Instrumented Ground Level Observer Of Hail Technical Specifications and Documentation

Howard Wen and Dylan MacPhail

August 2024

1 Introduction

There is a need to create reliable, accurate, and autonomous devices to collect data about hail. While current solutions exist, mainly disdrometers and hail pads, many solutions have downsides to them. For example, disdrometers are prohibitively expensive to maintain a network of them, and hail pads are too reliant on subjective and archaic forms of analysis. We have developed the Instrumented Ground Level Observer Of Hail (IGLOOH), a device primarily used to collect data about hail impact velocities which we then use to back-calculate hail size. Equipped with an environment sensor, decibel meter, radar and real time clock, it is capable of also recording various other environmental data. Solar power capabilities are also integrated into one of the models, allowing IGLOOH to be deployed both as a stationary and mobile device.

Contents

1 Introduction **1**

2 Design Overview **4**

3 Technical Specifications **5**
- 3.1 Physical Dimensions . . . 5
- 3.2 Electronic Components . . . 5
  - 3.2.1 Raspberry Pi 3B+ . . . 5
  - 3.2.2 Waveshare Solar Power Manager B . . . 5
  - 3.2.3 Large Solar Panel . . . 6
  - 3.2.4 Small Solar Panel . . . 6
  - 3.2.5 USB Splitter M-FF . . . 6
  - 3.2.6 Mini Fans . . . 6
  - 3.2.7 DS3231 Real Time Clock . . . 6
  - 3.2.8 Decibel Sound Level Meter Module . . . 6
  - 3.2.9 BME280 . . . 7
  - 3.2.10 uRAD . . . 7
  - 3.2.11 LED . . . 7
- 3.3 Auxiliary Components . . . 7
- 3.4 Raspberyy Pi 3B Pinout . . . 8

4 Prototype and Design Process **8**
- 4.1 Enclosure Development . . . 9
  - 4.1.1 Requirements . . . 9
  - 4.1.2 Prototyping and Testing . . . 9
  - 4.1.3 Design for manufacturing . . . 10
  - 4.1.4 Design for assembly . . . 11
- 4.2 Electronics and Software Development . . . 11
  - 4.2.1 Requirements . . . 11
  - 4.2.2 Code . . . 11
  - 4.2.3 Libraries . . . 12

5 Problems Encountered **12**
- 5.1 Enclosure . . . 12
- 5.2 Electronics and Software . . . 13
- 5.3 Deployment . . . 14

6 Future Direction **14**
6.1 Dome 14
6.2 ESP32 Integration 14
6.3 Wireless Enabled 14
6.4 Remote Data Upload 15
6.5 Multi-functional 15

A IGLOOH Drawings and Illustrations **17**
A.1 Mobile Variant 17
A.2 Stationary Variant 20

2 Design Overview

IGLOOH consists of a Raspberry Pi 3B powered by a Waveshare Solar Power Manager B. Peripherals include a DS3231 real time clock for time keeping, an LED to display when it is working, and a BME280 from Waveshare to record temperature, relative humidity, and air pressure. Most importantly, the IGLOOH has a uRAD radar set to continuous waveform mode to measure hail velocities radially (in this case, the Z direction).

IGLOOH has different configurations depending on method of deployment (supersite or mobile). When deployed at a supersite for long durations, it relies on a solar panel to recharge the battery it runs off of. It also requires an additional small solar panel and fans to properly manage thermals inside the enclosure. Because of this, the RPi is put into a specific enclosure (see Appendix A) with a waterproofed port for the solar panel cables as well as two fan pipes for air flow. Mobile IGLOOH deployment does not need solar power nor fans, and so it relies on the power manager itself (which holds 10,000mAh of charge) to power the RPi.

3 Technical Specifications

3.1 Physical Dimensions

	Stationary	Mobile
Dome Diameter	7.5”	
Dome Material	Acrylic	
Enclosure Height	90mm	80mm
Solar Panel Support	Yes	No
Exterior Bolts	#8-32	
Interior Bolts	M2.5	

IGLOOH originally currently used a 7.5” acrylic flanged dome. The enclosure that holds the RPi and peripherals is 3D printed with PETG and has an inner diameter of ⌀180 mm with a thickness of 2mm. The stationary model has a 90mm height to accommodate the bracket slot, while the mobile version is 80mm. Bolts used to secure the dome to the enclosure are #8-32, and M2.5s are used to mount the various electronics to the enclosure and lid as well.
See Appendix A for detailed IGLOOH drawings.

3.2 Electronic Components

3.2.1 Raspberry Pi 3B+

The ”master” of the IGLOOH is a Raspberry Pi 3B+ which is connected to the other sensors. Code on here is run primarily through Python, and uses various libraries and tools to effectively communicate with the other sensors.

3.2.2 Waveshare Solar Power Manager B

IGLOOH sustains itself off a Waveshare Solar Power Manager B which contains a 10,000mAh battery and components able to properly charge itself and control spikes/dips in sunlight. Purchased from Abra Electronics in Canada here.

3.2.3 Large Solar Panel

The Waveshare Solar Power Manager is charged with a SOLAR-19598 Polysilicon Solar Panel (18V 10W), High Conversion Efficiency panel. Also purchased from Abra here.

3.2.4 Small Solar Panel

We also bought a small solar panel to use for the fans. It's worth noting that we cannot control the power going into these fans, so as the sun gets brighter, the fans spin faster and vice versa.

3.2.5 USB Splitter M-FF

We noticed that the small solar panel was able to provide way more power than one single fan could handle, so we purchased a splitter to share the power from the panel to both fans.

3.2.6 Mini Fans

These are male USB ends, and connect to the splitter. They are very loud and vibrate a ton, so care is needed when deciding where to place them.

3.2.7 DS3231 Real Time Clock

Purchased from Amazon, this is used to prevent time drift on the Raspberry Pi, as the RPi is reliant on WiFi for an accurate time reading. Furthermore, in the case that the RPi loses power, the DS3231 will ensure that time will still be kept. This is connected to the RPi via I2C alongside the decibel meter. The I2C address of this should be on 0x78 but if properly connected, should show 0xUU. This must be connected PRIOR to turning the RPi on to ensure the correct RTC kernel is loaded.

3.2.8 Decibel Sound Level Meter Module

The decibel sound meter (DM) from PCB Artists is used to ensure autonomous detection of hail. It can read both dB sound levels and a frequency range, which are both valuable in determining whether it is hailing or raining. This is connected to the RPi also via I2C and a JST XH header (bought and assembled separately). We soldered the wires of the SDA and SCK on

parallel with the DS3231. It is possible to change the sample rate of the sensor, and we have it set either on 125ms or 1000ms. The I2C address of the DM is 0x48.

3.2.9 BME280

This Waveshare BME280 was originally used to measure thermals during the prototyping phase where heat management was a big priority. However, given that it provides useful data on temperature, relative humidity and pressure, we have chosen to continue to use it in our design. It is less important and useful on the mobile deployment currently, as the enclosure is entirely sealed so it doesn't read anything particularly useful. This is connected to the RPi via SPI and CS is on GPIO 5.

3.2.10 uRAD

The uRAD is used in the IGLOOH to track falling hail and record their impact velocities. The underlying theory is that we can back-calculate the size of hail from their fall speeds. Note that this radar only tracks velocities in the Z direction (i.e. up and down). This is connected to the RPi via a USB-A to USB-C connector. There are various settings on this worth looking into and changing, but primarily, altering the alpha and target values will change sensitivity and # of targets tracked respectively.

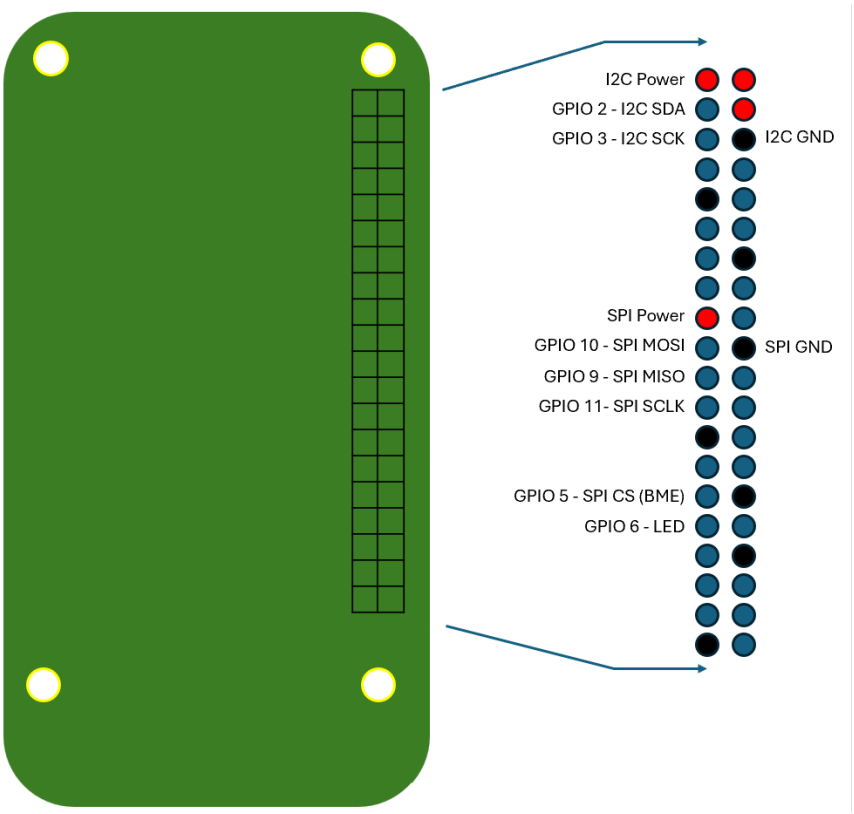
3.2.11 LED

There is a hackjob LED connected to GPIO 6 which, when on, indicates the program is functioning properly.

3.3 Auxiliary Components

To debug and interact with IGLOOH, we use a Logitech wireless keyboard + trackpad combo and a mobile screen powered by a portable charger. Using this configuration, it is possible to interface with IGLOOH on the move, and even without a screen using just the keyboard (granted typing blind is difficult). Using `Ctrl + Alt + T`, you can open up the terminal and interface from there. Debug the I2C sensors via `i2cdetect -y 1` or `i2cget -y 1 0xADDRESS` where the ADDRESS is the address of the sensor.

3.4 Raspberyy Pi 3B Pinout



The decibel meter and RTC are soldered in parallel for the I2C connections, while the BME280 is the only SPI connection and CS for the BME280 is connected to GPIO 5. The LED to display the functioning of IGLOOH is on GPIO 6.

4 Prototype and Design Process

The development of IGLOOH revolved around two main aspects: the enclosure and the electronics. Each had their own unique tribulations that this section will delve into.

4.1 Enclosure Development

4.1.1 Requirements

From the start, Julian had intended for IGLOOH (then unnamed) to be an autonomous hail impact velocity device, capable of being stationed the Water Valley supersite for extended periods of time. Think days, weeks, and even months if possible. Because of this, there were a few key requirements for the enclosure. Firstly, the enclosure obviously had to fit the dome. We were given a sample 3D print and the existing dome from the get-go, so we had to work with the dimensions of the dome from the start. Secondly, it had to be weather-proof. Since it would be outdoors in both the sun and in severe weather scenarios, it had to be able to vent heat, stay waterproof, and robust enough to survive moderate hail. Thirdly, the enclosure needed to be able to fit all the electrical components inside and retrieving data needed to be a smooth process. Finally, given our lack of machining equipment, our model had to be 3D printable.

4.1.2 Prototyping and Testing

Since Julian had already made a rough prototype, we mostly built off of his idea. We decided the best way to waterproof the dome was to keep the model in one piece, and flex seal if need be. Furthermore, to secure the seal between the dome and the enclosure, we found a suitable O-ring capable of surrounding the dome enough to create a seal.

To ensure our initial model of the dome was accurate, we printed a test enclosure that was just the connection between the dome and enclosure. This turned out great, and so we knew we had the correct measurements.

Then, according to Julian's initial goals for this to be a stationary hail monitor that would stay at a supersite 24/7, we designed a model with holes for fans to vent heat through and a port for solar panels to charge the battery and power the fans. Finding a way to keep the port for the solar panel cables weather proof was a challenge. We had to accommodate for the USB ends which are so much larger than the cable, which meant a big hole to fit through. Our first model had a roof and shield as an attempt to keep it weatherproof, but careless modeling had led to the port being too small to fit the usb through.

The solar panel usb cable was a lot larger than we anticipated, and we decided that the shield and roof wasn't going to be weatherproof enough if we were to accommodate the dimensions of the actual port. So, we came up with our second model which would have a large hole capable of fitting the USB through as well as two pieces to flex seal while they were clamped down on the cable. This setup would then be flex sealed again to the actual dome, allowing for a more robust weatherproof design. This is the current model that we use today.

Testing the device happened in multiple steps. To ensure the enclosure was waterproof and able to withstand torrential downpour, we put it in the shower and cranked the water as high as it could go, and then unscrewed it to see if any got in. When we were satisfied, we then filled up the sink with water and submerged the enclosure in it to ensure no water got in that way either. By this point, we were confident that the enclosure was weather-proof.

Testing thermals was the next step. Our acrylic dome was initially clear, and alongside black PETG, we were curious what baseline temperatures were. We put thermometer probes into both the dome and the enclosure underneath and then measured the temperatures of both on a hot sunny day every 5 minutes. We then painted both the dome and enclosure white and measured similarly again. We saw a noticeable drop in temperature just by painting the dome, dropping from 48℃ to 32 ℃. Addition of fans then lowered the temperature by around another 3℃. These temperatures were acceptable to safely store the raspberry pi and solar power manager in for prolonged periods of time.

4.1.3 Design for manufacturing

The primary limitation during our design phase for the enclosure was making sure it could be 3D printed. Since overhangs are generally difficult to print to a high quality, many of our designs needed chamfers to print well, which made space and design an ordeal.

Since the 3D printer was 250 x 210 x 220 mm, if we wanted a surefire way of keeping it weather-proof, printing in one piece would be the best way to do so. This meant that we also had to design an enclosure that was able to be printed within that printing bed size. - no overhangs, material strength,

enclosure size

4.1.4 Design for assembly

Designing for assembly meant taking into consideration how wires and cables would be threaded through, as well as how to connect sensors that would eventually be covered. The primary difficulty was finding a way to insert the solar panel cables into the enclosure while maintaining staying weatherproof. We solved this with two pieces that would clamp down on the cables and then we'd flex seal them to the enclosure which worked fine. Strength and robustness is an issue, which is something to be improved on.

We also had to account for wires and screws to mount the sensors and connect them to the pi. The only way to access the electronics was through the top, but with sensors being mounted on the lid, we needed a specific order of connecting the wires. Screws also needed easy ways to be accessed and had to be tight enough to keep them secure without being too tight that they'd break the enclosure.

4.2 Electronics and Software Development

4.2.1 Requirements

In terms of electronics and software, we needed to make sure our code could run autonomously, record down data in a logical and easy-to-read format, and lightweight enough to not burn through too much power.

4.2.2 Code

We chose to implement the code sensor-by-sensor, each as its own unique file (named `piSENSORNAME.py`). This meant installing the necessary libraries, finding an appropriate way to connect the sensor to the RPi, testing a basic interface with the sensor, before finally importing the library to the main code logic in a file called `iglooh.py`. This allowed for strong modularity, easy debugging, and building up familiarity with each of the sensors.

The code runs like so: Iglooh.py imports all sensor modules together, then runs a main while True: loop. In this are two timers, one that tells the RPi when to poll the DM and another that tells the RPi when to record and log environment variables (temp, RH, pressure, dB, freq_spec). If on one of these polls the dB reads a high enough level, it assumes it is hail,

and activates the radar. The radar will then run for 30 seconds, recording environment conditions every 15 seconds. While the radar is running, it will also be pinging the DM every 125ms, and if the sound still sounds like hail, it will continually extend the time to ensure that there is a 30 second running window. Basically, it will always run for 30s after the last detect hail noise.

4.2.3 Libraries

Since the RPi and Python both aren't exactly intended to be used as an embedded language, tons of libraries were installed to make this work. For Python, this list includes:

- `smbus2` for I2C interfacing
- `RPi.GPIO` for direct GPIO control for the LED
- `adafruit_circuitpython-bme280` includes all needed libraries for SPI interface and bme280 control
- `uRAD_USB_SDK11_Ns400doppler` is the sdk for the uRAD
- `serial` for serial communication via USB for uRAD
- `matplotlib` for DM meter visualization debugging and testing
- `numpy` for DM frequency bands
- `pandas` for csv file log recording

5 Problems Encountered

5.1 Enclosure

Solar panel cables: Our stationary model required solar panels for 24/7 usage. Finding a way to route the cables from the panels into the enclosure while maintaining its ability to remain weatherproof was difficult. Mainly this was because one of the ends of the solar panel cables was a female USB-A port. This was extremely bulky, and to accommodate this shape, we had to make the hole very large. This meant we needed some way to seal the hole. Our initial design was a mini-enclosure for the cable to go underneath

(see Appendix A), however, given the sheer size of the USB port, there was no way to 3D print it to be structurally sound and resistant to hail impacts.

The design we landed on was a larger hole with two smaller 3D printed parts with cable holes incorporated into them. These two parts would clamp down onto the cable and then be flex-sealed to the cable and enclosure. This would be more durable while also maintaining its ability to stay waterproof.

Vibration of fans: While testing its autonomous capabilities, we realized we were receiving a significant amount of data reading small and consistent values ($\tilde{5}$m/s). Unsure what this was at first, we conducted various tests. We manipulated air-flow thinking it might be particles in the air, we positioned fans far below the radar thinking it might've picked up on the fan blades, but we came to the conclusion that it was the vibration of the fans on the lid. Since the lid wasn't tightly attached to the enclosure, it would vibrate just enough that the radar would pick up on it and cause issues. We did not have the time to solve this in the summer, and more prototyping will be done in the winter to solve this issue.

5.2 Electronics and Software

Mobile Device Connections: For easier deployments, we wanted to be able to remotely activate IGLOOH. However, we found that the Raspberry Pi 3B and 4Bs were largely unreceptive to receiving commands from a laptop, be it Wi-Fi or Bluetooth. Lots of external configurations had to be modified, deleted, or added to get any sort of connection, and largely it was unstable and unfeasible. Thus, we stuck to blindly typing on a keyboard without a screen to execute commands and hoping the LED would shine (indicating the code was working). We hope to mitigate this problem in the future by switching to an ESP32, which is designed for wireless communication and IoT functionality.

JST XH Headers: The decibel meter came without any headers or wires, and so it was up to us to connect it with jumper cables. This caused severe issues since the connection was quite loose and would often disconnect, thus preventing the rest of the code from working. It took us way too long to figure out we could just buy JST XH headers to connect to the DM, but once we did, it worked fantastic.

5.3 Deployment

IGLOOH managed to be deployed six (6) times this field season. From rain to hail able to cause $2.8 billion in damage, IGLOOH has seen it all. We plan to spend the winter processing and analyzing the data it has saved. We also deployed it once at Water Valley, however, no storms passed by so it collected no data.

Unfortunately, during our August 5th chase with IBHS, IGLOOH's dome was destroyed by large hail, and is currently out of commission. We plan to use this opportunity to re-vamp and design a next-gen IGLOOH.

6 Future Direction

6.1 Dome

Besides being unable to withstand large hail, many of the radar data points seemed to experience attenuation and noise. After researching optimal radome design and 24 GHz radar properties, our conclusion is that acrylic domes are suboptimal for IGLOOH design. We will look into designs using Teflon and Polycarbonate, as well as potentially changing the shape of the enclosure.

6.2 ESP32 Integration

The initial use of Raspberry Pi was simple for prototyping, however, as we move into a more refined and sophisticated stage of design, it is important we are conscious optimizing our design. This includes integrating ESP32s as the primary microcontroller instead. While Raspberry Pis are fine, they are too power inefficient and bloated with software that is unnecessary for our requirements. ESP32s are lightweight, efficient, and design for IoT applications which is what this project's next steps are/

6.3 Wireless Enabled

Our current header-less design is quite a poor deployment experience. A lot of the time we are unsure whether the device is functioning properly or not, and we have no way to activate it other than unscrewing the dome and turning the device on. We plan on leveraging ESP32 IoT functionality alongside mobile and website apps to better control the device.

6.4 Remote Data Upload

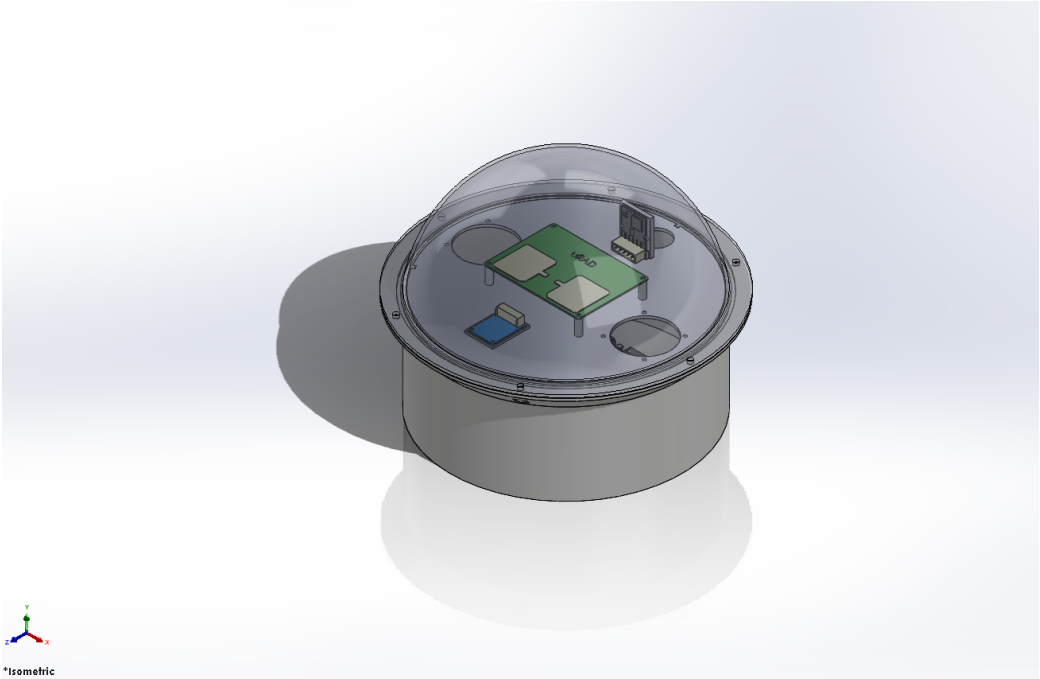
This device needs to be able to remotely upload data to a server when it is stationed and deployed. We plan to integrate 4G capabilities onto IGLOOH and build a back-end to accomplish this in the future.

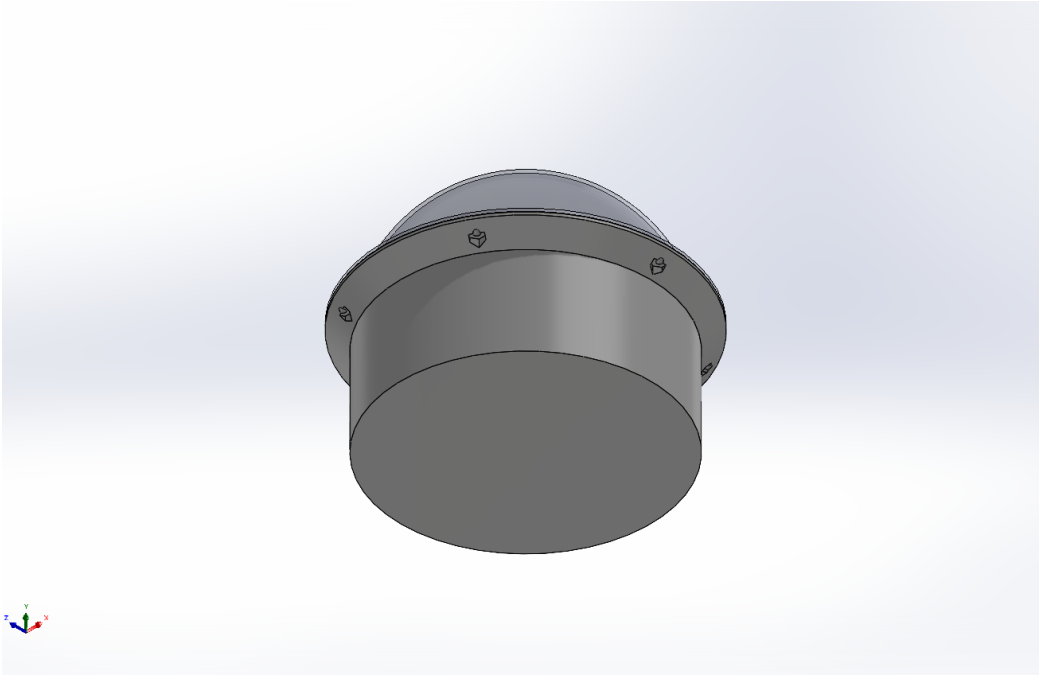
6.5 Multi-functional

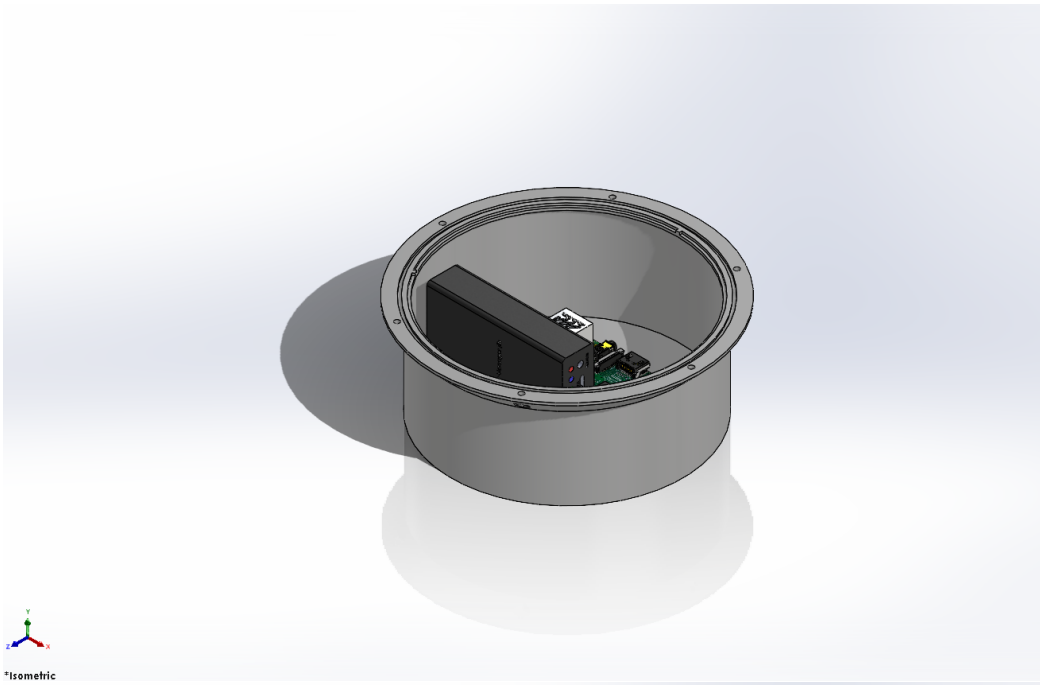
IGLOOH does not need to be limited to just radar data. We plan to better leverage the BME280 and sound sensor to make it a more comprehensive weather station. In the future, we also hope to explore adding cameras, tripod legs, and anemometers to the device to make it even more all-encompassing.

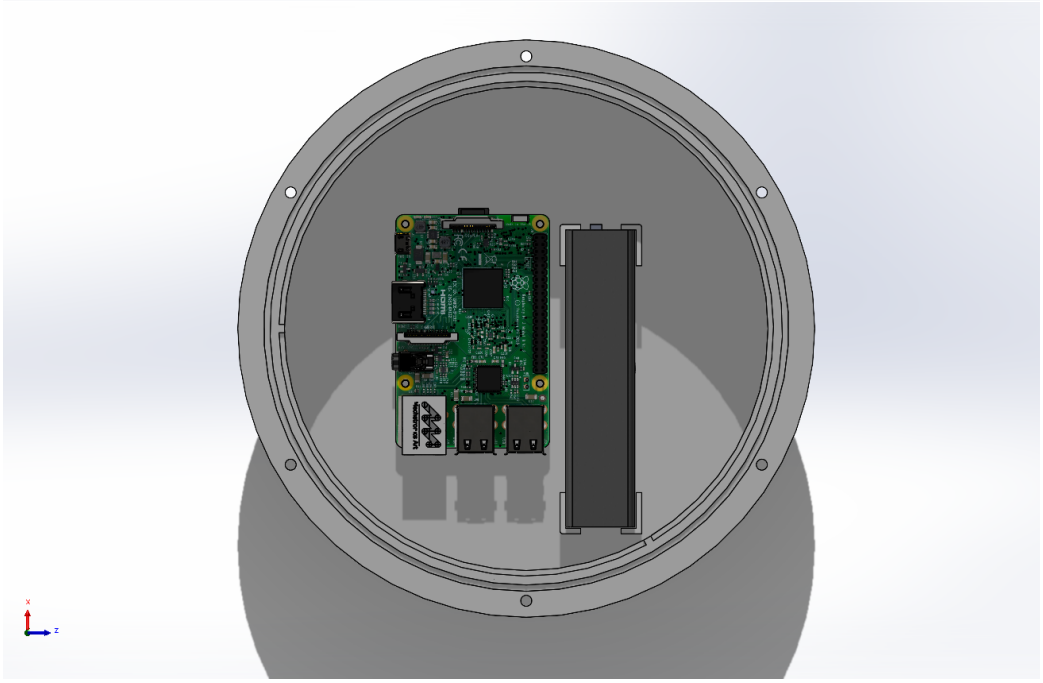
A IGLOOH Drawings and Illustrations

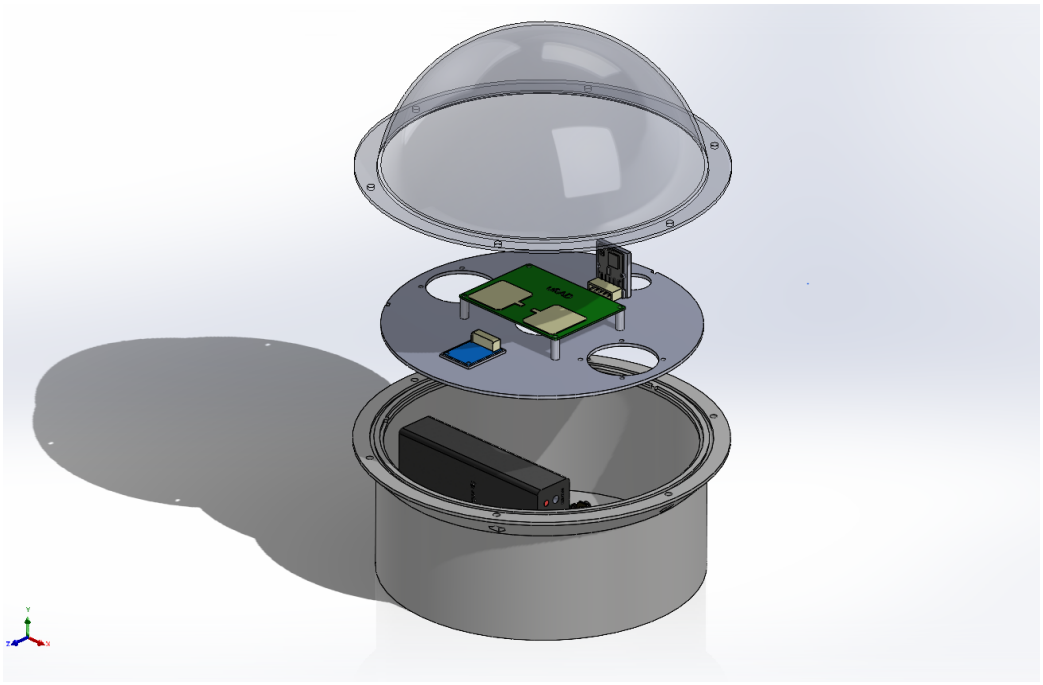
A.1 Mobile Variant



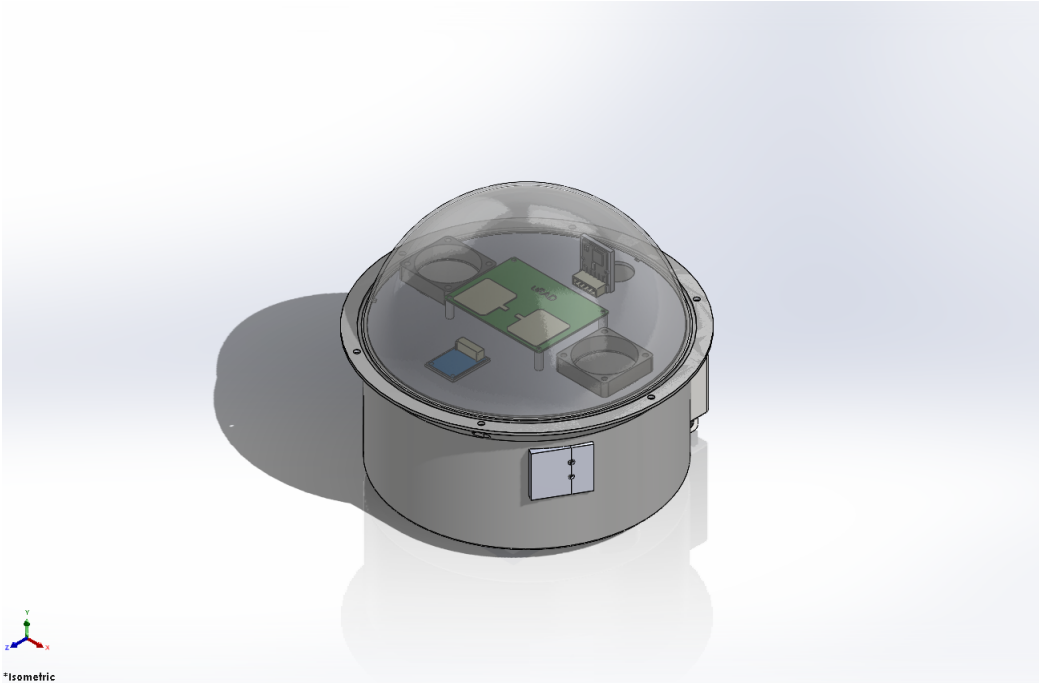


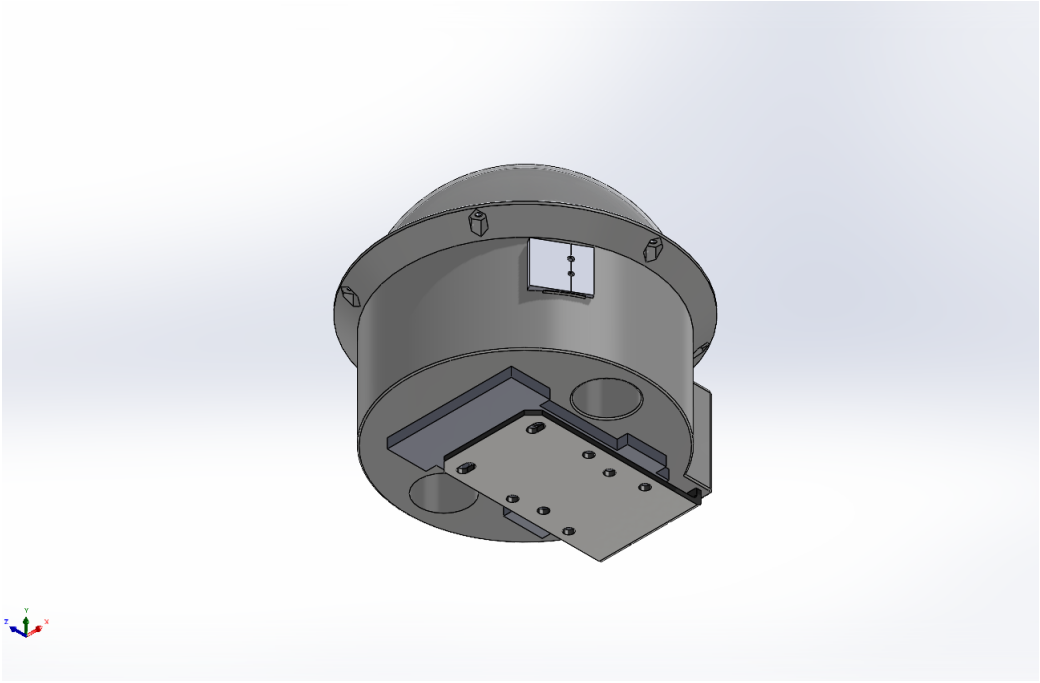

*Isometric

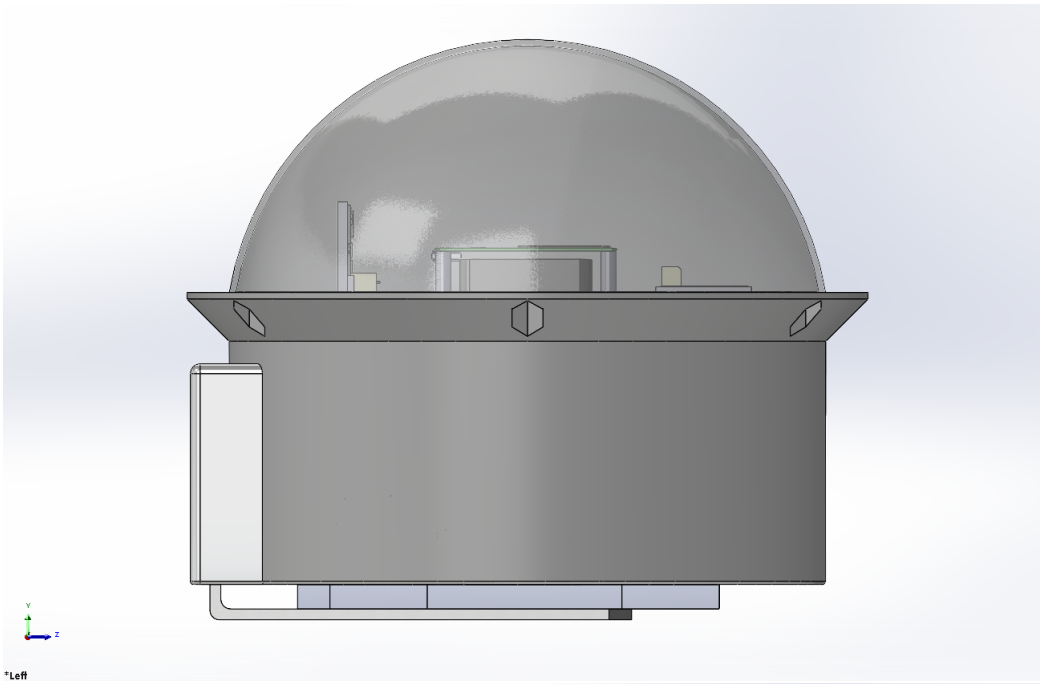


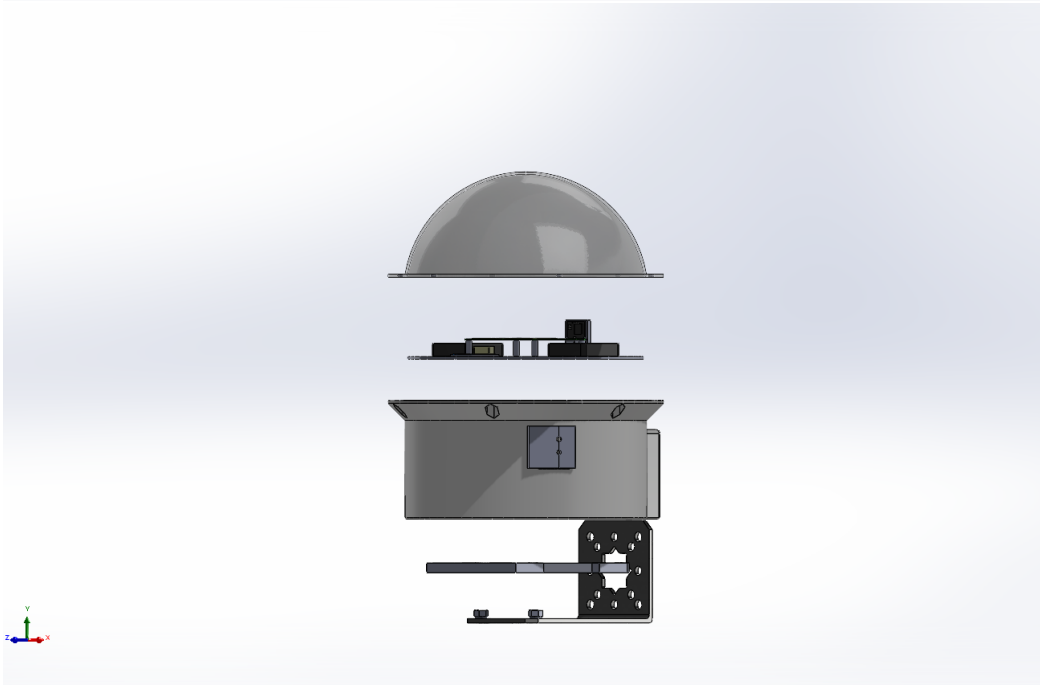


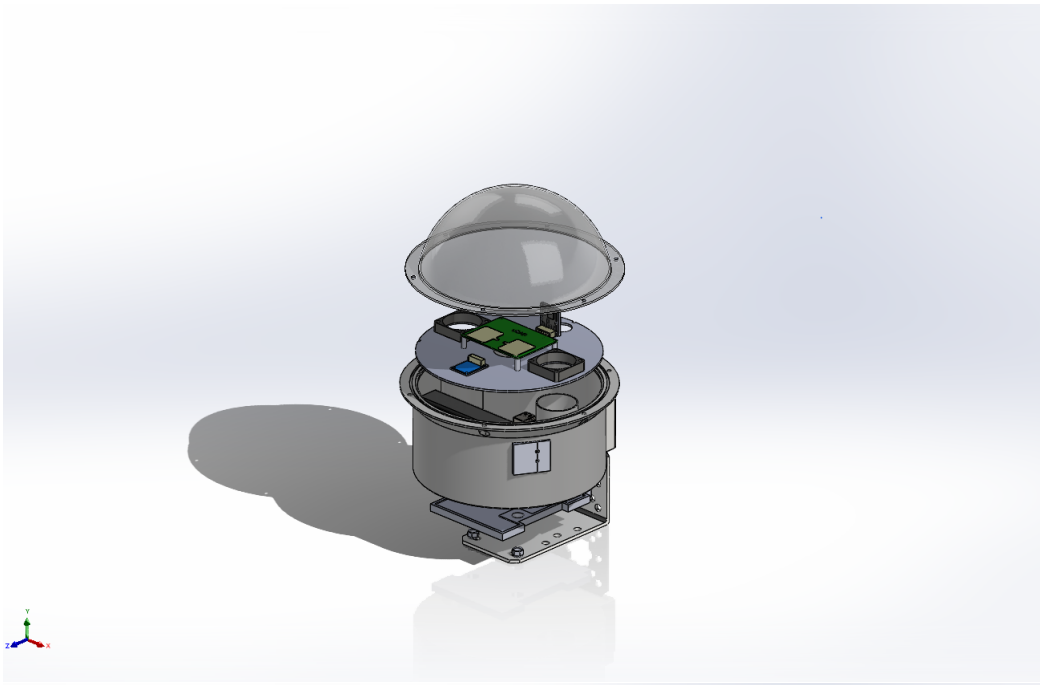
A.2 Stationary Variant

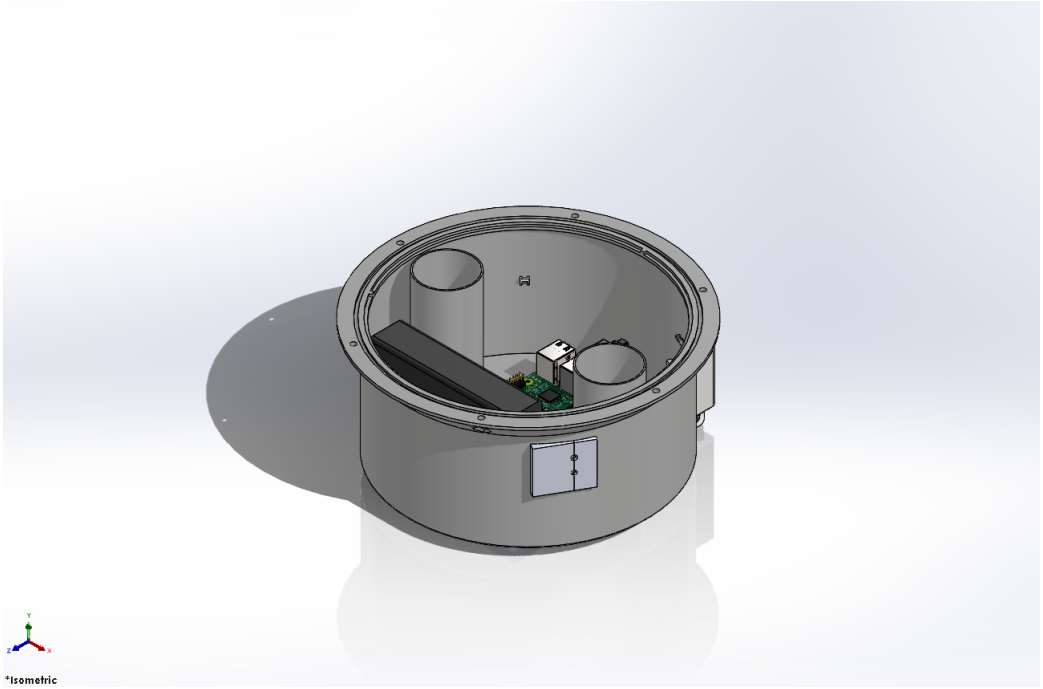




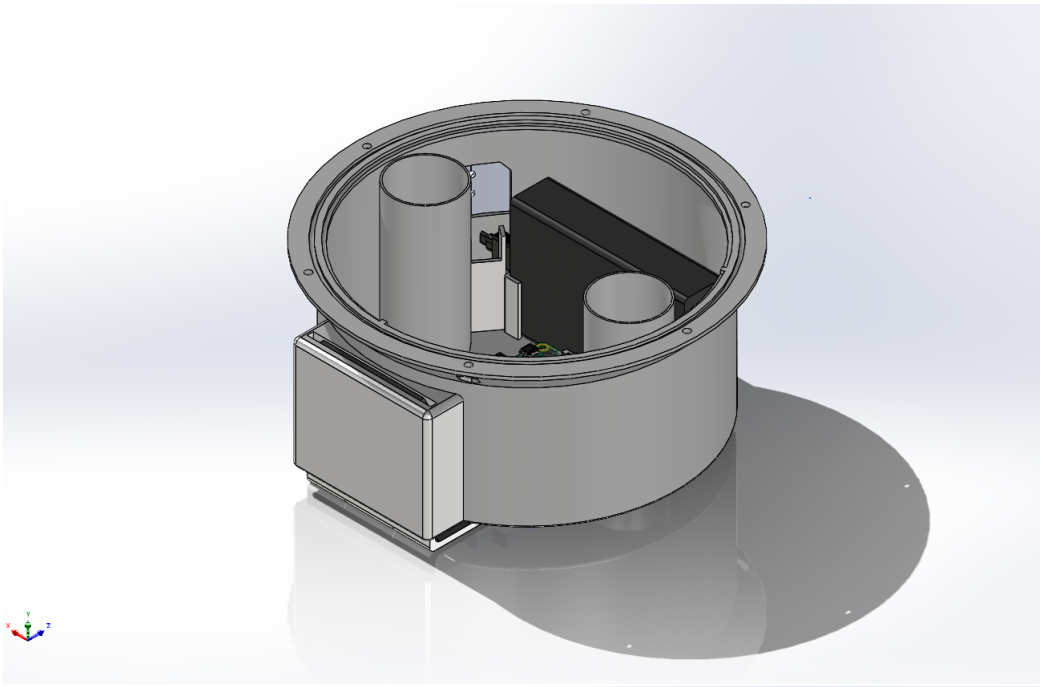

*Left

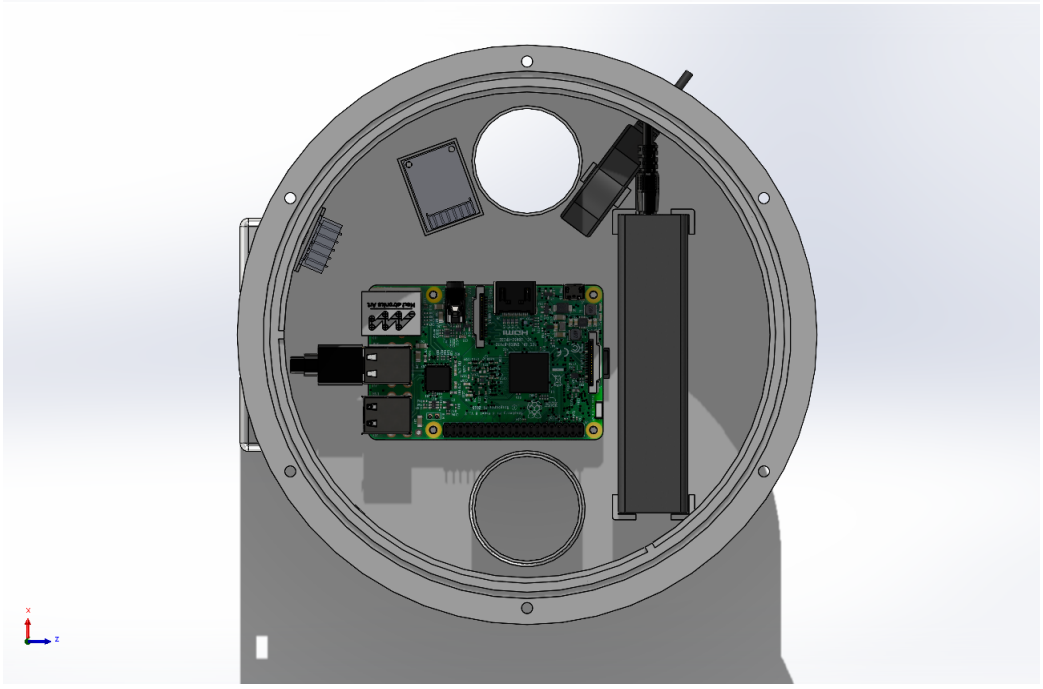






*Isometric





Revision History

Revision	Date	Author(s)	Description
1.0	09/03/2024	HW	Initial document upload
1.1	10/01/2024	HW	Future direction updates