidity = 59.00%  Air Temperature = 33.00°C
Water Temperature = 31.44°C
Water Level: Low
EC Level = 39ppm
pH Level = 7.16
```

*Figure 4.1a: Preliminary Sensor's data using Arduino*



*Figure 4.1b: Circuitry of the IoT Sensors with Arduino (Video link to the draft setup of the IoT Sensors - [Prototype.mp4](#))*



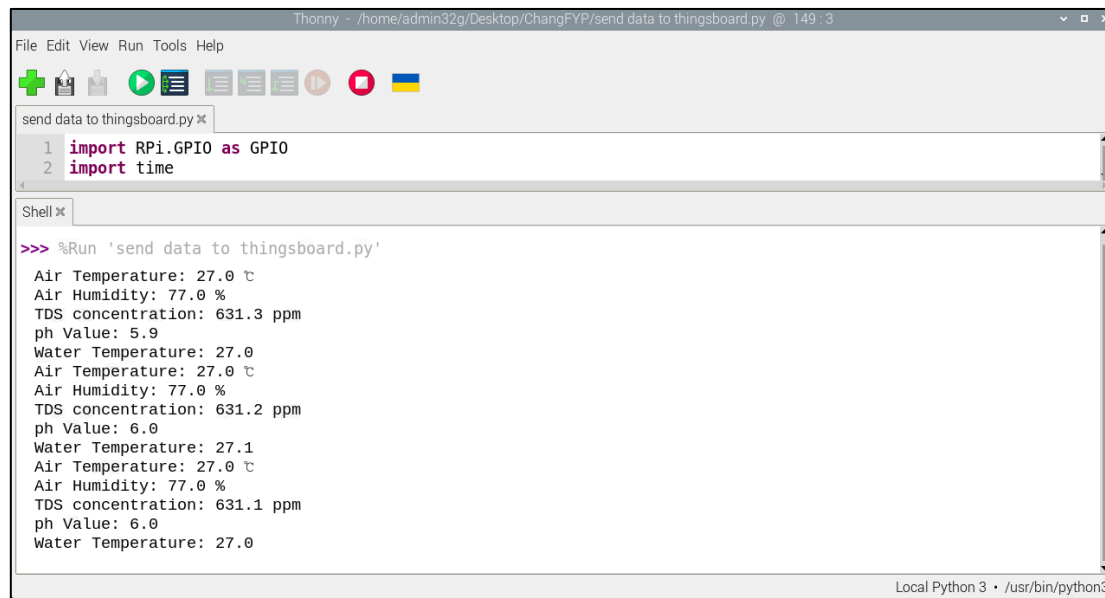
*Figure 4.1c: Cup of drinking water (Act as the hydroponic System)*

This section displays the initial findings from an Internet of Things sensor that uses an Arduino Uno microcontroller as its central processing unit. Since Arduino is simpler to use, it is being used to validate the sensor's accuracy before integrating it with the Raspberry Pi. The results of the sensors are shown in figure 4.1a. (pseudocode in [appendix I](#))

A cup of drinking water is used as an example for the hydroponic system for testing IoT sensors. The console of the Arduino Integrated Development Environment displays the sensor results. Drinking water should have a pH of 6.5 to 8.5, an EC of 50 to 150 ppm, and a temperature of 20 to 30 °C, respectively. According to the figure, the sensor's results were acquired with an accuracy of between 80% to 90%. To improve the accuracy of the sensors' results, some tuning of the algorithms and sensors is therefore necessary.



## 4.2 Results of the IoT Sensors (Raspberry Pi)



```
Thonny - /home/admin32g/Desktop/ChangFYP/send_data_to_thingsboard.py @ 149 : 3
File Edit View Run Tools Help
+ [Icons]
send_data_to_thingsboard.py x
1 import RPi.GPIO as GPIO
2 import time

Shell x
>>> %Run 'send_data_to_thingsboard.py'
Air Temperature: 27.0 °C
Air Humidity: 77.0 %
TDS concentration: 631.3 ppm
pH Value: 5.9
Water Temperature: 27.0 °C
Air Temperature: 27.0 °C
Air Humidity: 77.0 %
TDS concentration: 631.2 ppm
pH Value: 6.0
Water Temperature: 27.1 °C
Air Temperature: 27.0 °C
Air Humidity: 77.0 %
TDS concentration: 631.1 ppm
pH Value: 6.0
Water Temperature: 27.0 °C

Local Python 3 • /usr/bin/python3
```

*Figure 4.2a: Sensors' Data Output Using Raspberry Pi*

The integration of the IoT sensors with the Raspberry Pi may now be carried out after the IoT sensors have been tested and verified using the Arduino. An ADC is utilized to help convert some of the acquired Raspberry Pi digital data from the IoT sensors (pH and EC Sensor) into analogue data, as indicated in Chapter 3 section 3.3. This document's appendix ([appendix II](#)) contains the libraries and pseudocode code that were created and utilized to assist in getting the readings from IoT sensors. Figure 4.2a displays the output from IoT sensors gathered using the Thonny Python IDE, software that is included with the installation of the Raspberry Pi's operating system.

It is clear from the above figure that 5 outputs were anticipated. The DHT 11 sensor, which measures air temperature and humidity, is displayed in the first two outputs. The reading from the EC sensor, which is used to check the water's purity, is the next output. The pH sensor is next; it is clear from the name that this is a sensor used to determine the pH level. The reading from the sensor used to monitor the water temperature, the DSB18B20, comes last. The Things Board Cloud will then get all the sensor data for visualization. Two hours in the morning and two hours at night are spent monitoring the lettuce each day. The sensors' grouping interval is 6 seconds, therefore throughout the 2 hours of monitoring, the raspberry pi will send data to the Things Board Cloud every 6 seconds, resulting in a total of 1200 data requests.

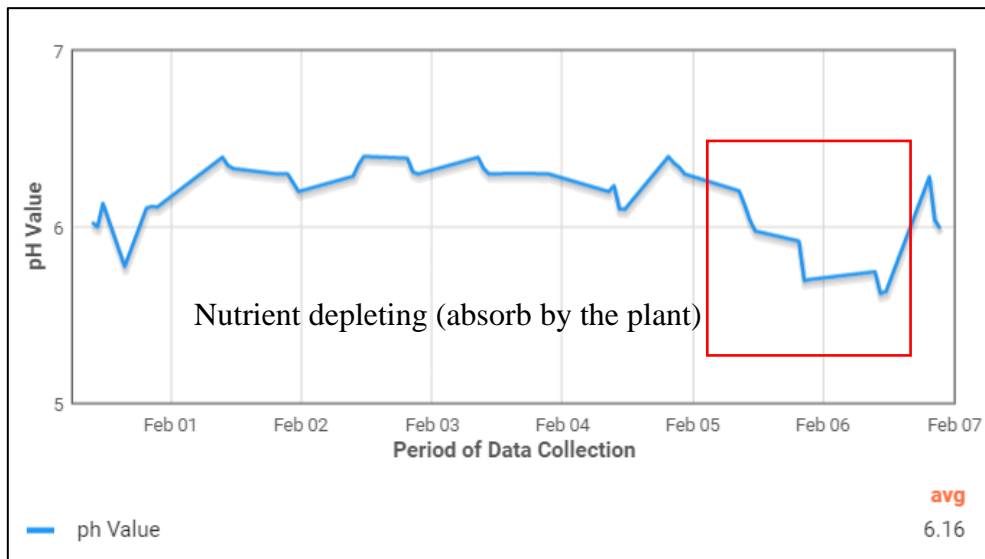


Figure 4.2b: Things Board Dashboard

The dashboard in figure 4.2b above was created using the Things Board Cloud, and the output data came from IoT sensors that were incorporated into the Raspberry Pi. In this report's Chapter 3 section 3.4, the method by which the data is sent is described. The data being obtained can be shown in a variety of ways; for example, a meter gauge widget on the dashboard above displays the air temperature and humidity. A widget that resembles a thermometer is used to display the water's temperature. A time series line chart is used to show the pH and EC sensor data since it is necessary to analyze the trend of these values. Also, the dashboard includes a table with all the sensor data that can be exported as an excel file and then utilized for data analysis.

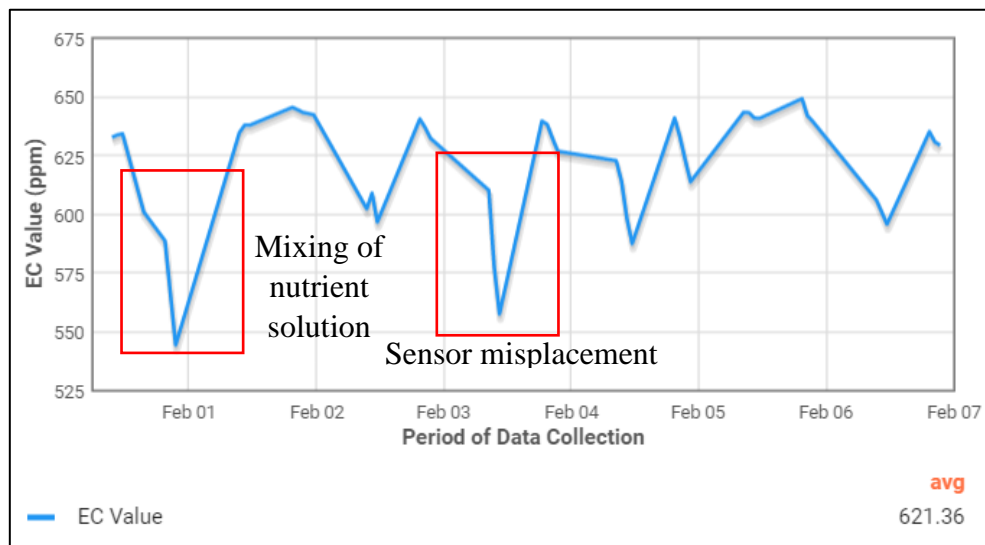
In addition to allowing users to change the visualization widgets, the Things Board Cloud Dashboard also provides features that let users view historical data from the dashboard as well as live data. As the user may obtain data based on whatever time and date they choose, this functionality is incredibly helpful.

*TimeSeries Line Chart of pH Value*



*Figure 4.2c: Timeseries Line Chart of pH Value*

*TimeSeries Line Chart of EC Value*



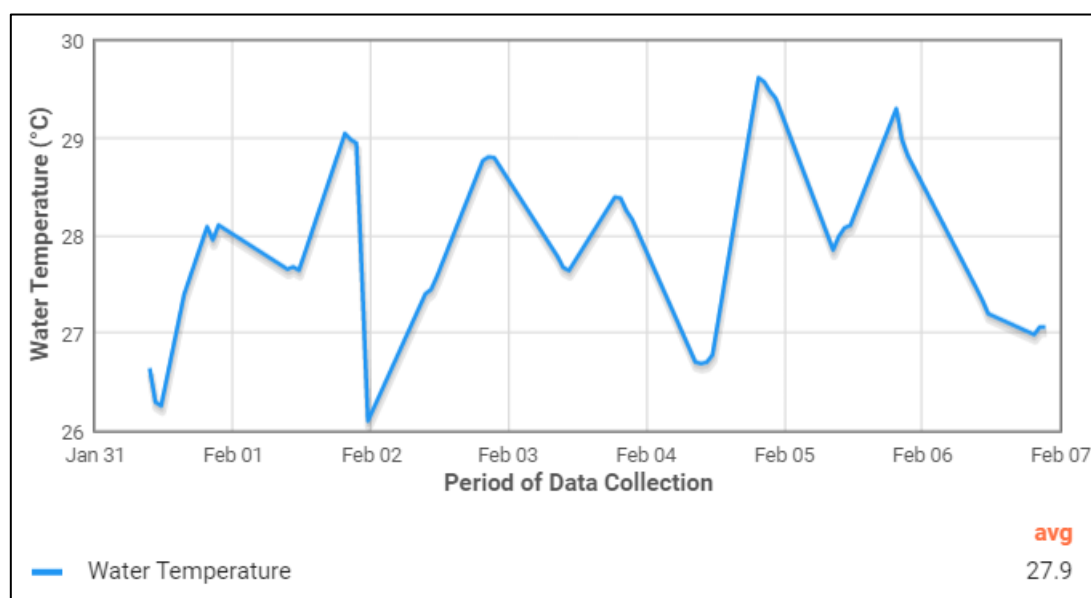
*Figure 4.2d: Timeseries Line Chart of EC Value*

The pH and EC values that were measured throughout the first seven days of the hydroponic lettuce plant's growing period are depicted in figures 4.2c and 4.2d, respectively. Figure 4.2c shows that the average pH over the course of seven days was 6.16, which is within the permissible range for hydroponically grown lettuce (5.5 – 6.5). As the data were captured using an ADC, there may have been some voltage/current loss, which is why there is some variation in the data. The loss has little impact on the outcomes because it is so tiny. About the substantial decline at day 6, this is because the nutrients in the solution are being progressively absorbed

by the lettuce plant and must be replaced. The nutritional solution needs to be replenished once a week, per the instructions that came with the hydroponic system that I purchased.

Similar to figure 4.3c, figure 4.2d displays that the pH averaged over seven days was 621.36 ppm, which is also in the acceptable range for lettuce cultivated hydroponically (560 – 840 ppm). The EC line chart only differs from the pH line chart in that the data fluctuates more dramatically there. The significant drop peak on day one because the nutrient solution is still being mixed in and takes some time for the sensor to respond to changes. As for the fall on day 4, it was brought on by the sensor's improper positioning.

***TimeSeries Line Chart of Water Temperature***



***Figure 4.2e: Timeseries Line Chart of Water Temperature***

In addition to pH and EC value, water temperature is another important factor in ensuring the plant grows as optimally as possible. The water temperature for the hydroponic lettuce plant's first seven days of growth is depicted in the above figure. The difference in temperature between day and night is the cause of the data's variation. The lettuce plant grew gradually over the course of the seven days, but unfortunately, after that time, it began to wilt. As a result, it is hypothesized that one of the causes may be because the water is excessively warm, which makes it harder for plants to absorb nutrients. The ideal water temperature for hydroponic plants is between 18 and 22 degrees Celsius. Furthermore, water temperature will also impact the readings of the pH and EC values.

### 4.3 Results of the CNN Model

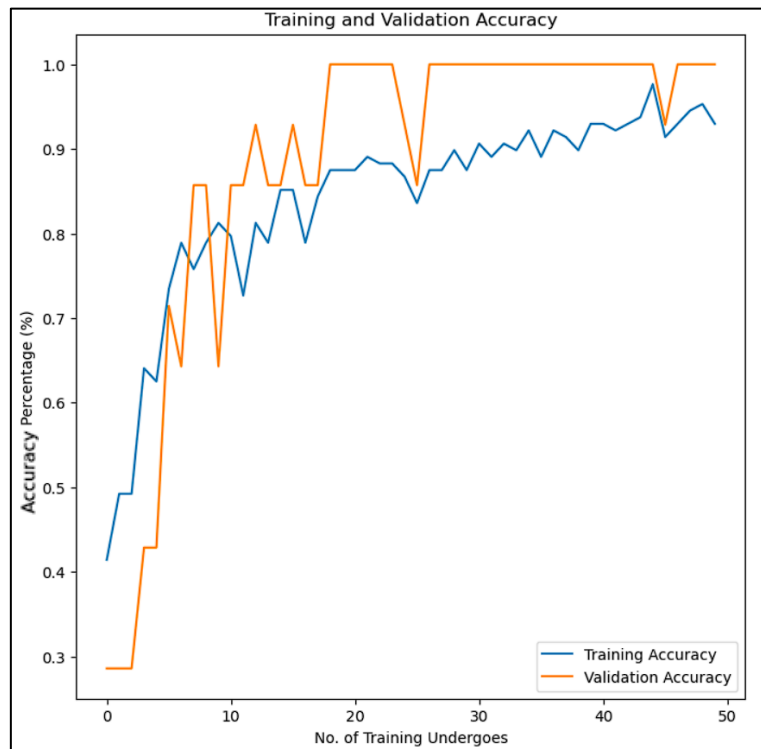


Figure 4.3a: Training and Validation Accuracy Chart

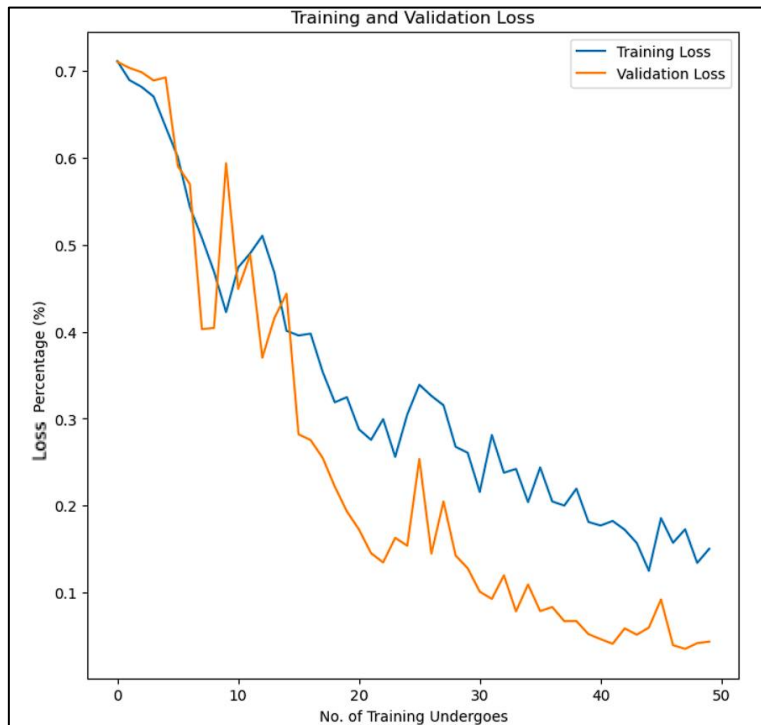


Figure 4.3b: Training and Validation Loss Chart

The training and validation accuracy and loss percentage of the CNN model developed for the hydroponic lettuce plant's health prediction are shown in the figure 4.3a and figure 4.3b above. As can be seen from the aforementioned figures, the accuracy of the model gradually increases as the number of times it is trained. Because the loss of the model decreases as accuracy rises, the relationship between the two is inverse. The model's output is better when the amount of losses is minimised because losses represent margin errors. Moreover, it is possible to see from the given graphs that the training and validation percentage fluctuates. The results of the model are overfitting when the validation percentage is higher than the training percentage, whereas the results are underfitting when the validation percentage is lower than the training percentage.

Overfitting and underfitting are common problems that might arise while training a Convolutional Neural Network (CNN) model for image categorization. Overfitting happens when a model is overly complicated and has too many parameters in comparison to the available training data. As a result, the model may become too specialised to the training data and perform badly on fresh, unknown data. In other words, the model is overfitting to the training data and does not generalise well to new data. Overfitting is frequently characterised by a high training accuracy but a low validation accuracy. Underfitting, on the other hand, typically happens when a model is too basic and lacks the capability to fully capture the underlying patterns in the training data. As a result, the model may not perform well on both the training and validation sets of data. In other words, the model is too generic and unable to represent the underlying complexity of the data. Low training accuracy and poor validation accuracy are frequent indicators of underfitting.

In order to obtain optimal performance, it is crucial to identify the ideal degree of model complexity and capacity in order to strike a balance between overfitting and underfitting. Frequently, this may be accomplished by varying the number of layers and neurons in each layer as well as by using regularisation techniques like dropout and weight decay. Also, adding additional instances for the model to learn from can assist minimise overfitting by increasing the amount of training data.

true label	Healthy	<b>TP</b> 22 (1.00)	<b>FN</b> 0 (0.00)
	Infected	<b>FP</b> 3 (0.12)	<b>TN</b> 21 (0.88)
		Healthy	Infected
		predicted label	

*Figure 4.3c: Confusion Matrix of the Model*

The model's confusion matrix is depicted in the above figure 4.3c. The matrix is constructed using the 2% test dataset, which was split from the larger dataset. The confusion matrix reveals that there are a total of 46 images of both healthy and infected lettuce. The matrix also reveals that 21 infected lettuce images are predicted to be infected with a chance of 88%, whereas 22 healthy lettuce images are predicted to be healthy with a chance of 100%. Moreover, 3 images of infected lettuce are predicted to be healthy with a chance of 12%. The accuracy and precision of the model may be determined using the information obtained from the confusion matrix and the formula below.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

As a result, it can be determined that the model's accuracy is 93.48% and its precision for making accurate predictions is 100%. In summary, some encouraging performance outcomes are being displayed by the trained CNN models.

The building blocks of the CNN model will be elaborated in this section, as well as some of the functions that are used in the code (pseudocode in [appendix III](#)). The Keras API from Tensorflow, a machine learning framework that is frequently used, is being used to create the CNN model. The CNN model was created to assist in classifying healthy and infected lettuce images.

The model is made to operate with images that are all the same size and have the same number of colours (for example, all black and white images that are 28 pixels by 28 pixels). The software takes the photos and preprocesses them (resizing them and making them all the same colour values). Then it employs a number of filters to scan the images for patterns and features. After that, the model performs a mathematical procedure to convert the output from those filters into a forecast of which of the two groups the image belongs to. The model "learns" by studying a large number of instances of images and their right labels, tweaking the filters and algorithms, and getting ever-closer to correctly predicting the contents of new, unobserved photos. In summary, the model is composed of several interconnected layers, such as filters, a flattening layer, and fully connected layers. Which of the two categories the image belongs to is predicted by the model's results. The CNN model must be trained on a collection of input data and associated labels after its architecture has been developed so that it may learn how to accurately classify incoming data. The model's weights and biases are updated during the training phase so that over time, it can produce better predictions. To accomplish this, we must outline the model's training procedure. This code specifically configures the optimizer, loss function, and training metrics:

- The optimizer acts as the training process's motor, modifying the model's weights and biases in response to mistakes it makes in label prediction. Because it adjusts the learning rate during training, the 'adam' optimizer is a well-liked option.
- The loss function calculates the discrepancy between the model's anticipated outputs and the actual labels found in the training set. Training's objective is to reduce this discrepancy, or 'loss,' as much as possible. The 'SparseCategoricalCrossentropy' loss function, which is appropriate for multi-class classification problems with integer labels, is employed in this instance. We instruct the model that its outputs should be probabilities of the various classes rather than raw numbers by setting 'from\_logits=False'.



- The performance of the model during training is assessed using metrics. The proportion of examples that are correctly classified is what the ‘accuracy’ metric measures. We can determine if the model is becoming better over time and whether we need to change the model or the training procedure to boost performance by keeping track of the accuracy during training.

By modifying its internal parameters (weights and biases) during training, the model discovers how to map the images to their proper labels (healthy or infected). By iterating over the training dataset repeatedly (the number of times is controlled by epochs) and changing the model's internal parameters each time, the ‘fit()’ method handles this procedure. The model does not examine every case in the dataset at once in order to maintain training efficiency. Instead, it examines a small number of samples (batch size-determined) at a time and modifies its internal settings in response to these examples. The training dataset's batches are all put through this process again.

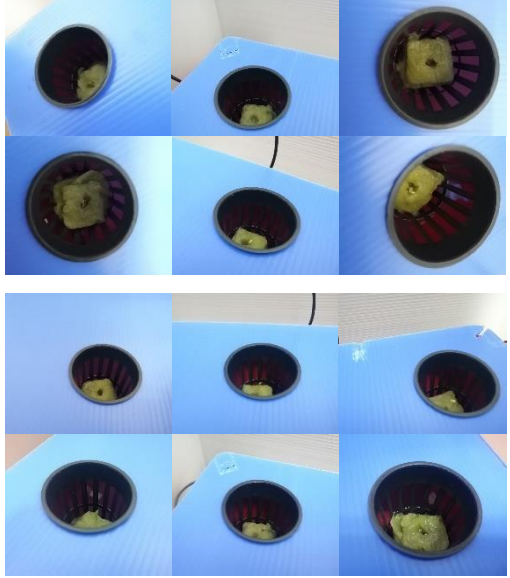
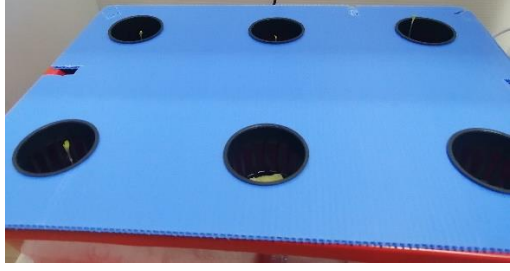
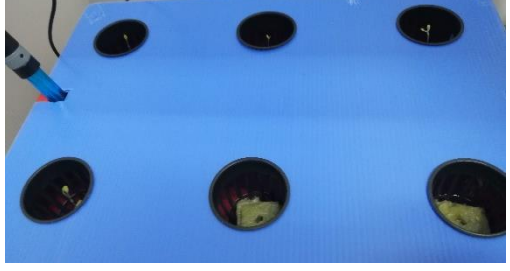
How much information is printed to the terminal during training is controlled by the ‘verbose’ parameter. If the parameter is set to 1, the procedure will print updates to the console following each epoch, informing us of the model's progress. The ‘validation\_data’ parameter is used to assess how well the model performs when applied to new data. On this validation dataset, the model's performance is assessed at each epoch, allowing us to identify overfitting and modify the model as necessary. Last but not least, the history object returned by the ‘fit()’ method includes details about the training procedure, including the model's accuracy and loss on the training and validation datasets at each epoch. With the use of this data, the model's performance can be examined and its accuracy can be increased.

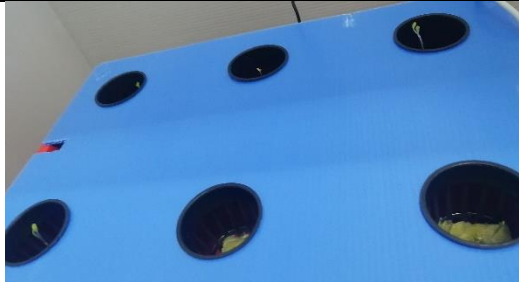
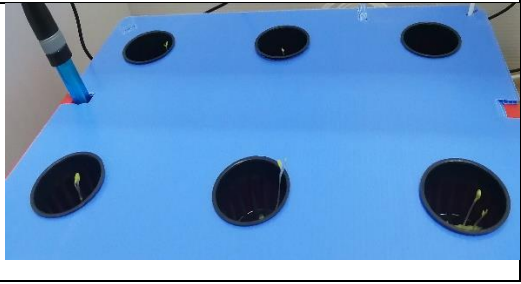
In a nutshell, the processed image array is then passed as an input to the ‘predict()’ method on the model object. A set of output predictions—a collection of probabilities for each potential class—is what this method generates. Then, the class with the highest probability will then be selected as the predicted class.

#### 4.4 Correlation between the sensors data and Machine Learning

This section of the paper will elaborate on the association between the sensor data and machine learning. What connection do they have? First, it is possible to forecast and estimate the precise pH and EC values that the nutrient solution should have using the sensor data that is being acquired. Also, the hydroponic lettuce photos based on data from various sensors can be utilised to gauge growth rate. The table 4.4 below displays the sensor readings and lettuce images that match to the readings as lettuce grows.

*Table 4.4a: Lettuce Growing Progress*

Days (Data of 1 week)	Sensors Reading (Average of the Day)	Outcomes (Images of the Plant)
1 to 3 (germination)	Air Humidity – 73 % Air Temperature – 28 °C Water Temperature – 28 °C EC Value – 622 ppm pH Value – 6.3	
4	Air Humidity – 77 % Air Temperature – 28 °C Water Temperature – 28 °C EC Value – 613 ppm pH Value – 6.3	
5	Air Humidity – 74 % Air Temperature – 28 °C Water Temperature – 28 °C EC Value – 616 ppm pH Value – 6.3	

6	Air Humidity – 73 % Air Temperature – 28 °C Water Temperature – 29 °C EC Value – 643 ppm pH Value – 6.0	
7	Air Humidity – 77 % Air Temperature – 27 °C Water Temperature – 27 °C EC Value – 617 ppm pH Value – 6.0	

As earlier mentioned, the sensor data can be utilised to make some straightforward predictions. As an illustration, consider the pH and EC values. We can perform the prediction using a basic machine learning model, such as linear regression, after gathering the sensor data and storing them in a CSV file. To put it simply, linear regression is a method of data analysis that predicts the value of unknowable data using the value of related and known data. It is used to determine whether the dependent and independent variables have a linear relationship (in this case, pH and EC).

So, how is the model being used? A simple Python programme is being used to apply the linear regression model (pseudocode in [appendix IV](#)). The programme first extracts the data from a CSV file that contains the pH and EC values. The two linear regression models will then be trained using the data. In contrast to the other model, which is used to forecast pH levels based on EC levels, the first one predicts EC levels based on pH levels. The models will then be utilised to create predictions about the EC and pH values after the training is complete. Lastly, the projected pH and EC values will be added as additional columns to the same CSV file.

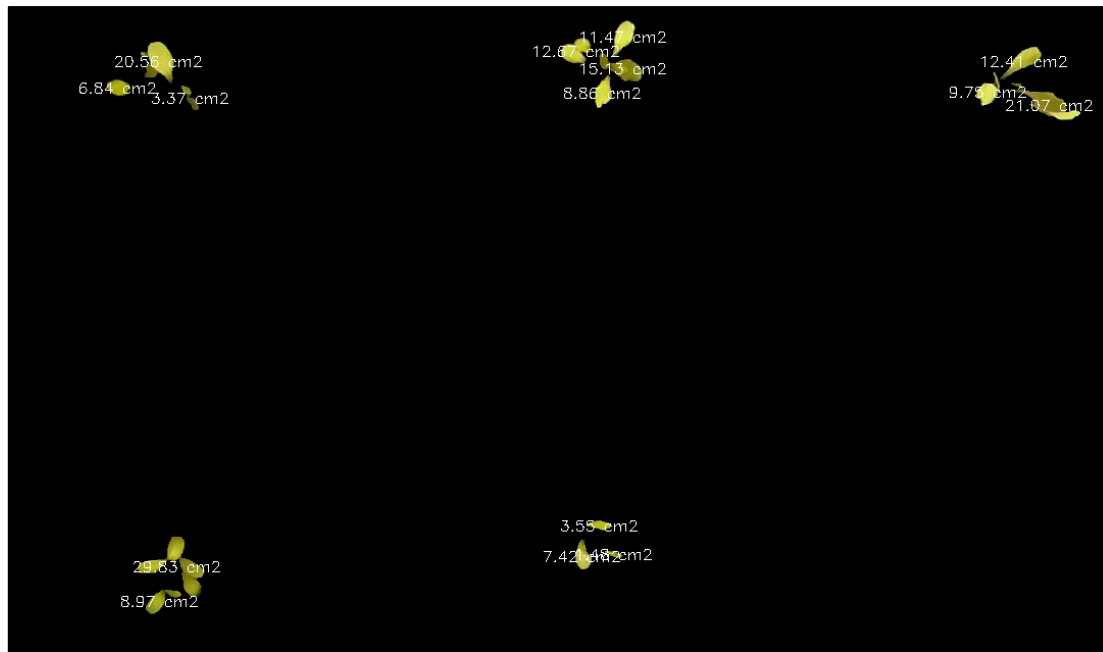
Overall, this programme employs linear regression to estimate the levels of two readings, EC and pH, based on one another to determine what are the exact porportion, and saves the predicted values to a CSV file for future examination.

EC Value	ph Value	Predicted EC	Predicted pH
632.4	6.1	726.4	5.8
632.2	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.3	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.2	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.2	6.1	726.4	5.8
632.3	6.1	726.4	5.8
632.3	6.1	726.4	5.8
632.1	6.1	726.4	5.7
632.3	6.1	726.4	5.8
632.1	6.1	726.4	5.7
632.2	6.1	726.4	5.8
632.3	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.5	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8
632.4	6.1	726.4	5.8

*Figure 4.4a: Example Output of the Linear Regression Model*

In addition to using sensor readings for prediction, the associated hydroponic lettuce images can be utilised to gauge the growth rate of the hydroponic lettuce. This can be used to observe how the sensor data influences the growth of the hydroponic lettuce. To achieve the given goal, a python programme (pseudocode in [appendix V](#)) employing OpenCV, a machine learning library, analyses the image of hydroponic lettuce leaves to identify and measure the size of the green and yellow segments. The programme converts the image to a new colour system known as HSV before identifying the spectrum of colours that correspond to green and yellow. It then generates a binary mask to highlight the green and yellow sections of the image, which it finally combines into a single mask.

The programme then blurs the image to decrease noise before creating a binary image with a thresholding technique to isolate the bits of the image that match to the lettuce leaves. The contours or edges of these isolated places are then identified and sorted by size. This data is used to create a contour around each lettuce leaf and compute its size in square centimetres. The original image is then displayed, with the contour and size information layered on top. It also displays an image with the background eliminated and only the lettuce leaves showing.



*Figure 4.4b: Example Output of the Segmented Images*

Overall, the programme is used to segment the image such that just the leaves of the hydroponic lettuce plant are visible. Measure the area of the leaves as well. The growth rate of the hydroponic lettuce plants can be calculated using the following formula:

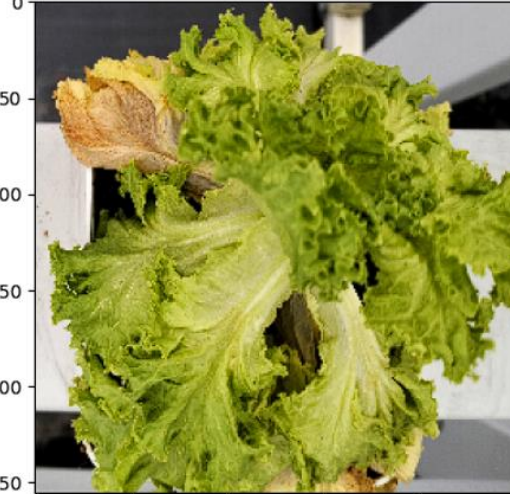
$$\text{Growth rate (\%)} = ((\text{Final leaf area} - \text{Initial leaf area}) / \text{Initial leaf area}) \times 100$$

Unfortunately, the growth rate of my hydroponic lettuce plants cannot be determined due to limited data collection and poor image quality. As a result, I'll demonstrate how to calculate the growth rate using a basic scenario. Assume the hydroponic lettuce plants have been monitored for 30 days. The lettuce leaf area was 20 centimetres square on day one and 50 centimetres square on day thirty. Using the aforementioned math equation  $[(50 - 20) / 20] \times 100 = 150$  percent, it can be concluded that the size of the lettuce leaf has increased by 150% throughout the growth period.

Similar to the growth rate discussed above, the CNN model may be used to the corresponding hydroponic lettuce images to determine the health of the hydroponic lettuce. In section 2.3.1 and 4.3 of this report, the use of a CNN model to predict the health condition of hydroponic lettuce was thoroughly discussed. It has been established that the CNN model, which is capable of classifying healthy and infected hydroponic lettuce leaves, can be used to predict the health condition of the lettuce. To achieve this, a python program, whose pseudocode is presented in

[appendix VI](#), is utilized. The CNN model, discussed in section 4.3, is loaded into the program. Subsequently, the hydroponic lettuce image is obtained using the IP camera application as described in section 3.4.3. The captured image is then resized to match the size of images used in the trained CNN model. The resized image is then used as input for the trained CNN model, which predicts the image's health condition. The predicted result and the confidence level of the input image can be obtained. An example of the CNN model's prediction results is presented in table 4.4b.

*Table 4.4b: Example Result of the CNN Model Prediction*

<p>Predicted Label: Healthy Confidence: 100.0 &lt;matplotlib.image.AxesImage at 0x217b939b130&gt;</p> 	<p>Predicted Label: Healthy Confidence: 99.88 &lt;matplotlib.image.AxesImage at 0x217b915c850&gt;</p> 
<p>Predicted Label: Infected Confidence: 74.7 &lt;matplotlib.image.AxesImage at 0x217b9403850&gt;</p> 	<p>Predicted Label: Infected Confidence: 66.41 &lt;matplotlib.image.AxesImage at 0x217bbd9d6d0&gt;</p> 



## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

One of the most well-known urban agriculture techniques, hydroponics, has been essential in addressing the problem of the growing global food demands. Particularly during a catastrophe, like the COVID-19 pandemic that the entire world is currently experiencing. Additionally, in addition to aiding in the resolution of the current crisis, it also works to prepare the world for future urbanization, which will lead to the gradual increase of food demand.

Hydroponics still faces several difficulties that are impeding its advancement despite the benefits it has brought. Because hydroponics requires a very sophisticated and organized system to be implemented, it is typically not the first choice when there are multiple options for agriculture techniques to be used. In addition, it is challenging for a novice without years of knowledge to assess the health of hydroponic plants. Therefore, the primary objective of this project is to build an IoT based Deep Water Culture Hydroponic Monitoring System that incorporated machine learning. The purpose of combining machine learning and IoT into one is to make a more robust and versatile system. The health state of the hydroponic plants can be forecasted using machine learning with the use of IoT sensors that help monitor and record the environmental condition of the hydroponic system. Machine learning also serves as human oversight to monitor the health of the hydroponic system.

Before concluding this report, it should be noted that there is always room for improvement, and the prototype of this project is no exception. Therefore, we would like to suggest some future improvements that could enhance the prototype's performance. Firstly, concerning the IoT sensors, more testing and experimentation should have been conducted to improve their accuracy, such as regular calibration, using higher quality sensors, validating the sensor data using historical data, monitoring, and analyzing the sensor's performance, and performing regular maintenance.

Regarding the machine learning algorithm, to ensure higher prediction accuracy, the number of images used for training and validation should be increased, such as using 1000 images for each category, and the quality of the images should be as clear as possible to achieve better results. Finally, performing more trial and error on the CNN architecture is necessary to obtain

the best possible results. In a nutshell, it is hoped that this project would be able to accomplish the following objectives: I) To create and build a cost-efficient, indoor-suitable prototype hydroponic monitoring system. II) To integrate the Internet of Things sensors on the system to measure the water's nutritional content, temperature, and air humidity & temperature. III) To integrate a machine learning-based system to predict the health and growth rate of the hydroponic plants. IV) To have a cloud database to house the data from IoT sensors and machine learning-based system, then display that database on an interactive dashboard. Likewise, this project will spread the word to more people about the benefits of hydroponics. Additionally, promote awareness of how IoT and machine learning applications may develop hydroponic technology.



## REFERENCES

- [1] Y. N. Chow, L. K. Lee, N. A. Zakaria, & K. Y. Foo, “New emerging hydroponic system,” In Symposium on Innovation and Creativity (*iMIT-SIC*), vol. 2, pp. 1-4, 2017.
- [2] R. Islam, & C. Siwar, “The analysis of urban agriculture development in Malaysia,” *Advances in Environmental Biology*, vol. 6, no. 3, pp. 1068-1078, 2021.
- [3] R. Murdad, M. Muhiddin, W. H. Osman, N. E. Tajidin, Z. Haida, A. Awang, & M. B. Jalloh, “Ensuring Urban Food Security in Malaysia during the COVID-19 Pandemic—Is Urban Farming the Answer A Review,” *Sustainability*, vol. 14, no. 7, pp. 4155, 2022.
- [4] M. H. Hamidon, S. Abd Aziz, T. Ahamed, & M. R. Mahadi, “Design and Development of Smart Vertical Garden System for Urban Agriculture Initiative in Malaysia,” *Jurnal Teknologi*, vol. 82, no. 1, 2020.
- [5] R. Lakshmanan, M. Djama, S. K. Selvaperumal, & R. Abdulla, “Automated smart hydroponics system using internet of things,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6389-6398, 2020.
- [6] A. Hadinata, “Internet of Things-based Hydroponic: Literature Review,” In *Journal of Physics: Conference Series*, vol. 2111, no. 1, Nov., pp. 012014, 2021.
- [7] Dudwadkar, Asawari, et al, “Automated hydroponics with remote monitoring and control using IoT,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 6, pp. 928-932, 2020.
- [8] N. Patil, S. Patil, A. Uttekar, & A. R. Suryawanshi, “Monitoring of Hydroponics System using IoT Technology,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 6, pp. 1455-1458, 2020.
- [9] V. Palande, A. Zaheer, & K. George, “Fully automated hydroponic system for indoor plant growth,” *Procedia Computer Science*, vol. 129, pp. 482-488, 2018.

- [10] V. Lakshmi, K. Avarna, D. N. Bhavani, & G. S. Spandana, "Hydroponic Farm Monitoring System Using IoT," *Iconic Research and Engineering Journals*, vol. 3, no. 10, pp. 257-261, 2020.
- [11] P. Koge, N. Deshmane, K. Chatwani, & P. S. Shetgar, "Development and Monitoring of Hydroponics using IoT," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 4876-4879, 2020.
- [12] J. K. Naik, R. Nayana, S. M. Bagade, S. Reddy, & H. S. Rohith, "IoT Based Automated Hydroponics System," *International Journal of Research in Engineering, Science and Management*, vol. 5, no. 6, pp. 309-310, 2022.
- [13] S. Tembe, S. Khan, & R. Acharekar, "IoT based Automated Hydroponics System," *International Journal of Scientific & Engineering Research*, vol. 9, no. 2, pp. 67-71, 2018.
- [14] M. Mehra, S. Saxena, S. Sankaranarayanan, R. J. Tom, & M. Veeramanikandan, "IoT based hydroponics system using Deep Neural Networks," *Computers and electronics in agriculture*, vol. 155, pp. 473-486, 2018.
- [15] K. P. Ferentinos, & L. D. Albright, "Predictive neural network modeling of pH and electrical conductivity in deep-trough hydroponics," *Transactions of the ASAE*, vol. 45, no. 6, 2007.
- [16] J. Pitakphongmetha, N. Boonnam, S. Wongkoon, T. Horanont, D. Somkiadcharoen, & J. Prapakornpilai, "Internet of things for planting in smart farm hydroponics style," In *2016 International Computer Science and Engineering Conference (ICSEC)*, Dec., pp. 1-5, 2016.
- [17] F. Ludwig, D. M. Fernandes, P. R. Mota, & R. L. V. Bôas, "Electrical conductivity and pH of the substrate solution in gerbera cultivars under fertigation," *Horticultura Brasileira*, vol. 31, pp. 356-360, 2013.

- [18] B. K. Hatuwal, A. Shakya, & B. Joshi, “Plant Leaf Disease Recognition Using Random Forest, KNN, SVM and CNN,” Polibits, vol. 62, pp. 13-19, 2020.
- [19] L. Tan, “CNN in Leaf Disease Classification,” Encyclopedia, Aug. 12, 2021. [Online]. Available: <https://encyclopedia.pub/entry/13083/>. [Accessed Nov. 12, 2022].
- [20] M. K. Gurucharan, “Basic CNN architecture: Explaining 5 layers of Convolutional Neural Network,” upGrad blog, Jul. 28, 2022. [Online]. Available: <https://www.upgrad.com/blog/basic-cnn-architecture/>. [Accessed Nov. 12, 2022].

# APPENDIX

## APPENDIX I: Pseudo Code on the Preliminary Results (Using Arduino IDE)

1. Initialize:
  - a. Set dht\_apin to 2
  - b. Set ONE\_WIRE\_BUS to 3
  - c. Set TdsSensorPin to A1
  - d. Set pHSensorPin to A2
  - e. Initialize DHT sensor
  - f. Initialize OneWire sensor
  - g. Initialize DallasTemperature sensor with OneWire sensor
  - h. Initialize GravityTDS sensor with TdsSensorPin
2. Loop:
  - a. Read air temperature and humidity using DHT sensor
  - b. Print air temperature and humidity
  - c. Read water temperature using DallasTemperature sensor
  - d. Print water temperature
  - e. Read electrical conductivity using GravityTDS sensor
  - f. Print electrical conductivity
  - g. Calculate pH value using pHSensor() function
  - h. Print pH value
  - i. Wait for 1 second
3. pHSensor:
  - a. buffer\_arr = empty array of size 10
  - b. For each i from 0 to 9:
  - c. buffer\_arr[i] = read analog value from pHSensorPin
  - d. Wait for 10 milliseconds
  - e. Sort buffer\_arr in ascending order
  - f. Return the 3rd value from buffer\_arr

## APPENDIX II: Pseudo Code on the Final Results (Using Thonny IDE in Raspberry Pi)

1. Import required libraries:
  - a. RPi.GPIO,
  - b. Time
  - c. Os
  - d. Adafruit\_DHT,
  - e. Glob
  - f. Adafruit\_ADS1x15
  - g. Board
  - h. Busio
  - i. adafruit\_ads1x15.ads1115
  - j. adafruit\_ads1x15.analog\_in
  - k. paho.mqtt.client
  - l. json
2. Define constants:
  - a. THINGSBOARD\_HOST = 'thingsboard.cloud'
  - b. ACCESS\_TOKEN = 'Oi4mExCcvGLAjrQOQnNL'
  - c. INTERVAL=6
  - d. DHT11\_SENSOR\_PIN = 26
  - e. i2c = busio.I2C(board.SCL, board.SDA)
  - f. ads = ADS.ADS1115(i2c, address=0x48)
  - g. tds\_channel = AnalogIn(ads, ADS.P1)
  - h. ph\_sensor\_channel = AnalogIn(ads, ADS.P2)
  - i. tds\_cf = 2000
  - j. vref = 3.3
3. Define functions:
  - a. read\_tds\_sensor(): reads TDS sensor voltage and returns TDS concentration
  - b. read\_ph\_sensor(): reads pH sensor voltage and returns pH value
  - c. read\_temp\_raw(): reads raw temperature data from the temperature sensor
  - d. read\_temp(): reads temperature from the temperature sensor and returns temperature in Celsius and Fahrenheit
4. Set up MQTT client:
  - a. Initialize MQTT client object
  - b. Set MQTT client username and password
  - c. Connect to the Thingsboard MQTT broker
  - d. Start the MQTT client loop
5. Try-except block:
  - a. Enter a while loop
    - i. Read air temperature and humidity using Adafruit\_DHT library and store in sensor\_data dictionary
    - ii. Read TDS concentration using read\_tds\_sensor() function and store in sensor\_data dictionary
    - iii. Read pH value using read\_ph\_sensor() function and store in sensor\_data dictionary
    - iv. Read water temperature using read\_temp() function and store in sensor\_data dictionary
    - v. Publish sensor data to Thingsboard MQTT broker using client.publish() method
    - vi. Calculate the time for the next reading and sleep until that time
  - b. If KeyboardInterrupt exception is raised, exit the while loop
6. Stop the MQTT client loop and disconnect the MQTT client.

### APPENDIX III: Pseudo Code on the Lettuce Health Prediction Model (Using CNN Model)

1. Import required libraries:
  - a. Numpy
  - b. Tensorflow
  - c. Models, layers from tensorflow.keras
  - d. Matplotlib.pyplot
  - e. Confusion\_matrix & accuracy\_score from sklearn.metrics
2. Define constants for image size, batch size, number of channels, and number of epochs.
3. Use `tf.keras.preprocessing.image_dataset_from_directory` to load the dataset from a directory and set class names.
4. Define a function `get_dataset_partitions_tf` to split the dataset into training, validation, and testing sets.
5. Apply caching, shuffling, and prefetching to each of the sets.
6. Define two data processing layers: `resize_and_rescale` and `data_augmentation`.
7. Define the input shape and number of classes for the model.
8. Define the model architecture using `layers.Conv2D`, `layers.MaxPooling2D`, `layers.Flatten`, `layers.Dense`, and `layers.Dropout`.
9. Compile the model using an optimizer, loss function, and metrics.
10. Train the model using `model.fit` and store the training history.
11. Plot the training and validation accuracy and loss using `matplotlib.pyplot`.
12. Define a function `predict` to predict the class of an image and its confidence.
13. Use `plt.imshow` and `predict` to plot a sample of images from the testing set with their actual and predicted classes and confidence.
14. Save the model using `model.save`.
15. Calculate the confusion matrix and accuracy score using `sklearn.metrics.confusion_matrix` and `sklearn.metrics.accuracy_score`.
16. Plot the confusion matrix using a helper function `plot_confusion_matrix`.

## APPENDIX IV: Pseudo Code on Simple Linear Regression to Predict pH and EC Value

1. Import required libraries:
  - a. Numpy
  - b. Pandas
  - c. LinearRegression from scikit-learn
2. Load the dataset from a .csv file named 'Sensorsdata.csv' using pandas read\_csv method.
3. Extract the EC and pH data from the dataset and store them in separate numpy arrays by reshaping the values to (-1, 1) using numpy's reshape method.
4. Create numpy arrays for EC and pH data with some values.
5. Create a linear regression model for EC based on pH using LinearRegression method from scikit-learn.
6. Fit the linear regression model on pH and EC data using the fit method.
7. Create a linear regression model for pH based on EC using LinearRegression method from scikit-learn.
8. Fit the linear regression model on EC and pH data using the fit method.
9. Test the EC model on new pH data by predicting EC values using predict method on the EC model and passing the ph\_data numpy array.
10. Print the predicted EC values.
11. Test the pH model on new EC data by predicting pH values using predict method on the pH model and passing the ec\_data numpy array.
12. Print the predicted pH values.
13. Predict EC and pH for the dataset by adding two new columns to the dataset for 'Predicted EC' and 'Predicted pH'.
14. Use predict method on the EC and pH models to predict values for the dataset using 'ph Value' and 'EC Value' columns respectively.
15. Save the updated dataset to the CSV file named 'Sensorsdata.csv' using the to\_csv method with the index parameter set to False.

## APPENDIX V: Pseudo Code on Segmentation and Size Measurement of Lettuce

1. Import required libraries:
  - a. Numpy
  - b. OpenCV
2. Set the path to the input image file
3. Load the image file using cv2.imread function
4. Convert the image from BGR to HSV color space using cv2.cvtColor function
5. Define the lower and upper bounds for the green color range in HSV space
6. Define the lower and upper bounds for the yellow color range in HSV space
7. Create masks for both the green and yellow color ranges using cv2.inRange function
8. Combine the green and yellow masks using cv2.bitwise\_or function
9. Apply a Gaussian blur to the mask using cv2.GaussianBlur function
10. Apply a binary threshold to the mask using cv2.threshold function
11. Find the contours in the thresholded mask using cv2.findContours function
12. Sort the contours in descending order of area using the sorted function and cv2.contourArea function
13. Create an empty mask using np.zeros\_like function
14. Loop through each contour in the sorted list of contours
15. Calculate the area and centroid of the contour using cv2.contourArea and cv2.moments functions
16. Convert the area from pixels to centimeters using a pixel\_conversion factor
17. Draw the contour on the empty mask using cv2.drawContours function
18. Add the area text to the mask and the original image using cv2.putText function
19. Use cv2.bitwise\_and function to apply the mask to the original image
20. Display the resulting masked image using cv2.imshow function
21. Wait for a key press using cv2.waitKey function
22. Close all windows using cv2.destroyAllWindows function



## APPENDIX VI: Pseudo Code on Using Captured Image to Predict the Lettuce Health Condition

1. Import required libraries
  - a. cv2
  - b. imutils
  - c. requests
  - d. numpy
  - e. matplotlib.pyplot
  - f. load\_model from tensorflow.keras.models
2. Define a list of class names for the classification task.
3. Define the URL for the camera feed.
4. Send a request to the camera feed URL and decode the image.
5. Resize the image to a width and height of 256 pixels using imutils.
6. Save the resized image to a file named "lettuce.jpg".
7. Load the pre-trained model from a file named "lettuce\_prediction\_model.h5".
8. Load the saved image from file "lettuce.jpg".
9. Resize the image to a size of (256, 256) using cv2.resize.
10. Expand the dimensions of the image to include a batch size of 1 using np.expand\_dims.
11. Use the loaded model to predict the class of the lettuce image.
12. Retrieve the predicted class label and the corresponding confidence level from the prediction array.
13. Convert the BGR image to RGB using cv2.cvtColor.
14. Display the RGB image using plt.imshow.
15. Print the predicted class label and the corresponding confidence level.