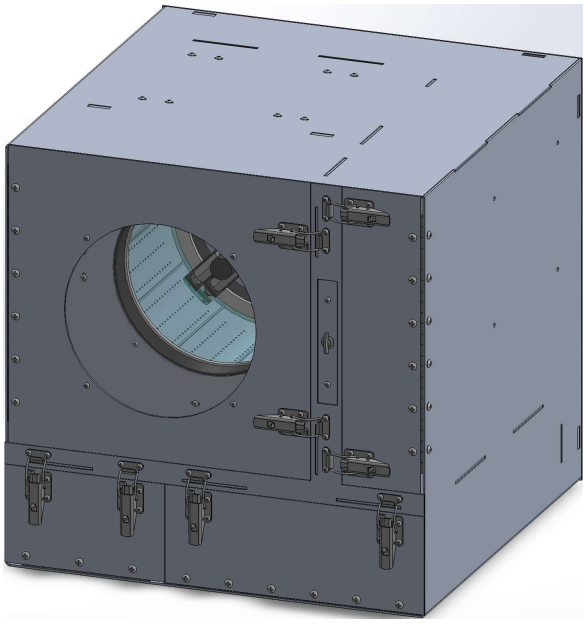
Final Report

Rotational Plant Growth R.P.G.



Name: Omar Odeh: (Abstract/Project Scope/Electrical Designs)
Vitas Diktanas (Mechanical Designs/Load Calculations)
Gregory Henry & Kevin Munoz: (Introduction/The Problem)
Christopher Cadenas: (Title Page/Project Plan/General Formatting of Report)

Design Project

EML 4551-001

Professor: Oscar Curet

Due: April 26th, 2022

Abstract

Long term space travel is severely hindered by the inability to create certain resources on demand; food and nutrients being of utmost importance for astronauts. In this report a novel design to grow microgreens is presented with a unique approach to resolve the complications that arise with the absence or reduction of gravity. Gravity influences most physics of everyday life which humanity has been forced to take into consideration with any engineering design. Infrastructure, aerodynamics, and many other subfields derive the behavior of a system often dependent on the gravitational constant. The absence of gravity not only creates a severe complication in many mathematical derivations but allows for other aspects of physics to have more influence over the system. The method presented has potential to grow microgreens by simulating a gravitational environment and allow plants to grow as they would on Earth.

Table of Contents

Introduction

As space exploration expands further from Earth, astronauts need plants for both aesthetic and practical reasons. Fresh plants play a critical role in the psychological well-being of space explorers [1]. The deficiency of vitamin C is the root cause of sailors' scurvy and can be an appropriate metaphor for similar health issues that astronauts can encounter during long space travel. Currently on the space station, a wide range of freeze-dried and prepackaged meals are delivered through regular shipments to fulfill their dietary needs. However, these prepacked vitamins break down over time when the crew explores deep space without any resupply shipments for months or even years [2]. Depending on the length of the mission, more mass and volume of prepackaged food would be needed. Due to this problem, scientists are looking for different ways to create a long-lasting and easily absorbed source of nutrients in the form of fresh fruits and vegetables in a closed environment without gravity and sunlight. In this report, we present a novel plant growing mechanism for the plantation of micro-red radishes in a closed environment which can provide fresh and nutritious food for astronauts. The red radishes have a relatively short cultivation period and are a great source of nutrition, making them the optimal choice for our experiment.

The ability to grow fresh, safe, and nutritious food in space for astronauts has been an important goal for many space exploration agencies, especially NASA. The Veggie system is a small, low-power plant growth chamber designed by Orbital Technologies Corporation (Now Sierra Nevada Corp, Madison, WI, United States) for growing different vegetables in space [3]. The first Veggie was launched to the International Space Station (ISS) back in 2014, along with eighteen different plant (rooting) pillows for the experiments. The Veggie plant growth system consists of adjustable LED lights for producing a different spectrum of light and fans which circulate ISS air through plants. An extensible and transparent bellow linked with a light unit via magnets holds the plants and debris in place. Moreover, the air is directed from the bottom of the canopy to the top of the growing volume as shown in Figure 1. In different tests termed VEG-01 and VEG-03, plants were grown from the seeds in plant pillows. The Veggie base plate holds these plant pillows which interact with root mat water. A mixture of controlled-release fertilizer with calcined clay substrate and wicks helps in the attachment of seeds in plant pillows. The seeds after surface sanitization were glued into the plant pillows wicks and pillows are arranged for flight under sterile air [4].

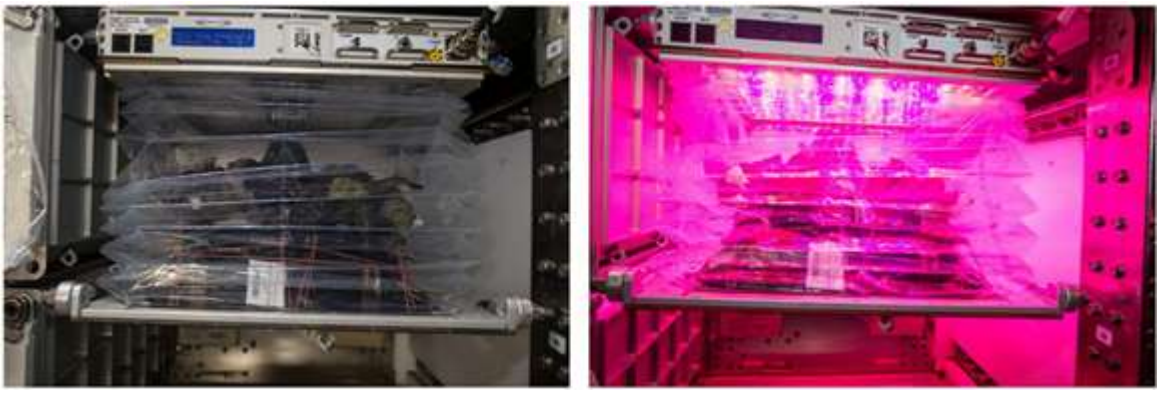


Figure 1: Veggie payload containing a crop of mature lettuce in Veggie Pillows aboard the ISS with the red-light panel on (Right) and light panel off (left)

The Advanced Plant Habitat (APH) is an improved version of Veggie for plant research in space. Similar to the Veggie, it utilizes LED lights with controlled-release fertilizer and clay substrate to provide nutrients, oxygen, and water to the plant roots. Unlike Veggie however, this module consists of 180 sensors and automated cameras for constant monitoring from the ground team [5]. Its atmospheric makeup, moisture levels, water recovery and distribution, and temperature are all adjusted automatically according to the requirements. Moreover, the APH consists of a wider wavelength range LEDs than the Veggie which includes green, blue, red, white, and infrared for night vision. Arabidopsis thaliana and dwarf wheat were grown on APH for the first time in spring 2018 on the International Space Station (ISS). When the crop is ready, crew members collect the samples and chemically preserve them to send them back for scientific studies [6].

Figure 2: The first growth test corps (Arabidopsis seeds and dwarf wheat) in Advanced Plant Habitat (APH) aboard the International Space Station

The Biological Research in Canisters (BRIC) is a facility developed by scientists to analyze the effect of microgravity on organisms such as microbes and yeast. The BRIC facility consists of a suite having six unique hardware configurations. Small biological specimens being studied are placed on a culture chamber or Petri plate contained inside an aluminum cylinder [7]. The BRIC-LED is the latest version integrated with multiple LEDs to assist the biology of algae, mosses, cyanobacteria, and plants that require light to make their food. Presently, BRIC LED is undergoing several types of hardware validation tests to ensure that LEDs don't get too hot for the plants and other checks [8].

Some of the major problems that NASA has faced with the farming system in space is water distribution, roots being starved of oxygen, reduction of evaporation, leaf temperature, and lack of sunlight energy for photosynthesis. Watering plants in space is an arduous task because of the low gravity. The water moves differently all the time, and it creates a sphere figure in which it

has air bubbles inside. It is important to notice that the water may disperse in a growth media due to no gravity. The technique that is being used to supply the plants needs is through injection. Better gaseous monitoring is needed because plants are starving of oxygen. Having no light energy causes the capsule to be at a lower temperature than needed, making the leaves a lower temperature and unable to successfully evaporate wasted water.[9]

Using these plants' growing mechanism as inspiration, we will design a volumetrically conscious chamber with multiple sensors and a centrally located LED for growing the micro red-radishes in space. As compared to the above-mentioned plant growing mechanism, our design utilizes more volumetric space which greatly enhances the production capabilities. The rest of the report is organized as follows: Project scope, conceptual design, and project plan.

Problem Definition and Objectives

Manned missions into the depths of space beyond the moon are on the cusp of human achievement but are hindered by the inability to create food on demand for the long expedition. Currently all food must be prepared and stored beforehand, increasing cost and reducing nutritional value the longer the mission. The objective of this project is to provide an efficient indoor gardening system for multi-gravitational environments to help alleviate this problem. This proposal will take into account two top space travel priorities; primary volume and crew time. Primary volume will be optimized by using an unorthodox structural configuration while crew time will be reduced via automation systems.

Concept of Operation

Assuming a Level 3 structure has been designed and tested, operation would begin by manual placement of seeds inside the structure. A resource profile of the plant being grown is selected/uploaded to the microcontroller which will adjust lighting, airflow, rotation, and water distribution rate. The system will operate autonomously for extended periods of time (days) with minimal to no intervention outside of periodic resource supplementation. During this autonomous period, data is regularly captured, parsed, and stored for later evaluation. Images will also be taken during this time to monitor plant growth and determine harvest time. At the time of harvest, usable plant matter is packaged for later consumption or scientific research.

System Level Requirements

Level 1: Grow

The garden shall be volumetrically conscious in shape (~0.125m^3) and be able to operate at a temperature within NASA specifications (18 to 30C +- 0.5C with 1C uniformity). An LED lighting arrangement will be used for photosynthesis. Fans capable of moving 300+ cfm will be positioned to regulate airflow/turbulence. A

pumped hydroponic based nutrient system will be able to create an ebb & flow system while rotating a chamber at ~250 RPM.

Level 2: Sense
The garden shall be able to monitor CO2/O2, temperature, and humidity. This data will be stored locally and displayed for the user. The garden shall be equipped with a camera system to monitor plant growth. The garden will regularly capture images with a reference point to be used to quantitatively monitor plant growth. The system will be in a roughly sealed environment to minimize pathogen contamination.

Level 3: Automate
A full automation system will control the light intensity/timing, airflow, and psychrometrics with appropriate adjustment of environmental conditions. An audible alarm system will be put in place if environmental conditions are outside specification. Fluid distribution will be a passive, adjustable nutrient delivery system. The garden should use lightweight materials to be weight competitive with enough structural integrity for surface habitat relocation.

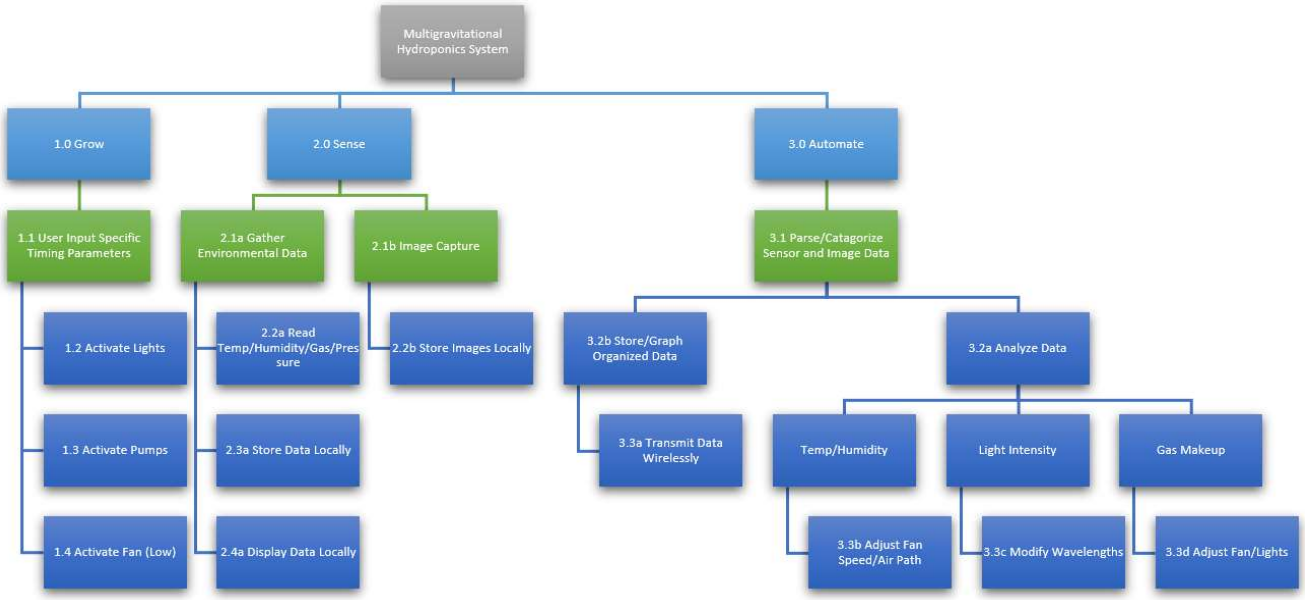
Success/Constraints

Success in this system will be judged by the continued ability to grow plants to maturity as milestones in Level 2 and Level 3 requirements are achieved. Because the concept of operation will take place in a unique environment, several constraints are assumed.
The garden system will be fixed to a rigid surface and be capable of operating in any orientation. Constant electrical power and water resources will be always available. The cabin systems during space flight should be designed to be able to handle the increased load in CO2 scrubbing, moisture capture, and any excess heat. All seeds are sterilized and prepackaged in a manner conducive to the current NASA method.

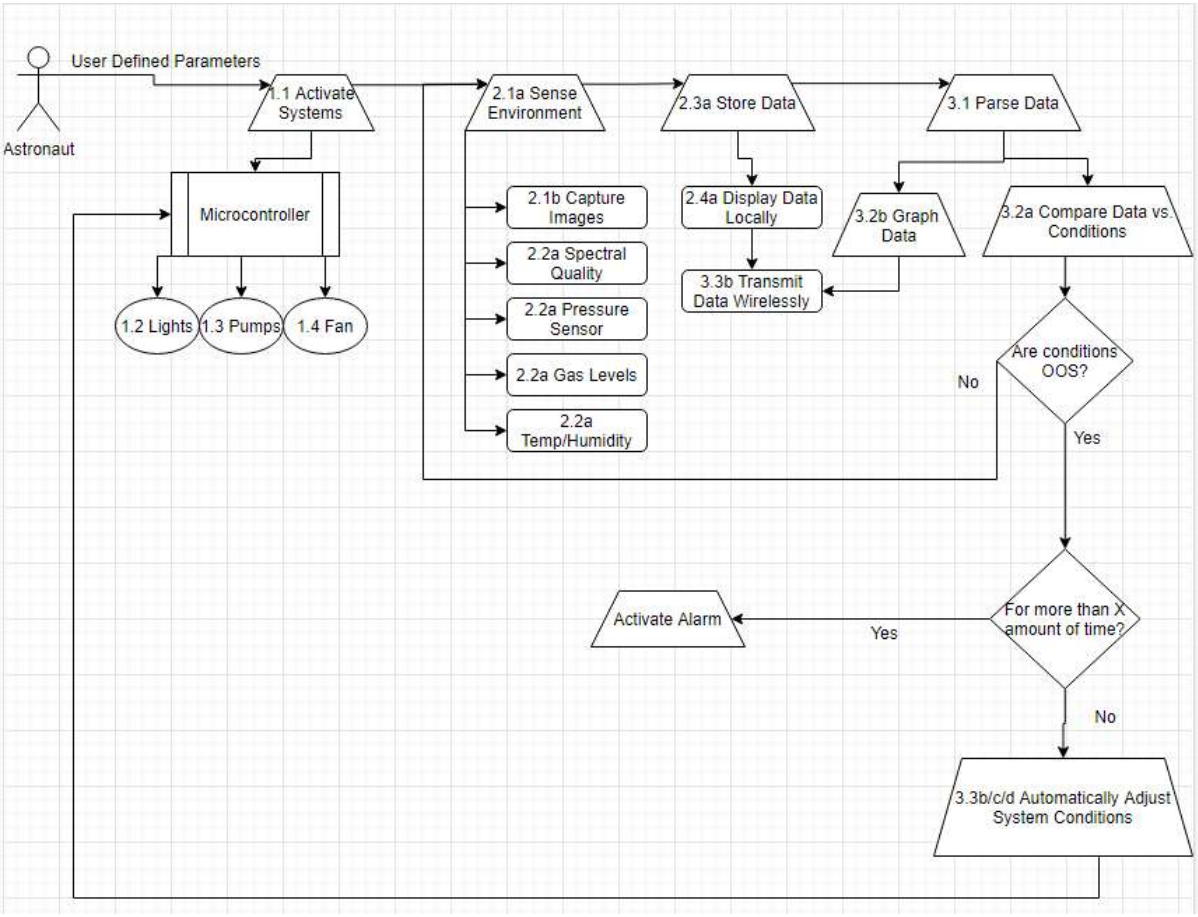
Functional Analysis/Hierarchy

The functional hierarchy operates under a similar profile to the system level requirements. Parent function 1.0 and children encompass the growth of a plant from seed to harvest without any automation or sensing systems. Achieving growth of microgreens is the foundation of the system and must be constant throughout all levels of the system. A resource profile will be manually entered by the astronaut to control lighting, fans, and pumps which will run continuously. Parent function 2.0 encompasses sensing broken into two sections; quantitative environmental sensing (2.1a) and image capture (2.1b). Either section requires its own scope of programming. Numerical data in section “a” will be displayed locally on the system. Both sections will store captured data (numerical or image) locally.

Parent function 3.0 revolves around automation and parsing of acquired data. After organizing the data into a more cohesive format, it will be graphed and transmitted wirelessly to another station for review. Testing will determine the conditions under which the system will automatically adjust lighting, fans, and pumps.



Functional Flow Diagram



Final Design Description

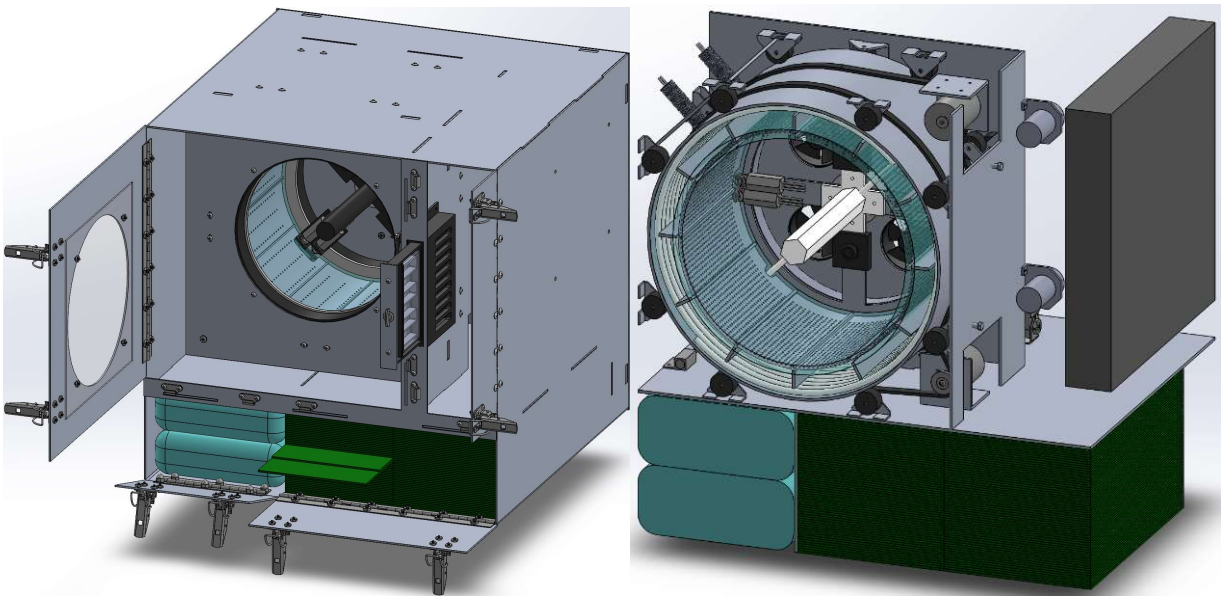


Figure 5. *Current standing centrifugal design portrayed in both housed (right) and exposed (left) form.*

The overall design houses a cylindrical growth chamber which utilizes a rotary mechanism in order to provide a centrifugal force over the water within the root zone of the system. This design uses baffles along the outer edge of the cylindrical housing and requires continuous operation of the rotatory systems to maintain water stability along the circumferential area of the cylinder.

This design not only maximizes the growth area through the use of the cylindrical housing, but also optimizes for durability through the implementation of several redundancies throughout the system. This includes the use of back up mechanical devices such as a set of peristaltic pumps and DC motors in order to ensure the longevity of the system. Currently, this system also provides the most consideration towards internal storage, being able to provide storage of microgreen seeds that can accommodate 60 growth cycles of the system.

Overall System

From the trade study, the design selected as most viable for the NASA indoor farming project is the centrifugal system. This system is most optimized to meet the recommended design criteria requested from NASA and provides a design that not only maximizes primary volume through the effect of producing an overwhelmingly higher yield of biomass when compared to the other two designs, but also requires a total crew time that is lower than the average crew time of the three designs as well.

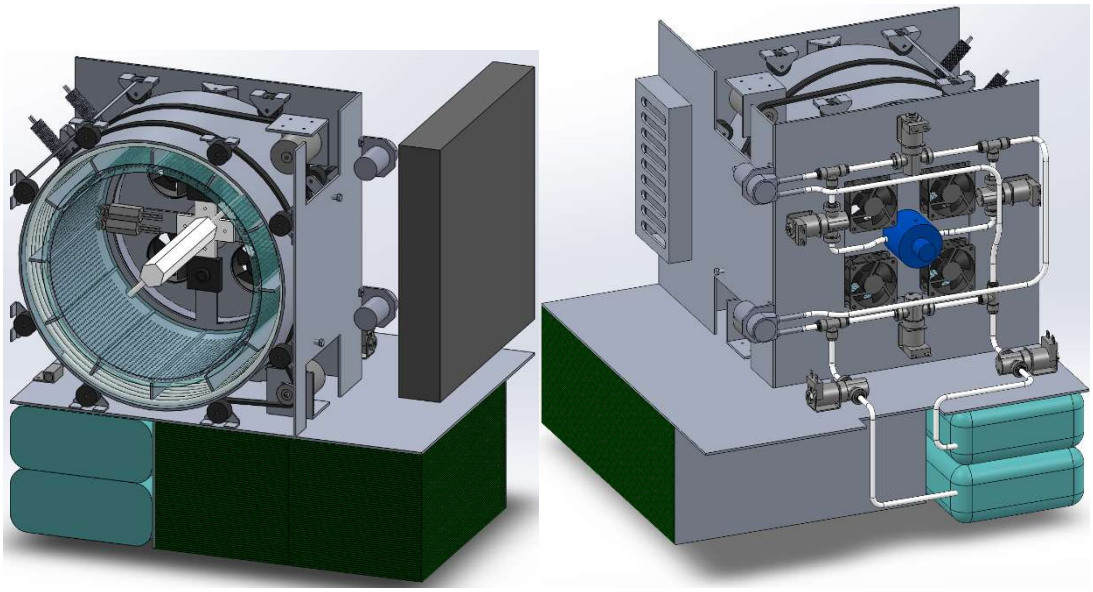


Figure 8. *Current standing centrifugal design, portrayed in both exposed-front (right) and exposed-rear (left) view.*

In this design, water is delivered through a set of peristaltic pumps and a system of solenoid bypass valves. Water and electricity passes into the growth chamber through a slip ring capable of handling constant rotation. Flexible water bladders are used for water storage. Reusable flexible aluminum screen are curled into place giving a stable base for plants to grow. Edible paper sheets with seeds attached are placed over the screen when planting.

Sub-systems

This design can be divided into 4 main subsystems. These subsystems provide the process by which a suitable environment, capable of promoting plant growth, can be obtained. The 4 systems within this design are the water delivery system, the lighting control system, the air quality control system, and the plant chamber rotary system. Each of these subsystems perform their duties parallel to one another and operate autonomously through the use of a Raspberry Pi microcontroller.

Water Delivery system

The water delivery system consists of a flexible collapsible water reservoir, polyethylene tubing with appropriate fittings, a set of solenoid valves, a set of water level laser sensors, a water pressure sensor, a set of peristaltic pumps, and a porous tube which dispenses and removes the water in and out of the rotating chamber. Currently, this system is designed to operate by cycling the water level within the root chamber between two levels during the growth cycle of the plant. The two levels being a soaking level for germination and a standard level for growth. 2 pairs of water level laser sensors, which are placed at varying radial positions within the chamber, are used to determine the water levels within the system and will help the microcontroller cycle between these desired water levels.

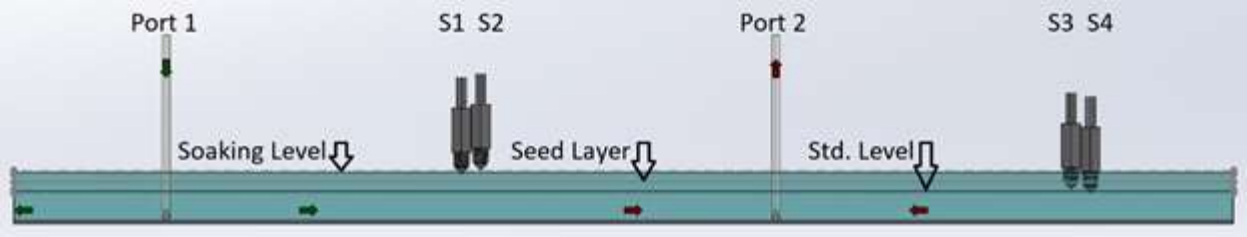


Figure 9. *Visual representation of varying water levels required during operational performance, with accompanying water level laser sensors and inlet and exit ports. .*

Lighting control system

The lighting control system consists of a 12 V grow light and a set of light intensity sensors. This system uses a centralized light source in order to provide light energy evenly to the plants. Plants predominantly use light levels within the 400 nm – 700 nm wavelength during photosynthesis so providing these wavelengths of light are critical in plant development. The system operates based on pre-programmed levels that can vary throughout the growth cycle of the plant. The pre-programmed levels will include parameters for brightness and timing intervals for the light within.

Figure 10. *A list of the colors present in the visible light spectrum and their effects on plant growth.*

Air quality control system

The air quality control system consists of a set of air circulating fans, an air heater with a fan component, and a set of air quality sensors including sensors for temperature, humidity, O2, and CO2. The fans within the system are currently how several properties within the growth chamber are maintained. By circulating air into and out of the system and filtering out gaseous content

within the growth chamber a neutral environment can be provided for the plants to grow in. Although this is currently how air quality is being handled, in future iterations, chances for more precise air quality handling can be pursued. The fans operate at pre-programmed levels and will be adjusted to ensure proper quality of air for the plants.

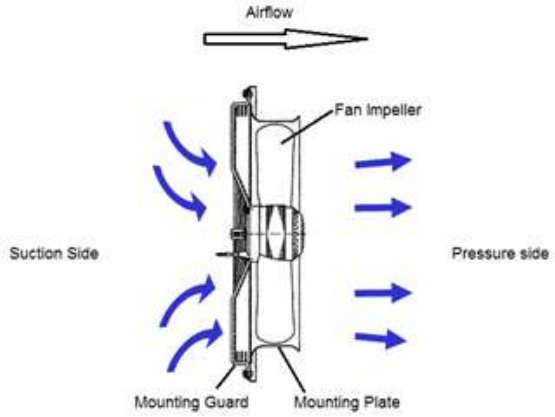


Figure 11. *Basic representation of axial flow air compressor during operation.*

Growth Chamber Rotary System

The Growth Chamber Rotary system consists of a 12V motor with low profile caster wheels, a 48’ drive belt, and belt tensors acting as auxiliary components for the design. In this system the motor would be required to operate continuously throughout the growth cycle of the vegetation in order to provide the necessary centripetal forces in order to produce an artificial gravity. The torque required for the motor will be subjected to varying loads throughout the operational life of the system due to the changing moment of inertia of the system as the water levels are cycled between their highs and lows.

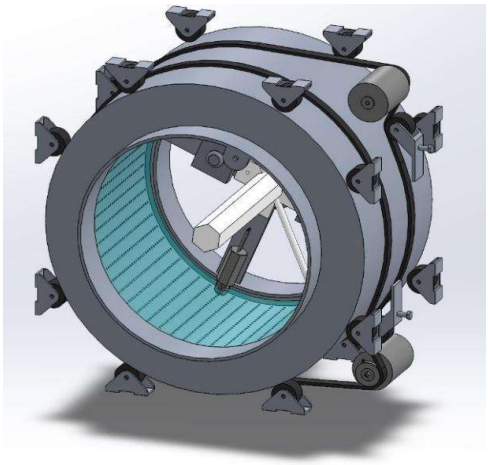
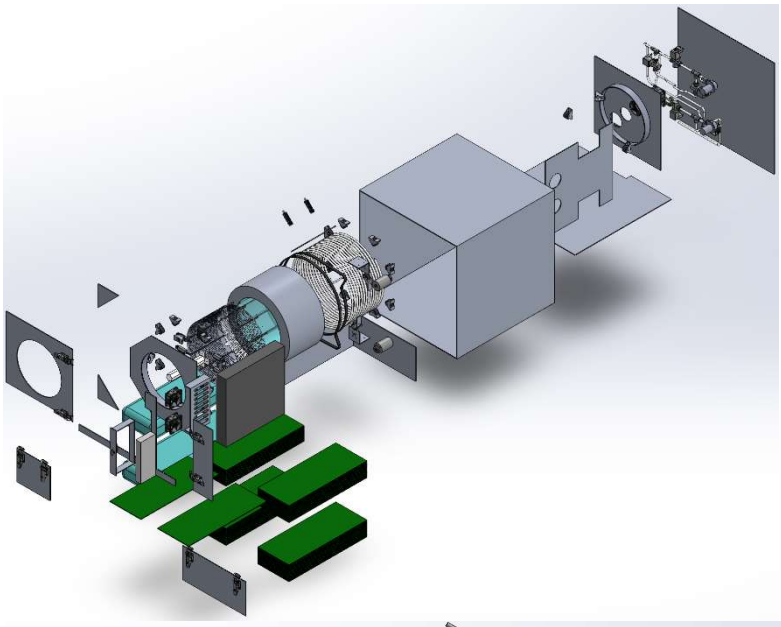
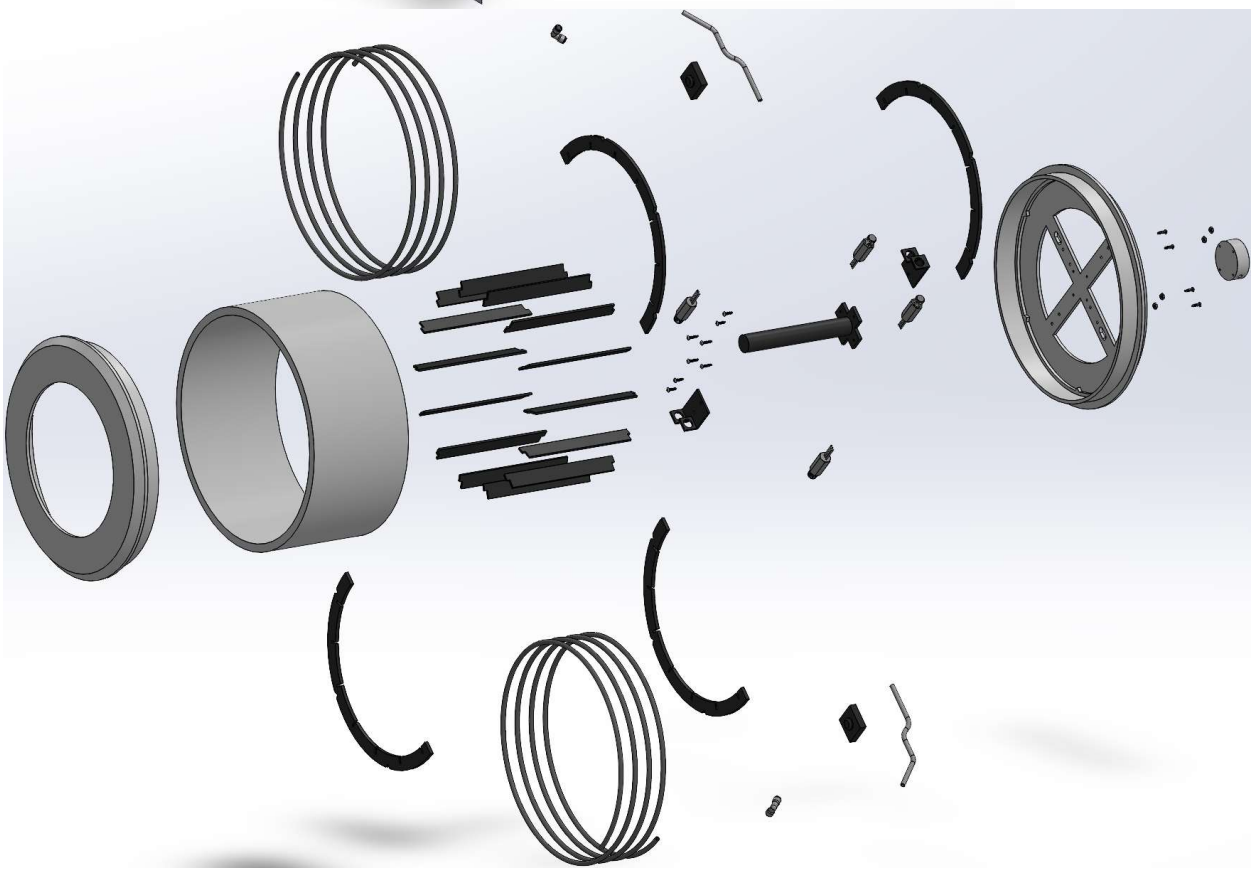


Figure 12. *Exposed growth chamber housing with accompanying caster wheels.*

Product Breakdown





Management Plan

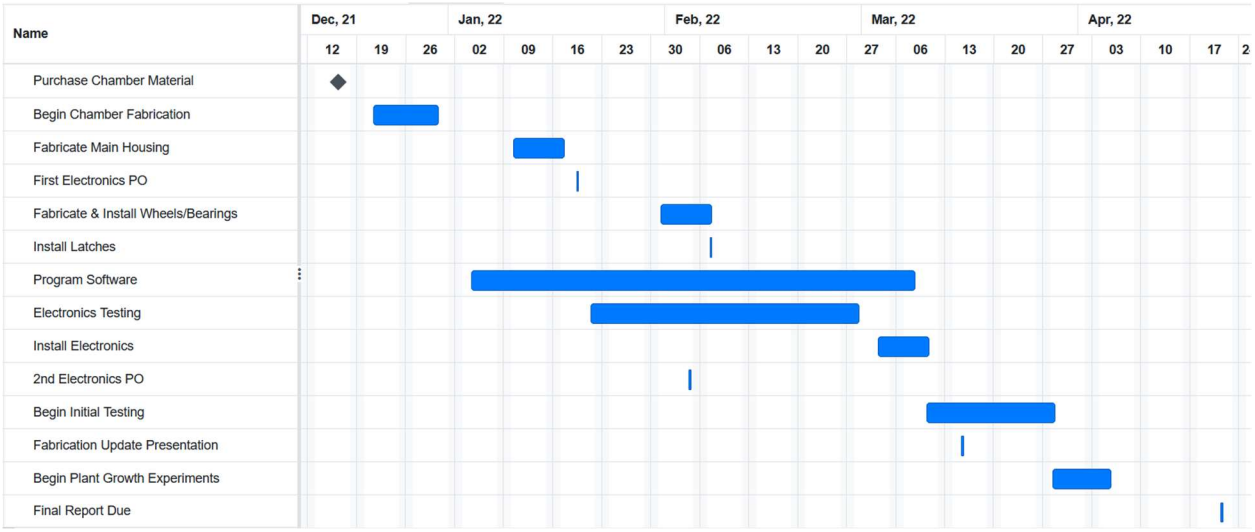
- Christopher Cadenas: Research of botanicals and advising the proper growing procedure/methodology that may influence design. Assisting electrical assembly and 3D printing.

- Vitas Diktanas: Team Lead and head mechanical designer utilizing SolidWorks and performing all necessary assemblies. Director of physical fabrication and assembly.

- Gregory Henry: Focus on air circulation and filtration in the system in order to control the O2/CO2 and micro gas concentrations. Assisting fabrication.

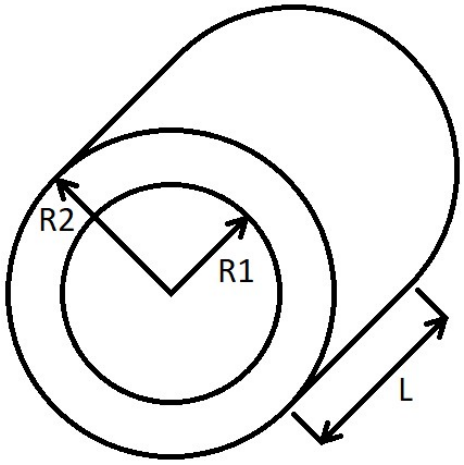
- Kevin Munoz: Focus on the required lighting and optical sensors in the system that will optimize plant growth and minimize energy consumption. Budget coordinator. Assisting fabrication.

- Omar Odeh: Communication Director and lead electrical circuit design and microcontroller programming which will automate the system. Software programmer and implementation of all electronics.



Design and Engineering (Mechanical)

The design dimensions and mass are based on desired growth area.



Minimum desired growth area: $A_{min} = 0.125\ m^2$

Inner radius of water based on plant height: $R_1 = 0.127\ m$

Outer radius of water based on root depth: $R_2 = 0.157\ m$

$$L_{min} = \frac{A_{min}}{2\pi * R_1}$$

The minimum length required to meet area requirement: $L_{min} = 0.1566\ m$

Added length for instruments: $L_{equip} = 0.017\ m$

$$L = L_{min} + L_{equip}$$

Total length of growth chamber: $L = 0.174\ m$

$$V = \pi * L * \left(R_2^2 - R_1^2\right)$$

Total volume of water in growth chamber: $V = 4.657\ x\ 10^{-3}\ m^3$

$$M_{water} = V * 1000$$

Mass of water in growth chamber: $M_{water} = 4.657\ Kg$

Mass of aluminum drum based on SOLIDWORKS analysis: $M_{drum} = 2.393\ Kg$

$$M_{tota} = M_{water} + M_{drum}$$

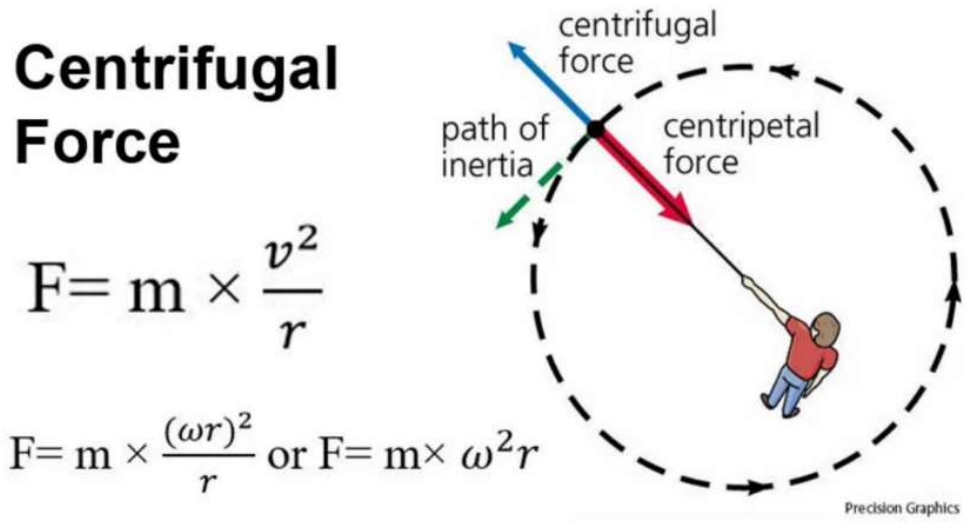
Total mass of growth chamber and water combined: $M_{total} = 7.05\ Kg$

Factor of safety for mass in calculations: $F.S._m = 1.5$

$$M_{eq} = M_{total} * F.S._m$$

Equivalent mass used in calculations: $M_{eq} = 10.6\ Kg$

The rotation required to create sufficient centrifugal force to counteract gravitational force is determined through Newton's 2nd Law of Motion.



Gravitational constant on Earth: $g = 9.81\ m/s^2$

$$F_{mass} = M_{eq} * g$$

Force due to mass: $F_{mass} = 103.6\ N$

$$F_{centr} = M_{eq} * \omega_{min}^2 * R_1$$

The centrifugal force, F_{centr}, must be greater than the force due to mass: $F_{centr} \geq F_{mass}$

$$\omega_{min} \geq \sqrt{\frac{g}{R_1}}$$

Minimum angular velocity to maintain centrifugal force equal to Earth gravity: $\omega_{min} \geq 8.79\ rad/s$

$$f_{min} = \frac{\omega_{min} * 60}{2\pi}$$

Minimum frequency to maintain centrifugal force equal to Earth gravity: $f_{min} = 83.9\ rev/min$

Frequency to maintain water distribution while in earth gravity found from tests: $f_t = 250\ rev/min$

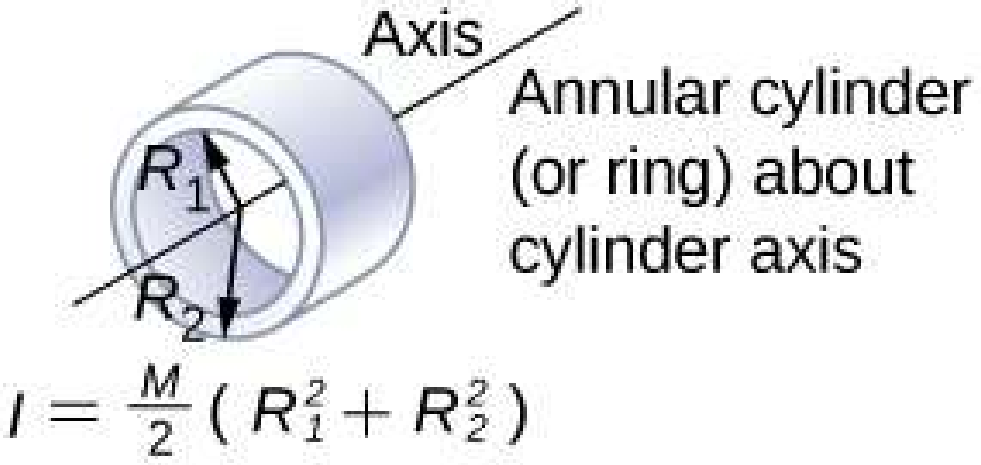
$$\omega_{eq} = \frac{2\pi * f_t}{60}$$

Equivalent angular velocity used in calculations: $\omega_{eq} = 26.18\ rad/s$

$$F_{eq} = M_{eq} * \omega_{eq}^2 * R_1$$

Equivalent centrifugal force used in calculations: $F_{eq} = 919.31\ N$

The torque required on the motor to reach desired rotation in a specified time is found using mass moment of inertia and gear ratios.



$$I = \frac{M_{eq} * \left(R_1^2 + R_2^2\right)}{2}$$

Mass moment of inertia for rotating growth chamber: $I = 0.2153\ m^4$

Estimated time to reach rotational requirements from stationary: $t = 1\ sec$

$$\alpha = \frac{\omega_{eq}}{t}$$

Angular acceleration: $\alpha = 26.18\ rad/s^2$

$$\tau_m = I * \alpha$$

Torque due to mass required to reach rotational requirements from stationary: $\tau_m = 5.64\ N * m$

Factor of safety for torque of rollers in calculations: $F.S._{\tau} = 1.5$

$$\tau_{meq} = \tau_m * F.S._{\tau}$$

Equivalent torque due to mass used in calculations: $\tau_{meq} = 8.46\ N * m$

Number of rollers in contact with rotating growth chamber: $N_r = 24$

Coefficient of friction for rollers: $\mu = 0.005$

Diameter of rollers in contact with rotating growth chamber: $d_r = 0.025\ m$

$$\tau_r = \frac{N_r * F_{eq} * \mu * d_r}{2}$$

Torque due to friction from rollers on growth chamber: $\tau_r = 1.38\ N * m$

$$\tau_{req} = \tau_r * F.S._{\tau}$$

Equivalent torque due to friction for rollers on growth chamber used in calculations: $\tau_{req} = 2.07\ N * m$

$$\tau_{rhp} = \frac{\tau_{req} * f_t}{7127}$$

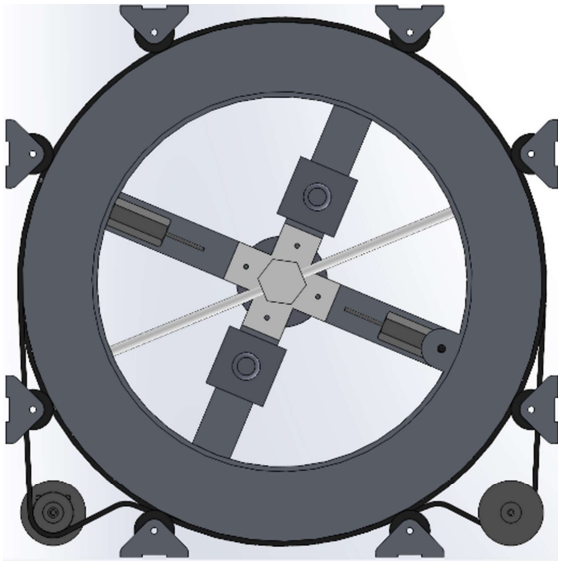
Torque due to friction from rollers on growth chamber in horsepower: $\tau_{rhp} = 0.0726\ hp$

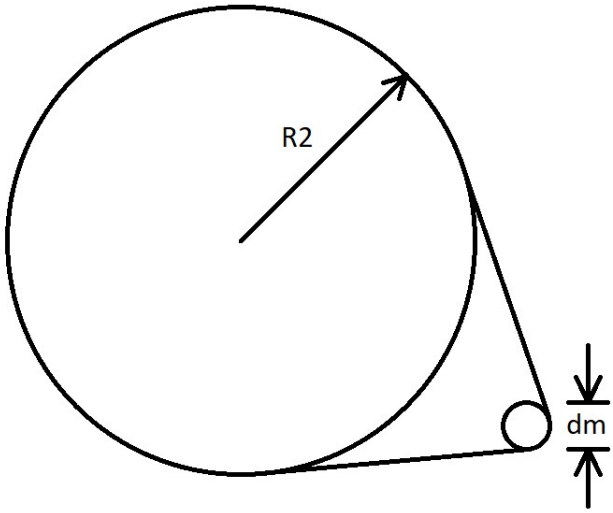
$$\tau_t = \tau_{meq} * \tau_{req}$$

Total torque on growth chamber required for startup: $\tau_t = 10.52\ N * m$

$$\tau_{thp} = \frac{\tau_t * f_t}{7127}$$

Total torque on growth chamber required for startup in horsepower: $\tau_{thp} = 0.3682\ hp$





The diameter of the drive wheel on the motor: $d_m = 0.03125\ m$

$$G_r = \frac{R_2 * 2}{d_m}$$

The gear ratio between the growth chamber and the drive motor wheel: $G_r = 10.05$

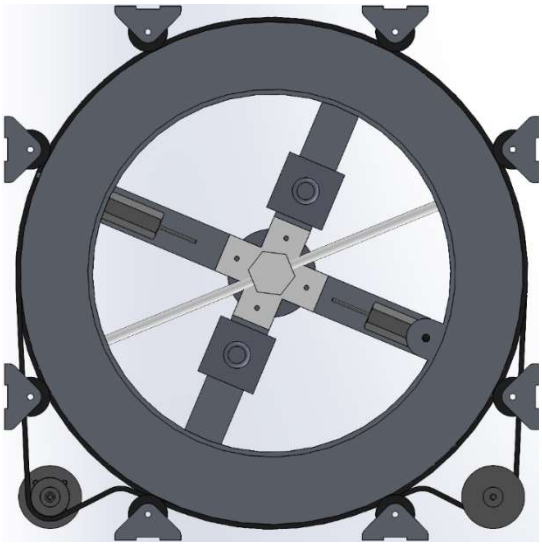
$$\tau_m = \frac{\tau_{thp}}{G_r}$$

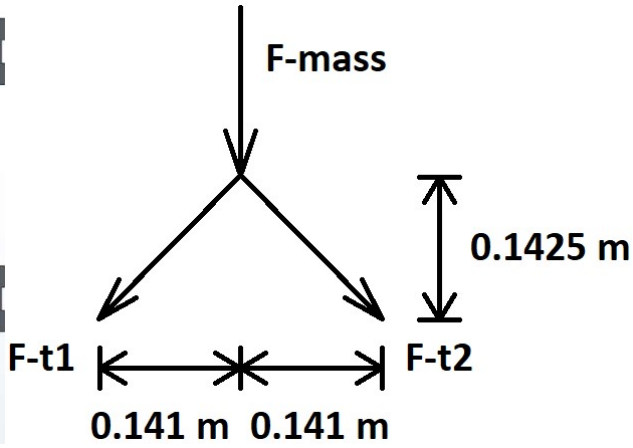
Total torque on the drive motor required for startup in horsepower: $\tau_m = 0.0367\ hp$

$$f_m = f_t * G_r$$

Frequency of drive motor at maximum rotation: $f_m = 2512\ rev/min$

The maximum applied force is determined by adding the combined tension from both drive belts in the direction of motors.





Assumed tension applied to each drive belt: $F_T = 25\ N$

$$\theta_T = tan^{-1}\left(\frac{0.1425}{0.141}\right)$$

Angle to each drive motor from y axis: $\theta_T = 45.30°$

$$F_{TY1} = F_T * sin(\theta_T)$$

Force of tension from drive belt 1 in the y direction: $F_{TY1} = 17.77\ N\ \downarrow$

By symmetry: $F_{TY1} = F_{TY2}$

Force of tension from drive belt 2 in the y direction: $F_{TY2} = 17.77\ N\ \downarrow$

$$\sum F_{TY} = F_{TY1} + F_{TY2}$$

Sum of forces due to tension in the y direction: $\sum F_{TY} = 35.54\ N\downarrow$

$$F_{TX1} = F_T * cos(\theta_T)$$

Force of tension from drive belt 1 in the x direction: $F_{TX1} = 17.58\ N\ \leftarrow$

By symmetry: $F_{TX1} = -F_{TX2}$

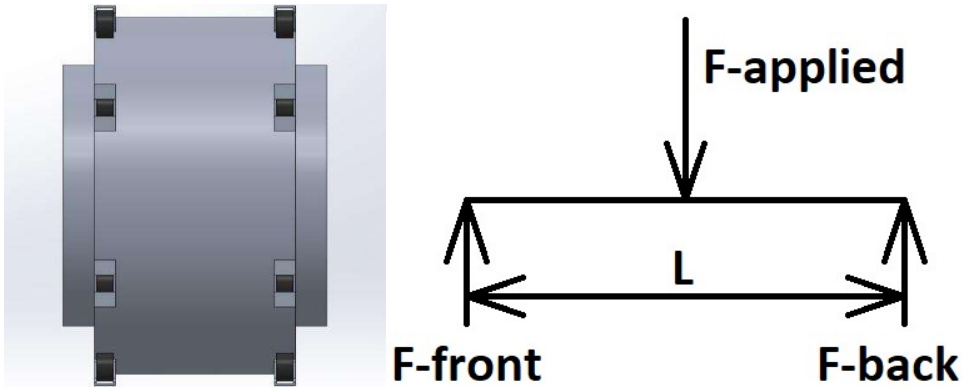
Force of tension from drive belt 2 in the x direction: $F_{TX2} = 17.58\ N\ \rightarrow$

Sum of forces due to tension in the x direction: $\sum F_{TX} = 0$

$$F_{applied} = F_{mass} + \sum F_{TY} + \sum F_{TX}$$

Sum of applied forces in the y direction: $F_{applied} = 139.15\ N\ \downarrow$

The force on the front and back rollers is determined by using symmetry.

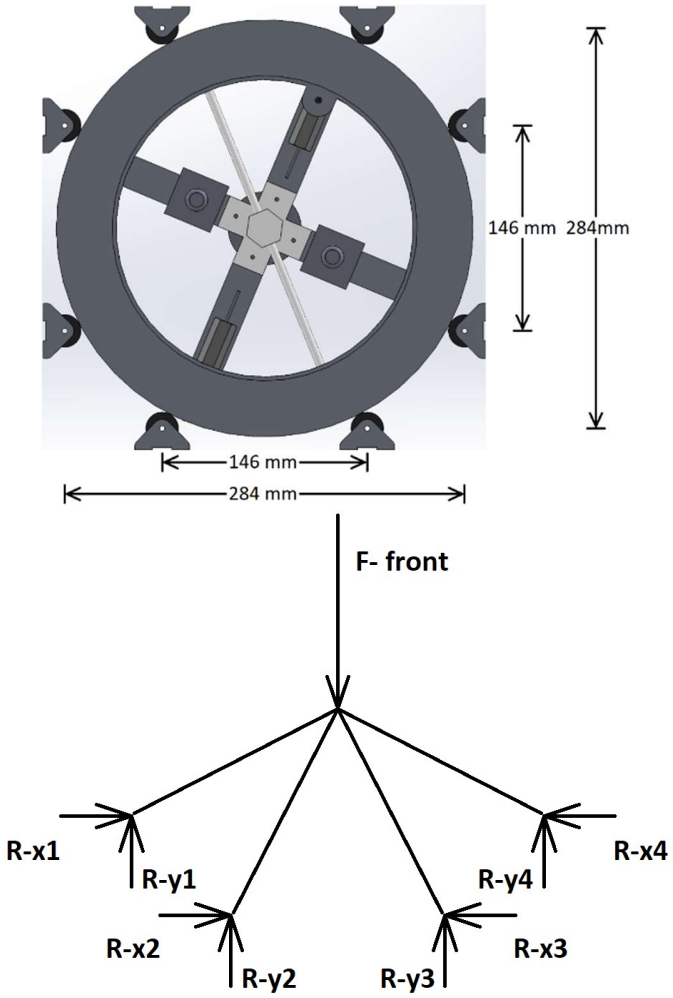


$$F_{front} = \frac{F_{applied}}{2}$$

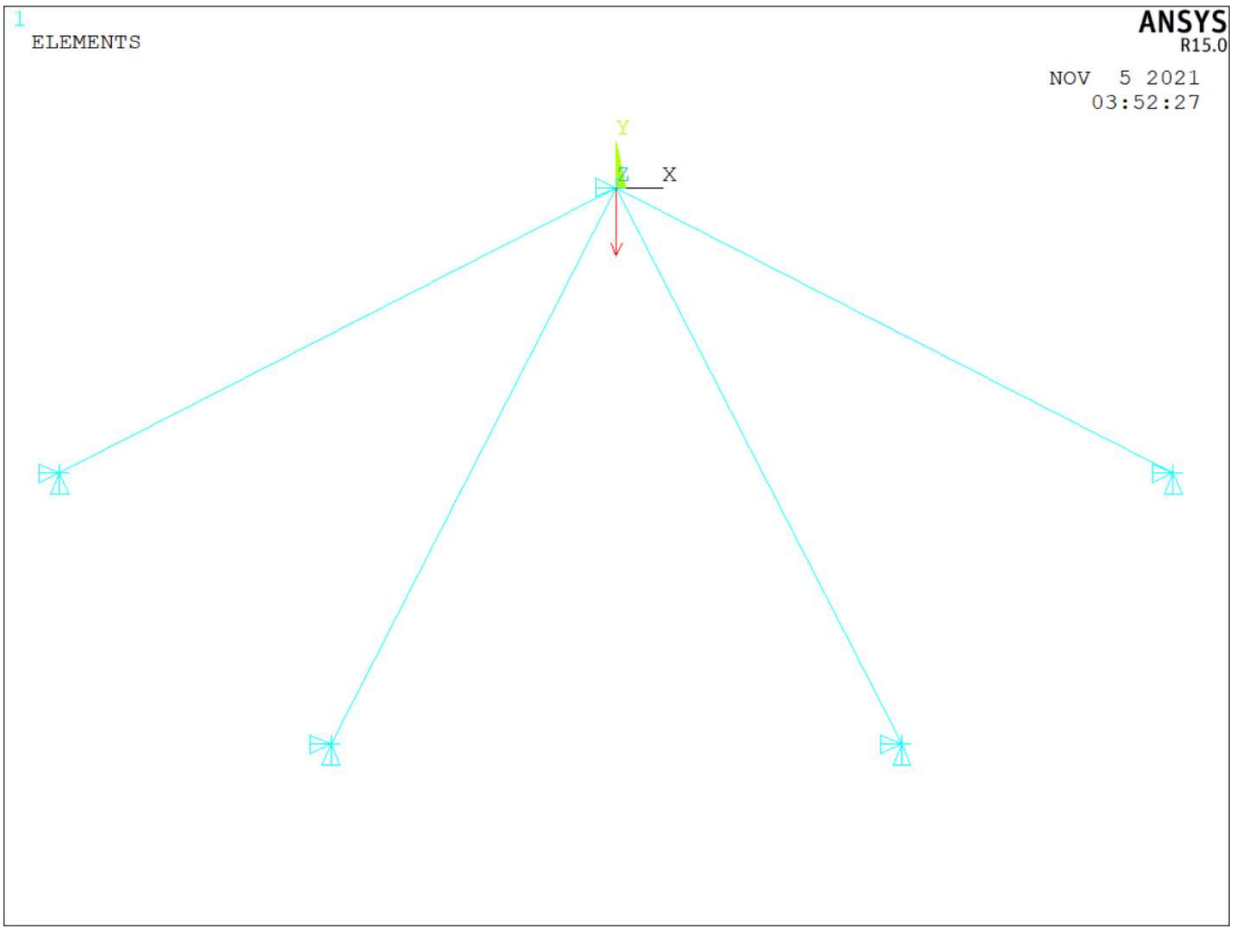
Force along front: $F_{front} = 69.6\ N$

By symmetry: $F_{front} = F_{back}$

Force along back: $F_{back} = 69.6\ N$



The force on each individual roller is determined using Ansys.



LIST SELECTED POINT LOADS ON ALL SELECTED KEYPOINTS
CURRENTLY SELECTED LOAD SET= FX FY FZ

KEYPOINT	LOAD LAB	VALUE(S)
5	FY	-69.575 0.0000

PRINT REACTION SOLUTIONS PER NODE

***** POST1 TOTAL REACTION SOLUTION LISTING *****

LOAD STEP= 1 SUBSTEP= 1
TIME= 1.0000 LOAD CASE= 0

THE FOLLOWING X,Y,Z SOLUTIONS ARE IN THE GLOBAL COORDINATE SYSTEM

NODE	FX	FY	FZ
1	0.0000		0.0000
2	14.123	7.2412	0.0000
3	14.123	27.546	0.0000
4	-14.123	27.546	0.0000
5	-14.123	7.2412	0.0000

TOTAL VALUES
VALUE 0.17764E-14 69.575 0.0000

The force on each side of the structure is determined by summing the reaction forces for front and back rollers found with Ansys.

$$\sum \quad R_Y = (R_{Y1} + R_{Y2} + R_{Y3} + R_{Y4}) * 2$$

Sum of reaction forces in the y direction: $\sum \quad R_Y = 139.15\ N\ \uparrow$

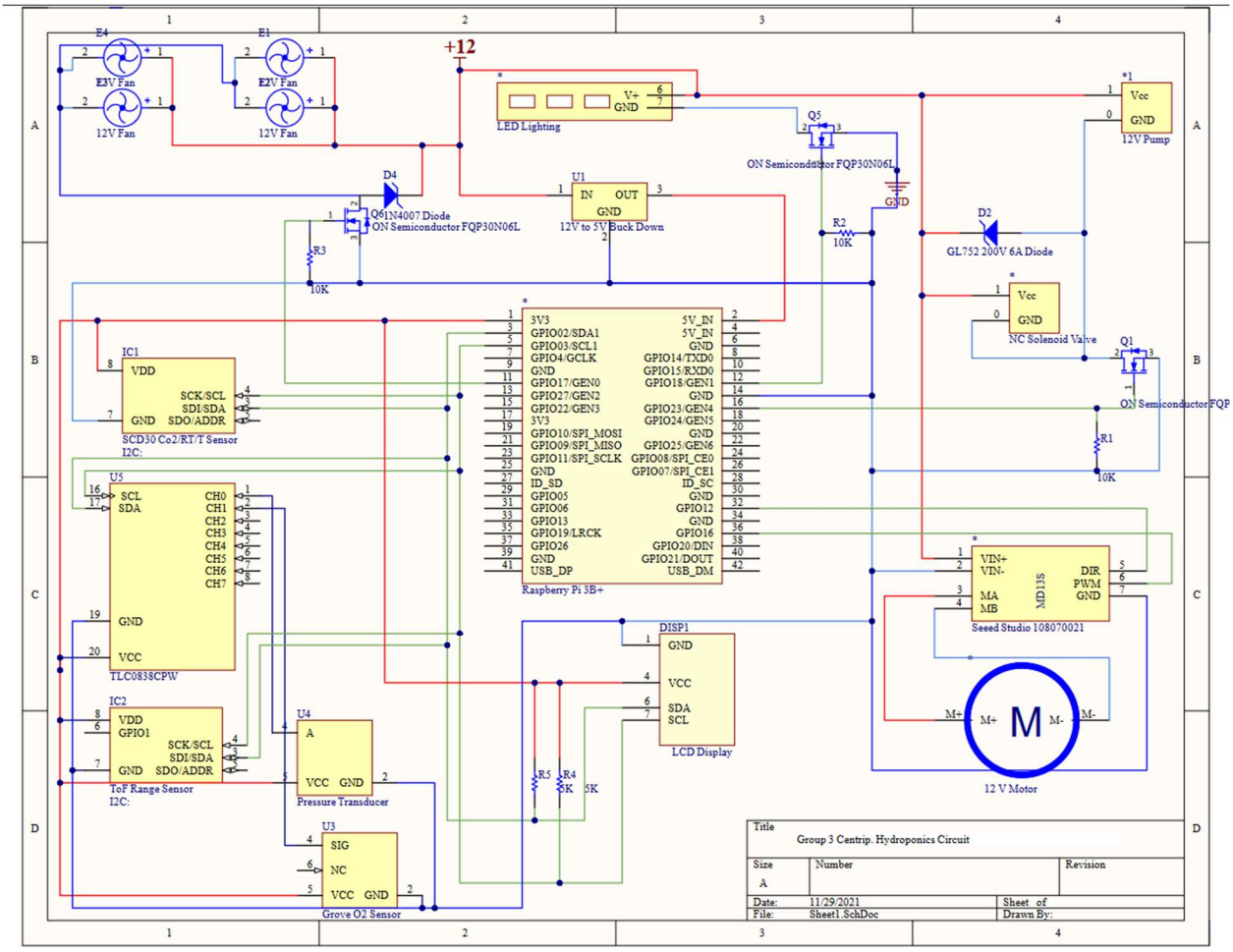
$$\sum \quad R_{X-left} = (R_{X1} + R_{X2}) * 2$$

Sum of reaction forces in the x direction along the left side: $\sum \quad R_{X-lef} = 56.49\ N\ \rightarrow$

$$\sum \quad R_{X-right} = (R_{X3} + R_{X4}) * 2$$

Sum of reaction forces in the x direction along the right side: $\sum \quad R_{X-right} = 56.49\ \leftarrow$

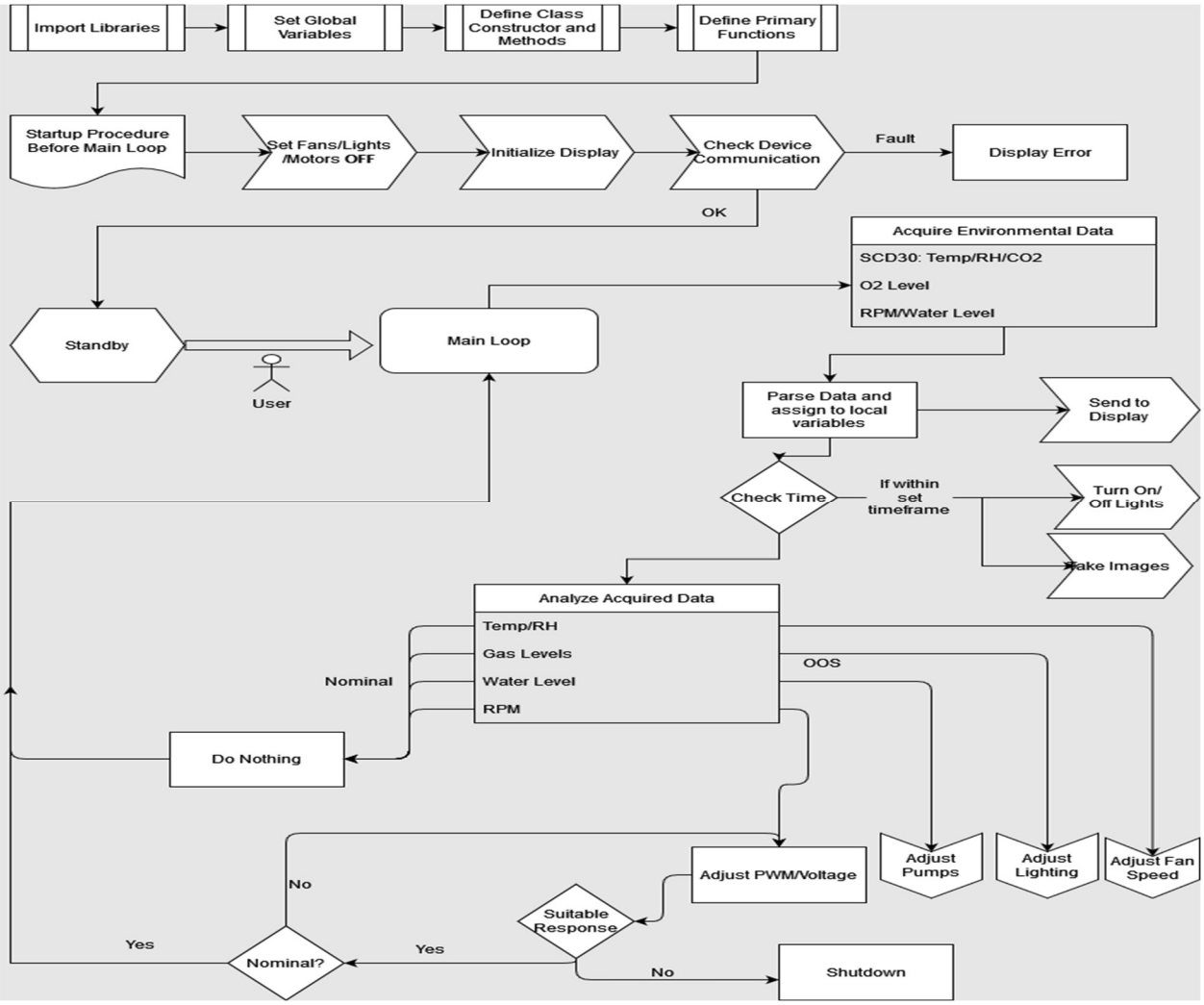
Design and Engineering (Electrical)



A Raspberry Pi 3B+ is used for all logic control. The GPIO pins provide the logic voltage to switch on MOSFETs for the fans, motors, pumps, and solenoid valves. Pull-down resistors are added to ensure proper shutoff of the MOSFET switch. Diodes are added to any inductive load in order to prevent inductive flyback and over voltages which may damage the circuitry. The low powered pump and fans can use a lower end 1N4007 flyback diode. However, the motor which will provide a much higher inductive load will require a diode with higher voltage capabilities. To alleviate the complications of higher power motor inductance, a Cytron Motor driver with equipped PWM capabilities is used.

Sensors connected to the Raspberry Pi operate either through I2C communication or via an A/D converter. The SDA and SCL lines have pull-up resistors to ensure proper communication. A 12V/30A power supply will provide enough power for the system. A buck down converter will power the Raspberry Pi from the 12V power supply. The coding is done in the Python language.

Program Algorithm Flow



Setup Procedure

- Import Libraries
- Define Variables/Classes/Methods
- Define Functions not contained in a Class

Startup Procedure

- Set Everything OFF
- Turn on Display
- Check Device Communication
- Enter Standby Menu

Main Loop

- Get Environmental Data

- Parse and Send Data to Display
- Check Timing Parameters
- Adjust System based on acquired data
- Repeat Main Loop

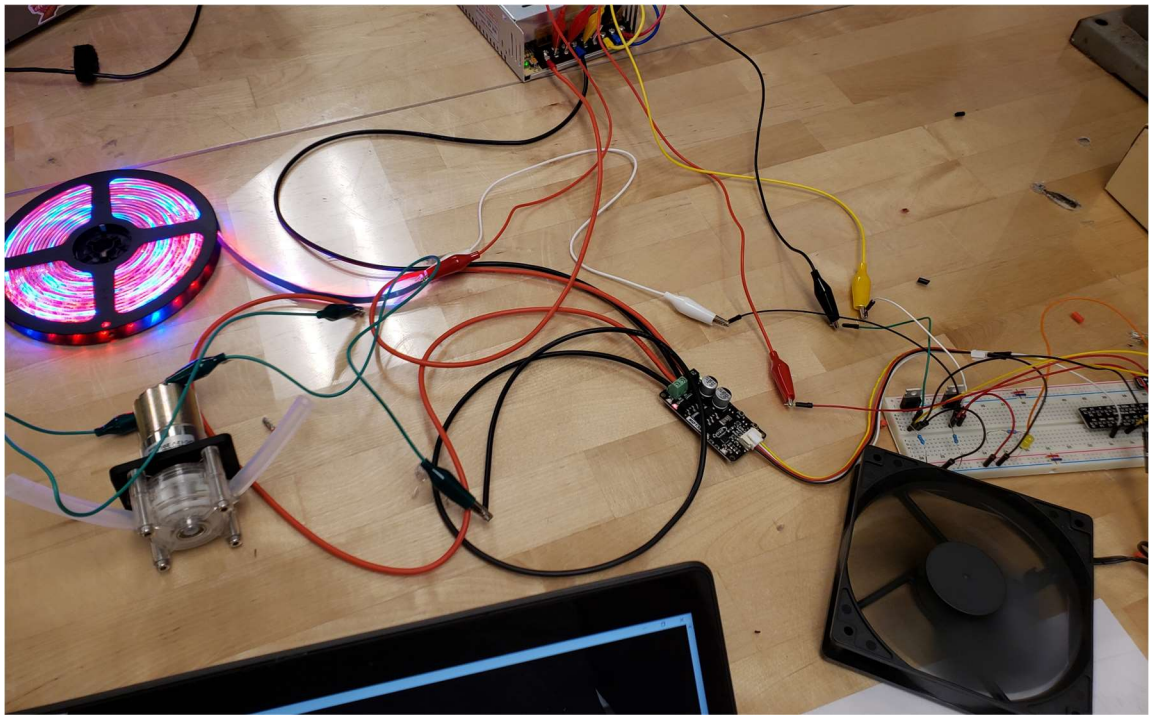
Electrical Software

Python was chosen as the software language due to its versatility in controlling I/O pins in conjunction with data processing for images and data. The program loop itself is a single while loop which runs a main menu function, propelling to other functions within the program as needed. Individual control over system functions allows for optimal testing of electronic components and measuring response.

Due to the high motor speed, extra care had to be considered in programming in order to prevent any sudden start/stops from the system. Such an extreme deceleration or acceleration could potentially damage the electronics and physical components of the system. To overcome this, all motor commands funneled through a ramping function, either accelerating or decelerating at 1% duty cycle per second until the desired duty cycle was achieved. If the program is exited during operation, a series of exit protocols including ramping down the motor and resetting all pins to a default state was integrated.

Ideally, the system will constantly acquire data in the background and respond accordingly while the user navigates the main menu simultaneously. Threading these two aspects into the programming is a more advanced task which was unfortunately not accomplished at the completion of this report.

Electrical Hardware



The hardware is controlled by wirelessly connecting to the Raspberry Pi via SSH and executing commands within a main menu. The motor controller is able to adjust PWM duty cycle and frequency which is essential in controlling rotational speed. Motor PWM is accomplished via software programming. The PWM pin is nonetheless connected to the designated hardware PWM pin on the Raspberry Pi to potentially use the more advanced option.

The LED lighting arrangement, fans, peristaltic pumps, and solenoid valves operate with FQP30N06L MOSFETs. The 3.3V logic on the Raspberry Pi requires this particular model and is capable of handling PWM signals as well.

Two sensors operate via I2C protocol; the SCD30 environmental sensor and OLED display. The minimum update period between measurements on the SCD30 is 3 seconds, limiting the update rate of environmental data to that value. The OLED display was decommissioned at the submission of this report. Original intentions were to mount the display on the housing so it can be viewed from the outside.

Two other sensors operate via an analog signal; the water level sensors and Grove O2 sensor. As the Raspberry Pi 3B+ is not equipped to handle an analog signal, an A/D converter HAT is used. Both sensors output an analog signal to the converter which is then used to convert a voltage value to the respective unit being measured.

System Construction and Assembly

Construction of the Growth Chamber

The main section of the growth chamber will be its outer wall. This is cut from the bell end of a sch40 PVC pressure pipe. This was cut and squared off the last week of the Fall 2021 semester.

Careful measurements were taken after the outer wall of the growth chamber was cut. Front and back end caps are designed to match diameters of both ends of the cut pipe. End caps are 3D printed using ABS filament.

Printed end caps are secured in place using solvent weld glue joints. Utilizing pre-existing methods for attaching ABS to PVC. Exposed printed surfaces that will come in contact with water are covered in a thin layer of epoxy to make a waterproof seal. This makes up the main body of the growth chamber.

After the main body of the growth chamber is completed, careful measurements were taken. Four removable supports were designed to fit two to each inside end of the growth chamber. These supports were printed out of ABS filament in addition to the baffles that line the inside circumference of the growth chamber wall. All surfaces of these supports and baffles are sealed with a thin layer of epoxy. 3D printing and construction of the growth chamber began over the winter break.

Constructing the Frame

Construction of the system begins with the outer walls. A single piece is cut from a full sheet of 0.090 in thickness 5055 aluminum using a water jet cutter. After cutting the piece is folded along the cut seams. The ends fit together with a tab and slot system. The corners are welded to secure the frame..

After the outer walls are assembled, careful measurements are taken. Inner walls, middle support, and front panel are then cut. These pieces fit into place with the tab and slot system. Opposite ends of tabs are then welded secure.

45 degree wall mount brackets for the back panel and growth chamber divider are cut from separate ¼ in thick aluminum. Threaded holes are tapped in the center of each bracket. These brackets fit into place using the tab and slot method. Tab ends are welded secure.

Once all welds are complete, excess welding material is smoothed out using angle grinders. After completion this will conclude all the welding requirements for this project.

Constructing the Access Panels

After construction of the frame is complete, careful measurements are taken once again. Measurements of hinges and latches are also taken into account. Front access panels are designed to accommodate clearances for existing structure, hinges and latches.

Front and back panels are then cut from the same 0.090 in thick 5055 aluminum as before using a water jet cutter. A clear access cover is cut from ⅛ in plexiglass using a vertical band saw. A reflective coating is also adhered to the inner side. Holes are then drilled in plexiglass using a drill press. Hinges, latches, panels, and access cover are secured into place using appropriately nuts, bolts, and washers.

Constructing Rollers

Twenty four pre-assembled roller wheels were ordered over the winter break. Once arrived they were carefully measured. Support brackets are designed to fit these wheels with a minimal clearance.

Support brackets are cut from the 0.090 in thick 5055 aluminum using a water jet cutter. Wheels are secured in place with M5 bolts and lock nuts. Assembled rollers are secured to the frame with appropriate nuts, bolts, and washers. These rollers and access panels have been constructed.

Installing Growth Chamber

Rollers are carefully measured for their clearances. Rollers along the walls of the growth chamber are secured into place using appropriate nuts, bolts, and washers. The rollers can be adjusted from outside along precut channels.

A growth chamber dividing wall is designed based on measurements taken of the frame. Motor mounts and slip ring brackets are based on measurements from their respective counterparts. Growth chamber dividing wall, motor mounts, and slip ring brackets are cut from the 0.090 in thick 5055 aluminum using the water jet cutter.

Spacers are designed to fill the gap at either end of the growth chamber based on the clearance of the rollers. The spacers were 3D printed from ABS.

Motors with attached drive wheels are attached to motor mount brackets before being installed inside the assembly. Motors, spacers, and rollers are installed in their designated positions using appropriate nuts, bolts, and washers.

Two drive belts are loosely fitted around the growth chamber as it is placed within the assembly. The growth chamber dividing wall is installed using predetermined bolts into the 45 degree brackets. The slip ring is then secured using the slip ring brackets with appropriate nuts, bolts, and washers.

Two belt tensioners were designed based on the type of drive belts chosen. The belt tensioners are cut from the remaining 0.090 in thick 5055 aluminum. The drive belts are then secured around the motor drive wheels and belt tensioners with appropriate nuts, bolts, and washers.

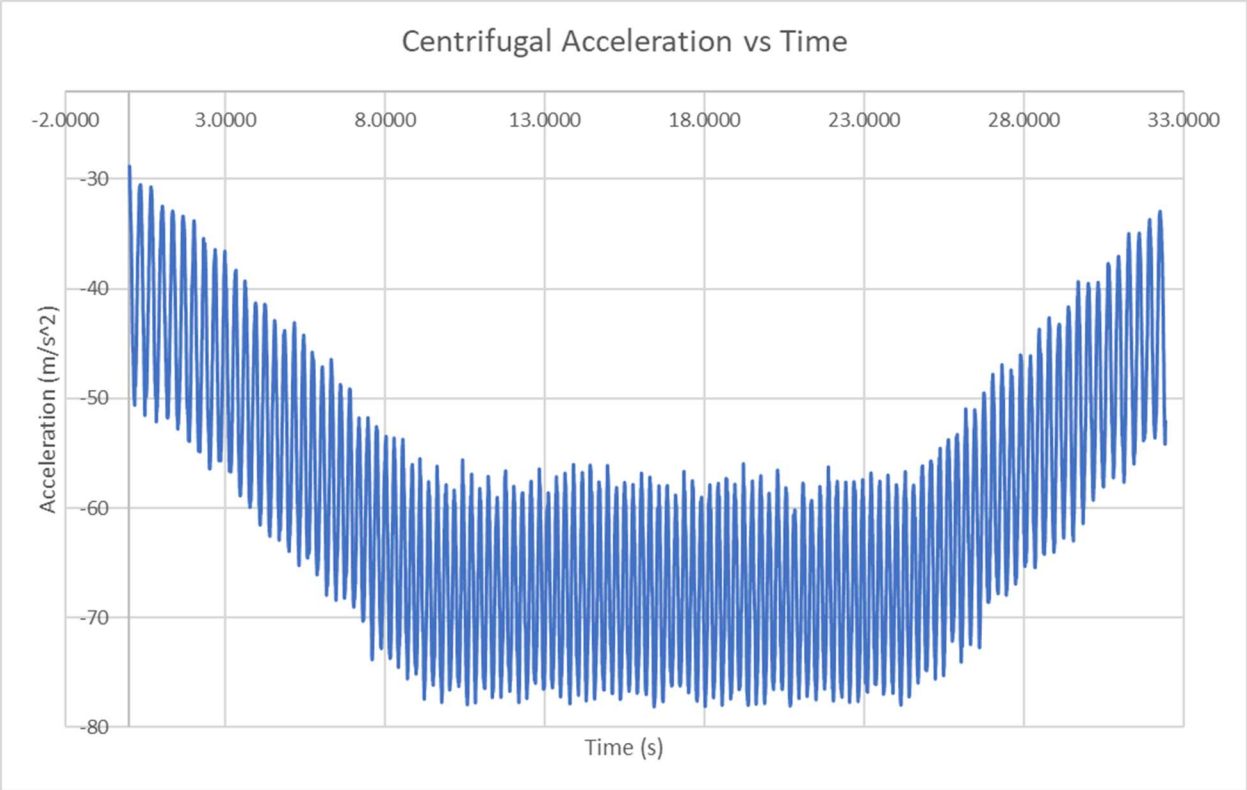
This will conclude the general assembly process of the project and begin the electrical installation phase.

Testing and Evaluation

At the time of submission of this report, several systems and subsystems have been tested and evaluated. The following are the tests complete.

Test	**Description**	**Results**
Motor Speed Control	Controlling the PWM of the motor via Cytron 13A Motor driver to achieve optimal RPM	Full control over motor speed and ramping functions. at 80% of Max PWM, optimal RPMs are achieved.
Sensor Communication	SCD30 (CO2, T, RH), Grove O2 Sensor, Analog Water Level Sensor, OLED Display	SCD30 and OLED integrated and working nominally. O2 sensor integrated but not calibrated. Water level sensors with A/D converter reading data appropriately.

Pump Control	Control water system via two peristaltic pumps and bypass solenoid valve system	Pumps function optimally for filling. Some obstacles encountered when draining system.
Light Control	Control 12V LED grow light strip and brightness via RPi	Light & brightness control accomplished.
Fan Control	Control four 12V fans for air circulation	Fans working nominally
Rotational Test	Connect motor physically to system to begin rotation testing	Goal of 250 rpm achieved at 80% duty cycle
Solenoid Valve	Six solenoid valves working in parallel with peristaltic pumps + bypass system	Solenoid valves installed and integrated with primary filling/emptying procedure. Bypass protocol needs additional programming
Imaging	Take images inside rotating chamber and transmit wirelessly	ESP32 cams integrated and connected to network. Can be accessed via browser and images can be captured/saved via the microcontroller
Centrifugal Acceleration	Measure the induced g-forces as the chamber rotates	Valid data. See chart



To confirm theoretical calculations, an MPU6050 accelerometer was placed inside the rotational chamber at a distance equivalent to where the seeds would be placed and brought up to near nominal speed (~230RPM). A moving average data filter of 7 points was used to clean up the data.

The first third and last third of the data set corresponds to the motor increasing and decreasing angular velocity, respectively. The center section displays the accelerometer data perpendicular to the axis of rotation as the system rotated over the course of ~14 seconds at a constant angular velocity before decelerating. As expected, the acceleration oscillated with a difference of +- 1g.

Detailed Budget

Our team received $1,600.00 from the Florida Space Grant Consortium in addition to $1200 from the OURI. Any other funding will go towards offsetting the total cost of completing this project. Funds were used to design and fabricate the indoor farming system in addition to supplementing travel to the Kennedy Space Center. It includes the following but is not limited to:

Botanicals					
Item Description	**Price**	**Quantity**	**Shipping**	**Tax**	**Sub Total**
Radish Seeds 1 lb.	$17.99	1		$0	$17.99
Edible Wafer Paper 9x12.6 in -50	$18.99	1		$0	$18.99
Edible Glue - 2oz	$7.69	2		$0	$15.38
Edible Glue	$13.27	1		$0	$13.27
Activated Carbon Pellets - 10oz	$11.95	1		$0	$11.95
		Botanicals Total =			$77.58

ELECTRONICS					
Item Description	**Price**	**Quantity**	**Shipping**	**Tax**	**Sub Total**

5M Waterproof Grow Light LED Strip	$20.99	1		$0	$20.99
Air Fans - 60mm - 5v or 24V - 4 Pack	$12.99	1		$0	$12.99
Cytron 13A DC Motor Driver PN00218-CYT3 MD13S	$10.89	1	$4.15	$0	$15.04
Raspberry Pi High-Precision AD/DA Expansion Board	$26.21	1	$2.99	$0	$29.20
Raspberry Pi 3 Model B+ Ultimate Kit	$129.95	1	$17.95	$0	$147.90
FQP30N06L Mosfets	$1.62	5	$7.80	$0	$15.90
DROK - 90010 Waterproof DC Buck Converter	$17.89	1		$0	$17.89
DC 12V 30A Power Supply 360W Universal Regulated Switching AC to DC Converter	$26.41	1		$0	$26.41
14 Gauge Silicone Wire Spool - 20ft. Red/Black	$22.98	1		$0	$22.98
Heater 12 v 100-watt w/ fan	$28.58	1		$0	$28.58
PN00218-CYT6 Cytron 30A DC Motor Driver MD30C	$34.38	1	$4.15	$0	$38.53
HiLetgo FT232RL FTDI Mini USB to TTL Serial Converter Adapter Module	$5.99	1		$0	$5.99
WAGO 2 Port (10) 3 Port (10) 5 Port (5) 221 Splicing Connector	$14.21	1		$0	$14.21

Sandisk 4GB MicroSDHC Memory Card	$8.89	2		$0	$17.78
UCTRONICS 0.96 Inch OLED Module 12864 128x64	$6.99	1		$0	$6.99
AC/DC Coverter	$37.41	1		$0	$37.41
Buck Converter	$9.95	1		$0	$9.95
Crimp Wire Connectors 18-22AGW	$5.95	1		$0	$5.95
Ring Terminal 18-22AGW #6 Stud	$4.95	1		$0	$4.95
2 pin connectors for 22-28AGW	$9.95	1		$0	$9.95
22AWG 6 colors 25 ft each-NOT SOLID-STRANDED	$15.95	1	$6.95	$0	$22.90
Gravity Water Level Sensor	$5.99	1		$0	$5.99
Gravity Water Level Sensor	$6.70	3	$21	$0	$41.10
Optical Tachometer	$165	1	$14.97	$0	$179.97
Motor Driver	$34.38	1	$4.15	$0	$38.53

Micro Right Angle Header Male 2 Pin 2 Row	$1.46	12		$0	$17.52
Micro Female 2 Socket 2 Row	$0.38	12		$0	$4.56
Micro Right Angle Header Male 4 Pin 2 Row	$2.21	11		$0	$24.31
Micro Female 4 Socket 2 Row - B/O 4-15-22	$0.39	15		$0	$5.85
Mini Right Angle Header Male 2 Pin 2 Row	$0.83	2		$0	$1.66
Mini Female 2 Socket 2 Row	$0.29	2		$0	$0.58
Mini Right Angle Header Male 4 Pin 2 Row	$2.02	2		$0	$4.04
Mini Female 4 Socket 2 Row	$0.36	2		$0	$0.72
Crimp Pin 18-24 AWG	$0.37	100		$0	$37
Crimp Socket 18-24 AWG	$0.52	100		$0	$52
Capacitor, 4.7uF / 325-475	$0.31	10		$0	$3.10
Diode, / 400-034	$0.38	10		$0	$3.80

FET, N-CH, 20V, / 402-044	$0.43	15		$0	$6.45
FET, N-CH, 100V, 10A, Dpak / 402-053	$1.18	3		$0	$3.54
FET, N-CH, 60V, 80A, D2pak / 402-078	$3.83	5		$0	$19.15
A/D Converter, 12bit, SAR / 424-055	$8.67	3		$0	$26.01
Motor Driver, 8-28V, 3A / 429-031	$1.74	5		$0	$8.70
Logic Gate, XOR, 1CH 2-IN / 440-068	$1.94	5		$0	$9.70
Fuse Block, NANO type / 550-012	$2.54	2		$0	$5.08
LitOrange 320PCS M2 Male Female Nylon Hex Spacer Standoff Screw Nut Assorted Assortment Kit (Black)	$13.88	1		$0	$13.88
		Electronics Total =			$1025.73

Lights, fans, and heater are all controlled by the Raspberry Pi. This microcontroller is capable of handling all aspects of automation to control the system. Wiring would connect all necessary components of the electrical system together. The wire splices and disconnects allow for easy maintenance and repair of the system.

FABRICATION

Item Description	Price	Quantity	Shipping	Tax	Sub Total
Aluminum - 1/8" - ($ by sq. ft.)	$197	1		$0	$197
Printer Filament - Kg	$21.99	4		$0	$87.96
#10-24 x 1in Screws - 100 pack	$7.20	1	$19.05	$0	$26.25
#10-24 x 0.5in screws - 100 pack	$4.88	1	$9.80	$0	$14.68
#10 lock washer - 100 pack	$4.31	1		$0	$4.31
#10 standard nut - 100 pack	$2.09	1		$0	$2.09
#10 std washer - 50 pack	$3.31	1		$0	$3.31
#10-24 lock nuts - 100 pack	$4.80	1		$0	$4.80
#8 Flat Washers - 30 pack	$2.88	1		$0	$2.88
#6-32 x 1.25in - 6 pack	$1,28	0		$0	Donated
#6-32 x 3/8in - 100 pack	$2.41	1		$0	$2.41
#6 standard nut - 100 pack	$1.41	1		$0	$1.41
M4-0.7 x 20mm Screws - 3 pack	$1.46	2		$0	$2.92
#0-80 x 1/4in Screws - 50 pack	$9.14	1		$0	$9.14
#0-80 Nuts - 100 pack	$5.02	1		$0	$5.02
12 inch Piano Hinges - 4 pack	$19.99	1		$0	$19.99

Cabinet Latch	$5.98	1		$0	$5.98
Cabinet Latch - REPLACEMENT	$11.90	1		$0	$11.90
1.77 x 0.75 in Chest Latches - 10 pack	$5.99	1		$0	$5.99
2.9in Spring Latch - 10 pack - REPLACEMENT	$12.99	1		$0	$12.99
Screen - Aluminum	$8.28	1		$0	Donated
		Fabrication Total =			$421.03

The system is aimed to be 0.125 m^2 in surface area and must be structurally sound enough to be transported into space. Aluminum sheeting was chosen for the frame because of its high strength to weight factor. Seams along the frame would be soldered. Nuts, bolts, and washers are used to secure individual components. Hinges, latches, and weather stripping are used to create easy access points while maintaining a partially sealed internal environment.

ROTARY MOTION					
Item Description	**Price**	**Quantity**	**Shipping**	**Tax**	**Sub Total**
12V 5310 RPM "CIM" Brushed DC Motor	$35.00	2	$17.33	$0	$104.74
Tachometer RPM Speed Meter	$18.99	1		$0	$18.99

RPM Gauges	$6.99	0		$0	Donated
Drive Belt - 48"	$13.07	3		$0	$39.21
3D Printer Wheels - 20 Pack	$15.99	1		$0	$15.99
3D Printer Wheels - 8 Pack	$7.99	1		$0	$7.99
Bearing Lubricant	$13.40	1		$0	$13.40
Sound Dampener 27 sq. ft.	$34.99	1		$0	$34.99
		Rotary Total =			$235.31

Centrifugal force would be used to create a pseudo-gravity. This would allow for control of water that would not normally be possible in space. The motors, drive belts, and belt tensioners provide a reliable method to maintain the needed rotation for the system. Caster wheels act as bearings for the rotating portion of the system in any orientation.

SENSORS					
Item Description	**Price**	**Quantity**	**Shipping**	**Tax**	**Sub Total**
Camera - Day/Night	$34.99	2		$0	Donated
CO_2 Humidity and Temperature Sensor - SCD30	$59.94	1	$8.86	$0	$68.81
Grove O2 Sensor	$54.90	1	$2.99	$0	$57.89
Analog Liquid Level Sensor	$2.99	2		$0	$5.98
		Sensors Total =			$132.68

Measuring the conditions inside the system is necessary to control the automation process. Water levels, humidity, temperature, light, gasses, pressure, and rotational velocity are all used to maintain internal settings. Cameras will document the results.

TRAVEL				
Item Description	**Price**	**Quantity**	**Tax**	**Sub Total**
Fuel Cost	$50	4	$0	$200

Our team has been awarded a grant through the Florida Space Grant Consortium which also includes the necessity to visit the Kennedy Space Center in Merritt Island, FL. With Covid concerns and social distancing guidelines individual rooms would be required for team members. These funds will be used to offset the personal cost for travel and lodging of six FAU students.

WATER DISTRIBUTION					
Item Description	**Price**	**Quantity**	**Shipping**	**Tax**	**Sub Total**

Slip Ring - 2 Port - 24, 2 Amp Wires	$105	1	$40	$0	$145
Peristaltic Pump - 12V	$25.80	2		$0	$51.60
Solenoid Valve - 1/4"	$6.19	6		$0	$37.14
Porous Tubing - 1/4" - 50'	$7.68	1		$0	$7.68
¼ in push to connect ball valve - 2 pack	$14.99	1		$0	$14.99
PEX Water Line - 1/4"	$15.45	0		$0	Donated
Braided Steel ¼'' water line - 12'' long	$8.25	2		$0	$16.50
Push Connect Tee - 1/4"	$3.73	4		$0	$14.92
Push Connect 90 - ⅓'' - 10 pack	$6.89	1		$0	$6.89
1/4 in Tube x 3/16in Barb	$1.53	2		$0	$3.06
1/4in tube x 5/16in Barb	$1.40	4		$0	$5.60
1/4in Push Connect x Universal 1/8in BSPP - 5 pack	$6.62	1		$0	$6.62
1/4 in MIP x 1/4in Push-to-Connect	$4.75	2		$0	$9.50
¼ NPT Tap and Dye Set	$4.30	1		$0	$4.30
Epoxy Resin - 8oz	$11.99	1		$0	$11.99

ABS to PVC Cement - 16oz.	$8.65	1		$0	$8.65
Purple Primer - 8oz.	$5.40	1		$0	$5.40
Clear PVC/ABS Cleaner - 8oz.	$8.14	1		$0	$8.14
Pasco Shower Pan Liner 40mil - /sq.ft	$7.50	0		$0	Donated
Pasco Shower Pan Cement - 1 pt.	$16.80	0		$0	Donated
		Water Dist. Total =			$357.98

The slip ring allows for water and electricity to be transferred from a stationary point to the rotating portion of the system. Self-priming pumps draw water from storage bladders into the system. Pipes and fittings are used to distribute water.

Project Total: ($2,450.31)

Conclusions

Hurdles in increased cost due to chip shortages and delays in shipping slowed our progress slightly, though excellent progress was made overall. The most difficult obstacle, inducing enough centrifugal acceleration to hold the water in the rotary chamber was overcome. The first grow cycle will begin shortly which will test the methodology of using edible wafer paper and glue to affix the seeds in place. The future is promising as grow cycles have already been performed using this media type. This project holds promise for future research in imitating gravitational environments for plant growth.

A great deal has been learned on behalf of all members regarding not only the esoteric topic of growing plants in space, but the specific hurdles encountered during the hands-on fabrication process (tolerances, machinery techniques, sound/vibration, wiring schematics). Many of these aspects were not covered in the FAU curriculum, challenging us to teach ourselves the knowledge required to accomplish our goal. Overall, the entire team is proud of our accomplishments in tackling a difficult problem that not even NASA engineers have been able to successfully solve.

References

[1] M. J. C. r. p. i. a. Musgrave, veterinary science, nutrition, and n. resources, "Growing plants in space," vol. 2, no. 065, 2007.

[2] C. L. Khodadad et al., "Microbiological and nutritional analysis of lettuce crops grown on the international space station," vol. 11, p. 199, 2020.

[3] R. Morrow, R. Remiker, M. Mischnick, L. Tuominen, M. Lee, and T. Crabb, "A low equivalent system mass plant growth unit for space exploration," SAE Technical Paper0148-7191, 2005.

[4] G. D. Massa, G. Newsham, M. E. Hummerick, R. C. Morrow, R. M. J. G. Wheeler, and S. Research, "Plant pillow preparation for the veggie plant growth system on the international space station," vol. 5, no. 1, 2017.

[5] M. Johnson, "Giving Roots and Shoots Their Space: The Advanced Plant Habitat," Apr 11, 2018.

[6] L. J. N. s. J. F. K. S. C. Herridge, "New Plant Habitat Will Increase Harvest on International Space Station," 2017.

[7] P. Fajardo-Cavazos, W. L. J. G. Nicholson, and S. Research, "Establishing Standard Protocols for Bacterial Culture in Biological Research in Canisters (BRIC) Hardware," vol. 4, no. 2, 2016.

[8] A. Caron, "Biological Research in Canisters–Light Emitting Diode (BRIC-LED)," 2016.

[9] L. Hall, “NASA is Everywhere: Farming Tech with Roots in Space,” 2019

Appendices

MATLAB Script

```
clear;clc;
% Given Data
Amin=0.125;             % minimum area in m^2
R1=0.127;               % inside diameter of maximum water depth in m
R2=0.157;               % outside diameter of water in m
Lex=0.017;              % extra length added for equipment in m
Md=2.393;               % mass of drum from solidworks in Kg
FSm=1.5;                % factor of safety for mass
g=9.81;                 % gravitational constant used in m/s^2
ft=250;                 % ideal frequency from tests in rev/min
t=1;                    % time to reach constant rotation in sec
mu=0.005;               % friction coefficient for rollers
dr=0.025;               % diameter of rollers in m
Nr=24;                  % number of rollers in contact with growth chamber
FSt=1.5;                % factor of safety for torque
dm=0.03125;             % diameter of motor drive wheel in m
Ft=25;                  % assumed tension on drive belt in N
x1=0.1425;              % position of motor along the x axis in m
x2=0.141;               % position of motor along the y axis in m
Nm=2;                   % number of motors in tension
% Equations for Dimensions and Mass
Lmin=Amin/(2*pi*R1);    % minimum length required in m
L=Lmin+Lex;             % length of growth chamber in m
V=pi*L*(R2^2-R1^2);     % volume of water in growth chamber in m^3
Mw=V*1000;              % mass of water in Kg and volume in liters
Mt=Mw+Md;               % total mass of growth chamber and water combined in Kg
Meq=Mt*FSm;             % equivalent mass used for calculations in Kg
% Equations for Frequency
Fmass=Meq*g;            % force of mass from gravity in N
Wmin=sqrt(g/R1);        % minimum angular velocity at surface of water in rad/s
fmin=Wmin*60/(2*pi);    % minimum frequency for gravitational equivalent in rev/min
Weq=2*pi*ft/60;         % angular velocity based on tests in rad/s
Feq=Meq*Weq^2*R1;       % centrifugal force based on tests in N
% Equations for Torque
I=Meq*(R1^2+R2^2)/2;    % moment of inertia in m^4
a=Weq/t;                % angular acceleration of growth chamber in rad/s^2
Tm=I*a;                 % torque due to mass at startup in N*m
Tmeq=Tm*FSt;            % equivalent torque for startup used in calculations in N*m
Tr=Nr*Feq*mu*dr/2;      % torque due to friction on rollers in N*m
Treq=Tr*FSt;            % equivalent torque from rollers used in calculations in N*m
Trhp=Treq*ft/7127;      % power to overcome friction from rollers in Hp
Tt=Tmeq+Treq;           % total torque at startup in N*m
Thp=Tt*ft/7127;         % power for growth chamber to reach constant rotation in Hp
Gr=R2/(dm*0.5);         % ratio between growth chamber radius and drive motor radius
Tm=Thp/Gr;              % maximum torque on motor
fm=ft*Gr;               % maximum frequency of motor in rev/min
```

```
% Equations for Force
thetaT=atan(x1/x2);     % angle to drive belt from the y axis in rad
Fty=Ft*sin(thetaT);     % force from drive belt along the y axis in N
Ftx=Ft*cos(thetaT);     % force from drive belt along the y axis in N
Fapp=Meq*g+Nm*Fty;      % force applied to system in N
fprintf('Length inside of growth chamber is %1.3f m.\n', L)
fprintf('Total water in growth chamber when soaking is %1.1f Kg / %1.1f L.\n', Mw,
Mw)
fprintf('Total weight of the growth chamber is %1.1f Kg.\n', Meq)
fprintf('Applied force across rollers is %1.0f N.\n', Fapp)
fprintf('Required speed to produce 1 g of force in microgravity is %1.1f rpm.\n',
fmin)
fprintf('Centripetal force inside the growth chamber at %1.0f rpm is %1.0f N.\n', ft,
Feq)
fprintf('Torque on growth chamber required to maintain constant rotation is %1.3f
Hp.\n', Trhp)
fprintf('Torque on growth chamber required to reach rotation in %1.0f sec is %1.3f
Hp.\n', t, Thp)
fprintf('Torque on motor required to reach rotation in %1.0f sec is %1.3f Hp.\n', t,
Tm)
fprintf('Frequency of motor at maximum rotation is %1.0f rev/min.\n', fm)
```

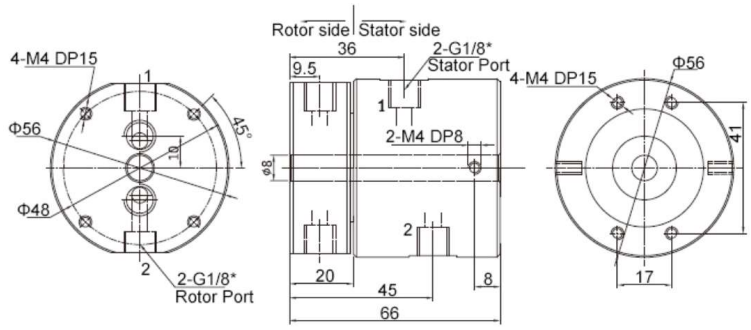
The series of 2-passage rotary union is a dual flow rotary union. It is 2 inlet 2 outlet, can rotate 360° and transfer fluid medium in two passages. It has the advantages of small size, light weight, small torque, and multiple gas passages without interference. In addition to standard models, other combinations of passages and transmission current can be customized.

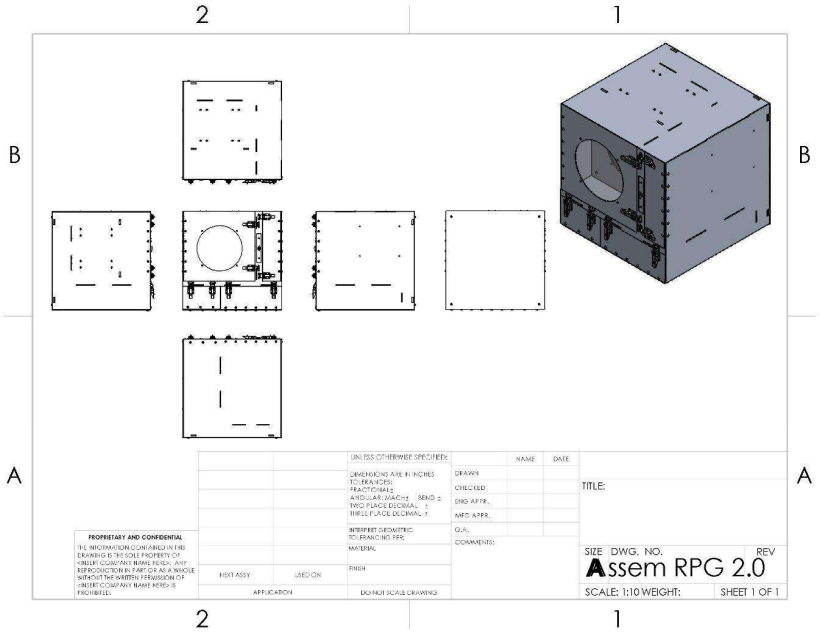
Specification

Model	ATO-3002006
Number of Passages	2 Passages
Pipe size	6~8mm Pipe (G1/8")
Max. Pressure	1Mpa, Vacuum -1.5kPa
Torque	<0.2N.m
Working Life	20 millions for ref (depends on working conditions)
Max. Speed	250rpm continously, 1000rpm intermittently
Working Temp.	-30℃~80℃
Work Humidity	0~85% RH
Contact Material	precious metal
Housing Material	aluminum alloy
Voltage Range	Power: 0~440VAC/VDC Signal: 0~440VAC/VDC
Insulation Resistance	Power: <500MΩ/300VDC Power: <500MΩ/300VDC
Lead Wire Spec.	Power:AWG17# Silverplated teflon wire Signal: AWG22# Silverplated teflon wire
Lead Wire Length	standard 250mm (can be customized)
Certification	CE, RoHS
Insulation Resistance	500VAC@50Hz,6s
Electrical Noise	<0.01Ω
IP Grade	IP51
Weight	About 1 kg

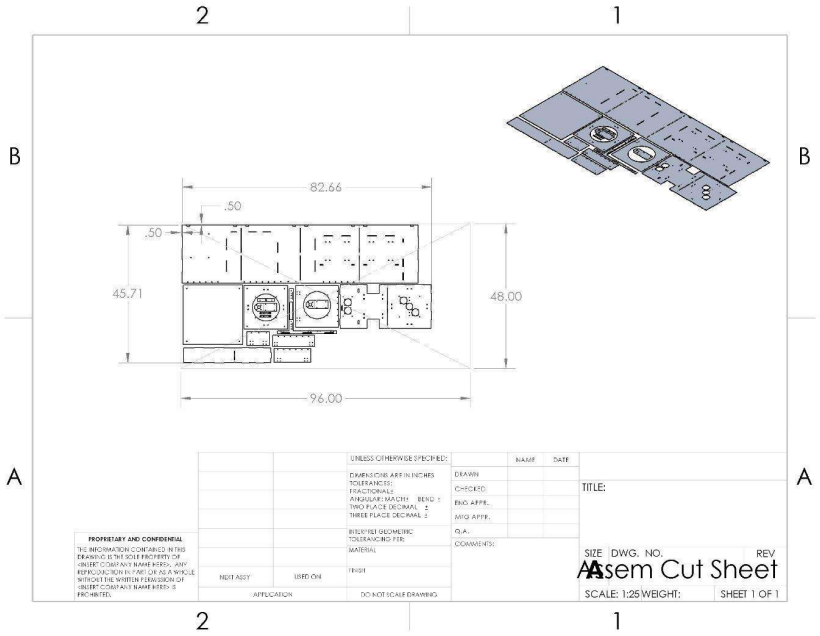
Note: Please contact us if you need the flange

Dimension Drawing of 2-Passage Rotary Joint (gas/flow transfer only)

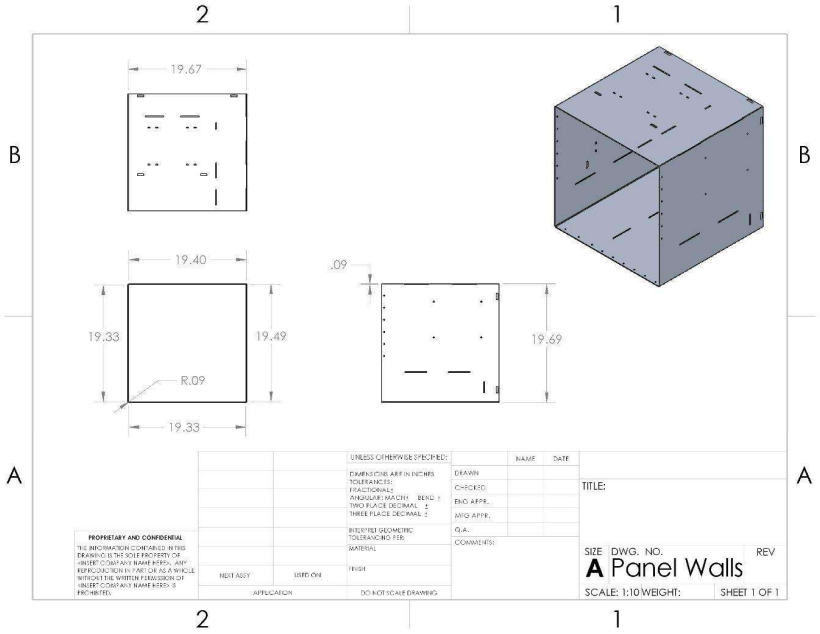




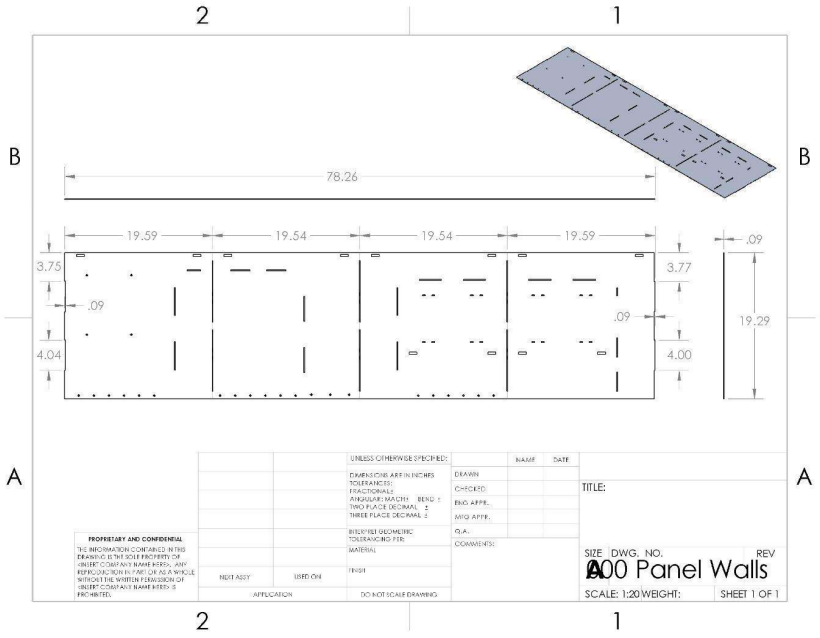
Overview of whole system



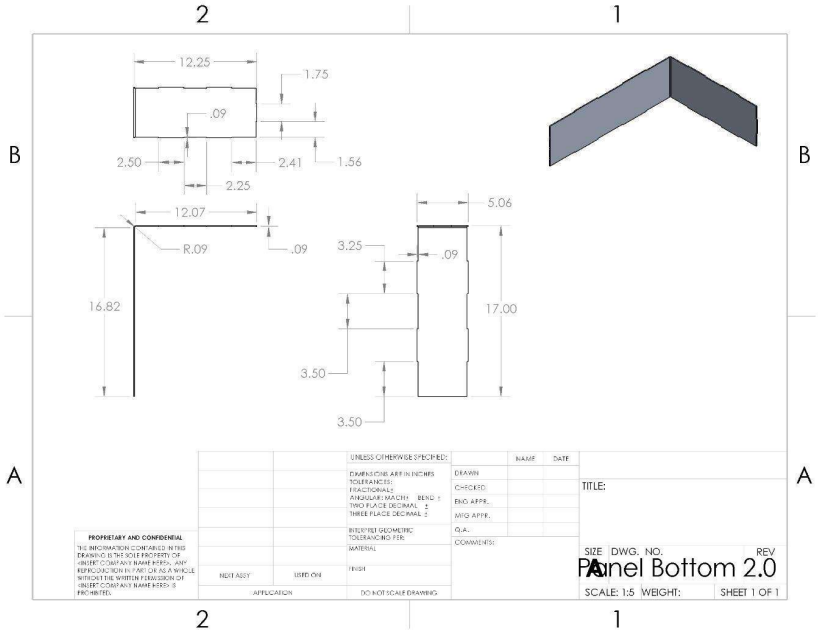
Cut sheet layout



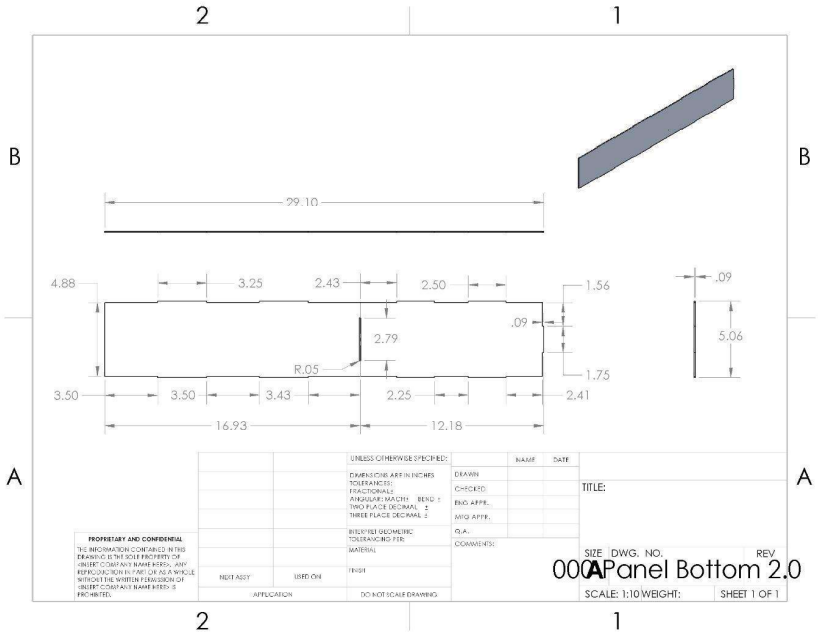
Outer Walls in Folded Position



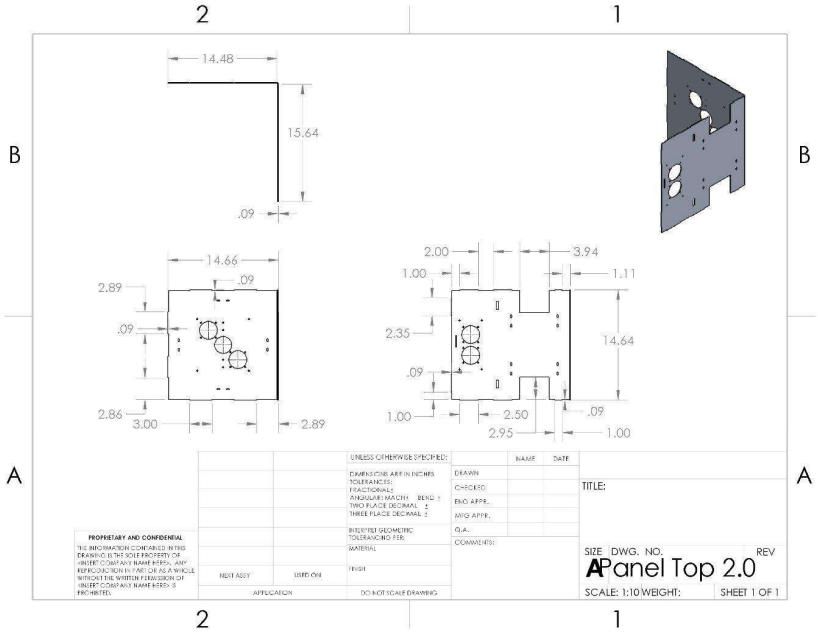
Outer Walls in Flattened Position



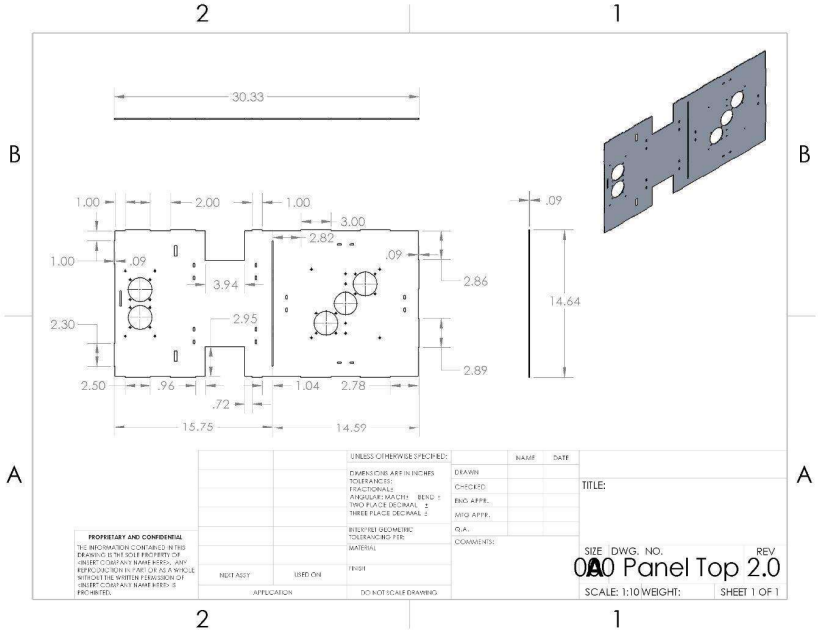
Bottom Panel in Folded Position



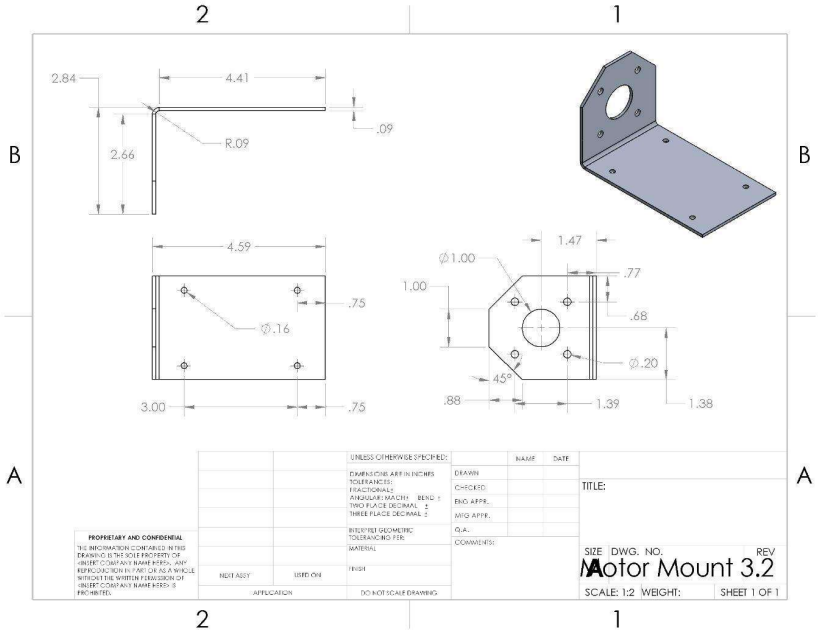
Bottom Panel in Flattened Position



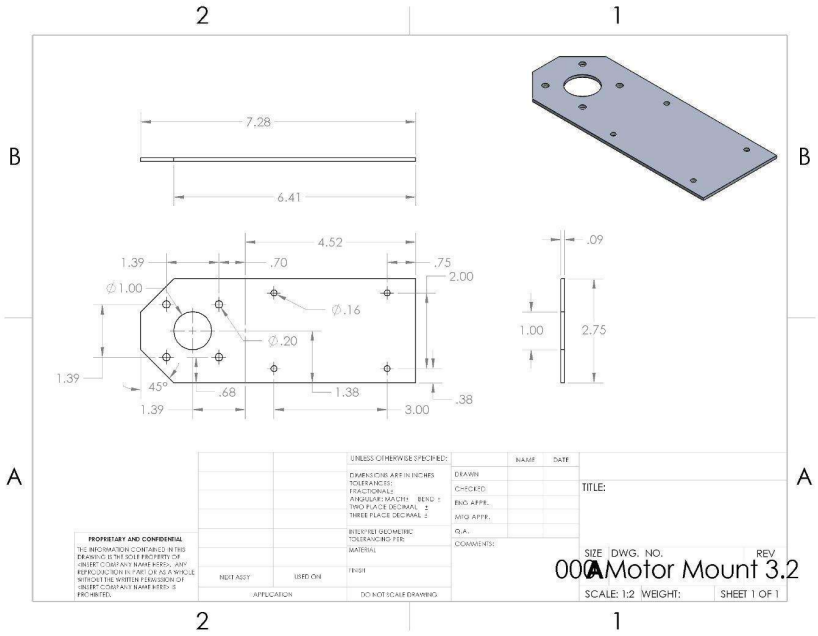
Upper Panel in Folded Position



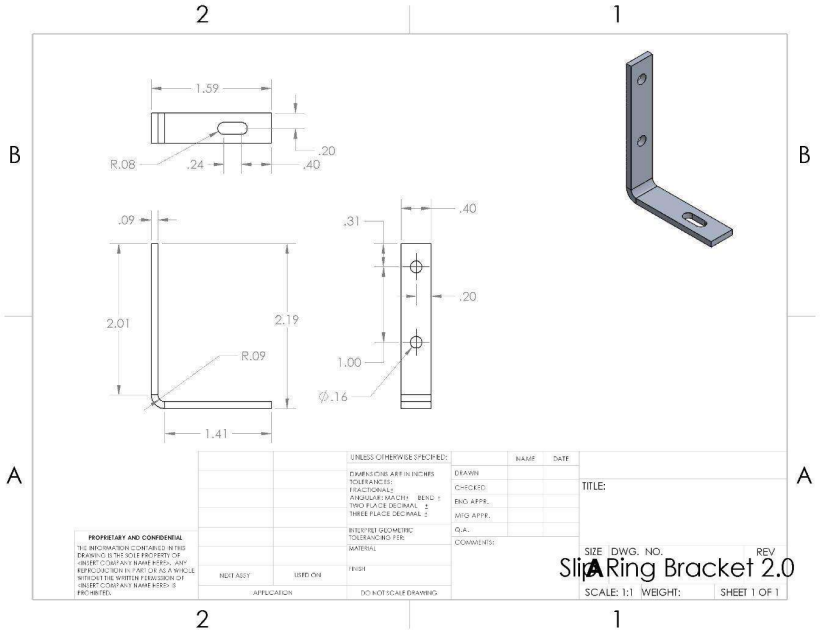
Upper Panel in Flattened Position



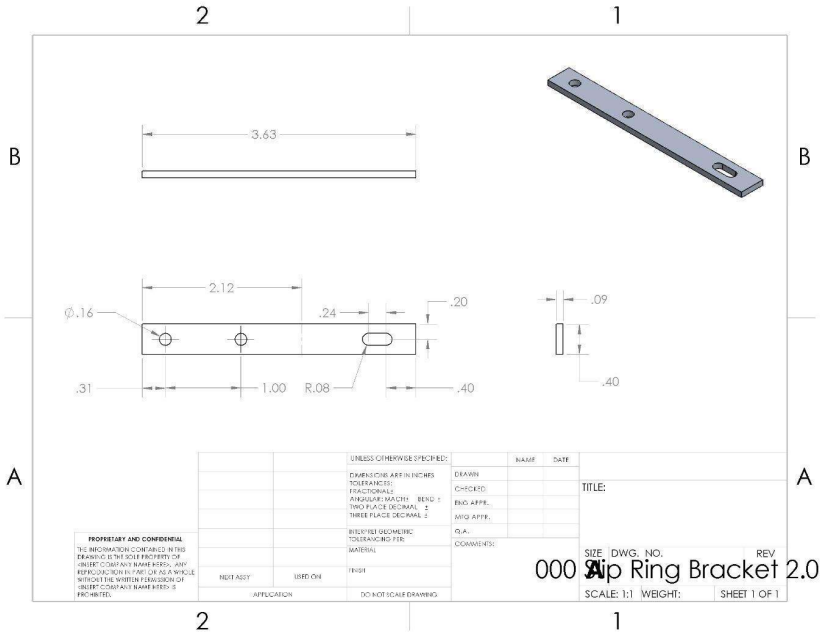
Motor Mount in Folded Position



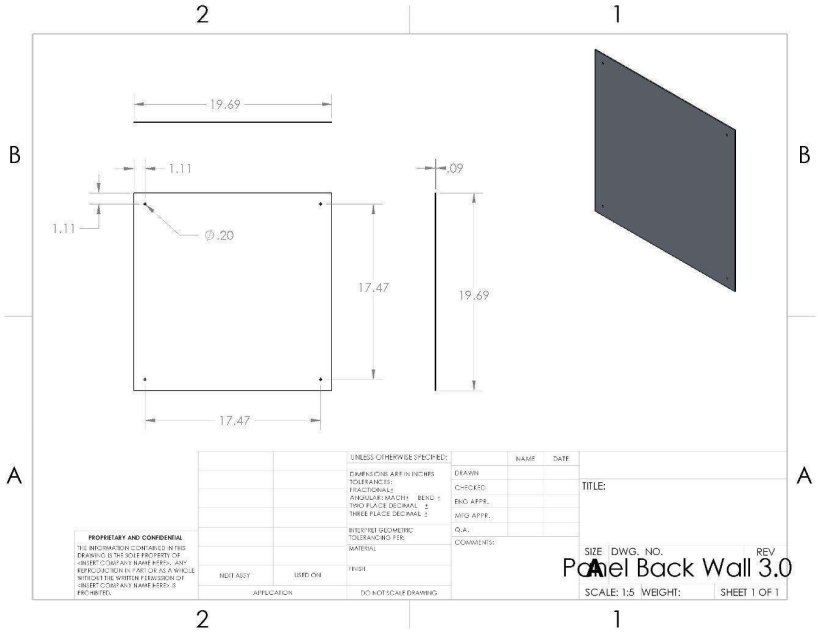
Motor Mount in Flattened Position



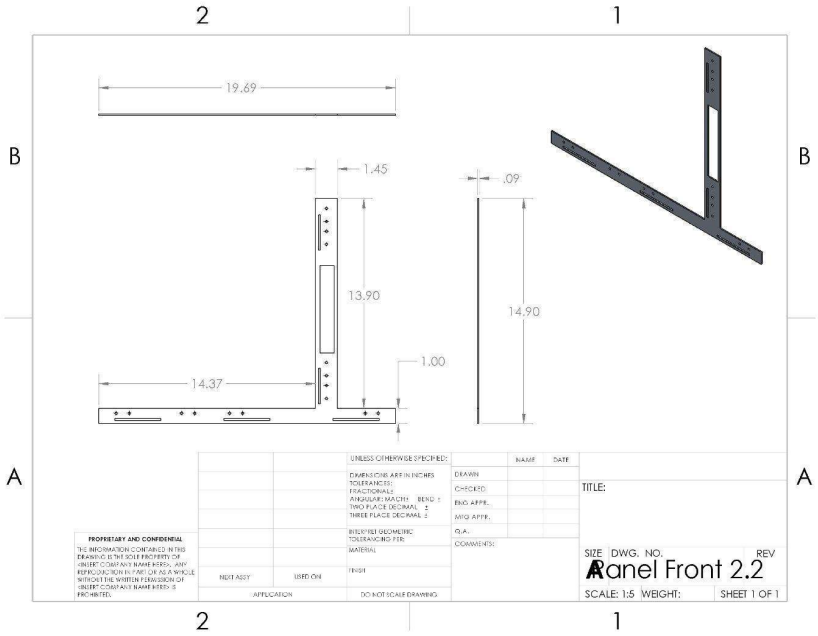
Slip Ring Mount in Folded Position



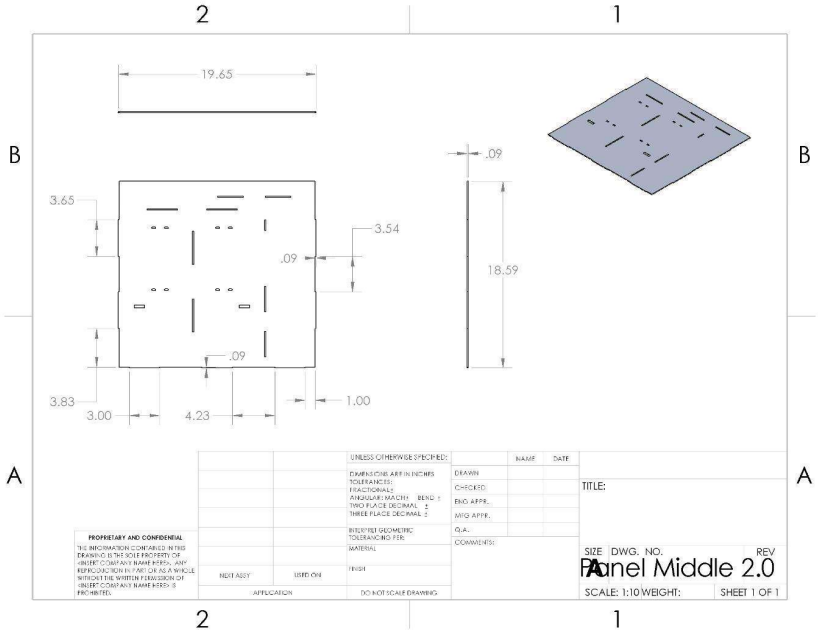
Slip Ring Mount in Flattened Position



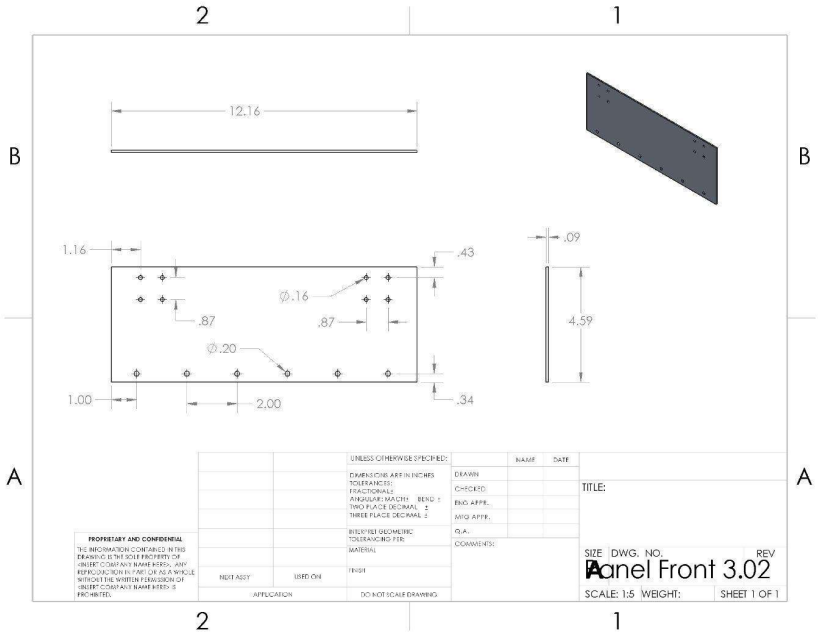
Back Panel



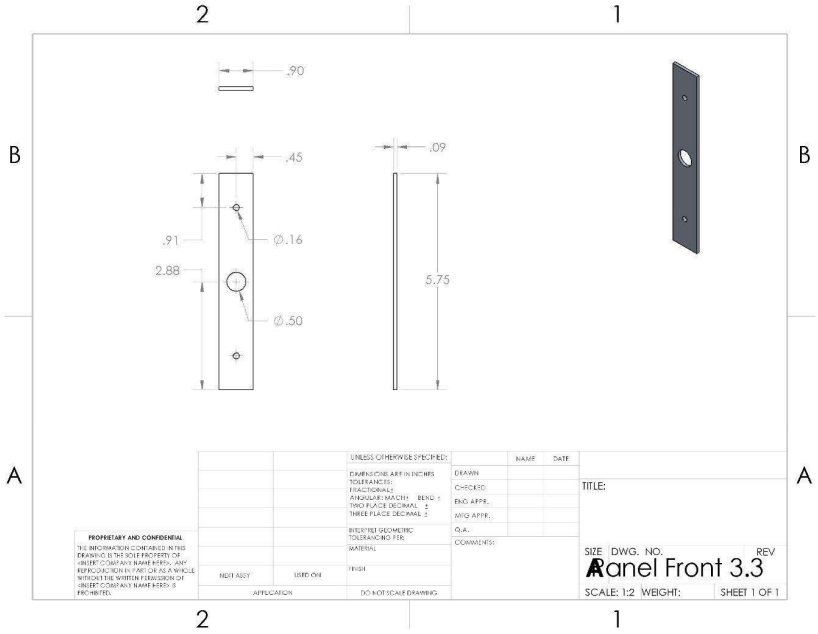
Front Panel



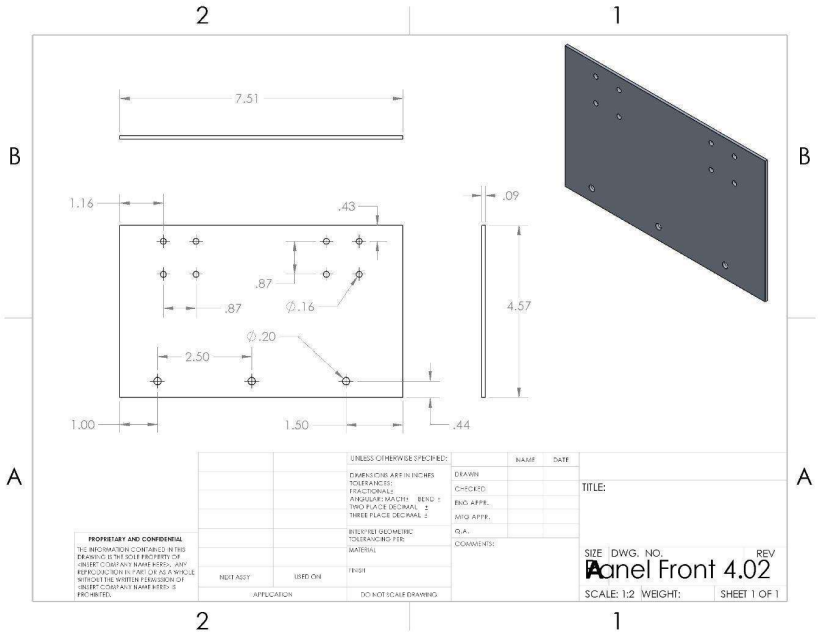
Middle Support Panel



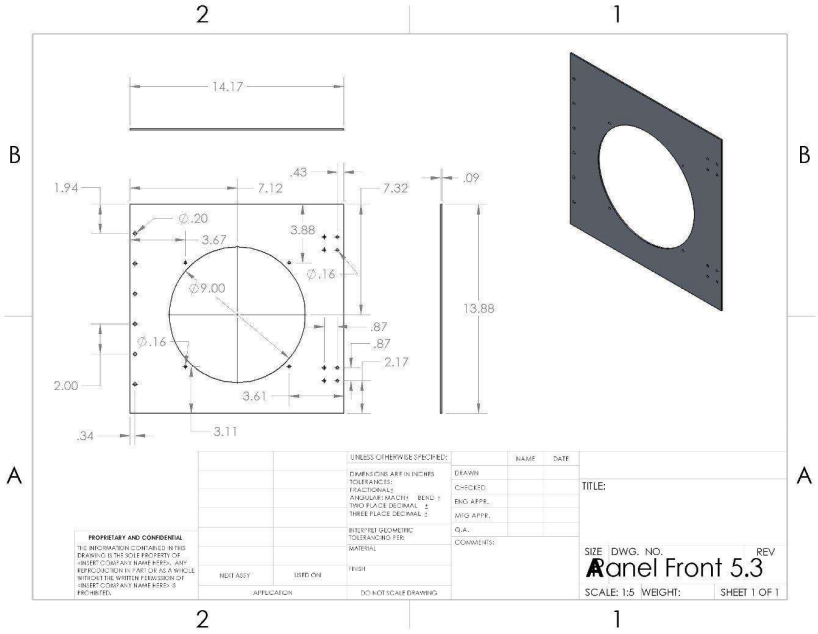
Seed Storage Door Panel



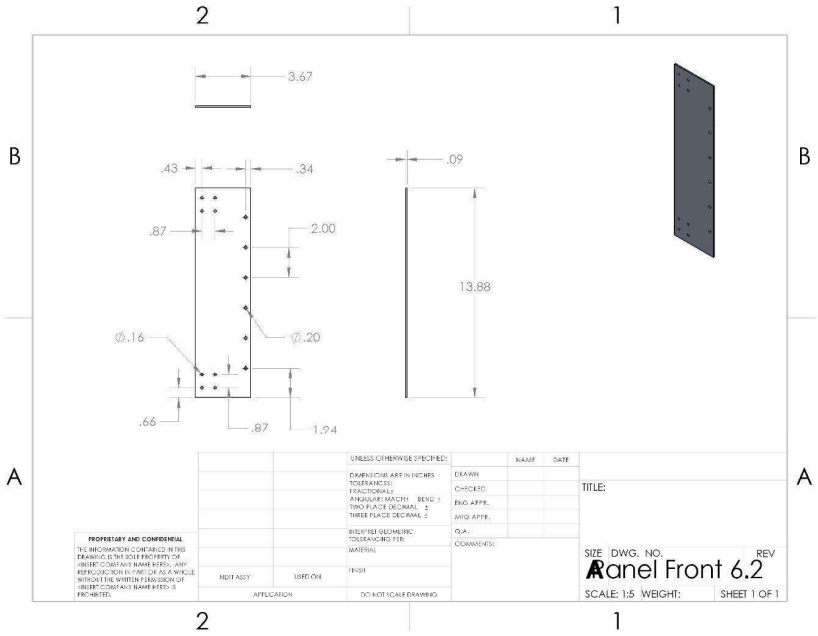
Air Filter Door Panel



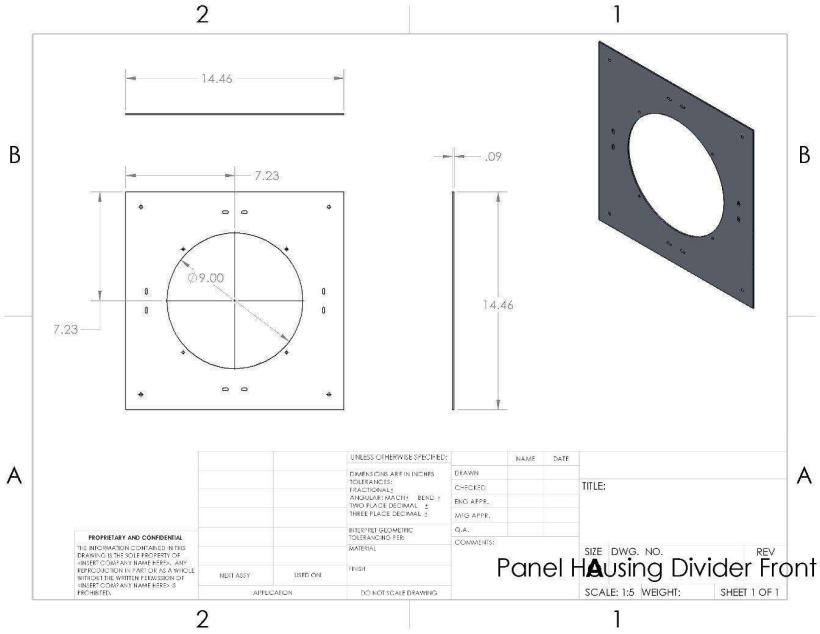
Water Storage Door Panel



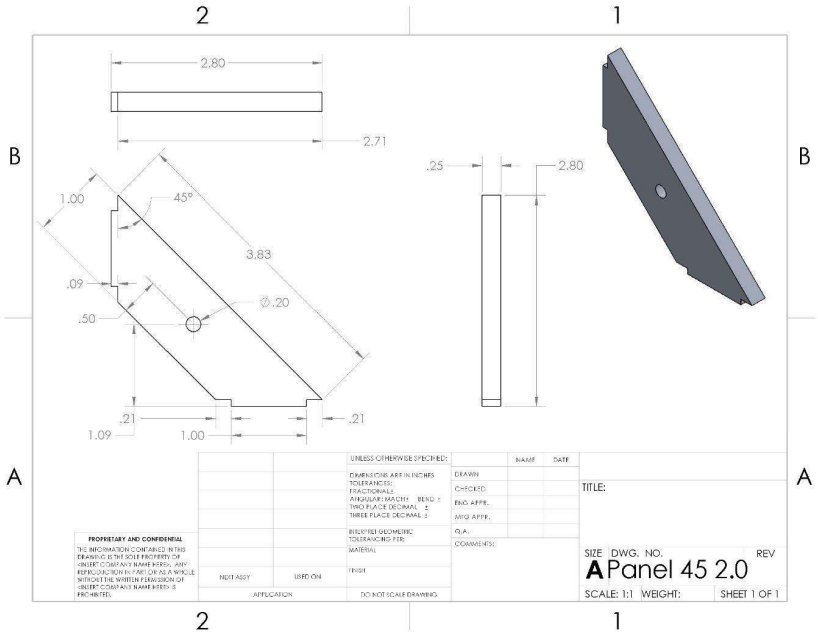
Growth Chamber Door Panel



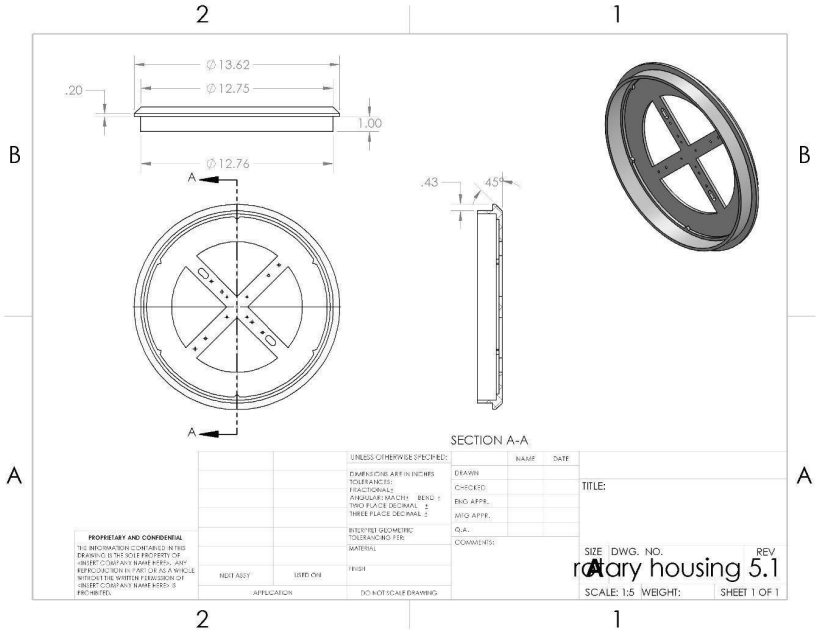
Electrical Door Panel



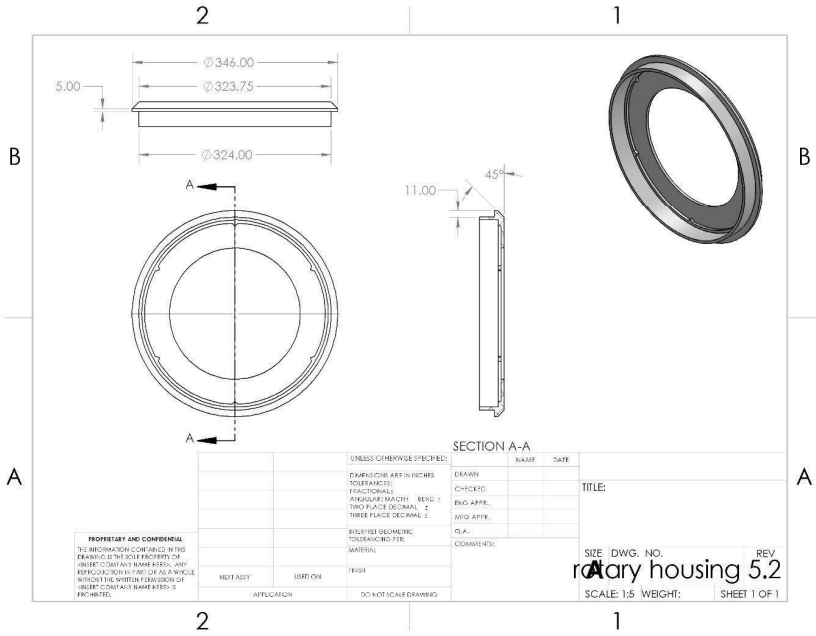
Growth Chamber Divider Panel



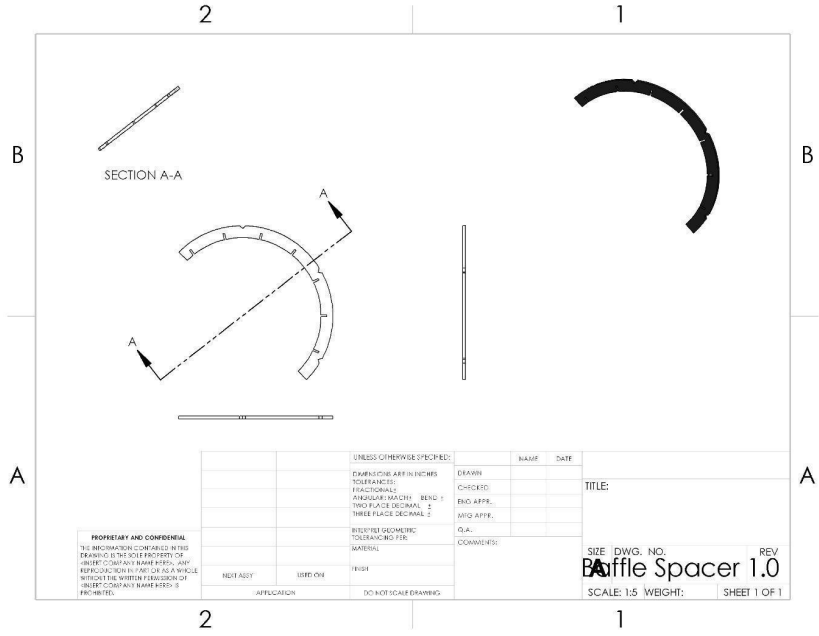
45 Degree Wall Mount Brackets



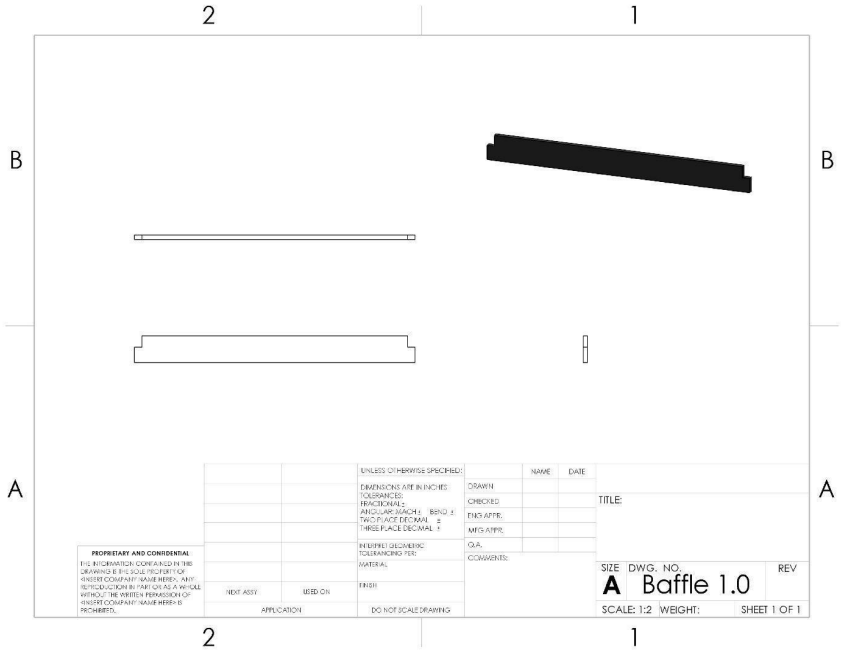
Growth Chamber Back End Cap



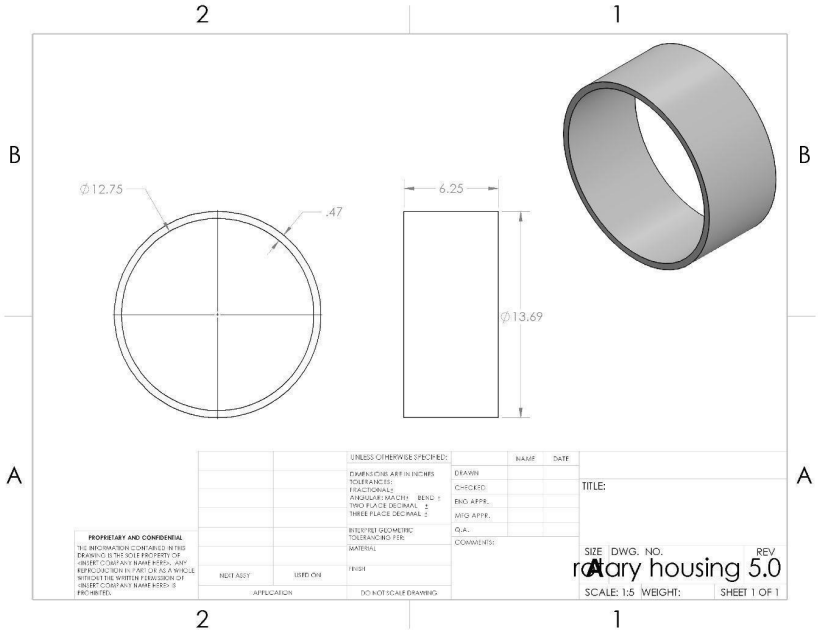
Growth Chamber Front End Cap



Baffle Spacers



Baffles



Growth Chamber Outer Ring

Python Code (04/20/22)

```
import os
import datetime
import keyboard
import Adafruit_SSD1306
from PIL import Image, ImageDraw, ImageFont
import RPi.GPIO as GPIO  #installed as sudo-apt get install python3-dev python3-rpi.gpio
import smbus
from scd30_i2c import SCD30
from time import sleep
import time
import ADS1256
import numpy as np
import urllib.request as URL

os.system('clear')
GPIO.setmode(GPIO.BCM)  #GPIO numbering
#display = Adafruit_SSD1306.SSD1306_128_64(rst=None)
GPIO.setwarnings(False)
print("Setting Pins")
MPWM = 13  #Set PWM line to pin 12 on Raspi (GPIO 12/Pin 32 has hardware PWM)
LPWM = 23  # Set PWM to pin 23 on Raspi for LED lights
DIR = 16  #Set directional pin to 8 on Raspi
M2PWM = 19
#DIR2 = 16
PUMP1 = 22
PUMP2 = 10
FAN = 9
#HPIN = 26
SOLA1 = 27
SOLA2 = 18
SOLB1 = 17
SOLB2 = 15
SOLC1 = 14
SOLC2 = 4
HEAT = 8
sleep(0.5)
print("Setting Default Duty/Freq")
MDUTY = 0  # Initial duty cycle set to 40%
PDUTY = 100  # Initial pump duty cycle set to 50%
LDUTY = 100  # LED Duty Cycle set to 100%
MFREQ = 1024  # PWM frequency 1024hz
LFREQ = 1024  # LED PWM Frequency 1024hz
PFREQ = 1024  # Pump PWM frequency 1024hz
p_initial = 1 #preset level for initial filling of chamber
p_preset = 1.3 #preset voltage level for water level sensor
preset = 1
sleep(0.5)
print("Setting GPIO as Outputs")
GPIO_initial_logic = (HEAT, MPWM, M2PWM, DIR, PUMP1, PUMP2, SOLA1, SOLA2, SOLB1, SOLB2,
SOLC1, SOLC2, LPWM, FAN) #, SOLA1, SOLA2, SOLB1, SOLB2, SOLC1, SOLC2)
GPIO.setup(GPIO_initial_logic, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(MPWM, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(M2PWM, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(DIR, GPIO.OUT)  # set motor PWM pin on Raspi as output
```

```
#GPIO.setup(PUMP1, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(PUMP2, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLA1, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLA2, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLB1, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLB2, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLC1, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(SOLC2, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(LPWM, GPIO.OUT)  # set motor PWM pin on Raspi as output
#GPIO.setup(FAN, GPIO.OUT)  # set motor PWM pin on Raspi as output

#GPIO.setup(HPIN, GPIO.IN)
sleep(0.5)
print("Setting All Pins to Low")
GPIO_initial_set = (HEAT, MPWM, M2PWM, DIR, PUMP1, PUMP2, SOLA1, SOLA2, SOLB1, SOLB2, SOLC1,
SOLC2, LPWM, FAN)
GPIO.output(GPIO_initial_set, GPIO.LOW)

mstatus = 0  #status condition for updating PWM
lstatus = 0  #status condition for updating PWM
p1status = 0  #status condition for updating PWM
p2status = 0
fstatus = 0
sleep(0.5)
print("Setting PWMs")
pwmi = GPIO.PWM(MPWM, MFREQ)  # Assign PWM and FREQ (channel + Freq) values to pwmi function
variable
pwms = GPIO.PWM(LPWM, LFREQ)  #Assign PWM and FREQ values to pwms (lights) function variable
pwmp1 = GPIO.PWM(PUMP1, LFREQ)
pwmp2 = GPIO.PWM(PUMP2, LFREQ)
i=1  #true false variable for program exit
sleep(0.5)

print("Pin Settings Complete \n")
sleep(0.5)
print("Setting I2C Devices")
try:
        scd30 = SCD30()
        scd30.set_measurement_interval(2)
        scd30.start_periodic_measurement()
except:
        print("SCD30 Error")
try:
        ADC = ADS1256.ADS1256()
        ADC.ADS1256_init()
        #Initialize the display
        display.begin()
        display.clear()
        display.display()
        displayWidth = display.width
        displayHeight = display.height
        image = Image.new('1', (displayWidth, displayHeight))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()
        draw.text((0,0), "Initializing...", font = font, fill = 255)
```

```
        display.image(image)
        display.display()
        sleep(2)
        display.clear()
        display.display()
        sleep(1)
        draw.rectangle((0,0,displayWidth,displayHeight), outline =0, fill=0)
        draw.text((0,16), "Awaiting Input", font = font, fill = 255)
        display.image(image)
        display.display()
except:
        print("OLED Error")

print("Complete")

#Analog Water Level Sensor Adjustement#
a1_adj = -0.0115  #zero out water level sensing so 0 volts is completely dry
a2_adj = -0.0115
a3_adj = -0.01
o_2 = str("NA")

def main_menu():
        global i
        print("""
        --------------------------------------------------
         Enter '1' to access motor controls \n
         Enter '2' to access light controls \n
         Enter '3' to access fan control \n
         Enter '4' to access sensor data \n
         Enter '5' to access pump control \n
         Enter '6' to access monitor mode (Press CTRL+C to Exit mode)\n
         Enter '7' to enter automation\n
         Enter '8' to exit program\n
         Enter '0' at any point to return to main menu\n
        --------------------------------------------------""")
        uinput = int(input("Please enter command: "))
        if uinput == 1:
                motor_menu()
        elif uinput == 2:
                lite_menu()
        elif uinput == 3:
                fan_menu()
        elif uinput == 4:
                sense_menu()
        elif uinput == 5:
                pump_menu()
        elif uinput == 6:
                monitor_mode()
        elif uinput == 7:
                auto()
                #rpm()
        elif uinput == 8:
                i = 0
        elif uinput == 9:
                image()
```

```
        elif uinput == 0:
                print("\n This is the main menu!! \n")
        else:
                print("Unknown Command")

def motor_menu():
        global MDUTY
        global MFREQ
        global mstatus
        print("-----------------------------\n"
                "Press 1 to Change PWM Duty Cycle\n"
                "Press 2 to Change PWM Frequency \n"
                "Press 3 to Activate Motor (Current Settings= " + str(MDUTY) + " %" + " & " + str(MFREQ) +
"Hz \n"
                "Enter 4 to Deactivate Motor (Current Setting= " + str(MDUTY) + " %" + " & " + str(MFREQ) +
"HZ)\n"
                "Enter 5 to Ramp Motor\n"
                "Enter 0 to return to Main Menu\n")
        minput = int(input("Please enter command: "))
        if minput == 1:
                try:
                        new_pwm = int(input("Please enter a new PWM Duty Cycle (0-100): \n"))
                        if new_pwm > 100:
                                print("Unacceptable High Value")
                                motor_menu()
                        elif new_pwm < 0:
                                print("Unacceptable Low Value")
                                motor_menu()
                        elif 100 >= new_pwm >=0:
                                MDUTY = new_pwm
                                print("New Value Assigned: " + str(MDUTY) + "% \n")
                                if mstatus == 1:
                                        pwmi.start(MDUTY)
                                        print("Duty Cycle Updated")
                                        sleep(0.5)
                                else:
                                        motor_menu()
                        else:
                                print("Unacceptable Input")
                except:
                        print("Unknown Try/Except Error")
        elif minput == 2:
                try:
                        new_freq = int(input("Please enter a new PWM Frequency (10-8000): \n"))
                        if new_freq > 20000:
                                print("Unacceptable High Value")
                                motor_menu()
                        elif new_freq < 10:
                                print("Unacceptable Low Value")
                                motor_menu()
                        elif 20000>= new_freq >= 10:
                                MFREQ = new_freq
                                print("New Value Assigned: " + str(MFREQ) + "Hz \n")
                                motor_menu()
                except:
```

```
                print("Unknown Try/Except Error")
        elif minput == 3:
                try:
                        RPWM = MDUTY
                        print("Max PWM: " + str(RPWM) + "% \n")
                        sleep(0.5)
                        RPWMM = (RPWM * 10)
                        GPIO.output(DIR, GPIO.HIGH)
                        for r in range(RPWMM):
                                RPWMD = r / 10
                                pwmi.start(RPWMD)
                                print("Duty Cycle= " + '{:.2f}'.format(RPWMD))
                                sleep(0.05)
                        mstatus = 1
                        sleep(0.5)
                        print("\n---------Motor at Full Speed---------- \n")
                        sleep(0.5)
                except:
                        print("Motor Exception Error")
        elif minput == 4:
                try:
                        RPWM = MDUTY
                        RPWMM = (RPWM * 10)
                        for d in range(RPWMM):
                                DPWMD = float(RPWM)-(d/10)
                                pwmi.start(DPWMD)
                                print("Duty Cycle= " + '{:.2f}'.format(DPWMD))
                                sleep(0.05)
                        GPIO.output(DIR, GPIO.LOW)
                        pwmi.start(0)
                        mstatus = 0
                        sleep(0.5)
                        print("\n---------Motor Deactivated---------- \n")
                        sleep(0.5)
                        motor_menu()
                except:
                        print("Motor Exception Error")
        elif minput == 5:
                try:
                        m_ramp()
                        motor_menu()
                except:
                        print("Motor Exception Error")
        elif minput == 0:
                try:
                        main_menu()
                except:
                        print("Motor Exception Error")

def lite_menu():
        global LDUTY
        global LFREQ
        global lstatus
        print("""----------------------\n
Enter 1 to adjust light PWM\n
```

```
Enter 2 to change light FREQ \n
Enter 3 to turn on lights\n
Enter 4 to turn off lights\n
Enter 0 to return to Main Menu\n""")
        linput = int(input("Please enter command: " ))
        if linput == 1:
                try:
                        new_lpwm = int(input("\nAdjust Lighting Duty Cycle (0-100: 0 to turn off, 100 full
brightness)\n" + "Current Setting: " + str(LDUTY) + "\n"+ "New Setting: "))
                        if new_lpwm > 100:
                                print("Unacceptable High Value")
                                lite_menu()
                        elif new_lpwm < 0:
                                print("Unacceptable Low Value")
                                lite_menu()
                        elif 100 >= new_lpwm >= 0:
                                LDUTY = new_lpwm
                                print("New Value Assigned: " + str(LDUTY) + "% \n")
                                lite_menu()
                                if lstatus == 1:
                                        pwms.start(LDUTY)
                                        print("Duty Cycle Updated")
                                        sleep(0.5)
                                        lite_menu()
                                else:
                                        pass
                        else:
                                print("Unacceptable Input")
                except:
                        print("Unknown Try/Except Error")
        elif linput == 2:
                try:
                        new_lfreq = int(input("Please enter a new Light PWM Frequency (10-8000): \n"
                        + "Current Setting: " + str(LFREQ) + "\nNew Setting: "))
                        if new_lfreq > 8000:
                                print("Unacceptable High Value")
                                lite_menu()
                        elif new_lfreq < 10:
                                print("Unacceptable Low Value")
                                lite_menu()
                        elif 8000 >= new_lfreq >= 10:
                                LFREQ = new_lfreq
                                print("New Value Assigned: " + str(LFREQ) + "Hz \n")
                                sleep(0.5)
                                lite_menu()
                except:
                        print("Unknown Try/Except Error")
        elif linput == 3:
                try:
                        GPIO.output(LPWM, GPIO.HIGH)
                        pwms.start(LDUTY)
                        lstatus=1
                        sleep(0.5)
                        print("\nLights Activated\n")
                        sleep(0.5)
```

```
                lite_menu()
            except:
                print("Unknown Try/Except Error")
        elif linput == 4:
            try:
                GPIO.output(LPWM, GPIO.LOW)
                pwms.start(0)
                lstatus=0
                sleep(0.5)
                print("\nLights Deactivated\n")
                lite_menu()
            except:
                print("Unknown Try/Except Error")
        elif linput == 0:
            try:
                main_menu()
            except:
                pass

def fan_menu():
        global fstatus
        finput=int(input("Enter 1 to activate fans. Enter 3 to turn off fans: "))
        if finput == 1:
            try:
                GPIO.output(FAN, GPIO.HIGH)
                sleep(0.5)
                print("Fans Activated")
                fstatus = 1
            except:
                print("Fan Exception")
        elif finput == 3:
            try:
                GPIO.output(FAN, GPIO.LOW)
                sleep(0.5)
                print("Fans Deactivated")
                fstatus = 0
            except:
                print("Fan Exception")
        else:
            pass

def m_ramp():
        global TPWM
        global RPWM
        print("-"*40)
        print("\nThis function will stop the motor, pause, and ramp up over a period of time to a specified PWM. "
              "What is the maximum PWM duty cycle you wish to stop at?")
        rpwm_input = int(input("Max Duty Cycle: "))
        TPWM = int(input("How long would you like to hold this max speed?: "))
        sleep(1)
        try:
            new_rpwm = rpwm_input
            if new_rpwm > 100:
                print("Unacceptable High Value")
            elif new_rpwm < 0:
```

```
                print("Unacceptable Low Value")
            elif 100 >= new_rpwm >= 0:
                RPWM = new_rpwm
                print("Max PWM: " + str(RPWM) + "% \n")
                sleep(0.5)
                print("Stopping Motor")
                pwmi.start(0)
                sleep(5)
                print("Starting Ramp")
                sleep(1)
                RPWMM = (RPWM * 10)
                for r in range(RPWMM):
                    RPWMD = r/10
                    pwmi.start(RPWMD)
                    print("Duty Cycle= " + '{:.2f}'.format(RPWMD))
                    sleep(0.05)
                print("Holding...")
                sleep(TPWM)
                print("Ramping Down Motor")
                for d in range(RPWMM):
                    DPWMD = float(RPWM)-(d/10)
                    pwmi.start(DPWMD)
                    print("Duty Cycle= " + '{:.2f}'.format(DPWMD))
                    sleep(0.05)
                pwmi.start(0)
                mstatus = 0
                sleep(5)
            else:
                print("unacceptable Input")
        except KeyboardInterrupt:
            pwmi.start(0)
            sleep(0.1)
        except:
            print("Unknown Try/Except Error")
def pump_menu():
        global p1status
        global p2status
        global PDUTY
        global p_preset
        GPIO_set_empty = [PUMP2, SOLB1, SOLB2] #Pins to activate to empty system
        GPIO_set_fill = [PUMP1, SOLA1, SOLA2] #Pins to activate to fill system
        pinput = int(input("""----------\n
        Enter 1 to adjust pump PWM\n
        Enter 2 to adjust pump PWM Frequency\n
        Enter 3 to activate Pump 1\n
        Enter 4 to activate Pump 2\n
        Enter 5 to deactivate Pump 1\n
        Enter 6 to deactivate Pump 2\n
        Enter 7 to fill to pre-set level\n
        Enter 8 to empty chamber\n
        Enter 0 to return to Main Menu\n"""))
        if pinput == 1:
            try:
                new_ppwm = int(input("\nAdjust PWM Duty Cycle (0-100)\n" +
                 "Current Setting: " + str(PDUTY) + "\n" + "New Setting: "))
```

```
                if new_ppwm > 100:
                        print("Unacceptable High Value")
                        pump_menu()
                elif new_ppwm < 0:
                        print("Unacceptable Low Value")
                        pump_menu()
                elif 100 >= new_ppwm >= 0:
                        PDUTY = new_ppwm
                        print("New Value Assigned: " + str(PDUTY) + "% \n")
                        if p1status == 1:
                                pwmp1.start(PDUTY)
                                print("Duty Cycle Updated")
                                sleep(0.5)
                                pump_menu()
                        else:
                                pump_menu()
                else:
                        print("Unacceptable Input")
        except:
                print("Pump Exception")
elif pinput == 2:
        try:
                new_pfreq = int(input("Please enter a new PWM Frequency (10-10000): \n"))
                if new_pfreq > 10000:
                        print("Unacceptable High Value")
                        pump_menu()
                elif new_pfreq < 10:
                        print("Unacceptable Low Value")
                        pump_menu()
                elif 10000 >= new_pfreq >= 10:
                        PFREQ = new_pfreq
                        print("New Value Assigned: " + str(PFREQ) + "Hz \n")
                        pump_menu()
        except:
                print("Pump Exception")
elif pinput == 3:
        try:
                GPIO.output(GPIO_set_fill, GPIO.HIGH)
                pwmp1.start(PDUTY)
                p1status = 1
                sleep(0.5)
                print("Filling Pump 1 Activated")
                pump_menu()
        except:
                print("Pump Exception")
elif pinput == 4:
        try:
                GPIO.output(PUMP2, GPIO.HIGH)
                GPIO.output(SOLB1, GPIO.HIGH)
                GPIO.output(SOLB2, GPIO.HIGH)
                pwmp2.start(PDUTY)
                p2status = 1
                sleep(0.5)
                print("Empty Pump 2 Activated")
                pump_menu()
```

```
                except:
                        print("Pump 2 Exception")
        elif pinput == 5:
                try:
                        GPIO.output(GPIO_set_fill, GPIO.LOW)
                        pwmp1.start(0)
                        sleep(0.5)
                        p1status = 0
                        sleep(0.5)
                        print("Filling Pump 1 Deactivated")
                        pump_menu()
                except:
                        print("Pump Exception")
        elif pinput == 6:
                try:
                        GPIO.output(PUMP2, GPIO.LOW)
                        pwmp2.start(0)
                        GPIO.output(SOLB1, GPIO.LOW)
                        GPIO.output(SOLB2, GPIO.LOW)
                        sleep(0.5)
                        p2status = 0
                        sleep(0.5)
                        print("Empty Pump 2 Deactivated")
                        pump_menu()
                except:
                        print("Pump Exception")
        elif pinput == 7:
                try:
                        pump_fill(preset)
                except:
                        print("Pump Filling Exception")
        elif pinput == 8:
                try:
                        pump_empty(preset)
                except:
                        print("Pump Empty Exception")
        elif pinput == 0:
                try:
                        main_menu()
                except:
                        pass

def sense_menu():
        global p1status
        global mstatus
        global lstatus
        global fstatus
        print("Finding I2C Devices:")
        bus = smbus.SMBus(1)
        for device in range(128):
                try:
                        bus.read_byte(device)
                        print(hex(device))
                except:
                        pass
```

```
	sleep(1)
	print("Hardware Systems (1 is Active. 0 is Inactive)\n" + "-"*20)
	print("PUMP: " + str(p1status) + "\nMOTOR: " + str(mstatus) + "\nLIGHTS: " + str(lstatus)+ "\nFANS: "
+ str(fstatus))
	sleep(1)
	print("Taking Environmental Measurements")
	if scd30.get_data_ready():
		m = scd30.read_measurement()
		if m is not None:
			print(f"CO2: {m[0]:.2f}ppm, Temp: {m[1]:.2f}'C, RH: {m[2]:.2f}%")
	main_menu()

def monitor_mode():
	try:
		while True:
			os.system('clear')
			ctime = datetime.datetime.now()
			ADC_Value = ADC.ADS1256_GetAll()
			if ADC_Value is not None:
				print("-"*40)
				print("1 ADC Voltage = %1f"%(ADC_Value[1]*5.0/0x7fffff))
				print("2 ADC Voltage = %1f"%(ADC_Value[2]*5.0/0x7fffff))
				print("3 ADC Voltage = %1f"%(ADC_Value[3]*5.0/0x7fffff+a1_adj))
				print("4 ADC Voltage = %1f"%(ADC_Value[4]*5.0/0x7fffff+a2_adj))
			else:
				pass
			sleep(1.5)
			try:
				m = scd30.read_measurement()
				if m:
					print(f"CO2: {m[0]:.2f}ppm\nTemp: {m[1]:.2f}'C\nRH:
{m[2]:.2f}%\nO2: "+ o_2)
					print("Press CTRL+C to Exit Monitor Mode")
					print("Last Updated: " + str(ctime))
					print("-"*40)
					display.clear()
					display.display()
					draw.rectangle((0,0,displayWidth,displayHeight), outline=0, fill=0)
					draw.text((0, 0), "CO2: " + str(m[0]), font=font, fill=255)
					draw.text((0, 16), "Temp: " + str(m[1]), font = font, fill=255)
					draw.text((0, 32), "RH: " + str(m[2]), font = font, fill = 255)
					draw.text((0, 48), "O2: " + str(o_2), font = font, fill = 255)
					display.image(image)
					display.display()
				else:
					pass
				sleep(3.5)
			except:
				pass
	except KeyboardInterrupt:
		main_menu()
	except:
		print("Monitor Mode Error")
def rpm():
	print("Currently Disabled")
```

```
        #run_time = 30 #seconds
        #i = 0
        #HPIN_status = False
        #start = time.time()
        #while (time.time()-start) < run_time:
        #       if GPIO.input(HPIN) == True and HPIN_status == False:
        #               i += 1
        #               HPIN_status = True
        #               print(i)
        #       elif GPIO.input(HPIN) == False:
        #               HPIN_status = False
        #               pass
        #       else:
        #               pass
        #rpm_reading = (i/run_time) * 60
        #print(rpm_reading)

def pump_fill(preset):
        global p_preset
        global p_avg
        try:
                fill_status = True
                while fill_status:
                        p_array = []
                        for i in range(3):
                                ADC_Value_3 = ADC.ADS1256_GetChannalValue(3)
                                p = float(ADC_Value_3 * 5.0 / 0x7fffff + a1_adj)
                                p_array.append(p)
                                print("Getting Data Points")
                                print(p)
                                sleep(3)
                        p_avg = np.sum((p_array)/3)
                        print("Average: ", p_avg)
                        p_diff = float(p_array[0] - p_array[2])
                        print("Difference: ", p_diff)
                        GPIO_set_fill = (PUMP1, SOLA1, SOLA2) #Pins to activate to fill system
                        if p_avg < preset and abs(p_diff) < 0.05:
                                GPIO.output(GPIO_set_fill, GPIO.HIGH)
                                pwmp1.start(PDUTY)
                                p1status = 1
                                print("Filling...\n")
                                sleep(3)
                                GPIO.output(GPIO_set_fill, GPIO.LOW)
                                pwmp1.start(0)
                        elif p_avg >= preset and abs(p_diff) < 0.05:
                                GPIO.output(GPIO_set_fill, GPIO.LOW)
                                pwmp1.start(0)
                                p1status = 0
                                print("Water Level Reached\n")
                                fill_status = False
        except KeyboardInterrupt:
                pump_menu()

def pump_empty(preset):
        global p_preset
```

```
	global p_avg
	try:
		empty_status = True
		GPIO_set_empty = [PUMP2, SOLC1, SOLC2] #Pins to empty pump
		while empty_status:
			p_array = []
			for i in range(3):
				ADC_Value_3 = ADC.ADS1256_GetChannalValue(3)
				p = float(ADC_Value_3 * 5.0 / 0x7fffff + a1_adj)
				p_array.append(p)
				print("Getting Data Point: ", p)
				sleep(3)
			p_avg = np.sum(p_array)/3
			print("Average: ", p_avg)
			p_diff = float(p_array[0] - p_array[2])
			print("Difference: ", p_diff)
			if p_avg > preset and abs(p_diff) < 0.05:
				GPIO.output(GPIO_set_empty, GPIO.HIGH)
				pwmp2.start(PDUTY)
				p2status = 1
				print("Emptying...\n")
				sleep(3)
				GPIO.output(GPIO_set_empty, GPIO.LOW)
				pwmp2.start(0)
			elif p_avg <= preset and abs(p_diff) < 0.05:
				p2status = 0
				print("Water Level Reaced\n")
				empty_status = False

	except KeyboardInterrupt:
		pump_menu()

def light_pulse(x): #x is number of times to pulse the light
	for i in range(x):
		GPIO.output(LPWM, GPIO.HIGH)
		pwms.start(LDUTY)
		sleep(0.2)
		GPIO.output(LPWM, GPIO.LOW)
		pwms.start(0)
		sleep(0.2)


def auto():
	start_time = datetime.time() #take the time to reference future automation
	try:
		ainput = input("This option will start the automation system. Enter 1 to start. 0 to go back. ")
		if ainput == 1:
			RPWM = MDUTY
			print("Max PWM: " + str(RPWM) + "% \n")
			sleep(0.5)
			RPWMM = (RPWM * 10)
			GPIO.output(DIR, GPIO.HIGH)
			for r in range(RPWMM):
				RPWMD = r / 10
				pwmi.start(RPWMD)
```

```
                                        print("Duty Cycle= " + '{:.2f}'.format(RPWMD))
                                        sleep(0.05)
                                mstatus = 1
                                sleep(0.5)
                                print("\n---------Motor at Full Speed---------- \n")
                                light_pulse(2)
                                print("Waiting 10 seconds...")
                                sleep(10)
                                try:
                                        GPIO.output(FAN, GPIO.HIGH)
                                        sleep(0.5)
                                        print("Fans Activated")
                                        fstatus = 1
                                        light_pulse(3)
                                except:
                                        print("Fan Exception")
                                try:
                                        pump_fill(p_initial)
                                except:
                                        print("Auto Pump Exception")
                        elif ainput == 0:
                                main_menu()
                except KeyboardInterrupt:
                        pass
                except:
                        print("Automation Exception Error")

#def image():
#       URL.urlretrieve("http://192.168.137.44/capture", "Images/%d.jpg",
if __name__ == '__main__':
                while i==1:
                        try:
                                main_menu()
#                               os.system('clear')
                        except KeyboardInterrupt:
                                i=0
                        except:
                                print("Secondary Except Error")
                else:
                        if mstatus == 1:
                                for r in range(MDUTY):
                                        DDUTY = MDUTY - r
                                        pwmi.start(DDUTY)
                                        sleep(0.1)
                                        print(str(DDUTY))
                        GPIO.output(LPWM, GPIO.LOW)
                        GPIO.output(MPWM, GPIO.LOW)
                        GPIO.output(FAN, GPIO.LOW)
                        GPIO.output(PUMP1, GPIO.LOW)
                        GPIO.output(PUMP2, GPIO.LOW)
```