INC281 Capstone Project Report

"Monitoring System for Hydroponic Medical Marijuana"

Present to

Dr. Santi Nuratch

Members

Mr. Pattarapon Buathong	Student ID: 62070504012
Ms. Suchada Panyawiraphat	Student ID: 62070504016
Ms. Pawornrat Supavit Pranayanuntana	Student ID: 62070504030

Semester 2/2021

Department of Control System and Instrumentation Engineering

King Mongkut's University of Technology Thonburi

Abstract

This paper presents the development of a monitoring system for a medical marijuana hydroponic farm, providing its automation and quantitative assessment of the farm performance. An increasing popularity of indoor hydroponic farming has been seen in the last decade thanks to the development of energy-efficient and affordable LED lights. However, the optimal configuration of such systems (i.e., the amount of nutrients and water, ambient temperature, light intensity etc.) is still mainly determined by the farmers' experience and empirical guidelines. In addition, even if simple, the maintenance of such systems is labor-intensive and time-consuming because it demands periodically supplying water, nutrient mixing, and other tasks for cultivation. To unlock the full potential of farming, a quantitative understanding of the impact of how each factor contributes to plant growth is needed, as well as adequate knowledge of automation and IoT (Internet of Things). The monitoring system proposed in this paper is a step towards filling this knowledge and the technological gap in agricultural farming. It allows for the collection and viewing of sensor data such as temperature, humidity, light intensity, pH, water pump status, air pump status, movement detection, and water level detection without requiring any human interaction within the farm. Data can be accessed remotely via a simple web interface. The proposed platform can quantitatively optimize the cultivation of the plants and automate some of the most labor-intensive maintenance activities. Moreover, such a monitoring system can also potentially be used for high-level decision making once enough data are collected. In the project, medical marijuana was chosen as the output of cultivation to demonstrate how this system would be beneficial in a real-life scenario.

Table of Content

Abstract i

Table of Content ii

Table of Figure iii

Introduction iv

Section I: System Overview 1

General Circuit Designing 4

Converting the Range 0-3.33V to 0-1023 6

Section II: Sensors 7

Analog Sensors 7

Temperature Sensor 7

Light Intensity Sensor 10

Relative Humidity Sensor 12

pH Sensor 15

Digital Sensors 16

Water Level Detector 16

Motion Detector 17

The Air Pump 18

The Water Pump 19

Section III: Web-based User Interface (Application) 20

Section IV: Conclusion 24

Gantt chart 25

Appendix 26

References 35

Table of Figure

Figure 1. The Nutrient Film Technique 1
Figure 2. Microcontroller Circuit 2
Figure 3. System Block Diagram 3
Figure 4 Non-Inverting Op-Amp Circuit Design 5
Figure 5. Circuit Design 5
Figure 6. Functional Flow Block Diagram for Temperature Sensor 7
Figure 7. Circuit Design Temperature Sensor 8
Figure 8. MAX1044 Circuit from Datasheet 9
Figure 9. Temperature/Cooling Fan Circuit 9
Figure 10 Functional Flow Block Diagram for Light Dependent Resistor 10
Figure 11. Circuit Design Light Dependent Resistor 11
Figure 12. LED/Grow Lamp Circuit 11
Figure 13 Functional Flow Block Diagram for Relative Humidity Sensor 12
Figure 14 Inverting Op-Amp Circuit (Gain = 1) 13
Figure 15 Circuit Design Relative Humidity Sensor 14
Figure 16 HIH-5030 Circuit from Datasheet 14
Figure 17. Humidifier Fan Circuit 15
Figure 18. Functional Flow Block Diagram for pH Sensor 15
Figure 19. Functional Flow Block Diagram for Water Level Detector 16
Figure 20. Functional Flow Block Diagram for Motion Detector 17
Figure 21. Buzzer Circuit 17
Figure 22. Functional Flow Block Diagram for Air Pump 18
Figure 23. Air Pump Circuit 18
Figure 24. Functional Flow Block Diagram for Water Pump 19
Figure 25. Water Pump Circuit 19
Figure 26. Website Dashboard 20
Figure 27. The Home Page of the Website 20
Figure 28. Farm Information Page 21
Figure 29. Customer Reviews Section 22

Introduction

In the past few years, there has been a rise in the usage and demand for medical marijuana as more countries continue to legalize the drug. Entrepreneurs wanting to profit from this growing demand, are looking to mass-produce the plant to sell to their consumers. However, the traditional farms used to cultivate medical marijuana alongside farms used for growing food crops, are putting unsustainable pressure on Earth's ecosystems. Establishing more of these traditional systems for business will only be detrimental to the surrounding environment and further deplete our limited resources. Hence, it is certain that we need an alternative way to effectively grow produce without taking up as much space or resources as traditional farming would. Hydroponics, combined with an automatic monitoring system, present a compelling solution that could benefit both producers and consumers in the long term. Our project illustrates what a simulation of this system would look like for monitoring the cultivation of medical marijuana.

This paper is organized as follows. The system's overall infrastructure and circuit used to manage the farm and collect the data is illustrated in Section I. Subsequently, the choice of sensors and calculations converting sensor signal to microcontroller circuit input is discussed in Section II, essentially describing each unit of the circuit in Section I in more detail. The theory for general circuit designing will also be discussed prior to this section. Next, the data collection and transmission platform will be explained in Section III along with how to use it. Finally, some conclusions are reported in Section IV, together with suggestions for further work.

Section I: System Overview

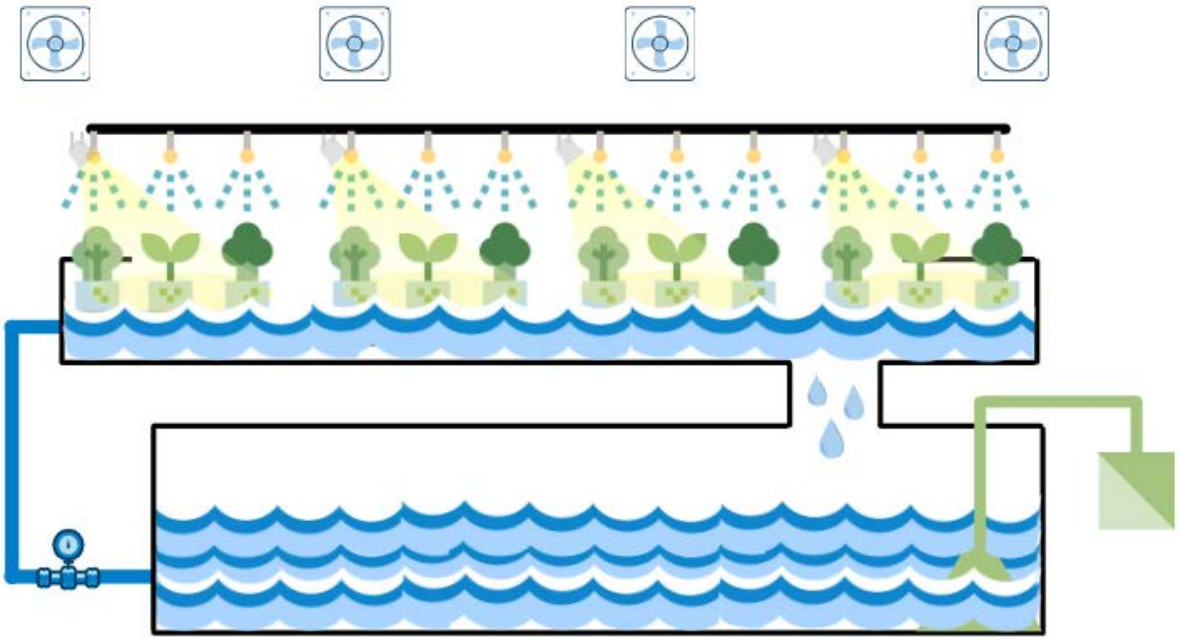


Figure 1. The Nutrient Film Technique

Figure 1 illustrates how the hydroponic system would be arranged inside our greenhouse. The setup follows the Nutrient Film Technique, or NFT for short. There are several benefits to NFT: its simplicity, low cost, and low maintenance, which explains the popularity of the method amongst both hydroponic hobbyists and commercial-scale farms despite varying production sizes. NFT is also very modular and expandable, another reason why farm factories can easily construct them for large-scale production.

The system uses a water pump to deliver fertilized water from the water tank to the grow tray. The drainpipe then recycles the unused nutrient solution. Due to gravity, the nutrient solution is constantly flowing over the roots in NFT. The grow tray is at an angle that allows water to flow down the drainpipe and back into the water tank as a new solution continuously feeds into the grow tray's high end. Unfortunately, this "angle" is not shown in Figure 1 due to the designer wanting the picture to look aesthetically pleasing. Nevertheless, the illustration still depicts the functionality of how each actuator and sensor contributes to the system overall. NFT enables the nutrient solution to flow in a thin film over the roots, watering and feeding them without completely soaking them.

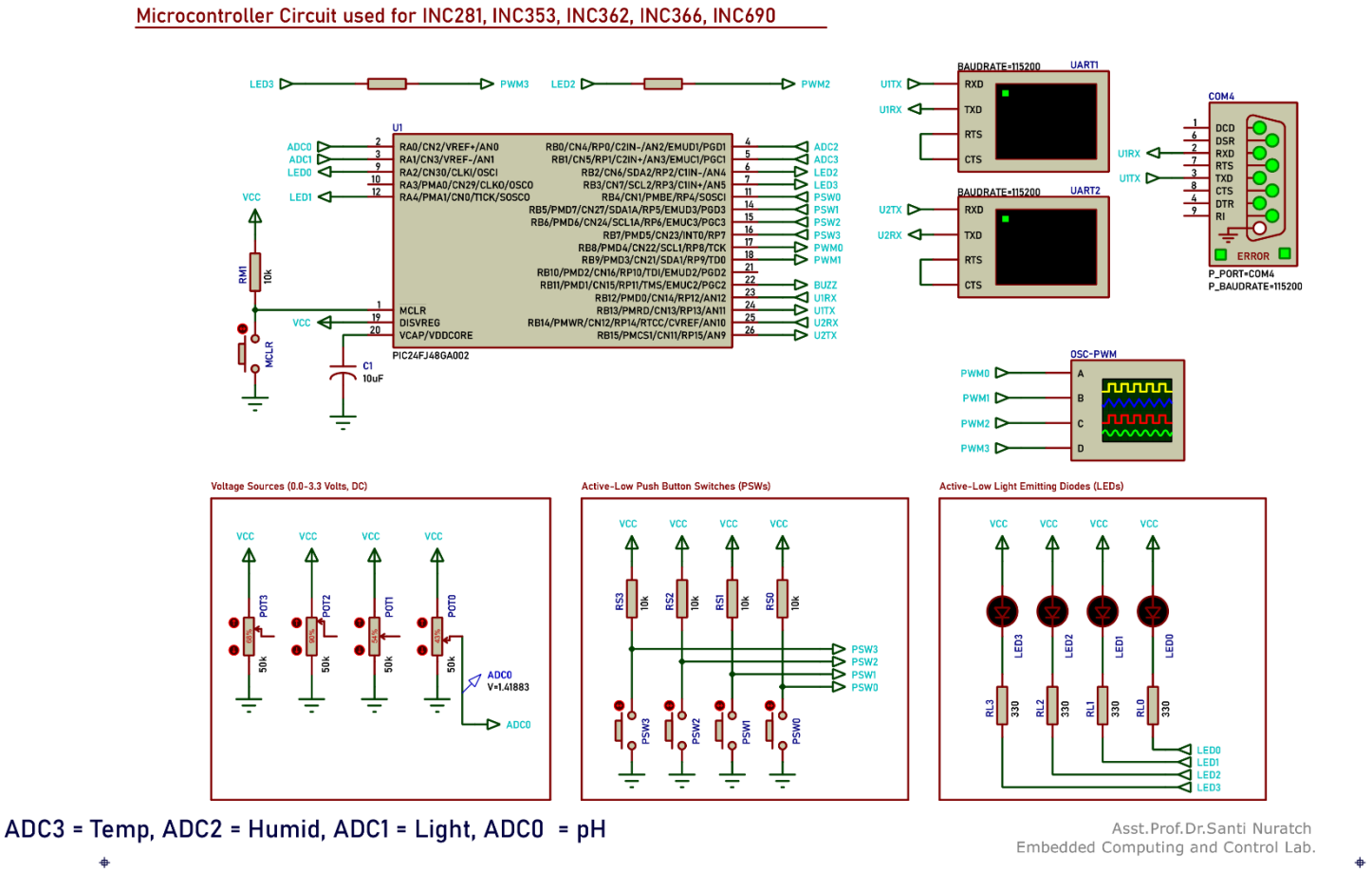


Figure 2. Microcontroller Circuit

The monitoring system described in this paper features a comprehensive set of commonly used sensors for temperature, light intensity, humidity, pH, water level detection, pest detection, air pump status and water pump status to evaluate the conditions of the environment inside the greenhouse. The system is modular and built around a microcontroller circuit provided by our professor (Figure 2). Note that the Node MCU of this circuit requires the input voltage range to be 0-3.33V. Sensor data is converted and sent to a remote web-user interface, which serves as an access point for current data and time series to be displayed on a webpage that is accessible from anywhere, allowing the farm to be monitored without users needing to be on-site. If necessary, feedback control over sensory data can be easily accomplished. Figure 3 below, shows a Block Diagram of the system overall.

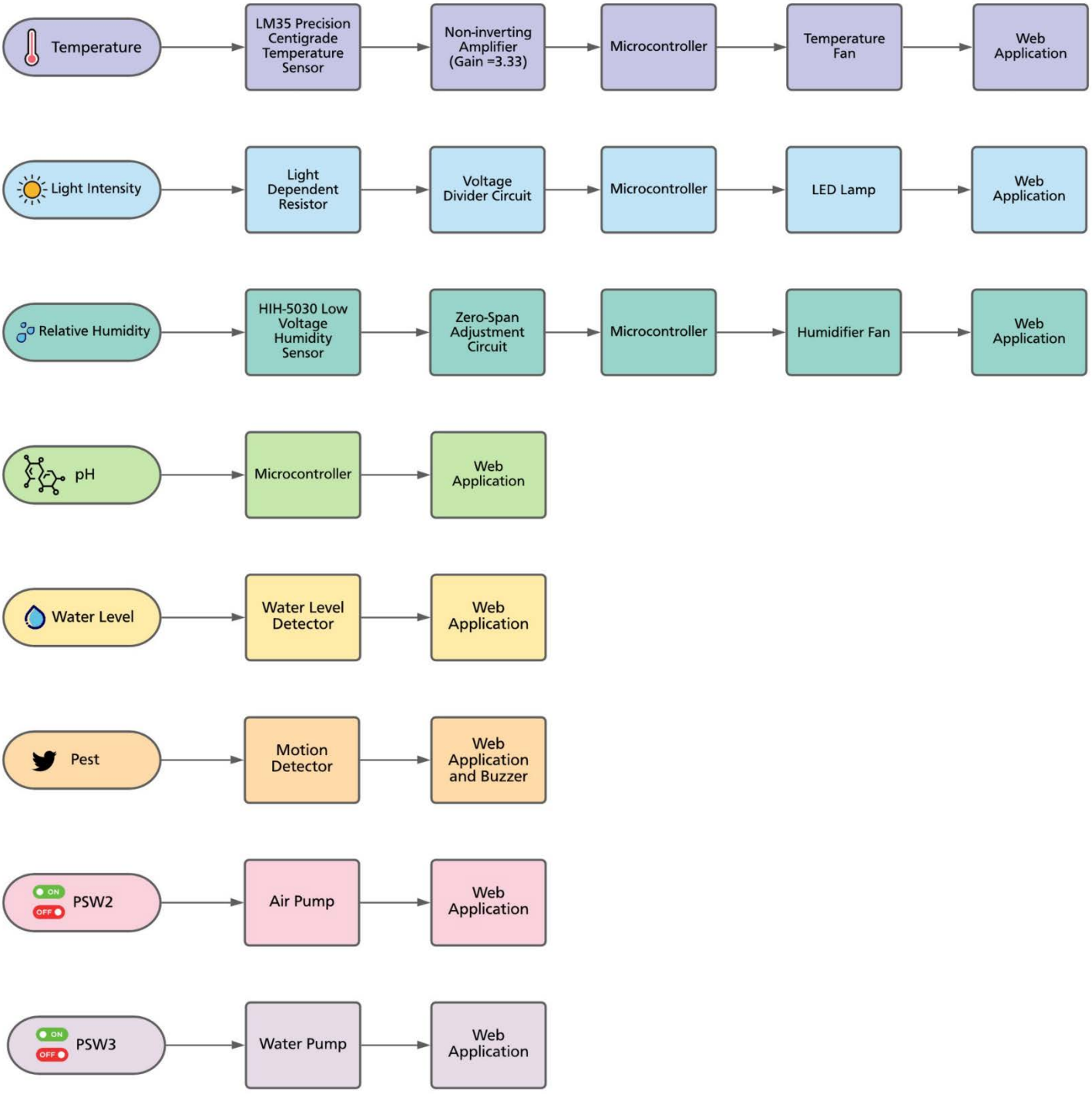


Figure 3. System Block Diagram

General Circuit Designing

Before introducing each of the sensors, we would like to mention the equations for designing an operational amplifier circuit in a general form as it will be used in the following text to demonstrate calculations for sensor signals that need converting.

The general equation used is

$$Vout = m\,Ein + \frac{R_f}{R_{OS}}Vos \qquad (1)$$

To calculate for op-amp design circuit, we first need to take note of the input signal range ($Ein_{min} - Ein_{max}$) and the desired output signal range ($Vout_{min} - Vout_{max}$). Calculate the voltage gain m from these two ranges

$$m = \frac{Vout_{max} - Vout_{min}}{Ein_{max} - Ein_{min}} \qquad (2)$$

Next, we find Vos using the linear equation

$$Vout_{min} = m\,Ein_{min} + Vos \qquad (3)$$

We then check whether or not our obtained Vos is gives an acceptable value that is near $Vout_{max}$ by using the linear equation

$$Vout_{max} = m\,Ein_{max} + Vos \qquad (4)$$

If Vos is not equal to 0V, we can select resistor values (R_1 and R_2) from the voltage divider equation

$$Vos = \frac{R_2}{R_1 + R_2} * V_{DC} \qquad (5)$$

However, if Vos is equal to 0, the step above is not needed and Vos will connect to ground. In addition, we will be instead, designing a non-inverting op-amp circuit. When picking resistor values (R_f and R_i) from the previously calculated voltage gain m in a non-inverting op-amp circuit

$$m = 1 + \frac{R_f}{R_i} \qquad (6)$$

If Vos is a nonzero value, the next step from equation (5) would be to select resistor values (R_f and R_i) from the previously calculated voltage gain m using the equation

$$m = \frac{R_f}{R_i} \qquad (7)$$

For Vos not equal to 0V, we make R_{OS} equal to R_f from the general equation (1) so we can easily calculate R_{IOS}

$$R_{IOS} = R_f // R_{OS} // R_i \qquad (8)$$

If Vos is equal to 0V, we do not have to calculate for R_{OS} and R_{IOS}.

Lastly, we substitute the obtained values into the circuit in either Figure 4 (for non-inverting op-amp circuit) or Figure 5 (for nonzero Vos).

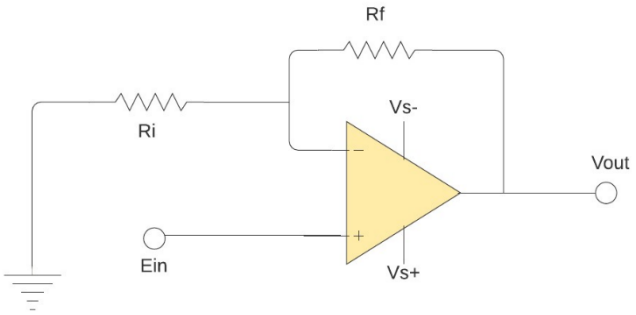


Figure 4. Non-Inverting Op-Amp Circuit Design

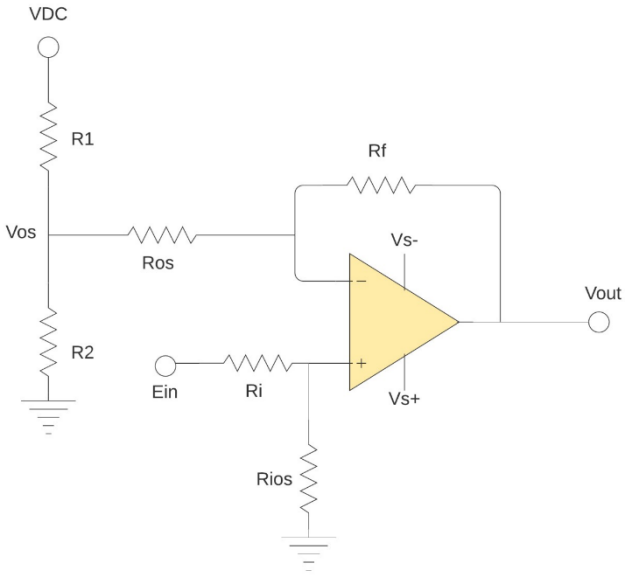


Figure 5. Circuit Design

If the values above were calculated correctly, we would get the desired output signal when feeding our input signal into the designed op-amp circuit. Note: We may have to check values in the simulation and adjust according to trial and error still.

Converting the Range 0-3.33V to 0-1023

The microcontroller contains an analog-to-digital converter (ADC) circuit that reads the changing voltage and converts it to a number between 0 and 1023. ADCs can vary greatly between microcontrollers. In our case, the ADC is 10-bit as it can detect 1,024 (2^10) discrete analog levels (0-1023). At the minimum, the pin is able to receive 0 V and thus, the analog input value is 0. Whereas at the maximum , the pin receives 3.33 V, the value is 1023. Between those two values, analogRead() returns a number between 0 and 1023 that is proportional to the voltage amount supplied to the microcontroller pin. To convert the range 0-3.33V to 0-1023, we use the equation

$$SensorValue = Voltage * \frac{1023}{3.33} \qquad (9)$$

The converted value will be used to compare to the threshold range in the program code which, in turn, will determine the PWM duty cycle of the corresponding actuator connected to microcontroller (whichever it may be).

Section II: Sensors

The sensors adopted into the system have been selected according to two main criteria: collect useful information about the conditions of the hydroponic farm within the greenhouse and ensure those conditions are within optimal range for plant growth. Such sensors can be broadly classified into two categories: analog sensors and digital sensors.

Analog Sensors

Temperature Sensor

Part of optimal plant growth is temperature. Hence, our system is established to monitor and control the temperature of the environment the plants are in. If sensors detect temperatures exceeding the limit, cooling fans will automatically turn on to decrease the ambient temperature. Since our farm is in a closed greenhouse, the growing environment is stabilized as ambient temperatures are buffered and the plants are protected from extreme weather. In addition, we have decided to locate the farm in a tropical country to save the costs of buying a heating system, so temperatures will never go below the optimum 25 degrees Celsius. The functional flow of the temperature sensor is shown in Figure 6 below.

Figure 6. Functional Flow Block Diagram for Temperature Sensor

When simulated in Proteus, the LM35 Precision Centigrade Temperature Sensor gives an output voltage range of 0-1V for temperatures between 0 to 100 ºC. For the sensor to be able to input into Node MCU in Figure 2, the voltage range must be 0-3.33V. Thus, we designed an operational amplifier circuit to convert the voltage signal range from 0-1V to 0-3.33V. The calculations are as follows:

From (2),

$$m = \frac{3.33 - 0}{1 - 0} = 3.33$$

From (3),

$$0 = 3.33 * 0 + Vos$$

$$Vos = 0$$

From (4), values are acceptable.

$$3.33 = 3.33 * 1 + 0 = 3.33$$

(5) is not needed as Vos is equal to 0V, we will be designing a non-inverting op-amp circuit.

Therefore, we use equation (6). From (6), 2.2k Ω was selected for R_f and 1k Ω for R_i to make m as close as possible to 3.33 while, at the same time, using standard resistor values.

$$3.3 = 1 + \frac{2.2k}{1k} = 3.2$$

As a result, we can use Figure 4 and get the following circuit design for the temperature sensor.

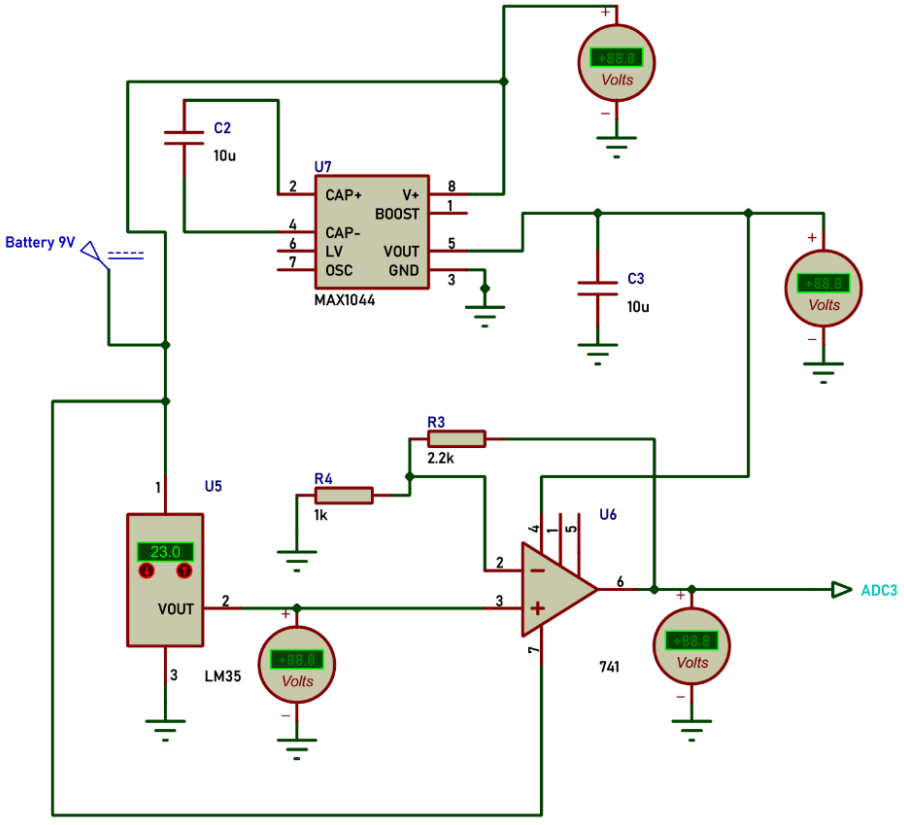


Figure 7. Circuit Design Temperature Sensor

In general, we use a 9V power supply for all the components unless a sensor is specified to use a different value by its datasheet. MAX1044, is used to invert the voltage in our circuit so we can use -9V for the op-amp supply and voltage divider (if needed) as well and it is connected according to the datasheet (Figure 8).

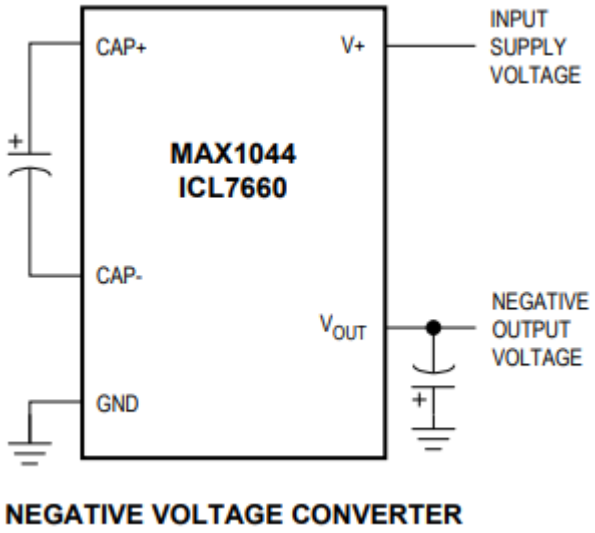


Figure 8. MAX1044 Circuit from Datasheet

Going back to the flowchart in Figure 6, once the voltage is converted by the op-amp signal, it inputs into the microcontroller pin ADC3 (Figure 7 and Figure 2). The voltage is converted to an analog value and is compared with the threshold range of the program. This determines the Temperature Fan's PWM duty cycle which is our corresponding input actuator to the sensor.

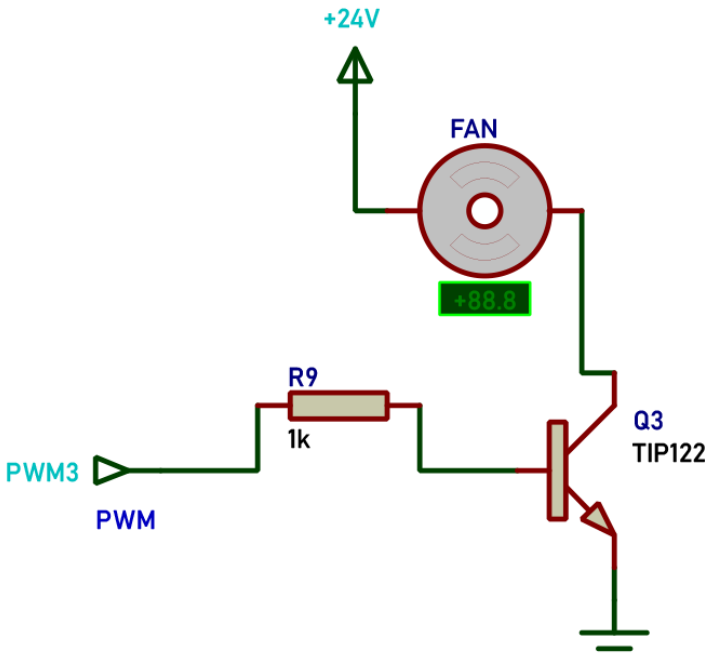


Figure 9. Temperature/Cooling Fan Circuit

Light Intensity Sensor

In the greenhouse factory, it is possible to provide the plants with consistent and constant amount of light for optimal growth. Our plants need about 10,000 Lux of light. If the LDR sensor detects the light to be below this value, the LED Lamp's brightness intensity automatically increases. The functional flow of the LDR sensor is shown in Figure 10 below.

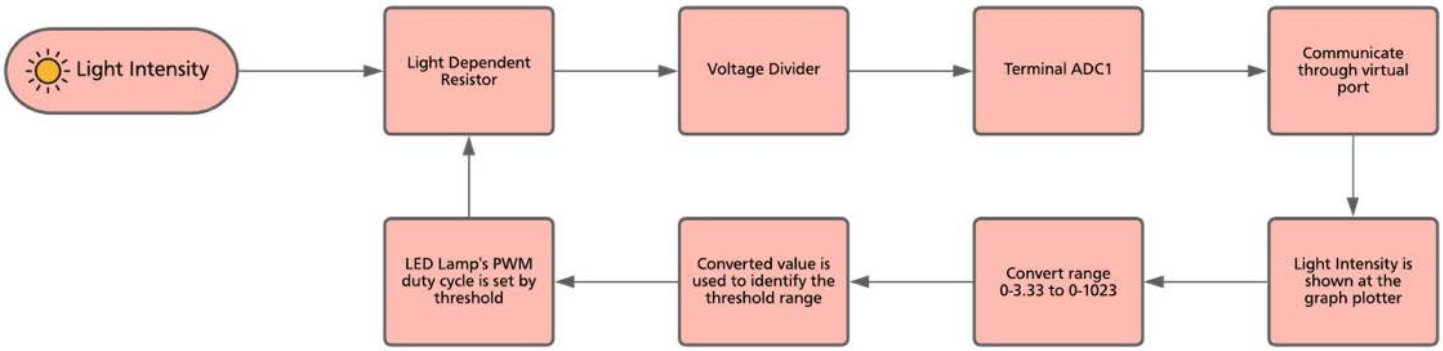


Figure 10 Functional Flow Block Diagram for Light Dependent Resistor

When simulated in Proteus, the LDR Sensor is able to measure the light intensity ranging from 0 to 10,000 lux, giving a voltage signal range of 0-8.60V. Although, we would have preferred the LDR to be able measure light intensity at higher levels (i.e. 30,000 lux), we worked with what proteus has provided us with. For the sensor to be able to input into Node MCU in Figure 2, the voltage range must be 0-3.33V. But since the voltage range from the sensor is larger than the range we want to input into the microcontroller, we cannot use an operational amplifier circuit to convert the voltage signal range as it does not reduce the voltage signal. Instead, we simply use voltage divider.

$$Output\ Voltage = \frac{R_2}{R_1 + R_2} * LDR\ Voltage \qquad (10)$$

From (10), we selected 16k Ω for R_1 and 10k Ω for R_2

$$3.33 = \frac{10k}{16k + 10k} * 8.60 = 3.31$$

The values for R_1 and R_2 gives us the circuit in Figure 11 below.

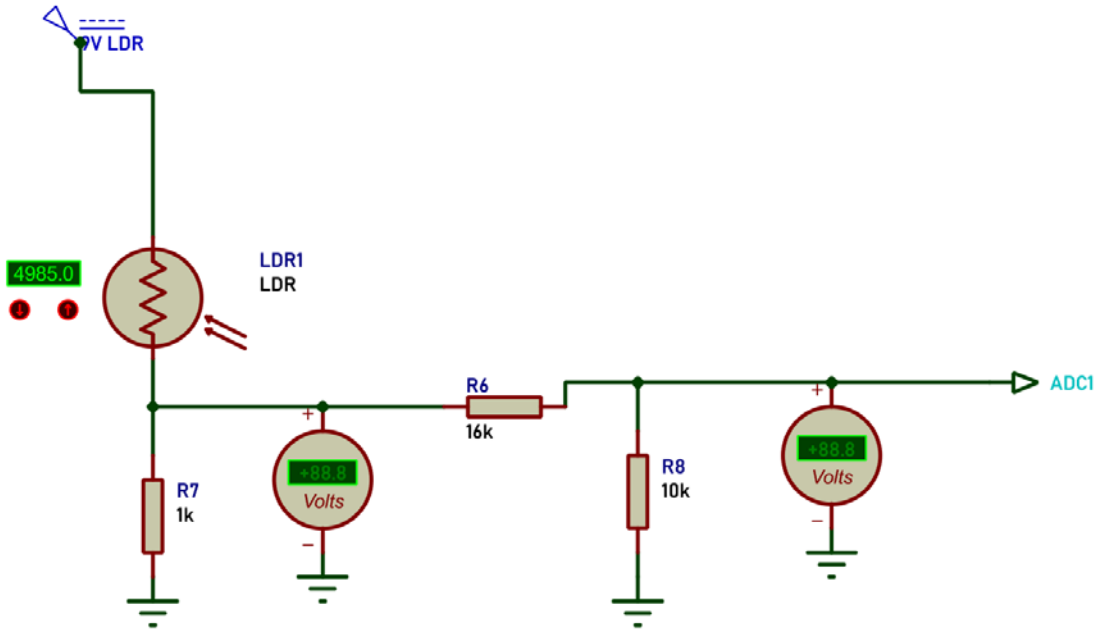


Figure 11. Circuit Design Light Dependent Resistor

Like the Temperature Sensor, the voltage supply used is 9V. Going back to the flow diagram (Figure 10), the converted voltage inputs into ADC1 (Figure 11 and Figure 2). The voltage is converted to an analog value and is compared with the threshold range of the program. This determines the LED Lamp's PWM duty cycle which is our corresponding input actuator to the sensor.

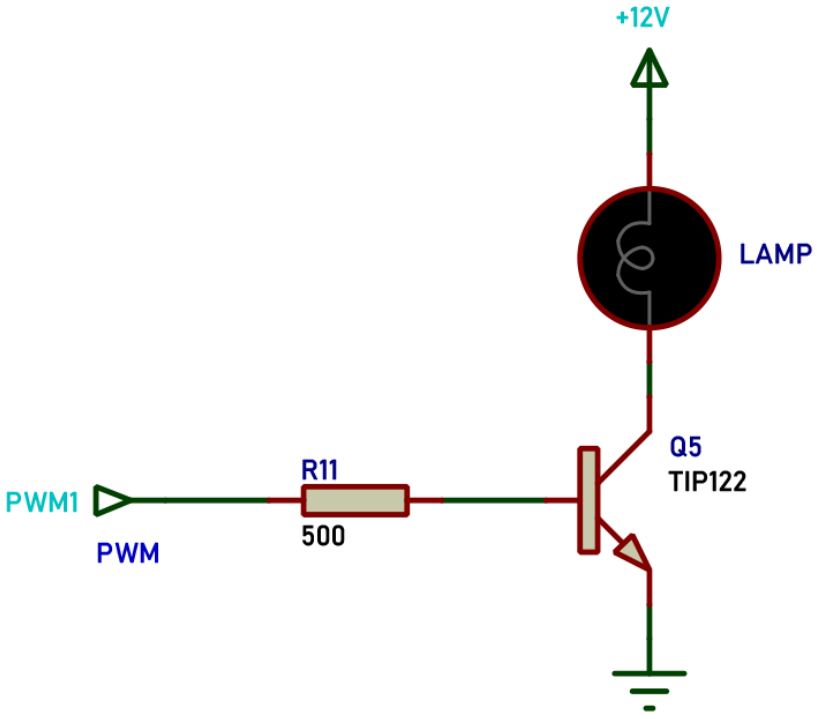


Figure 12. LED/Grow Lamp Circuit

Relative Humidity Sensor

Humidity is also essential to our plants' growth. Ideally, the room needs to be at 70% humidity for the marijuana plants to thrive. If the sensor detects humidity in the room to be too low, a humidity fan will turn on to raise the air moisture content by directly spraying water on our plants. If the humidity becomes too high, the fan turns off. Since our farm is in a closed environment, controlling the relative humidity is rather easy. The functional flow of the relative humidity sensor is shown in Figure 13 below.

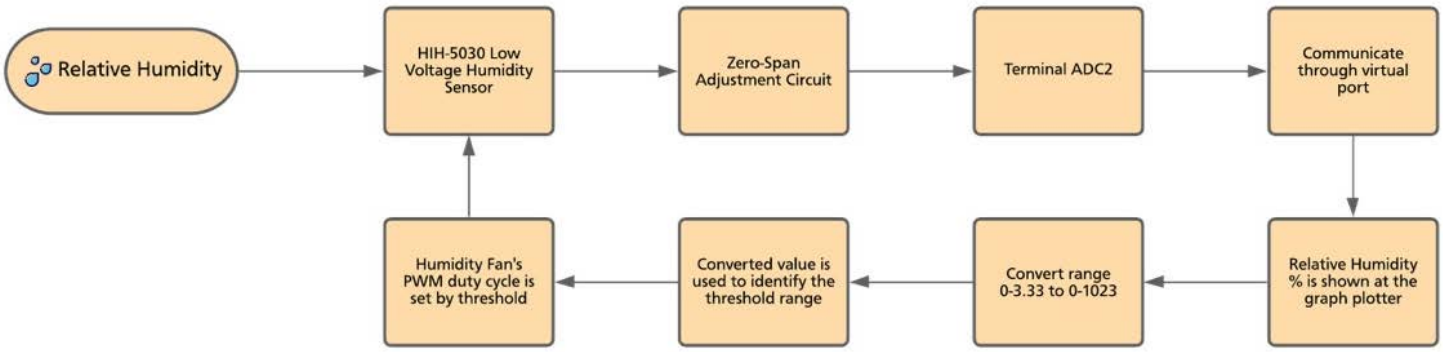


Figure 13. Functional Flow Block Diagram for Relative Humidity Sensor

When simulated in Proteus, the HIH-5030 Low Voltage Humidity Sensor gives an output voltage range of 0.5-2.58V for the relative humidity range between 0 to 100 %. For the sensor to be able to input into Node MCU in Figure 2, the voltage range must be 0-3.33V. Thus, we designed an operational amplifier circuit to convert the voltage signal range from 0.5-2.58V to 0-3.33V. The calculations are as follows:

From (2),

$$m = \frac{3.33 - 0}{2.58 - 0.5} = 1.6$$

From (3),

$$0 = 1.6 * 0.5 + Vos$$

$$Vos = -0.8$$

From (4), values are acceptably close.

$$3.33 = 1.6 * 2.58 - 0.8 = 3.328$$

Because Vos is a nonzero voltage, we can use equation (5) to help select values for R_1 and R_2. We selected 91k Ω for R_1 and 9.1k Ω for R_2. Note that all resistor values selected are standard resistor values.

$$0.80 = \frac{9.1k}{91k + 9.1k} * 9 = 0.81$$

V_{DC} is from the supply 9V.

The next step is to select values for R_f and R_i from the previously calculated voltage gain m using equation (7). We selected 160k Ω for R_f and 100k Ω for R_i.

$$1.6 = \frac{160k}{100k} = 1.6$$

We let R_{OS} equal to R_f from the general equation (1) so we can easily calculate R_{IOS} from equation (8).

$$R_{IOS} = 160k//160k//100k = 80k//100k = \ 44.4k$$

We picked 47k Ω for R_{IOS}.

With this, we designed a circuit that will provide us with the voltage range we need for our output from Figure 5. However, the circuit is not non-inverting and thus our voltage output right now is the inverted version -3.33-0V of what we want 0-3.33V. To fix this, we add a simple inverting amplifier to the circuit design after our op-amp circuit design, making sure that the gain of this amplifier circuit is 1 so the signal won't be amplified again, only inverted.

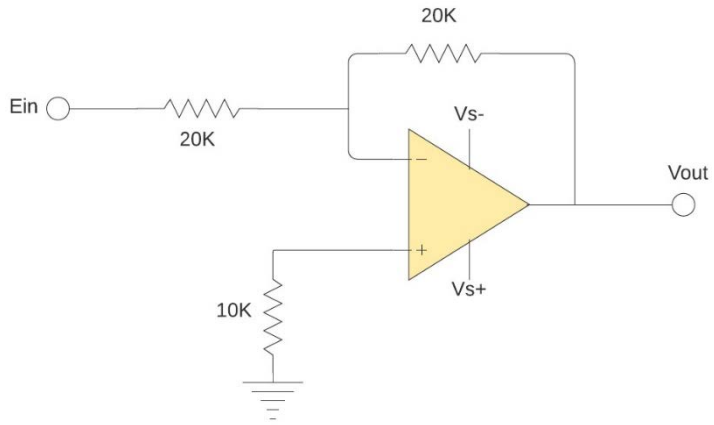


Figure 14. Inverting Op-Amp Circuit (Gain = 1)

Putting everything together, we are able to design the circuit to achieve the desired voltage output.

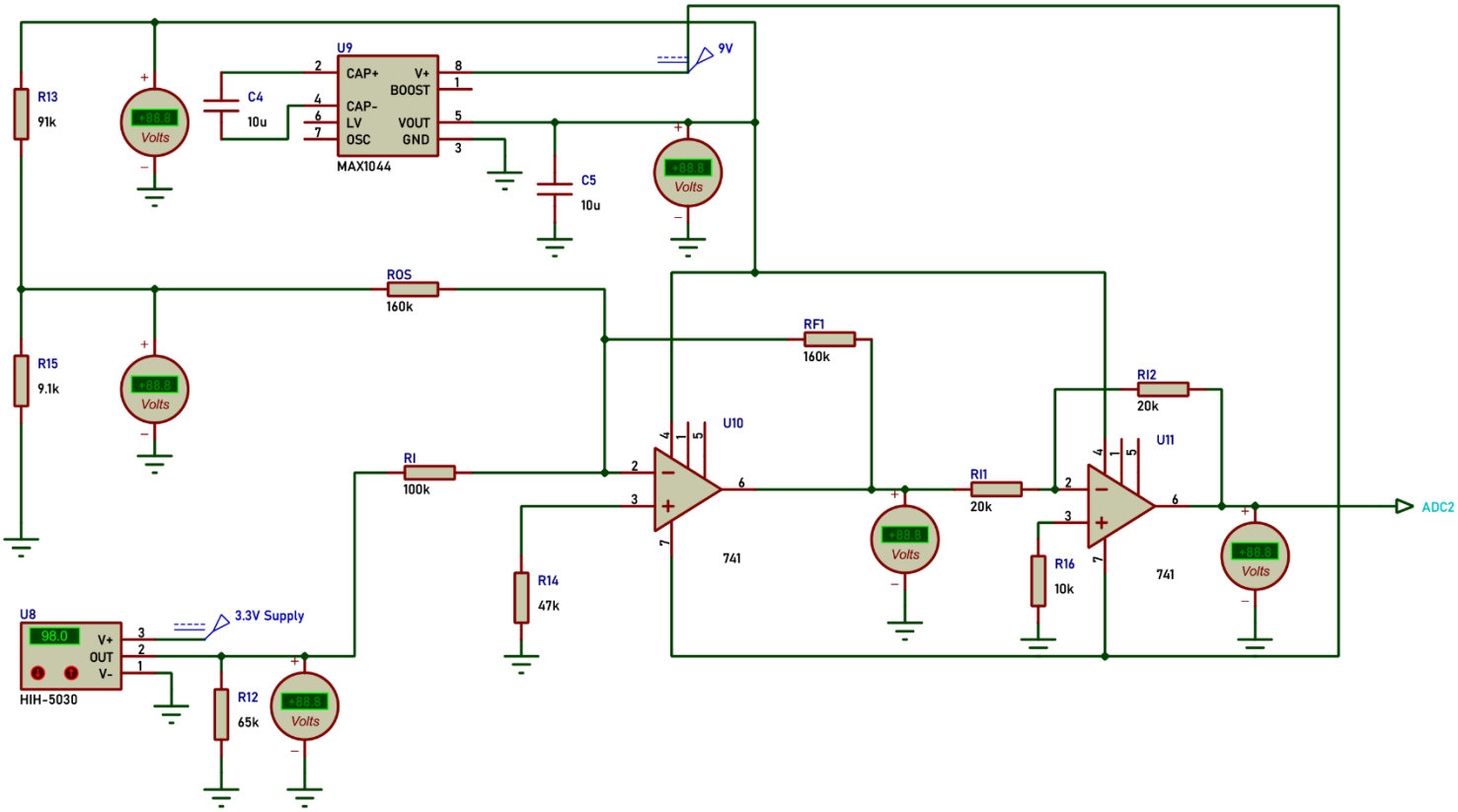


Figure 15. Circuit Design Relative Humidity Sensor

In Figure 15, 9V is mainly used as the power supply for the circuit, but a 3.3V power supply is needed for the HIH-5030 sensor specified by the datasheet (check Figure 16). MAX1044, connected like in Figure 8, is used to invert the voltage in our circuit so we can use -9V for offset voltage and the op-amp power supply.

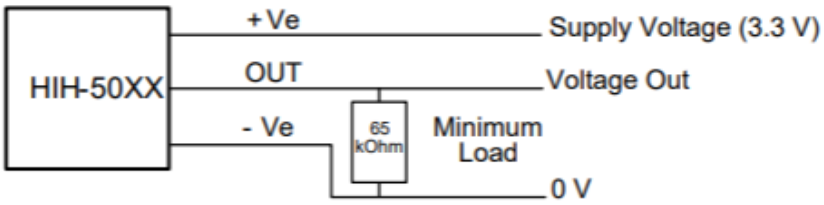


Figure 16. HIH-5030 Circuit from Datasheet

Going back to the flow diagram in Figure 13, once the voltage is converted by the op-amp signal, it inputs into the microcontroller pin ADC2 (Figure 15 and Figure 2). The voltage is converted to an analog value and is compared with the threshold range of the program. This determines the Humidifier Fan's PWM duty cycle which is our corresponding input actuator to the sensor.

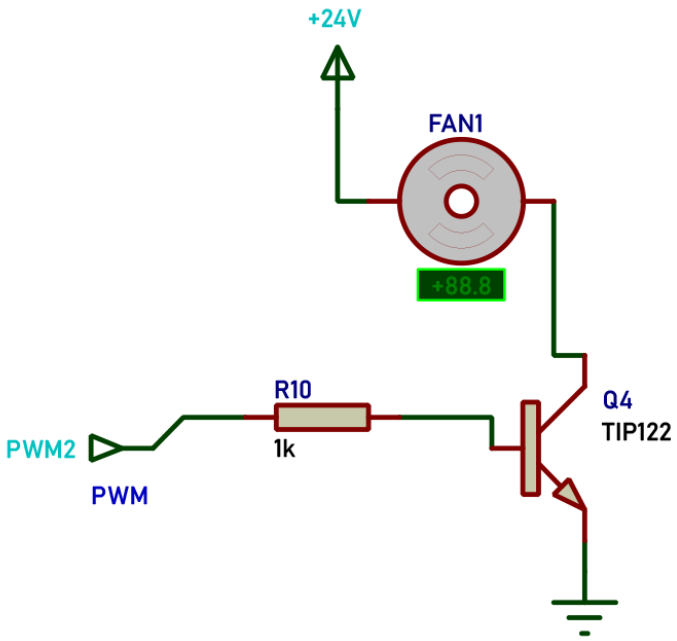


Figure 17. Humidifier Fan Circuit

pH Sensor

Because pH influences the nutritional availability of growing plants, maintaining it to be at a certain level is important. A pH level that is too high or alkaline might prevent nutrients from being absorbed, resulting in a deficiency in our plants. Marijuana pH levels in hydroponics should be 5.8–6.0 optimally. The pH of the nutrient solution must be constantly monitored and maintained, as pH can change considerably more quickly than in soil gardens, causing problems if not correctly maintained. Therefore, our system can monitor the pH of the water passing through the plants in real-time in which users can check via the dashboard. If an anomaly is detected, they may have to change their water supply. The functional flow of the pH sensor is shown in Figure 18 below.

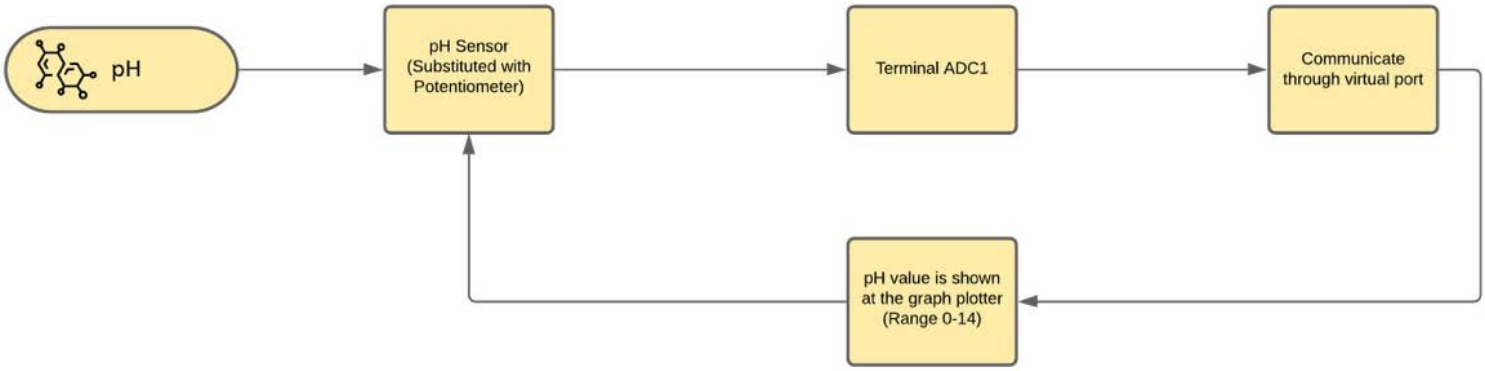


Figure 18. Functional Flow Block Diagram for pH Sensor

Although by default the pH sensor is not available in the Proteus software, we figured it was too significant to leave out in a hydroponic farming. Therefore, to emulate how the pH sensor would work, we used a potentiometer to change the values of the voltage input entering the microcontroller pin ADC1 (Figure 2). This is enough to show the pH ranging from 0 to 14 in the graph on the dashboard.

Digital Sensors

Water Level Detector

A water level detector is needed to prevent the overflow of the water reservoir. The LED will turn on if the water level reaches the "danger" height in the tank (80 percent of the water tank's height), telling the user the water supply needs to be reduced. Although this scenario seems unlikely to happen, precaution should be taken to prevent losses of our production. The functional flow of the water level detector is shown in Figure 19 below.

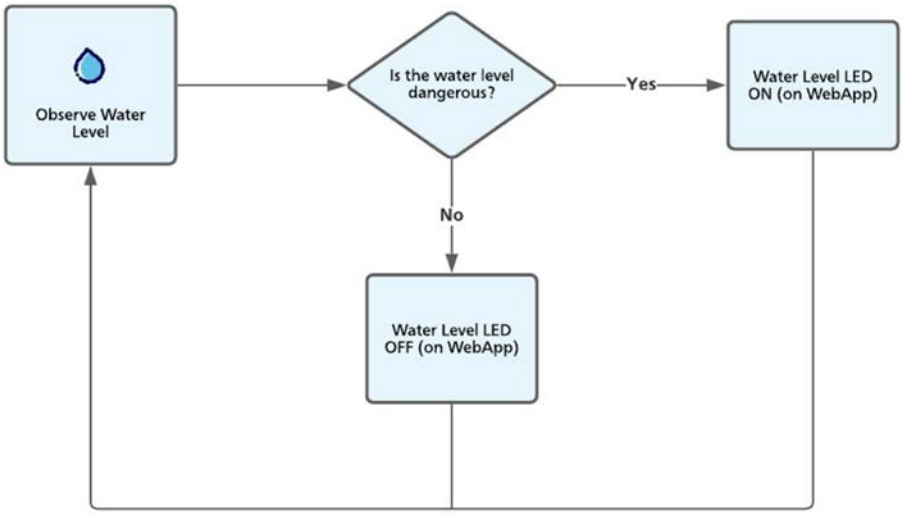


Figure 19. Functional Flow Block Diagram for Water Level Detector

Like the pH sensor, the Water Level Detector is not available in the Proteus software by default. Fortunately, to emulate how the output will be displayed on the dashboard, we only have to “feed” the input manually. This is enough to show on the dashboard whether or not the water is at a dangerous level in the tank.

Motion Detector

In cases where insects and birds do end up in the greenhouse, this could be detrimental to the health and quality of our plants. Thus, a motion sensor is needed to detect these creatures, the LED will turn on in the dashboard when it senses movement, and the buzzer will sound. However, if a person needs to enter factory for inspection, the sensor will be turned off beforehand preventing the LED and buzzer from unnecessarily turning on. This motion sensor can also prevent theft and burglary on occasions where an unauthorized person enters the factory since the sensor will not be turned off. The functional flow of the motion detector is shown in Figure 20 below.

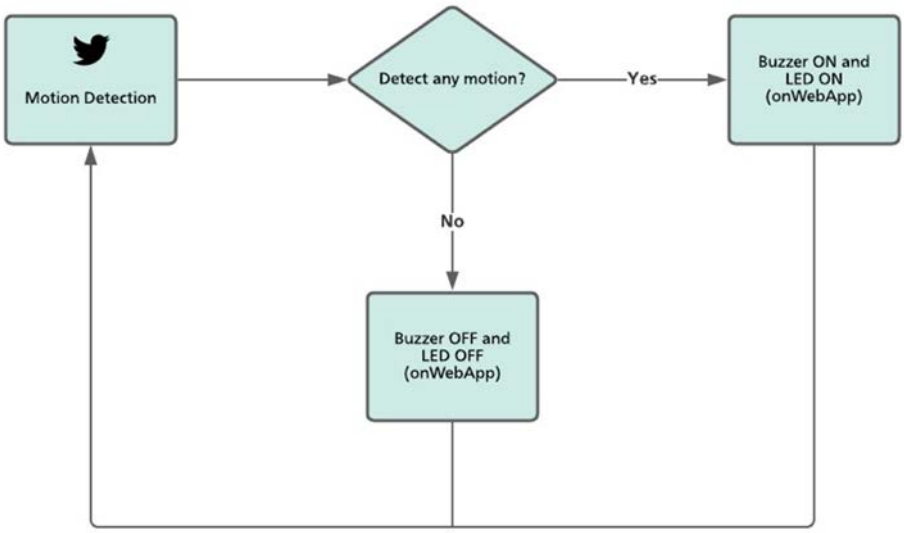


Figure 20. Functional Flow Block Diagram for Motion Detector

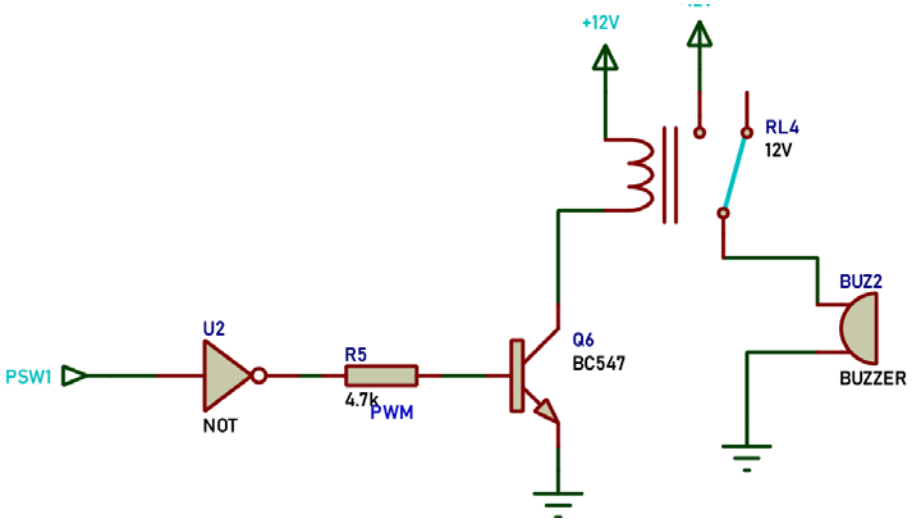


Figure 21. Buzzer Circuit

The Air Pump

Like humans, plants need oxygen to respire as well. When a plant's root system is submerged in the nutrient solution in hydroponics, the air pump is necessary to deliver the oxygen to the air diffuser in the reservoir, which subsequently breaks the air stream into tiny bubbles that are easier for the roots to absorb. The air pump's status (on or off) is also displayed on the dashboard. If the air pump is on, the LED will be on and vice versa. The functional flow of the air pump is shown in Figure 22 below.

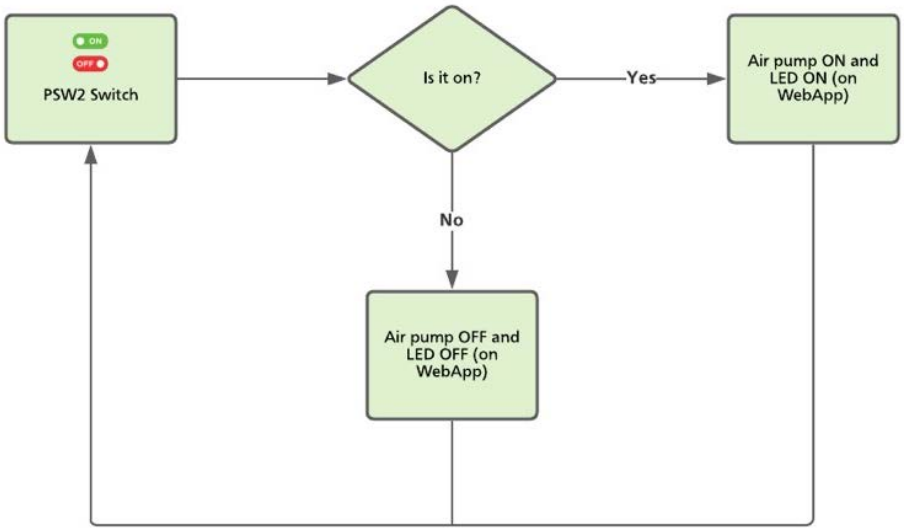


Figure 22. Functional Flow Block Diagram for Air Pump

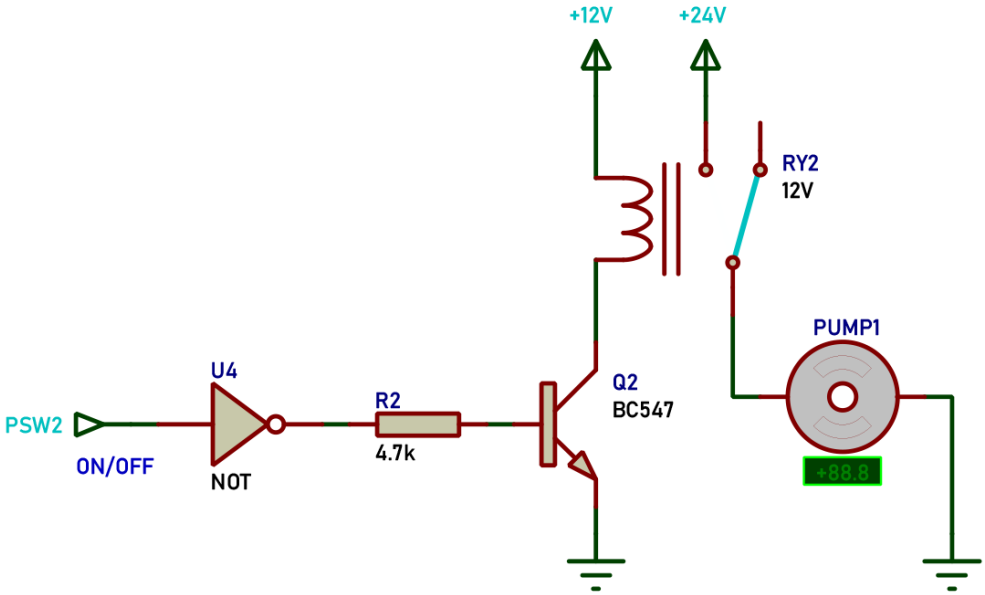


Figure 23. Air Pump Circuit

The Water Pump

The water pump is essentially for transporting the water by increasing their pressure from the reservoir to the plants as shown in Figure 1. The user is able to view the water pump's status (on or off) via the dashboard. If the water pump is on, the LED will be on and vice versa. The functional flow of the water pump is shown in Figure 24 below.

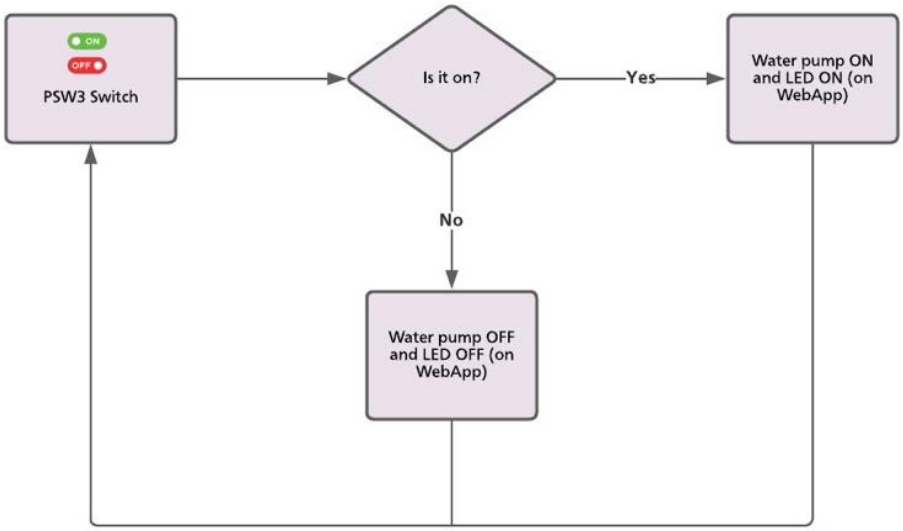


Figure 24. Functional Flow Block Diagram for Water Pump

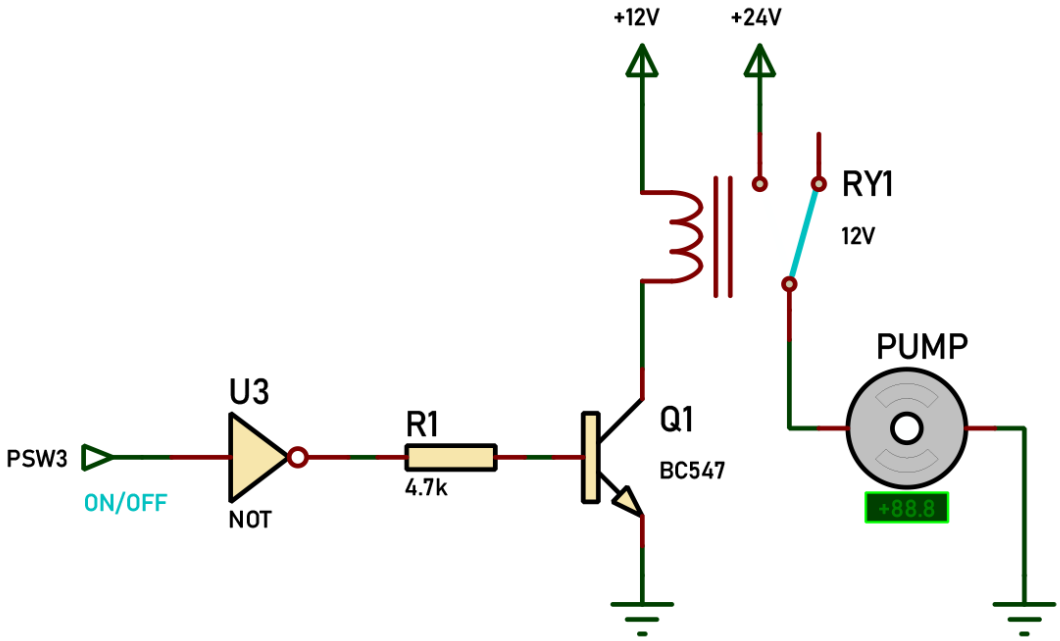


Figure 25. Water Pump Circuit

Section III: Web-based User Interface (Application)

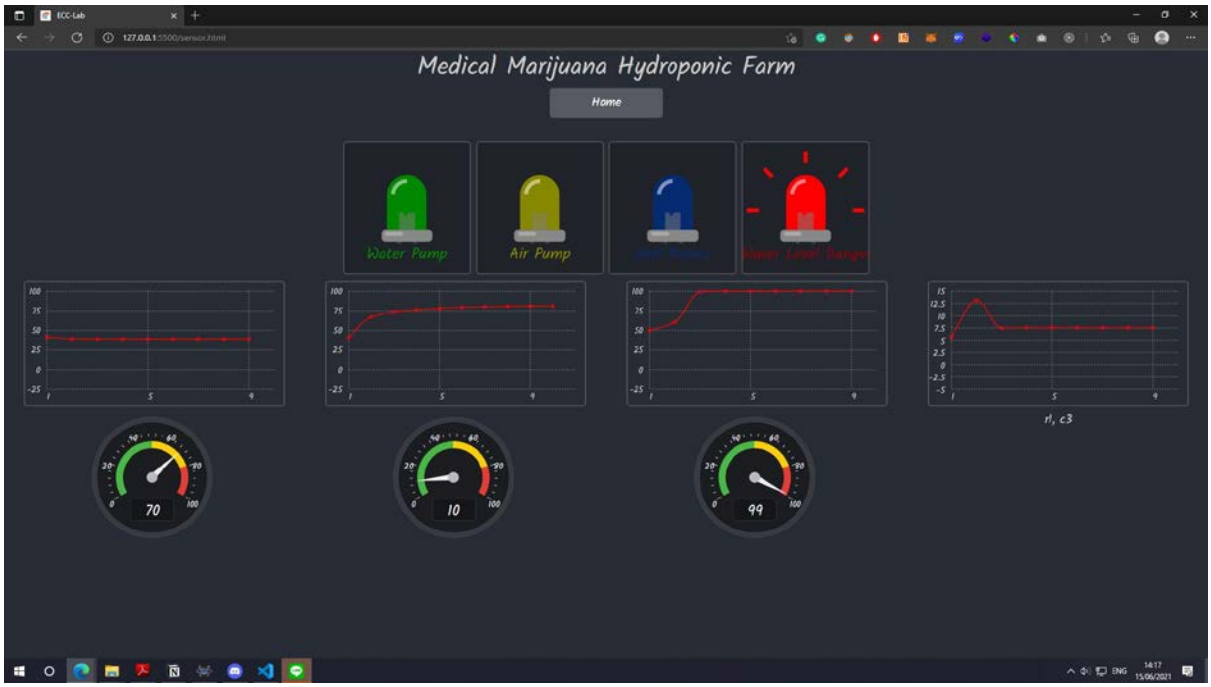


Figure 26. Website Dashboard

The dashboard (Figure 26) consists of the headline, the back-to-the-home button, and three components that indicate the value from the farm, including LEDs, graph plotters, and PWM gauges.

First, there are four LEDs in the web application, which tell the current state of, water pump, air pump, buzzer alarm, and water level danger, respectively. To clarify, if the PSW of the water pump or air pump is on, the pump is turned on and the LED will turn on as well to indicate that the pump is working right now. Next, if motion is detected, we can imply that it might be pests around the farm (e.g., birds, rats), the alarm buzzer will play the loud sound to chase the pests away, as well as turn on the LED on the web application to tell the user that the alarm buzzer is working. Furthermore, if the water level in the water reservoir is at the specified danger level (set by the farmers), the red LED will be turned on, and the user might come and release and adjust the water manually.

Second, the web application page contains four graph plotters for plotting temperature (in Celsius), relative humidity percent, light intensity (lux), and pH level, respectively. The first graph receives the temperature value (in the voltage range of 0-3.33V form) from ADC3 in Proteus (say the temperature sensor circuit), and it will plot the current temperature in the farm in Celsius unit. The second graph receives the relative humidity percent value (in the

voltage range of 0-3.33V form) from ADC2 in Proteus (say the humidity sensor circuit), and it will plot the current relative humidity percentage in the farm in percent unit. The third graph receives the light intensity (in the voltage range of 0-3.33V form) value from ADC1 in Proteus (say the light intensity circuit), and it will show the lux value. This one is a special case since LDR has a working range of only 0-10,000 lux, but the light intensity level that our plant needed is exceeded 15,000 lux. Thus, the graph at the daytime will be at maximum. The last graph receives the pH value (in the voltage range of 0-3.33V form) from ADC0 in Proteus, but since the pH sensor is inaccessible in Proteus. Hence, we substituted the sensor with the potentiometer. This graph will plot the pH value from 0-14 because it is the normal range of pH values.

Third, there are three PWM gauges in the web application, which indicate the duty cycle of the Temperature fan, the Humidifier fan, and the grow lamp, respectively. Each actuator will receive the duty ratio from the JavaScript sensor code. To elaborate, the “ecc-uart-websockets-cli” will receive the ADC values from the Proteus via virtual port communication, these values of each sensor will fall to the specified threshold for each actuator. For example, if the temperature is more than 25 Celsius, set the duty ratio of the temperature fan to be 0.33. This is the value that will be shown that the gauge as well. It tells that how “powerful” the PWM actuators will be, the higher the PWM value, the brighter the grow lamp will be, or the faster the temperature lamp will reach the specific speed.

The Home Page of the Website, in Figure 27, contains 5 hyperlinks in the navigation bar, which are About us, Types, Reviews, Contact Us, and Dashboard, respectively. The first four buttons in the navigation bar will take the user to the specific part of the same page. But the last button, Dashboard, will take the user to the web application part of the website (say Figure 26).

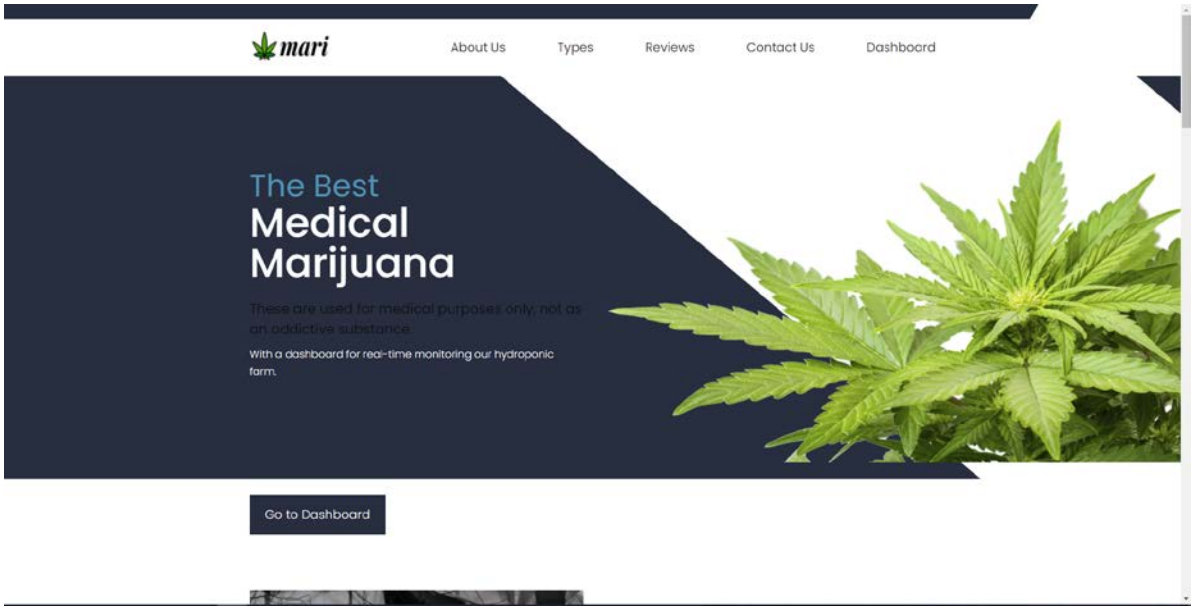


Figure 27. The Home Page of the Website

In the About us section, there will be a “Read more” button. If the user clicks the button, it will take the user to the information page of the farm, which shows in Figure 28 below

Figure 28. Farm Information Page

On this page, there will be information about the farm, i.e., temperature function, humidity function, light function, pH function, water pump control, air pump control, motion detection, water level detector, and a short summary of the functions. Which is very helpful to help the user understands the indicators in the web application of the farm.

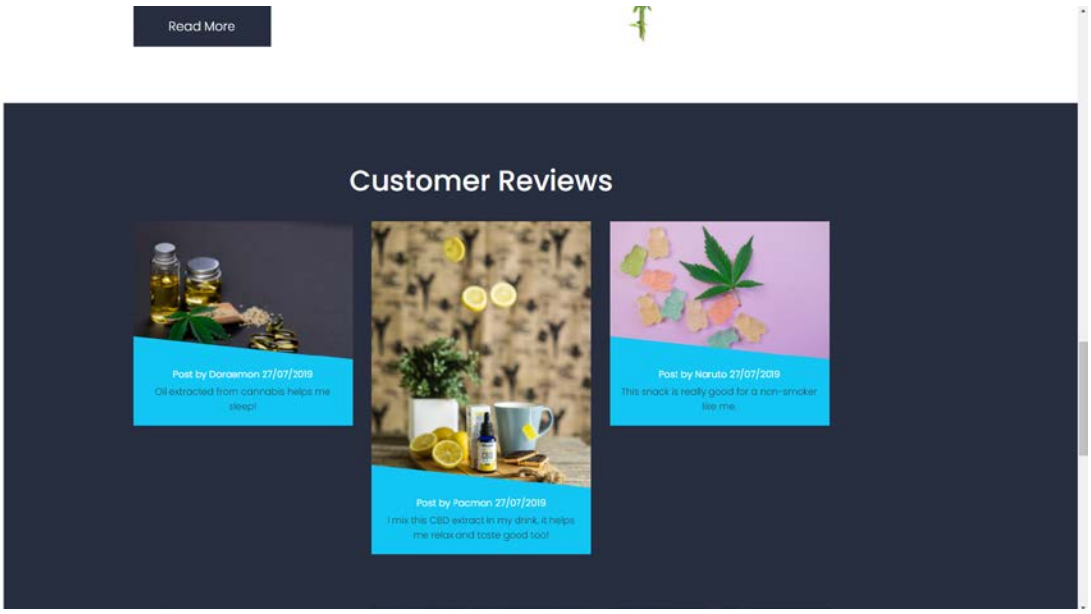


Figure 29. Customer Reviews Section

There is also the customer reviews part of the website, which show the real and sincere comments or writings from the user. For example, “Post by Doraemon 27/07/2019 Oil extracted from cannabis helps me sleep!”. This help builds the customers trust and loyalty from reading the reviews.

Section IV: Conclusion

This paper describes the developed monitoring system for a hydroponic medicinal marijuana farm, including the farm itself, that can be used to remotely monitor the system and help quantitatively understand the effects each parameter of the environment contributes plant growth. Because the suggested platform is made up of common inexpensive sensors and modular components, it is both economical and simple to set up and operate. As a result, the system may simply be recreated, and additional sensors can be added as needed. The data collected by the sensors is displayed in a user-friendly manner on a webpage and the online website may be accessed from anywhere, allowing users to stay up to date on the farm's conditions.

Once enough data has been collected, the system can be utilized to collect sensor data to do a quantitative assessment of the farm useful for high-level decision making. Furthermore, the suggested monitoring platform serves as a starting point for making the farm completely autonomous by using sensor data as input signals to control fans, valves, lights, motors, and other devices.

Most importantly, users could fundamentally implement this versatile smart farm to cultivate other hydroponic plants if they wish to do so. Medical marijuana was used in the farm only to show how useful this method would be in an actual scenario with highly demanded plants. To monitor other plants in the farm, users would only have to modify the thresholds in the code of the sensors according to the plant species' optimal conditions. Minimal human interaction is needed as the farm will automate itself, marginally reducing costs and time wasted. This system could unlock the full potential of hydroponics, making it a viable (maybe even better) alternative to traditional agriculture in the face of future climate-related issues.

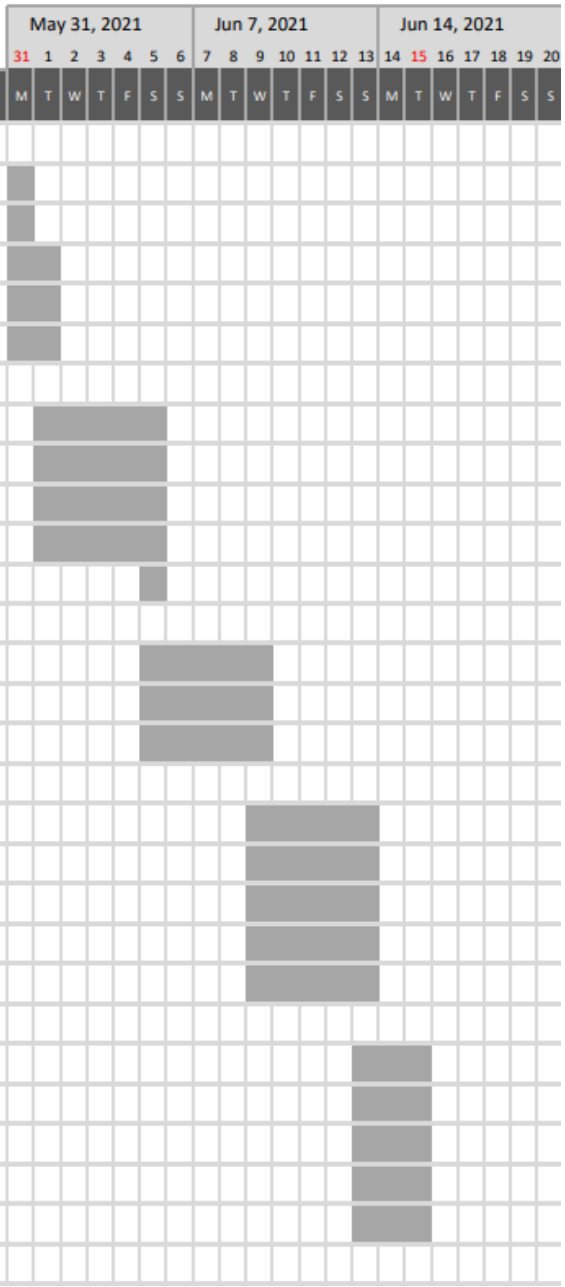
Gantt chart

Medical Marijuana Hydroponic Farm

Pattarapon 4012 = PB
Suchada 4016 = S
Pawornrat 4030 = PP

Project Start: Mon, 5/31/2021
Today: -
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Project Planning				
Project Topic Decision	Everyone	100%	5/31/21	5/31/21
Components Choosing	Everyone	100%	5/31/21	5/31/21
Op-Amp Circuit Calculations	PP	100%	5/31/21	6/1/21
Information/Documentation Gathering	S	100%	5/31/21	6/1/21
Circuit Drafting	PB	100%	5/31/21	6/1/21
Proteus Circuit Building				
Digital Output Sensors	PB	100%	6/1/21	6/5/21
Analog Output Sensors (with Op-Amp)	PP	100%	6/1/21	6/5/21
Digital Input Actuators	S	100%	6/1/21	6/5/21
PWM Input Actuators	PB, S	100%	6/1/21	6/5/21
Connect Everything Together	Everyone	100%	6/5/21	6/5/21
Web Application Coding				
HTML Page	PP, S	100%	6/5/21	6/9/21
CSS Page	PP, S	100%	6/5/21	6/9/21
JavaScript Page	PB	100%	6/5/21	6/9/21
Report Preparation				
Introduction Part	PP, S	100%	6/9/21	6/13/21
System Overview Part	S	100%	6/9/21	6/13/21
Electrical System Part	PP	100%	6/9/21	6/13/21
Coding Part	PB	100%	6/9/21	6/13/21
Web-based User Interface Part	PB, S	100%	6/9/21	6/13/21
Presentation Video Production				
Introduction Part	PP	100%	6/13/21	6/15/21
System Overview Part	S	100%	6/13/21	6/15/21
Electrical System Part	PP, S	100%	6/13/21	6/15/21
Coding Part	PB	100%	6/13/21	6/15/21
Web-based User Interface Part	PB	100%	6/13/21	6/15/21



Appendix

The JavaScript code below is the algorithm for communication with Proteus and display information on the web application.

```
const button_push = [];  // Globalize Push Button.
var temp;                // Globalize Temperature Variable.
var humid;               // Globalize Humidity Variable.
var light;               // Globalize Light Intensity Variable.
var temp_pwm;            // Globalize Temperature Fan PWM Variable.
var humid_pwm;           // Globalize Humidifier Fan PWM Variable.
var light_pwm;           // Globalize Grow Lamp PWM Variable.

// Grid Initiate
function newGrid(r, c){  // Declare Grid Function.
    grids = new Grids({  // Initiate a new grid.
        rows: r,         // Row value for the grid.
        cols: c          // Column value for the grid.
    });
    grids.hideBorders(); // Set grid settings to hide its border line.
}

// Grid Size
function gridChild(){                       // Function to create a new grid.
    newGrid(2, 4);                          // This grid has 2 rows, and 4 columns.
    grids.addChildTo(0, 0, plotTemp);       // Add Temperature Plotter to row: 0, column: 0.
    grids.addChildTo(0, 1, plotHumid);      // Add Humidity Plotter to row: 0, column: 1.
    grids.addChildTo(0, 2, plotLight);      // Add Light Intensity Plotter to row: 0, column: 2.
    grids.addChildTo(0, 3, plotPh);         // Add pH Plotter to row: 0, column: 3.
    grids.addChildTo(1, 0, gaugeTemp);      // Add Temperature Fan PWM Gauge to row: 1 , column: 0.
    grids.addChildTo(1, 1, gaugeHumid);     // Add Humidifier Fan PWM Gauge to row: 1, column: 1.
    grids.addChildTo(1, 2, gaugeLamp);      // Add Grow Lamp PWM Gauge to row: 1, column: 2.
```

```
}

function psw(){                                                                    // Create Switches function.
    const labels = ["Water Pump", "Air Pump", "Bird Detect", "Water Level Danger"];    // Label of the switches (LED).
    const colors = ["green", "yellow", "blue", "red"];                             // Color of the switches (LED).

    const leds = [];                  // Initially empty array used to store LED object.

    labels.map( (txt, idx) => {       // Create the LEDs by mapping the labels, colors, and uids.

        const led = new Led({         // Create a new LED.
            label: txt,               // Use the txt of the labels as its label.
            type: colors[idx],        // Use the color corresponding to txt (at idx position) to be its color.
            uid: 3-idx                // Set uid to be start with 3, and decreasing until 0.
        });

        leds.push(led);               // Send the created LED to store in leds array.
    });

    const linkLED = new WSLink({                          // Create a new link to Proteus.
        pswCallback: (uid, status, state, error) => {     // Initiate the callback from web application or Proteus.
            leds[3-uid].setStatus(status);                // Set the status of LED (to be either ON/OFF).
        }
    });
}

// Temperature Graph
function plot_temp(){           // Function to create graph for plotting temperature value.
    plotTemp = new Plotter({    // Create a new plotter.
        mix: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
```

```
        step: 4                 // The graph will step at the range of 4.
    });
    plotTemp.clear();           // Initially clear the graph (at each page refresh).
}

// Humidity Graph
function plot_humid(){          // Function to create graph for plotting relative humidity value.
    plotHumid = new Plotter({   // Create a new plotter.
        mix: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
        step: 4                 // The graph will step at the range of 4.
    });
    plotHumid.clear();          // Initially clear the graph (at each page refresh).
}

// Light Intensity Graph
function plot_light(){          // Function to create graph for plotting light intensity value.
    plotLight = new Plotter({   // Create a new plotter.
        mix: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
        step: 4                 // The graph will step at the range of 4.
    });
    plotLight.clear();          // Initially clear the graph (at each page refresh).
}

// pH in scientific soil Graph
function plot_ph(){             // Function to create graph for plotting light intensity value.
    plotPh = new Plotter({      // Create a new plotter.
        mix: 0,                 // Minimum scale of the graph is 0.
        max: 14,                // Maximum scale of the graph is 14.
        step: 4                 // The graph will step at the range of 4.
```

```
    });
    plotPh.clear();             // Initially clear the graph (at each page refresh).
}

function temp_fan(){            // Function to create the gauge for temperature fan.
    gaugeTemp = new Gauge({     // Create a new gauge.
        min: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
        ticksMajor: 20,         // The larger scale of the gauge has a range of 20.
        ticksMinor: 4           // The smaller scale of the gauge has a range of 4.
    });
}

function humid_fan(){           // Function to create the gauge for humidifier fan.
    gaugeHumid = new Gauge({    // Create a new gauge.
        min: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
        ticksMajor: 20,         // The larger scale of the gauge has a range of 20.
        ticksMinor: 4           // The smaller scale of the gauge has a range of 4.
    });
}

function grow_lamp(){           // Function to create the gauge for grow lamp.
    gaugeLamp = new Gauge({     // Create a new gauge.
        min: 0,                 // Minimum scale of the graph is 0.
        max: 100,               // Maximum scale of the graph is 100.
        ticksMajor: 20,         // The larger scale of the gauge has a range of 20.
        ticksMinor: 4           // The smaller scale of the gauge has a range of 4.
    });
}
```

```
function temp_threshold(linkPWM){           // Function to specify the threshold for temperature fan.
    if(temp>=250 && temp<300){              // If the temperature is roughly >= 25C and < 30C
        linkPWM.pwmDutyRatio(3, 0.33);      // Set its duty cycle (pin 3) to be 33%
        temp_pwm = 0.33;                    // Assign global variable temp_pwm to be 0.33 (use to show at gauge).
    }
    else if(temp>=300 && temp<350){         // Else If the temperature is roughly >= 30C and < 35C
        linkPWM.pwmDutyRatio(3, 0.50);      // Set its duty cycle (pin 3) to be 50%
        temp_pwm = 0.50;                    // Assign global variable temp_pwm to be 0.5 (use to show at gauge).
    }
    else if(temp>=350 && temp<400){         // Else If the temperature is roughly >= 35C and < 40C
        linkPWM.pwmDutyRatio(3, 0.60);      // Set its duty cycle (pin 3) to be 60%
        temp_pwm = 0.60;                    // Assign global variable temp_pwm to be 0.6 (use to show at gauge).
    }
    else if(temp>=400 && temp<450){         // Else If the temperature is roughly >= 40C and < 45C
        linkPWM.pwmDutyRatio(3, 0.70);      // Set its duty cycle (pin 3) to be 70%
        temp_pwm = 0.70;                    // Assign global variable temp_pwm to be 0.7 (use to show at gauge).
    }
    else if(temp>=450 && temp<500){         // Else If the temperature is roughly >= 45C and < 50C
        linkPWM.pwmDutyRatio(3, 0.80);      // Set its duty cycle (pin 3) to be 80%
        temp_pwm = 0.80;                    // Assign global variable temp_pwm to be 0.8 (use to show at gauge).
    }
    else if(temp>=500){                     // Else If the temperature is roughly >= 50C
        linkPWM.pwmDutyRatio(3, 0.99);      // Set its duty cycle (pin 3) to be 99% (Maximum PWM without overflow).
        temp_pwm = 0.99;                    // Assign global variable temp_pwm to be 0.99 (use to show at gauge).
    }
    else if(temp<250){                      // Else If the temperature is roughly < 25C
        linkPWM.pwmDutyRatio(3, 0.1);       // Set its duty cycle (pin 3) to be 99% (For Air Ventilation).
        temp_pwm = 0.1;                     // Assign global variable temp_pwm to be 0.1 (use to show at gauge).
    }
}
```

```
function humid_threshold(linkPWM){          // Function to specify the threshold for humidifier fan.
    if(humid >= 725){                       // If the relative humidity is roughly >= 70%
        linkPWM.pwmDutyRatio(2, 0.1);       // Set its duty cycle (pin 2) to be 10%
        humid_pwm = 0.1;                    // Assign global variable humid_pwm to be 0.1 (use to show at gauge).
    }
    else if(humid<725 && humid >= 700){     // If the relative humidity is roughly < 70% and >= 68%
        linkPWM.pwmDutyRatio(2, 0.6);       // Set its duty cycle (pin 2) to be 60%
        humid_pwm = 0.6;                    // Assign global variable humid_pwm to be 0.6 (use to show at gauge).
    }
    else if(humid<700 && humid >= 650){     // If the relative humidity is roughly < 68% and >= 63%
        linkPWM.pwmDutyRatio(2, 0.7);       // Set its duty cycle (pin 2) to be 70%
        humid_pwm = 0.7;                    // Assign global variable humid_pwm to be 0.7 (use to show at gauge).
    }
    else if(humid<650 && humid >= 600){     // If the relative humidity is roughly < 63% and >= 58%
        linkPWM.pwmDutyRatio(2, 0.8);       // Set its duty cycle (pin 2) to be 80%
        humid_pwm = 0.8;                    // Assign global variable humid_pwm to be 0.8 (use to show at gauge).
    }
    else if(humid<600 && humid >= 550){     // If the relative humidity is roughly < 58% and >= 53%
        linkPWM.pwmDutyRatio(2, 0.9);       // Set its duty cycle (pin 2) to be 90%
        humid_pwm = 0.9;                    // Assign global variable humid_pwm to be 0.9 (use to show at gauge).
    }
    else if(humid<550){                     // If the relative humidity is roughly < 53%
        linkPWM.pwmDutyRatio(2, 0.99);      // Set its duty cycle (pin 2) to be 99% (Maximum PWM without overflow).
        humid_pwm = 0.99;                   // Assign global variable humid_pwm to be 0.99 (use to show at gauge).
    }
}

function lamp_threshold(linkPWM){           // Function to specify the threshold for grow lamp.
    if(light<1023){                         // If light intensity < 10,000 lux.
        linkPWM.pwmDutyRatio(1, 0.99);      // Set its duty cycle (pin 1) to be 99% (Maximum PWM without overflow).
        light_pwm = 0.99;                   // Assign global variable light_pwm to be 0.99 (use to show at gauge).
```

```
    }
}

function thresh_time(linkPWM){          // Function to specify the threshold checking time.
    linkPWM.pwmFrequency(0, 5);         // Set PWM Frequency to be 5Hz (applied to every pin).
    linkPWM.pwmDutyRatio(0, 0);         // Initially set duty cycle of pin 0 to be 0.
    linkPWM.pwmDutyRatio(1, 0);         // Initially set duty cycle of pin 1 to be 0.
    linkPWM.pwmDutyRatio(2, 0);         // Initially set duty cycle of pin 2 to be 0.
    linkPWM.pwmDutyRatio(3, 0);         // Initially set duty cycle of pin 3 to be 0.

    setInterval(()=>{                   // Set time interval function.
        lamp_threshold(linkPWM);        // Call the lamp_threshold function with linkPWM link.
    },3000);                            // Iterate every 3 seconds.

    setInterval(()=>{                   // Set time interval function.
        temp_threshold(linkPWM);        // Call the temp_threshold function with linkPWM link.
    },3000);                            // Iterate every 3 seconds.

    setInterval(()=>{                   // Set time interval function.
        humid_threshold(linkPWM);       // Call the humid_threshold function with linkPWM link.
    },3000);                            // Iterate every 3 seconds.
}

function main(){                        // Main function of this JavaScript code
    const link = new WSLink();          // Create a new link named “link”.
    const linkPWM = new WSLink({        // Create a new link named “linkPWM”.
        conCallback: () => {            // Callback function.
            thresh_time(linkPWM);       // Call the function “thresh_time” with the linkPWM link as a callback.
        }
    });
```

```
psw();              // Call the psw() function

// Graph plot
plot_temp(link);    // Call the plot_temp() function with link assigned
plot_humid(link);   // Call the plot_humid() function with link assigned
plot_light(link);   // Call the plot_light() function with link assigned
plot_ph(link);      // Call the plot_ph() function with link assigned

// Gauge plot
temp_fan(link);     // Call the temp_fan() function with link assigned
humid_fan(link);    // Call the humid_fan() function with link assigned
grow_lamp(link);    // Call the grow_lamp() function with link assigned

// setTimeout and *setInterval* are the only native functions of the JavaScript to execute code asynchronously.
// Temperature
setInterval(()=>{                                       // Set time interval function.
    link.adcGet(3, (id, val, err)=>{                    // Get the ADC3 value in Proteus from link.
        if(err==null){                                  // If there is no error, do.
            temp = val;                                 // Assign val value to global variable temp.
            plotTemp.addPoint(val*100/1023);            // Add val*100/1023 value to plotTemp plotter.
            // gaugeTemp.setValue([val*100/1023]);      // In case user want to show plotter value at the gauge.
            gaugeTemp.setValue([temp_pwm*100]);         // Set temperature gauge value to be temp_pwm*100
            }
    });
}, 3500);                                               // Iterate every 3.5 seconds.

// Humidity
setInterval(()=>{                                       // Set time interval function.
    link.adcGet(2, (id, val, err)=>{                    // Get the ADC2 value in Proteus from link.
        if(err==null){                                  // If there is no error, do.
            humid = val;                                // Assign val value to global variable humid.
```

```
            plotHumid.addPoint(val*100/1023);           // Add val*100/1023 value to plotHumid plotter.
            // gaugeHumid.setValue([val*100/1023]);     // In case user want to show plotter value at the gauge.
            gaugeHumid.setValue([humid_pwm*100]);       // Set humidity gauge value to be humid_pwm*100
            }
    });
}, 3500);                                               // Iterate every 3.5 seconds.

// Light Intensity
setInterval(()=>{                                       // Set time interval function.
    link.adcGet(1, (id, val, err)=>{                    // Get the ADC1 value in Proteus from link.
        if(err==null && val==val){                      // If there is no error, do.
            light = val;                                // Assign val value to global variable light.
            plotLight.addPoint(val*100/1023);           // Add val*100/1023 value to plotLight plotter.
            // gaugeLamp.setValue([val*100/1023]);      // In case user want to show plotter value at the gauge.
            gaugeLamp.setValue([light_pwm*100]);        // Set lamp gauge value to be light_pwm*100
            }
    });
}, 3500);                                               // Iterate every 3.5 seconds.

// pH (range 0 - 14)
setInterval(()=>{                                       // Set time interval function.
    link.adcGet(0, (id, val, err)=>{                    // Get the ADC0 value in Proteus from link
        if(err==null){                                  // If there is no error, do.
            // plotPh.addPoint(val*100/1023);           // This is the 0-100 scale.
            plotPh.addPoint(val*14/1023);               // Set pH plot value from 0 to 14
            }
    });
}, 4000);                                               // Iterate every 4 seconds.

gridChild();                                            // Call the gridChild() function to add objects into grid
```

References

Arduino. (2018, February 5). Read Analog Voltage. https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage.

Freshbox Farm. (n.d.). Controlled Environment Agriculture (CEA): More than Hydroponics. http://freshboxfarms.com/articles/controlled-environment-agriculture-ceahydroponics

Green CulturED. (n.d.). Essential Marijuana pH Measurements. https://www.greencultured.co/proper-marijuana-ph-levels-for-optimal-growth/.

Honeywell. (2010, March). HIH-5030/5031 Series Low Voltage Humidity Sensors. https://sensing.honeywell.com/index.php?ci_id=49692.

I49 USA. (2020, March 10). Optimal Lights for Cannabis. https://i49.net/blog/gardening-tools-and-equipment/types-of-lights/optimal-lights-for-cannabis/.

James. (2013, April 21). NFT Systems, my experiences with them and a brief rundown on how they work. https://hydroponicsinfomation.wordpress.com/2013/04/21/nfthydroponics/.

Kubala, J., & Chen, J. (2021, May 6). Can You Eat Weed? All You Need to Know About Marijuana Edibles. https://www.healthline.com/nutrition/eating-weed.

Lubbe, G. (2021, February 21). What Countries are Next to Legalize Cannabis in 2021?. https://cannabistraininguniversity.com/blog/marijuana-laws/what-countries-next-to-legalize-cannabis-2021/.

Michael. (2019). What is Nutrient Film Technique - NFT Hydroponics?. https://www.nosoilsolutions.com/nutrient-film-technique-nft-hydroponics/.

Myster. (2020, June 6). What Is The Best Humidity For Growing Cannabis Plants?. https://www.getmyster.com/blogs/blog/what-is-the-best-humidity-for-growing-cannabis-plants.

Nuratch, S. (2021). INC281-2021. https://github.com/drsanti/INC281-2021.

O'Neill, L. (2020, August 28). More People Are Consuming Cannabis During COVID-19. https://thegreenfund.com/more-people-are-consuming-cannabis-during-covid-19

Royal Queen Seeds. (2018, August 28). How To Use A Lux Meter To Increase Your Cannabis Yields. https://www.royalqueenseeds.com/blog-how-to-use-a-lux-meter-to-increase-your-cannabis-yields-n977.

RS Components. (1997, March). Light dependent resistors. https://components101.com/asset/sites/default/files/component_datasheet/LDR%20Datasheet.pdf.

Sensorex. (2021). HYDROPONICS WATER QUALITY. https://sensorex.com/hydroponics/.

Texas Instrument. (2017, October). LM35 Precision Centigrade Temperature Sensors. https://www.ti.com/lit/ds/symlink/lm35.pdf.

Thompson, E. (2019, August 12). Why more people are consuming cannabis edibles. https://theleafdesk.com/guide/why-more-people-are-consuming-cannabis-edibles/

Vernon, J. (2019, December 16). WHY DO PLANTS GROW BETTER IN A GREENHOUSE?. https://hartley-botanic.com/magazine/plants-grow-better-greenhouse/.

Whipker, B., Smith, J. T., Cockson, P., & Landis, H. (2019, November 6). New Research Results: Optimal pH for Cannabis. https://www.cannabisbusinesstimes.com/article/new-research-results-optimal-ph-for-cannabis/.