



TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

Ayala Blvd, Ermita, Manila

COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT



electronics@tup.edu.ph

**WSN-based Smart Farming through Soil Health and Ambience Condition Analysis
with Monitoring and Correction System**

by

Cabanding, Jonard Bryan S.

Caibigan, Jholrey C.

Indiano, Ricky M.

Moreno, John Nicholas B.

Virtucio, Angelika Mae C.

August 2023

APPROVAL SHEET

This project study entitled "**WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System**", has been prepared and submitted by the following proponents:

Cabanding, Jonard Bryan S.

Moreno, John Nicholas B.

Caibigan, Jholrey C.

Virtucio, Angelika Mae C.

Indiano, Ricky M.

In partial fulfillment of the requirements for the degree of **Bachelor in Science in Electronics Engineering** is hereby recommended for approval:

ENGR. GILFRED ALLEN M. MADRIGAL
Project Adviser

ENGR. JOHN PETER RAMOS
Panel Chair

ENGR. CHERRY G. PASCION
Panel Member

ENGR. EDGAR A. GALIDO
Panel Member

ENGR. LEJAN ALFRED ENRIQUEZ
Panel Member

ENGR. ROMEO L. JORDA JR.
Panel Member

Accepted and approved in partial fulfillment of the requirements for the degree of **Bachelor of Science in Electronics Engineering**.

ENGR. TIMOTHY M. AMADO
Head, ECE Department

ENGR. NILO M. ARAGO
Dean, College of Engineering

ABSTRACT

Most Filipino farmers rely on conventional ways of farming, which limit their yields and production efficiency. To aid farmers in improving the soil quality and to provide alternative farming solutions, the researchers proposed a Wireless Sensor Network system that will revolutionize farming through the integration of sensors, IOT, and automated systems for drip fertilization, irrigation, and ambient condition correction. The system is composed of two sensor nodes powered by a solar panel and a large-capacity battery, corrective devices, a gateway, and a mobile application for remote monitoring. Each sensor node measures ten parameters, namely light intensity, ambient temperature and humidity, soil's NPK, pH level, electrical conductivity, moisture, and temperature. The data acquired by the sensors will be processed under Fuzzy Logic Algorithm for correction, which will result in the automation of the feedback system to maintain optimal soil and ambient conditions. The data from sensors will be transmitted to the gateway using the ZigBee protocol and stored in the database, which can be viewed through a mobile application for remote monitoring with a notification system. The system is deployed in a greenhouse and monitors four crops, specifically lettuce, tomatoes, bell peppers, and pechay. The data gathered from the plant's growth was compared to the traditional planting and showed significant differences in terms of physical changes. The effectiveness of the correction system showed consistency in maintaining the required parameters within the threshold. In comparison to the traditional measuring tools, the system has 90% accuracy and has proven to be reliable in data transmission and effective in correcting parameters, which improve the plant's growth and yield.

ACKNOWLEDGEMENT

The proponents would like to express their deepest gratitude and appreciation to the individuals and organizations who have contributed to the successful completion of this study, SOIL-N-WAN. Without their invaluable support and contributions, this project would not have been possible.

First and foremost, the proponents would like to extend their sincere appreciation to their Adviser, **Engr. Gilfred Allen M. Madrigal**. His guidance, expertise, and mentorship have been instrumental in shaping the research. His invaluable insights, feedback, and continuous support have been invaluable throughout the entire journey.

Special thanks go to **RPA Lorenzo M. Jacinto**, whose expertise and honest evaluation of the system have been vital in refining and enhancing the functionalities of SOIL-N-WAN. His recommendations and valuable inputs have played a significant role in improving the overall quality of the system.

The proponents are grateful to **Brgy. Bagong Pag-Asa** for providing the opportunity to test and deploy SOIL-N-WAN in a favorable setup, crucial for successful implementation.

The **Green Thumb Organization** deserves special recognition as the primary user, whose collaboration, deployment assistance, and effective techniques greatly contributed to the development of SOIL-N-WAN.

The Panelists, whose expertise and valuable feedback during the evaluation process, have significantly improved the system and its operational efficiency. Their insights and

recommendations have been instrumental in enhancing the overall quality and functionality of SOIL-N-WAN.

Finally, the proponents deeply thank the **Almighty God** for His grace, wisdom, and blessings, which enabled the development of this technology and solution, serving as the cornerstone of their perseverance and success in completing this study.

The Proponents

TABLE OF CONTENTS

<i>Title</i>	i
<i>Approval Sheet</i>	ii
<i>Abstract</i>	iii
<i>Acknowledgement</i>	iv
<i>Table of Contents</i>	vi
<i>List of Figures</i>	xii
<i>List of Tables</i>	xvii
<i>Annexes</i>	xix
 CHAPTER 1: THE PROBLEM AND ITS SETTING		 1
1.1 Introduction	1
1.2 Background of the Study	3
1.3 Research Gap	5
1.4 Objectives	6
1.5 Significance of the Study	7
1.6 Scope and Limitation	8
1.7 Definition of Terms	9
 CHAPTER 2: REVIEW OF RELATED LITERATURE		 12

2.1 Smart Farming	12
2.2 Soil Nutrient/Content Conditions	13
2.2.1 Temperature	13
2.2.2 Moisture	15
2.2.3 pH Level	17
2.2.4 Electrical Conductivity	18
2.2.5 NPK Nutrient	19
2.3 Ambience Condition	22
2.3.1 Temperature	22
2.3.2 Humidity	23
2.3.3 Luminance	25
2.4 Soil Monitoring / Analysis	27
2.4.1 Most Common Sensors	27
2.5 Wireless Sensor Network	32
2.5.1 Zigbee	38
2.6 Data Analysis	41
2.6.1 Data Monitoring	41
2.6.2 Feedback and Control Process	43
2.6.3 Web Server	45
2.6.4 Notification System	46
2.7 Power Management	48
2.7.1 Battery	48
2.7.2 Buck Converter and Boost Converter	51

2.7.3 Solar Panel	52
CHAPTER 3: METHODOLOGY	55
3.1 Research Design	55
3.2 Research Process Flow	58
3.3 Development of the Sensor Nodes with an Efficient Power Management	59
3.3.1 Materials, Equipment, and Software to be Used	59
3.3.1.1 Hardware Components and Equipment	59
3.3.1.2 Software Application	66
3.3.2 Designing and Construction of the Sensor Nodes	66
3.3.3 Programming the Microcontroller	67
3.3.4 Sensor Calibration	67
3.3.5 Design of Power Management for Sensor Nodes	68
3.4 Development of Database and Gateway, and Establishment of Connection	68
3.4.1 Materials, Equipment, and Software to be Used	68
3.4.1.1 Hardware Components	68
3.4.1.2 Software to be Used	70
3.4.2 Designing and Construction of Gateway	71
3.4.3 Programming the Microprocessor as a Gateway	71
3.4.4 Establishing Secure Connection Between WSN and Database	72
3.5 Development of Mobile Application for Monitoring and Selection System	72
3.5.1 Software to be Used	72

3.5.2 Android Application Development	73
3.5.3 Programming the Option System	74
3.5.4 Establishing Connection Between Server and Mobile Application	74
3.6 Development of Automatic Correction System through Relays	75
3.6.1 Materials and Equipment Required	75
3.6.2 Design and Construction of the Feedback Mechanism	76
3.6.3 Programming the Calibration of the Correction Methodology	77
3.7 System Architecture	78
3.8 System Overall Flowchart	79
3.9 Testing the Device Overall Operation	80
3.9.1 Deployment of the WSN	80
3.9.2 Monitoring the System's Capabilities, Efficiency and Maintenance ...	80
3.9.3 Comparing the Accuracy of the System to other Monitoring System/Methods	81
3.9.4 Conducting User Acceptance Test	82
3.10 Statistical Analysis	83
3.11 Project Workplan	83
 CHAPTER 4: DATA AND RESULTS	 84
4.1 Project Technical Description	84
4.2 Project Structural Organization	85
4.2.1 Parts of the System	85

4.2.1.1 Sensor Nodes	85
4.2.1.2 Gateway	86
4.2.1.3 Power Management	86
4.2.1.4 Fuzzy Logic Algorithm-based Correction System	87
4.2.1.5 Monitoring Application	89
4.2.1.6 Deployment Area	92
4.3 Project Limitation and Capabilities	97
4.4 Project Evaluation	98
4.4.1 Sensor Accuracy	98
4.4.2 Power Management	107
4.4.2.1 Energy Consumption Table and Computation	107
4.4.2.2 Battery and Solar Panel Specification	109
4.4.2.3 Computation of Estimated Charge Time	110
4.4.2.4 Computation of Discharge Battery Percentage	110
4.4.2.5 Computation of Estimated Battery Life at Full Load	111
4.4.3 Calculated Values and Measured Parameters for Different Plants	112
4.4.3.1 Tomato	112
4.4.3.1.1 Measured Values for Tomato	114
4.4.3.1.2 Measured Values Parameters Measured for Tomato	115
4.4.3.2 Pechay	121
4.4.3.2.1 Measured Values for Pechay	122
4.4.3.2.2 Parameters Measured for Pechay	124

4.4.3.3 Pepper	130
4.4.3.3.1 Measured Values for Pepper Plant	132
4.4.3.3.2 Parameters Measured for Pepper	133
4.4.3.4 Lettuce	138
4.4.3.4.1 Measured Values for Lettuce Plant	140
4.4.3.4.1 Parameters Measured for Lettuce	141
CHAPTER 5: CONCLUSIONS, AND RECOMMENDATIONS	147
5.1 Summary	147
5.2 Conclusion	148
5.3 Recommendations	150

LIST OF FIGURES

Figure 1	Process Flow of the Study	58
Figure 2	Soil NPK Sensor	59
Figure 3	Soil pH Sensor	59
Figure 4	Soil Moisture Sensor Module	60
Figure 5	Waterproof Temperature Sensor DS18B20	60
Figure 6	3 n 1 Soil Sensor	61
Figure 7	Light Intensity Module GY-302 BH1750	61
Figure 8	Temperature and Humidity Sensor Module	62
Figure 9	XBee S2C Module	62
Figure 10	Solar Panel	63
Figure 11	Solar Charge Controller	63
Figure 12	Lead Acid Battery	64
Figure 13	Boost Converter Module (BQ25570)	64
Figure 14	Buck Converter (LM2596)	65
Figure 15	Arduino Mega 2560	65
Figure 16	Arduino IDE	66
Figure 17	Sensor Node Architecture Without Power Management Module	66
Figure 18	Power Management Architecture	68
Figure 19	Raspberry Pi	69
Figure 20	XBee S2C Module	69
Figure 21	Thonny IDE	70
Figure 22	Firebase Realtime Database	70

Figure 23	Gateway Architecture	71
Figure 24	MIT App Inventor	72
Figure 25	Data Transmission and Processing from Gateway to Monitoring Device	73
Figure 26	Relay	75
Figure 27	Water Pumps	75
Figure 28	Humidifier	76
Figure 29	Feedback Mechanism Architecture	77
Figure 30	System Architecture	78
Figure 31	System Flowchart	79
Figure 32	Sensor Node Design	85
Figure 33	Gateway	86
Figure 34	Power Management and Relays	87
Figure 35	Fuzzy Logic Designer using MATLAB	88
Figure 36	Rule Editor	88
Figure 37	Membership Function Editor	89
Figure 38	Rule Viewer of Fuzzy Logic using MATLAB	89
Figure 39	Graphical User Interface of Monitoring Application	90
Figure 40	Graphical View of the Parameters Via the Monitoring Application ...	91
Figure 41	Representation of Actual Device in Greenhouse	92
Figure 42	Deployment Area	92
Figure 43	Full Deployment of the System	93
Figure 44	Node 1 Used for Tomato Plant	93

Figure 45	Node 2 Used for Pechay Plant	93
Figure 46	Water Tanks for Correction System	94
Figure 47	Solar Panel and Power Management System	94
Figure 48	Uncontrolled Group of Plants for Conventional Farming	95
Figure 49	Node 1 Used for Pepper Plant	95
Figure 50	Node 2 Used for Lettuce Plant	96
Figure 51	Mr. Lorenzo Jacinto, a Registered Professional Agriculturist, Representing as the Expert for the Project Study	96
Figure 52	Deployment Site Visit with Room 2 Panel Chair (Engr. John Peter Ramos), Research Adviser (Engr. Gilfred Allen Madrigal), Research Extension Coordinator (Engr. Mark Melgrito).	97
Figure 53	Tomato's Height	114
Figure 54	Tomato's Number of Leaves	115
Figure 55	Tomato's Ambient Temperature	115
Figure 56	Tomato's Soil Electrical Conductivity	116
Figure 57	Tomato's Ambient Humidity	116
Figure 58	Tomato's Soil Moisture	117
Figure 59	Tomato's Perceived Light Intensity	117
Figure 60	Tomato's Soil Temperature	118
Figure 61	Tomato's Soil pH Level	119
Figure 62	Tomato's Soil Nitrogen	119
Figure 63	Tomato's Soil Phosphorus	120
Figure 64	Tomato's Soil Potassium	120

Figure 65	Pechay's Height	122
Figure 66	Pechay's Number of Leaves	123
Figure 67	Pechay's Average Fresh Body Weight	124
Figure 68	Pechay's Ambient Temperature	124
Figure 69	Pechay's Soil Electrical Conductivity	125
Figure 70	Pechay's Ambient Humidity	125
Figure 71	Pechay's Soil Moisture	126
Figure 72	Pechay's Perceived Light Intensity	126
Figure 73	Pechay's Soil Temperature	127
Figure 74	Pechay's Soil pH Level	127
Figure 75	Pechay's Soil Nitrogen	128
Figure 76	Pechay's Soil Phosphorus	129
Figure 77	Pechay's Soil Potassium	129
Figure 78	Pepper's Height	132
Figure 79	Pepper's Number of Leaves	132
Figure 80	Pepper's Number of Flowers	133
Figure 81	Pepper's Ambient Temperature	133
Figure 82	Pepper's Soil Electrical Conductivity	134
Figure 83	Pepper's Ambient Humidity	134
Figure 84	Pepper's Soil Moisture	135
Figure 85	Pepper's Perceived Light Intensity	135
Figure 86	Pepper's Soil Temperature	136
Figure 87	Pepper's Soil pH Level	136

Figure 88	Pepper's Soil Nitrogen	137
Figure 89	Pepper's Soil Phosphorus	137
Figure 90	Pepper's Soil Potassium	138
Figure 91	Lettuce's Height	140
Figure 92	Lettuce's Number of Leaves	140
Figure 93	Lettuce's Average Fresh Body Weight	141
Figure 94	Lettuce's Ambient Temperature	141
Figure 95	Lettuce's Soil Electrical Conductivity	142
Figure 96	Lettuce's Ambient Humidity	142
Figure 97	Lettuce's Soil Moisture	143
Figure 98	Lettuce's Perceived Light Intensity	143
Figure 99	Lettuce's Soil Temperature	144
Figure 100	Lettuce's Soil pH Level	144
Figure 101	Lettuce's Soil Nitrogen	145
Figure 102	Lettuce's Soil Phosphorus	145
Figure 103	Lettuce's Soil Potassium	146

LIST OF TABLES

Table 1	Summary of Related Studies in Smart Farming	13
Table 2	Summary of Related Studies in Soil Nutrient/Content Conditions, Temperature	15
Table 3	Summary of Related Studies in Soil Nutrient/Content Conditions, Moisture	17
Table 4	Summary of Related Studies in Soil Nutrient/Content Conditions, pH Level	18
Table 5	Summary of Related Studies in Soil Nutrient/Content Conditions, Electrical Conductivity	19
Table 6	Summary of Related Studies in Soil Nutrient/Content Conditions, NPK Nutrient	21
Table 7	Summary of Related Studies in Ambience Conditions, Temperature	23
Table 8	Summary of Related Studies in Ambience Conditions, Humidity	25
Table 9	Summary of Related Studies in Ambience Conditions, Luminance	26
Table 10	Summary of Related Studies in Soil Monitoring/Analysis, Most Common Sensors	31
Table 11	Summary of Related Studies in Wireless Sensor Network	36
Table 12	Summary of Related Studies in Wireless Sensor Network, Zigbee	41
Table 13	Summary of Related Studies in Data Analysis, Data Monitoring	43
Table 14	Summary of Related Studies in Data Analysis, Feedback and Control Process	45
Table 15	Summary of Related Studies in Data Analysis, Web Server	46

Table 16	Summary of Related Studies in Data Analysis, Notification System	47
Table 17	Summary of Related Studies in Power Management, Battery	50
Table 18	Summary of Related Studies in Power Management, Buck, and Boost Converter	52
Table 19	Summary of Related Studies in Power Management, Solar Panel	54
Table 20	Table of the Thresholds for the Crops to be Used	80
Table 21	Table of Accepted Percentage Error of Sensor Accuracy	82
Table 22	Ambient Temperature Sensor Accuracy	99
Table 23	Ambient Humidity Sensor Accuracy	100
Table 24	Soil Moisture Sensor Accuracy	101
Table 25	Soil Temperature Sensor Accuracy	102
Table 26	pH Level Sensor Accuracy	103
Table 27	Nitrogen Sensor Accuracy	104
Table 28	Phosphorus Sensor Accuracy	105
Table 29	Potassium Sensor Accuracy	106
Table 30	Energy Consumption	107
Table 31	Power Readings of the Device	109
Table 32	Calculated Values for Tomato	113
Table 33	Calculated Values for Pechay	122
Table 34	Calculated Values for Pepper	131
Table 35	Calculated Values for Lettuce	139

ANNEXES

ANNEX I	Schematic Diagram	158
ANNEX II	Component's Data Sheet	160
ANNEX III	Bill of Materials	171
ANNEX IV	Expert Evaluation Form	175
ANNEX V	User Acceptance Test	178
ANNEX VI	Soil-N-WAN Node Codes	187
ANNEX VII	Soil-N-WAN Gateway Codes	234
ANNEX VIII	Soil-N-WAN Monitoring Application Blocks and Codes	238
ANNEX IX	Progress Documentation	274
ANNEX X	Photos From Previous Defense	281
ANNEX XI	Turnitin Results	286
ANNEX XII	Certificate of Proofreading	292
ANNEX XIII	User's Manual	294
ANNEX XIV	Manual for Duplication of Prototype	297
ANNEX XV	Proponent's Information	308

CHAPTER 1

THE PROBLEM AND ITS SETTING

1.1 Introduction

The Philippines is one of the countries that relies heavily on agriculture due to its local resources and agricultural landscape. Because of this, it reflects the country's lifestyle in terms of cropping systems, barter systems, local and international trade, and localized labor [1]. The Philippines consists of more than 7,000 islands, which cover around 30 million hectares. According to the statistics of the World Bank, the agricultural land percentage of the country in 2018 was 41.7% [2]. Filipinos who work in agricultural sectors focus on farming, fisheries, forestry, and livestock production. In 2020, this sector generated a GVA (gross value added) of around 1.8 trillion Philippine pesos, which is about 10% of the country's GDP (gross domestic product) [3]. Due to these statistics, agriculture is a major contributor to the country's economic status. It serves as a work industry for Filipinos, a supplier of food for local and international markets, and the country's largest subsector.

Soil-based agriculture is the earliest form of strategy in cropping systems. Due to its nature, soil provides essential nutrients, oxygen, water, and support for a crop to grow. But without proper conditions, most crops are unable to grow in an area and require the absolute best conditions to produce their maximum quality. Based on the regions, some crops are so prevalent that multiple regions produce the same type of crop due to the status of an area [1]. According to the Nations Encyclopedia, out of thousands of crops, the main agricultural crops of the Philippines include rice, mango, coconut, corn, sugarcane, pineapple, abaca, coffee, and tobacco. Some crops considered next to these crops are garlic, onion, eggplant, kamote, calamansi, etc. [4].

Cultivating such crops leads to many parameters that need to be taken into consideration as they affect the quality and yield of the crop. These parameters include ambient conditions and soil content. This can be seen as some regions can grow a specific crop but not others. Thus, it results in ambience monitoring to determine whether the atmosphere is appropriate for the crops and the soil testing.

Most soil testing is done in a laboratory, which uses chemical reagents that create a chemical reaction in the soil samples to look for the specific properties that they are trying to achieve. This method proved to be accurate but takes time, especially since laboratories in the Philippines are in urban areas and most are far from rural agricultural areas. Thus, the Bureau of Soil and Water Management, the only soil resource agency in the country, provided Soil Test Kits where it tested soil samples using chemicals to gather NPK and pH levels, which proved to be useful and price-friendly for the farmers [5].

With these testing kits, people interpreting results badly is still prevalent since most people are not used to testing soil samples and human errors in testing can occur. Here comes the modern technologies where it lessens human intervention and error at a high rate which improves quality of life. There are multiple studies done but the technology does not reach the agricultural areas.

Even though it is the largest subsector, the growth of this industry is sluggish, and the major factors include industrialization and urbanization. Another factor that affects this sector is the vulnerability of the country to natural disasters. Innovation in technologies is greatly encouraged in this sector to improve the quality of life and the production of goods and services [3]. Soil may be renewable, but the formation takes a long time and may not meet certain nutrients for a crop to grow. Thus, the soil monitoring system spreads across the globe to secure the contents of the soil

that will be used for planting. But due to this advancement, the money generated from this sector takes years to compensate for the prices of the newer technologies. This study aims to provide a smart farming system where soil monitoring can happen simultaneously, providing a network and automatic correction system that would be beneficial for a rural area or wide-range farmlands.

1.2 Background of the Study

Soil health and nutrient content, as well as environmental conditions, are the primary factors that affect agricultural yields. To accurately detect these parameters, wireless sensor networks apt for smart soil health and ambience analysis with correction systems could be introduced.

The Soil NPK and Moisture analysis using WSN is capable of monitoring the actual real-time NPK, pH, and temperature of the soil. The system is equipped with nodes containing soil moisture, NPK, temperature, and pH sensors to detect the soil's macronutrients. An Arduino is incorporated into the system to control the sensors and communication module, Node MCU (ESP8266). Consequently, the data gathered by the sensors is uploaded to the cloud storage service Amazon Web Services through the ESP8266, allowing access on mobile devices equipped with the monitoring application. The software features a user-friendly monitoring system that is also capable of recommending the fertilizers required by the plants or suggesting suitable plants in the area depending on the sensor data. As a future enhancement, several sensors to measure soil micronutrients such as copper, iron, manganese, molybdenum, and zinc could be added to increase the system's reliability [6].

On the other hand, another study about real-time monitoring of Soil Nutrient Analysis using WSN analyzes the real-time NPK content of the soil through different sensors integrated on

a single node. The data collected by the sensors will be fed by the microcontroller to the n-Gate through Zigbee technology. The n-Gate is responsible for sending the acquired data to the IBM Bluemix Cloud, allowing remote monitoring of the system on mobile devices. On the software application, the present values of NPK on the soil will be displayed, and it will recommend the best crops appropriate to the reading or the best fertilizer to enhance the soil's health, depending on the crop planted. Additional sensors are being considered to enhance the system, including those that measure humidity, pH, and a number of micronutrients crucial to plant growth. Likewise, enhancements can be made to the monitoring system's algorithm in order to improve its ability to recommend crops and fertilizers to farmers [7].

An autonomous wireless sensor can be used to estimate the soil parameters to reduce human intervention in measuring the soil's condition. A sensor node is a device that will measure the data in the soil. With a total of 4 sensor nodes deployed, good accuracy is acquired in measuring phosphorus levels and other parameters. To achieve good accuracy and reliability, a sensor node is integrated with different modules, such as power management and wireless module technology. Dynamic Power Management is introduced to ensure low power consumption per node. Zigbee is used as the wireless module since it performs better in terms of power consumption with a 250-kbps data rate [8].

Soil moisture plays an important role in determining the condition of the plants in the weather. It is a critical parameter since plants can easily get stressed if there is climate change. By measuring soil moisture using a DHT22 sensor, such circumstances can be prevented. A machine learning-based prediction of the soil moisture is introduced together with the irrigation system to maintain the plant's health by measuring the soil's condition [9]. The system is integrated with the

ESP8266 Wi-Fi module and is used to transmit the data from the sensor nodes. The data transmitted is stored in the database (firebase) through a dummy server.

In soil quality management, a study is done using wireless sensor networks. Multiple sensors are used, such as the temperature sensor of the LM35, moisture, pH, and NPK sensors. For the WSN, a nRF transceiver with a GPS module embedded in a microcontroller is used for the sensor nodes, and a Raspberry Pi acts as a gateway node. Measuring the water content of the soil also aids in water conservation by allowing farmers to irrigate their fields only when necessary. pH sensors, coupled with NPK sensors, are used to maintain soil quality by advising fertilizer application [10].

1.3 Research Gap

For plants to grow to their utmost quality, it is ideal to have the best conditions for both their soil and ambience. Most of the cited research achieved soil health detection but failed to consider the crop's surroundings. There are multiple-parameter single sensors that have recently been developed, which these researchers failed to utilize. Another technology that is required and crucial for monitoring is a notification system with great accuracy and real-time speed. Aside from those, the primary problem encountered with WSN technology is the large power consumption of the components integrated into the sensor nodes and the communication module. Lastly, the usual feedback mechanism of the cited studies is the suggestion and recommendation of plants or fertilizers appropriate to the reading of the sensors.

Thus, the development of a WSN-enabled soil and ambience parameter monitoring system with feedback is introduced. Particularly, sensor nodes that are capable of detecting 10 parameters and transmitting the sensor data wirelessly to the monitoring system simultaneously while utilizing

a sustainable power source will be developed. The proposed monitoring software can be installed on Android devices, and it is capable of displaying the acquired data in numerical and graphical form. The application also features a notification system to inform the farmer or user whenever anomalies or sudden changes are observed. Additionally, the challenge for this system is the development of the automatic feedback system that triggers the relays for the minimal correction of the soil's NPK, pH, and ambient temperature and humidity depending on the user's crop choice.

1.4 Objectives

The main objective of the study is to develop a wireless sensor network enabled soil health and ambience condition monitoring system with correction feedback through IoT using ZigBee protocol.

Specifically, it aims:

1. To develop a sustainable and efficient sensor node that monitors soil's content such as temperature, pH level, electrical conductivity, moisture and NPK nutrient and the surrounding temperature, humidity, and luminance.
2. To establish WSN connection between the sensor nodes, the database and the gateway that is capable of receiving multiple simultaneous packets using ZigBee protocol.
3. To develop a monitoring system application that features algorithm, data processing, notification and plant selection for the monitoring system.
4. To design a closed-loop correction system that minimally corrects the soil's NPK and pH through fertilizer application and drip irrigation as well as the ambient temperature and humidity through relays.

5. To test and validate the device operation in different scenarios and conditions.

1.5 Significance of the Study

The study will focus on soil and ambience monitoring using a wireless sensor network with a sensor node that will help monitor the parameters. Since the Philippines primarily relies on its agricultural industry, research like this would be beneficial to improve crop yield production and sustain the needs of the country. The researchers aim to assist farmers in improving the soil quality so that they can produce better crops and harvests. Farmers in the Philippines will benefit from this research since it allows for more efficient monitoring. Our country is dependent on agriculture, and research like this aids production and labor efficiency.

Regarding the agricultural sectors and agendas in the country, this study under the Harmonized National Research and Development Agenda (HNRDA) specifically focuses on Section III Agriculture, Aquatic, and Natural Resources (AANR) as it tries to develop electronic innovations with a significant impact on agriculture. Sustainable Development Goals focus on helping the country in terms of developing, which varies based on the field of study. This study falls under Goal 9: Industries, Innovation, and Infrastructure since it also provides easier access and transparent knowledge about the versatility of ZigBee transmission and the possibilities that wireless sensor networks can offer.

With the main purpose of this study focusing on developing a Real-time Soil Monitoring System that can be used in future inventions, the study will benefit the body of knowledge as well as a varied but specific group of people. Soil nutrients are an important component in land-based agriculture, as they are required for plant growth. From plant varieties to fertilizer types, the content varies. It is easier to select a type of crop and fertilizer using a specific set of data, making

the procedure more efficient and cost-effective. This will assist the end-user in monitoring soil across a greenhouse. As a result, crop growth will be healthier and able to bear a larger mass. With the help of the device that the researcher will produce. It will be a great help for the farming community since monitoring soil conditions can offer the data needed to develop and implement strategies to maximize yield and productivity.

This research will provide more knowledge about wireless sensor network technology, network configuration, and data processing algorithms, thus enhancing and developing the proponents technical and analytical skills. Future researchers might use the concepts offered as a reference to test the validity, improve the project, or as a roadmap to perform a new study.

1.6 Scope and Limitation

The study aims to develop a soil monitoring system using WSN (Wireless Sensor Network) with various sensor nodes. The following sensors are to be used: a soil moisture sensor, a soil temperature sensor, an NPK sensor, a soil pH sensor, an electrical conductivity (EC) sensor, a light intensity sensor, ambient temperature, and humidity, all of which will be controlled by an Arduino Mega 2560. The soil monitoring should be carried out by this device, which has an integrated XBee module for the transmission of the data. It also includes a mobile application for real-time monitoring as well as a notification system. The system will have a feedback mechanism that will indicate what parameters should be corrected based on the user's input as it will be the basis for the required parameters for the specific crops, and it is integrated with an automatic correcting system that will correct the parameters using devices such as an exhaust fan for high temperature, a humidifier for low relative humidity, and a water pump for the macronutrients of the soil in terms of a fertigation system. The system will feature one sensor node for data gathering. The intended

deployment will be in a controlled area, specifically a greenhouse, as the system is integrated with correcting devices.

However, multiple limitations have been reached. The gateway device requires an internet connection to send the sensor-acquired data to the cloud to monitor the parameters. Some sensors are vulnerable when exposed to water, specifically drowning, which may affect the system as a whole. Likewise, some sensors are integrated into a single device, which leads to difficult maintenance and troubleshooting purposes. Due to the advancement of the sensors, the price is expensive, which led the researchers to restrict the number of nodes to one due to financial constraints. Also, the system is intended to utilize only commercially available products to correct the greenhouse's parameters. Particularly for the soil's content, water additive fertilizers would be used, and for ambience condition control, regular exhaust fans and humidifiers would be utilized. Lastly, the number of crops that the system is capable of monitoring is up to 4, such as lettuce, tomatoes, bell peppers, and pechay.

1.7 Definition of Terms

- WSN

The term "Wireless Sensor Network" (WSN) refers to a wireless network without any physical infrastructure that is deployed ad hoc using a large number of wireless sensors to monitor system, physical, or environmental factors. In WSN, sensor nodes with an integrated CPU are utilized to control and monitor a specific area's environment. They are linked to the Base Station, which serves as the WSN System's processing hub. The data can be processed, analyzed, stored, and mined using WSN.

- ZigBee

For the specific needs of low-cost, low-power wireless IoT data networks, Zigbee is a wireless technology that was developed as an open global market connectivity standard. The IEEE 802.15.4 physical board radio specification, together with unlicensed radio bands including 2.4 GHz, 900 MHz, and 868 MHz, are the foundations for the Zigbee connectivity standard.

- LiPO

Lithium Polymer batteries are rechargeable batteries that use polymer electrolytes, which makes them safer and more efficient to use. With the availability of varied sizes and shapes, the nature of these batteries makes them versatile and lightweight. As these lithium polymer batteries provide low discharge rates, their application mostly consists of small devices and gadgets that are safer to use.

- Packet Delivery Ratio (PDR)

The Packet Delivery Ratio can be achieved by dividing the total number of data packets that have reached their destinations by the total number of packets that have been delivered from sources. To put it another way, the packet delivery ratio is the proportion of packets transmitted from the source to those received at the destination.

- NPK

NPK, or Nitrogen, Phosphorus, and Potassium, are the macronutrients that plants require for survival and growth. Specifically, Nitrogen encourages the development of new plant tissues, Phosphorus promotes robust roots, fruiting, and flowering, and Potassium fortifies plants against disease and increases their resilience.

- Electrical Conductivity

The electrical conductivity of soil is a measurement of the capacity of soil water to conduct an electrical current. It influences crop yields, crop suitability, plant nutrient availability, and the activity of soil microorganisms, which in turn influence important soil processes such as the emission of greenhouse gases like nitrogen oxides, methane, and carbon dioxide.

CHAPTER 2

REVIEW OF RELATED LITERATURE

This chapter discusses all the related literature and studies regarding the topic WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System.

2.1 Smart Farming

In a greenhouse with WSN Arduino-based monitoring and control system, the application of WSN happened with the use of a wifi module ESP8266 connected to a microcontroller, specifically Arduino UNO and analog sensors. These transceivers are known to transmit packets in 2ms while consuming power of less than 1mW. Based on the findings, the wireless sensor network in a greenhouse would work better if the sensor nodes can monitor more parameters and additional hardware [11]. This would indicate that each node requires WiFi connectivity but with an advantage as these radio modules can be configured to read and transmit analog signals directly from the sensors and generate an equivalent of its digital data.

The study used Arduino Uno Nodes, which collect data from various locations in the greenhouse and communicate with the Raspberry-Pi node in real time in order to efficiently control light, soil moisture, humidity, and temperature inside a greenhouse by actuating lights or sliders, Water pump, Heater, and Fan according to the crops' essential conditions. The different values received from the various sensors and the status of the various devices are also displayed on an LCD. The results provided a better quality and productivity on crops that are controlled by the smart farming compared to the crops without controlled parameters [12].

Table 1. Summary of Related Studies in Smart Farming

Author	Year	Title	Relevant Findings	Relationship to the Study
R. Bhowmik and D. A. Pathak	2017	An Arduino Based WSN To Control and Monitor the Greenhouse Parameters	A sensor node that has more sensor parameters is much better in monitoring a wireless sensor network in a greenhouse.	Both studies use Arduino based microcontroller nodes.
Shinde, D., and Siddiqui, N.	2018	IOT Based Environment change Monitoring Controlling in Greenhouse using WSN.	A smart farming that provides controlled parameters that corrects the measured data will result in the quality of the crops.	The microcontroller used on the nodes are Arduino based and the correction system proposed have similarities.

2.2 Soil Nutrient/Content Conditions

2.2.1 Temperature

Soil temperatures average around 50 to 75 degrees Fahrenheit for bioactivity. Exceeding hot or cold temperatures can cause soil particles to crumble. As such, high temperatures cause evaporation on land areas, which makes them crack due to the lack of water. In a strong case, this can be related to plant growth. Warm temperatures help vegetation develop, mainly through water and nutrient uptake. Coldness can cause water to have a lower viscosity, which results in a slower process of photosynthesis. Focusing on

the chemical properties, warm soil proved to have more water-soluble phosphorus, which is better for plants. pH levels increase at high temperatures due to organic acid denaturation [13].

According to the study by Onwuka, soil moisture, aeration, and the availability of plant nutrients are all affected by soil temperature. Plant growth is also affected by soil temperature, which affects water and nutrient uptake as well as root and shoot growth. Water uptake decreases when the temperature decreases because of the increased viscosity and decreased absorption rate of water. The rate of photosynthesis is decreased by reduced water uptake. The viscosity of soil water and root nutrient transport are both affected by soil temperature. Low soil temperatures reduce plant nutrient uptake due to high soil water viscosity and low root nutrient transport activity, according to Grossnickle. Due to the increased metabolic activity of root cells and the production of lateral roots, an increase in soil temperature improves root growth. Reduced tissue nutrient concentrations and, as a result, reduced root growth are the results of low soil temperatures [14].

Table 2. Summary of Related Studies in Soil Nutrient/Content Conditions, Temperature

Author	Year	Title	Relevant Findings	Relationship to the Study
Eos	2020	Soil Temperature as A Factor Of Crops Development	The growth of crops is mostly influenced by the intake of water and nutrients in warm climates. Water can become less viscous when it is cold, which slows the photosynthesis process.	The one parameter that the study focuses is on soil temperature.
Brownmang onwuka	2018	Effects of soil temperature on some soil properties and plant growth.	The impact on soil temperature includes changes in soil aeration, soil moisture and the availability of plant nutrients. Temperature of the soil has an impact on plant growth, including root and stem development, nitrogen fixation, and water and water absorption.	Temperature is one of the soil parameters to be tested

2.2.2 Moisture

Determining the soil moisture helps to prevent the plant from having an unhealthy state or stress. Monitoring the condition of the soil may result in planning or deciding a strategy that can reduce the possibility of a plant being damaged due to uncertainty [15].

According to the study [16], researchers in agriculture must monitor soil moisture levels, particularly if they prefer to look at how much water plants take up during their development. To do so, a system is required that can store the collected data and alert the researcher to the percentage of monitored soil moisture. Depending on how much water volume is delivered to the soil, the rate of percentage increase in sensors and data readings about soil moisture conditions tends to increase faster. This implies that the sensor and monitoring system's performance is in line with the water infiltration process and is functioning properly. The rate of moisture deterioration in sensor data readings is slower since it is dependent on evaporation processes in the soil. The evaporation process accelerates the loss of soil moisture. The percentage reduction will be slower if the evaporation process is slow. This rate of percentage drop will necessitate a warning system that can alert the researcher if the observed soil is equal to or less than the parameter limit that the researcher is monitoring.

Table 3. Summary of Related Studies in Soil Nutrient/Content Conditions, Moisture

Author	Year	Title	Relevant Findings	Relationship to the Study
S. Unninayar	2015	Monitoring, Observations, and Remote Sensing – Global Dimensions	A strategy that can lessen the possibility of a plant being damaged by uncertainty may be planned or decided as a result of monitoring the soil's condition.	Soil moisture is a factor that the study aims to detect.
B. Siregar et al.	2018	Soil Moisture Monitoring System using Wireless Sensor Network	Particularly if they prefer to look at how much water plants take during their development, the proponents of agriculture must monitor the soil moisture levels.	The proposed study will analyze the soil moisture to efficiently monitor the crops.

2.2.3 pH Level

pH level affects the growth of the plants since it affects the availability of essential nutrients. The most common pH value of the soil ranges from 3.5 to 10 but it depends on the area because the typical pH value in rainfall areas is about 5 to 7 while in drier areas ranges into 6.5 to 9. The pH level of a soil can be classified as neutral (6.5 to 7.5), alkaline (over 7.5), and acidic (less than 6.5), it can also be classified as strongly acidic (less than 5.5) [17].

Soil pH became a parameter on monitoring because it is used to predict chemical elements contains in the soil. With this data, plant growth conditions can be used to determine the activities such as variety of plants, location, fertilizer to be used, and nutrient

content within the soil. The toxicity of elements can also be determined through soil pH because acidity serves as a major factor in chemical reactions [18].

Table 4. Summary of Related Studies in Soil Nutrient/Content Conditions, pH Level

Author	Year	Title	Relevant Findings	Relationship to the Study
Queensland.	2013	Soil pH Soil properties.	A pH level impacts the availability of vital nutrients; hence it affects how well plants grow.	The study will analyze the soil pH level.
R. Mylavaram et al.	2020	Soil pH and Electrical Conductivity: A County Extension Soil Laboratory Manual	As soil acidity plays a significant role in chemical reactions, it is possible to evaluate the toxicity of elements by its pH value.	One of the parameters that the study aims to detect is pH level.

2.2.4 Electrical Conductivity

Electrical Conductivity, or EC, became a soil parameter as it measured the capability of the sample's conductivity. Salinity is another term for the presence of salts or ions, which is directly proportional to the EC of the soil. But measuring the EC does not equate to the amount of salts present, for which deeper learning and testing are required and must be done in a laboratory [18].

The salinity of the soil can be measured using the soil electrical conductivity sensor. It is used to monitor the soil's content, such as its texture, nutrient availability and loss, and water capacity. Salts in the soil play a role in plant growth since they affect the balance of

the soil and water. Furthermore, cropping, irrigation, and land management can increase salt levels. EC does not provide a way to measure specific ions and salt compounds; instead, it focuses on the concentration of nitrates, potassium, ammonia, chloride, sulfate, and sodium [19].

Table 5. Summary of Related Studies in Soil Nutrient/Content Conditions, Electrical Conductivity

Author	Year	Title	Relevant Findings	Relationship to the Study
R. Mylavaram et al.	2020	Soil pH and Electrical Conductivity: A County Extension Soil Laboratory Manual	Another term for the measurement of the number of ions or salts, salinity, is directly correlated with the soil's EC.	The proposed system will have an algorithm to calculate the soil electrical conductivity.
USDA	1994	Inherent Factors Affecting Soil Phosphorus	The equilibrium of soil and water is affected by salts in the soil, which has an impact on plant growth. Additionally, salt levels can rise as a result of framing, irrigation, and soil conservation.	One of the soil parameters that the study aims to evaluate is the soil electrical conductivity

2.2.5 NPK Nutrient

The reason there are so many quality plants and crops coming from the farm is because of the soil they use. Soil also has nutrients such as nitrogen, phosphorus, and potassium, which are considered the main nutrients. Nitrogen is one of the factors affecting

plant growth since it is part of all the plant cells. For plants to be able to absorb the nutrient, nitrogen should be converted into nitrate when applied to the soil. Phosphorus is also a factor in plant growth. It is used to carry sunlight's energy to plants. Furthermore, potassium is one of the major elements in plant growth since it helps reduce the risk of a plant having a disease. It also plays an important role in maintaining the quality of the plants [20].

There are many elements required for plants and their development, but the main macronutrient that is measured is NPK. Nitrogen is used to link the formation of proteins, which can be seen as a major component of amino acids. This serves as a major nutrient, especially when plant tissues are renewed over time. Phosphorus also applies those services and shows a visible change of color on leaves as a warning of deficiency. With this, growth can slow down. Potassium also shows a visible mark in the form of spots of yellow or brown as a sign of deficiency. This element is important for the relationship between energy and water, which can be useful in colder temperatures and weather. Correcting these macronutrients can provide toxic tendencies to the plants, whether under evaluation or over application of certain fertilizers [21].

Correcting the nutrients in the soil, especially NPK, should have a positive impact on its quality and yield. The journal published by Kulumsa Agricultural Research Center was able to study the impact of combining organic and inorganic fertilizers on both soil fertility and productivity. The study demonstrates a significant difference between using a correction system for fertilizers and conventional methods. Furthermore, the application of inorganic fertilizer leads to an increase in root residues, indirectly contributing to the buildup of organic matter. Inorganic fertilizers are beneficial for promoting the rapid

growth of plants due to the presence of readily water-soluble nutrients [22]. Hence, utilizing the system should have a significant impact on the growth rate of crops.

Table 6. Summary of Related Studies in Soil Nutrient/Content Conditions, NPK Nutrient

Author	Year	Title	Relevant Findings	Relationship to the Study
Department of Primary Industries	2017	Plant nutrients in the soil	Since nitrogen is a part of every plant cell, it is one of the factors affecting plant growth. Phosphorus is used to carry sunlight's energy to plants. Furthermore, potassium minimizes plant disease risk.	NPK are the soil macronutrients that the study aims to detect and control
R. L. Mahler	2004	Nutrients plants require for Growth	Nitrogen links amino acids to build proteins. Phosphorus also changes leaf color to indicate a deficit. Potassium leaves yellow or brown patches.	The study will utilize NPK sensor to analyze soil health and it is one of the parameters to be controlled
Journal of Ecology & Natural Resources	2021	The Effect of Integrated Organic and Inorganic Fertilizer on soil fertility and productivity	The use of inorganic fertilizer has an impact on the growth rate of the crops.	NPK are the soil nutrients that the study aims to correct by using organic and inorganic nutrients.

2.3 Ambience Conditions

2.3.1 Temperature

Ambient temperature is one of the driving factors in a plant's processes of photosynthesis, transpiration, and respiration, which are essential to its growth and development. As stated by [23], a rise in temperature accelerates those processes, while a decrease in temperature slows down those transitions. Temperature also influences the transition from vegetative to reproductive growth when it is combined with day length.

But at high temperatures, pollination of plants is delicate, and it decreases the production of grains and seeds, leading to a lower yield. Overexposure to extreme temperatures also reduces the grain-filling period of crops. Aside from that, if the crop is exposed to high temperatures throughout its growing stage, it would grow faster but its leaves wouldn't prosper. As an example, a corn plant exposed to extreme heat while at the grain-filling stage would produce 80–90 percent less [24].

Some of the effects of extremely low or high temperatures on crops include stress, slender appearance, damage to the foliage, or loss of the entire foliage. As stated by [25], plants prefer cool nighttime temperatures to grow rather than high temperatures.

Table 7. Summary of Related Studies in Ambience Conditions, Temperature

Author	Year	Title	Relevant Findings	Relationship to the Study
G. E. T. Involved and C. Us	2022	Environmental Factors Affecting Plant Growth	Photosynthesis, transpiration, and respiration are all processes that are enhanced by temperature increases, while they are slowed down by temperature decreases.	The study aims to detect and correct the ambient temperature depending on the crop.
J. L. Hatfield and J. H. Prueger	2015	Temperature extremes: Effect on plant growth and development	High temperatures make plant pollination delicate, which in turn reduces the yield of grains and seeds.	Temperature of the surrounding is one of the parameters that the study will evaluate and correct.
Texas A&M Agrilife Extension		Light, Temperature and Humidity	Stress, a narrow appearance, damage to the foliage, or loss of the entire foliage are some effects of high temperatures on crops.	One of the focuses of the proposed system is the ambient temperature detection and control

2.3.2 Humidity

Humidity is defined as the amount of water vapor or moisture in the air, and it is a factor to consider in plant growth and development [25]. Generally, transpiration water loss and stomatal opening are the direct impacts of atmospheric moisture on crops. Low air humidity causes water loss in plants while at the same time reducing carbon dioxide, resulting in growth limitation. On the other hand, too much humidity on the crops

surrounding it could attract the growth of molds, bacteria, pests, and fungus that cause diseases and rot on several parts of the plant [26]. When the temperature rises and there is no air humidity, water vapor enters the leaf through the stomata opening. Hence, transpiration is accelerated on hot, dry, and windy days, and it slows down when the temperature is cool and no wind is present [23].

Table 8. Summary of Related Studies in Ambience Conditions, Humidity

Author	Year	Title	Relevant Findings	Relationship to the Study
Texas A&M Agrilife Extension		Light, Temperature and Humidity	An aspect to consider for plant growth and development is the amount of water molecules or moisture in the air because it has an impact on how plants transpire.	Air Humidity is one of the factors that the proposed system will detect and correct
T. W. Tibbitts	1979	Humidity and Plants	An excessive amount of humidity nearby crops may lead to the accumulation of molds, bacteria, bugs and fungus, which can lead to illness and decay on different areas of the plant.	The study aims to evaluate the humidity on the crop's surroundings and to correct it depending on what is necessary for the plant.
G. E. T. Involved and C. Us	2022	Environmental Factors Affecting Plant Growth	On days that are warm, dry and windy, transpiration increases, while on days that are chilly and windless, it decreases.	The proposed system will utilize ambience humidity sensor and correction system to improve the plants growth and development

2.3.3 Luminance

Luminance, or light intensity, simply refers to the concentration or amount of sunlight. It differs every season, and it is an important factor in plants growth and

development since it enables photosynthesis. The more solar luminance it receives, the more food plants could generate [23]. Moreover, the effects of exposure to proper light intensity are visible in the plant's physical appearance and characteristics. A plant grown outdoors that receives enough sunlight tends to have stronger branches and larger, darker green leaves than the same plant grown in a shaded area [25]. Each variety of crop has a different optimal intensity of light. When the sunlight received by the plant is optimal, photosynthesis is at its maximum, and the plant could potentially grow faster. On the other hand, the growth is less if the luminance is lower [27].

Table 9. Summary of Related Studies in Ambience Conditions, Luminance

Author	Year	Title	Relevant Findings	Relationship to the Study
G. E. T. Involved and C. Us	2022	Environmental Factors Affecting Plant Growth	Receiving solar luminance will result in more production of food plants.	The study aims to detect the light intensity received by plants since it is one of the factors that affect crop growth
Texas A&M Agrilife Extension		Light, Temperature and Humidity	Exposure to the right amount of light intensity has an impact on the physical properties and appearance of the plant.	On the system, a light sensor will be utilized to detect the luminance received by plants
J. Badgery	1999	Light in the greenhouse	The ideal light intensity differ depends on the crops.	The study also measures crop perceived light intensity.

2.4 Soil Monitoring/Analysis

2.4.1 Most Common Sensors

The parameters in measuring soil can be divided into two which are soil parameters and environment parameters. The soil parameters include soil moisture and soil temperature sensors while environmental parameters use light intensity, relative humidity, and ambient temperature. The study compares the used components for soil moisture sensors which are the resistance-based and capacitance-based. As a result, a resistance-based is cheaper but has a poor accuracy while capacitance-based has a cost with a reasonable accuracy [28].

The designed system of intelligent monitoring of environmental data of soil in farmlands is capable of sensing the soil temperature and humidity, light intensity and carbon dioxide concentration for every interval of 5 minutes. While formulating the design, the proponents consider that sensor nodes should occupy only little space for convenience. [29].

The sensor node is composed of different hardware elements that are used to collect specific values regarding the soil, plant and the surrounding area. Specifically, a temperature sensor is used to measure the coldness and hotness of the ambient, a humidity sensor to detect the air moisture, a soil moisture sensor to measure the volumetric water content of the land and lastly the biosensor to analyze the plant's health by determining microorganisms and antibodies. All of these sensors are combined to provide the best solution for agricultural needs [30].

The proposed LoRaFarm [31] is an integration of different IoT technologies aiming to monitor the soil and environmental conditions of farms through sensor data. Four sensor nodes were placed on the Greenhouse and Vineyard to evaluate the soil's moisture and temperature, along with the humidity and temperature of the surroundings. Specifically, the sensors installed on the Vineyard module were the SHT-10 for soil humidity and temperature evaluation and the DS18B29 for soil temperature. On the Greenhouse module, AM2032 was placed to measure the air humidity and temperature, along with the PlanDeploSHT-10 to detect the soil condition. Aside from that, some nodes support feedback, which comes with actuator nodes to enable automation on watering and humidity control. Each end node was equipped with a LoRa module to forward data on the Network Server through the LoRa Gateway. Upon evaluating the results gathered by the sensors after deployment for a span of 81 days, it can be inferred that the sensed parameters follow a pattern that is appropriate to the soil and ambient conditions during night and day hours.

To collect data and display it on the Ubidots website, all sensors will function 24 hours a day. As a result, the data will be represented on a graph for analysis. This type of data will give the farmers extremely useful information for improving the quality of treatment for the African Violet Plant. This [32] study uses various sensors, such as pH level sensors, moisture sensors, turbidity sensors, and DHT-22 sensors. For the African Violet Plant to thrive in the best possible conditions, the humidity should be between 50% and 60%, the temperature should be between 15°C and 26°C, the soil pH should be between 5.8 and 6.2, the soil wetness should be moist to dry, and the water quality should be pure. The pH Level Sensor: The data from this sensor indicates the pH level in the water supply began at 6.7, indicating the water was at its natural level. Unfortunately, there was a change in the graph,

which appeared to be a sensor malfunction because the water changed again afterwards. The pH of the water remained at 7.5 for the next few days, indicating that it was slightly acidic. For the next sensor, the Turbidity sensor, it is the responsibility of the Nephelometric Turbidity Units to know the water quality that will be used to feed the plant. The NTU fluctuated from 0 to 580 NTU on the first days but then remained steady for the next four days until the last day. Next is the Moisture sensor; the information obtained from the soil moisture graph can assist the farmer in determining whether the soil is wet or dry. Lastly, for the DHT-22 (Temperature and Humidity Sensor), it is run every 30 minutes for the whole 5 days. From the first to the last day, the average temperature was between 27°C and roughly 23°C. Every day, the data is nearly identical. The data also shows the lowest and maximum temperatures for each day. Humidity rises to roughly 95% before dropping at midnight. After midnight on each day, the amount of humidity dropped to roughly 60% before increasing again.

According to the results of this experiment, the sensors successfully collected data from the African Violet Plant and transferred it to the gateway. The obtained result provides vital information for agriculturists in developing a way to assist farmers in improving the quality of their agriculture [32].

The LM35 sensor is used to determine the temperature of the soil. A moisture sensor is used to determine the amount of water present in the soil. This allows you to keep track of the crop's water requirements. Overwatering the plant can impede its development. Farmers can irrigate their lands only, when necessary, by measuring the water content of the soil. pH and NPK sensors are used to maintain soil quality by advising on fertilizer application. Sensors with selective electrodes are employed to measure NPK ions [10].

The Soil Health Monitoring Unit, or SHMU, was deployed to different parts of the testing farmland, and it is configured to send sensed data every 10 minutes. Each SHMU is equipped with sensors to efficiently evaluate soil parameters such as temperature, moisture, electrical conductivity, salinity, and carbon dioxide concentration. In order to find the required parameters, the system used Teros 12 for soil temperature, moisture, and EC and GMP 251 to evaluate the carbon dioxide concentration of the soil. Using the data acquired by the sensors, it was found that soil temperature has a direct relationship to the concentration of carbon dioxide, and soil moisture affects the soil's electrical conductivity. [33].

Table 10. Summary of Related Studies in Soil Monitoring/Analysis, Most Common Sensors

Author	Year	Title	Relevant Findings	Relationship to the Study
S. Heble et al.	2018	A low power IoT network for smart agriculture	<p>Temperature and moisture content of the soil as well as the ambient temperature and humidity.</p> <p>Capacitance-based devices are more expensive, but they have better accuracy than resistance-based devices.</p>	The parameters measured in soil and environment are similar to the study.
G. Codeluppi et al.	2020	LoraFarM: A LoRaWAN-based smart farming modular IoT architecture	It can be concluded from an analysis of the sensor data collected over the period of 81 days that the measured parameters exhibit a pattern that the soil and ambient conditions both during the day and at night.	Aside from the sensor used in the sensor node, it has a similarity in feedback integrated using the actuator in automation of watering and humidity control.
S. R. J. Ramson et al.	2021	A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System	They measure soil carbon dioxide, moisture, EC, and GMP251. The sensors indicated that soil wetness affects electrical conductivity and that soil temperature and carbon dioxide levels are linked.	Both studies will use a sensor that analyzes the soil parameters such as temperature, moisture, electrical conductivity or salinity.

2.5 Wireless Sensor Network

In the field of agriculture, the most common applications of wireless sensor nodes are for soil health analysis, irrigation control, environmental and livestock monitoring, and the chemical condition of plants [34].

Utilizing wireless sensor networks in agriculture will lead to high-yield production of crops while spending less money since it requires less human intervention. In the paper Modern Agriculture Using Wireless Sensor Networks, the soil's moisture, the plant's health, and the ambient temperature and humidity are detected using the different sensors. The data collected is transmitted to the web server so that farmers can access their farm's monitored values anywhere [30].

Using a collection of strategically distributed sensors, Wireless Sensor Networks are used to investigate environmental conditions. This is accomplished by exchanging data among nodes in a cooperative manner until it reaches the Gateway node. The monitoring of soil conditions is one of the many uses of WSN. By carefully monitoring the quality of the soil in relation to the crop, soil management can boost production. The suggested Soil Quality Management System can detect electrical conductivity, pH, temperature, NPK (Nitrogen Phosphate Potassium) concentration, and light intensity thanks to the use of a wireless sensor network. The system was proven to be built with less effort, well-maintained, and has the benefit of being low-cost and easily scalable [10].

Moreover, the researchers also achieved the best crop suggestion appropriate to the measured values of NPK on farmlands, which will undoubtedly help farmers in their decision-making to maximize their yields. Furthermore, the proponents considered the addition of parameters such as humidity, pH, and several micronutrients essential for plant growth to improve

the system. Likewise, the monitoring system's algorithm can be enhanced to increase accuracy in terms of providing crop and fertilizer recommendations.

The system to analyze the soil's temperature, moisture, salinity, electrical conductivity, and carbon dioxide concentration was developed through the integration of IoT technology. A total of nine sensor nodes were scattered on an agricultural field to monitor the parameters essential for crop growth and production. After several weeks of being deployed, it has been determined that the system is apt for wireless sensing of soil and can be used for smart agriculture, specifically decision-making. The results have proven that the design was better than other systems in terms of real-time soil monitoring. Specifically, the system deployed was able to sense soil condition and was scalable, wide-range, low-power, and sustainable. The SHMU was flexible since it could accommodate additional sensors. It achieved the goal of covering 3442 m² of land with a packet delivery rate of 85%. Lastly, the unit consumes only 13 mA of current, which may be operated for several days with the installed 2500 mAh battery. Moreover, the unit was designed to power itself, and it can fully charge the battery for 14 days. Somehow, the system still lacks important parameters and features that could be improved. Particularly, the sensor nodes could be mapped or their location could be accessed through geolocation. The addition of sensors to measure other soil nutrient content and chemical conditions such as nitrate, phosphate, redox, pH, and oxygen concentration should be considered. Finally, the system could be enhanced by adding a feedback system to increase crop production through machine learning and data processing algorithms [33].

WSN, or wireless sensor network, is the technology behind the designed wireless system that is capable of accurately measuring the environmental data on farmlands in real time. The system should be low-cost and low-power to ensure that it is stable and efficient. Lastly, the radio frequency that will be used should be capable of improving the operation of the system [29]. The

data sensed by the nodes is transmitted to the monitoring system through the MAX3232 communication module, which is Zigbee-based. The acquired data is displayed on the host computer, and it can also be accessed by other networks connected to the host PC.

According to the study of an IoT-based Environment Change Monitoring and control system in greenhouses using WSN, WSN will interact with the actual world as a result of the IoT. This system includes sensors that allow agricultural fields to be connected to the IoT. Product quality will increase as a result of this system. The connection sets up connections between agronomists and farms, thereby improving agricultural product output. In continuation of this study, it has built a wireless sensor network based on ZigBee in this system that was designed in 2017. The data collection and storage of greenhouse air temperature, air humidity, and soil temperature may be easily arranged by installing wireless sensor nodes in various locations around the greenhouse. In addition, the greenhouse equipment is automatically controlled [12].

In another study about soil moisture monitoring, the researchers used Wireless Sensor Network (WSN) as a technology that transmits data collected from a field of interest over a wireless link. WSN can be used in a variety of fields, including monitoring, wireless measurements, and control. It is essential to continuously monitor the fields in precision agriculture and organic farming since they are site specific. Monitoring plant health is critical for increasing food grain productivity. One of the most basic factors influencing plant health is soil moisture. The water that remains in the soil as a thin film aids plant growth by supplying nutrients. According to the study, WSN has a major problem on power consumption as there is also a control process. Using Exponential Weighted Moving Average, external switches can be removed and as an algorithm, the process will only occur when conditions are met resulting for nodes to save some power. [35].

The developed wireless sensor network can be deployed to monitor soil conditions and environmental parameters in an agricultural area and send the data wirelessly to an online web server for data tracking and observation. The system is practical for deployment in agricultural areas with weak network signals, such those common in the Philippines [36].

Table 11. Summary of Related Studies in Wireless Sensor Network

Author	Year	Title	Relevant Findings	Relationship to the Study
U. Madhura et al.	2018	Soil Quality Management Using Wireless Sensor Network	A low-cost and easily accessible WSN with nRF transceivers can be developed with minimal effort and well maintained.	A WSN utilized in both studies in measuring environmental conditions.
S. N. Shylaja and M. B. Veena	2017	Real-time monitoring of soil nutrient analysis using WSN	A wireless sensor network for soil health monitoring and analysis can determine soil nutrients. Adding humidity, pH, and other micronutrients detection were recommended to improve the system.	Both studies used a WSN-based monitoring for soil health and nutrients.
S. R. J. Ramson et al.	2021	A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System	A superior architecture for real-time soil monitoring is scalable, wide-ranging, low-power, and sustainable. The research covered 3442 m with 85% packet delivery.	Both studies utilized the system to analyze the soil's temperature, moisture, salinity integrated in IoT technology.

X. Shipu et al.	2019	Design and Research on the Farmland Soil Environmental Monitoring System Based on Agricultural Network	To guarantee that the system is reliable and effective, it should be inexpensive and power efficient. Furthermore, radio frequency affects system performance.	A low cost and low power consumption system will be implemented in both studies.
Ezhilazhahi, A. M., & Bhuvaneswari, P. T. V.	2017	IoT enabled plant soil moisture monitoring using wireless sensor networks	Power consumption is major problem encountered in using a WSN. External switches can be eliminated using the Exponential Weighted Moving Average since the algorithm only operates when certain conditions are met, hence, nodes can save some power.	Both studies will use the WSN in monitoring, wireless measurements, and control.
Guico, M. et al.	2019	Wireless Sensor Network for Soil Monitoring	For deployment in agricultural areas with poor network signals, such those typical in the Philippines, the WSN with Adafruit Feather FONA is practical.	A web server will be utilized in storing and monitoring the data in both studies.

2.5.1 Zigbee

The study by A. Mahir et al. describe a crop field area that will be monitored through a ZigBee wireless sensor network. To do this, sensors for measuring soil moisture can be placed across the area to identify any areas where the water table is low. Proponents can irrigate that specific area of the field using those results as a starting point. By doing so, the proponents may conserve water and lessen the issue of water logging on the land. An efficient and reliable system for monitoring agricultural parameters is a ZigBee-based system. When different agriculture farms' sensor nodes are required for gathering data on microclimate, atmospheric conditions, and plant data, this is highly desirable. The researchers recommended utilizing an excess heating control system and a frost management system to improve crop quality and yield. [37].

According to [38], The Zigbee wireless network has been proposed by many researchers as a means of monitoring environmental variables. It provides a number of benefits, such as affordability, low power consumption, mesh topology, and multi-hop data transfer. By using the Zigbee/IEEE802.15.4 protocol, wireless communication between indoor servers and weather stations can be stored. PLCs are utilized as network masters since they are industrial controllers. The Modbus RTU industrial protocol, which was used to gather the transmitted data in the wireless network, has thus also been deployed. A Zigbee wireless connection in a star topology connects the weather stations to the indoor server. Xbee modules are arranged for RFD. It allows for wireless Zigbee network communication between indoor servers and weather stations. Using a UART port, RFD is connected to the Arduino.

A star topology was used to link the ZigBee module. ZigBee modules wirelessly transmit data to a data center. When necessary, this information can be transferred to the client's mobile device as well as presented and evaluated on the primary monitoring software. Based on the basic diagram of the system [39], every ZED device collects data from its own cover area, sending it to the data center after measuring the temperature, atmospheric pressure, soil moisture, and light intensity. Temperature, pressure, moisture, and light intensity are all monitored by all nodes. A star topology links the monitoring nodes together. As a result, each node is permitted to send data from associated sensors to the data center via the node that is closest to it. The data center is situated in a prominent building. ZigBee Coordinator (ZC) integrates monitoring node data that is routinely sent through a GSM modem to the data center database. Also, it shows the structure of their Zigbee module and their unit nodes. The CC2530 ZigBee SoC-based monitoring node has a controller, an analog module with light intensity and soil moisture sensors, a digital module with air pressure and temperature sensors, and a connection to DC power. An independent battery powers the monitoring nodes. A star topology is used to connect one ZC (ZigBee Coordinator) and three ZED (ZigBee End Device) modules. ZC receives measurements from all of ZED's associated sensors on a regular basis. A direct connection to the data center is made using ZC. ZC gathers data from ZigBee nodes and provides it to the main monitoring application. The monitoring program shows the data from each ZigBee node independently. The set-up of their monitoring is that one ZigBee end node's cover area serves as the division between monitoring zones. The database contains the values for all sensors and the states of all systems [39].

For the implementation of an agricultural monitoring system with the Zigbee network. There are two weather stations, each located in a different part of an agricultural field, that have delivered data that will be monitored and recorded by an inside server. The indoor server uses a Programmable Logic Controller (PLC) as the primary controller, interacting with an Xbee module that serves as a coordinator and being supplemented by an Arduino board to enable wireless connectivity. The suggested system gathers data through the Zigbee wireless network using the Modbus RTU protocol. Based on their findings, they determine that each interior server may connect with each weather station up to 200 meters away using Zigbee wireless [37].

In conclusion, ZigBee end nodes wirelessly transmit their own sensing data to a ZigBee coordinator (ZC). ZC transfers the data to the data center, which gathers, stores, and makes it possible for analysis. Whenever necessary, this data can be shown on the primary monitoring software. Connecting a wireless module to soil moisture and photocell sensors, then completing the necessary actions to identify each attribute. They recommend that for future researchers, energy consumption, technical aspects, and routing algorithms be investigated for mesh networks. This system's development and implementation allow for the monitoring of growth stages, advance planning of the arrangement, and crop failure prevention. Additionally, the technology needs to be tested in actual agricultural settings since they didn't have a chance to deploy it. Zigbee mesh networking must be used to monitor large agricultural areas [39].

Table 12. Summary of Related Studies in Wireless Sensor Network, Zigbee

Author	Year	Title	Relevant Findings	Relationship to the Study
A. Mahir	2018	Soil Monitoring System using Zigbee for Smart Agriculture	The study was able to transmit measured data using Zigbee.	Both studies used Zigbee as a microprocessor.
O. Ayurzana and S. Tsagaanchuluun	2021	Monitoring System of Agriculture Fields using Zigbee Modules	The study found that a Zigbee module can transmit and receive data with the range of 200 meters away to each other with the use of Modbus RTU protocol.	The XBee module was utilized to transmit measured data from the sensor node.
W. Koodtalang and T. Sangsuwan	2020	Agricultural Monitoring System with Zigbee and PLC based on Modbus RTU Protocol	The study was able to transfer data which gathers and store in the database through Zigbee coordinator.	Both studies transmit the acquired data from the sensor node through Zigbee communication.

2.6 Data Analysis

2.6.1 Data Monitoring

A farmer can acquire real-time test results of important soil parameters, which are important for increasing yield, using the proposed approach. The embedded systems used

in this system are open-source Arduino and Raspberry Pi. The purpose of soil testing is to determine the soil's nutritional potential. To determine the best fertilizer application rate for the studied soil, data analysis is performed. Fertilization in excess can harm both the crops and the environment. The system is used to help people make informed ecological and economic land-use decisions. It advocates for better soil nutrient management. For archiving previous soil quality readings, a cloud database is used. The database also includes NPK ratings for a variety of fertilizers as well as crop NPK requirements. Data from newly obtained soil test results will be uploaded to this cloud database for examination. Using shell scripting, the uploading procedure is also automated. A GSM module is utilized by the Raspberry Pi to connect to the internet and access the cloud. This data is displayed on a dashboard after the amount of fertilizer needed for NPK is calculated. Other information, such as the amount of water needed by the crop, is also provided. All of the information is sent to the farmer's mobile phone via SMS [10].

All of the data sensed by the end nodes was transmitted to the network server, and it can be viewed on the website monitoring platform and on the VegIoT Garden mobile app. On the website's dashboard, the exact location of every end node can be seen over the farm's map, and the data gathered by the system is displayed on the plot. Further, the system's mobile app functions similarly to the web dashboard, where the farm's data can be observed. Aside from that, the mobile app is integrated with settings and a notification system that the user or farmer can modify to determine when and at what threshold the user wants to be notified. Note that mobile phone monitoring was possible due to the application of the Constrained Application Protocol Sensors Module [31].

Table 13. Summary of Related Studies in Data Analysis, Data Monitoring

Author	Year	Title	Relevant Findings	Relationship to the Study
C. Z. Myint et al.	2017	WSN-based reconfigurable water quality monitoring system in IoT environment	The study utilizes a monitoring website that displays the sensed farm data. Accordingly, the proponents were able to provide an effective notification system which was incorporated via SMS.	Both studies have the same method on data analysis in determining the fertilizer that is needed in measured NPK.
G. Codeluppi et al.	2020	LoraFarM: A LoRaWAN-based smart farming modular IoT architecture	Utilizing CoAP, with sensor nodes being constrained, the data transmission to the monitoring device was possible using this protocol. This provided a wider communication between the nodes and the Internet.	The notification and recommendation as the feedback in the mobile application is utilized in both studies.

2.6.2 Feedback and Control Process

ASF is an automatic fertilizer suggestion system where farmers are registered to have specific crops of choice, and the algorithm provides fertilizer suggestions automatically. The soil analysis will occur in a laboratory setting, and the data results must be manually entered while the algorithm determines what nutrients the soil lacks. The algorithm provides about 90% accuracy between the system and manual computation. The 10% deviation was checked by the proponents, who concluded that it was due to human error,

thus making the algorithm design more accurate, time-efficient, and simpler for record management for the farmers [40].

A closed-loop system was designed in the Arduino-based WSN for greenhouses, as its feedback focuses on controlling the parameters using actuators as it meets certain conditions. The system reads the raw data and solves the deficiency or abundance of a parameter to meet the requirements. Based on the result, the actuators will automatically work regardless of whether the threshold is on the lower or upper spectrum that the proponents set. As shown by the study, temperature varies during the morning, hence the temperature sensors monitor the values, and actuators in line with the sensor are connected to cooling fans and heaters. The accuracy of the feedback process based on the resulting table was above 90%, as the temperature does not exceed more than 27 degrees and does not falter below 21 degrees, which is the optimal temperature for the greenhouse [11].

Table 14. Summary of Related Studies in Data Analysis, Feedback and Control Process

Author	Year	Title	Relevant Findings	Relationship to the Study
M. K. Dharani and K. R. P. Kumar	2019	Automated Soil Nutrient Content Analysis and Fertilizer Suggestion for Farmers.	The algorithm achieves roughly 90% accuracy between the system and manual computation. The 10% variance is due to human error therefore a high accuracy for the algorithm design, time efficient, and easier record management for farmers.	Both studies have an algorithm that suggests what's best for the crops.
R. Bhowmik and D. A. Pathak	2017	an Arduino Based Wsn To Control and Monitor the Greenhouse Parameters.	Provided that the temperature does not fall below 21 degrees or rise above 27 degrees, the ideal temperature for a greenhouse is 25 degrees, the accuracy of the feedback mechanism based on the resulting table was above 90%.	Both studies will have an actuator as a correcting system of the parameters.

2.6.3 Web Server

The web server that was used is a standalone web server. This web server will act as a hub for data processing and servicing between Wasp mote, the database, and the client. With the support of the 3G module, the web server received soil moisture data, ground temperature, room temperature, and battery capacity supplied directly by the Wasp mote board. This information will then be stored in a database and processed before being given to the client in the form of charts, graphs, and plug-ins [10].

ThingSpeak is open-source software capable of cloud data and data monitoring using IoT technology. With this software, monitoring is done through mobile devices, as the main people assigned for monitoring are the plantation workers. Data results along with graphs are shown through this software, and for real-time monitoring, every two (2) minutes, a sample of data is taken and uploaded to the cloud. In terms of accuracy, based on the table results, the 2-minute interval per data gathered is accurate, but the accuracy of data for temperature and tilt cannot be confirmed as these are not compared to another medium for measuring these parameters [41].

Table 15. Summary of Related Studies in Data Analysis, Web Server

Author	Year	Title	Relevant Findings	Relationship to the Study
M. S. M. Saleh et al.	2021	IoT Real-Time Soil Monitoring Based on LoRa for Palm Oil Plantation	Using Thingspeak, real time monitoring became possible with a high accuracy in 2 minutes interval in transmission of the measured data except for temperature and tilt since it is not comparable to other medium of measuring such parameters.	The data gathered from the sensor node will then be stored in the database on both studies.

2.6.4 Notification System

The customer can specify notification parameters for soil moisture to be monitored in the monitoring application system; in this case, the notification parameter scale is 1–100 percent. When the client sets the notification parameters, such as 39%, and the web server receives data from the soil moisture sensor with the same percentage level as the regulated

notification parameters, the client will be notified that the soil moisture content monitored has reached the set parameter limits. The client will also receive notification about percentage levels that are less than the parameters set if the data received from the server is less than or smaller than the notification parameters set, and the system will inform the customer of the percentage levels of pre-set humidity upon receiving the notice [10].

Table 16. Summary of Related Studies in Data Analysis, Notification System

Author	Year	Title	Relevant Findings	Relationship to the Study
U. Madhura, P. Akshay, A. J. Bhattad, and G. Nagaraja	2018	Soil Quality Management Using Wireless Sensor Network	If the data received from the server percentage is less than or smaller than the notification parameters set, the client will receive notification about percentage levels that are below those parameters. The system will then inform the client about what percentage levels of pre-set humidity and levels percentage of humidity after receiving the notification.	Both studies used the same technique in implementing the notification system wherein the user will be notified that the parameter is at limit depending on the crops.

2.7 Power Management

2.7.1 Battery

AQUAlity device, an automated water monitoring and correction device based on IoT technology, runs on a 12VDC, 12Ah sealed lead-acid battery for the whole system that includes the Packetduino (an Arduino clone created by Packetworx), multiple relays, and multiple types of sensors. Using this battery, the device is powered for five (5) straight days while maintaining up to 50% depth of discharge of the battery [42].

To identify the total operation time of the Soil Health Monitoring Unit without the battery being recharged, the proponents used the INA233 current monitor developed by Texas Instruments. SHMU was found to consume an average of 34 mA in active mode and 328 nA in sleep mode. The SHMU's overall average current consumption was 13 mA, taking into account all other factors and specifications. The SHMU with a 2500 mAh battery can run for up to 163.46 hours or 6.8 days without recharging, according to the calculations [33].

The study by Guico et. al shows that the sensor node system is powered by 6600mAh batteries and a 2200 mAh battery dedicated for the microcontroller. The whole circuit is connected to two solar panels connected in parallel with 1.3W, 5.5V ratings [35].

Another study relating to Soil moisture monitoring in LoRaWAN [43] evaluates and analyzes batteries before selecting the most appropriate battery for the prototype, which is the Saft LS14500. The device environment, particularly the temperature, as well as information about the MCU's power consumption, must be known in order to select an

appropriate battery. This enables the project to incorporate a battery holder. Because of the risk of corrosion on the connections, a battery holder would not be incorporated into the final product design. The end product would include a battery with wire leads, tags, or PCB pins. The ability to change the battery quickly is more vital than avoiding corrosion during the prototype's testing phase. The gadget will have a low to medium-low discharge rate (between uA and mA). For devices with such behavior, a non-rechargeable battery has a lower drain rate than a rechargeable. The Saft LS14500 is the best battery for this project because of its 2600 mAh capacity. According to the datasheet for the Saft LS14500, it has a self-discharge rate of less than 1% per year of storage at 20°C, which makes it a good choice because the device will be turned off from late fall to late spring.

Table 17. Summary of Related Studies in Power Management, Battery

Author	Year	Title	Relevant Findings	Relationship to the Study
L. K. Tolentino et al.	2021	IoT-based automated water monitoring and correcting modular device via LoRaWAN for aquaculture	The device runs on a 12 VDC, 12 Ah sealed lead-acid battery to which tested in a 5-day straight runtime which showed a maintaining depth of discharge of up to 50%.	Both studies used a battery that can power multiple components in a long time.
S. R. J. Ramson et al.	2021	A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System	The calculations shows that the SHMU with 2500 mAh battery may operate approximately 163.46 hours or 6.8 days without recharging.	The proponents used a method in maintaining a long battery life while monitoring the soil health.
M. Johansson	2021	Soil Moisture Monitoring System Using LoRaWAN Technology	To choose the right battery, one must take into account the device's surroundings, especially the temperature, and the MCU's power requirements. Fast battery replacement is more important than preventing corrosion while testing the prototype.	Both studies used a most effective battery since a device will have multiple components that will consume power.

2.7.2 Buck Converter and Boost Converter

A WSN was used to transmit the estimated soil parameters using sensor nodes. The sensor node is composed of a variety of technologies to ensure reliability along with a high accuracy. It is divided into different units such as sensor module, microcontroller unit, wireless module, and power management module. A proper power management is advisable in conducting a WSN since it consumes more power that's why a BQ25570 is introduced (a boost converter) integrated with PV cell together with the DPM strategy which help to have a maximum power consumption of 7.37mWh at 1.97mAh [8].

The Soil Health Monitoring Unit is integrated with a power management module comprised of different IC and components to sustain the energy requirement of the sensors and other circuitry. Particularly, BQ25505 IC for energy harvesting and battery charging, MCP16252T or dc to dc boost converter to produce 5V output and a low-dropout regulator to have 3.3 V output. The Teros 12 sensor will be powered by 5 volts, while the GPS module, LoRa radio, and microcontroller will be powered by 3.3 volts [33].

Table 18. Summary of Related Studies in Power Management, Buck, and Boost Converter

Author	Year	Title	Relevant Findings	Relationship to the Study
J. J. Estrada-Lopez et al.	2018	Smart Soil Parameters Estimation System Using an Autonomous Wireless Sensor Network with Dynamic Power Management Strategy	Since a WSN requires more power, effective power management is advised. To that end, the BQ25570 is introduced (a boost converter), which is integrated with a PV cell and the DPM method, allowing for a maximum energy usage of 7.37mWh at 1.97 mAh.	BQ25570 was used as the buck converter in both studies.
S. R. J. Ramson et al.	2021	A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System	BQ25505 IC for charging batteries and energy generation. A low-dropout regulator with 3.3 V output and an MCP16252T or DC to DC boost converter are used.	The proposed study has a similar power management that contains a buck converter.

2.7.3 Solar Panel

The Soil Health Monitoring Unit's battery is recharged using solar cells, making it environmentally friendly and long-lasting. Based on acquired data, the solar panel can harvest 345.8 mW of power for solar irradiance of 997 W/m². Energy generated by the solar panel and BQ25505 was fed into the system's circuitry and the excess was used to recharge the battery. After 14 days of deployment, the battery's voltage was still at 4.14V

which is above the discharged voltage, indicating that the panel was able to sustain the energy required by the system [33].

A 3.7 V with 1800 mA capacity of Lithium Polymer battery is used to power each of the end nodes of the LoRaFarm network. Using the LiPo charger, the 1 W solar panel recharges the batteries. Based on the evaluated results upon 81 days of deployment, the solar panel is capable of sustaining the node's power requirement for months. It was found that under 25 °C temperature average, it takes 13-14 hours to completely charge the installed LiPo battery. However, on shady days or if the solar panel is placed in a closed vicinity, it will take two to three days to recharge the battery. On the instance that the battery completely discharges, end nodes will resume operation as the battery receives sufficient energy. Note that the farm soil and environmental data from the end nodes is collected every 10 minutes for three sensor nodes and every 30 minutes for the fourth sensor. Thus, end nodes are disabled at times that do not fall within the interval [31].

Table 19. Summary of Related Studies in Power Management, Solar Panel

Author	Year	Title	Relevant Findings	Relationship to the Study
S. R. J. Ramson et al.	2021	A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System	After 14 days after deployment, the battery's voltage was still at 4.14 V, which is above the drained level, suggesting that the panel was able to sustain the energy needed by the system. A solar panel can generate 345.8 mW of power for sun irradiation of 997 W/m ²	In recharging a sensor node, solar cells were used in both studies.
G. Codeluppi et al.	2020	LoraFarM: A LoRaWAN-based smart farming modular IoT architecture	The solar panel can support the node's power needs for months based on evaluations made after 81 days of deployment. The installed LiPo battery was found to require 13-14 hours to fully charge at an average temperature of 25 °C.	Both studies use Lithium Polymer as the battery of the device.

CHAPTER 3

METHODOLOGY

3.1 Research Design

The study focuses on developing an innovative system or technology. A sensor node will be developed that is composed of different sensors that will measure the soil's parameters together with the ambient condition and relays for the correction system of the parameters. Along with the sensor node, a WSN-based gateway will also be developed to transmit the data measured by the sensors. A monitoring application will be implemented to monitor the soil health and the ambient conditions. Thus, the study will be a **Developmental Research** as the proponents will develop a device that will monitor the soil.

The system that the proponents proposed also requires user testing and feedback. The intended respondents for this study are the farmers, as the deployment area is in the greenhouse. The device should be tested to see if it can help the farmer monitor the crops. The respondents' feedback will help to prove that the system is effective enough to monitor the soil and maintain its health condition to prevent the loss of crops. Hence, a **Descriptive Research** will be implemented in the study.

Input	Process	Output
<p>Knowledge Requirements:</p> <ul style="list-style-type: none"> • Application Development • Arduino and Raspberry Pi Programming • Circuit Analysis and Development • Establishing Database • Automation and Control Systems • Crop's nutrient and condition <p>Software Requirements:</p> <ul style="list-style-type: none"> Arduino IDE Thonny IDE 	<p>Hardware Development:</p> <ul style="list-style-type: none"> • Sensor Nodes • Power management system • Feedback Control Unit for the drip irrigation and fertilizer application, fan and humidifier control • Gateway <p>Software Development:</p> <ul style="list-style-type: none"> • Web Server for data storage • Monitoring Application with feedback and notification system. • Program that will send data to the drip irrigation, humidifier, and fan control 	<ul style="list-style-type: none"> • A wireless sensor network enabled soil health and ambience monitoring analysis that detects soil nutrients and atmosphere parameters. • A feedback mechanism that controls certain parameters on plants appropriate to the data acquired by the sensor nodes. <p>Plant selection system available on the mobile monitoring device for the basis of plant requirement parameters.</p>

Android Development Software Multisim and Proteus Firebase Hardware Requirements: XBee S2C Module Arduino Mega 2560 Raspberry Pi 4 Soil NPK Sensor Soil Moisture Sensor Module Waterproof Temperature Sensor DS18B20 Soil pH Sensor Soil EC sensor Light Intensity Module GY-302 BH1750 DHT11 Temperature and Humidity Sensor		
---	--	--

Module		
Solar Panel		
LiPo Battery		
Boost Converter		
Module		
Buck Converter		
Relays		
Water Pump		
Humidifier		
Exhaust Fan		

3.2 Research process Flow

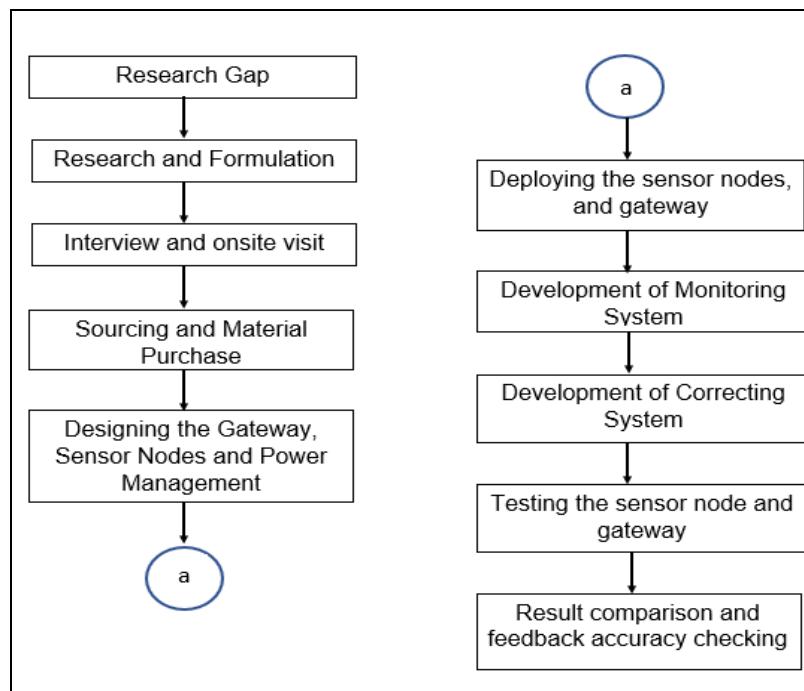


Figure 1. Process Flow of the Study

3.3 Development of the sensor nodes with an efficient power management system.

3.3.1 Materials, Equipments, and Softwares to be Used

3.3.1.1 Hardware Components and Equipment

Soil NPK Sensor by Renkeer

The Soil NPK sensor is used for detecting the content of nitrogen, phosphorus, and potassium in the soil, and judging the fertility of the soil.

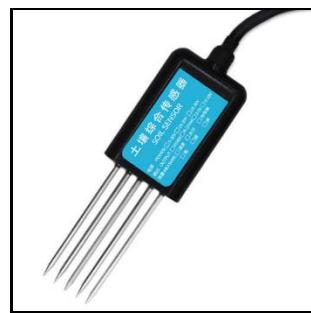


Figure 2. Soil NPK Sensor

Soil pH Sensor by Renkeer

The Soil pH Sensor is a device that measures the soil's current pH. Two stainless steel probes are inserted vertically into the soil to detect the pH value. A soil data logger can be used with this sensor.



Figure 3. Soil pH Sensor

Soil Moisture Sensor Module

The Soil Moisture Sensor Module can be used to monitor soil moisture or determine whether there is water near the sensor, allowing your garden plants to seek human help. When the moisture level is higher or lower than the threshold, the sensor output logic will be HIGH or LOW.

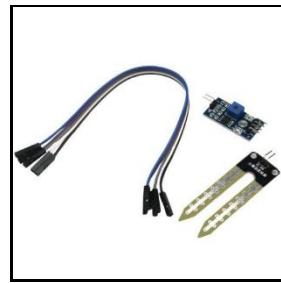


Figure 4. Soil Moisture Sensor Module

Waterproof Temperature Sensor DS18B20

The Waterproof Temperature Sensor DS18B20 will precisely measure temperatures in wet environments with a simple 1-wire interface. It uses a 1-wire interface to provide 9 to 12-bit temperature readings, allowing only one wire and ground to be connected from a central microprocessor.



Figure 5. Waterproof Temperature Sensor DS18B20

Soil EC Sensor by Renkeer

The 3-in-1 soil sensor will be utilized for soil electrical conductivity monitoring. Since it is 3 in 1, 3 parameters are measured by the sensor, which are electrical conductivity, humidity, and temperature. For a better and more accurate measurement, the two parameters will be disregarded and the focus will be on measuring the soil's electrical conductivity. By measuring the dielectric constant of the soil, it can directly and accurately reflect the true moisture content of various soils.



Figure 6. 3 in 1 Soil sensor by Renkeer

Light Intensity Module GY-302 BH1750

The photo-resistor in the Light Intensity Module GY-302 BH1750 detects the intensity of light, and when the intensity of light increases, the resistance of the photo-resistor decreases when the intensity of light increases.

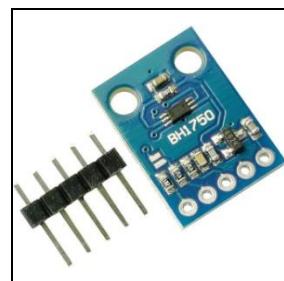


Figure 7. Light Intensity Module GY-302 BH1750

DHT11 Temperature and Humidity Sensor Module

The DHT11 is a simple digital temperature and humidity sensor with a low price. It measures the ambient air with a capacitive humidity sensor and a thermistor and outputs a digital signal on the data pin with no analog input pins needed.

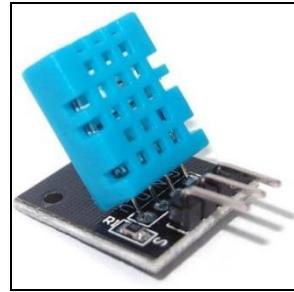


Figure 8. Temperature and Humidity Sensor Module

XBee Module

XBee S2C is a RF module designed for wireless communication or data exchange, and it works on ZigBee mesh communication protocols that sit on top of IEEE 802.15.4 PHY. The module provides wireless connectivity to end-point devices in any ZigBee mesh network, including devices from other vendors. This will be used as the main communication device between the sensor node and the gateway.

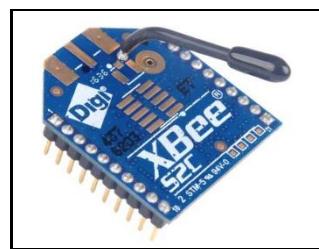


Figure 9. XBee S2C Module

Solar Panel

Solar panels convert sunlight into electrical energy either through photovoltaic (PV) panels or through mirrors that concentrate solar radiation. This will be used in the gateway system.



Figure 10. Solar Panel

Solar Charge Controller

A solar charge controller regulates the flow of energy from the solar array into the battery bank. It ensures that the deep-cycle batteries are not overcharged during the day and that the batteries are not drained overnight by electricity flowing backwards to the solar panels. Charge controllers are available with other features like lighting and load control, although their primary function is power management.



Figure 11. Solar Charge Controller

Lead Acid Battery

Lead acid batteries represent the most widely utilized form of battery. Even though lead acid batteries have a low energy density, only modest efficiency, and significant maintenance requirements, they have a long lifespan and are relatively inexpensive compared to other battery types.



Figure 12. Lead Acid Battery

Boost Converter Module (BQ25570)

The Boost Converter Module (BQ25570) is designed to capture and manage power from a variety of DC power sources, including photovoltaic (solar), thermoelectric generators, and piezoelectric modules, in the microwatts (W) to milliwatts (mW) range.

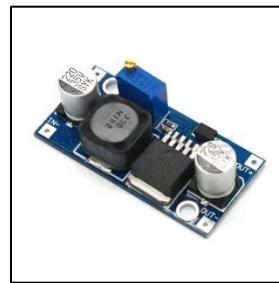


Figure 13. Boost Converter Module (BQ25570)

Buck Converter (LM2596)

The Buck Converter (LM2596) is a step-down (buck) switching regulator with excellent line and load regulation that can drive a 3-A load.



Figure 14. Buck Converter (LM2596)

Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board containing 54 digital input/output pins, 16 analog inputs, 4 UARTs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. This will hold all the components to be used on the sensor node.

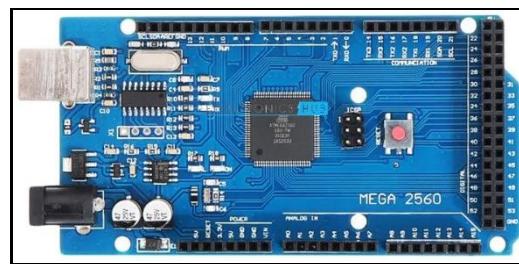


Figure 15. Arduino Mega 2560

3.3.1.2 Software Application

Arduino IDE

Arduino Software used for writing codes and uploading them to the board.

The programming of Analog to Digital Converter and Serial Communication using XBee will be done using this IDE.



Figure 16. Arduino IDE

3.3.2 Designing and Construction of the Sensor nodes

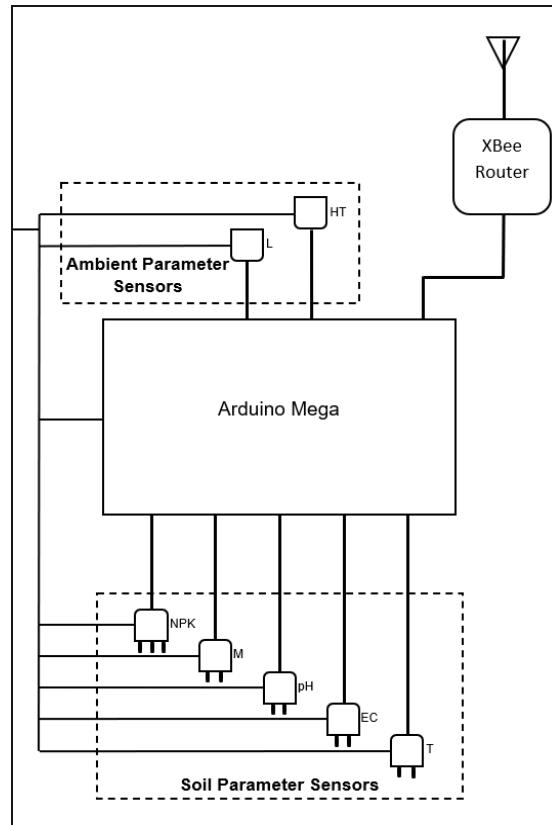


Figure 17. Sensor Node Architecture Without Power Management Module

The design will be combined into one sensor node that includes all of the following sensors: soil NPK sensor, EC sensor, temperature and humidity sensor, light intensity sensor, temperature sensor, soil moisture sensor, and soil pH sensor. To transmit the sensor data acquired, the node is integrated with the ZigBee transceiver module. The sensor nodes' design will ensure that the chassis material is waterproof and sturdy, ensuring the safety of the sensor nodes.

3.3.3 Programming the Microcontroller

All the instructions and functionalities of the sensor nodes will be written on the Arduino IDE. The line of codes will include the commands for data conversion to suitable form for better transmission, the accepted threshold range for each sensor parameter and the control of the relays whenever anomalies are detected. Generally, the microcontroller will be programmed to instruct the sensors, communication module, power management system and the relays what to do in different circumstances.

3.3.4 Sensor Calibration

The sensors are needed to be calibrated first before deployment to test its accuracy and to ensure that no anomaly on the result will be gathered. The sensor will be calibrated with the help of the Department of Science and Technology (DOST). Also, before being inspected by the Department of Science and Technology, the sensor will be tested on a sample of soil and compared to other sensors to confirm that it is functional and precise.

3.3.5 Design of Power Management for Sensor nodes

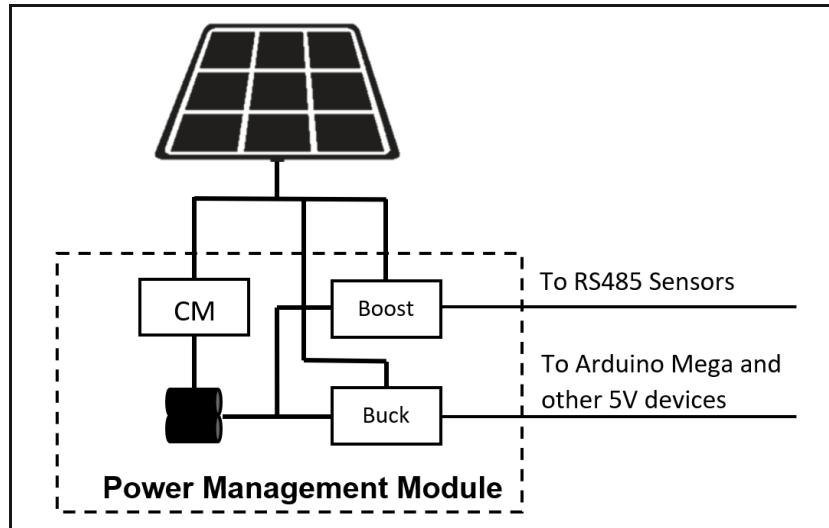


Figure 18. Power Management Architecture

The sensor node's power management is in charge of keeping the sensor operating by using a rechargeable Lithium-Polymer battery attached to the solar panel. A Charging Module (CM) is connected to the battery in order to recharge it from the solar panel. A boost converter and buck converter can be powered by the solar panel and the battery. A boost converter is used for 12V devices while the buck converter is used for the Arduino and 5V sensors.

3.4 Development of Database and Gateway, and Establishment of Connection

3.4.1 Materials, Equipment, and Software to be Used

3.4.1.1 Hardware Components

Raspberry Pi

For the gateway system, researchers will use Raspberry pi as a microcontroller. It is a low-cost computer the size of a credit card that plugs to a

computer monitor or television and uses a standard keyboard and mouse. This microcontroller has an integrated wifi module, so it can connect to a Wi-Fi access point.



Figure 19. Raspberry Pi

XBee Module

XBee S2C is a RF module designed for wireless communication or data exchange and it works on ZigBee mesh communication protocols that sit on top of IEEE 802.15.4 PHY. The module provides wireless connectivity to end-point devices in any ZigBee mesh networks including devices from other vendors. This will be used as the main communication device between the sensor node and the gateway.

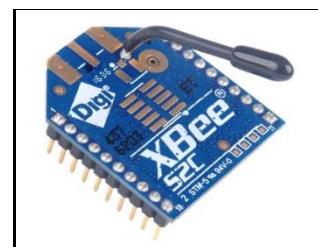


Figure 20. XBee S2C Module

3.4.1.2 Softwares to be Used

Thonny

Thonny is a free, dedicated IDE for Python designed for beginners. It is an IDE bundled with the latest version of the Raspbian (Raspberry Pi OS) with PIXEL operating system. This will be used as the main programming environment for the connection of raspberry pi to the database.

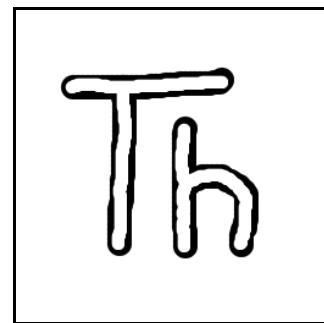


Figure 21. Thonny IDE

Firebase

Firebase is a cloud-hosted NoSQL database for storing data and syncing data and end users for real time monitoring. The proponents will be utilizing this database as it is used for real time and free. The data sent from the gateway will be stored here in Firebase.



Figure 22. Firebase Realtime Database

3.4.2 Designing and Construction of Gateway

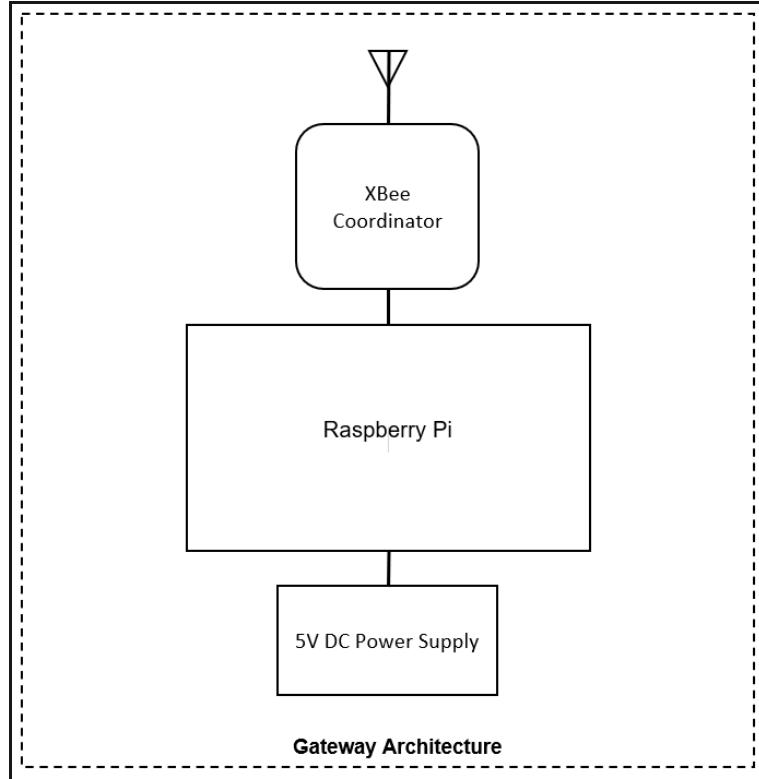


Figure 23. Gateway Architecture

The proponents will design and construct a gateway system that includes a Raspberry Pi as a microprocessor with an integrated Wi-Fi module and an XBee S2C Module. The gateway system also includes an antenna that has a proper placement, height, and dimension, and a case that will protect the components inside the gateway. This will be powered by plugging the gateway system into the main outlet.

3.4.3 Programming the Microprocessor as a Gateway

The microprocessor will be programmed using Thonny IDE. The Raspberry Pi will be utilized to act as a gateway as it has a built-in WiFi module. The Thonny IDE will be used to program the data gathering and transmission from the sensor nodes and database.

3.4.4 Establishing Secure Connection Between WSN and Database

The proponents will utilize Firebase for secure connection between gateway and Database. All the data that was sent from the gateway system will be stored in Firebase. This database is utilized since it is free and most systems use it for real-time monitoring.

3.5 Development of Mobile Application for Monitoring and Selection System

3.5.1 Softwares to be Used

MIT App Inventor



Figure 24. MIT App Inventor

For the creation of monitoring software, the proponents will utilize the MIT App Inventor which is a web application IDE for android app development. MIT App Inventor offers features like visual layout editor, open blocks programming, Intelligent code editor, and flexible build system which will ease the building of the monitoring application.

3.5.2 Android Application Development

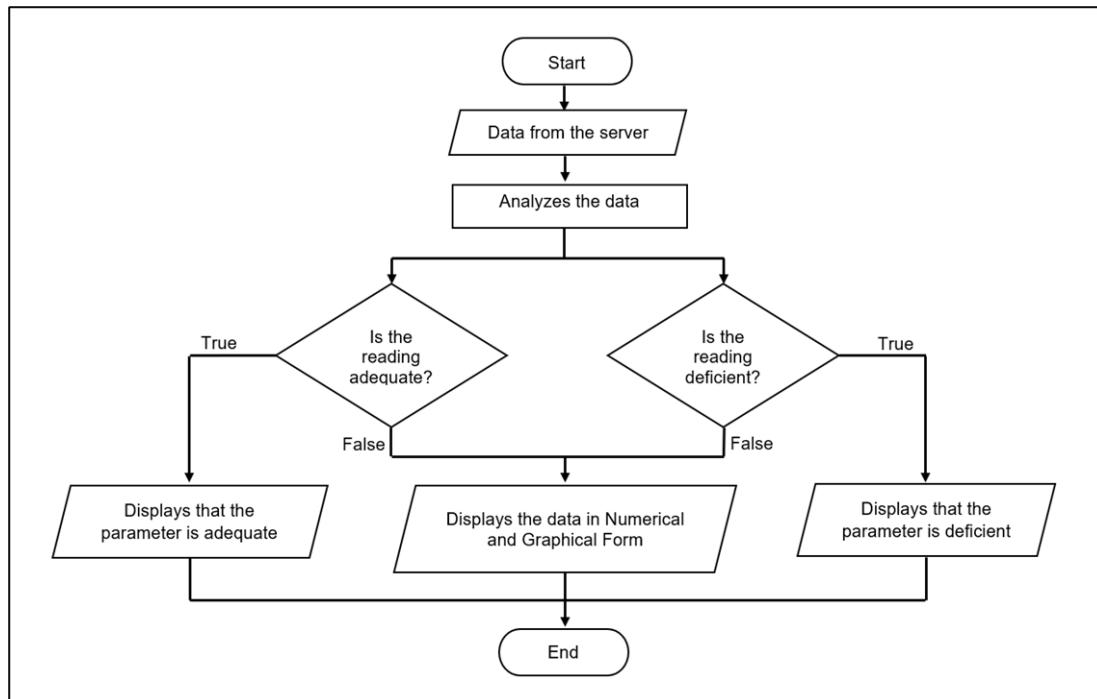


Figure 25. Data Transmission and Processing from Gateway to Monitoring Device

Using the MIT App Inventor, the proponents will develop the monitoring system following the algorithm on the flowchart illustrated above. The sensor data from the network server will undergo a process to determine whether the sensed parameter is adequate, enough, or deficient for the specific crop. Simultaneously, the data will proceed to two conditional statements. If the data exceeds the threshold range of a specific parameter, the application will display the reading and also notify the farmer that the specific parameter is adequate. On the other hand, if the read data falls below the acceptable range of a particular parameter, the data will still be shown on the software, and it will alarm the farmer that the specific parameter is deficient.

3.5.3 Programming the Option System for Plant Requirement Conditions

The data monitoring application for the smart farming system will display the categorized real-time sensor data acquired in numerical and graphical form. Particularly, the main window of the monitoring app will consist of different windows for each node deployed. Each window will show the soil NPK, temperature, moisture, pH, and ambient temperature, humidity, and light intensity readings. As well, the app features an option window where the user can set or assign the crop planted in the area where the soil monitoring node was installed since different crops require different soil and ambience parameters. The crop selected by the user will be the basis of the system's threshold for when the automatic correction system will be actuated to control these essential parameters. Aside from that, the monitoring system is capable of notifying the user whenever drastic changes occur in the soil and ambient conditions, specifically what is deficient and adequate on the farm. Further, the threshold values and correction system calibration were adjustable on the mobile application to also address the desires of the user. Lastly, the user can switch to the different tabs to monitor the other sensor nodes deployed simultaneously.

3.5.4 Establishing Connection Between Server and Mobile Application

On the data server, the API Key and the channel feed URL will be generated and that info will be embedded on the source code of the mobile application. In that way, the monitoring software will be connected to the database capable of accessing the real time sensor data anytime and anywhere if internet connection is available.

3.6 Development of Automatic Correction System through Relays

3.6.1 Materials and Equipment Required

Relay

A relay contains a set of input terminals for single or multiple control signals, as well as a set of working contact terminals.



Figure 26. Relay

Water Pumps

A water pump is a centrifugal water pump that uses a motor to power an impeller that rotates and pushes water outwards.



Figure 27. Water Pumps

Humidifier

The Humidifier warms up the water, and the water evaporates and turns into water vapor, increasing the level of humidity.



Figure 28. Humidifier

3.6.2 Design and Construction of the Feedback mechanism

The design and construction of the feedback mechanism will depend on the construction of the sensor node. A relay will be connected to the microcontroller since it will be utilized in multiple control signals and will be connected to the water pumps and correcting devices. Relay plays an important role in feedback mechanisms because it will serve as a switch that will direct or cut current to the correcting device to control its operation and functionality. Correcting devices will be integrated with the relays as they correct the parameters that are improper for the soil, which may affect the plant growth of the crops. Correcting devices include water pumps, which will be used for the fertigation system whenever soil needs to be fertilized, and a fan and humidifier, which will alter the humidity and temperature inside the greenhouse.

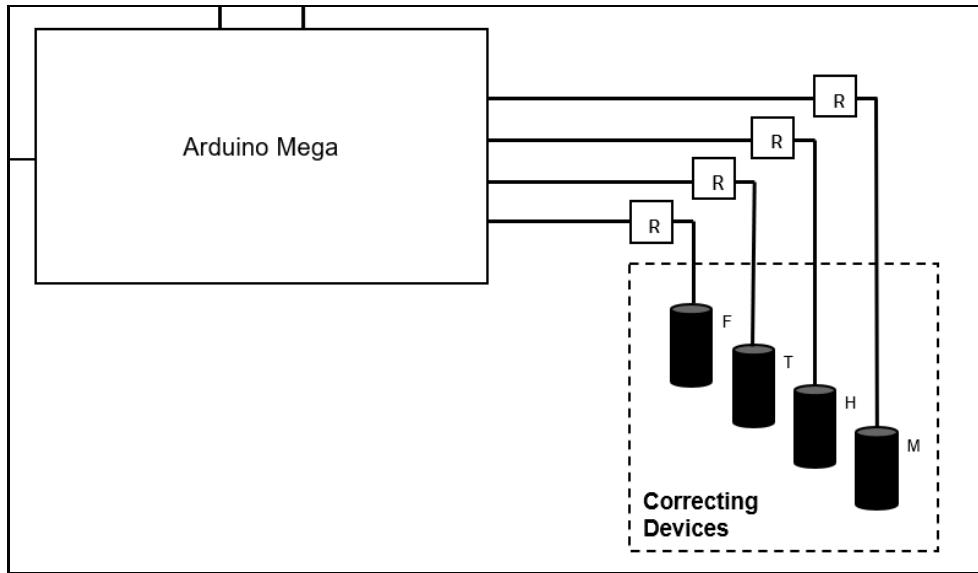


Figure 29. Feedback Mechanism Architecture

3.6.3 Programming the Calibration of the Correction Methodology

A correction method is intended to vary the improperly measured data into corrected data for the soil's parameters. In correcting a parameter, it will need a necessary device such as the water pumps, fan, and humidifier. The software that will be used in programming the correction system is Arduino IDE, as it uses the Arduino as the microcontroller. The program will set a range value for the specific parameter depending on the crops planted on the soil. The range value is intended for the correct data for the specific crops. Fuzzy Logic will be used in the programming of the correcting system as it is used for decision-making. Specifically, whenever data is measured, it will decide if the parameter should be corrected or not. From the sensor, it will continuously measure the parameters, and then the program will decide if the parameter gets above or below a certain value, turn on the correcting devices, or take no action.

3.7 System Architecture

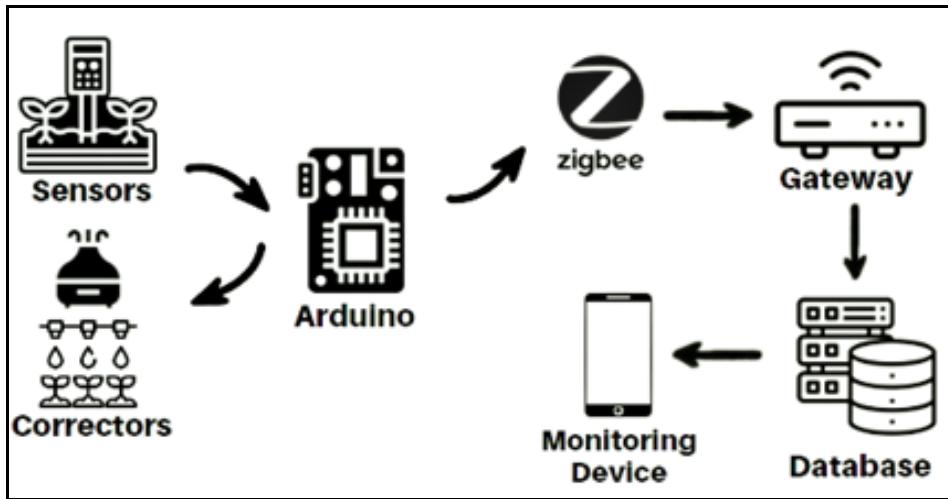


Figure 30. System Architecture

The figure above depicts the overall process of the system. The Arduino is integrated with multiple sensors that gathers data from the soil and transmit it to the gateway through Zigbee protocol that stores to the database and can be monitored to the monitoring application. Furthermore, the Arduino is also integrated with correctors that automatically corrects the crop's threshold parameters.

3.8 System Overall Flowchart

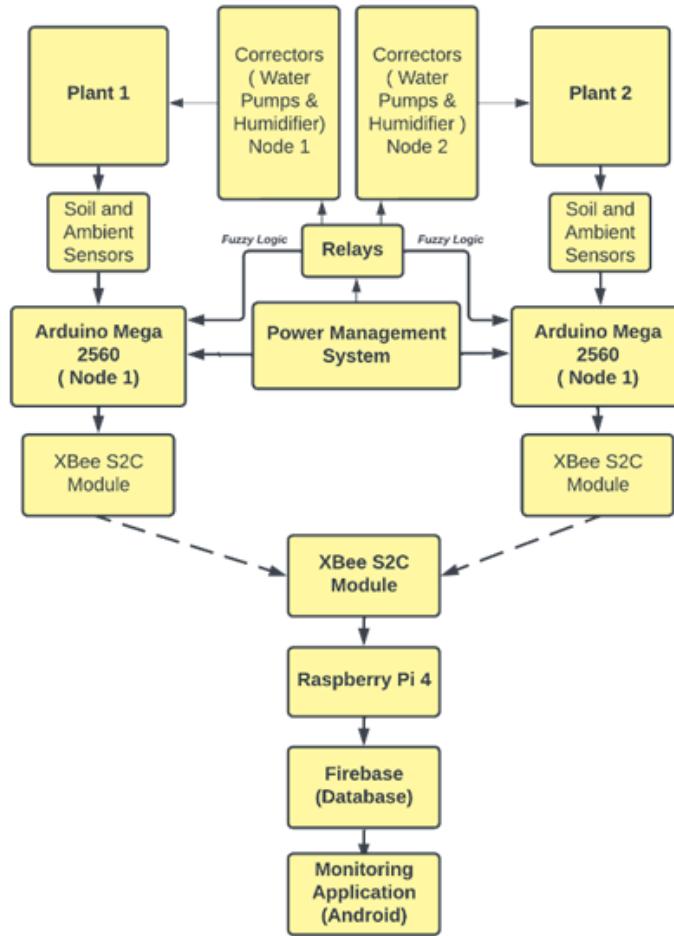


Figure 31. System Flowchart

The system flowchart represents the entire process of the system. The nodes are integrated with soil and ambient sensors that gather data on soil parameters. Additionally, each node is equipped with an XBee S2C Module used to transmit the collected data to the gateway. The gateway stores the data in Firebase, which can be monitored through the monitoring application. The nodes are powered by the power management system, which also supplies power to the relays connecting to the corrective devices. The Fuzzy Logic algorithm was employed to automate the feedback mechanism.

3.9 Testing the Device Overall Operation

3.9.1 Deployment of the WSN

The WSN will be deployed in a controlled area, specifically on a greenhouse. The data gathering and monitoring of plant's growth will happen here as well as the traditional way of farming which involves minimal monitoring and correction on soil parameters. Included on the deployment is the planting of varied plants that was chosen by the proponents. Crops to be planted and its required parameters are shown below:

Table 20. Table of Thresholds for the Crops to be Used

Crops	Soil pH level	Soil Temperature	Nitrogen	Phosphorus	Potassium	Soil Moisture	Air Temperature	Air Humidity
Lettuce	5.4 - 6.7	16 - 24 °C	2.5 - 5	10 - 18	12 - 20	65 - 85%	16 - 26 °C	40 - 70%
Pepper	5-7	18 - 28 °C	2.5 - 5	5 - 10	5 - 10	60 - 80%	21 - 27 °C	50 - 70%
Tomato	4.3 - 5.5	18 - 29 °C	9 - 18	9 - 18	9 - 18	65 - 80%	26 - 29 °C	65 - 85%
Pechay	5.4 - 6.8	20 - 34 °C	7 - 14	7 - 14	7 - 14	55 - 65%	26 - 29 °C	40 - 60%

3.9.2 Monitoring the System's Capabilities, Efficiency, and Maintenance

The system will be monitored for its capabilities and efficiency as the plant's grow. This process will take a long time as the growth of plant's takes weeks or months. Through that timespan, maintenance is a big factor since minimal human intervention should happen for the whole system's automatic process. In monitoring the capabilities, this will include the speed and accuracy of the system on both sensor nodes data transmission and gateway to the mobile application.

3.9.3 Comparing the Accuracy of the System to other Monitoring System/Methods

The data results of the whole system will be compared to other monitoring systems and methods available for soil and ambience parameters. The comparison will occur using statistics and graphical representations to check the accuracy and precision between the study's system and a verified system or methods. With this being the basis of statistics, the computation for percent accuracy and the t test will also occur in this part.

The assessment of accuracy involves comparing the percentage difference obtained from manual parameter testing with the measurements obtained from the system's sensors. The following presents the percentage values representing the permissible error tolerance, which should ideally be close to or lower than the specified values. These acceptance criteria are determined by the researchers based on the parameters being measured by each sensor and the significance of the observed differences and their potential impact.

The data results of the whole system will be compared to other monitoring systems and methods available for soil and ambience parameters. The comparison will occur using statistics and graphical representations to check the accuracy and precision between the study's system and a verified system or methods. With this being the basis of statistics, the computation for percent accuracy and the t test will also occur in this part.

The assessment of accuracy involves comparing the percentage difference obtained from manual parameter testing with the measurements obtained from the system's sensors. The following presents the percentage values representing the permissible error tolerance, which should ideally be close to or lower than the specified values. These acceptance criteria are determined by the researchers based on the parameters being measured by each sensor and the significance of the observed differences and their potential impact.

Table 21. Accepted Percent Error of Sensor Accuracy

Parameters	Percent Error Tolerance
Ambient Temperature	$\leq 5\%$
Ambient Humidity	$\leq 5\%$
Soil Moisture	In Range
Soil Temperature	$\leq 5\%$
pH Level	$\leq 5\%$
Nitrogen	In Range
Phosphorus	In Range
Potassium	In Range

3.9.4 Conducting Expert Evaluation and User Acceptance Test

The proponents decided to use ISO 9001 standards. A product quality evaluation system for software engineering. The system consists of both hardware and software development but with the main quality characteristics involved, for the system as a whole, the evaluation can be described using the ISO 9001. The main quality characteristics identified as Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability in which these can also be broken down into sub categories to be more specific since the system has both hardware and software characteristics.

3.10 Statistical Analysis

In order to determine if the results of the study differ significantly from conventional methods that measure the same parameters, the research will use paired T-test statistical analysis. Similarly, the study will use the same test to identify the effectiveness of the correction system as compared to the traditional method of farming in terms of yield and plant growth.

A t-test statistical analysis is used to determine if there is a significant difference between the means of two groups as well as to identify whether the hypothesis actually improved the subject.

3.11 Project Work Plan

Activities Description	2022												2023				
	MONTHS																
Preparation for Choosing a Topic (Feb)	February	March	April	May	June	July	August	September	October	November	December	January	February	March	April	May	June
Topic Defense (March)																	
Canvassing of Materials																	
Consultation with Advisers																	
Preparing for Title Defense and Draft of Manuscript (June)																	
Designing and Construction of the Sensor Nodes																	
Buying Materials and Equipment needed																	
Programming the Microcontroller																	
Sensor Calibration																	
Designing of Power Management for Sensor Nodes																	
Design and Construction of Gateway																	
Programming the Microprocessor as a Gateway																	
Establishing Secure Connection between WSN and Database																	
Android Application Development																	
Progress Defense (November)																	
Programming the Option System for Plant Requirement Conditions																	
Establishing Connection between Server and Mobile Application																	
Finalization of Monitoring System																	
Pre-Final Defense (January)																	
Design and Construction of the Feedback Mechanism																	
Programming the Calibration of Correction Methodology																	
Deployment (February)																	
Preparation of the Deployment Area and Adjustments of the System																	
Planting Crops																	
Data Gathering (April)																	
Testing and Evaluating of the Project																	
APPRECIATE (May)																	
Finalization of the Manuscript																	
Final Defense and Pitching (June)																	
Finishing the Manuscript and Bookbinding																	

Chapter 4

DATA AND RESULTS

This chapter presents the presentation, analysis of relative findings to tests conducted and interpretation of data.

4.1 Project Technical Description

The study SOIL-N-WAN: WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System is focused on preventing plant growth deficiencies by monitoring soil and ambient parameters with a correction system. It uses a wireless sensor network consisting of sensor stations that monitor real-time embedded transmitters and a gateway.

In the sensor stations, it monitors different parameters such as Light intensity, moisture, Ambient temperature, pH, humidity, electrical conductivity, and NPK (Nitrogen, Phosphorus, Potassium). The following sensors are to be used: soil moisture sensor, Soil pH sensor, Soil temperature sensor, Soil EC sensor, BH1750 light intensity sensor, DHT11, Soil NPK sensor, all of which will be programmed and controlled by an Arduino Mega 2560.

The proponents used Raspberry Pi 4 Model B and Xbee S2C as microprocessors; both components are established by different software: Thony IDE for Raspberry Pi 4 Model B and XCTU for Xbee S2C. The gateway will be connected to the internet to monitor the data gathered via mobile applications. The mobile application was established by an MIT app inventor and is used for data monitoring and plant selection on the basis of parameters utilized on the correcting system.

4.2 Project Structural Organization

4.2.1 Parts of the System

The system is divided into four different stations to gather the data for the mobile application. The first station is the sensor node, which consists of seven different sensors that can measure up to 10 parameters, mainly soil and ambient conditions. The data gathered from the sensor nodes will be transmitted to the database through the gateway that serves as the medium for the data. The data from the database will then be transmitted to the mobile application.

4.2.1.1 Sensor Nodes

The purpose of the sensor nodes is to measure data from the soil and ambient condition. It uses an Arduino Mega as the microcontroller, the sensors that are integrated to the arduino are DHT11 (Humidity), DS18B20 (Soil Temperature), BH1750 (Light Intensity), Soil Moisture, NPK, EC, and pH sensors.

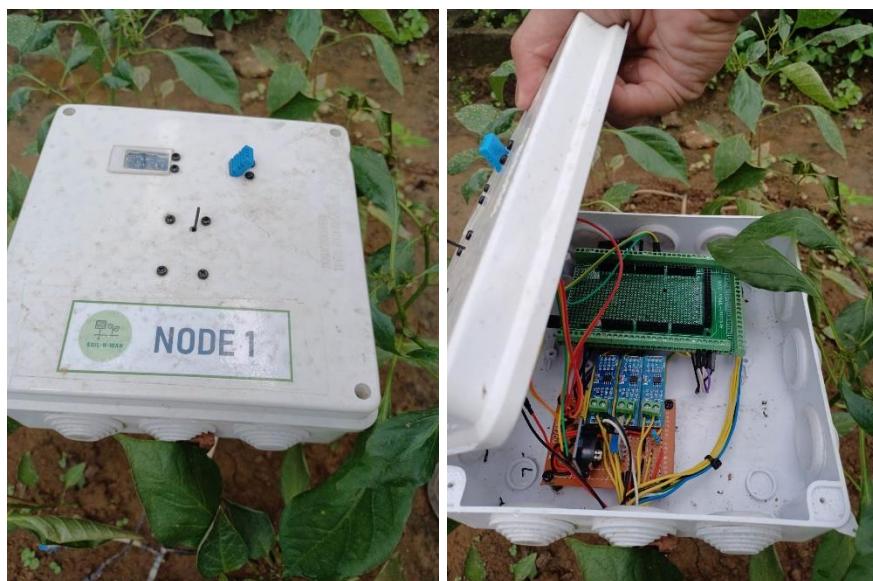


Figure 32. Sensor Node Design

4.2.1.2 Gateway

The gateway receives the data gathered from the sensor nodes using the Raspberry Pi 4 and XBee S2C. The measured data will then be transmitted to the database to store and have the records of the data.



Figure 33. Gateway

4.2.1.3 Power Management

The power management system consists of a battery, a solar panel, a solar charge controller, a buck converter, and a boost converter. The battery serves as the main power source for the sensor nodes and the correction system. The battery can be charged using the solar panel with the help of the solar charge controller. The buck converter is used to decrease the input voltage to 9V for the Arduino Mega. The boost converter is used to increase the input voltage to 24V for the humidifier.

and correction device. The load on the solar charge controller is 12V for the relays that are connected to the other correction devices.

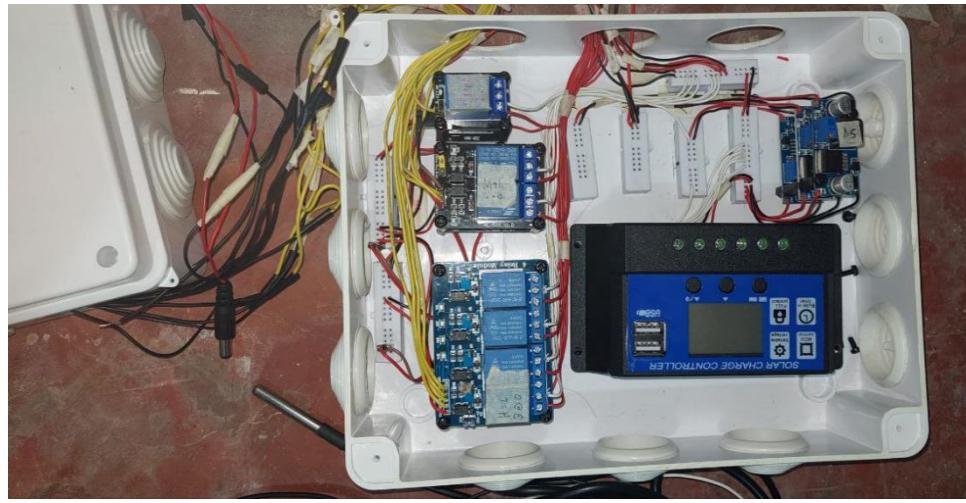


Figure 34. Power Management and Relays

4.2.1.4 Fuzzy Logic Algorithm-based Correction System

The correction system involves an algorithm and relays. The fuzzy logic algorithm is used for the application of relays. The parameters and timing are predefined by the proponents, and if a certain range is met, the timing output of the relays to power the correctors will be dependent. The relays are connected to the sensor nodes for signals and correctors such as water pumps and humidifiers through a wired connection.

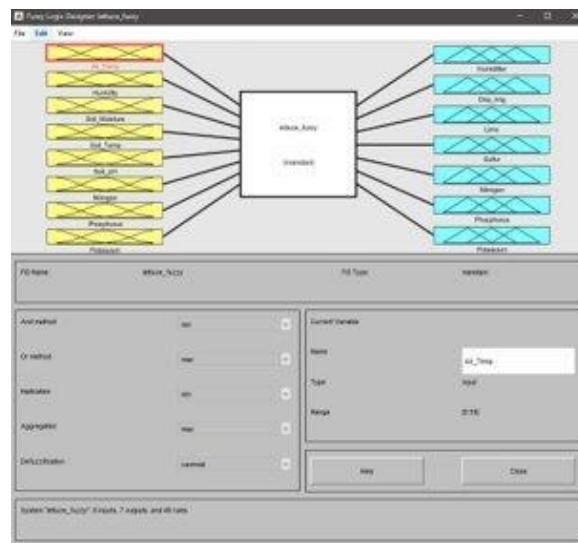


Figure 35. Fuzzy Logic Designer using MATLAB

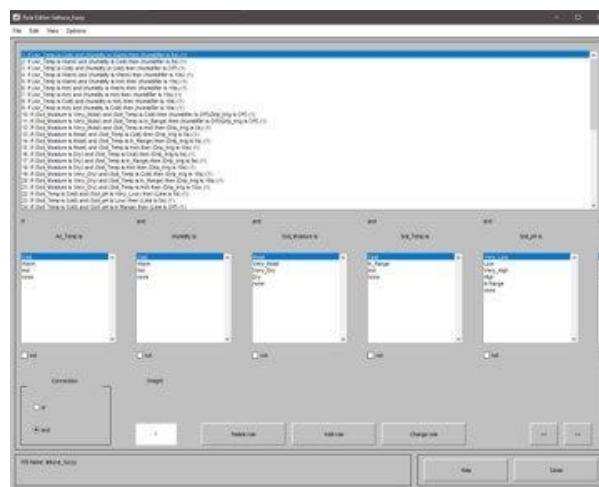


Figure 36. Rule Editor

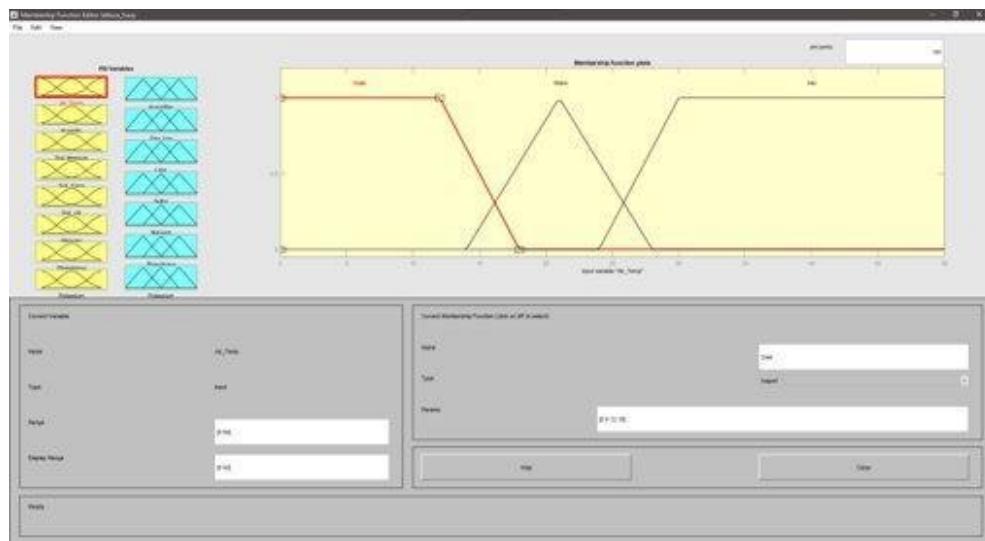


Figure 37. Membership Function Editor

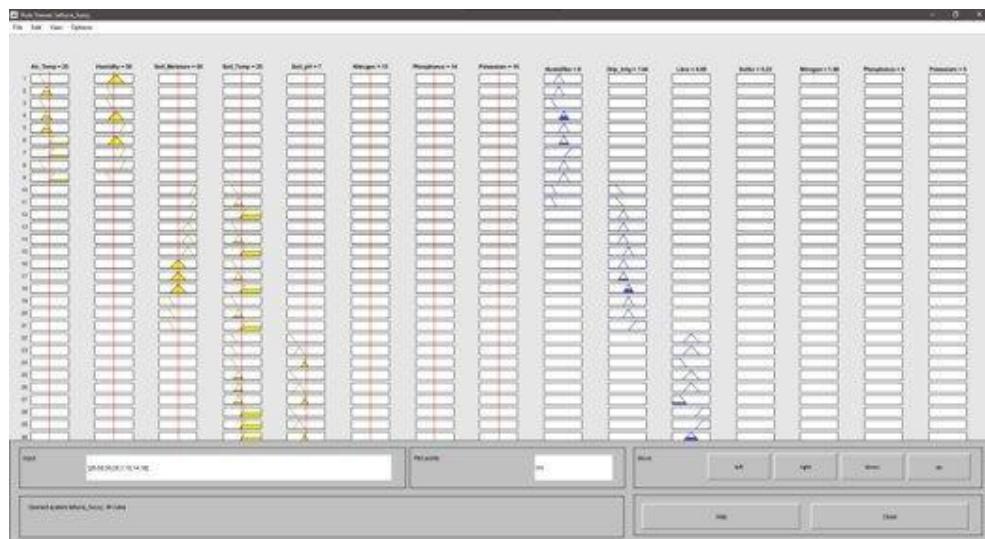
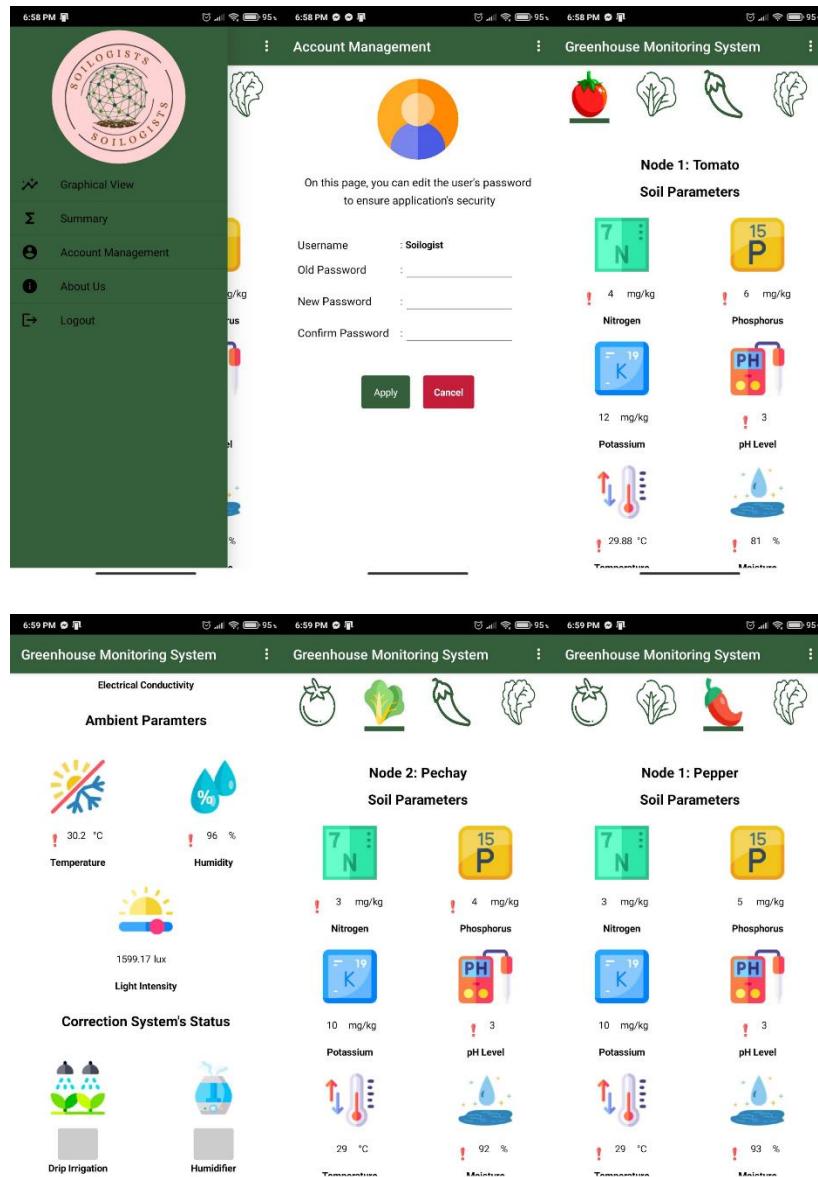


Figure 38. Rule Viewer of Fuzzy Logic using MATLAB

4.2.1.5 Monitoring Application

The system includes a monitoring application for the parameters which is connected to the Firebase Database. The data transferred are split for the two nodes in which it can be monitored through the application simultaneously. This data is secured in the gateway to ensure that the application shows the correct data for the

nodes. It will also include a graphical representation of the data that shows the changes of parameters through time.



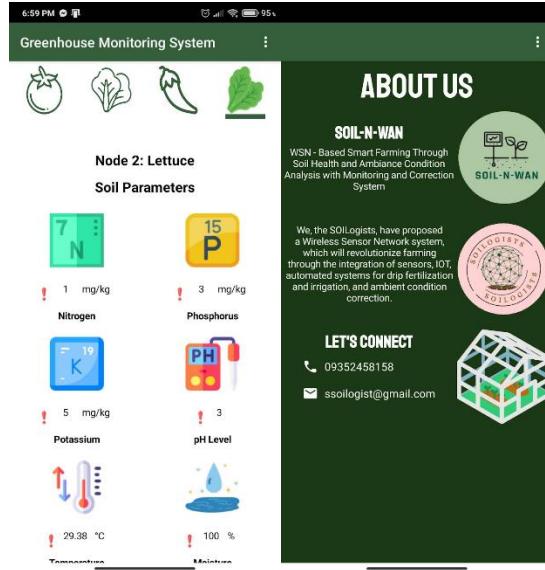


Figure 39. Graphical User Interface of Monitoring Application

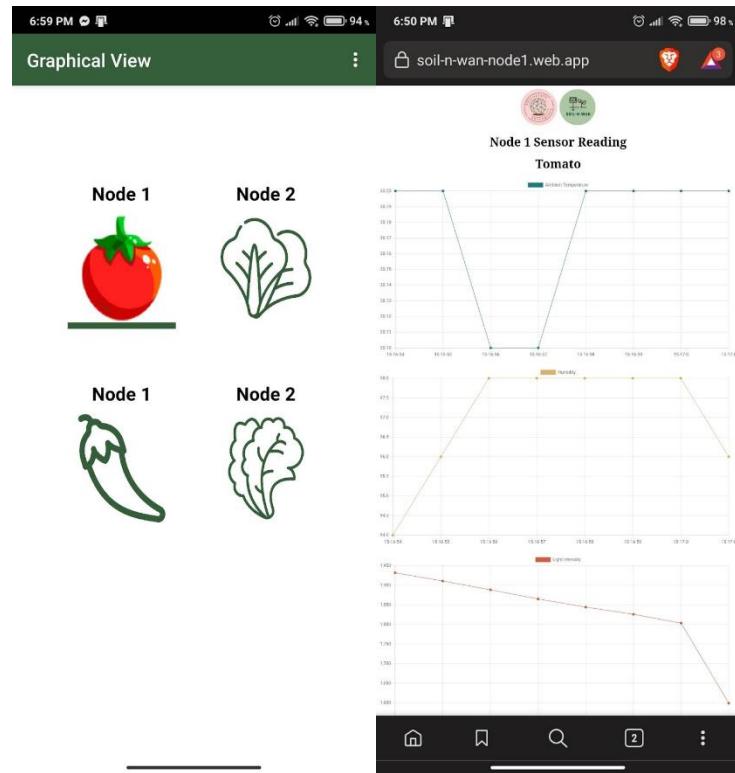


Figure 40. Graphical View of the Parameters Via the Monitoring Application

4.2.1.6 Deployment Area

The deployment of the system consists of a greenhouse in which 4 grow beds are involved with varied plants. Since there are only two nodes, after weeks of monitoring, the sensor nodes will be transferred to another grow bed for a new range of parameters and codes. The deployment area is located at Brgy. Bagong Pag-aso Urban Farm at Road 9, Bagong Pag-aso, Quezon City.



Figure 41. Representation of Actual Device in Greenhouse



Figure 42. Deployment Area



Figure 43. Full Deployment of the System



Figure 44. Node 1 Used for Tomato Plant



Figure 45. Node 2 Used for Pechay Plant



Figure 46. Water Tanks for Correction System



Figure 47. Solar Panel and Power Management System



Figure 48. Uncontrolled Group of Plants for Conventional Farming



Figure 49. Node 1 Used for Pepper Plant



Figure 50. Node 2 Used for Lettuce Plant



Figure 51. Mr. Lorenzo Jacinto, a Registered Professional Agriculturist,
Representing as the Expert for the Project Study



Figure 52. Deployment Site Visit with Room 2 Panel Chair (Engr. John Peter Ramos), Research Adviser (Engr. Gilfred Allen Madrigal), Research Extension Coordinator (Engr. Mark Melgrito)

4.3 Project Limitation and Capabilities

The system consists of two nodes and one gateway, with a correction system embedded in the sensor nodes. It can monitor up to 10 parameters, which are Humidity, Ambient Temperature, Light Intensity, Soil Moisture, Soil Temperature, Soil pH, Soil Electrical conductivity, and NPK (Nitrogen, Phosphorus, Potassium). With the transmission of Serial communication using XBee S2C, the two nodes are securely transmitted to the necessary classification. The system has an automatic correction system based on the Fuzzy Logic Algorithm. All in all, the system is designed to be power-sustainable with the use of a solar panel and a high-capacity battery.

The main limitation of the system is the wired connection of power and signals for relays. The proponents used a single battery with a power management system to energize all of the

systems, including the sensor nodes and correction system. For the monitoring application, the plant selection system is limited to four options: lettuce, tomato, pechay, and pepper, which are the most commonly cultivated plants in a greenhouse setting. The monitoring app also doesn't have the function of manually controlling the sensor node's power and parameter threshold values. Since the correction system is based on Fuzzy Logic, it is not possible to switch a sensor node's threshold from one plant to another through the app because the codes need to be changed and hardcoded. In regards to the correction system, its feature is to activate when the parameters are lacking in value, but when a sensor value is above the threshold, the correction system does not lower that specific value as its function is to add.

After formulating the hypotheses, the uncontrolled and controlled groups were subjected to paired t-tests to examine the mean differences, leading to different conclusions for each plant type.

4.4 Project Evaluation

4.4.1 Sensor Accuracy

After the calibration of the sensors per node, the proponents found out that the percent error between the manual and the sensor is not more than 10%, as the sensor gives data that is not much less or greater than the calibration tool. However, the calibration tool for moisture is set only with wet, dry, and normal as it is only compared by range, just like the NPK. The testing for the sensors and the calibration tool is set at at least 10 minutes as the interval, with a total of 10 tests per sensor and a 15-minute interval for each NPK sensor.

Table 22. Ambient Temperature Sensor Accuracy

Ambient Temperature	Sensor Node 1				Sensor Node 2			
	Time	Digital Temperature Meter	DHT11	Percent Error	Time	Digital Temperature Meter	DHT11	Percent Error
Test 1	8:00 pm	29.2	28.70	1.71 %	8:00 pm	28.4	29.20	2.82 %
Test 2	8:10 pm	29.4	29	1.36 %	8:10 pm	28.7	28.8	0.35 %
Test 3	8:20 pm	29.3	28.70	2.05 %	8:20 pm	28.7	28.8	0.35 %
Test 4	8:30 pm	29.1	29.1	0 %	8:30 pm	28.6	29.20	2.1 %
Test 5	8:40 pm	29.1	28.6	1.72 %	8:40 pm	28.5	29.20	2.45 %
Test 6	8:50 pm	29	28.5	1.72 %	8:50 pm	28.5	28.3	0.70%
Test 7	9:00 pm	28.9	28.40	1.73 %	9:00 pm	29.1	29.2	0.34%
Test 8	9:10 pm	28.9	28.5	1.38 %	9:10 pm	29.1	28.6	1.72 %
Test 9	9:20 pm	28.9	28.4	1.73 %	9:20 pm	29.1	28.6	1.72 %
Test 10	9:30 pm	28.9	28.30	2.08 %	9:30 pm	29.3	28.70	2.05 %

The tests conducted above represent the percent error in the calibration of each node. Node 1 exhibits an average percent error of 1.55%, while Node 2 shows an average percent error of 1.46%. The percentage difference between the manual testing and the measurements obtained from the system's sensors falls within the predefined acceptable percentage.

Table 23. Ambient Humidity Sensor Accuracy

Ambient Humidity	Sensor Node 1				Sensor Node 2			
	Time	Digital Humidity Meter	DHT11	Percent Error	Time	Digital Humidity Meter	DHT11	Percent Error
Test 1	3:09 pm	56	58	3.57 %	3:09 pm	58	59	1.72 %
Test 2	3:19 pm	58	59	1.72 %	3:19 pm	59	60	1.69 %
Test 3	3:29 pm	59	60	1.69 %	3:29 pm	61	60	1.64 %
Test 4	3:39 pm	57	60	5.26 %	3:39 pm	58	58	0 %
Test 5	3:49 pm	59	59	0 %	3:49 pm	29	29.75	2.59 %
Test 6	3:59 pm	58	58	0 %	3:59 pm	58	58	0 %
Test 7	4:09 pm	57	58	1.75 %	4:09 pm	59	59	0 %
Test 8	4:19 pm	59	58	1.69 %	4:19 pm	59	59	0 %
Test 9	4:29 pm	59	58	1.69 %	4:29 pm	58	58	0 %
Test 10	4:39 pm	58	60	3.45 %	4:39 pm	59	60	1.69 %

The table above presents the calibration results of ambient humidity sensors obtained through ten tests conducted at ten-minute intervals. The percent error of each test was calculated. Additionally, the average percent error for sensor node 1 is 2.082%, while for sensor node 2, it is 0.93%. The percentage difference between the manual testing and the measurements obtained from the system's sensors falls within the predefined acceptable percentage.

Table 24. Soil Moisture Sensor Accuracy

Soil Moisture	Sensor Node 1				Sensor Node 2			
	Time	4 in 1 Soil Tester	Soil Moisture Sensor	In Range	Time	4 in 1 Soil Tester	Soil Moisture Sensor	In Range
Test 1	1:00 pm	70-100%	87%	✓	1:00 pm	30-69%	58%	✓
Test 2	1:10 pm	70-100%	100%	✓	1:10 pm	30-69%	67%	✓
Test 3	1:20 pm	70-100%	83%	✓	1:20 pm	30-69%	54%	✓
Test 4	1:30 pm	70-100%	83%	✓	1:30 pm	30-69%	63%	✓
Test 5	1:40 pm	70-100%	82%	✓	1:40 pm	30-69%	46%	✓
Test 6	1:50 pm	70-100%	81	✓	1:50 pm	30-69%	38%	✓
Test 7	2:00 pm	70-100%	81%	✓	2:00 pm	30-69%	44%	✓
Test 8	2:10 pm	70-100%	81%	✓	2:10 pm	30-69%	51%	✓
Test 9	2:20 pm	70-100%	79%	✓	2:20 pm	30-69%	47%	✓
Test 10	2:30 pm	70-100%	80%	✓	2:30 pm	30-69%	48%	✓

As shown in table above, it indicates the calibration results of soil moisture that was acquired through ten different testing conducted at a ten-minutes interval. All of the test calibrations of soil moisture for both sensor node 1 and sensor node 2 are in range. The percentage difference between the manual testing and the measurements obtained from the system's sensors falls within the predefined acceptable range.

Table 25. Soil Temperature Sensor Accuracy

Soil Temperature	Sensor Node 1				Sensor Node 2			
	Time	Digital Thermometer	DS18B20	Percent Error	Time	Digital Thermometer	DS18B20	Percent Error
Test 1	4:42 pm	29	28.06	3.24 %	4:42 pm	29	27.50	5.17 %
Test 2	4:52 pm	29	27.44	5.38 %	4:52 pm	29	27.69	4.52 %
Test 3	5:02 pm	30	27.88	7.07 %	5:02 pm	29	27.38	5.59 %
Test 4	5:12 pm	29	27.44	5.38 %	5:12 pm	29	28.06	3.24 %
Test 5	5:22 pm	28	27.25	2.68 %	5:22 pm	29	28.31	2.38 %
Test 6	5:32 pm	30	27.65	7.83 %	5:32 pm	29	28.31	2.38 %
Test 7	5:42 pm	29	28	3.45 %	5:42 pm	29	27.50	5.17 %
Test 8	5:52 pm	30	28.06	6.45 %	5:52 pm	28	27.25	2.68 %
Test 9	6:02 pm	30	28.31	5.63 %	6:02 pm	30	27.65	7.83 %
Test 10	6:12 pm	29	28.50	1.72 %	6:12 pm	30	27.88	7.07 %

After conducting ten tests, the sensor calibration reveals an average percent error of 4.88% for node 1 and 4.6% for node 2. Hence, the accuracy of each sensor node is close to the actual instrument. The percentage difference between the manual testing and the measurements obtained from the system's sensors falls within the predefined acceptable percentage.

Table 26. pH Level Sensor Accuracy

pH Level	Sensor Node 1				Sensor Node 2			
	Time	4 in 1 Soil Tester	pH Sensor	Percent Error	Time	4 in 1 Soil Tester	pH Sensor	Percent Error
Test 1	6:15 pm	6	5.50	8.33 %	6:15 pm	6.5	6.1	6.15 %
Test 2	6:25 pm	6.5	6.1	6.15 %	6:25 pm	6.5	6.80	4.61 %
Test 3	6:35 pm	5.5	5.0	9.09 %	6:35 pm	6	5.50	8.33 %
Test 4	6:45 pm	6	5.50	8.33 %	6:45 pm	5.5	5.0	9.09 %
Test 5	6:55 pm	6.5	6.80	4.61 %	6:55 pm	6	5.6	6.67 %
Test 6	7:05 pm	6	6.50	8.33 %	7:05 pm	6	5	20 %
Test 7	7:15 pm	6	5	20 %	7:15 pm	5.5	5.5	0 %
Test 8	7:25 pm	5.5	5.2	5.45 %	7:25 pm	5.5	5.5	0 %
Test 9	7:35 pm	5.5	5.5	0 %	7:35 pm	5.5	5.10	7.27 %
Test 10	7:45 pm	5.5	5.10	7.27 %	7:45 pm	5.5	5.0	9.09 %

The table above provides the results of ten test calibrations performed by pH level at intervals of ten minutes. In comparison to the sensor node 2, sensor node 1 shows an

average percent error of 7.76% whereas sensor node 2 shows an average percent error of 7.12%. The percentage difference between the manual testing and the measurements obtained from the system's sensors did not fall within the predefined acceptable percentage.

Table 27. Nitrogen Sensor Accuracy

Nitrogen	Sensor Node 1				Sensor Node 2			
	Time	Soil Test Kit	NPK Sensor	In Range	Time	Soil Test Kit	NPK Sensor	In Range
Test 1	8: 56 pm	0- 9	2	✓	8: 56 pm	0- 9	1	✓
Test 2	9:11 pm	0- 9	1	✓	9:11 pm	0- 9	1	✓
Test 3	9:26 pm	0- 9	2	✓	9:26 pm	0- 9	2	✓
Test 4	9:41 pm	0- 9	0	✓	9:41 pm	0- 9	1	✓
Test 5	9:56 pm	0- 9	1	✓	9:56 pm	0 - 9	1	✓
Test 6	10:11 pm	0- 9	1	✓	10:11 pm	0- 9	1	✓
Test 7	10:26 pm	0- 9	0	✓	10:26 pm	0- 9	0	✓
Test 8	10:41 pm	0- 9	1	✓	10:41 pm	0- 9	1	✓
Test 9	10:56 pm	0- 9	1	✓	10:56 pm	0- 9	1	✓
Test 10	11:11 pm	0- 9	0	✓	11:11 pm	0- 9	1	✓

The accuracy of each sensor node's NPK sensor was tested by comparing the output of the actual instrument with the measured data. The data indicates that both sensors were within the expected range, in comparison to the manual testing of nitrogen.

Table 28. Phosphorus Sensor Accuracy

Phosphorus	Sensor Node 1				Sensor Node 2			
	Time	Soil Test Kit	NPK Sensor	In Range	Time	Soil Test Kit	NPK Sensor	In Range
Test 1	8: 56 pm	0 - 19	2	✓	8: 56 pm	0 - 19	2	✓
Test 2	9:11 pm	0 - 19	2	✓	9:11 pm	0 - 19	2	✓
Test 3	9:26 pm	0 - 19	2	✓	9:26 pm	0 - 19	2	✓
Test 4	9:41 pm	0 - 19	1	✓	9:41 pm	0 - 19	1	✓
Test 5	9:56 pm	0 - 19	1	✓	9:56 pm	0 - 19	2	✓
Test 6	10:11 pm	0 - 19	1	✓	10:11 pm	0 - 19	2	✓
Test 7	10:26 pm	0 - 19	0	✓	10:26 pm	0 - 19	1	✓
Test 8	10:41 pm	0 - 19	1	✓	10:41 pm	0 - 19	1	✓
Test 9	10:56 pm	0 - 19	1	✓	10:56 pm	0 - 19	2	✓
Test 10	11:11 pm	0 - 19	1	✓	11:11 pm	0 - 19	2	✓

Justified above was the accuracy of sensor node 1 and sensor node 2 in terms of measuring the amount of Phosphorus in soil in the unit of mg/kg. Given that the same sample of soil was analyzed on each test, all the sensor readings on both nodes fell in the range of results of conventionally acquiring Phosphorus content on soil.

Table 29. Potassium Sensor Accuracy

Potassium	Sensor Node 1				Sensor Node 2			
	Time	Soil Test Kit	NPK Sensor	In Range	Time	Soil Test Kit	NPK Sensor	In Range
Test 1	8:56 pm	0 - 19	6	✓	8: 56 pm	0 - 19	4	✓
Test 2	9:11 pm	0 - 19	6	✓	9:11 pm	0 - 19	5	✓
Test 3	9:26 pm	0 - 19	5	✓	9:26 pm	0 - 19	5	✓
Test 4	9:41 pm	0 - 19	2	✓	9:41 pm	0 - 19	3	✓
Test 5	9:56 pm	0 - 19	3	✓	9:56 pm	0 - 19	4	✓
Test 6	10:11 pm	0 - 19	3	✓	10:11 pm	0 - 19	5	✓
Test 7	10:26 pm	0 - 19	2	✓	10:26 pm	0 - 19	2	✓
Test 8	10:41 pm	0 - 19	4	✓	10:41 pm	0 - 19	3	✓
Test 9	10:56 pm	0 - 19	3	✓	10:56 pm	0 - 19	4	✓
Test 10	11:11 pm	0 - 19	3	✓	11:11 pm	0 - 19	3	✓

The accuracy of the Potassium sensors on both nodes were tested through comparison with results of the conventional testing. All of the sensor outputs were in range of the manual testing results justifying that the sensors were accurate in detecting the Phosphorus content of Soil.

4.4.2 Power Management

4.4.2.1 Energy Consumption Table and Computation

Table 30. Energy Consumption

Device	Quantity	Voltage	Current (mA)	Power (W)	Time Active (s)	Time Interval (h)	Energy Consumption per Day (Wh)
Arduino Mega	1	9	40	360×10^{-3}	86400	24	8.640
Max485	3	5	0.5	8×10^{-3}	86400	24	0.180
DS3231 Timer	1	3	0.2	1×10^{-3}	86400	24	0.016
Xbee S2C	1	3	31	102×10^{-3}	86400	24	2.455
Ambient Humidity and Temperature Sensor	1	4	0.3	1×10^{-3}	86400	24	0.025
Light Intensity Sensor	1	3	0.12	360×10^{-6}	86400	24	0.009
Soil Temperature Sensor	1	3	1.5	5×10^{-3}	86400	24	0.119

Soil Moisture Sensor	1	3	15	50×10^{-3}	86400	24	1.188
pH Sensor	1	12	20	240×10^{-3}	86400	24	5.760
EC Sensor	1	12	20	240×10^{-3}	86400	24	5.760
NPK Sensor	1	12	20	240×10^{-3}	86400	24	5.760
Water Pump	1	12	1000	12	60	24	0.200
Nutrient Pump	5	12	1000	60	60	24	1.000
Electric Solenoid Valve	6	12	400	28.8	60	24	0.480
Fan	1	12	200	2.4	180	24	0.120
Humidifier	1	24	1000	24	180	24	1.200
Total							32.912 Wh

Energy generated per day (theoretical) = $100W * 24\text{ hours} = 2.4\text{kWh}$

Energy consumed per day (theoretical) = **32.912 Wh**

Total energy consumed per day (2 Nodes) = **65.823 Wh**

Energy efficient per day = $[(\text{energy generated per day} - \text{energy consumed per day}) / \text{energy generated per day}] * 100$

Energy efficiency per day = $[(2.4 \text{ kWh} - 32.912 \text{ Wh}) / 2.4 \text{ kWh}] * 100 = \underline{\underline{98.63\%}}$

Table 31. Power Readings of the Device

Battery Operated			
Time	Voltage (V)	Current (A)	Power (W)
10:42 am	11.7	0.4	4.68
10:52 am	11.6	0.39	4.52
11:02 am	11.6	0.4	4.64
11:12 am	11.5	0.39	4.49
11:22 am	11.5	0.42	4.83
11:32 am	11.7	0.41	4.80
11:42 am	11.7	0.41	4.80

4.4.2.2 Battery and Solar Panel Specifications

Battery Voltage = 12 V

Battery Amp Hours (Capacity) = 100Ah

Battery Type = Lead Acid

Battery Charge Efficiency = 85%

Battery Depth of Discharge (DoD) = 25%

Absorption Charging of Lead Acid Battery = 1.176 hours

Solar Panel Wattage = 100 W

Solar Charge Controller Type = PWM

Charge Controller Efficiency = 75%

System Loses = 20%

4.4.2.3 Computation of Estimated Charge Time

1. To estimate the maximum charge current output by the solar charge controller, divide the solar panel wattage by the battery voltage:

$$100 \text{ W} / 12 \text{ V} = 8.33 \text{ A}$$

2. Multiply current by rule-of-thumb system losses and charge controller efficiency.

$$8.33 (1 - 0.2)(0.75) = 5 \text{ A}$$

3. Divide battery capacity by its battery charge efficiency.

$$100 \text{ Ah} / 0.85 = 117.65 \text{ Ah}$$

4. Divide the solved battery capacity by current to measure how long it would take to charge the entire battery.

$$117.65 \text{ Ah} / 5 \text{ A} = 23.53 \text{ hours}$$

5. Multiply the charge time by the battery's DOP or depth of discharge to measure how long it would take to charge the battery at its current level (from 70% level to 100% level).

$$23.53(0.3) = \mathbf{7 \text{ hours}}$$

6. To account for the Lead Acid battery and PWM charge controller absorption charging stage, add time.

$$7 + 1.176 = \mathbf{8.176 \text{ hours}}$$

4.4.2.4 Computation of Discharge Battery Percentage

Average Power Reading = 4.68 W

Sun Hours = 5 hours, 19 hours of continuous discharging

- Multiply t=energy efficiency, battery capacity and battery voltage. Then divide them by the load (power readings).

$$0.9863 \left(\frac{100 \times 12}{4.68} \right) = 252.90 \text{ hours}$$

- Divide 100% by its hours to drain the battery to get the decreasing percentage per hour.

$$\left(\frac{100\%}{252.90 \text{ hours}} \right) = 0.40\% / \text{hour}$$

- Multiply decreasing percentage and the hours of continuous discharging.

$$0.40 \frac{\%}{\text{hour}} (19 \text{ hours}) = 7.6 \%$$

The previous tables and computations justify that the combination of a 100 Watts Solar Panel and a 100 Ah lead acid battery is sufficient for the system to operate continuously given that only the sensors, microcontrollers, and Xbee modules are working most of the time. Meanwhile, in consideration of the minimum charging time to fully energize the battery, it is recommended to utilize a higher-power solar panel to ensure that the battery has sufficient stored energy for the system even in cloudy and unfavorable weather conditions. Upgrading the battery is also advised when the system is utilized on larger farms.

4.4.2.5 Computation of Estimated Battery Life at Full load

$$B_l = \frac{B_l}{B_c}$$

Where, B_l = Battery life

B_c = Battery Capacity

I_l = Load Current

$$B_l = \frac{100 \text{ Ah}}{19.499 \text{ A}} = 5.128 \text{ hours}$$

At full load, without a solar panel energizing the battery, the supply itself can power the system continuously for 5.128 hours.

4.4.3 Calculated Values and Measured Parameters for Different Plants

4.4.3.1 Tomato

The results are based on the measured values in terms of plant's height and number of leaves for both controlled and uncontrolled groups for the Tomato plants. Paired differences are gathered and subjected to two-tailed t-tests. For the hypothesis testing, two variables are considered. Hence, two hypotheses are formulated.

Ho: Controlled = Uncontrolled (There is no significant difference between the use of the system and conventional planting in terms of growth on number of leaves)

Ha: Controlled ≠ Uncontrolled (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% $\quad df = 9$

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth of the plant's height)

Ha: **Controlled ≠ Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% $df = 9$

Table 32. Calculated Values for Tomato

Variable	Number of Sample	MEAN Difference	Standard Deviation	t-value	Two Tailed paired t-test				
					df = 9				
					$\alpha = 5\%$				
					Critical t- value				
No. of Leaves	10	-0.038	1.463	-0.081	2.262				
Height	10	0.524	0.379	9.417					

Using the calculated values for table 9, for the Number of Leaves, with the t-value having a lower value compared to the Critical t-value, it is concluded that the null hypothesis is subjected to fail to reject. As for the Plant's Height, as t-value has a higher value than the Critical t-value, it is concluded that the null hypothesis is rejected.

4.4.3.1.1 Measured Values for Tomato

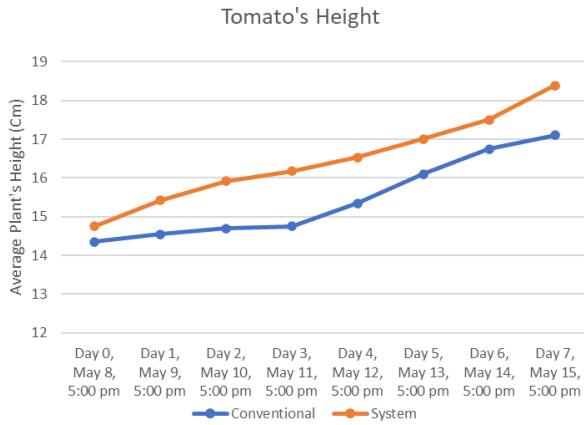


Figure 53. Tomato's Height

As shown in the figure above, on the first day, both the conventional and the system have almost the same height. However, as days pass, the height varies, and the controlled crops are significantly taller than the conventional ones. As a result, after seven days of surveillance, tomato grew at a higher rate on the controlled setup with 24.68% while tomato on conventional setup grew at 19.16%. This is supported by [22], which states that correcting soil nutrients, specifically NPK with inorganic fertilizers, will result in accelerated plant growth due to the water solubility of the nutrients.

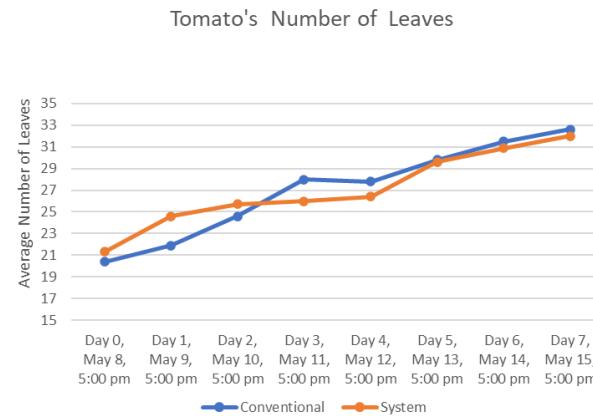


Figure 54. Tomato's Number of Leaves

The figure above displays the graphical representation of the number of tomato leaves. The data shows both increases and decreases in leaf count for both conventional and controlled plants. Furthermore, the decrease in the number of leaves is attributed to a viral infection and wilting, as confirmed by our consultation with an expert.

4.4.3.1.2 Measured Values Parameters Measured for Tomato

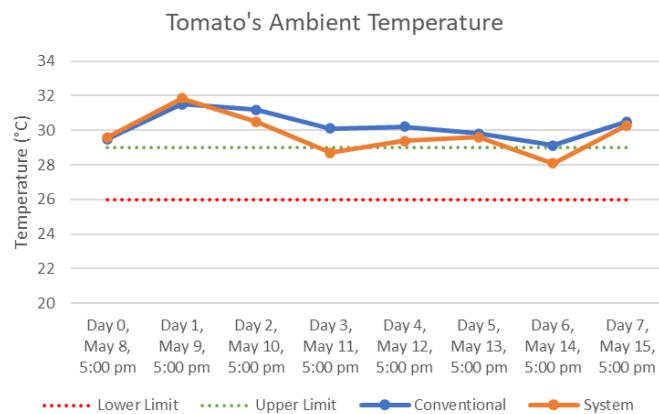


Figure 55. Tomato's Ambient Temperature

The graph above shows the measured value of ambient temperature of conventional and with the system for 7 days. It shows that the data is still in the threshold for tomatoes.

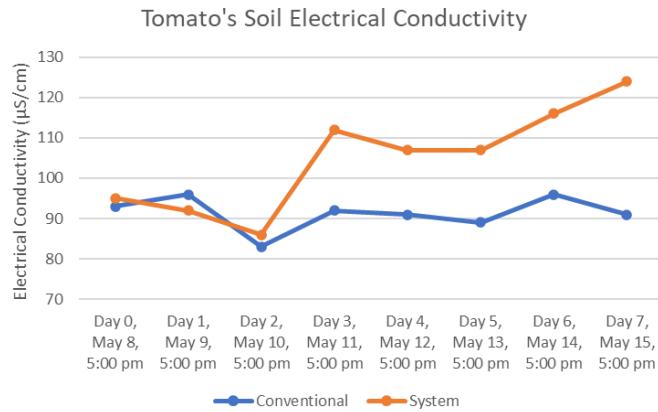


Figure 56. Tomato's Soil Electrical Conductivity

The graphical representation above shows the different measured value in electrical conductivity for both conventional and with the system. The controlled ones are high due to the soil used and since most of the soil used for conventional is loam soil.

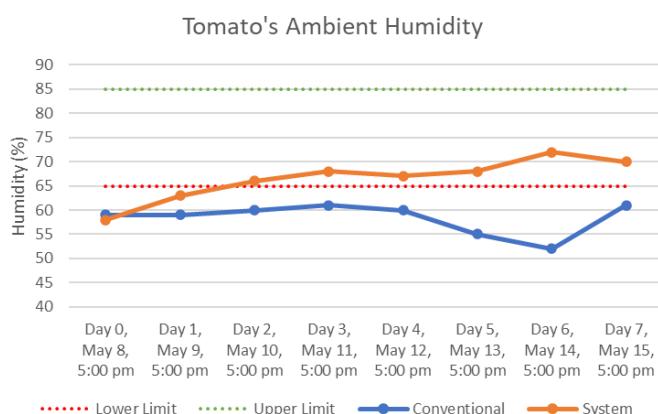


Figure 57. Tomato's Ambient Humidity

As the figure shown above, the measured data of humidity is on the threshold of the tomato since it has a correction device which is the humidifier.

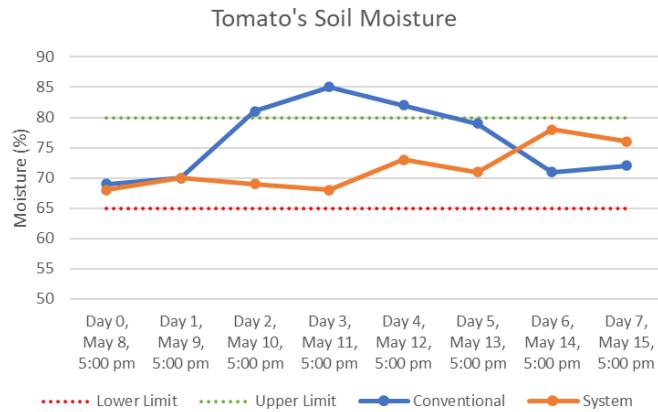


Figure 58. Tomato's Soil Moisture

The figure above displays the measured value of soil moisture, which remains within the threshold range due to daily watering. The controlled plants show greater effectiveness from day 1 to day 7, as their values consistently stay within the threshold.

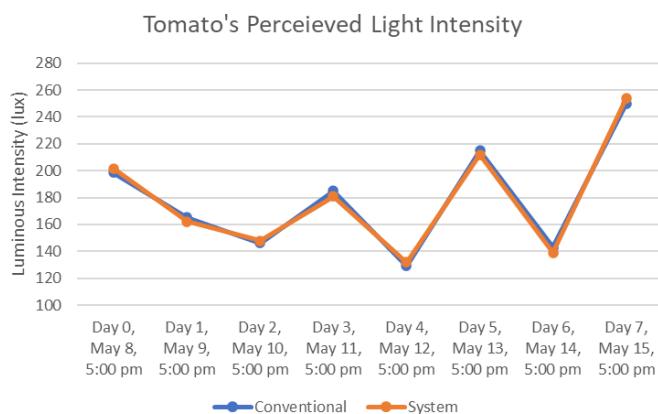


Figure 59. Tomato's Perceived Light Intensity

The figure above displays the graphical representation of the measured values for both the conventional and system setups. The graph indicates that both exhibit similar values as they are located inside the greenhouse.

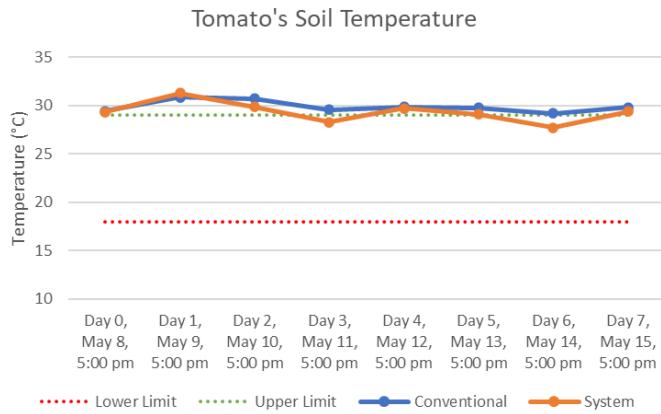


Figure 60. Tomato's Soil Temperature

The graph above shows the measured values of soil temperature. Both the conventional and controlled setups exhibit similar values since they are situated inside the greenhouse. However, there is a slight difference, with the controlled ones consistently staying within the threshold on certain days.

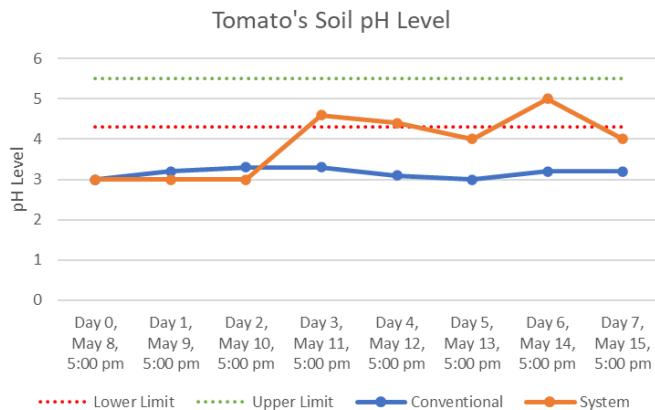


Figure 61. Tomato's Soil pH Level

The figure above displays the measured values of the pH level. The pH level of the controlled plants is adjusted on certain days using sulfur and lime. Additionally, there is an initial spike followed by a sudden decrease since pH levels take time to stabilize, as supported by research and consultation with experts.

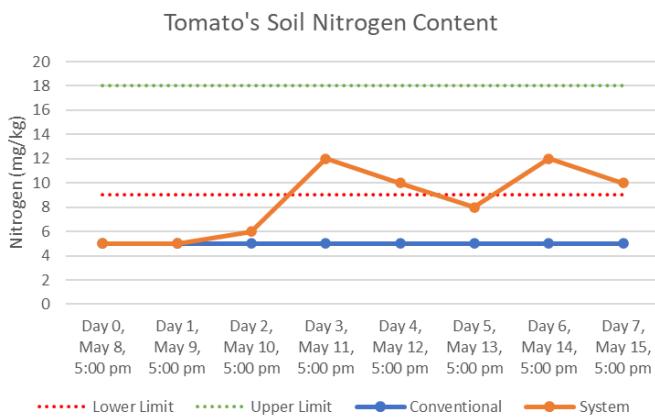


Figure 62. Tomato's Soil Nitrogen

The graph above displays the measured values of nitrogen. The controlled plants show corrections on specific days, as the fertilizer

correction device operates only twice a week but there is a significant difference between conventional and controlled ones.

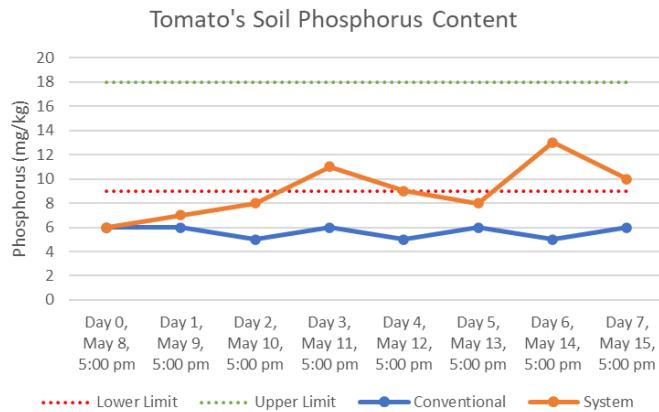


Figure 63. Tomato's Soil Phosphorus

The graph above displays the measured values of phosphorus. The controlled plants show corrections on specific days, as the fertilizer correction device operates only twice a week but there is a significant difference between conventional and controlled ones.

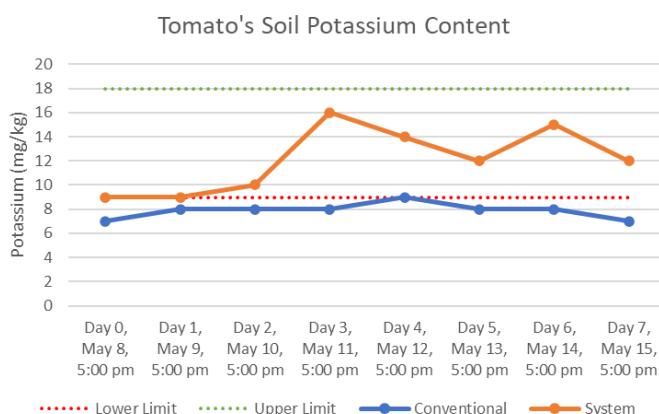


Figure 64. Tomato's Soil Potassium

The graph above displays the measured values of potassium. The controlled plants indicate that the data gathered are in the range of threshold for pepper.

4.4.3.2 Pechay

The results are based on the measured values in terms of plant's height and number of leaves for both controlled and uncontrolled groups for the Pechay plants. Paired differences are gathered and subjected to two-tailed t-test. For the hypothesis testing, two variables are considered. Hence, two hypotheses are formulated.

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth on number of leaves.)

Ha: **Controlled** \neq **Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% df = 13

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth of the plant's height.)

Ha: **Controlled \neq Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% df = 13

Table 33. Calculated Values for Pechay

Variable	Number of Sample	MEAN Difference	Standard Deviation	t-value	Two Tailed paired t-test
					df = 9
					a = 5%
No. of Leaves	14	0.669	1.057	2.364	Critical t-value
Height	14	2.139	1.359	5.879	2.16

Using the calculated values for table 11, for the Number of Leaves, with the t-value having a higher value compared to Critical t-value, it is concluded that the null hypothesis is rejected. As for the Plant's Height, as t-value has a higher value than the Critical t-value, it is concluded that the null hypothesis is rejected.

4.4.3.2.1 Measured Values for Pechay

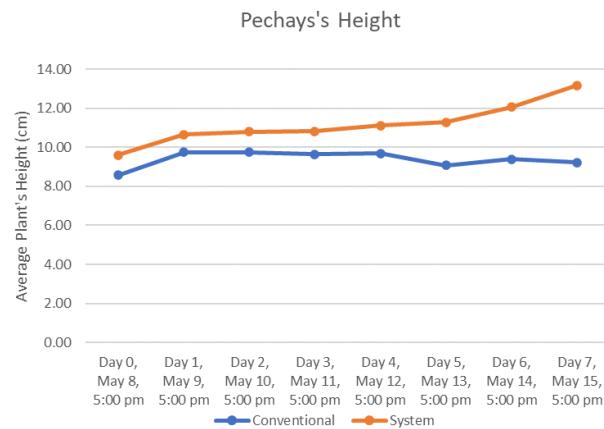


Figure 65. Pechay's Height

There are minor similarities in Pechay's height which can be seen in the figure shown above. But, as the day goes by, the plants on conventional

setup remain steady while the system goes taller resulting in a higher growth rate of 37.17% as compared to 7.69%. This finding is comparable with [22], which suggests that plant quality and yield can be improved by adjusting soil nutrients. Factor to consider that affects the height on the conventional setup were the presence of pests that ate plant's leaves decreasing the overall height of the plant.

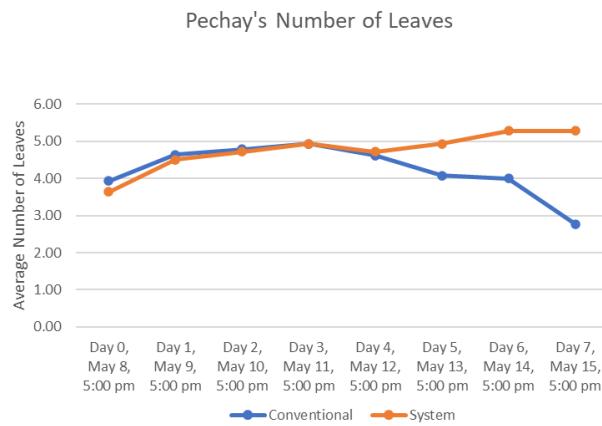


Figure 66. Pechay's Number of Leaves

Based on the figure above, both conventional and system have the same number of leaves in their first four days; however, in the next three days, the system has a lot of advantages because of the nutrients they take from the correctors. The conventional has a lot of leaves that are lost in the plant due to wilting and a pest called cutworm.

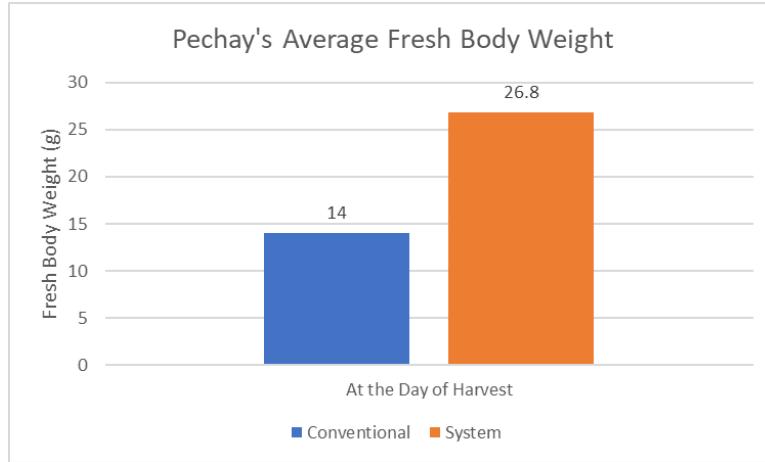


Figure 67. Pechay's Average Fresh Body Weight

As shown in the figure above, it can be seen in the graph that the system has significantly different values than the conventional one. A higher fresh body weight value may be helpful for assessing yield outcomes prior to product sale. This result correlates with the findings of [22] which states that correcting soil parameters thru application of inorganic fertilizers lead to increase in quality and yield of plants.

4.4.3.2.2 Parameters Measured for Pechay

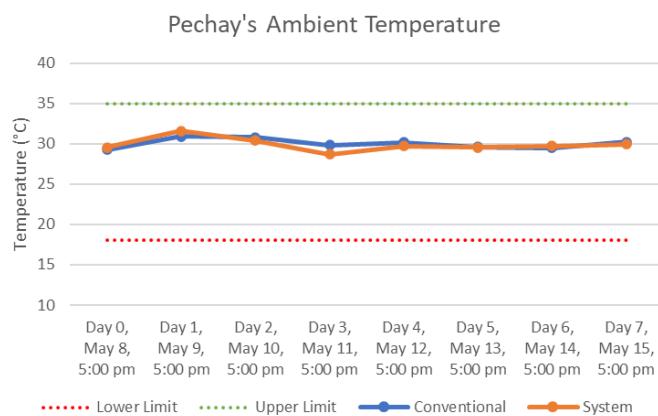


Figure 68. Pechay's Ambient Temperature

Figure 68 shows that both system and conventional planting has the same ambient temperature for the last 7 days and it also shows that the data gathered is within the threshold.

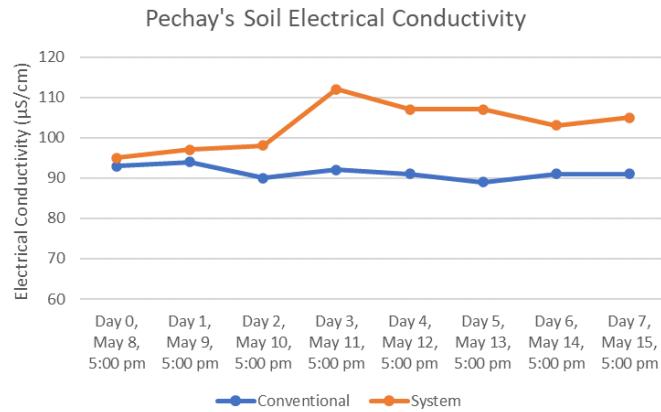


Figure 69. Pechay's Soil Electrical Conductivity

The data gathered in both system and conventional shows different measures in electrical conductivity. The system has a high value since the soil is mixed with pebbles and sand, whereas the conventional has the lowest value since it uses pure loam soil.

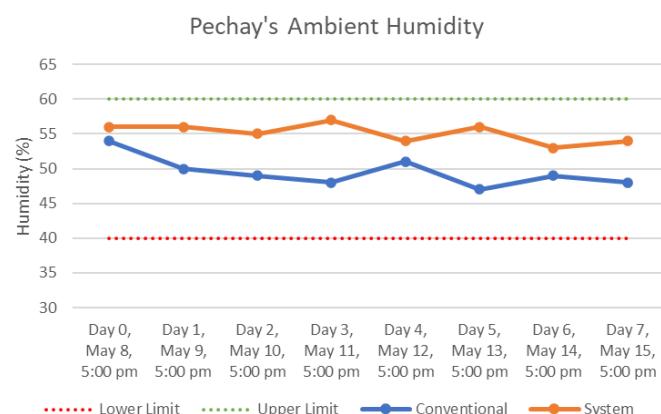


Figure 70. Pechay's Ambient Humidity

As shown in the graph above, the humidity of the system is better since it utilizes a humidifier compared to the conventional. Both system and conventional are within the threshold.

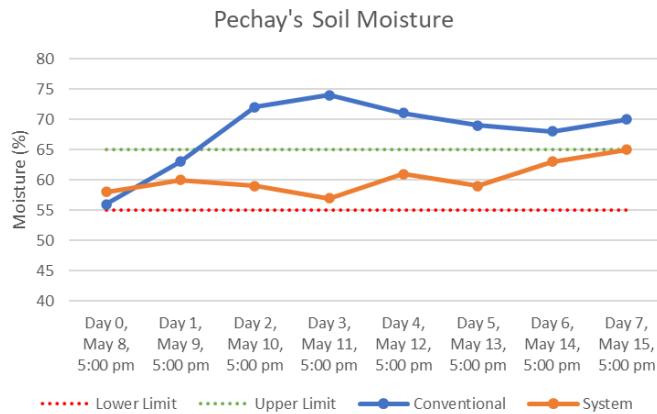


Figure 71. Pechay's Soil Moisture

The graph illustrates that the soil moisture readings for both the system and conventional are fairly close to the specified threshold. The fact that the system remained below the threshold from day 1 to day 7 suggests that it is more reliable.

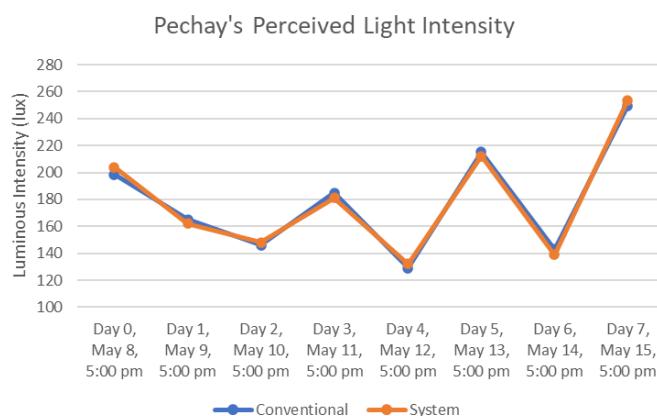


Figure 72. Pechay's Perceived Light Intensity

Figure 72 shows the graphical representation of the measured values for both system and conventional. The figure above specifies that both methods show similarities of the measured values in light intensity since it is located inside the greenhouse.

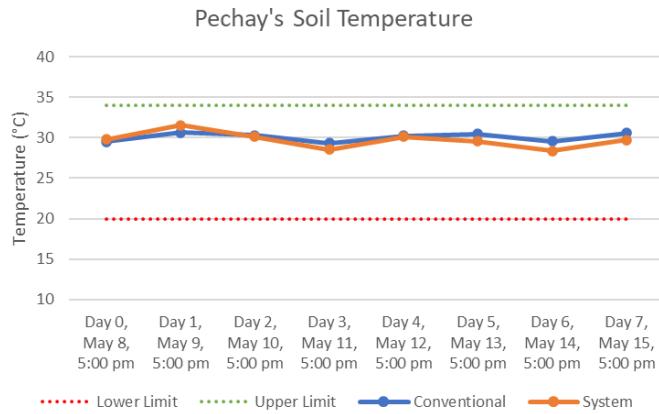


Figure 73. Pechay's Soil Temperature

Based on the graph above, both methods show similar measure values in soil temperature since they're both established inside a greenhouse, and it also shows that both methods stay within the threshold.

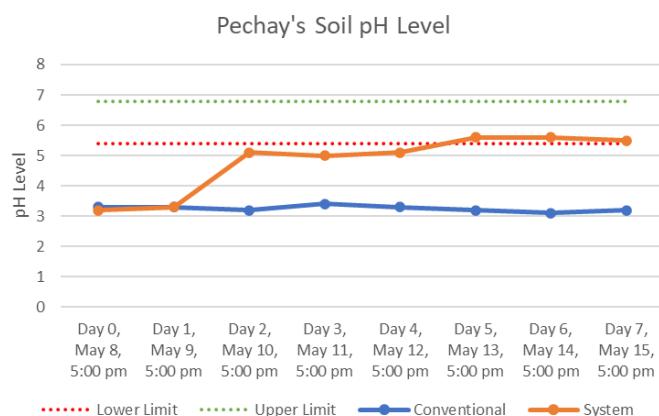


Figure 74. Pechay's Soil pH Level

The graph shows the major differences in the values for both systems and conventional methods. Lime and sulfur are being used to adjust the pH level of the system. The conventional method stays the same up to day 7, while the system fluctuates on day 1 due to pH level stabilization, according to experts. The system and conventional method are not within the threshold, as can be seen in Figure 81.

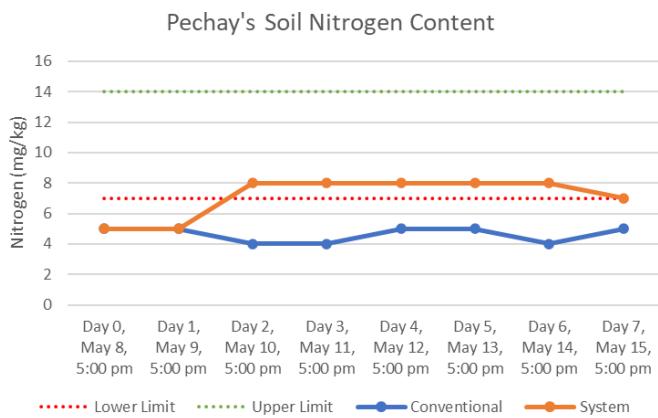


Figure 75. Pechay's Soil Nitrogen

Based on the graph above, there is a big difference between system and conventional methods. The system shows that the plot is within the threshold since it uses fertilizer called Urea compared to conventional.

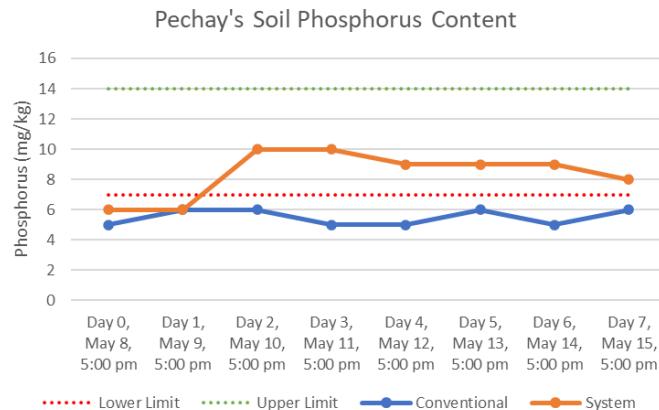


Figure 76. Pechay's Soil Phosphorus

The measured value of phosphorus is shown in the figure above.

The system is within the threshold given since it has a fertilizer called Duofos and a correction system that operates twice a week compared to conventional.

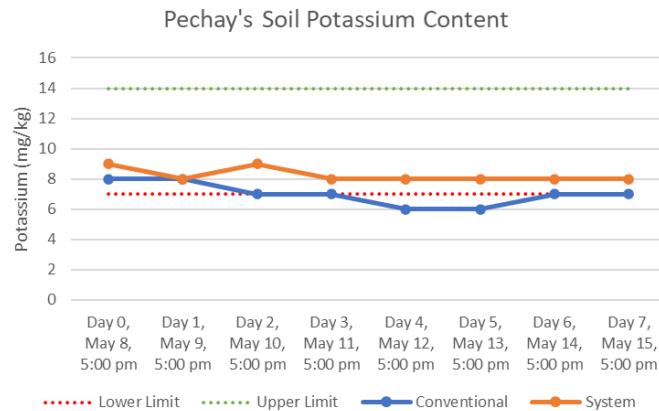


Figure 77. Pechay's Soil Potassium

The measured value of potassium is shown in the figure above. The system is within the threshold given since it has a fertilizer called Muriate of Potash and a correction system that operates twice a week compared to conventional.

4.4.3.3 Pepper

The results are based on the measured values in terms of plant's height, number of leaves, and number of leaves for both controlled and uncontrolled groups for the Pepper plants. Paired differences are gathered and subjected to two-tailed t-test. For the hypothesis testing, two variables are considered. Hence, two hypotheses are formulated.

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth on number of leaves.)

Ha: **Controlled \neq Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth of the plant's height.)

Ha: **Controlled** \neq **Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth on number of flowers.)

Ha: **Controlled** \neq **Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% $df = 9$

Table 34. Calculated Values for Pepper

Variable	Number of Sample	MEAN Difference	Standard Deviation	t- value	Two Tailed paired t-test				
					df = 9				
					$\alpha = 5\%$				
					Critical t-value				
No. of Leaves	10	2.167	2.478	2.741					
Height	10	0.567	3.079	0.581	2.262				
No. of Flowers	10	0.08	0.57	0.444					

With the Number of Leaves, with the t-value having a lower value compared to the Critical t-value, it is concluded that the null hypothesis is subjected to fail to reject. As for the Plant's Height, as t-value has a higher value than the Critical t-value, it is concluded that the null hypothesis is subjected to fail to reject. For the Number of Flowers, as the t-value is less than the Critical t-value, it is concluded that the null hypothesis is subjected to fail to reject.

4.4.3.3.1 Measured Values for Pepper plant

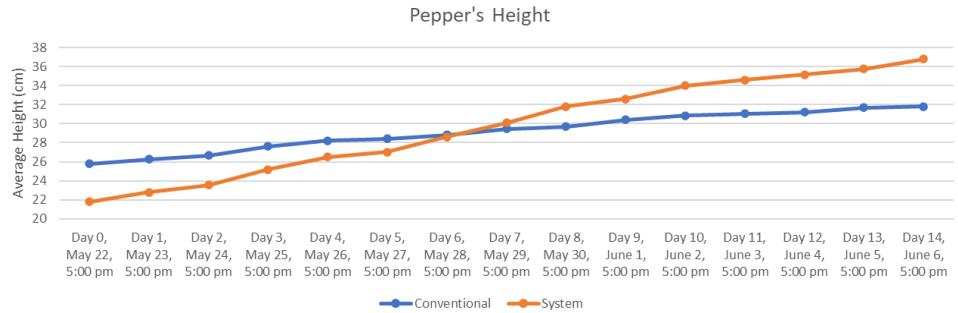


Figure 78. Pepper's Height

From the day 0 up to day 14 shows more effectiveness in using the system based on the height measured from the plants. Furthermore, the growth rate of the system is much higher than the conventional with computed rate of 68.81% and 23.26% respectively as the evidence [22] claims that using an inorganic fertilizer in correcting soil nutrients boosts the growth rate of the plants.

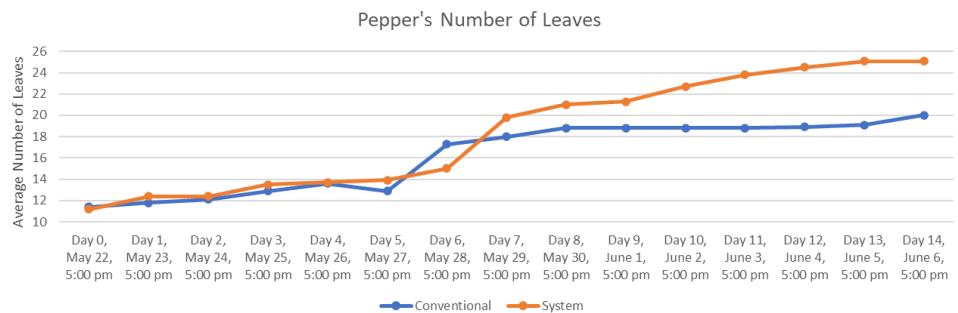


Figure 79. Pepper's Number of Leaves

The average number of leaves indicates a significant difference between using the system for planting and conventional methods. The graph

of the controlled plants displays a higher number of leaves compared to the others.

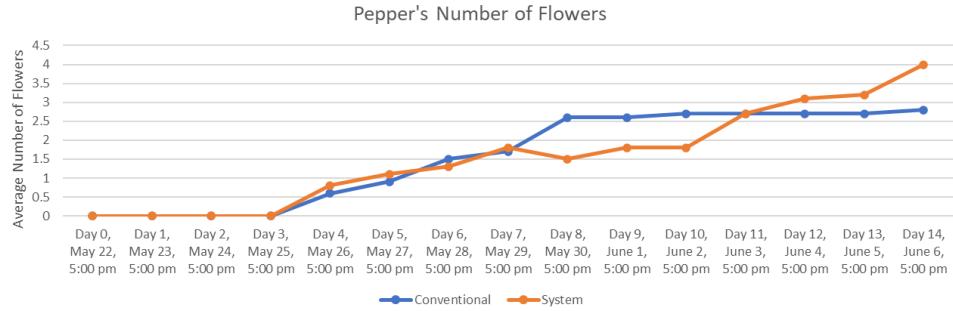


Figure 80. Pepper's Number of Flowers

There are more flowers in bloom on the plants that are managed by the system compared to the conventional ones. However, the conventional plants still have a significant number of flowers, as the soil used for planting contains a higher proportion of loam soil compared to the managed plants.

4.4.3.3.2 Parameters Measured for Pepper

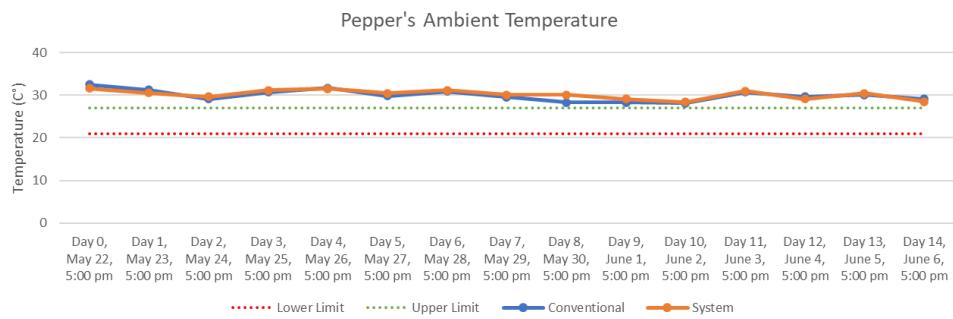


Figure 81. Pepper's Ambient Temperature

The figure shown above displays the measured values of ambient temperature from day 1 to day 14. It demonstrates that both the conventional and system setups exhibit similar trends, as they are located inside the greenhouse and show a mutual inclination towards each other.

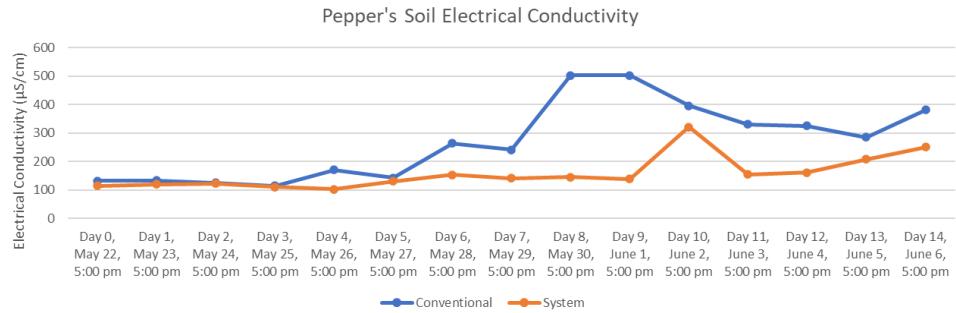


Figure 82. Pepper's Soil Electrical Conductivity

The graphical representation above shows the different measured values of electrical conductivity for both the conventional and system setups. The controlled plants exhibit lower values, which can be attributed to the soil being specifically managed for the peppers.

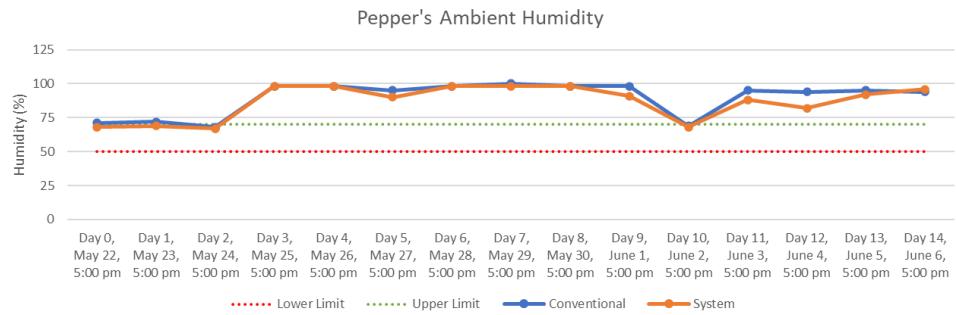


Figure 83. Pepper's Ambient Humidity

As shown in the figure above, the measured data of humidity for the peppers does not fall within the threshold range. This is primarily due to the rainy weather conditions experienced during most of those days.

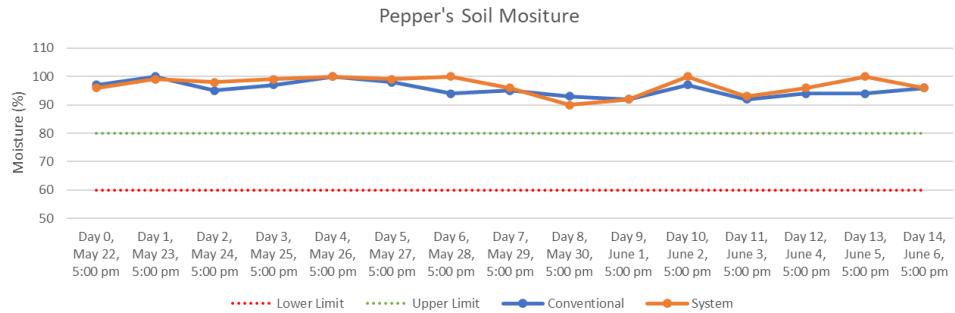


Figure 84. Pepper's Soil Moisture

The graph displays the measured value of soil moisture, which, contrary to expectations, does not remain within the threshold range due to the rainy weather conditions.

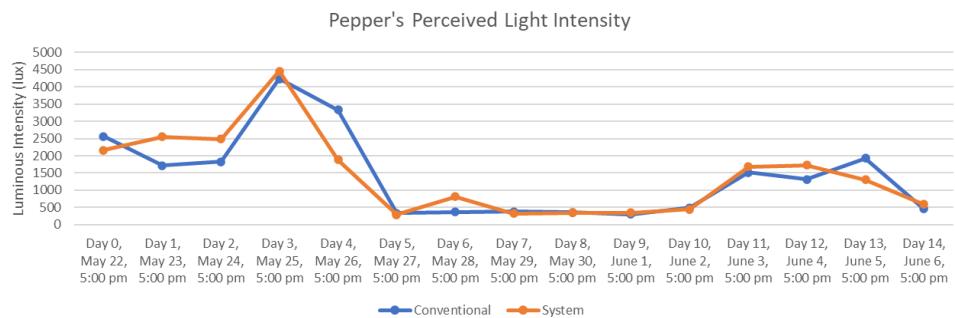


Figure 85. Pepper's Perceived Light Intensity

The figure above displays the graphical representation of the measured values for both the conventional and system setups. The graph indicates that both exhibit similar values as they are located inside the greenhouse.

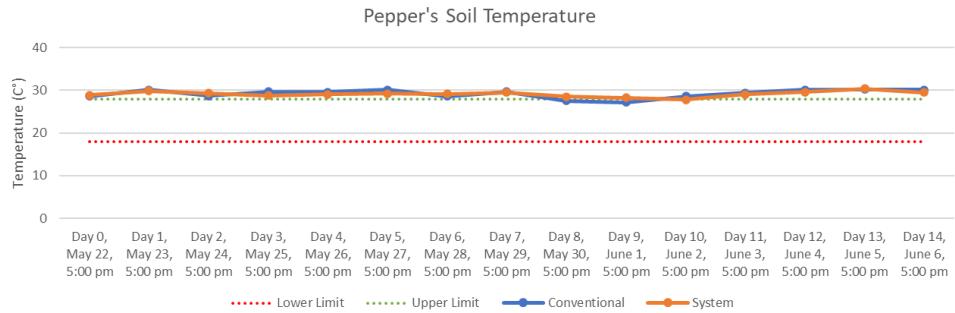


Figure 86. Pepper's Soil Temperature

The graph above shows the measured values of soil temperature. Both the conventional and controlled setups exhibit similar values since they are situated inside the greenhouse.

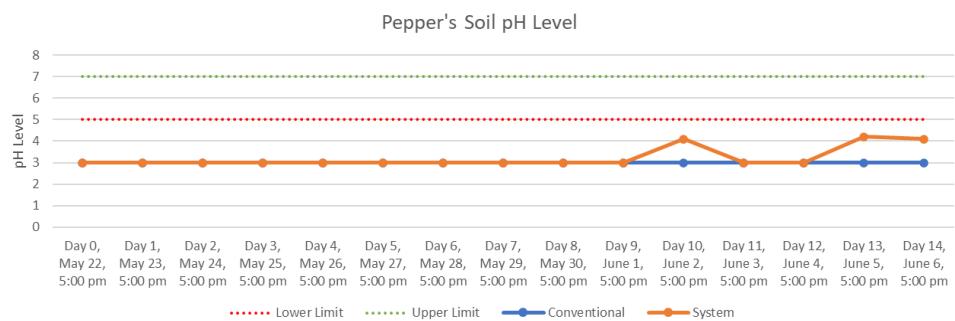


Figure 87. Pepper's Soil pH Level

The graph displays the measured values of the pH level. The pH level of the controlled plants is adjusted on certain days using sulfur and lime. Additionally, there is an initial spike followed by a sudden decrease since pH levels take time to stabilize, as supported by research and consultation with experts.

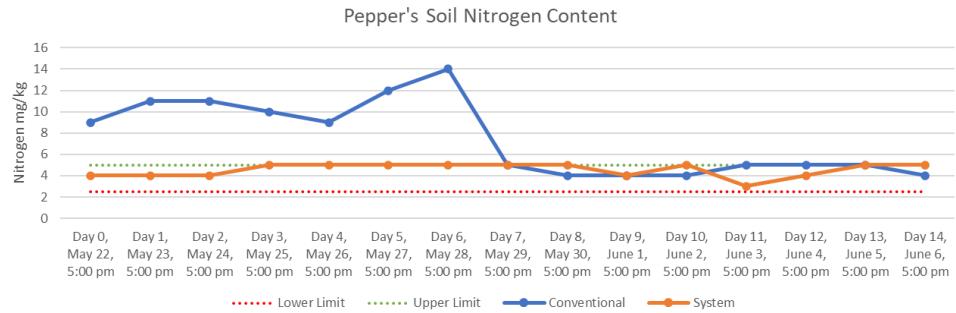


Figure 88. Pepper's Soil Nitrogen

The graph represents that there is a significant difference in using the system for planting than the conventional specifically in the nitrogen of the soil.

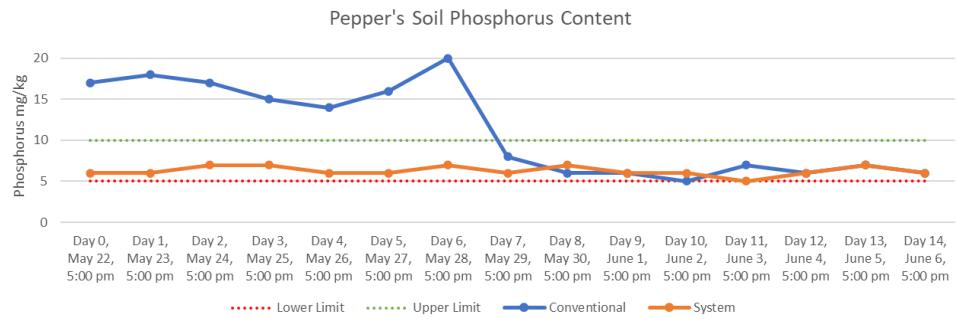


Figure 89. Pepper's Soil Phosphorus

The controlled plants show corrections on specific days, as the fertilizer correction device operates only twice a week but there is a significant difference between conventional and controlled ones.

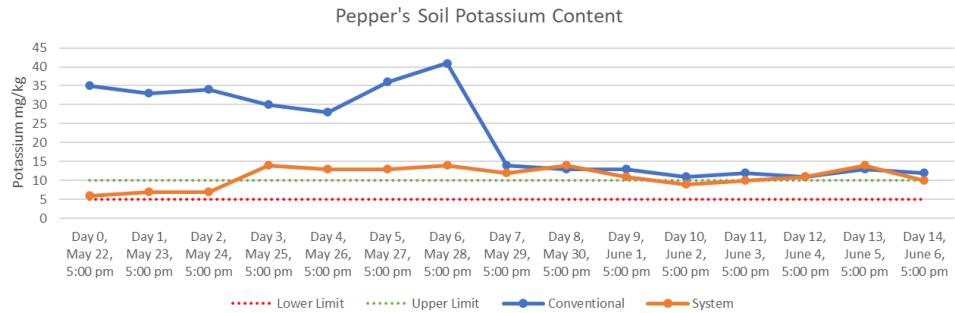


Figure 90. Pepper's Soil Potassium

The use of the system for planting has greatly assisted in maintaining the threshold of potassium for peppers. However, it is noticeable that on certain days, the levels are outside the desired range due to the fertilizer correction system operating only twice a week

4.4.3.4 Lettuce

The results are based on the measured values in terms of plant's height, number of leaves, and number of leaves for both controlled and uncontrolled groups for the Pepper plants. Paired differences are gathered and subjected to two-tailed t-test. For the hypothesis testing, two variables are considered. Hence, two hypotheses are formulated.

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth on number of leaves.)

Ha: **Controlled ≠ Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95%

$df = 13$

Ho: **Controlled = Uncontrolled** (There is no significant difference between the use of the system and conventional planting in terms of growth of the plant's height.)

Ha: **Controlled ≠ Uncontrolled** (There is a significant difference)

Level of significance: $\alpha = 0.05$

Confidence level: 95% $df = 13$

Table 35. Calculated Values for Lettuce

Variable	Number of Sample	MEAN Difference	Standard Deviation	t- value	Two-Tailed paired t-test				
					df = 9				
					$\alpha = 5\%$				
					Critical t-value				
No. of Leaves	14	1.53	0.47	10.18	2.16				
Height	14	3.04	2.07	5.49					

Using the calculated values for table 13, for the Number of Leaves, with the t-value having a lower value compared to the Critical t-value, it is concluded that the null hypothesis is rejected. As for the Plant's Height, as t-value has a higher value than the Critical t-value, it is concluded that the null hypothesis is rejected.

4.4.3.4.1 Measured Values for Lettuce Plant

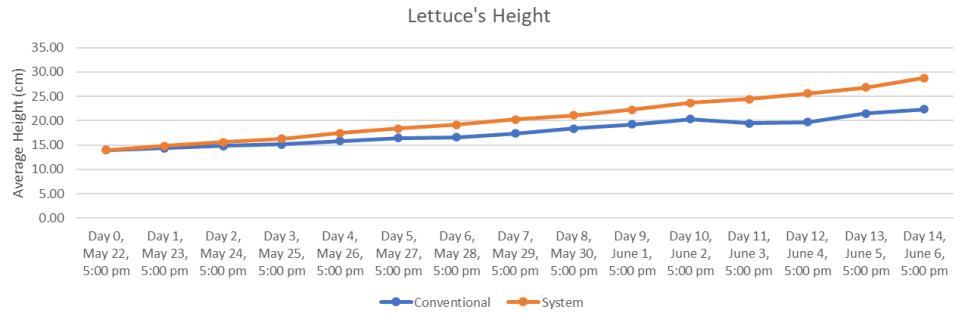


Figure 91. Lettuce's Height

The graph shows that both the system and conventional methods increase in height as the day progresses. However, as day progresses, the system expresses the increase in its height compared to the conventional method. Overall, the Lettuces on conventional setup grew at a rate of 60.64% while on the system, plants progressed at 106.05%. Additionally, the growth rate stated can be supported by the study [22] which claims that inorganic or water-based fertilizers really help with the development of the plants.

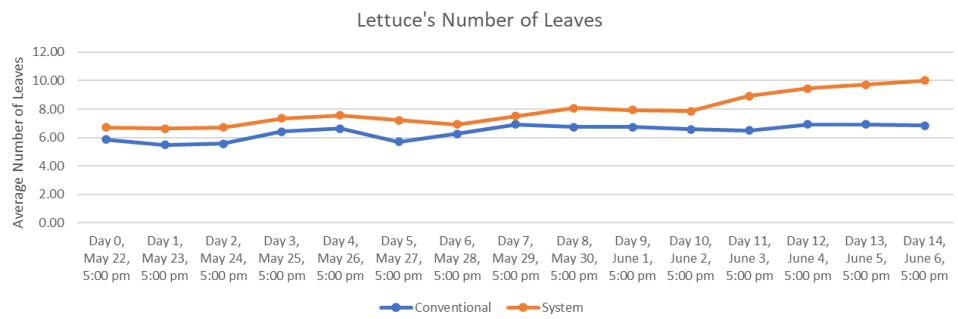


Figure 92. Lettuce's Number of Leaves

With regards to the number of leaves, the graph indicates that the system has more leaves than conventional since it uses a correction system

that operates twice a week with fertilizers. Even yet, wilting and pests cause an instant decrease in the number of leaves on some days for conventional and system.

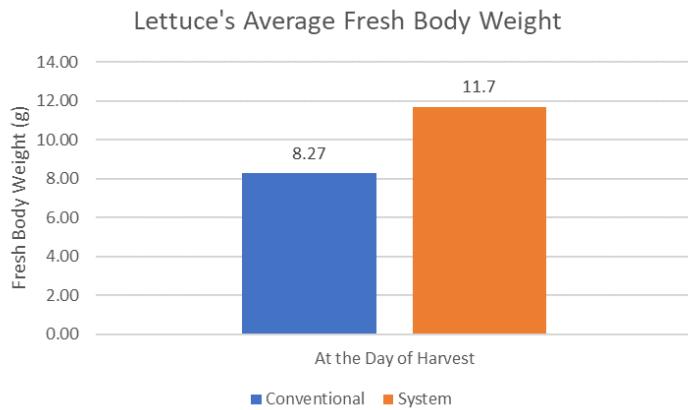


Figure 93. Lettuce's Fresh Body Weight

In terms of fresh body weight, Lettuces from the system has significant advantage as compared to the conventional given that nutrients on the system setup were modified to suit Lettuces growth.

4.4.3.4.2 Parameters Measured for Lettuce

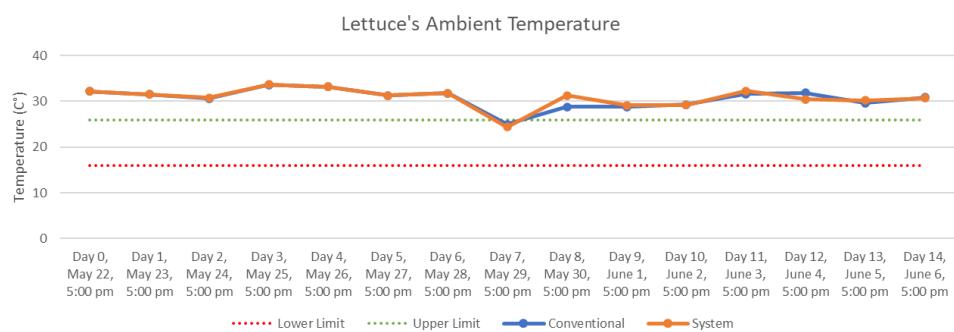


Figure 94. Lettuce's Ambient Temperature

Both system and conventional are not within the threshold given which can be seen in the figure above. It indicates in the graph that both methods appear to be identical since it is both established in the greenhouse.

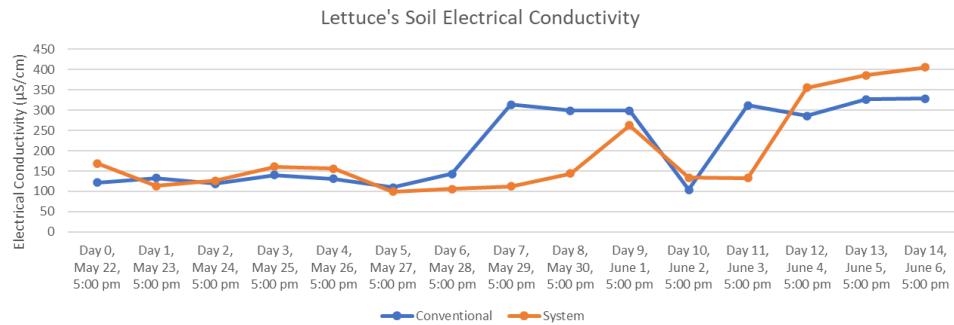


Figure 95. Lettuce's Soil Electrical Conductivity

The different measured electrical conductivity values for both conventional and system ways are displayed in the graph above. Some days, the system has a high value since the soil is mixed with pebbles and sand, whereas the conventional has the lowest value since it uses pure loam soil.

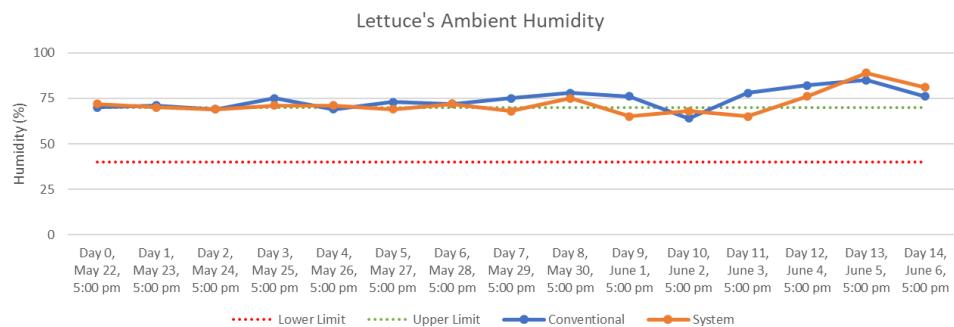


Figure 96. Lettuce's Ambient Humidity

The measured value in humidity of lettuce for both conventional and system does not meet the specified threshold. Due to rainy weather

conditions, the graph expresses the similarities of the measured value for both methods.

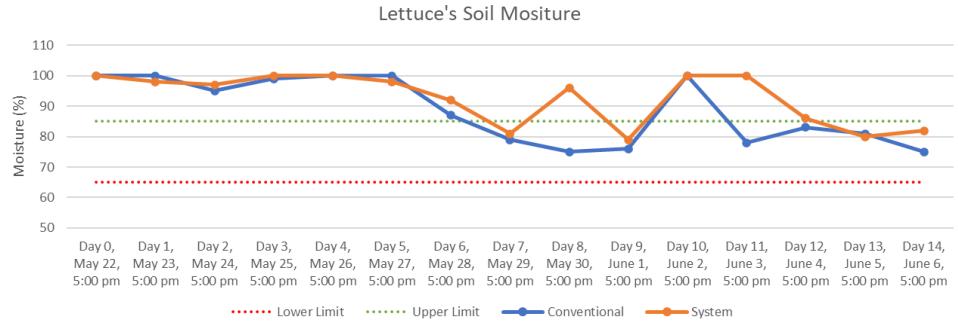


Figure 97. Lettuce's Soil Moisture

Both system and conventional methods do not fall within the specified threshold. It shows in the figure that it has a sudden fluctuation due to the rainy season.

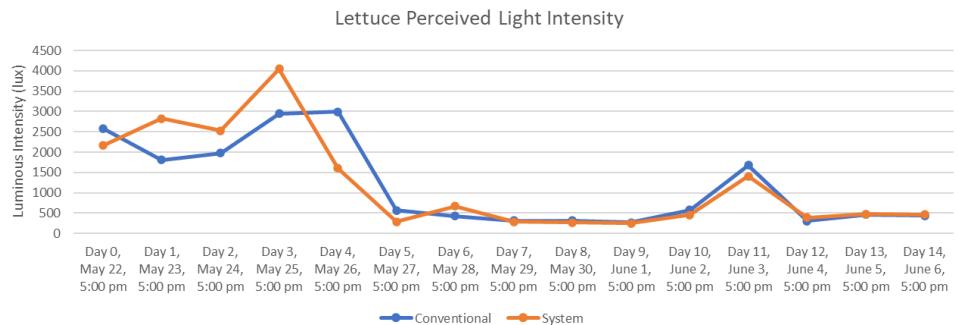


Figure 98. Lettuce's Perceived Light Intensity

The graph indicates the measured values in light intensity of lettuce for both conventional and system methods. As the day goes by, the graph shows the comparison of the measured values for both system and conventional since they are deployed in a greenhouse.

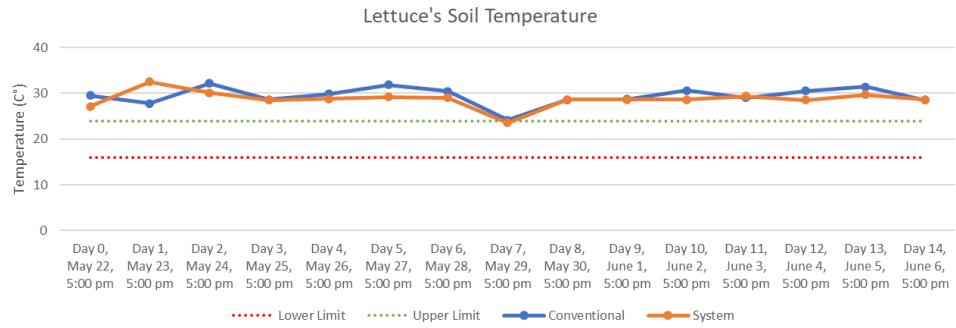


Figure 99. Lettuce's Soil Temperature

Since both setups were situated in the same greenhouse and utilized the same type of soil, the soil temperature readings were nearly identical.

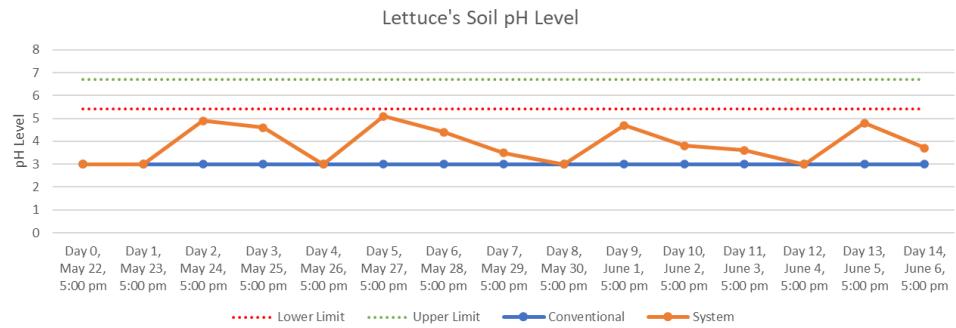


Figure 100. Lettuce's pH Level

For the pH level of soil used in both conditions, it can be drawn that the system attempted to correct the acidity of soil at certain days, whereas on the conventional setup, pH level remains constant.

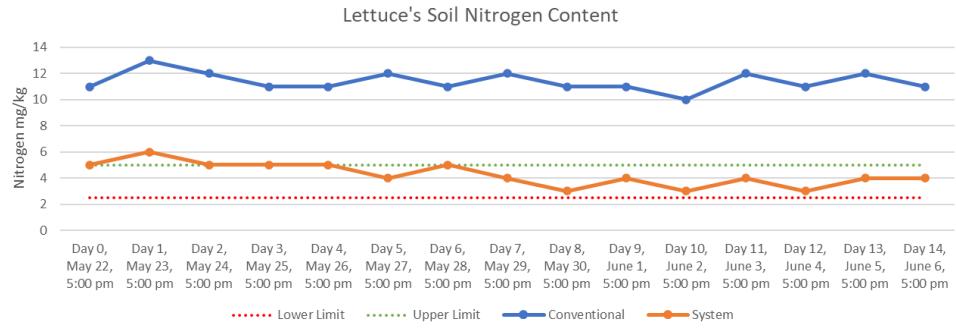


Figure 101. Lettuce's Soil Nitrogen

The correction system was able to maintain the Nitrogen level of the soil within the identified threshold value suited for Lettuce, whereas on the conventional setup, the Nitrogen level of soil remains high without proper intervention.

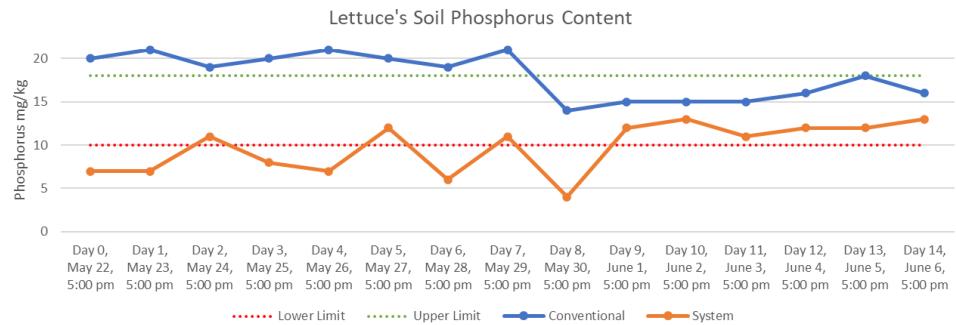


Figure 102. Lettuce's Soil Phosphorus

The graph of phosphorus shows the measure values for both system and conventional methods. It expresses that the system is more inclined toward the specified threshold compared to conventional systems from day 1 to day 14. Due to the fact that the corrective system only operates twice a week, on certain days the measured value of the system did not fall inside the threshold range.

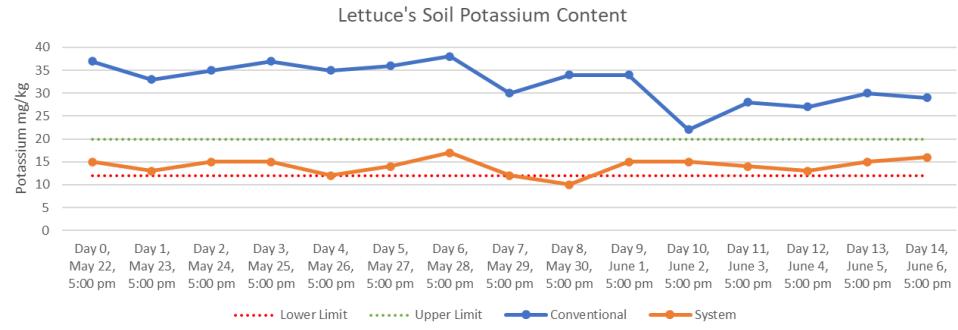


Figure 103. Lettuce's Soil Potassium

The graphical representation of potassium shows that the controlled ones are inclined in the threshold from day 1 up to day 14. However, there are some days that it is out of range since the correction system for fertilizers operates twice a week.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

This chapter presents the summary, conclusion of the objectives, and recommendations about the study.

5.1 Summary

The project study entitled "WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System" aims to develop a wireless sensor network-enabled soil health and ambience condition monitoring system with correction feedback through IoT using ZigBee protocol. The project focused on the monitoring and correction system with a 12-V lead-acid battery and solar panel that will help automate monitoring and correcting the soil and ambient parameters for varied plants in a greenhouse located in Brgy. Bagong Pagasa, Quezon City.

The testing conducted by the researchers yielded the following results, which are now presented and discussed: To assess the accuracy of the Sensor Nodes compared to commonly used measuring tools, a series of tests were conducted for Humidity, Moisture, Soil Temperature, pH level, and Ambient Temperature. The percent errors obtained from these tests were used to determine the accuracy of the Sensor Nodes. For Humidity, Sensor Node 1 exhibited an accuracy of 98%, while Sensor Node 2 showed an accuracy of 99%. Regarding Moisture, the measuring tool provided a range of values, and both Sensor Nodes 1 and 2 consistently fell within their respective ranges for all trials. In terms of Soil Temperature, both Sensor Node 1 and Sensor Node 2 demonstrated an accuracy of 95%. For pH level, Sensor Node 1 achieved an accuracy of 92%, while Sensor Node 2 achieved an accuracy of 93%. For Ambient Temperature, both Sensor Node

1 and Sensor Node 2 exhibited an accuracy of 99%. In regards to the NPK, the test used provided a range of values for the corresponding colors shown for the soil testing. Both Sensor Nodes 1 and 2 proved to be accurate as the measurements fell within their range for all trials. These findings highlight the performance of the Sensor Nodes compared to traditional measuring tools across various parameters, showcasing their accuracy in measuring Humidity, Soil Temperature, pH level, Moisture, Ambient Temperature, and NPK.

After formulating the hypotheses, the uncontrolled and controlled groups were subjected to paired t-tests to examine the mean differences, leading to different conclusions for each plant type. The growth of Tomato plants exhibited no significant difference in the number of leaves but a significant difference in height. Pechay plants, on the other hand, showed a significant difference in both the number of leaves and height. The Pepper plant analysis considered three factors. While there was a significant difference in the number of leaves, no significant differences were observed in height or the number of flowers. As for Lettuce plants, both height and the number of leaves demonstrated significant differences. These significant differences favored the controlled group, indicating that the system's use provided an advantage in terms of height growth due to parameter correction. However, regarding the number of leaves, growth varied across different plant types and living conditions, suggesting no significant effect.

5.2 Conclusion

Based on the findings and results of the study, the proponents conclude the followings:

1. Using the market-available devices and the existing technology, SOIL-N-WAN was established, featuring different soil and ambient condition sensors. Specifically, the system was able to determine the soil's temperature, moisture, pH level, electrical conductivity, and NPK content, as well as the surrounding temperature and humidity, while using solar energy.

2. The designed sensor nodes were able to transmit data simultaneously to the gateway with ease using the Zigbee protocol. In fact, most of the sent data was successfully received, regardless of the weather conditions. Also, the gateway was able to upload the sensor data to the Firebase Realtime database, enabling remote monitoring via a mobile application.
3. The group successfully developed the SOIL-N-WAN App, a real-time monitoring system that displays sensor measurements on each node, the visualization of data on each parameter via graphs, and the status of the correction system. The app includes a plant/node selection system to separate sensor readings on each node, as well as a notification system to inform the user of which parameter requires adjustment.
4. SOIL-N-WAN was integrated with a correction system based on the Fuzzy Logic Algorithm. For soil parameters, drip fertigation was utilized to minimally improve the nutrient content suited for each plant, and for the surrounding area, a humidifier was deployed. Based on the results of the testing, the closed-loop correction system was able to minimally adjust the parameters within the range of thresholds specified for each plant.
5. To assess sensor accuracy, the proponents used soil testing instruments and compare the measured values to the sensors through percent errors. The sensors employed in measuring Ambient Temperature, Ambient Humidity, Soil Moisture, Temperature, and NPK demonstrated satisfactory accuracy, as the calculated percent errors equates to an average below 5% which fell within the acceptable range. However, when examining the pH level measurements, it was evident that the accuracy was not optimal. The average percent error was below 8%, whereas the acceptable threshold was defined as 5% or below. After the deployment period, the SOIL-N-WAN was evaluated by the expert and users, which involved their resident agriculturist and a few local farmers, respectively. The expert concluded that the system is excellently reliable

and effective while still being accurate and intuitive in displaying results. The average rating for the system's overall performance among the users was 4.5, indicating their complete satisfaction.

5.3 Recommendations

The following recommendations were made for improvement of the system:

1. To test the system on large scale applications.
2. To enhance the power management system to maximize power efficiency and ensure the continuous usage of the devices
3. To identify and use fertilizers to address high values of NPK content of soil.
4. To use organic fertilizers instead of artificial sources of nutrients.
5. To integrate a more powerful device to correct the ambient conditions.
6. Consider utilizing pesticide solution to deal with the presence of pests and insects.

References

- [1] Bureau of Soils and Water Management, “Why BSWM?” <http://bswm.da.gov.ph/why-bswm/>
- [2] Food and Agriculture Organization - World Bank, “Agricultural land (% of land area) - Philippines,” 2018. <https://data.worldbank.org/indicator/AG.LND.AGRI.ZS?locations=PH>
- [3] Statista Research Department, “Agriculture in the Philippines - statistics & facts,” 2021. https://www.statista.com/topics/5744/agriculture-industry-in-the-philippines/#topicHeader_wrapper (accessed Jun. 08, 2022).
- [4] Nations Encyclopedia, “Philippines - Agriculture.” <https://www.nationsencyclopedia.com/economies/Asia-and-the-Pacific/Philippines-AGRICULTURE.html#:~:text=The%20country's%20main%20agricultural%20crops,%2C%20rubber%2C%20and%20cotton.>
- [5] Bureau of Soils and Water Management. (n.d.). Soil Test Kit (STK). <http://bswm.da.gov.ph/download/soil-test-kit-stk/>
- [6] R. Madhumathi, T. Arumuganathan, and R. Shruthi, “Soil NPK and Moisture analysis using Wireless Sensor Networks,” in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Jul. 2020, pp. 1–6. doi: 10.1109/ICCCNT49239.2020.9225547.
- [7] S. N. Shylaja and M. B. Veena, “Real-time monitoring of soil nutrient analysis using WSN,” in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Aug. 2017, pp. 3059–3062. doi: 10.1109/ICECDS.2017.8390018.

- [8] J. J. Estrada-Lopez, A. A. Castillo-Atoche, J. Vazquez-Castillo, and E. Sanchez-Sinencio, “Smart Soil Parameters Estimation System Using an Autonomous Wireless Sensor Network with Dynamic Power Management Strategy,” IEEE Sens. J., vol. 18, no. 21, pp. 8913–8923, 2018, doi: 10.1109/JSEN.2018.2867432.
- [9] R. S. Ghumatkar, M. K. Hulbatte, M. M. Gaikwad, and M. R. B. Gurav, “IoT Based Temperature and Soil Monitoring System with Motor Pump Control,” Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, no. 4, pp. 1165–1168, 2022, doi: 10.22214/ijraset.2022.41352.
- [10] U. Madhura, P. Akshay, A. J. Bhattad, and G. Nagaraja, “Soil Quality Management Using Wireless Sensor Network,” 2nd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2017, pp. 108–112, 2018, doi: 10.1109/CSITSS.2017.8447860.
- [11] R. Bhowmik and D. A. Pathak, “an Arduino Based Wsn To Control and Monitor the Greenhouse Parameters.,” Int. J. Adv. Res., vol. 5, no. 3, pp. 1501–1508, 2017, doi: 10.21474/ijar01/3658.
- [12] Shinde, D., & Siddiqui, N. (2018). IOT Based Environment change Monitoring Controlling in Greenhouse using WSN. *2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018*, 1–5.
<https://doi.org/10.1109/ICICET.2018.8533808>
- [13] Eos, “Soil Temperature As A Factor Of Crops Development,” 2020.
<https://eos.com/blog/soil-temperature/>

- [14] Brownmang onwuka, “Effects of soil temperature on some soil properties and plant growth,” vol. Volume 8, no. Issue 1, 2018, [Online]. Available: <https://medcraveonline.com/APAR/APAR-08-00288.pdf>
- [15] S. Unninayar, “Monitoring, Observations, and Remote Sensing – Global Dimensions,” 2015. <https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/soil-moisture>
- [16] B. Siregar, A. B. Azmi Nasution, L. Adlin, U. Andayani, and F. Fahmi, “Soil Moisture Monitoring System using Wireless Sensor Network,” J. Phys. Conf. Ser., vol. 1028, no. 1, 2018, doi: 10.1088/1742-6596/1028/1/012058.
- [17] Queensland. (2013). Soil pH | Soil properties. <https://www.qld.gov.au/environment/land/management/soil/soil-properties/ph-levels>
- [18] R. Mylavarapu, J. Bergeron, N. Wilkinson, and E. A. Hanlon, “Soil pH and Electrical Conductivity: A County Extension Soil Laboratory Manual,” Edis, vol. 2020, no. 1, pp. 1–10, 2020, doi: 10.32473/edis-ss118-2020.
- [19] USDA, “Inherent Factors Affecting Soil Phosphorus,” no. Figure 1, pp. 1–6, 1994.
- [20] Department of Primary Industries, “Plant nutrients in the soil,” 2017. <https://www.dpi.nsw.gov.au/agriculture/soils/soil-testing-and-analysis/plant-nutrients>
- [21] R. L. Mahler, “Nutrients plants require for Growth,” Coll. Agric. Life Sci., vol. CIS 1124, pp. 1–4, 2004.
- [22] “The Effect of Integrated Organic and Inorganic Fertilizer on Soil Fertility and Productivity” Journal of Ecology & Natural Resources, vol. 5, no. 3, 2021, doi: <https://doi.org/10.23880/jenr-16000248>.

[23] G. E. T. Involved and C. Us, “Environmental Factors Affecting Plant Growth,” Oregon State University, 2022. <https://extension.oregonstate.edu/gardening/techniques/environmental-factors-affecting-plant-growth>

[24] J. L. Hatfield and J. H. Prueger, “Temperature extremes: Effect on plant growth and development,” Weather and Climate Extremes, 2015.

<https://www.ars.usda.gov/research/publications/publication/?seqNo115=313172>

[25] Texas A&M Agrilife Extension, “Light, Temperature and Humidity.” <https://aggie-horticulture.tamu.edu/ornamental/a-reference-guide-to-plant-care-handling-and-merchandising/light-temperature-and-humidity/#:~:text=Light%20intensity%20influences%20the%20manufacture,have%20larger%2C%20dark%20green%20leaves>.

[26] T. W. Tibbitts, “Humidity and Plants,” Bioscience, vol. 29, no. 6, pp. 358–363, 1979, doi: 10.2307/1307692.

[27] J. Badgery, “Light in the greenhouse,” NSW Agriculture, 1999.
https://www.dpi.nsw.gov.au/__data/assets/pdf_file/0007/119365/light-in-greenhouse.pdf

[28] S. Heble, A. Kumar, K. V. V. D. Prasad, S. Samirana, P. Rajalakshmi, and U. B. Desai, “A low power IoT network for smart agriculture,” IEEE World Forum Internet Things, WF-IoT 2018 - Proc., vol. 2018-Janua, pp. 609–614, 2018, doi: 10.1109/WF-IoT.2018.8355152.

[29] X. Shipu, W. Yunsheng, L. Yong, Z. Jingyin, and Z. Chenxi, “Design and Research on the Farmland Soil Environmental Monitoring System Based on Agricultural Network,” 2019 IEEE 2nd Int. Conf. Comput. Commun. Eng. Technol. CCET 2019, pp. 46–49, 2019, doi: 10.1109/CCET48361.2019.8989247.

- [30] D. K. Rathinam, D. Surendran, A. Shilpa, A. Santhiya Grace, and J. Sherin, “Modern Agriculture Using Wireless Sensor Network (WSN),” 2019 5th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2019, pp. 515–519, 2019, doi: 10.1109/ICACCS.2019.8728284.
- [31] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari, “LoraFarM: A LoRaWAN-based smart farming modular IoT architecture,” Sensors (Switzerland), vol. 20, no. 7, 2020, doi: 10.3390/s20072028.
- [32] M. M. Suwaid, M. H. Habaebi, and S. Khan, “Embedded LoRaWAN for Agricultural Sensing Applications,” ICETAS 2019 - 2019 6th IEEE Int. Conf. Eng. Technol. Appl. Sci., 2019, doi: 10.1109/ICETAS48360.2019.9117346.
- [33] S. R. J. Ramson et al., “A Self-Powered, Real-Time, LoRaWAN IoT-Based Soil Health Monitoring System,” IEEE Internet Things J., vol. 8, no. 11, pp. 9278–9293, Jun. 2021, doi: 10.1109/JIOT.2021.3056586.
- [34] B. Citoni, F. Fioranelli, M. A. Imran, and Q. H. Abbasi, “Internet of Things and LoRaWAN-Enabled Future Smart Farming,” 2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput., no. December 2019, pp. 3507–3508, 2017.
- [35] Ezhilazhahi, A. M., & Bhuvaneswari, P. T. V. (2017). IoT enabled plant soil moisture monitoring using wireless sensor networks. *Proceedings of 2017 3rd IEEE International Conference on Sensing, Signal Processing and Security, ICSSS 2017*, 345–349.
<https://doi.org/10.1109/SSPS.2017.8071618>

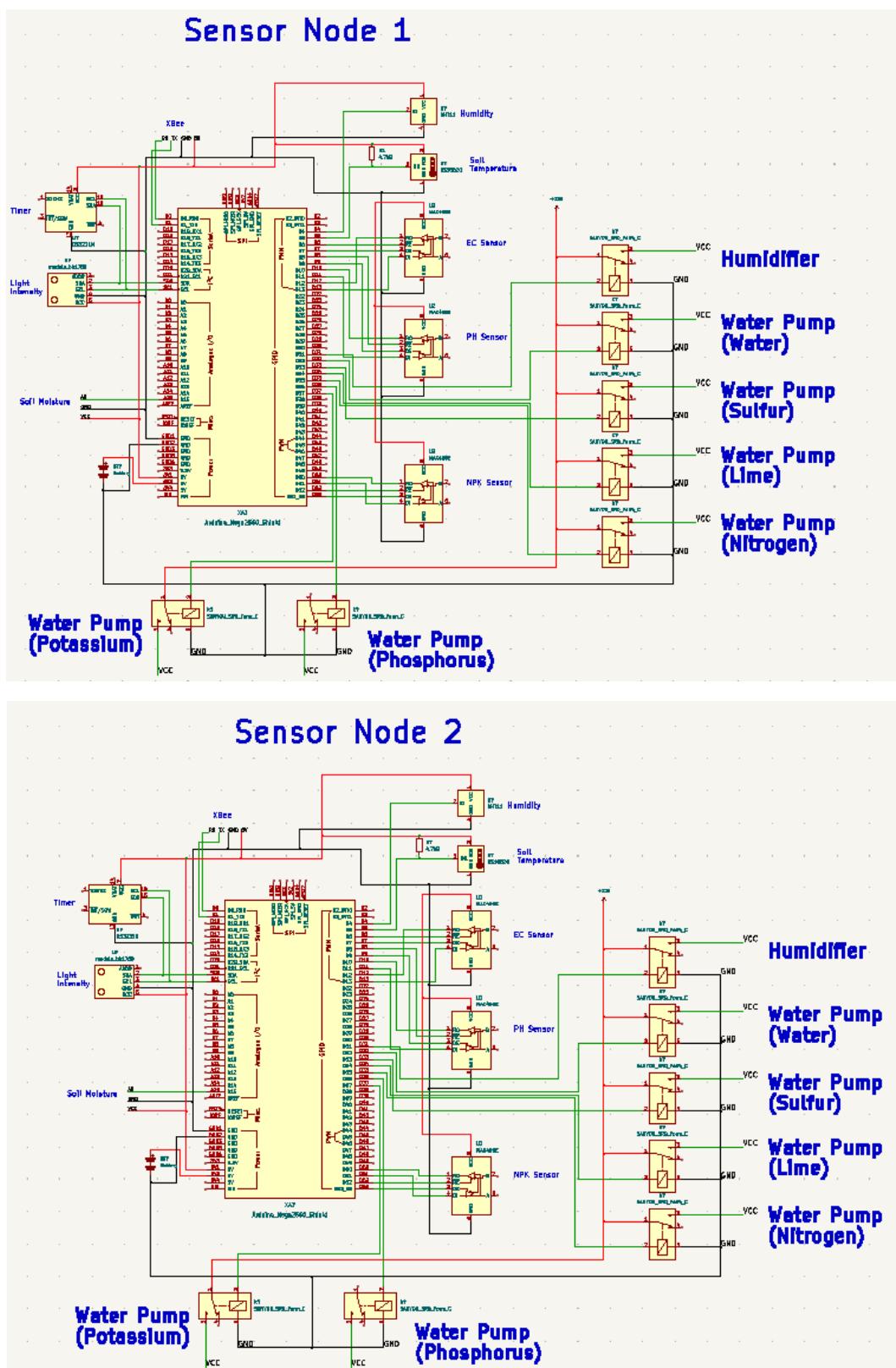
- [36] Guico, M. L., Monje, J. C., Oppus, C., Domingo, A., Ngo, G., & Torres, J. B. (2019). Wireless Sensor Network for Soil Monitoring. *2019 IEEE Conference on Wireless Sensors, ICWiSe 2019*, 2019-Janua, 1–5. <https://doi.org/10.1109/ICWISE47561.2019.8971831>
- [37] A. Mahir, “Soil Monitoring System using Zigbee for Smart Agriculture,” *IJSTE - Int. J. Sci. Technol. Eng.*, vol. 4, no. 7, pp. 32–38, 2018.
- [38] W. Koodtalang and T. Sangsuwan, “Agricultural Monitoring System with Zigbee Network and PLC based on Modbus RTU Protocol,” *Proc. 2020 Int. Conf. Power, Energy Innov. ICPEI 2020*, no. Icpei, pp. 201–204, 2020, doi: 10.1109/ICPEI49860.2020.9431470.
- [39] O. Ayurzana and S. Tsagaanchuluun, “Monitoring System of Agriculture Fields using ZigBee Modules,” *Int. J. Adv. Smart Converg.*, vol. 10, no. 1, pp. 89–96, 2021, [Online]. Available: <http://dx.doi.org/10.7236/IJASC.2021.101.89>
- [40] M. K. Dharani and K. R. P. Kumar, “Automated Soil Nutrient Content Analysis and Fertilizer Suggestion for Farmers,” *Int. J. Recent Technol. Eng.*, vol. 8, no. 4, pp. 12830–12833, 2019, doi: 10.35940/ijrte.d8004.118419.
- [41] M. S. M. Saleh, A. A. B. Sajak, R. Mohamad, and M. A. M. Zaaba, “IoT Real-Time Soil Monitoring Based on LoRa for Palm Oil Plantation,” *J. Phys. Conf. Ser.*, vol. 1874, no. 1, pp. 1–12, 2021, doi: 10.1088/1742-6596/1874/1/012047.
- [42] L. K. Tolentino et al., “IoT-based automated water monitoring and correcting modular device via LoRaWAN for aquaculture,” *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 533–544, 2021, doi: 10.12785/IJCDS/100151.

[43] M. Johansson, “Soil Moisture Monitoring System Using LoRaWAN Technology,” pp. 1–51, 2021.

ANNEX I

Schematic Diagram

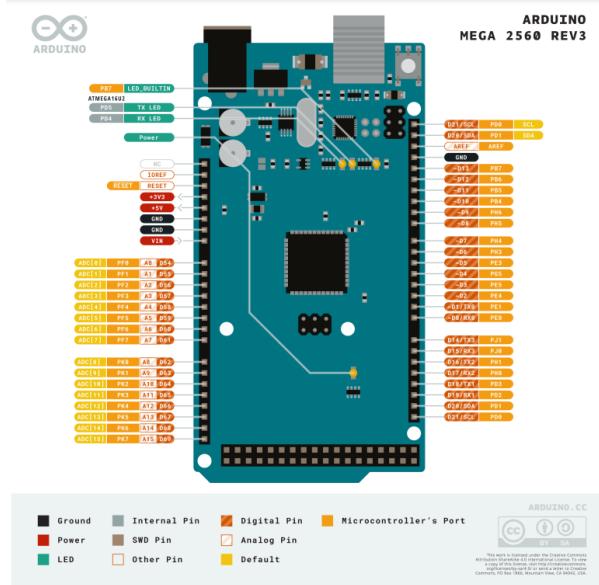
Circuit Diagram of Nodes with Correction System



ANNEX II

Component's Data Sheet

Arduino Mega



Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage ^a	$V_{DD_IO} = 3.3V$	-	-	TBD	V
V_{IH}	Input high voltage ^a	$V_{DD_IO} = 3.3V$	TBD	-	-	V
I_{IL}	Input leakage current	$TA = +85^\circ C$	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage ^b	$V_{DD_IO} = 3.3V, I_{OL} = -2mA$	-	-	TBD	V
V_{OH}	Output high voltage ^b	$V_{DD_IO} = 3.3V, I_{OH} = 2mA$	TBD	-	-	V
I_{OL}	Output low current ^c	$V_{DD_IO} = 3.3V, VO = 0.4V$	TBD	-	-	mA
I_{OH}	Output high current ^c	$V_{DD_IO} = 3.3V, VO = 2.3V$	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	k Ω
R_{PD}	Pulldown resistor	-	TBD	-	TBD	k Ω

Features

ATmega2560 Processor

- Up to 16 MIPS Throughput at 16MHz
- 256k bytes (of which 8k is used for the bootloader)
- 4k bytes EEPROM
- 8k bytes Internal SRAM
- 32 × 8 General Purpose Working Registers
- Real Time Counter with Separate Oscillator
- Four 8-bit PWM Channels
- Four Programmable Serial USART
- Controller/Peripheral SPI Serial Interface

ATmega16U2

- Up to 16 MIPS Throughput at 16 MHz
- 16k bytes ISP Flash Memory
- 512 bytes EEPROM
- 512 bytes SRAM
- USART with SPI master only mode and hardware flow control (RTS/CTS)
- Master/Slave SPI Serial Interface

Sleep Modes

- Idle
- ADC Noise Reduction
- Power-save
- Power-down
- Standby
- Extended Standby

Power

- USB Connection
- External AC/DC Adapter

I/O

- 54 Digital
- 16 Analog
- 15 PWM Output

XBee S2C

Performance Specifications

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C	XBee Zigbee S2D
Indoor/urban range	Up to 60 m (200 ft)	Up to 90 m (300 ft)	Up to 60 m (200 ft)
Outdoor RF line-of-sight range	Up to 1200 m (4000 ft)	Up to 3200 m (2 mi)	Up to 1200 m (4000 ft)
Transmit power output (maximum)	6.3 mW (+8 dBm), boost mode 3.1 mW (+5 dBm), normal mode channel 26 max power is +3 dBm	63 mW (+18 dBm)	6.3 mW (+8 dBm) channel 26 max power is +1 dBm
RF data rate	250,000 b/s		
Receiver sensitivity	-102 dBm, boost mode -100 dBm, normal mode	-101 dBm	-102 dBm, boost mode -100 dBm, normal mode

Power Requirements

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C	XBee Zigbee S2D
Adjustable power	Yes		
Supply voltage	2.1 - 3.6 V 2.2 - 3.6 V for programmable version	2.7 - 3.6 V	2.1 - 3.6 V
Operating current (transmit)	45 mA (+8 dBm, boost mode) 33 mA (+5 dBm, normal mode)	120 mA @ +3.3 V, +18 dBm	45 mA
Operating current (receive)	31 mA (boost mode) 28 mA (normal mode)	31 mA	31 mA
Power-down current	< 1 µA @ 25°C		< 3 uA @ 25°C

General Specifications

Specification	XBee Zigbee S2C	XBee-PRO Zigbee S2C	XBee Zigbee S2D
Operating frequency band	ISM 2.4 - 2.5 GHz		
Form factor	through-hole, surface-mount		surface-mount
Dimensions	through-hole: 2.438 x 2.761 cm (0.960 x 1.087 in) surface-mount: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)	through-hole: 2.438 x 3.294 cm (0.960 x 1.297 in) surface-mount: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)	surface-mount: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)
Operating temperature	-40 to 85 °C (industrial)		
Antenna options	through-hole: PCB antenna, U.FL connector, RPSMA connector, or integrated wire surface-mount: RF pad, PCB antenna, or U.FL connector		

Communication Interface

Interface options	
UART	250 Kb/s maximum
SPI	5 Mb/s maximum (burst)

Raspberry Pi 4 Model B

Features

Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)

- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
 - Up to 6x UART
 - Up to 6x I2C
 - Up to 5x SPI
 - 1x SDIO interface
 - 1x DPI (Parallel RGB Display)
 - 1x PCM
 - Up to 2x PWM channels
 - Up to 3x GPCLK outputs

Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
 - Recent Linux kernel support
 - Many drivers upstreamed
 - Stable and well supported userland
 - Availability of GPU functions using standard APIs

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage ^a	$V_{DD_IO} = 3.3V$	-	-	TBD	V
V_{IH}	Input high voltage ^a	$V_{DD_IO} = 3.3V$	TBD	-	-	V
I_{IL}	Input leakage current	$TA = +85^{\circ}C$	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage ^b	$V_{DD_IO} = 3.3V, I_{OL} = -2mA$	-	-	TBD	V
V_{OH}	Output high voltage ^b	$V_{DD_IO} = 3.3V, I_{OH} = 2mA$	TBD	-	-	V
I_{OL}	Output low current ^c	$V_{DD_IO} = 3.3V, V_O = 0.4V$	TBD	-	-	mA
I_{OH}	Output high current ^c	$V_{DD_IO} = 3.3V, V_O = 2.3V$	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	$k\Omega$
R_{PD}	Pulldown resistor	-	TBD	-	TBD	$k\Omega$

^a Hysteresis enabled

^b Default drive strength (8mA)

^c Maximum drive strength (16mA)

MAX485

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Differential Driver Output (no load)	VOD1				5	V
Differential Driver Output (with load)	VOD2	R = 50Ω (RS-422)	2			V
		R = 27Ω (RS-485), Figure 4	1.5		5	
Change in Magnitude of Driver Differential Output Voltage for Complementary Output States	ΔVOD	R = 27Ω or 50Ω, Figure 4			0.2	V
Driver Common-Mode Output Voltage	VOC	R = 27Ω or 50Ω, Figure 4			3	V
Change in Magnitude of Driver Common-Mode Output Voltage for Complementary Output States	ΔVOD	R = 27Ω or 50Ω, Figure 4			0.2	V
Input High Voltage	VIH	DE, DI, RE	2.0			V
Input Low Voltage	VIL	DE, DI, RE			0.8	V
Input Current	IIN1	DE, DI, RE			±2	μA
Input Current (A, B)	IIN2	DE = 0V; VCC = 0V or 5.25V, all devices except MAX487/MAX1487	VIN = 12V		1.0	mA
			VIN = -7V		-0.8	
		MAX487/MAX1487, DE = 0V, VCC = 0V or 5.25V	VIN = 12V		0.25	mA
			VIN = -7V		-0.2	
Receiver Differential Threshold Voltage	VTH	-7V ≤ VCM ≤ 12V		-0.2	0.2	V
Receiver Input Hysteresis	ΔVTH	VCM = 0V		70		mV
Receiver Output High Voltage	VOH	IO = -4mA, VID = 200mV	3.5			V
Receiver Output Low Voltage	VOL	IO = 4mA, VID = -200mV			0.4	V
Three-State (high impedance) Output Current at Receiver	IOZR	0.4V ≤ VO ≤ 2.4V			±1	μA
Receiver Input Resistance	RIN	-7V ≤ VCM ≤ 12V, all devices except MAX487/MAX1487		12		kΩ
		-7V ≤ VCM ≤ 12V, MAX487/MAX1487		48		kΩ

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
No-Load Supply Current (Note 3)	I _{CC}	MAX488/MAX489, DE, DI, RE = 0V or V _{CC}	120	250		μA
		MAX490/MAX491, DE, DI, RE = 0V or V _{CC}	300	500		
		MAX481/MAX485, RE = 0V or V _{CC}	DE = V _{CC}	500	900	
			DE = 0V	300	500	
		MAX1487, RE = 0V or V _{CC}	DE = V _{CC}	300	500	
			DE = 0V	230	400	
		MAX483/MAX487, RE = 0V or V _{CC}	DE = 5V	350	650	
			MAX483	250	400	
			DE = 0V	120	250	
Supply Current in Shutdown	I _{SHDN}	MAX481/483/487, DE = 0V, RE = V _{CC}	0.1	10		μA
Driver Short-Circuit Current, VO = High	I _{OSD1}	-7V ≤ VO ≤ 12V (Note 4)	35	250		mA
Driver Short-Circuit Current, VO = Low	I _{OSD2}	-7V ≤ VO ≤ 12V (Note 4)	35	250		mA
Receiver Short-Circuit Current	I _{OSR}	0V ≤ VO ≤ V _{CC}	7	95		mA

DHT11

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			± 1%RH	
Accuracy	25 °C		± 4%RH	
	0-50 °C			± 5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0 °C	30%RH		90%RH
	25 °C	20%RH		90%RH
	50 °C	20%RH		80%RH
Response Time (Seconds)	1/e(63%) 25 °C , 1m/s Air	6 S	10 S	15 S
Hysteresis			± 1%RH	
Long-Term Stability	Typical		± 1%RH/year	
Temperature				
Resolution		1 °C	1 °C	1 °C
		8 Bit	8 Bit	8 Bit
Repeatability			± 1 °C	
Accuracy		± 1 °C		± 2 °C
Measurement Range		0 °C		50 °C
Response Time (Seconds)	1/e(63%)	6 S		30 S

VDD=5V, T = 25 °C (unless otherwise stated)

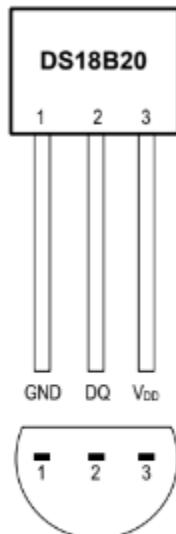
	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

Note: Sampling period at intervals should be no less than 1 second.

BH1750

Parameter	Symbol	Limits			Units	Conditions
		Min.	Typ.	Max.		
Supply Current	Icc1	—	120	190	µA	EV = 100 lx ^{*1}
Powerdown Current	Icc2	—	0.01	1.0	µA	No input Light
Peak Wave Length	λp	—	560	—	nm	
Measurement Accuracy	S/A	0.96	1.2	1.44	times	Sensor out / Actual lx EV = 1000 lx ^{*1, *2}
Dark (0 lx) Sensor out	S0	0	0	3	count	H-Resolution Mode ^{*3}
H-Resolution Mode Resolution	tHR	—	1	—	lx	
L-Resolution Mode Resolution	tLR	—	4	—	lx	
H-Resolution Mode Measurement Time	tHR	—	120	180	ms	
L-Resolution Mode Measurement Time	tLR	—	16	24	ms	
Incandescent / Fluorescent Sensor out ratio	rIF	—	1	—	times	EV = 1000 lx
ADDR Input 'H' Voltage	VAH	0.7 * VCC	—	—	V	
ADDR Input 'L' Voltage	VAL	—	—	0.3 * VCC	V	
DVI Input 'L' Voltage	VDVL	—	—	0.4	V	
SCL, SDA Input 'H' Voltage 1	ViH1	0.7 * DVI	—	—	V	DVI \geq 1.8V
SCL, SDA Input 'H' Voltage 2	ViH2	1.26	—	—	V	1.65V \leq DVI < 1.8V
SCL, SDA Input 'L' Voltage 1	VIL1	—	—	0.3 * DVI	V	DVI \geq 1.8V
SCL, SDA Input 'L' Voltage 2	VIL2	—	—	DVI - 1.26	V	1.65V \leq DVI < 1.8V
SCL, SDA, ADDR Input 'H' Current	IiH	—	—	10	µA	
SCL, SDA, ADDR Input 'L' Current	IiL	—	—	10	µA	
I ² C SCL Clock Frequency	fsCL	—	—	400	kHz	
I ² C Bus Free Time	tbUF	1.3	—	—	µs	
I ² C Hold Time (repeated) START Condition	tHDSTA	0.6	—	—	µs	
I ² C Set up time for a Repeated START Condition	tsUSTA	0.6	—	—	µs	
I ² C Set up time for a Repeated STOP Condition	tsUSTD	0.6	—	—	µs	
I ² C Data Hold Time	tHDDAT	0	—	0.9	µs	
I ² C Data Setup Time	tsUDAT	100	—	—	ns	
I ² C 'L' Period of the SCL Clock	tLOW	1.3	—	—	µs	
I ² C 'H' Period of the SCL Clock	tHIGH	0.6	—	—	µs	
I ² C SDA Output 'L' Voltage	Vol	0	—	0.4	V	IOL = 3 mA

DS18B20



BOTTOM VIEW

**TO-92
(DS18B20)**

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS		
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V		
Pullup Supply Voltage	V_{PU}	Parasite power (Notes 1, 2)		+3.0		+5.5	V		
		Local power		+3.0		V_{DD}			
Thermometer Error	t_{ERR}	-10°C to +85°C -30°C to +100°C -55°C to +125°C		(Note 3)	± 0.5		°C		
					± 1				
					± 2				
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V		
Input Logic-High	V_{IH}	Local power Parasite power		(Notes 1,6)	+2.2	The lower of 5.5 or $V_{DD} + 0.3$	V		
					+3.0				
Sink Current	I_L	$V_{I/O} = 0.4V$		4.0			mA		
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA		
Active Current	I_{DD}	$V_{DD} = 5V$ (Note 9)			1	1.5	mA		
DQ Input Current	I_{DQ}	(Note 10)			5		μA		
Drift		(Note 11)			± 0.2		°C		

Electrical Conductivity Sensor

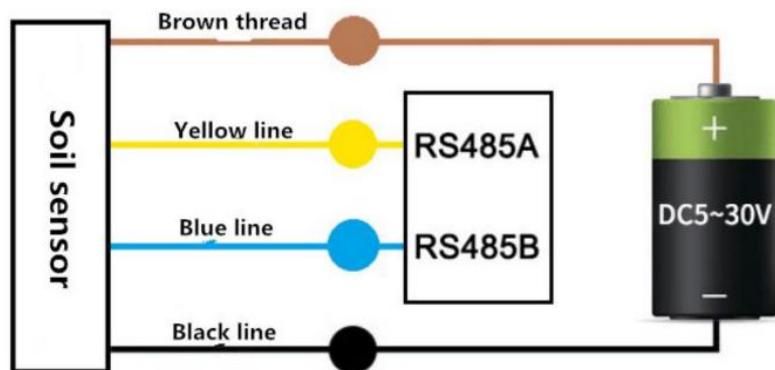


Parameters	Content
Power supply	12-24V DC
Output signal	RS485
Temperature range	-40°C-80°C
Temperature accuracy	±0.5°C
Temperature resolution	0.1°C
Water measurement range	0-100%
Moisture accuracy	±3% Within 0-53%; ±5% within 53-100%
Water resolution	0.10%
Electrical conductivity measurement range	0-10000us/cm
Conductivity resolution	10us/cm
Storage environment	-20°C-60°C
Working Pressure Range	0.9-1.1atm
Response Time	<1s
Protection Level	IP68

PARAMETERS	CONTENT
Protocol	Modbus RTU
Data bits	8 bit
Parity bit	No
Stop bit	1 bit
Error checking	CRC (redundant loop code)
Baud rate	2400 bps/ 4800 bps/ 9600 bps can be set factory defaults to 9600 bps

pH Sensor

Thread color	Explanation	Remarks
brown	Power is positive	5~30V DC
black	Power ground	GND
yellow	485-A	485-A
blue	485-B	485-B



DC power supply (default)	DC 5-30V
Maximum power consumption	0.5W (24V DC powered by)
Range	3—9 PH
Resolution	0.1
Accuracy	±0.3PH
Operating temperature	-20°C~60°C
Long-term stability	≤5%/year
Response time	≤10S
Protection level	IP68
Probe material	Anti-corrosion special electrode
Sealing material	Black flame retardant epoxy resin
Dimensions	45*15*123mm
output signal	RS485(Modbus protocol)

NPK Sensor



Function	Cable Color	Specs
Power	Brown	Power supply +
	Black	Power supply -
Communication	Yellow (grey)	485-A
	Blue	485-B

Parameters	Technical Specs
Measure Range	0-1999mg/kg
Accuracy	±2%F.s
Resolution	1mg/kg(mg/l)
Response Time (T90, Seconds)	<10s
Working Temperature	5-45°C
Working Humidity	5-95%RH (Relative humidity), no condensation
Baud Rate	2400/4800/9600
Communication Port	RS485
Power Supply	12V-24V DC

Parameters	Specs
Coding	8-bit binary
Data bit	8-bit
Parity bit	no
Stop bit	1-bit
Error check	CRC(redundant cyclic code)
Baud Rate	2400bps/4800bps/9600bps.Can customize. Default 9600bps

ANNEX III

Bill of Materials

Cost of Materials for 2 Sensor Nodes

SENSOR NODES			
Qty	Materials	Price/Unit	Total
8	Max485	50	400
2	DHT11	125	250
3	BH175	80	240
2	DS18B20 / Temperature Sensor	100	200
2	Soil Moisture	55	110
2	pH Sensor	5000	10000
2	Electrical Conductivity	3,030.18	6060.32
2	NPK Sensor	3,856.23	7712.12
2	Arduino Mega	849	1698
2	DS3231 RTC module	150	300
2	CR2032	40	80
2	Xbee S2C	1800	3600
2	Xbee USB Adapter	350	700
TOTAL			PHP 31,350.44

Cost of Materials for the Gateway

GATEWAY			
Qty	Materials	Price/Unit	Total
1	Raspberry Pi 4	3800	3800
1	Xbee S2C	1800	1800
1	Xbee USB Adapter	350	350
TOTAL			PHP 5,950

Bill of Materials for the Correction System

CORRECTION SYSTEM			
Qty	Materials	Price/Unit	Total
12	ARD Pump	110	1320
1	100 pcs 3/5mm Threaded Connector Irrigation System	139	139
1	100pcs Drip Irrigation Tee Fittings (4/7 mm Tee Pipe)	149	149
2	24V Humidifier	350	700
1	2 Channel Relay	75	75
1	4 Channel Relay 5V	129	129
20m	Aquarium Hose	8	160
2	Floating Atomizer for Humidifier	99	198
1	Phosphate Organic Fertilizer (0-22-0 + Ca 30) 1kg	57	57
1	Urea 46-0-0 Granular Fertilizer 1 kg (Sources of Nitrogen)	76.31	76.31

1	1kg Lime Organic Fertilizer	26	26
1	260 g 0-0-60 Potash Fertilizer	28	28
1	100 g Sulfur Powder	198	198
12	12V Electronic Solenoid Valve	187	2244
7	Plastic Drum	70	490
2	Epoxy	130	260
TOTAL		PHP 6,249.31	

Bill of Materials for the Power Management System

POWER MANAGEMENT SYSTEM			
Qty	Materials	Price/Unit	Total
1	Boost Converter	100	100
2	Buck Converter	150	139
2	Universal PCB (Small)	25	149
1	100 W Solar Panel	2600	2600
1	30 A Solar Charger Controller	220	220
2	Battery Clamp Terminal	75	150
2	9V Battery Clip	15	30
TOTAL		PHP 3,388	

Cost of Calibration Materials and Tools

CALIBRATOR			
Qty	Materials	Price/Unit	Total
1	NPK Extractant Testing Reagent Kit Set	472	472
1	4 -n- 1 (Soil Tester, PH Meter, Moisture, EC) Tester	399	399
1	Humidity and Temperature Digital Thermometer	323	323
TOTAL		PHP 1,194	

Cost of Other Necessary Materials

OTHER MATERIALS			
Qty	Materials	Price/Unit	Total
1	12V Battery	95	95
2	Triple A Battery	10	20
1	Junction Box 150 x 150 x70mm	125	125
2	Junction Box 200 x 200 x 80mm	195	390
1	Junction Box 255 x 200 x 80 mm	220	220
1	Breadboard	150	150
2	Terminal Block	100	200
1	15m Solid Wires	75	150
10	5pcs Female to Male Jumper Wire	57	570

3	3m Solid Wires	25	75
1	12 x 12 Acrylic sheet	125	125
1	Female to Female Jumper Wire	120	120
1	180 pcs M3 Nylon Standoffs	165	165
TOTAL			PHP 2410

Total Cost of Materials to Develop SOIL-N-WAN

Overall	
Description	Total
Sensor Nodes	31350.44
Gateway	5950
Correction System	3,388
Power Management System	2600
Calibrator	1,194
Other Materials	2410
TOTAL	
PHP 46, 892.44	

ANNEX IV

Expert Evaluation Form

Evaluation Form

Direction: Rate the following by placing a check (✓) mark on the box provided

5 – Excellent 4 – Very Satisfactory 3 – Satisfactory 2 – Fair 1 – Poor

I. General Impact

	5	4	3	2	1
1. Novelty of the Project	✓				
2. Industrial applicability	✓				
3. Societal impact		✓			

II. System

Structural Design	5	4	3	2	1
1. Design structure's effectiveness	✓				
2. Use of appropriate materials	✓				
3. Aesthetics	✓				
Graphical User Interface (GUI)	5	4	3	2	1
1. User-friendliness		✓			
2. Informativeness	✓	.			
3. Aesthetics		✓			

III. Results Evaluation

	5	4	3	2	1
1. Accuracy		✓			
2. Reliability	✓				
3. Effectivity	✓				
4. Intuitive results		✓			

Comments & Suggestion:

Try to test in large scale of Urban Farm and other crops.

Project Leader:

RICKY INDIANO

Evaluated by:

~~LORENZO m. JACINTO JR~~
AgricTourist consultant
Green thumb Assoc

ANNEX V

User Acceptance Test

User Acceptance Test (UAT) questionnaire based on ISO 9001 standards. The questionnaire aims to assess the acceptance and usability of the developed WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System

1. System Functionality

- a. Rate the system 1-5 based on the ability how the device monitors the following parameters:

In the scale of 1-5, 5 is Very Satisfied and 1 is Very Dissatisfied

Parameters	5	4	3	2	1
Ambient Temperature	/				
Humidity	/				
Light Intensity	/				
Electrical Conductivity	/				
Soil Moisture	/				
PH	/				
Soil Temperature	/				
Nitrogen		/			
Phosphorus		/			
Potassium		/			

- b. Did the monitoring application accurately collect and display the real-time data from the nodes?

(Nakuha ba ng monitoring application ang mga datos nang tumpak at nasusunod sa oras galing sa mga nodes?)

Yes

No

- c. Were you able to easily understand and interpret the collected data that flashes on the monitoring application? (Madali mo bang naintindihan at nainterpret ang mga nakolektang datos na nakikita sa monitoring application?)

Yes

No

2. WSN Connection

In the scale of 1-5, 5 is Excellence and 1 is Poor

	5	4	3	2	1
1. How does the system establish a reliable connection between the sensor nodes and gateway using the ZigBee protocol? (Maasahan ba ang koneksyon ng mga sensor nodes at gateway gamit ang Zigbee?)		/			
2. How is the system overall performance in terms of Wireless Sensor Network (WSN) connectivity? (Gaano kaganda ang performance ng proyekto sa paggamit ng WSN?)		/			

3. Monitoring System

- e. Was the Monitoring System Application easy to access and user-friendly? (Madali bang ma-access at madaling gamitin ang Monitoring System Application?)

Yes

No

- f. Did the monitoring system application provide clear and concise data visualization and representation? (Naibigay ba ng monitoring system ang malinaw at maigsi na visualization at representasyon ng data?)

Yes

No

- g. Was the monitoring system application consistent in terms of notification ability? (Naging epektibo ba sa pagaabiso ang monitoring system?)

Yes

No

4. Closed - Loop Correction System

- h. Did the closed-loop correction system effectively adjust the soil's NPK nutrient content? (Naging epektibo ba ang correction system sa pagwawasto ng tamang sukat ng NPK sa lupa?)

Yes

No

- i. Did the closed-loop correction system effectively adjust the required threshold for the soil's moisture required to each plant? (Epektibo bang naayos ng closed-loop correction system ang kinakailangang threshold para sa tamang basa ng lupa na kinakailangan sa bawat halaman?)

Yes

No

- j. Did the closed-loop correction system effectively adjust the soil's pH level? (Epektibo bang naiwasto ng closed-loop correction system ang pH ng lupa?)

Yes

No

5. Testing and Validation

- k. How would you rate the overall performance and accuracy of the device in different scenarios and conditions? (1 - Poor, 5 - Excellent) (Gaano kataas ang mabibigay mo sa pang kalahatan na kalidad at katumpakan ng device sa iba't-ibang senario at kundisyon?)

5 4 3 2 1

- l. Were there any specific scenarios or conditions where you are not satisfied in the device performance? Please provide details. (Meron bang senaryo or kondisyon na kung saan ka hindi na siyahan sa peformance ng proyekto?)

MAHINA ANG BATERY

Questions:

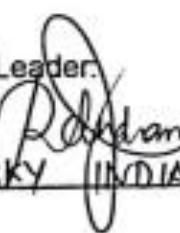
In the scale of 1-5, 5 is Very Likely and 1 is Not Likely

	5	4	3	2	1
1. Rate your overall satisfaction with WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System (I'rate mo ang overall na kaluguran sa pag gamit ng aming proyekto)		/			
2. How likely are you to recommend this system to others? (Gaano mo ni re'rekomenda ang proyekto na ito sa iba?)	/				

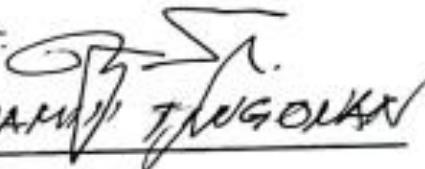
Do you have any suggestions for improvements to the system based on your experience and usage? (Meron ka bang suhestyon para sa ikaka'buti ng system base sa iyong karanasan sa pag gamit?)

LAGAY SA MAS MALAWAK NA
PAGTANIMAM

Project Leader:


RICKY INLAND

End User:


BENJAMIN TUGOLAK
/

User Acceptance Test (UAT) questionnaire based on ISO 9001 standards. The questionnaire aims to assess the acceptance and usability of the developed WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System

1. System Functionality

- a. Rate the system 1-5 based on the ability how the device monitors the following parameters:

In the scale of 1-5, 5 is Very Satisfied and 1 is Very Dissatisfied

Parameters	5	4	3	2	1
Ambient Temperature	✓				
Humidity	✓				
Light Intensity	✓				
Electrical Conductivity	✓				
Soil Moisture	✓				
PH	/				
Soil Temperature	✓				
Nitrogen	✓				
Phosphorus	✓				
Potassium	✓				

- b. Did the monitoring application accurately collect and display the real-time data from the nodes?

(Nakuha ba ng monitoring application ang mga datos nang tumpak at nasusunod sa oras galing sa mga nodes?)

Yes

No

- c. Were you able to easily understand and interpret the collected data that flashes on the monitoring application? (Madali mo bang naintindihan at nainterpret ang mga nakolektang datos na nakikita sa monitoring application?)

Yes

No

2. WSN Connection

In the scale of 1-5, 5 is Excellence and 1 is Poor

	5	4	3	2	1
1. How does the system establish a reliable connection between the sensor nodes and gateway using the ZigBee protocol? (Maasahan ba ang koneksyon ng mga sensor nodes at gateway gamit ang Zigbee?)	/				
2. How is the system overall performance in terms of Wireless Sensor Network (WSN) connectivity? (Gaano kaganda ang performance ng proyekto sa paggamit ng WSN?)	✓				

3. Monitoring System

- e. Was the Monitoring System Application easy to access and user-friendly? (Madali bang ma-access at madaling gamitin ang Monitoring System Application?)

Yes

No

- f. Did the monitoring system application provide clear and concise data visualization and representation? (Naibigay ba ng monitoring system ang malinaw at maigsi na visualization at representasyon ng data?)

Yes

No

- g. Was the monitoring system application consistent in terms of notification ability? (Naging epektibo ba sa pagaabiso ang monitoring system?)

Yes

No

4. Closed - Loop Correction System

- h. Did the closed-loop correction system effectively adjust the soil's NPK nutrient content? (Naging epektibo ba ang correction system sa pagwawasto ng tamang sukat ng NPK sa lupa?)

Yes

No

- i. Did the closed-loop correction system effectively adjust the required threshold for the soil's moisture required to each plant? (Epektibo bang naayos ng closed-loop correction system ang kinakailangang threshold para sa tamang basa ng lupa na kinakailangan sa bawat halaman?)

Yes

No

- j. Did the closed-loop correction system effectively adjust the soil's pH level? (Epektibo bang nailwasto ng closed-loop correction system ang pH ng lupa?)

Yes

No

5. Testing and Validation

- k. How would you rate the overall performance and accuracy of the device in different scenarios and conditions? (1 - Poor, 5 - Excellent) (Gaano kataas ang mabilbigay mo sa pang kalahatan na kalidad at katumpakan ng device sa iba't-ibang senario at kundisyon?)

5 4 3 2 1

- l. Were there any specific scenarios or conditions where you are not satisfied in the device performance? Please provide details. (Meron bang senaryo or kondisyon na kung saan ka hindi na siyahan sa performance ng proyekto?)

nalolowbat po ang system pag maylan

Questions:

In the scale of 1-5, 5 is Very Likely and 1 is Not Likely

	5	4	3	2	1
1. Rate your overall satisfaction with WSN-based Smart Farming through Soil Health and Ambience Condition Analysis with Monitoring and Correction System (I'rate mo ang overall na kaluguran sa pag gamit ng aming proyekto)	✓				
2. How likely are you to recommend this system to others? (Gaano mo ni re'rekomenda ang proyekto na ito sa iba?)	✓				

Do you have any suggestions for improvements to the system based on your experience and usage? (Meron ka bang suhestyon para sa ikaka'buti ng system base sa iyong karanasan sa pag gamit?)

needs po ang may malawak at capat na aray.

Project Leader:

End User:

BERNADETTE C. BELTRAN

Pn Bernadette

GREEN THUMB URBAN PARA
BAONG PAG-AQA
Q-C.

ANNEX VI

SOIL-N-WAN Node Codes

Arduino Codes of Node 1 Tomato

```
#include <DS3231.h>
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <BH1750.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "fis_header.h"

#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define ONE_WIRE_PIN 5
#define RE 6
#define DE 7
#define RF 52
#define DF 53
#define RG 8
#define DG 9
#define FIS_TYPE float
#define FIS_RESOLUTION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);
typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_ARR_OP);

SoftwareSerial mod1(12,13);
SoftwareSerial mod2(10,11);
SoftwareSerial mod3(50,51);
const byte ec[] = {0x01, 0x03, 0x00, 0x15, 0x00, 0x01, 0x95, 0xCE};
const byte ph[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x84, 0x0A};
const byte nitro[] = {0x01, 0x03, 0x00, 0x1e, 0x00, 0x01, 0xe4, 0x0c};
const byte phos[] = {0x01, 0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc};
```

```
const byte pota[] = {0x01, 0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xc0};
byte value[11];

// Number of inputs to the fuzzy inference system
const int fis_gcI = 8;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 7;
// Number of rules to the fuzzy inference system
const int fis_gcR = 48;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);
BH1750 lightMeter;
DS3231 myRTC;

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

int prev=0;
int pres=0;

float Value_EC = 0;
float Value_PH = 0;
float Value_N = 0;
float Value_Pho = 0;
float Value_Pot = 0;
float Value_Ambient_Temp = 0;
float Value_Humidity = 0;
float Value_Light_Int = 0;
float Value_Soil_Temp = 0;
float Value_Moisture = 0;
float Water_On = 0;
float Correction_On = 0;
float Humidifier_On = 0;
```

```

byte year;
byte month;
byte date;
byte dow;
byte hour;
byte minute;
byte second;

bool century = false;
bool h12Flag;
bool pmFlag;

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

  mod1.begin(9600);
  mod2.begin(4800);
  mod3.begin(9600);

  Wire.begin();
  lightMeter.begin();
  dht.begin();
  sensors.begin();

  myRTC.setClockMode(false); // set to 24h

  myRTC.setYear(23);
  myRTC.setMonth(5);
  myRTC.setDate(15);
  myRTC.setDoW(2);
  myRTC.setHour(15); //IIBAHIN KUNG
ANONG ORAS NA
  myRTC.setMinute(50);
  myRTC.setSecond(30);

  pinMode(RE, OUTPUT);
  pinMode(DE, OUTPUT);
  pinMode(RG, OUTPUT);
  pinMode(DG, OUTPUT);
  pinMode(RF, OUTPUT);
  pinMode(DF, OUTPUT);
  delay(1000);

  // initialize the Analog pins for output.
  // Pin mode for Output: Humidifier
  pinMode(31, OUTPUT);

  // Pin mode for Output: Drip_Irrig
  pinMode(32, OUTPUT);
  // Pin mode for Output: Lime
  pinMode(34, OUTPUT);
  // Pin mode for Output: Sulfur
  pinMode(33, OUTPUT);
  // Pin mode for Output: Nitrogen
  pinMode(35, OUTPUT);
  // Pin mode for Output: Phosphorus
  pinMode(36, OUTPUT);
  // Pin mode for Output: Potassium
  pinMode(37, OUTPUT);

}

void loop() {
  SetRelay();
  Soil_EC();
  Soil_PH();
  Soil_NPK();
  Soil_Temperature();
  Soil_Moisture();
  Ambient_Temperature();
  Ambient_Humidity();
  Light_Intensity();
  Serial_Print();
  Alarm();
  delay(50000);
  SetOff();
}

void Serial_Print(){
  Serial.println(" "+ String("Node1") +
"+String(Value_Humidity)+" "+
String(Value_Ambient_Temp)+" "+
"+String(Value_Soil_Temp)+" "+
String(Value_Light_Int)+" "+
String(Value_Moisture)+" "+
"+String(Value_EC)+" "+
String(Value_PH)+" "+ String(Value_N)+" "+
"+ String(Value_Pho)+" "+
String(Value_Pot)+" "+
(myRTC.getMonth(century))+" "+
(myRTC.getDate())+" "+
(myRTC.getDoW())+" "+
(myRTC.getHour(h12Flag, pmFlag))+ " "+

```



```

        value[iii] = mod3.read();
    }
}

return value[4];
}

void Soil_Temperature(){
    sensors.requestTemperatures();
    Value_Soil_Temp =
    sensors.getTempCByIndex(0);
}

void Soil_Moisture(){
    Value_Moisture=analogRead(A15);

    Value_Moisture=map(Value_Moisture,0,982
    ,148,0);
    pres=Value_Moisture;
    if(Value_Moisture>100)
    Value_Moisture=100;
    else if(Value_Moisture<0)
    Value_Moisture=0;
    prev=Value_Moisture;
}

void Ambient_Humidity(){
    Value_Humidity = dht.readHumidity() + 3;
//+3
}

void Ambient_Temperature(){
    Value_Ambient_Temp =
    dht.readTemperature() - 1; // -1
}

void Light_Intensity(){
    Value_Light_Int =
    lightMeter.readLightLevel();
}

void Water(){
    Water_On = 1;
    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Phosphorus;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Drip_Irrig
    if ((g_fisOutput[1] > 0) &&
(g_fisOutput[1] <= 5))
    {
        digitalWrite(32, HIGH);
        delay(300000); //TRIAL
    }

    if ((g_fisOutput[1] > 5) &&
(g_fisOutput[1] <= 7))
    {
        digitalWrite(32, LOW);
        delay(20000);
        digitalWrite(32, HIGH);
        delay(300000); //TRIAL
    }

    if ((g_fisOutput[1] > 7) &&
(g_fisOutput[1] > 9))
    {
        digitalWrite(32, LOW);
        delay(40000);
    }
}

```

```

digitalWrite(32, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[1] > 9)
{
digitalWrite(32, LOW);
delay(60000);
digitalWrite(32, HIGH);
delay(300000); //TRIAL
}
}

void Humidifier(){
    Humidifier_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Pho;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Humidifier
}

if ((g_fisOutput[0] > 0) &&
(g_fisOutput[0] <= 5))
{
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

if ((g_fisOutput[0] > 5) &&
(g_fisOutput[0] <= 7))
{
digitalWrite(31, HIGH);
delay(60000);
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

if ((g_fisOutput[0] > 7) &&
(g_fisOutput[0] <= 9))
{
digitalWrite(31, HIGH);
delay(120000);
digitalWrite(31, LOW);
delay(300000);
}

if (g_fisOutput[0] > 9)
{
digitalWrite(31, HIGH);
delay(180000);
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

void Correction()
{
    Correction_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
}

```

```

// Read Input: Soil_pH
g_fisInput[4] = Value_PH;
// Read Input: Nitrogen
g_fisInput[5] = Value_N;
// Read Input: Phosphorus
g_fisInput[6] = Value_Pho;
// Read Input: Potassium
g_fisInput[7] = Value_Pot;

g_fisOutput[0] = 0;
g_fisOutput[1] = 0;
g_fisOutput[2] = 0;
g_fisOutput[3] = 0;
g_fisOutput[4] = 0;
g_fisOutput[5] = 0;
g_fisOutput[6] = 0;

fis_evaluate();

// Set output vlaue: Lime
if ((g_fisOutput[2] > 0) &&
(g_fisOutput[2] <= 3.332))
{
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[2] > 3.332) &&
(g_fisOutput[2] <= 6.663))
{
    digitalWrite(34, LOW);
    delay(20000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[2] > 6.663)
{
    digitalWrite(34, LOW);
    delay(40000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Sulfur

```

```

if ((g_fisOutput[3] > 0) &&
(g_fisOutput[3] <= 3.332))
{
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[3] > 3.332) &&
(g_fisOutput[3] <= 6.663))
{
    digitalWrite(33, LOW);
    delay(20000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[3] > 6.663)
{
    digitalWrite(33, LOW);
    delay(40000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Nitrogen
if ((g_fisOutput[4] > 0) &&
(g_fisOutput[4] <= 3.332))
{
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[4] > 3.332) &&
(g_fisOutput[4] <= 6.663))
{
    digitalWrite(35, LOW);
    delay(20000);
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[4] > 6.663)
{

```

```

digitalWrite(35, LOW);
delay(40000);
digitalWrite(35, HIGH);
delay(300000); //TRIAL
}

// Set output vlaue: Phosphorus
if ((g_fisOutput[5] > 0) &&
(g_fisOutput[5] <= 3.332))
{
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[5] > 3.332) &&
(g_fisOutput[5] <= 6.663))
{
digitalWrite(36, LOW);
delay(20000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[5] > 6.663)
{
digitalWrite(36, LOW);
delay(40000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

// Set output vlaue: Potassium
if ((g_fisOutput[6] > 0) &&
(g_fisOutput[6] <= 3.332))
{
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[6] > 3.332) &&
(g_fisOutput[6] <= 6.663))
{
digitalWrite(37, LOW);

delay(20000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

}

//MAGKAIBANG TIME NODE 1 NODE
2()
void Alarm(){
uint8_t checkHour =
myRTC.getHour(h12Flag, pmFlag);
uint8_t checkDoW = myRTC.getDoW();

if (checkHour == 9) {
Water();
//digitalWrite(31, HIGH);
} if (checkHour == 17 & (checkDoW != 3 | checkDoW != 6)) {
Water();
}
if (checkDoW == 3 & checkHour == 16) {
Correction();
delay(600000);
} if (checkDoW == 6 & checkHour == 16) {
Correction();
delay(600000);
} //if (checkHour == 16) {
// digitalWrite(31, LOW);
// } if (checkHour == 17) {
// Humidifier();
// }
}

void SetRelay(){
digitalWrite(31,LOW);

```

```

digitalWrite(32,HIGH);
digitalWrite(33,HIGH);
digitalWrite(34,HIGH);
digitalWrite(35,HIGH);
digitalWrite(36,HIGH);
digitalWrite(37,HIGH);
}

void SetOff(){
    Water_On = 0;
    Correction_On = 0;
    Humidifier_On = 0;
}

//*****
// Support functions for Fuzzy Inference
System
//*****
/*
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d
= p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0
: ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0
: ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis.trimf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return
(FIS_TYPE) (x == a);
}

if (a == b) return (FIS_TYPE) (t2*(b <=
x)*(x <= c));
if (b == c) return (FIS_TYPE) (t1*(a <=
x)*(x <= b));
t1 = min(t1, t2);
return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a,
FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a,
FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE
*array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}

//*****
// Data for Fuzzy Inference System
//*****
/*

```

```

// Pointers to the implementations of
// member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

// Count of member function for each Input
int fis_gIMFCount[] = { 3, 3, 4, 3, 5, 3, 3, 3
};

// Count of member function for each Output
int fis_gOMFCount[] = { 4, 4, 3, 3, 3, 3, 3 };

// Coefficients for the Input Member
// Functions
FIS_TYPE fis_gMFI0Coeff1[] = { 0, 0,
23.5, 28 };
FIS_TYPE fis_gMFI0Coeff2[] = { 24, 27.5,
31 };
FIS_TYPE fis_gMFI0Coeff3[] = { 27, 31.5,
50, 50 };
FIS_TYPE* fis_gMFI0Coeff[] = {
fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { 0, 0, 55,
67 };
FIS_TYPE fis_gMFI1Coeff2[] = { 63, 75,
87 };
FIS_TYPE fis_gMFI1Coeff3[] = { 83, 95,
100, 100 };
FIS_TYPE* fis_gMFI1Coeff[] = {
fis_gMFI1Coeff1, fis_gMFI1Coeff2,
fis_gMFI1Coeff3 };
FIS_TYPE fis_gMFI2Coeff1[] = { 63, 72.5,
82 };
FIS_TYPE fis_gMFI2Coeff2[] = { 78, 86,
100, 100 };
FIS_TYPE fis_gMFI2Coeff3[] = { 0, 0, 20,
34.5 };
FIS_TYPE fis_gMFI2Coeff4[] = { 30.5,
48.75, 67 };
FIS_TYPE* fis_gMFI2Coeff[] = {
fis_gMFI2Coeff1, fis_gMFI2Coeff2,
fis_gMFI2Coeff3, fis_gMFI2Coeff4 };

FIS_TYPE fis_gMFI3Coeff1[] = { 0, 0, 13,
20 };
FIS_TYPE fis_gMFI3Coeff2[] = { 16, 23.5,
31 };
FIS_TYPE fis_gMFI3Coeff3[] = { 27, 34,
50, 50 };
FIS_TYPE* fis_gMFI3Coeff[] = {
fis_gMFI3Coeff1, fis_gMFI3Coeff2,
fis_gMFI3Coeff3 };
FIS_TYPE fis_gMFI4Coeff1[] = { 0, 0, 1.5,
2.65 };
FIS_TYPE fis_gMFI4Coeff2[] = { 1.65,
3.225, 4.8 };
FIS_TYPE fis_gMFI4Coeff3[] = { 8.95,
11.75, 14.6, 19.3 };
FIS_TYPE fis_gMFI4Coeff4[] = { 4.4,
7.175, 9.95 };
FIS_TYPE fis_gMFI4Coeff5[] = { 3.8, 4.6,
5.4 };
FIS_TYPE* fis_gMFI4Coeff[] = {
fis_gMFI4Coeff1, fis_gMFI4Coeff2,
fis_gMFI4Coeff3, fis_gMFI4Coeff4,
fis_gMFI4Coeff5 };
FIS_TYPE fis_gMFI5Coeff1[] = { 0, 0, 5,
10 };
FIS_TYPE fis_gMFI5Coeff2[] = { 8, 13.5,
19 };
FIS_TYPE fis_gMFI5Coeff3[] = { 17, 22,
30, 30 };
FIS_TYPE* fis_gMFI5Coeff[] = {
fis_gMFI5Coeff1, fis_gMFI5Coeff2,
fis_gMFI5Coeff3 };
FIS_TYPE fis_gMFI6Coeff1[] = { 0, 0, 5,
10 };
FIS_TYPE fis_gMFI6Coeff2[] = { 8, 13.5,
19 };
FIS_TYPE fis_gMFI6Coeff3[] = { 17, 22,
30, 30 };
FIS_TYPE* fis_gMFI6Coeff[] = {
fis_gMFI6Coeff1, fis_gMFI6Coeff2,
fis_gMFI6Coeff3 };
FIS_TYPE fis_gMFI7Coeff1[] = { 0, 0, 5.5,
11.5 };
FIS_TYPE fis_gMFI7Coeff2[] = { 9.5,
15.75, 22 };

```

```

FIS_TYPE fis_gMFI7Coeff3[] = { 20, 26,
30, 30 };
FIS_TYPE* fis_gMFI7Coeff[] = {
fis_gMFI7Coeff1, fis_gMFI7Coeff2,
fis_gMFI7Coeff3 };
FIS_TYPE** fis_gMFICoeff[] = {
fis_gMFI0Coeff, fis_gMFI1Coeff,
fis_gMFI2Coeff, fis_gMFI3Coeff,
fis_gMFI4Coeff, fis_gMFI5Coeff,
fis_gMFI6Coeff, fis_gMFI7Coeff };

// Coefficients for the Output Member
Functions
FIS_TYPE fis_gMFO0Coeff1[] = { 4, 6, 8
};
FIS_TYPE fis_gMFO0Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO0Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO0Coeff4[] = { 0, 0, 3,
5 };
FIS_TYPE* fis_gMFO0Coeff[] = {
fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3, fis_gMFO0Coeff4 };
FIS_TYPE fis_gMFO1Coeff1[] = { 0, 0, 3,
6 };
FIS_TYPE fis_gMFO1Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO1Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO1Coeff4[] = { 4, 6, 8
};
FIS_TYPE* fis_gMFO1Coeff[] = {
fis_gMFO1Coeff1, fis_gMFO1Coeff2,
fis_gMFO1Coeff3, fis_gMFO1Coeff4 };
FIS_TYPE fis_gMFO2Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO2Coeff2[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE fis_gMFO2Coeff3[] = { 2.833,
5, 7.166 };
FIS_TYPE* fis_gMFO2Coeff[] = {
fis_gMFO2Coeff1, fis_gMFO2Coeff2,
fis_gMFO2Coeff3 };
FIS_TYPE fis_gMFO3Coeff1[] = { 0, 0, 2,
3.83 };

```

```

FIS_TYPE fis_gMFO3Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO3Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO3Coeff[] = {
fis_gMFO3Coeff1, fis_gMFO3Coeff2,
fis_gMFO3Coeff3 };
FIS_TYPE fis_gMFO4Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO4Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO4Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO4Coeff[] = {
fis_gMFO4Coeff1, fis_gMFO4Coeff2,
fis_gMFO4Coeff3 };
FIS_TYPE fis_gMFO5Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO5Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO5Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO5Coeff[] = {
fis_gMFO5Coeff1, fis_gMFO5Coeff2,
fis_gMFO5Coeff3 };
FIS_TYPE fis_gMFO6Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO6Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO6Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO6Coeff[] = {
fis_gMFO6Coeff1, fis_gMFO6Coeff2,
fis_gMFO6Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = {
fis_gMFO0Coeff, fis_gMFO1Coeff,
fis_gMFO2Coeff, fis_gMFO3Coeff,
fis_gMFO4Coeff, fis_gMFO5Coeff,
fis_gMFO6Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 1, 0 };
int fis_gMFI1[] = { 0, 1, 0 };
int fis_gMFI2[] = { 1, 0, 0, 1 };
int fis_gMFI3[] = { 0, 1, 0 };
int fis_gMFI4[] = { 0, 1, 0, 1, 1 };

```

```

int fis_gMFI5[] = { 0, 1, 0 };
int fis_gMFI6[] = { 0, 1, 0 };
int fis_gMFI7[] = { 0, 1, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1,
fis_gMFI2, fis_gMFI3, fis_gMFI4,
fis_gMFI5, fis_gMFI6, fis_gMFI7};

// Output membership function set
int fis_gMFO0[] = { 1, 1, 0, 0 };
int fis_gMFO1[] = { 0, 1, 0, 1 };
int fis_gMFO2[] = { 0, 0, 1 };
int fis_gMFO3[] = { 0, 1, 0 };
int fis_gMFO4[] = { 0, 1, 0 };
int fis_gMFO5[] = { 0, 1, 0 };
int fis_gMFO6[] = { 0, 1, 0 };
int* fis_gMFO[] = { fis_gMFO0,
fis_gMFO1, fis_gMFO2, fis_gMFO3,
fis_gMFO4, fis_gMFO5, fis_gMFO6};

// Rule Weights
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Type
int fis_gRTYPE[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI1[] = { 2, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI2[] = { 1, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI3[] = { 2, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI4[] = { 2, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI5[] = { 3, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI6[] = { 3, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI7[] = { 1, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI8[] = { 3, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI9[] = { 0, 0, 2, 1, 0, 0, 0, 0, 0 };
int fis_gRI10[] = { 0, 0, 2, 2, 0, 0, 0, 0, 0 };
int fis_gRI11[] = { 0, 0, 2, 3, 0, 0, 0, 0, 0 };
int fis_gRI12[] = { 0, 0, 1, 1, 0, 0, 0, 0, 0 };
int fis_gRI13[] = { 0, 0, 1, 2, 0, 0, 0, 0, 0 };

```

```
int fis_gRI14[] = { 0, 0, 1, 3, 0, 0, 0, 0 };  
int fis_gRI15[] = { 0, 0, 4, 1, 0, 0, 0, 0 };  
int fis_gRI16[] = { 0, 0, 4, 2, 0, 0, 0, 0 };  
int fis_gRI17[] = { 0, 0, 4, 3, 0, 0, 0, 0 };  
int fis_gRI18[] = { 0, 0, 3, 1, 0, 0, 0, 0 };  
int fis_gRI19[] = { 0, 0, 3, 2, 0, 0, 0, 0 };  
int fis_gRI20[] = { 0, 0, 3, 3, 0, 0, 0, 0 };  
int fis_gRI21[] = { 0, 0, 0, 1, 1, 0, 0, 0 };  
int fis_gRI22[] = { 0, 0, 0, 1, 2, 0, 0, 0 };  
int fis_gRI23[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI24[] = { 0, 0, 0, 2, 1, 0, 0, 0 };  
int fis_gRI25[] = { 0, 0, 0, 2, 2, 0, 0, 0 };  
int fis_gRI26[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI27[] = { 0, 0, 0, 3, 1, 0, 0, 0 };  
int fis_gRI28[] = { 0, 0, 0, 3, 2, 0, 0, 0 };  
int fis_gRI29[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI30[] = { 0, 0, 0, 1, 3, 0, 0, 0 };  
int fis_gRI31[] = { 0, 0, 0, 1, 4, 0, 0, 0 };  
int fis_gRI32[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI33[] = { 0, 0, 0, 2, 3, 0, 0, 0 };  
int fis_gRI34[] = { 0, 0, 0, 2, 4, 0, 0, 0 };  
int fis_gRI35[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI36[] = { 0, 0, 0, 3, 3, 0, 0, 0 };  
int fis_gRI37[] = { 0, 0, 0, 3, 4, 0, 0, 0 };  
int fis_gRI38[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI39[] = { 0, 0, 0, 0, 0, 1, 0, 0 };  
int fis_gRI40[] = { 0, 0, 0, 0, 0, 2, 0, 0 };  
int fis_gRI41[] = { 0, 0, 0, 0, 0, 3, 0, 0 };  
int fis_gRI42[] = { 0, 0, 0, 0, 0, 0, 1, 0 };  
int fis_gRI43[] = { 0, 0, 0, 0, 0, 0, 2, 0 };  
int fis_gRI44[] = { 0, 0, 0, 0, 0, 0, 3, 0 };  
int fis_gRI45[] = { 0, 0, 0, 0, 0, 0, 0, 1 };  
int fis_gRI46[] = { 0, 0, 0, 0, 0, 0, 0, 2 };  
int fis_gRI47[] = { 0, 0, 0, 0, 0, 0, 0, 3 };  
int* fis_gRI[] = { fis_gRI0, fis_gRI1,  
    fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5,  
    fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9,  
    fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13,  
    fis_gRI14, fis_gRI15, fis_gRI16, fis_gRI17,  
    fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21,  
    fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25,  
    fis_gRI26, fis_gRI27, fis_gRI28, fis_gRI29,  
    fis_gRI30, fis_gRI31, fis_gRI32, fis_gRI33,  
    fis_gRI34, fis_gRI35, fis_gRI36, fis_gRI37,  
    fis_gRI38, fis_gRI39, fis_gRI40, fis_gRI41,
```

```
fis_gRI42, fis_gRI43, fis_gRI44, fis_gRI45,
fis_gRI46, fis_gRI47 };
```

// Rule Outputs

```
int fis_gRO0[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO1[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO2[] = { 4, 0, 0, 0, 0, 0, 0 };
int fis_gRO3[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO4[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO5[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO6[] = { 3, 0, 0, 0, 0, 0, 0 };
int fis_gRO7[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO8[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO9[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO10[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO11[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO12[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO13[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO14[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO15[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO16[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO17[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO18[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO19[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO20[] = { 0, 3, 0, 0, 0, 0, 0 };
int fis_gRO21[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO22[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO23[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO24[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO25[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO26[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO27[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO28[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO29[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO30[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO31[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO32[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO33[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO34[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO35[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO36[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO37[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO38[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO39[] = { 0, 0, 0, 0, 3, 0, 0 };
int fis_gRO40[] = { 0, 0, 0, 0, 2, 0, 0 };
int fis_gRO41[] = { 0, 0, 0, 0, 1, 0, 0 };
```

```
int fis_gRO42[] = { 0, 0, 0, 0, 0, 3, 0 };
int fis_gRO43[] = { 0, 0, 0, 0, 0, 2, 0 };
int fis_gRO44[] = { 0, 0, 0, 0, 0, 1, 0 };
int fis_gRO45[] = { 0, 0, 0, 0, 0, 0, 3 };
int fis_gRO46[] = { 0, 0, 0, 0, 0, 0, 2 };
int fis_gRO47[] = { 0, 0, 0, 0, 0, 0, 1 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1,
fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9,
fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15,
fis_gRO16, fis_gRO17, fis_gRO18,
fis_gRO19, fis_gRO20, fis_gRO21,
fis_gRO22, fis_gRO23, fis_gRO24,
fis_gRO25, fis_gRO26, fis_gRO27,
fis_gRO28, fis_gRO29, fis_gRO30,
fis_gRO31, fis_gRO32, fis_gRO33,
fis_gRO34, fis_gRO35, fis_gRO36,
fis_gRO37, fis_gRO38, fis_gRO39,
fis_gRO40, fis_gRO41, fis_gRO42,
fis_gRO43, fis_gRO44, fis_gRO45,
fis_gRO46, fis_gRO47 };
```

// Input range Min

```
FIS_TYPE fis_gIMin[] = { 0, 0, 0, 0, 0, 0, 0,
0 };
```

// Input range Max

```
FIS_TYPE fis_gIMax[] = { 50, 100, 100,
50, 14, 30, 30, 30 };
```

// Output range Min

```
FIS_TYPE fis_gOMin[] = { 0, 0, 0, 0, 0, 0,
0 };
```

// Output range Max

```
FIS_TYPE fis_gOMax[] = { 15, 15, 10, 10,
10, 10, 10 };
```

```
*****  
*****  
*
```

// Data dependent support functions for
Fuzzy Inference System

```

//*****
***** *
FIS_TYPE fis_MF_out(FIS_TYPE**
fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut =
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 -
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut,
fuzzyRuleSet[1][r]);
    }
    return
fis_array_operation(fuzzyRuleSet[0],
fis_gcR, fis_max);
}

FIS_TYPE
fis_defuzz_centroid(FIS_TYPE**
fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] -
fis_gOMin[o]) / (FIS_RESOLUTION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve
    formed by the MF outputs
    for (i = 0; i < FIS_RESOLUTION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step *
fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] +
fis_gOMin[o]) / 2) : (momentum / area));
}

//*****
***** *
// Fuzzy Inference System
//*****
***** *
***** *
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput2[] = { 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput3[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput4[] = { 0, 0, 0, 0, 0
};;
    FIS_TYPE fuzzyInput5[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput6[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput7[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] = {
fuzzyInput0, fuzzyInput1, fuzzyInput2,
fuzzyInput3, fuzzyInput4, fuzzyInput5,
fuzzyInput6, fuzzyInput7, };

    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput2[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput3[] = { 0, 0, 0 };
}

```

```

FIS_TYPE fuzzyOutput4[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput5[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput6[] = { 0, 0, 0 };
FIS_TYPE* fuzzyOutput[fis_gcO] = {
fuzzyOutput0, fuzzyOutput1, fuzzyOutput2,
fuzzyOutput3, fuzzyOutput4, fuzzyOutput5,
fuzzyOutput6, };
FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
FIS_TYPE* fuzzyRuleSet[] = {
fuzzyRules, fuzzyFires };
FIS_TYPE sW = 0;

// Transforming input to fuzzy Input
int i, j, r, o;
for (i = 0; i < fis_gcI; ++i)
{
    for (j = 0; j < fis_gIMFCount[i]; ++j)
    {
        fuzzyInput[i][j] =
(fis_gMF[fis_gMFI[i][j]])(g_fisInput[i],
fis_gMFICoeff[i][j]);
    }
}

int index = 0;
for (r = 0; r < fis_gcR; ++r)
{
    if (fis_gRType[r] == 1)
    {
        fuzzyFires[r] = FIS_MAX;
        for (i = 0; i < fis_gcI; ++i)
        {
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], fuzzyInput[i][index -
1]);
            else if (index < 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
            else
}
}
}
else
{
    fuzzyFires[r] = fis_min(fuzzyFires[r], 1);
}
}
else
{
    fuzzyFires[r] = FIS_MIN;
    for (i = 0; i < fis_gcI; ++i)
    {
        index = fis_gRI[r][i];
        if (index > 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
        else if (index < 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
        else
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 0);
    }
}
}
else
{
    fuzzyFires[r] = fis_gRWeight[r] *
fuzzyFires[r];
    sW += fuzzyFires[r];
}

if (sW == 0)
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] = ((fis_gOMax[o] +
fis_gOMin[o]) / 2);
    }
}
else
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] =
fis_defuzz_centroid(fuzzyRuleSet, o);
    }
}
}

```

Arduino Codes of Node 2 Pechay

```
#include <DS3231.h>
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <BH1750.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "fis_header.h"

#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define ONE_WIRE_PIN 5
#define RE 6
#define DE 7
#define RF 52
#define DF 53
#define RG 8
#define DG 9
#define FIS_TYPE float
#define FIS_RESOLUTION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);
typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_ARR_OP);

SoftwareSerial mod1(12,13);
SoftwareSerial mod2(10,11);
SoftwareSerial mod3(50,51);
const byte ec[] = {0x01, 0x03, 0x00, 0x15, 0x00, 0x01, 0x95, 0xCE};
const byte ph[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x84, 0x0A};
const byte nitro[] = {0x01, 0x03, 0x00, 0x1e, 0x00, 0x01, 0xe4, 0x0c};
const byte phos[] = {0x01, 0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc};

const byte pota[] = {0x01, 0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xc0};
byte value[11];

// Number of inputs to the fuzzy inference system
const int fis_gcI = 8;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 7;
// Number of rules to the fuzzy inference system
const int fis_gcR = 48;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);
BH1750 lightMeter;
DS3231 myRTC;

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

int prev=0;
int pres=0;

float Value_EC = 0;
float Value_PH = 0;
float Value_N = 0;
float Value_Pho = 0;
float Value_Pot = 0;
float Value_Ambient_Temp = 0;
float Value_Humidity = 0;
float Value_Light_Int= 0;
float Value_Soil_Temp = 0;
float Value_Moisture = 0;
float Water_On = 0;
float Correction_On = 0;
float Humidifier_On = 0;
```

```

byte year;
byte month;
byte date;
byte dow;
byte hour;
byte minute;
byte second;

bool century = false;
bool h12Flag;
bool pmFlag;

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

  mod1.begin(9600);
  mod2.begin(4800);
  mod3.begin(9600);

  Wire.begin();
  lightMeter.begin();
  dht.begin();
  sensors.begin();

  myRTC.setClockMode(false); // set to 24h

  myRTC.setYear(23);
  myRTC.setMonth(5);
  myRTC.setDate(15);
  myRTC.setDoW(2);
  myRTC.setHour(17);
  myRTC.setMinute(8);
  myRTC.setSecond(0);

  pinMode(RE, OUTPUT);
  pinMode(DE, OUTPUT);
  pinMode(RG, OUTPUT);
  pinMode(DG, OUTPUT);
  pinMode(RF, OUTPUT);
  pinMode(DF, OUTPUT);
  delay(1000);

  // initialize the Analog pins for output.
  // Pin mode for Output: Humidifier
  pinMode(31, OUTPUT);
  // Pin mode for Output: Drip_Irrig
  pinMode(32, OUTPUT);
  // Pin mode for Output: Lime
  pinMode(34, OUTPUT);
  // Pin mode for Output: Sulfur
  pinMode(33, OUTPUT);
  // Pin mode for Output: Nitrogen
  pinMode(35, OUTPUT);
  // Pin mode for Output: Phosphorus
  pinMode(36, OUTPUT);
  // Pin mode for Output: Potassium
  pinMode(37, OUTPUT);

}

void loop() {
  SetRelay();
  Soil_EC();
  Soil_PH();
  Soil_NPK();
  Soil_Temperature();
  Soil_Moisture();
  Ambient_Temperature();
  Ambient_Humidity();
  Light_Intensity();
  Serial_Print();
  Alarm();
  delay(50000);
  SetOff();
}

void Serial_Print() {
  Serial.println(" "+ String("Node2") +
"+String(Value_Humidity)+" "+
String(Value_Ambient_Temp)+" "+
"+String(Value_Soil_Temp)+" "+
String(Value_Light_Int)+" "+
String(Value_Moisture)+" "+
"+String(Value_EC)+" "+
String(Value_PH)+" "+ String(Value_N)+" "+
"+ String(Value_Pho)+" "+
String(Value_Pot)+" "+
(myRTC.getMonth(century))+" "+
(myRTC.getDate())+" "+
(myRTC.getDoW())+" "+
(myRTC.getHour(h12Flag, pmFlag))+ " "+

```



```

        value[iii] = mod3.read();
    }
}

return value[4];
}

void Soil_Temperature(){
    sensors.requestTemperatures();
    Value_Soil_Temp =
    sensors.getTempCByIndex(0);
}

void Soil_Moisture(){
    Value_Moisture=analogRead(A15);

    Value_Moisture=map(Value_Moisture,0,982
    ,148,0);
    pres=Value_Moisture;
    if(Value_Moisture>100)
    Value_Moisture=100;
    else if(Value_Moisture<0)
    Value_Moisture=0;
    prev=Value_Moisture;
}

void Ambient_Humidity(){
    Value_Humidity = dht.readHumidity() + 3;
//+3
}

void Ambient_Temperature(){
    Value_Ambient_Temp =
    dht.readTemperature() - 1; // -1
}

void Light_Intensity(){
    Value_Light_Int =
    lightMeter.readLightLevel();
}

void Water(){
    Water_On = 1;
    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Phosphorus;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Drip_Irrig
    if ((g_fisOutput[1] > 0) &&
(g_fisOutput[1] <= 5))
    {
        digitalWrite(32, HIGH);
        delay(300000); //TRIAL
    }

    if ((g_fisOutput[1] > 5) &&
(g_fisOutput[1] <= 7))
    {
        digitalWrite(32, LOW);
        delay(20000);
        digitalWrite(32, HIGH);
        delay(300000); //TRIAL
    }

    if ((g_fisOutput[1] > 7) &&
(g_fisOutput[1] > 9))
    {
        digitalWrite(32, LOW);
        delay(40000);
    }
}

```

```

digitalWrite(32, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[1] > 9)
{
digitalWrite(32, LOW);
delay(60000);
digitalWrite(32, HIGH);
delay(300000); //TRIAL
}
}

void Humidifier(){
    Humidifier_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Pho;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Humidifier
}

if ((g_fisOutput[0] > 0) &&
(g_fisOutput[0] <= 5))
{
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

if ((g_fisOutput[0] > 5) &&
(g_fisOutput[0] <= 7))
{
digitalWrite(31, HIGH);
delay(60000);
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

if ((g_fisOutput[0] > 7) &&
(g_fisOutput[0] <= 9))
{
digitalWrite(31, HIGH);
delay(120000);
digitalWrite(31, LOW);
delay(300000);
}

if (g_fisOutput[0] > 9)
{
digitalWrite(31, HIGH);
delay(180000);
digitalWrite(31, LOW);
delay(300000); //TRIAL
}

void Correction()
{
    Correction_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
}

```

```

// Read Input: Soil_pH
g_fisInput[4] = Value_PH;
// Read Input: Nitrogen
g_fisInput[5] = Value_N;
// Read Input: Phosphorus
g_fisInput[6] = Value_Pho;
// Read Input: Potassium
g_fisInput[7] = Value_Pot;

g_fisOutput[0] = 0;
g_fisOutput[1] = 0;
g_fisOutput[2] = 0;
g_fisOutput[3] = 0;
g_fisOutput[4] = 0;
g_fisOutput[5] = 0;
g_fisOutput[6] = 0;

fis_evaluate();

// Set output vlaue: Lime
if ((g_fisOutput[2] > 0) &&
(g_fisOutput[2] <= 3.332))
{
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[2] > 3.332) &&
(g_fisOutput[2] <= 6.663))
{
    digitalWrite(34, LOW);
    delay(20000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[2] > 6.663)
{
    digitalWrite(34, LOW);
    delay(40000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Sulfur

```

```

if ((g_fisOutput[3] > 0) &&
(g_fisOutput[3] <= 3.332))
{
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[3] > 3.332) &&
(g_fisOutput[3] <= 6.663))
{
    digitalWrite(33, LOW);
    delay(20000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[3] > 6.663)
{
    digitalWrite(33, LOW);
    delay(40000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Nitrogen
if ((g_fisOutput[4] > 0) &&
(g_fisOutput[4] <= 3.332))
{
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[4] > 3.332) &&
(g_fisOutput[4] <= 6.663))
{
    digitalWrite(35, LOW);
    delay(20000);
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[4] > 6.663)
{
}

```

```

digitalWrite(35, LOW);
delay(40000);
digitalWrite(35, HIGH);
delay(300000); //TRIAL
}

// Set output vlaue: Phosphorus
if ((g_fisOutput[5] > 0) &&
(g_fisOutput[5] <= 3.332))
{
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[5] > 3.332) &&
(g_fisOutput[5] <= 6.663))
{
digitalWrite(36, LOW);
delay(20000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[5] > 6.663)
{
digitalWrite(36, LOW);
delay(40000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

// Set output vlaue: Potassium
if ((g_fisOutput[6] > 0) &&
(g_fisOutput[6] <= 3.332))
{
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[6] > 3.332) &&
(g_fisOutput[6] <= 6.663))
{
digitalWrite(37, LOW);

delay(20000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(31, HIGH);
}

if (g_fisOutput[6] > 3.332)
{
digitalWrite(31, LOW);
}

void Alarm(){
uint8_t checkHour =
myRTC.getHour(h12Flag, pmFlag);
uint8_t checkDoW = myRTC.getDoW();

if (checkHour == 9) {
Water();
//digitalWrite(31, HIGH);
} if (checkHour == 17 & (checkDoW != 2 | checkDoW != 5)) {
Water();
}
if (checkDoW == 2 & checkHour == 16) {
Correction();
delay(600000);
} if (checkDoW == 5 & checkHour == 16) {
Correction();
delay(600000);
// } if (checkHour == 16) {
// digitalWrite(31, LOW);
// } if (checkHour == 17) {
// Humidifier();
// }
}

void SetRelay(){
}

```

```

digitalWrite(31,LOW);

digitalWrite(32,HIGH);
digitalWrite(33,HIGH);
digitalWrite(34,HIGH);
digitalWrite(35,HIGH);
digitalWrite(36,HIGH);
digitalWrite(37,HIGH);
}

void SetOff(){
    Water_On = 0;
    Correction_On = 0;
    Humidifier_On = 0;
}

//*****
***** Support functions for Fuzzy Inference
System
***** //*****
* // Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d
= p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0
: ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0
: ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return
(FIS_TYPE) (x == a);
    if (a == b) return (FIS_TYPE) (t2*(b <=
x)*(x <= c));
    if (b == c) return (FIS_TYPE) (t1*(a <=
x)*(x <= b));
    t1 = min(t1, t2);
    return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a,
FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a,
FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE
*array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}

//*****
***** // Data for Fuzzy Inference System
***** //*****
* 
```

```

// Pointers to the implementations of
// member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

// Count of member function for each Input
int fis_gIMFCount[] = { 3, 3, 4, 3, 5, 3, 3, 3
};

// Count of member function for each Output
int fis_gOMFCount[] = { 4, 4, 3, 3, 3, 3, 3 };

// Coefficients for the Input Member
// Functions
FIS_TYPE fis_gMFI0Coeff1[] = { 0, 0, 8,
20 };
FIS_TYPE fis_gMFI0Coeff2[] = { 16, 26.5,
37 };
FIS_TYPE fis_gMFI0Coeff3[] = { 33, 45,
49.88, 49.88 };
FIS_TYPE* fis_gMFI0Coeff[] = {
fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { 0, 0, 30,
42 };
FIS_TYPE fis_gMFI1Coeff2[] = { 38, 50,
62 };
FIS_TYPE fis_gMFI1Coeff3[] = { 58, 70,
100, 100 };
FIS_TYPE* fis_gMFI1Coeff[] = {
fis_gMFI1Coeff1, fis_gMFI1Coeff2,
fis_gMFI1Coeff3 };
FIS_TYPE fis_gMFI2Coeff1[] = { 63, 72.5,
82 };
FIS_TYPE fis_gMFI2Coeff2[] = { 78, 88,
100, 100 };
FIS_TYPE fis_gMFI2Coeff3[] = { 0, 0,
47.5, 57 };
FIS_TYPE fis_gMFI2Coeff4[] = { 53, 60,
67 };
FIS_TYPE* fis_gMFI2Coeff[] = {
fis_gMFI2Coeff1, fis_gMFI2Coeff2,
fis_gMFI2Coeff3, fis_gMFI2Coeff4 };

FIS_TYPE fis_gMFI3Coeff1[] = { 0, 0, 13,
22 };
FIS_TYPE fis_gMFI3Coeff2[] = { 18, 27,
36 };
FIS_TYPE fis_gMFI3Coeff3[] = { 32, 41,
50, 50 };
FIS_TYPE* fis_gMFI3Coeff[] = {
fis_gMFI3Coeff1, fis_gMFI3Coeff2,
fis_gMFI3Coeff3 };
FIS_TYPE fis_gMFI4Coeff1[] = { 0, 0, 1,
3.2 };
FIS_TYPE fis_gMFI4Coeff2[] = { 2.2, 4,
5.9 };
FIS_TYPE fis_gMFI4Coeff3[] = { 10,
11.75, 14, 14 };
FIS_TYPE fis_gMFI4Coeff4[] = { 6.2,
8.525, 10.85 };
FIS_TYPE fis_gMFI4Coeff5[] = { 4.9, 6.05,
7.2 };
FIS_TYPE* fis_gMFI4Coeff[] = {
fis_gMFI4Coeff1, fis_gMFI4Coeff2,
fis_gMFI4Coeff3, fis_gMFI4Coeff4,
fis_gMFI4Coeff5 };
FIS_TYPE fis_gMFI5Coeff1[] = { 0, 0, 3, 8
};
FIS_TYPE fis_gMFI5Coeff2[] = { 6, 10.5,
15 };
FIS_TYPE fis_gMFI5Coeff3[] = { 13, 18,
30, 30 };
FIS_TYPE* fis_gMFI5Coeff[] = {
fis_gMFI5Coeff1, fis_gMFI5Coeff2,
fis_gMFI5Coeff3 };
FIS_TYPE fis_gMFI6Coeff1[] = { 0, 0, 3, 8
};
FIS_TYPE fis_gMFI6Coeff2[] = { 6, 10.5,
15 };
FIS_TYPE fis_gMFI6Coeff3[] = { 13, 18,
30, 30 };
FIS_TYPE* fis_gMFI6Coeff[] = {
fis_gMFI6Coeff1, fis_gMFI6Coeff2,
fis_gMFI6Coeff3 };
FIS_TYPE fis_gMFI7Coeff1[] = { 0, 0, 3, 8
};
FIS_TYPE fis_gMFI7Coeff2[] = { 6, 10.5,
15 };

```

```

FIS_TYPE fis_gMFI7Coeff3[] = { 13, 18,
30, 30 };
FIS_TYPE* fis_gMFI7Coeff[] = {
fis_gMFI7Coeff1, fis_gMFI7Coeff2,
fis_gMFI7Coeff3 };
FIS_TYPE** fis_gMFICoeff[] = {
fis_gMFI0Coeff, fis_gMFI1Coeff,
fis_gMFI2Coeff, fis_gMFI3Coeff,
fis_gMFI4Coeff, fis_gMFI5Coeff,
fis_gMFI6Coeff, fis_gMFI7Coeff };

// Coefficients for the Output Member
Functions
FIS_TYPE fis_gMFO0Coeff1[] = { 4, 6, 8
};
FIS_TYPE fis_gMFO0Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO0Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO0Coeff4[] = { 0, 0, 3,
5 };
FIS_TYPE* fis_gMFO0Coeff[] = {
fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3, fis_gMFO0Coeff4 };
FIS_TYPE fis_gMFO1Coeff1[] = { 0, 0, 3,
6 };
FIS_TYPE fis_gMFO1Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO1Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO1Coeff4[] = { 4, 6, 8
};
FIS_TYPE* fis_gMFO1Coeff[] = {
fis_gMFO1Coeff1, fis_gMFO1Coeff2,
fis_gMFO1Coeff3, fis_gMFO1Coeff4 };
FIS_TYPE fis_gMFO2Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO2Coeff2[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE fis_gMFO2Coeff3[] = { 2.833,
5, 7.166 };
FIS_TYPE* fis_gMFO2Coeff[] = {
fis_gMFO2Coeff1, fis_gMFO2Coeff2,
fis_gMFO2Coeff3 };
FIS_TYPE fis_gMFO3Coeff1[] = { 0, 0, 2,
3.83 };

```

```

FIS_TYPE fis_gMFO3Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO3Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO3Coeff[] = {
fis_gMFO3Coeff1, fis_gMFO3Coeff2,
fis_gMFO3Coeff3 };
FIS_TYPE fis_gMFO4Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO4Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO4Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO4Coeff[] = {
fis_gMFO4Coeff1, fis_gMFO4Coeff2,
fis_gMFO4Coeff3 };
FIS_TYPE fis_gMFO5Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO5Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO5Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO5Coeff[] = {
fis_gMFO5Coeff1, fis_gMFO5Coeff2,
fis_gMFO5Coeff3 };
FIS_TYPE fis_gMFO6Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO6Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO6Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO6Coeff[] = {
fis_gMFO6Coeff1, fis_gMFO6Coeff2,
fis_gMFO6Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = {
fis_gMFO0Coeff, fis_gMFO1Coeff,
fis_gMFO2Coeff, fis_gMFO3Coeff,
fis_gMFO4Coeff, fis_gMFO5Coeff,
fis_gMFO6Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 1, 0 };
int fis_gMFI1[] = { 0, 1, 0 };
int fis_gMFI2[] = { 1, 0, 0, 1 };
int fis_gMFI3[] = { 0, 1, 0 };
int fis_gMFI4[] = { 0, 1, 0, 1, 1 };

```

```

int fis_gMFI5[] = { 0, 1, 0 };
int fis_gMFI6[] = { 0, 1, 0 };
int fis_gMFI7[] = { 0, 1, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1,
fis_gMFI2, fis_gMFI3, fis_gMFI4,
fis_gMFI5, fis_gMFI6, fis_gMFI7};

// Output membership function set
int fis_gMFO0[] = { 1, 1, 0, 0 };
int fis_gMFO1[] = { 0, 1, 0, 1 };
int fis_gMFO2[] = { 0, 0, 1 };
int fis_gMFO3[] = { 0, 1, 0 };
int fis_gMFO4[] = { 0, 1, 0 };
int fis_gMFO5[] = { 0, 1, 0 };
int fis_gMFO6[] = { 0, 1, 0 };
int* fis_gMFO[] = { fis_gMFO0,
fis_gMFO1, fis_gMFO2, fis_gMFO3,
fis_gMFO4, fis_gMFO5, fis_gMFO6};

// Rule Weights
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Type
int fis_gRTYPE[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI1[] = { 2, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI2[] = { 1, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI3[] = { 2, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI4[] = { 2, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI5[] = { 3, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI6[] = { 3, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI7[] = { 1, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI8[] = { 3, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI9[] = { 0, 0, 2, 1, 0, 0, 0, 0, 0 };
int fis_gRI10[] = { 0, 0, 2, 2, 0, 0, 0, 0, 0 };
int fis_gRI11[] = { 0, 0, 2, 3, 0, 0, 0, 0, 0 };
int fis_gRI12[] = { 0, 0, 1, 1, 0, 0, 0, 0, 0 };
int fis_gRI13[] = { 0, 0, 1, 2, 0, 0, 0, 0, 0 };

```

```
int fis_gRI14[] = { 0, 0, 1, 3, 0, 0, 0, 0 };  
int fis_gRI15[] = { 0, 0, 4, 1, 0, 0, 0, 0 };  
int fis_gRI16[] = { 0, 0, 4, 2, 0, 0, 0, 0 };  
int fis_gRI17[] = { 0, 0, 4, 3, 0, 0, 0, 0 };  
int fis_gRI18[] = { 0, 0, 3, 1, 0, 0, 0, 0 };  
int fis_gRI19[] = { 0, 0, 3, 2, 0, 0, 0, 0 };  
int fis_gRI20[] = { 0, 0, 3, 3, 0, 0, 0, 0 };  
int fis_gRI21[] = { 0, 0, 0, 1, 1, 0, 0, 0 };  
int fis_gRI22[] = { 0, 0, 0, 1, 2, 0, 0, 0 };  
int fis_gRI23[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI24[] = { 0, 0, 0, 2, 1, 0, 0, 0 };  
int fis_gRI25[] = { 0, 0, 0, 2, 2, 0, 0, 0 };  
int fis_gRI26[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI27[] = { 0, 0, 0, 3, 1, 0, 0, 0 };  
int fis_gRI28[] = { 0, 0, 0, 3, 2, 0, 0, 0 };  
int fis_gRI29[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI30[] = { 0, 0, 0, 1, 3, 0, 0, 0 };  
int fis_gRI31[] = { 0, 0, 0, 1, 4, 0, 0, 0 };  
int fis_gRI32[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI33[] = { 0, 0, 0, 2, 3, 0, 0, 0 };  
int fis_gRI34[] = { 0, 0, 0, 2, 4, 0, 0, 0 };  
int fis_gRI35[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI36[] = { 0, 0, 0, 3, 3, 0, 0, 0 };  
int fis_gRI37[] = { 0, 0, 0, 3, 4, 0, 0, 0 };  
int fis_gRI38[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI39[] = { 0, 0, 0, 0, 0, 1, 0, 0 };  
int fis_gRI40[] = { 0, 0, 0, 0, 0, 2, 0, 0 };  
int fis_gRI41[] = { 0, 0, 0, 0, 0, 3, 0, 0 };  
int fis_gRI42[] = { 0, 0, 0, 0, 0, 0, 1, 0 };  
int fis_gRI43[] = { 0, 0, 0, 0, 0, 0, 2, 0 };  
int fis_gRI44[] = { 0, 0, 0, 0, 0, 0, 3, 0 };  
int fis_gRI45[] = { 0, 0, 0, 0, 0, 0, 0, 1 };  
int fis_gRI46[] = { 0, 0, 0, 0, 0, 0, 0, 2 };  
int fis_gRI47[] = { 0, 0, 0, 0, 0, 0, 0, 3 };  
int* fis_gRI[] = { fis_gRI0, fis_gRI1,  
    fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5,  
    fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9,  
    fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13,  
    fis_gRI14, fis_gRI15, fis_gRI16, fis_gRI17,  
    fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21,  
    fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25,  
    fis_gRI26, fis_gRI27, fis_gRI28, fis_gRI29,  
    fis_gRI30, fis_gRI31, fis_gRI32, fis_gRI33,  
    fis_gRI34, fis_gRI35, fis_gRI36, fis_gRI37,  
    fis_gRI38, fis_gRI39, fis_gRI40, fis_gRI41,
```

```
fis_gRI42, fis_gRI43, fis_gRI44, fis_gRI45,
fis_gRI46, fis_gRI47 };
```

// Rule Outputs

```
int fis_gRO0[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO1[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO2[] = { 4, 0, 0, 0, 0, 0, 0 };
int fis_gRO3[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO4[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO5[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO6[] = { 3, 0, 0, 0, 0, 0, 0 };
int fis_gRO7[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO8[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO9[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO10[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO11[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO12[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO13[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO14[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO15[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO16[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO17[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO18[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO19[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO20[] = { 0, 3, 0, 0, 0, 0, 0 };
int fis_gRO21[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO22[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO23[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO24[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO25[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO26[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO27[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO28[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO29[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO30[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO31[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO32[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO33[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO34[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO35[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO36[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO37[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO38[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO39[] = { 0, 0, 0, 0, 3, 0, 0 };
int fis_gRO40[] = { 0, 0, 0, 0, 2, 0, 0 };
int fis_gRO41[] = { 0, 0, 0, 0, 1, 0, 0 };
```

```
int fis_gRO42[] = { 0, 0, 0, 0, 0, 3, 0 };
int fis_gRO43[] = { 0, 0, 0, 0, 0, 2, 0 };
int fis_gRO44[] = { 0, 0, 0, 0, 0, 1, 0 };
int fis_gRO45[] = { 0, 0, 0, 0, 0, 0, 3 };
int fis_gRO46[] = { 0, 0, 0, 0, 0, 0, 2 };
int fis_gRO47[] = { 0, 0, 0, 0, 0, 0, 1 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1,
fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9,
fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15,
fis_gRO16, fis_gRO17, fis_gRO18,
fis_gRO19, fis_gRO20, fis_gRO21,
fis_gRO22, fis_gRO23, fis_gRO24,
fis_gRO25, fis_gRO26, fis_gRO27,
fis_gRO28, fis_gRO29, fis_gRO30,
fis_gRO31, fis_gRO32, fis_gRO33,
fis_gRO34, fis_gRO35, fis_gRO36,
fis_gRO37, fis_gRO38, fis_gRO39,
fis_gRO40, fis_gRO41, fis_gRO42,
fis_gRO43, fis_gRO44, fis_gRO45,
fis_gRO46, fis_gRO47 };
```

// Input range Min

```
FIS_TYPE fis_gIMin[] = { 0, 0, 0, 0, 0, 0, 0,
0 };
```

// Input range Max

```
FIS_TYPE fis_gIMax[] = { 50, 100, 100,
50, 14, 30, 30, 30 };
```

// Output range Min

```
FIS_TYPE fis_gOMin[] = { 0, 0, 0, 0, 0, 0,
0 };
```

// Output range Max

```
FIS_TYPE fis_gOMax[] = { 15, 15, 10, 10,
10, 10, 10 };
```

```
*****  
*****  
*
```

// Data dependent support functions for
Fuzzy Inference System

```

//*****
***** *
FIS_TYPE fis_MF_out(FIS_TYPE**
fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut =
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 -
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut,
fuzzyRuleSet[1][r]);
    }
    return
fis_array_operation(fuzzyRuleSet[0],
fis_gcR, fis_max);
}

FIS_TYPE
fis_defuzz_centroid(FIS_TYPE**
fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] -
fis_gOMin[o]) / (FIS_RESOLUTION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve
    formed by the MF outputs
    for (i = 0; i < FIS_RESOLUTION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step *
fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] +
fis_gOMin[o]) / 2) : (momentum / area));
}

//*****
***** *
// Fuzzy Inference System
//*****
***** *
***** *
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput2[] = { 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput3[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput4[] = { 0, 0, 0, 0, 0
};;
    FIS_TYPE fuzzyInput5[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput6[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput7[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] =
fuzzyInput0, fuzzyInput1, fuzzyInput2,
fuzzyInput3, fuzzyInput4, fuzzyInput5,
fuzzyInput6, fuzzyInput7, };

    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput2[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput3[] = { 0, 0, 0 };
}

```

```

FIS_TYPE fuzzyOutput4[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput5[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput6[] = { 0, 0, 0 };
FIS_TYPE* fuzzyOutput[fis_gcO] = {
fuzzyOutput0, fuzzyOutput1, fuzzyOutput2,
fuzzyOutput3, fuzzyOutput4, fuzzyOutput5,
fuzzyOutput6, };
FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
FIS_TYPE* fuzzyRuleSet[] = {
fuzzyRules, fuzzyFires };
FIS_TYPE sW = 0;

// Transforming input to fuzzy Input
int i, j, r, o;
for (i = 0; i < fis_gcI; ++i)
{
    for (j = 0; j < fis_gIMFCount[i]; ++j)
    {
        fuzzyInput[i][j] =
(fis_gMF[fis_gMFI[i][j]])(g_fisInput[i],
fis_gMFICoeff[i][j]);
    }
}

int index = 0;
for (r = 0; r < fis_gcR; ++r)
{
    if (fis_gRTyp[r] == 1)
    {
        fuzzyFires[r] = FIS_MAX;
        for (i = 0; i < fis_gcI; ++i)
        {
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], fuzzyInput[i][index -
1]);
            else if (index < 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
            else
                fuzzyFires[r] =
fis_min(fuzzyFires[r], 1);
    }
}
else
{
    fuzzyFires[r] = FIS_MIN;
    for (i = 0; i < fis_gcI; ++i)
    {
        index = fis_gRI[r][i];
        if (index > 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
        else if (index < 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
        else
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 0);
    }
}
fuzzyFires[r] = fis_gRWeight[r] *
fuzzyFires[r];
sW += fuzzyFires[r];
}

if (sW == 0)
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] = ((fis_gOMax[o] +
fis_gOMin[o]) / 2);
    }
}
else
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] =
fis_defuzz_centroid(fuzzyRuleSet, o);
    }
}
}
}

```

Arduino Codes of Node 1 Pepper

```
#include <DS3231.h>
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <BH1750.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "fis_header.h"

#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define ONE_WIRE_PIN 5
#define RE 6
#define DE 7
#define RF 52
#define DF 53
#define RG 8
#define DG 9
#define FIS_TYPE float
#define FIS_RESOLUTION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);
typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_ARR_OP);

SoftwareSerial mod1(12,13);
SoftwareSerial mod2(10,11);
SoftwareSerial mod3(50,51);
const byte ec[] = {0x01, 0x03, 0x00, 0x15, 0x00, 0x01, 0x95, 0xCE};
const byte ph[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x84, 0x0A};
const byte nitro[] = {0x01, 0x03, 0x00, 0x1e, 0x00, 0x01, 0xe4, 0x0c};
const byte phos[] = {0x01, 0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc};

const byte pota[] = {0x01, 0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xc0};
byte value[11];

// Number of inputs to the fuzzy inference system
const int fis_gcI = 8;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 7;
// Number of rules to the fuzzy inference system
const int fis_gcR = 48;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);
BH1750 lightMeter;
DS3231 myRTC;

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

int prev=0;
int pres=0;

float Value_EC = 0;
float Value_PH = 0;
float Value_N = 0;
float Value_Pho = 0;
float Value_Pot = 0;
float Value_Ambient_Temp = 0;
float Value_Humidity = 0;
float Value_Light_Int = 0;
float Value_Soil_Temp = 0;
float Value_Moisture = 0;
float Water_On = 0;
float Correction_On = 0;
float Humidifier_On = 0;
```

```

byte year;
byte month;
byte date;
byte dow;
byte hour;
byte minute;
byte second;

bool century = false;
bool h12Flag;
bool pmFlag;

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

  mod1.begin(9600);
  mod2.begin(4800);
  mod3.begin(9600);

  Wire.begin();
  lightMeter.begin();
  dht.begin();
  sensors.begin();

  myRTC.setClockMode(false); // set to 24h

  myRTC.setYear(23);
  myRTC.setMonth(6);
  myRTC.setDate(28);
  myRTC.setDoW(4);
  myRTC.setHour(13);
  myRTC.setMinute(55);
  myRTC.setSecond(0);

  pinMode(RE, OUTPUT);
  pinMode(DE, OUTPUT);
  pinMode(RG, OUTPUT);
  pinMode(DG, OUTPUT);
  pinMode(RF, OUTPUT);
  pinMode(DF, OUTPUT);
  delay(1000);

  // initialize the Analog pins for output.
  // Pin mode for Output: Humidifier
  pinMode(31, OUTPUT);
  // Pin mode for Output: Drip_Irrig
  pinMode(32, OUTPUT);
  // Pin mode for Output: Lime
  pinMode(34, OUTPUT);
  // Pin mode for Output: Sulfur
  pinMode(33, OUTPUT);
  // Pin mode for Output: Nitrogen
  pinMode(35, OUTPUT);
  // Pin mode for Output: Phosphorus
  pinMode(36, OUTPUT);
  // Pin mode for Output: Potassium
  pinMode(37, OUTPUT);

}

void loop() {
  SetRelay();
  Soil_EC();
  Soil_PH();
  Soil_NPK();
  Soil_Temperature();
  Soil_Moisture();
  Ambient_Temperature();
  Ambient_Humidity();
  Light_Intensity();
  Serial_Print();
  Alarm();
  delay(50000);
  SetOff();
}

void Serial_Print(){
  Serial.println(" "+ String("Node1") +
"+String(Value_Humidity)+" "+
String(Value_Ambient_Temp)+" "+
"+String(Value_Soil_Temp)+" "+
String(Value_Light_Int)+" "+
String(Value_Moisture)+" "+
"+String(Value_EC)+" "+
String(Value_PH)+" "+ String(Value_N)+" "+
"+ String(Value_Pho)+" "+
String(Value_Pot)+" "+
(myRTC.getMonth(century))+" "+
(myRTC.getDate())+" "+
(myRTC.getDoW())+" "+
(myRTC.getHour(h12Flag, pmFlag))+ " "+

```



```

        value[iii] = mod3.read();
    }
}

return value[4];
}

void Soil_Temperature(){
    sensors.requestTemperatures();
    Value_Soil_Temp =
    sensors.getTempCByIndex(0);
}

void Soil_Moisture(){
    Value_Moisture=analogRead(A15);

    Value_Moisture=map(Value_Moisture,0,982
    ,148,0);
    pres=Value_Moisture;
    if(Value_Moisture>100)
    Value_Moisture=100;
    else if(Value_Moisture<0)
    Value_Moisture=0;
    prev=Value_Moisture;
}

void Ambient_Humidity(){
    Value_Humidity = dht.readHumidity() + 3;
//+3
}

void Ambient_Temperature(){
    Value_Ambient_Temp =
    dht.readTemperature() - 1; // -1
}

void Light_Intensity(){
    //Value_Light_Int =
    lightMeter.readLightLevel();
    Value_Light_Int = 0;
}

void Water(){
    Water_On = 1;
    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
}

// Read Input: Humidity
g_fisInput[1] = Value_Humidity;
// Read Input: Soil_Moisture
g_fisInput[2] = Value_Moisture;
// Read Input: Soil_Temp
g_fisInput[3] = Value_Soil_Temp;
// Read Input: Soil_pH
g_fisInput[4] = Value_PH;
// Read Input: Nitrogen
g_fisInput[5] = Value_N;
// Read Input: Phosphorus
g_fisInput[6] = Value_Phosphorus;
// Read Input: Potassium
g_fisInput[7] = Value_Pot;

g_fisOutput[0] = 0;
g_fisOutput[1] = 0;
g_fisOutput[2] = 0;
g_fisOutput[3] = 0;
g_fisOutput[4] = 0;
g_fisOutput[5] = 0;
g_fisOutput[6] = 0;

fis_evaluate();

// Set output vlaue: Drip_Irrig
if ((g_fisOutput[1] > 0) &&
(g_fisOutput[1] <= 5))
{
    digitalWrite(32, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[1] > 5) &&
(g_fisOutput[1] <= 7))
{
    digitalWrite(32, LOW);
    delay(20000);
    digitalWrite(32, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[1] > 7) &&
(g_fisOutput[1] > 9))
{
    digitalWrite(32, LOW);
}

```

```

delay(40000);
digitalWrite(32, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[1] > 9)
{
digitalWrite(32, LOW);
delay(60000);
digitalWrite(32, HIGH);
delay(300000); //TRIAL
}
}

void Humidifier(){
    Humidifier_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Pho;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Humidifier
        if ((g_fisOutput[0] > 0) &&
(g_fisOutput[0] <= 5))
    {
        digitalWrite(31, LOW);
        delay(300000); //TRIAL
    }

        if ((g_fisOutput[0] > 5) &&
(g_fisOutput[0] <= 7))
    {
        digitalWrite(31, HIGH);
        delay(60000);
        digitalWrite(31, LOW);
        delay(300000); //TRIAL
    }

        if ((g_fisOutput[0] > 7) &&
(g_fisOutput[0] <= 9))
    {
        digitalWrite(31, HIGH);
        delay(120000);
        digitalWrite(31, LOW);
        delay(300000);
    }

        if (g_fisOutput[0] > 9)
    {
        digitalWrite(31, HIGH);
        delay(180000);
        digitalWrite(31, LOW);
        delay(300000); //TRIAL
    }
}

void Correction()
{
    Correction_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
}

```

```

// Read Input: Soil_pH
g_fisInput[4] = Value_PH;
// Read Input: Nitrogen
g_fisInput[5] = Value_N;
// Read Input: Phosphorus
g_fisInput[6] = Value_Pho;
// Read Input: Potassium
g_fisInput[7] = Value_Pot;

g_fisOutput[0] = 0;
g_fisOutput[1] = 0;
g_fisOutput[2] = 0;
g_fisOutput[3] = 0;
g_fisOutput[4] = 0;
g_fisOutput[5] = 0;
g_fisOutput[6] = 0;

fis_evaluate();

// Set output vlaue: Lime
if ((g_fisOutput[2] > 0) &&
(g_fisOutput[2] <= 3.332))
{
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[2] > 3.332) &&
(g_fisOutput[2] <= 6.663))
{
    digitalWrite(34, LOW);
    delay(20000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[2] > 6.663)
{
    digitalWrite(34, LOW);
    delay(40000);
    digitalWrite(34, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Sulfur

```

```

if ((g_fisOutput[3] > 0) &&
(g_fisOutput[3] <= 3.332))
{
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[3] > 3.332) &&
(g_fisOutput[3] <= 6.663))
{
    digitalWrite(33, LOW);
    delay(20000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[3] > 6.663)
{
    digitalWrite(33, LOW);
    delay(40000);
    digitalWrite(33, HIGH);
    delay(300000); //TRIAL
}

// Set output vlaue: Nitrogen
if ((g_fisOutput[4] > 0) &&
(g_fisOutput[4] <= 3.332))
{
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if ((g_fisOutput[4] > 3.332) &&
(g_fisOutput[4] <= 6.663))
{
    digitalWrite(35, LOW);
    delay(20000);
    digitalWrite(35, HIGH);
    delay(300000); //TRIAL
}

if (g_fisOutput[4] > 6.663)
{
}

```

```

digitalWrite(35, LOW);
delay(40000);
digitalWrite(35, HIGH);
delay(300000); //TRIAL
}

// Set output value: Phosphorus
if ((g_fisOutput[5] > 0) &&
(g_fisOutput[5] <= 3.332))
{
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[5] > 3.332) &&
(g_fisOutput[5] <= 6.663))
{
digitalWrite(36, LOW);
delay(20000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[5] > 6.663)
{
digitalWrite(36, LOW);
delay(40000);
digitalWrite(36, HIGH);
delay(300000); //TRIAL
}

// Set output value: Potassium
if ((g_fisOutput[6] > 0) &&
(g_fisOutput[6] <= 3.332))
{
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if ((g_fisOutput[6] > 3.332) &&
(g_fisOutput[6] <= 6.663))
{
digitalWrite(37, LOW);

delay(20000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(300000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(31, HIGH);
}
if (checkHour == 16 & (checkDoW != 3 |
checkDoW != 6)) {
Water();
}
if (checkDoW == 3 & checkHour == 16) {
Correction();
delay(600000);
} if (checkDoW == 6 & checkHour == 16)
{
Correction();
delay(600000);
} //if (checkHour == 16) {
// digitalWrite(31, LOW);
// } if (checkHour == 17) {
// Humidifier();
// }

void SetRelay(){
digitalWrite(31,LOW);
}

```

```

digitalWrite(32,HIGH);
digitalWrite(33,HIGH);
digitalWrite(34,HIGH);
digitalWrite(35,HIGH);
digitalWrite(36,HIGH);
digitalWrite(37,HIGH);
}

void SetOff(){
    Water_On = 0;
    Correction_On = 0;
    Humidifier_On = 0;
}

//*****
// Support functions for Fuzzy Inference
System
//*****
/*
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d
= p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0
: ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0
: ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis.trimf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return
(FIS_TYPE) (x == a);
}

if (a == b) return (FIS_TYPE) (t2*(b <=
x)*(x <= c));
if (b == c) return (FIS_TYPE) (t1*(a <=
x)*(x <= b));
t1 = min(t1, t2);
return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a,
FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a,
FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE
*array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}

//*****
// Data for Fuzzy Inference System
//*****
*/

```

```

// Pointers to the implementations of
// member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

// Count of member function for each Input
int fis_gIMFCount[] = { 3, 3, 4, 3, 5, 3, 3, 3
};

// Count of member function for each Output
int fis_gOMFCount[] = { 4, 4, 3, 3, 3, 3, 3 };

// Coefficients for the Input Member
// Functions
FIS_TYPE fis_gMFI0Coeff1[] = { 0, 0, 17,
23 };
FIS_TYPE fis_gMFI0Coeff2[] = { 19, 24,
29 };
FIS_TYPE fis_gMFI0Coeff3[] = {
24.9614494988435, 30.9614494988435,
49.9614494988435, 49.9614494988435 };
FIS_TYPE* fis_gMFI0Coeff[] = {
fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { 0, 0, 40,
52 };
FIS_TYPE fis_gMFI1Coeff2[] = { 48, 60,
72 };
FIS_TYPE fis_gMFI1Coeff3[] = { 68, 80,
100, 100 };
FIS_TYPE* fis_gMFI1Coeff[] = {
fis_gMFI1Coeff1, fis_gMFI1Coeff2,
fis_gMFI1Coeff3 };
FIS_TYPE fis_gMFI2Coeff1[] = { 58, 70,
82 };
FIS_TYPE fis_gMFI2Coeff2[] = { 78, 90,
100, 100 };
FIS_TYPE fis_gMFI2Coeff3[] = { 0, 0, 18,
32 };
FIS_TYPE fis_gMFI2Coeff4[] = { 28, 45,
62 };
FIS_TYPE* fis_gMFI2Coeff[] = {
fis_gMFI2Coeff1, fis_gMFI2Coeff2,
fis_gMFI2Coeff3, fis_gMFI2Coeff4 };

```

```

FIS_TYPE fis_gMFI3Coeff1[] = { 0, 0, 13,
20 };
FIS_TYPE fis_gMFI3Coeff2[] = { 16, 23,
30 };
FIS_TYPE fis_gMFI3Coeff3[] = { 26, 33,
50, 50 };
FIS_TYPE* fis_gMFI3Coeff[] = {
fis_gMFI3Coeff1, fis_gMFI3Coeff2,
fis_gMFI3Coeff3 };
FIS_TYPE fis_gMFI4Coeff1[] = { 0, 0, 1, 3
};
FIS_TYPE fis_gMFI4Coeff2[] = { 2, 3.75,
5.5 };
FIS_TYPE fis_gMFI4Coeff3[] = { 10, 11.5,
14, 14 };
FIS_TYPE fis_gMFI4Coeff4[] = { 6.5, 8.75,
11 };
FIS_TYPE fis_gMFI4Coeff5[] = {
4.48920585967618, 5.98920585967618,
7.48920585967618 };
FIS_TYPE* fis_gMFI4Coeff[] = {
fis_gMFI4Coeff1, fis_gMFI4Coeff2,
fis_gMFI4Coeff3, fis_gMFI4Coeff4,
fis_gMFI4Coeff5 };
FIS_TYPE fis_gMFI5Coeff1[] = { 0, 0, 1,
3.5 };
FIS_TYPE fis_gMFI5Coeff2[] = { 1.5, 3.75,
6 };
FIS_TYPE fis_gMFI5Coeff3[] = { 4, 6.5,
30, 30 };
FIS_TYPE* fis_gMFI5Coeff[] = {
fis_gMFI5Coeff1, fis_gMFI5Coeff2,
fis_gMFI5Coeff3 };
FIS_TYPE fis_gMFI6Coeff1[] = { 0, 0, 2, 6
};
FIS_TYPE fis_gMFI6Coeff2[] = { 4, 7.5, 11
};
FIS_TYPE fis_gMFI6Coeff3[] = { 9, 12.5,
30, 30 };
FIS_TYPE* fis_gMFI6Coeff[] = {
fis_gMFI6Coeff1, fis_gMFI6Coeff2,
fis_gMFI6Coeff3 };
FIS_TYPE fis_gMFI7Coeff1[] = { 0, 0, 2, 6
};
FIS_TYPE fis_gMFI7Coeff2[] = { 4, 7.5, 11
};

```

```

FIS_TYPE fis_gMFI7Coeff3[] = { 9, 12.5,
30, 30 };
FIS_TYPE* fis_gMFI7Coeff[] = {
fis_gMFI7Coeff1, fis_gMFI7Coeff2,
fis_gMFI7Coeff3 };
FIS_TYPE** fis_gMFICoeff[] = {
fis_gMFI0Coeff, fis_gMFI1Coeff,
fis_gMFI2Coeff, fis_gMFI3Coeff,
fis_gMFI4Coeff, fis_gMFI5Coeff,
fis_gMFI6Coeff, fis_gMFI7Coeff };

// Coefficients for the Output Member
Functions
FIS_TYPE fis_gMFO0Coeff1[] = { 4, 6, 8
};
FIS_TYPE fis_gMFO0Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO0Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO0Coeff4[] = { 0, 0, 3,
5 };
FIS_TYPE* fis_gMFO0Coeff[] = {
fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3, fis_gMFO0Coeff4 };
FIS_TYPE fis_gMFO1Coeff1[] = { 0, 0, 3,
6 };
FIS_TYPE fis_gMFO1Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO1Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO1Coeff4[] = { 4, 6, 8
};
FIS_TYPE* fis_gMFO1Coeff[] = {
fis_gMFO1Coeff1, fis_gMFO1Coeff2,
fis_gMFO1Coeff3, fis_gMFO1Coeff4 };
FIS_TYPE fis_gMFO2Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO2Coeff2[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE fis_gMFO2Coeff3[] = { 2.833,
5, 7.166 };
FIS_TYPE* fis_gMFO2Coeff[] = {
fis_gMFO2Coeff1, fis_gMFO2Coeff2,
fis_gMFO2Coeff3 };
FIS_TYPE fis_gMFO3Coeff1[] = { 0, 0, 2,
3.83 };

FIS_TYPE fis_gMFO3Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO3Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO3Coeff[] = {
fis_gMFO3Coeff1, fis_gMFO3Coeff2,
fis_gMFO3Coeff3 };
FIS_TYPE fis_gMFO4Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO4Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO4Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO4Coeff[] = {
fis_gMFO4Coeff1, fis_gMFO4Coeff2,
fis_gMFO4Coeff3 };
FIS_TYPE fis_gMFO5Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO5Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO5Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO5Coeff[] = {
fis_gMFO5Coeff1, fis_gMFO5Coeff2,
fis_gMFO5Coeff3 };
FIS_TYPE fis_gMFO6Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO6Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO6Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO6Coeff[] = {
fis_gMFO6Coeff1, fis_gMFO6Coeff2,
fis_gMFO6Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = {
fis_gMFO0Coeff, fis_gMFO1Coeff,
fis_gMFO2Coeff, fis_gMFO3Coeff,
fis_gMFO4Coeff, fis_gMFO5Coeff,
fis_gMFO6Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 1, 0 };
int fis_gMFI1[] = { 0, 1, 0 };
int fis_gMFI2[] = { 1, 0, 0, 1 };
int fis_gMFI3[] = { 0, 1, 0 };
int fis_gMFI4[] = { 0, 1, 0, 1, 1 };

```

```

int fis_gMFI5[] = { 0, 1, 0 };
int fis_gMFI6[] = { 0, 1, 0 };
int fis_gMFI7[] = { 0, 1, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1,
fis_gMFI2, fis_gMFI3, fis_gMFI4,
fis_gMFI5, fis_gMFI6, fis_gMFI7};

// Output membership function set
int fis_gMFO0[] = { 1, 1, 0, 0 };
int fis_gMFO1[] = { 0, 1, 0, 1 };
int fis_gMFO2[] = { 0, 0, 1 };
int fis_gMFO3[] = { 0, 1, 0 };
int fis_gMFO4[] = { 0, 1, 0 };
int fis_gMFO5[] = { 0, 1, 0 };
int fis_gMFO6[] = { 0, 1, 0 };
int* fis_gMFO[] = { fis_gMFO0,
fis_gMFO1, fis_gMFO2, fis_gMFO3,
fis_gMFO4, fis_gMFO5, fis_gMFO6};

// Rule Weights
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Type
int fis_gRTYPE[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI1[] = { 2, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI2[] = { 1, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI3[] = { 2, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI4[] = { 2, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI5[] = { 3, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI6[] = { 3, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI7[] = { 1, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI8[] = { 3, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI9[] = { 0, 0, 2, 1, 0, 0, 0, 0, 0 };
int fis_gRI10[] = { 0, 0, 2, 2, 0, 0, 0, 0, 0 };
int fis_gRI11[] = { 0, 0, 2, 3, 0, 0, 0, 0, 0 };
int fis_gRI12[] = { 0, 0, 1, 1, 0, 0, 0, 0, 0 };
int fis_gRI13[] = { 0, 0, 1, 2, 0, 0, 0, 0, 0 };

```

```
int fis_gRI14[] = { 0, 0, 1, 3, 0, 0, 0, 0 };  
int fis_gRI15[] = { 0, 0, 4, 1, 0, 0, 0, 0 };  
int fis_gRI16[] = { 0, 0, 4, 2, 0, 0, 0, 0 };  
int fis_gRI17[] = { 0, 0, 4, 3, 0, 0, 0, 0 };  
int fis_gRI18[] = { 0, 0, 3, 1, 0, 0, 0, 0 };  
int fis_gRI19[] = { 0, 0, 3, 2, 0, 0, 0, 0 };  
int fis_gRI20[] = { 0, 0, 3, 3, 0, 0, 0, 0 };  
int fis_gRI21[] = { 0, 0, 0, 1, 1, 0, 0, 0 };  
int fis_gRI22[] = { 0, 0, 0, 1, 2, 0, 0, 0 };  
int fis_gRI23[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI24[] = { 0, 0, 0, 2, 1, 0, 0, 0 };  
int fis_gRI25[] = { 0, 0, 0, 2, 2, 0, 0, 0 };  
int fis_gRI26[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI27[] = { 0, 0, 0, 3, 1, 0, 0, 0 };  
int fis_gRI28[] = { 0, 0, 0, 3, 2, 0, 0, 0 };  
int fis_gRI29[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI30[] = { 0, 0, 0, 1, 3, 0, 0, 0 };  
int fis_gRI31[] = { 0, 0, 0, 1, 4, 0, 0, 0 };  
int fis_gRI32[] = { 0, 0, 0, 1, 5, 0, 0, 0 };  
int fis_gRI33[] = { 0, 0, 0, 2, 3, 0, 0, 0 };  
int fis_gRI34[] = { 0, 0, 0, 2, 4, 0, 0, 0 };  
int fis_gRI35[] = { 0, 0, 0, 2, 5, 0, 0, 0 };  
int fis_gRI36[] = { 0, 0, 0, 3, 3, 0, 0, 0 };  
int fis_gRI37[] = { 0, 0, 0, 3, 4, 0, 0, 0 };  
int fis_gRI38[] = { 0, 0, 0, 3, 5, 0, 0, 0 };  
int fis_gRI39[] = { 0, 0, 0, 0, 0, 1, 0, 0 };  
int fis_gRI40[] = { 0, 0, 0, 0, 0, 2, 0, 0 };  
int fis_gRI41[] = { 0, 0, 0, 0, 0, 3, 0, 0 };  
int fis_gRI42[] = { 0, 0, 0, 0, 0, 0, 1, 0 };  
int fis_gRI43[] = { 0, 0, 0, 0, 0, 0, 2, 0 };  
int fis_gRI44[] = { 0, 0, 0, 0, 0, 0, 3, 0 };  
int fis_gRI45[] = { 0, 0, 0, 0, 0, 0, 0, 1 };  
int fis_gRI46[] = { 0, 0, 0, 0, 0, 0, 0, 2 };  
int fis_gRI47[] = { 0, 0, 0, 0, 0, 0, 0, 3 };  
int* fis_gRI[] = { fis_gRI0, fis_gRI1,  
    fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5,  
    fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9,  
    fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13,  
    fis_gRI14, fis_gRI15, fis_gRI16, fis_gRI17,  
    fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21,  
    fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25,  
    fis_gRI26, fis_gRI27, fis_gRI28, fis_gRI29,  
    fis_gRI30, fis_gRI31, fis_gRI32, fis_gRI33,  
    fis_gRI34, fis_gRI35, fis_gRI36, fis_gRI37,  
    fis_gRI38, fis_gRI39, fis_gRI40, fis_gRI41,
```

```
fis_gRI42, fis_gRI43, fis_gRI44, fis_gRI45,
fis_gRI46, fis_gRI47 };
```

// Rule Outputs

```
int fis_gRO0[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO1[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO2[] = { 4, 0, 0, 0, 0, 0, 0 };
int fis_gRO3[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO4[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO5[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO6[] = { 3, 0, 0, 0, 0, 0, 0 };
int fis_gRO7[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO8[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO9[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO10[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO11[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO12[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO13[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO14[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO15[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO16[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO17[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO18[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO19[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO20[] = { 0, 3, 0, 0, 0, 0, 0 };
int fis_gRO21[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO22[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO23[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO24[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO25[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO26[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO27[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO28[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO29[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO30[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO31[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO32[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO33[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO34[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO35[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO36[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO37[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO38[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO39[] = { 0, 0, 0, 0, 3, 0, 0 };
int fis_gRO40[] = { 0, 0, 0, 0, 2, 0, 0 };
int fis_gRO41[] = { 0, 0, 0, 0, 1, 0, 0 };
```

```
int fis_gRO42[] = { 0, 0, 0, 0, 0, 3, 0 };
int fis_gRO43[] = { 0, 0, 0, 0, 0, 2, 0 };
int fis_gRO44[] = { 0, 0, 0, 0, 0, 1, 0 };
int fis_gRO45[] = { 0, 0, 0, 0, 0, 0, 3 };
int fis_gRO46[] = { 0, 0, 0, 0, 0, 0, 2 };
int fis_gRO47[] = { 0, 0, 0, 0, 0, 0, 1 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1,
fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9,
fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15,
fis_gRO16, fis_gRO17, fis_gRO18,
fis_gRO19, fis_gRO20, fis_gRO21,
fis_gRO22, fis_gRO23, fis_gRO24,
fis_gRO25, fis_gRO26, fis_gRO27,
fis_gRO28, fis_gRO29, fis_gRO30,
fis_gRO31, fis_gRO32, fis_gRO33,
fis_gRO34, fis_gRO35, fis_gRO36,
fis_gRO37, fis_gRO38, fis_gRO39,
fis_gRO40, fis_gRO41, fis_gRO42,
fis_gRO43, fis_gRO44, fis_gRO45,
fis_gRO46, fis_gRO47 };
```

// Input range Min

```
FIS_TYPE fis_gIMin[] = { 0, 0, 0, 0, 0, 0, 0,
0 };
```

// Input range Max

```
FIS_TYPE fis_gIMax[] = { 50, 100, 100,
50, 14, 30, 30, 30 };
```

// Output range Min

```
FIS_TYPE fis_gOMin[] = { 0, 0, 0, 0, 0, 0,
0 };
```

// Output range Max

```
FIS_TYPE fis_gOMax[] = { 15, 15, 10, 10,
10, 10, 10 };
```

```
*****  
*****  
*
```

// Data dependent support functions for Fuzzy Inference System

```

//*****
***** *
FIS_TYPE fis_MF_out(FIS_TYPE**
fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut =
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 -
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut,
fuzzyRuleSet[1][r]);
    }
    return
fis_array_operation(fuzzyRuleSet[0],
fis_gcR, fis_max);
}

FIS_TYPE
fis_defuzz_centroid(FIS_TYPE**
fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] -
fis_gOMin[o]) / (FIS_RESOLUTION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve
    formed by the MF outputs
    for (i = 0; i < FIS_RESOLUTION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step *
fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] +
fis_gOMin[o]) / 2) : (momentum / area));
}

//*****
***** *
// Fuzzy Inference System
//*****
***** *
***** *
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput2[] = { 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput3[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput4[] = { 0, 0, 0, 0, 0
};;
    FIS_TYPE fuzzyInput5[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput6[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput7[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] = {
fuzzyInput0, fuzzyInput1, fuzzyInput2,
fuzzyInput3, fuzzyInput4, fuzzyInput5,
fuzzyInput6, fuzzyInput7, };

    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0, 0
};;
    FIS_TYPE fuzzyOutput2[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput3[] = { 0, 0, 0 };
}

```

```

FIS_TYPE fuzzyOutput4[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput5[] = { 0, 0, 0 };
FIS_TYPE fuzzyOutput6[] = { 0, 0, 0 };
FIS_TYPE* fuzzyOutput[fis_gcO] = {
fuzzyOutput0, fuzzyOutput1, fuzzyOutput2,
fuzzyOutput3, fuzzyOutput4, fuzzyOutput5,
fuzzyOutput6, };
FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
FIS_TYPE* fuzzyRuleSet[] = {
fuzzyRules, fuzzyFires };
FIS_TYPE sW = 0;

// Transforming input to fuzzy Input
int i, j, r, o;
for (i = 0; i < fis_gcI; ++i)
{
    for (j = 0; j < fis_gIMFCount[i]; ++j)
    {
        fuzzyInput[i][j] =
(fis_gMF[fis_gMFI[i][j]])(g_fisInput[i],
fis_gMFICoeff[i][j]);
    }
}

int index = 0;
for (r = 0; r < fis_gcR; ++r)
{
    if (fis_gRTyp[r] == 1)
    {
        fuzzyFires[r] = FIS_MAX;
        for (i = 0; i < fis_gcI; ++i)
        {
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], fuzzyInput[i][index -
1]);
            else if (index < 0)
                fuzzyFires[r] =
fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
            else
                fuzzyFires[r] =
fis_min(fuzzyFires[r], 1);
    }
}
else
{
    fuzzyFires[r] = FIS_MIN;
    for (i = 0; i < fis_gcI; ++i)
    {
        index = fis_gRI[r][i];
        if (index > 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
        else if (index < 0)
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-
index - 1]);
        else
            fuzzyFires[r] =
fis_max(fuzzyFires[r], 0);
    }
}
fuzzyFires[r] = fis_gRWeight[r] *
fuzzyFires[r];
sW += fuzzyFires[r];
}

if (sW == 0)
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] = ((fis_gOMax[o] +
fis_gOMin[o]) / 2);
    }
}
else
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] =
fis_defuzz_centroid(fuzzyRuleSet, o);
    }
}
}
}

```

Arduino Codes of Node 2 Lettuce

```

#include <DS3231.h>
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <BH1750.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include "fis_header.h"

#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
#define ONE_WIRE_PIN 5
#define RE 6
#define DE 7
#define RF 52
#define DF 53
#define RG 8
#define DG 9
#define FIS_TYPE float
#define FIS_RESOLUTION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE,
FIS_TYPE*);
typedef
FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE,
FIS_TYPE);
typedef
FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int,
_FIS_ARR_OP);

SoftwareSerial mod1(12,13);
SoftwareSerial mod2(10,11);
SoftwareSerial mod3(50,51);
const byte ec[] = {0x01, 0x03, 0x00, 0x15,
0x00, 0x01, 0x95, 0xCE};
const byte ph[] = {0x01, 0x03, 0x00, 0x00,
0x00, 0x01, 0x84, 0x0A};
const byte nitro[] = {0x01,0x03, 0x00, 0x1e,
0x00, 0x01, 0xe4, 0x0c};
const byte phos[] = {0x01,0x03, 0x00, 0x1f,
0x00, 0x01, 0xb5, 0xcc};

const byte pota[] = {0x01,0x03, 0x00, 0x20,
0x00, 0x01, 0x85, 0xc0};
byte value[11];

// Number of inputs to the fuzzy inference system
const int fis_gcI = 8;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 7;
// Number of rules to the fuzzy inference system
const int fis_gcR = 48;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(ONE_WIRE_PIN);
DallasTemperature sensors(&oneWire);
BH1750 lightMeter;
DS3231 myRTC;

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 =
3, d7 = 2;

int prev=0;
int pres=0;

float Value_EC = 0;
float Value_PH = 0;
float Value_N = 0;
float Value_Pho = 0;
float Value_Pot = 0;
float Value_Ambient_Temp = 0;
float Value_Humidity = 0;
float Value_Light_Int= 0;
float Value_Soil_Temp = 0;
float Value_Moisture = 0;
float Water_On = 0;
float Correction_On = 0;
float Humidifier_On = 0;

```

```

byte year;
byte month;
byte date;
byte dow;
byte hour;
byte minute;
byte second;

bool century = false;
bool h12Flag;
bool pmFlag;

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

  mod1.begin(9600);
  mod2.begin(4800);
  mod3.begin(9600);

  Wire.begin();
  lightMeter.begin();
  dht.begin();
  sensors.begin();

  myRTC.setClockMode(false); // set to 24h

  myRTC.setYear(23);
  myRTC.setMonth(5);
  myRTC.setDate(29);
  myRTC.setDoW(7);
  myRTC.setHour(17);
  myRTC.setMinute(0);
  myRTC.setSecond(0);

  pinMode(RE, OUTPUT);
  pinMode(DE, OUTPUT);
  pinMode(RG, OUTPUT);
  pinMode(DG, OUTPUT);
  pinMode(RF, OUTPUT);
  pinMode(DF, OUTPUT);
  delay(1000);

  // initialize the Analog pins for output.
  // Pin mode for Output: Humidifier
  pinMode(31, OUTPUT);

  // Pin mode for Output: Drip_Irrig
  pinMode(32, OUTPUT);
  // Pin mode for Output: Lime
  pinMode(34, OUTPUT);
  // Pin mode for Output: Sulfur
  pinMode(33, OUTPUT);
  // Pin mode for Output: Nitrogen
  pinMode(35, OUTPUT);
  // Pin mode for Output: Phosphorus
  pinMode(36, OUTPUT);
  // Pin mode for Output: Potassium
  pinMode(37, OUTPUT);

}

void loop() {
  SetRelay();
  Soil_EC();
  Soil_PH();
  Soil_NPK();
  Soil_Temperature();
  Soil_Moisture();
  Ambient_Temperature();
  Ambient_Humidity();
  Light_Intensity();
  Serial_Print();
  Alarm();
  delay(50000);
  SetOff();
}

void Serial_Print(){
  Serial.println(" " + String("Node2") +
  "+String(Value_Humidity)+" "+
  String(Value_Ambient_Temp)+" "+
  "+String(Value_Soil_Temp)+" "+
  String(Value_Light_Int)+" "+
  String(Value_Moisture)+" "+
  "+String(Value_EC)+" "+
  String(Value_PH)+" "+ String(Value_N)+" "+
  "+ String(Value_Pho)+" "+
  String(Value_Pot)+" "+
  (myRTC.getMonth(century))+" "+
  (myRTC.getDate())+" "+
  (myRTC.getDoW())+" "+
  (myRTC.getHour(h12Flag, pmFlag))+ " "+
}

```



```

        value[iii] = mod3.read();
    }
}

return value[4];
}

void Soil_Temperature(){
    sensors.requestTemperatures();
    Value_Soil_Temp =
    sensors.getTempCByIndex(0);
}

void Soil_Moisture(){
    Value_Moisture=analogRead(A15);

    Value_Moisture=map(Value_Moisture,0,982
    ,148,0);
    pres=Value_Moisture;
    if(Value_Moisture>100)
    Value_Moisture=100;
    else if(Value_Moisture<0)
    Value_Moisture=0;
    prev=Value_Moisture;
}

void Ambient_Humidity(){
    Value_Humidity = dht.readHumidity() + 3;
//+3
}

void Ambient_Temperature(){
    Value_Ambient_Temp =
    dht.readTemperature() - 1; // -1
}

void Light_Intensity(){
    Value_Light_Int =
    lightMeter.readLightLevel();
}

void Water(){
    Water_On = 1;
    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Phosphorus;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Drip_Irrig
    if ((g_fisOutput[1] > 0) &&
(g_fisOutput[1] <= 5))
    {
        digitalWrite(32, HIGH);
        delay(30000); //TRIAL
    }

    if ((g_fisOutput[1] > 5) &&
(g_fisOutput[1] <= 7))
    {
        digitalWrite(32, LOW);
        delay(20000);
        digitalWrite(32, HIGH);
        delay(30000); //TRIAL
    }

    if ((g_fisOutput[1] > 7) &&
(g_fisOutput[1] > 9))
    {
        digitalWrite(32, LOW);
        delay(40000);
    }
}

```

```

digitalWrite(32, HIGH);
delay(30000); //TRIAL
}

if (g_fisOutput[1] > 9)
{
digitalWrite(32, LOW);
delay(60000);
digitalWrite(32, HIGH);
delay(30000); //TRIAL
}
}

void Humidifier(){
    Humidifier_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
    // Read Input: Soil_pH
    g_fisInput[4] = Value_PH;
    // Read Input: Nitrogen
    g_fisInput[5] = Value_N;
    // Read Input: Phosphorus
    g_fisInput[6] = Value_Pho;
    // Read Input: Potassium
    g_fisInput[7] = Value_Pot;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;
    g_fisOutput[2] = 0;
    g_fisOutput[3] = 0;
    g_fisOutput[4] = 0;
    g_fisOutput[5] = 0;
    g_fisOutput[6] = 0;

    fis_evaluate();

    // Set output vlaue: Humidifier
}

if ((g_fisOutput[0] > 0) &&
(g_fisOutput[0] <= 5))
{
digitalWrite(31, LOW);
delay(30000); //TRIAL
}

if ((g_fisOutput[0] > 5) &&
(g_fisOutput[0] <= 7))
{
digitalWrite(31, HIGH);
delay(60000);
digitalWrite(31, LOW);
delay(30000); //TRIAL
}

if ((g_fisOutput[0] > 7) &&
(g_fisOutput[0] <= 9))
{
digitalWrite(31, HIGH);
delay(120000);
digitalWrite(31, LOW);
delay(30000);
}

if (g_fisOutput[0] > 9)
{
digitalWrite(31, HIGH);
delay(180000);
digitalWrite(31, LOW);
delay(30000); //TRIAL
}

void Correction()
{
    Correction_On = 1;

    // Read Input: Air_Temp
    g_fisInput[0] = Value_Ambient_Temp;
    // Read Input: Humidity
    g_fisInput[1] = Value_Humidity;
    // Read Input: Soil_Moisture
    g_fisInput[2] = Value_Moisture;
    // Read Input: Soil_Temp
    g_fisInput[3] = Value_Soil_Temp;
}

```

```

// Read Input: Soil_pH
g_fisInput[4] = Value_PH;
// Read Input: Nitrogen
g_fisInput[5] = Value_N;
// Read Input: Phosphorus
g_fisInput[6] = Value_Pho;
// Read Input: Potassium
g_fisInput[7] = Value_Pot;

g_fisOutput[0] = 0;
g_fisOutput[1] = 0;
g_fisOutput[2] = 0;
g_fisOutput[3] = 0;
g_fisOutput[4] = 0;
g_fisOutput[5] = 0;
g_fisOutput[6] = 0;

fis_evaluate();

// Set output vlaue: Lime
if ((g_fisOutput[2] > 0) &&
(g_fisOutput[2] <= 3.332))
{
    digitalWrite(34, HIGH);
    delay(30000); //TRIAL
}

if ((g_fisOutput[2] > 3.332) &&
(g_fisOutput[2] <= 6.663))
{
    digitalWrite(34, LOW);
    delay(20000);
    digitalWrite(34, HIGH);
    delay(30000); //TRIAL
}

if (g_fisOutput[2] > 6.663)
{
    digitalWrite(34, LOW);
    delay(40000);
    digitalWrite(34, HIGH);
    delay(30000); //TRIAL
}

// Set output vlaue: Sulfur
if ((g_fisOutput[3] > 0) &&
(g_fisOutput[3] <= 3.332))
{
    digitalWrite(33, HIGH);
    delay(30000); //TRIAL
}

if ((g_fisOutput[3] > 3.332) &&
(g_fisOutput[3] <= 6.663))
{
    digitalWrite(33, LOW);
    delay(20000);
    digitalWrite(33, HIGH);
    delay(30000); //TRIAL
}

if (g_fisOutput[3] > 6.663)
{
    digitalWrite(33, LOW);
    delay(40000);
    digitalWrite(33, HIGH);
    delay(30000); //TRIAL
}

// Set output vlaue: Nitrogen
if ((g_fisOutput[4] > 0) &&
(g_fisOutput[4] <= 3.332))
{
    digitalWrite(35, HIGH);
    delay(30000); //TRIAL
}

if ((g_fisOutput[4] > 3.332) &&
(g_fisOutput[4] <= 6.663))
{
    digitalWrite(35, LOW);
    delay(20000);
    digitalWrite(35, HIGH);
    delay(30000); //TRIAL
}

if (g_fisOutput[4] > 6.663)
{
    digitalWrite(35, LOW);
    delay(40000);
    digitalWrite(35, HIGH);
    delay(30000); //TRIAL
}

```

```

digitalWrite(35, LOW);
delay(40000);
digitalWrite(35, HIGH);
delay(30000); //TRIAL
}

// Set output value: Phosphorus
if ((g_fisOutput[5] > 0) &&
(g_fisOutput[5] <= 3.332))
{
digitalWrite(36, HIGH);
delay(30000); //TRIAL
}

if ((g_fisOutput[5] > 3.332) &&
(g_fisOutput[5] <= 6.663))
{
digitalWrite(36, LOW);
delay(20000);
digitalWrite(36, HIGH);
delay(30000); //TRIAL
}

if (g_fisOutput[5] > 6.663)
{
digitalWrite(36, LOW);
delay(40000);
digitalWrite(36, HIGH);
delay(30000); //TRIAL
}

// Set output value: Potassium
if ((g_fisOutput[6] > 0) &&
(g_fisOutput[6] <= 3.332))
{
digitalWrite(37, HIGH);
delay(30000); //TRIAL
}

if ((g_fisOutput[6] > 3.332) &&
(g_fisOutput[6] <= 6.663))
{
digitalWrite(37, LOW);

delay(20000);
digitalWrite(37, HIGH);
delay(30000); //TRIAL
}

if (g_fisOutput[6] > 6.663)
{
digitalWrite(37, LOW);
delay(40000);
digitalWrite(37, HIGH);
delay(30000); //TRIAL
}

//MAGKAIBANG TIME NODE 1 NODE
2()
void Alarm(){
uint8_t checkHour =
myRTC.getHour(h12Flag, pmFlag);
uint8_t checkDoW = myRTC.getDoW();

if (checkHour == 9) {
Water();
//digitalWrite(31, HIGH);
} if (checkHour == 16 & (checkDoW != 2 |
checkDoW != 5)) {
Water();
}
if (checkDoW == 2 & checkHour == 16) {
Correction();
delay(120000);
} if (checkDoW == 5 & checkHour == 16)
{
Correction();
delay(120000);
// } if (checkHour == 16) {
// digitalWrite(31, LOW);
// } if (checkHour == 17) {
// Humidifier();
// }
}

void SetRelay(){
digitalWrite(31,LOW);
digitalWrite(32,HIGH);
}

```

```

digitalWrite(33,HIGH);
digitalWrite(34,HIGH);
digitalWrite(35,HIGH);
digitalWrite(36,HIGH);
digitalWrite(37,HIGH);
}

void SetOff(){
    Water_On = 0;
    Correction_On = 0;
    Humidifier_On = 0;
}

//*****
*****  

*  

// Support functions for Fuzzy Inference  

System  

//*****  

*****  

*  

// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d
= p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0
: ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0
: ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x,
FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return
(FIS_TYPE) (x == a);
    if (a == b) return (FIS_TYPE) (t2 * (b <
= x) * (x <= c));
}

if (b == c) return (FIS_TYPE) (t1 * (a <=
x) * (x <= b));
    t1 = min(t1, t2);
    return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a,
FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a,
FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE
*array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}

//*****
*****  

*  

// Data for Fuzzy Inference System  

//*****  

*****  

*  

// Pointers to the implementations of  

member functions

```

```

FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

// Count of member function for each Input
int fis_gIMFCount[] = { 3, 3, 4, 3, 5, 3, 3, 3
};

// Count of member function for each Output
int fis_gOMFCount[] = { 4, 4, 3, 3, 3, 3, 3 };

// Coefficients for the Input Member
Functions
FIS_TYPE fis_gMFI0Coeff1[] = { 0, 0, 12,
18 };
FIS_TYPE fis_gMFI0Coeff2[] = { 14, 21,
28 };
FIS_TYPE fis_gMFI0Coeff3[] = { 24, 30,
50, 50 };
FIS_TYPE* fis_gMFI0Coeff[] = {
fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { 0, 0, 30,
45 };
FIS_TYPE fis_gMFI1Coeff2[] = { 35, 55,
75 };
FIS_TYPE fis_gMFI1Coeff3[] = { 65, 80,
100, 100 };
FIS_TYPE* fis_gMFI1Coeff[] = {
fis_gMFI1Coeff1, fis_gMFI1Coeff2,
fis_gMFI1Coeff3 };
FIS_TYPE fis_gMFI2Coeff1[] = { 60, 75,
90 };
FIS_TYPE fis_gMFI2Coeff2[] = { 80, 95,
100, 100 };
FIS_TYPE fis_gMFI2Coeff3[] = { 0, 0, 25,
40 };
FIS_TYPE fis_gMFI2Coeff4[] = { 30, 50,
70 };
FIS_TYPE* fis_gMFI2Coeff[] = {
fis_gMFI2Coeff1, fis_gMFI2Coeff2,
fis_gMFI2Coeff3, fis_gMFI2Coeff4 };
FIS_TYPE fis_gMFI3Coeff1[] = { 0, 0, 11,
18 };

FIS_TYPE fis_gMFI3Coeff2[] = { 14, 20,
26 };
FIS_TYPE fis_gMFI3Coeff3[] = { 22, 29,
50, 50 };
FIS_TYPE* fis_gMFI3Coeff[] = {
fis_gMFI3Coeff1, fis_gMFI3Coeff2,
fis_gMFI3Coeff3 };
FIS_TYPE fis_gMFI4Coeff1[] = { 0, 0, 1,
3.5 };
FIS_TYPE fis_gMFI4Coeff2[] = { 2.5, 4.5,
6.5 };
FIS_TYPE fis_gMFI4Coeff3[] = { 10,
11.75, 14.6, 19.3 };
FIS_TYPE fis_gMFI4Coeff4[] = { 6.5, 8.75,
11 };
FIS_TYPE fis_gMFI4Coeff5[] = { 5.5, 6.5,
7.5 };
FIS_TYPE* fis_gMFI4Coeff[] = {
fis_gMFI4Coeff1, fis_gMFI4Coeff2,
fis_gMFI4Coeff3, fis_gMFI4Coeff4,
fis_gMFI4Coeff5 };
FIS_TYPE fis_gMFI5Coeff1[] = { 0, 0, 1,
3.5 };
FIS_TYPE fis_gMFI5Coeff2[] = { 1.5, 3.75,
6 };
FIS_TYPE fis_gMFI5Coeff3[] = { 4, 6.5,
31.4, 41.4 };
FIS_TYPE* fis_gMFI5Coeff[] = {
fis_gMFI5Coeff1, fis_gMFI5Coeff2,
fis_gMFI5Coeff3 };
FIS_TYPE fis_gMFI6Coeff1[] = { 0, 0, 6,
11 };
FIS_TYPE fis_gMFI6Coeff2[] = { 9, 14, 19
};
FIS_TYPE fis_gMFI6Coeff3[] = { 17, 22,
29.11, 38.43 };
FIS_TYPE* fis_gMFI6Coeff[] = {
fis_gMFI6Coeff1, fis_gMFI6Coeff2,
fis_gMFI6Coeff3 };
FIS_TYPE fis_gMFI7Coeff1[] = { 0, 0, 7.5,
13 };
FIS_TYPE fis_gMFI7Coeff2[] = { 11, 16,
21 };
FIS_TYPE fis_gMFI7Coeff3[] = { 19, 24.5,
33.3, 44 };

```

```

FIS_TYPE* fis_gMFI7Coeff[] = {
fis_gMFI7Coeff1, fis_gMFI7Coeff2,
fis_gMFI7Coeff3 };

FIS_TYPE** fis_gMFICoeff[] = {
fis_gMFI0Coeff, fis_gMFI1Coeff,
fis_gMFI2Coeff, fis_gMFI3Coeff,
fis_gMFI4Coeff, fis_gMFI5Coeff,
fis_gMFI6Coeff, fis_gMFI7Coeff };

// Coefficients for the Output Member
Functions
FIS_TYPE fis_gMFO0Coeff1[] = { 4, 6, 8
};
FIS_TYPE fis_gMFO0Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO0Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO0Coeff4[] = { 0, 0, 3,
6 };
FIS_TYPE* fis_gMFO0Coeff[] = {
fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3, fis_gMFO0Coeff4 };
FIS_TYPE fis_gMFO1Coeff1[] = { 0, 0, 3,
6 };
FIS_TYPE fis_gMFO1Coeff2[] = { 6, 8, 10
};
FIS_TYPE fis_gMFO1Coeff3[] = { 8, 11,
15, 15 };
FIS_TYPE fis_gMFO1Coeff4[] = { 4, 6, 8
};
FIS_TYPE* fis_gMFO1Coeff[] = {
fis_gMFO1Coeff1, fis_gMFO1Coeff2,
fis_gMFO1Coeff3, fis_gMFO1Coeff4 };
FIS_TYPE fis_gMFO2Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO2Coeff2[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE fis_gMFO2Coeff3[] = { 2.833,
5, 7.166 };
FIS_TYPE* fis_gMFO2Coeff[] = {
fis_gMFO2Coeff1, fis_gMFO2Coeff2,
fis_gMFO2Coeff3 };
FIS_TYPE fis_gMFO3Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO3Coeff2[] = { 2.833,
5, 7.166 };

```

```

FIS_TYPE fis_gMFO3Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO3Coeff[] = {
fis_gMFO3Coeff1, fis_gMFO3Coeff2,
fis_gMFO3Coeff3 };
FIS_TYPE fis_gMFO4Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO4Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO4Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO4Coeff[] = {
fis_gMFO4Coeff1, fis_gMFO4Coeff2,
fis_gMFO4Coeff3 };
FIS_TYPE fis_gMFO5Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO5Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO5Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO5Coeff[] = {
fis_gMFO5Coeff1, fis_gMFO5Coeff2,
fis_gMFO5Coeff3 };
FIS_TYPE fis_gMFO6Coeff1[] = { 0, 0, 2,
3.83 };
FIS_TYPE fis_gMFO6Coeff2[] = { 2.833,
5, 7.166 };
FIS_TYPE fis_gMFO6Coeff3[] = { 6.16, 8,
10.4, 13.73 };
FIS_TYPE* fis_gMFO6Coeff[] = {
fis_gMFO6Coeff1, fis_gMFO6Coeff2,
fis_gMFO6Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = {
fis_gMFO0Coeff, fis_gMFO1Coeff,
fis_gMFO2Coeff, fis_gMFO3Coeff,
fis_gMFO4Coeff, fis_gMFO5Coeff,
fis_gMFO6Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 1, 0 };
int fis_gMFI1[] = { 0, 1, 0 };
int fis_gMFI2[] = { 1, 0, 0, 1 };
int fis_gMFI3[] = { 0, 1, 0 };
int fis_gMFI4[] = { 0, 1, 0, 1, 1 };
int fis_gMFI5[] = { 0, 1, 0 };
int fis_gMFI6[] = { 0, 1, 0 };

```

```

int fis_gMFI7[] = { 0, 1, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1,
fis_gMFI2, fis_gMFI3, fis_gMFI4,
fis_gMFI5, fis_gMFI6, fis_gMFI7};

// Output membership function set
int fis_gMFO0[] = { 1, 1, 0, 0 };
int fis_gMFO1[] = { 0, 1, 0, 1 };
int fis_gMFO2[] = { 0, 0, 1 };
int fis_gMFO3[] = { 0, 1, 0 };
int fis_gMFO4[] = { 0, 1, 0 };
int fis_gMFO5[] = { 0, 1, 0 };
int fis_gMFO6[] = { 0, 1, 0 };
int* fis_gMFO[] = { fis_gMFO0,
fis_gMFO1, fis_gMFO2, fis_gMFO3,
fis_gMFO4, fis_gMFO5, fis_gMFO6};

// Rule Weights
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1 };

// Rule Type
int fis_gRTType[] = { 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI1[] = { 2, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI2[] = { 1, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI3[] = { 2, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI4[] = { 2, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI5[] = { 3, 2, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI6[] = { 3, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI7[] = { 1, 3, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI8[] = { 3, 1, 0, 0, 0, 0, 0, 0, 0 };
int fis_gRI9[] = { 0, 0, 2, 1, 0, 0, 0, 0, 0 };
int fis_gRI10[] = { 0, 0, 2, 2, 0, 0, 0, 0, 0 };
int fis_gRI11[] = { 0, 0, 2, 3, 0, 0, 0, 0, 0 };
int fis_gRI12[] = { 0, 0, 1, 1, 0, 0, 0, 0, 0 };
int fis_gRI13[] = { 0, 0, 1, 2, 0, 0, 0, 0, 0 };
int fis_gRI14[] = { 0, 0, 1, 3, 0, 0, 0, 0, 0 };
int fis_gRI15[] = { 0, 0, 4, 1, 0, 0, 0, 0, 0 };

```

```

int fis_gRI16[] = { 0, 0, 4, 2, 0, 0, 0, 0, 0 };
int fis_gRI17[] = { 0, 0, 4, 3, 0, 0, 0, 0, 0 };
int fis_gRI18[] = { 0, 0, 3, 1, 0, 0, 0, 0, 0 };
int fis_gRI19[] = { 0, 0, 3, 2, 0, 0, 0, 0, 0 };
int fis_gRI20[] = { 0, 0, 3, 3, 0, 0, 0, 0, 0 };
int fis_gRI21[] = { 0, 0, 0, 1, 1, 0, 0, 0, 0 };
int fis_gRI22[] = { 0, 0, 0, 1, 2, 0, 0, 0, 0 };
int fis_gRI23[] = { 0, 0, 0, 1, 5, 0, 0, 0, 0 };
int fis_gRI24[] = { 0, 0, 0, 2, 1, 0, 0, 0, 0 };
int fis_gRI25[] = { 0, 0, 0, 2, 2, 0, 0, 0, 0 };
int fis_gRI26[] = { 0, 0, 0, 2, 5, 0, 0, 0, 0 };
int fis_gRI27[] = { 0, 0, 0, 3, 1, 0, 0, 0, 0 };
int fis_gRI28[] = { 0, 0, 0, 3, 2, 0, 0, 0, 0 };
int fis_gRI29[] = { 0, 0, 0, 3, 5, 0, 0, 0, 0 };
int fis_gRI30[] = { 0, 0, 0, 1, 3, 0, 0, 0, 0 };
int fis_gRI31[] = { 0, 0, 0, 1, 4, 0, 0, 0, 0 };
int fis_gRI32[] = { 0, 0, 0, 1, 5, 0, 0, 0, 0 };
int fis_gRI33[] = { 0, 0, 0, 2, 3, 0, 0, 0, 0 };
int fis_gRI34[] = { 0, 0, 0, 2, 4, 0, 0, 0, 0 };
int fis_gRI35[] = { 0, 0, 0, 2, 5, 0, 0, 0, 0 };
int fis_gRI36[] = { 0, 0, 0, 3, 3, 0, 0, 0, 0 };
int fis_gRI37[] = { 0, 0, 0, 3, 4, 0, 0, 0, 0 };
int fis_gRI38[] = { 0, 0, 0, 3, 5, 0, 0, 0, 0 };
int fis_gRI39[] = { 0, 0, 0, 0, 0, 1, 0, 0, 0 };
int fis_gRI40[] = { 0, 0, 0, 0, 0, 2, 0, 0, 0 };
int fis_gRI41[] = { 0, 0, 0, 0, 0, 3, 0, 0, 0 };
int fis_gRI42[] = { 0, 0, 0, 0, 0, 0, 1, 0, 0 };
int fis_gRI43[] = { 0, 0, 0, 0, 0, 0, 2, 0, 0 };
int fis_gRI44[] = { 0, 0, 0, 0, 0, 0, 3, 0, 0 };
int fis_gRI45[] = { 0, 0, 0, 0, 0, 0, 0, 1, 0 };
int fis_gRI46[] = { 0, 0, 0, 0, 0, 0, 0, 2, 0 };
int fis_gRI47[] = { 0, 0, 0, 0, 0, 0, 0, 3, 0 };
int* fis_gRI[] = { fis_gRI0, fis_gRI1,
fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5,
fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9,
fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13,
fis_gRI14, fis_gRI15, fis_gRI16, fis_gRI17,
fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21,
fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25,
fis_gRI26, fis_gRI27, fis_gRI28, fis_gRI29,
fis_gRI30, fis_gRI31, fis_gRI32, fis_gRI33,
fis_gRI34, fis_gRI35, fis_gRI36, fis_gRI37,
fis_gRI38, fis_gRI39, fis_gRI40, fis_gRI41,
fis_gRI42, fis_gRI43, fis_gRI44, fis_gRI45,
fis_gRI46, fis_gRI47 };

```

```

// Rule Outputs
int fis_gRO0[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO1[] = { 1, 0, 0, 0, 0, 0, 0 };
int fis_gRO2[] = { 4, 0, 0, 0, 0, 0, 0 };
int fis_gRO3[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO4[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO5[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO6[] = { 3, 0, 0, 0, 0, 0, 0 };
int fis_gRO7[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO8[] = { 2, 0, 0, 0, 0, 0, 0 };
int fis_gRO9[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO10[] = { 4, 1, 0, 0, 0, 0, 0 };
int fis_gRO11[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO12[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO13[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO14[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO15[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO16[] = { 0, 4, 0, 0, 0, 0, 0 };
int fis_gRO17[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO18[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO19[] = { 0, 2, 0, 0, 0, 0, 0 };
int fis_gRO20[] = { 0, 3, 0, 0, 0, 0, 0 };
int fis_gRO21[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO22[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO23[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO24[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO25[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO26[] = { 0, 0, 1, 0, 0, 0, 0 };
int fis_gRO27[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO28[] = { 0, 0, 2, 0, 0, 0, 0 };
int fis_gRO29[] = { 0, 0, 3, 0, 0, 0, 0 };
int fis_gRO30[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO31[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO32[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO33[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO34[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO35[] = { 0, 0, 0, 1, 0, 0, 0 };
int fis_gRO36[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO37[] = { 0, 0, 0, 3, 0, 0, 0 };
int fis_gRO38[] = { 0, 0, 0, 2, 0, 0, 0 };
int fis_gRO39[] = { 0, 0, 0, 0, 3, 0, 0 };
int fis_gRO40[] = { 0, 0, 0, 0, 2, 0, 0 };
int fis_gRO41[] = { 0, 0, 0, 0, 1, 0, 0 };
int fis_gRO42[] = { 0, 0, 0, 0, 0, 3, 0 };
int fis_gRO43[] = { 0, 0, 0, 0, 0, 2, 0 };
int fis_gRO44[] = { 0, 0, 0, 0, 0, 1, 0 };

int fis_gRO45[] = { 0, 0, 0, 0, 0, 0, 3 };
int fis_gRO46[] = { 0, 0, 0, 0, 0, 0, 2 };
int fis_gRO47[] = { 0, 0, 0, 0, 0, 0, 1 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1,
fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9,
fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15,
fis_gRO16, fis_gRO17, fis_gRO18,
fis_gRO19, fis_gRO20, fis_gRO21,
fis_gRO22, fis_gRO23, fis_gRO24,
fis_gRO25, fis_gRO26, fis_gRO27,
fis_gRO28, fis_gRO29, fis_gRO30,
fis_gRO31, fis_gRO32, fis_gRO33,
fis_gRO34, fis_gRO35, fis_gRO36,
fis_gRO37, fis_gRO38, fis_gRO39,
fis_gRO40, fis_gRO41, fis_gRO42,
fis_gRO43, fis_gRO44, fis_gRO45,
fis_gRO46, fis_gRO47 };

// Input range Min
FIS_TYPE fis_gIMin[] = { 0, 0, 0, 0, 0, 0, 0,
0 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 50, 100, 100,
50, 14, 30, 28, 32 };

// Output range Min
FIS_TYPE fis_gOMin[] = { 0, 0, 0, 0, 0, 0,
0 };

// Output range Max
FIS_TYPE fis_gOMax[] = { 15, 15, 10, 10,
10, 10, 10 };

//***** *****
***** *****
*
// Data dependent support functions for
// Fuzzy Inference System
//***** *****
***** *****
*
FIS_TYPE fis_MF_out(FIS_TYPE**  

fuzzyRuleSet, FIS_TYPE x, int o)

```

```

{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut =
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 -
(fis_gMF[fis_gMFO[o][index]])(x,
fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut,
fuzzyRuleSet[1][r]);
    }
    return
fis_array_operation(fuzzyRuleSet[0],
fis_gcR, fis_max);
}

FIS_TYPE
fis_defuzz_centroid(FIS_TYPE**
fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] -
fis_gOMin[o]) / (FIS_RESOLUTION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve
    // formed by the MF outputs
    for (i = 0; i < FIS_RESOLUTION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step *
fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] +
fis_gOMin[o]) / 2) : (momentum / area));
}

//*****
***** *
// Fuzzy Inference System
//*****
***** *
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput2[] = { 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput3[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput4[] = { 0, 0, 0, 0, 0
};

    FIS_TYPE fuzzyInput5[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput6[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput7[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] = {
fuzzyInput0, fuzzyInput1, fuzzyInput2,
fuzzyInput3, fuzzyInput4, fuzzyInput5,
fuzzyInput6, fuzzyInput7, };

    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0
};
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0, 0
};
    FIS_TYPE fuzzyOutput2[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput3[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput4[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput5[] = { 0, 0, 0 };
    FIS_TYPE fuzzyOutput6[] = { 0, 0, 0 };
}

```


ANNEX VII
SOIL-N-WAN Gateway
Codes

Codes to Retrieve Data from the Sensor Nodes and Upload them to the Database

```

import sys
import time
import io
import decimal
import pyrebase
import serial
import RPi.GPIO as GPIO

ser = serial.Serial('/dev/ttyUSB0',9600)

while True:
    try:
        if ser.in_waiting > 0:
            x = ser.readline().decode('utf-
8').strip();
            x1 = x.split();
            print(x1);

            if x1[0] == 'Node1':
                NodeNo, humN1, amtempN1,
                stempN1, luxN1, moiN1, ecN1, phN1,
                val1nN1,           val2pN1, val3kN1,
                Month1, Date1, DoW1, Hour1, Minute1,
                Water_On1, Hum_On1, Cor_On1 = x1
            else:
                NodeNo, humN2, amtempN2,
                stempN2, luxN2, moiN2, ecN2, phN2,
                val1nN2, val2pN2, val3kN2, Month2,
                Date2, DoW2, Hour2, Minute2, Water_On2,
                Hum_On2, Cor_On2 = x1

                timestamp1 = str(Date1) + str("-") +
                str(Hour1) + str("-") + str(Minute1);
                timestamp2 = str(Date2) + str("-") +
                str(Hour2) + str("-") + str(Minute2);

            config = {
                "apiKey": "AIzaSyAICGl1t7c6dlPUx2noP_qzkQwCd
                QGJc8A",
                "authDomain": "monitoringapplication-
                5ce24.firebaseio.com",

```

```

                "databaseURL": "https://monitoringapplication-5ce24-
                default- rtbd.firebaseio.com",
                "projectId": "monitoringapplication-5ce24",
                "storageBucket": "monitoringapplication-5ce24.appspot.com",
                "messagingSenderId": "576472957352",
                "appId": "1:576472957352:web:9a63b5052b4dc8fe6
                bdd9d",
                "measurementId": "G-
                1F044JHVSF"
            };

            firebase =
            pyrebase.initialize_app(config)
            db = firebase.database()

            # Set Data for Graph
            db.child('Node 1 -
            Humidity').update({timestamp1: humN1})
            db.child('Node 1 - Ambient
            Temperature').update({timestamp1:
            amtempN1})
            db.child('Node 1 -
            Temperature').update({timestamp1:
            stempN1})
            db.child('Node 1 - Light
            Intensity').update({timestamp1: luxN1})
            db.child('Node 1 -
            Moisture').update({timestamp1: moiN1})
            db.child('Node 1 - Electrical
            Conductivity').update({timestamp1: ecN1})
            db.child('Node 1 - pH
            Level').update({timestamp1: phN1})
            db.child('Node 1 -
            Nitrogen').update({timestamp1: val1nN1})
            db.child('Node 1 -
            Phosphorus').update({timestamp1:
            val2pN1})
            db.child('Node 1 -
            Potassium').update({timestamp1: val3kN1})

```

```

        db.child('Node 2 -
Humidity').update({timestamp2: humN2})
        db.child('Node 2 - Ambient
Temperature').update({timestamp2:
amtempN2})
        db.child('Node 2 -
Temperature').update({timestamp2:
stempN2})
        db.child('Node 2 - Light
Intensity').update({timestamp2: luxN2})
        db.child('Node 2 -
Moisture').update({timestamp2: moiN2})
        db.child('Node 2 - Electrical
Conductivity').update({timestamp2: ecN2})
        db.child('Node 2 - pH
Level').update({timestamp2: phN2})
        db.child('Node 2 -
Nitrogen').update({timestamp2: val1nN2})
        db.child('Node 2 -
Phosphorus').update({timestamp2:
val2pN2})
        db.child('Node 2 -
Potassium').update({timestamp2: val3kN2})

#Set Data
        db.child("Node
1").child("Humidity").set(humN1)
        db.child("Node 1").child("Ambient
Temperature").set(amtempN1)
        db.child("Node
1").child("Temperature").set(stempN1)
        db.child("Node 1").child("Light
Intensity").set(luxN1)
        db.child("Node
1").child("Moisture").set(moiN1)
        db.child("Node 1").child("Electrical
Conductivity").set(ecN1)
        db.child("Node 1").child("pH
Level").set(phN1)
        db.child("Node
1").child("Nitrogen").set(val1nN1)
        db.child("Node
1").child("Phosphorus").set(val2pN1)
        db.child("Node
1").child("Potassium").set(val3kN1)

        db.child("Node
1").child("Year").set(Year1)
        db.child("Node
1").child("Month").set(Month1)
        db.child("Node
1").child("Date").set(Date1)
        db.child("Node 1").child("Day of the
Week").set(DoW1)
        db.child("Node
1").child("Hour").set(Hour1)
        db.child("Node
1").child("Minute").set(Minute1)
        db.child("Node
1").child("Water_On").set(Water_On1)
        db.child("Node
1").child("Humidifier_On").set(Hum_On1)
        db.child("Node
1").child("Correction_On").set(Cor_On1)

        db.child("Node
2").child("Humidity").set(humN2)
        db.child("Node 2").child("Ambient
Temperature").set(amtempN2)
        db.child("Node
2").child("Temperature").set(stempN2)
        db.child("Node 2").child("Light
Intensity").set(luxN2)
        db.child("Node
2").child("Moisture").set(moiN2)
        db.child("Node 2").child("Electrical
Conductivity").set(ecN2)
        db.child("Node 2").child("pH
Level").set(phN2)
        db.child("Node
2").child("Nitrogen").set(val1nN2)
        db.child("Node
2").child("Phosphorus").set(val2pN2)
        db.child("Node
2").child("Potassium").set(val3kN2)
        db.child("Node
2").child("Month").set(Month2)
        db.child("Node
2").child("Date").set(Date2)
        db.child("Node 2").child("Day of the
Week").set(DoW2)

```

```
        db.child("Node
2").child("Hour").set(Hour2)
        db.child("Node
2").child("Minute").set(Minute2)
        db.child("Node
2").child("Water_On").set(Water_On2)
        db.child("Node
2").child("Humidifier_On").set(Hum_On22)

        db.child("Node
2").child("Correction_On").set(Cor_On2)
except Exception:
    continue
```

ANNEX VIII

SOIL-N-WAN

**Monitoring Application Blocks and
Codes**

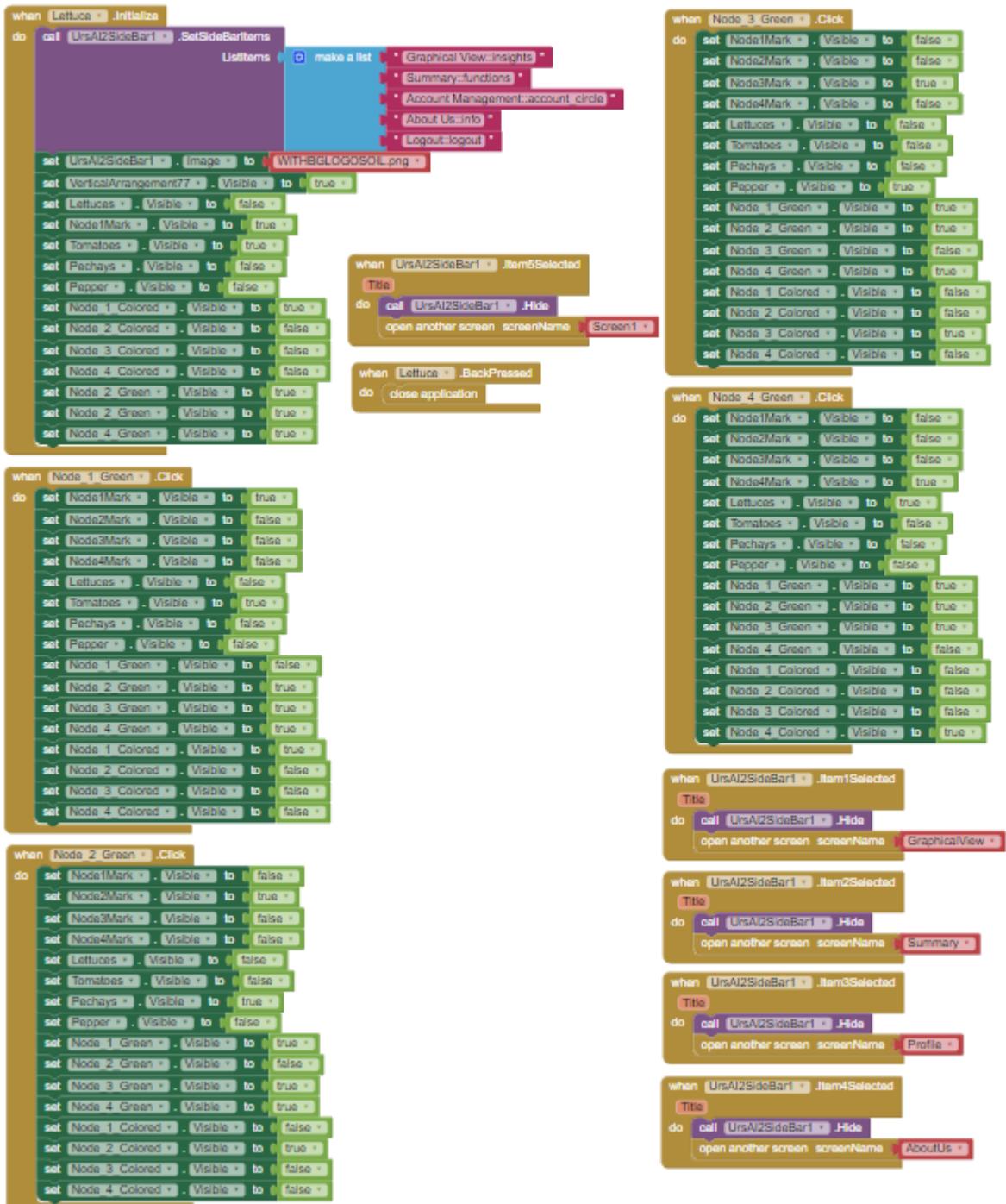
Login Screen Code Blocks

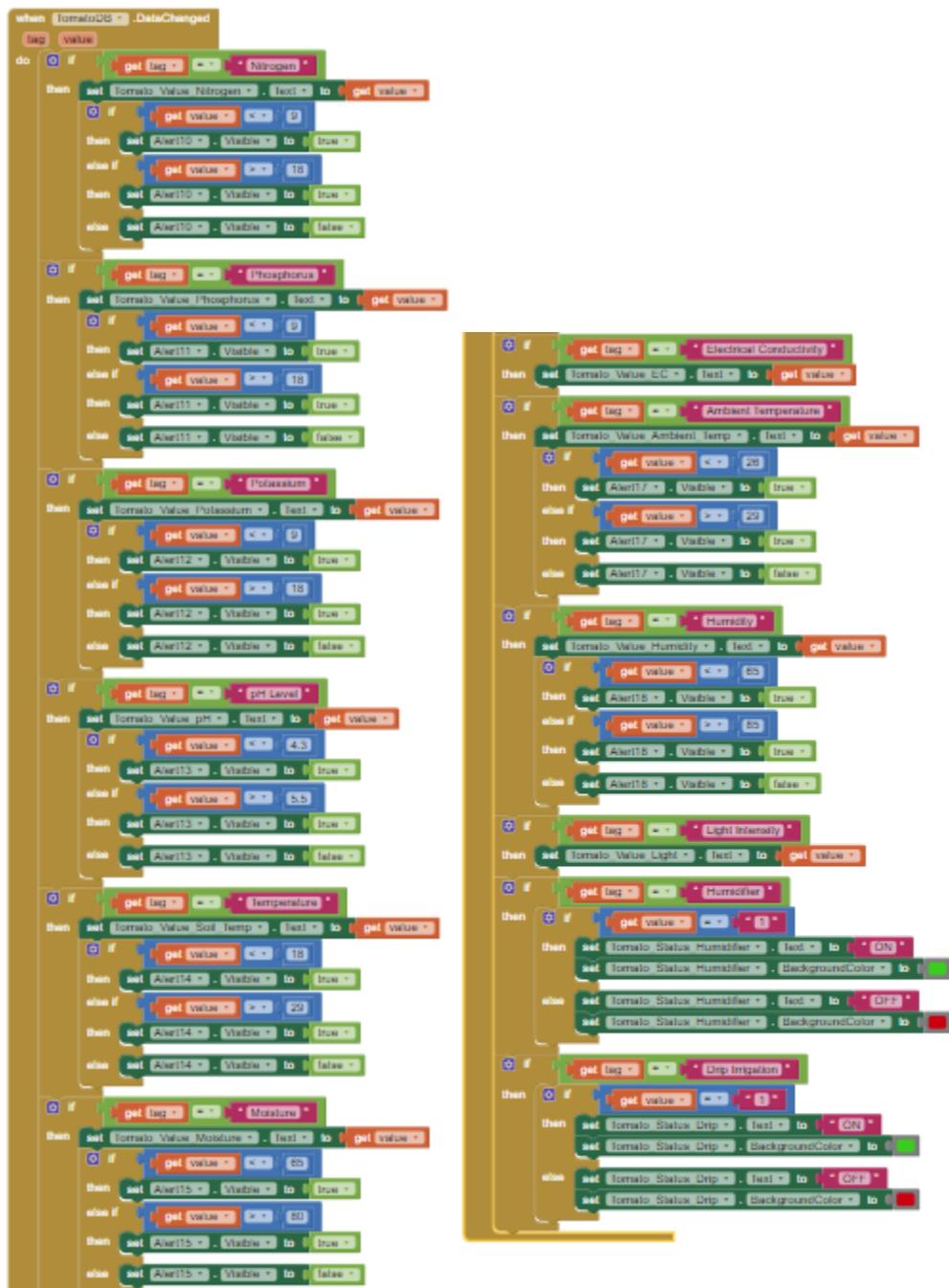
```
when Login .Click
do call FirebaseDatabase1 .GetValue
    tag Username .Text
    valueIfTagNotThere " NA "
call Screen1 .HideKeyboard

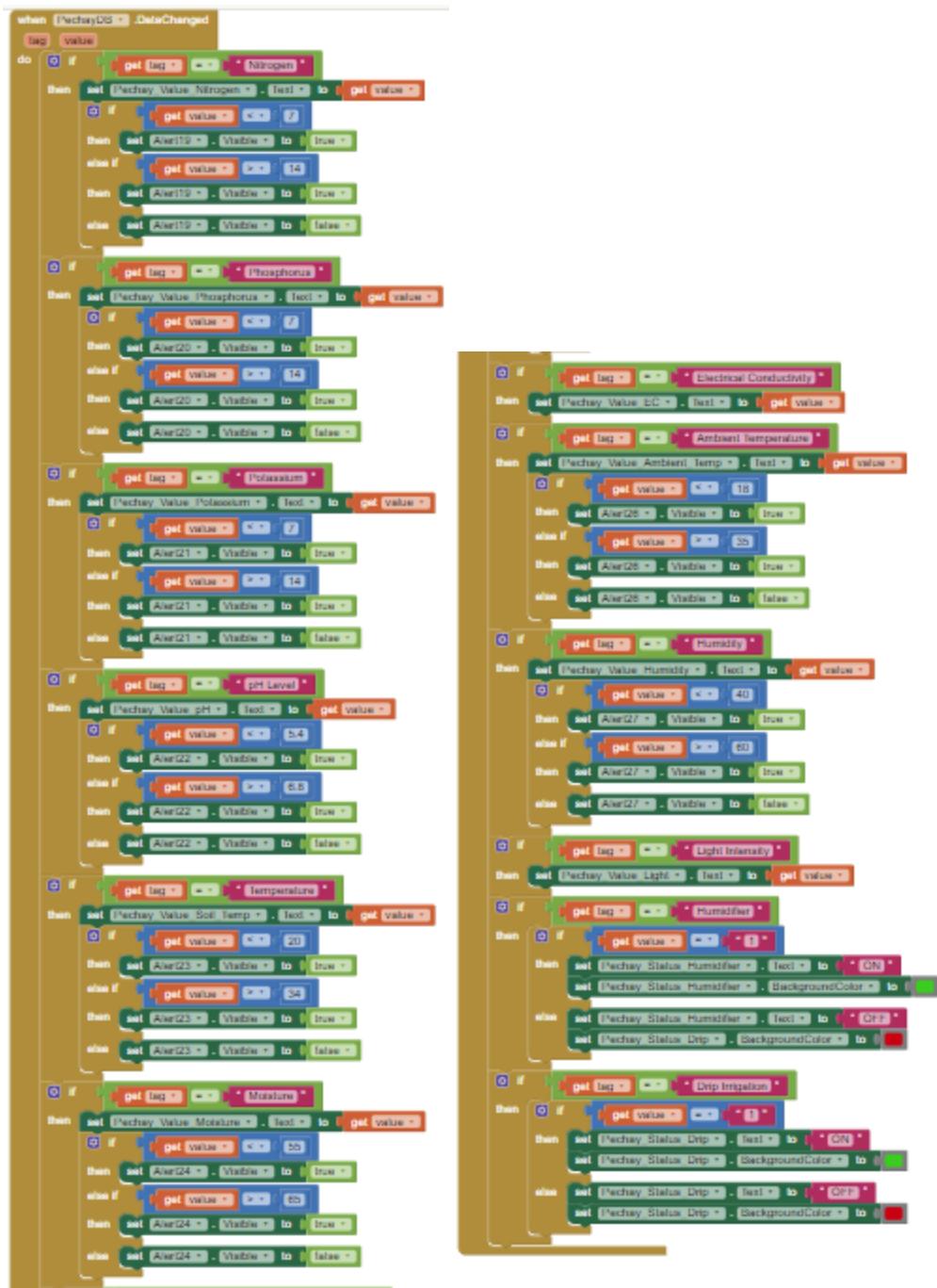
when FirebaseDatabase1 .GetValue
tag value
do if get tag = Username .Text
then if get value = PasswordTextBox1 .Text
then open another screen screenName Lettuce
else set Username .Text to " "
set PasswordTextBox1 .Text to " "
set label .Visible to true
call Screen1 .HideKeyboard

when Screen1 .BackPressed
do close application
```

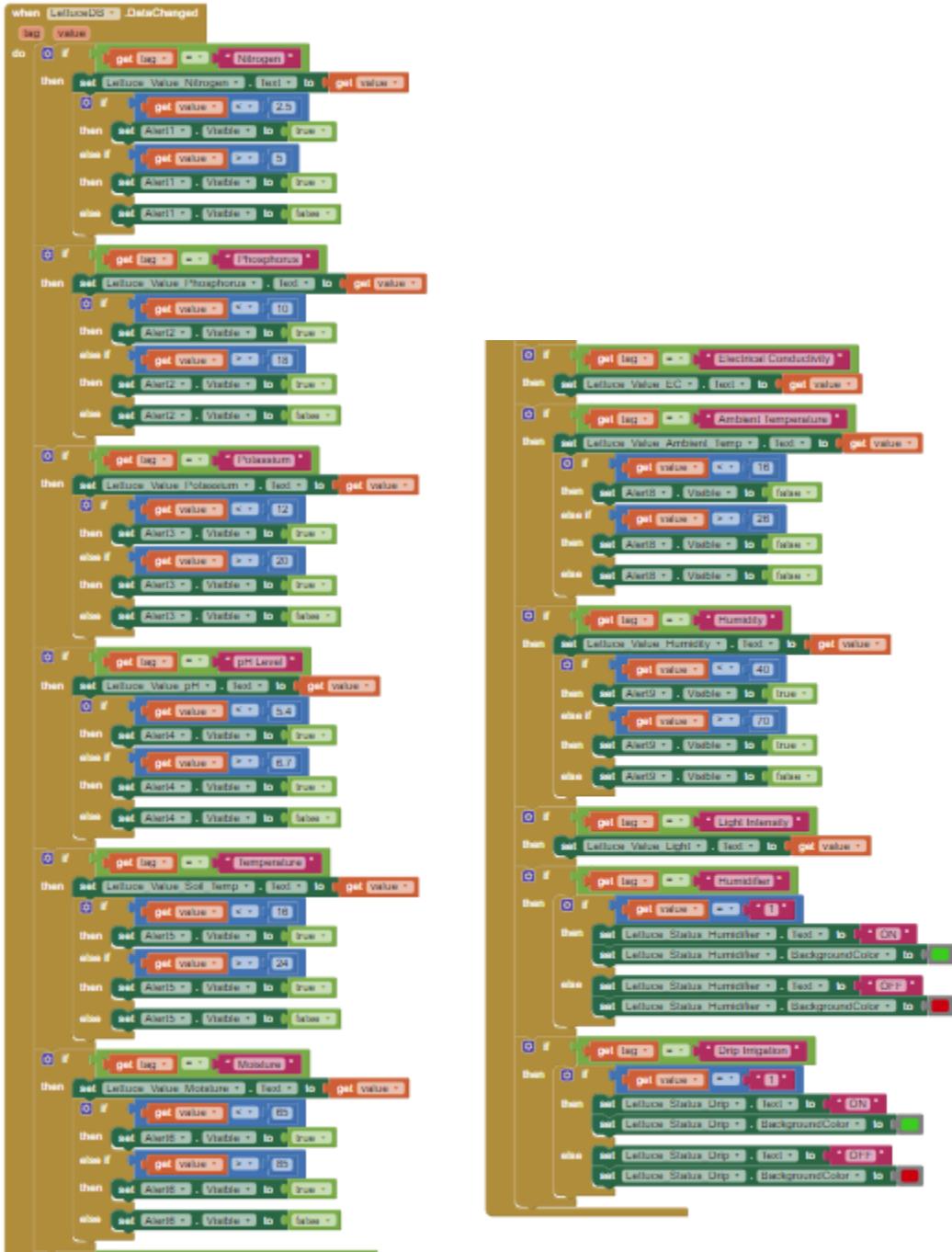
Monitoring Screen Code Blocks











Profile Management Code Blocks

```

when [Apply v] Click
do call [FirebaseDatabase1 v] GetValue
    tag: Soloist
    valueIfTagNotThere: Na

when [FirebaseDatabase1 v] GetValue [tag: value]
do if [value = OldPass Text and NewPass Text = OldPass Text and NewPass Text = ConPass Text and ConPass Text = ConPass Text]
then call [FirebaseDatabase1 v] StoreValue
    tag: Soloist
    valueToStore: ConPass Text
call [Invisible v] HideKeyboard
open another screen screenName: Screen1
else
set Invalid [Visible to true]
set Spacer [Visible to false]
set OldPass [Text to .]
set NewPass [Text to .]
set ConPass [Text to .]
call [Invisible v] HideKeyboard

when [Cancel v] Click
do set OldPass [Text to .]
set NewPass [Text to .]
set ConPass [Text to .]
call [Invisible v] HideKeyboard
open another screen screenName: Lettuce

```

Graphical View Code Blocks

```

when [GraphicalView v] Initialize
do
set VerticalArrangement77 [Visible to true]
set Node_1_Green [Visible to true]
set Node_2_Green [Visible to true]
set Node_3_Green [Visible to true]
set Node_4_Green [Visible to true]

when [GraphicalView v] BackPressed
do
close screen

when [Node_1_Green v] Click
do
set [ActivityStarter1 v] DataUri to "https://sol-n-wan-node1.web.app"
call [ActivityStarter1 v] StartActivity
set Node1Mark [Visible to true]
set Node2Mark [Visible to false]
set Node3Mark [Visible to false]
set Node4Mark [Visible to false]
set Node_1_Green [Visible to false]
set Node_2_Green [Visible to true]
set Node_3_Green [Visible to true]
set Node_4_Green [Visible to true]
set Node_1_Colored [Visible to true]
set Node_2_Colored [Visible to false]
set Node_3_Colored [Visible to false]
set Node_4_Colored [Visible to false]

when [Node_2_Green v] Click
do
set [ActivityStarter1 v] DataUri to "https://sol-n-wan-node2.web.app"
call [ActivityStarter1 v] StartActivity
set Node1Mark [Visible to false]
set Node2Mark [Visible to true]
set Node3Mark [Visible to false]
set Node4Mark [Visible to false]
set Node_1_Green [Visible to true]
set Node_2_Green [Visible to false]
set Node_3_Green [Visible to true]
set Node_4_Green [Visible to true]
set Node_1_Colored [Visible to false]
set Node_2_Colored [Visible to true]
set Node_3_Colored [Visible to true]
set Node_4_Colored [Visible to false]

when [Node_3_Green v] Click
do
set [ActivityStarter1 v] DataUri to "https://sol-n-wan-node3.web.app"
call [ActivityStarter1 v] StartActivity
set Node1Mark [Visible to false]
set Node2Mark [Visible to false]
set Node3Mark [Visible to true]
set Node4Mark [Visible to false]
set Node_1_Green [Visible to true]
set Node_2_Green [Visible to true]
set Node_3_Green [Visible to false]
set Node_4_Green [Visible to true]
set Node_1_Colored [Visible to true]
set Node_2_Colored [Visible to false]
set Node_3_Colored [Visible to false]
set Node_4_Colored [Visible to true]

when [Node_4_Green v] Click
do
set [ActivityStarter1 v] DataUri to "https://sol-n-wan-node4.web.app"
call [ActivityStarter1 v] StartActivity
set Node1Mark [Visible to false]
set Node2Mark [Visible to false]
set Node3Mark [Visible to false]
set Node4Mark [Visible to true]
set Node_1_Green [Visible to true]
set Node_2_Green [Visible to true]
set Node_3_Green [Visible to true]
set Node_4_Green [Visible to false]
set Node_1_Colored [Visible to false]
set Node_2_Colored [Visible to true]
set Node_3_Colored [Visible to false]
set Node_4_Colored [Visible to true]

```

HTML Codes for Node 1

```
<!DOCTYPE html>
<html>
<head>
<title>Live Chart from Firebase Realtime
Database using Chart.js</title>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-app.js"></script>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-database.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/chart.js"><
/script>
</head>
<body>
<center>
</center>
<h1><center>Node 1 Sensor
Reading</center></h1>
<h1><center>Tomato</center></h1>
<canvas
id="AmbientTempN1"></canvas>
<h1></h1>
<canvas id="HumidityN1"></canvas>
<h1></h1>
<canvas
id="LightIntensityN1"></canvas>
<h1></h1>
<canvas
id="ElectricalConductivityN1"></canvas>
<h1></h1>
<canvas id="SoilMoistureN1"></canvas>
<h1></h1>
<canvas id="pHN1"></canvas>
<h1></h1>
<canvas id="SoilTempN1"></canvas>
<h1></h1>
<canvas id="NitrogenN1"></canvas>
<h1></h1>
<canvas id="PhosphorusN1"></canvas>
<h1></h1>
```

```
<canvas id="PotassiumN1"></canvas>
<h1></h1>
<script>
// Set up Firebase
var firebaseConfig = {
  apiKey:
  "AIzaSyAICG1t7c6dlPUx2noP_qzkQwCd
  QGJc8A",
  authDomain:
  "monitoringapplication-
  5ce24.firebaseio.com",
  databaseURL:
  "https://monitoringapplication-5ce24-
  default-rtdb.firebaseio.com",
  projectId: "monitoringapplication-
  5ce24",
  storageBucket:
  "monitoringapplication-
  5ce24.appspot.com",
  messagingSenderId:
  "576472957352",
  appId:
  "1:576472957352:web:9a63b5052b4dc8fe6
  bdd9d",
  measurementId: "G-1F044JHVSF"
};
firebase.initializeApp(firebaseConfig);
var database = firebase.database();

// Set up Chart.js
var ctx1 = document.getElementById("AmbientTempN
1").getContext("2d");
var AmbientTempN1 = new Chart(ctx1, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Ambient Temperature", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', // Chart color
      }
    ]
  }
});
```

```

        borderColor: '#22797F', // Chart
border color
        borderWidth: 1, // Chart border
width

    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var AmbientTempRef = database.ref('Node 1 - Ambient
Temperature');
AmbientTempRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    AmbientTempN1.data.labels = labels;
    AmbientTempN1.data.datasets[0].data
= values;
    AmbientTempN1.update();
});

var ctx2 = document.getElementById("HumidityN1").getContext("2d");
var HumidityN1 = new Chart(ctx2, {
    type: "line",
    data: {
        labels: [], // X-axis labels
datasets: [
    {
        label: "Humidity", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', // Chart color
        borderColor: '#DFAF67', // Chart border color
        borderWidth: 1, // Chart border width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var humidityRef = database.ref('Node 1 - Humidity');
humidityRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    HumidityN1.data.labels = labels;
    HumidityN1.data.datasets[0].data
= values;
    HumidityN1.update();
});

// Set up Chart.js

```

```

var ctx3 = document.getElementById("LightIntensityN1").getContext("2d");
var LightIntensityN1 = new Chart(ctx3, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Light Intensity", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#D65B3F', // Chart color
        borderColor: '#D65B3F', // Chart border color
        borderWidth: 1, // Chart border width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});

// Set up listener for changes in Firebase Realtime Database data
var lightIntensityRef = database.ref('Node 1 - Light Intensity');
lightIntensityRef.on("value", function(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  LightIntensityN1.data.labels = labels;
});

```

```

LightIntensityN1.data.datasets[0].data = values;
LightIntensityN1.update();
});

var ctx4 = document.getElementById("ElectricalConductivityN1").getContext("2d");
var ElectricalConductivityN1 = new Chart(ctx4, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Electrical Conductivity", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#634082', // Chart color
        borderColor: '#634082', // Chart border color
        borderWidth: 1, // Chart border width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});

// Set up listener for changes in Firebase Realtime Database data
var ElectricalConductivityRef = database.ref('Node 1 - Electrical Conductivity');
ElectricalConductivityRef.on("value", function(snapshot) {
  var data = snapshot.val();
  var labels = [];

```

```

var values = [];

// Convert Firebase Realtime Database
// data to Chart.js format
for (var key in data) {
  labels.push(key);
  values.push(data[key]);
}

// Update Chart.js chart with new data
ElectricalConductivityN1.data.labels =
labels;

ElectricalConductivityN1.data.datasets[0].da
ta = values;
ElectricalConductivityN1.update();
});

var ctx5 = document.getElementById("SoilMoistureN1")
.getContext("2d");
var SoilMoistureN1 = new Chart(ctx5, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Moisture", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', // Chart color
        borderColor: '#22797F', // Chart border color
        borderWidth: 1, // Chart border width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase
// Realtime Database data
var SoilMoistureRef = database.ref('Node 1 - Moisture');
SoilMoistureRef.on("value", function(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilMoistureN1.data.labels = labels;
  SoilMoistureN1.data.datasets[0].data =
values;
  SoilMoistureN1.update();
});

var ctx6 = document.getElementById("pH1").getCon
text("2d");
var pH1 = new Chart(ctx6, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "pH", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', // Chart color
        borderColor: '#DFAF67', // Chart border color
        borderWidth: 1, // Chart border width
      },
    ],
  },
});

```

```

options: {
  responsive: true,
  maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var pHRef = database.ref('Node 1 - pH
Level');
pHRef.on("value", function (snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  pHN1.data.labels = labels;
  pHN1.data.datasets[0].data = values;
  pHN1.update();
});

var ctx7 = document.getElementById("SoilTempN1").getContext("2d");
var SoilTempN1 = new Chart(ctx7, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Temperature", // Chart
        title
        data: [], // Y-axis values
        backgroundColor: '#D65B3F', // Chart
        Chart color
        borderColor: '#D65B3F', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  }
});

// Set up listener for changes in Firebase
Realtime Database data
var SoilTempRef = database.ref('Node 1 -
Temperature');
SoilTempRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilTempN1.data.labels = labels;
  SoilTempN1.data.datasets[0].data =
values;
  SoilTempN1.update();
});

var ctx8 = document.getElementById("NitrogenN1").g
etContext("2d");
var NitrogenN1 = new Chart(ctx8, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Nitrogen", // Chart title
        data: [] // Y-axis values
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  }
});

```

```

        backgroundColor: '#634082', //
Chart color
        borderColor: '#634082', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var NitrogenRef = database.ref('Node 1 -
Nitrogen');
NitrogenRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    NitrogenN1.data.labels = labels;
    NitrogenN1.data.datasets[0].data =
values;
    NitrogenN1.update();
});

var ctx9 =
document.getElementById("PhosphorusN1")
.getContext("2d");
var PhosphorusN1 = new Chart(ctx9, {
    type: "line",
    data: {
        labels: [], // X-axis labels
        datasets: [
            {
                label: "Phosphorus", // Chart title
                data: [], // Y-axis values
                backgroundColor: '#22797F', //
Chart color
                borderColor: '#22797F', // Chart
border color
                borderWidth: 1, // Chart border
width
            },
            ],
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
            },
        });
// Set up listener for changes in Firebase
Realtime Database data
var PhosphorusRef = database.ref('Node
1 - Phosphorus');
PhosphorusRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    PhosphorusN1.data.labels = labels;
    PhosphorusN1.data.datasets[0].data =
values;
    PhosphorusN1.update();
});

```

```

var ctx10 = document.getElementById("PotassiumN1").getContext("2d");
var PotassiumN1 = new Chart(ctx10, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Potassium", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', // Chart color
        borderColor: '#DFAF67', // Chart border color
        borderWidth: 1, // Chart border width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase Realtime Database data
var PotassiumRef = database.ref('Node 1 - Potassium');
PotassiumRef.on("value", function(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  PotassiumN1.data.labels = labels;
  PotassiumN1.data.datasets[0].data = values;
  PotassiumN1.update();
});
</script>
</body>
</html>

```

HTML Codes for Node 2

```

<!DOCTYPE html>
<html>
<head>
<title>Live Chart from Firebase Realtime
Database using Chart.js</title>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-app.js"></script>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-database.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/chart.js"><
/script>
</head>
<body>
<center>
</center>
<h1><center>Node 2 Sensor
Reading</center></h1>
<h1><center>Pechay</center></h1>
<canvas
id="AmbientTempN2"></canvas>
<h1></h1>
<canvas id="HumidityN2"></canvas>
<h1></h1>
<canvas
id="LightIntensityN2"></canvas>
<h1></h1>
<canvas
id="ElectricalConductivityN2"></canvas>
<h1></h1>
<canvas id="SoilMoistureN2"></canvas>
<h1></h1>
<canvas id="pHN2"></canvas>
<h1></h1>
<canvas id="SoilTempN2"></canvas>
<h1></h1>
<canvas id="NitrogenN2"></canvas>
<h1></h1>
<canvas id="PhosphorusN2"></canvas>
<h1></h1>

```

```

<canvas id="PotassiumN2"></canvas>
<h1></h1>
<script>
// Set up Firebase
var firebaseConfig = {
  apiKey:
  "AIzaSyAICG1t7c6dlPUx2noP_qzkQwCd
  QGJc8A",
  authDomain:
  "monitoringapplication-
  5ce24.firebaseio.com",
  databaseURL:
  "https://monitoringapplication-5ce24-
  default-rtdb.firebaseio.com",
  projectId: "monitoringapplication-
  5ce24",
  storageBucket:
  "monitoringapplication-5ce24.appspot.com",
  messagingSenderId:
  "576472957352",
  appId:
  "1:576472957352:web:9a63b5052b4dc8fe6
  bdd9d",
  measurementId: "G-1F044JHVSF"
};
firebase.initializeApp(firebaseConfig);
var database = firebase.database();

// Set up Chart.js
var ctx1 =
document.getElementById("AmbientTempN
2").getContext("2d");
var AmbientTempN2 = new Chart(ctx1,
{
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Ambient Temperature", //
        Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', //
        Chart color
      }
    ]
  }
});

```

```

        borderColor: '#22797F', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var AmbientTempRef =
database.ref('Node 2 - Ambient
Temperature');
AmbientTempRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    AmbientTempN2.data.labels = labels;
    AmbientTempN2.data.datasets[0].data
= values;
    AmbientTempN2.update();
});

var ctx2 =
document.getElementById("HumidityN2").getContext("2d");
var HumidityN2 = new Chart(ctx2, {
    type: "line",
    data: {
        labels: [], // X-axis labels
datasets: [
    {
        label: "Humidity", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
Chart color
        borderColor: '#DFAF67', // Chart
border color
        borderWidth: 1, // Chart border width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var humidityRef = database.ref('Node 2 -
Humidity');
humidityRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    HumidityN2.data.labels = labels;
    HumidityN2.data.datasets[0].data =
values;
    HumidityN2.update();
};

// Set up Chart.js

```

```

var ctx3 =
document.getElementById("LightIntensityN
2").getContext("2d");
var LightIntensityN2 = new Chart(ctx3, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Light Intensity", // Chart title
data: [], // Y-axis values
backgroundColor: '#D65B3F', // Chart
color
borderColor: '#D65B3F', // Chart
border color
borderWidth: 1, // Chart border width
},
],
},
options: {
responsive: true,
maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var lightIntensityRef = database.ref('Node
2 - Light Intensity');
lightIntensityRef.on("value", function
(snapshot) {
var data = snapshot.val();
var labels = [];
var values = [];

// Convert Firebase Realtime Database
data to Chart.js format
for (var key in data) {
labels.push(key);
values.push(data[key]);
}

// Update Chart.js chart with new data
LightIntensityN2.data.labels = labels;
}

```

```

LightIntensityN2.data.datasets[0].data =
values;
LightIntensityN2.update();
});

var ctx4 =
document.getElementById("ElectricalCondu
ctivityN2").getContext("2d");
var ElectricalConductivityN2 = new
Chart(ctx4, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Electrical Conductivity", // Chart
title
data: [], // Y-axis values
backgroundColor: '#634082', // Chart
color
borderColor: '#634082', // Chart
border color
borderWidth: 1, // Chart border
width
},
],
},
options: {
responsive: true,
maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var ElectricalConductivityRef =
database.ref('Node 2 - Electrical
Conductivity');
ElectricalConductivityRef.on("value",
function (snapshot) {
var data = snapshot.val();
var labels = [];

```

```

var values = [];

// Convert Firebase Realtime Database
// data to Chart.js format
for (var key in data) {
  labels.push(key);
  values.push(data[key]);
}

// Update Chart.js chart with new data
ElectricalConductivityN2.data.labels =
labels;

ElectricalConductivityN2.data.datasets[0].da-
ta = values;
ElectricalConductivityN2.update();
});

var ctx5 =
document.getElementById("SoilMoistureN2")
.getContext("2d");
var SoilMoistureN2 = new Chart(ctx5, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Moisture", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', //
        Chart color
        borderColor: '#22797F', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase
// Realtime Database data
var SoilMoistureRef =
database.ref('Node 2 - Moisture');
SoilMoistureRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilMoistureN2.data.labels = labels;
  SoilMoistureN2.data.datasets[0].data =
values;
  SoilMoistureN2.update();
});

var ctx6 =
document.getElementById("pH2").getCon-
text("2d");
var pH2 = new Chart(ctx6, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "pH", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
        Chart color
        borderColor: '#DFAF67', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
    ],
  },
});

```

```

options: {
  responsive: true,
  maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var pHRef = database.ref('Node 2 - pH
Level');
pHRef.on("value", function (snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  pHN2.data.labels = labels;
  pHN2.data.datasets[0].data = values;
  pHN2.update();
}

var ctx7 =
document.getElementById("SoilTempN2").getContext("2d");
var SoilTempN2 = new Chart(ctx7, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Temperature", // Chart
        title
        data: [], // Y-axis values
        backgroundColor: '#D65B3F', //
        Chart color
        borderColor: '#D65B3F', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});

// Set up listener for changes in Firebase
Realtime Database data
var SoilTempRef = database.ref('Node 2
- Temperature');
SoilTempRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilTempN2.data.labels = labels;
  SoilTempN2.data.datasets[0].data =
values;
  SoilTempN2.update();
}

var ctx8 =
document.getElementById("NitrogenN2").g
etContext("2d");
var NitrogenN2 = new Chart(ctx8, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Nitrogen", // Chart title
        data: [] // Y-axis values
      }
    ]
  },
});

```

```

        backgroundColor: '#634082', //
Chart color
        borderColor: '#634082', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var NitrogenRef = database.ref('Node 2 -
Nitrogen');
NitrogenRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    NitrogenN2.data.labels = labels;
    NitrogenN2.data.datasets[0].data =
values;
    NitrogenN2.update();
});

var ctx9 =
document.getElementById("PhosphorusN2")
.getContext("2d");
var PhosphorusN2 = new Chart(ctx9, {
    type: "line",
    data: {
        labels: [], // X-axis labels
        datasets: [
            {
                label: "Phosphorus", // Chart title
                data: [], // Y-axis values
                backgroundColor: '#22797F', //
Chart color
                borderColor: '#22797F', // Chart
border color
                borderWidth: 1, // Chart border
width
            },
            ],
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
            },
        });
// Set up listener for changes in Firebase
Realtime Database data
var PhosphorusRef = database.ref('Node
2 - Phosphorus');
PhosphorusRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    PhosphorusN2.data.labels = labels;
    PhosphorusN2.data.datasets[0].data =
values;
    PhosphorusN2.update();
});

```

```

var ctx10 =
document.getElementById("PotassiumN2").getContext("2d");
var PotassiumN2 = new Chart(ctx10, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Potassium", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
        Chart color
        borderColor: '#DFAF67', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
      ],
    },
    options: {
      responsive: true,
      maintainAspectRatio: true,
    },
  });
// Set up listener for changes in Firebase
Realtime Database data
var PotassiumRef = database.ref('Node 2
- Potassium');
PotassiumRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  PotassiumN2.data.labels = labels;
  PotassiumN2.data.datasets[0].data =
  values;
  PotassiumN2.update();
});
</script>
</body>
</html>

```

HTML Codes for Node 3

```
<!DOCTYPE html>
<html>
<head>
<title>Live Chart from Firebase Realtime
Database using Chart.js</title>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-app.js"></script>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-database.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/chart.js"><
/script>

</head>
<body>
<center>
</center>

<h1><center>Node 1 Sensor
Reading</center></h1>
<h1><center>Pepper</center></h1>
<canvas
id="AmbientTempN3"></canvas>
<h1></h1>
<canvas id="HumidityN3"></canvas>
<h1></h1>
<canvas
id="LightIntensityN3"></canvas>
<h1></h1>
<canvas
id="ElectricalConductivityN3"></canvas>
<h1></h1>
<canvas id="SoilMoistureN3"></canvas>
<h1></h1>
<canvas id="pHN3"></canvas>
<h1></h1>
<canvas id="SoilTempN3"></canvas>
<h1></h1>
<canvas id="NitrogenN3"></canvas>
<h1></h1>
```

```
<canvas id="PhosphorusN3"></canvas>
<h1></h1>
<canvas id="PotassiumN3"></canvas>
<h1></h1>
<script>
// Set up Firebase
var firebaseConfig = {
  apiKey:
  "AIzaSyAICG1t7c6dlPUx2noP_qzkQwCd
  QGJc8A",
  authDomain:
  "monitoringapplication-
  5ce24.firebaseio.com",
  databaseURL:
  "https://monitoringapplication-5ce24-
  default-rtdb.firebaseio.com",
  projectId: "monitoringapplication-
  5ce24",
  storageBucket:
  "monitoringapplication-5ce24.appspot.com",
  messagingSenderId:
  "576472957352",
  appId:
  "1:576472957352:web:9a63b5052b4dc8fe6
  bdd9d",
  measurementId: "G-1F044JHVSF"
};
firebase.initializeApp(firebaseConfig);
var database = firebase.database();

// Set up Chart.js
var ctx1 =
document.getElementById("AmbientTempN
3").getContext("2d");
var AmbientTempN3 = new Chart(ctx1,
{
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Ambient Temperature", //
        Chart title
        data: [], // Y-axis values
      }
    ]
  }
});
```

```

        backgroundColor: '#22797F', //
Chart color
        borderColor: '#22797F', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var AmbientTempRef =
database.ref('Node 3 - Ambient
Temperature');
AmbientTempRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    AmbientTempN3.data.labels = labels;
    AmbientTempN3.data.datasets[0].data
= values;
    AmbientTempN3.update();
});

var ctx2 =
document.getElementById("HumidityN3").getContext("2d");
var HumidityN3 = new Chart(ctx2, {
    type: "line",
    data: {
        labels: [], // X-axis labels
        datasets: [
            {
                label: "Humidity", // Chart title
                data: [], // Y-axis values
                backgroundColor: '#DFAF67', //
Chart color
                borderColor: '#DFAF67', // Chart
border color
                borderWidth: 1, // Chart border width
            },
        ],
    },
    options: {
        responsive: true,
        maintainAspectRatio: true,
    },
});
// Set up listener for changes in Firebase
Realtime Database data
var humidityRef = database.ref('Node 3 -
Humidity');
humidityRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    HumidityN3.data.labels = labels;
    HumidityN3.data.datasets[0].data =
values;
    HumidityN3.update();
});

// Set up Chart.js

```

```

var ctx3 =
document.getElementById("LightIntensityN
3").getContext("2d");
var LightIntensityN3 = new Chart(ctx3, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Light Intensity", // Chart title
data: [], // Y-axis values
backgroundColor: '#D65B3F', // Chart
color
borderColor: '#D65B3F', // Chart
border color
borderWidth: 1, // Chart border width
},
],
},
options: {
responsive: true,
maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var lightIntensityRef = database.ref('Node
3 - Light Intensity');
lightIntensityRef.on("value", function
(snapshot) {
var data = snapshot.val();
var labels = [];
var values = [];

// Convert Firebase Realtime Database
data to Chart.js format
for (var key in data) {
labels.push(key);
values.push(data[key]);
}

// Update Chart.js chart with new data
LightIntensityN3.data.labels = labels;
}

```

```

LightIntensityN3.data.datasets[0].data =
values;
LightIntensityN3.update();
});

var ctx4 =
document.getElementById("ElectricalCondu
ctivityN3").getContext("2d");
var ElectricalConductivityN3 = new
Chart(ctx4, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Electrical Conductivity", // Chart
title
data: [], // Y-axis values
backgroundColor: '#634082', // Chart
color
borderColor: '#634082', // Chart
border color
borderWidth: 1, // Chart border
width
},
],
},
options: {
responsive: true,
maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var ElectricalConductivityRef =
database.ref('Node 3 - Electrical
Conductivity');
ElectricalConductivityRef.on("value",
function (snapshot) {
var data = snapshot.val();
var labels = [];

```

```

var values = [];

// Convert Firebase Realtime Database
// data to Chart.js format
for (var key in data) {
  labels.push(key);
  values.push(data[key]);
}

// Update Chart.js chart with new data
ElectricalConductivityN3.data.labels =
labels;

ElectricalConductivityN3.data.datasets[0].da-
ta = values;
  ElectricalConductivityN3.update();
});

var ctx5 =
document.getElementById("SoilMoistureN3")
.getContext("2d");
var SoilMoistureN3 = new Chart(ctx5, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Moisture", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', //
        Chart color
        borderColor: '#22797F', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase
// Realtime Database data
var SoilMoistureRef =
database.ref('Node 3 - Moisture');
SoilMoistureRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilMoistureN3.data.labels = labels;
  SoilMoistureN3.data.datasets[0].data =
values;
  SoilMoistureN3.update();
});

var ctx6 =
document.getElementById("pHN3").getCon-
text("2d");
var pHN3 = new Chart(ctx6, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "pH", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
        Chart color
        borderColor: '#DFAF67', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
    ],
  },
});

```

```

options: {
  responsive: true,
  maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var pHRef = database.ref('Node 3 - pH
Level');
pHRef.on("value", function (snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  pHN3.data.labels = labels;
  pHN3.data.datasets[0].data = values;
  pHN3.update();
}

var ctx7 =
document.getElementById("SoilTempN3").getContext("2d");
var SoilTempN3 = new Chart(ctx7, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Temperature", // Chart
        title
        data: [], // Y-axis values
        backgroundColor: '#D65B3F', //
        Chart color
        borderColor: '#D65B3F', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      }
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});

// Set up listener for changes in Firebase
Realtime Database data
var SoilTempRef = database.ref('Node 3
- Temperature');
SoilTempRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilTempN3.data.labels = labels;
  SoilTempN3.data.datasets[0].data =
values;
  SoilTempN3.update();
}

var ctx8 =
document.getElementById("NitrogenN3").g
etContext("2d");
var NitrogenN3 = new Chart(ctx8, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Nitrogen", // Chart title
        data: [], // Y-axis values
      }
    ],
  },
});

```

```

        backgroundColor: '#634082', //
Chart color
        borderColor: '#634082', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase
Realtime Database data
var NitrogenRef = database.ref('Node 3 -
Nitrogen');
NitrogenRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    NitrogenN3.data.labels = labels;
    NitrogenN3.data.datasets[0].data =
values;
    NitrogenN3.update();
});

var ctx9 =
document.getElementById("PhosphorusN3")
.getContext("2d");
var PhosphorusN3 = new Chart(ctx9, {
    type: "line",
    data: {
        labels: [], // X-axis labels
        datasets: [
            {
                label: "Phosphorus", // Chart title
                data: [], // Y-axis values
                backgroundColor: '#22797F', //
Chart color
                borderColor: '#22797F', // Chart
border color
                borderWidth: 1, // Chart border
width
            },
            ],
            },
            options: {
                responsive: true,
                maintainAspectRatio: true,
            },
        });
// Set up listener for changes in Firebase
Realtime Database data
var PhosphorusRef = database.ref('Node
3 - Phosphorus');
PhosphorusRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    PhosphorusN3.data.labels = labels;
    PhosphorusN3.data.datasets[0].data =
values;
    PhosphorusN3.update();
});

```

```

var ctx10 =
document.getElementById("PotassiumN3").getContext("2d");
var PotassiumN3 = new Chart(ctx10, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Potassium", // Chart title
data: [], // Y-axis values
backgroundColor: '#DFAF67', //
Chart color
borderColor: '#DFAF67', // Chart border color
borderWidth: 1, // Chart border width
},
],
},
options: {
responsive: true,
maintainAspectRatio: true,
},
});
// Set up listener for changes in Firebase Realtime Database data
var PotassiumRef = database.ref('Node 3 - Potassium');
PotassiumRef.on("value", function(snapshot) {
var data = snapshot.val();
var labels = [];
var values = [];

// Convert Firebase Realtime Database data to Chart.js format
for (var key in data) {
labels.push(key);
values.push(data[key]);
}

// Update Chart.js chart with new data
PotassiumN3.data.labels = labels;
PotassiumN3.data.datasets[0].data = values;
PotassiumN3.update());
});
</script>
</body>
</html>

```

HTML Codes for Node 4

```

<!DOCTYPE html>
<html>
<head>
<title>Live Chart from Firebase Realtime
Database using Chart.js</title>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-app.js"></script>
<script
src="https://www.gstatic.com/firebasejs/8.6.
7.firebaseio-database.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/chart.js"><
/script>
</head>
<body>
<center>
</center>
<h1><center>Node 2 Sensor
Reading</center></h1>
<h1><center>Lettuce</center></h1>
<canvas
id="AmbientTempN4"></canvas>
<h1></h1>
<canvas id="HumidityN4"></canvas>
<h1></h1>
<canvas
id="LightIntensityN4"></canvas>
<h1></h1>
<canvas
id="ElectricalConductivityN4"></canvas>
<h1></h1>
<canvas id="SoilMoistureN4"></canvas>
<h1></h1>
<canvas id="pHN4"></canvas>
<h1></h1>
<canvas id="SoilTempN4"></canvas>
<h1></h1>
<canvas id="NitrogenN4"></canvas>
<h1></h1>
<canvas id="PhosphorusN4"></canvas>
<h1></h1>

```

```

<canvas id="PotassiumN4"></canvas>
<h1></h1>
<script>
// Set up Firebase
var firebaseConfig = {
  apiKey:
  "AIzaSyAICGl1t7c6dlPUx2noP_qzkQwCd
  QGJc8A",
  authDomain:
  "monitoringapplication-
  5ce24.firebaseio.com",
  databaseURL:
  "https://monitoringapplication-5ce24-
  default-rtdb.firebaseio.com",
  projectId: "monitoringapplication-
  5ce24",
  storageBucket:
  "monitoringapplication-5ce24.appspot.com",
  messagingSenderId:
  "576472957352",
  appId:
  "1:576472957352:web:9a63b5052b4dc8fe6
  bdd9d",
  measurementId: "G-1F044JHVSF"
};
firebase.initializeApp(firebaseConfig);
var database = firebase.database();

// Set up Chart.js
var ctx1 =
document.getElementById("AmbientTempN
4").getContext("2d");
var AmbientTempN4 = new Chart(ctx1,
{
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Ambient Temperature", //
        Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', //
        Chart color
      }
    ]
  }
});

```

```

        borderColor: '#22797F', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var ctx1 =
document.getElementById("AmbientTempN
4").getContext("2d");
var AmbientTempN4 = new Chart(ctx1, {
type: "line",
data: {
labels: [], // X-axis labels
datasets: [
{
label: "Ambient Temperature", // Chart
title
data: [], // Y-axis values
backgroundColor: '#22797F', // Chart
color
borderColor: '#22797F', // Chart border
color
borderWidth: 1, // Chart border width
},
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});
}

// Set up listener for changes in Firebase
Realtime Database data
var AmbientTempRef = database.ref('Node 4
- Ambient Temperature');

```

```

AmbientTempRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database data
    // to Chart.js format
    var keys = Object.keys(data).sort(); // Sort
    the keys

    for (var key in data) {
        var key = keys[i];
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    AmbientTempN4.data.labels = labels;
    AmbientTempN4.data.datasets[0].data =
values;
    AmbientTempN4.update();
});

// Set up listener for changes in Firebase
Realtime Database data
var humidityRef = database.ref('Node 4 -
Humidity');
humidityRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    HumidityN4.data.labels = labels;
    HumidityN4.data.datasets[0].data =
values;
    HumidityN4.update();
}

```

```

    });

    // Set up Chart.js
    var ctx3 =
        document.getElementById("LightIntensityN4").getContext("2d");
    var LightIntensityN4 = new Chart(ctx3, {
        type: "line",
        data: {
            labels: [], // X-axis labels
            datasets: [
                {
                    label: "Light Intensity", // Chart title
                    data: [], // Y-axis values
                    backgroundColor: '#D65B3F', // Chart color
                    borderColor: '#D65B3F', // Chart border color
                    borderWidth: 1, // Chart border width
                },
            ],
        },
        options: {
            responsive: true,
            maintainAspectRatio: true,
        },
    });

    // Set up listener for changes in Firebase Realtime Database data
    var lightIntensityRef = database.ref('Node 4 - Light Intensity');
    lightIntensityRef.on("value", function(snapshot) {
        var data = snapshot.val();
        var labels = [];
        var values = [];

        // Convert Firebase Realtime Database data to Chart.js format
        for (var key in data) {
            labels.push(key);
            values.push(data[key]);
        }
    });

    // Update Chart.js chart with new data
    LightIntensityN4.data.labels = labels;
    LightIntensityN4.data.datasets[0].data = values;
    LightIntensityN4.update();
}

var ctx4 =
document.getElementById("ElectricalConductivityN4").getContext("2d");
var ElectricalConductivityN4 = new Chart(ctx4, {
    type: "line",
    data: {
        labels: [], // X-axis labels
        datasets: [
            {
                label: "Electrical Conductivity", // Chart title
                data: [], // Y-axis values
                backgroundColor: '#634082', // Chart color
                borderColor: '#634082', // Chart border color
                borderWidth: 1, // Chart border width
            },
        ],
    },
    options: {
        responsive: true,
        maintainAspectRatio: true,
    },
});

// Set up listener for changes in Firebase Realtime Database data
var ElectricalConductivityRef = database.ref('Node 4 - Electrical Conductivity');
ElectricalConductivityRef.on("value", function(snapshot) {

```

```

var data = snapshot.val();
var labels = [];
var values = [];

// Convert Firebase Realtime Database
data to Chart.js format
for (var key in data) {
  labels.push(key);
  values.push(data[key]);
}

// Update Chart.js chart with new data
ElectricalConductivityN4.data.labels =
labels;

ElectricalConductivityN4.data.datasets[0].da
ta = values;
  ElectricalConductivityN4.update();
});

var ctx5 =
document.getElementById("SoilMoistureN4
").getContext("2d");
var SoilMoistureN4 = new Chart(ctx5, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Soil Moisture", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#22797F', //
Chart color
        borderColor: '#22797F', // Chart
border color
        borderWidth: 1, // Chart border
width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase
Realtime Database data
var SoilMoistureRef =
database.ref('Node 4 - Moisture');
SoilMoistureRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

// Convert Firebase Realtime Database
data to Chart.js format
for (var key in data) {
  labels.push(key);
  values.push(data[key]);
}

// Update Chart.js chart with new data
SoilMoistureN4.data.labels = labels;
SoilMoistureN4.data.datasets[0].data =
values;
  SoilMoistureN4.update();
});

var ctx6 =
document.getElementById("pHN4").getCon
text("2d");
var pHN4 = new Chart(ctx6, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "pH", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
Chart color
        borderColor: '#DFAF67', // Chart
border color
        borderWidth: 1, // Chart border
width
      },
    ],
  },
});

```

```

    },
    options: {
      responsive: true,
      maintainAspectRatio: true,
    },
  });

  // Set up listener for changes in Firebase
  // Realtime Database data
  var pHRef = database.ref('Node 4 - pH
Level');
  pHRef.on("value", function (snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    // data to Chart.js format
    for (var key in data) {
      labels.push(key);
      values.push(data[key]);
    }

    // Update Chart.js chart with new data
    pHN4.data.labels = labels;
    pHN4.data.datasets[0].data = values;
    pHN4.update();
  });

  var ctx7 =
  document.getElementById("SoilTempN4").getContext("2d");
  var SoilTempN4 = new Chart(ctx7, {
    type: "line",
    data: {
      labels: [], // X-axis labels
      datasets: [
        {
          label: "Soil Temperature", // Chart
          title
          data: [], // Y-axis values
          backgroundColor: '#D65B3F', // Chart
          Chart color
          borderColor: '#D65B3F', // Chart
          border color
        }
      ],
      options: {
        responsive: true,
        maintainAspectRatio: true,
      };
    }
  });

  // Set up listener for changes in Firebase
  // Realtime Database data
  var SoilTempRef = database.ref('Node 4
- Temperature');
  SoilTempRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  // data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  SoilTempN4.data.labels = labels;
  SoilTempN4.data.datasets[0].data =
values;
  SoilTempN4.update();
});

  var ctx8 =
  document.getElementById("NitrogenN4").getContext("2d");
  var NitrogenN4 = new Chart(ctx8, {
    type: "line",
    data: {
      labels: [], // X-axis labels
      datasets: [
        {
          label: "Nitrogen", // Chart title
        }
      ],
      options: {
        responsive: true,
        maintainAspectRatio: true,
      };
    }
  });

```

```

        data: [], // Y-axis values
        backgroundColor: '#634082', //
Chart color
        borderColor: '#634082', // Chart
border color
        borderWidth: 1, // Chart border
width
    },
],
},
options: {
    responsive: true,
    maintainAspectRatio: true,
},
});

// Set up listener for changes in Firebase
Realtime Database data
var NitrogenRef = database.ref('Node 4 -
Nitrogen');
NitrogenRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    NitrogenN4.data.labels = labels;
    NitrogenN4.data.datasets[0].data =
values;
    NitrogenN4.update();
});

var ctx9 =
document.getElementById("PhosphorusN4")
.getContext("2d");
var PhosphorusN4 = new Chart(ctx9, {
    type: "line",

```

```

        data: {
        labels: [], // X-axis labels
        datasets: [
        {
            label: "Phosphorus", // Chart title
            data: [], // Y-axis values
            backgroundColor: '#22797F', //
Chart color
            borderColor: '#22797F', // Chart
border color
            borderWidth: 1, // Chart border
width
        },
        ],
        },
        options: {
            responsive: true,
            maintainAspectRatio: true,
        },
    });

// Set up listener for changes in Firebase
Realtime Database data
var PhosphorusRef = database.ref('Node
4 - Phosphorus');
PhosphorusRef.on("value", function
(snapshot) {
    var data = snapshot.val();
    var labels = [];
    var values = [];

    // Convert Firebase Realtime Database
    data to Chart.js format
    for (var key in data) {
        labels.push(key);
        values.push(data[key]);
    }

    // Update Chart.js chart with new data
    PhosphorusN4.data.labels = labels;
    PhosphorusN4.data.datasets[0].data =
values;
    PhosphorusN4.update();
});

```

```

var ctx10 =
document.getElementById("PotassiumN4").getContext("2d");
var PotassiumN4 = new Chart(ctx10, {
  type: "line",
  data: {
    labels: [], // X-axis labels
    datasets: [
      {
        label: "Potassium", // Chart title
        data: [], // Y-axis values
        backgroundColor: '#DFAF67', //
        Chart color
        borderColor: '#DFAF67', // Chart
        border color
        borderWidth: 1, // Chart border
        width
      },
    ],
  },
  options: {
    responsive: true,
    maintainAspectRatio: true,
  },
});
// Set up listener for changes in Firebase
Realtime Database data
var PotassiumRef = database.ref('Node 4
- Potassium');
PotassiumRef.on("value", function
(snapshot) {
  var data = snapshot.val();
  var labels = [];
  var values = [];

  // Convert Firebase Realtime Database
  data to Chart.js format
  for (var key in data) {
    labels.push(key);
    values.push(data[key]);
  }

  // Update Chart.js chart with new data
  PotassiumN4.data.labels = labels;
  PotassiumN4.data.datasets[0].data =
  values;
  PotassiumN4.update();
});
</script>
</body>
</html>

```

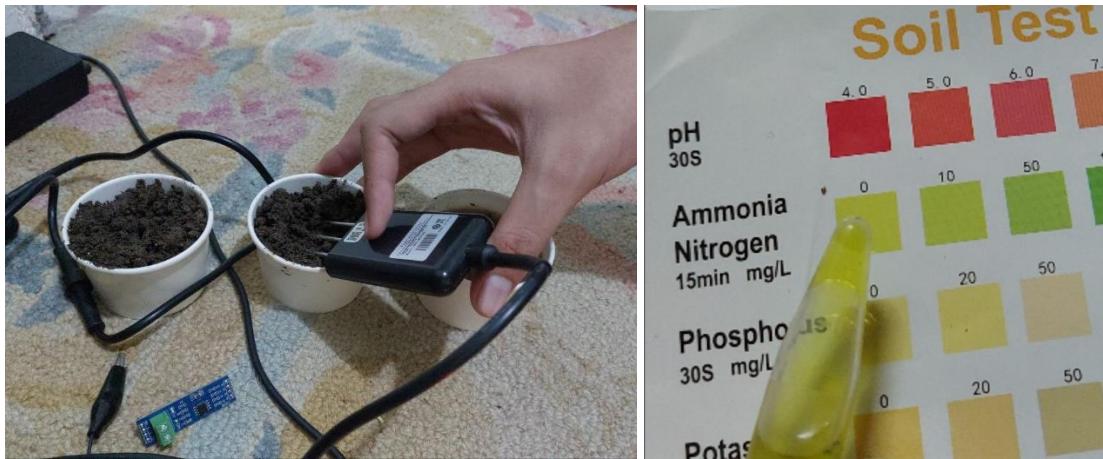
ANNEX IX

Progress Documentation

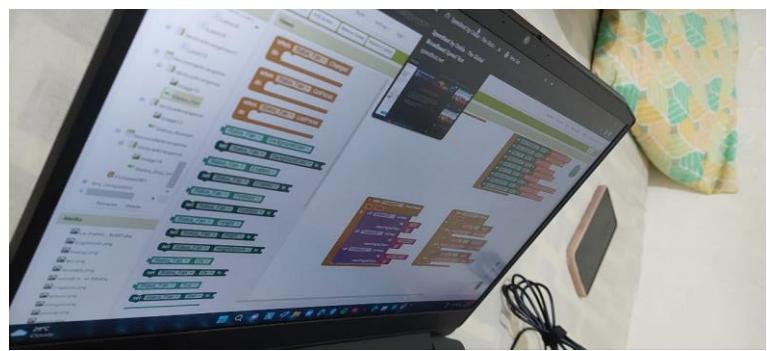


Coding and Integration of Sensors to a Node

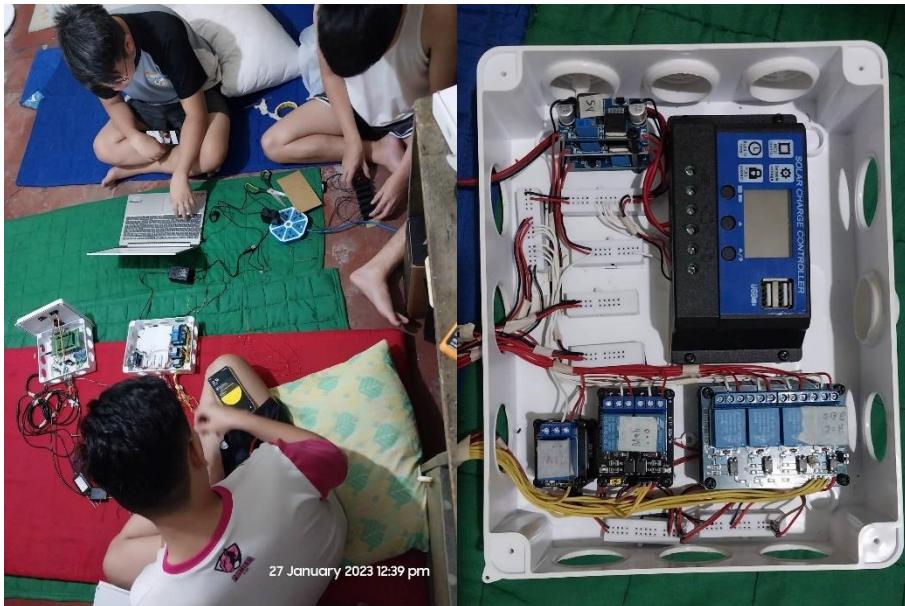




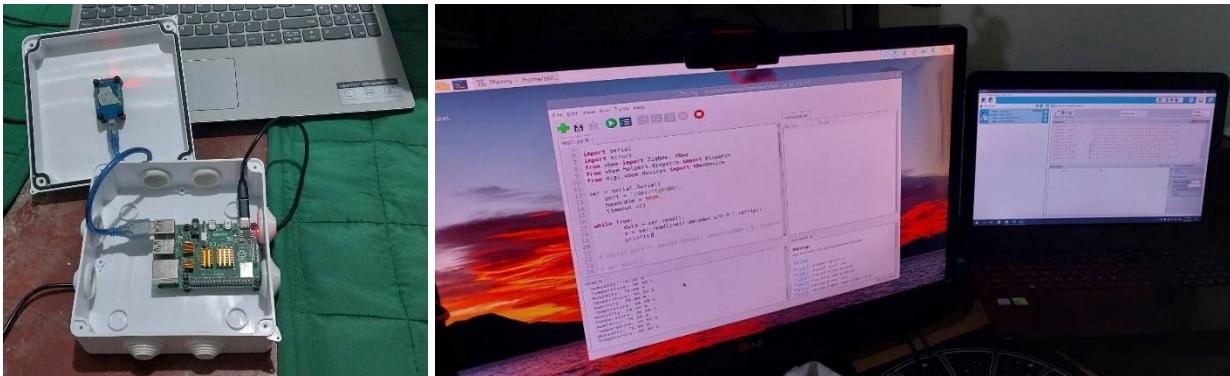
Calibration and Testing the Accuracy of Sensors



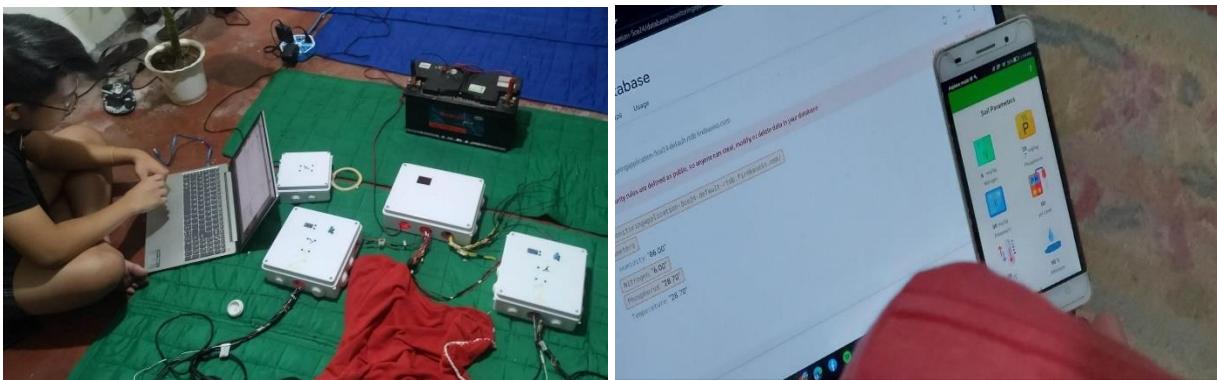
Monitoring Application Development



Development of the Power Management System



Developing and Configuring the Gateway



Testing the Overall System Before Deployment



The Deployment Area



Preparation of the Greenhouse for System Deployment



Installed Correction System in the Greenhouse



The Conventional (Right) and the Controlled (Right) Plants Setup



Gathering Data from Plants



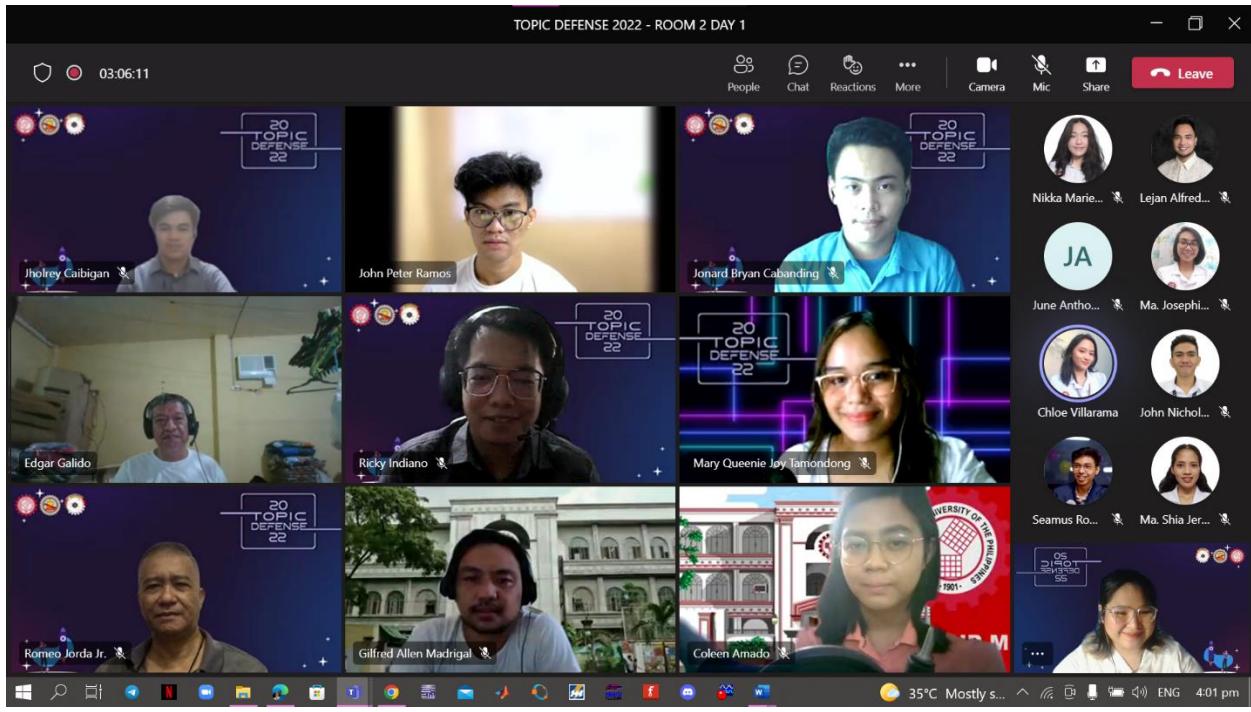
Harvested Pechays and Lettuces



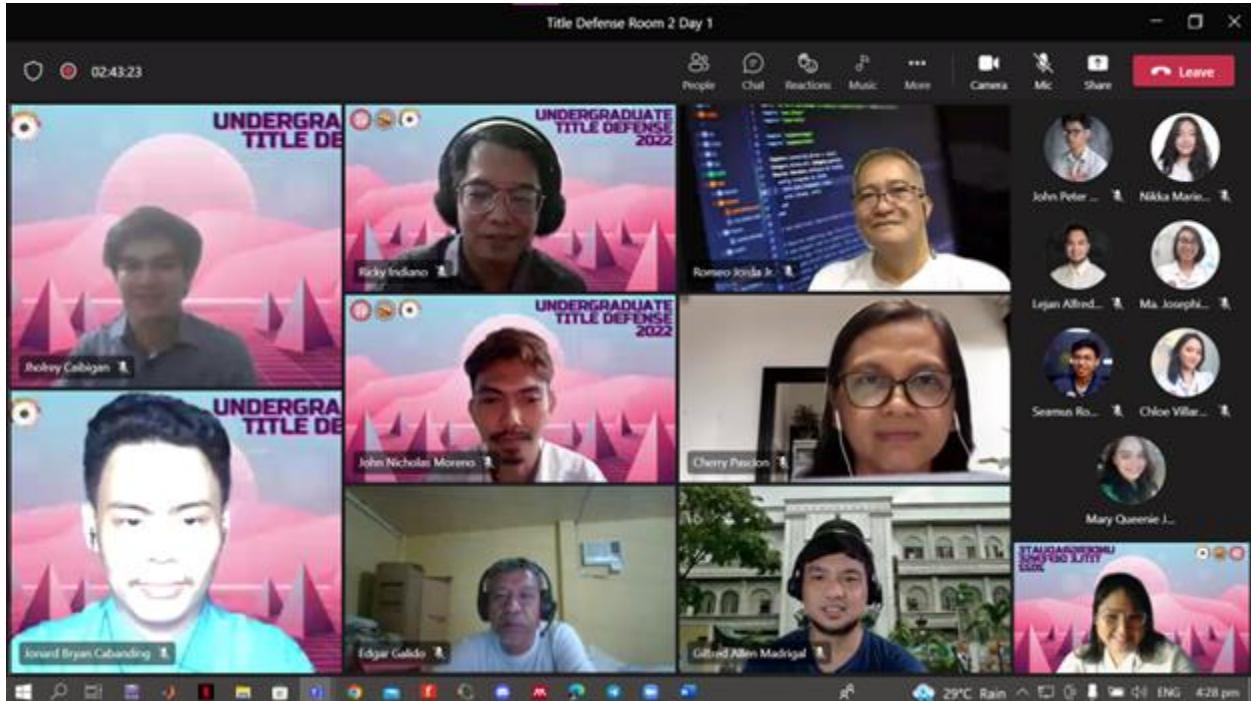
Site Visit with Thesis Adviser and Engr. Mark Melegrito

ANNEX X

Photos From Previous Defense



Topic Defense



Title Defense



Pre-final Defense



Progress Presentation



Appreciate 2023



Final Defense



Final Defense (with the Thesis Adviser)



Final Defense (with the Panelist)

ANNEX XI

Turnitin Results



Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author: Angelika Mae Virtucio
Assignment title: Assignment
Submission title: SOILNWAN
File name: SOILNWAN_MANUSCRIPT.pdf
File size: 10.18M
Page count: 324
Word count: 57,582
Character count: 308,956
Submission date: 30-Jun-2023 01:27AM (UTC-0500)
Submission ID: 2124654920



Copyright 2023 Turnitin. All rights reserved.

PRIMARY SOURCES

- | | | |
|---|--|------|
| 1 | Baihaqi Siregar, AB Azmi Nasution, Lukman Adlin, Ulfi Andayani, Fahmi Fahmi. "Soil Moisture Monitoring System using Wireless Sensor Network", Journal of Physics: Conference Series, 2018
Publication | 1 % |
| 2 | UK Madhura, P Akshay, Akshay J Bhattad, GS Nagaraja. "Soil Quality Management Using Wireless Sensor Network", 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), 2017
Publication | 1 % |
| 3 | A. Bhattacharya. "Physiological Processes in Plants Under Low Temperature Stress", Springer Science and Business Media LLC, 2022
Publication | <1 % |
| 4 | Dattatraya Shinde, Naseem Siddiqui. "IOT Based Environment change Monitoring & Controlling in Greenhouse using WSN", 2018 | <1 % |
-

**International Conference on Information ,
Communication, Engineering and Technology
(ICICET), 2018**

Publication

-
- 5 S.R. Jino Ramson, Walter D. Leon-Salas, Zachary Brecheisen, Erika J. Foster et al. "A Self-Powered, Real-Time, LoRaWAN IoT-based Soil Health Monitoring System", IEEE Internet of Things Journal, 2021 <1 %
Publication
- 6 "Internet of Things and Analytics for Agriculture, Volume 3", Springer Science and Business Media LLC, 2022 <1 %
Publication
- 7 "The Fourth Industrial Revolution and Beyond", Springer Science and Business Media LLC, 2023 <1 %
Publication
- 8 "International Conference on Mobile Computing and Sustainable Informatics", Springer Science and Business Media LLC, 2021 <1 %
Publication
- 9 Islamic Azad University-Isfahan Branch <1 %
Publication
- 10 Submitted to Universiti Teknikal Malaysia Melaka <1 %
Student Paper
-

11	repository.ub.ac.id Internet Source	<1 %
12	Laurence Alec M. Burce, Dionis A. Padilla, John Lawrence M. Nagayo. "Soil Quality Monitoring System using Low-Powered Wireless Sensor Network", 2022 14th International Conference on Computer and Automation Engineering (ICCAE), 2022 Publication	<1 %
13	"Intelligent Sustainable Systems", Springer Science and Business Media LLC, 2022 Publication	<1 %
14	Hanoi National University of Education Publication	<1 %
15	Sudha M, Ravisankar Kandasamy. "Artificial Intelligence-Based Intravenous Fluid Monitoring and Control System for Health Care Using WSN", Research Square Platform LLC, 2022 Publication	<1 %
16	Ercan Avşar, Md. Najmul Mowla. "Wireless communication protocols in smart agriculture: A review on applications, challenges and future trends", Ad Hoc Networks, 2022 Publication	<1 %
17	mro.massey.ac.nz	

Internet Source

<1 %

18 Journal of Facilities Management, Volume 11,
Issue 1 (2013-02-02) <1 %

Publication

19 S. R. Jino Ramson, Walter D. Leon-Salas,
Zachary Brecheisen, Erika J. Foster et al. "A
Self-Powered, Real-Time, LoRaWAN IoT-Based
Soil Health Monitoring System", IEEE Internet
of Things Journal, 2021 <1 %

Publication

Exclude quotes On

Exclude matches Off

Exclude bibliography On

ANNEX XII

Certificate of Proofreading

ANNEX XIII

User's Manual

To change the password

Your Best Farm Companion!

1. Hover to the upper left side of the main screen to reveal the sidebar and choose the Account Management.

2. Fill up the required credentials then choose apply to change the default password.
3. Note that you can't change the default username.
4. You will now be redirected to the login menu. Make sure to enter the new password to pass through.

Account Management ::



On this page, you can edit the user's password to ensure application's security

Username :	Ssologist
Old Password :	<input type="text"/>
New Password :	<input type="text"/>
Confirm Password :	<input type="text"/>

SOIL-N-WAN USER'S MANUAL

WSN - Based Smart Farming Through
Soil Health and Ambiance Condition
Analysis with Monitoring and
Correction System

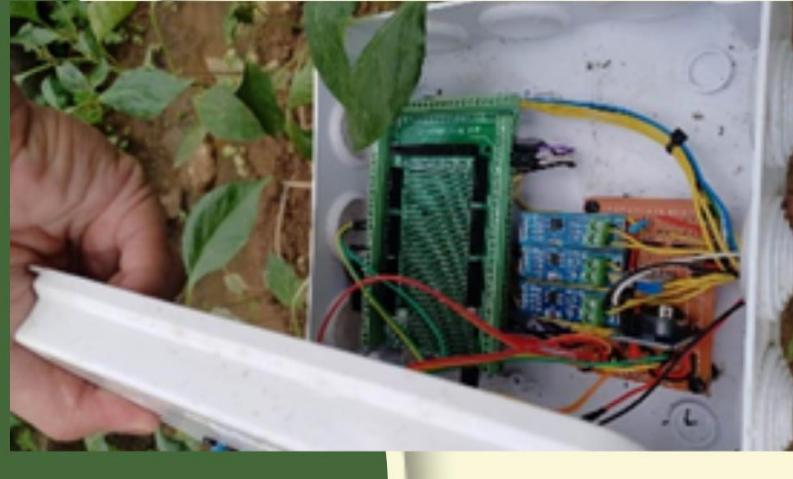
About Us

We are the SOILOGISTS, and we proposed a Wireless Sensor Network system, which will revolutionize farming through the integration of sensors, IoT, automated systems for drip fertilization and irrigation, and ambient condition correction to assist farmers in improving soil quality and provide alternative farming solutions.

Questions? Ask us

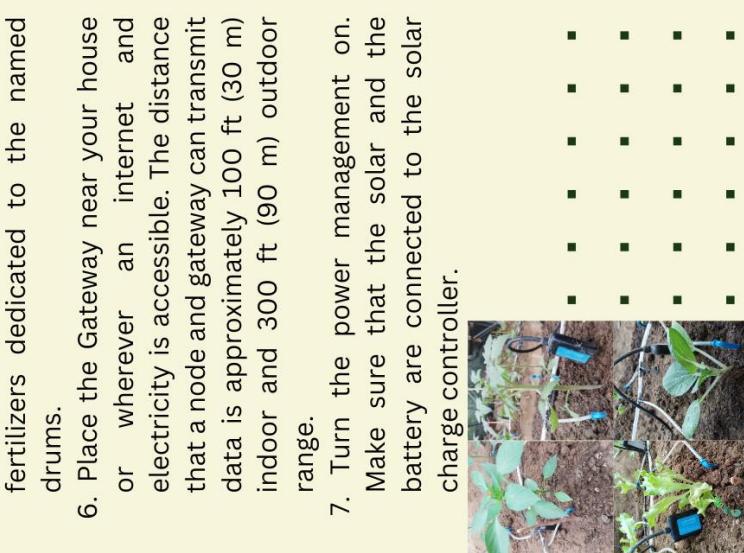
09352458158

ssologist@gmail.com



How to Setup the System

1. Place the Node near the soil.
2. Pin the sensors to the soil 7 cm apart from the plants to ensure that the pins do not touch the roots of the plants. Place down the sensors in appropriate distance with one another to maximize the usage of the node.
3. Sensors should be pinned all the way to the soil.
4. Connect the plastic tubes to the drum and pin the drip arrow to the soil 2.5 cm apart from the plants.
5. Fill the drums with water and mix the fertilizers dedicated to the named drums.
6. Place the Gateway near your house or wherever an internet and electricity is accessible. The distance that a node and gateway can transmit data is approximately 100 ft (30 m) indoor and 300 ft (90 m) outdoor range.
7. Turn the power management on. Make sure that the solar and the battery are connected to the solar charge controller.



Guide to proper ratio of fertilizer and water

1. Note that the container is 18 liters of capacity.
2. For Nitrogen - 1 tbsp per 4 liters of water, thus add 4.5 tbsp if empty
3. For Phosphorus - 1 tbsp per liter of water, thus add 18 tbsp if empty
4. For Potassium - 3 tbsp per 4 liters of water, thus add 13.5 tbsp if empty
5. For Lime - 1 tbsp per 5 liters of water, thus add 3.6 tbsp if empty
6. Sulfur - 1 tbsp per liter of water, thus add 18 tbsp if empty



Reminders!

1. Fill the drum with water and fertilizers every week in case the drum becomes empty.
2. Change and charge the battery if there is no sunlight throughout the day.
3. Checking of water lines in case of leaks
4. Checking of wire connections when some functions are not working.



Exploring the SoilNWAN App

1. Scan the QR code to download the SoilNWAN App



Note: this app is only available for android devices.

2. Install the application on your android device.
3. Use the default username: Soilologist and password: 1234 to get through the monitoring app.

4. You can now monitor your nodes remotely.
5. Select the plant's/node's tab where the sensor nodes are placed.

To Check the Graphs of the Sensor Data

1. Hover to the upper left side of the main screen to reveal the sidebar and choose the Graphical View.
2. Then, select the plant/node where the sensor nodes are placed.
3. You are now redirected to the graphs of sensor data detected by your selected node.
4. Press back to return to the application and explore graphs of the other node.
5. To return to the main monitoring window, just press back.

ANNEX XIV

Manual for Duplication of Prototype



TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES
Ayala Blvd, Ermita, Manila
COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT



electronics@tup.edu.ph

**WSN-based Smart Farming through Soil Health and Ambience Condition Analysis
with Monitoring and Correction System
(MANUAL FOR DUPLICATION OF PROTOTYPE)**

by

Cabanding, Jonard Bryan S.

Caibigan, Jholrey C.

Indiano, Ricky M.

Moreno, John Nicholas B.

Virtucio, Angelika Mae C.

August 2023

I. INTRODUCTION

Welcome to the manual for duplicating the SOIL-N-WAN system. This comprehensive guide is designed to provide you with step-by-step instructions on how to successfully recreate the prototype of our innovative SOIL-N-WAN system. Whether you are an engineer, developer, or enthusiast looking to replicate our technology, this manual will serve as your resource.

The purpose of this manual is to enable you to duplicate the prototype of the BLANK system with accuracy and precision. By following the instructions outlined herein, you will gain a deep understanding of the inner workings of our system and be able to replicate it effectively. Duplicating the prototype will allow you to explore its functionality, test its performance, and gain hands-on experience with the SOIL-N-WAN system.

Please note that this manual assumes a basic understanding of electronics and related concepts. If you are new to the field, it is recommended to familiarize yourself with the fundamentals before proceeding. Additionally, if you encounter any difficulties or have questions along the way, our support team is available to provide assistance and clarification.

In order to successfully duplicate the system, the following are skills required for the team:

- Circuit Analysis and Hardware Troubleshoot
- Python Programming
- Arduino Programming
- Firebase Database Knowledge
- Basic MIT App Inventor Debug
- MATLAB Simulink and Fuzzy Logic Implementation Knowledge

NOTE: In terms of changing a plant to be used, the proponents must be consulted in this case since another code must be created as there will be a change in parameter requirement. The following plants are used by the proponents thus the only codes suitable for the system.

- TOMATO
- PEPPER
- PECHAY
- LETTUCE

II. MATERIALS

The following figures provides an overview of the materials required in order to duplicate the system. All the materials listed is required to create the system.

For 2 Sensor Nodes:

8	Max485
2	DHT11
3	BH175
2	DS18B20 / Temperature Sensor
2	Soil Moisture
2	pH Sensor
2	Electrical Conductivity

- 2 NPK Sensor
- 2 Arduino Mega
- 2 DS3231 RTC module
- 2 CR2032
- 2 Xbee S2C
- 2 Xbee USB Adapter

For the Gateway:

- 1 Raspberry Pi 4
- 1 Xbee S2C
- 1 Xbee USB Adapter

For the Correction System:

- 12 ARD Pump
- 1 100 pcs 3/5mm Threaded Connector Irrigation System
- 1 100pcs Drip Irrigation Tee Fittings (4/7 mm Tee Pipe)
- 2 24V Humidifier
- 1 2 Channel Relay
- 1 4 Channel Relay 5V
- 20m Aquarium Hose
- 2 Floating Atomizer for Humidifier
- 1 Phosphate Organic Fertilizer (0-22-0 + Ca 30) 1kg
- 1 Urea 46-0-0 Granular Fertilizer 1 kg (Sources of Nitrogen)

For the Power Management System:

- 1 Boost Converter
- 2 Buck Converter
- 2 Universal PCB (Small)
- 1 100 W Solar Panel
- 1 30 A Solar Charger Controller
- 2 Battery Clamp Terminal
- 2 9V Battery Clip

Calibration Materials:

- 1 12V Battery
- 2 Triple A Battery
- 1 Junction Box 150 x 150 x70mm
- 2 Junction Box 200 x 200 x 80mm
- 1 Junction Box 255 x 200 x 80 mm
- 1 Breadboard
- 2 Terminal Block
- 1 15m Solid Wires
- 10 5pcs Female to Male Jumper Wire

III. TOOLS AND EQUIPMENT

The following list provides the necessary tools required and their functionality in order to duplicate the system.

- Soldering Iron and Lead
- Shrinkable Tubes
- 5mm Phillips Screwdriver
- PCB Standoff Spacers
- Scissors
- Electrical Tape
- Hacksaw
- Hand-held corded Electric Drill & Small drill bits
- Waterproof Silicone Sealant
- Epoxy (All Purpose)
- Copper Wire (22 gauge Solid)
 - Length depends on the area you want to install the system

IV. DUPLICATION PROCESS

The following steps provide a brief overview on how to duplicate the system. The steps will be divided into parts: Sensor Nodes Development, Gateway Development and WSN Connection, Correction System Integration, and Testing Process.

Sensor Node Development

1. Using the schematic diagram provided, connect all the necessary connections starting from the Arduino Mega 2560 to the sensors and relays using copper wires through soldering. To connect the series and parallel connections, a universal PCB must be used.

Note: a factory PCB can be used but with the system made by the proponents, a layout design is not present.

2. Install the Arduino Mega 2560 and Circuit board in the chassis used using PCB Standoff Spacers.
3. Make holes (if not present) for the wiring of the sensors to be used. Using this hole, the connections from the sensor node and power management will also go through this.
4. For the Arduino IDE, install the necessary libraries on the laptop to prepare the uploading of codes.
5. Upload the necessary codes using a laptop to the Arduino Mega 2560.

Note: changes in the code must be made specifically, in void setup where the time is indicated and must be change for the scheduled correction.

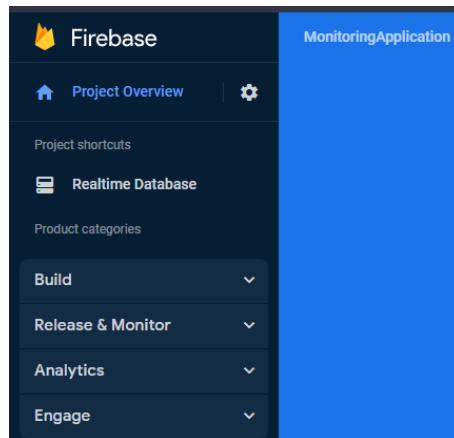
```

myRTC.setYear(__);
myRTC.setMonth(__);
myRTC.setDate(__);
myRTC.setDoW(__);
myRTC.setHour(__);
myRTC.setMinute(__);
myRTC.setSeconds(__);

```

Gateway Development and WSN Connection

1. Reformat a Raspberry Pi (model 3 onwards) and create a Python file with the codes provided.
2. Connect the ZigBee S2C module to the Raspberryy Pi using the USB connector.
3. Create a project at Firebase (<https://console.firebaseio.google.com>) specifically Realtime Database at Build. This is where the data will be sent from the sensor nodes and to the application.
4. Install the necessary libraries for the python program.
5. Using the main Python program, change the **config{}** to the necessary credentials of the created Firebase project. This setting can be from through the SETTING SYMBOL beside project overview on the top left corner of the website as seen on the picture.



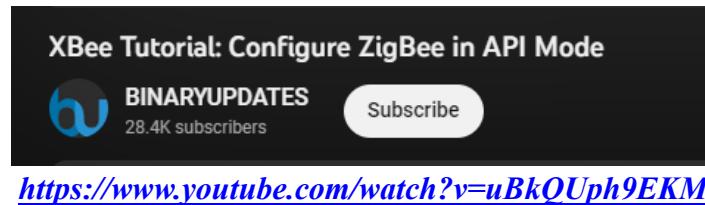
Go to project settings. Scroll through the settings until Your Apps. Web apps can be seen here where the necessary credentials can be found below **\$ npm install firebase**. The picture below shows the required credentials is between the curly brackets **{}**. *Note: the censored characters are for security purposes whereas if viewed through the website, it will not be censored.*

```

const firebaseConfig = {
  apiKey: "XXXXXXXXXXXXXX",
  authDomain: "XXXXXXXXXXXXXX",
  databaseURL: "XXXXXXXXXXXXXX",
  projectId: "XXXXXXXXXXXXXX",
  storageBucket: "XXXXXXXXXXXXXX",
  messagingSenderId: "XXXXXXXXXXXXXX",
  appId: "XXXXXXXXXXXXXX",
  measurementId: "XXXXXXXXXXXXXX"
};

```

6. For the connection of the ZigBee modules, **install XCTU**, a software for configuration and connection of different modules that uses ZigBee protocol, and **watch the video** linked below to configure what modules will serve as the Coordinator (for gateway) and Routers (for nodes).



<https://www.youtube.com/watch?v=uBkQUph9EKM>

7. After this configuration, the python is ready to run for the Raspberry Pi to act as the gateway.

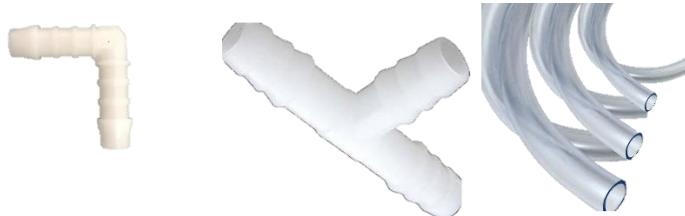
Correction System Integration

1. Gather the materials necessary and clean the water tanks inside and out. Make sure there are no water or solution inside. For water tanks, cut the necessary parts in order to properly install the water pumps and make sure it has room for the process of mixing the solutions.
2. Drill a small hole on the bottom part of the water tank as a passageway for the wire of the water pump. Put the sealant inside where the wire enters as to seal it to prevent flow of water. Install the electric solenoid valve using All Purpose Epoxy roughly above the maximum amount of water/solution that will be put inside the tank.

Note: Use the picture below for reference, if two Solenoid Valves and Water pumps will be installed, it can be installed side by side.



3. Connect the wires of the solenoid valve and water pump in a parallel connection and connect them on its dedicated relay.
4. For the drip irrigation line, use a 5mm water plastic pipe for the water line and use 5mm L-shaped and T-shape Connectors to connect all the different water lines for each container as shown in the picture.



5. For the main water line to the plants, connect a single water line to the different lines. The pictures will guide what will be the connection for the main water line.



The white circle indicates the connection of the main water line to the water lines for different containers.

Note: The size of the water lines will depend on the deployment area, thus adjustments and distance must be considered.

6. For the drip arrows, make a drip line for each plant in which the design will be based on the number of crops. Use the picture as a guide to make a drip line.



Testing Process

1. For the Arduino and Correction system, utilize the codes provided specifically **Correction_trial.ino**, as this code produces Serial print of the parameters, and activates the correction system with corresponding Serial print on what water pumps should be currently working.
2. For the Gateway and Firebase, utilize the main codes provided with the additional line of
print(x1);

after the line in which the variable x1 is used. This testing should provide an array of values and strings like the picture below:

```
[ 'Node1', '98.00', '31.80', '29.56', '3329.17', '97.00', '174.00', '3.00', '9.00', '13.00', '27.00', '5', '26', '7', '15', '33', '0.00',
  '0.00', '0.00' ],
[ 'Node2', '71.00', '33.70', '29.81', '2554.17', '100.00', '129.00', '3.00', '1.00', '2.00', '5.00', '5', '26', '3', '17', '14', '0.00',
  '0.00', '0.00' ]
```

3. For the Monitoring Application, make sure the necessary credentials of the Firebase are correct for the connection of the monitoring application. To check if the values are correct, compare the values of the Firebase values and what the monitoring application presented.

Frequently Asked Questions (FAQs):

- In addition of another node, where will I connect its power?
 - To the same power management system, but make sure the current value can handle the addition of a new node.
- Can we add or change the plants to be planted?
 - In order add or change the plants to be planted, the proponents need to change the program in Fuzzy Logic Algorithm based on the threshold of the plants it needs.
- Where can we buy the fertilizers?
 - The fertilizers can be bought at any soil fertilizer store or through online shopping. Ensure that you specifically purchase sulfur and lime for soil. For nitrogen, look for urea 46-0-0. For phosphorus, choose phosphate 0-22-0. Lastly, for potassium, opt for potash 0-0-60.

Troubleshooting Guide:

- If some sensors don't work normally, like maximum value or negative value, check the connections of its power and data signal. Loose connections can greatly affect this problem.
- If the uploading of codes doesn't work on the Arduino, remove the Jumper Wires for the XBee S2C Module of RX and TX, upload it, then after uploading put it back.



V. SUMMARY

The process of duplicating the SOIL-N-WAN system involves several key steps: sensor node development, gateway development and WSN connection, correction system integration, and testing.

During the sensor node development phase, the manual provides detailed instructions on assembling and configuring the sensor nodes, which play a crucial role in collecting data related to soil parameters. This section covers the selection of appropriate sensors, their placement, and the necessary programming to ensure accurate data acquisition.

The next step involves the development of the gateway and establishing a wireless sensor network (WSN) connection. The manual outlines the process of setting up the gateway, configuring the communication protocols, and establishing a reliable connection with the sensor

nodes. This section also addresses any necessary network security measures to protect data integrity.

The correction system integration phase focuses on incorporating the correction system into the overall setup. This system aims to refine and adjust parameters based on the sensor readings, ensuring high accuracy and reliability. The manual provides guidance on integrating the correction algorithms and implementing necessary calibration techniques to enhance the performance of the system.

Lastly, the testing process is an essential step to verify the functionality and performance of the duplicated system. The manual outlines various tests and procedures to validate the accuracy of the sensor readings, evaluate the stability of the WSN connection, and assess the overall system performance under different conditions. This section emphasizes the importance of thorough testing to ensure the system meets the desired specifications.

In summary, the manual provides a comprehensive guide for duplicating the SOIL-N-WAN system, covering sensor node development, gateway setup and WSN connection, correction system integration, and testing procedures. By following these step-by-step instructions, users can successfully replicate the system and gain a deep understanding of its components, functionality, and performance.

For any assistance or clarification needed, please don't hesitate to reach out. We are here to support you throughout the process of duplicating the SOIL-N-WAN system. Additional resources, references, and contact information can be provided upon request. Feel free to contact us through the number and email provided below.

Number: 09352458158

Email address: ssoilogist@gmail.com

ANNEX XV

Proponent's Information



SUMMARY

Jonard Bryan S. Cabanding



09760657122



Cavite, Philippines



jonardbryancabanding@gmail.com

I am an avid enthusiast of assembling things and have a genuine passion for hands-on projects. I find joy in bringing different components together to create functional and innovative solutions. Furthermore, my curiosity drives me to constantly seek new knowledge and stay up-to-date with the latest advancements in technology. I am particularly intrigued by the ever-evolving world of technology and continuously strive to expand my understanding of its various applications. With a keen eye for detail and a strong desire to learn, I am eager to contribute my skills and enthusiasm to a dynamic organization that values innovation and embraces technological advancements.

CAREER

AUG – OCT 2022

Gakken
Philippines, Inc.

OJT/Trainee

- Manage assessment and repair of printing machines.
- Manage quality checking of the machines.
- Collaborate with other colleague in assessing machines.

EDUCATION

TERTIARY

BS ELECTRONICS ENGINEERING

Technological University of the Philippines - Manila
2019 – 2023

SECONDARY

SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

San Sebastian College - Recoletos de Cavite
2017 – 2019

St. Mary Magdalene School
2013 - 2017

SKILLS

- Proficient in MS Office applications.
- Basic Knowledge on Multisim, Matlab, Cisco Packet Tracer, and Python
- Troubleshooting
- Computer Skills
- Versatile

CERTIFICATIONS

- Master IP Addressing and Subnetting for CCNA
- Fortinet NSE1, NSE2, NSE3
- PacketHACKS 2023 Hackathon Finalist



Jholrey C. Caibigan



09693116609



Mandaluyong City, Philippines



jholreycaibigan@gmail.com

SUMMARY

Desires to work with a strong organization suited to hone my skills and utilize my knowledge in electronics engineering to grow and develop professionally and personally. Is flexible in adapting new tasks and welcomes new challenges to become efficient in life specially in technological field.

CAREER

AUG – SEPT 2022

Microdata Systems
and Management,
Inc.

OJT/Trainee

- Utilizing Cisco Meraki Dashboard to monitor network
- Troubleshooting simple network problems
- Simulating different network configurations
- Installation of endpoint security software
- Monitoring endpoint security through dashboard
- Managing tickets

EDUCATION

TERTIARY

BS ELECTRONICS ENGINEERING

Technological University of the Philippines - Manila
2019 - 2023

SECONDARY

SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

Calaca Senior High School
2017 - 2019

Calaca Academy Inc.
2013 - 2017

SKILLS

- Proficient in MS Office applications.
- Proficient on Multisim, Matlab, Cisco Packet Tracer, and Python
- Remarkable communication skills
- Complex problem solver
- Service-focused
- Adaptability and capability to work under pressure
- Willing to learn new things to improve work efficacy

CERTIFICATIONS

- Master IP Addressing and Subnetting for CCNA
- (ISC)2 Certified in Cybersecurity
- Fortinet NSE1, NSE2, NSE3
- PacketHACKS 2023 Hackathon Finalist



Ricky M. Indiana

09773822355
 Quezon City, Philippines
 indianoricky@gmail.com

SUMMARY

To use and enhance my skills and knowledge with quality service and work with a respected company focusing on IT / Communications industry while being flexible and eager to learn.

CAREER

AUG – SEPT 2022

Synetcom
Philippines, Inc.

OJT/Trainee

- Demonstration in managing traffic between network devices
- Troubleshooting and conducting different configuration to be deployed on Versa devices
- Created a presentation for practice pitching on solution products

EDUCATION

TERTIARY

BS ELECTRONICS ENGINEERING

Technological University of the Philippines - Manila
2019 - 2023

SECONDARY

SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

Dr. Carlos S. Lanting College
2017 - 2019

Ernesto Rondon High School
2013 - 2017

SKILLS

- Data Handling and Analysis
- Proficient user of circuit analysis software
- Quality circuit analysis and troubleshooter
- Has Knowledge in Network Handling (Cisco Packet Tracer)
- Quality Programming Skills | Python, MATLAB, Arduino
- Research and Development
- Flexible, Quick Learner, and Adaptable



SUMMARY

John Nicholas B. Moreno

09070133608
Pasay City, Philippines
Morenojohn167@gmail.com

Motivated and successful professional with a track record of success. centered on producing exceptional results by putting diligence first and making proactive moves. Constantly looking to improve both personally and professionally. Quick to adapt. Excellent problem-solving skills and a collaborative approach allow for success in hectic environments. energized to provide knowledge and abilities to support organizational achievement.

CAREER

AUG – OCT 2022

Tronix Master Inc.

OJT/Trainee

- Repairing different home appliances
- Managing customers concerns
- Storage of equipment

EDUCATION

TERTIARY

BS ELECTRONICS ENGINEERING

Technological University of the Philippines – Manila
2019 – 2023

SECONDARY

SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

Manila Tytana Colleges
2017 – 2019

Pasay City West High School
2013 - 2017

SKILLS

- Proficient in MS Office applications
- Proficient on Multisim, Matlab, Cisco Packet Tracer, and HTML
- Communication skills
- Electronics Hardware
- Time management
- Fast learner
- Adaptability and capability to work under pressure
- eager to expand my knowledge and acquire new skills to enhance my effectiveness at work

CERTIFICATIONS

- Master IP Addressing and Subnetting for CCNA
- (ISC)2 Certified in Cybersecurity
- Fortinet NSE1, NSE2, NSE3
- PachetHACKS 2023 Hackathon Finalist

T



Angelika Mae C. Virtucio

09352458158
Las Piñas, City
gelikavirtucio@gmail.com

SUMMARY

She is persistent in her efforts. Wanting to learn, progress, and excel in the field of Electronics Engineering. To provide the greatest possible service to my field and community. A highly motivated and adaptable individual with a strong foundation in programming, office tools, and networking. A flexible and hardworking professional with excellent communication skills. Quick to learn and easily adaptable to new environments and challenges.

CAREER

AUG – SEPT 2022	OJT/Trainee
GoSolar Philippines	<ul style="list-style-type: none">• Troubleshooting wirings• Simulating different roof design for solar panel• Answering call and inquiries• Wiring circuit box• Managing tickets

EDUCATION

TERTIARY	BS ELECTRONICS ENGINEERING <i>Technological University of the Philippines - Manila</i> 2019 - 2023
SECONDARY	SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS <i>Centro Escolar Las Piñas</i> 2017 - 2019

Las Piñas College
2013 - 2017

SKILLS

- Basic Python Language Program
- Basic Octave/ MATLAB Language Program
- Basic Microsoft Office Techniques
- Proficiency in IP Addressing and Subnetting
- Flexible Worker
- Hard Working
- Good Communication Skills
- Easily to adopt

CERTIFICATIONS

- Master IP Addressing and Subnetting for CCNA
- (ISC)² Candidate
- Fortinet NSE1, NSE2, NSE3