Instrumented Ground Level Observer Of Hail (IGLOOH) Technical Specifications and Documentation

Howard Wen and Dylan MacPhail

August 2024

1 Introduction

The Instrumented Ground Level Observer Of Hail (IGLOOH) is a device used to collect primarily data about hail impact velocities. Equipped with an environment sensor, decibel meter, radar and real time clock, it is capable of also recording various other environmental data. With solar power capabilities, the IGLOOH can be deployed both as a stationary and mobile device.

Contents

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, a BME280 from Waveshare to record temperature, relative humidity, and air pressure, and the IGLOOH has a uRAD radar to measure hail velocities in the Z direction.

Dependent on method of deployment (supersite vs mobile), IGLOOH has a variety of different configurations. When deployed at a supersite for long durations, it relies on a solar panel to recharge the power manager battery. 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 with a waterproofed port for the two solar panels to enter from as well as two fan pipes to allow better 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.

Deployment mainly occurs like so:

1. Mount and connect all sensors and wires
2. Turn on Waveshare and ensure RPi built-in LEDs are on
3. Put on lid
4. Turn on Logitech keyboard and press `Ctrl + Alt + T` to open up the terminal
5. Type `sudo ./mobile.sh` or `sudo ./iglooh.sh` depending on the usage
6. Wait to see if red LED turns on
7. Close and screw in dome
8. Leave for deployment either at Supersite or in the hail swath (path)

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 currently uses a 7.5” acrylic flanged dome. The enclosure that holds the RPi and peripherals is 3D printed with either PLA or PETG. The enclosure 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.

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. 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 Schematic and Pinout

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.

4.1.3 Design for manufacturing

3D printing limitations - no overhangs, material strength, enclosure size

4.1.4 Design for assembly

- solar panel cables, screws, wiring

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

Vibration of fans

5.2 Electronics and Software

Wireless - Bluetooth - Wifi

5.3 Deployment

5.4 Dome Capabilities/death

6 Future Direction

6.1 Dome

Diff materials - polycarbonate, different shapes

6.2 ESP32 Integration

More lightweight and power efficient, designed for IoT

6.3 Wireless Enabled

Mobile app/website to interface with IGLOOH

6.4 Remote Capabilities

4G enabled to be stationed at sites

Revision History

Revision	Date	Author(s)	Description
1.0	08/25/2024	HW	Initial Document Upload