Z-Axis Stabilized High Altitude Balloon Instrumentation Platform

Steve Giewont[1], Lincoln Glauser[2], Connor Goldberg[3], Sydney Kaminski[4], Ian Prechtl[5], Chris Schwab[6], Adam Steenkamer[7], Matthew Zachary[8]
Rochester Institute of Technology, Rochester, NY, 14623

[1] Steve Giewont, Electrical Engineering, for first author.
[2] Lincoln Glauser, Electrical Engineering, for second author.
[3] Connor Goldberg, Electrical Engineering, for third author.
[4] Sydney Kaminski, Mechanical Engineering, for fourth author.
[5] Ian Prechtl, Mechanical Engineering, for fifth author.
[6] Chris Schwab, Electrical Engineering, for sixth author.
[7] Adam Steenkamer, Electrical Engineering, for seventh author.
[8] Matthew Zachary, Electrical Engineering, for eighth author.

I. Abstract

Many High Altitude Balloon (HAB) projects have a reputation for being extremely simple and with minimal functionality. This High Altitude Balloon Instrumentation Platform combines many subsystems together to form a more advanced platform that can be used for future experiments in near space environments. To better understand the environment where these experiments will be performed, internal and external sensors were placed on the HAB. These sensors include temperature, pressure, humidity, and GPS. In addition, cameras were used to capture imagery of the environment. For all of these sensor nodes, including the cameras, the data was saved locally onto SD cards and transmitted via 2m/70cm amateur radio bands down to Earth. In order to provide platform stabilization in the z-axis, a stabilization system was developed consisting of an Inertial Measurement Unit (IMU) and a reaction wheel. This, in theory, would allow for more stable video capture and in the future would allow for the placement of a telescope for precise image captures from the stratosphere. The launch was completed as an overall success, however it had multiple subsystem anomalies. These anomalies consisted of a configuration issue with the GPS preventing some data to be acquired for the entire duration of the flight, some wiring issues which prevented the acquisition of some additional data from two sensor nodes, and an unknown issue which prevented the operation of reaction wheel controller during the flight. With the exception of these anomalies, that will be addressed and resolved for future missions, the platform was recovered and a large amount of sensor information was captured and analyzed. The platform was proven to be a successful starting point for future additions and improvements to the project.

II. Introduction

High altitude balloons (HAB) are used by both researchers and hobbyists to explore near space altitudes, while operating within an affordable budget. The RIT Space Exploration (SPEX) group performs high altitude balloon launches. They plan launches into a near space environment (>100,000 ft.) in order to test CubeSat systems. Instrumentation for such flights can be designed to collect, record, and transmit various sensor data, including temperature, pressure, and images. An HAB instrumentation platform has been constructed to collect this vital data along with transporting experimental subsystem modules for future operations. There have been platforms and balloon launches in the past at RIT (e.g., METEOR 2005), but they were not successful and did not provide a great deal of functionality.

The goals of this project were to create and test an instrumentation platform to acquire and transmit telemetry information, location information, and video. It was to interface with GPS and transceivers, receive and execute ground commands, and be stabilized in one axis. Captured data includes temperature, pressure, inertial measurements, and digital images. An analog video stream is transmitted to the ground, with select sensor data overlaid on the captured video. Captured sensor data is stored on local SD cards and packaged for telemetry transmissions. A reaction wheel assembly has been designed and implemented to stabilize the instrumentation platform in the z-axis. The resulting design forms a functional prototype, which should operate nominally up to 120,000 feet in elevation for at least 3 hours. There is a ground recovery signaling system (GRSS) that should be operational for at least 24 hours after launch. The work has been split across Data Acquisition and Control Systems (DAQCS) and Communication System (COMMS) sub-teams. The resulting design and prototype will be well documented and tested with the launch of the functioning system on a HAB mission.

III. Communication System

The goal of this project is to create a system to acquire and transmit location and telemetry information and video, interface with GPS and transceivers, and receive and execute ground commands. The system is incorporated with the Data Acquisition and Control System (DAQCS) portion of the project.

A. Communication System Hardware

The main processing in the communication system of the platform is performed on a Raspberry Pi Zero. It sits on a custom printed circuit board (PCB) that contains various sensors, devices, and power management. Figure 1 shows a block diagram of the communication system.

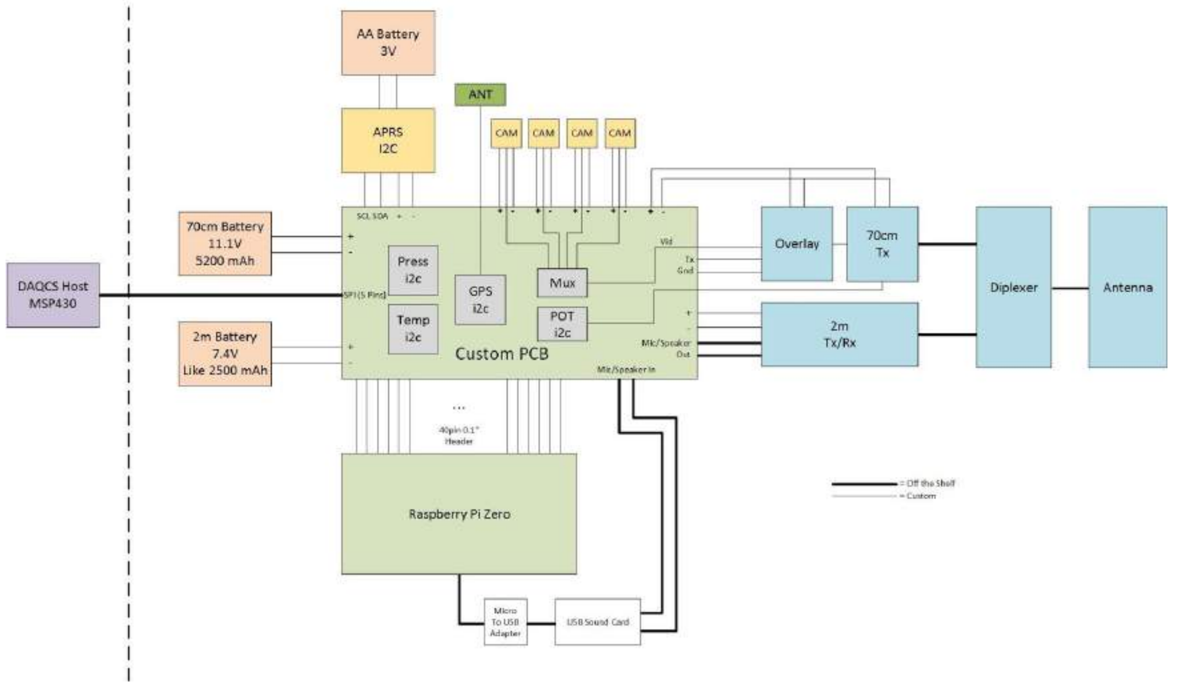

Figure 1. Communication System Block Diagram.

There is a standalone APRS (Automatic Packet Reporting System) that transmits the platform location to the ground. There is also a 70cm Amateur TV (ATV) transmitter to transmit analog video to the ground and a 2m transceiver for telemetry and commands. The ATV transmitter and the transceiver are attached to the same antenna with a diplexer as shown in Fig. 2.

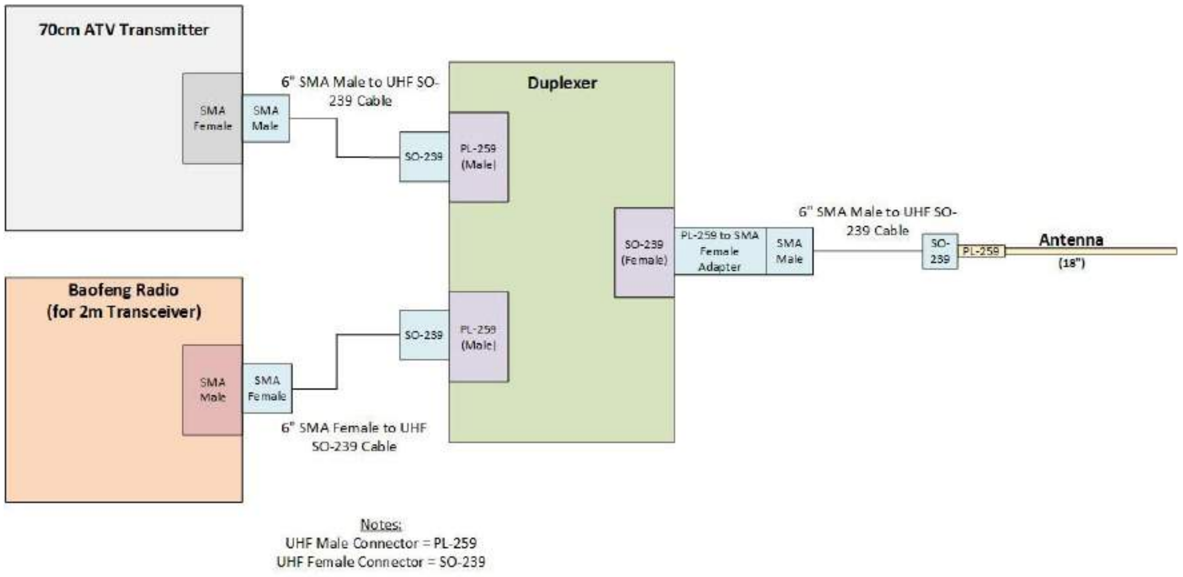
70cm ATV Transmitter
SMA Female
SMA Male
6" SMA Male to UHF SO-239 Cable
SO-239
PL-259 (Male)
Duplexer
SO-239 (Female)
PL-259 to SMA Female Adapter
SMA Male
6" SMA Male to UHF SO-239 Cable
SO-239
PL-259
Antenna
(18")
Baofeng Radio (for 2m Transceiver)
SMA Male
SMA Female
6" SMA Female to UHF SO-239 Cable
SO-239
PL-259 (Male)
Notes:
UHF Male Connector = PL-259
UHF Female Connector = SO-239

Figure 2. Platform RF Connections.

1. 2m Chain Hardware

The main communication with the platform takes place on the 2m Amateur Radio band. A handheld VHF/UHF radio transceiver (Baofeng UV-5RA) is used as a transceiver. The radio has mic in and speaker out ports that are used to send audio in to be transmitted and to bring out received audio. With audio cables, audio breakout on the custom PCB, and a USB sound card, the Raspberry Pi Zero is attached to the radio. Software on the platform is used to transmit and receive audio tones. For transmission, telemetry is sent from the platform to the ground station about once a minute. This is a stream of all of the sensor data on the platform (temperature sensors, pressure sensors, power monitor information, IMU information, and more). When the 2m transceiver is not transmitting, it is waiting for incoming commands. Operators can send various commands from the ground station to the platform; these commands include the following: changing what sensor information is shown on the video overlay, changing which analog video camera is used for the video transmission, commanding the reaction wheel to turn the platform, power cycling various hardware, and more. Figure 3 shows a block diagram of the hardware set-up for the 2m chain.

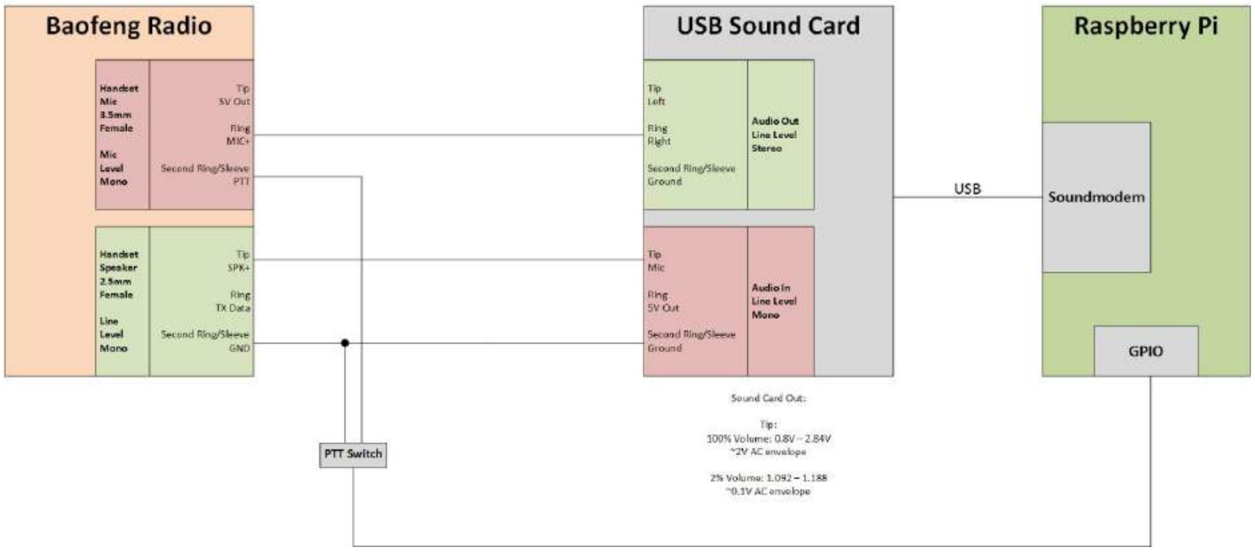

Figure 3. 2m Chain Hardware.

2. 70cm Chain Hardware

One of the customer requirements of the instrumentation platform is to provide a real-time, live, video stream from the platform to the base station. Ideally this stream would be of digital video, however there currently are not any solutions that could transmit information at a high enough bandwidth to carry the digital video stream required. To accomplish the task of providing a video stream, the ATV radio bands are used.

Four analog NTSC cameras are each connected to an analog video multiplexer which can be digitally controlled via GPIO pins from the Raspberry Pi. The output of the multiplexer provides the data into an analog video on-screen display (OSD) board (specifically the OSD-232 board). The Raspberry Pi can provide data to the OSD via serial communication so that miscellaneous sensor data and telemetry can be overlaid on top of the camera image. The output of the OSD then is input into an ATV transmitter that was purchased. For it to be possible to transmit both the ATV image and the data over the 2m interface, a diplexer is used before the antenna so there is no interference on the signals.

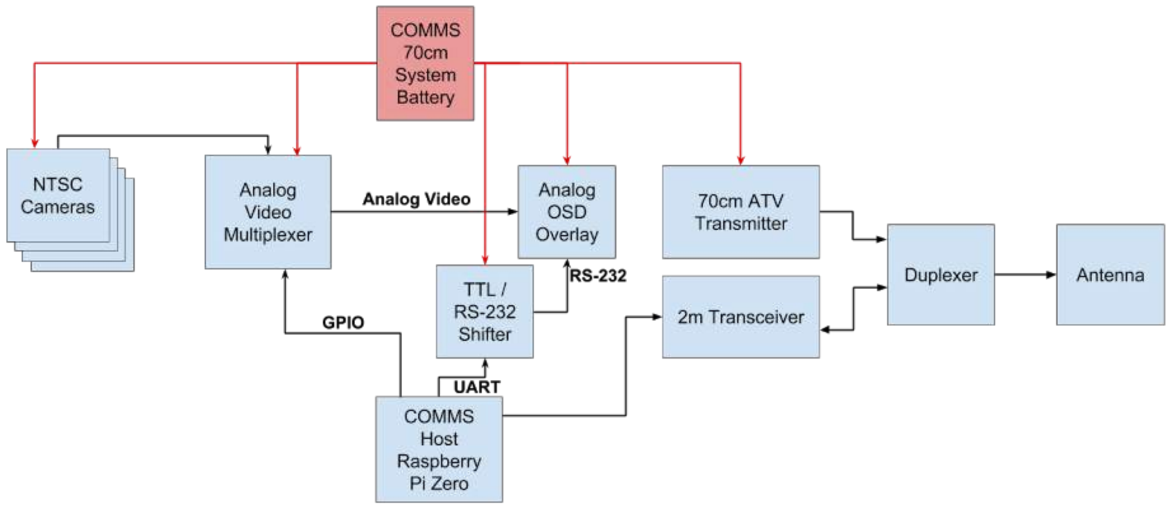

Figure 4. 70cm Chain Hardware.

3. *Tracksoar APRS*

In order to geolocate the HAB, a Tracksoar branded APRS (Automatic Packet Reporting System) was purchased and installed (see Fig. 5). The Tracksoar has a built-in Ublox MAX-M8Q GPS, capable of providing latitude, longitude, and altitude measurements for the purposes of locating the HAB in space. In order to transmit this GPS data to the ground, the 2m Amateur Band is used, on the APRS network frequency (144.390 MHz).

Figure 5. Tracksoar APRS.

The APRS is a network of repeaters and beacons, within a local geographic area, all operating on the APRS frequency. A given node ("digipeater") that receives a message from another node (in this case, the HAB), will pass the message it receives to neighboring digipeaters. These neighbors will do the same, and eventually, the message will be broadcasted to many nodes within a large radius. Therefore, this message can travel very far geographically, to nodes that aren't even within receiving range of the HAB.

Consequently, the HAB can report its position to the RIT METEOR lab even if it travels miles or even hundred of miles away. Some digipeaters even collect messages received, and post them to http://APRS.fi, which is an online mapping tool used to view the GPS data reported by nodes on the APRS frequency, as will be done with the HAB.

4. *Custom PCB*

A custom PCB was designed in Eagled, manufactured by Advanced Circuits, and assembled in RIT's surface mount lab. Fig. 1 shows the main pieces of the custom board. It contains a pressure sensor, temperature sensor, digital potentiometer, video multiplexer, GPS, and various power management circuitry. There are also many connectors so that all pieces of the communication system can be connected. There is a header on the board for the Raspberry Pi Zero to mount on to easily interface with the various sensors and hardware on the custom board. The pressure sensor, temperature sensor, GPS, digital potentiometer, and APRS are on I2C buses. GPIO are used to control the analog video multiplexer, power control to the video overlay board (OSD), power control to the analog video cameras, and more. The pieces for the 70cm chain (cameras, video multiplexer, OSD, ATV transmitter, and digital potentiometer) are on a power management system from an 11.1V Lithium Ion battery. The pieces for the 2m chain and main communication hardware (Pi Zero, Temperature Sensor, Pressure Sensor, GPS, and radio) are on a power management system from a 7.4V Lithium Ion Battery. The APRS has its own battery so that it can be an independent system; the I2C interface is only connected to it to allow the Pi Zero to be able to take data from it in addition to allowing the APRS to transmit on its own. There is also a SPI connection to the DAQCS host board. The communication main board will send commands and data requests to the DAQCS side, and the DAQCS host board will send sensor data back to be displayed on the video overlay and sent to the ground in the telemetry stream.

Several sensors are on the custom board to be consistent with all of the others PCBs in the platform. The two sensors that are common to all of the other custom PCBs in the platform are the temperature and pressure sensors. These sensors are polled periodically for data collection purposes. The additional sensor that is of interest to the communications system is the Global Positioning System (GPS) sensor. The GPS is an advanced digital sensor made by u-blox. The main method for tracking the platforms location during flight is the APRS system, however this can be used as a redundant location method in case an anomaly occurs in the APRS system.

B. Communication System Software

As previously mentioned, the main processing in the communication system is on the Pi Zero. There is software to collect sensor data, store sensor data on the SD card, transmit telemetry, receive and execute ground commands, communicate with the DAQCS host board, control the camera source and data overlay for the 70cm chain, and more.

1. GPS Software

The GPS can be used to retrieve the date and time. This is useful because the Raspberry Pi boards do not contain a real-time clock. Therefore if one of the boards needs to reset or reboots, the GPS can provide a reliable time frame for use in logs and data transmission. The GPS communicates to the Raspberry Pi via the Inter-Integrated Circuit (I2C) protocol.

2. 2m Chain Software

In order to modulate/demodulate the AFSK (Audio Frequency Shift Keying) audio signals transmitted/received by the Raspberry Pi through the Baofeng radio, open source Soundmodem software is used. Soundmodem is capable of accessing the USB sound card attached to the Raspberry Pi, and sends/receives the digital data/commands required by the system as AFSK audio signals. Since the Raspberry Pi doesn’t have hardware to perform AFSK, the modulation and demodulation is done in software, by Soundmodem.

Soundmodem is a process that operates as a background thread on the Raspberry Pi, and provides a way for other executables to perform either send or receive transactions via AFSK. For example, the traffic heard over the radio can be monitored using the ‘axlisten’ command. Anytime an AFSK packet is present on the frequency at which the radio is tuned to, Soundmodem will demodulate the packet, and ‘axlisten’ will present the modulated data to the console. Alternatively, ‘beacon’ can be used to send any string of ASCII characters to the radio, as AFSK modulated audio signals. In this way, commands and telemetry can be communicated, in both directions, between the HAB and the ground station during the flight.

IV. Physicalities

A. Platform Design

The HAB platform subsystem serves as the sole means of mechanical support for all electronic systems. The physical design of the platform was severely geometrically constrained to minimize upon three parameters: centroid radius, weight, and inertial. The three parameter constraint was imposed due to reaction wheel feasibility. To satisfy the design requirement, the subsystem was built using low density polystyrene boards as the bulk material, reinforced with tempered aluminum. This allowed for a maximum volume to weight ratio, which was beneficial for component allocation, while keeping package weight at a minimum.

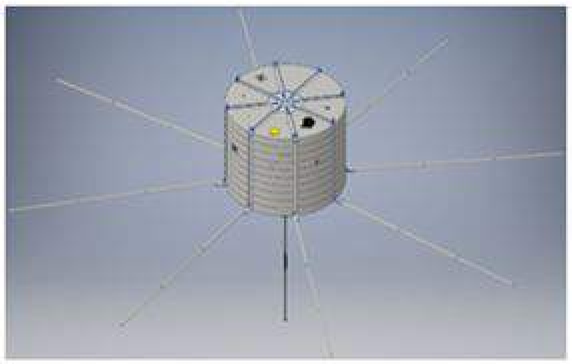
Figure 6: HAB Platform Design with Components

To improve upon stabilization feasibility the platform was designed to be a geometric cylinder (see Fig. 6) – mimicking the cylindrical geometry of the reaction wheel. Using this geometry the external surface area of the package was minimized, thus decreasing the effect that ambient air current disturbance would have upon the system. While minimizing the external surface area provided a clear benefit to system control it also decreased the internal surface area that could be utilized for component allocation.

In order to better adapt the internals of the system for subsystem integration the internals of the platform were "sectioned" off in a series of two unique compartments (see Fig. 7). The uppermost compartment contains the stability control components, data acquisition components, and retrieval system components. The bottommost compartment contains data transmission components and power system components. This arrangement effectively centralized the weight of the package at the bottom due to the power system's batteries which passively vertically stabilizes the packages. This allocation also isolates the reaction wheel subsystem from the visual overlay components, beneficially in the event of a failure.

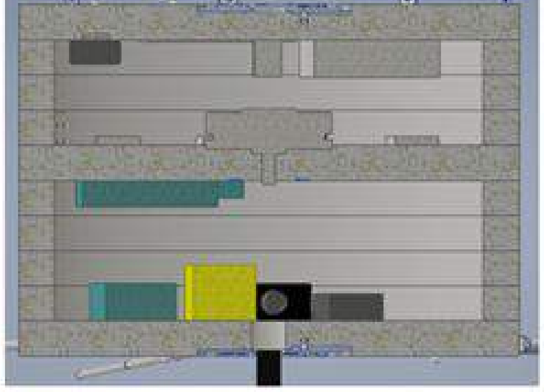
Figure 7: HAB Platform Internal Compartments

The compartment design further reduces intra-system wiring as component systems are secular. Extensive wiring is necessary for entire system integration, requiring wire routing at the radial ends of the platform to avoid physical interference with the reaction wheel – resulting in suboptimal inertial effects.

Among the batteries utilized in the power system are a selection of lithium based units. Usage of the units are hazardous during the flight profile of the platform due to decreasing pressure and temperature, meaning that in order to maintain system safety, these batteries are required to be kept suitably within operational ranges. The solution to this issue was found intrinsically in the lower compartment's data transmission system. One of the subsystem's components, the Amateur Television transmitter (ATV), requires an additional heat sink for ground level operation, magnifying the necessity during the mission profile. The unit produces waste heat at a high wattage continuously, which if left uncompensated would result in early mission component failure. A compensation action of transferring the excess energy to both the external surfaces and the power system batteries resulted in an expected mission survival. This assertion was made based on dynamic environment modeling where the continuously transient external convection transfer was approximated by assumed dominant environment pressure. This method was chosen due to the fact that the system internal factors (internal current, pressure, and temperature) could not be accurately defined as the system is partially open to the environment. Modeling of this system would require dynamic discretization of the compartments over an expected flight profile of two plus hours which for the project's scope was deemed unnecessary. Furthermore the effects of internal vapor could not be defined in any sensible manner, further driving the performance uncertainty up. With the above being true, simple first degree methods of performance shaping were surmised to be as effective as any higher degree consideration.

V. Data Acquisition and Control System

The goals of this subsystem are to collect sensor data, save it, control the motor and the angular movement of the entire system, as well as provide a way to safely recover the entire platform after flight.

A. Data Acquisition and Control System (DAQCS) Host Board

The DAQCS subsystem has a central host controller board that is primarily responsible for handling communication between the four DAQCS Sensor Nodes and the COMMS host board. Furthermore, the DAQCS Host Board is responsible for reaction wheel motor control, sensor data logging and manipulating the emergency cut down mechanism. The main processing units powering the DAQCS Host Board are two MSP430FR5994 mixed-signal microcontrollers from Texas Instruments. The MSP430FR5994 uses embedded FRAM technology to allow for ultra-low power operation and robust operation in near space environments. This microcontroller is capable of 16MHz operation and a variety of peripherals including UART, SPI, I2C. Other features include a real time clock module and a 12-bit analog to digital converter. Overall, these capabilities make the MSP430FR5994 microcontroller an ideal choice for the high altitude balloon application. The required functionality is divided between two independent MSP430FR4994 microcontrollers. This is necessary to maintain performance and due to hardware peripheral limitations. The communication between all of the DAQCS sensor nodes and the COMMS host board is constrained to a single MSP430FR5994. This MSP430FR5994 will be referred to as the Host MSP430FR5994. This includes the communicating between the four DAQCS Sensor Nodes over UART interfaces, COMMS Host board over a SPI interface, and the second MSP430FR5994 over a SPI interface. All of these interfaces are implemented on separate hardware buses to allow for optimal performance. Furthermore, the Host MSP430FR5994 is responsible for controlling the safety cut-down mechanism and reading the reaction wheel battery voltage levels using the AC. The second MSP430FR5994, is responsible for reaction wheel motor control, acquiring local sensor data and storing data local to an SD card. This MSP430FR5994 will be referred to as the Motor MSP430FR5994. The Motor MSP430FR5994 will be able to receive commands and send data back to the Host MSP430FR5994 over a SPI interface. The Host MSP430FR5994 will be a master device and the Motor MSP430FR5994 will be a slave device. Overall, the DACQS Host board capable of executing all necessary functionality and also provides additional GPIO for future development. A high level diagram of the DAQCS Host controller board can be seen in Fig. 8 below.

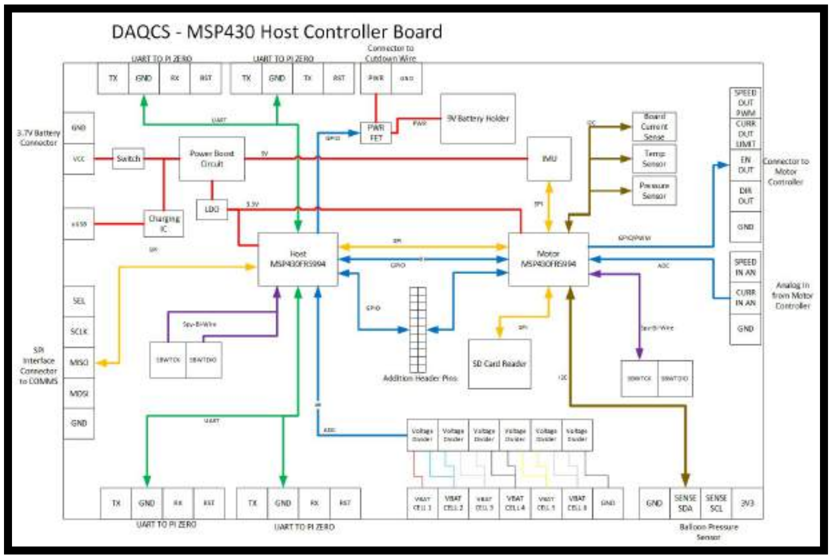

Figure 8. High Level Block Diagram of DAQCS Host Controller Board.

1. Power Management Design

The DAQCS host board is to be powered via its own independent single cell 3.7V Li-Po battery pack. Therefore, proper voltage regulation is necessary to support all the devices on host board. The two main voltage levels that need to be obtained are 3.3V for the MSP430FR5994 and all I2C sensors, and 5V for the IMU. In order to accomplish this, a step-up DC-DC converter IC is used to step up the nominal 3.7V to 5V. A Texas Instruments, TPS61240 is chosen for this conversion. It supports a wide rage of input voltage levels and operates at efficiency rates at over 90%. This 5V supply is used to power the IMU. From the 5V supply, a LDO is used to step down the voltage to 3.3V. A Texas Instruments LP5907 is used for the conversion. It has a wide range of operating voltage levels and can provide up to 250mA of output current. This 3.3V supply provides power for both of the MSP430FR5994 microcontrollers. The 3.3V supply also provides power to the temperature sensor, pressure sensor, current sensor and the SD card. A detailed schematic can be seen in Fig. 9 below.

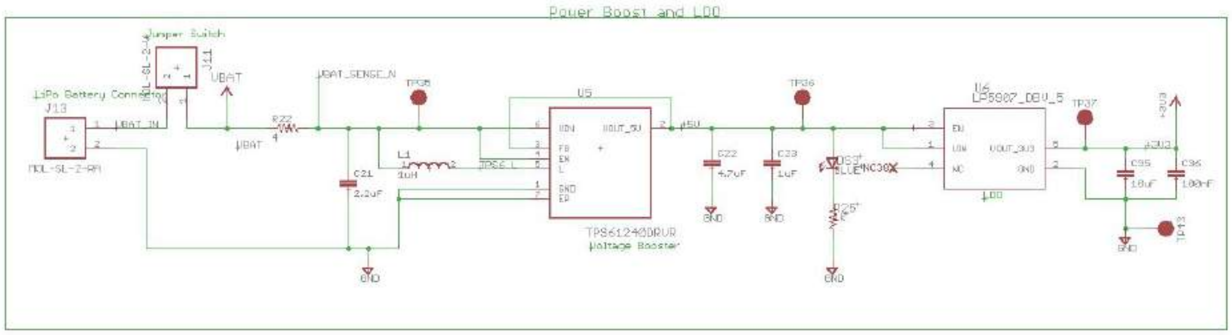

Figure 9. Power Management Schematic.

2. Host MSP430FR5994 System Design

The host MSP430FR5994 is primarily used as a communication terminal between all of the external subsystems. On the DAQCS side, communication between each of the Raspberry Pi Zero sensor nodes is accomplished using UART. The MSP430FR5994 is capable of four separate UART buses, therefore four sensor nodes will be used. This will allows data to be received from the sensor nodes and can interrupt the primary routine of the code to ensure the data is received. Data can also be sent to the sensor nodes to either update software parameters or to request data. Furthermore, a GPIO bus is connected to the sensor node which allows the Host MSP430FR5994 to hard reset the sensor node in case of failure.

The next major interface that the Host MSP430FR5994 handles is a SPI interface between the COMMS host Raspberry Pi Zero. In this configuration, the Host MSP430FR5994 is configured as a slave. This allows the COMMS Raspberry Pi to provide the master clock and send commands to request data from the Host MSP430FR5994. In this configuration, the Host MSP430FR5994 will receive and parse commands from the COMMS Host. If the command is directed at one of the Raspberry Pi sensor nodes or the Motor MSP430FR5994, the Host MSP430FR5994 will grab from the local register file in RAM and respond to the COMMS Host immediately. The Host MSP430FR5994 regularly polls each Raspberry Pi and the Motor MSP430FR5994 for all sensor values and stores the latest value in the local register file in RAM.

A SPI interface is also used to communicate between the Host and Motor MSP430FR5994. This allows commands to be sent from COMMS to be forwarded to the Motor MSP430FR5994. Such commands will include changing reaction wheel parameters and requesting current data. In this configuration, the Host MSP430FR5994 acts as the master SPI device and the Motor MSP430FR5994 acts as the slave device.

Apart from handling communications, the Host MSP430FR5994 is responsible for polling the status of the reaction wheel motor battery cells voltage levels. This feature is important since the Li-Po battery used in this initial application does not have built in under-voltage protection. Therefore, the voltage needs to be observed to ensure the voltage of each cell does not drop below a safe level. The 12-bit ADC will be able to acquire the analog voltage from each of the battery cells. The ADC will only operate up to a 2.5V reference. Each Li-Po cell has a maximum voltage of 4.2V. A voltage divider circuit is used to drop each cell down to a 2V max range. For protection, a 3V Zener diode is placed in parallel to each voltage divider to ensure a voltage over 3V is never seen at the output of each voltage divider. A detailed schematic of the voltage dividers is seen in Fig. 10 below.

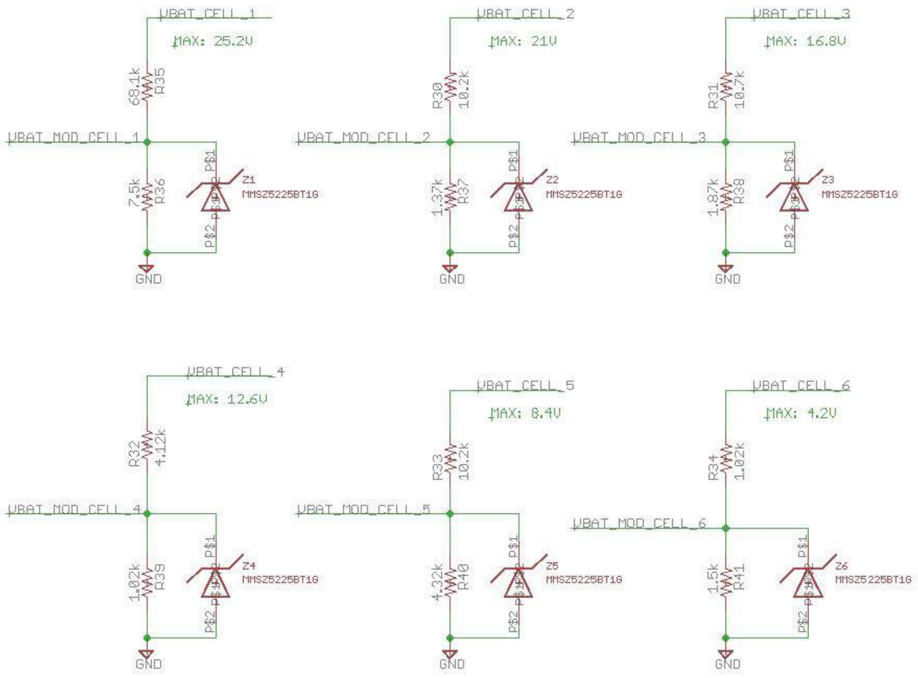

Figure 10. Voltage Divider Schematic.

3. Cutdown Mechanism

Finally, the Host MSP430FR5994 is responsible for controlling the cutdown mechanism. The cutdown mechanism is a small piece of nichrome wire at the end of a long copper wire cable. The end of the cable will be inside of the balloon envelope and resting along the side with the nichrome wire directly in contact with the balloon material. Shorting approximately one amp of current through this nichrome wire makes the nichrome heat up and pops the balloon. A separate 9V battery will be used as the power supply to heat up the nichrome wire. In order to safely create a short on the board, the circuit must be isolated from the main board circuitry. An optocoupler is used to isolate the control of a power MOSFET. A detailed schematic is seen in Fig. 11 below. When GPIO from the MSP430 is enabled high, this allows current to flow across the opposing side of the optocoupler. When this happens, the gate of the power MOSFET is pulled to GND (with respect the 9V battery) and therefore allows 1A of current to short across the source and drain of the MOSFET. This will consequently heat up the nichrome wire and pop the balloon. The Host controller will receive a command from the ground to initiate the cutdown sequence. In addition to this, when reaching a specific altitude calculated from the pressure sensors, the Host controller will initiate the cutdown sequence in the case that the ground loses communication with the platform.

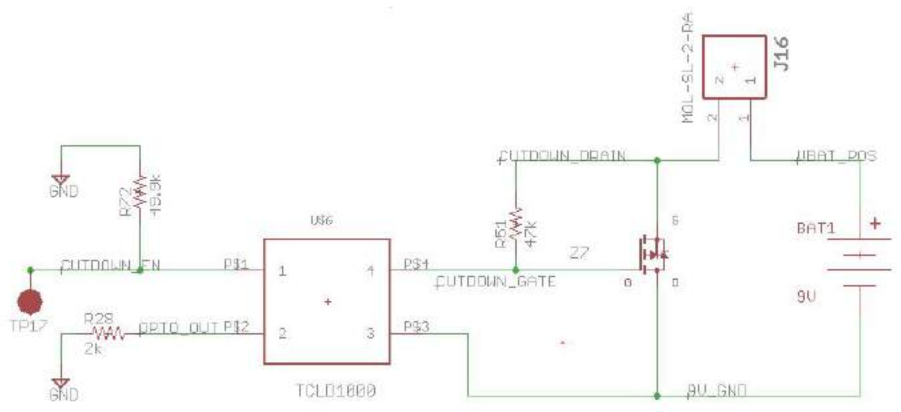

Figure 11. Cutdown Circuit.

4. Motor MSP430FR5994 System Design

The motor MSP430FR5994 has five different sensors integrated on the board. Four of these sensors communicate over the same I2C bus. This include the temperature, pressure, current and backup IMU sensor. The current sensor allows to determine the current draw of the board throughout its operation. The backup IMU is a small IMU that is primarily used in cell phone based applications. The Host MSP430FR5994 also interfaces with an external I2C pressure sensor that will be located in the nozzle of the balloon. This sensor is on a separate I2C bus to minimize any failures or shorts that may occur.

The main IMU that will be used in the reaction wheel application is mounted on the board and interfaces with the MSP430FR5994 over a SPI interface. This IMU will be polled as fast as possible and used as an input to the reaction wheel controller algorithm. Detailed schematics of all sensors can be seen in Fig. 12 below.

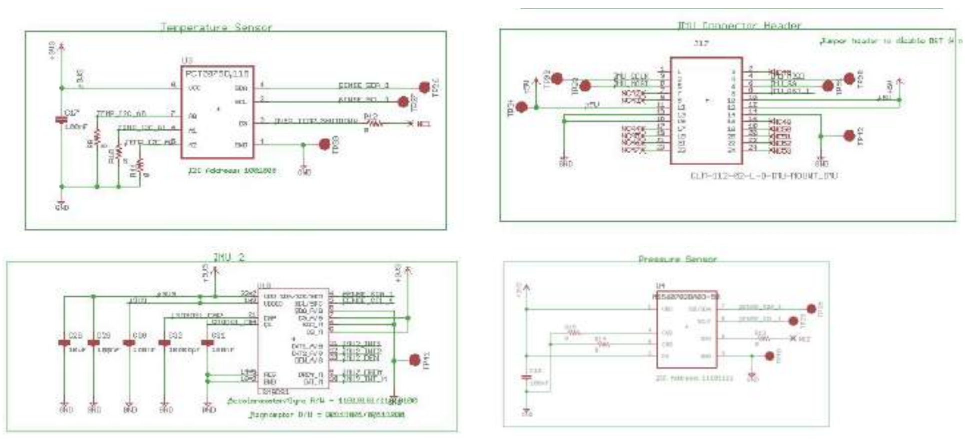

Figure 12. Sensors.

5. GPIO/PWM Motor Controller

In order to control the motor controller, digital PWM signals and TTL signals are used. These signals are generated using the MSP430FR5994. The PWM is generated using built in timers on the MSP430FR5994 that are set to toggle a GPIO pin when the timer reaches a certain value. The timer count value can easily be modified on the fly to change the PWM duty cycle. Two separate PWM signals are generated. One is used to control the speed of the motor and the other is used to limit the max current consumed by the controller. Two GPIO pins are used to control if the controller is enabled and the direction to spin the motor.

6. ADC

The actual speed of the motor must be acquired as feedback for the reaction wheel controller. The Maxon motor controller outputs the current speed of the motor as an analog output that is scaled from 0-2.5V. Using the 12-bit ADC, the analog voltage can be acquired and converted to RPM.

All of the acquired data from sensors must be logged to an SD card such that the data can be analyzed after flight. A microSD card connector is mounted to the Host board and is interfaced with the Motor MSP430FR5994 over a SPI interface. This allows the MSP430FR5994 to create files and log data to the microSD card. The Motor MSP430FR5994 is a master device and the microSD card is the slave device. A detailed schematic of the microSD card interface can be seen below in Fig. 13.

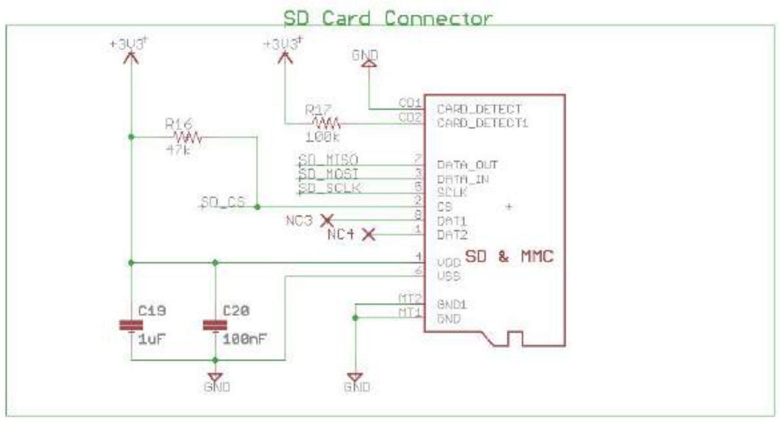

Figure 13. MicroSD Card Interface.

7. SPI-By-Wire

A SPI-By-Wire interface is pinned out to a Molex SL connector. This allows each of the MSP430FR5994 microcontrollers to be flashed via a debugging interface such as a MSP430FR5994 development board.

B. GRSS

Another custom board, the global recovery signalling system (GRSS), was created. The board is composed of a 9V battery, 9 super bright LEDs, a loud buzzer, a voltage regulator, 555 timer and power MOSFET. The 555 timer sinks the LEDs and buzzer through the power MOSFET at a rate of 2 times per second. The main purpose, is to locate the platform after it lands. The GPS location will only show us roughly where the platform lands and the GRSS will help the team locate it, especially if caught high up in a tree. The GRSS is also useful for other people to locate our platform and contact us when they find it.

1. Ground Recovery/Cutdown

The plan for ground recovery is as follows. First, the GPS location will be monitored on a map as stated in the communications section and the horizontal speed of the platform will be monitored. If the horizontal speed of the launch exceeds a limit, meaning it would travel too far away to feasibly acquire it, the communications team will trigger a cutdown. Second, the vertical speed of the platform will be monitored. If the vertical speed of the platform is too slow, the platform will take too long to reach altitude and then travel too far horizontally. Therefore, a cutdown would be triggered. Third, the path of the platform will be monitored. If the platform is headed towards a large body of water or land that is not accessible, a cutdown will be triggered. Fourth, as mentioned above, at a certain altitude the balloon is designed to burst (roughly 100k ft.) and the platform will start its descent. In all descending circumstances, there is a parachute attached to the platform that will bring it down safely. For recovery, if a GPS signal is still available, that will be utilized for guiding the recovery team and the GRSS will help locate the platform at that GPS location. However, if GPS signal was lost at some point and never re-established, the platform recovery will solely be relied on a pedestrian noticing the platform via the GRSS and contacting the team.

C. Raspberry Pi Zero Sensor Nodes

One of the main goals of the DAQCS High Altitude Balloon project is remote environmental sensing. This includes sensing of payload temperature, external ambient temperature, relative humidity, pressure, altitude, payload acceleration, photo/video capture, and more. Continuous flight data allows for monitoring of flight status during the launch but also allows post processing of the recorded data to learn more about the environment at high altitudes. In order to satisfy this requirement, a reliable, accurate, lightweight, and low-power system needed to be developed. Some level of sensor redundancy also needed to be integrated in case a few of the sensors malfunctioned during flight.

1. Hardware Design Architecture

With the large amount of sensors needed, it was evident that some form of custom PCB solution was necessary. The high level requirements were: record/store environmental sensor data and record/store photos and videos. The initial design revolved around what type of camera would be used to take pictures and videos. Most sensors use standard serial busses such as I2C, SPI, or 1-Wire which have dedicated hardware integrated in most microcontrollers. So interfacing with these sensors is fairly trivial. The original plan was to use NTSC cameras and store the analog video feed using an ADC. This was quickly scrapped since the NTSC camera feed is very low quality given the abundance of affordable HD cameras on the market. The final design settled on using a Raspberry Pi Zero (RasPi Zero) as a base computing platform for the sensor nodes. The RasPi Zero was chosen primarily for the small form factor and the integrated MIPI camera interface which allowed easy connection of a 15pin flex cable to an HD camera module. The RasPi Zero uses a Broadcom BCM2835 SoC which at its core uses an ARM1176 processor and dedicated GPU for camera capture acceleration. Added benefits of using the RasPi Zero included being able to run a full Linux OS, which simplified software development and allowed easy data storage to the onboard microSD card. With the RasPi Zero GPIO header pinning out all the necessary serial interfaces for the environmental sensors, the only drawback to the RasPi Zero module was its increased power consumption over smaller microcontrollers such as the MSP430.

The remaining design choices were to pick environmental sensors, decide how to power the RasPi Zero, and integrate system redundancy and failsafes. All of these features were designed into a RasPi HAT (where HAT refers to “Hardware At the Top”) which is a custom PCB that sits on top of the base RasPi Zero module. This board houses a 5V Boost Converter to boost up a single cell LiPo/LiIon battery to power the Raspberry Pi Zero along with various sensors. A boost converter was chosen in order to keep the battery used at a single cell configuration to limit weight. A high efficiency TPS61232 booster was chosen to reduce conversion loss when boosting the 3.2V-4.2V battery voltage to the required 5V. To simplify charging, an in-system LiPo/LiIon charger was integrated so that the sensor node RasPi HATs would not have to be removed from the HAB platform to recharge. When used with the RasPi Zero base board, the RasPi HAT supplies 5V to power the Raspberry Pi Zero and receives 3.3V from the

RasPi to power all of the I2C bus sensors. An optional 3.3V LDO was integrated onto the HAT so that the HAT could be used in the future with any other microcontroller of choice.

All environmental sensors chosen used the I2C bus or 1-Wire protocol to communicate with the RasPi Zero. This allowed for easy integration since almost all sensors are made to support the standard I2C bus protocol. The only sensors in the HAB that use the 1-Wire protocol are the external temperature sensors, which were chosen based on the sensor package. The DS18B20+ 1-Wire sensors have three long leads and allow for easy wire harnessing and placement outside the HAB platform. To summarize the sensors used, each RasPi HAT houses the following sensors:

- 2x on-board PCT2075D I2C Temperature sensors
- 1x MS5607 I2C Pressure sensor
- 1x MS5803 I2C Pressure sensor (zero-pressure rated)
- 1x SI7021 I2C Humidity sensor
- 1x INA219 I2C Power monitor
- 3x external DS18B20+ 1-Wire Temperature sensors
- 1x LSM9DS1 I2C 9-axis IMU (optional)
- 1x MAX11613 I2C 12-bit 4-channel ADC (optional)

With a planned use of four RasPi HATs in the HAB platform, the requirement of sensor redundancy was more than completed. It is important to note that two different pressure sensors are used on each HAT. The MS5803 is rated for standard sensor accuracy at zero-pressure whereas the accuracy of the MS5607 only goes down to 300mBar. The MS5607 claims to reach down to zero-pressure using the "extended ADC range," but it most likely not accurate. Both modules were used so that a flight comparison could show if the MS5803 sensors, which cost 3 times as much as the MS5607 sensors, are indeed more accurate. If this is the case, the MS5803 sensor will be used for future flights. A final note should be mentioned about the "optional" IMU and ADC included on the RaspPi HATs. These components were added to "future-proof" the RasPi HAT. The one peripheral that the RasPi Zero base platform does not have is an integrated ADC. Adding a small and cheap 4-channel ADC allows the potential to do basic analog sampling with the HAT. The IMU was also included so that the RasPi HAT could function as a stand alone HAB computer without relying on inertial and compos measurements from other boards within the HAB.

At upwards of 100,000ft, something is bound to go wrong. Maybe the Linux OS on the RasPi HAT crashes or an I2C sensor locks up. A system reboot would be necessary to recover from this type of soft failure. Included on the RasPi HAT is a reset functionality that allows one of three reset sources to either disable the boost converter and hard reset the board or simply soft reset the BCM2835 SoC. The three reset sources include a GPIO (externally driven by the DAQCS HOST controller), an on-board MAX6371 hardware watchdog timer, or a pushbutton (only used for development). Fig. 14 belows shows a hardware block diagram which overviews all the components covered in this section.

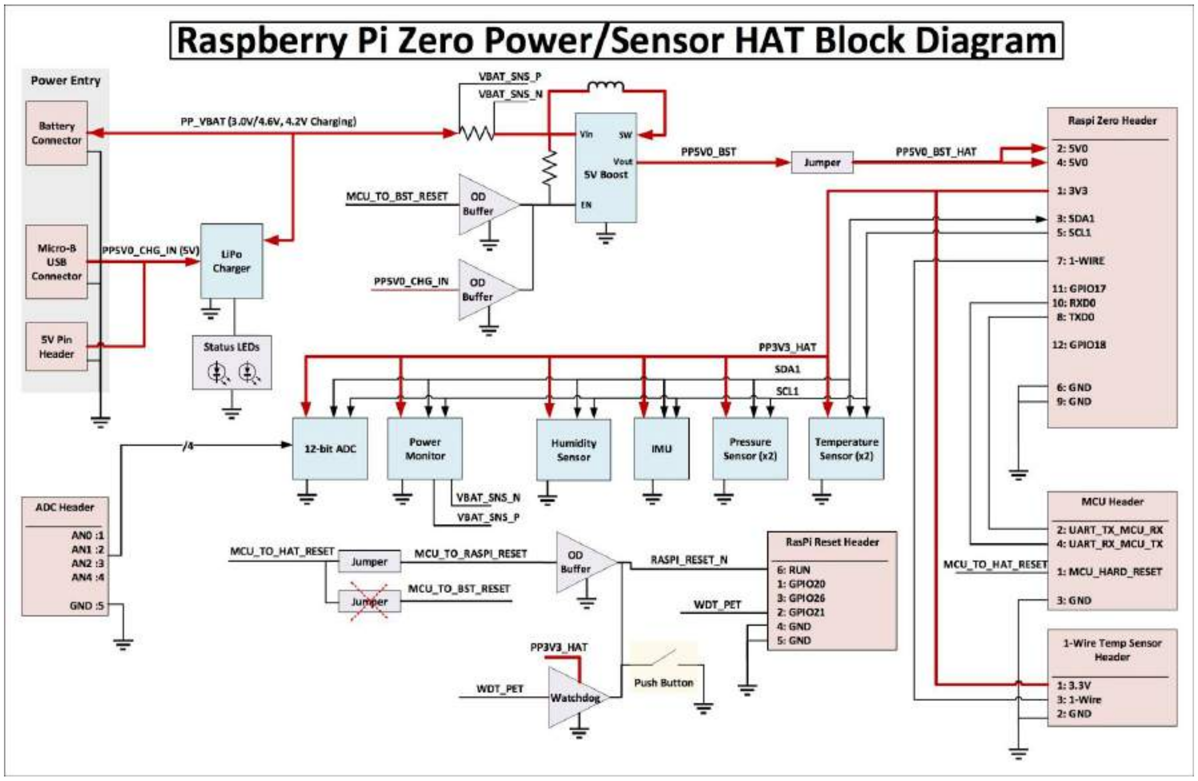

Figure 14. Raspberry Pi HAT Block Diagram.

2. *Software Overview*

Running a Linux OS on the RasPi Zero allowed for more smooth software development. The main tasks of the software running on the Raspbian OS include sensor logging, video/camera capture, interfacing with the DAQCS HOST board over UART, and toggling the on-board watchdog timer to avoid reset. All software running on the RasPi was written using Python. Three main scripts are used and are executed when the RasPi boots. The first script is the environmental sensor logger. This script reads data from all I2C and 1-Wire sensors and logs the recorded data to separate sensor CSV files along with one combined CSV log. The CSV logger script also incorporates toggling the hardware watchdog timer every one second. This is done inside the CSV logger in case the python application crashes during flight, which would reset the CPU and continue to execute the CSV logger. The second Python script used records 60 seconds of video and then captures four photos continuously. This script toggles the built in BCM2835 watchdog timer for the same reason. Finally, the third Python script is the UART serial monitor. This script handles UART transactions with the DAQCS HOST board. Transactions include requests for sensor data and setting the OS's epoch timer for accurate timestamping.

3. *Custom PCB Solution*

Using Eagle for PCB design entry, the RasPi HAT was designed and fabricated. Initial testing showed that all I2C and 1-Wire sensors functioned correctly along with the boost converter, battery charger, and reset circuitry. Fig. 15 belows shows a completely assembled RasPi HAT Sensor Node.

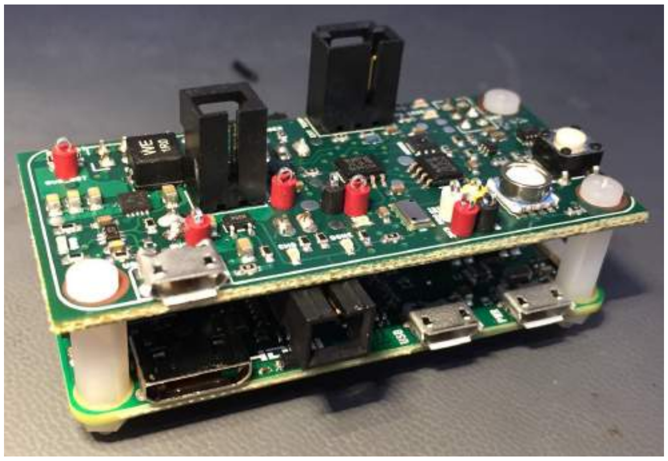
Figure 15. Assembled Raspberry Pi HAT.

D. Reaction Wheel

One important component of the instrumentation platform is its reaction wheel. One requirement for this project was to stabilize the platform in the z-axis, above an atmospheric height of 60,000 feet. At this elevation, the wind speeds are greatly reduced in comparison to lower altitudes. This requirement was created with the thought that better pictures and videos may be taken during flight. Position control may also be achieved in later revisions of the project.

1. Motor Determination

In order to compensate for the instrumentation platform's movement due to winds, a motor and flywheel design was initially proposed. However, after the final platform weight was determined, it was found that the rotor mass and shaft of the chosen motor were able to provide enough angular momentum to counteract its movement. Theoretically, it was determined that the motor itself must spin approximately 170 times faster than the instrumentation platform to provide a net zero angular velocity.

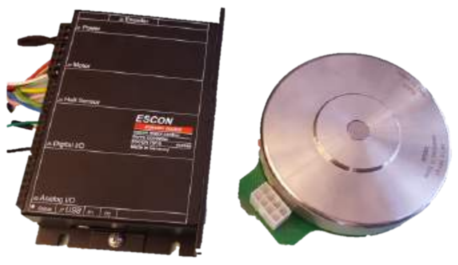
Figure 16. Maxon Motor & Motor Controller.

Figure 16, seen above, shows the motor and motor controller being used by the team to control the angular velocity and position of the instrumentation platform. The motor is a Maxon EC 90 flat brushless motor with hall sensors. The controller that was chosen is an ESCON 70/10 4-Q servo-controller.

2. *Motor Testing*

Initial testing of the motor was done using the setup outlined below in Fig. 17. The motor was powered by a 6 amp DC power supply, with an Arduino controlling the PWM signals being sent to the ESCON servo-controller.

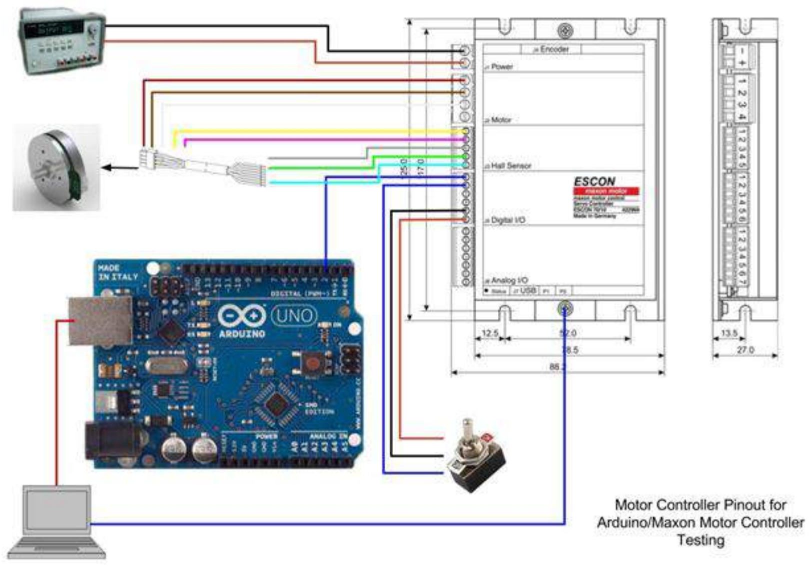

Figure 17. Motor Test Setup.

After learning more about the software and motor, testing was done to determine the characteristics of the motor in a no-load situation. As there will be no flywheel, the motor will actually be in a no-load situation, enabling the team to analyze the motor's abilities by itself.

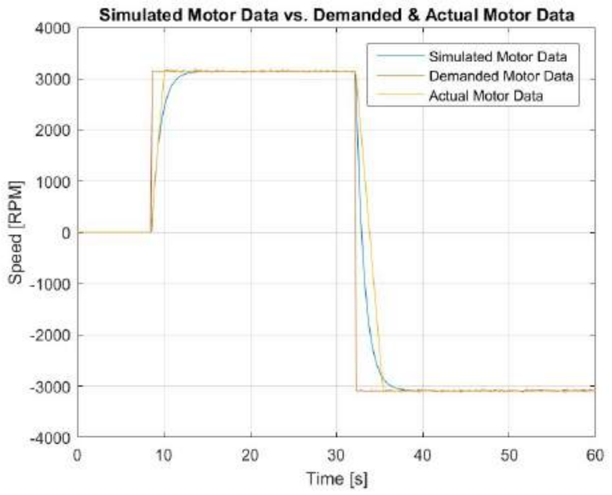

Figure 18. No-Load Motor Simulation & Test Results.

Figure 18 above shows the results gained from one test, in which the motor was spun using a PWM. It can be seen that the motor has a low time constant. The transfer function being used for this system may be seen below:

$$TF(s) = \frac{135}{(0.0211s+1)(s)} \tag{1}$$

This transfer function, shown within the s-domain, showed a very good fit, which can be seen in the plot above. The 21.1 ms time constant is the mechanical time constant of the system. The other time constant, which belongs to the electrical components of the system, is much smaller.

3. Controller Determination

A similar test to the one outlined above for the motor is planning on being done with the instrumentation platform. In the case of the entire platform however, an IMU will be acquiring data about the angular speed, while motor speed and the PWM input are also monitored. With these inputs and outputs, a model should be able to be derived and validated through multiple testing scenarios. Multiple models may be determined at different sampling rates. This will allow for a best-case scenario to be determined. A high sampling rate is desired for the motor controller; however, due to the amount of time it takes to process and save the data on the MSP430, a lower rate may be used. The "best-practice" rule of having the sampling rate be less than the 0.1 times the maximum time constant of the system is hoped to be used. This maximum time constant is determined by finding the place on a step response plot where the system is at 63% of its final steady-state value.

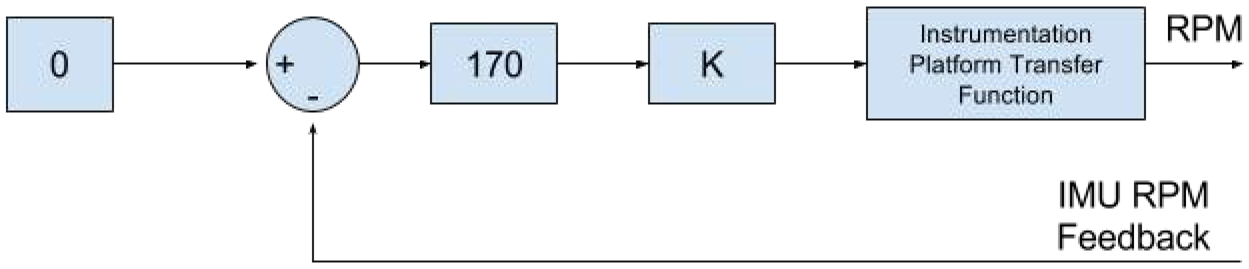

Figure 19. Instrumentation Platform Control System.

Figure 19 above outlines the theoretical model of the controller for this system. It can be seen that the desired angular velocity of the system is equal to 0, meaning that the team does not want the platform to spin. Feedback on the angular velocity is then taken from the IMU. This signal is then multiplied by 170, which gives a motor command for the system. K is the compensator gain of the system. It is hoped that a proportional controller may be used, as this would be much simpler to implement into the MSP430 code for the system than a PID controller. It will also take less processing time. This compensated motor signal is then fed into the motor controller, causing the platform to spin.

This system is currently being tested and will be determined by the time of the flight.

VI. Conclusions & Results

Conclusions and results were drawn after the launch that took place on Saturday, April 29, 2017.

A. Communication System

Unfortunately, multiple communication subsystems failed during the launch. The GPS data from the redundant module in the telemetry stream cut out above 40,000 feet. Once the platform can back below 40,000 feet, the GPS functioned properly again. The GPS data from the APRS cut out above 60,000 feet. It was never updated afterward. The APRS kept transmitting the same coordinates. Upon recovery, it was found that the APRS module fell inside of the platform due to poor mounting, which explains why the GPS data failed to update afterward. The redundant GPS module cut out at high altitudes due to a software issue. The module should have been configured for an airborne mode to allow it to operate up to about 164,000 feet.

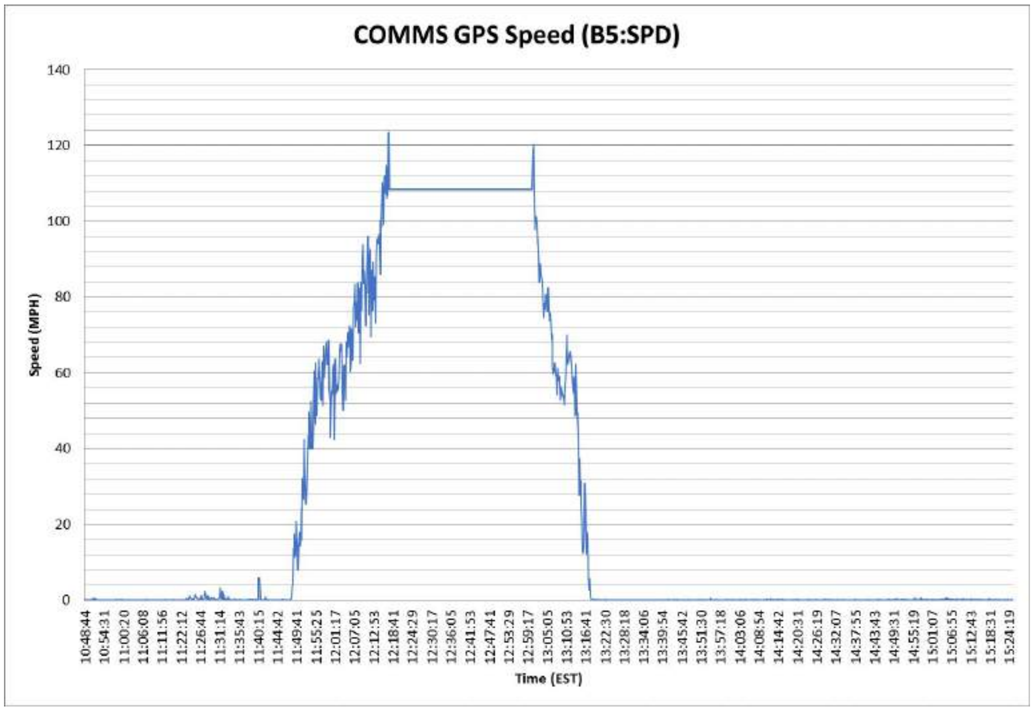

Figure 20. Comms Horizontal GPS Sensor Launch Data

Figure 20 shows the speed obtained from the GPS sensor throughout the flight. The data cut off in the middle, while GPS lock was lost (at high altitudes). A max speed of about 124 mph was recorded. Figure 21 shows the altitude obtained from the GPS sensor throughout the flight. As previously stated, the GPS data was lot above 40,000 feet (12,000m = 39,370ft).

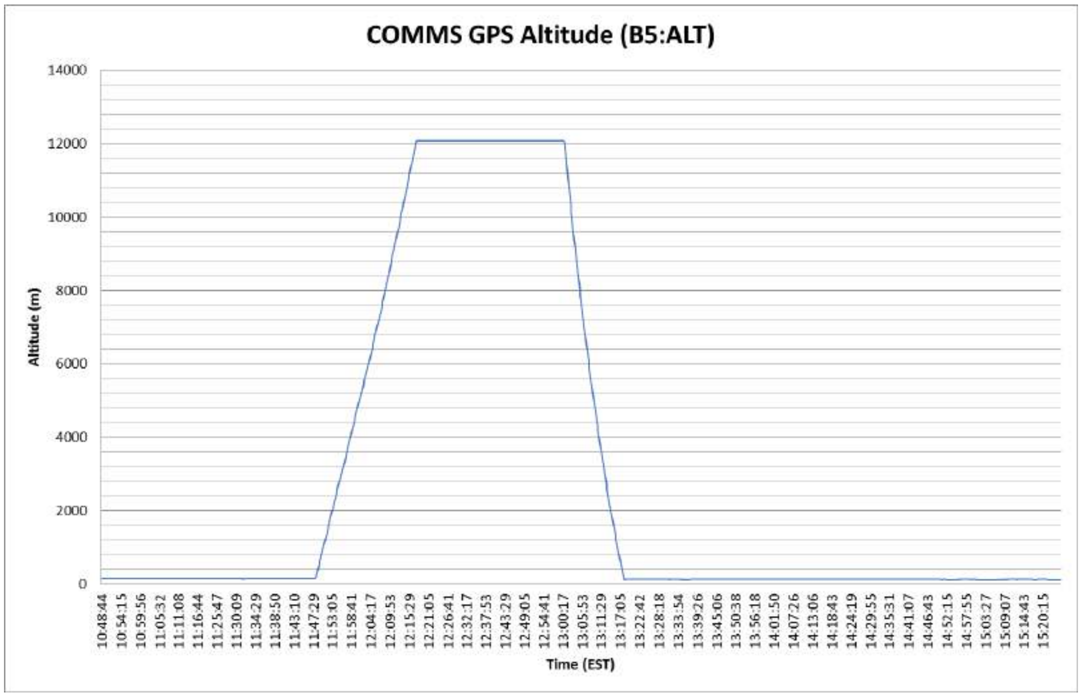

Figure 21. Comms Altitude GPS Sensor Launch Data

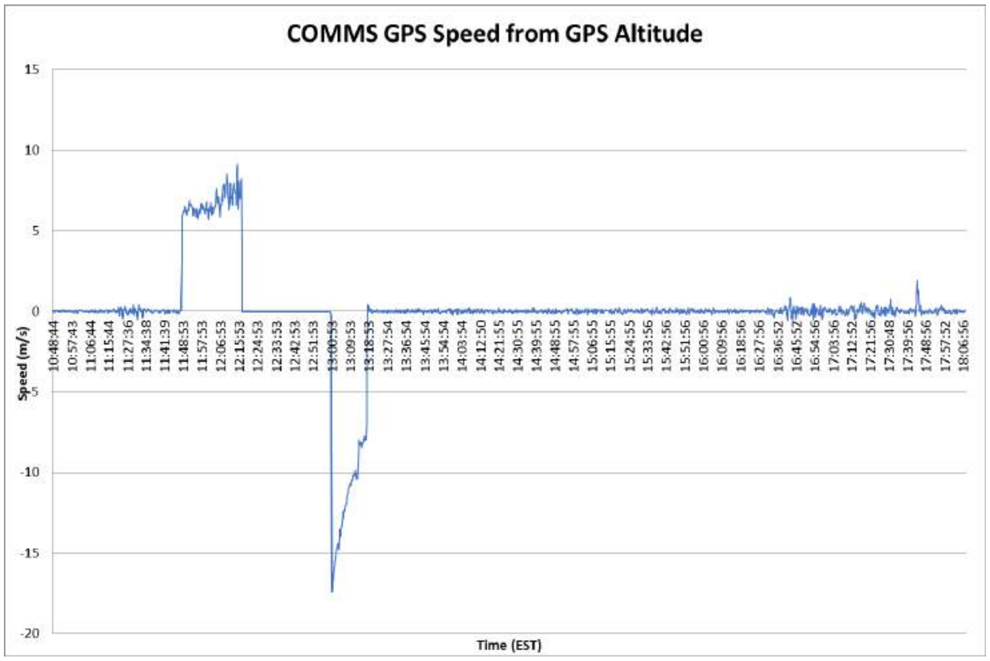

Figure 22. Comms Vertical Velocity GPS Launch Data

Using the GPS altitude over time, the vertical velocity of the platform was calculated. Figure 22 shows the results of the vertical velocity calculations.

In addition to the GPS issues, there were issues with the analog video stream. Prior to launch, audio from the platform was received by the ground station. However, video was not received. The platform was launched due to time constraints, with the hope that the video would start working. The ground station never received the analog video and quickly lost the audio transmission. A HAM in Irondequoit was actually able to receive the analog video stream and recorded some of it. So, there was definitely an issue with the ground station setup for receiving video. The recorded video was observed. The data overlay was functioning properly. However, the video multiplexer did not seem to be working. The overlay would change camera number, but the video was always from the bottom camera (analog camera #1). There were some issues close to launch day with the video multiplexer that were not investigated due to time constraints. For some reason, the board's video multiplexer had issues, despite working for weeks throughout testing.

Better results were obtained on the 2m transceiver side. Telemetry transmissions worked and were received about every 40 seconds (the goal was about once per minute). Unfortunately, not all sensors were working properly on the data acquisition side, but all sensor that was received by the communication host board was transmitted to the ground. Since the APRS cut out early, the GPS data in the telemetry transmissions was used to track the platform. The information was lost at about 9,000 feet when the ground station lost line-of-sight communications with the platform. Based on data, the platform was headed toward Syracuse, NY. HAM operators in the Syracuse area were contacted in an attempt to find the platform. Luckily, someone was able to receive and decode the telemetry stream to get GPS data. Coordinates to the platform were obtained, allowing for the platform to ultimately be recovered. Unfortunately, commands did not work throughout the launch. Some worked during launch preparation, but none were received and executed upon by the platform during the flight. After recovery, a weird software error for soundmodem was found that caused the software to stop listening, but not crash. There was software in place to restart the application if it crashed, but it never did.

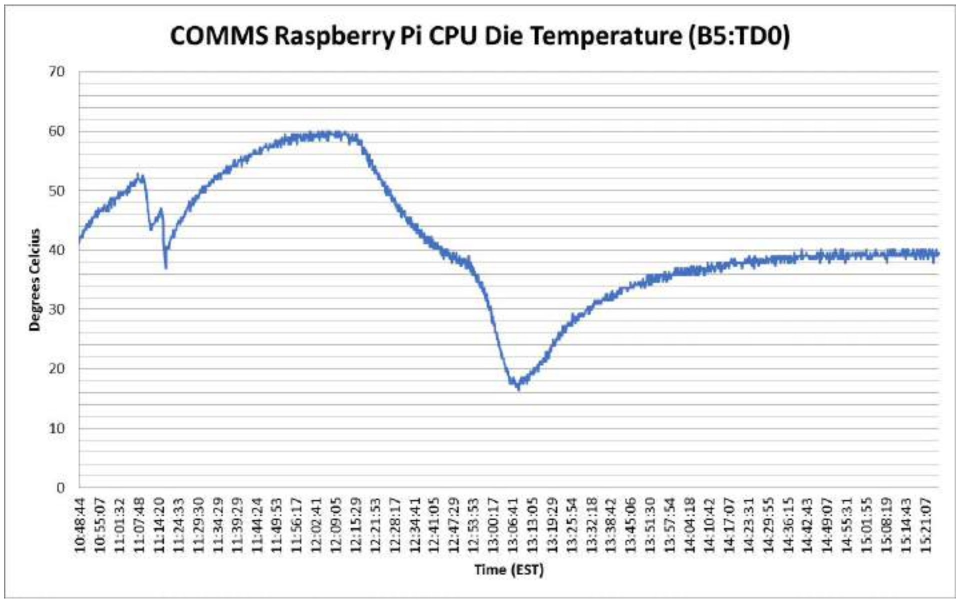

Figure 23. Comms Pi Die Temperature Sensor Launch Data

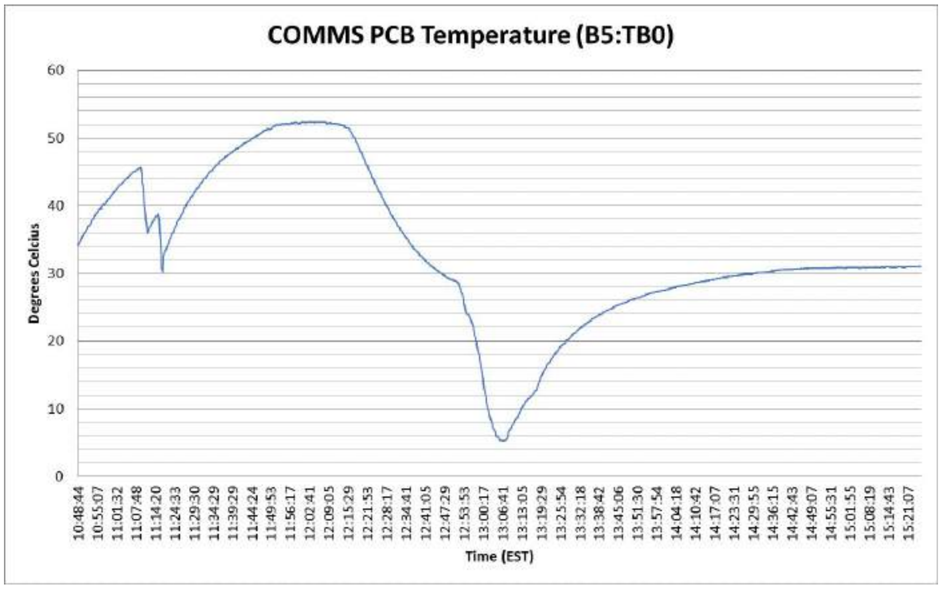

Figure 24. Comms Board Temperature Sensor Launch Data

The temperature and pressure sensors in the communication system worked properly throughout the flight. Figure 23 shows the data from the Raspberry Pi Zero die temperature sensor. Figure 24 shows the data from the temperature sensor on the communication host board. The temperature changes for both boards matched up. Both boards got much hotter than expected (above 50 degrees Celsius). They were in the lower portion of the platform where the ATV Transmitter was placed, which probably caused the high temperatures. Figure 25 shows the data from the pressure sensor on the communication host board. The pressure sensor value went down to 11.8 mBar, when the platform was at its maximum altitude.

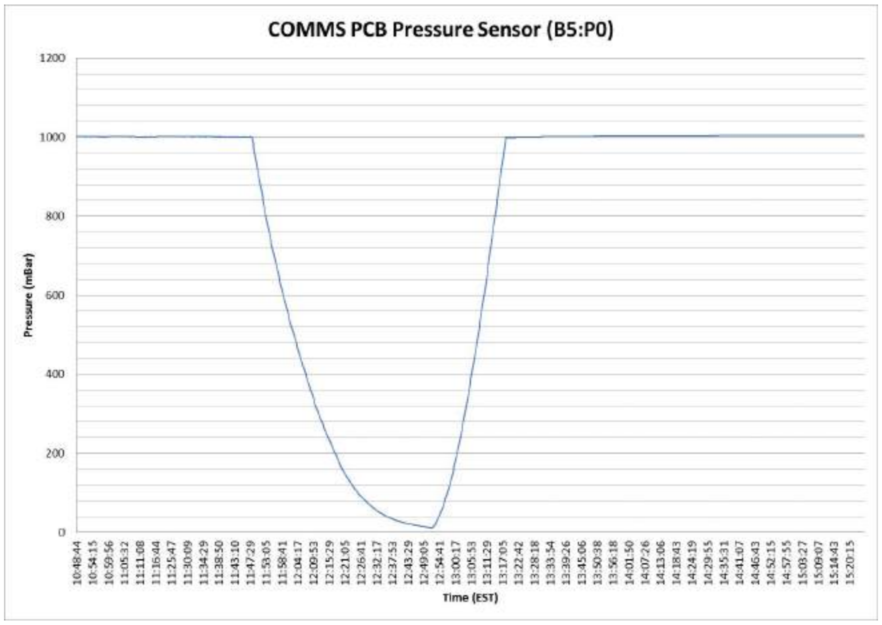

Figure 25. Comms Pressure Sensor Launch Data

Overall, platform communication was a partial success. The telemetry stream worked fine and commands worked until a new software error appeared. The APRS module was working properly, but poorly mounted. The GPS module worked as it was configured. The temperature and pressure sensor on the communication host board worked. Video with data overlay was transmitted from the platform, even though it wasn't received by the ground station.

Minimum future work could be done to greatly improve the communication system. The APRS mounting should be improved. The GPS software can be updated to change its operating mode to an airborne mode to allow it to work at higher altitudes. The soundmodem listening software could be restarted periodically or be restarted if a log parser sees an error message. More work would have to go into determining why there was an issue receiving the analog video stream and why the video multiplexer was not working.

B. Platform

1. *Impact and Damage*

When the HABIP was recovered from the flight, the platform was recovered on a sloped hill. The recovery team had explained that it appeared as though the platform had hit the ground and rolled along the slope of the hill. This was later confirmed when examining the recovered platform. Approximately half of the antenna grounds were either bent or missing. According to the recovered speed data, the HABIP had also been traveling at a significantly faster rate than was planned for - with respect to the parachute. Extrapolating the impact speed from Figures 21 and 22 the platform impacted the ground in excess of 30 mph, with a vertical impact of approximately 20 mph. According to parachute specs the platform was projected to impact with a vertical speed of 10 mph, however it did not account for any horizontal velocity. It is theorized that this deviation was caused by the balloon neck plug falling on top of the parachute during re-entry and causing a portion of the parachute to fail, resulting in a decrease in effective area, and thus a faster descent rate. In future flights the aluminum balloon plug will be replaced with a plastic printed balloon plug.The decrease is weight will hopefully prevent this adverse event.

Aside from the damaged antenna grounds, the only other external part with damage was the antenna itself. The antenna was snapped on impact, even with the flexible connection. This again is proof of the higher than expected impact speed. Upon impact it appears as though the antenna tip hit the ground, rebounded somewhat, and the opposite end of the antenna “butted” up against the metal coupling plate of the cage preventing any further movement causing the antenna to snap. There is no easy solution to this issue and it has been reiterated by the customer that future launches will incorporate a non-flexible coupling, regardless of the effects it will have on component health.

A handful of internal components were dislodged upon impact, causing damage to one of the Pi HATs. The motor controller was dislodged and collided with a Pi HAT, destroying the stack. An interesting fact of this occurring too is that it appeared as though when the motor controller fell, it disconnected from the DAQCS PCB, which may have prevented further component damage (as the DAQCS PCB was already damaged from an unrelated event). No other components in the DAQCS compartment were dislocated. Within the COMMS compartment three components had been dislodged although did not damage any adjacent components. With the COMMS compartment directly located at the impact face, the batteries were expected to experience most of the direct impact. This was a design failure, however in retrospect may have proved beneficial in this scenario. The batteries dislodging mean that they did not absorb all of the impact, as they would have if they were rigidly connected. This could have potentially damaged the batteries, or caused a series of damages. Future iterations will employ a modification of the applied design. The batteries will not be rigidly connected to the platform, however need a “safety net” to prevent them from potentially colliding with any internal components.

2. Temperature Response

The data recorded from the COMMS and DAQCS components indicate that the platform successfully maintain the internal temperature at a temperature above the minimum operating value, circa 0 C across all components, generally speaking. The Pi HATs were kept at a temperature between 70F and 100F, see the data plots in the following section.Both the reaction wheel and motor controller were not operational during flight and this is expected to have aided in not “overheating” the compartment. The DAQCS external sensors acquired an ambient temperature history as expected, ranging from ground level temperature to -50 Fahrenheit. During periods of flight the temperature gradient across the platform wall exceeded 100 Fahrenheit.

The COMMS compartment did not have as many temperature sensors as the DAQCS compartment, however the COMMS PCB did have the only two sensors. The COMMS Raspberry Pi achieved a temperature of 140 Fahrenheit (Figure 23) and the PCB achieving the minimum temperature of 40 Fahrenheit (Figure 24). The minimum temperature was achieved during descent, which was expected as the ATV heat sink would conduct/convect to a greater degree on descent due to the increase in airflow. The highest temperatures were experienced as the platform neared the highest point of its flight. This is also expected as the air at this point carries nearly no density preventing convection, and thus components heating up. The temperature at this point peaked, and the component seemed to have achieved the near steady-state heat exchange with the compartment air. Unfortunately this is the extend of temperature data for the COMMS compartment. The ATV, which was the highest risk item, did not fail, and did not cause any other component to fail.

C. Data Acquisition and Reaction Wheel Control System

1. Pi HATs

For the first HAB launch, all four PiHATs were used in the platform. All I2C sensors were logged except for the IMU and ADC, which were deemed optional sensors during the design phase. Each board logged data from two 1-Wire sensors. The original plan was to have the 1-Wire sensors external to the platform for all PiHATs, but due to time constraints only two of the PiHATs had their 1-Wire sensors outside the platform. Each PiHAT camera captured four photos and then sixty seconds of video continuously. The PiHAT board names were labeled B0 through B3 for identification. The below table documents the PiHAT launch configurations.

Table 1. PiHAT Sensor Node Configurations.

PiHAT Board	1-Wire Configuration	I2C Configuration	Camera Configuration
B0	2x External, sampled every 1.5 seconds	All sensors except IMU and ADC, sampled every 0.5 seconds	4x Photo, 60s Video, Pointed down to Earth
B1	2x Internal, sampled every 1.5 seconds		4x Photo, 60s Video, Pointed at the Horizon
B2	2x Internal, sampled every 1.5 seconds		4x Photo, 60s Video, Pointed at the Horizon
B3	2x External, sampled every 1.5 seconds		4x Photo, 60s Video, Pointed up at the Balloon

During pre-launch checks, it was determined that PiHATs B2 and B3 were not powering on. There was no time to debug the issues, so the mission proceeded with PiHATs B0 and B1 properly working. Post launch, it was determined that the two UART cables connected to B2 and B3 were not wired correctly. The UART Host Tx was tied to the RST input on the PiHAT and therefore disabled the PiHAT's power supply, keeping B2 and B3 powered off. No data was recovered from either of the two boards. Also, a plastic lens cover was placed on the bottom camera of B0 to protect during HAB pre-launch checks. This cover was not removed before launch and thus the camera pictures and videos were tinted. B0 I2C and 1-Wire data was successfully logged. PiHAT B1 was the only completely functioning board. Post-processing the sensor data showed that the maximum mission altitude achieved was between 82416ft and 78396ft. External HAB temperatures were measured down to -58F. The mission ascent time was approximately 63 minutes. The calculated ascent rate using the maximum altitude was ~1275ft/min. The mission descent time was approximately 25 minutes, with a calculated descent rate of ~3175ft/min. The two below plots document the recorded data through the mission. Following the plots in an image of Earth and space captured by PiHAT B1.

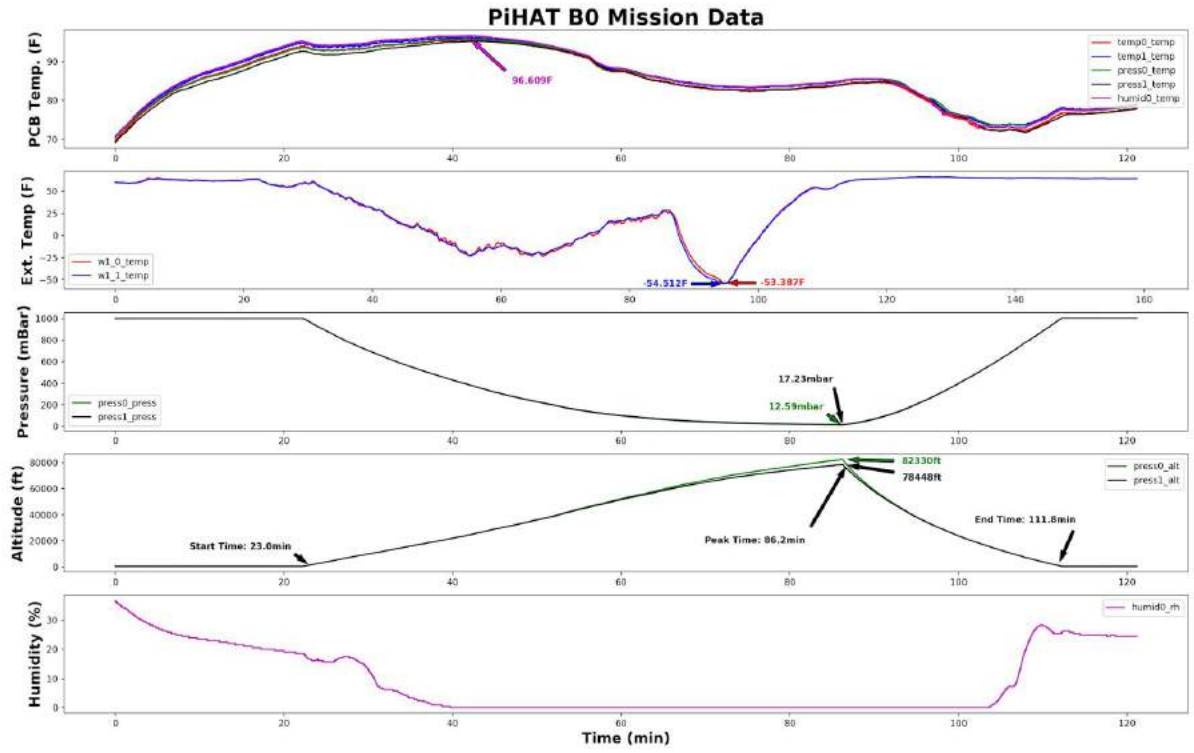

Figure 26. PiHAT B0 Mission Data Plots.

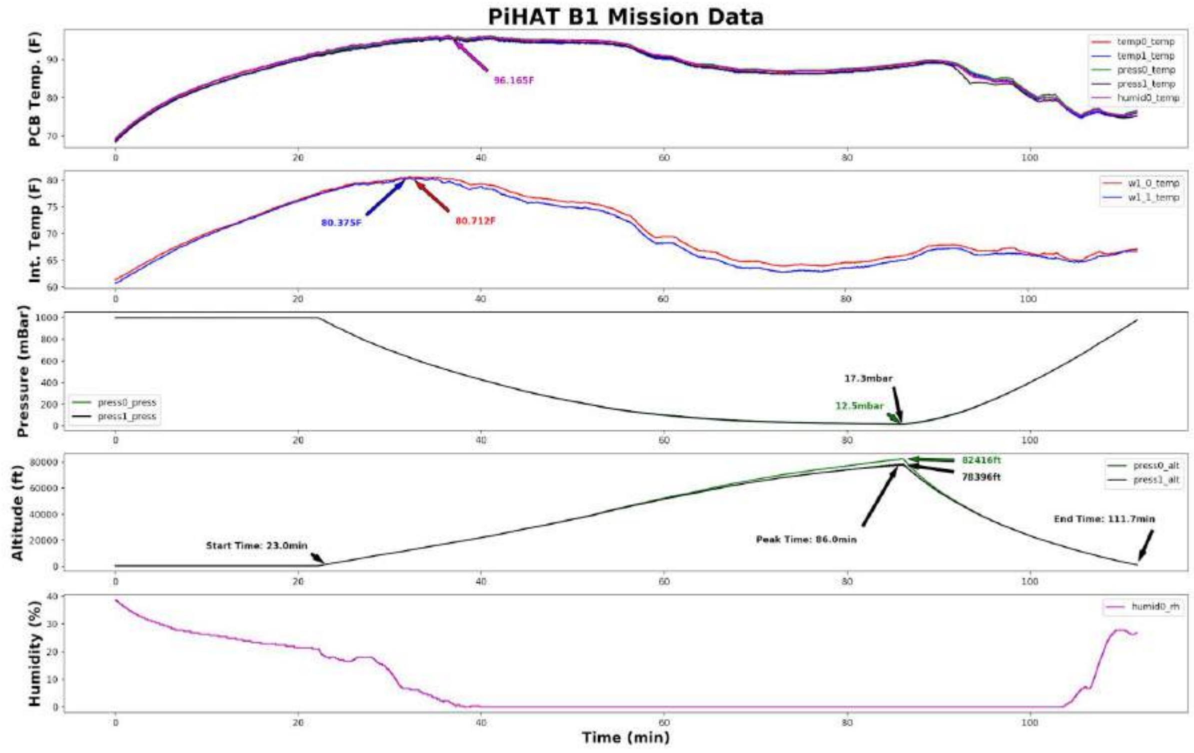

Figure 27. PiHAT B1 Mission Data Plots.

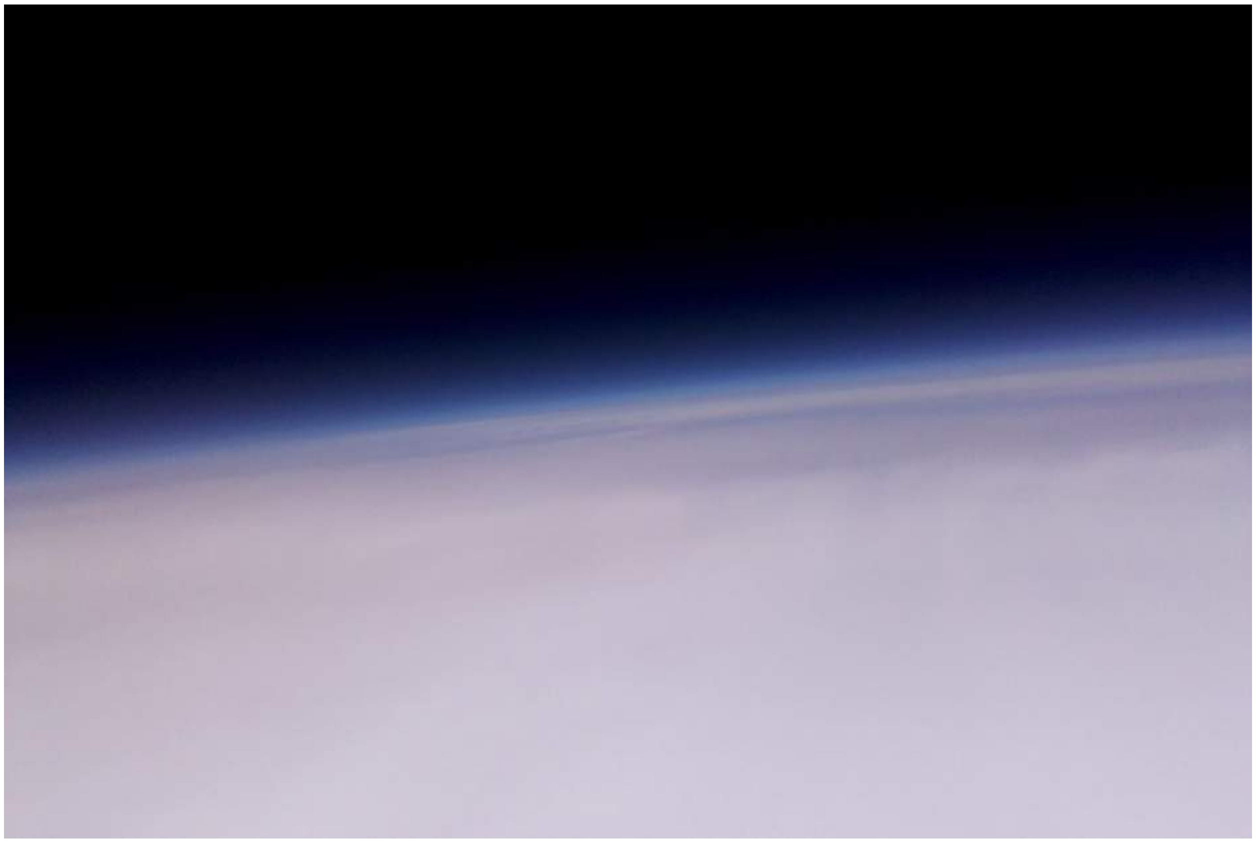

Figure 28. PiHAT B1 Photograph of Earth.

For future launches, some modifications should be made to the PiHAT sensor nodes. First, correct UART cables should be made so that boards B2 and B3 can function properly. Second, each the IMUs on each board should be populated and firmware written to record data. The IMU data can be very useful to determine how the platform moves over time. Third, it was found that camera photos were much more clear and picturesque than the recorded video. Therefore, many more photos should be taken instead of video. For the horizon and earth facing cameras, a suggested change would be 15 seconds of video followed by 10 photos. For the upwards, balloon facing camera, the original 60 seconds of video followed by 4 photos is sufficient since the main goal of the camera is to capture the balloon burst. More testing can be conducted, however, to determine if still photos or video can capture the balloon burst with more clarity since still photos are taken at a much higher quality.

2. *DAQCS Host Board and Reaction Wheel Control*

For the first HAB launch, only the critical functionality was implemented for the DAQCS Host board due to time constraints and the large number of features that could be implemented. On the Host MSP430FR5994 micro-controller, the UART communication between all of the Raspberry Pi Sensor Nodes, SPI communication between the Host and the COMMS Raspberry Pi Board, cutdown mechanism control, reaction wheel battery voltage sensing and simple GPIO based control of the Motor MSP430FR5994. On a software level, SPI communication between the two MSP430FR5994s was also completed but due to some bugs, the functionality was decided to be removed from the final launch platform. Regardless, the Host MSP430FR5994 was able to receive all commands from the COMMS board and either send commands to the Motor MSP430FR5994 or forward commands to a particular Raspberry Pi Sensor Node. The Host would then be able to store local copies of all incoming data from both the Raspberry Pi sensor nodes and the Motor MSP430FR5994. This data would then be sent to COMMS for transmission back to earth.

On the Motor MSP430FR5994, only limited functionality was implemented due to timing constraints and to optimized the reaction wheel control algorithm. All data from the IMU was able to be acquired over a SPI interface. An analog signal from the reaction wheel motor controller was able to be acquired and converted to received the actual speed of the motor over time. Using the built in Real-Time clock unit, all data collected was able to be timestamped down to millisecond accuracy and up to hour accuracy. The read/write speeds of data to the SD card limited the overall speed of the reaction wheel control algorithm when constantly acquiring and storing data to the SD card. In order to compensate for the slow write speeds, built in FRAM was utilized to quickly save over 2 minutes of data collection while the reaction wheel is being used. This allowed the reaction wheel controller to operate at approximately 200Hz. After 2 minutes of running the controller, the controller would be disabled and the data in FRAM would be dumped to a single text file on the SD card. When the motor controller was not operating, data would continue to be logged to the SD card approximately every 100ms, and would log data for 5 minutes at a time. Finally, using all of this data, PWM signals were generated to control the speed and direction of the motor through the motor controller. The final assembled DAQCS board can be seen in Fig. 29 below.

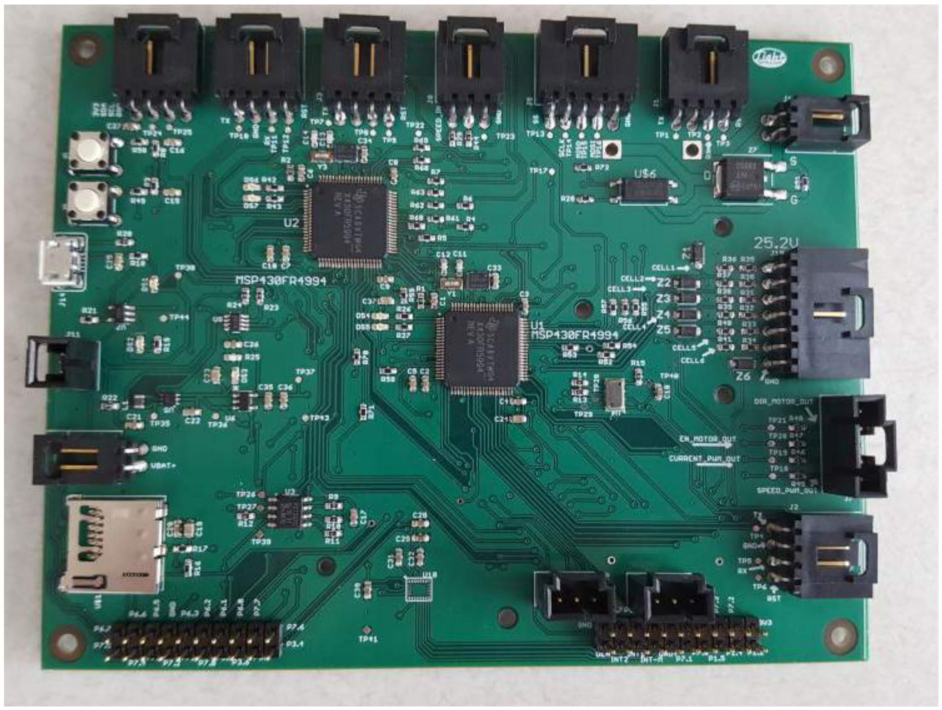

Figure 29. DAQCS Host Board Assembled.

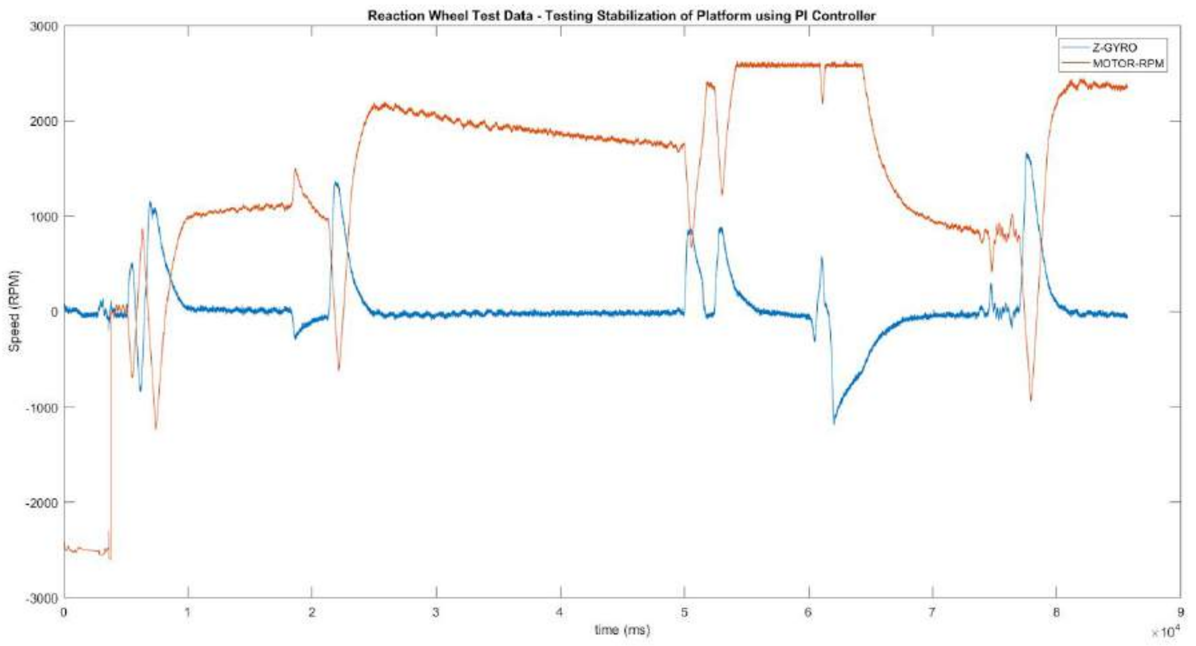

Figure 30. Reaction Wheel and Instrumentation Platform Test Data.

The above plot shows the final test data of the reaction wheel and the instrumentation platform. It can be seen that the approximated 1:170 proportion between the platform and the reaction wheel was proven to be correct. However, unlike what was first thought, the controller ended up becoming a proportional-integral controller, rather than a proportional one. This left two gains to be determined; they were found through testing to be equal to approximately 120 and -0.2 for the Kp and Ki, respectively. It can be seen that after being implemented, the platform maintained its zero RPM angular velocity relatively well, with a steady state being reached within approximately 0.5 second. There was some oscillation around the zero point, which was mitigated by filtering the IMU with an exponential moving average. A boundary was then set at +/-0.2 RPM to further try to mitigate the "bouncing" of the IMU data.

The main DAQCS host board demonstrated full functionality for communication and the reaction wheel prior to the launch. See the next section for details on the reaction wheel. In regards to communication prior to launch, communication via SPI from the COMMs Raspberry Pi Zero to the Host MSP430FR5994 was demonstrated, communication via UART between four UART busses on the Host MSP430FR5994 and all four Pi HAT boards was demonstrated and communication via SPI between the Host MSP430FR5994 and the Motor MSP430FR5994 was demonstrated. Due to the communication between the Host MSP430FR5994 and the Motor MSP430FR5994 being unreliable prior to launch, the interface was removed and a single GPIO pin was utilized to communicate when to turn on the reaction wheel. On launch day, the communication chain of sensor data from each Pi HAT to the COMMs Raspberry Pi to the ground station was demonstrated. At time of launch, the board was assumed to be working and within minutes of launching, the ground station noticed no updated data coming from any of the Pi HATs. When the platform was recovered, the LDO powered both microcontrollers was visibly blown. In addition to this, the SD card was utilized to see logs for the Motor MSP430FR5994 and the SD card wasn't readable on multiple computers. On the third attempt to read the SD card, only two files with roughly 10 minutes were found and the SD card started to melt the adapter. Therefore, one of the potential failures of the board could have been the SD card malfunctioning and causing over-current issues across the board. Thankfully, each Pi HAT had SD cards to store sensor information locally, so the board failure only affected the streaming of sensor information during flight and the entire reaction wheel system. The GRSS maintained full functionality throughout the entire mission and was essential to the recovery of the platform due to the loudness of the buzzer.

Paper Contributions of Each Team Member:

COMMS - P17104:

- Connor Goldberg: 70 cm Chain & Communications Sensors Sections
- Ian Prechtl: Physicalities Section
- Adam Steenkamer: Introduction, Custom PCB, 2m Chain, and Communications Intro
- Matt Zachary: Tracksoar APRS & 2m Chain Sections

DAQCS - P17105:

- Steve Giewont: Host Board Section
- Lincoln Glauser: GRSS Section and assisted with Host Board Section
- Sydney Kaminski: Reaction Wheel Section & Intro to DAQCS
- Chris Schwab: Raspberry Pi Zero Sensor Nodes