SMART HYDROPONICS FOR LETTUCES WITH THE USE OF INTERNET OF THINGS

A Senior Design Project

Presented to the Faculty of the Engineering Department

College of Science and Information Technology

ATENEO DE ZAMBOANGA UNIVERSITY

by

Hairun, Al-Fitri C.

Tan, Alvin T.

BS Computer Engineering V

Engr. Louie Virgil A. Gallardo

Adviser

April 2023

ABSTRACT

As the population of the world continues to grow at a rapid rate, the food demand is also expected to increase at the same time (Elferink and Schierhorn, 2016). This was impressed upon as a pressing concern to the two (2) student-researchers of Ateneo de Zamboanga University. Solution is sought to a rising global demand for food. Attention is given to hydroponics which provides a higher production yield and lower water consumption when compared to soil, which makes it the ideal solution for improving food security, particularly in third world countries (Hydroponicsspace.com, 2023).

Hydroponics, however, is complicated because there are more growing factors that can affect the growth of plants. This can be addressed by constantly monitoring the system parameters. Such led to the project of these researchers to develop an automated hydroponic system that cultivates and grows lettuces. It is to be noted that lettuce is one of the most commonly grown vegetables in hydroponics. It is a green leafy vegetable that is an excellent source of essential nutrients and antioxidants. It is known as a rich source of vitamin K and vitamin A and has numerous impressive health benefits (Tadimalla, 2023).

To underscore, hydroponics is the cultivation of plants without using soil, it is a technique where the plants are grown using a water-based nutrient solution. The hydroponic system requires less water that saves up to 80 to 90% of water compared to soil gardening. It can be performed in small areas, has faster growth rate, and no pest control products are needed. But generally, the hydroponic system requires human interaction when it comes to regulation and monitoring of parameters, such as, pH

value, TDS value, water temperature; and, maintaining water level in growing of plants, which are both daunting tasks.

This study aims to expand and improve the utilization of hydroponics as well as to create an environmentally independent system for indoor plant growth. The researchers ventured into Smart Hydroponics for Lettuces (SHL) which is automated and requires less human interaction after placing the germinated plant into the system. The SHL with the use of Internet of Things (IOT) seeks to automate the feeding of nutrient solutions, irrigation, monitoring and optimization of plant growth of the hydroponic system. Nutrient Film Technique (NFT) is the hydroponic method chosen by the researchers in growing the lettuces.

The SHL has two control panels for the user. SHL web application serves as the main control panel for SHL system. The web application is designed for the user to set the desired values for pH, TDS, temperature, and water level. It is also designed for the user to monitor the values, configure the grow light schedule, configure the temperature for turning on or off the thermoelectric peltier cooler, view the live footage of the growing lettuces, monitor the sensor logs, and adjust the log recording and log clearing. The recordings will be stored in a database. While the Control and Display module serves as the supplementary control panel. This module allows the user to view the sensor readings on LCD, and use a keypad to change the target values for pH, TDS, temperature, and water level to maintain the system.

The developed system will lessen human involvement and aid the production or cultivation of lettuces. Indeed, the results of the tests showed that the SHL was able to address the objectives for the development of this project.

ACKNOWLEDGMENT

We, the researchers give honor and glory to our Almighty God for the endless grace that have allowed us to accomplish this project successfully. We surpassed challenges and trials, and gained determination and strength to pursue this undertaking because of His divine guidance. We offer this endeavor to Him for the knowledge, peace of mind and good health bestowed upon us; and, for the blessing of people in our lives.

Indeed, this design project comes to fruition with the kind support, help and understanding of certain individuals who are deserving of recognition. We take this opportunity to express our utmost appreciation.

We are highly indebted to our indefatigable and brilliant adviser, Engr. Louie Virgil A. Gallardo for the immense wisdom and patient mentoring. Our adviser's expertise and guidance have helped us surmount what initially seemed to be an uphill task. For being the source of our motivation and enthusiasm, we could have not imagined a better adviser. Hence our profound gratitude to Sir Louie.

To our classmates, teachers, and friends whose words of hope have been extremely invaluable, we return such with nothing less than kindness.

Finally, to our respective families, for having been our constant in terms of support in all aspects, even inspiring us to always be the best versions of ourselves and to finish this piece of work, you have given significant meaning to selflessness and love

that know no limits and boundaries. We are truly blessed to share this milestone with each one.

Table of Contents

CHAPTER 1 ENGINEERING DESIGN 1
1.1 Hardware Components 3
1.1.1 Sensor Reading Module 3
1.1.1.1 Analog TDS Sensor 4
1.1.1.2 Analog pH Sensor 5
1.1.1.3 Analog Signal Isolator 6
1.1.1.4 Temperature Sensor 7
1.1.1.5 Ultrasonic Sensor 8
1.1.1.6 Arduino Uno R3 9
1.1.2 Hydroponic Maintaining and Monitoring Module 10
1.1.2.1 Hydroponic Maintaining Module 10
1.1.2.1.1 Relay Module 11
1.1.2.1.2 Dosing Pumps 12
1.1.2.1.3 Water Pumps 13
1.1.2.1.4 Thermoelectric Peltier Cooler 15
1.1.2.1.5 Grow Lights 17
1.1.2.1.6 Power Supply 17
1.1.2.2 Control and Display Module 18
1.1.2.2.1 LCD 18
1.1.2.2.2 Membrane Keypad 19
1.1.2.3 Camera Module 21
1.1.2.4 Raspberry Pi 3 Model B 21
1.2 Hardware Interconnection 22
1.3 Software Design 27
1.3.1 Arduino Software Modules 28
1.3.1.1 TDS Reading Module 28
1.3.1.1.1 Libraries and Variables for TDS Reading Module 28
1.3.1.1.2 Initializing the TDS Reading Module 29
1.3.1.1.3 Obtaining the TDS Reading 29

1.3.1.2 pH Reading Module 30
1.3.1.2.1 Initializing Variables for the pH Sensor 30
1.3.1.2.2 Obtaining the Voltage Reading of the pH Sensor 31
1.3.1.2.3 Median Filtering System 31
1.3.1.2.4 Getting the Calibration Data from EEPROM 32
1.3.1.2.5 pH Calibration 33
1.3.1.2.6 Obtaining the pH Reading 37
1.3.1.3 Water Level Reading Module 37
1.3.1.3.1 Initializing Variables for the Water Level Reading 38
1.3.1.3.2 Obtaining Distance Reading 38
1.3.1.3.3 Smoothening Data with Running Average 39
1.3.1.4 Temperature Reading Module 40
1.3.1.4.1 Libraries and Variables for Temperature Reading 40
1.3.1.4.2 Obtaining Temperature Reading 41
1.3.1.5 Serial Communication Module 41
1.3.1.5.1 Printing Sensor Data for Serial Communication 41
1.3.2 Raspberry Pi Software Modules 42
1.3.2.1 Hydroponics Maintaining Module 42
1.3.2.1.1 Receiving Sensor Data from Arduino 42
1.3.2.1.2 Obtaining the Set Values 43
1.3.2.1.3 Maintaining Water Level and Temperature 44
1.3.2.1.4 Setting Up Thresholds for pH and TDS 45
1.3.2.1.5 Calculating the Dosing Duration for the pH and TDS 45
1.3.2.1.6 Dosing Nutrient Solutions and pH Buffer Solutions 47
1.3.2.1.7 Turning the Grow Lights on or off 50
1.3.2.2 Keypad Membrane Module 50
1.3.2.2.1 Setting up the GPIO Pins for the Keypad Membrane 50
1.3.2.2.2 Setting up the Keypad Membrane 51
1.3.2.2.3 Menu Selection 51
1.3.2.2.4 Storing Inputs into String 51

1.3.2.2.5 Selecting Exit or Enter 52
1.3.2.2.6 Password Check 53
1.3.2.2.7 Changing the Set Values 53
1.3.2.3 LCD Module 56
1.3.2.3.1 Displaying the Home Screen 56
1.3.2.3.2 Displaying pH Menu 57
1.3.2.3.3 Displaying TDS Menu 58
1.3.2.3.4 Displaying Temperature Menu 59
1.3.2.3.5 Displaying Water Level Menu 60
1.3.2.4 Log Recording Module 61
1.3.2.4.1 Log Recording 61
1.3.2.4.2 Log Clearing 62
1.3.3 Web Application Software 62
1.3.3.1 Configure Settings in the Web Application 63
1.3.3.1.1 Password System 63
1.3.3.1.2 Setting pH in Web Application 64
1.3.3.1.3 Setting TDS in Web Application 65
1.3.3.1.4 Setting Temperature in Web Application 65
1.3.3.1.5 Setting Water Level 66
1.3.3.1.6 Setting Grow Lights Schedule 67
1.3.3.1.7 Changing Password 68
1.3.3.1.8 Setting Log Interval 69
1.3.3.2 Viewing the Web Application 71
1.3.3.2.1 Displaying the Main Menu 71
1.3.3.2.2 Displaying the Logs 74
CHAPTER 2 RESULTS AND DISCUSSION 75
2.1 Sensors Test 75
2.1.1 pH Sensor Test 75
2.1.1.1 Measuring an Acidic Solution 76
2.1.1.2 Measuring a Neutral Solution 77

2.1.1.3 Measuring an Alkaline Solution 79
2.1.2 TDS Sensor Test 81
2.1.3 Temperature Sensor Test 83
2.1.4 Ultrasonic Sensor Test 84
2.2 Functionality Test 86
2.2.1 Modifying Set pH Test 86
2.2.2 Modifying Set TDS Test 87
2.2.3 Modifying Set Temperature Test 88
2.2.4 Modifying Set Water Level Test 89
2.2.5 Scheduling Grow Lights Test 90
2.2.6 Adjusting Log Intervals 91
2.2.6.1 Adjusting Log Saving Interval Test 91
2.2.6.2 Adjusting Log Clearing Interval 92
2.2.7 Changing Passcode Test 93
2.3 Membrane Keypad Test 94
2.3.1 Keypad Button Test 94
2.3.2 Menu Selection Test 96
2.3.3 Modifying Set Values in Keypad Test 97
2.3.3.1 Modifying Set pH in Keypad Test 97
2.3.3.2 Modifying Set TDS in Keypad Test 98
2.3.3.3 Modifying Set Temperature in Keypad Test 99
2.3.3.4 Modifying Set Water Level in Keypad Test 100
2.4 System Demo Test 101
2.4.1 Nutrient Solution Dosing Test 101
2.4.2 pH Buffer Solution Dosing Test 103
2.4.3 Water Pump Test 105
2.4.3.1 Water in Pump Test 105
2.4.3.2 Water Pump TDS Reduction Test 107
2.4.4 Thermoelectric Peltier Cooler Test 108
2.4.5 Grow Lights Test 109

2.4.6 RPI Camera Test 111
CHAPTER 3 CONCLUSION 112
CHAPTER 4 RECOMMENDATIONS 114
BIBLIOGRAPHY 116
USER'S MANUAL 117
APPENDICES 156
APPENDIX A HARDWARE DESIGN AND INTEGRATION 157
APPENDIX B SOFTWARE DESIGN 158
APPENDIX C KEYPAD 163
APPENDIX D RESUME 164

LIST OF FIGURES

Figure 1 System Architecture of SHL with the use of IOT 2
Figure 1.1.1.1 Analog TDS Sensor SEN0244 4
Figure 1.1.1.2 Analog pH Sensor SEN0161 5
Figure 1.1.1.3 Analog Signal Isolator 6
Figure 1.1.1.4 DS18B20 Temperature Sensor 7
Figure 1.1.1.5 Ultrasonic Sensor HC-SR04 8
Figure 1.1.1.6 Arduino Uno R3 9
Figure 1.1.2.1.1 8-Channel Relay 11
Figure 1.1.2.1.2 Peristaltic Dosing Pump 13
Figure 1.1.2.1.3 Submersible Water Pump 14
Figure 1.1.2.1.4.1 Assembled Thermoelectric Peltier Cooler 16
Figure 1.1.2.1.4.2 TEC1-12706 Peltier 16
Figure 1.1.2.1.5 Grow Lights 17
Figure 1.1.2.1.6 12,V 20A Power Supply 18
Figure 1.1.2.2.1 20x4 LCD 18
Figure 1.1.2.2.2.1 4x4 Membrane Keypad 19
Figure 1.1.2.2.2.2 Pin Out of 4x4 Membrane Keypad 20
Figure 1.1.2.3 RPI Camera 21
Figure 1.1.2.4 Raspberry Pi 3 Model B 22
Figure 1.2.1 Overall Hardware Interconnection 23
Figure 1.2.2 Sensors to Arduino Interconnection 24
Figure 1.2.3 USB Cable Connector 25
Figure 1.2.4 Interconnection of the Display and Control Module 25
Figure 1.2.5 Relay Connection to the RPI 26
Figure 1.2.6 Relay Connection 27
Figure 1.2.7 RPI Camera Slot 27

LIST OF TABLES

Table 2.1.1.1 Summary Table for Acidic Solution Test 76
Table 2.1.1.2 Summary Table for Neutral Solution Test 78
Table 2.1.1.3 Summary Table for Alkaline Solution Test 80
Table 2.1.2 Summary Table for TDS Sensor Test 82
Table 2.1.3 Summary Table for Temperature Sensor Test 83
Table 2.1.4 Summary Table for Ultrasonic Sensor Test 85
Table 2.2.1 Sumary of Modifying Set pH Test Results 87
Table 2.2.2 Summary of Modifying Set TDS Test Results 88
Table 2.2.3 Summary of Modifying Set Temperature Test Results 89
Table 2.2.4 Summary of Modifying Set Water Level Test Results 90
Table 2.2.5 Summary of Scheduling Grow Lights Test Results 91
Table 2.2.6.1 Summary of Adjusting Log Saving Interval Test Results 92
Table 2.2.6.2 Summary of Adjusting Log Clearing Interval Test Results 93
Table 2.2.7 Summary of Changing Passcode Test Results 94
Table 2.3.1 Results of the Keypad Buttons Test 95
Table 2.3.2 Results of the Menu Selection Test 96
Table 2.3.3.1 Results of the Modifying Set pH in Keypad Test 98
Table 2.3.3.2 Results of the Modifying Set TDS in Keypad Test 99
Table 2.3.3.3 Results of the Modifying Set Temperature in Keypad Test 99
Table 2.3.3.4 Results of the Modifying Set Water Level in Keypad Test 100
Table 2.4.1 Results of Nutrient Solution Dosing Test 102
Table 2.4.2 Results of pH Buffer Solution Dosing Test 104
Table 2.4.3.1 Test Summary of Water in Pump 105
Table 2.4.3.2 Test Summary of Water Pump TDS Reduction 107

Table 2.4.4 Test Summary of Thermoelectric Peltier Cooler 108
Table 2.4.5 Test Summary of Grow Lights..... 110
Table 2.4.6 Test Summary of RPI Camera 111

PROGRAM LISTINGS

Listing 1a Libraries and Variables Needed for TDS Reading Module 29

Listing 1b Code Snippet of Initializing the TDS Reading Module 29

Listing 1c Code Snippet of Obtaining the TDS Reading.....29

Listing 2a Code Snippet of Initializing Variables for the pH Sensor.....30

Listing 2b Code Snippet of Obtaining the Voltage Reading of the pH Senor 31

Listing 2c Code Snippet of Median Filtering System 32

Listing 2d Code Snippet of Getting Calibration Data from EEPROM 32

Listing 2e Code Snippet of pH Calibration Commands 33

Listing 2f Code Snippet of pH Calibration Initialization.....34

Listing 2g Code Snippet of Entering Calibration Mode.....34

Listing 2h Code Snippet of Acid Calibration 35

Listing 2i Code Snippet of Alkali Calibration.....36

Listing 2j Code Snippet of Completing pH Calibration.....37

Listing 2k Code Snippet of Obtaining the pH Reading 37

Listing 3a Code Snippet of Initializing Variables for the Water Level Reading ..38

Listing 3b Code Snippet of Obtaining Distance Reading 39

Listing 3c Code Snippet of Smoothened Data with Running Average.....40

Listing 4a Libraries and Variables Needed for TDS Reading Module 41

Listing 4b Code Snippet of Obtaining the Temperature Reading 41

Listing 5 Code Snippet for Printing Sensor Values for Serial Communication 42

Listing 6 Code Snippet of Receiving Sensor Data from Arduino 43
Listing 7 Code Snippet of Obtaining the Set Values.....44
Listing 8 Code Snippet of Maintaining Water Level and Temperature 45
Listing 9 Code Snippet of Setting Up Thresholds for pH and TDS 45
Listing 10a Getting the Difference of Set Parameters and Reading 46
Listing 10b Code Snippet of Calculating for the Dosing Duration.....47
Listing 11 Code Snippet of Dosing Nutrient and pH Buffer Solutions.....49
Listing 12 Code Snippet of Turning the Grow Lights On or Off 50
Listing 13 Assigned GPIO Pins for the Keypad Membrane 50
Listing 14 Setting up the Keypad Membrane.....51
Listing 15 Code Snippet of Menu Selection.....51
Listing 16 Code Snippet of Storing Inputs to String 52
Listing 17 Code Snippet of Selecting Exit or Enter.....52
Listing 18 Code Snippet of Password Check 53
Listing 19a Code Snippet of Changing the Set TDS 54
Listing 19b Code Snippet of Changing the Set Water Level.....55
Listing 19c Code Snippet of Changing the Set Temperature 55
Listing 19d Code Snippet of Changing the Set pH 56
Listing 20 Code Snippet of Displaying the Home Screen 56
Listing 21 Code Snippet of Displaying pH Menu 57
Listing 22 Code Snippet of Displaying TDS Menu.....58
Listing 23 Code Snippet of Displaying Temperature Menu 59
Listing 24 Code Snippet of Displaying Water Level Menu.....60

Listing 25 Code Snippet of Log Recording 61
Listing 26 Code Snippet of Log Clearing 62
Listing 27a Form Field for Passcode 63
Listing 27b Code Snippet of Retrieving Password and Masterkey 64
Listing 28a Form Field for Set pH 64
Listing 28b Code Snippet of Setting pH in Web Application 64
Listing 29a Form Field for Set TDS 65
Listing 29b Code Snippet of Setting TDS in Web Application 65
Listing 30a Form Field for Set Temperature 66
Listing 30b Code Snippet of Setting Temperature in Web Application 66
Listing 31a Form Field for Set Water Level 66
Listing 31b Code Snippet of Setting Water Level in Web Application 67
Listing 32a Form Field for Grow Light Schedule 67
Listing 32b Code Snippet of Changing Grow Lights Schedule 68
Listing 33a Form Field for Changing Passcode 69
Listing 33b Code Snippet of Changing Passcode 69
Listing 34a Form Field for Log Saving Interval 69
Listing 34b Code Snippet of Changing Log Saving Interval 70
Listing 34c Form Field for Log Clearing Interval 70
Listing 34d Code Snippet of Changing Log Clearing Interval 70
Listing 35a Code Snippet of Creating Buttons for Menus 71
Listing 35b Code Snippet of Displaying Grow Lights Schedule 72
Listing 35c Code Snippet of Displaying pH Reading and Set pH 72
Listing 35d Code Snippet of Displaying TDS Reading and Set TDS 73

Listing 35e Code Snippet of Displaying Temperature Reading and Set Temperature 73

Listing 35f Code Snippet of Displaying Water Level Reading and Set Water Level.....74

Listing 36 Code Snippet of Displaying Logs 74

CHAPTER 1 ENGINEERING DESIGN

The Smart Hydroponics for Lettuces (SHL) with the use of Internet of Things (IOT) is designed for the user to constantly monitor and maintain vital parameters, such as, pH level, TDS level, water temperature, and water level of the growing lettuces. It is also designed for the user to configure the schedule of the grow lights, configure the thermoelectric peltier cooler, view live footage of the growing lettuces, monitor the sensor logs, and adjust the log recording and log clearing intervals through SHL Web-Application. The system then stores the recordings in a database.

The user can also monitor and maintain the parameters in another control panel using LCD and membrane keypad.

The SHL automates the feeding of nutrient solutions, irrigation, monitoring and optimization of plant growth of the hydroponic system. The user must first set the parameters. Once done, the system is able to maintain the parameters and promote healthy plant growth.

Figure 1 shows the System Architecture of the Smart Hydroponics for Lettuces (SHL) with the use of Internet of Things (IOT).

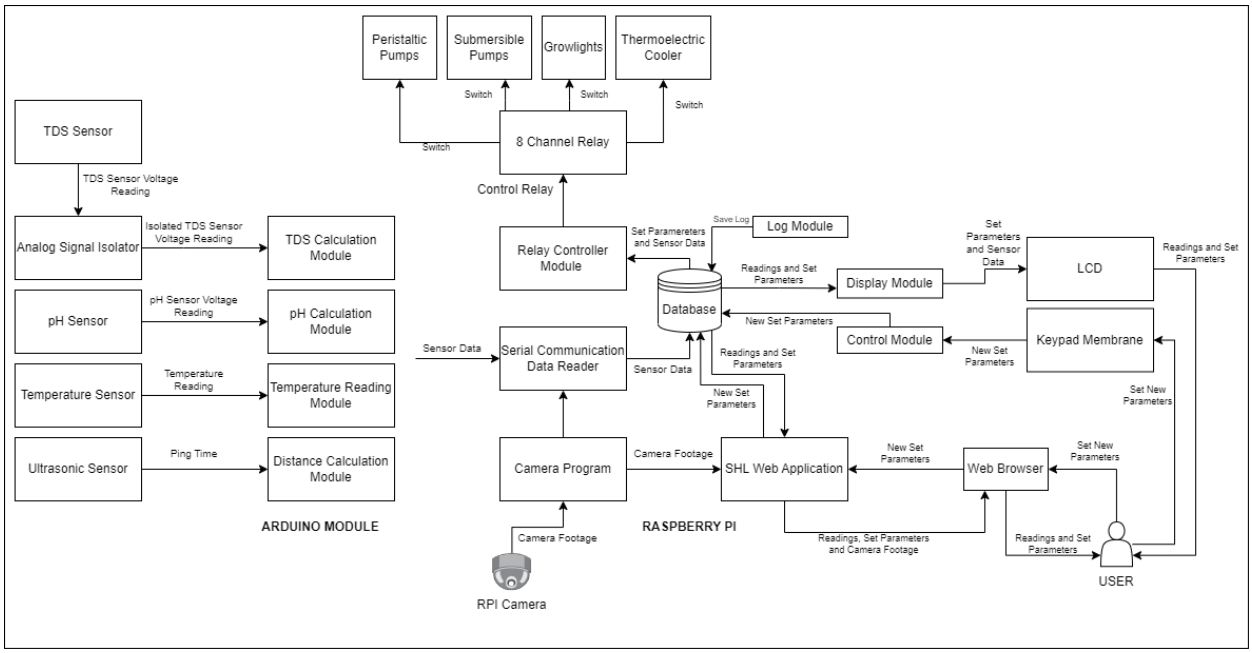


Figure 1. System Architecture of SHL with the use of IOT

The SHL system revolves around the development of an automated hydroponic system. In Figure 1, the Analog TDS sensor voltage is isolated by means of an Analog Signal Isolator to prevent interference with the Analog pH Sensor. The Analog TDS sensor voltage reading is then used by the TDS calculation module to calculate the TDS reading. The Analog pH sensor voltage reading is used by the pH calculation module to calculate the pH reading. The temperature sensor reading is used by the temperature reading module to get the temperature. The ultrasonic sensor ping time is used by the distance calculating module to calculate the distance. The sensor data is then sent via serial communication to the Raspberry Pi.

The Raspberry Pi first reads the Serial Input via the Serial Communication Reader then stores it in the database. The relay controller module uses sensor data and

set parameters to determine which relay needs to be open or close. The display module displays the sensor data and the set parameters on the LCD. The set parameters can be changed in the control module which is accessed by the user in the keypad membrane.

The Raspberry Pi is also responsible for hosting the web application which is accessible using a web browser where the sensor data and set parameters are also displayed. The user can also change the set parameters, view logs, and view live footage of the hydroponic. The logs are saved by the log module on an established interval which is dependent on the interval set by the user.

1.1 Hardware Components

The system is comprised of 2 main modules: (1) Sensor Reading Module, and (2) Hydroponic Maintaining and Monitoring Module.

1.1.1 Sensor Reading Module

This module is a collection of sensors connected to an Arduino. It is responsible for handling these key functions of the system: (1) Obtaining the TDS Reading, (2) Obtaining the pH Reading, (3) Preventing interference between the Analog pH Sensor and the Analog TDS Sensor, (4) Obtaining the Temperature Reading, (5) Obtaining the Distance Reading, and (6) Sending the Sensor Data to the Raspberry Pi.

The Sensor Reading Module consists of six (6) modules: (1) Analog TDS Sensor, (2) Analog pH Sensor, (3) Analog Signal Isolator, (4) Temperature Sensor, (5) Ultrasonic Sensor, and (6) Arduino Uno R3.

1.1.1.1 Analog TDS Sensor

The function of this module is to obtain the TDS reading of the hydroponic solution. The TDS reading is used to determine the nutrient level of the hydroponic solution. The TDS Sensor that is used in the system is the Analog TDS Sensor SEN0244 as shown in Figure 1.1.1.1.

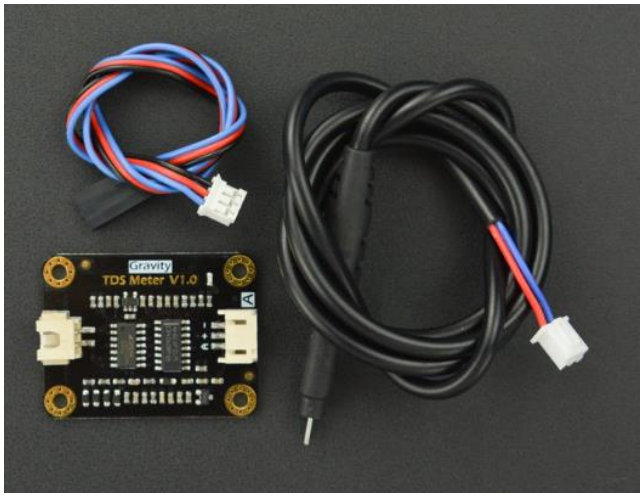


Figure 1.1.1.1 Analog TDS Sensor SEN0244

The TDS Sensor supports 3.3-5.5V wide voltage input and 0-2.3V analog voltage output, which makes it compatible with Arduino Uno. The TDS meter measurement ranges ideally from 0ppm to 1000ppm with the accuracy of ±10% F.S. at 25°C, which

is enough to measure the optimum TDS level of the lettuces that are between 560ppm and 840ppm (Homehydrosystems.com, n.d.). The TDS probe is waterproof so it can be immersed in water for long time measurement. The device requires 5V DC voltage and has three wires, namely, Ground, VCC, and Analog.

1.1.1.2 Analog pH Sensor

The function of this module is to obtain the pH reading of the hydroponic solution. The pH level is an important factor in growing lettuces. It affects which nutrients and chemicals are dissolved in the solution. The pH Sensor that is used in the system is the Analog pH Sensor SEN0161 as shown in Figure 1.1.1.2.

Figure 1.1.1.2 Analog pH Sensor SEN0161

The Sensor is simple with cost-effective pH meter that is designed for use with Arduino controllers, which comes with LED that works as the power indicator, Bayonet Neil-Concelman (BNC)

connector, and PH2.0 sensor interface. The analog pH meter has good accuracy range providing pH measurements at ± 0.1pH (25°C), thereby making it ideal for use in a research project. The Sensor is also designed to return values quickly. It can measure pH of 0 to 14. The meter has its potentiometer for adjusting the gain potential for calibration purposes. The device requires 5V DC voltage and has three wires, namely, Ground, VCC, and Analog.

1.1.1.3 Analog Signal Isolator

The purpose of the Analog Signal Isolator is to prevent electric interference between the Analog pH Sensor and the Analog TDS Sensor. The researchers added this module to the project since it was noticed that the pH and TDS Sensors interfered with each other and as a result, these did not measure properly in the same container. Figure 1.1.1.3 below shows the Analog Signal Isolator.

Figure 1.1.1.3 Analog Signal Isolator

This module has on board 5V isolation power which is directly supplied to the sensor, so the user does not need to prepare external power. The connector wires are specially designed for DFRobot 3-pin analog compatibility, making it easy plug and play with no need for soldering.

1.1.1.4 Temperature Sensor

The purpose of this module is to obtain the temperature reading of the hydroponic solution. Temperature is another important factor in growing lettuces since it influences most of the plant processes. The Temperature Sensor used in the system is the DS18B20 Temperature Sensor as shown in Figure 1.1.1.4.

Figure 1.1.1.4 DS18B20 Temperature Sensor

The DS18B20 is a 1-wire programmable Temperature Sensor. It can measure a wide range of temperature from 55°C to +125°C with a decent accuracy of ±0.5°C. The device requires 5.0V and has three wires, namely, Ground input, +5V Input, and

Signal. A resistor with a minimum resistance of 4700 Ω should be connected to the Signal and +5V Input. The signal wire is connected to one of the digital pins of the microcontroller to retrieve temperature data.

1.1.1.5 Ultrasonic Sensor

The purpose of this module is to measure the distance of the water surface from the sensor to be able to calculate the water level of the hydroponic solution. The sensor used in the system is the Ultrasonic Sensor HC-SR04. Figure 1.1.1.5 below shows the Ultrasonic Sensor HC-SR04.

Figure 1.1.1.5 Ultrasonic Sensor HC-SR04

The Ultrasonic Sensor HC-SR04 has 4-pin modules, namely, VCC, Ground, Trigger, and Echo. The Sensor provides excellent non-contact range detection between 2cm to 400cm with the accuracy of 3mm, or it can measure distance from 2cm to 80cm with the accuracy up to 3mm.

The current consumed by the Sensor is less than 15mA, therefore it can be directly powered by the MCU. The VCC supplies power to the Sensor to connect the VCC pin to the 5V pin on the MCU and the GND pin to the ground pin. The Trigger and Echo pins are connected to I/O digital pins of the MCU. The Ultrasonic Sensor uses sound wave to calculate the distance.

1.1.1.6 Arduino Uno R3

The Arduino Uno R3 is used to interconnect the Analog TDS Sensor, Analog pH Sensor, Analog Signal Isolator, Temperature Sensor, and Ultrasonic Sensor. It serves as the processor for the sensor calculation and calibration. It is also responsible for sending the sensor data to the Raspberry Pi. Figure 1.1.1.6 shows the Arduino Uno R3.

Figure 1.1.1.6 Arduino Uno R3

The Arduino Uno R3 is the most popular development board among the Arduino family. Based on Atmel's ATmega328

microcontroller, this board offers 14 digital input/output pins wherein 6 pins can be used as PWM outputs and 6 as analogue pins. The board can be programmed via USB port by using the Arduino IDE software.

1.1.2 Hydroponic Maintaining and Monitoring Module

The Hydroponic Maintaining and Monitoring Module consists of four (4) modules: (1) Hydroponic Maintaining Module, (2) Control and Display Module, (3) Camera Module, and (4) Raspberry Pi 3 Model B.

1.1.2.1 Hydroponic Maintaining Module

The Hydroponic Maintaining Module is a collection of components that maintains the growing conditions for lettuces. This module is responsible for handling six (6) key functions of the system: (1) Maintaining TDS Level, (2) Maintaining pH Level, (3) Maintaining Water Level, (4) Cooling the Hydroponic Solution, (5) Providing Alternative Sunlight for Lettuces, and (6) Providing Power to the Components.

The Hydroponic Maintaining Module consists of six (6) components: (1) Relay Module, (2) Dosing Pumps, (3) Water Pumps, (4) Thermoelectric Peltier Cooler, (5) Grow Lights, and (6) Power Supply.

1.1.2.1.1 Relay Module

The purpose of this component is to control the submersible pumps, peristaltic dosing pumps, grow lights, and thermoelectric peltier cooler.

The 5-Volt, 8-Channel Relay is used for this project which is made up of four dosing pumps, two submersible pumps, grow lights, and a thermoelectric peltier cooler. Figure 1.1.2.1.1 shows the 8-Channel Relay Module.

Figure 1.1.2.1.1 8-Channel Relay

A relay is an electrically-operated switch. This relay uses an electromagnet to mechanically operate the switch and provide electrical isolation between two circuits. Meanwhile, optocouplers generate another isolation layer between the inputs and the load. The input jumper consists of GND, VCC, and input pins. For this purpose, it provides easy connectivity with female jumper wires. Each submodular of the relay has one NC (normal close), one

NO (normal open), and one COM (common). NC stands for the normal close port contact and the state without power; NO stands for the normal open port contact and the state with power; while, COM means the common port. NC port or NO port can be chosen, depending if with power or not. Additionally, eight individual loads connect to the relay's NO terminals. In this case, a live wire connects to each relay's common terminal. The load powers up once it connects to the live wire after turning on the relays.

1.1.2.1.2 Dosing Pumps

The purpose of this component is to dose nutrient and pH buffer solutions into the hydroponic reservoir, to result to an increase in the TDS level and an increase or decrease in the pH level of the hydroponic solution. A precise dosing pump is needed for the dosing of nutrient and pH buffer solutions, since extra drops of solutions can affect the TDS level or the pH level of the hydroponic solution. Figure 1.1.2.1.2 shows the Peristaltic Dosing Pump.

Figure 1.1.2.1.2 Peristaltic Dosing Pump

The Peristaltic Dosing Pump is highly precise and can be calibrated to be highly accurate. The peristaltic pump has an input voltage of 12volts DC. It has a flow rate of 0-100 ml/min, motor rpm of 5000rpm, rotate speed from 0.1 to 100rpm, and tube size of 2mm inner diameter x 4mm outer diameter. With Snap-in type design, it is easy to remove the pump head, which is very convenient for pump tube replacement and cleaning.

1.1.2.1.3 Water Pumps

The water pumps are used for pumping water in and out of the hydroponic reservoir and for water circulation. A high flow rate is needed to facilitate transfer of water. Figure 1.1.2.1.3 shows the Submersible Water Pump which is used for the system.

Figure 1.1.2.1.3 Submersible Water Pump

The submersible water pump has an Input voltage of 12volts DC, 3 meters maximum head, 4.2W power consumption, 30000 hours lifespan. This pump is a brushless direct current motor (BLDC) that works smoothly and steadily, with low noise that is about 40decibels (dB), and permanent magnetic rotor and ceramic shaft to ensure excellent performance. The submersible water pump has a maximum flow rate of 240 L/H which is enough to supply the need for this project. A tube is connected to the pump for transferring the water. To note, the pump has low consumption, high efficiency and long working life. The submersible water pump has two terminals: positive and negative connections.

1.1.2.1.4 Thermoelectric Peltier Cooler

The purpose of the Thermoelectric Peltier Cooler is for cooling the hydroponic solution in the hydroponic reservoir. The Thermoelectric Peltier Cooler turns on when the temperature reading is above the set temperature, while it turns off when the temperature reading is below the set temperature.

The Thermoelectric Peltier Cooler comes with the TEC1-12706 Peltier, heatsink and fans. Connect the wires to a power supply to 12V adapter. The 12V/6A Peltier module is mounted onto a 12V heatsink and fan assembly with thermal paste and an aluminum clamp. Simply connect to 12V 6A+ power supply and the frost will appear on the aluminum plate within a minute. The Thermoelectric Peltier Cooler has a power consumption of 50 watts. Figure 1.1.2.1.4.1 below shows the Assembled Thermoelectric Peltier Cooler.

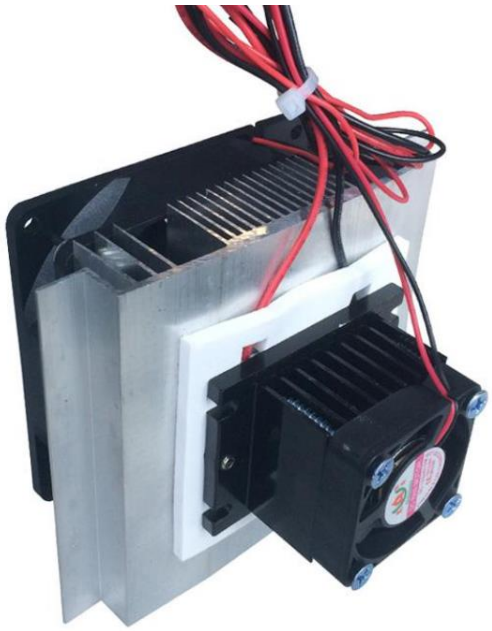


Figure 1.1.2.1.4.1 Assembled Thermoelectric Peltier Cooler

The TEC1-12706 Peltier consists mainly of semiconductor material sandwiched between two ceramic plates. It can be used either for heating or for cooling. In this project, it is used for cooling. Figure 1.1.2.1.4.2 below shows the TEC1-12706 Peltier.

Figure 1.1.2.1.4.2 TEC1-12706 Peltier

1.1.2.1.5 Grow Lights

Figure 1.1.2.1.5 shows the Light-Emitting Diodes (LED) Strip Light Full Spectrum 5M DC12V 5050 LED Grow Light. This is added to act as an alternative sunlight for the photosynthesis of lettuces. The grow light has adhesive at the back which can be attached to anywhere. It is waterproof and is composed of three (3) red LEDs and one (1) blue LED.

Figure 1.1.2.1.5 Grow Lights

1.1.2.1.6 Power Supply

A power supply is used to power up components of the system. The main function of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load. The power supply has a built-in over-voltage, over current, and short circuit protection. The power supply has an output

voltage of 12V DC and an output current of 20A. Figure 1.1.2.1.6 shows the centralized power supply.

Figure 1.1.2.1.6 12V, 20A Power Supply

1.1.2.2 Control and Display Module

This module allows the user to view the sensor readings displayed on LCD, and use a keypad to change the target values for pH, TDS, temperature, and water level.

1.1.2.2.1 LCD

The purpose of the LCD is to display the sensor readings and the different menus selected by the user. Figure 1.1.2.2.1 below shows the inter-integrated circuit LCD.

Figure 1.1.2.2.1 20x4 LCD

This 20×4 LCD display with I2C adapter board is based on the common HD44780 parallel interface chipset. The LCD display includes an I2C adapter board based on PCF8574 I2C chip. It converts I2C serial data to parallel data for the LCD display, thus significantly reducing the number of I/O pins used on the microcontroller to send data to the LCD.

1.1.2.2.2 Membrane Keypad

The membrane keypad is being used as an input device for the user to access the menu and set target values for pH, TDS, temperature, and water level. A 4x4 membrane keypad is used for the system. Figure 1.1.2.2.2.1 below shows the 4×4 Membrane Keypad.

Figure 1.1.2.2.2.1 4x4Membrane Keypad

The 4×4 Matrix Membrane Keypad which provides a useful human interface component for a microcontroller project it is made of a thin, flexible membrane material with an adhesive back so it can be attached to nearly anything. Given the matrix arrangement, only 8 microcontroller pins (4 columns and 4 rows) are needed to scan through the pad. The membrane keypad has a dimension of 69mm x 76mm and a cable length of 85mm. The switch pad has 8 pin female headers at the end of a flat cable. Figure 1.1.2.2.2.2 shows the pin out of 4x4 membrane keypad. To detect which button is pressed, the Raspberry Pi has to send a pulse to each of the four rows of the keyboard. When a user presses a button that is connected to the line which is currently pulled high, the corresponding column is also pulled high. By decoding the combination of line and column, it is easy to determine which button got pressed.

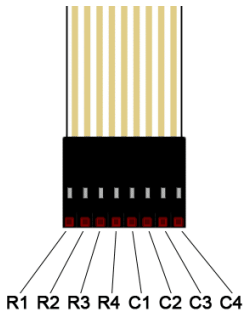


Figure 1.1.2.2.2.2 Pin Out of 4x4 Membrane Keypad

1.1.2.3 Camera Module

Figure 1.1.2.3 shows the Raspberry Pi Camera Module 1080p 720p Mini Camera 5MP Webcam Video Camera. The researchers chose this camera not solely due to affordability, but most importantly, it has a high-definition recording video of 1080p. The camera has a 5-megapixel resolution. It is compatible with the Raspberry Pi that facilitates plug and record. The purpose of this module is to capture video of lettuces as part of the monitoring activity.

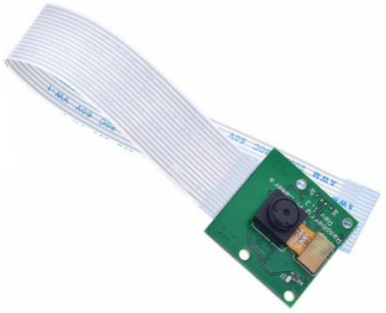


Figure 1.1.2.3 RPI Camera

1.1.2.4 Raspberry Pi 3 Model B

The Raspberry Pi 3 Model B is responsible for all the processes in the Hydroponic Maintaining and Monitoring Module. It is also responsible for hosting the web application and receiving the data from the Arduino Uno.

The Raspberry Pi 3 Model B is the third generation Raspberry Pi (RPI) with Bluetooth 4.1 and Wi-Fi. The RPI is

equipped with a 64-bit quad code processor running at 1.2GHz, which is enough for the project. This product should only be connected to an external power supply rated at 5V/2.5 A DC. The RPI has 40-pin GPIO header that is already enough for the connection of other components. It has 4 USB 2.0 ports that are also enough for connections. The RPI has CSI camera port that is needed since the project uses the Raspberry Pi camera for the live footage of lettuces for monitoring purpose. Figure 1.1.2.4 shows the image of the Raspberry Pi 3 Model B.

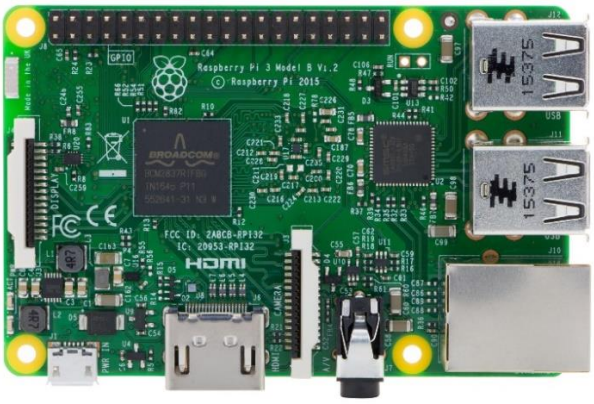


Figure 1.1.2.4 Raspberry Pi 3 Model B

1.2 Hardware Interconnection

Hardware components of the SHL are interconnected by means of different kinds of wires, USB connector cable, terminal block, and a Printed Circuit Board (PCB). The hardware interconnection of the components will be discussed from module to module. Figure 1.2.1 shows the overall hardware interconnection of components of the SHL.

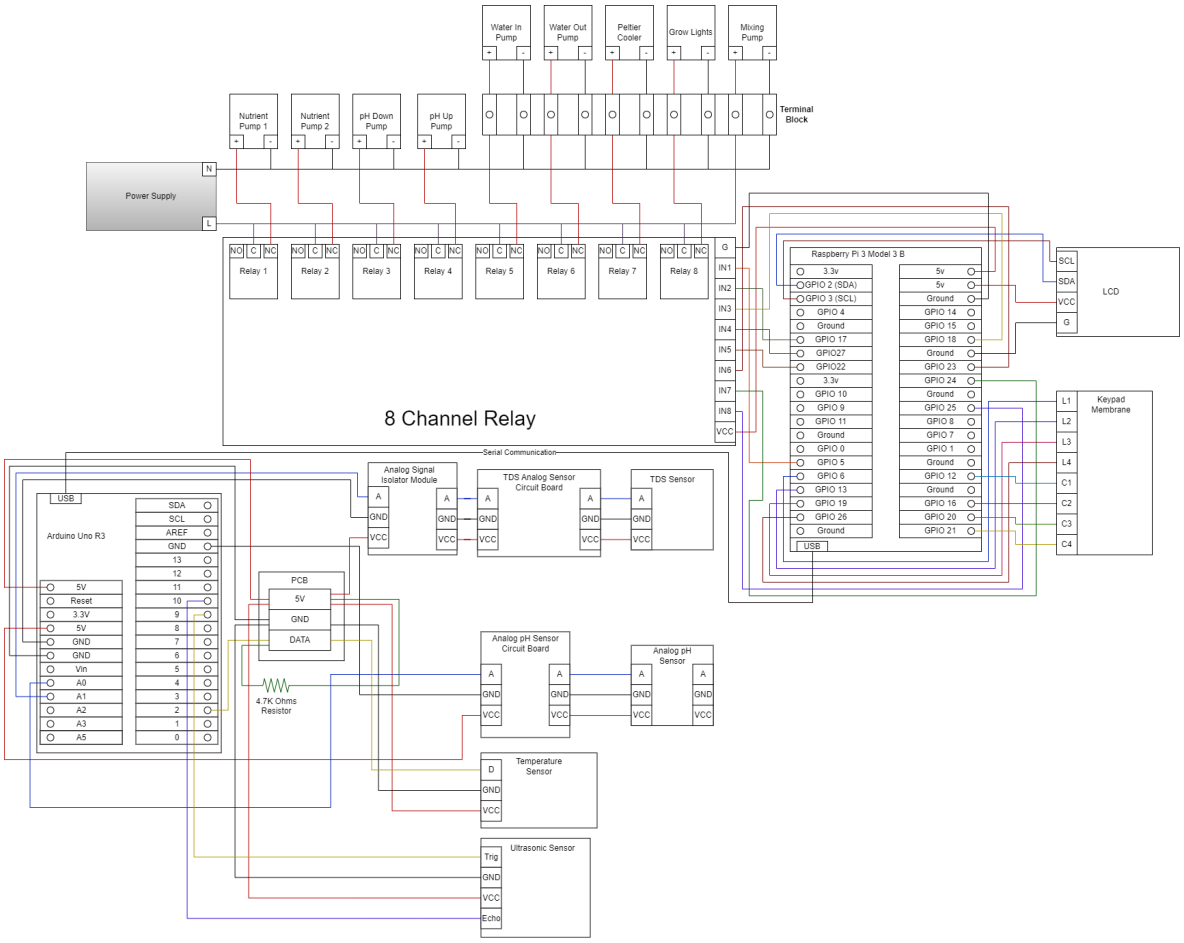


Figure 1.2.1 Overall Hardware Interconnection

A Printed Circuit Board (PCB) is used in the interconnection of the sensors since Arduino Uno has a limited 5V pin and GND pin. Male-to-male and male-to-female jumper wires are used to connect the sensors to the Arduino.

The Arduino Uno serves as the interconnection module between the pH Sensor, TDS Sensor, Temperature Sensor, and Ultrasonic Sensor. For the pH Sensor, connect the VCC pin to the 5V pin, the GND pin to the ground pin, and the signal pin to the analog A0 pin. For the TDS Sensor, connect it to the Analog Signal

Isolator, then connect the VCC pin to the 5V pin, the GND pin to the ground pin, and the signal pin to the analog A1 pin. For the Ultrasonic Sensor, connect the VCC pin to the 5V pin, the GND pin to the ground pin, then connect the trigger pin to the digital pin 9, and the echo pin to the digital pin 10. For the Temperature Sensor, a 4.7k ohms resistor should be connected to the digital pin 2 and +5V Input, then the GND pin to the ground pin. Figure 1.2.2 shows the connection of the Sensors to Arduino Interconnection.

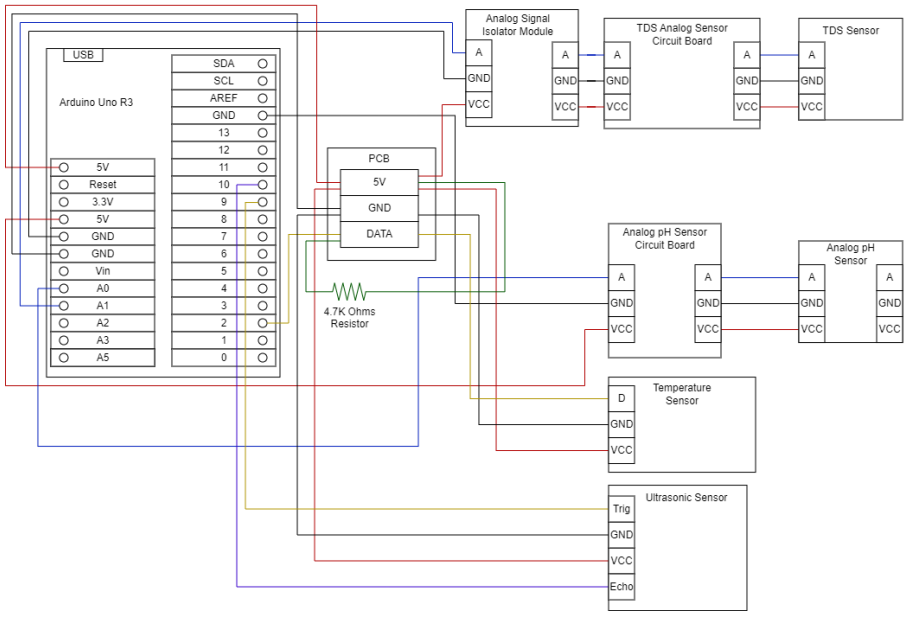


Figure 1.2.2 Sensors to Arduino Interconnection

The Arduino is thereafter connected to the Raspberry Pi via USB 2.0 cable connector. This is for the serial communication between the Arduino and the Raspberry Pi, aside from providing power to the Arduino. Figure 1.2.3 shows the USB Cable Connector.

Figure 1.2.3 USB Cable Connector

An LCD and membrane keypad is connected to the GPIO pins of the RPI. The VCC pin of the LCD is connected to the 5V pin of the RPI, the GND pin to the ground pin of the RPI, the SCL to the SCL pin of the RPI, and the SDA to the SDA pin of the RPI. The membrane keypad is connected to eight (8) GPIO pins of the RPI. Four (4) pins on the left side are for the rows and the other four (4) for the columns. Figure 1.2.4 shows the interconnection for the Display and Control Module.

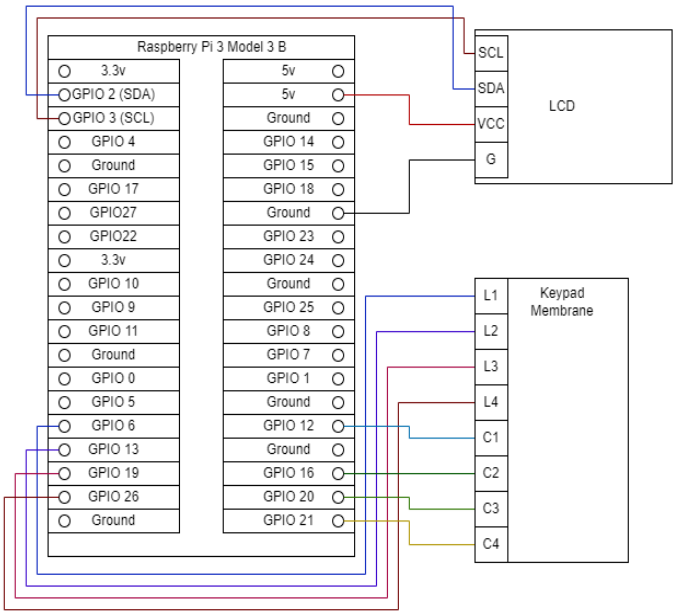


Figure 1.2.4 Interconnection of the Display and Control Module

A relay is connected to 8 GPIO pins, a 5V pin and a GND pin of the Raspberry Pi using female-to-female jumper wires as shown in Figure 1.2.5.

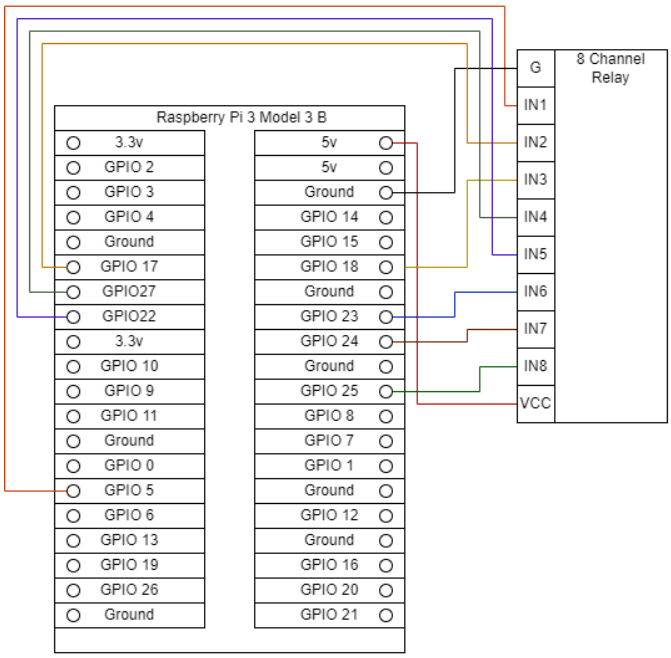


Figure 1.2.5 Relay Connection to the RPI

In this project, a terminal block is utilized for portability and more organized connectivity of the system. AWG #22 stranded wires are soldered to extend the wirings of the grow lights, thermoelectric peltier cooler and submersible pumps, then glued into the crimp terminals. Screw the crimp terminals to the terminal block.

The four (4) peristaltic pumps, two (2) submersible water pumps, thermoelectric peltier cooling module, and grow lights are connected to the relay using AWG #22 stranded wires to be powered by a 12V, 20A power supply which is shown in Figure 1.2.6. Then an additional submersible water pump is directly connected to the power supply for water circulation.

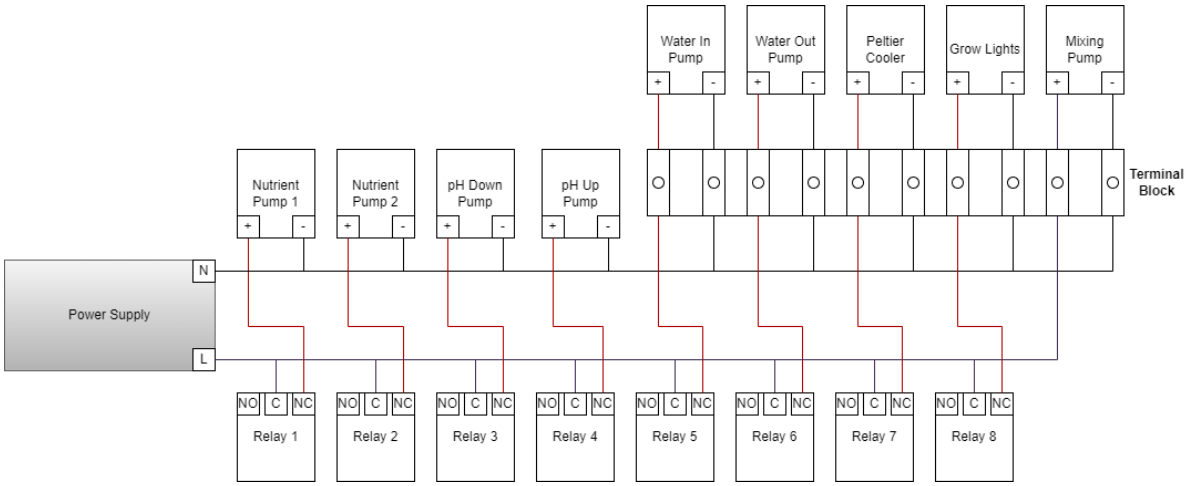


Figure 1.2.6 Relay Connection

The RPI camera is inserted into the camera slot of the RPI which is shown in Figure 1.2.7.

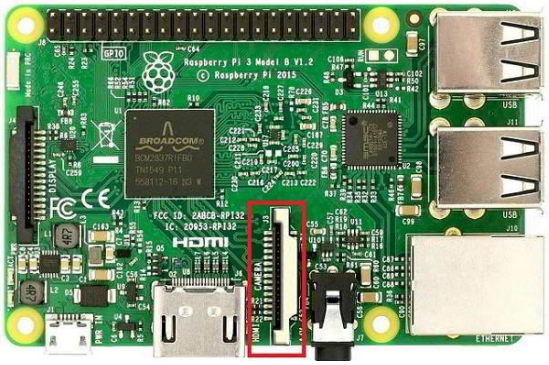


Figure 1.2.7 RPI Camera Slot

1.3 Software Design

The Arduino Integrated Development Environment (IDE) is used in coding the TSHL Arduino program, where the sensor data acquisition and sensor calibration are done. The latest version (1.8.19) of the IDE was utilized at the time of the project's implementation stage.

Thonny (IDE) is used for programming the Raspberry Pi wherein the programming language used is Python 3. And Geany (IDE) is used for programming the Web server wherein the programming language used is PHP and the web server used is Apache.

1.3.1 Arduino Software Modules

1.3.1.1 TDS Reading Module

The TDS Reading Module is responsible for calculating and obtaining the TDS reading of the hydroponic solution using an Analog TDS Sensor.

1.3.1.1.1 Libraries and Variables for TDS Reading Module

The TDS Reading Module requires these libraries and variables to function. Listing 1a line 1 includes the <*EEPROM.h*>. This library is used to ensure that the TDS calibration data are kept even after the Arduino Board is powered off. In line 2, the *"GravityTDS.h"* is included. This library is for calibration and obtaining the TDS reading of the Analog TDS Sensor. The TDS Sensor utilizes the A1 pin of the Arduino Uno. In line 3, the A1 pin is defined as *TdsSensorPin*. And in line 4, the class *GravityTDS* is initialized.

```
1 #include <EEPROM.h>
2 #include "GravityTDS.h"
3 #define TdsSensorPin A1
4 GravityTDS gravityTds;
```

Listing 1a Libraries and Variables Needed for TDS Reading Module

1.3.1.1.2 Initializing the TDS Reading Module

Before the TDS Sensor can operate, it needs to be initialized. In listing 1b line 39, the *setPin* uses the *TdsSensorPin* for the library to know what pin is being utilized. Line 40 gives the voltage reference. Since the TDS Sensor uses 5V, the reference voltage is 5.0. Line 41 sets the Analog to Digital Conversion range in which 1024 is used, and line 42 begins the initialization.

```
39 gravityTds.setPin(TdsSensorPin);
40 gravityTds.setAref(5.0);
41 gravityTds.setAdcRange(1024);
42 gravityTds.begin();
```

Listing 1b Code Snippet of Initializing the TDS Reading Module

1.3.1.1.3 Obtaining the TDS Reading

The TDS reading can be obtained using the library in listing 1c line 72. The library is used to calculate the TDS reading. And in line 73, the TDS reading is returned.

```
72     gravityTds.update();
73     tdsValue = gravityTds.getTdsValue();
```

Listing 1c Code Snippet of Obtaining the TDS Reading

1.3.1.2 pH Reading Module

This module is responsible for calculating and obtaining the pH reading of the hydroponic solution using an Analog pH Sensor.

1.3.1.2.1 Initializing Variables for the pH Sensor

The pH sensor also utilizes the *<EEPROM.h>* library. In listing 2a lines 13 to 16, *EEPROM_write* and *EEPROM_read* address are defined. This is where the pH calibration data are saved. Lines 17 to 19 define the *BufferLength* and *BufferIndex* to store the serial command. Lines 20 to 22 are for averaging of the pH readings. Lines 23 to 26 are for calibration. While line 27 defines A0 as the *SensorPin*, as the pH Sensor utilizes the A0 pin of the Arduino Uno; and, line 28 defines the voltage reference in mV for the pH sensor wherein 5000mV is used.

```
13 #define EEPROM_write(address, p) {int i = 0; byte *pp =
14 (byte*)&(p);for(; i < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}
15 #define EEPROM_read(address, p) {int i = 0; byte *pp =
16 (byte*)&(p);for(; i < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}
17 #define ReceivedBufferLength 20
18 char receivedBuffer[ReceivedBufferLength+1];
19 byte receivedBufferIndex = 0;
20 #define SCOUNT 30
21 int analogBuffer[SCOUNT];
22 int analogBufferIndex = 0;
23 #define SlopeValueAddress 0
24 #define InterceptValueAddress (SlopeValueAddress+4)
25 float slopeValue, interceptValue, averageVoltage;
26 boolean enterCalibrationFlag = 0;
27 #define SensorPin A0
28 #define VREF 5000
```

Listing 2a Code Snippet of Initializing Variables for the pH Sensor

1.3.1.2.2 Obtaining the Voltage Reading of the pH Sensor

The voltage reading of the pH Sensor is used to calculate for the pH reading. In listing 2b lines 100 to 105, the voltage is read and stored into the array and increments *analogBufferIndex* every 40 milliseconds. And in lines 106 to 109, once the *analogBufferIndex* is equal to the SCOUNT, the median filtering system function is used to get a smoother voltage reading.

```
100 static unsigned long sampleTimepoint = millis();
101  if(millis()-sampleTimepoint>40U)
102  {
103  sampleTimepoint = millis();
104  analogBuffer[analogBufferIndex] = analogRead(SensorPin)/1024.0*VREF;
105  analogBufferIndex++;
106  if(analogBufferIndex == SCOUNT)
107  analogBufferIndex = 0;
108  averageVoltage = getMedianNum(analogBuffer,SCOUNT);
109  }
```

Listing 2b Code Snippet of Obtaining the Voltage Reading of the pH

1.3.1.2.3 Median Filtering System

In order to get a smoother reading, a median filtering system is used in listing 2c lines 253 to 257. The *bArray* contents are stored into the *bTab* array, then lines 258 to 270 rearrange the Array in ascending order. Lines 271 to 272 equate *bTemp* to the middle number in the *bTab* array, if *iFilterLen* is odd; while lines 273 to 274 equate *bTemp* to the average of the 2 middle values in the *bTemp* array, if *iFilterLen* is even. And line 275, returns *bTemp*.

```
251 int getMedianNum(int bArray[], int iFilterLen)
252 {
253  int bTab[iFilterLen];
254  for (byte i = 0; i<iFilterLen; i++)
255  {
256  bTab[i] = bArray[i];
257  }
258  int i, j, bTemp;
259  for (j = 0; j < iFilterLen - 1; j++)
260  {
261  for (i = 0; i < iFilterLen - j - 1; i++)
262  {
263  if (bTab[i] > bTab[i + 1])
264  {
265  bTemp = bTab[i];
266  bTab[i] = bTab[i + 1];
267  bTab[i + 1] = bTemp;
268  }
269  }
270  }
271  if ((iFilterLen & 1) > 0)
272  bTemp = bTab[(iFilterLen - 1) / 2];
273  else
274  bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) /2;
275  return bTemp;
276 }
```

Listing 2c Code Snippet of Median Filtering System

1.3.1.2.4 Getting the Calibration Data from EEPROM

In listing 2d lines 279 to 280, the *slopeValue* and the *interceptValue* are retrieved. In lines 281 to 286, if there is no saved *interceptValue*, the program equates to 3.5 as its slope value; while in lines 287 to 292, if there is no saved *slopeValue*, the program equates to 0 as its intercept value.

```
277 void readCharacteristicValues()
278 {
279  EEPROM_read(SlopeValueAddress, slopeValue);
280  EEPROM_read(InterceptValueAddress, interceptValue);
281  if(EEPROM.read(SlopeValueAddress)==0xFF && EEPROM.read(SlopeValueAddress+1)==0xFF
282  && EEPROM.read(SlopeValueAddress+2)==0xFF && EEPROM.read(SlopeValueAddress+3)==0xFF)
283  {
284  slopeValue = 3.5;
285  EEPROM_write(SlopeValueAddress, slopeValue);
286  }
287  if(EEPROM.read(InterceptValueAddress)==0xFF && EEPROM.read(InterceptValueAddress+1)==0xFF
288  && EEPROM.read(InterceptValueAddress+2)==0xFF && EEPROM.read(InterceptValueAddress+3)==0xFF)
289  {
290  interceptValue = 0;
291  EEPROM_write(InterceptValueAddress, interceptValue);
292  }
293 }
```

Listing 2d Code Snippet of Getting Calibration Data from EEPROM

1.3.1.2.5 pH Calibration

In order to get accurate readings, the pH Sensor needs to be calibrated. The pH calibration software needs specific commands in the serial monitor in order to start the calibration. In listing 2e line 64, variable *modeIndex* is declared as an indicator in the calibration process. Lines 165 to 166, if the system receives “CALIBRATION”, the *modeIndex* is equal to 1 which starts the pH calibration. In lines 167 to 168, if “EXIT” is received, *modeIndex* is equal to 4 which exits the pH calibration mode. In lines 169 to 170, if “ACID:” is received, *modeIndex* is equal to 2 which is acidic solution calibration. And in lines 171 to 172, if “ALKALI:” is received, *modeIndex* is equal to 3 which is alkaline solution calibration.

```
162 byte uartParse()
163 {
164  byte modeIndex = 0;
165  if(strstr(receivedBuffer, "CALIBRATION") != NULL)
166  modeIndex = 1;
167  else if(strstr(receivedBuffer, "EXIT") != NULL)
168  modeIndex = 4;
169  else if(strstr(receivedBuffer, "ACID:") != NULL)
170 modeIndex = 2;
171  else if(strstr(receivedBuffer, "ALKALI:") != NULL)
172  modeIndex = 3;
173  return modeIndex;
174 }
```

Listing 2e Code Snippet of pH Calibration Commands

In order to calculate for the slope and intercept values, four values must be acquired. The *acidValue* is the

pH of the acidic solution used; *alkaliValue* is the pH of the alkaline solution used; *acidVoltage* is the voltage reading of the acidic solution; and, *alkaliVoltage* is the voltage reading of the alkaline solution. In listing 2f lines 177 to 178, the *receivedBufferPtr*, *acidCalibrationFinish*, and *alkaliCalibrationFinish* are declared. These are used to know what step is completed in the calibration process. In lines 179 to 180, *acidValue, alkaliValue, acidVoltage,* and *alkaliVoltage* are declared. These 4 values are obtained during the acid and alkali calibration.

```
175 void phCalibration(byte mode)
176 {
177  char *receivedBufferPtr;
178  static byte acidCalibrationFinish = 0, alkaliCalibrationFinish = 0;
179  static float acidValue,alkaliValue;
180  static float acidVoltage,alkaliVoltage;
181  float acidValueTemp,alkaliValueTemp,newSlopeValue,newInterceptValue;
```

Listing 2f Code Snippet of pH Calibration Initialization

In listing 2g lines 190 to 193, if the *recievedBuffer* receives “CALIBRATION”, the *enterCalibrationFlag* is equal to 1, *acidCalibrationFinish* is equal to 0, and *alkaliCalibrationFinish* is equal to 0.

```
189  case 1:
190  receivedBufferPtr=strstr(receivedBuffer, "CALIBRATION");
191  enterCalibrationFlag = 1;
192  acidCalibrationFinish = 0;
193  alkaliCalibrationFinish = 0;
194  Serial.println(F("Enter Calibration Mode"));
195  break;
```

Listing 2g Code Snippet of Entering Calibration Mode

In this function, *acidValue* and *acidVoltage* are obtained. In Listing 2h lines 198 to 202, if the *enterCalibrationFlag* is equal to 1 and the *receivedBuffer* receives “ACID:”, the system gets the float in the *receivedBuffer* string and equates the *acidValueTemp* to the float value received. In lines 203 to 213, if the *acidValue* is greater than 3 and less than 5, the *acidValueTemp* is saved to *acidValue* and *averageVoltage* is saved to *acidVoltage*; and the *acidCalibrationFinish* equates to 1, otherwise the *acidCalibrationFinish* equates to 0.

```
197 case 2:
198 if(enterCalibrationFlag)
199 {
200 receivedBufferPtr=strstr(receivedBuffer, "ACID:");
201 receivedBufferPtr+=strlen("ACID:");
202 acidValueTemp = strtod(receivedBufferPtr,NULL);
203 if((acidValueTemp>3)&&(acidValueTemp<5))
204 {
205 acidValue = acidValueTemp;
206 acidVoltage = averageVoltage/1000.0; // mV ->V
207 acidCalibrationFinish = 1;
208 Serial.println(F("Acid Calibration Successful"));
209 }else {
210 acidCalibrationFinish = 0;
211 Serial.println(F("Acid Value Error"));
212 }
213 }
214 break;
```

Listing 2h Code Snippet of Acid Calibration

In this function, *alkaliValue* and *alkaliVoltage* are obtained. In listing 2i lines 217 to 221, if the *enterCalibrationFlag* is equal to 1 and the *receivedBuffer*

receives “ALKALI:”, the system receives the float in the *receivedBuffer* string and equates the *alkaliValueTemp* to the float received. In lines 222 to 233, if the *alkaliValue* is greater than 8 and less than 11, the *alkaliValueTemp* is saved to *alkaliValue* and *averageVoltage* is saved to *alkaliVoltage*; and the *alkaliCalibrationFinish* equates to 1, otherwise the *alkaliCalibrationFinish* equates to 0.

```
216 case 3:
217 if(enterCalibrationFlag)
218 {
219 receivedBufferPtr=strstr(receivedBuffer, "ALKALI:");
220 receivedBufferPtr+=strlen("ALKALI:");
221 alkaliValueTemp = strtod(receivedBufferPtr,NULL);
222 if((alkaliValueTemp>8)&&(alkaliValueTemp<11))
223 {
224 alkaliValue = alkaliValueTemp;
225 alkaliVoltage = averageVoltage/1000.0;
226 alkaliCalibrationFinish = 1;
227 Serial.println(F("Alkali Calibration Successful")
228 );
229 }else{
230 alkaliCalibrationFinish = 0;
231 Serial.println(F("Alkali Value Error"));
232 }
233 }
234 break;
```

Listing 2i Code Snippet of Alkali Calibration

With the four values needed to compute for slope and intercept values, the calibration can be completed. In listing 2j lines 236 to 244, if the *entercalibrationFlag*, *acidCalibrationFinish*, and *alkaliCalibrationFinish* are equal to 1, the *newslopeValue* will be equal to (*acidValue*-*alkaliValue*) / (*acidVoltage* – *alkaliVoltage*) and the *newIntercepValue* will be equal to *acidValue* –

(*slopeValue*acidVoltage*). The *slopeValue* and *interceptValue* are saved on the EEPROM. In lines 249 to 251, the flags reset back to 0.

```
235  case 4:
236  if(enterCalibrationFlag)
237  {
238  if(acidCalibrationFinish && alkaliCalibrationFinish)
239  {
240  newSlopeValue = (acidValue-alkaliValue)/(acidVoltage
241 - alkaliVoltage);
242  EEPROM_write(SlopeValueAddress, newSlopeValue);
243  newInterceptValue = acidValue - (slopeValue*acidVoltage);
244  EEPROM_write(InterceptValueAddress, newInterceptValue);
245  Serial.print(F("Calibration Successful"));
246  }
247  else Serial.print(F("Calibration Failed"));
248  Serial.println(F(",Exit Calibration Mode"));
249  acidCalibrationFinish = 0;
250  alkaliCalibrationFinish = 0;
251  enterCalibrationFlag = 0;
252  }
253  break;
254  }
255 }
```

Listing 2j Code Snippet of Completing pH Calibration

1.3.1.2.6 Obtaining the pH Reading

The pH Value can be obtained by using the formula *averageVoltage/1000.0*slopeValue+interceptValue*. This is shown in listing 2k line 74.

```
74    pHValue = averageVoltage/1000.0*slopeValue+interceptValue
```

Listing 2k Code Snippet of Obtaining the pH Reading

1.3.1.3 Water Level Reading Module

This module is responsible for calculating the water level reading of the hydroponic reservoir using an Ultrasonic Sensor by measuring the distance of the water surface from the sensor.

1.3.1.3.1 Initializing Variables for the Water Level Reading

In listing 3a line 27, integer *distanceavg* is declared. This is where the average reading is stored. Line 28 declares the integer array *distancearray[16]* where the 16 readings are stored. Line 29 declares integer *cdistance*, the counter for the array. Line 30 integer *distancesum* is declared, where the sum of the readings is stored. Line 31 declares integer *distancecc* to divide the distance sum to get the integer *distanceavg*. The Ultrasonic Sensor utilizes digital pin 9 and pin 10 of the Arduino Uno. In lines 32 to 33, the *trigPin* is declared as pin 9 and *echoPin* as pin 10. In line 35, long integer *duration* is declared to store the ping time. Data type long is used to enable storing larger numbers. And line 36 declares integer distance, where the distance reading is stored.

```
27 int distanceavg =0;
28 int distancearray[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
29 int cdistance =0;
30 int distancesum =0;
31 int distancecc=1; // for averaging distance
32 const int trigPin = 9;
33 const int echoPin = 10;
34 // defines variables
35 long duration;
36 int distance;
```

Listing 3a Code Snippet of Initializing Variables for the Water Level Reading

1.3.1.3.2 Obtaining Distance Reading

In listing 3b lines 79 to 83, the sensor begins by giving low signal to the *trigPin* for 2 microseconds,

followed by a high signal for 10 microseconds then low signal again. In line 84, the duration is equal to the time it takes for the *echoPin* to go from low to high in microseconds. Line 85 calculates the distance using the formula *duration * 0.034 / 2.*

```
79    digitalWrite(trigPin, LOW);
80    delayMicroseconds(2);
81    digitalWrite(trigPin, HIGH);
82    delayMicroseconds(10);
83    digitalWrite(trigPin, LOW);
84    duration = pulseIn(echoPin, HIGH);
85    distance = duration * 0.034 / 2;
```

Listing 3b Code Snippet of Obtaining Distance Reading

1.3.1.3.3 Smoothening Data with Running Average

The running average of the distance reading is used in order to have a more stable distance reading. In listing 3c line 86, the distance reading is stored in the array. Then in lines 87 to 90, the sum of the *distancearray* is stored in distance sum. Line 91 gets the *distanceavg* by dividing *distancesum* by *distancecc*. Line 92 resets *distancesum* to 0, and line 93 increments the counter for *cdistance*. Lines 94 to 97 increase the counter of *distancecc* if the *distancecc* is less than or equal to 15, to match the number of readings stored in *distancearray*. In lines 98 to 101, if the *cdistance* is greater than or equal to

16, the *cdistance* resets to always overwrite the oldest reading in the *distancearray*.

```
86    distancearray[cdistance] = distance;
87    for(int dc=0;dc<16;dc++)
88    {
89    distancesum=distancesum+distancearray[dc];
90    }
91    distanceavg = distancesum/distancecc;
92    distancesum = 0;
93    cdistance++;
94    if(distancecc<=15)
95    {
96    distancecc++;
97    }
98    if(cdistance>=16)
99    {
100   cdistance=0;
101   }
102   delay(300);
```

Listing 3c Code Snippet of Smoothening Data with Running Average

1.3.1.4 Temperature Reading Module

This module is responsible for receiving the hydroponic solution temperature reading using the Temperature Sensor.

1.3.1.4.1 Libraries and Variables for Temperature Reading

The Temperature Reading Module requires two libraries to operate. Listing 4a lines 6 and 7 include the libraries *<Onewire.h>* and *<DallasTemperature.h>.* Lines 8 to 10 initialize the Temperature Reading Module and line 5 defines the digital pin for the Temperature Sensor to use.

```
5  #define ONE_WIRE_BUS 2
6  #include <OneWire.h>
7  #include <DallasTemperature.h>
8  OneWire oneWire(ONE_WIRE_BUS);
9  DallasTemperature sensors(&oneWire);
10 DeviceAddress insideThermometer;
```

Listing 4a Libraries and Variables Needed for TDS Reading Module

1.3.1.4.2 Obtaining Temperature Reading

The temperature can be retrieved using this function. In listing 4b line 59, the temperature is retrieved from the sensor. And in line 60, the temperature is printed.

```
57 void printTemperature(DeviceAddress deviceAddress)
58 {
59   float tempC = sensors.getTempC(deviceAddress);
60   Serial.print(tempC);
61 }
```

Listing 4b Code Snippet of Obtaining Temperature Reading

1.3.1.5 Serial Communication Module

The sensor data needs to be sent to the Raspberry Pi. This can be done through serial communication.

1.3.1.5.1 Printing Sensor Data for Serial Communication

The sensor readings of TDS, distance, temperature, and pH value need to be separated with a comma, so it can be retrieved by the RPI program. Listing 5 lines 125 to 128 print the TDS reading and distance reading; line 129 calls the function that prints the temperature reading; and, line 131 prints the pH reading.

```
125  Serial.print(tdsValue,0);
126  Serial.print(",");
127  Serial.print(distance);
128  Serial.print(",");
129  printTemperature(insideThermometer);
130  Serial.print(",");
131  Serial.println(pHValue);
```

Listing 5 Code Snippet of Printing Sensor Data for Serial

1.3.2 Raspberry Pi Software Modules

1.3.2.1 Hydroponics Maintaining Module

This module is responsible for handling the dosing, irrigation, controlling grow light, and cooling of the hydroponics using the sensor data and set parameters.

1.3.2.1.1 Receiving Sensor Data from Arduino

In order to know the condition of the hydroponic solution, the sensor data must first be received and processed to usable data. The Arduino is connected to the Raspberry Pi using a USB connector cable. The Arduino sends the sensor data using serial communication. The data received by the Raspberry Pi is a string that consists of all the sensor readings which are split by commas. In listing 6 lines 1069 to 1070, the data is split to pH, TDS, temperature, and distance. In line 1066, the container height is defined with the height of the hydroponic reservoir used in centimeter which is equal to 22; and, in line 1067, the container height is subtracted by 2. This is

to give allowance and avoid the risk of overflowing of the hydroponic solution. In line 1071, the distance is subtracted from the container height to get the water level of the hydroponic reservoir. And in line 1072, the water level is converted to percentage form.

```
1066    containerheight = 22
1067    subcontainerheight = containerheight - 2
1068    if ser.in_waiting > 0:
1069        line = ser.readline().decode('utf-8').rstrip()
1070        (tds,ultrason,temp,ph) = [float(s) for s in line.split(',')]
1071        distance = containerheight - ultrason
1072        distance = distance/subcontainerheight * 100
```

Listing 6 Code Snippet of Receiving Sensor Data from Arduino

1.3.2.1.2 Obtaining the Set Values

The set values data are stored in text files, so the program and the web application can both have access. In listing 7 lines 1121 to 1125, the set pH is retrieved; in lines 1227 to 1231, the set TDS is retrieved; in lines 1233 to 1237, the set temperature is retrieved; in lines 1239 to 1241, the set water level is retrieved; in lines 1245 to 1247, the current time is retrieved; and, in lines 1249 to 1259, the set grow light schedule is retrieved.

```
1221        fetch_1 = open("/var/www/html/4_set_phlvl.txt","r")
1222        fetched_data_1 = float(fetch_1.read())
1223        fetch_1.close()
1224
1225        setph = fetched_data_1
1226
1227        fetch_2 = open("/var/www/html/1_set_tds.txt","r")
1228        fetched_data_2 = float(fetch_2.read())
1229        fetch_2.close()
1230
1231        settds = fetched_data_2
1232
1233        fetch_3 = open("/var/www/html/3_set_temperature.txt","r")
1234        fetched_data_3 = float(fetch_3.read())
1235        fetch_3.close()
1236
1237        settemperature = fetched_data_3
1238
1239        fetch_4 = open("/var/www/html/2_set_waterlvl.txt","r")
1240        fetched_data_4 = float(fetch_4.read())
1241        fetch_4.close()
1242
1243        setdistance = fetched_data_4
1244
1245        current_time2 = datetime.datetime.now()
1246        dt_string2 = current_time2.strftime("%H%M")
1247        growlight_current = int(dt_string2)
1248
1249        fetch_5 = open("/var/www/html/growlights_start.txt","r")
1250        fetched_data_5 = int(fetch_5.read())
1251        fetch_5.close()
1252
1253        growlight_start = fetched_data_5
1254
1255        fetch_6 = open("/var/www/html/growlights_end.txt","r")
1256        fetched_data_6 = int(fetch_6.read())
1257        fetch_6.close()
1258
1259        growlight_end = fetched_data_6
```

Listing 7 Code Snippet of Obtaining the Set Values

1.3.2.1.3 Maintaining Water Level and Temperature

The program compares the water level reading to the set water level, and the temperature reading to the set temperature; to determine if the submersible pump and Thermoelectric Peltier Cooler will turn on or off.

Listing 8 lines 1235 to 1236 are for maintaining the water level. If the water level reading is below the set water level, relay 5 turns on which is hooked to the water pump in the water reservoir. Lines 1238 to 1239 are for maintaining the water temperature. If the set temperature

is above the temperature reading, relay 7 turns on which is hooked to the Thermoelectric Peltier Cooler.

```
1235        if distance < distancemin:
1236            GPIO.output(relay5, GPIO.LOW)
1237
1238        if temp > tempmax:
1239            GPIO.output(relay7, GPIO.LOW)
```

Listing 8 Code Snippet of Maintaining Water Level and Temperature

1.3.2.1.4 Setting Up Thresholds for pH and TDS

Thresholds are set up in order to avoid indefinite dosing. In listing 9 lines 1220 and 1221, a threshold is set to +0.15 for the maximum pH and -0.15 for the minimum pH. In lines 1222 to 1223, a threshold is set to +25 for the maximum TDS and -25 for the minimum TDS.

```
1220        phmin = setph - 0.15
1221        phmax = setph + 0.15
1222        tdsmin = settds - 25
1223        tdsmax = settds + 25
```

Listing 9 Code Snippet of Setting Up Thresholds for pH and TDS

1.3.2.1.5 Calculating the Dosing Duration for the pH and TDS

In order to get right amount of doses, the duration as to how long the dosing pumps needs to turn on is acquired. The dosing duration highly depends on how far the current reading is to the set reading. In listing 10a line 1297, the set TDS is subtracted from the TDS reading to get *tdsmaxdif* to calculate for the duration of the water out pump. In line 1298, the TDS reading is subtracted from the set TDS to get *tdsmindif* to calculate for the duration of the

nutrient solution dosing pumps. In line 1299, the set pH is subtracted from pH reading to get *phmaxdif* to calculate for the pH down solution dosing duration. In line 1300, pH reading is subtracted from set pH to get *phmindif* to calculate for the duration of the pH up solution dosing.

```
1297        tdsmaxdif = tds - settds
1298        tdsmindif = settds - tds
1299        phmaxdif = ph - setph
1300        phmindif = setph - ph
```

Listing 10a Getting the Difference of Set Parameters and Readings

The dosing duration depends on (1) the concentration of the nutrient solution used, (2) concentration of the pH buffer solution used, (3) the size of the Hydroponic Reservoir, (4) the current water level reading, and (5) the difference of the current readings with the set values. In listing 10b line 1310, the *tdsmindif* is divided by 40 to get *tdsmindifd*. The divisor can be changed depending on how concentrated the nutrient solution is. The more concentrated the nutrient solution, the bigger the divisor needs to be. In line 1311, the *tdsmaxdif* is divided by 25 to get *tdsmaxdifd*. The divisor can be changed depending on the size of the reservoir. The bigger the reservoir, the lower the divisor needs to be. In line 1312, the distance is divided by 40 to get distanced. The divisor can be changed depending on how big the

reservoir is. The bigger the reservoir, the bigger the divisor should be. In line 1313, *phmaxdif* is multiplied by 2 to get *phmaxdifd* and in line 1314, *phmindif* is multiplied by 2 to get *phmindif*. The multiplier can be changed depending on how concentrated the pH buffer solutions are. The more concentrated, the lower the number. Finally, the dosing durations can be acquired. Line 1315 gets the *tdsminc* by multiplying distanced and *tdsmindifd*, wherein *tdsminc* is used as a counter for nutrient solution dosing. Line 1316 gets the *tdsmaxc* by multiplying distanced and *tdsmaxdifd*, wherein *tdsmaxc* is used as a counter for pumping out hydroponic solution. Line 1317 gets the *phmaxc* by multiplying distanced and *phmaxdifd*. Line 1318 acquires *phminc* by multiplying distanced and *phmindifd*. The *phmaxc* and *phminc* are the counters for dosing of pH buffer up and down solutions.

```
1310        tdsmindifd = tdsmindif/40
1311        tdsmaxdifd = tdsmaxdif/25
1312        distanced = distance/40
1313        phmaxdifd = phmaxdif * 2
1314        phmindifd = phmindif * 2
1315        tdsminc = distanced * tdsmindifd
1316        tdsmaxc = distanced * tdsmaxdifd
1317        phmaxc = distanced * phmaxdifd
1318        phminc = distanced * phmindifd
```

Listing 10b Code Snippet of Calculating for the Dosing Duration

1.3.2.1.6 Dosing Nutrient Solutions and pH Buffer Solutions

To ensure correct dosing, the system is given time before it starts to dose. This gives the sensors enough

time to stabilize. It also gives time for hydroponic solution to mix properly. In listing 11 line 1321, an initial counter of 240 ticks needs to finish before the dosing of pH and nutrient solution starts.

The first 240 ticks of the loop is for the TDS dosing. In lines 1323 to 1326, if the *counter_alternate* is greater than or equal to *tdsminc* and if current TDS reading is less than the TDS minimum, relay 1 and 2 turn on which are hooked to the peristaltic dosing pumps that dose nutrient solution to the hydroponic reservoir resulting in an increase of the TDS in the hydroponic solution. In lines 1327 to 1329, if the *counter_alternate* is less than or equal to *tdsmaxc* and if current TDS reading is greater than the TDS maximum, relay 6 turns on which is hooked to the submersible pump that flows water out of the hydroponic reservoir, so that the water gets replaced by fresh water resulting in the drop in TDS level.

The 240 to 480 ticks in the loop are for the pH dosing. In lines 1330 to 1332, if the *counter_alternate* is greater than or equal to 240 and the *counter_alternate* is less than or equal to 240+*phminc*, and if current pH reading is less than the pH minimum, relay 4 turns on

which is hooked to the peristaltic dosing pump that doses pH up buffer solution into the hydroponic reservoir resulting in an increase of the pH of the hydroponic solution. In lines 1333 and 1334, if the *counter_alternate* is greater than or equal to 240 and the *counter_alternate* is less than or equal to 240+*phmaxc*, and if current pH reading is greater than the pH maximum, relay 3 turns on which is hooked to the peristaltic dosing pump that doses pH down buffer solution into the Hydroponic Reservoir resulting in a decrease of the pH of the hydroponic solution. Lines 1338 increments the *counter_alternate*; line 1342 increments *counter_setup*; and, in lines 1339 to 1340, the program resets the counter for the cycle to continue.

```
1321        if counter_setup >= 240:
1322            distancemin = setdistance
1323            if counter_alternate <=tdsminc:
1324                if tds < tdsmin:
1325                    GPIO.output(relay1, GPIO.LOW)
1326                    GPIO.output(relay2, GPIO.LOW)
1327            if counter_alternate <=tdsmaxc:
1328                if tds > tdsmax:
1329                    GPIO.output(relay6, GPIO.LOW)
1330            if counter_alternate >= 240 and counter_alternate <= 240+phminc:
1331                if ph < phmin:
1332                    GPIO.output(relay4, GPIO.LOW)
1333            if counter_alternate >= 240 and counter_alternate <= 240+phmaxc:
1334                if ph > phmax:
1335                    GPIO.output(relay3, GPIO.LOW)
1336
1337                print("relays online")
1338            counter_alternate+=1
1339            if counter_alternate >= 480:
1340                counter_alternate=0
1341
1342        counter_setup+=1
```

Listing 11 Code Snippet of Dosing Nutrient and pH Buffer Solutions

1.3.2.1.7 Turning the Grow Lights on or off

The grow lights turn on or off depending on the set schedule and current time. In listing 12, if the current time is greater than the start time and less than the end time, the relay 8 turns on which is hooked to the grow lights.

```
1241        if growlight_current >= growlight_start and growlight_current < growlight_end:
1242                GPIO.output(relay8, GPIO.LOW)
```

Listing 12 Code Snippet of Turning the Grow Lights On or Off

1.3.2.2 Keypad Membrane Module

This module enables the user to change the set values using a keypad membrane.

1.3.2.2.1 Setting up the GPIO Pins for the Keypad Membrane

The keypad membrane utilizes the GPIO pins of the Raspberry Pi. In listing 13 lines 11 to 14, GPIO pins 6, 13, 19, and 26 are defined for rows, and in lines 16 to 19, the GPIO pins 12, 16, 20, and 21 are defined for columns.

```
11  L1 = 6
12  L2 = 13
13  L3 = 19
14  L4 = 26
15
16  C1 = 12
17  C2 = 16
18  C3 = 20
19  C4 = 21
```

Listing 13 Assigned GPIO pins for the Keypad Membrane

1.3.2.2.2 Setting up the Keypad Membrane

Each row and column combination will have an equivalent character. In listing 14 Lines 1058 to 1061, the equivalent character is declared.

```
1058    readLine(L1, ["1","2","3","A"])
1059    readLine(L2, ["4","5","6","B"])
1060    readLine(L3, ["7","8","9","C"])
1061    readLine(L4, ["*","0","#","D"])
```

Listing 14 Setting up the Keypad Membrane

1.3.2.2.3 Menu Selection

The Keypad Membrane button A is assigned to pH, B is assigned to TDS, C is assigned to temperature, and D is assigned to water level. In listing 15 line 100, the current character pressed is printed. In lines 101 to 105, if the character printed is A, B, C, or D, the *buttonstate* equals to the button pressed, and the *checkstate* equates to 1 to know that a menu is selected.

```
100         print(characters[0])
101
102         if checkstate == 0:
103             if characters[0] == "A" or characters[0] == "B" or characters[0] == "C" or characters[0] == "D":
104                 checkstate = 1
105                 buttonstate = (characters[0])
```

Listing 15 Code Snippet of Menu Selection

1.3.2.2.4 Storing Inputs into String

The inputs must first be stored into strings so such can be used to change values or compare passwords. In listing 16 lines 253 to 256, the program determines if password check is initiated. If initiated, the inputs are

appended into the *appended_string_password* which is used to compare passwords. And in lines 258 to 261, if the password check is not initiated, the inputs are appended into the *appended_string* which is used to change the set values.

```
253             elif password_check_ok == 0 and password_check_initiated == 1:
254                 appended_string_password = appended_string_password + characters[0]
255                 print("entering password")
256                 display.lcd_clear()
257 
258             else:
259                 appended_string = appended_string + characters[0]
260                 print("value added")
261                 display.lcd_clear()
```

Listing 16 Code Snippet of Storing Inputs into String

1.3.2.2.5 Selecting Exit or Enter

The menu needs enter and exit buttons for it to be operational. In listing 17 lines 120 to 128, if exit is pressed in which "*" is assigned, the system clears the *appended_strings* and *appended_string_password*, and resets all the flags. And in line 130, if enter is pressed in which "#" is assigned, the *password_check_initiated* equals to 1 to proceed to the password check.

```
120             elif characters[0] == "*":
121                 print("EXIT SELECTED")
122                 checkstate = 0
123                 buttonstate = ""
124                 appended_string = ""
125                 appended_string_password = ""
126                 password_check_initiated = 0
127                 lcd_already_on = 0
128                 display.lcd_clear()
129 
130             elif characters[0] == "#":
131 
132                 print("Enter selected, prompt password")
133                 display.lcd_clear()
134 
135                 if buttonstate != "" and appended_string != "":
136                     password_check_initiated = 1
```

Listing 17 Code Snippet of Selecting Exit or Enter

1.3.2.2.6 Password Check

To ensure that only authorized users can change the set values, the system asks for the password before changing the set values. The password.txt contains the regular password used by the user and masterkey.txt is the backup password kept in case the user forgets the password. In listing 18 lines 152, if the flag *password_check_initiated* is equal to 1 and the string *appended_string_password* is not empty, the password check will initiate. And lines 154 to 160 retrieve the password and masterkey from the text documents. If the string *appended_string_password* matches the password or matches the masterkey, the program then proceeds to changing the set value.

```
152             if password_check_initiated == 1 and appended_string_password != "":
153
154                 fetch = open("/var/www/html/password.txt","r")
155                 fetched_password = str(fetch.read())
156                 fetch.close()
157
158                 fetch2 = open("/var/www/html/masterkey.txt","r")
159                 masterkey = str(fetch2.read())
160                 fetch2.close()
161
162                 if appended_string_password == fetched_password or appended_string_password == masterkey:
```

Listing 18 Code Snippet of Password Check

1.3.2.2.7 Changing the Set Values

After the password check is completed, the program proceeds to changing the value, but it must first know what menu is selected using the *buttonstate*. In listing 19a lines 164 to 178, if the *buttonstate* equals to

"B", it checks if the input is within the allowed range of 1500. If it is within the allowed range, it saves the *appended_string* to the 1_set_tds.txt file. Then it clears the *appended_string* and *appended_string_password* and equates the *password_check_initiated* to 0, otherwise it just clears the *appended_string* and *appended_string_password* and equates *password_check_initiated* to 0.

```
164                 if buttonstate == "B" and appended_string != "":
165                     if int(appended_string) <= 1500:
166                         file1 = open("/var/www/html/1_set_tds.txt","w+")
167                         print("TDS Accessed")
168                         file1.write(str(appended_string))
169                         print("New SET TDS Saved")
170                         file1.close()
171                         appended_string = ""
172                         appended_string_password = ""
173                         password_check_initiated = 0
174 
175                     else:
176                         appended_string = ""
177                         appended_string_password = ""
178                         password_check_initiated = 0
```

Listing 19a Code Snippet of Changing Set TDS

In listing 19b lines 180 to 194, if the *buttonstate* equals to "D", it checks if the input is within the allowed range of 100. If it is within the allowed range, the string *appended_string* is saved in the 2_set_waterlvl.txt then it clears the *appended_string* and *appended_string_password* and equates *password_check_initiated* to 0, otherwise it just clears the *appended_string* and *appended_string_password* and equates *password_check_initiated* to 0.

```
180                     elif buttonstate == "D" and appended_string != "":
181                         if int(appended_string) <= 100:
182                             file1 = open("/var/www/html/2_set_waterlvl.txt","w+")
183                             print("WTR Accessed")
184                             file1.write(str(appended_string))
185                             print("New SET WTR Saved")
186                             file1.close()
187                             appended_string = ""
188                             appended_string_password = ""
189                             password_check_initiated = 0
190 
191                         else:
192                             appended_string = ""
193                             appended_string_password = ""
194                             password_check_initiated = 0
```

Listing 19b Code Snippet of Changing Set Water Level

In listing 19c lines 196 to 209, if the *buttonstate* equals to “C”, the *appended_string* is first converted to float then divided by 100 to allow decimal inputs. The divided quotient is then saved in the 3_set_temperature.txt, and then it clears the *appended_string* and *appended_string_password* and equates *password_check_initiated* to 0.

```
196                     elif buttonstate == "C" and appended_string != "":
197                         file1 = open("/var/www/html/3_set_temperature.txt","w+")
198                         print("TEMP Accessed")
199 
200                         #coversion
201                         x = appended_string
202                         y = float(x)/100
203 
204                         file1.write(str(y))
205                         print("New SET TEMP Saved")
206                         file1.close()
207                         appended_string = ""
208                         appended_string_password = ""
209                         password_check_initiated = 0
```

Listing 19c Code Snippet of Changing Set Temperature

In listing 19d lines 196 to 209, if the *buttonstate* equals to “A”, the *appended_string* is first converted to float then divided by 100 to allow decimal inputs. Then it checks if the input is within the allowed range of 0 to 14, the float is then saved in the 3_set_temperature.txt. Then it clears the *appended_string_password* and

appended_string and equates *password_check_initiated* to 0, otherwise it just clears the *appended_string* and *appended_string_password* and equates *password_check_initiated* to 0.

```
211                     elif buttonstate == "A" and appended_string != "":
212                         x = appended_string
213                         y = float(x)/100
214
215                         if y <= 14 and y >= 0:
216                             file1 = open("/var/www/html/4_set_phlvl.txt","w+")
217                             print("PHlvl Accessed")
218                             file1.write(str(y))
219                             print("New SET PHlvl Saved")
220                             file1.close()
221                             appended_string = ""
222                             appended_string_password = ""
223                             password_check_initiated = 0
224
225                         else:
226                             appended_string = ""
227                             appended_string_password = ""
228                             password_check_initiated = 0
```

Listing 19d Code Snippet of Changing Set pH

1.3.2.3 LCD Module

This module displays the sensor readings and the different menus selected by the user.

1.3.2.3.1 Displaying the Home Screen

The home screen displays the pH Level, TDS, Temperature, and Water Level. In listing 20 line 796, the LCD is first cleared, then in lines 797 to 800, the pH reading, TDS reading, Temperature Reading, and Water Level reading are displayed on the LCD.

```
796             display.lcd_clear()
797             display.lcd_display_string("PH Level: " + fetched_data_1, 1)
798             display.lcd_display_string("TDS     : " + fetched_data_2 + " ppm", 2)
799             display.lcd_display_string("Temp    : " + fetched_data_3 + " C", 3)
800             display.lcd_display_string("WtrLvl  : " + fetched_data_4 + " %", 4)
```

Listing 20 Code Snippet of Displaying the Home Screen

1.3.2.3.2 Displaying pH Menu

When the pH menu is selected, it needs to display the set pH and pH reading, and show the inputs made by the user on the LCD. In listing 21 line 947, the program checks if the *buttonstate* equals to A. If it is equal to A, the menu for the pH is displayed. In lines 953 to 965, the set pH, pH reading, and instruction on how to enter and exit are displayed. In lines 967 to 972, if a number is inputted, the number is displayed. In lines 974 to 979, if the user confirms, the system asks for the password and displays current password inputs.

```
947     if buttonstate == "A":
948 
949         lcd_already_on = 0
950 
951         if appended_string == "":
952 
953             fetch_1 = open("/var/www/html/4_set_phlvl.txt","r")
954             fetched_data_1 = str(fetch_1.read())
955             fetch_1.close()
956 
957             display.lcd_display_string("Set PH: " + fetched_data_1, 1)
958 
959             fetch = open("/var/www/html/4_phlvl.txt","r")
960             fetched_data = str(fetch.read())
961             fetch.close()
962 
963             display.lcd_display_string("PH: " + fetched_data, 2)
964             display.lcd_display_string("# to enter", 3)
965             display.lcd_display_string("* to exit", 4)
966 
967         elif appended_string != "" and password_check_initiated != 1:
968 
969             display.lcd_display_string("Set New PH ", 1)
970             display.lcd_display_string("Value: " + appended_string, 2)
971             display.lcd_display_string("# to enter", 3)
972             display.lcd_display_string("* to exit", 4)
973 
974         elif appended_string != "" and password_check_initiated == 1:
975 
976             display.lcd_display_string("Enter Password", 1)
977             display.lcd_display_string("Value: " + appended_string_password, 2)
978             display.lcd_display_string("# to enter", 3)
979             display.lcd_display_string("* to exit", 4)
980 
981         else:
982             None
```

Listing 21 Code Snippet of Displaying pH Menu

1.3.2.3.3 Displaying TDS Menu

When the TDS menu is selected, it needs to display the set TDS and TDS reading, and show the inputs made by the user on the LCD. In listing 22 line 830, the program checks if the *buttonstate* equals to B. If it is equal to B, the menu for the TDS is displayed. In lines 836 to 848, the set TDS, TDS reading, and instruction on how to enter and exit are displayed. In lines 850 to 854, if a number is inputted, the number is displayed. In lines 857 to 861, if the user confirms, the system asks for the password and displays current password inputs.

```
830     if buttonstate == "B":
831
832         lcd_already_on = 0
833
834         if appended_string == "":
835
836             fetch_1 = open("/var/www/html/1_set_tds.txt","r")
837             fetched_data_1 = str(fetch_1.read())
838             fetch_1.close()
839
840             display.lcd_display_string("Set TDS: " + fetched_data_1, 1)
841
842             fetch = open("/var/www/html/1_tds.txt","r")
843             fetched_data = str(fetch.read())
844             fetch.close()
845
846             display.lcd_display_string("TDS: " + fetched_data, 2)
847             display.lcd_display_string("# to enter", 3)
848             display.lcd_display_string("* to exit", 4)
849
850         elif appended_string != "" and password_check_initiated != 1:
851             display.lcd_display_string("Set New TDS ", 1)
852             display.lcd_display_string("Value: " + appended_string, 2)
853             display.lcd_display_string("# to enter", 3)
854             display.lcd_display_string("* to exit", 4)
855
856
857         elif appended_string != "" and password_check_initiated == 1:
858             display.lcd_display_string("Enter Password", 1)
859             display.lcd_display_string("Value: " + appended_string_password, 2)
860             display.lcd_display_string("# to enter", 3)
861             display.lcd_display_string("* to exit", 4)
862
863         else:
864             None
```

Listing 22 Code Snippet of Displaying TDS Menu

1.3.2.3.4 Displaying Temperature Menu

When the temperature menu is selected, it needs to display the set temperature and temperature reading, and show the inputs made by the user on the LCD. In listing 23 line 912, the program checks if the *buttonstate* equals to C. If it is equal to C, the menu for the temperature is displayed. In lines 918 to 930, the set temperature, temperature reading, and instruction on how to enter and exit are displayed. In lines 932 to 936, if a number is inputted, the number is displayed. In lines 938 to 942, if the user confirms, the system asks for the password and displays current password inputs.

```
912     if buttonstate == "C":
913 
914         lcd_already_on = 0
915 
916         if appended_string == "":
917 
918             fetch_1 = open("/var/www/html/3_set_temperature.txt","r")
919             fetched_data_1 = str(fetch_1.read())
920             fetch_1.close()
921 
922             display.lcd_display_string("Set Temp: " + fetched_data_1, 1)
923 
924             fetch = open("/var/www/html/3_temperature.txt","r")
925             fetched_data = str(fetch.read())
926             fetch.close()
927 
928             display.lcd_display_string("Temp: " + fetched_data, 2)
929             display.lcd_display_string("# to enter", 3)
930             display.lcd_display_string("* to exit", 4)
931 
932         elif appended_string != "" and password_check_initiated != 1:
933             display.lcd_display_string("Set New Temp ", 1)
934             display.lcd_display_string("Value: " + appended_string, 2)
935             display.lcd_display_string("# to enter", 3)
936             display.lcd_display_string("* to exit", 4)
937 
938         elif appended_string != "" and password_check_initiated == 1:
939             display.lcd_display_string("Enter Password", 1)
940             display.lcd_display_string("Value: " + appended_string_password, 2)
941             display.lcd_display_string("# to enter", 3)
942             display.lcd_display_string("* to exit", 4)
943 
944         else:
945             None
```

Listing 23 Code Snippet of Displaying Temperature Menu

1.3.2.3.5 Displaying Water Level Menu

When the water level menu is selected, it needs to display the set water level and water level reading, and show the inputs made by the user on the LCD. In listing 24 line 866, the program checks if the *buttonstate* equals to D. If it is equal to D, the menu for the water level is displayed. In lines 872 to 884, the set water level, temperature water level, and instruction on how to enter and exit are displayed. In lines 886 to 890, if a number is inputted, the number is displayed. In lines 891 to 896, if the user confirms, the system asks for the password and displays current password inputs.

```
866     if buttonstate == "D":
867 
868         lcd_already_on = 0
869 
870         if appended_string == "":
871 
872             fetch_1 = open("/var/www/html/2_set_waterlvl.txt","r")
873             fetched_data_1 = str(fetch_1.read())
874             fetch_1.close()
875 
876             display.lcd_display_string("Set WtrLvl: " + fetched_data_1, 1)
877 
878             fetch = open("/var/www/html/2_waterlvl.txt","r")
879             fetched_data = str(fetch.read())
880             fetch.close()
881 
882             display.lcd_display_string("WtrLvl: " + fetched_data, 2)
883             display.lcd_display_string("# to enter", 3)
884             display.lcd_display_string("* to exit", 4)
885 
886         elif appended_string != "" and password_check_initiated != 1:
887             display.lcd_display_string("Set New WtrLvl ", 1)
888             display.lcd_display_string("Value: " + appended_string, 2)
889             display.lcd_display_string("# to enter", 3)
890             display.lcd_display_string("* to exit", 4)
891         elif appended_string != "" and password_check_initiated == 1:
892 
893             display.lcd_display_string("Enter Password", 1)
894             display.lcd_display_string("Value: " + appended_string_password, 2)
895             display.lcd_display_string("# to enter", 3)
896             display.lcd_display_string("* to exit", 4)
897 
898         else:
899             None
```

Listing 24 Code Snippet of Displaying Water Level Menu

1.3.2.4 Log Recording Module

This module is responsible for recording the logs and clearing the logs. The logs saving and clearing interval can be adjusted by the user in the Web Application.

1.3.2.4.1 Log Recording

The log contains the date and time of recording, TDS reading, water level reading, temperature reading, and pH reading. The log is saved in a text file. In listing 25 Lines 1117 to 1120, the program first retrieves the log saving interval, then multiplies it by 60 since it is in minutes. In lines 1122 to 1134, once the counter is reached, the program writes the current date and time, TDS reading, water level reading, temperature reading, and pH reading into the *log_string*. In lines 1136 to 1139, the *log_string* is stored in the sensor_logs.txt.

```
1117        fetch = open("/var/www/html/interval_logs_saving.txt","r")
1118        logs_interval = int(fetch.read())
1119        fetch.close()
1120        logs_interval = logs_interval * 60
1121
1122        if counter_logs >= logs_interval:
1123            log_string = ""
1124            current_time = datetime.datetime.now()
1125            dt_string = current_time.strftime("%d/%m/%Y %H:%M:%S")
1126            log_string += dt_string
1127            log_string += " TDS= "
1128            log_string += str(tds)
1129            log_string += " WaterLvl= "
1130            log_string += str(distance)
1131            log_string += " Temperature= "
1132            log_string += str(temp)
1133            log_string += " PhLvl= "
1134            log_string += str(ph)
1135
1136            file1 = open("/var/www/html/sensor_logs.txt","a+")
1137            file1.write(log_string)
1138            file1.write("\n")
1139            file1.close()
```

Listing 25 Code Snippet of Log Recording

1.3.2.4.2 Log Clearing

The system has a log clearing system where the user is able to set the interval for clearing the logs. Listing 26 lines 1151 to 1153 retrieve the log clearing interval, and line 1154 multiplies the log clearing by 86400 since it is in days. In lines 1156 to 1159, if the *counter_logs_reset* is greater than or equal to *clearing_interval*, the system overwrites an empty string to the sensor_logs.txt file, then in line 1160 the *counter_logs_reset* is reset back to 0.

```
1151        fetch = open("/var/www/html/interval_logs_clearing.txt","r")
1152        clearing_interval = float(fetch.read())
1153        fetch.close()
1154        clearing_interval = clearing_interval * 86400
1155
1156        if counter_logs_reset >= clearing_interval:
1157            file1 = open("/var/www/html/sensor_logs.txt","w")
1158            file1.write("")
1159            file1.close()
1160            counter_logs_reset = 0
```

Listing 26 Code Snippet of Log Clearing

1.3.3 Web Application Software

The web application should allow the users to view sensor data, view camera footage, view logs, and configure settings using a web browser. An Apache web server is used to run the PHP files.

1.3.3.1 Configure Settings in the Web Application

This module allows the user to set the pH, TDS, water level, temperature, grow lights schedule, log intervals; and to change the passcode.

1.3.3.1.1 Password System

To ensure that only authorized individuals can modify the set parameters, a password system is used. First, the system needs to allow users to input the password. In listing 27a lines 68 to 69, a form field for the password is created.

```
68    Enter Passcode: <input style="font-size:150%;"
69    type="password" name="pass_code"/>
```

Listing 27a Form Field for Passcode

The password used for the web application is the same one used in the keypad membrane discussed in 1.3.2.2.6. The password and masterkey need to be retrieved and compared if these match the contents of the passcode form field. Listing 27b lines 15 to 18 retrieve the password and store it in *$contents*. Lines 20 to 23 retrieve the masterkey and store it in *$contents2*. Line 25 takes the contents of the form field *pass_code* and stores it in *$y* so it can be compared with the *$contents* and *$contents2*.

```
15      $filename = "password.txt";
16      $handle = fopen($filename, "r");
17      $contents = fread($handle, filesize($filename));
18      fclose($handle);
19
20      $filename2 = "masterkey.txt";
21      $handle2 = fopen($filename2, "r");
22      $contents2 = fread($handle2, filesize($filename2));
23      fclose($handle);
24
25      $y=$_POST['pass_code'];
```

Listing 27b Code Snippet of Retrieving Password and Masterkey

1.3.3.1.2 Setting pH in Web Application

The pH range is only 0 to 14. In listing 28a lines 52 to 54, a form field that only allows number inputs that are within 0 to 14 is made as an input form for changing the set pH.

```
52      <form method="post"  style="font-size:200%;">
53      Set TDS value: <input style="font-size:150%;"
54      type="number" name="fnum_1" min="0" max="1500"/>
```

Listing 28a Form Field for Set pH

In listing 28b lines 27 to 29, if the password input is correct, the contents of the form field are put inside *$x*. And in lines 30 to 34, if *$x* is not empty, the *$x* is then saved in the 4_set_phplvl.txt.

```
27      if (($contents == $y) || ($contents2 == $y))
28      {
29          $x=$_POST['fnum_1'];
30      if (!empty($x))
31      {
32          $file = fopen("4_set_phlvl.txt", "w");
33          $action =  fwrite($file, $x);
34          fclose($file);
```

Listing 28b Code Snippet of Setting pH in Web Application

1.3.3.1.3 Setting TDS in Web Application

The allowed input range is 0 to 1500, though the ideal range for lettuces is 560 to 840. Setting it too low or high can damage lettuces. In listing 29a lines 52 to 54, a form field that only allows number inputs that are within 0 to 1500 is made as an input form for changing the set TDS.

```
52    <form method="post"  style="font-size:200%;">
53    Set TDS value: <input style="font-size:150%;"
54    type="number" name="fnum_1" min="0" max="1500"/>
```

Listing 29a Form Field for Set TDS

In listing 29b line 28, if the password input is correct, the contents of the form field are put inside *$x*. And in lines 29 to 35, if *$x* is not empty, the *$x* is then saved in the 1_set_tds.txt.

```
28        if (($contents == $y) || ($contents2 == $y))
29        {
30        $x=$_POST['fnum_1'];
31        if (!empty($x))
32        {
33            $file = fopen("1_set_tds.txt", "w");
34            $action =  fwrite($file, $x);
35            fclose($file);
```

Listing 29b Code Snippet of Setting TDS in Web Application

1.3.3.1.4 Setting Temperature in Web Application

The set temperature serves as deciding factor if the thermoelectric peltier cooler is to be turned on or off. In listing 30a lines 52 to 54, a form field that only allows

number inputs is made as an input form for changing the set temperature.

```
52    <form method="post"  style="font-size:200%;">
53    Set Temperature Target Value: <input style="font-size:150%;"
54    type="number" name="fnum_1"/>
```

Listing 30a Form Field for Set Temperature

In listing 30b lines 27 to 34, if the password input is correct, the contents of the form field are put inside *$x*. And if *$x* is not empty, the *$x* is saved in the 3_set_temperature.txt.

```
27          if (($contents == $y) || ($contents2 == $y))
28          {
29          $x=$_POST['fnum_1'];
30          if (!empty($x))
31          {
32              $file = fopen("3_set_temperature.txt", "w");
33              $action =  fwrite($file, $x);
34              fclose($file);
```

Listing 30b Code Snippet of Setting Temperature in Web Application

1.3.3.1.5 Setting Water Level

The water level is in percentage form and the allowed range is only 0 to 100. In listing 31a lines 52 to 54, a form field that only allows number inputs that are within 0 to 100 is made as an input form for changing the set water level.

```
52    <form method="post"  style="font-size:200%;">
53    Set Water Level Value: <input style="font-size:150%;"
54    type="number" name="fnum_1" min="0" max="100"/>
```

Listing 31a Form Field for Set Water Level

In listing 31b lines 28 to 35, if the password input is correct, the contents of the form field are put inside *$x*.

And if *$x* is not empty, the *$x* is saved in the 2_set_waterlvl.txt.

```
28          if (($contents == $y) || ($contents2 == $y))
29          {
30          $x=$_POST['fnum_1'];
31          if (!empty($x))
32          {
33              $file = fopen("2_set_waterlvl.txt", "w");
34              $action =  fwrite($file, $x);
35              fclose($file);
```

Listing 31b Code Snippet of Setting Water Level in Web Application

1.3.3.1.6 Setting Grow Lights Schedule

The grow lights schedule uses the 24 hour format using values from 0 to 2400. The system needs the start time and end time in order to operate the grow lights. In listing 32a lines 57 to 59, a form field fnum_1 allowing number inputs from 0 to 2400 is used as an input for the start time. And in lines 63 to 64, a form field fnum_2 allowing number inputs from 0 to 2400 is used as an input for the end time.

```
57      <form method="post"  style="font-size:200%;">
58      Set start time (HH:MM): <input style="font-size:150%;"
59      type="number" name="fnum_1" min="0" max="2400"/>
60      <br>
61      <br>
62
63      Set end time (HH:MM): <input style="font-size:150%;"
64      type="number" name="fnum_2" min="0" max="2400"/>
```

Listing 32a Form Field for Grow Light Schedule

In listing 32b line 36, if the password input is correct, the contents of the *fnum_1* are put inside *$x*. And if *$x* is not empty, the *$x* is saved in the growlights_starts.txt. And in lines 43 to 46, the contents of

the *fnum_2* are stored in *$x*. And if *$x* is not empty, the *$x* is saved in the growlights_end.txt.

```
36          if (($contents == $y) || ($contents2 == $y))
37          {
38          $x=$_POST['fnum_1'];
39          $file = fopen("growlights_start.txt", "w");
40          $action =  fwrite($file, $x);
41          fclose($file);
42
43          $x=$_POST['fnum_2'];
44          $file = fopen("growlights_end.txt", "w");
45          $action =  fwrite($file, $x);
46          fclose($file);
```

Listing 32b Code Snippet of Changing Grow Lights Schedule

1.3.3.1.7 Changing Password

In case of compromised password, the system allows the user to change password using the current password or the masterkey. The password should only consist of numbers since it is also used by the keypad membrane. In listing 33a lines 66 to 68, a form field allowing numeric inputs named *fnum_1* is created as an input for the current password. In lines 72 to 73, a form field allowing numeric inputs named *pass_code* is created as an input for the new password. And in lines 77 to 78, a form field allowing numeric inputs named *cpass_code* is created as an input for the confirm password.

```
66  <form method="post"  style="font-size:200%;">
67  Enter Old Passcode: <input style="font-size:150%;"
68  type="password" pattern="[0-9]*" inputmode="numeric" name="fnum_1"/>
69  <br>
70  <br>
71
72  Enter New Passcode: <input style="font-size:150%;"
73  type="password" pattern="[0-9]*" inputmode="numeric" name="pass_code"/>
74  <br>
75  <br>
76
77  Enter Confirm New Passcode: <input style="font-size:150%;"
78  type="password" pattern="[0-9]*" inputmode="numeric" name="cpass_code"/>
```

Listing 33a Form Field for Changing Passcode

In listing 33b line 37, if the new password matches the confirm password, then it proceeds to check if the old password input is correct. If the password input is correct, it proceeds to check if the new password string is not empty. If it is not empty, the new password is saved in the password.txt.

```
37  if ($_POST["pass_code"] === $_POST["cpass_code"]){
38      if (($contents == $y) || ($contents2 == $y))
39      {
40      $x=$_POST['pass_code'];
41      if (!empty($x))
42      {
43      $file = fopen("password.txt", "w");
44      $action =  fwrite($file, $x);
45      fclose($file);
```

Listing 33b Code Snippet of Changing Passcode

1.3.3.1.8 Setting Log Interval

The log saving interval and log clearing interval can be configured in the web application. In listing 34a lines 47 to 49, a form field for the log saving interval is made.

```
47  <form method="post"  style="font-size:200%;">
48  Set new value (second/s): <input style="font-size:150%;"
49  type="text" name="fnum_1"/>
```

Listing 34a Form Field for Log Saving Interval

In listing 34b lines 32 to 37, if the password input is correct, the contents of the form field are put inside *$x*. And if *$x* is not empty, the *$x* is then saved in the interval_logs_saving.txt.

```
32          if (($contents == $y) || ($contents2 == $y))
33          {
34          $x=$_POST['fnum_1'];
35          $file = fopen("interval_logs_saving.txt", "w");
36          $action =  fwrite($file, $x);
37          fclose($file);
```

Listing 34b Code Snippet of Changing Log Saving Interval

In listing 34c lines 47 to 49, a form field for the log clearing interval input is made.

```
47      <form method="post"  style="font-size:200%;">
48      Set new value (days): <input style="font-size:150%;"
49      type="text" name="fnum_1"/>
```

Listing 34c Form Field for Log Clearing Interval

In listing 34d lines 32 to 37, if the password input is correct, the contents of the form field are put inside *$x*. And if *$x* is not empty, the *$x* is then saved in the interval_logs_clearing.txt.

```
32          if (($contents == $y) || ($contents2 == $y))
33          {
34          $x=$_POST['fnum_1'];
35          $file = fopen("interval_logs_clearing.txt", "w");
36          $action =  fwrite($file, $x);
37          fclose($file);
```

Listing 34d Code Snippet of Changing Log Clearing Interval

1.3.3.2 Viewing the Web Application

This function is to allow the user to view the sensor data, set parameters, and view logs in the web application.

1.3.3.2.1 Displaying the Main Menu

The main menu displays the current readings, set parameters and assigned buttons for opening menus. Listing 35a lines 17 to 19 create the button for viewing the camera footage; lines 20 to 22 create the button for viewing the logs; lines 23 to 25 create the button for changing the log intervals; and, lines 26 to 28 create the button for changing the passcode.

```
17    <button style="padding: 5px 5px;font-size:250%;" type="button"
18    onclick="window.open('http://tanhairunpi:8080/?action=stream','
19    _blank','height=800,width=800');">Initialize Camera</button>
20    <br><br><button style="padding: 5px 5px;font-size:250%;" type="button"
21    onclick="window.open('http://tanhairunpi/sensor_data_logs.php','
22    _blank','height=800,width=800');">Sensor Logs</button>
23    <button style="padding: 5px 5px;font-size:250%;" type="button"
24    onclick="window.open('http://tanhairunpi/adjust_intervals.php','
25    _blank','height=800,width=800');">Log Intervals</button>
26    <br><br><button style="padding: 5px 5px;font-size:250%;" type="button"
27    onclick="window.open('http://tanhairunpi/reset_password.php','
28    _blank','height=800,width=800');">Change Passcode</button>
```

Listing 35a Code Snippet of Creating Buttons for Menus

Listing 35b lines 34 to 42 retrieve and display the grow lights start time, while lines 45 to 53 retrieve and display the grow lights end time.

```
30  <p style="color:blue;font-size:200%;text-align:center"><b>
31
32      Grow Lights Schedule
33      <br>
34      Start:
35      <?php
36          $filename = "growlights_start.txt";
37          $handle = fopen($filename, "r");
38          $contents = fread($handle, filesize($filename));
39          fclose($handle);
40          echo($contents);
41      ?>
42      HH:MM
43
44      <br>
45      End:
46      <?php
47          $filename = "growlights_end.txt";
48          $handle = fopen($filename, "r");
49          $contents = fread($handle, filesize($filename));
50          fclose($handle);
51          echo($contents);
52      ?>
53      HH:MM
```

Listing 35b Code Snippet of Displaying Grow Lights Schedule

Listing 35c lines 60 to 67 retrieve and display the current pH reading, while lines 69 to 75 retrieve and display the set pH.

```
60      pH Value:
61      <?php
62          $filename = "4_phlvl.txt";
63          $handle = fopen($filename, "r");
64          $contents = fread($handle, filesize($filename));
65          fclose($handle);
66          echo($contents);
67      ?>
68
69      <br>pH Target Value:
70      <?php
71          $filename = "4_set_phlvl.txt";
72          $handle = fopen($filename, "r");
73          $contents = fread($handle, filesize($filename));
74          fclose($handle);
75          echo($contents);
76      ?>
```

Listing 35c Code Snippet of Displaying pH Reading and Set pH

Listing 35d lines 82 to 89 retrieve and display the current TDS reading, while lines 91 to 98 retrieve and display the set TDS.

```
82          TDS Value:
83          <?php
84              $filename = "1_tds.txt";
85              $handle = fopen($filename, "r");
86              $contents = fread($handle, filesize($filename));
87              fclose($handle);
88              echo($contents);
89          ?>
90
91          <br>TDS Target Value:
92          <?php
93              $filename = "1_set_tds.txt";
94              $handle = fopen($filename, "r");
95              $contents = fread($handle, filesize($filename));
96              fclose($handle);
97              echo($contents);
98          ?>
```

Listing 35d Code Snippet of Displaying TDS Reading and Set TDS

Listing 35e lines 106 to 112 retrieve and display the current temperature reading, while lines 114 to 120 retrieve and display the set temperature.

```
105         Temperature:
106         <?php
107             $filename = "3_temperature.txt";
108             $handle = fopen($filename, "r");
109             $contents = fread($handle, filesize($filename));
110             fclose($handle);
111             echo($contents);
112         ?>
113
114         <br>Temperature Target Value:
115         <?php
116             $filename = "3_set_temperature.txt";
117             $handle = fopen($filename, "r");
118             $contents = fread($handle, filesize($filename));
119             fclose($handle);
120             echo($contents);
121         ?>
```

Listing 35e Code Snippet of Displaying Temperature Reading and Set Temperature

Listing 35f lines 127 to 135 retrieve and display the current water level reading, while lines 137 to 144 retrieve and display the set water level.

```
127     <p style="color:blue;font-size:300%;text-align:center"><b>
128          Water Level:
129         <?php
130             $filename = "2_waterlvl.txt";
131             $handle = fopen($filename, "r");
132             $contents = fread($handle, filesize($filename));
133             fclose($handle);
134             echo($contents);
135         ?>
136
137         <br>Water Level Target Value:
138         <?php
139             $filename = "2_set_waterlvl.txt";
140             $handle = fopen($filename, "r");
141             $contents = fread($handle, filesize($filename));
142             fclose($handle);
143             echo($contents);
144         ?>
```

Listing 35f Code Snippet of Displaying Water Level Reading and Set Water Level

1.3.3.2.2 Displaying the Logs

The logs are stored in the sensor_logs.txt file. In listing 36 line 10, the contents of the sensor_logs.txt are stored in the $file. Then in lines 12 to 17, the logs are displayed.

```
 7     <p style="font-size:120%;text-align:center"><b>
 8
 9         <?php
10         $file = fopen("sensor_logs.txt","r");
11
12         while(! feof($file))
13         {
14         echo fgets($file). "<br />";
15         }
16
17         fclose($file);
18         ?>
19
20     </p></b>
```

Listing 36 Code Snippet of Displaying Logs

CHAPTER 2 RESULTS AND DISCUSSION

This chapter contains the results and discussion of the tests. There are four (4) tests performed during the testing phase of SHL. These are the (1) Sensors Test, (2) Functionality Test, (3) Membrane Keypad Test, and (4) System Demo Test. The tests are done to ensure that the system will fully function and will be ready for use.

2.1 Sensors Test

The Sensors Test is a series of tests done to ensure that the sensors exhibit utmost accuracy by checking if the readings are within the ± accuracy of the said sensor.

2.1.1 pH Sensor Test

This test is done to ensure that the pH Sensor used exhibits good accuracy which is vital since the pH level is crucial in growing lettuces.

To conduct the test, the researchers prepared a graduated cylinder, empty glasses, distilled water, pH buffer powders, and an absorbent paper.

This test has three different phases: (1) measuring an acidic solution, (2) measuring a neutral solution, and (3) measuring an alkaline solution. Each test has 5 trials, with each trial having 50 recorded readings. This test will be considered successful if 100% of the pH Sensor reading is within the ± 0.1 pH accuracy.

2.1.1.1 Measuring an Acidic Solution

In this testing phase, the pH sensor measures an acidic solution. The researchers first cleaned the pH Sensor with distilled water and dried it using an absorbent paper. Then the researchers used a graduated cylinder to fill up 250ml distilled water and transferred the water into an empty glass. Thereafter, a 4.01 pH buffer powder was properly mixed with the distilled water. The researchers opened the Arduino IDE and used the serial monitor to get the pH readings. Then the pH Sensor was dipped into the acidic solution. The researchers waited until the readings have stabilized before recording the readings. Table 2.1.1.1 shows the summary reading of this test.

Table 2.1.1.1 Summary Table for Acidic Solution Test

Trial Number	Total Number of Readings Recorded	Number of Readings within the pH Range of 4.01 ± 0.1	Percentage of Readings within the pH Range of 4.01 ± 0.1
1	50	50	100%
2	50	50	100%
3	50	50	100%
4	50	50	100%
5	50	50	100%

Table 2.1.1.1 reflects the summary of results for the test measuring an acidic solution wherein a 4.01 pH buffer powder mixed with 250ml of distilled water was measured. Column 1 shows the trial number; Column 2, the number of readings recorded;

Column 3, the number of readings within the pH range of 4.01 ± 0.1; and, Column 4, the percentage of readings within the pH range of 4.01 ± 0.1.

In trial 1, 50 of the readings are within the range of 4.01 ± 0.1 wherein 100% of the readings are within the range of 4.01 ± 0.1. In trial 2, 50 of the readings are within the range of 4.01 ± 0.1 wherein 100% of the readings are within the range of 4.01 ± 0.1. In trial 3, 50 of the readings are within the range of 4.01 ± 0.1 wherein 100% of the readings are within the range of 4.01 ± 0.1. In trial 4, 50 of the readings are within the range of 4.01 ± 0.1 wherein 100% of the readings are within the range of 4.01 ± 0.1. In trial 5, 50 of the readings are within the range of 4.01 ± 0.1 wherein 100% of the readings are within the range of 4.01 ± 0.1. Given the results, the testing showed that the pH Sensor used has good accuracy in measuring acidic solution.

2.1.1.2 Measuring a Neutral Solution

After measuring the acidic solution, the researchers proceeded to the next testing phase where the pH Sensor measures a neutral solution. First, the researchers removed the pH Sensor from the glass containing the acidic solution. Then the pH Sensor was cleaned with distilled water and dried using an absorbent paper. The researchers went on to fill up a graduated

cylinder with 250ml distilled water and transferred the water into an empty glass. Thereafter, a 6.86 pH buffer powder was properly mixed with the distilled water. The researchers opened the Arduino IDE and used the serial monitor to get the pH readings. Then the pH Sensor was dipped into the neutral solution. The researchers waited until the readings have stabilized before recording the readings. Table 2.1.1.2 shows the summary reading of this test.

Table 2.1.1.2 Summary Table for Neutral Solution Test

Trial Number	Total Number of Readings Recorded	Number of Readings within the pH Range of 6.86 ± 0.1	Percentage of Readings within the pH Range of 6.86 ± 0.1
1	50	50	100%
2	50	50	100%
3	50	50	100%
4	50	50	100%
5	50	50	100%

Table 2.1.1.2 reflects the summary of results for the test measuring a neutral solution wherein a 6.86 pH calibrating powder mixed with 250ml of distilled water was measured. Column 1 shows the trial number; Column 2, the number of readings recorded; Column 3, the number of readings within the pH range of 6.86 ± 0.1; and, Column 4, the percentage of readings within the pH range of 6.86 ± 0.1.

In trial 1, 50 of the readings are within the range of 6.86 ± 0.1 wherein 100% of the readings are within the range of 6.86 ± 0.1. In trial 2, 50 of the readings are within the range of 6.86 ± 0.1 wherein 100% of the readings are within the range of 6.86 ± 0.1. In trial 3, 50 of the readings are within the range of 6.86 ± 0.1 wherein 100% of the readings are within the range of 6.86 ± 0.1. In trial 4, 50 of the readings are within the range of 6.86 ± 0.1 wherein 100% of the readings are within the range of 6.86 ± 0.1. In trial 5, 50 of the readings are within the range of 6.86 ± 0.1 wherein 100% of the readings are within the range of 6.86 ± 0.1. Given the results, the testing showed that the pH Sensor used has good accuracy in measuring neutral solution.

2.1.1.3 Measuring an Alkaline Solution

After measuring the neutral solution, the researchers proceeded to the last testing phase where the pH Sensor measures an alkaline solution. First, the researchers removed the pH Sensor from the glass containing the neutral solution. Then the pH Sensor was cleaned with distilled water and dried using an absorbent paper. The researchers went on to fill up a graduated cylinder with 250ml distilled water and transferred the water into an empty glass. Thereafter, a 9.18 pH buffer powder was properly mixed with the distilled water. The researchers opened the Arduino IDE and used

the serial monitor to get the pH readings. Then the pH Sensor was dipped into the alkaline solution. The researchers waited until the readings have stabilized before recording the readings. Table 2.1.1.3 shows the summary reading of this test.

Table 2.1.1.3 Summary Table for Alkaline Solution Test

Trial Number	Total Number of Readings Recorded	Number of Readings within the pH Range of 9.18 ± 0.1	Percentage of Readings within the pH Range of 9.18 ± 0.1
1	50	50	100%
2	50	50	100%
3	50	50	100%
4	50	50	100%
5	50	50	100%

Table 2.1.1.3 reflects the summary of results for the test measuring an alkaline solution wherein a 9.18 pH calibrating powder mixed with 250ml of distilled water was measured. Column 1 shows the trial number; Column 2, the number of readings recorded; Column 3, the number of readings within the pH range of 9.18 ± 0.1; and, Column 4, the percentage of readings within the pH range of 9.18 ± 0.1.

In trial 1, 50 of the readings are within the range of 9.18 ± 0.1 wherein 100% of the readings are within the range of 9.18 ± 0.1. In trial 2, 50 of the readings are within the range of 9.18 ± 0.1 wherein 100% of the readings are within the range of 9.18 ± 0.1. In trial 3, 50

of the readings are within the range of 9.18 ± 0.1 wherein 100% of the readings are within the range of 9.18 ± 0.1. In trial 4, 50 of the readings are within the range of 9.18± 0.1 wherein 100% of the readings are within the range of 9.18± 0.1. In trial 5, 50 of the readings are within the range of 9.18 ± 0.1 wherein 100% of the readings are within the range of 9.18 ± 0.1. Given the results, the testing showed that the pH Sensor used has good accuracy in measuring alkaline solution.

After having completed the testing phase, the pH Sensor is deemed to have good accuracy which in turn assures of correct dosing of pH buffer solutions leading to a healthy plant growth.

2.1.2 TDS Sensor Test

This test is done to ensure that the TDS Sensor used exhibits good accuracy. The accuracy of the TDS Sensor is significant because it measures the hydroponic solution which is a crucial part in cultivation of lettuces.

To conduct the test, the researchers prepared a 1000ppm calibrating solution. For this test, the TDS Sensor measured a 1000ppm calibrating solution. The researchers opened the Arduino IDE and used the serial monitor to get the TDS readings. Then the TDS Sensor was dipped into the solution. The researchers waited for the readings to be stable before recording such.

This test consists of 5 trials, with 50 recorded readings per trial. This test will be considered successful if 100% of the readings are within the ± 25 ppm range. Table 2.1.2 shows the summary of readings for this test.

Table 2.1.2 Summary Table for TDS Sensor Test

Trial Number	Total Number of Readings Recorded	Number of TDS Readings within the 1000 ± 25 ppm range	Percentage of TDS Readings within the 1000 ± 25 ppm range
1	50	50	100%
2	50	50	100%
3	50	50	100%
4	50	50	100%
5	50	50	100%

Column 1 of 2.1.2 shows the trial number; Column 2, the number of readings recorded; Column 3, the number of readings within the 1000 ± 25 ppm range; and, Column 4, the percentage of readings within the 1000 ± 25 ppm range.

In trial 1, 50 of the readings are within the 1000 ± 25 ppm range wherein 100% of the readings are within the 1000 ± 25 ppm range. In trial 2, 50 of the readings are within the 1000 ± 25 ppm range wherein 100% of the readings are within the 1000 ± 25 ppm range. In trial 3, 50 of the readings are within the 1000 ± 25 ppm range wherein 100% of the readings are within the 1000 ± 25 ppm range. In trial 4, 50 of the readings are within the 1000 ± 25 ppm range wherein 100% of the readings are within the 1000 ± 25 ppm range. In trial 5, 50 of the readings are within the 1000 ± 25 ppm

range wherein 100% of the readings are within the 1000 ± 25 ppm range. Given the results, this test is considered successful.

After having completed the testing, the TDS Sensor is deemed to have good accuracy which in turn assures of correct amount of nutrient solutions given to lettuces.

2.1.3 Temperature Sensor Test

This test is done to check if the Temperature Sensor reading is accurate. The reading of the Temperature Sensor is compared to the 5 in 1 water quality tester. To conduct this test, the researchers submerged both sensors in a glass of water. Then the researchers opened the Arduino IDE and used the serial monitor to get the temperature reading. Thereafter, the temperature reading was recorded and compared with the temperature reading of 5 in 1 water quality tester.

Table 2.1.3 Summary Table for Temperature Sensor Test

Trial Number	Total Number of Readings Recorded	Number of Temperature Readings within the ±1 °C of the thermometer	Percentage of Temperature Readings within the ±1 °C of the thermometer
1	50	50	100%
2	50	50	100%
3	50	50	100%
4	50	50	100%
5	50	50	100%

Table 2.1.3 reflects the summary of results of test done. Column 1 shows the trial number; Column 2, the number of readings recorded; Column 3, the number of temperature readings within the ±1 °C of the thermometer; and, Column 4, the percentage of temperature readings within the ±1 °C of the thermometer.

In trial 1, 50 readings are within the range of ±1 °C of the thermometer wherein 100% of the readings are within the range of ±1 °C of the thermometer. In trial 2, 50 readings are within the range of ±1 °C of the thermometer wherein 100% of the readings are within the range of ±1 °C of the thermometer. In trial 3, 50 readings are within the range of ±1 °C of the thermometer wherein 100% of the readings are within the range of ±1 °C of the thermometer. In trial 4, 50 readings are within the range of ±1 °C of the thermometer wherein 100% of the readings are within the range of ±1 °C of the thermometer. In trial 5, 50 readings are within the range of ±1 °C of the thermometer wherein 100% of the readings are within the range of ±1 °C of the thermometer. After having performed the test, the Temperature Sensor is deemed to have good accuracy and is reliable.

2.1.4 Ultrasonic Sensor Test

This test is done to check if the Ultrasonic Sensor can measure distances accurately. For this test, a container was filled with water. The Ultrasonic Sensor was held above the water surface to measure the distance. The distance from the water surface to the Ultrasonic Sensor was

determined using a measuring tape, and compared to the distance readings of the Ultrasonic Sensor.

This test consists of 3 scenarios, with 10 recorded readings per scenario. This test will be considered successful if all of the Ultrasonic Sensor distance readings are within ±1cm range of the actual distance.

Table 2.1.4 Summary Table for Ultrasonic Sensor Test

Scenario	Number of Ultrasonic Sensor Readings Recorded	Number of Ultrasonic Sensor Readings within ±1cm range of the actual distance	Percentage of Ultrasonic Sensor Reading within the ±1cm of the actual distance
5cm	10	10	100%
10cm	10	10	100%
15cm	10	10	100%

Table 2.1.4 shows the summary of results of test done. Column 1 shows the scenario; Column 2, the number of Ultrasonic Sensor readings recorded; Column 3, the number of Ultrasonic Sensor readings that are within the ±1cm range of the actual distance; and, Column 4, the percentage of Ultrasonic Sensor readings that are within the ±1cm range of the actual distance.

In the 5cm scenario, 10 of the readings are within the ±1cm range of the actual distance wherein 100% of the readings are within the ±1cm range of the actual distance. In the 10cm scenario, 10 of the readings are within the ±1cm range of the actual distance wherein 100% of the readings are

within the ±1cm range of the actual distance. In the 15cm scenario, 10 of the readings are within the ±1cm range of the actual distance wherein 100% of the readings are within the ±1cm range of the actual distance. Given the results, this test is considered successful. This test showed that the Ultrasonic Sensor can measure distance accurately, thereby ensuring that the system will have accurate water level.

2.2 Functionality Test

The Functionality Test is a set of tests that checks the capability of each module of the web application to perform its task correctly. The series of tests for the Functionality Test are Modifying Set pH Test, Modifying Set TDS Test, Modifying Set Temperature Test, Modifying Set Water Level Test, Scheduling Grow Lights Test, Adjusting Interval Test, and Changing Passcode Test.

To conduct this test, the researchers tried all possible scenarios in modifying the values or setting. This test is vital to ensure that all the modules work flawlessly considering that the SHL web application serves as the main control panel for this system.

2.2.1 Modifying Set pH Test

In this test, all possible scenarios for modifying the set pH are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set pH to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set pH not to change. In the third scenario, the

input is not within the allowed range, but the passcode is correct where it is expected for the set pH not to change. In the fourth scenario, the input is not within the range and the passcode is incorrect where it is expected for the set pH not to change. Table 2.2.1 shows the results of the Modifying Set pH Test.

Table 2.2.1 Summary of Modifying Set pH Test Results

Is the Input within the allowed Range?	Is Passcode Correct?	Did the Set pH Change?	Is the Set pH expected to Change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the set pH can only be changed if the input is within the allowed range and if the passcode input is correct. With this, it can be concluded that the modification of set pH in web application is working as intended.

2.2.2 Modifying Set TDS Test

In this test, all possible scenarios for modifying the set TDS are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set TDS to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set TDS not to change. In the third scenario, the input is not within the allowed range, but the passcode is

correct where it is expected for the set TDS not to change. In the fourth scenario, the input is not within the range and the passcode is incorrect where it is expected for the set TDS not to change. Table 2.2.2 shows the results of the Modifying Set TDS Test.

Table 2.2.2 Summary of Modifying Set TDS Test Results

Is the Input within the allowed Range?	Is Passcode Correct?	Did the Set TDS Change?	Is the Set TDS expected to Change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the set TDS can only be changed if the input is within the allowed range and if the passcode input is correct. With this, it can be concluded that the modification of set TDS in web application is working as intended.

2.2.3 Modifying Set Temperature Test

In this test, all possible scenarios for modifying the set Temperature are tested. In the first scenario, the passcode is correct where it is expected for the set temperature to change. In the second scenario, the passcode is incorrect where it is expected for the set temperature not to change. Table 2.2.3 shows the results of the Modifying Set Temperature Test.

Table 2.2.3 Summary of Modifying Set Temperature Test Results

Is Passcode Correct?	Did the Set Temperature Change?	Is the Set Temperature expected to change?	Result
YES	YES	YES	Successful
NO	NO	NO	Successful

The summary of results shows that the set temperature can only be changed if the passcode input is correct. Therefore, it can be concluded that the modification of set temperature in web application is working as intended.

2.2.4 Modifying Set Water Level Test

In this test, all possible scenarios for modifying the set water level are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set water level to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set water level not to change. In the third scenario, the input is not within the allowed range, but the passcode is correct where it is expected for the set water level not to change. In the fourth scenario, the input is not within the range and the passcode is incorrect where it is expected for the set water level not to change. Table 2.2.4 shows the results of the Modifying Set Water Level Test.

Table 2.2.4 Summary of Modifying Set Water Level Test Results

Is the Input within the allowed Range?	Is Passcode Correct?	Did the Set Water Level Change?	Is the Set Water Level expected to change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of this test shows that the set water level can only be changed if the input is within the allowed range and if the passcode input is correct. With this, it can be concluded that the modification of set water level in web application is working as intended.

2.2.5 Scheduling Grow Lights Test

In this test, all possible scenarios for setting the grow light schedule are tested. In the first scenario, the inputs are within the allowed range and the passcode is correct where it is expected for the grow lights schedule to change. In the second scenario, the inputs are within the allowed range, but the passcode is incorrect where it is expected for the grow lights schedule not to change. In the third scenario, the inputs are not within the allowed range, but the passcode is correct where it is expected for the grow lights schedule not to change. In the fourth scenario, the inputs are not within the range and the passcode is incorrect where it is expected for the grow lights schedule not to change. Table 2.2.5 shows the results of the Scheduling Grow Lights Test.

Table 2.2.5 Summary of Modifying Grow Lights Schedule Test Results

Are the Inputs within the allowed Range?	Is Passcode Correct?	Did the Grow Lights Schedule Change?	Is the Grow Lights Schedule Expected to Change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the grow lights schedule successfully changes if the inputs are within the allowed range and if the passcode input is correct. With this, it can be concluded that the scheduling of grow lights is working as intended.

2.2.6 Adjusting Log Intervals

This test is conducted to verify if the modules of adjusting the intervals are working as intended. In this test, there will be two (2) modules that will be tested: (1) Adjusting Log Saving Interval, and, (2) Adjusting Log Clearing Interval.

2.2.6.1 Adjusting Log Saving Interval Test

In this test, all possible scenarios for adjusting the log saving interval are tested. In the first scenario, the passcode is correct where it is expected for the log saving interval to change. In the second scenario, the passcode is incorrect where it is expected for

the log saving interval not to change. Table 2.2.6.1 shows the results of the Adjusting Log Saving Interval Test.

Table 2.2.6.1 Summary of Adjusting Log Saving Interval Test Results

Is Passcode Correct?	Did the Log Saving Interval Change?	Is it expected for the Log Saving Interval to Change?	Result
YES	YES	YES	Successful
NO	NO	NO	Successful

The summary of results shows that the log saving interval can only be changed if the passcode input is correct. With this, it can be concluded that the adjustment of log saving interval is working as intended.

2.2.6.2 Adjusting Log Clearing Interval

In this test, all possible scenarios for adjusting the log clearing interval are tested. In the first scenario, the passcode is correct where it is expected for the log clearing interval to change. In the second scenario, the passcode is incorrect where it is expected for the log clearing interval not to change. Table 2.2.6.2 shows the Summary of the Adjusting Log Clearing Interval Test Result.

Table 2.2.6.2 Summary of Adjusting Log Clearing Interval Test Results

Is Passcode Correct?	Did the Log Clearing Interval Change?	Is it expected for the Log Clearing Interval to Change?	Result
YES	YES	YES	Successful
NO	NO	NO	Successful

The summary of results shows that the log clearing interval can only be changed if the passcode input is correct. With this, it can be concluded that the adjustment of log clearing interval is working as intended.

2.2.7 Changing Passcode Test

In this test, all possible scenarios for changing the passcode are tested. In the first scenario, the current passcode is correct and the new passcode matches the confirmed passcode where it is expected for the passcode to change. In the second scenario, the current passcode is correct, but the new passcode does not match the confirmed passcode where it is expected for the passcode not to change. In the third scenario, the passcode is incorrect, but the new passcode matches the confirmed passcode where it is expected for the passcode not to change. In the fourth scenario, the current passcode is incorrect and the new passcode does not match the confirmed passcode where it is expected for the passcode not to change. Table 2.2.7 shows the results of the Changing Passcode Test.

Table 2.2.7 Summary of Changing Passcode Test Results

Is the Current Passcode Input Correct?	Are the New Passcode and Confirmed Passcode the Same?	Did the Passcode Change?	Is it expected for the Passcode to Change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the passcode successfully changes if the current passcode input is correct and if the new passcode matches the confirm passcode. It can therefore be concluded that the changing of passcode is working as intended.

2.3 Membrane Keypad Test

The membrane keypad is used as a supplementary control panel to change the target values for pH, TDS, temperature, and water level to maintain the system. The Membrane Keypad Test is a set of tests to ensure that the keypad is fully functional and is working as intended.

2.3.1 Keypad Button Test

This test can be considered as the most vital part of the Membrane Keypad Test. In this test, the researchers tested each keypad button to ensure that the keypad receives inputs and displays the readings on the LCD. The researchers expect that all keypads are functioning, otherwise there is a faulty jumper wires or the wiring of the keypad to the GPIO pins of

the Raspberry Pi is incorrect. To conduct this test, each keypad button is pressed five (5) times individually. Table 2.3.1 shows the results for the Keypad Button Test.

Table 2.3.1 Results of the Keypad Button Test

Keypad Button Function	Number of Trials	Number of Successful Trials	Percentage of Successful Trials
When keypad "1" is pressed	5	5	100%
When keypad "2" is pressed	5	5	100%
When keypad "3" is pressed	5	5	100%
When keypad "A" is pressed	5	5	100%
When keypad "4" is pressed	5	5	100%
When keypad "5" is pressed	5	5	100%
When keypad "6" is pressed	5	5	100%
When keypad "B" is pressed	5	5	100%
When keypad "7" is pressed	5	5	100%
When keypad "8" is pressed	5	5	100%
When keypad "9" is pressed	5	5	100%
When keypad "C" is pressed	5	5	100%
When keypad "*" is pressed	5	5	100%
When keypad "0" is pressed	5	5	100%
When keypad "#" is pressed	5	5	100%
When keypad "D" is pressed	5	5	100%

olumn 1 shows what keypad button was pressed; Column 2, the number of trials of pressing the button; Column 3, the number of successful trials; and, Column 4, the percentage of successful trials.

The results show that all the keypad buttons got 100% percentage of successful trials when pressed, thus this proves the keypad membrane is perfectly functional.

2.3.2 Menu Selection Test

For this test, each button assigned as a menu button will be tested. Button A is assigned to pH where it is expected to select the pH menu when pressed. Button B is assigned to TDS where it is expected to select the TDS menu when pressed. Button C is assigned to Temperature where it is expected to select the Temperature menu when pressed. And, Button D is assigned to Water Level where it is expected to select the Water Level menu when pressed. Table 2.3.2 shows the results of the Menu Selection Test.

Table 2.3.2 Results of the Menu Selection Test

Keypad Button Press	What does it select?	What is expected to be selected?	Result
When button "A" is pressed	pH Menu	pH Menu	Successful
When button "B" is pressed	TDS Menu	TDS Menu	Successful
When button "C" is pressed	Temperature Menu	Temperature Menu	Successful
When button "D" is pressed	Water Level Menu	Water Level Menu	Successful

The summary of results shows that all the assigned buttons select the correct menu when pressed. It can therefore be concluded that the menu selection is working as intended.

2.3.3 Modifying Set Values in Keypad Test

This test is done to verify if the user can change the set values in the keypad, whether successfully or not. To conduct this test, the researchers tried all possible scenarios in modifying the values.

2.3.3.1 Modifying Set pH in Keypad Test

In this test, all possible scenarios for modifying the set pH in keypad are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set pH to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set pH not to change. In the third scenario, the input is not within the allowed range, but the password is correct where it is expected for the set pH not to change. In the fourth scenario, the input is not within the range and the password is incorrect where it is expected for the set pH not to change. Table 2.3.3.1 shows the test results for Modifying Set pH in Keypad.

Table 2.3.3.1 Results of the Modifying Set pH in Keypad Test

Is the Input within the Allowed Range?	Is Passcode Correct?	Did the set pH change?	Is the set pH expected to change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the pH can only be changed if the input is within the allowed range and if the password input is correct. It can therefore be concluded that the modification of set pH in the keypad is functional.

2.3.3.2 Modifying Set TDS in Keypad Test

In this test, all possible scenarios for modifying the set TDS are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set TDS to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set TDS not to change. In the third scenario, the input is not within the allowed range, but the passcode is correct where it is expected for the set TDS not to change. In the fourth scenario, the input is not within the range and the passcode is incorrect where it is expected for the set TDS not to change. Table 2.3.3.2 shows the test results for Modifying Set TDS in Keypad.

Table 2.3.3.2 Results of the Modifying Set TDS in Keypad Test

Is the Input within the Allowed Range?	Is Passcode Correct?	Did the set TDS change?	Is the set TDS expected to change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful
NO	NO	NO	NO	Successful

The summary of results shows that the TDS can only be changed if the input is within the allowed range and if the password input is correct. . It can therefore be concluded that the modification of set TDS in the keypad is functional.

2.3.3.3 Modifying Set Temperature in Keypad Test

In this test, all possible scenarios for modifying the set Temperature are tested. In the first scenario, the passcode is correct where it is expected for the set temperature to change. In the second scenario, the passcode is incorrect where it is expected for the set temperature not to change. Table 2.3.3.3 shows the test results for Modifying Set Temperature in Keypad.

Table 2.3.3.3 Results of the Modifying Set Temperature in Keypad Test

Is Passcode Correct?	Did the Set Temperature Change?	Is the Set Temperature expected to change?	Result
YES	YES	YES	Successful
NO	NO	NO	Successful

The summary of results shows that the temperature can only be changed if the password input is correct. It can therefore be concluded that the modification of set temperature in the keypad is functional.

2.3.3.4 Modifying Set Water Level in Keypad Test

In this test, all possible scenarios for modifying the set water level are tested. In the first scenario, the input is within the allowed range and the passcode is correct where it is expected for the set water level to change. In the second scenario, the input is within the allowed range, but the passcode is incorrect where it is expected for the set water level not to change. In the third scenario, the input is not within the allowed range, but the passcode is correct where it is expected for the set water level not to change. In the fourth scenario, the input is not within the range and the passcode is incorrect where it is expected for the set water level not to change. Table 2.3.3.4 shows the test results for Modifying Set Water Level in Keypad.

Table 2.3.3.4 Results of the Modifying Set Water Level in Keypad Test

Is the Input within the Allowed Range?	Is Passcode Correct?	Did the water level change ?	Is the set Water Level expected to change?	Result
YES	YES	YES	YES	Successful
YES	NO	NO	NO	Successful
NO	YES	NO	NO	Successful

NO	NO	NO	NO	Successful

The summary of results shows that the water level can only be changed if the input is within the allowed range and if the password input is correct. It can therefore be concluded that the modification of set water level in the keypad is functional.

2.4 System Demo Test

In the System Demo Test, the system is tested as a whole to ensure that it is working as intended. This test is conducted to verify six things: (1) if the nutrient solution is dosed correctly by reaching the set TDS threshold, (2) if the pH buffer solution is dosed correctly by reaching the set pH set threshold, (3) if the water pumps turn on or off depending on the conditions, (4) if the thermoelectric peltier cooler turns on or off when comparing the set temperature to the temperature reading, (5) if the grow light turns on or off when comparing the set schedule to the current time of the system, and (6) if the RPI camera is functioning.

2.4.1 Nutrient Solution Dosing Test

This test is done to ensure that the nutrient solution dosing of the system is working as intended. This test checks if the dosing pumps of nutrients solutions are functioning, and if there is no dosing of nutrient solutions once the TDS reading is already within the threshold.

The test commenced by running the system with fresh water, with TDS reading at 93ppm. From having a set TDS value, the researchers checked after one hour the reading of the TDS value.

This test will be considered successful if the TDS reading after one hour is within ±50ppm of the set TDS.

Table 2.4.1 Result of Nutrient Solution Dosing Test

Test Number	TDS Reading of the system	Set TDS	TDS Reading after 1 hour	Is TDS Reading within ±50ppm of the set TDS?
Test 1	93ppm	250ppm	255ppm	YES
Test 2	255ppm	550ppm	565ppm	YES
Test 3	565ppm	750ppm	763ppm	YES
Test 4	763ppm	755ppm	760ppm	YES

Table 2.4.1 shows the results of the Nutrient Solution Dosing Test. Column 1 shows the test number; Column 2, the TDS reading of the system; Column 3, the set TDS; Column 4, the TDS reading after one hour; and, Column 5, the TDS reading, to ascertain if within the ±50ppm threshold of the set TDS. In test number 1, the initial TDS reading was 93ppm and the set TDS was 250ppm. After an hour of running the system, the TDS reading became 255ppm. In test number 2, the initial TDS reading was 255ppm and the set TDS was 550ppm. After leaving the system to run, particularly after an hour, the TDS reading became 565ppm. In test number 3, the initial TDS reading was 565ppm and the set TDS was 750ppm. Then after an hour, the TDS reading became 763ppm. In test number 4, the initial

TDS reading was 763ppm and the set TDS was 755ppm. After an hour, the TDS reading became 760ppm. Since the set TDS is already within the TDS reading threshold, there should be no dosing. The researchers waited for an hour and verified there was no dosing.

After having conducted the four tests, the results show that the TDS reading approaches the set TDS and after an hour, it is within the threshold. The system does not dose more nutrient solution if the TDS reading is within the threshold. It can therefore be concluded that the Nutrient Solution Dosing works as intended.

2.4.2 pH Buffer Solution Dosing Test

This test is done to ensure that the pH buffer solution dosing of the system is working as intended. This test checks if the dosing pumps for pH up and pH down are functioning, and if there is no dosing of buffer solution once the set pH is already within the threshold.

The test started by running the system with fresh water which has a pH reading of 7.32. From having a set pH value, the researchers checked after one hour the reading of the pH value.

This test will be considered successful if the pH reading after one hour is within ±0.3 of the Set pH.

Table 2.4.2 Result of pH Buffer Solution Dosing Test

Test Number	pH Reading of the system	Set pH	pH Reading after 1 hour	Is pH Reading within ± 0.3 of the set pH?
Test 1	7.32	6.5	6.71	YES
Test 2	6.71	6.0	6.12	YES
Test 3	6.12	7.0	6.98	YES
Test 4	6.98	7.0	6.93	YES

Table 2.4.2 shows the results of the pH Buffer Solution Dosing Test. Column 1 shows the test number; Column 2, the pH reading of the system; Column 3, the set pH; Column 4, the pH reading after one hour; and, Column 5, the pH reading, to ascertain if within the ±0.3 threshold of the set pH value. In test number 1, the initial pH reading was 7.32 and the set pH was 6.5. After an hour of running the system, the pH reading became 6.71. In test number 2, the initial pH reading was 6.71 and the set pH was 6.0. After leaving the system to run for an hour, the pH reading became 6.12. In test number 3, the initial pH reading was 6.12 and the set pH was 7.0. Then after an hour, the pH reading became 6.98. In test number 4, the initial pH reading was 6.98 and the set pH was still 7.0. After an hour, the pH reading became 6.93. Since the set pH is already within the pH reading threshold, the system should not anymore dose pH buffer solution. The researchers waited for an hour and verified there was no dosing.

After having conducted the four tests, the results show that the pH reading approaches the set pH and after an hour, it is within the threshold. The system does not dose any more pH buffer solution if the pH reading is

within the threshold. It can therefore be concluded that the pH maintaining module works as intended.

2.4.3 Water Pump Test

This test is done to ensure that the water pumps used are able to pump water in and out on correct conditions.

2.4.3.1 Water in Pump Test

This test is done to ensure that the water pump from the water reservoir transfers water to the hydroponic reservoir if the right conditions are met. It is done by setting the water level at varying values. This test will be considered successful if the water pump from the water reservoir will pump in water to the hydroponic reservoir at a time when the water level reading is lower than the set water level, and the pump will not turn on if the water level reading is greater or equal to the set water level.

Table 2.4.3.1 Test Summary of Water in Pump

Test Number	Water Level Reading	Set Water Level	Should the Water Pump turn on?	Did the Water Pump turn on?
Test 1	50	50	NO	NO
Test 2	50	75	YES	YES
Test 3	65	30	NO	NO
Test 4	60	90	YES	YES

Table 2.4.3.1 shows the summary of the Water in Pump Test. Column 1 shows the test number; Column 2, the water level reading; Column 3, the set water level; Column 4, if the water in pump should turn on; and, Column 5, if the water in pump turned on. In test number 1, when the water level reading was 50 and the set water level was 50, the water pump did not turn on because the water level reading was equal to the set water level. In test number 2, when the water level reading was 50 and the set water level was 75, the water pump turned on because the water level reading was lower than the set water level. In test number 3, when the water level reading was 65 and the set water level was 30, the water pump did not turn on because the water level reading was higher than the set water level. In test number 4, when the water level reading was 60 and the set water level was 90, the water pump turned on because the water level reading was lower than the set water level.

After having conducted the four tests, the results show that the water pump is functioning and it turns on until the current reading reaches the set water level. The water pump does not turn on if the water level reading is equal or greater than the set water level. It can thus be concluded that the water level maintaining module is fully functional.

2.4.3.2 Water Pump TDS Reduction Test

This test is done to ensure that the water pump for reducing the TDS value pumps out hydroponic solution from the hydroponic reservoir. For this test, the researchers set the TDS at varying values. The test will be considered successful if the water pump for TDS reduction will pump out hydroponic solution from the hydroponic reservoir when the TDS reading is higher than the set TDS.

Table 2.4.3.2 Test Summary of Water Pump TDS Reduction

Test Number	TDS Reading	Set TDS	Should the Water Pump turn on?	Did the Water Pump turn on?
Test 1	766 ppm	100ppm	YES	YES
Test 2	541ppm	550ppm	NO	NO

Table 2.4.3.2 shows the summary of the Water Pump TDS Reduction Test. Column 1 shows the test number; Column 2, the current TDS reading of the system; Column 3, the set TDS; Column 4, if the water pump should turn on; and, Column 5, if the water pump for TDS reduction turned on. In test number 1, the TDS reading of the system was 766ppm while the researchers set the TDS value to 100ppm. The water pump turned on and continued to remove water from the hydroponic reservoir since the TDS reading was higher than the set TDS. In test number 2, the

TDS reading of the system was 541ppm while the researchers set the TDS value to 550ppm. The water pump did not turn on since the set TDS was already within the TDS threshold of the system.

After having conducted the tests, the results show that the water pump is functioning and it turns on whenever the TDS reading is higher than the set TDS threshold. And it does not turn on when the TDS reading is below the set TDS threshold. It can thus be concluded that the system can reduce TDS level at correct circumstances.

2.4.4 Thermoelectric Peltier Cooler Test

This test is done to ensure that the thermoelectric peltier cooler turns on given correct circumstances. For this test, the researchers set the temperature at varying values. This test will be considered successful if the thermoelectric peltier cooler will turn on when the temperature reading is higher than the set temperature, and turn off when the temperature reading is lower than the set temperature.

Table 2.4.4 Test Summary of Thermoelectric Peltier Cooler

Test Number	Temperature Reading	Set Temperature	Should the Cooler Turn On?	Did Cooler Turn On?
Test 1	29°C	25°C	YES	YES
Test 2	29°C	35°C	NO	NO
Test 3	29°C	20°C	YES	YES
Test 4	29°C	40°C	NO	NO

Table 2.4.4 shows the summary of the Thermoelectric Peltier Cooler Test. Column 1 shows the test number; Column 2, the temperature reading; Column 3, the set temperature; Column 4, if the cooler should turn on; and, Column 5 if the cooler did turn on. In all the test numbers, the temperature reading was constant at 29°C. In test number 1, when the set temperature was 25°C, the cooler turned on as should be expected. In test number 2, when the set temperature was 35°C, the cooler turned off as should be expected. In test number 3, when the set temperature was 20°C, the cooler turned on as should be expected. In test number 4, when the set temperature was 40°C, the cooler turned off as should be expected.

After having conducted the four tests, the results show that the system turns on the thermoelectric peltier cooler whenever the set temperature is lower than the temperature reading. This test also verified that the thermoelectric peltier cooler is functioning which is important since it helps in cooling the hydroponic solution in the hydroponic reservoir.

2.4.5 Grow Lights Test

This test is done to ensure that the grow lights turn on whenever the current time is within the set schedule by the user. The test is done by setting the grow lights schedule at varying time. This test will be considered successful if the grow lights will turn on when the current time is within the set schedule and will turn off when the current time is not within the set schedule.

Table 2.4.5 Test Summary of Grow Lights

Test Number	Current Time	Set Schedule	Should the Grow Lights Turn on?	Did the Grow Lights Turn on?
Test 1	22:31	00:00-23:59	YES	YES
Test 2	22:34	22:34-22:35	YES	YES
Test 3	22:35	00:00-00:01	NO	NO
Test 4	22:36	22:34-22:35	NO	NO

Table 2.4.5 shows the summary of the Grow Lights Test. Column 1 shows the test number; Column 2, the current time of the system; Column 3, the set schedule; Column 4, if the grow lights should turn on; and, Column 5, if the grow lights did turn on. In test number 1, when the current time was 22:31 and the set schedule was 00:00-23:59, the grow lights turned on as should be expected. In test number 2, when the current time was 22:34 and the set schedule was 22:34-22:35, the grow lights turned on as should be expected. In test number 3, when the current time was 22:35 and the set schedule was 00:00-00:01, the grow lights did not turn on as should be expected. In test number 4, when the current time was 22:36 and the set schedule was 22:34-22:35, the grow lights did not turn on as should be expected.

After having conducted the test, it can be concluded that the system turns on the grow lights whenever the current time is within the set schedule. This test verified that the grow lights are functioning which is

important since the grow lights act as alternative sunlight for the photosynthesis of lettuces.

2.4.6 RPI Camera Test

This test is done to ensure that the RPI camera is functioning. For this test, the researchers accessed the camera and checked if it is functioning. This test will be considered successful if the camera can show real-time monitoring of lettuces.

Table 2.4.6 Test Summary of RPI Camera

Test Number	Can the Video be Viewed?
Camera Test 1	YES
Camera Test 2	YES
Camera Test 3	YES
Camera Test 4	YES

Table 2.4.6 shows the summary of the RPI Camera Test. Column 1 shows the test number and Column 2 shows if the video can be viewed. For this test, the RPI camera was accessed and showed the live footage.

CHAPTER 3 CONCLUSION

In summary, the student-researchers have successfully developed the Smart Hydroponics for Lettuces (SHL) that aids in the cultivation of growing lettuces. The system was successfully run and demonstrated.

The study was focused on the automation of feeding of nutrient and pH buffer solutions; irrigation; and, monitoring and maintenance of vital parameters, such as, pH value, TDS value, temperature, and water level of the hydroponic system.

The system was designed to have a web-application that is user friendly, particularly one that allows access through a web browser for real time monitoring, maintaining the values, configuring the settings, and viewing of live footage of lettuces. The system also has supplementary control panel where the user can view the sensor readings on LCD and use a keypad to change the target values for pH, TDS, temperature, and water level to maintain the system. Given this developed system, the user easily monitors and maintains the cultivation of lettuces with less involvement.

In conclusion, the development of this research was satisfactorily met as proven from the results of tests done. Each module was tested and specific goals were achieved from and after the conduct of trials on each module. Based on the results of the Sensor Testing, Functionality Test, Membrane Keypad Test, and System Demo Test, the modules and the system are working satisfactorily and are fully functional.

Though the SHL system is fully functional, the system also has its limits or problems. When the system turns on and there is no internet connection, the system

time does not come out correct. This leads to the grow light turning on or off not within the set schedule since the time is not correct. It needs internet connection to have the system time updated.

Thus, the researchers thoroughly inspected the housing of lettuces. The researchers made sure that the pipes were secured and tightened to make sure there was no leaking in the housing. In general, given the test results, it is concluded that the SHL has met the objectives of the study and addressed issues identified at the beginning of the research. The final system resulted to ease of feeding of the nutrient solutions, proper distribution of water, systematic monitoring and maximum optimization of plant growth of lettuces.

CHAPTER 4 RECOMMENDATIONS

The Smart Hydroponics for Lettuces (SHL) has proven to be less hassle and more efficient in the growing of lettuces. Nonetheless, the researchers believe that SHL may further be innovated in the future to maximize the full potential of the system. The following are recommendations of the researchers for future development of the system:

First, the researchers recommend for SHL web application to have a login module. It may be in different language as long as it has its login module to have better security system. In addition, the researchers believe that the User Interface (UI) design of SHL web-application has much more to improve.

Second, the researchers recommend for upgrading of the thermoelectric peltier cooler and adding of grow lights. Although such would translate to higher power consumption, the researchers maintain this recommendation since both modules are essential in growing lettuces. When upgrading the thermoelectric peltier cooler and adding grow lights, the power supply should also be upgraded to 12V, 30A.

Third, the researchers also recommend the addition of a fill valve to the water reservoir, to ensure that the water reservoir is always full.

Fourth, the researchers recommend the addition of a Real-Time Clock (RTC) module, to ensure that the system time is always correct even without internet connection.

Fifth, the researchers recommend for upgrading of the current pH sensor into an industrial analog pH sensor. This is a professional Arduino pH Sensor Meter Kit with an

industrial electrode. This is suitable for long term monitoring. Since this will be an upgraded version, this product will greatly improve the precision and user experience.

Lastly, the researchers recommend to have a backup power system, like an uninterruptible power supply (UPS), to ensure that the project will continue to function despite power fluctuations or power outages.

BIBLIOGRAPHY

Elferink, M. & Schierhorn, F. (2016, April 7). Global Demand for Food Is Rising. Can We Meet It? Retrieved April 22, 2023 from https://hbr.org/2016/04/global-demand-for-food-is-rising-can-we-meet-it

Homehydrosystems.com. (n.d.). Vegetable Requirements. Retrieved April 22, 2023 from http://www.homehydrosystems.com/ph_tds_ppm/ph_vegetables_page.html

Hydroponicsspace.com. (2023). How Can Hydroponics Improve Food Security? Retrieved April 22, 2023 from https://www.hydroponicsspace.com/how-can-hydroponics-improve-food-security/

Tadimalla, R.T. (2023, March 6). 16 Scientifically Proven Health Benefits Of Lettuce. Retrieved April 22, 2023 from https://www.stylecraze.com/articles/best-benefits-of-lettuce-for-skin-hair-and-health/

USER'S MANUAL

Table of Contents

1. Setting up SHL

1.1 Setting up the Hydroponic Reservoir

1.2 Setting up the Water Reservoir

1.3 Setting up the SHL System

1.4 Setting up the Hydroponic Housing

2. Getting Started with SHL System

2.1 Starting up the SHL System

2.2 Connecting the SHL to a WIFI

2.3 Accessing the Web Application

2.4 Handling Main Menu Window

2.4.1 Initialize Camera

2.4.2 Viewing Sensor Logs

2.4.3 Log Intervals

2.4.4 Changing of Passcode

2.4.5 Adjusting Grow lights Schedule

2.4.6 Modify pH Settings

2.4.7 Modify TDS Setting

2.4.8 Modify Temperature Setting

2.4.9 Modify Water Level Settings

3. Keypad Guide

3.1 Numerical Keypad Buttons

3.2 Symbol Keypad Buttons

3.3 Alphabetical Keypad Buttons

4. Sensor Calibration

4.1 pH Sensor Calibration

4.2 TDS Sensor Calibration

5. Troubleshooting

1. Setting up SHL

SHL consists of 4 parts: (1) Hydroponic Housing, (2) Hydroponic Reservoir, (3) Water Reservoir, and (4) SHL System.

Figure M1. Hydroponic Housing

Figure M2. Hydroponic Reservoir

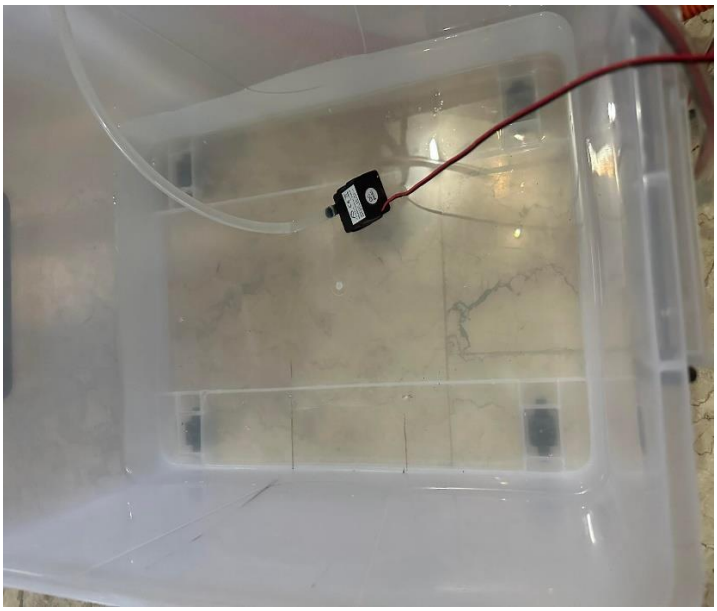


Figure M3. Water Reservoir

Figure M4. SHL System

1.1 Setting up the Hydroponic Reservoir

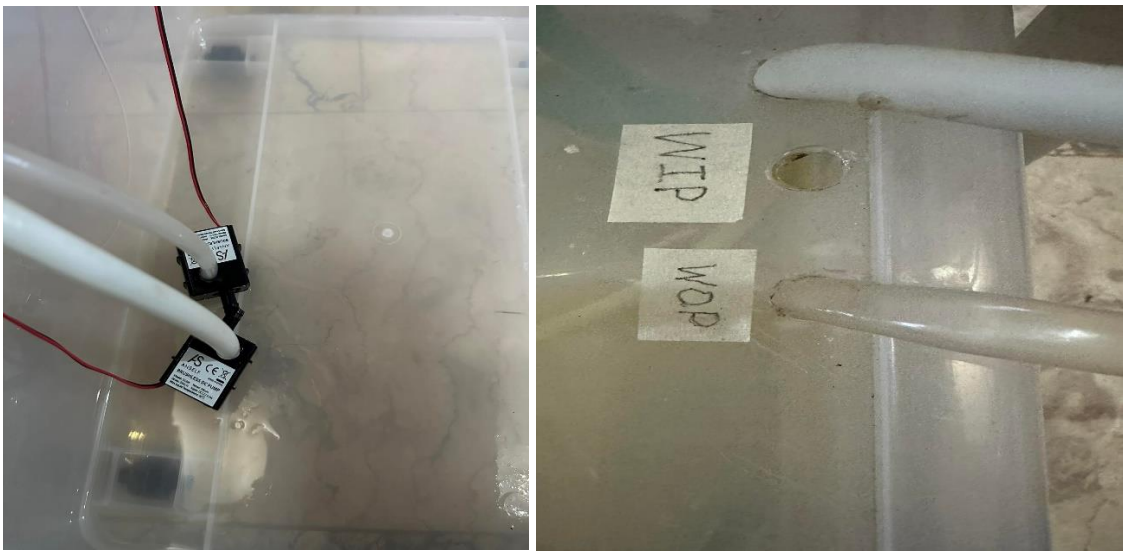
1. Place the Hydroponic Reservoir beneath the Hydroponic Housing.

2. Insert the flexible hose into the hydroponic container’s lid hole.

3. Place the water out pump and the mixing pump inside the container then attach the white tube to the mixing pump.

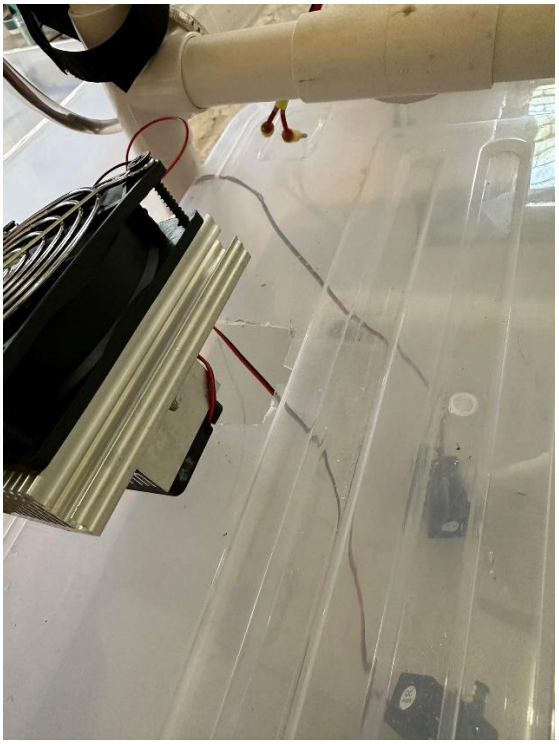
4. Pass the hose through the hole of the lid and connect it to the Hydroponic Housing.

5. Attach the longer transparent tube to the water out pump then insert it through the lid hole labeled WOP, and place the other end of the tube into a drainage.



6. Fill up the Hydroponic Reservoir with water measuring around 30% or more of its total volume.

7. Lastly, place the thermoelectric peltier cooler in its designated placement.



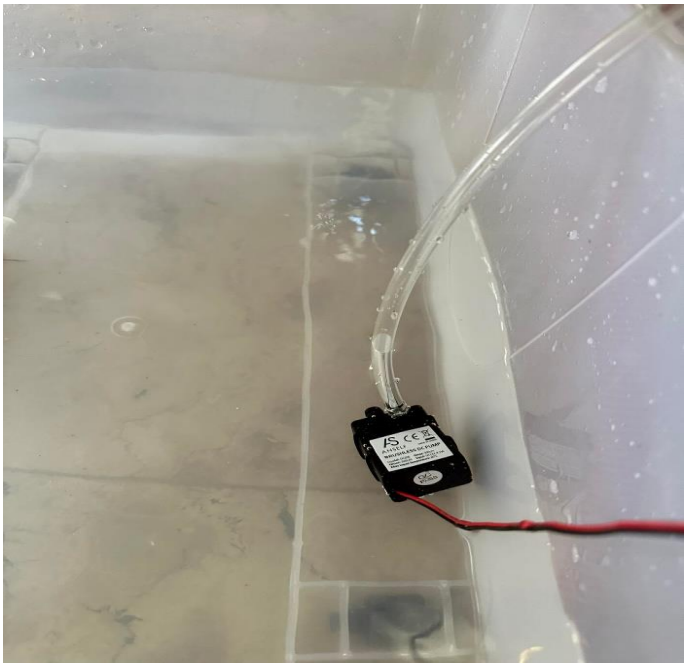
1.2 Setting up the Water Reservoir

1. Place the Water Reservoir near the Hydroponic Reservoir and Hydroponic Housing.

2. Place the water in pump inside the Water Reservoir then attach the tube to the pump.

3. Insert the other end of the tube to the Hydroponic Reservoir's lid hole labeled WIP.

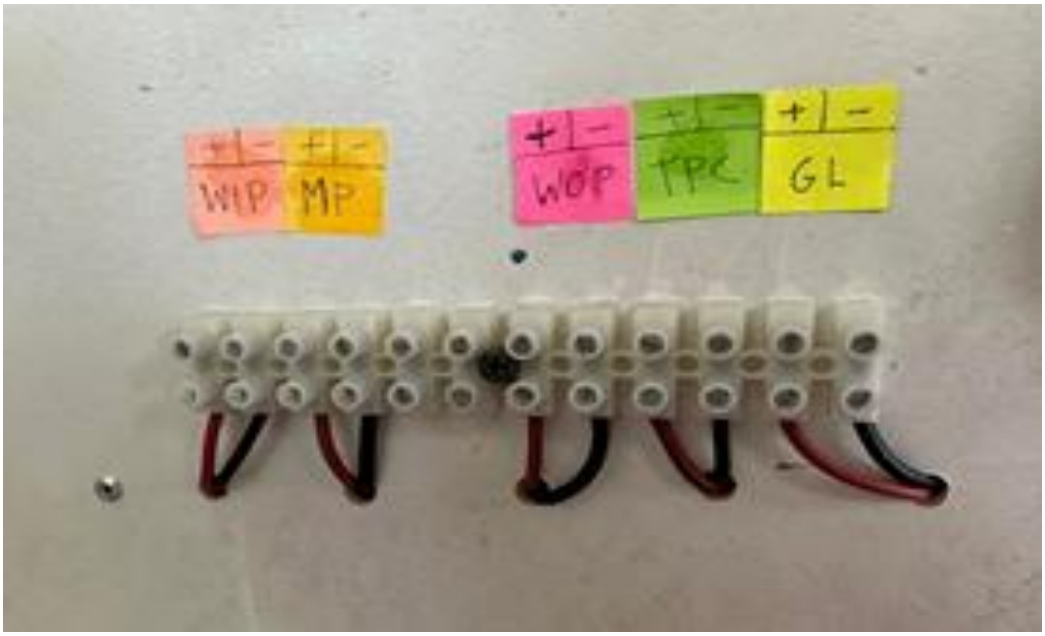
4. Fill up the Water Reservoir with water measuring around 50% or more of its total volume.



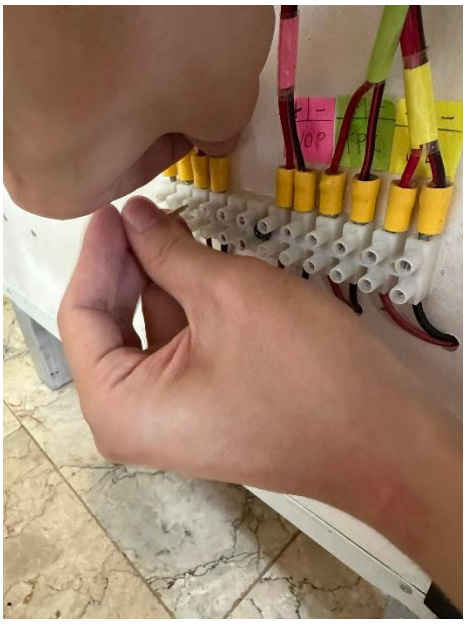
1.3 Setting up the SHL System

1. Three hoses contain different sets of wires. The sink hose is connected to the Hydroponic Reservoir. The hose contains 3 sets of wires. It is identified using different colors which are green, pink and orange. A transparent hose connected to the Hydroponic Housing contains 1 set of wire with yellow tag, and another transparent hose connected to the Water Reservoir contains 1 set of wire with peach tag. Secure the hoses in place.

2. These wires are connected to the terminal block. And above the terminal block are labels.



3. Connect the terminal crimps to the corresponding colors in the terminal block, with the red wire below the “+” sign and the black wire below the “–” sign. Then tighten the screws on the terminal block using a screw driver.



4. Insert the tubes of the peristaltic pumps into the lid hole of the Hydroponic Reservoir together with the flexible hose.

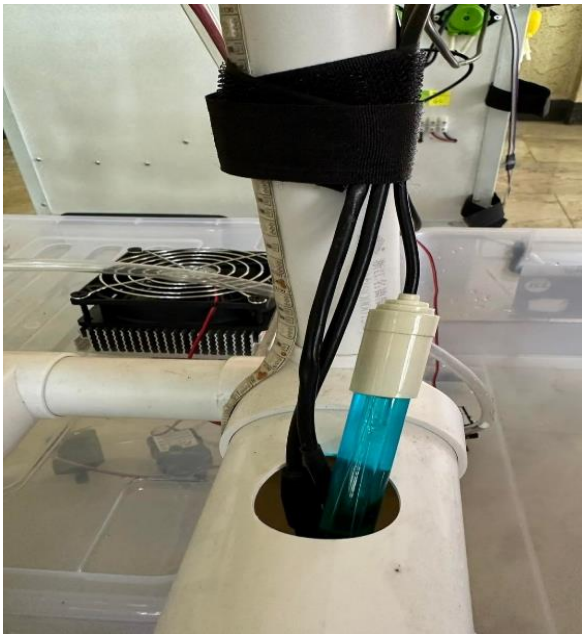
5. Place the RPI Camera on its camera stand. Then put an adhesive at the back of the RPI Camera.

6. Connect the USB Cable to the Arduino USB Port.

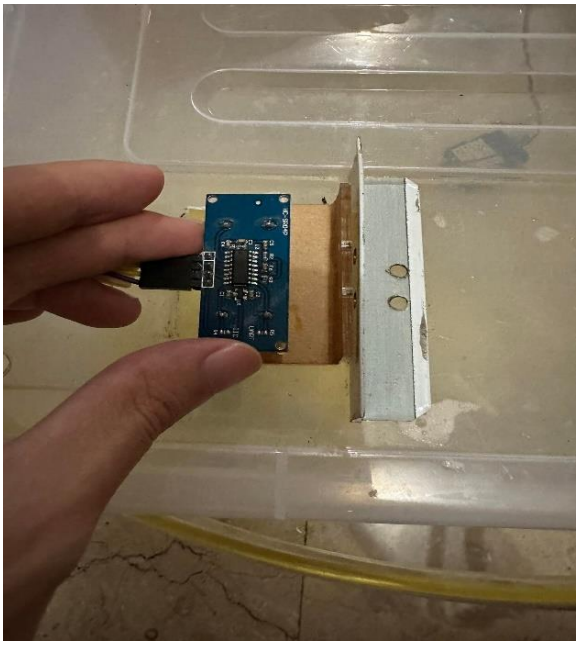
1.4 Setting up the Hydroponic Housing

1. Place the pH Sensor, TDS Sensor, and Temperature Sensor in the nearest hole of the Hydroponic Housing.

2. Using a velcro tape, wrap around the wires of the sensors.



3. Insert the Ultrasonic Sensor into its housing.



2. Starting the SHL System

2.1 Starting the SHL System

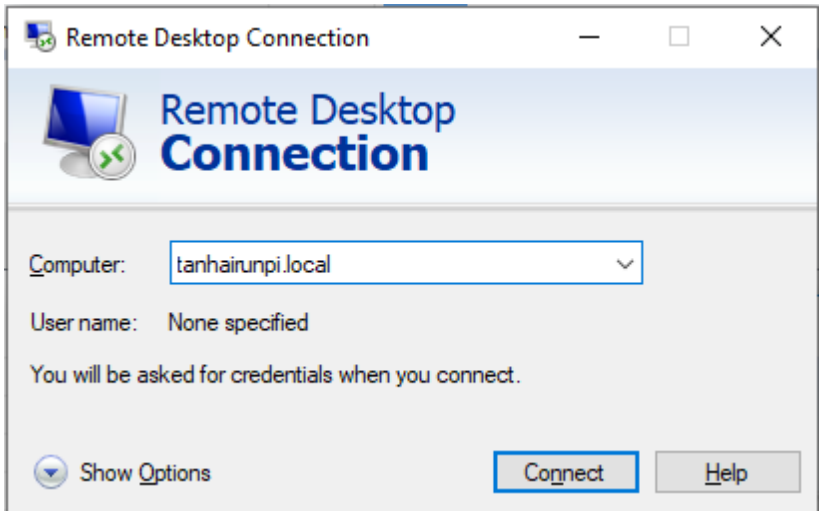
1. Plug the Power Supply and AC Adapter of the RPI to a power outlet to run the system.

2.2 Connecting the SHL to a WIFI

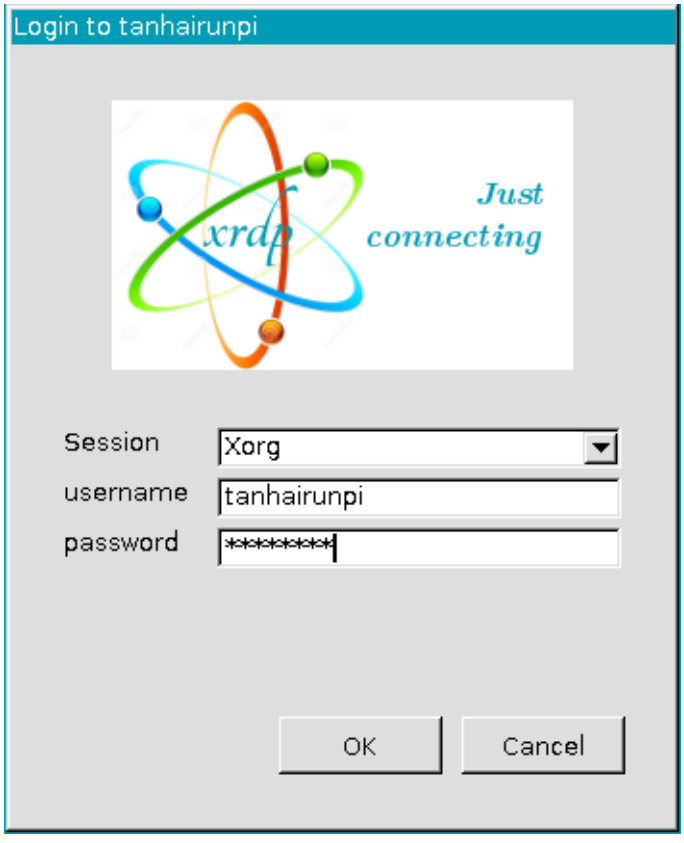
The SHL must be connected to a router in order to access the web application.

1. The user must have a computer with an ethernet cable. Connect the ethernet cable to the RPI ethernet port and connect the other end to the computer ethernet port or the router.

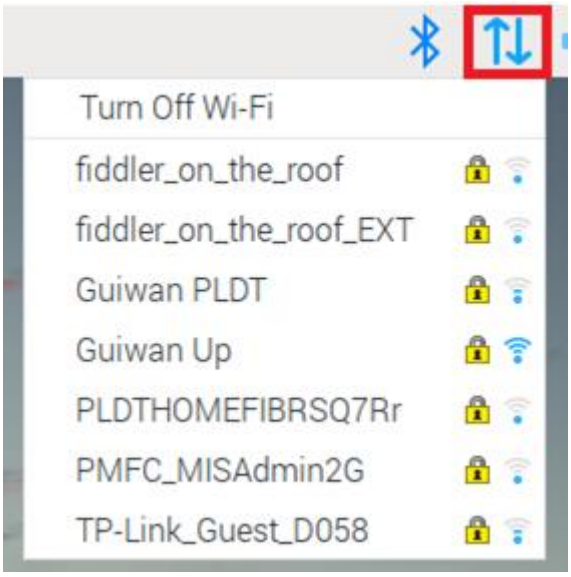
2. Once connected, open the remote desktop connection and type the RPI hostname then click the connect button. Press yes if there is a pop-up.

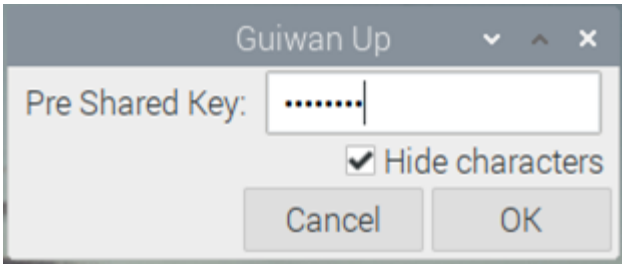


Then the user will have to login the username and password in the RPI login system.



Access the Wi-Fi network for internet connection.





2.3 Accessing the Web Application

To access the SHL Web Application, open a web browser and navigate to web server by typing "http://tanhairunpi/plants_main_menu.php" or http://192.168.0.106/plants_main_menu2.php then you will be directed to the main menu of SHL.

2.4 Handling Main Menu Window

Main Menu Window shows the 9 buttons, namely, Initialize Camera, Sensor Logs, Logs Intervals, Change Passcode, Adjust Schedule, Modify pH Settings,

Modify TDS Settings, Modify Temperature Settings, and Modify Water Level Settings; and, the current readings of grow lights schedule, pH Value, pH Target Value, TDS Value, TDS Target Value, Temperature, Temperature Target Value, Water Level and Water Level Target Value.

2.4.1 Initialize Camera

To view the live footage of the system, just click the Initialize Camera button.

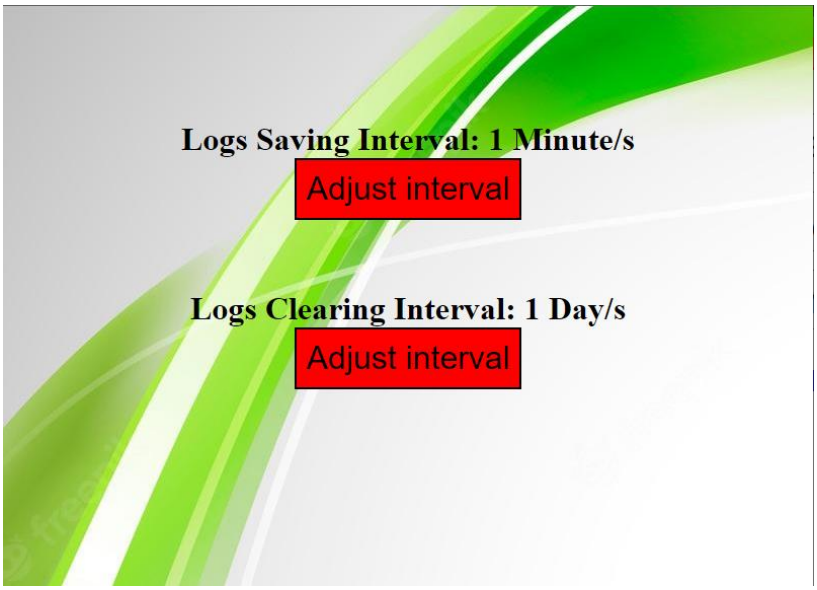
2.4.2 Viewing Sensor Logs

To view the sensor logs, press the Sensor Logs button.

10/03/2023 19:58:53 TDS= 785.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.68
10/03/2023 19:59:55 TDS= 785.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.61
10/03/2023 20:00:56 TDS= 785.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.69
10/03/2023 20:01:57 TDS= 791.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.6
10/03/2023 20:02:59 TDS= 782.0 WaterLvl= 100 Temperature= 29.0 PhLvl= 6.59
10/03/2023 20:04:00 TDS= 788.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.66
10/03/2023 20:05:02 TDS= 778.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.62
10/03/2023 20:06:03 TDS= 791.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.71
10/03/2023 20:07:05 TDS= 782.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.68
10/03/2023 20:08:07 TDS= 782.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.72
10/03/2023 20:09:08 TDS= 782.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.62
10/03/2023 20:10:10 TDS= 788.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.63
10/03/2023 20:11:11 TDS= 785.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.58
10/03/2023 20:12:13 TDS= 788.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.62
10/03/2023 20:13:14 TDS= 782.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.55
10/03/2023 20:14:16 TDS= 782.0 WaterLvl= 100 Temperature= 29.0 PhLvl= 6.59
10/03/2023 20:15:17 TDS= 785.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.6
10/03/2023 20:16:19 TDS= 785.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.67
10/03/2023 20:17:20 TDS= 782.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.63
10/03/2023 20:18:22 TDS= 782.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.57
10/03/2023 20:19:24 TDS= 788.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.66
10/03/2023 20:20:25 TDS= 791.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.61
10/03/2023 20:21:27 TDS= 788.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.67
10/03/2023 20:22:29 TDS= 782.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.57
10/03/2023 20:23:30 TDS= 785.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.67
10/03/2023 20:24:32 TDS= 785.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.51
10/03/2023 20:25:34 TDS= 782.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.66
10/03/2023 20:26:36 TDS= 778.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.6
10/03/2023 20:27:38 TDS= 778.0 WaterLvl= 50.0 Temperature= 29.0 PhLvl= 6.66
10/03/2023 20:28:39 TDS= 778.0 WaterLvl= 55.0 Temperature= 29.0 PhLvl= 6.73

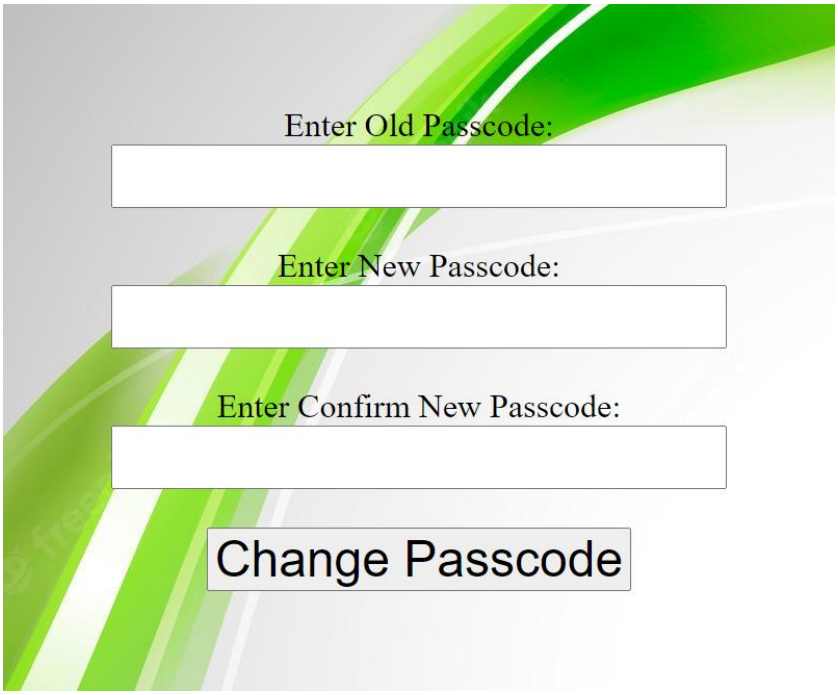
2.4.3 Log Intervals

To adjust the Log Saving Interval and the Log Clearing Interval, access the Log Interval tab.



2.4.4 Changing of Passcode

To change passcode, click the Change Passcode button to open the Change Passcode window.

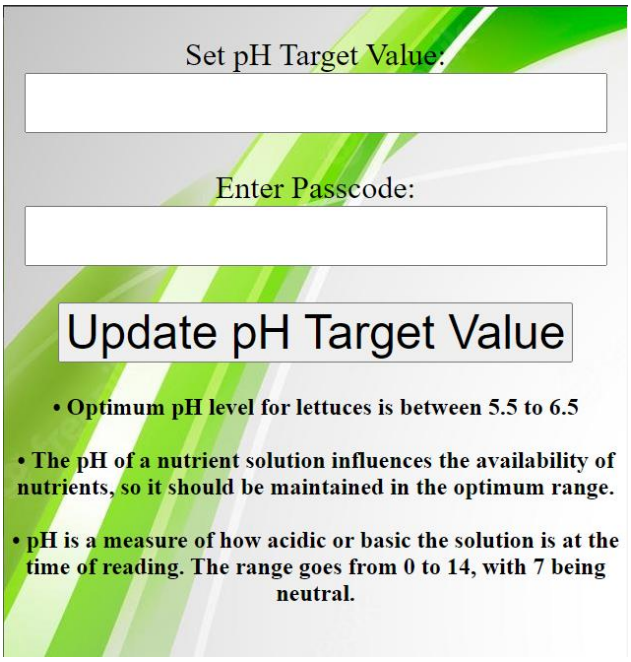


2.4.5 Adjusting Grow lights Schedule

To adjust he grow lights schedule, click the Adjust Schedule button to open the Grow Lights Schedule window.

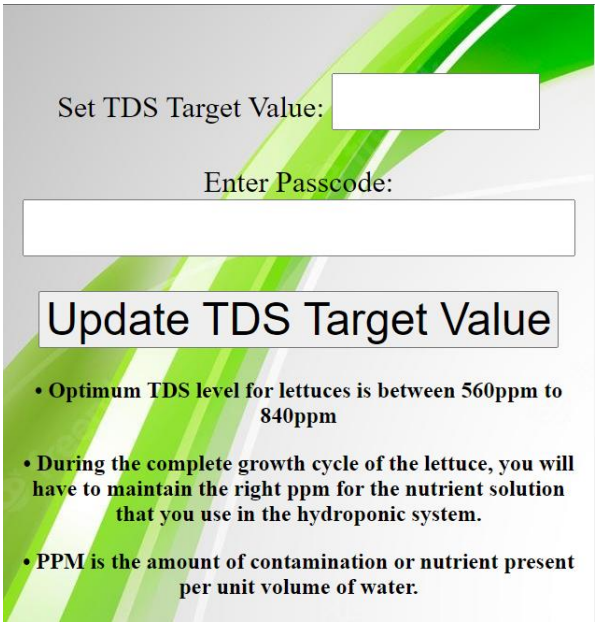
2.4.6 Modify pH Settings

To modify the pH Settings, just get into the Modify pH Settings tab. This tab contains the Set pH Target Value, Enter Passcode, Update pH Target Value Button, and information about pH and ideal pH Level of lettuces.



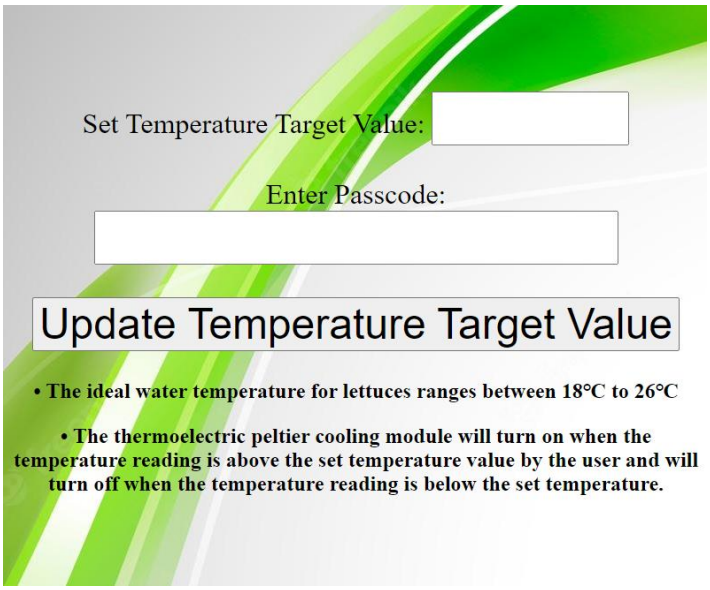
2.4.7 Modify TDS Setting

To modify the TDS Settings, just get into the Modify TDS Settings tab. In this tab, Set TDS Target Value, Enter Passcode, Update TDS Target Value Button, and a little information about TDS and ideal TDS Level are displayed.



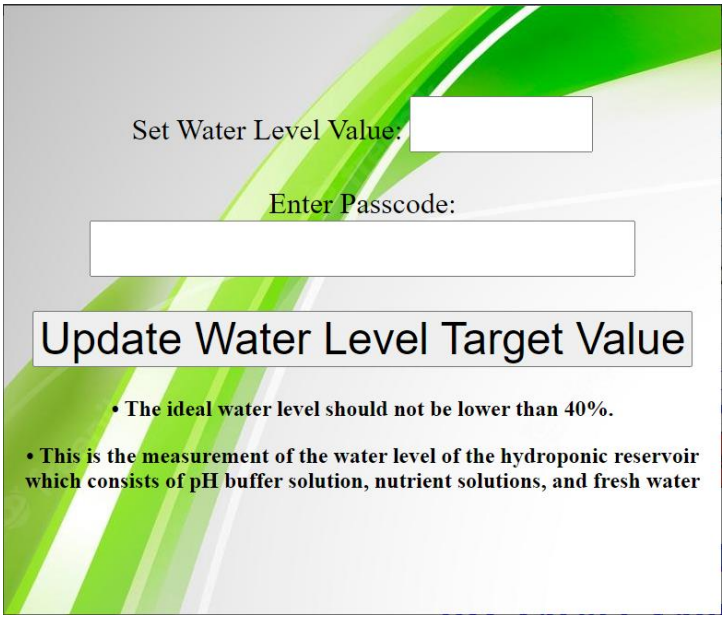
2.4.8 Modify Temperature Setting

To modify the Temperature Settings, just get into the Modify Temperature Settings tab. In this tab, Set Temperature Target Value, Enter Passcode, Update Temperature Target Value Button, and a little information about ideal water temperature and function of thermoelectric peltier cooling module are displayed.



2.4.9 Modify Water Level Settings

To modify the Water Level Settings, just get into the Modify Water Level Settings tab. In this tab, Set Water Level Value, Enter Passcode, Update Water Level Target Value Button, and a little information about water level and ideal water level are displayed.



3. Keypad Guide

Note: In using the keypad, the user must hold or press the button to be able to register.

3.1 Numerical Keypad Buttons

The numerical keypad buttons which are color blue as highlighted below, are used for inputting the desired set values and the user passcode for successfully changing the set values inputted by the user.

3.2 Symbol Keypad Buttons

The symbol keypad buttons which are highlighted below, function as the enter and return buttons. The number sign (#) is used if the user wants to confirm the new set target values and the passcode to successfully change the set values set by the user. The asterisk (*) is used if the user wants to go back to the menu of the LCD or to cancel the input set target values or the input user passcode.

3.3 Alphabetical Keypad Buttons

The alphabetical keypad buttons which are highlighted below, are used in configuring the pH value, TDS value, temperature, and water level of SHL. Each alphabetical keypad button is assigned in configuring the designated parameters. The keypad “A” is used in configuring the pH value. The keypad “B” is used in configuring the TDS value. The keypad “C” is used in configuring the Temperature. The keypad “D” is used in configuring the Water Level.

4. Sensor Calibration

4.1 pH Sensor Calibration

1. Measure 250ml of distilled water in any measuring cup or graduated cylinder, to place it into a glass.

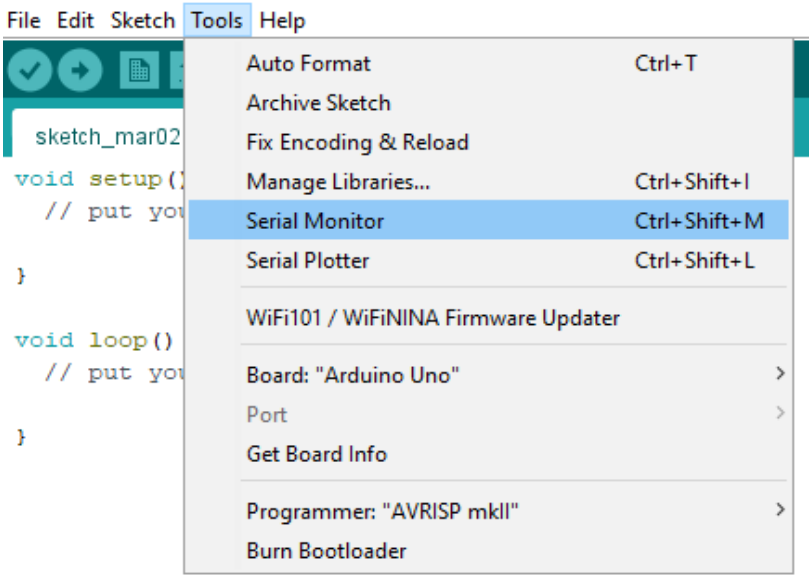
2. Put the pH Buffer Powder into the glass, empty it, then stir it until the powder has completely dissolved.

3. To get started with software calibration, the user must take note that this method uses mathematical method that to draw a line using two points, i.e. using the Acid standard solution, pH = 4.01 and alkaline pH = 9.18 to draw the linear relation between the voltage and the pH value.

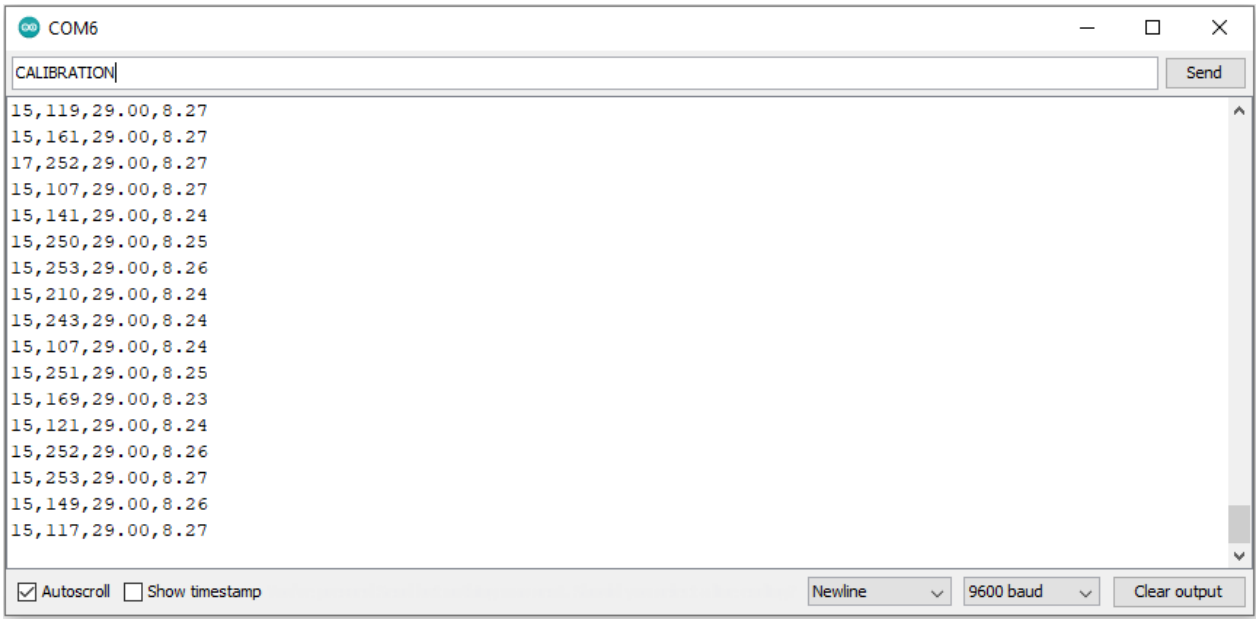
4. The user needs to open the ARDUINO Integrated Development Environment (IDE).

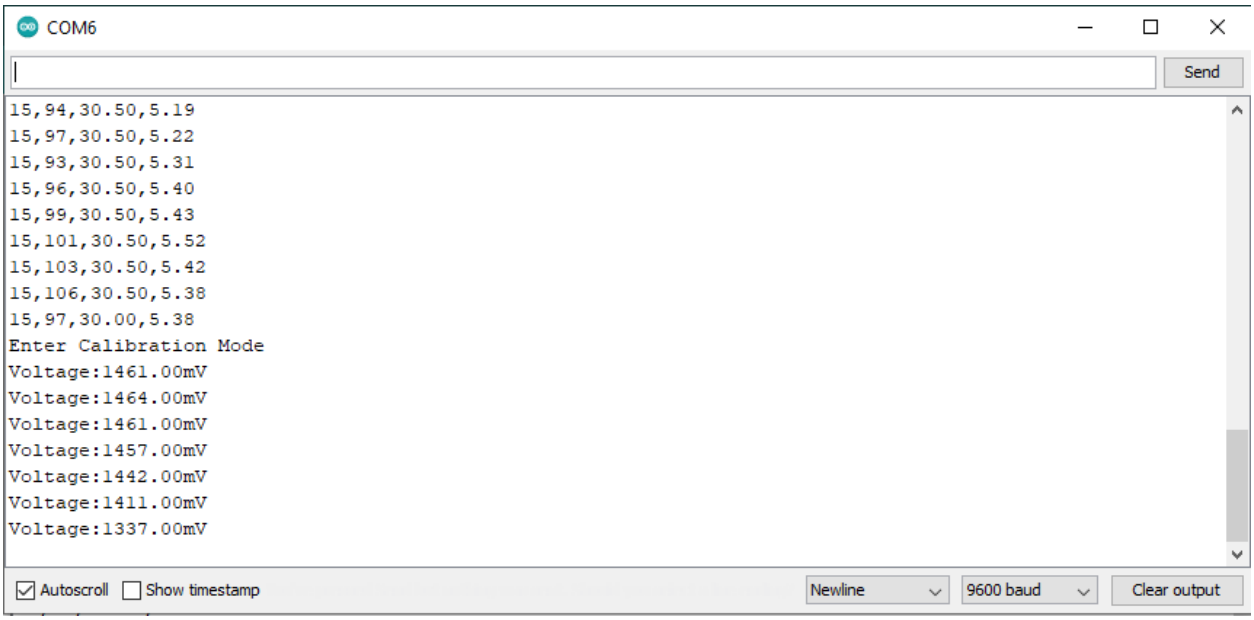
5. Go to Tools then open the Serial Monitor.

Note: Do not upload any file in the Arduino or it will overwrite the existing code in the Arduino.



6. Type the word “Calibration” then press enter to enter the Calibration Mode. After, the user will see “Enter Calibration Mode”. Then, proceed with Acid Calibration





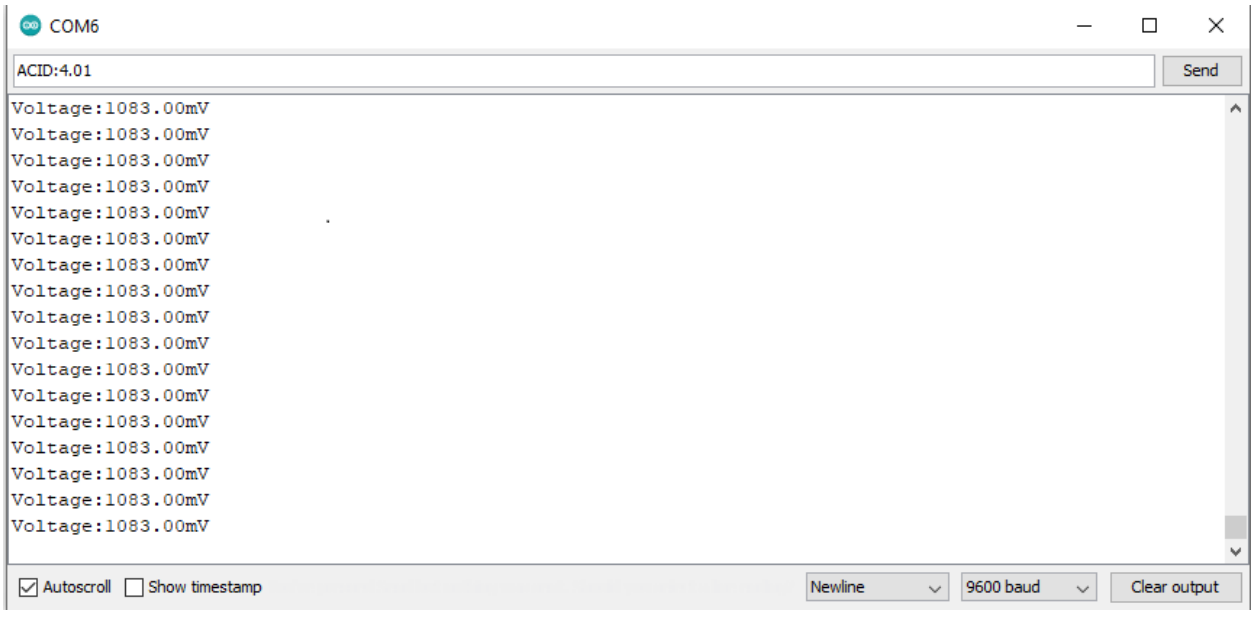
5.1.1 Acid Calibration

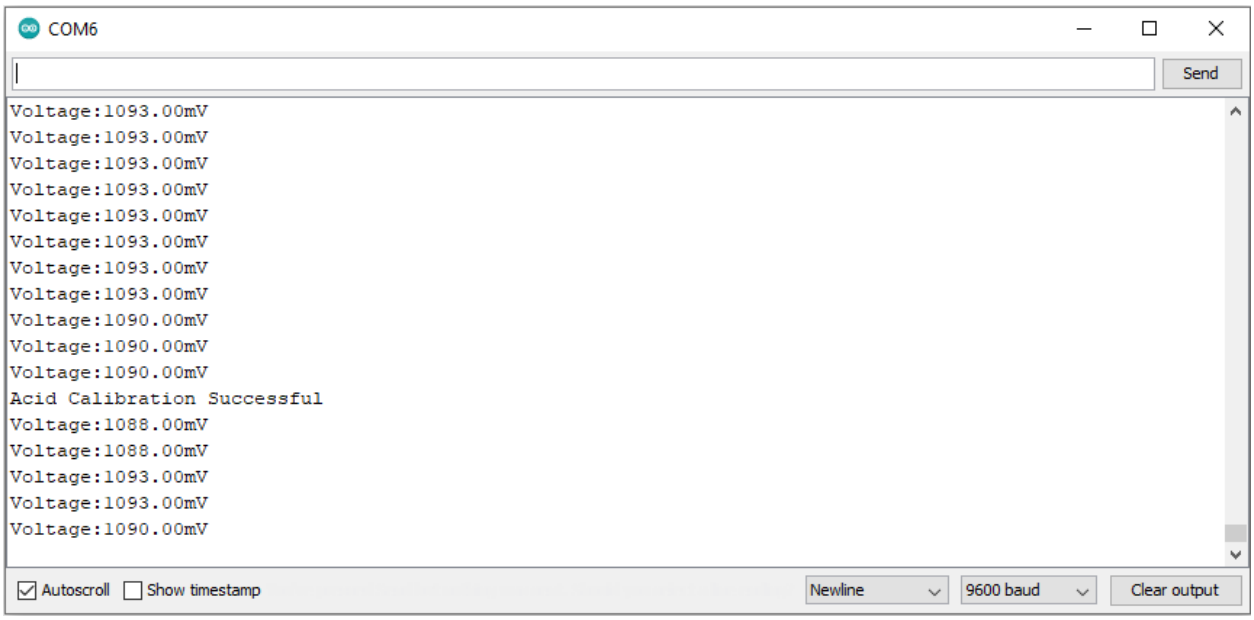
1. Clean the pH probe with pure water, but the Researchers suggest to clean the pH probe with distilled water.

2. Dry the pH probe using with absorbent paper.

3. Place it into standard acid solution of 4.01. Wait for not less than a minute till the readings get relative stable.

4. Input command “ACID:4.01” then press enter, after the user will see “Acid Calibration Successful”.





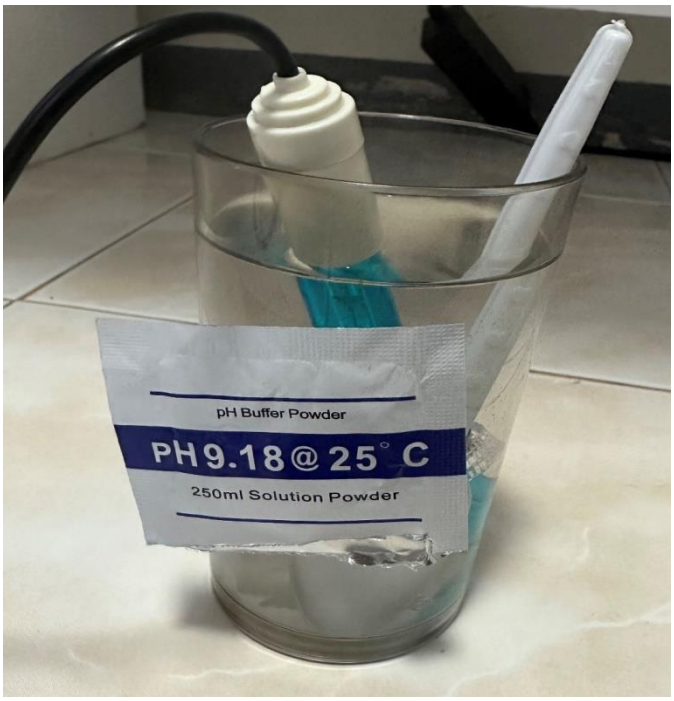
5. Take out the probe out of the acid solution, after proceed with Alkaline Calibration.

4.1.2 Alkaline Calibration

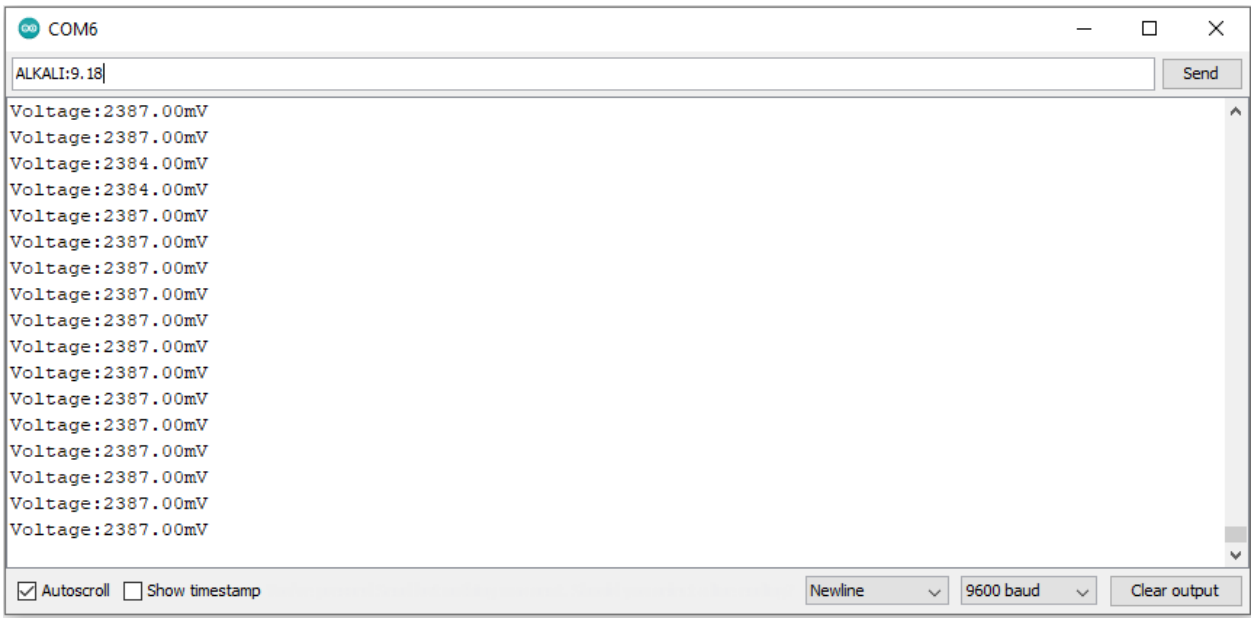
1. Clean again the pH probe with distilled water.

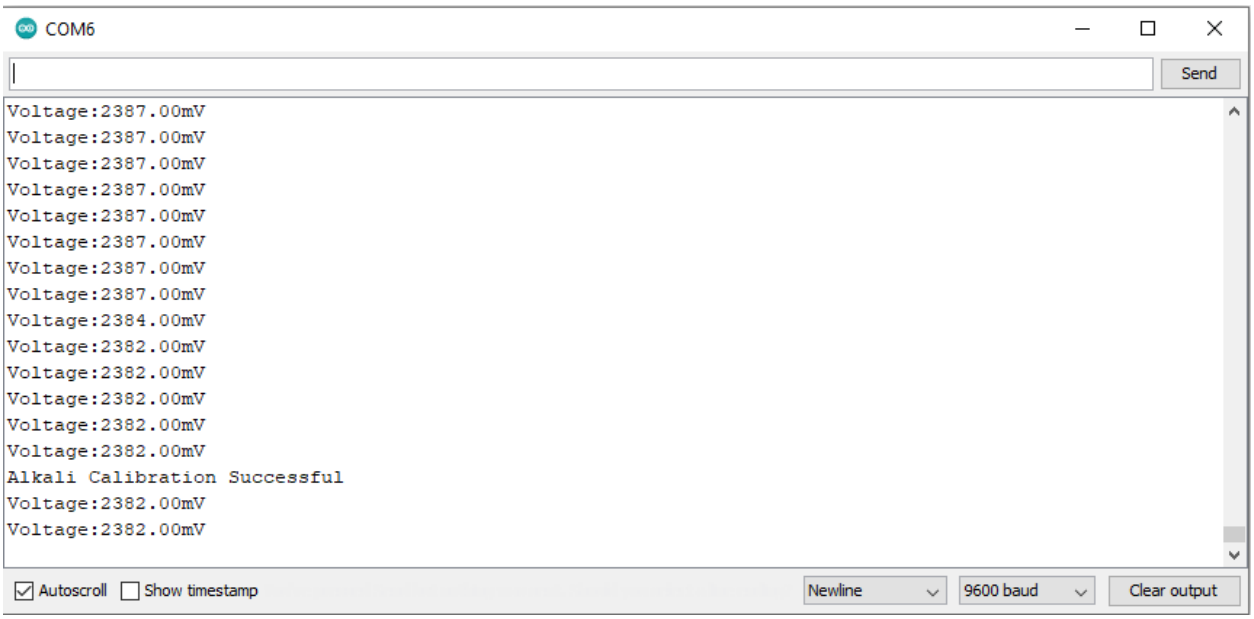
2. Dry the pH probe using with absorbent paper.

3. Place it into the standard alkaline solution with pH 9.18 solution. Wait for the readings to be stable.

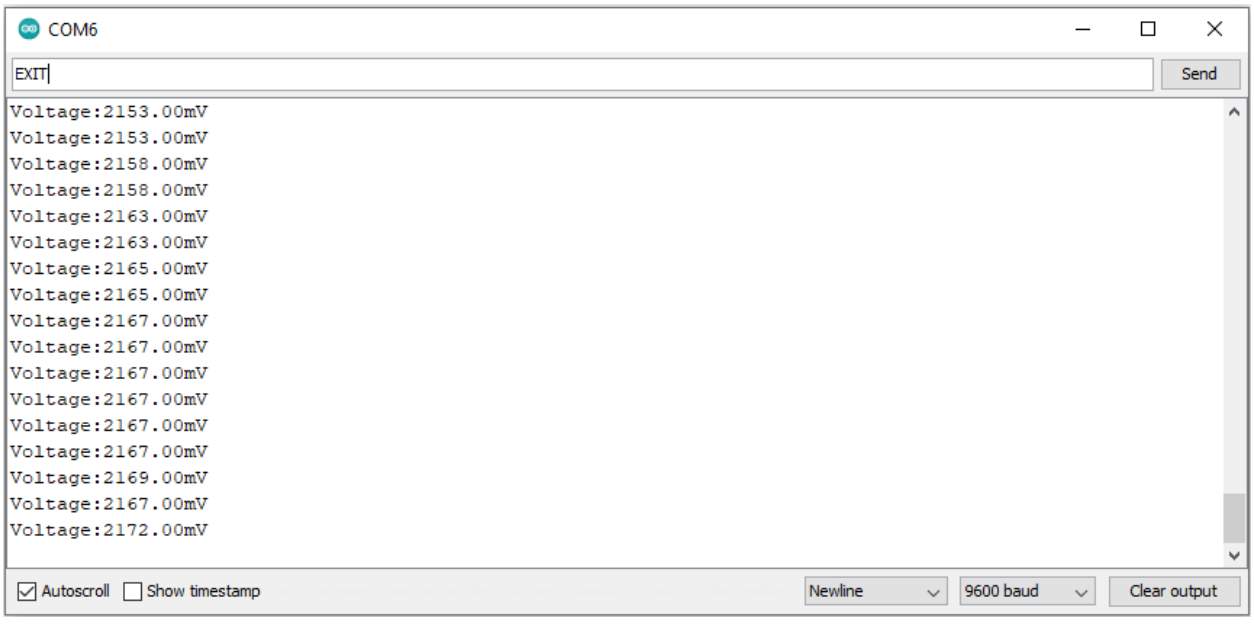


4. Input command “ALKALI:9.18” then press enter. After the user will see “Alkali Calibration Successful”





5. Input Command “EXIT” then press enter. After the user will see “Calibration Successful, Exit Calibration Mode”.



6. The pH Sensor is now well-calibrated.

Note: Check if the pH meter was calibrated successfully with the designated pH solutions. If the readings are with the error of 0.1, the user successfully calibrated the pH meter.

4.2 TDS Sensor Calibration

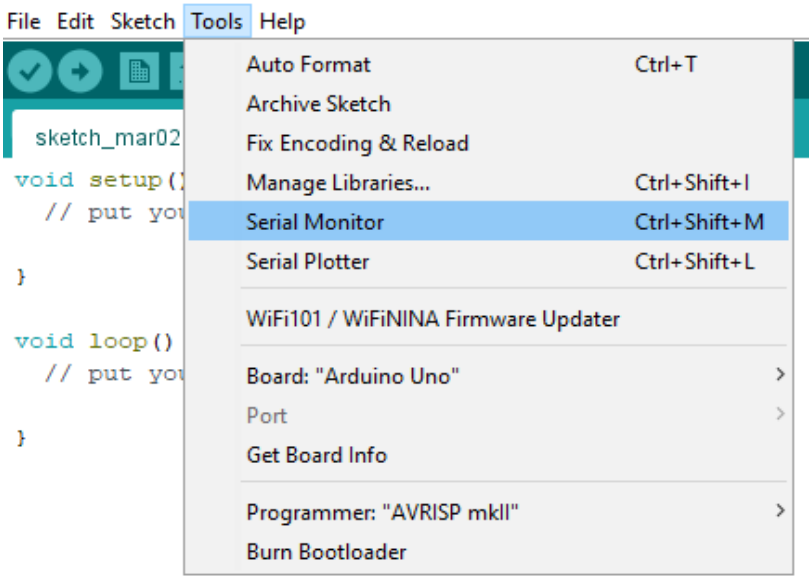
1. Clean the TDS probe with pure water, but the Researchers suggest to clean the TDS probe with distilled water.

2. Dry the TDS probe using with absorbent paper.

3. The user needs to open the ARDUINO Integrated Development Environment (IDE).

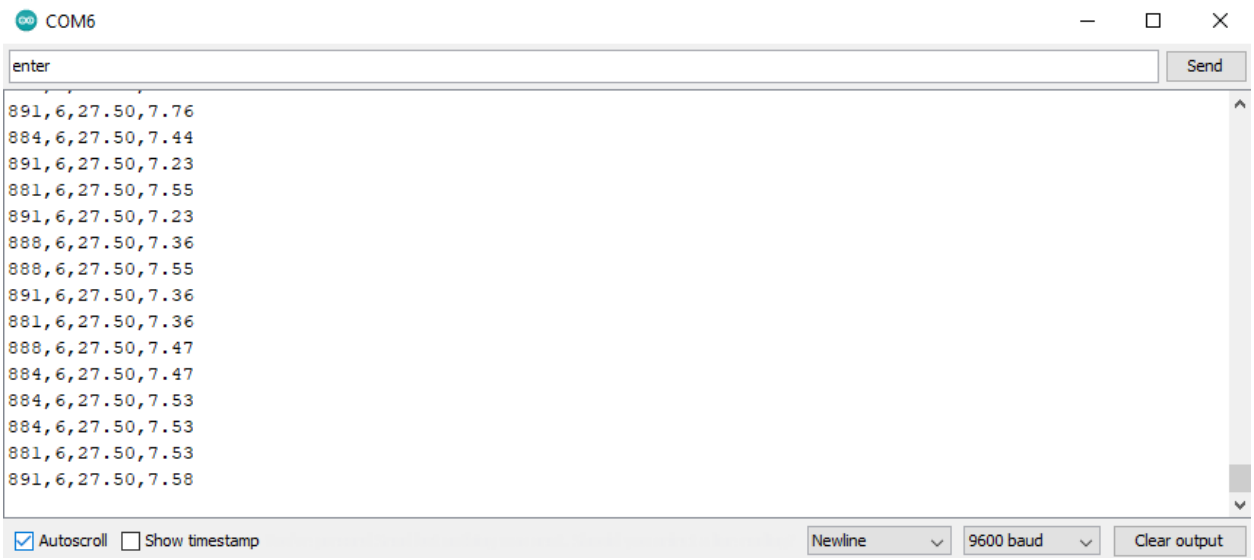
4. Go to Tools then open the Serial Monitor.

Note: Do not upload any file in the Arduino or it will overwrite the existing code in the Arduino.



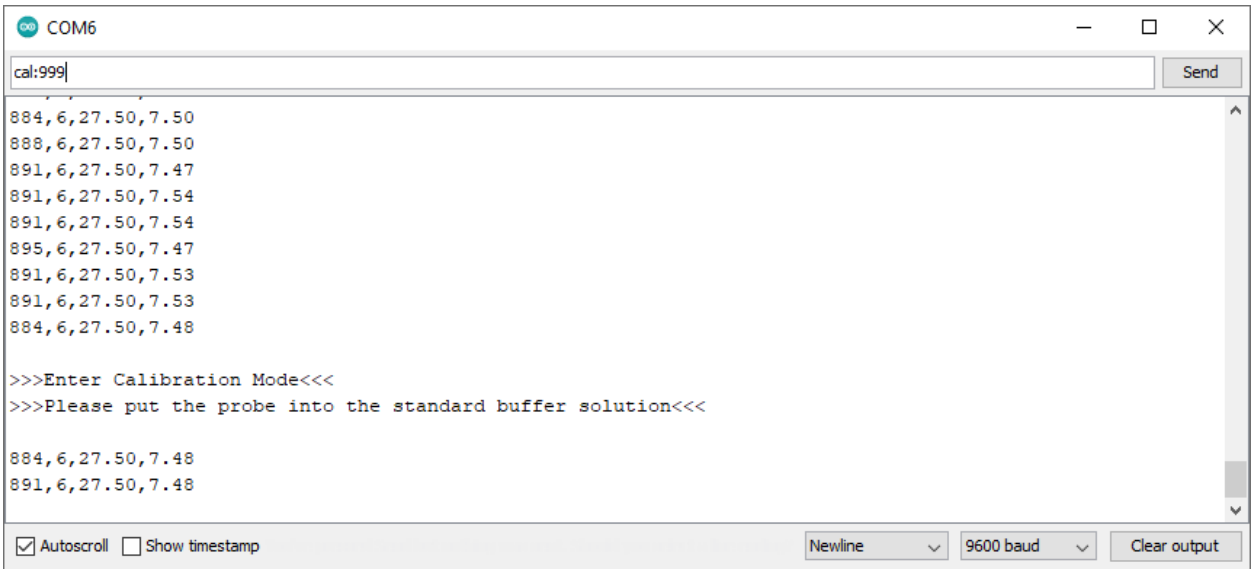
5. Place the probe into any known TDS value buffer solution. The Researchers suggest using the buffer solution as shown below. Then wait for the readings to be stable.

6. The user must input command “enter” to enter the calibration mode.

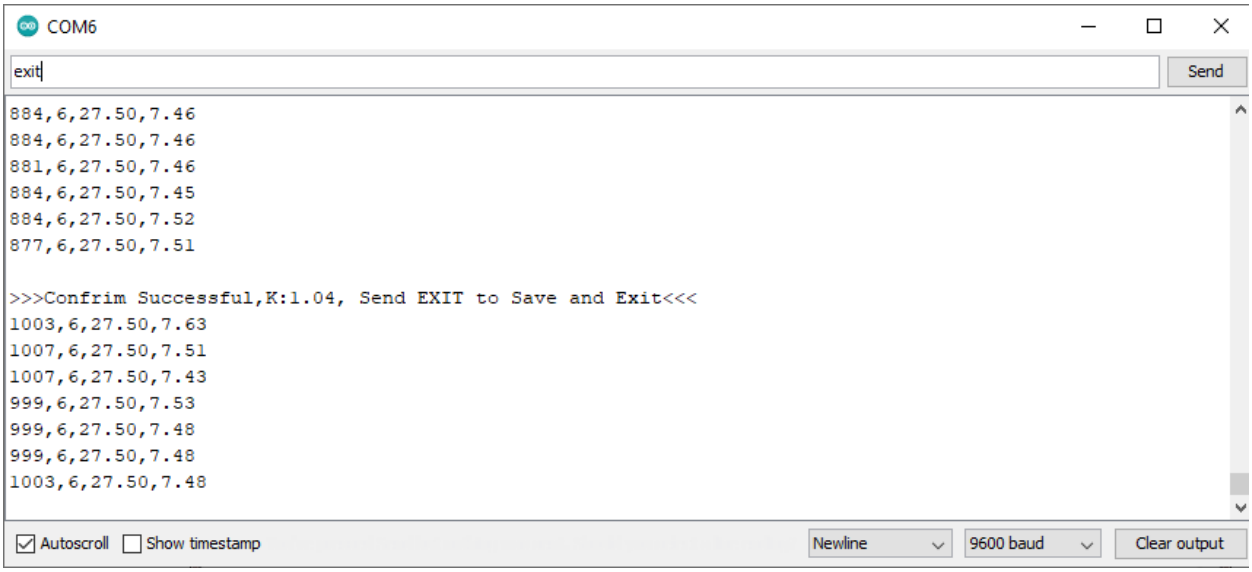


7. Input command “cal:tds value:” to calibrate the sensor. In this case input command “cal:999”.

Note: The calibration mode accepts the TDS value only up to 999ppm which is the maximum value.



8. Input command “exit”, to save and exit the calibration mode.



9. The TDS sensor is now well-calibrated.

5. Troubleshooting Guide

If the situation arose in which some features of the system are not working properly, please do take action by following the troubleshooting methods given below to resolve the problem and recover the functionality of the system. Along with the methods given below are the issues that may arise and may result to the impairment of system's functionality.

Issues/Problems	Troubleshooting Measures
The system is not running	**1.** Please make sure that Arduino is using the USB0. This can be done by making sure that only the Arduino is connected to the RPI USB ports, and then rebooting the RPI by unplugging and re-plugging the RPI's power supply.
Grow Lights Schedule is turning on or off on the wrong schedule	**1.** Check your internet connection to make sure you are connected to internet.
pH reading is not accurate	**1.** Please make sure that the pH sensor is well calibrated. Please refer to USER's MANUAL 5.1 for pH calibration
TDS reading is not accurate	**1.** Please make sure that the TDS sensor is well calibrated. Please refer to USER's MANUAL 5.2 for TDS calibration
Water level reading is not accurate	**1.** Please make sure that the Ultrasonic Sensor is placed properly.
No video is showing in the camera window	**1.** Please make sure that the camera is hooked properly. **2**. Replace the Flex Cable.

APPENDICES

APPENDIX A

HARDWARE DESIGN AND INTEGRATION

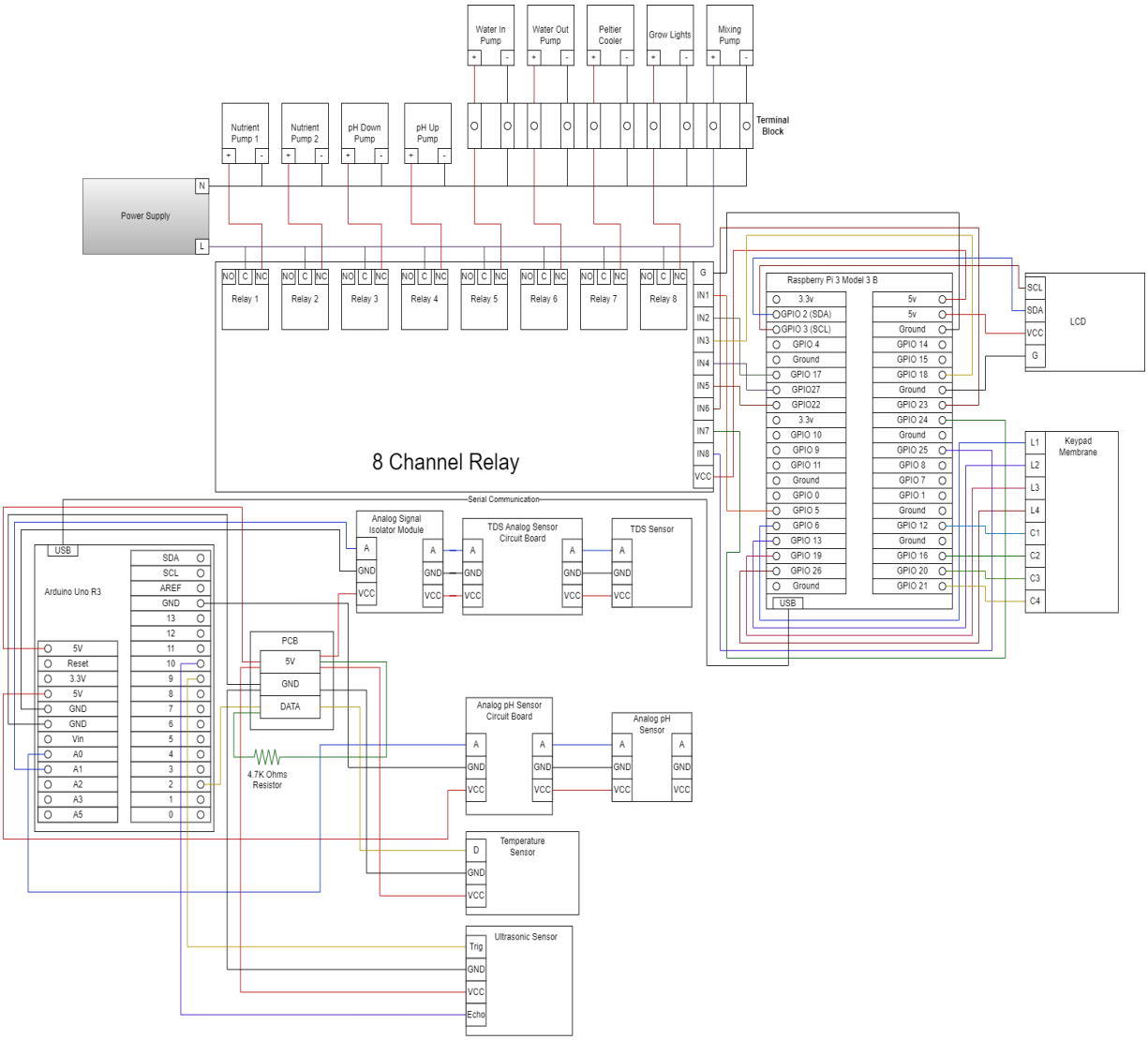


Figure A1. Overall Hardware Interconnection

APPENDIX B

SOFTWARE DESIGN

Figure B.1: TSHL MAIN MENU

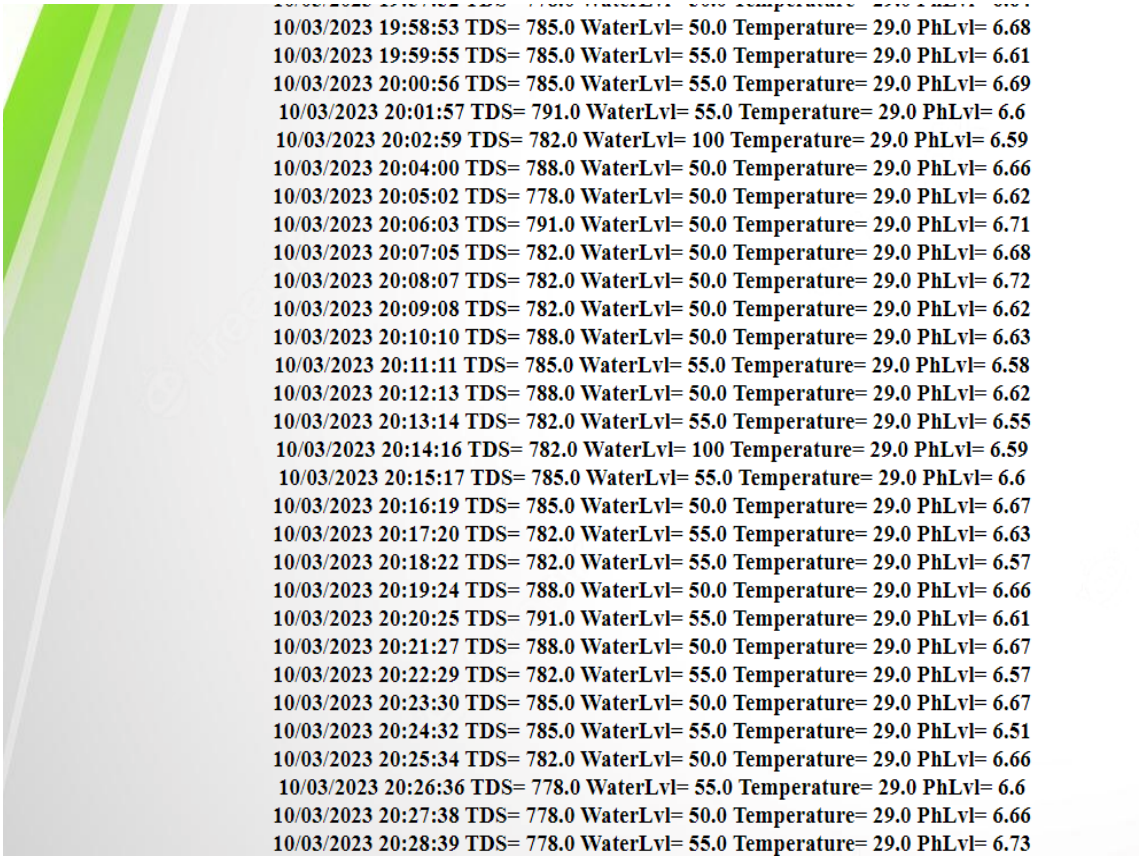


Figure B.2: History Log

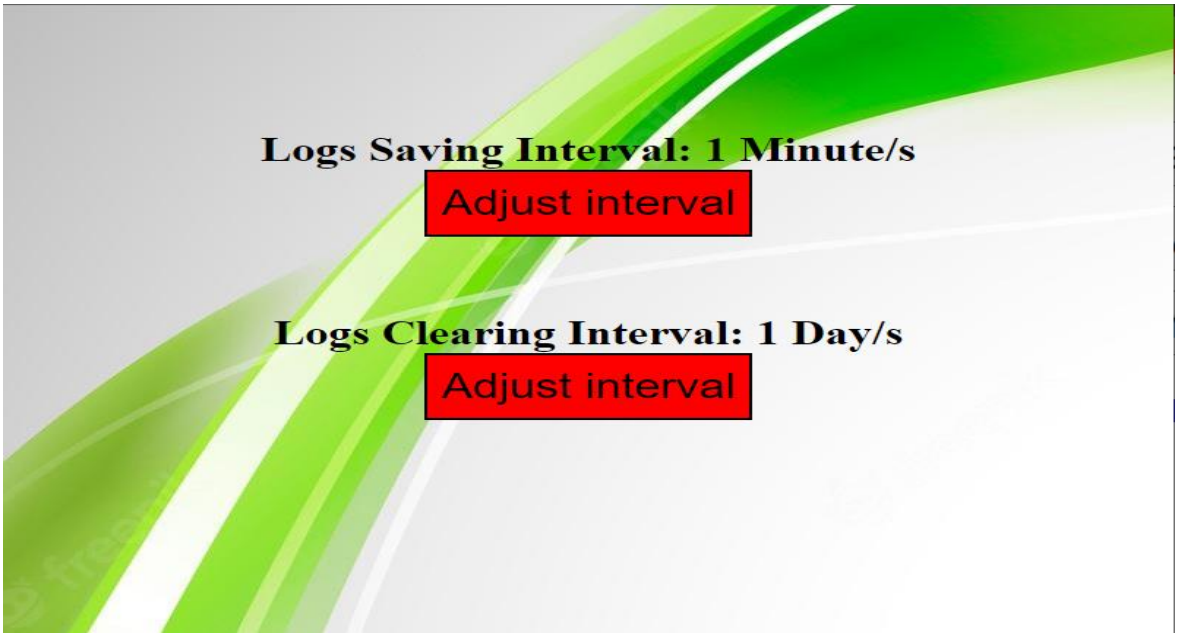


Figure B.3: Adjusting Intervals Page

Figure B.4: Form field for Adjusting Log Saving Interval

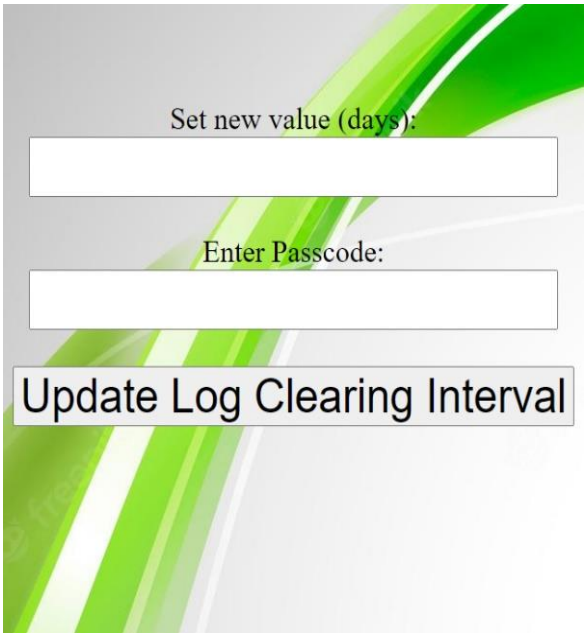


Figure B.5: Form field for Adjusting Log Clearing Interval

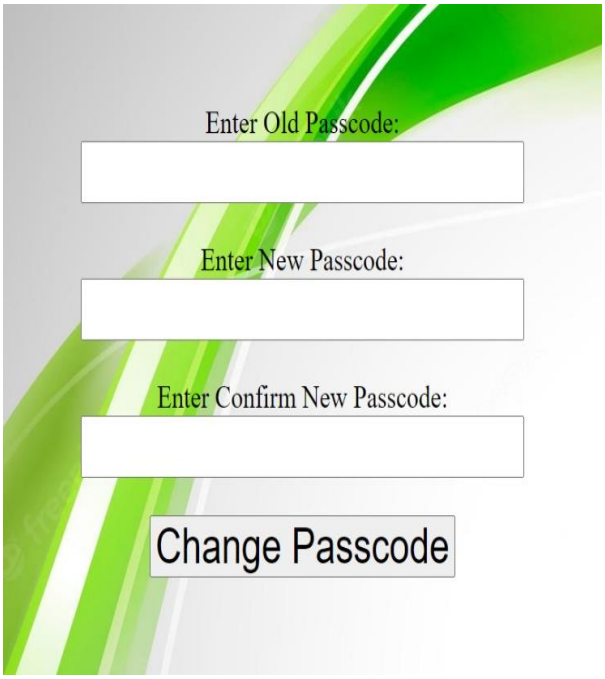


Figure B.6: Changing User Passcode

Figure B.7: Grow Light Scheduling

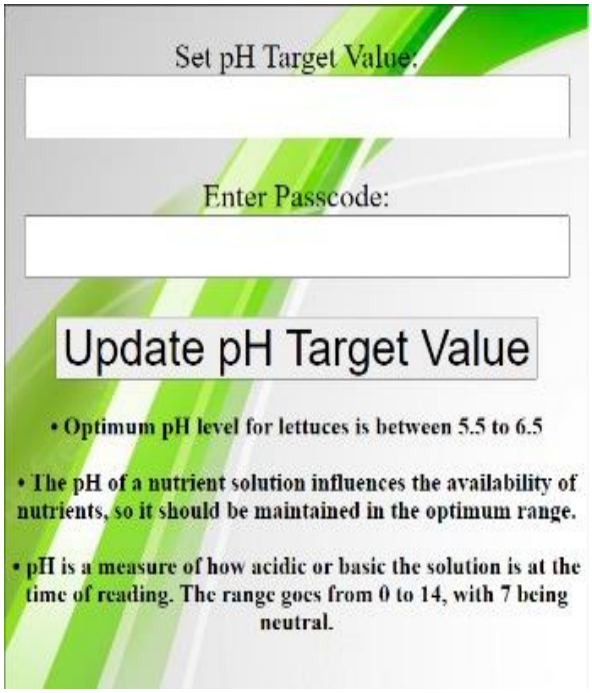


Figure B.8: Form field for Modifying pH Value

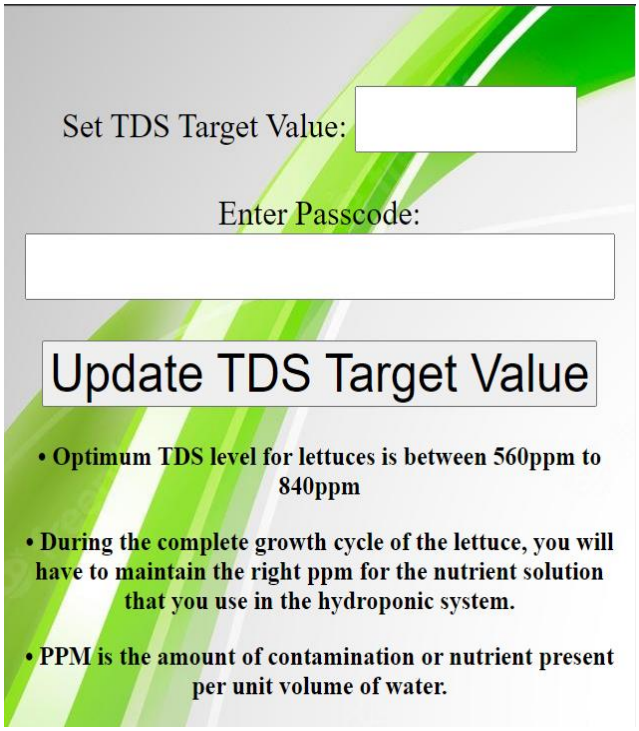


Figure B.9: Form field for Modifying TDS Value

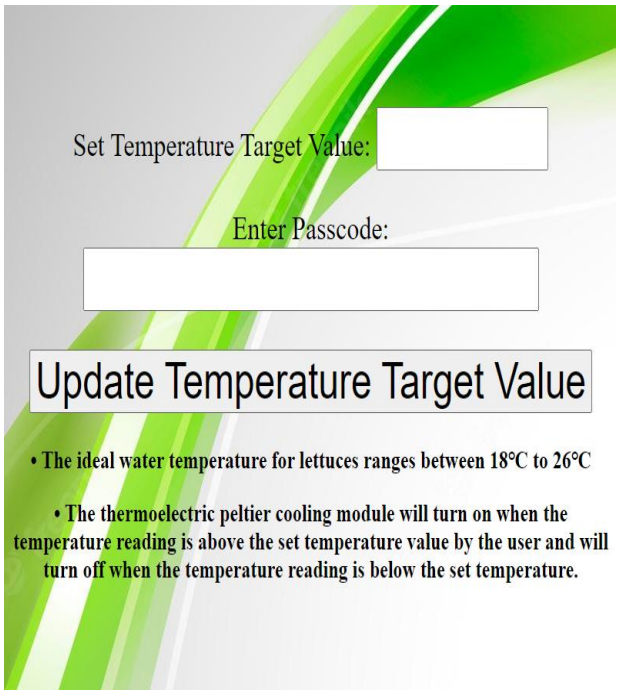


Figure B.10: Form field for Modifying Temperature

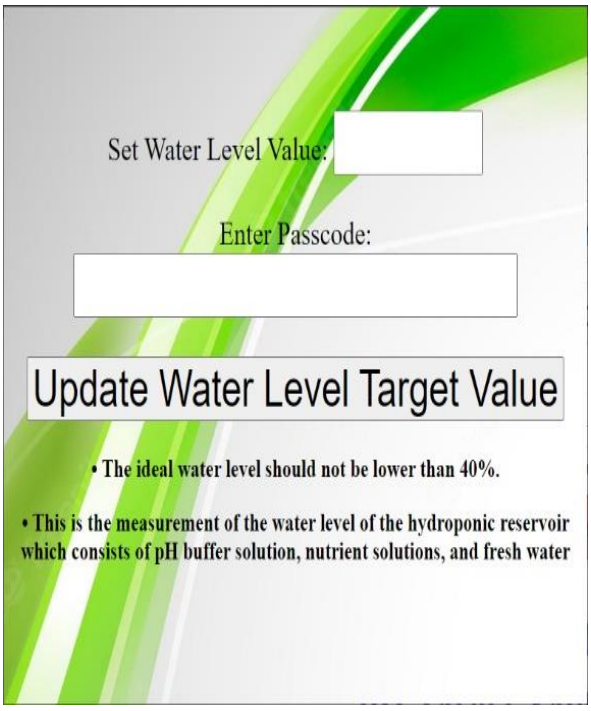


Figure B.11: Form field for Modifying Water Level

Figure B.12: Sample View of Camera Live Footage

APPENDIX C KEYPAD

MEMBRANE MANUAL

Please Hold the Button for at least 1 second to register.
Press "#" To Enter or for Confirmation
Press "*" To go back

Press keypad "A" to configure pH Value

After pressing the keypad "A", if the user wants to change the set target value, input the desired value that is within the allowed range from 0 to 1400. The inputted value will be divided by 100. For example, the inputted value is 650, divide it by 100, so the equivalent pH value is 6.5, then press "#" to confirm the new set target value. Enter the correct user passcode for the change to successfully happen, then press again "#" for confirmation. If the user wants to go back to the Main Menu, just press the "*". The optimum pH level for lettuces is between 5.5 to 6.5.

Press keypad "B" to configure TDS Value

After pressing the keypad "B", if the user wants to change the set target value, input the desired value that is within the allowed range from 0pmm to 1500ppm. After inputting the value, press "#" to confirm the new set target value. Enter the correct user passcode for the change to successfully happen, then press again "#" for confirmation. If the user wants to go back to the Main Menu, just press the "*". The optimum TDS level for lettuces is between 560pmm to 840ppm.

Press keypad "C" to configure the Temperature

After pressing the keypad "C", if the user wants to change the set target value, input the desired value. The inputted value will be divided by 100. For example, the inputted value is 2800, divide it by 100 so the equivalent temperature value is 28°C, then press "#" to confirm the new set target value. Enter the correct user passcode for the change to successfully happen, then press again "#" for confirmation. If the user wants to go back to the Main Menu, just press the "*".

Press keypad "D" to configure Water Level

After pressing the keypad "D", if the user wants to change the set target value, input the desired value that is within the allowed range from 0 to 100. This will be the target percent of the water level of the hydroponic reservoir. Then press "#" to confirm the new set target value. Enter the correct user passcode for the change to successfully happen, then press again "#" for confirmation. If the user wants to go back to the Main Menu, just press the "*".

APPENDIX D

RESUME

AL-FITRI C. HAIRUN

fitrihairun@gmail.com

09276961997

3rd Road, A.S. Alvarez Drive, D.A. Candido Subdivision, Don Alfaro Street, Tetuan, Zamboanga City, Zamboanga del Sur, Philippines, 7000

CAREER OBJECTIVE

To pursue a challenging and rewarding career in software development, hardware design, or systems engineering using my strong technical skills and knowledge in computer engineering. I aim to contribute to the development of innovative technologies and solutions that improve the efficiency, security, and functionality of computer systems, while continuously expanding my knowledge and expertise in emerging trends and tools in the field.

PERSONAL DATA

Age	:	26
Sex	:	Male
Civil Status	:	Single
Date of Birth	:	February 8, 1997
Citizenship	:	Filipino
Father	:	Hasan H. Hairun (Deceased)
Occupation	:	Retired school teacher
	:	Zamboanga Chong Hua Highschool
Mother	:	Emma C. Hairun
Occupation	:	Businesswoman
Religion	:	Islam
Languages/Dialects	:	English, Filipino, Chinese (fair), Chavacano, Visayan, Tausug and Samal

EDUCATIONAL BACKGROUND

College Studies	:	Bachelor of Science in Computer Engineering Ateneo de Zamboanga University Zamboanga City; Class 2023
Secondary Studies	:	Zamboanga Chong Hua High School Zamboanga City; Class 2014
Elementary Studies	:	Zamboanga Chong Hua High School Zamboanga City; Class 2010

SKILLS AND ABILITIES

- Computer literate and Programmer. Has knowledge of basic computer programs and languages, respectively.
- Knowledgeable in electronic hardware and computer repair
- Highly trainable
- Good interpersonal skills
- Professional, optimistic, flexible, results-oriented and with good work ethics

SEMINARS/TRAINING ATTENDED

Seminar on "2nd Regional ICpEP Convention and 4th Regional CpE Student Challenge 2019", Grand Astoria Hotel, Zamboanga City – September 12-14, 2019

Seminar on "REACT JS Laboratory Session", Ateneo de Zamboanga University, Zamboanga City – September 12-14, 2019

Seminar on "ROBOTICS Tech Talk", KCC Mall de Zamboanga, Zamboanga City – September 12-14, 2019

Seminar on "Building an IOT-Based System Tech Talk", KCC Mall de Zamboanga, Zamboanga City – September 12-14, 2019

Seminar on "Cisco: Router and Switching", Fr Salvador Campus, Ateneo de Zamboanga University, La Purisima St., Zamboanga City – August 10, 2019

On-The-Job Training 300 hours at Lenin Computer Systems, INC. Zamboanga City – April 8, 2019 to July 11, 2019

Seminar Workshop on “Microcontrollers and the Internet of Things”, FWS Building, Ateneo de Zamboanga University, La Purisima St., Zamboanga City – January 25, 2019

Seminar on “Antenna and Radio Transmission”, Ateneo de Zamboanga University, Zamboanga City – January 26, 2018

Seminar on “Cybersecurity: Building a Safe and Productive Online Environment”, Ateneo de Zamboanga University, Zamboanga City – May 2, 2017

Seminar on “Zamboanga Tech Caravan & Hackathon 2017”, STI College, Zamboanga City – April 23, 2017

Seminar on “CyberSecurity Caravan”, Summit Center, Universidad de Zamboanga, Zamboanga City – April 20, 2017

Seminar on “Strengthening the Culture of Excellence”, New Laboratory Building, Universidad de Zamboanga, Tetuan, Zamboanga City – September 3, 2016

Seminar on “Environmental Engineering and Safety”, Ateneo de Zamboanga University, Zamboanga City – August 31, 2016

CERTIFICATE

NCII in Computer Systems Servicing
Issued on: July 5, 2019

REFERENCES

Engr. Louie Virgil A. Gallardo
Engineering Department Chair
Ateneo de Zamboanga University

Engr. Niño Christon Lazarte
Engineering Department Faculty
Ateneo de Zamboanga University

Engr. Zaeefa A. Pandangan
Engineering Department Faculty
Ateneo de Zamboanga University

ALVIN T. TAN

140729alvintan@gmail.com

09777762125

Aurora Street, Guiwan, Zamboanga City 7000

CAREER OBJECTIVE

To be able engage in training where I can enhance my skills in the field of computer engineering and gain experience to start a career as a computer engineer with the hopes of eventually becoming a fully qualified computer engineer.

PERSONAL DATA

Age	:	25
Sex	:	Male
Civil Status	:	Single
Date of Birth	:	March 7, 1998
Citizenship	:	Filipino
Father	:	Alan T. Tan
Occupation	:	Businessman
Mother	:	Ledwina T. Tan
Occupation	:	Businesswoman
Religion	:	Roman Catholic
Languages/Dialects	:	English, Filipino, Tausug and Chavacano

EDUCATIONAL BACKGROUND

College Studies	**:**	Bachelor of Science in Computer Engineering Ateneo de Zamboanga University Zamboanga City; Class 2023
Secondary Studies	**:**	Pilar College Zamboanga City; Class 2014
Elementary Studies	**:**	Pilar College Zamboanga City; Class 2010

SKILLS AND ABILITIES

- Intermediate programming knowledge on C++, Java, PHP, Arduino, Python, and Assembly Language.
- Average skills on electronic hardware and has experience in programming MCU Arduino and Raspberry Pi.
- Great Communication Skills
- Good Planner and Organizer

SEMINARS/TRAINING ATTENDED

Seminar on "2nd Regional ICpEP Convention and 4th Regional CpE Student Challenge 2019", Grand Astoria Hotel, Zamboanga City – September 12-14, 2019

Seminar on "REACT JS Laboratory Session", Ateneo de Zamboanga University, Zamboanga City – September 12-14, 2019

Seminar on "ROBOTICS Tech Talk", KCC Mall de Zamboanga, Zamboanga City – September 12-14, 2019

Seminar on "Data Mining Tech Talk", KCC Mall de Zamboanga, Zamboanga City – September 12-14, 2019

Seminar on "Building an IOT-Based System Tech Talk", KCC Mall de Zamboanga, Zamboanga City – September 12-14, 2019

Seminar on "Cisco: Router and Switching", Fr Salvador Campus, Ateneo de Zamboanga University, La Purisima St., Zamboanga City – August 10, 2019

Seminar Workshop on "Microcontrollers and the Internet of Things", FWS Building, Ateneo de Zamboanga University, La Purisima St., Zamboanga City – January 25, 2019

Seminar on "LTE Advanced: Road Towards 5G Convergence; Internet of Things: Identifying the Best Fit Access Solution For Your Business Needs", Terra Cotta, Tumaga, Zamboanga City – May 12, 2018

On-The-Job Training 300 hours at Emedia Production Network INC. Zamboanga City – April 5, 2018 to June 8, 2018

Seminar on "Antenna and Radio Transmission", Ateneo de Zamboanga University, Zamboanga City – January 26, 2018

Seminar on “Cybersecurity: Building a Safe and Productive Online Environment”, Ateneo de Zamboanga University, Zamboanga City – May 2, 2017

Seminar on “Zamboanga Tech Caravan & Hackathon 2017”, STI College, Zamboanga City – April 23, 2017

Seminar on “Strengthening the Culture of Excellence”, New Laboratory Building, Universidad de Zamboanga, Tetuan, Zamboanga City – September 3, 2016

REFERENCES

Engr. Louie Virgil A. Gallardo
Engineering Department Chair
Ateneo de Zamboanga University

Engr. Niño Christon Lazarte
Engineering Department Faculty
Ateneo de Zamboanga University

Engr. Zaeefa A. Pandangan
Engineering Department Faculty
Ateneo de Zamboanga University