Term Project – Final Report

Phys 3361
Digital Signal Processing
Mount Allison University
11 December 2019

Michael McEachern
Brian Beardsall
Logan DiAdams

1 INTRODUCTION

The basic concept for the project was to construct an indoor automated gardening system. It was to provide optimal growing conditions for any plant left inside with minimal input from the user. This concept can be seen applied in real world settings in areas such as agriculture, aquaculture, scientific research, and others where environmental control is crucial when dealing with living matter. The three basic control systems needed to achieve a basic controlled climate in the case of this project were identified as: water, temperature, and light.

The water system encompassed monitoring and maintaining of soil hydration levels over set thresholds. The main working components in this system were a water pump, moisture sensor, and water reservoir. The system was to be able to detect the moisture level of the soil the plant was planted in and if dryer than the directed by the user, activate the pump until the moisture levels returned accepted levels.

The light system was comprised of two LED lamps and a photo-resistor to monitor and control the light levels inside the growing chamber. Plants are constantly experiencing photo-inhibition, a breakdown in the molecular machinery that carries out photosynthesis, due to damage by light-induced radicals. Exposure of a low-light preferring species to high light levels can inhibit photosynthesis, and negatively impact growth (Studies on the Mechanism of Photoinhibition in Higher Plants, http://www.plantphysiol.org/content/plantphysiol/67/6/1161.full.pdf). Therefore, it would be ideal to have the ability to vary light levels, depending on the plant being used. The LEDs were emitting light in the red and violet wavelength range, optimal for plant growth, peaking at 460 and 660 nm. The photoresistor was used to maintain light levels specified by the user.

The temperature system included a Peltier device, two fans, and temperature sensor. Temperature is also a primary factor for determining plant growth. Each species has an optimal temperature range, and vegetative or reproductive stages of growth can have different temperature optimums (Temperature extremes: Effect on plant growth and development:
https://www.sciencedirect.com/science/article/pii/S22120947153001160). Close control of temperature is vital for a useful automated gardening system.

These three systems outline the basic working model of the project. The objective was to then establish and expand on these systems to achieve as much automation in the various operations as possible. The end goal was to create a system that required no human interaction to successfully tend to plant inside.

2 THEORY

2.1 Hysteresis Temperature Control

For temperature control, it was necessary to introduce some form of hysteresis so that the Peltier would not be constantly switching between cooling and heating when the input temperature oscillated around the setpoint.

The idea for a simple Peltier controller with hysteresis was found from a temperature controller manufacturer, uwetronic GmbH. A hysteresis value defines the start and end of the heating and cooling functions.

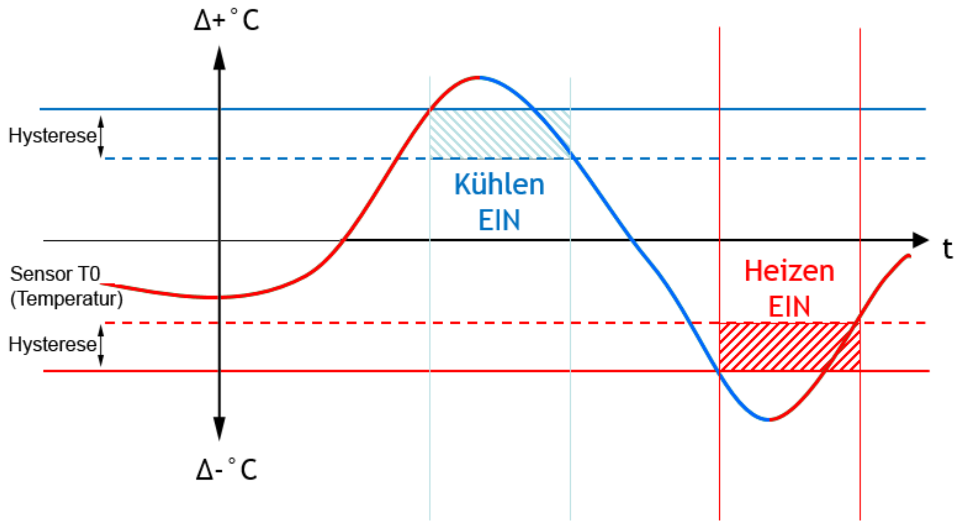


Figure 1: Peltier Hysteresis.

(Figure from https://www.uweelectronic.de/de/temperaturmanagement-2/temperaturregler/regelung-peltierelementen.html).

In the design for this project, the Peltier has three states- cooling, heating, and off. Once a state is entered, the high and low thresholds change to make it more difficult to leave the current state. A diagram of this behaviour is shown below.

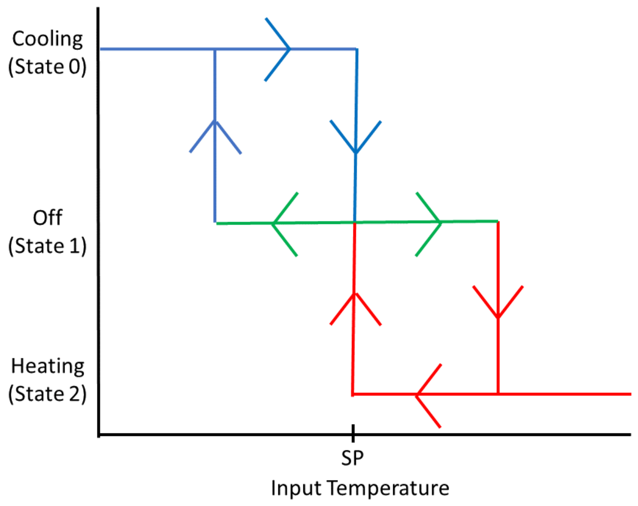


Figure 2: Peltier Hysteresis.

2.2 PID Control Algorithm:

To keep the light intensity at a set level, PID control was used. PID (proportional–integral–derivative) controllers are widely used in many applications, where precise, continuous control is desired. In a survey of over 11 000 controllers in the refining, chemicals, and pulp paper industries, 97% of controllers used a PID control algorithm (Increasing Customer Value of Industrial Control Performance Monitoring—Honeywell's Experience, Desborough & Miller, 2001).

In a feedback controlled process, it is desired that the output variable, y, follows the setpoint value, r, by varying the manipulated variable, u. The error, e, is calculated as r-y.

The first element of PID control, P, is proportional to the current value of the error term, e. The I element is proportional to the integral of previous error values, up to the current time t, which is analogous to "past error". The D element is proportional to the derivative of the error at time t, which represents predicted "future error". (Araki, M. "PID Control", http://www.eolss.net/ebooks/Sample %20Chapters/C18/E6-43-03-03.pdf).

In a full PID controller, all three elements summed to set the manipulated variable such that control error is minimized, through the following equation:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{1}$$

(https://www.pdx.edu/nanogroup/sites/www.pdx.edu.nanogroup/files/2013_Arduino%20PID%20Lab_0.pdf)

Where Kp, Ki, and Kd are constant parameters which must be set.

A block diagram of this process is as follows:

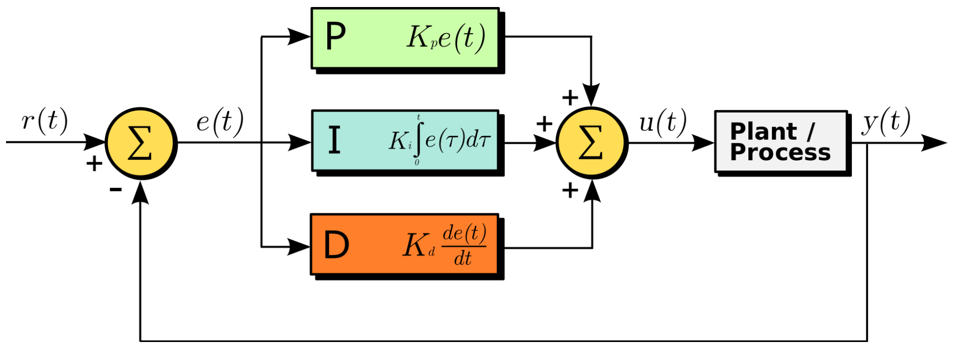


Figure 3: Block diagram of a PID controller.
(Arturo Urquizohttp://commons.wikimedia.org/wiki/File:PID.svg)

However, the PID controller used in this project is just an "I" controller- due to the fast response time of the LED, it was found that introducing P and D control increased oscillations and error.

In addition to the circuit, a fixed laser sits in front of the pivoting mirror of the moving arm. The laser light is reflected back onto a nearby surface and is used to determine the displacement between the parallel plates by the use of a decided reference point.

3 DESIGN

3.1 Construction

The physical layout was a major aspect of the project to ensure the effectiveness of the various systems. For steady operation, volume, insulation, ventilation, and accessibility were all considerations. The design went through several drafts until one was found to meet all requirements and was feasible with the materials available to us.

The layout consisted of an approximately 1'x1'x1' square chamber with a 1'x1' outcropping to serve as a hub for the components of the project. The primary material was plywood with white Polystyrene foam insulation lining the inside of the cube chamber. Two entries were made into the chamber to allow for the mounting of the temperature system and lights to the interior. Lastly, a wall was made into an access door to the chamber.

The initial designs of the project included slanted walls and windows to allow for outside light to enter into the chamber and help maintain the health of the plant. This was ultimately changed to a completely isolated chamber. This would allow for a simpler design, more effective insulation and therefor control of the internal climate of the chamber, and would better serve as a proof of concept. The final design would better demonstrate the achievement of the project's goals by allowing the systems working with as little external input as possible.

3.2 Components

Peltier:

A peltier device is a solid-state thermoelectric device capable of exchanging heat. It operates using the thermoelectric effect, which is the direct conversion between temperature differences and electric voltage. In this case, electric voltage was used to create temperature differences between the two sides of the device.

Photoresistor:

A photoresistor is an active component that decreases resistance with respect to light levels on the component's surface. The resistance of a photoresistor decreases with increasing incident light intensity. It is made of a high resistance semiconductor. In the dark, a photoresistor exhibits a high resistance, and gradually decreases with light levels.

Moisture sensor:

We made use of a soil moisture sensor to monitor plant levels. The moisture sensor is designed with two probes, these probes are electrodes which allow current to pass through the soil between them. When the soil is wet, it conducts electricity better than when it is dry. In this way the component acts as a resistive sensor connected to an output module.

Relay:

Relays are essentially switches which you can turn on and off by applying electricity to a coil inside the relay. When the coil is energized, it creates a magnetic field which attracts a metal contact inside and mechanically closes the switch. When the power source is withdrawn, there is no longer a magnetic field and the switch opens again. Different relays require different voltages to energize the coil, for our project we made use of 5V and 9V relays, which means it requires 5V and 9V DC respectively to energize the coil inside to open and close the contact.

4 OPERATIONAL SUMMARY

Computing:

An Crowduino (Arduino UNO knockoff) contained all the control logic for the systems. The Arduino has digital output pins, 3 of which have PWM capabilities, and most importantly 6 analog input pins, which use an ADC to measure input voltage.

The Arduino communicates with the Raspberry Pi over a USB serial connection. The Arduino sends the current readings and thresholds to the Pi, and receives new threshold readings from the Pi.

A simple Tkinter user interface ran on the Pi, to display current values from the Arduino for light, temperature, and moisture, and allow the user to input new setpoints.

The Raspberry Pi writes the current values, setpoints, and a timestamp to a CSV file, which is uploaded to a POST form on a Web App, allowing the user to view graphs of the values with respect to time, as well as the most recent readings written to the CSV.

Light Input:

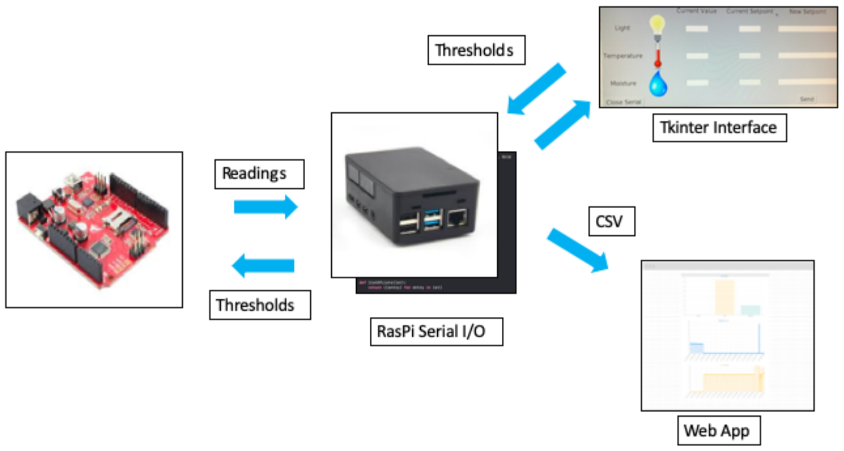


Figure 4: Overview of communication between components.

A photoresistor was used to measure light intensity. The photoresistor was placed in a voltage divider configuration with a 10 k ohm resistor, between the 5V arduino supply and ground. An Arduino analog input pin was connected between the 10 k ohm resistor and the photoresistor.

Moisture Input:

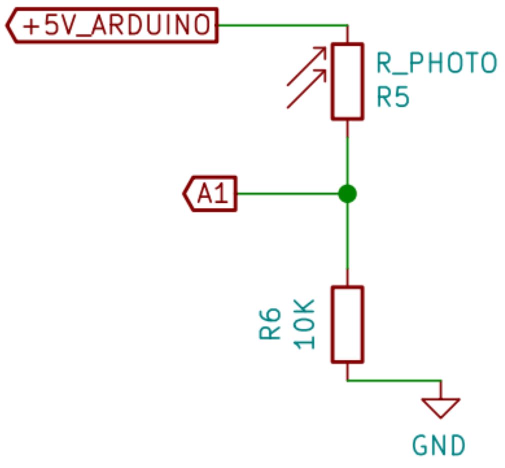


Figure 5: Light input.

A resistive soil moisture sensor was used (Model YL-69), attached to an output module (Model YL-38). The module was powered by the arduino's 5V supply, and its analog output connected to an arduino analog pin. The digital output, which outputs a high signal when voltage is above a cutoff value set by a potentiometer, was not used.

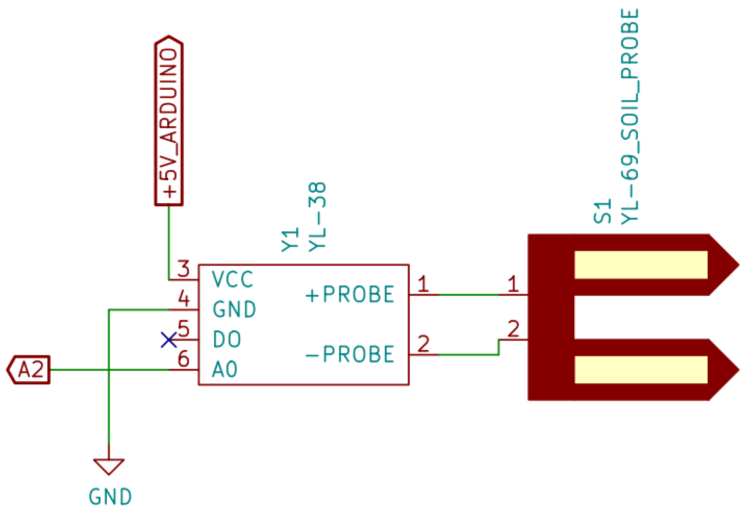


Figure 6: Soil probe circuit.

Temperature Input:

An LM35 precision temperature sensor IC was used to monitor temperature. The datasheet specifications state that it is accurate to 0.5°C, with a linear 10 mV/degree celsius scaling factor (Texas Instruments LM35, http://www.ti.com/lit/ds/symlink/lm35.pdf). However, a large variation in temperature readings was present when the LM35 was wired up at first. After consulting the datasheet, large wire leads could act as antennae, increasing noise. To remedy this, the wire leads were shortened, and a 2 k ohm pulldown resistor was added to the output of the LM35. These modifications greatly decreased noise.

Light Output:

An LED grow light was used to provide light for the system, with peak wavelengths at 460 & 660 nm. The grow light came as a desk clamp, with two light bars connected by a flexible metal housing. In the final design, the clamp and metal housing was removed, leaving two separate LED light bars, each in 10" long cylindrical plastic casings, with a diameter of 0.7".

The grow lights came with a manual control switch PCB, with four buttons to control the brightness, set a timer, and switch the colour of the lights (only blue, only red, or both colours). This manual switching was not desired, so an alternative was needed.

The grow light operated at 5V (supplied by an included USB wall adapter), and at full brightness drew a combined 2A of current, to give a power consumption of 10 W. This current draw was much too high to be controlled by the Arduino, or even a low-power BJT.

The control circuit that came with the lights was examined, and found to include two surface-mount MOSFETs (model A2SHB). The gate of the MOSFETs was identified from their datasheet (https://datasheetspdf.com/pdf-file/1381523/HAOHAI/A2SHB/1), and through testing it was determined that one MOSFET controlled the blue LEDs, while the other controlled the red.

The emitter of a 2N222A BJT transistor was wired to the gate of the MOSFETs, with its collector attached to the 5V grow light supply. The base of the 2N222A was connected to an arduino PWM digital pin, with a 10K current limiting resistor.

Temperature Output:

A Peltier device (Model TEC1-12706) was used to provide both heating and cooling power. The device has a maximum voltage rating of 14.4 V, with a maximum delta T (difference in temperature between the two sides) of 66°Celsius (TEC1-12706 datasheet, https://peltiermodules.com/peltier.datasheet/TEC1-12706.pdf). However, the Amazon product page stated that it was designed to use 12 V. When connected to a 12 V power supply, 2 A of current were drawn, giving a power consumption of 12 W.

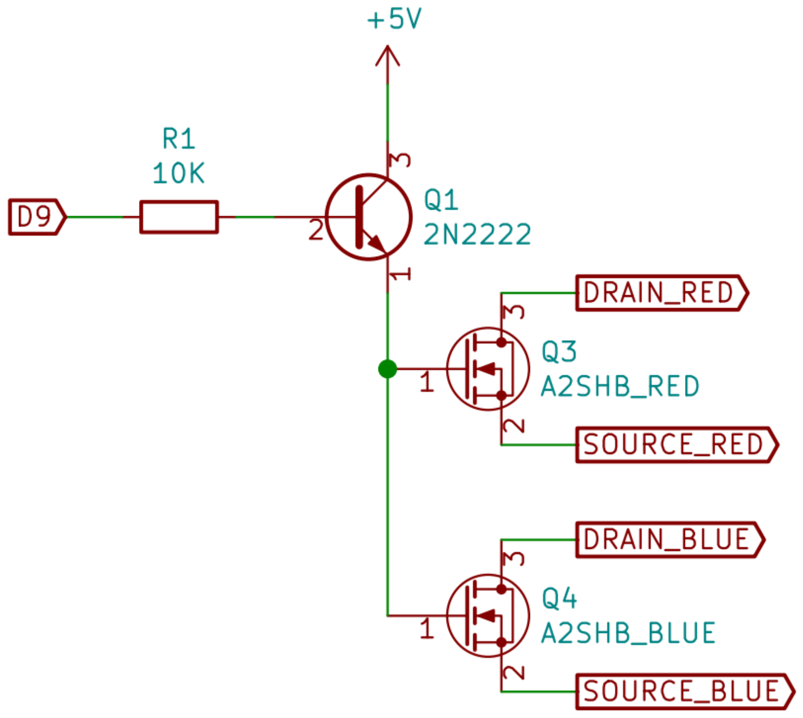


Figure 7: Light output circuit.

The power supply used the Peltier device was a 12 V wall adapter, with a maximum current output of 3 A. This wall adapter was dedicated only to the Peltier.

A useful feature of the Peltier device is that reversing the polarity of the applied voltage will swap the hot and cold side. To add the capability of switching polarity, an H-Bridge integrated circuit was considered, but not used due to its current maximum of 1 A. One Double-Pole-Double-Throw (DPDT) relay could be used to simply reverse the voltage, but would not allow for turning off the Peltier. In the final design, two DPDT relays (G5V-2-H1-DC9) were used, in conjunction with two NPN BJT transistors (2N2222) acting as switches (controlled by Arduino digital pins). The layout of the circuit is shown below.

When both Q5 and Q6 are off, both of the Peltier terminals are at +12V (no voltage difference).

Similarly, when both Q5 and Q6 are on, both terminals are grounded. When Q5 is on, (Digital pin 2 is High), but Q6 is off, the top terminal is connected to Ground, while the bottom terminal is at +12V. There is a 12V difference, so the Peltier is on. If Q5 turns off, and Q6 turns on, the

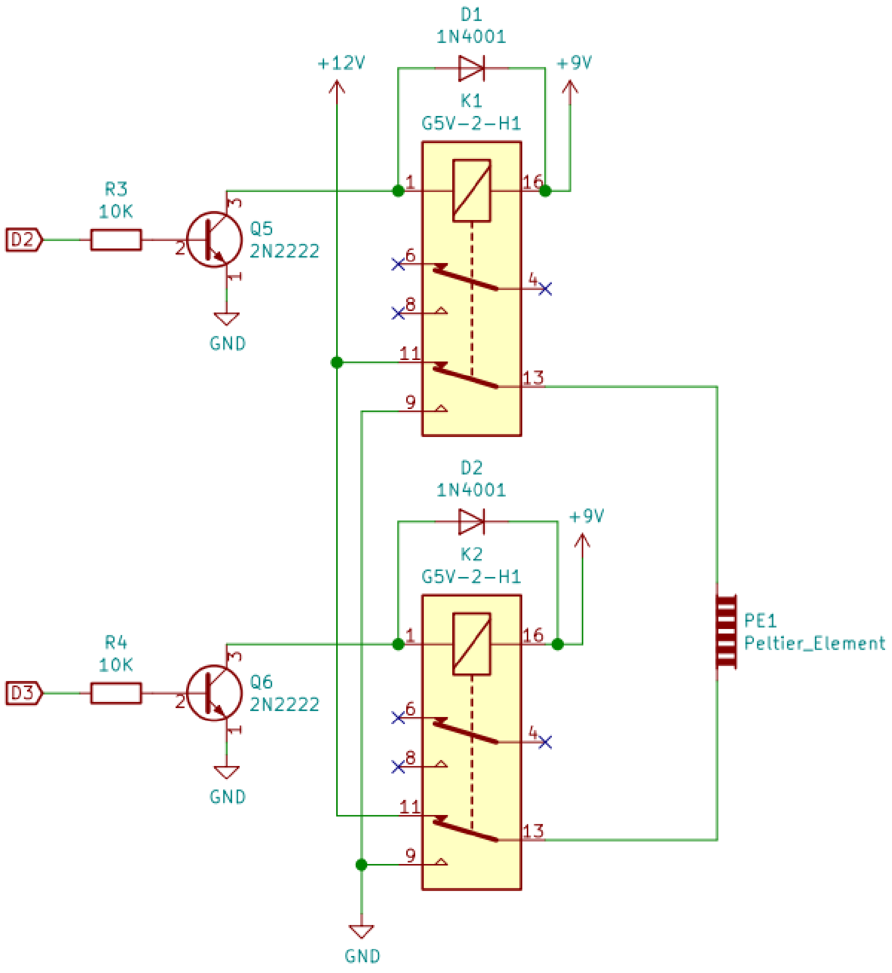


Figure 8: Relay circuit.

top terminal is at +12V, while the bottom is grounded. Current flows in the opposite direction, reversing the hot and cold sides of the Peltier.

Because the relay coils are inductors, reverse biased diodes were placed across the coils to dissipate the back EMF and protect the transistors when switching on and off.

The Peltier device will rapidly burn up without a heatsink on the hot side, so two PC heatsinks with 12V fans (generously donated by Downtown Digital) were mounted on either side, with metal oxide thermal paste applied to improve heat transfer.

The fans were controlled by a BJT transistor, with a 510 ohm resistor at the base connected

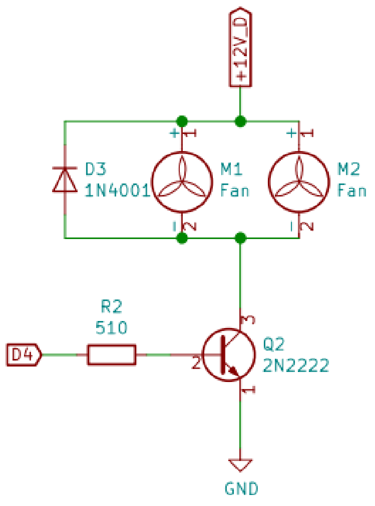


Figure 9: Fan circuit.

to a digital output pin. The lower resistance was necessary at the base to ensure that enough base current flowed to be near saturation.

At first the same 12V supply as the Peltier was used, but when the Peltier was on the voltage dropped considerably, lowering the fan speed. We didn't have any other 12V power supplies, so instead a 13.5V power supply was dropped to approximately 12V by connecting two diodes (1N4001) in series .

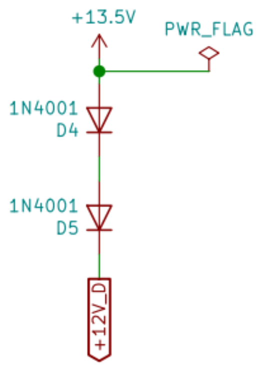


Figure 10: 12V from 13.5V supply using series diodes.

Water output:

A peristaltic pump was used to bring water from an outside reservoir into the enclosure, through vinyl tubing. The pump used the same 12V power supply (13.5V dropped through two diodes) as the fans, and was controlled by a 5V relay module (SRD-05VDC-SL-C), attached directly to an Arduino digital output pin.

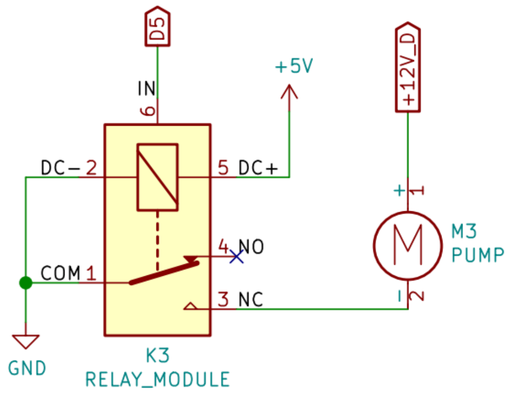


Figure 11: Pump circuit.

The flow rate of the pump was measured as being approximately 1 mL/s.

Web Interface and Database:

The website was created using a high-level web framework for Python called Django. The appeal to Django was its versatility. It can be used to design practically any type of webapp and can deliver content through HTML format easily, and comes with a lot of features, straight out of the box. Coupled with Django and HTML5, we also made use of a JavaScript library for creating charts with an HTML5 canvas, this library was Chart.js, as it was able to create elegant graphs with appealing animations as well. With this, another software library that was made use of was Pandas, used in the web app for reading in columns and rows of a CSV, and then sending that data to be used to make our graphs.

The Web App design was primarily focused on a Home View which displayed 4 graphs in total. However the application also makes use of another view '/upload/' when making POST requests from a python script on the Raspberry Pi itself. There are other views included within the code that were only designed for testing purposes, but the Home and Upload Views are what was made use of within the final product.

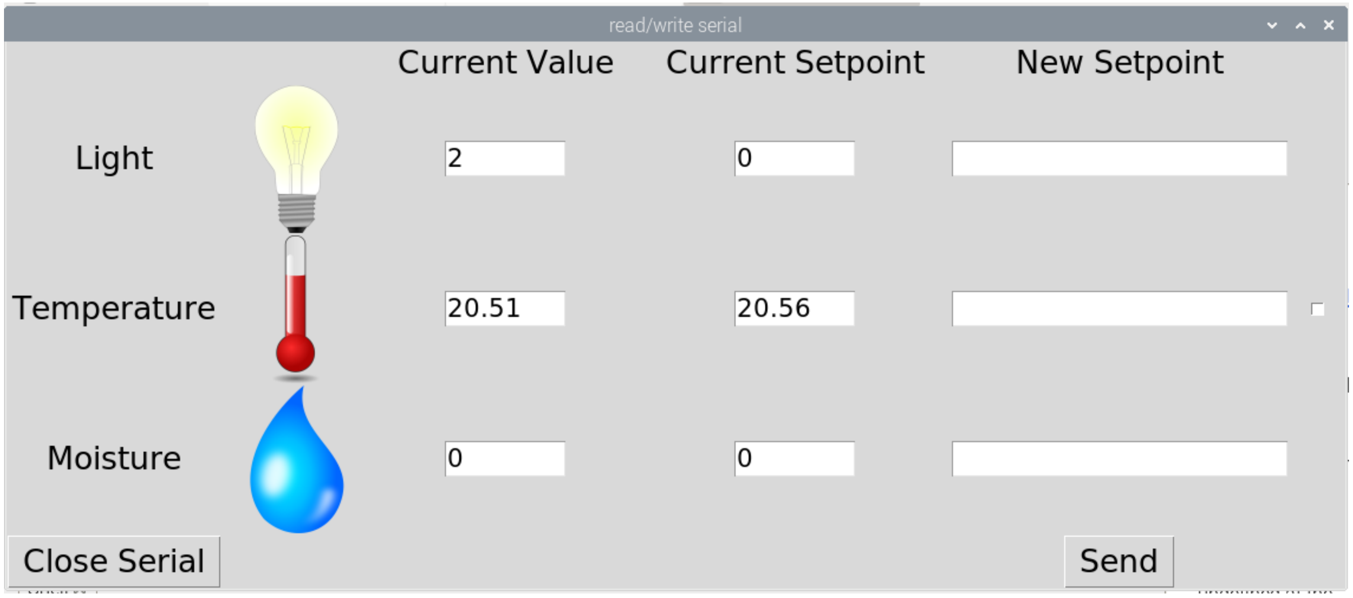
On the main page of the site, a user is presented with 4 graphs. An initial graph which displays the current readings of moisture, light, and temperature values. As well as graphs showing the values of moisture, light and temperature at each timestamp (in our demo, at each second) as the website receives these readings.

The upload view is useful if a user wishes to upload one csv file to the webapp, with matching headers to the csv given by the Pi. They will be prompted to select a file and upload it, once they click 'upload' the home page will be reset with the new data, and the user can notice this if they return to the main view. How the python script on the pi is working is that it is making these requests to this upload form in an infinite loop every second with the CSV as it is written to with current values from the Pi, every second, until we cancel that script. Ideally, if we were to have this read in values for a very long time, we would only have the CSV file update every few minutes or every hour, so as to minimize a ridiculous number of data points.

Graphical User Interface (GUI):

A GUI, designed to be run on the Raspberry Pi (connected to the Arduino), was created using the python GUI package Tkinter. The program displays current values and setpoints, received from the Arduino, and allows users to input desired setpoints, which are sent to the Arduino. The user was also given the option of turning the temperature control on or off (via a checkbox). This feature was added mostly for demonstration purposes, to prevent the noisy fans from turning on and off.

The program writes received values and setpoints to a CSV file with a timestamp, which is accessed by the webapp. This CSV file was also used to graph the response of the system to setpoint changes during testing.



	Current Value	Current Setpoint	New Setpoint
Light	2	0	
Temperature	20.51	20.56	
Moisture	0	0	

Figure 12: Screenshot of the GUI.

5 RESULTS

Temperature Control:

Heating was found to be more efficient than cooling. With an ambient temperature of 20 deg C, the system was able to reach a temperature range of 16- 30 deg C.

The response of the system to temperature setpoint changes was recorded, and results from the CSV file were graphed using the R package ggplot2.

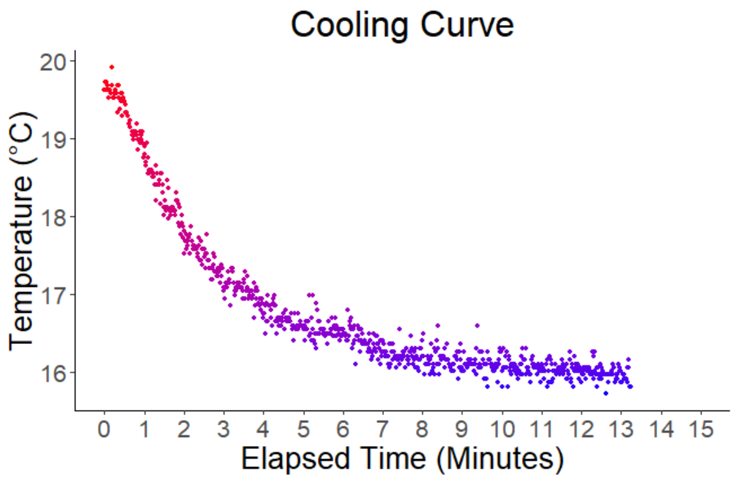


Figure 13: Change in temperature over time when cooling.

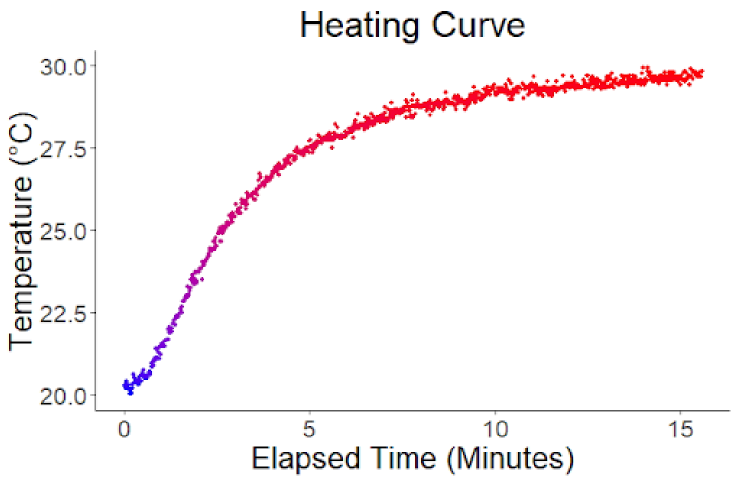


Figure 14: Change in temperature over time when heating.

Light Control:

After receiving a light setpoint, the system took approximately 1 second to respond.

At a setpoint of 74% brightness, the mean error ($\bar{e}$) was 0.4, with a standard deviation (σ_e) of 1.7. The mean measured value was 74.4%.
At 49% brightness, $\bar{e} = 0.2$, $\sigma_e = 2.2$, and mean measured value = 49.2.

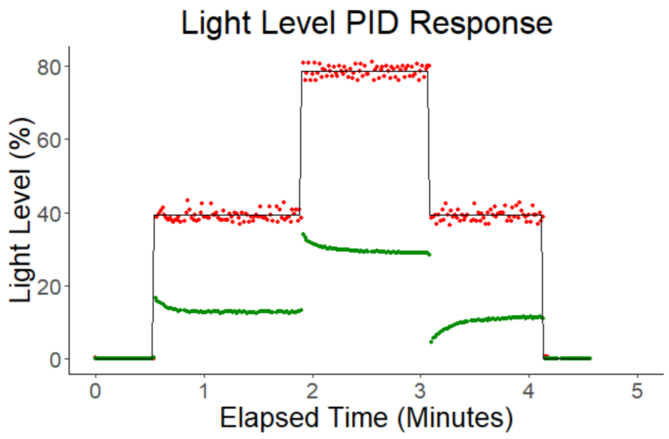


Figure 15: Black lines represent the current setpoint, red dots are measured values by the photoresistor, and green dots are the output voltage sent to the light bar.

Soil Moisture: The soil moisture sensor was the least sensitive of the three sensors. When the pump turned on, soil moisture went from 0-40% almost immediately.

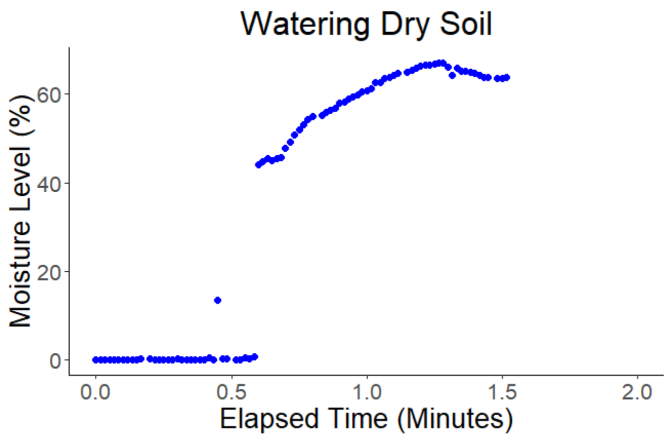


Figure 16: Response of the system when the moisture setpoint was set to 60%.

Webpage plots:

Below are examples of data provided by the website, as the page is refreshed, each graph will update respectively to show changes to the plant's environment.

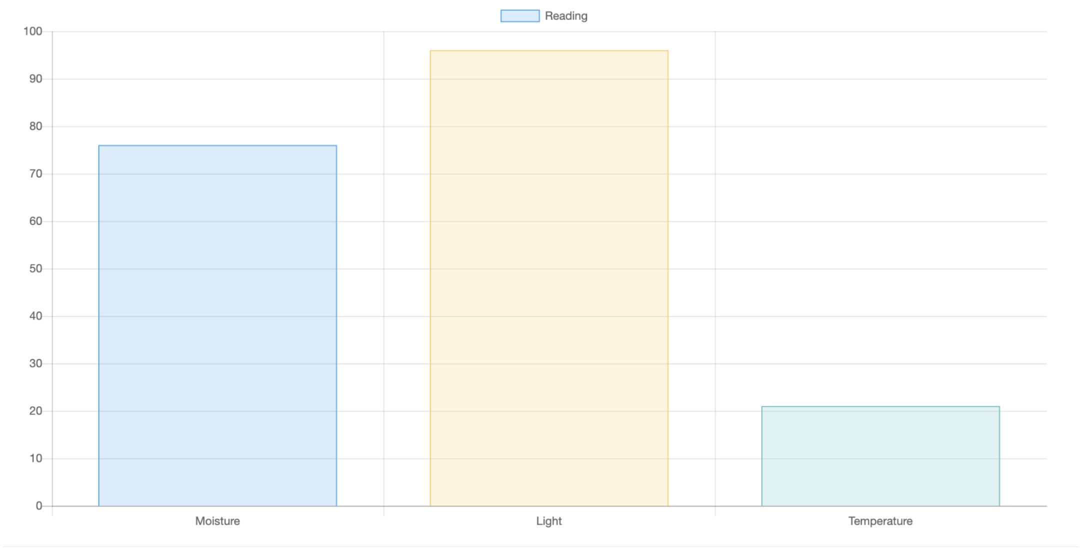


Figure 17: Notice the scale on the left goes to 100, this is because the values of moisture and light are given in terms of percentage, while temperature remains in celsius.

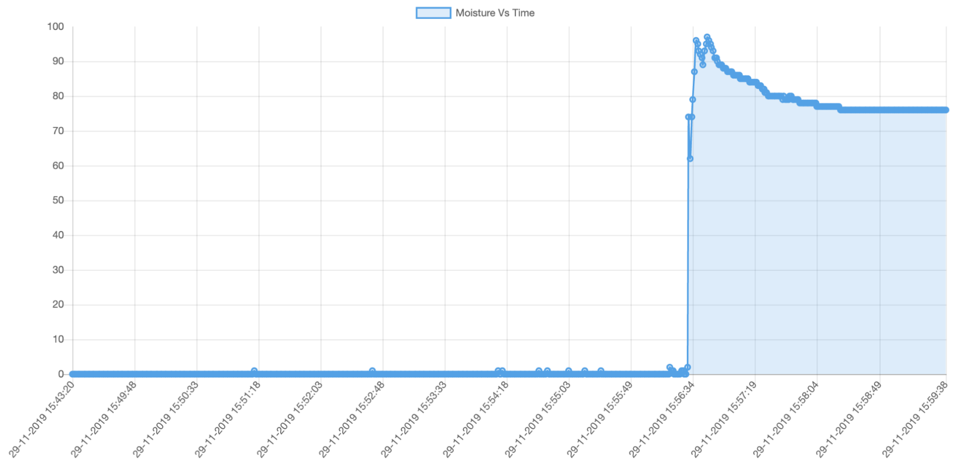


Figure 18: Illustrates the readings of the soil moisture sensor as it goes from completely dry to receiving water from the reservoir, and it then settles around a 76% moisture level reading with time.

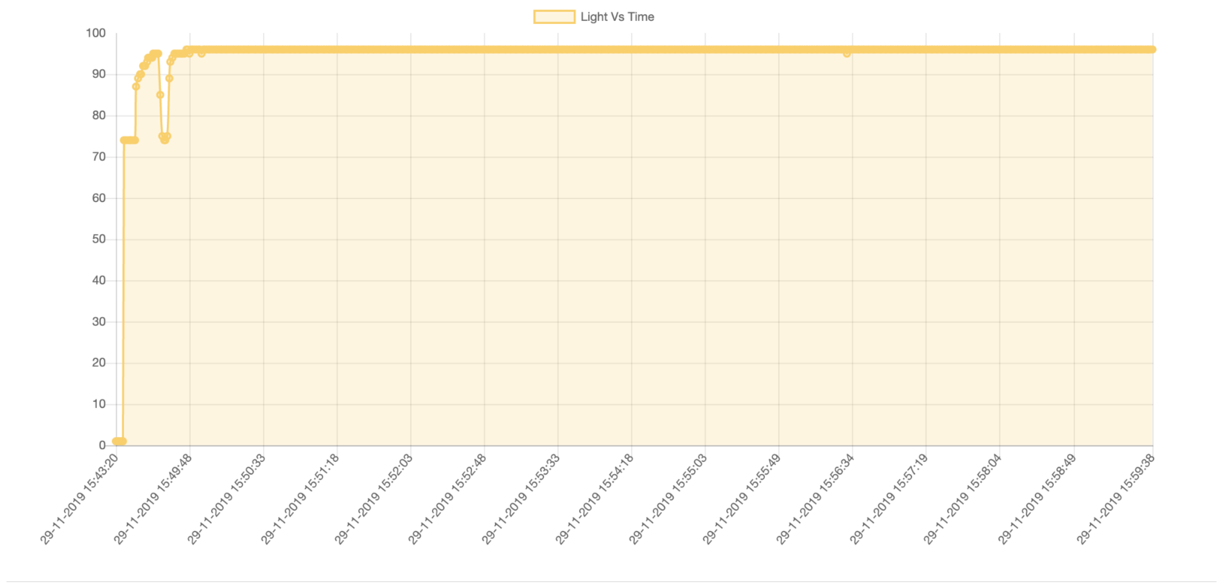


Figure 19: Demonstrates a plant going from receiving no light from both outside the enclosure or from the grow lights, to receiving complete exposure when the lights are turned on to near full brightness.

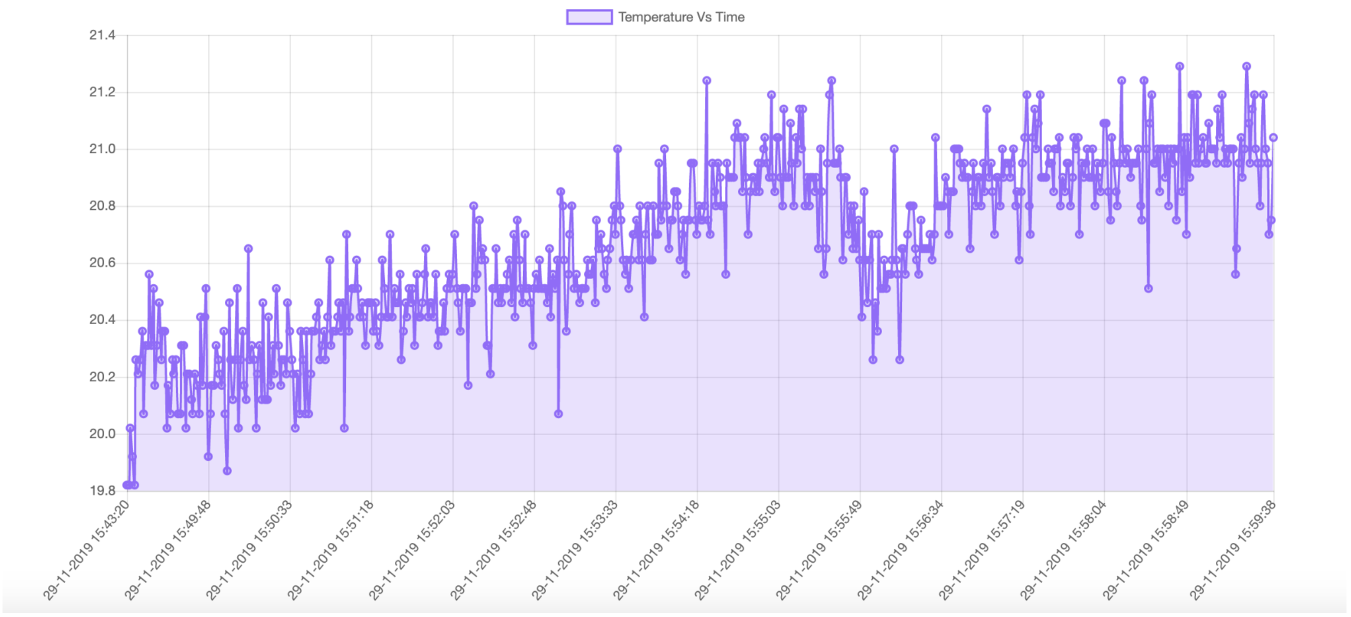


Figure 20: Variations in temperature of the plant in time, as shown, the plant here is resting in an environment at about 19.8°C, and at some points, the surrounding temperature reaches levels of above 21.2 degrees in this demonstration.

6 DISCUSSION & CONCLUSION

Web Design:

Web Development can often be challenging, as it is no easy task. It is important to consider speed, scalability and security when designing a Web Application, and sometimes these factors conflict with the actual task which you are trying to achieve through your Web Interface. This is a problem which was encountered in the creation of our Web App to interface with the Raspberry Pi. The initial goal of the Application was to have everything communicate through the deployed website, both input and output. We envisioned a web-based platform which would display readings of our sensors in real-time, organize data into graphs, adjust the readings through web scraping of current weather conditions with the functionality to match those if a user chooses, and we had hoped to have the capability of using the App to set thresholds similar to the way the GUI application does directly from the Pi.

It was not anticipated that these tasks would be far from easy given the capacities of the framework chosen. It could be a potential lack of experience in using Django, or the approach to the task or perhaps a mixture of both, however, in the end, the only thing which was truly accomplished as anticipated was real time data and graphing from the online interface, this turned out to be difficult enough given the obscure process required to achieve this concurrently with the Pi. Communicating between a Web App initially designed on another computer, and a Raspberry Pi device involved a deep dive into Django security and required some new knowledge of the system set up which is beneficial for future projects, but seemed to only be useful for a handful of scenarios similar to that which was experienced here.

Getting the Django application working and displaying graphs given a CSV turned out to be not so difficult after the incorporation of "Chart.js", a Javascript Library which coupled with the HTML code. And tweaking this to function properly from our specific CSV into a Django Application was a smoother process. Difficulties came when dealing with some more logic-based preliminaries. The Pi only appended to a CSV with every second as it read in the values, so it never overwrites itself, this is something which needed to be considered as the Django Application read in the data. If we wanted real time data, with the way it was being designed, we would need to read in the same CSV (given by the RasPi) and overwrite the data here every time the CSV received an update / a new row for the graphing. The Django application needed to read in the same CSV every time, and overwrite that file if it had the same name as the previous so that the new data could be displayed, it turned out to be too difficult to apply only any changes to the file given our setup, so the approach of overwriting the file each time while potentially slower, functions well for our purposes since the old data never changes in the Pi's CSV.

We needed the functionality of having a file automatically overwrite itself when a file of the same name and type was uploaded through a form, so that we could continuously upload our information from the Pi as it's updated, and have the Web App (which only reads from a CSV called values.csv) autonomously receive a POST request and make use of the data for graphing purposes. There were intricacies involved in the process of having a form on the website (/upload/) with this function.

Finally once this was achieved, and the website could be deployed, we needed one more script which would be hosted on the Raspberry Pi to send the CSV over and over again, this turned out to be a bit of an obscure task and really quite difficult without previous knowledge on Django's 'CSRF Tokens' which protect the Website from "Cross-Site Request Forgery Attacks" where you are not the only one with HTTP request access control of the website. Because this was not really considered when writing the Pi script to post the CSV to the upload form, it turned out to be a relatively big problem before it was realized because the error was very inspecific: "Rejected Requests: 403 Forbidden Response". Fortunately, the internet is an ocean of knowledge and it was realized that we needed to fetch a CSRF Cookie first and then use it for submitting the file. After this final adjustment, the Web App and Raspberry Pi script were working as designed, and everything was running smoothly with the other components of the project including the other GUI interface for altering thresholds and setpoints!

Structure Design:

If this project were to be expanded for general purpose plant care, there are three things that need to be re-evaluated or improved in the future; insulation, humidity control, capacity. All three are interconnected and can directly or indirectly affect the others. For the initial scope of the project, a lightweight design was acceptable for demonstrating the various achievements of the project. It still remains to be shown however, the current design's long-term drawbacks as there was insufficient time to allow these inherent flaws to become revealed.

If capacity of the housing chamber were to be increased, this would have a direct effect on the abilities of the temperature control system currently in place. The peltier device currently used may need to be exchanged for a more effective model, or perhaps traded for more commercial options such as vapor-compression systems. Either change would then have implications on the power draw of the system. Further investment into better insulation and heat exchange would help to compensate for the greater strain.

With temperature and capacity satisfied, a further goal would be to implement a humidity control system. Any long-term use of the system as is would eventually encounter issues with condensation within the growing chamber. This poses a risk to any nearby electronics exposed. With the ability to control humidity, there is less chance of complete saturation of the electronics, but even so, all open wiring must be water sealed to properly protect the function of the systems in place.

Electrical Design:

The light control BJT, attached directly to the gates of MOSFETs in the PCB which came with the light bar, sunk a significant amount of current when turned on. Theoretically, the MOSFETs themselves should not sink any current, so this current draw could be coming from the rest of the light bar circuit. Wherever the current draw came from, it was significant enough to heat up the transistor, requiring a makeshift heat sink (aluminum from a pop can attached to the casing with thermal paste). For economical reasons, this was sufficient, but future designs could ditch the light bar PCB and use standalone MOSFETs.