Calculating Orientation and Measuring Pointing Angle for Scientific Systems (COMPASS)

HASP 2021 Final Flight and Science Report

Louisiana State University

Jeanne Garriz, Sabrina Huezo, Harrison Gietz, Jesse Frank, Aaron Ryan

Table of Contents

1. Abstract

Observations of the Sun during a high-altitude balloon flight, such as during a total solar eclipse, must consider factors such as the position of the Sun in the sky for the duration of the flight, the angular sensitivity of instrument solar detectors, and potentially large rotational and pendular fluctuations in payload orientation. In 2021, a student-led team developed an orientation system called COMPASS (Calculating Orientation and Measuring Pointing Angle for Scientific Systems) for flight on the High-Altitude Student Platform (HASP). The goal of COMPASS is to determine the orientation of HASP using two independent systems. The first method uses an accelerometer and magnetometer system. The second system is a camera array to capture the image of the sun and determine the orientation of the payload from the observed sun position. This camera system consists of two Arducam cameras with wide angle lenses used to cover a total horizontal angle of ~190° and total vertical angle of ~60° attached to a Raspberry Pi. The performance of these cameras as an orientation system will be compared with the results of the magnetometer/accelerometer system to ascertain whether these cameras are a viable method for determining the orientation of a balloon payload.

2. Goals and Objectives

2.1. Mission Goal

The goal of COMPASS is to determine the orientation of a payload in space by measuring the payload's magnetic heading, acceleration, and verifying these measurements with an array of cameras using the Sun as a reference. These measurements were taken at float altitude of around 120k feet. Determining the direction in which a payload points is important for making measurements using field of view dependent sensors when the sensor placement is not fixed, such as on a future balloon payload.

2.2. Objectives

2.2.1. Scientific Objectives

1. Measure azimuth and elevation of the payload's pointing vector.
2. Verify azimuth and elevation measurements using an array of cameras with the Sun as a reference.
3. Determine right ascension and declination of the payload's pointing vector.

2.2.2. Technical Objectives

1. Adhere to HASP physical requirements.
2. Interface with HASP power system before and during flight.
3. Interface with HASP control/communication system before and during flight.
4. Process sensor and camera data with a microprocessor.
5. Store data on an SD card for analysis after flight.

3. Background

3.1 Scientific Background

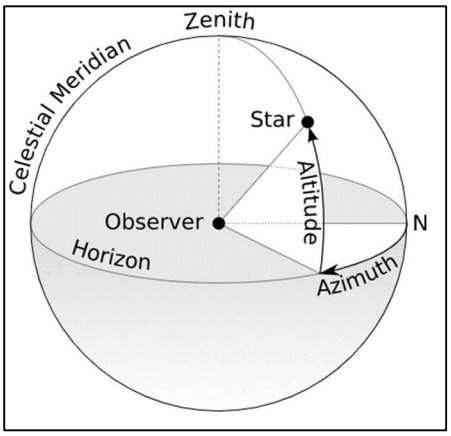


Figure 1 This figure shows what the azimuth and elevation mean physically.

The azimuth is defined as the angle between the magnetic north vector and the observed object's vector on the horizontal plane. In the case of this payload, it would be the angle between north and the direction in which the payload points. Azimuth can be measured with a magnetic compass (magnetometer) by relating the measured magnetic field readings to the Earth's magnetic field to find a relative angle. The elevation (or altitude) is the angle between an object (or in this case the payload pointing vector) and the observer's local horizon. Elevation can be measured with a tilt sensor/accelerometer, using the Earth's gravity as a reference. These two measurements are to describe where the payload is pointing in the Earth's reference frame. Right ascension and declination can also then be calculated using the orientation of the payload, latitude, longitude, and time of observation.

To verify whether the readings from the magnetometer are correct, an array of cameras placed on the outside of the payload were used to orient the payload with respect to the Sun and the Earth's reference frame. These cameras cover the total range of the Sun's motion relative to the payload from an early estimate of float time to sunset. This results in an estimated total change of 190 degrees in azimuth and 50 degrees in elevation. Snapshots from these cameras and their timestamps are then be sorted, and pictures not containing the Sun are be disregarded. The rest were run through image processing software to pinpoint the location of the Sun relative to the payload and relative to the Earth at different times and will be compared to the magnetometer readings at these times. The field of view and sensor orientation issue arose in a previous LaACES sounding balloon payload called Student Payload for UV Detection (SPUD), designed and built by undergraduates during the spring semester of 2020. SPUD used UV sensors to measure UV and ozone in the atmosphere. The UV sensor response was dependent on the angle at which the radiation from the Sun hit the sensor. A magnetometer was placed inside the payload with the intention of using the magnetic field readings to determine the payload's orientation and solar incidence angle at any given time and therefore correct the UV irradiance readings. However, this did not work as planned due to what the team believed to be unforeseen electromagnetic interference on the magnetometer from the payload electronics.

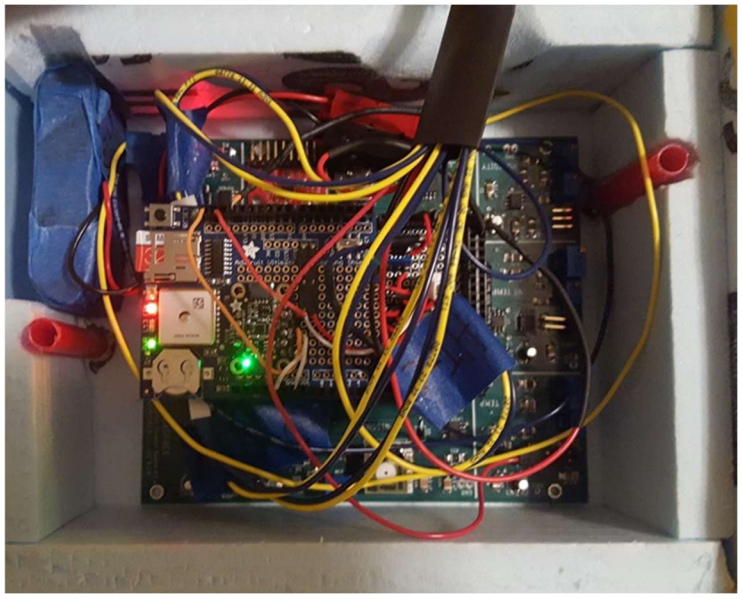


Figure 2 This figure shows the interior of our previous LaACES payload - SPUD.

COMPASS' use of an orientation system (magnetometer and accelerometer) placed on a boom away from the payload, as well as one inside the payload, will be able to provide insight into the functionality of a magnetometer when used for scientific ballooning applications. Specifically, COMPASS will provide information on the use of a magnetometer to determine the azimuth and elevation at which a payload points. The "front" of the payload was defined to be the -x direction of the magnetometer. To verify that these magnetometer readings are correct, an array of cameras placed around the payload took take snapshots during flight to orient the payload with respect to the Sun.

3.2. Technical Background

Because a future goal of such an orientation system would be to ascertain when detectors are pointed at the Sun, we would like to determine the azimuth and elevation of the payload's pointing vector within half a degree of the Sun's angular size (0.5 degrees). Therefore, our desired accuracy is 0.25 degrees. This is required because COMPASS' verification system uses the Sun as a reference. These magnetometer measurements and camera snapshots were taken once HASP reaches float altitude and is stable. Ascent and descent were considered because the motion of the payload during these time periods is too erratic to take measurements within the specified accuracy. To take measurements at roughly every $1/8^{th}$ (0.125) degree interval (half the required accuracy, for thoroughness and redundancy), the payload needed to sample data twice every second at float. This sampling rate was determined by estimating the rate of rotation of HASP from previous flight videos to be 0.23 degrees/second at float.

$$0.23\frac{degrees}{second} * \frac{1}{0.125\ degrees} = \frac{1.84}{second} \approx 2/second$$

Because of the importance of cameras in this payload, a Raspberry Pi was used as the microprocessor because of its many compatible camera options and full camera library.

4. Payload Design

4.1. Principle of Operation

COMPASS determines the orientation of a balloon payload in space by measuring the payload's magnetic heading with a magnetometer and the payload's tilt with an accelerometer. These measurements are verified using an array of cameras on the main payload to take snapshots of the Sun in real time and downlink them to HASP. These snapshots are compared to the orientation data that can be correlated with tracking the position of the Sun. To minimize electronic interference between the magnetometer and the rest of the electrical components, one of the magnetometers was placed on a boom situated away from the main HASP gondola. COMPASS also contains multiple temperature sensors to monitor the temperature of electrical components during flight. All sensor and camera data were processed using a Raspberry Pi on the main payload. The data from the sensors on the boom was processed through an Arduino Mega, which converts the signals from analog to digital and then send the serial communication to the RPi on the main payload.

System Design Diagram

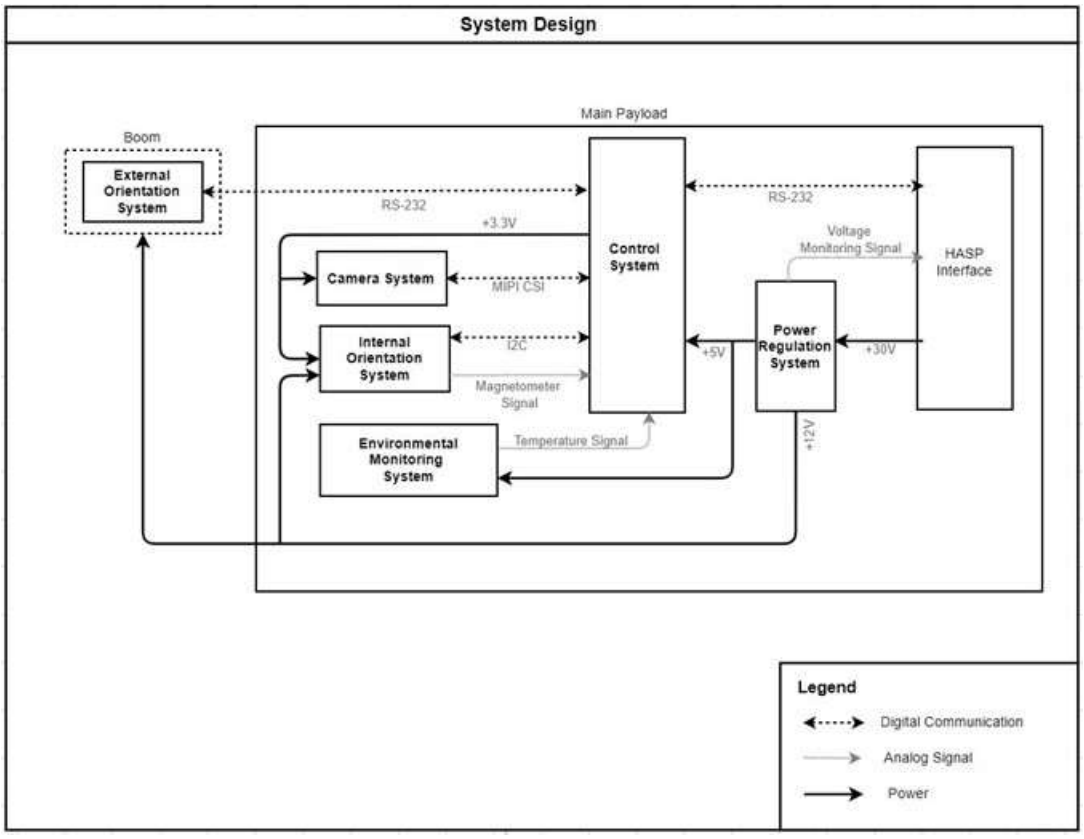


Figure 3 This diagram shows the main subsystems that make up COMPASS.

4.2. Electrical Design

4.2.1. Sensors

4.2.1.1. Internal Orientation system

This system measures the azimuth and elevation of the payload. This orientation module contains a 3-axis magnetometer used to measure a magnetic field vector at each timestamp. Given that we know what Earth's magnetic field is, we can compare this to our magnetic field readings to determine the azimuthal direction of the payload in space. Based on the pointing accuracy requirement, we were able to determine that the magnetometer shall have an accuracy of about +/-0.1uT. This system also contains a 3-axis accelerometer which is used to measure the elevation of our payload. The accelerometer allows us to detect which direction our payload is pointing down towards Earth which can be used to determine the angle of elevation.

After testing multiple magnetometer options, the Bartington Mag648 magnetometer produced the most consistent results compared to a calibrated lab magnetometer. This procedure involved supplying direct current to a Helmholtz coil to produce a magnetic field. The expected value of this magnetic field was calculated, then compared to the actual value read by the magnetometer (minus the background field measured by the lab magnetometer). Overall, the Mag648, consistently gave us results close to what we expected to measure. The Mag648 was supplied 12VDC by the COMPASS power regulation system and has a range of ±100uT that reads out 6 analog signals to ADC1 (MCP3208) of the control system. This orientation system also contains a LSM303AGR accelerometer that has a selectable range from +-2g to +-8g that measures acceleration in 3-axis and communicates with the Raspberry Pi via I2C communication.

4.2.1.2. External Orientation System

The External Orientation System is mounted on a fiberglass boom away from our main payload (and the HASP gondola) to avoid possible electronic interference. This orientation system includes its own magnetometer, accelerometer, 12-bit ADC, +5V reference, RS-232 Level Shifter, and Arduino Mega for processing data. This system communicates with the Raspberry Pi on the main payload via a DB-9 serial cable that runs from both RS-232 level shifters (one on the main payload and one on the boom).

Mounting the module far from the microcontroller can cause potential communication issues, because sensors cannot be directly read over such long distances. RS232 communication is used to communicate such long distances. However, because the accelerometer on the External OS requires communication via the I2C bus, there would have to be some way of converting from I2C to TTL-UART, then to RS232 before it can communicate with the RPi4 on the main payload. We also require a way to convert the magnetometer and environmental monitor analog signals to digital signals before it can communicate with the RPi4. Our solution to solve both these problems was to add a 12-bit ADC, an Arduino Mega microcontroller, and

an RS232 level shifter to the boom that can read the detector with an ADC, and then convert the digital signal to RS232 before it can be transmitted to the control system on the HASP payload plate.

We cannot use the 10-bit ADC on the Arduino to read in the magnetometer signal because this does not fulfill the accuracy needs for the magnetometers. To satisfy this requirement, we use a 12-bit ADC between the magnetometer and the SPI digital pins on the Arduino to read detector data and convert the analog signals to digital signals, then to TTL-UART. The Arduino is used to convert the accelerometer I2C signals to TTL-UART.

4.2.1.3. Camera system

An array of two cameras takes simultaneous snapshots of the Sun to verify the measurements taken by each of the payloads orientation systems.

These cameras cover an angular area of at least 190 degrees in azimuth and 50 degrees in elevation, which is the range of motion of the Sun relative to the payload during float and have a max angular error of <1/4 degree. This camera array takes snapshots at a rate of around 1 picture per minute.

Images are stored on the SD card and processed post-flight to determine the location of the payload in the Earth's reference frame, and then these results are compared to magnetometer readings at the same timestamp to verify magnetometer measurements.

The camera system used is the Camarray-Arducam 1MP Stereoscopic Camera Bundle Kit. With an M40160 M12 lens, these cameras have a 118-degree horizontal and 92-degree vertical field of view. It attaches directly to the RPi camera module, communicates through serial communication, and has an input voltage of 3.3V, which was supplied by the Rpi.

4.2.1.4. Environmental Monitoring System

This system includes internal and external temperature sensors to help monitor the health of main electrical components. Three temperature sensors are placed inside the main payload box and two are placed inside the electronics box on the boom.

We used AD22100 temperature sensors that have a 200° C temperature span, where the output voltage is proportional to the temperature varying from 0.25V at -50°C to 4.75 at 150°C at a single +5V supply. Though we do not expect to see temperatures as high as +150°C, no attenuation circuity is needed for the application of these sensors unless we want very precise temperature data, which in this case is not of high importance. These sensors also feature built in signal conditioning designed to eliminate the need for linearization circuitry or trimming. In addition to the built-in signal conditioning, the temperature monitors also include external signal conditioning including filtering out high frequency noise and analog-to-digital signal conversion.

4.2.2. Sensor Interfacing

The main system that required sensor interfacing was the Environmental Monitoring System. This system contains five temperature sensors (AD22100s) that are used to monitor the internal temperature of the payload for each of the main electrical components. Each sensor was connected to an RC low pass filter between its output voltage pin and a designated analog input pin of ADC2 (MCP3208). This low pass filter is simply a technique used to reduce noise by attenuating high frequency signals. The ADC then converts the input analog signal to a digital signal to be read by the Raspberry Pi (Refer to control electronics section for all ADC pin connections).

The Mag648 magnetometer also requires analog-to-digital conversion using the MCP3208 ADC. Other than this, no other signal conditioning is needed with the magnetometers.

4.2.3. Control Electronics

The central component of the COMPASS control system is a Raspberry Pi 4 Model B (8GB RAM) microcontroller. The RPi4 has various interface options responsible for communicating with each of the major system components to receive and transmit data. In addition, there is another microcontroller, an Arduino Mega 2560, on the boom that is responsible for collecting data from the external orientation system components and communicating this data to the control system on the main payload using the necessary serial communication.

Also included in the control system are two 12-bit ADCs (MCP3208), each with up to 8 input channels for analog-to-digital conversion between the Rpi and any analog devices used. There is also an ADC level shifter connected to the output of the ADC to step down from 5V to the RPi's 3.3V logic. There is also an onboard SD card to store data as well as two RS-232 level shifters to convert between RS-232 and CMOS-UART logic for communications with the HASP interface and the electronics on the boom.

The last component of the control system is the NEO-M9N GPS Module by Sparkfun which can operate at a maximum of approximately 262,500ft and can also mate with the Sparkfun SMA GPS antenna that includes an onboard low noise amplifier. To use this GPS module, we drilled a hole through the payload cap so the antenna can be mounted outside of the payload box. The onboard GPS was sampled at the same rate as the magnetometer to correlate GPS and magnetometer readings. In addition, we planned on requesting HASP GPS data periodically (at a slower rate than the onboard GPS) for redundancy.

4.2.4. Power

COMPASS is supplied ~+30V by the HASP interface system and has its own power regulation system to regulate the required voltage going to each of its components. On the bottom shelf of the main payload, the power subsystem consists of two DC-DC converters that step down the +30V into two output voltages, one to +5V and the other to +12V. The +5V line powered the temperature monitors on the same shelf, as well as the USB-C breakout board that supplies +5V to the RPi. The +12V line directly powers the Arduino (on the boom), both magnetometers (one on the same shelf and one on the boom), as well as the +5V reference chips (one on the boom and one on the top shelf).

The RPi4 on the top shelf of the main payload is supplied +5V from a USB-C cable connected to the USB-C board from the bottom shelf. The RPi has its own voltage regulation system including +5V and +3.3V voltage regulators. The +5V line provides power to both ADCs, the 5V-3.3V logic level shifter and the MAX232 level shifter. The +3.3V line supplies power to the accelerometer, camera system, GPS, NS-RS232 level shifter, and the 5V-3.3V logic level shifter.

In addition to the +5V reference and magnetometer on the boom, the Arduino Mega is powered by +12V line running from the main payload to the boom. The Arduino also has its own onboard +5V and +3.3V voltage regulators that power +3.3V to the accelerometer, and +5V to the temperature monitors, ADC, and MAX232 level shifter.

The P78xx-2000 series offers step-down switching regulators at +5V/+12V that can operate at a maximum input of 36Vdc and supply up to 2A of output current. We currently have access to a custom converter board, the “Murata DCDC,” that is designed for the Murata LM7805 and LM812 converters. However, they do not supply enough current output that we need to satisfy our power requirements, so we switched these out for the P7805 and P7812 converters that meet out power requirements.

To ensure that the DC-DC voltage converters operated properly in a low-pressure environment, vacuum testing was conducted on both the 12V and 5V converters. Based on the expected current draw of the payload components, a unique resistive load was applied to either converter for 30 minutes while they were under vacuum conditions. This was to ensure that the converters did not overheat, as well as to get an idea of how vacuum conditions affect their efficiency. From this testing, we found that the converters operated for the whole duration without overheating, even while bearing loads larger than what the payload components provide. However, a decreased converter efficiency (compared to the datasheet) was recorded in the vacuum chamber; it was found that the 5V and 12V converters had efficiencies of 80% and 86% respectively when using the expected payload power consumption.

One issue we encountered with the power design, was that the maximum start-up current specified in the datasheet for the Bartington magnetometers is 400mA, which is just barely under the maximum current limit our payload can exceed. Thus, we had to test what the

actual start up current of these magnetometers were to make sure they meet the requirements of our total power budget. To test the inrush current, we simply connected a 5ohm resistor between a +12V power supply and the voltage input of the magnetometer. Using an oscilloscope, we measured the voltage drop across the resistor immediately after we flipped the power on to the magnetometer. The measured voltage drop across the resistor was about 100mV, which (using Ohms law) means the startup current was about 20mA. This value is very low compared to the maximum current draw specified in the datasheet therefore we were able to conclude that it won't have a damaging effect on our payloads total power consumption. The steady state current draw of the magnetometer was also measured, but by using an ammeter instead of an oscilloscope. The current draw came out to be as little as 3uA which did not have much of a significance to our power budget. This board also includes voltage monitoring circuits that communicate with the analog pins on the EDAC pin to make sure the converters are providing stable power.

There is also a series of logic level N-channel MOSFETS (PSMN022) that controlled the powering on/off of different electronic components. MOSFETs are actually just transistors that act as power 'switches' and were commanded by the control system to switch power on or off to the designated electrical component when/or if necessary.

4.3. Software Design

4.3.1. Flight Software

Because COMPASS utilized two separate microcontrollers, these microcontrollers both had their own software routines. However, they also needed to be synced up and made to take data simultaneously. The following two sections detail how the in-flight software worked on both the main payload and the boom payload.

4.3.1.1. Main Payload

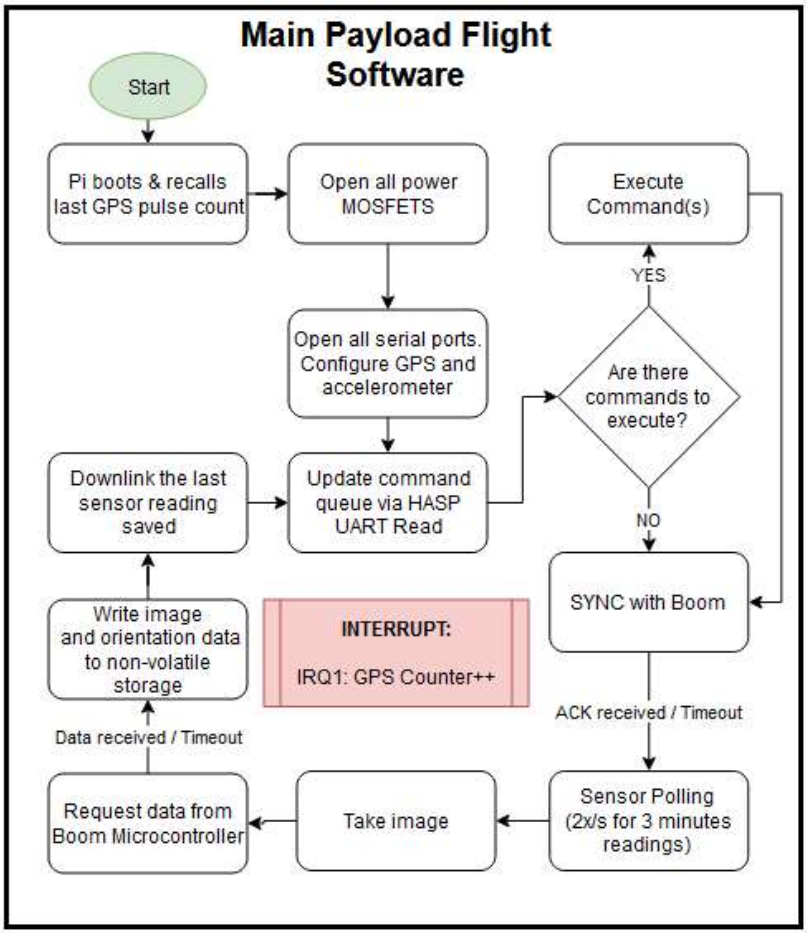


Figure 4: : High-level diagram of the main payload flight software on the main payload Raspberry Pi. Detailed block diagrams of each subroutine are found in Appendix C.

Once the main microcontroller has booted, all devices controlled by the power MOSFETS are activated (GPS, magnetometers (x2), and Boom microcontroller (Arduino Mega)). Serial ports are also opened, and the GPS module is configured to output GGA and RMC strings. The accelerometer was also configured via I2C. At this point, the main microcontroller finishes start-up procedures and enters its main loop.

In order to maintain absolute timing accuracy relative to GPS time, the main microcontroller maintains an interrupt that counts the number of pulses received from the GPS module. Once a satellite lock is attained, the GPS module begins outputting a very accurate (+/- 60 ns) 1-second pulse. The rising edge of this pulse triggers the above interrupt and increment the pulse count by 1.

The main microcontroller begins the top of its loop by reading the contents of the UART for the HASP serial connection. This subroutine checks if byte characters are available in the UART and appends them to an empty command string. This action continues until either a line

feed <LF> character is reached (signaling the end of the command), or the subroutine has gone on for a reasonable amount of time without receiving a new character and times out. Completed commands are added to the commanding queue and will be executed in the next step. If there are commands in the commanding queue, the Command Execution subroutine will attempt to execute them until there are no more queued commands. First a command is extracted from the queue and is verified to be a complete and valid command. Once verified, the command will be compared to commanding table (vector table) and used to look up to matching function call. Once the matching function is executed and presents a return value, the command status will be updated with either “DONE” (successful execution) or “FAIL” (execution error or failure. Once the commanding queue is empty, the main microcontroller approaches the synchronization phase.

To ensure the timing accuracy of 0.5s, a synchronization between the main and boom microcontrollers is necessary. First a “SYN” request consisting of the following:

SYN<Unique Identifier (Current GPS Pulse count)>,<system time>

is sent from the main microcontroller. Upon receiving the synchronization packet, the boom microcontroller updates its GPS pulse count and records the synchronization attempt; responding with an acknowledgment packet, which consists of the following:

ACK<Unique Identifier (Matching GPS Pulse count),<local millisec.>

Upon proper receipt of the matching acknowledgement to the synchronization attempt, the main microcontroller enters the sensor polling phase. However, if an incorrect ACK is received, or the synchronization attempt times out without receiving an ACK, the main microcontroller attempts to synchronize again. If the synchronization fails 10 times, given the simplicity of the network, it is reasonable to assume that the synchronization is currently impossible. The main microcontroller must move on to the sensor polling phase. The Boom microcontroller times out at approximately the same time and begin its own asynchronous sensor polling.

The main microcontroller has a 180s (3-minute) sensor polling phase. At a rate of 2 polls per second, this amounts to 360 data entries. At the beginning and end of each sensor poll, the start and end time are recorded. These are used to calculate how long the routine should pause between each sensor polling, such that each poll occurs 500 milliseconds apart. This also helps ensure that each sensor poll was in fact less than 500 milliseconds long. The GPS pulse count is updated by the GPS Pulse Counter interrupt subroutine while the latitude, longitude, altitude,

and timestamp are gathered. At the end of the 360 data entries, the main microcontroller goes on to take an image. An image is taken at the end of every 3 minutes polling phase. The beginning and end time of the picture are recorded to ensure the timing accuracy of 0.5s. This image should be taken 0.5s of the last sensor poll.

To verify that the boom microcontroller is functioning and recording data as expected, the main microcontroller was scheduled to request the last data entry recorded by the boom microcontroller. A data request is sent, and if the request receives no reply or a garbled reply, will re-send the data request. Given the simplicity of the network, if a data request fails 10 times, the main microcontroller gives up on requesting data during the current call to the data request subroutine.

The data up to this point is currently stored in volatile memory. To preserve the data through failure or shutdown, the data must be written to non-volatile storage. The reason for waiting to write data to storage until this moment was to ensure the timing accuracy. The files are created with the appropriate names. Once saved, the last data entry containing data taken by both main and boom microcontrollers is copied to the variable “data_to_downlink” and passed to the downlink subroutine.

The last phase in main loop of main microcontroller is to transmit the latest orientation data recorded. This telemetry packet is used to verify that all components are working as intended and that all command received were executed properly. More detailed block diagrams of each individual subroutine are included in Appendix C.

4.3.1.2. Boom Payload

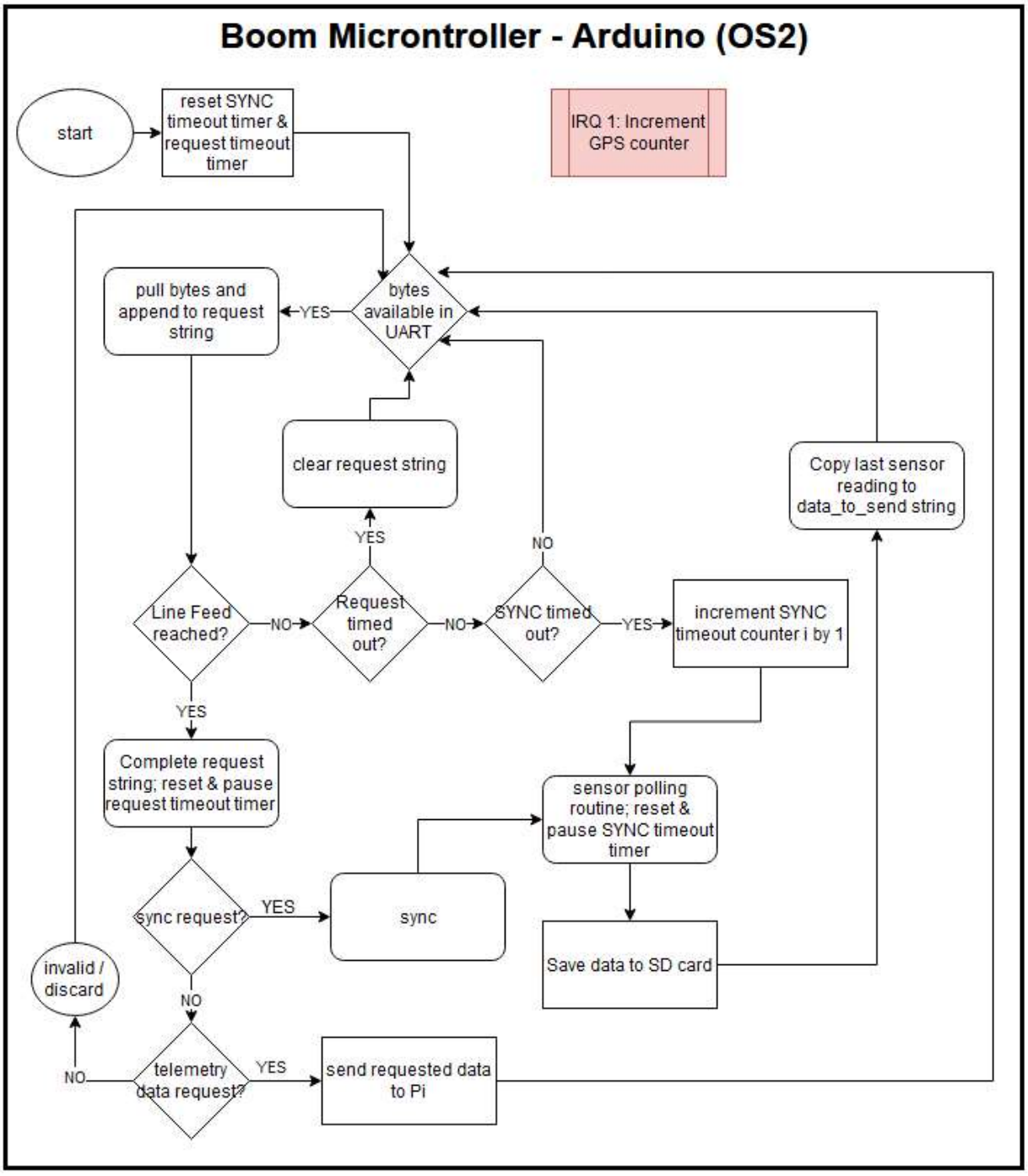


Figure 5: High-level block diagram of software running on boom payload during flight.

The main function of the boom microcontroller is to wait for a synchronization attempt and respond with an acknowledgement. After waiting an appropriate amount of time, if the boom microcontroller fails to receive a sync attempt, it "assumes" that the serial connection to the main payload has failed and thus, will go on to the sensor polling routine. At a rate of 2

sensor polls per second over a 180s polling period, this amounts to 360 data entries. At the beginning and end of each sensor poll, the start and end time are recorded. The start and end time are used to calculate how long the routine should pause between each sensor polling, such that each poll occurs 500 milliseconds apart.

4.3.2. Payload Commands

The main types of commands created for COMPASS were commands used to power on and off different components in case they malfunctioned during flight and needed to be reset. The commands used are shown in the table below, and more detailed flowcharts of the command subroutines are included in Appendix C.

Instrumentation Command Bytes			
Command Group	**Command**	**Hex Value**	**Description**
Power	Main Magnetometer (OS1) ON	0x11	Turns power to the OS1 Magnetometer ON
	Boom Magnetometer (OS2) ON	0x12	Turns power to the OS2 Magnetometer ON
	Main GPS ON	0x16	Turns power to the payload GPS ON
	Boom microcontroller ON	0x17	Turns power to the Arduino ON
	Main Magnetometer (OS1) OFF	0x19	Turns power to the OS1 Magnetometer OFF
	Boom Magnetometer (OS2) OFF	0x1A	Turns power to the OS2 Magnetometer OFF
	Main GPS OFF	0x1E	Turns power to the payload GPS OFF
	Boom microcontroller OFF	0x1F	Turns power to the Arduino OFF
Resets	Magnetometer (OS1) Reset	0x30	Powercycles the OS1 Magnetometer MOSFETS
	Magnetometer (OS2) Reset	0x31	Powercycles the OS2 Magnetometer MOSFETS
	Payload GPS Reset	0x35	Powercycles the GPS MOSFETS
	Boom microcontroller Reset	0x36	Powercycles the Boom microcontroller

Table 1: Commands used to change the state of sensors on COMPASS in flight.

4.3.3. Data Storage and Downlink

Both microcontrollers used included their own SD cards used to store data taken during flight, and example data entries are shown in Appendix J. Sets of 360 of these entries were taken and then recorded on each SD card.

The information downlinked every 3 minutes was essentially the boom payload data entry at the end of a polling session appended to the main payload data entry at the end of a polling session. An example of downlinked data is also shown in Appendix H.

4.4. Mechanical/Thermal Design

4.4.1. Main Payload

The main payload housing was designed to contain and protect all internal components of the payload from atmospheric conditions. The payload frame was constructed of 20mm extruded aluminum, and the walls of the payload were made of aluminum sheets that were painted white for thermal regulation. The payload cap was also designed as a mount for the two cameras that would be used to image the Sun, and this mount was 3D printed out of ABS plastic. The exterior of the main payload is shown in the figure below.

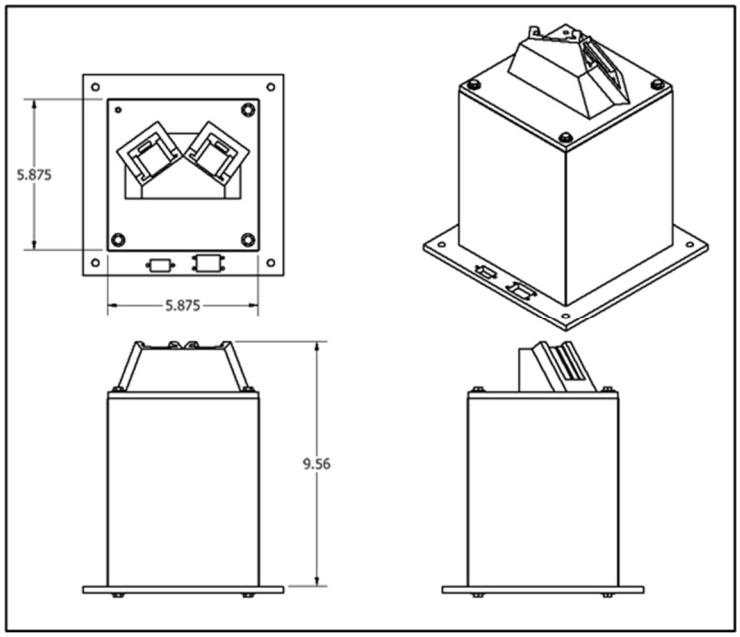
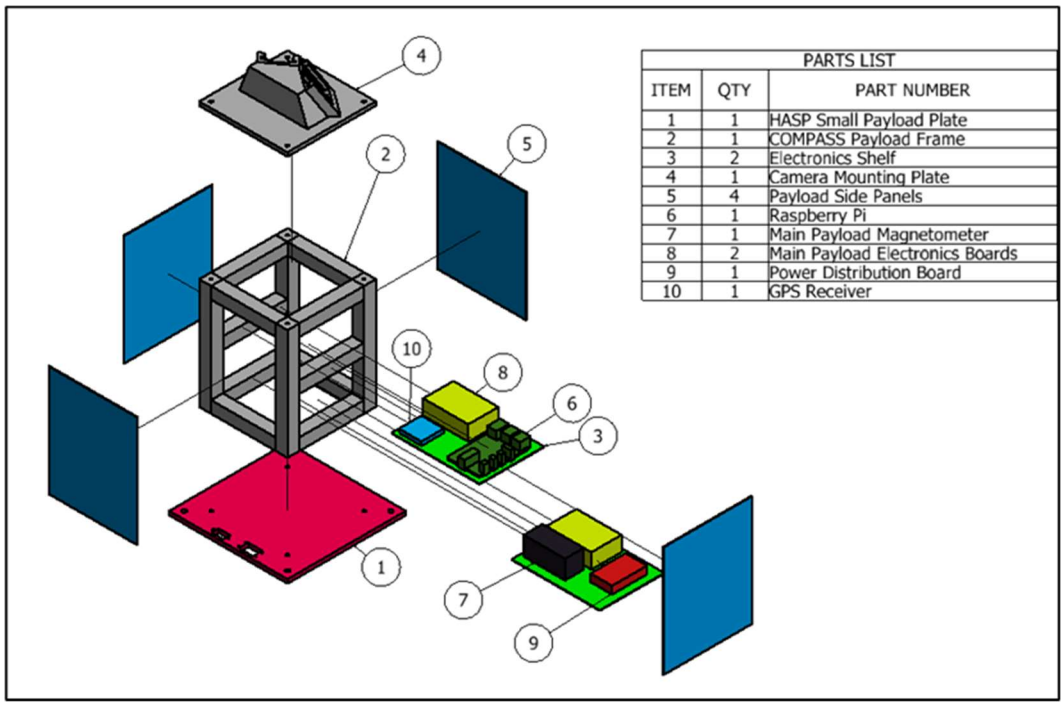


Figure 6: Dimensions and design of main payload exterior. Shown mounted to payload plate.

The main payload interior consisted of 2 shelves that contained the electronics and sensors of the payload. The top shelf included the Raspberry Pi and the attached cameras, which were fed through the payload cap into the mount. The interior structure of the payload is shown below. More detailed drawings of specific parts of the main payload design are shown in Appendix D, and a detailed weight budget of both payloads can be found in Appendix E.



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	HASP Small Payload Plate
2	1	COMPASS Payload Frame
3	2	Electronics Shelf
4	1	Camera Mounting Plate
5	4	Payload Side Panels
6	1	Raspberry Pi
7	1	Main Payload Magnetometer
8	2	Main Payload Electronics Boards
9	1	Power Distribution Board
10	1	GPS Receiver

Figure 7: Interior structure of the main payload that shows the shelves the arrangement of electrical components on these shelves.

4.4.2. Boom Payload

The boom payload was offset from the HASP gondola using a 96” fiberglass boom. More detailed diagrams and drawings of how the boom was attached to the main payload can be seen in Appendix D, and the boom stress test that was completed to ensure that the boom would survive flight is shown in Appendix F. The boom payload components were enclosed in a plastic electronics box that was attached to the end of the boom as shown in the figure below.

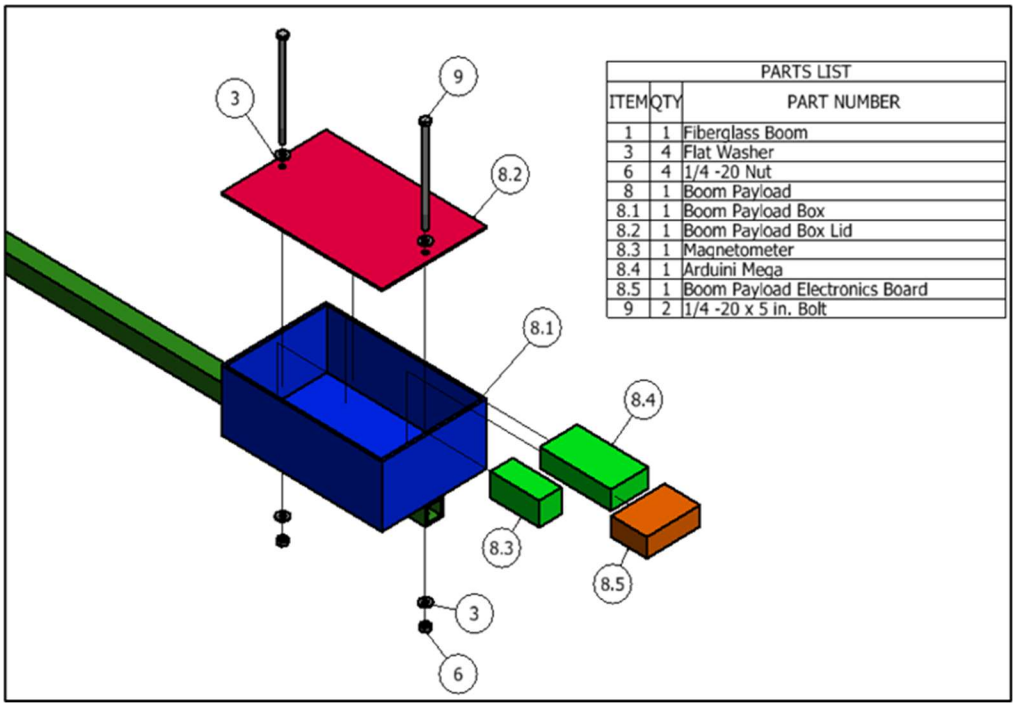


Figure 8: Interior of the boom payload and arrangement of boom payload components.

4.4.3. Thermal Design

COMPASS was designed with the expectation that the payload would encounter ambient temperatures from -30°C to 45°C according to the HASP Call for Payloads document. The different operating temperatures of the major system components were taken into account and tested to ensure that they would perform under flight conditions. In addition, the payload was constructed out of polished aluminum that was painted white on the outside to insulate the interior components. The white exterior would reflect visible light from the Sun, and the polished exterior would reflect internal heat from the components.

The Raspberry Pi 4 we tested specially in the thermal/vacuum chamber multiple times because the data sheet specified operating temperature range did not fit the range of atmospheric temperatures required. However, after multiple vacuum tests, the Pi 4 did operate as required and neither froze nor overheated and throttled CPU speeds when running a program designed to put stress on the CPU. Overall, no major issues related to thermal regulation occurred during testing or during flight.

5. Integration

On July 23rd 2021, the COMPASS team traveled to Palestine, TX to integrate the payload with HASP. During integration week, there were two thermal/vacuum tests performed to determine if the payload was ready for flight. Before being integrated with HASP for the first test, the team assembled the payload and checked that all commands worked properly. Once it was determined that the commands worked and any bugs were fixed, COMPASS was placed on HASP for the first test.

The first thermal/vacuum test took place on July 28th, 2021 and began at 9:26 am local time. During the tests, we monitored the health of the payload by plotting the downlinked data and determining if there were any issues during the tests. We also sent commands to our payload such as turning the boom payload off and then back on, turning the main payload magnetometer off and then back on, and turning the GPS off and then back on. We also tested all of the reset commands. We determined that these commands successfully worked based on what we saw in the downlinked data. The main issue we saw during the first test was that the x-component of the main payload magnetometer readings spiked seemingly at random. To fix this issue, we made sure to secure the magnetometer inside the payload and also double checked that we were downlinking and saving the correct data. The only other fix that was made between the two tests was securing the DB9 that connected the boom payload to the main payload better. The plots from these tests are included below.

Figure 9: This plot shows the magnetometer readings taken during integration on the first thermal/vacuum test day (7/28/21)

Figure 10: This plot shows the accelerometer readings taken during integration on the first thermal/vacuum test day (7/28/21)

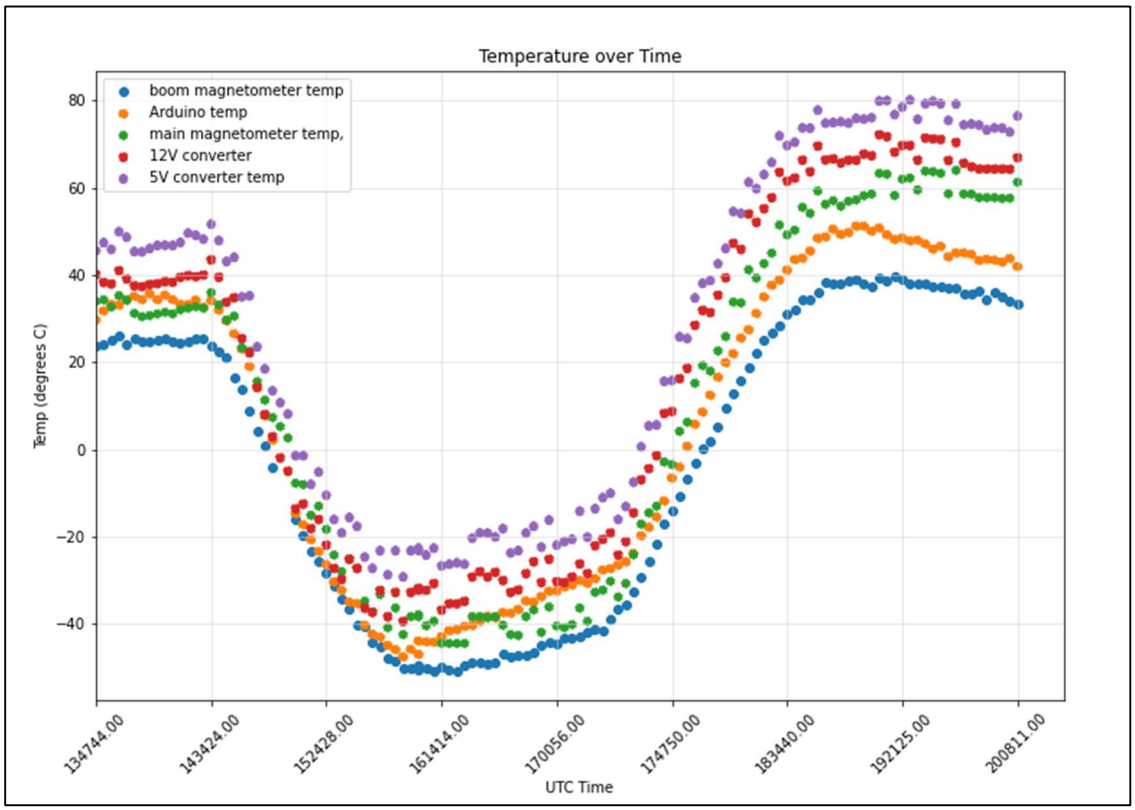


Figure 11: This plot shows the temperature data taken during integration on the first thermal/vacuum test day (7/28/21)

One of the other interesting features we saw from the plots from the first day of integration was the acceleration increasing in the x component and decreasing in the y component with what appears to be low temperatures because these times corresponded to the same times that the vacuum chamber was cooled.

During the second test on July 30th, 2021, the magnetometer did not exhibit the same issues and everything else was running as expected. However, we did still see the temperature dependence feature of the accelerometer. After the second test, the payload was packed up to be sent with HASP to Ft. Sumner, NM for flight.

6. HASP Flight

The HASP flight took place on September 14, 2021. HASP was launched at 14:03 UTC. During ascent, the boom payload SD card initialization flag showed that the SD card was not working, so the COMPASS team tried resetting the entire boom payload (using the boom reset command) to remedy this. This appeared to work for a few datasets but then it went back to saying the SD card was not initialized. The COMPASS team power cycled the payload at around 15:35 UTC to see if that fixed the SD card issue. When this did not work, the team decided to just keep running without the boom payload SD card data because boom data was still being downlinked correctly. HASP reached float at 15:53 UTC.

The GPS also stopped sending data at around 14:29 UTC, at an altitude of around 11,000 meters. The GPS kept sending time and date information, but no location or altitude information. However, HASP took GPS data during flight as well, so we still ended up with usable GPS data.

Apart from these two issues (which are discussed in greater detail in the next section), COMPASS took data and downlinked it as expected for the rest of the time it was powered on. It also continued to take images during flight. Because our payload depended on seeing the Sun, we shut it off after the Sun set because there was no point in taking data afterwards. Our payload was shut off around 2:06 UTC on September 15th, but HASP itself was not terminated until 6:03 UTC. This gave COMPASS around 10 hours and 13 minutes taking data at float, while the total HASP float time was 14 hours and 10 minutes. HASP landed in northeast Arizona at 6:46 UTC for a total flight time of 16 hours and 44 minutes.

A flight timeline is included below.

UTC Time	**Event**
9/14/21 14:03	Launch
9/14/21 14:29	GPS stopped
9/14/21 15:35	COMPASS Power off
9/14/21 15:41	COMPASS power on
9/14/21 15:53	Float
9/15/21 2:06	COMPASS power off
9/15/21 6:03	HASP flight terminated
9/15/21 6:46	Impact

Table 2: This table shows the timeline of COMPASS' flight on September 14th/15th 2021.

7. Data Analysis and Results

7.1. Image Analysis

COMPASS took a total of 250 images during the flight. The Sun was seen in 90/250 images from the right camera and 50/250 images from the left camera, for a total of 119/250 images where the Sun could be seen in either one or both cameras. The camera system took images simultaneously with both the left and right camera, but the images were exported stitched together as one file, as shown below. Because of the way the cameras had to be mounted, the images were also upside down.

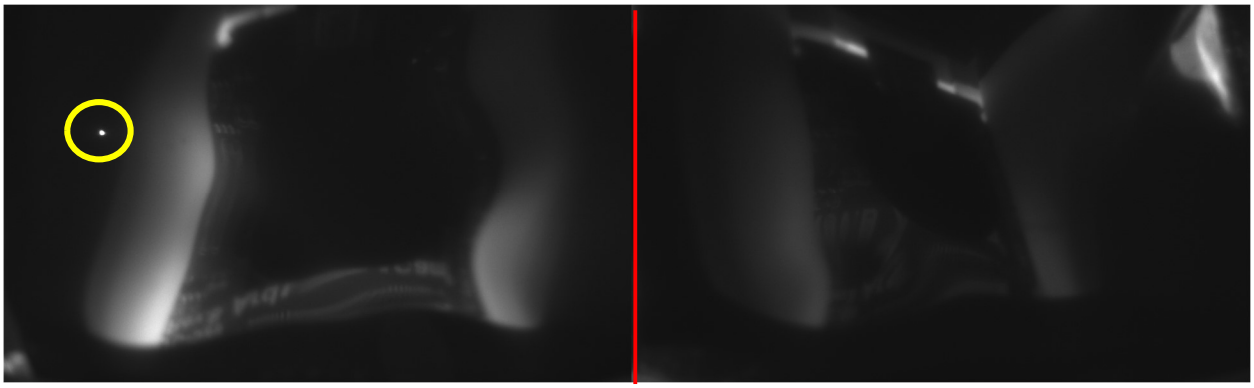


Figure 12: An example of the images taken with the cameras in flight. The red line divides the image into the pictures from the left and right cameras. The yellow circle shows where the Sun is in this picture.

In addition to the Sun, the above image contains a distorted reflection of the cameras themselves on the solar filter. This was because the solar filter was not completely flat against the filter mount. This was not a problem for our analysis however, because we were able to subtract the background of the images by altering the brightness threshold of the image and only keeping pixels that were a certain brightness (in this case, the brightness of the Sun). Once the background was subtracted, we used the Hough Gradient Method circle transform (using an OpenCV Python function) to detect circles and to obtain the pixel coordinates of the center of the Sun in these images. The pixel area of the Sun was around ~50.3 square pixels. The below images show the left half of the image before and after the background subtraction was performed.

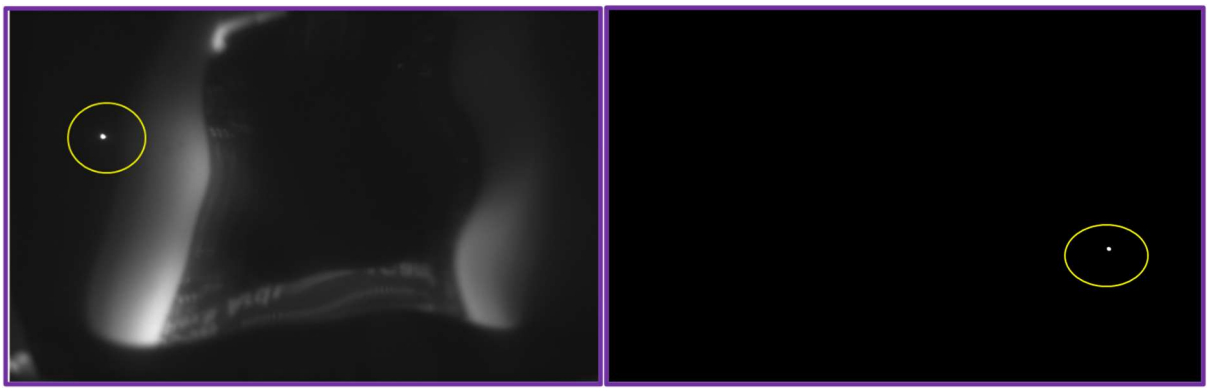


Figure 13: A camera image before and after image processing. This image processing included subtracting the background of the images and also rotating the image 180 degrees.

7.2. Raw data

The payload took data for the whole time it was on during flight. The following plots are the plots of the data downlinked during flight.

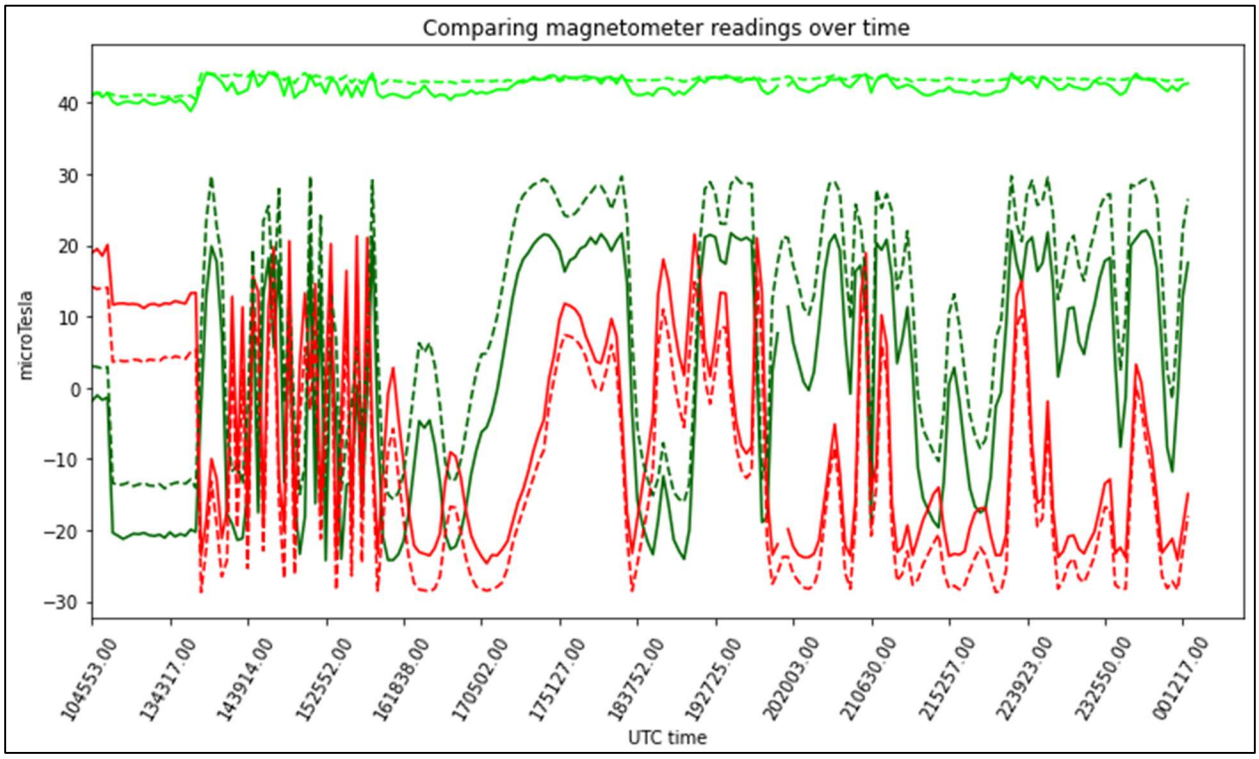


Figure 14: This plot shows the magnetometer readings taken during the flight

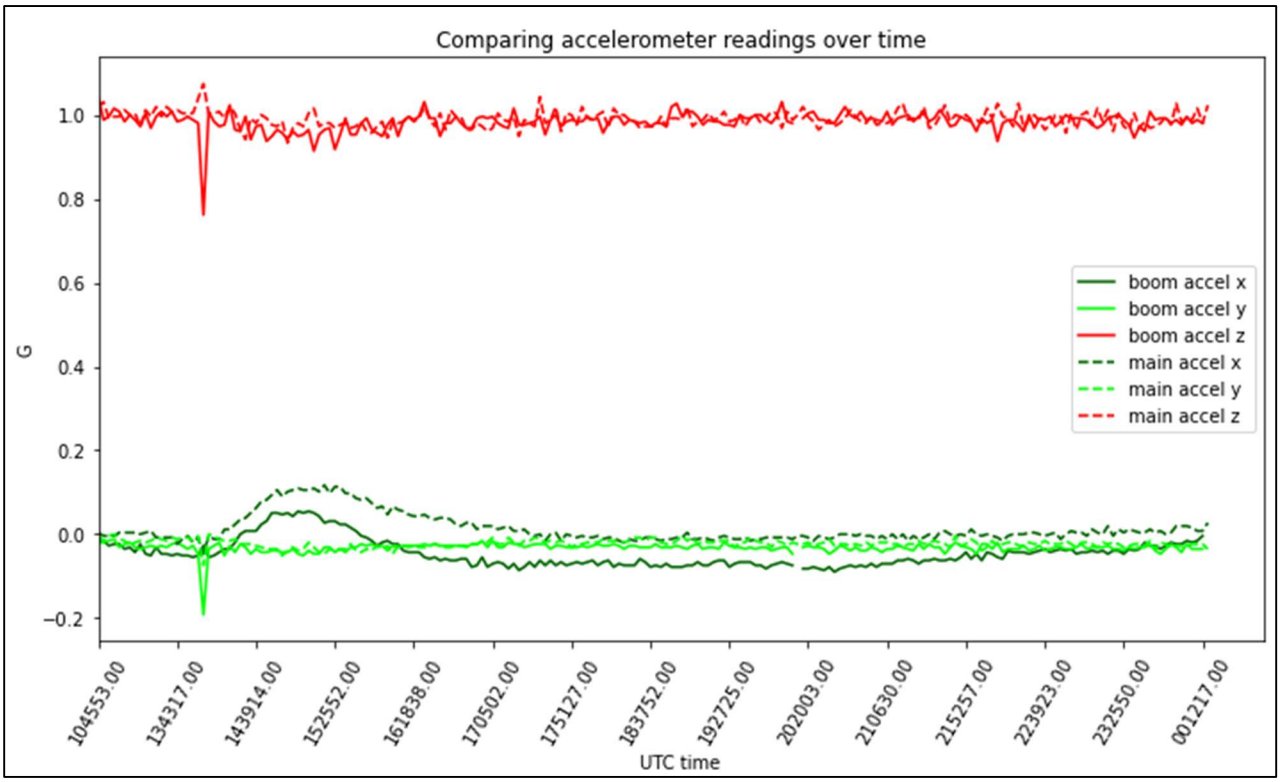


Figure 15: This plot from the accelerometer readings during flight.

.

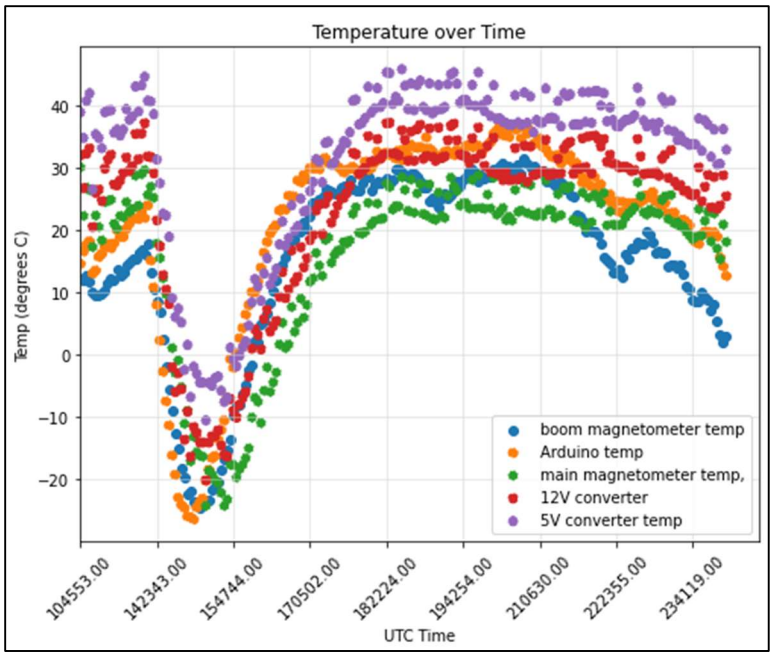


Figure 16: This plot shows the temperature of the different components with temperature sensors during flight.

The main thing to note from these plots is the increase in the x component of both accelerometers that happened when the temperature dipped during flight. This was the same behavior seen during integration.

7.3. Azimuth and Elevation of Payload

7.3.1. Coordinate systems

To continue the description of the analysis, an understanding of the coordinate axes of each of the components and each of the frames of reference being rotated between is necessary.

Magnetometer axes:

The coordinate axes of the magnetometer were labeled as shown in the picture below. When looking at a top-down picture of the payload, the magnetometer x component is pointed down the pae, the z component is pointed to the right, and the y component is pointed into the page. The cameras are pointing in the negative x direction in this coordinate system. The main payload and boom payload magnetometers were oriented the same way, so this applies to both.

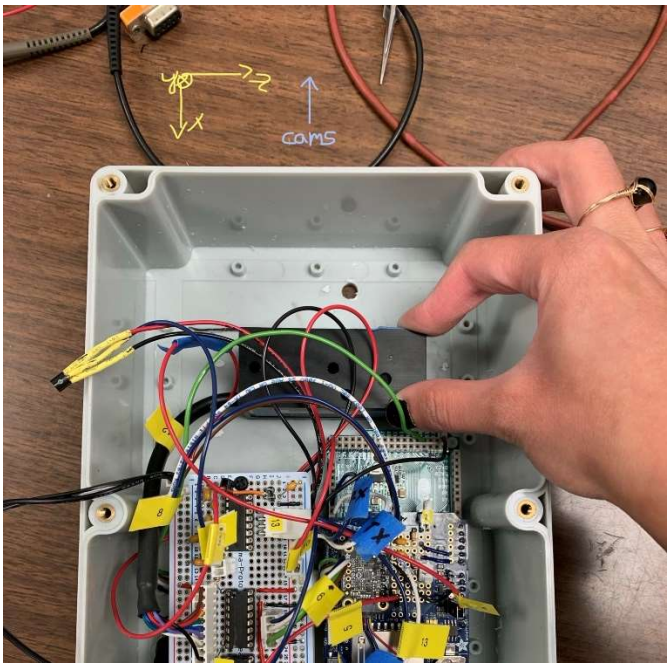


Figure 17: This picture shows the coordinate axes of the magnetometers on the payload.

Accelerometer Axes:

When looking at a top-down view of the payload where the cameras are pointing up: the accelerometer x component is pointed up the page, the y component is pointed to the right, and the z component is pointed into the page. The accelerometers were oriented the same way on both payloads as well.

Camera Frame Coordinate system:

The camera coordinate system is shown in the image below.

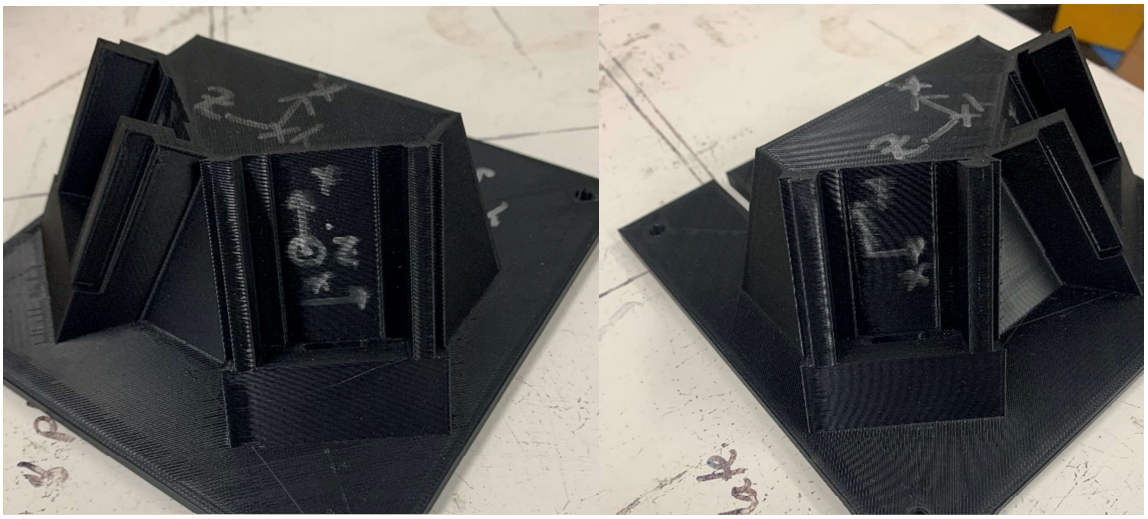


Figure 18: These pictures show the coordinate axes of the cameras/camera mount.

When looking straight at either of the cameras, the x component is to the right, the y component is up the page, and the z component is out of the page. This is because the cameras are upside down in the camera mount.

Earth Frame coordinate system:

The coordinate system of the Earth frame used in this data analysis is North, East, and Down.

Payload Frame coordinate system:

When looking at a top-down view of the payload, the x component of the payload frame is up the page, the y component is to the right, and the z component is down.

Using the magnetometer and accelerometer measurements, we were able to calculate the azimuth and elevation of the payload pointing vector (aka the orientation of the payload in the Earth frame). To do this, we used IGRF readings that correspond to the latitude, longitude, and altitude of each magnetic field/acceleration reading. Because the GPS we put on COMPASS stopped working mid-flight, we used the HASP GPS measurements to obtain the IGRF readings. Once these were obtained, we found rotations between the expected IGRF values/expected gravitational acceleration and the magnetic field/accelerometer readings we measured with our payload. This was done using a SciPy implementation of the Kabsch algorithm (called scipy.spatial.transform.Rotation.align_vectors) that aligns sets of vectors in two different frames. This put the magnetometer readings in the Earth frame. These rotations were then applied to the vector defined to be the payload's "pointing vector" ([1, 0, 0] in the payload frame) to get the payload pointing vector in the Earth's reference frame.

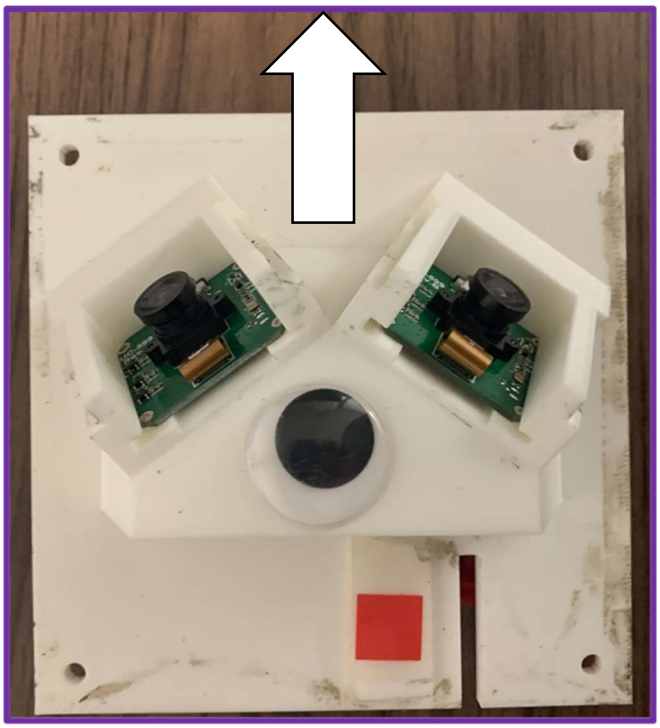


Figure 19: The payload "pointing vector" is shown in white. The goal of this payload is to find the orientation of this vector. This vector corresponds to the negative x direction of the magnetometers as they were oriented on both the payloads.

The resulting payload pointing vectors were in the components North, East, and Down. We then converted these vectors into azimuth and elevation to compare the payload's azimuth and elevation to the azimuth and elevation of the Sun at the same time. These calculations were done using the main payload data and then also the boom payload data, and a comparison of the two can be seen in the next section.

7.4. Azimuth Comparison Results

The following results comparing the calculated pointing vector to what we expect only compare the azimuth measurements, not elevation measurements. This is because we do not have expected elevations of the payload, but we do have expected rotation rates of the payload based on previous flights. Because azimuth is a measure of the angle swept from North, the azimuth of the payload corresponds to the rotation of the payload.

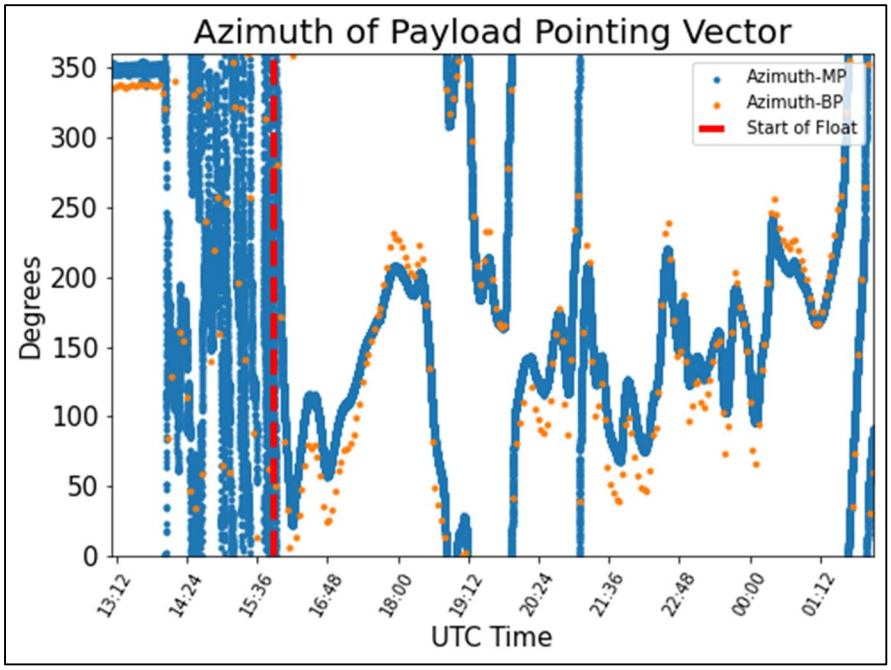


Figure 20: This plot shows the azimuth of the payload pointing vector with time. The main payload is shown in blue and the boom payload is shown in orange. The time at which HASP reached float is denoted by the dashed red line.

In the above plot, the boom payload has fewer data points because we were only able to save what was downlinked during flight because the boom payload SD card stopped working towards the beginning of the flight. At the beginning of the flight, the azimuth jumps up and down very quickly, and then becomes more gradual after reaching float (denoted by the dashed red line). This type of rotation is what we expect based on previous flights and the videos that were taken during flight. A more detailed view of the main payload rotation rates are shown in the next plot.

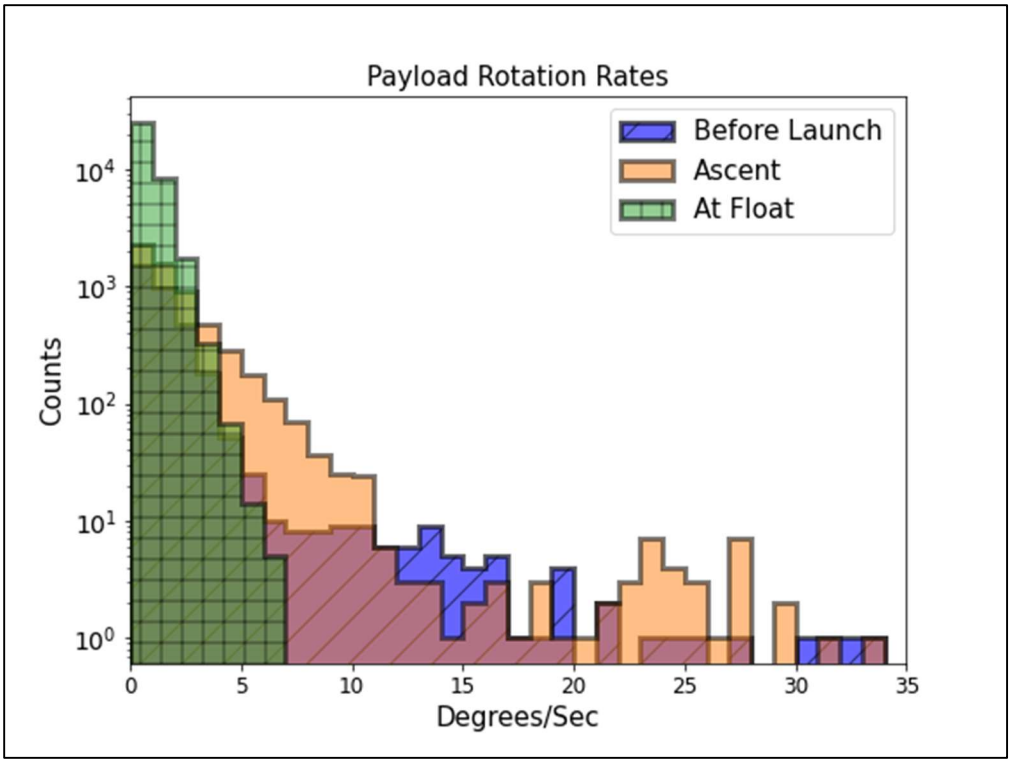


Figure 21: This plot shows the rotation rates calculated at different times during the flight- before launch, during ascent, and at float.

This plot shows the rotation rates of the payload at different periods during the flight- before launch, on ascent, and at float.The differences in rotation rates between these time periods makes sense with what we know is happening to the payload at these times based on previous flights and flight video. All of the rotation rates before launch are very random and correspond to HASP being moved around and situated on the launch vehicle. During ascent, the payload rotates somewhat quickly, however, once it reaches float, the rotation does not exceed around 7 degrees/second. This is consistent with what we would expect based on the flight videos from past flights.

While the boom payload data points appear to follow the main payload data points in Figure 9, this is not exactly the case, as shown by the next plot.

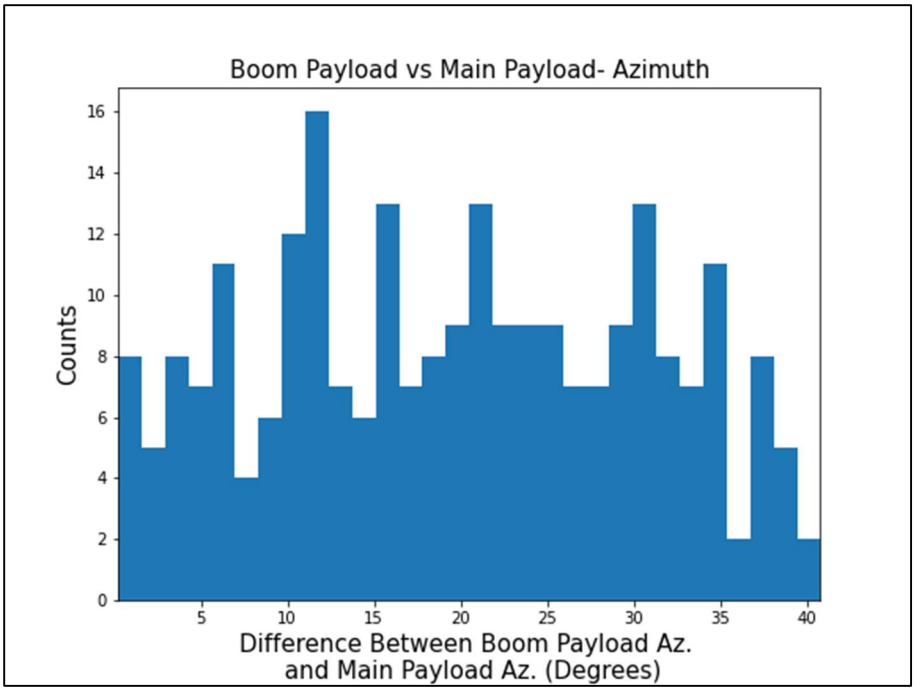


Figure 22: This histogram shows the frequency of the differences in azimuth angle between the pointing vector as calculated by the main payload data and calculated by the boom payload data.

The differences between payload azimuth calculated using boom data and payload azimuth calculated by main payload data range from less than 5 degrees to over 40 degrees. There also does not appear to be a very Gaussian relationship in the difference of the payload measurements, so there's no average difference we can use to correct between the two. To determine which is actually more accurate, these measurements need to be compared to the Sun's actual position, and more analysis must be done to make this comparison.

Despite the differences between the main payload and boom payload azimuths, we can verify that we are on the right track and that our azimuth measurements are not completely off using the data we have from the cameras.

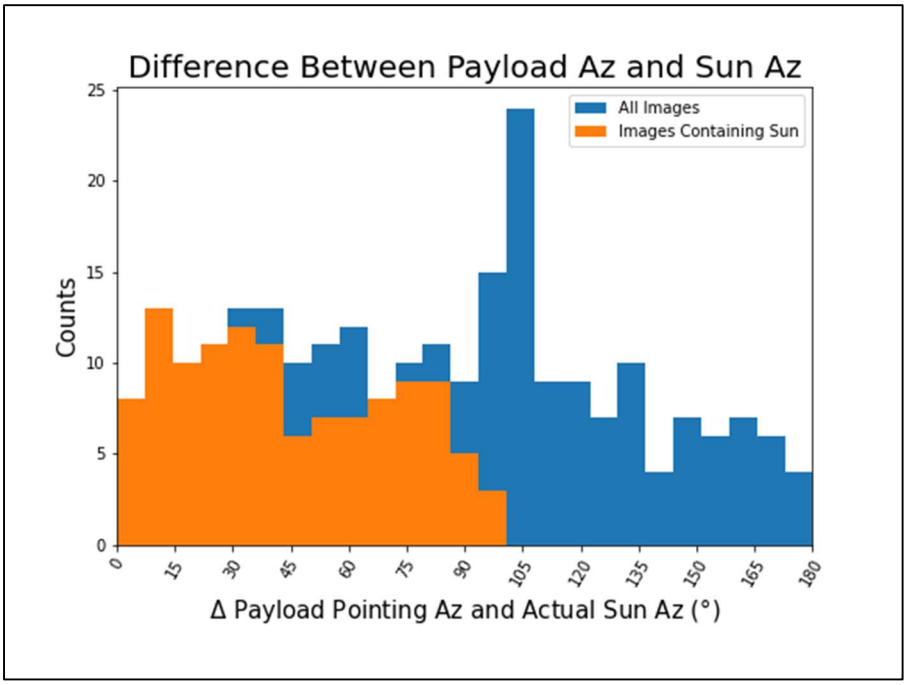


Figure 23: This plot shows the differences between the azimuth of the payload pointing vector and the actual azimuth of the Sun. The orange bars are times when we have images containing the Sun, and the blue bars show the set of all images.

The plot above shows the differences between the azimuth of the payload pointing vector and the actual azimuth of the Sun. The payload azimuths for this plot were calculated using main payload data. This plot serves as a sanity check because it tells us that we only see the Sun in our images when the difference between our calculated pointing vector azimuth is less than 100 degrees different from the actual Sun azimuth. This type of cut off is what we would expect based on the field of view of the wide-angle camera lenses we used. It also tells us that we do not see the Sun when the difference between the two angles is 180 degrees, meaning, as expected, we don't see the Sun when it's directly behind the front of the payload.

7.5. Improvements/Future Work

There are a few improvements that could've been made on this payload before flight. For one, the SD card on the boom should have been secured better so that we could've had a full dataset for both the main payload and the boom payload. The SD card never stopped working during bench testing or integration. Based on the data that was still on the SD card from previous times the payload had been turned on, the last time the SD card was saving data was actually very early in the morning on September 8th, the attempted (but scrubbed) flight date. However, we were unaware that this was an issue until HASP was already on the launch vehicle on September 14th because the SD card flag in the downlinked data was still saying the SD card was initialized, so the flag also did not work as planned. The flag in the downlinked data probably could have been tested more thoroughly as well. We still ended up with downlinked data from the boom, but it was not as complete of a set as we had planned on having.

The GPS stopped sending data mid-flight. Because we were only downlinking the parsed data and not the raw NMEA strings, it was unclear if this was from not having a fix or an issue with the range setting of the GPS. We think it was probably that the range was not set correctly because there should not have been a reason to lose a fix during flight, and because it was still sending time and date information, it was still on. This could've been prevented by reading the documentation more clearly. It ended up not being too much of an issue because we were able to use HASP GPS data, but it was definitely preventable. The other improvement that could've been made was securing the solar filters better to prevent any light from leaking in and causing the distorted camera reflections on the images. This did not end up being too much of a problem, however, because we were able to subtract the background of the images.

There is also more analysis that needs to be done on the collected data to fully achieve the goals of this payload. While we were able to verify that the magnetometer and accelerometer give results that make sense qualitatively (i.e. the rotation rate plot and the plot comparing the pointing vector to camera images), much more rigorous analysis needs to be done to quantitatively identify how far off our calculation of the Sun's azimuth and elevation would be compared to the actual Sun's azimuth and elevation. To characterize the difference between our calculation of the Sun's azimuth and elevation and the actual Sun azimuth and elevation, the camera pixel vector needs to be rotated into the Earth frame (using rotation matrices found with the magnetometer/accelerometer readings) and then the differences between this vector and the Sun's vector in the Earth frame need to be compared. Once this is done, we can figure out how well our orientation system of the magnetometer and accelerometer work to orient the payload in space.

The differences between the boom payload and the main payload can also be quantitatively analyzed. The reason for having a payload on a boom in the first place was to determine if magnetic fields coming from other payloads/HASP electronics had an effect on the onboard magnetometer and would skew our results. To characterize the differences between the boom payload and main payload orientation systems, one must first be able to compare the differences in calculated azimuth/elevation and actual azimuth/elevation of the Sun (as described in the above paragraph). This must be done with both payloads and then the differences between the two payloads can be characterized.

The effects of the temperature dependence of the magnetometer can also be characterized by comparing the error in the calculated azimuth and elevation of the payload at times when the accelerometer is fluctuating and when it is leveled off.

8. Demographic Information

Name	Start Date	End Date	Role	Student Status	Race	Ethnicity	Gender	Disabled
Jeanne Garriz	9/15/20	Present	Project manager/Mechanical	Undergrad	White	Non-Hispanic/Latino	Female	No
Sabrina Huezo	9/15/20	Present	Electrical Lead	Undergrad	White	Hispanic	Female	No
Harrison Gietz	12/1/20	10/1/21	Software/Mechanical secondary	Undergrad	White	Non-Hispanic/Latino	Male	No
Jesse Frank	12/15/20	10/1/21	Software lead	Undergrad	White	Non-Hispanic/Latino	Male	No
Aaron Ryan	9/15/20	Present	Faculty advisor	Staff	White	Non-Hispanic/Latino	Male	No

Table 3: Demographic information on the COMPASS Team members.

9. Publications

Garriz, Jeanne; Huezo, Sabrina. “A Compact, Low-Cost Balloon Flight Attitude Orientation System.” American Geophysical Union Fall Meeting. 13 December 2021, Ernst N. Morial Convention Center, New Orleans, LA. Poster Presentation

10. References

1. ‘Angle of View’ (2020) Wikipedia, Available at https://en.wikipedia.org/wiki/Angle_of_view

2. ‘Azimuth’ (2020) Wikipedia, Available at https://en.wikipedia.org/wiki/Angle_of_view

3. Angle/Path of the Sun calculator application, ‘Suncalc,’ Available at https://www.suncalc.org/

4. Adafruit, “LSM303 Accelerometer + Compass Breakout,” Last updated Jan 8, 2020, Available at https://cdn-learn.adafruit.com/downloads/pdf/lsm303-accelerometer-slashcompass-breakout.pdf?timestamp=1608269052

5. STMicroelectronics, “Ultra-compact high-performance eCompass module: ultra-lowpower 3D accelerometer and 3D magnetometer,” November 2018, Available at https://www.st.com/resource/en/datasheet/lsm303agr.pdf

6. NXP, “Sensor Toolbox Development Boards for a 9-Axis Solution using MMA8652FC, FXAS21002C and MAG3110,” October 2015, Available at https://www.nxp.com/design/software/development-software/sensor-toolbox-sensordevelopment-ecosystem/evaluation-boards/sensor-toolbox-development-boards-for-a-9-axis-solution-using-mma8652fc-fxas21002c-and-mag3110:AGM04-SOLUTION

7. OKW Enclosures, “C7012011,” Available at https://www.okwenclosures.com/en/InBox/C7012011.htm

8. McMaster-Carr, “Structural FRP Fiberglass Square Tube,” 2020, Available at https://www.mcmaster.com/8548K31/

9. LaACESProgram (2019), “HASP 2019 Downward Looking Camera #2 Stream Part 2,”

Available at https://www.youtube.com/watch?v=Ler8qjh9bHw

10. Arducam, “Camarray – Arducam 1MP Stereoscopic Camera Bundle Kit,” December 7,

2020, Available at https://www.arducam.com/docs/cameras-for-raspberrypi/synchronized-stereo-camera-hat/camarry-1mp2-stereoscopic-camera-hat/

11. Arducam, “Arducam 1/4” M12 Mount 1.6mm Focal Length Camera Lens M40160M12,”

March 20, 2019, Available at https://www.arducam.com/product/u105001-4-m12-mount1-6mm-focal-length-camera-lens-ls-0002-for-raspberry-pi/

12. Arducam, “Introduction to Arducam USB Camera Shields,” August 13, 2020, Available

at https://www.arducam.com/docs/usb-cameras/introduction-to-arducam-usb-camerashields/

13. Arducam, “Arducam 0.36MP MT9V022 Monochrome 1/3” CMOS Camera Module,”

March 22, 2019, Available at https://www.arducam.com/product/arducam-mt9v022-

monochrome-1-3-inch-cmos-0-36mp-camera-module/

14. Arducam, “Arducam USB3.0 Camera Shield,” January 30, 3019, Available at https://www.arducam.com/product/usb3-0-camera-shield-2/

15. Uctronics, “Stereo Camera Adapter Board,” Available at https://www.uctronics.com/stereo-camera-adapter-board.html

16. Arduam, “Arducam 1/3” M12 mount 2.25mm Focal Length Camera Lens M30225H10,”

March 20, 2019, Available at https://www.arducam.com/product/m30225h10-m12-

mount-2-25mm-focal-length-camera-lens-ls-30188-for-raspberry-pi/

17. Pultex, “The New and Improved Pultex Pultrusion Design Manual,” 2004, Available at

https://www.creativepultrusions.com/default/assets/File/DMV5R11.pdf

18. Raspberry Pi, “FAQs,” Available at https://www.raspberrypi.org/documentation/faqs/#pipower-gpioout

19. Freescale Semiconductor Inc, “Xtrinsic MMA8451Q 3-Axis, 14-bit/8-bit Digital

Accelerometer,” 2013, Available at https://cdn-.adafruit.com/datasheets/MMA8451Q1.pdf

20. Trimble, “Copernicus II GPS Receiver Reference Manual,” 2009. Available at

http://cdn.sparkfun.com/datasheets/Sensors/GPS/63530-10_Rev-B_Manual_CopernicusII.pdf

21. LECO Plastics, “Zip Ties- Cable Ties,” Available at https://lecoplastics.com/cableties.aspx

22. Arducam, “1MP OV9281 1/4” CMOS Global Shutter Standalone Camera UC9281M1

MIPI Interface,” Available at https://www.arducam.com/product/1-4-cmos-ov9281-

global-shutter-standalone-camera-uc9281m1-mipi-interface/

23. Engineers Edge, “AWG Copper Wire Size and Data Table Chart @ 100 Degrees F,”

Available at https://www.engineersedge.com/copper_wire.htm

24. McMaster-Carr, “Black-Oxide Alloy Steel Socket Head Screw,” Available at

https://www.mcmaster.com/90044A125/

25. Microchip, “2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial

Interface,” 2008, Available at https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

26. RobotShop, “Arduino Mega 2560 Datasheet,” Available at

http://eprints.polsri.ac.id/4598/8/File%20VIII%20%28Lampiran%29.pdf

27. Murata Power Solutions, “78xxSR Series- 3.3V/5V/12V Outputs High-Efficiency

Switching Regulators with LM78xx Pinouts,” 2015, Available at

COMPASS 3 Pre-CDR v2.0
https://www.murata.com/products/productdata/8807028260894/dms78xxsr.pdf?1583754811000

28. Bartington, “Mag648 & Mag649® Low Power Three-Axis Magnetic Field Sensors,”

Available at
https://www.bartington.com/wpcontent/uploads/pdfs/datasheets/Mag648_649_DS2298.pdf

29. Millan, R. et al, “The Balloon Array for RBSP Realistic Electron Losses (BARREL),”

Space Science Reviews, DOI 10.1007/s11214-013-9971-z, Available at

http://www.dartmouth.edu/~barrel/documents/10.1007_s11214-013-9971-z.pdf

30. Raspberry Pi Trading Ltd, “Raspberry Pi 4 Computer- Model B,” 2020, Available at

https://static.raspberrypi.org/files/productbriefs/200521+Raspberry+Pi+4+Product+Brief.pdf

31. Sparkfun, “RS-232 vs TTL Serial Communication,” 2010, Available at

https://www.sparkfun.com/tutorials/215

32. Adafruit, “Raspberry Pi Analog to Digital Converters,” Available at

https://www.digikey.com/en/maker/projects/raspberry-pi-analog-to-digitalconverters/72388f5f1a0843418130f56c53a1276c

33. Adafruit, “Analog Inputs for Raspberry Pi Using the MCP3008,” Available at

https://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-theraspberry-pi?view=all

34. Raspberry Pi, “Raspberry Pi Temperature Monitoring,” 2020, available at

https://www.raspberrypi-spy.co.uk/2020/11/raspberry-pi-temperature-monitoring/

35. CUI INC, “P78-2000-S Non-Isolated Switching Regulator,” Feb 21, 2020, Available at

https://www.mouser.com/datasheet/2/670/p78_2000_s-1729053.pdf

36. Cooperhill Technologies, “Raspberry Pi Proven to Withstand Extended Temperature

Range,” 2016, Available at https://copperhilltech.com/blog/raspberry-pi-proven-towithstand-extended-temperature-range/

37. Azimuth and Elevation Schematic by TWCarlson - File:Azimut_altitude.svg, CC BY-SA

3.0, https://commons.wikimedia.org/w/index.php?curid=17727911

38. Alken, P., Thébault, E., Beggan, C.D. et al. International Geomagnetic Reference Field: the thirteenth generation. Earth Planets Space 73, 49 (2021), Available at https://doi.org/10.1186/s40623-020-01288-x

11. Appendices

Appendix Table of Contents

Appendix A: Power Budget

Arduino Power Consumption

COMPONENT	OPERATING VOLTAGE (V)	UPPER LIMIT CURRENT DRAW (mA)	Method	POWER CONSUMED (W)	UNC. (W)
Accelerometer	5	1.3	Measured	<1	±0.01
Environmental sensors	5	~2	Estimated	<1	±0.01
Bare-board Arduino Mega	12	200	Datasheet	2.4	±0.24
TOTAL		203.3		4.4	±0.3

Table 4: This table shows the power consumption of the bare board Arduino Mega and also the parts that were powered through the Mega.

12V Power Budget

COMPONENT	OPERATING VOLTAGE (V)	UPPER LIMIT CURRENT DRAW (mA)	Method	POWER CONSUMED (W)	UNC. (W)
Arduino Mega (includes load of accelerometer, temperature sensor, and Active Bare-Board Mega)	12	203	Estimated	2.44	±0.25
Magnetometers	12	40	Measured	0.48	±0.05
+5V Voltage Reference	12	~1	Estimated	<1	
TOTAL		244.3		2.9	±0.3

Table 5: Power budget for all electronics connected to the 12V supply from the P7812-2000-S converter. These are the electronics that were housed on the boom.

Raspberry Pi Power Consumption

COMPONENT	OPERATING VOLTAGE (V)	UPPER LIMIT CURRENT DRAW (mA)	Method	POWER CONSUMED (W)	UNC. (W)
Camera System*	3.3	97	Estimated	0.32	±0.03
Rasp Pi Active Bare-Board	5	530	Measured	2.65	±0.27
GPS Module	3.3	43	Measured	0.14	±0.01
Accelerometer	3.3	1.3	Measured	<1	± 0.01
RS-232 Level Shifters	3.3	1	Datasheet	<1	± 0.01
ADCs	5	<1	Datasheet	<1	± 0.01
TOTAL		672.3		3.1	±0.3

Table 6: The maximum power to be used from the Raspberry Pi and the loads it carries.

*Power consumption could not be found for the intended camera system, so it was estimated using a similar single 0.36MP camera. It is likely that the actual current draw will be larger than this estimate, as two cameras will be used on COMPASS.

5V Power Budget

COMPONENT	OPERATING VOLTAGE(V)	UPPER LIMIT CURRENT DRAW (mA)	Method	POWER CONSUMED(W)	UNC. (W)
RPi4 Model B (includes loads of GPS, Camera System, Accelerometer, and Active Bare-Board RPi)	5	672.3	Measured	3.36	±0.27
Environmental sensors	5	~2	Estimated	0.01	±0.05
TOTAL		673.3		~3.4	±0.3

Table 7: Power budget for all electronics connected to the 5V supply from the P7805-2000-S converter. These are the electronics that were housed on the main payload mounting plate

Power Consumption of DC-DC Converters

COMPONENT	Voltage Out (V)	Current Out Max (A)	Power Out Max (W)	Efficiency (%)	Power In Max (W)	Power In Max Unc. (W)	Current in @30 V (A)
P7812 Converter	12	0.244	2.9	86	3.37	±0.34	0.11
P7805 Converter	5	0.673	3.4	80	4.25	±0.43	0.14
TOTAL (entire payload)					**7.62**	**±0.77**	**0.25**

Table 8: Current and power consumption information for the DC-DC converters, and the total power consumption of the COMPASS payload (given in the bottom right).

Appendix B: Detailed Electrical Diagrams

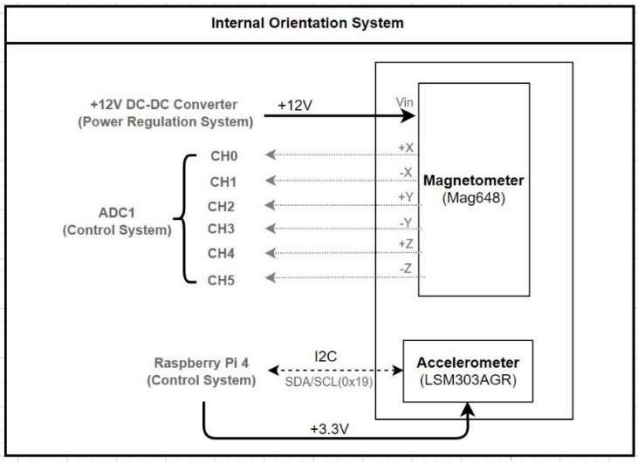


Figure 24: This diagram shows the orientation system that was housed in the main payload. It utilizes the Mag648 and the LSM303AGR Accelerometer.

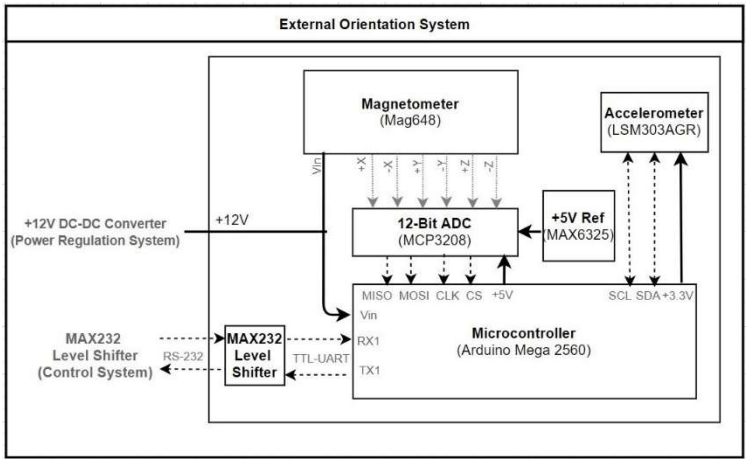


Figure 25: This diagram shows the external orientation system that was housed at the end of the boom. It used the same magnetometer/accelerometer combination as the internal orientation system.

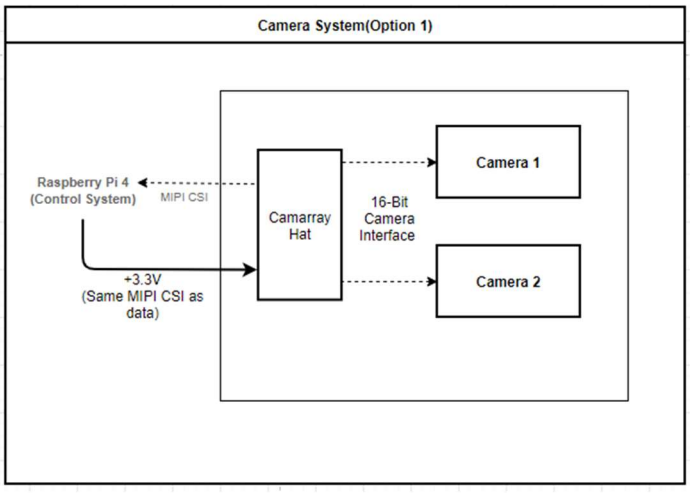


Figure 26: This diagram shows how the camera system is connected to the RPi and the rest of the payload.

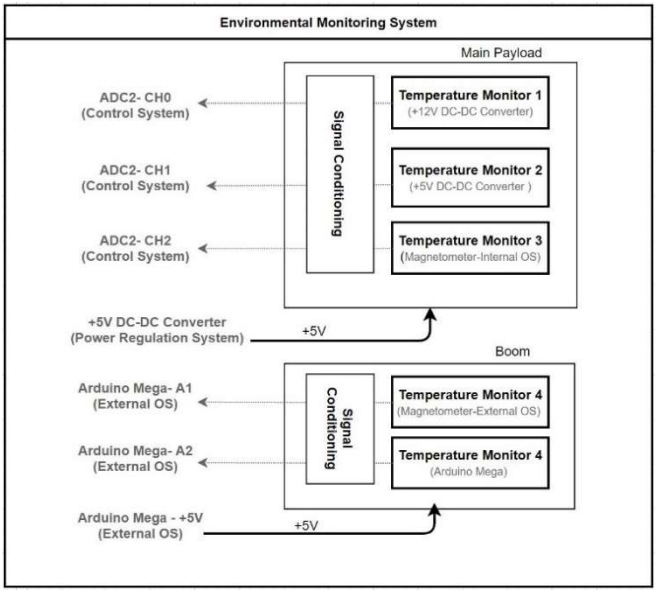


Figure 27: This diagram shows the connections between the temperature sensors and the ADCs they are connected to.

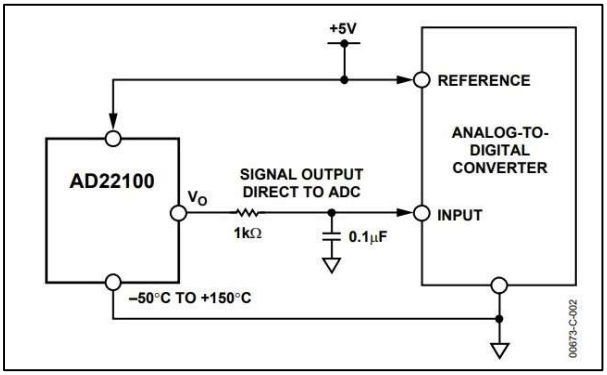


Figure 28: This circuit diagram shows the suggested low-pass filter for the temperature sensors.

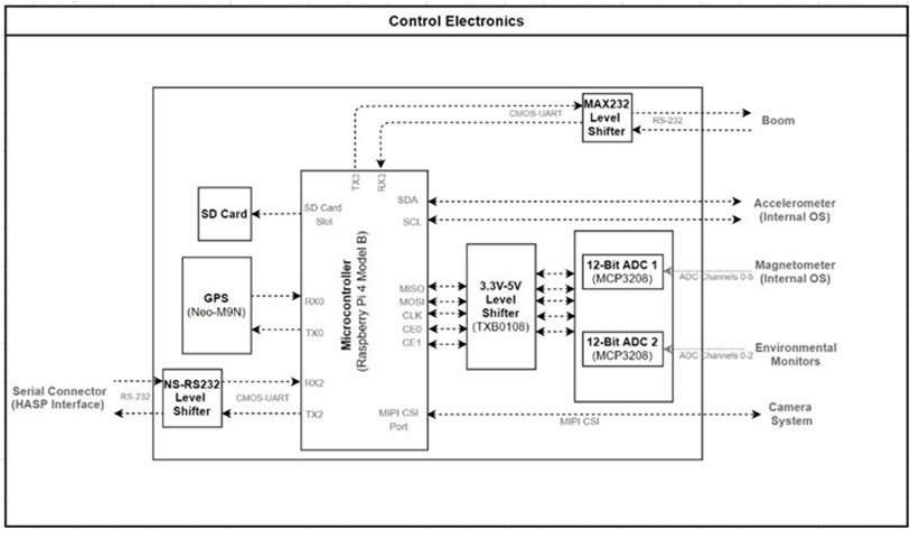


Figure 29: This figure shows how the Raspberry Pi controls the rest of the payload.

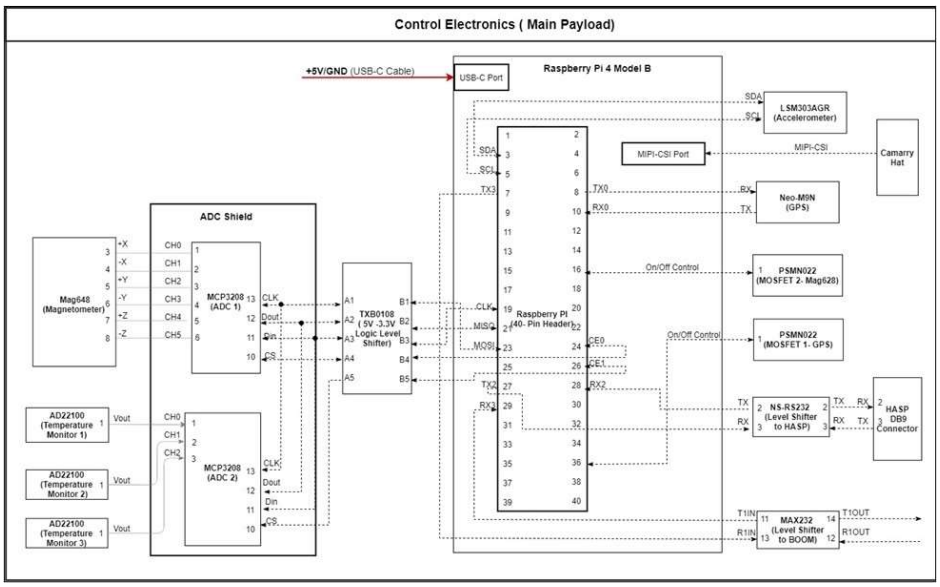


Figure 30: This figure shows the detailed schematic of communication pin connections between the between the components of the control system on the main payload.

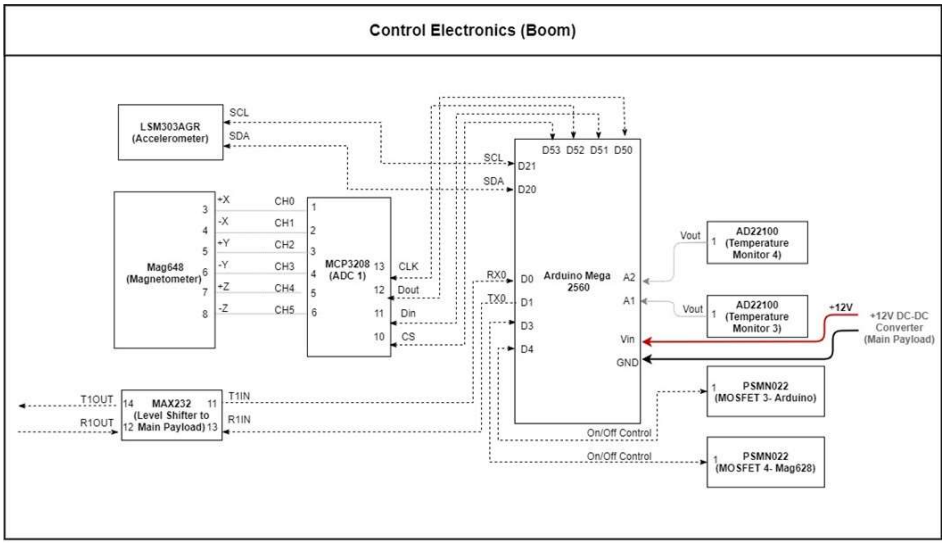


Figure 31: This figure shows the detailed schematic of the communication pin connections between the components of the boom payload.

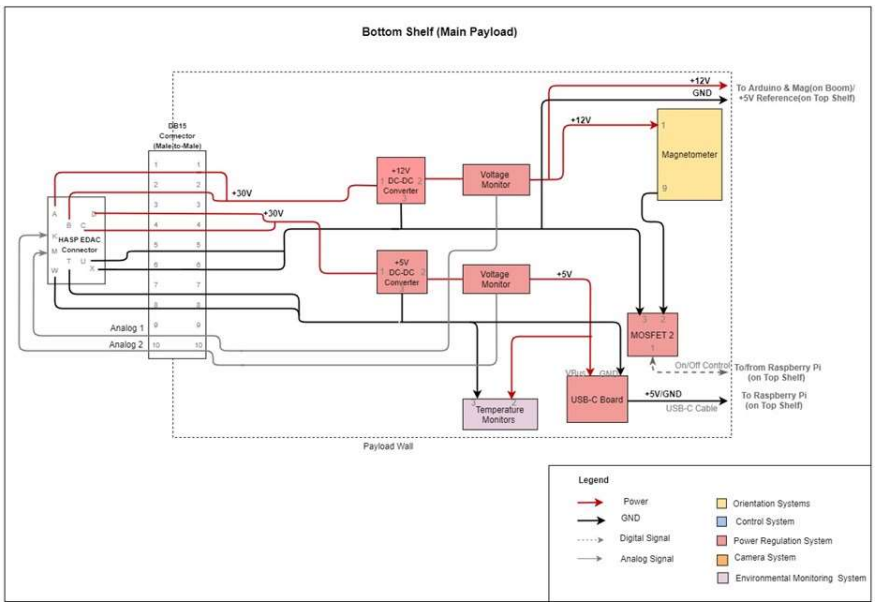


Figure 32: This figure shows how power connects from HASP to all elements of the bottom shelf of the main payload.

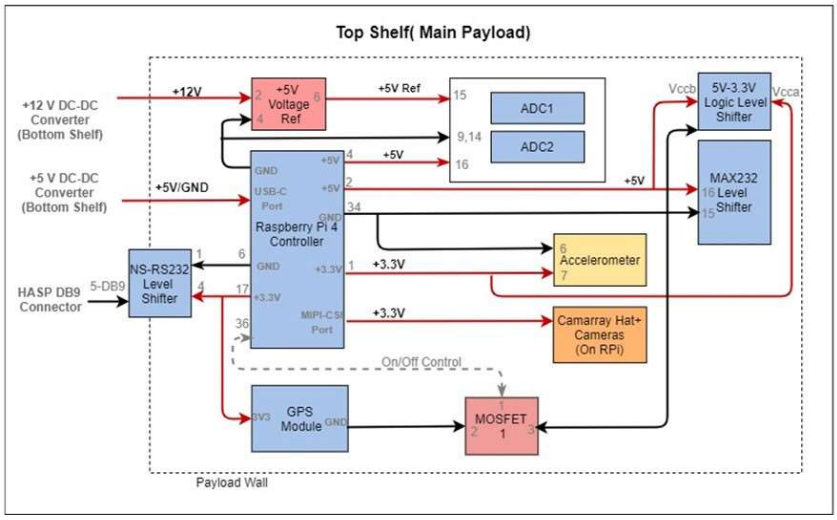


Figure 33: This diagram shows how power gets from HASP to components on the top shelf of the main payload.

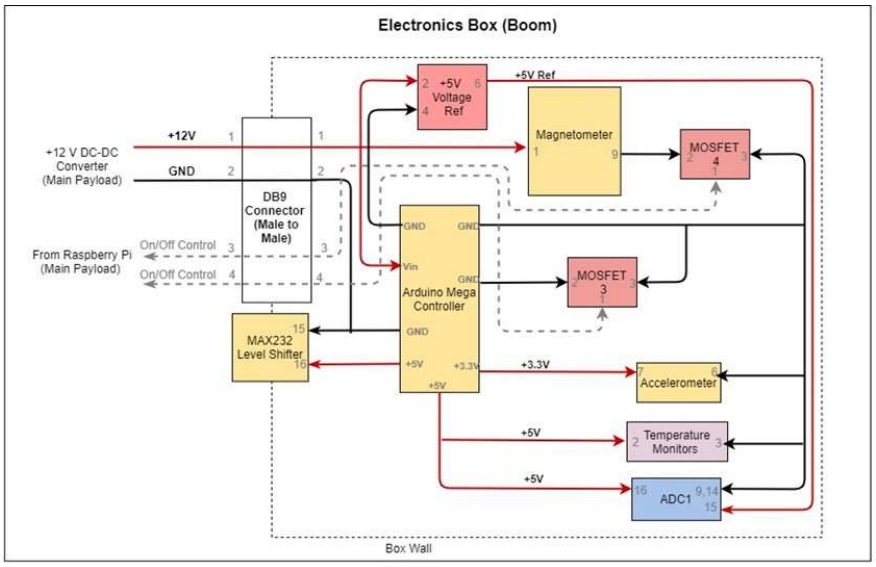


Figure 34: This diagram shows how power gets to all components of the boom payload.

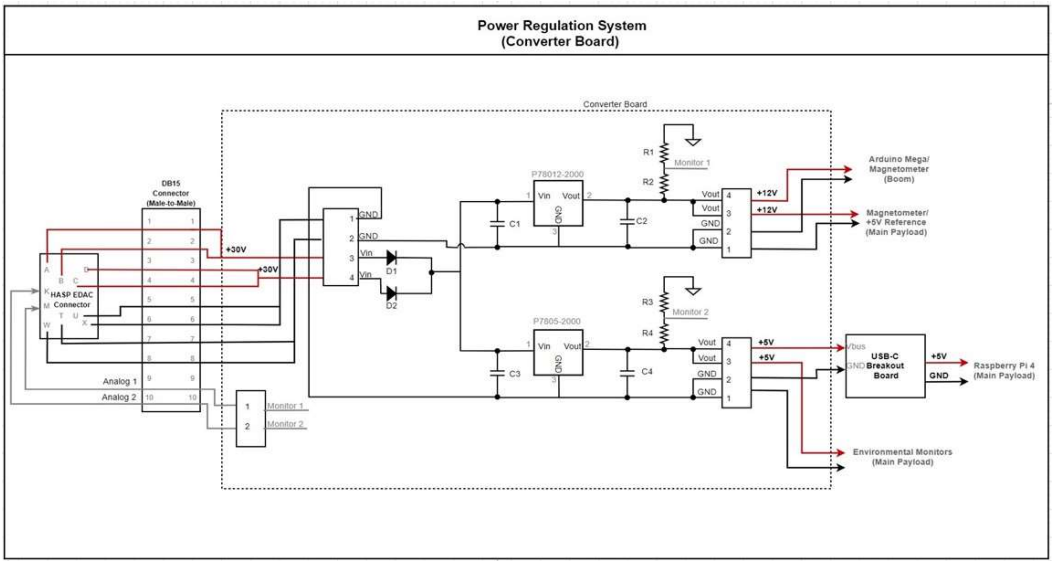


Figure 35: This detailed circuit diagram shows in depth how the power converter board works to supply power from HASP to the COMPASS payload and what electrical components are used.

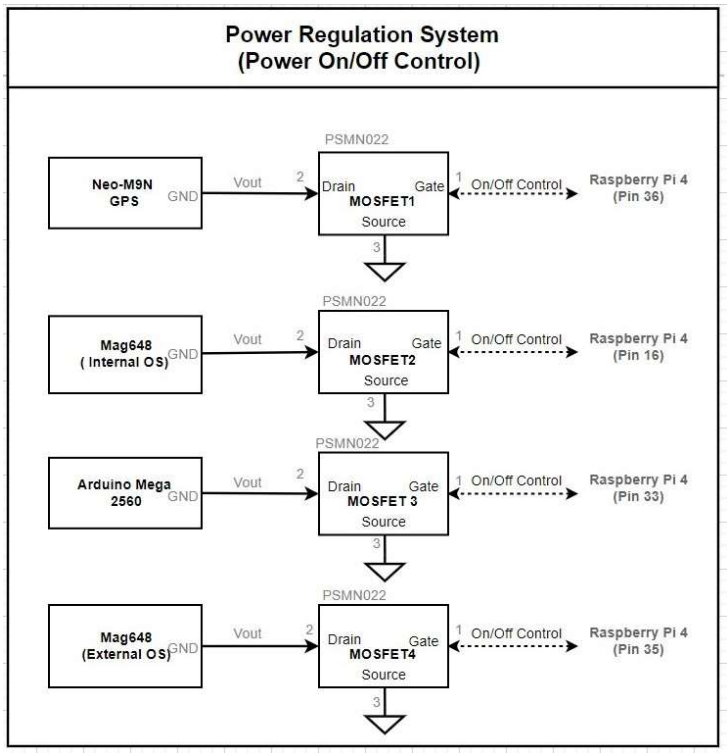


Figure 36: Diagram of the power regulation system that controls the different components that were able to be turned on and off.

Appendix C: Software Flowcharts

1. Main Payload Flowcharts:

The following flowcharts show the more detailed processes of the subroutines in the main payload flight software shown in section 4.3.

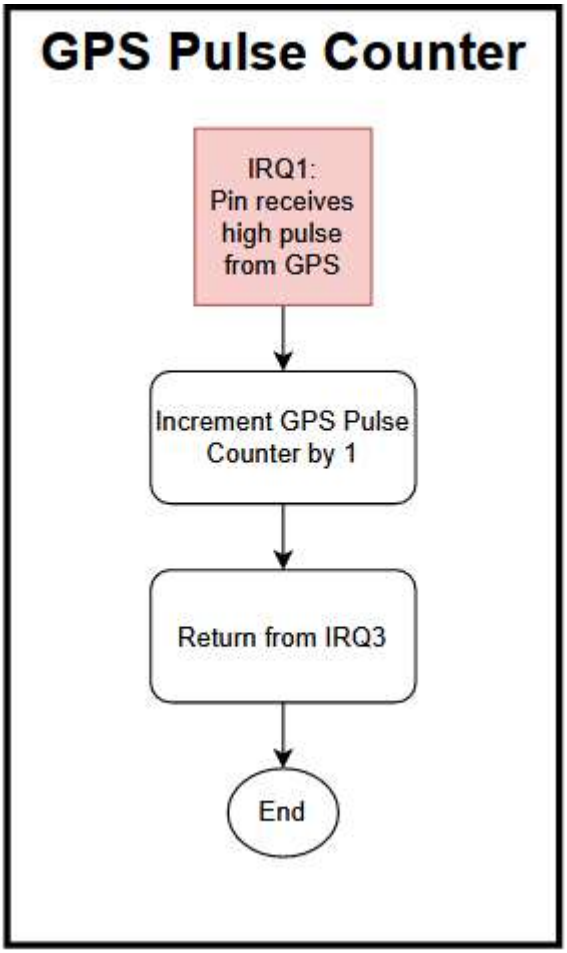


Figure 37: High level flowchart for GPS Pulse counter interrupt.

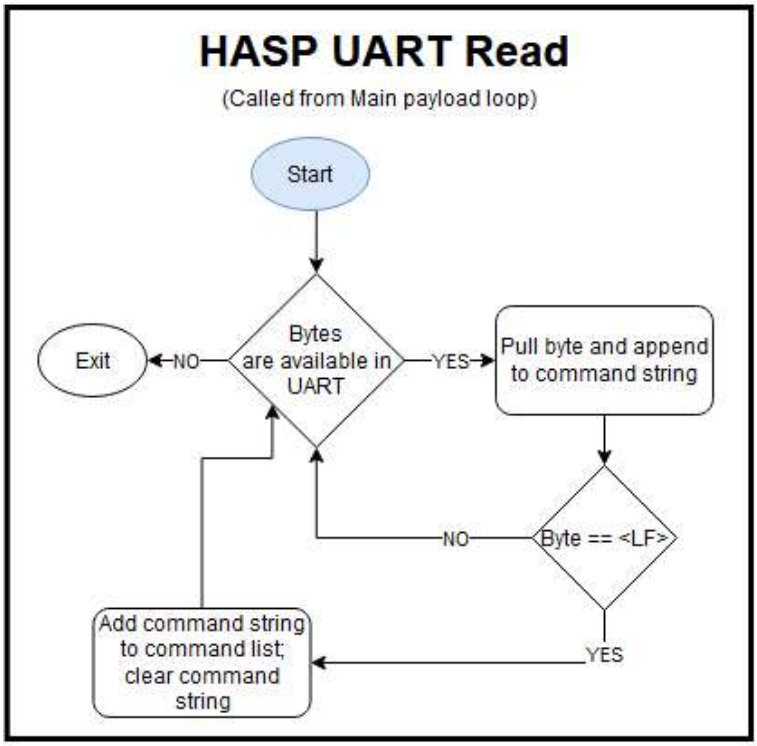


Figure 38: Diagram showing the subroutine the main payload calls to read the HASP UART.

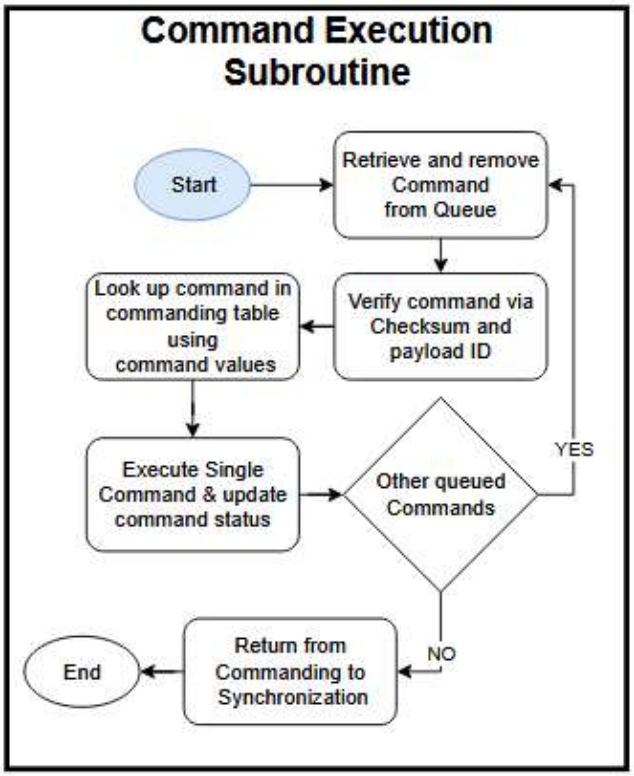


Figure 39: Subroutine called by main payload to execute uplinked commands.

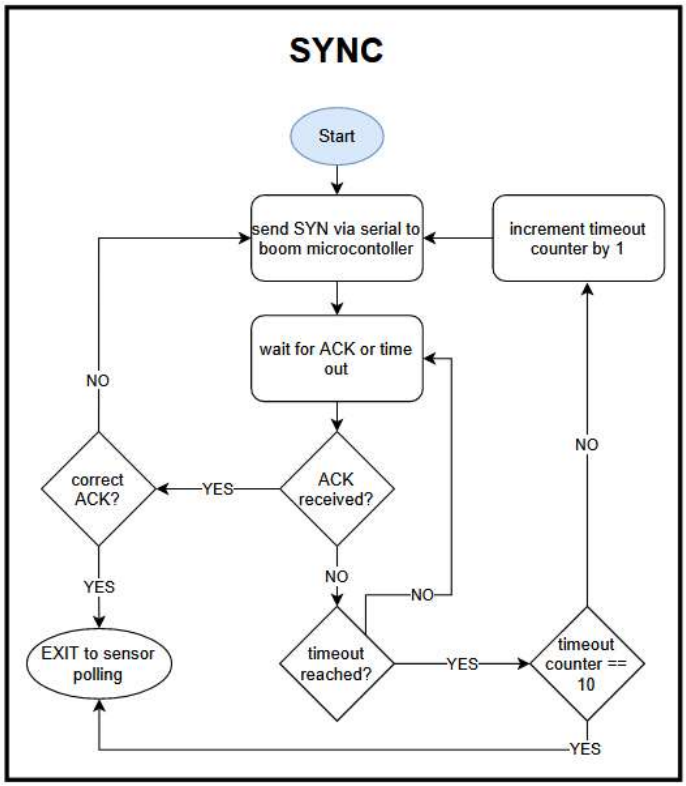


Figure 40: Flowchart showing the synchronization routine between the main payload and boom payload microcontrollers used to maintain timing accuracy.

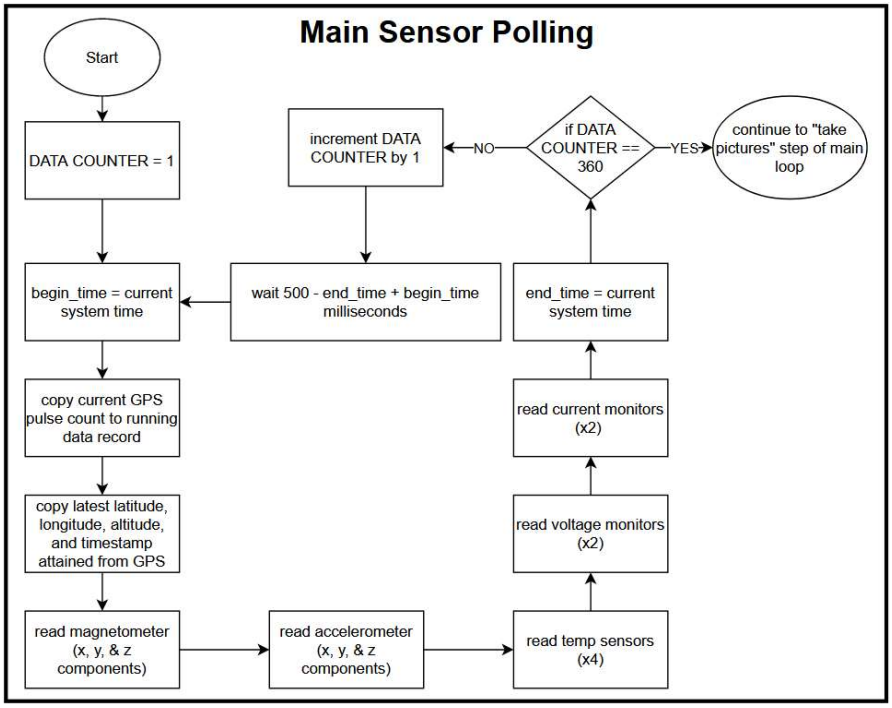


Figure 41: Main payload sensor polling routine showing the order in which sensors are polled.

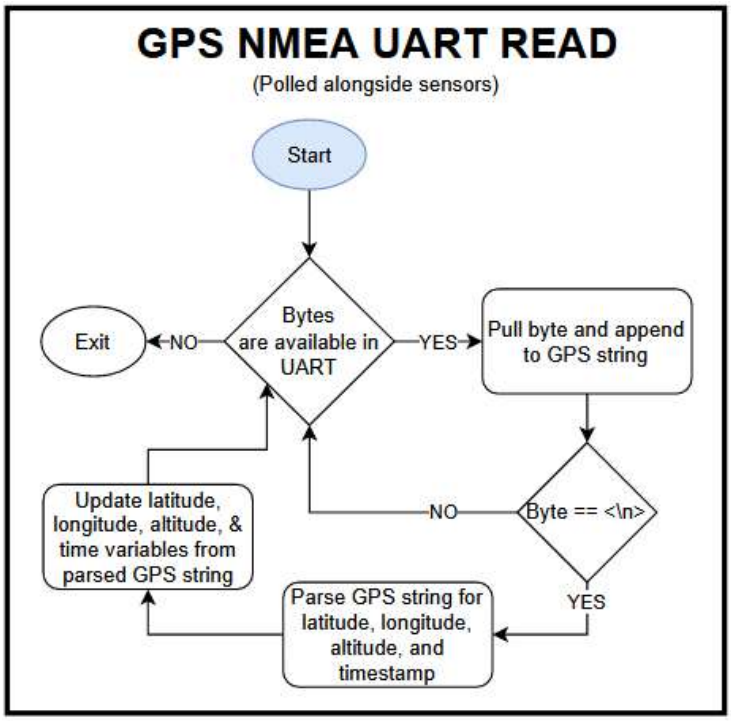


Figure 42: Shows subroutine that polls GPS and parses received NMEA strings

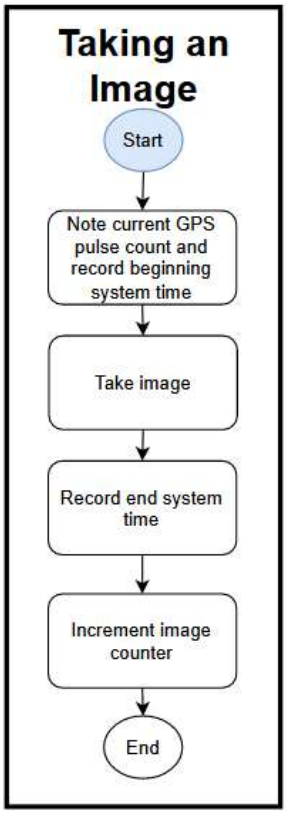


Figure 43: Subroutine used by main payload to take images of Sun.

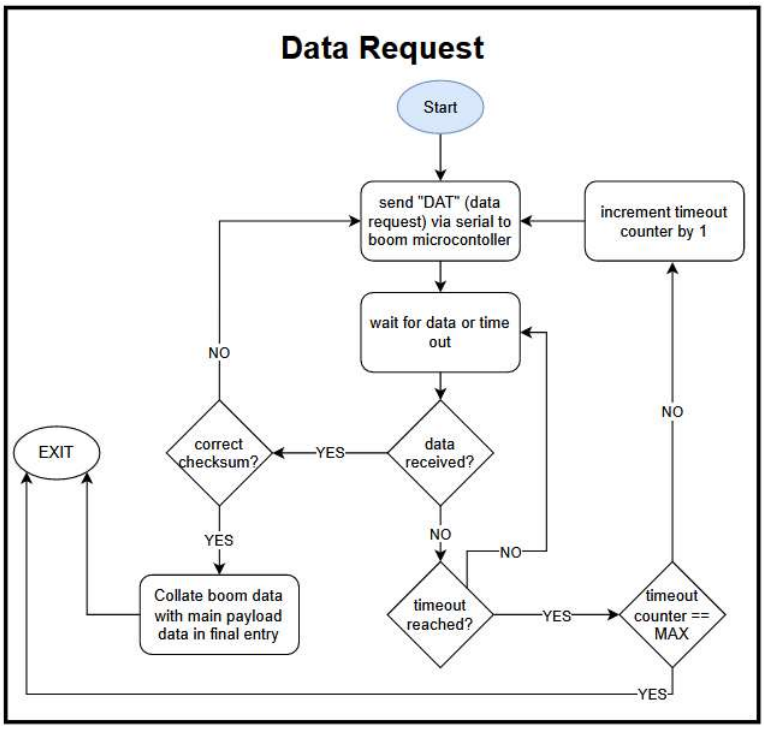


Figure 44: Subroutine used by main payload to request data from the boom payload.

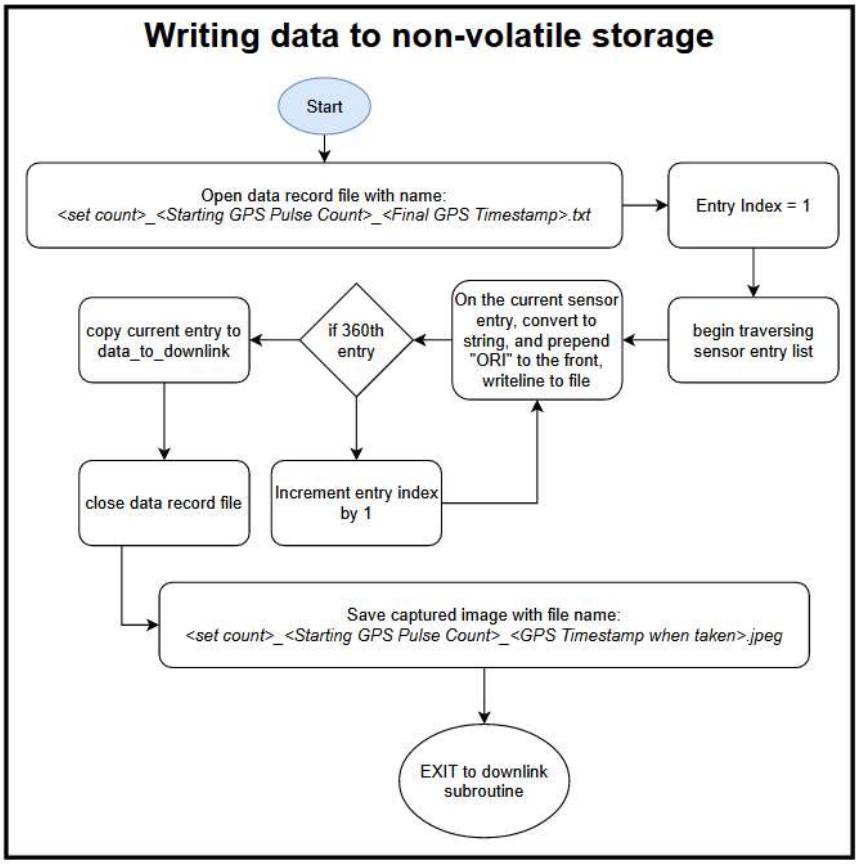


Figure 45: Subroutine used to save data to SD card on main payload

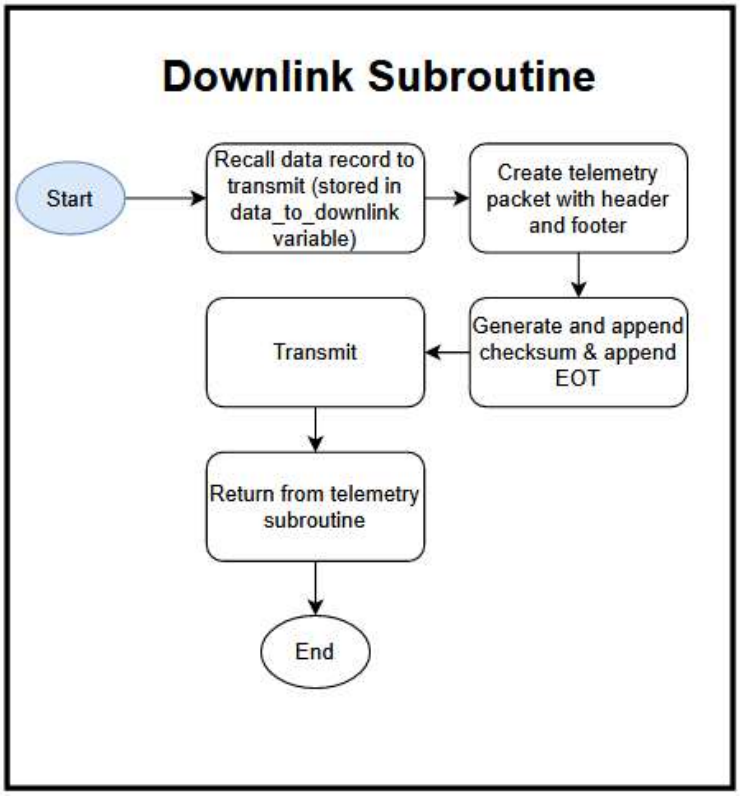


Figure 46: Subroutine used to downlink data from the main payload to the ground during flight.

2. Boom Payload Software

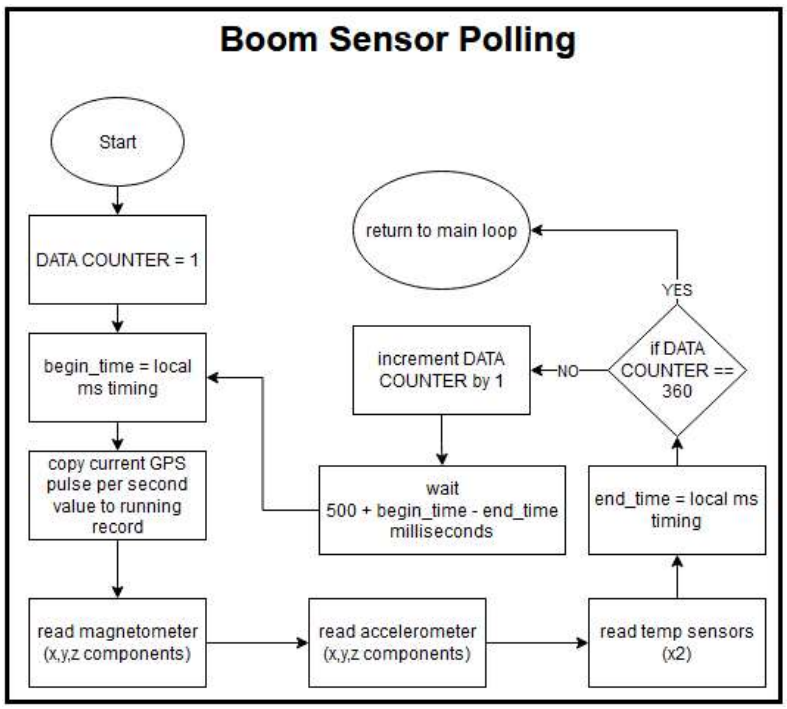


Figure 47: Shows sensor polling order of the sensors on the boom payload.

Appendix D: Detailed Mechanical Drawings

The following figures are more detailed mechanical drawings that expand upon the general drawings from section 4.4. All dimensions in these drawings are in inches.

1. Main Payload Drawings

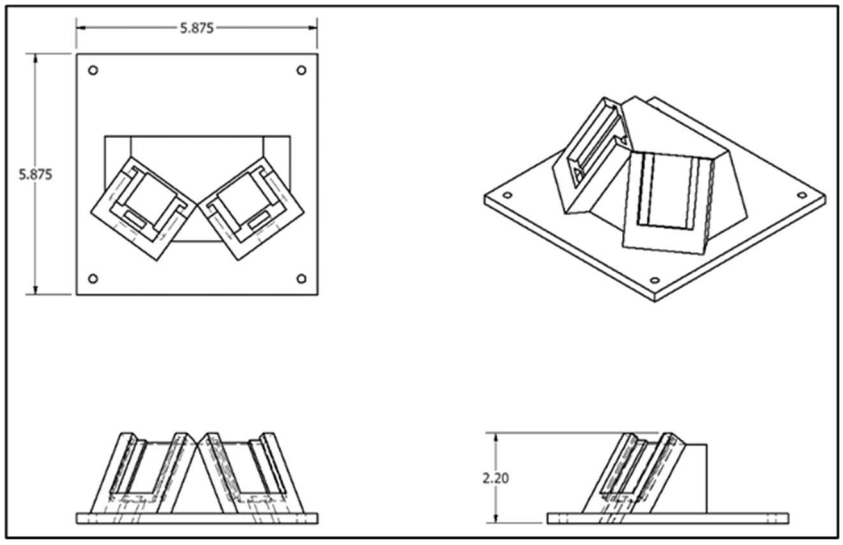


Figure 48: Mechanical design and dimensions of the camera mount/payload cap.

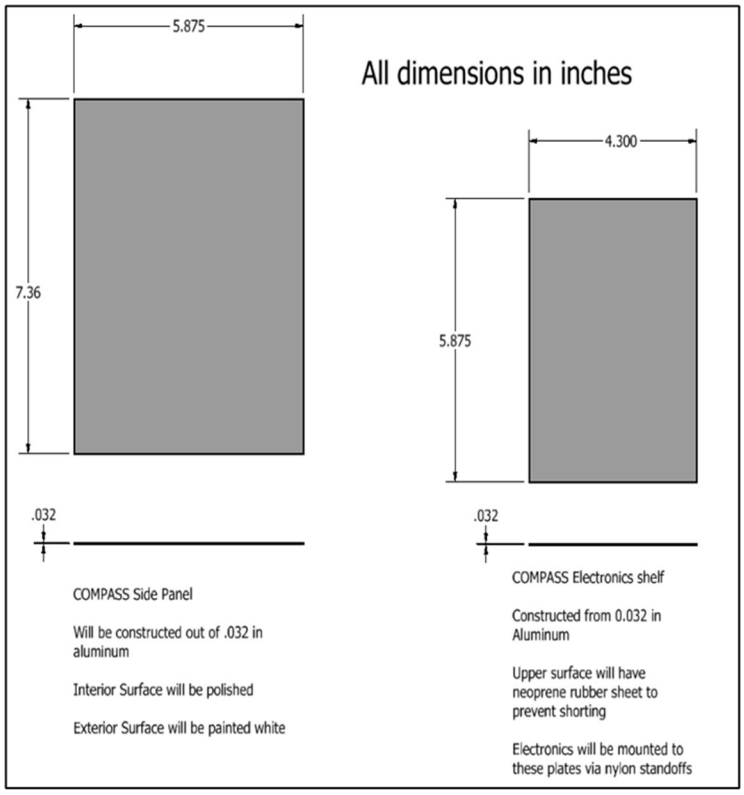


Figure 49: Dimensioned drawing of sheet metal side panels (left) and electronics mounting shelves (right).

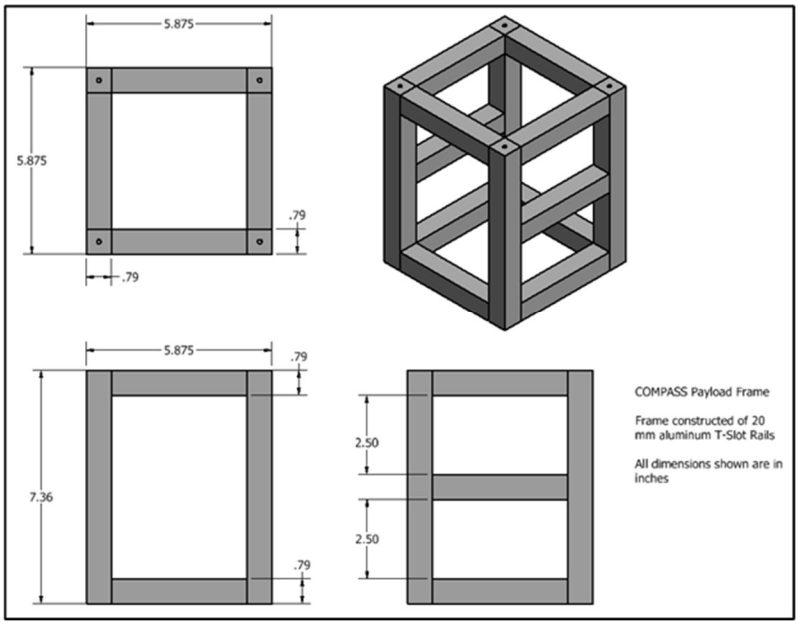


Figure 50: Main payload frame.

2. Payload interface to HASP

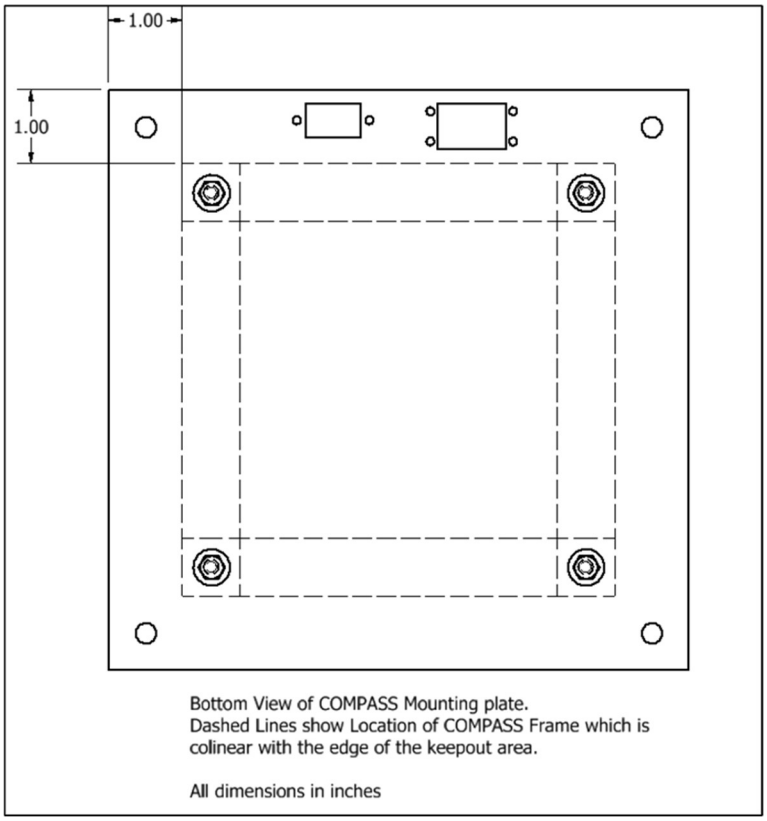


Figure 51: HASP mounting plate and locations of COMPASS frame relative to it.

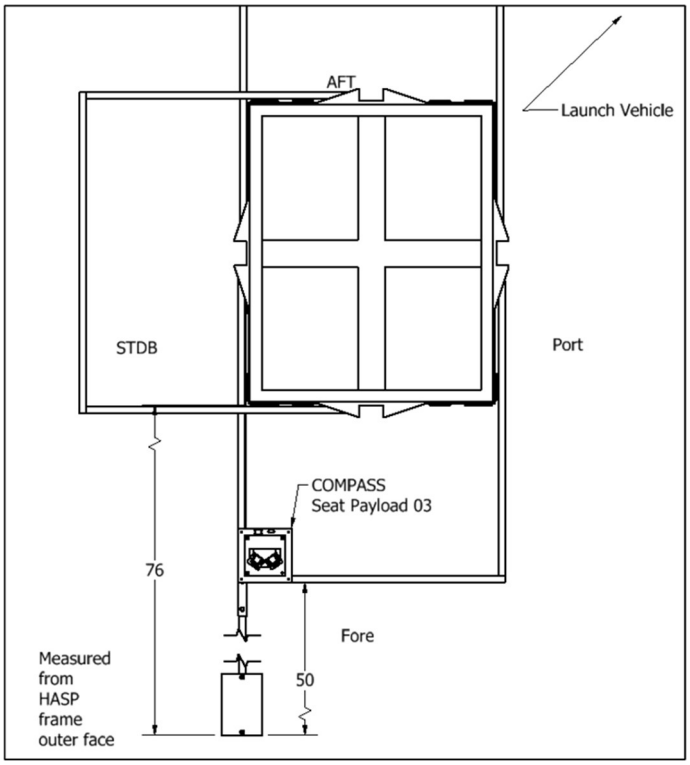


Figure 52: Diagram showing where the boom was placed relative to the HASP gondola and the main payload.

3. Boom/Boom Payload Drawings

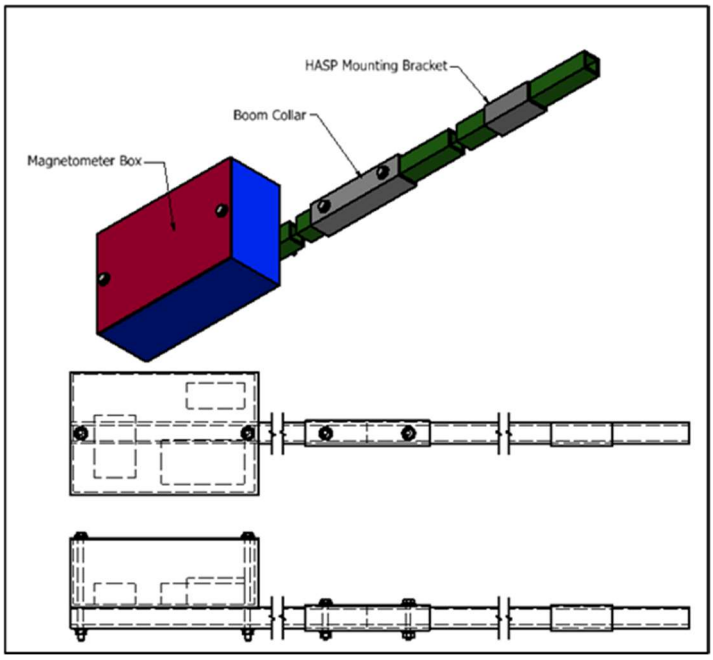


Figure 53: Arrangement of components of the boom and placement of external orientation system on the boom.

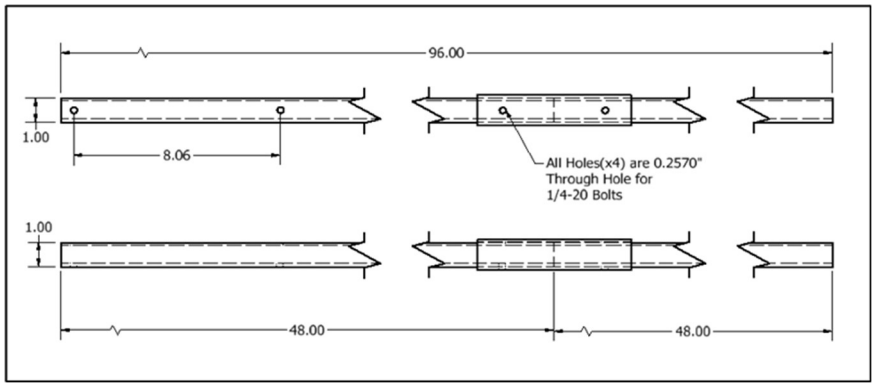


Figure 54: How sections of the boom were arranged and connected.

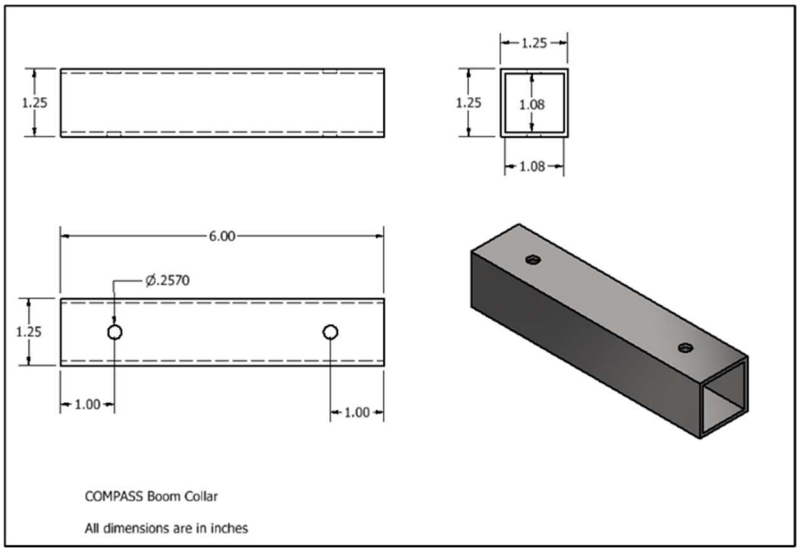
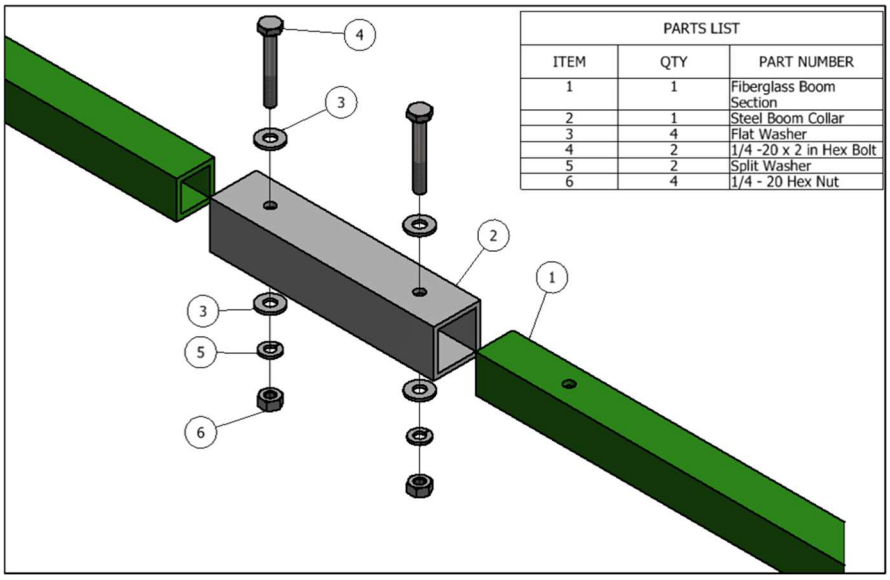


Figure 55: Dimensions of boom collar used to connect two pieces of fiberglass that comprised the boom.



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Fiberglass Boom Section
2	1	Steel Boom Collar
3	4	Flat Washer
4	2	1/4 -20 x 2 in Hex Bolt
5	2	Split Washer
6	4	1/4 - 20 Hex Nut

Figure 56: Drawing showing construction of boom and how two sections were connected

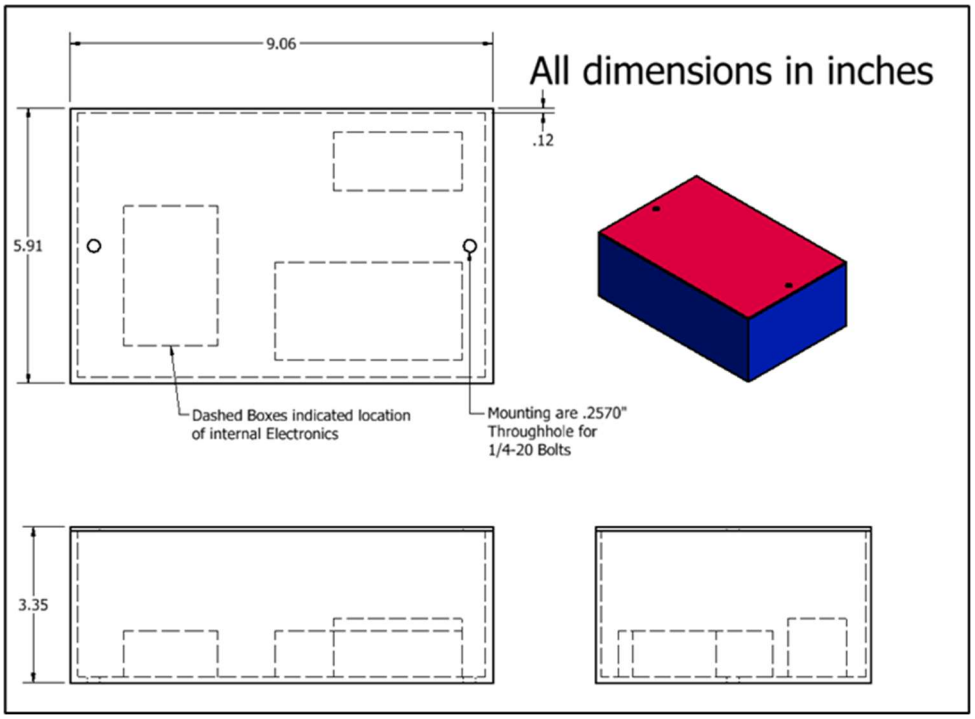


Figure 57: Interior arrangement and exterior dimensions of sensor box. Box lid was held to box by additional fasteners not pictured.

Appendix E: Weight Budget

The following tables are the weight budgets for the main payload, boom payload, and boom.

Main Payload Weight Budget

Component	**Mass (g)**	**Uncertainty (g)**
Camera mount	145	±5
Camera filters	20	±5
Payload walls (including connectors)	265	±5
Payload shelves	85	±5
Payload frame + bolts + washers	1130	±5
screws/standoffs	<1	±1
RPi + CamArray Hat	85	±5
Top shelf circuit board (includes accelerometer, level shifter, and GPS MOSFET)	20	±5
GPS + antenna	30	±5
Magnetometer	110	±5
Power converter board	30	±5
Bottom shelf circuit boards (includes ADC circuits and MOSFETs, bidirectional level shifter, and USBC breakout board)	55	±5
Wiring	30	±5
TOTAL:	**2006**	**±17**

Table 9: Total measured weight budget of the main payload.

Boom/Boom Payload Weight Budget

Component	**Mass (g)**	**Uncertainty (g)**
Fiberglass boom	1180	±5
Boom collar	280	±5
Bolts	100	±5
Zip ties	40	±5
Boom cable	225	±5
Electronics box	480	±5
Arduino Due (includes SD card shield with accelerometer and temperature sensor circuits)	55	±5
Magnetometer	110	±5
Boom circuit boards (includes ADC, 5V reference, and Max232 level shifter and bidirectional level shifter)	45	±5
Wires	15	±5
TOTAL:	**2530**	**±16**

Table 10: This table shows the measured weights of all the components making up the boom and the payload that was attached to the end of the boom.

Appendix F: Boom Stress Analysis

8 G vertical boom test:

1. A proof test was conducted on the boom.
    a. First, using the information from the weight budget the load of the boom and the sensor box was determined.
    b. Next, that load was multiplied by a factor of 8 to determine the load under 8G conditions
    c. Then that load was multiplied by a factor of 1.2 factor of safety to determine the desired load for the test. These values are show in the table below.
    d. The prototype boom and collar where then attached to the HASP frame using the upper payload boom brackets.
    e. Then weight loads were hung from the boom at the center of the boom (for the boom itself) and 1 in. from the end of the boom (to simulate the sensor box) to simulate the desired load. Note the applied load for the boom listed below includes the weight of the existing boom.
    f. This load was maintained for 3 min after which the boom was disassembled, and the boom sections and collars were inspected for any signs of damage.

	Budgeted Weight	**8*G Load**	**1.2 x Margin**	**Applied Load**
Sensor Box	714 gram	5712 gram	6854 gram	6860 gram
Boom	1540 gram	12320 gram	14784 gram	14790 gram

Table 11: Weight Loads for 8g vertical proof test.

4G Horizontal boom

	Budgeted Weight	**4*G Load**	**1.2 x Margin**	**Applied Load**
Sensor box	714 g	2856 g	3427 g	6860 g
Boom	1540 g	6120 g	7382 g	14790 g

Table 12: Loads for 4G horizontal proof test

2. The 4g horizon proof test was conducted in an identical manner to vertical test except for the base load being multiplied by a factor of 4 instead of 8 and the boom was rotated 90 degrees around its long axis when loaded so that the hanging loads would be in the direction of a horizontal load.

3. An image of the boom under load during the test is shown below in figure 50.

Figure 58: Boom under load during horizontal proof test.

ii. Results:

No failure or damage to the was noted following either test

Appendix G: Calibration Data Tables

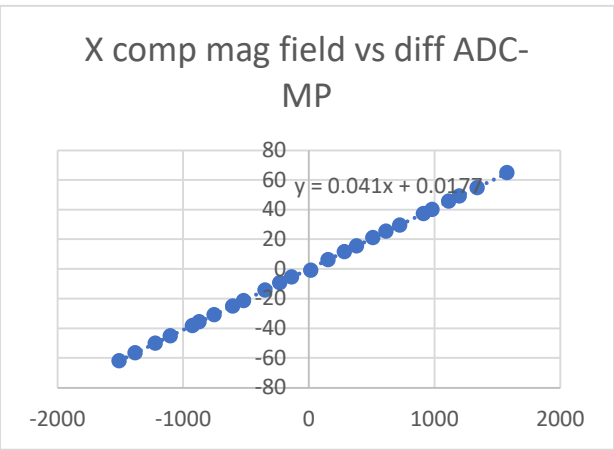
1. Magnetometer Calibration

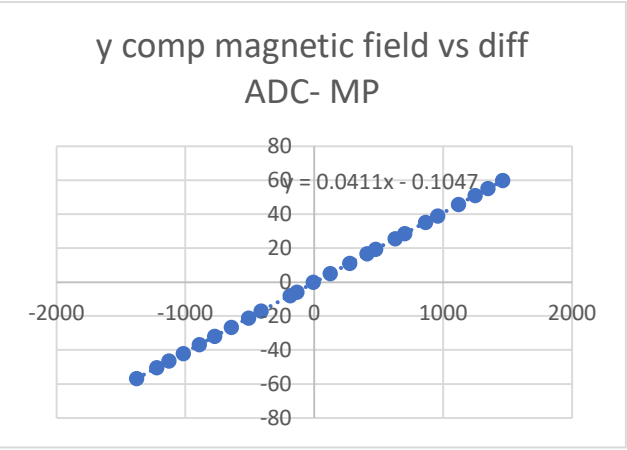
The following calibration was performed on both magnetometers for the x, y, and z components of each. The background magnetic field was taken with a lab magnetometer. Our magnetometer was then placed in a Helmholtz coil and the magnetic field was increased in each direction. Magnetic field readings from the lab magnetometer were taken and then the corresponding ADC values of the positive and negative outputs were taken. The differential ADC value was then plotted against magnetic field and the linear relationship was used to go between ADC and magnetic field.

Background	Helmoltz coil mag field	corresponding current for Helmholtz coil	magnetic field values	actual mag field values (uT)	Xplus	xminus	Vxplus	Vxminus	differential ADC Value	differential V	mV/uT
16.2	76.2	0.319533166	-60	-61.79	589	2098	0.72	2.56	-1509	-1.84	29.77828
**in up direc	71.2	0.298566423	-55	-56.39	654	2036	0.8	2.49	-1382	-1.69	29.96985
**Assuming	66.2	0.27759968	-50	-49.91	731	1952	0.89	2.38	-1221	-1.49	29.85374
X DIRECTION	61.2	0.256632936	-45	-45.13	791	1892	0.97	2.31	-1101	-1.34	29.692
	56.2	0.235666193	-40	-38.04	881	1807	1.08	2.21	-926	-1.13	29.70557
	51.2	0.21469945	-35	-35.58	906	1778	1.11	2.17	-872	-1.06	29.79202
	46.2	0.193732707	-30	-30.87	966	1719	1.18	2.1	-753	-0.92	29.8024
	41.2	0.172765964	-25	-24.95	1040	1646	1.27	2.01	-606	-0.74	29.65932
	36.2	0.151799221	-20	-21.3	1084	1604	1.32	1.96	-520	-0.64	30.04695
	31.2	0.130832477	-15	-14.25	1171	1519	1.43	1.85	-348	-0.42	29.47368
	26.2	0.109865734	-10	-9.33	1228	1460	1.5	1.78	-232	-0.28	30.01072
	21.2	0.088898991	-5	-5.42	1276	1412	1.56	1.72	-136	-0.16	29.5203
	16.2	0.067932248	0	-0.88	1353	1338	1.65	1.63	15	0.02	-22.7273
	11.2	0.046965505	5	6.34	1421	1268	1.73	1.55	153	0.18	28.39117
	6.2	0.025998762	10	11.72	1485	1202	1.81	1.47	283	0.34	29.01024
	1.2	0.005032018	15	15.55	1535	1155	1.87	1.41	380	0.46	29.58199
	3.8	0.015934725	20	21.1	1600	1090	1.95	1.33	510	0.62	29.38389
	8.8	0.036901468	25	25.41	1654	1040	2.02	1.27	614	0.75	29.51594
	13.8	0.057868211	30	29.67	1707	984	2.08	1.2	723	0.88	29.65959
	18.8	0.078834954	35	37.35	1802	891	2.2	1.09	911	1.11	29.71888
	23.8	0.099801697	40	40.23	1837	857	2.24	1.05	980	1.19	29.57992
	28.8	0.120768441	45	45.76	1902	790	2.33	0.97	1112	1.36	29.72028
	33.8	0.141735184	50	49.18	1946	749	2.38	0.92	1197	1.46	29.68686
	38.8	0.162701927	55	54.73	2017	679	2.46	0.83	1338	1.63	29.78257
	43.8	0.18366867	60	64.91	2135	558	2.61	0.68	1577	1.93	29.73348

Table 13: Example of the data taken from the Main payload X component calibration. This was done for the main payload Y and Z components, as well as the X, Y, and Z of the boom payload magnetometer as well. There were a total of 6 tables similar to this one.

The plots shown below are the plots of the main payload magnetometer calibration, with differential ADC on the x-axis and magnetic field on the y-axis.





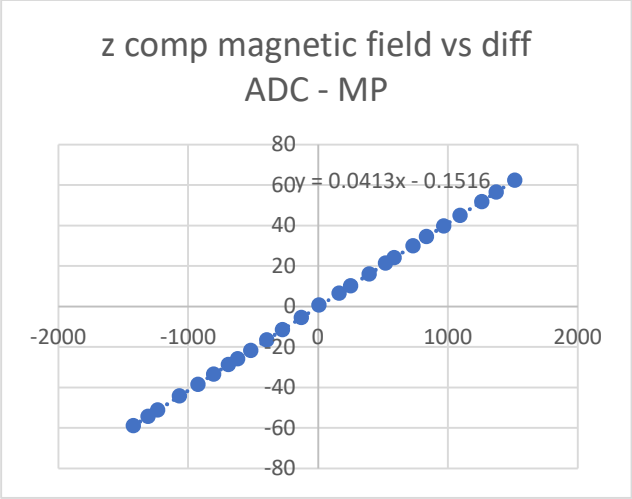


Figure 59: These plots show the magnetic field values vs the differential ADC for each component of the main payload magnetometer calibration.

The plots shown below are the plots of the boom payload magnetometer calibration, with differential ADC on the x-axis and magnetic field on the y-axis.

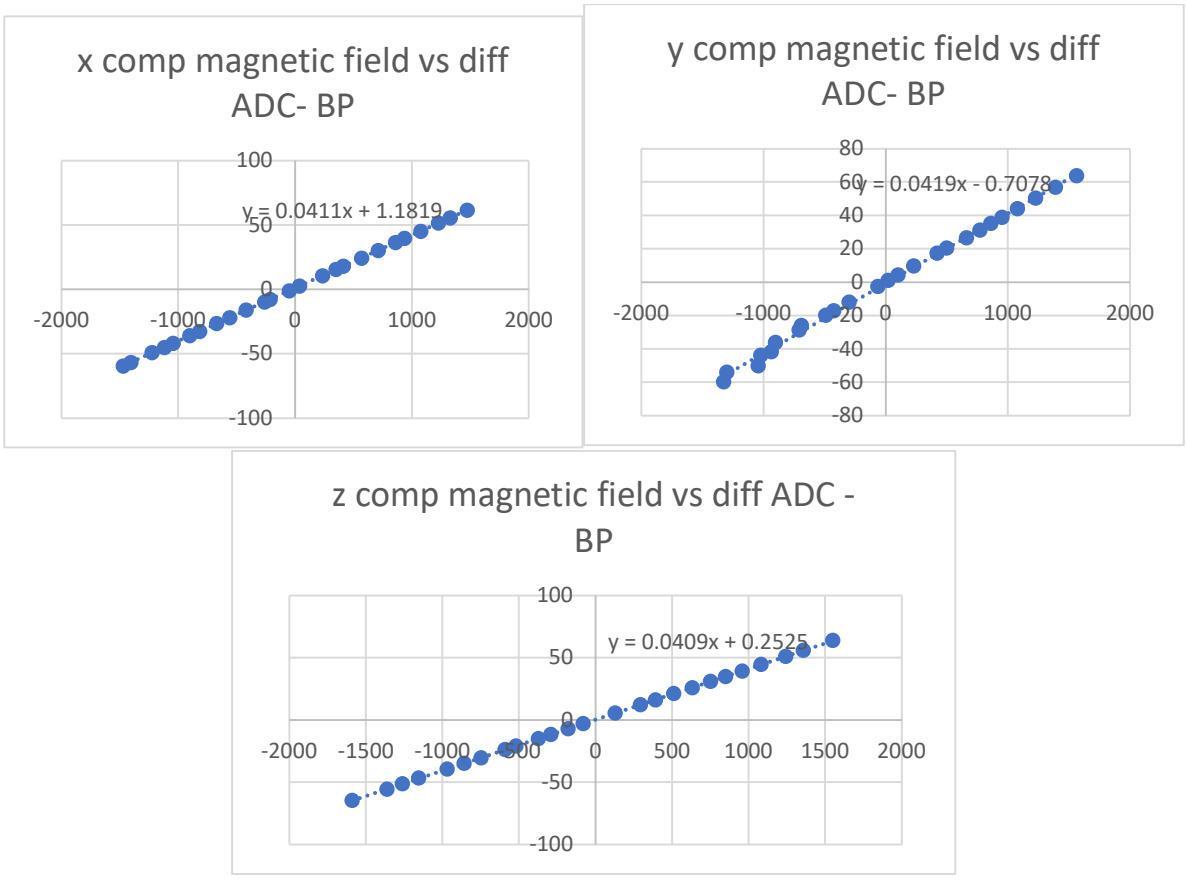


Figure 60: These plots show the magnetic field values vs the differential ADC for each component of the boom payload magnetometer calibration.

2. Temperature Calibration

The temperature calibration was a bit difficult to do physically because of how short the wires on the temperature sensor were. However, we took as many measurements as possible for the 3 main payload temperature sensors and then averaged the slope and intercepts of the lines of best fit to get the calibration equation for all of the temperature sensors. We did this because the values of slope and intercept we were getting were close in relation to each other and the temperature was not the most important feature of this payload and was mainly used to monitor general health.

Temp	Actual Temp	ADC	Voltage	Expected Voltage
-50				0.25
-45				0.3625
-40				0.475
-35				0.5875
-30				0.7
-25				0.8125
-20	-21.4	851	1.04	0.925
-15	-12.4	1032	1.24	1.0375
-10	-10	1055	1.29	1.15
-5				1.2625
0				1.375
5				1.4875
8	7.9	1382	1.69	1.555
10				1.6
15	14.8	1431	1.75	1.7125
25	21.5	1542	1.88	1.9375
30	29.7	1630	1.99	2.05
35	36.9	1793	2.19	2.1625
40	38.6	1825	2.23	2.275

Table 14: This table shows and example of the data taken for one of the temperature sensors we took while calibrating it. The targeted temperatures are in the left most column and the expected voltages based on the datasheet are in the right most column. As can be seen, the actual data taken was somewhat more sporadic because of the difficulty in actually taking the measurements.

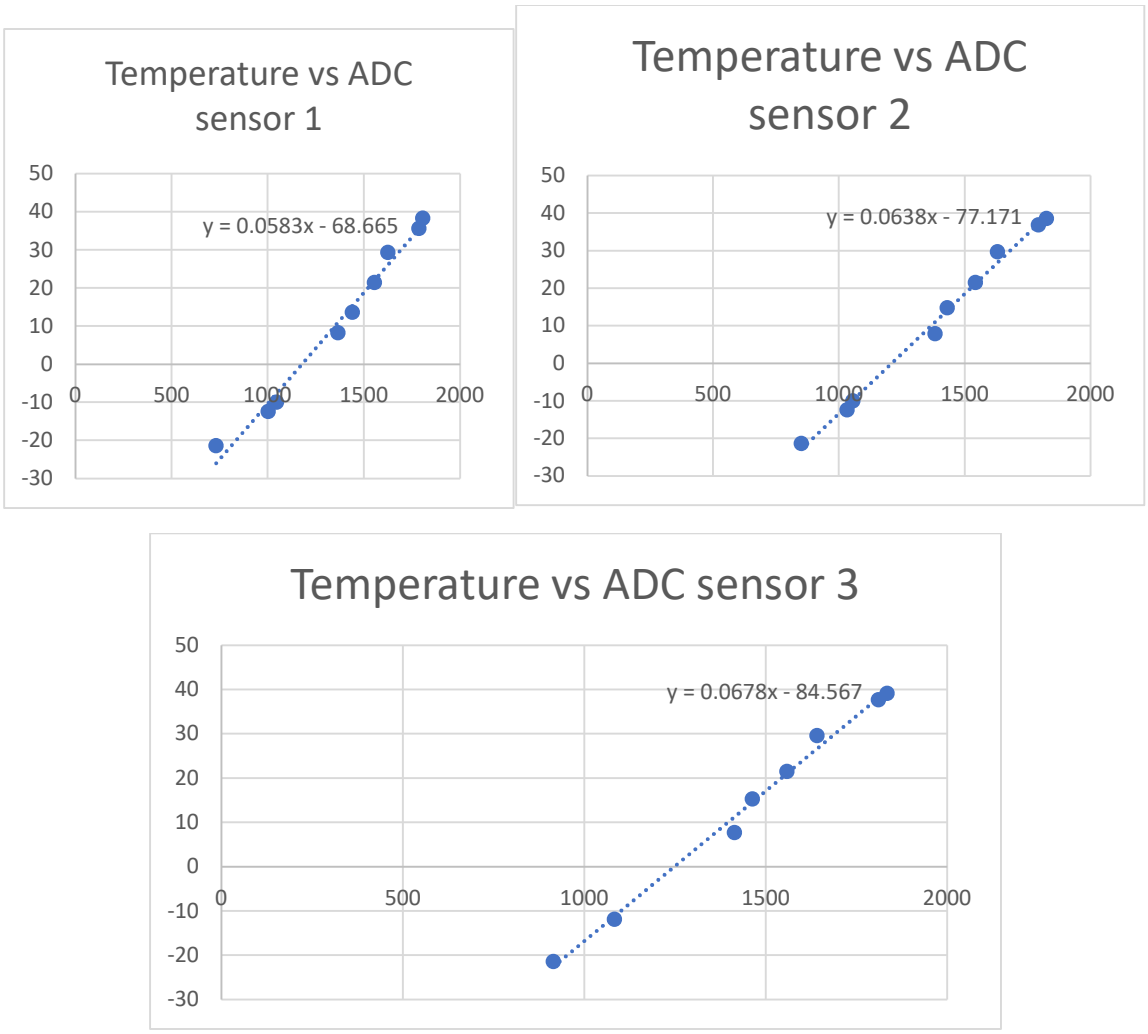


Figure 61: These plots show the temperature vs ADC value from the temperature sensor calibrations.

The equations from the above plots were averaged to give the equation below:

$$y = 0.0633x - 76.801$$

Appendix H: Downlink Packet

Key	**Example**
Start of Tel	“000000”
Payload ID	“3”
Data length	335
Time of Transmission	3740041
ORI	"ORI"
Set	57
Entry	360
"MAIN" Delim.	"MAIN"
Start Millis	1503504
End Milli	1503677
Temp 3	1752.0
Temp 4	1852.0
Temp 5	2000.0
Mag X +	1276.0
Mag X -	1396.0
Mag Y +	1448.0
Mag Y -	1229.0
Mag Z +	1584.0
Mag Z -	1084.0
Acc X	-24.0
Acc Y	0.0
Acc Z	1044.0
Latitude	3429.39685
Latitude Direction	N
Longitude	10413.39411
Longitude Direction	W
Altitude	1275.2
GPS Timestamp	050921_193033.00
FixQuality	2
SatelliteCount	12
Number Commands Received	0
Number Commands Successful	0
Number Commands Failed	0
Number Commands Garbled	0
Last Successful Command	BRS
Last Failed Command	
Last Command Received	BRS

Time Received	3543771
Time Flight Started	1626132454
Time power up	1626132454
Image Count	57
Millis Before Image	3735826
Millis After Image	3736057
"BOOM" delim.	"BOOM"
DAT	"DAT"
LastSynUID	85
SD Status	1
Start Millis	191002
End Millis	191004
Temp 1	1495
Temp 2	1593
MagX +	1486
MagX -	1198
MagY +	1874
MagY -	802
MagZ +	1101
MagZ -	1562
Acc X	-52
Acc Y	-28
Acc Z	1008
TimeSincPower	191502

Table 15: Table showing what was downlinked in each data packet and example data.

Appendix I: Uplink Command Packet

Communication Command Bytes			
Command Group	**Command**	**Hex Value**	**Description**
NULL	NULL Command	0xFF	Initialization value for Last Command Received (Does nothing)
Packet Request	Data Packet	0x00	Sends most recent sendable data packet
Set payload flight mode	Shutdown	0x05	Safely shut down the payload microcontroller

Table 16: Uplink Commands - Communication: Used to safely shutdown the main payload's microcontroller prior to a power cycle or when descending. The null command is defined to differentiate between repeated commands and determine those commands were received correctly

Instrumentation Command Bytes				
Command Group	**Command**	**Hex Value**	**Description**	**Requirements**
Power	Main Magnetometer (OS1) ON	0x11	Power ON the Main payload's Magnetometer (OS1)	12.3.1
	Main GPS ON (OS1)	0x16	Power ON the Main payload's GPS (OS1)	
	Boom Payload (OS2) ON	0x12	Power ON the entire Boom payload (OS2)	
	Main Magnetometer (OS1) OFF	0x19	Power OFF the Main payload's Magnetometer (OS1)	
	Main GPS OFF (OS1)	0x1E	Power OFF the Main payload's GPS (OS1)	
	Boom Payload (OS2) OFF	0x1A	Power OFF the entire Boom payload (OS2)	
Resets	Main Magnetometer (OS1) Reset	0x30	Powercycles the Main payload's Magnetometer	
	Main Payload GPS Reset	0x35	Powercycles the Main payload's GPS	
	Boom Payload Reset	0x36	Powercycles the entire Boom payload	12.3

Table 17: Uplink Commands – Instrumentation: Used to turn on/off power to specific components and initiate resets of reset-able components.

Uplink Command Wrapper			
Byte	**bits**	**Hex Value**	**Description**
1	0 : 7	0x01	Start of Heading (SOH)
2	0 : 7	0x02	Start of Text (STX)
3	0 : 7	Command Byte 1	Checksum
4	0 : 7	Command Byte 2	Actionable Command
5	0 : 7	0x03	End of Text (ETX)
6	0 : 7	0x0D	Carriage Return (CR)
7	0 : 7	0x0A	Line Feed (LF)

Table 18: Uplink Commands Wrapper – This is the standard uplink command wrapper. It contains the appropriate header, footer, and 2 command bytes as mandated by HASP.

Appendix J: Data Record Packet and Examples

1. Main payload

Key	Example
ORI	"ORI"
Set	57
Entry	360
"MAIN" Delim.	"MAIN"
Start Millis	1503504
End Milli	1503677
Temp 3	1752.0
Temp 4	1852.0
Temp 5	2000.0
Mag X +	1276.0
Mag X -	1396.0
Mag Y +	1448.0
Mag Y -	1229.0
Mag Z +	1584.0
Mag Z -	1084.0
Acc X	-24.0
Acc Y	0.0
Acc Z	1044.0
Latitude	3429.39685
Latitude Direction	N
Longitude	10413.39411
Longitude Direction	W
Altitude	1275.2
GPS Timestamp	050921_193033.00
FixQuality	2
SatelliteCount	12

Table 19: Shows what was contained in each data entry string stored on the main payload.

The following is an example of the saved main payload data packet. In each file contained on the main payload, there was a heading of payload health data, then rows of data labelled “ORI” for orientation data.

```
Commands Received: 0
Commands Successful: 0
Commands Failed: 0
Commands Garbled: 0
Last Succesful Command:
Last Failed Command:
Last Command Received:
Time Received:
Time Flight Started: 1626132454.446765
Time of Power Up: 1626132454.4443648
Image Count: 2
Millis Before Image: 572467
Millis After Image: 572661
SYN,3,391401,U
ACK,3,387079,x
ORI,3,1,MAIN,392623,392990,1689.0,1754.0,1890.0,1276.0,1394.0,1446.0,1226.0,1586.0,1090.0,-8.0,-12.0,1044.0,3429.39649,N,10413.39428,W,1267.2,050921_191202.00,1,12
ORI,3,2,MAIN,393123,393319,1696.0,1760.0,1885.0,1288.0,1391.0,1444.0,1226.0,1583.0,1089.0,-8.0,-12.0,1028.0,3429.39647,N,10413.39425,W,1267.3,050921_191203.00,1,12
ORI,3,3,MAIN,393623,393747,1755.0,1850.0,1966.0,1280.0,1394.0,1444.0,1226.0,1585.0,1090.0,-4.0,-24.0,1048.0,3429.39647,N,10413.39425,W,1267.3,050921_191203.00,1,12
ORI,3,4,MAIN,394123,394330,1729.0,1808.0,1930.0,1282.0,1393.0,1444.0,1226.0,1588.0,1090.0,-4.0,-4.0,1028.0,3429.39646,N,10413.39422,W,1267.3,050921_191204.00,1,12
ORI,3,5,MAIN,394623,394741,1725.0,1807.0,1962.0,1282.0,1390.0,1442.0,1226.0,1582.0,1086.0,8.0,-8.0,976.0,3429.39646,N,10413.39422,W,1267.3,050921_191204.00,1,12
ORI,3,6,MAIN,395123,395331,1726.0,1808.0,1930.0,1280.0,1394.0,1444.0,1236.0,1582.0,1084.0,-8.0,-16.0,1016.0,3429.39645,N,10413.39421,W,1267.3,050921_191205.00,1,12
ORI,3,7,MAIN,395623,395741,1733.0,1801.0,1965.0,1282.0,1394.0,1447.0,1225.0,1586.0,1084.0,-12.0,-16.0,1052.0,3429.39645,N,10413.39421,W,1267.3,050921_191205.00,1,12
ORI,3,8,MAIN,396123,396304,1743.0,1818.0,1941.0,1288.0,1394.0,1444.0,1226.0,1590.0,1089.0,-8.0,-12.0,1040.0,3429.39643,N,10413.39419,W,1267.3,050921_191206.00,1,12
```

Figure 62: Main payload file data example

2. Boom Payload

Key	Example
ORI	"ORI"
LastSynUID	85
SD Status	1
Start Millis	191002
End Millis	191004
Temp 1	1495
Temp 2	1593
MagX +	1486
MagX -	1198
MagY +	1874
MagY -	802
MagZ +	1101
MagZ -	1562
Acc X	-52
Acc Y	-28
Acc Z	1008
TimeSincPower	191502

Table 20: Shows what was saved in each data string on the boom payload SD card.

The following is an example of the boom payload data that was saved on the Arduino SD card. Each file contained on the boom payload began with the SYNC and ACK numbers that were transmitted between the boom and main payload and the timestamps of each. After, each data string is denoted by “ORI.”

```
SYN,6,949635,Z
ACK,6,945287,z
ORI,6,1,945967,945969,1603,1670,1302.00,1370.00,1520.00,1155.00,1656.00,1007.00,-52.0000,-12.0000,1020.0000,1125967
ORI,6,2,946467,946469,1588,1691,1299.00,1369.00,1514.00,1156.00,1667.00,1020.00,-44.0000,-20.0000,1036.0000,1125967
ORI,6,3,946967,946969,1602,1730,1300.00,1370.00,1514.00,1151.00,1660.00,1010.00,-56.0000,-12.0000,1040.0000,1125967
ORI,6,4,947467,947469,1597,1689,1302.00,1368.00,1514.00,1154.00,1658.00,1008.00,-52.0000,-12.0000,1032.0000,1125967
ORI,6,5,947967,947969,1594,1687,1309.00,1362.00,1515.00,1152.00,1658.00,1013.00,-48.0000,-12.0000,1012.0000,1125967
ORI,6,6,948467,948469,1589,1700,1297.00,1366.00,1516.00,1155.00,1657.00,1012.00,-52.0000,-12.0000,1032.0000,1125967
ORI,6,7,948967,948969,1591,1674,1304.00,1363.00,1516.00,1155.00,1657.00,1034.00,-56.0000,-20.0000,1032.0000,1125967
ORI,6,8,949467,949469,1587,1700,1295.00,1364.00,1512.00,1147.00,1666.00,1013.00,-56.0000,-16.0000,1004.0000,1125967
ORI,6,9,949967,949969,1590,1705,1301.00,1369.00,1515.00,1155.00,1656.00,1013.00,-48.0000,-24.0000,1008.0000,1125967
ORI,6,10,950467,950469,1575,1681,1304.00,1369.00,1522.00,1151.00,1655.00,1022.00,-52.0000,-20.0000,1032.0000,1125967
```

Figure 63: Boom Payload data file example.

Appendix K: COMPASS Flight Data Documentation

Processed data:

The COMPASS flight data arrays are saved as .npy arrays that have already been processed and saved.The arrays that are just float type data and not datetime objects or rotation objects have also been converted to .txt files and saved.

To import a .npy array (in this example an array called “variable.npy”), you need numpy already imported (usually as np), then type the following:

```
name_of_variable = np.array([])

name_of_variable = np.load('variable.npy')
```

The following arrays include all of the pieces of the COMPASS processed data that have been used in the data analysis code.

1. Main payload processed data: the full main payload data set has 45805 datapoints.

Array name	Description	Length of array
actual_alt.npy	the altitudes corresponding to the main payload data points, taken at 1 sec intervals. These were taken from the HASP GPS.	45805
actual_lat.npy	the latitudes corresponding to the main payload data points, taken at 1 sec intervals. These were taken from the HASP GPS.	45805
actual_lon.npy	the longitudes corresponding to the main payload data points, taken at 1 sec intervals. These were taken from the HASP GPS.	45805
sub_calAccelX.npy	X component of accelerometer in Gs. The “sub” in the first part of the following names means the data set corresponds to the 1 second intervals and is a subset of the full MP dataset (the MP actually took data more quickly than 1/sec)	45805
sub_calAccelY.npy	Y component of accelerometer in Gs	45805
sub_calAccelZ.npy	Z component of accelerometer in Gs	45805
sub_calTemp3.npy	calibrated temperature data from temp sensor on main payload magnetometer	45805
sub_calTemp4.npy	calibrated temperature data from temp sensor on main payload 12V converter	45805

sub_calTemp5.npy	calibrated temperature data from temp sensor on main payload 5V converter	45805
sub_magxplus.npy	x plus component of magnetometer, uncalibrated ADC values	45805
sub_magxminus.npy	x minus component of magnetometer, uncalibrated ADC values	45805
sub_magyplus.npy	y plus component of magnetometer, uncalibrated ADC values	45805
sub_magyminus.npy	y minus component of magnetometer, uncalibrated ADC values	45805
sub_magzplus.npy	z plus component of magnetometer, uncalibrated ADC values	45805
sub_magzminus.npy	z minus component of magnetometer, uncalibrated ADC values	45805
sub_calX.npy	calibrated x component of main payload magnetometer, in uT	45805
sub_calY.npy	calibrated y component of main payload magnetometer, in uT	45805
sub_calZ.npy	calibrated z component of main payload magnetometer, in uT	45805
new_MP_unix.npy	unix time stamps corresponding to every MP datapoint at 1 sec intervals. These are taken from the GPS and are the true time	45805
MP_az_el.npy	azimuth and elevation of the payload pointing vector, calculated using the main payload data	45805
MP_date_objects.npy	datetime objects corresponding to each MP datapoint	45805
MP_to_BP_mask.npy	Boolean mask that if applied to any array with full 45805 datapoint length will give just the points corresponding to BP datapoints/images	45805
p_to_e_A_MP.npy	“payload to earth” rotation objects obtained by aligning the expected magnetic field readings given by IGRF/expected gravitational acceleration to the measured mag/accel values. These were calculated using MP values but correspond only to the subset that has a matching boom datapoint. Uses the function referenced below*	250

p_to_e_rsq_MP.npy	Root mean square distance (weighted) between the given set of vectors after alignment for MP data, but only subset with corresponding boom datapoints	250
p_to_E_MP_full.npy	Rotation objects calculated with MP data, but full set	45805
p_to_E_rsq_MP_full.npy	Root mean square distance between aligned vectors for MP, full set	45805
north_IGRF_MP.npy	The north component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	45805
east_IGRF_MP.npy	The east component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	45805
down_IGRF_MP.npy	The down component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	45805
Total_IGRF_MP.npy	Total magnetic field based on position of the payload calculated from IGRF data in uT	45805

*
https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.align_vectors.html#scipy.spatial.transform.Rotation.align_vectors

2. Boom payload processed data: boom payload dataset only has 250 datapoints bc SD card didn't work during flight but we still had the downlinked data.

Array name	**Description**	**Length of array**
BP_data_object.npy	Datetime objects corresponding to each boom datapoint	250
BP_unix.npy	Unix timestamps corresponding to each boom datapoint	250
BP_sub_alt.npy	Altitude corresponding to boom payload datapoints	250
BP_sub_lat.npy	latitude corresponding to boom payload datapoints	250
BP_sub_lon.npy	longitude corresponding to boom payload datapoints. These are taken from the same data points as the main payload GPS data, but this array only has the set of GPS data corresponding to boom data	250
accel_x.npy	X component of BP accelerometer	250
accel_y.npy	Y component of BP accelerometer	250
accel_z.npy	Z component of BP accelerometer	250

calTemp1.npy	Calibrated temperatures from the BP magnetometer temp monitor	250
calTemp2.npy	Calibrated temperatures from the BP arduino temp monitor	250
calX_mag.npy	Calibrated x component BP magnetometer values in uT	250
calY_mag.npy	Calibrated y component BP magnetometer values in uT	250
calZ_mag.npy	Calibrated z component BP magnetometer values in uT	250
mag_x_minus.npy	Uncalibrated ADC value for x minus component of BP magnetometer	250
mag_x_plus.npy	Uncalibrated ADC value for x plus component of BP magnetometer	250
mag_y_minus.npy	Uncalibrated ADC value for y minus component of BP magnetometer	250
mag_y_plus.npy	Uncalibrated ADC value for y plus component of BP magnetometer	250
mag_z_minus.npy	Uncalibrated ADC value for z minus component of BP magnetometer	250
mag_z_plus.npy	Uncalibrated ADC value for z plus component of BP magnetometer	250
sun_az_el.npy	Azimuth and elevation of the Sun at each datapoint for which there's an image/boom payload datapoint	250
bool_array_cam0.npy	Boolean array mask that when applied to an array length 250 will give only the resulting points that correspond to the sun being detected in cam0	250
bool_array_cam1.npy	Boolean array mask that when applied to an array length 250 will give only the resulting points that correspond to the sun being detected in cam1	250
undistorted_left_points.npy	Pixel coordinates from cam1 after they have been undistorted using the camera matrix and distortion coeffs	50
undistorted_right_points.npy	Pixel coordinates from cam0 after they have been undistorted using the camera matrix and distortion coeffs	90
distorted_left.npy	Pixel coordinates from cam1 before being undistorted using camera matrix	50
distorted_left.npy	Pixel coordinates from cam0 before being undistorted using camera matrix	90
payload_to_earth_A.npy	Rotation objects obtained by aligning the expected magnetic field readings given by	250

	IGRF/expected gravitational acceleration to the measured mag/accel values. These were calculated using BP values	
p_to_e_rsq.npy	Root mean square distance (weighted) between the given set of vectors after alignment for BP data2	250
north_IGRF_BP.npy	The north component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	250
east_IGRF_BP.npy	The east component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	250
down_IGRF_BP.npy	The down component of the magnetic field based on the position of the payload calculated from the IGRF data in uT	250
Total_IGRF_BP.npy	Total magnetic field based on position of the payload calculated from IGRF data in uT	250

Other files:

- Camera matrices.txt : gives the camera matrices and distortion coefficients for both cameras. The camera matrix is a 3x3 matrix and the distortion coefficients are a 1x5 matrix. These were given by running the camera calibration code on the RPi multiple times and averaging the output matrices.
- The matrices are pasted below:
- **cam0Mtx** = ([[488.7601901, 0., 633.776699], [0., 488.8880742, 386.0706219], [0., 0., 1.]])
- **cam0DistCoeffs** = ([[0.235529324, -0.290618838, 8.398753206E-4, 6.707909302E-4, 0.0924374093]])
- **cam1Mtx** = ([[487.2260587, 0., 672.2442943], [0., 487.85669902, 376.668898], [0., 0., 1.]])
- **cam1DistCoeffs** = ([[0.2673417117, -0.3241992797, -5.82929372E-4, 0.0010265586, 0.1030181927]])

Software:

This is a list of the jupyter notebooks (.ipynb files) contained in the vault and an explanation of what they pretty much do.

General Notation/array/vector names:

- **"A" Matrix:** Payload to earth rotations, already calculated for both boom and main payload. Inverse goes from earth to payload.
- **P_c:** sun position from camera in CAMERA FRAME

- **P_p:** sun position from camera in PAYLOAD FRAME
- **P_e:** sun position from camera in EARTH FRAME
- **T_e:** known sun position from date and time in EARTH FRAME
- **T_p:** known sun position from date and time in PAYLOAD FRAME
- **"C" matrix:** rotation from payload to camera frame.

Notebooks and descriptions:

- **MP data analysis.ipynb:** This notebook loads in a bunch of the already processed ("processed" meaning correct length/set of data) data and uses it to get the IGRF readings corresponding to the main payload data. It then calculates the rotation vectors between the calculated and measured mag/accel readings. These rotations are then applied to the pointing vector of the payload in the earth frame. This is then converted to azimuth and elevation to get the az/el of the payload using MP data.
    - Dependencies: GS_functions.py*
- **BP data analysis.ipynb:** This notebook does the same basic thing as **MP data analysis.ipynb** but with boom payload data.
    - Dependencies: GS_functions.py*
- **camera analysis MP.ipynb:** uses the processed MP data and the rotations between payload and earth (from previous MP notebook) to attempt to find rotations between camera frame and payload frame (C matrix). This was attempted using the physical measurements of the camera frame. This is also where we used the gs functions called "gs.cam0_pix_to_xyz" and "gs.cam1_pix_to_xyz" to get the undistorted pixels into the camera frame/3 dimensions. We're not 100% sure if this function is correct and more research may have to be done on it. We then tried to use the C rotations and the A rotations to rotate the sun position from pixel coordinates between frames and compare them. This didn't really get v far and we didn't have time to really troubleshoot this.
    - Dependencies: GS_functions.py*
- **Camera analysis BP.ipynb:** basically the same intended function as above but for the boom payload data except that we tried to find the C matrices in a different way (noted in the comments). It's also incomplete because it wasn't a priority because we couldn't get it to work for the main payload.
    - Dependencies: GS_functions.py*
- **Plotting.ipynb:** basically just plots all the stuff I had, this is what I used to make the poster plots.
- **GS_functions.py:** this is a .py file used only for imports in the above notebooks. Written by Harrison Gietz and has a lot of useful functions for analyzing the data. This file also needs the file "igrf_utils.py" which is why that file is included also.
- **MP full set array processing.ipynb:** this is a jupyter notebook that was used to take HASP GPS data and interpolate it to give values every 1 second (which resulted in the 45805 data points). This notebook also selects the main payload datapoints that correspond to these every second GPS measurements. The arrays that result from this are already saved as .npy files in the processed data arrays and this notebook shouldn't need to be touched.

- **Filter_imgs_w_dcoeff-C.ipynb:** this notebook was used to find the center pixel of the sun for the images where the Sun was detected. This *mostly* worked, but I did have to go back and pick out images where it was subtracting the background but it was detecting another bright point and not the sun, so I also had to manually take these pixel points out of the arrays. The resulting arrays of undistorted pixel coordinates are already included in the processed data.
- **chessCalCode.py:** This code was used on the Pi to determine the camera matrices and distortion coefficients. This link contains the references about this code: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html