5-4-2018

Development of a 2U CubeSat for Imaging the 2017 Solar Eclipse

Sepehr Zangeneh

Development of a 2U CubeSat for imaging the 2017 solar eclipse

By

Sepehr Zangeneh

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Aerospace Engineering
in the Department of Aerospace Engineering

Mississippi State, Mississippi
May 2018

Development of a 2U CubeSat for imaging the 2017 solar eclipse

By

Sepehr Zangeneh

Approved:

_____

Keith Koenig
(Major Professor)

_____

Gregory Olsen
(Committee Member)

_____

Donghoon Kim
(Committee Member)

_____

Ratneshwar Jha
(Graduate Coordinator)

_____

Jason Keith
Dean
Bagley College of Engineering

Name: Sepehr Zangeneh

Date of Degree: May 5, 2018

Institution: Mississippi State University

Major Field: Aerospace Engineering

Title of Study: Development of a 2U CubeSat for imaging the 2017 solar eclipse

Pages in Study 57

Candidate for Degree of Master of Science

The entire contiguous United States experienced a solar eclipse on August 21st, 2017 which passed from the Pacific to the Atlantic Coasts. The path of totality crossed 14 states while other states had partial eclipse. Due to the rarity of this event, it was known as "The Great American Eclipse" and NASA collaborated with 52 universities across the United States to launch weather balloon payloads to record this impactful event. Although Montana State University designed a workshop for all universities involved in order to assist those not experienced in the area, Mississippi State University decided to design our own payload.

Our system was designed in order to meet the standards of a 2U CubeSat. One key aspect of our payload is that it is entirely made from 3D printed parts with over 100 prototype parts made over the length of two years. Instead of buying an off the shelf flight computer, we designed and built a custom Hexa-Processor Computer Board which gave us flexibility with the computation needs. A turret was also developed that housed two cameras and could spin 360 degrees, allowing it to counter act the rotations of the payload in order to obtain a stabilized image. The payload was launched in Kentucky and was a successful flight without any damages to the payload.

DEDICATION

This thesis is dedicated to my parents and Dr. Keith Koenig. Without them this thesis would be blank.

ACKNOWLEDGEMENTS

I would like to begin by thanking Dr. Keith Koenig for agreeing to be my advisor and mentor during my graduate studies. His immense knowledge on literally everything and sense of humor made this journey an absolute joy. I appreciate the time and patience he provided during the past three years. Dr. Koenig never stopped supporting me and without him this thesis would never exist, and for that I am forever in his debt. I also would like to thank Dr. Gregory Olsen and Dr. Donghoon Kim for their advice and evaluation of my thesis.

I like to thank Jacob Rogers for his support in the development of this project. A special thanks to the high-altitude research group, Cayla Hummel, Tyler Howell and Nick Briggs who made all the test flights possible, particularly the solar eclipse event. I also would like to thank Reed Clay for his mentorship in the use of the 3D printers.

Lastly, I would like to thank Ms. Margaret Schaff and Dr. Thomas E. Lacy for providing the funding to make this project a reality.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ABS	Acrylonitrile Butadiene Styrene
CAD	Computer Aided Design
COTS	Commercial Off The Shelf
I2C	Inter-Integrated Circuit
Li-ion	Lithium-ion
Li-Po	Lithium-Polymer
PCB	Printed Circuit Board
PLA	Polylactic Acid
SMD	Surface Mount Device
UART	Universally Asynchronous Receiver Transmitter

CHAPTER I

INTRODUCTION

On June 8, 1918, a total solar eclipse occurred that crossed the United States from Washington state to Florida. [1] Although there have been solar eclipse occurrences since then, none have covered the entire United States until almost a century later. On August 21st, 2017, the contiguous United States experienced a total solar eclipse which passed from the Pacific to the Atlantic Coasts. The path of totality crossed 14 states while other states had partial eclipse. Due to the rarity of this event, it was known as "The Great American Eclipse". [2] NASA's Eclipse Ballooning Project was a nationwide collaboration among high schools, colleges and universities in an effort to live stream the solar eclipse at a hundred thousand feet of altitude. The project consisted of 55 teams across 30 state-based Space Grant Consortia. [3] Montana State University provided a kit and workshop for those involved in the project. [4] Although the majority of the participants of this collaboration used the system provided by Montana State University, it was not mandatory and so the high-altitude research group at Mississippi State University (MSU) decided to design and develop an independent system, one that met the standards of a 2U CubeSat.

CubeSats are a category of spacecraft known as nanosatellites and are commonly used for research purposes. The CubeSat Standard was developed at California Polytechnic University, San Luis Obispo and Stanford University in 1999. CubeSats are classified by their size in units of (U) where each unit is a (100mm x 100mm x 100mm)

cube with mass below 1.3 kg. Common sizes are 1U, 2U, 3U, 6U and 12U. [5][6][7] Although CubeSats are generally meant to be launched into space, the high-altitude balloon group at MSU decided to design the balloon payload to the 2U CubeSat standard for the purpose of gaining experience.

The payload developed in this research includes features not found in normal weather balloon payloads. Two of the more important of these features are 3D printed components and multi-core processing for flight control and data acquisition. This thesis documents the evolution through two prototypes to the system that flew to monitor the 2017 eclipse.

CHAPTER II

PROTOTYPE ONE

2.1 Structure

Many high altitude balloon projects fly on weather balloons, which are typically made of latex. The balloons are categorized by the mass of latex. The largest commercially available latex balloons are 3 kg in mass. It is also typical for high altitude balloon projects to use insulation foam board (such as used in house construction) from which to make the payload box or container. Insulation foam board maintains the heat properly inside the box without requiring a sophisticated active thermal control system. It is a cost-effective method of building, and openings and modifications can be easily cut out of without requiring special tools or machining. Styrofoam containers are also widely used.

In the current project, the structure of the first prototype was constructed from two hollow Styrofoam hemispheres of 12 in diameter. The two hemispheres were held together using three latches as shown in Figure 2.1. The foam was coated with a thin layer of resin to give it further structural strength. The objective of this version was to test some of the electrical components such as the battery and long-range communications before integrating them into a more complex structure of the next versions. Although most weather balloons typically use a rectangular payload shape, a spherical geometry was used

so that the payload would be less likely to get stuck between branches if the system landed in a tree.

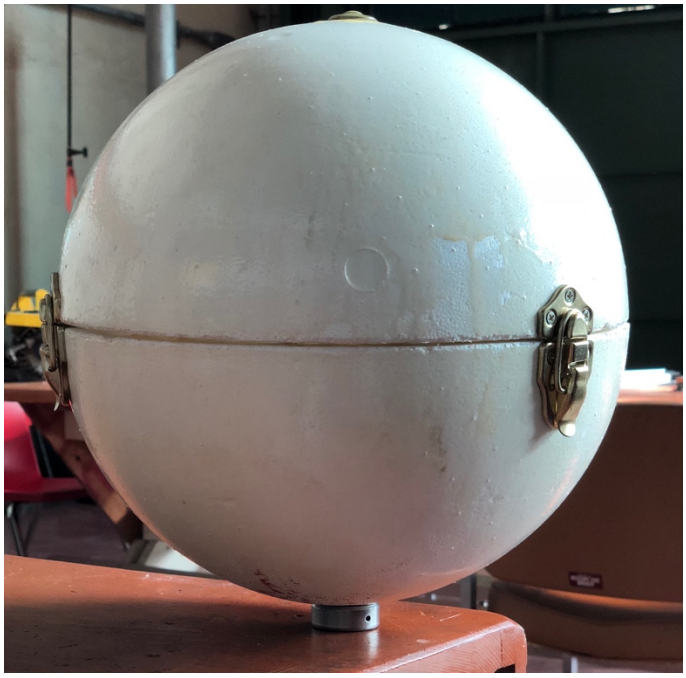

Figure 2.1 Prototype One

2.2 Electronics

The onboard electronics in Prototype One consisted of a flight computer and components for tracking, communication, power and power control and recovery mechanisms.

2.2.1 Flight Computer

An Arduino Uno Board is a prototyping board commonly used by students due to its ease of use and low cost. It has a single Atmel ATmega328 micro-controller running at 16MHz with a variety of peripherals for connecting different sensors and communication devices. These benefits made it perfect for the flight computer. Figure 2.2 is a schematic

which shows how the electronic components are connected to each other. The two headers are connected to nichrome wires which serve as the release mechanism which is explained later in this chapter.

Time constraints for this phase of the project prevented development of a custom Printed Circuit Board (PCB) so the wiring was done on a perfboard. A perfboard is a circuit board with an array of holes across the entire board which allows for wires and components to be mounted and connected.

The 78xx is the voltage regulator that lowers the battery voltage down to 5V which is what the system requires. (Red lines represent positive 5V and black lines represent Ground). The two components on each side of the regulator are capacitors which maintain stability of the regulator. XTend Digi modem is the transceiver which is the primary communication system. The transmit (Tx) pin of the Arduino is connected to the Data In (DI) of the transceiver and the receive (Rx) pin of the Arduino is connected to the Data Out (DO) of the transceiver. The SUP500F is the GPS module where Tx and Rx pins of it are connected to digital pins D2 and D3 of the Arduino. The ULN2003A is the transistor array which activates the nichrome wires. The Y shaped parts represent the nichrome wire connectors. Digital pin D4 and D5 are connected to the inputs of the ULN2003A which controls the activation.

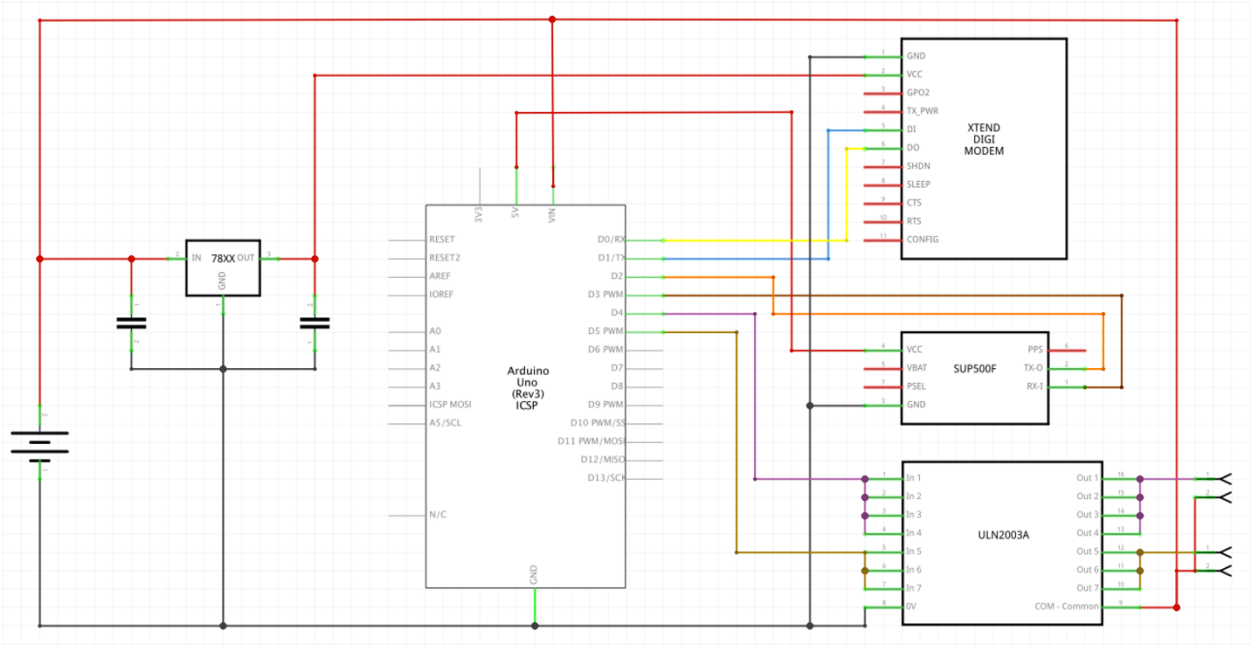

Figure 2.2 Circuit diagram of the electronics in Prototype One

2.2.2 Sensors

A GPS receiver from USGlobalSat (Figure 2.4) was used to track the balloon. This unit runs on 5V and consumes 50mA while trying to get a lock but it will consume as low as 34mA once it is locked. It uses serial communication (UART) to output GPS data using the NMEA standard. The GPS data include UTC Time, Latitude, N/S indicator, Longitude, E/W indicator, Position Fix, Satellites Used and Altitude.

2.2.3 Communication

A wireless communication was a requirement so that the location and status of the payload could be known at all times. The communication range required could be as long as 40 km due to both the altitude and cross-range distance of the balloon from the ground-based tracking equipment. With the exception of amateur “ham” radio equipment, which requires licenses from the Federal Communication Commission (FCC) to operate, there

are very few devices that can provide mobile communication at these ranges. The Digi XTend 900MHz is a license free transceiver which is rated for over 60 km of range when the antennas are in line of sight. It uses a 3 to 5V CMOS UART for the serial data interface therefore it can easily work with the UART port on the Arduino Uno. One XTend would be connected to a ground station computer using a serial to USB converter. The Arduino software has a serial data monitor which can display serial data coming in and also it is capable of sending messages through the serial port. For Prototype One, commands to activate certain features were sent using this method and the location and status of the balloon was displayed here.

2.2.4 Power

2.2.4.1 Battery Selection

Lithium Polymer (Li-Po) batteries are rechargeable batteries that are light weight and have large energy density. Each cell has a nominal voltage of 3.4V, but full charge is at 4.2V and it is recommended that they do not go below 3.2 V for longevity. Cells are categorized by their discharge rate and capacity. Discharge rate shows how much current can the battery provide at instant and capacity is the amount of charge, or energy that can be stored in the cell. Cells can be combined to increase output voltage or capacity depending on the requirements. Since 4.2V is not enough to run the electronics, two cells are put together in series (hence 2S) to double the voltage to 8.4V which then can be reduced to 5V. Both cells have a capacity of 5000mAh which means if 5A are consumed per hour, the battery will last one hour.

2.2.4.2 Voltage Regulator

The flight computer, XTend transceiver and the GPS sensor ran on 5V like the other electronic components. However, the battery pack had a nominal voltage of 8V which would damage these modules. In order to step down the higher voltage to the lower voltage, a DC/DC converter, also known as a voltage regulator, was required. There are two types of regulators, classified by the type of conversion method; linear or switching. Linear regulators use a resistive load to control the output voltage, which can lead to significant heat generation and low efficiency. However, linear regulators are simple to implement and few components are needed. Switching regulators on the other hand are very complex in design and require many components but have high efficiency and low heat generation. [8]

Since simplicity was the main idea in the first prototype, linear regulators were chosen. An LM7805 Regulator was used. This regulator can reduce input voltages of up to 25V down to 5 V at a max current of 1.5 Amps. [9] The Arduino Uno board has an onboard 5V regulator, but can only supply 500mA before it starts to generate heat and shut down. Since the XTend can consume up to 1A when running at its full power of 1W, the external regulator was used to power the communications while the onboard regulator powered the Arduino itself and the GPS module.

2.3 Release Mechanism

The FAA requires all high-altitude balloons (HAB) to have two methods of flight termination according to FAR 101.35. [10] Since at high altitude balloons will burst, it can be considered one of the two termination methods. For the second method, a physical mechanism is required to release the payload from the balloon. Prototype One used

nichrome wire wrapped around the rope linking the balloon and payload. Voltage applied to the wire, caused it to heat up and cut the wire. Because the current through the wire could reach up to 2A a ULN2003 was implemented. This IC is known as a Darlington array which contains 7 transistors each capable of conducting 500 mA. Four transistors were paired together to allow 2A to flow through the wire. These transistors were controlled by the Arduino Uno's digital pins as seen in Figure 2.3. The command to activate these digital pins were sent through the wireless communications. The landing zone of weather balloons are not controllable and they will often land in a tree, making recovery difficult. To remedy this situation, a second nichrome wire was added between the parachute and payload. This wire would cut the payload away from the parachute and ropes and possibly let it drop to the ground.

2.4 Flight and Recovery

In July 2015, Prototype One was flown at Mississippi State University North Farm using a 1200-gram weather balloon. Communication was lost at around an altitude of 14,0000 meters and never re-established. Ten months later the system was found in Alabama. The battery and flight computer were damaged excessively. The loss of communication was concluded to be the extended range between the transceivers, specially because their antennas were no longer in line of sight.

CHAPTER III

PROTOTYPE TWO

3.1 Structure

3.1.1 Additive manufacturing

Additive Manufacturing (AM) refers to a method of manufacturing 3D objects by adding layers on top of previous layers of material. 3D printing or rapid prototyping are other names used for this method of manufacturing. Materials can range from plastics to metals with plastic being more common and affordable. There are many types of additive manufacturing technologies which are suitable for a wide variety of users. Selective Laser Sintering (SLS), Stereolithography (SLA) and Fused Deposition Modeling (FDM) are the most common technologies used today.

SLS uses a high-powered laser to fuse small particles together to create a single piece. These particles can be made of plastic, metal, ceramic or glass. The advantage of this technology is that support material is not required since not particles that are not fused will act as support. The price of a machine using this methodology is very expensive, specially with a large build volume.

SLA uses a laser to cure a photopolymer resin layer after layer. Unlike SLS it will require support material. A tank filled with the resin and a build platform lowers and submerges into the resin. The laser then traces the cross section of the model and cures the

resin which creates one layer. This process is repeated until all layers have been made followed by an alcohol bath which removes uncured resin and finally cured to the optimum state using UV light. This method is similar to SLS in which it is expensive and large parts are not easily printed in one piece.

FDM uses thermoplastics to create the model. Plastic filaments are fed through a heated nozzle which traces the cross-section pattern and then deposited onto a platform to create a layer. The nozzle is then raised by a step and the process is repeated until the build is completed. The smaller the size of the steps, the finer the piece will be. This technology allows for larger build volumes and requires less post processing on the finished parts.

Aleph Objects Incorporated produces the Lulzbot TAZ 6, a 3D printer that uses the FDM technique and is widely used due its simplicity and ease of repair. It has a print volume of 280mm by 280mm by 250mm which is sufficient enough to print a 2U CubeSat. It can print using a variety of materials like ABS, PLA, NGEN.

3.1.2 Design

Prototype Two was radically different from Prototype One. The entire system was modeled in Solidworks CAD software so that it could be 3D printed. Originally the design was supposed to meet the CubeSat 2U dimensions, however the electronics package and battery increased the overall size to 110mm by 110 mm and a height of 220mm. The system was composed of a rotating turret, or payload module, housing a camera, a service module containing the power and communication equipment and solar arrays for increased energy. Figure 3.1 shows the exploded view of the Prototype Two. Solar arrays are installed on the sides of the CubeSat and on foldable panels which are hinged at the bottom.

The flight computer is the red component in the bottom and the payload is the top unit, both are discussed in detail later in this chapter.

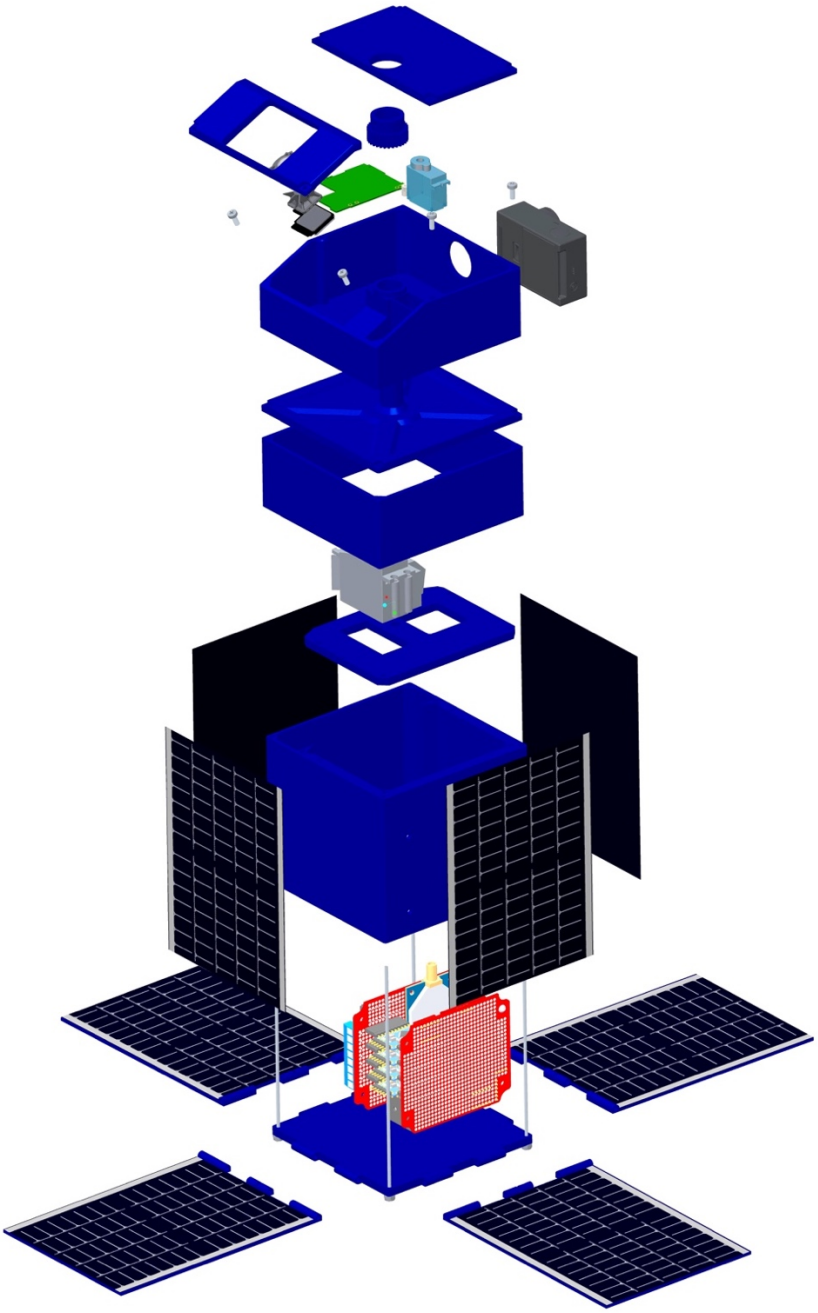

Figure 3.1 Exploded view of the Prototype Two.

To create a robust structure, the side walls were printed in one single piece shown in Figure 3.2 rather than separate panels like in traditional CubeSats. The flight computer and batteries are secured inside this structure. Aluminum threaded rods went through each corner, holding the entire CubeSat together.

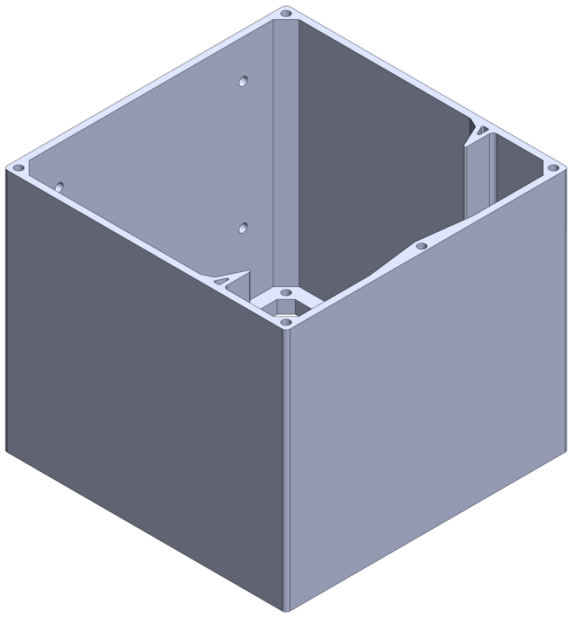

Figure 3.2 Primary structure of Prototype Two

3.1.3 Manufacturing

All the plastic components of prototype one where built by two 3D printers, a Lulzbot TAZ 5 and TAZ 6. PLA filament was used for the entire structure while the solar panel supports where printed with white ABS.

3.2 Electronics

3.2.1 Flight Computer

To improve the processing capabilities of the flight computer, four micro-controllers were used in parallel. Each processor ran a portion of the overall code unlike

Prototype One which used only one processor. This method allowed for more sensors and system functionality. Although the Arduino Uno board has many ports and connectors, it comes at a cost of size. In order to use four of them together and keep the dimensions of the flight computer small, the pro mini version of the same board was implemented. This board has the same processor as the Uno but all the connectors and ports have been removed to make it smaller.

Each processor needed to be able to communicate with each other in order to share data and values from the sensors. I2C, SPI and UART were three methods of communications available. Table 3.1 shows a comparison between the three. There is only one UART module on the processor which is connected to the transceiver. SPI is faster than I2C and is full-duplex, meaning it can read and write at the same time. SPI requires 3 main pins and a chip select pin for every slave. Therefore, a total of 6 pins would be required. I2C is half-duplex which makes it slower but it only requires 2 pins and therefore it was chosen as the method of communication among the processors.

Table 3.1 Comparison between UART, SPI, I2C [11]

Features	UART	SPI	I2C
Type of Communication	Asynchronous	Synchronous	Synchronous
Data Rate	230Kbps to 460Kbps	10Mbps to 20Mbps	100Kbps to 3.4 Mbps
Pins Required	2	4+	2
Duplex	Half	Full	Half

3.2.2 Sensors

An assortment of sensors was added to the flight computer in order to better understand the behavior and conditions of the payload during its flight. To determine the attitude of the payload, a 9 Degrees of Freedom (DOF) Inertial Measurement Unit (IMU)

was used. A 9DOF sensor houses a triple axis accelerometer, gyroscope and compass in one single chip. It was connected directly to the I2C bus of the flight computer. A Kalman Filter was used to convert the raw data to Pitch, Roll and Yaw.

Altitude determination is important when using weather balloons. Prototype One relied solely on GPS data of measuring the altitude. Atmospheric pressure provides an alternative measure of altitude and can be compared to the value from the GPS for better accuracy. MPL3115A2 is a pressure sensor that can measure from 50kPa to 110kPa. It is internally temperature compensated and has an altitude resolution of 0.3 m. It also provides temperature measurements. It only supports I2C for data output. This sensor was used in Prototype Two.

A humidity sensor was also added to monitor the level of humidity during flight to determine if waterproofing the payload is a requirement. It also provided measurement of temperature.

Because the chances of losing communication during flight is highly possible, it is best to record the sensor's data on a storage device so it can be analyzed later. SD cards are the best device to record and access with a personal computer. Processor Two took the data from all the sensors and placed it in a single text string. The string was recorded on the SD card. The process was repeated throughout the flight. If any power losses occurred, stored data was not lost. In addition, if power was lost intermittently, the current data on the SD card was not overwritten.

3.2.3 Communication

The primary communication continued to be the same as Prototype One which was the Digi XTend 900MHz. However, since it might be difficult to locate the payload after

landing, a secondary communication method was necessary. To provide this redundancy, a cellular board or shield was connected to the second processor. A cellular connection allowed for GPS readings to be sent by the payload using text messages.

3.2.4 Power

Due to the additional sensors and processors, a 5V regulator with higher current limit was required. Also, because the IMU and pressure sensor ran at 3.3V, a secondary regulator was required to provide that voltage.

Figure 3.3 shows the flight computer in its full assembled status. Figure 3.4 displays all the components taken apart with labels to identify each part.

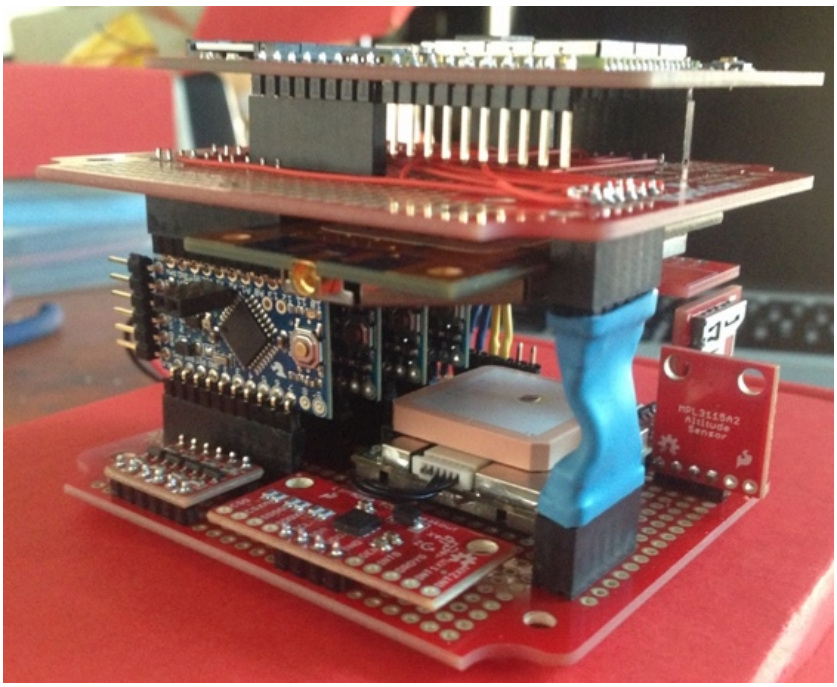

Figure 3.3 The flight computer fully assembled

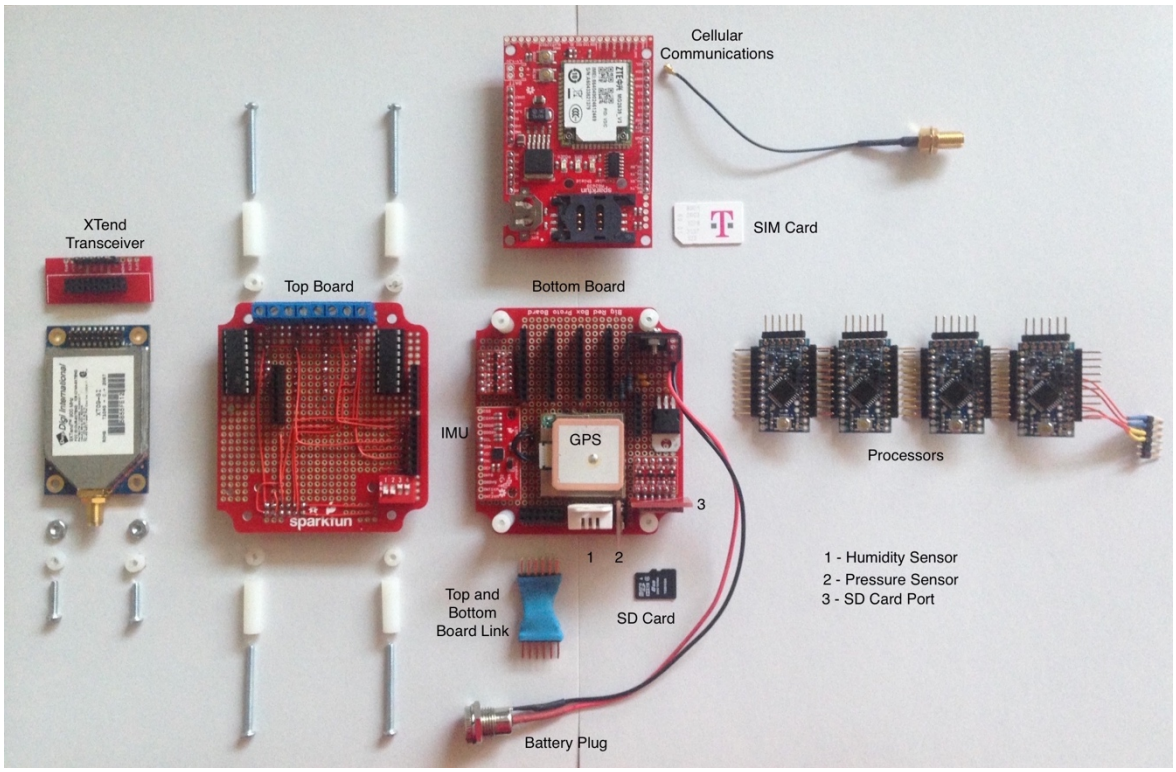

Figure 3.4 An overview of all the components inside the electronics module

3.3 Release Mechanism

The nichrome wire used in Prototype One as a release mechanism resulted in inconsistent results. Ambient temperature would affect its ability to fully cut the rope and the time the wire was powered had to be adjusted each time a different rope was used. For Prototype Two, a new release mechanism was designed where a metal pin holds the rope between two supports. Once the command for release was sent, a servo-motor pulled the pin back, releasing the rope. This system was more reliable and easier to set up. Figure 3.5 shows the mechanism. The image on the right shows the pin (in red) in its fully locked position. A rope (not shown) looped around the pin. Once the release command was sent, the pin was pulled back as shown on the left. The rope loop slipped off the pin allowing the payload to fall away from the balloon.

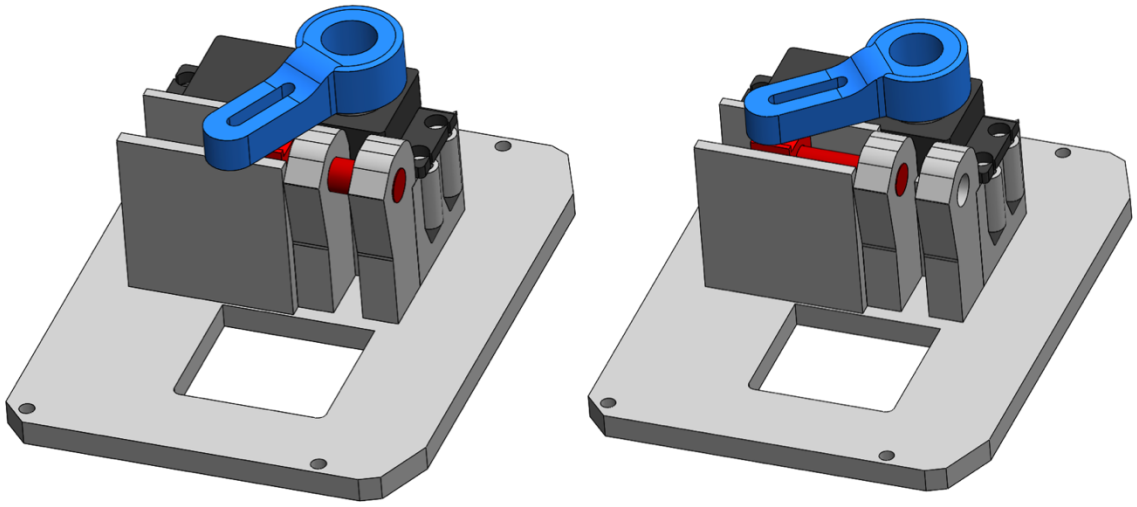

Figure 3.5 Release Mechanism on Prototype Two. Left image – locked, right image – released.

3.4 Payload

The payload of Prototype Two was intended to image the solar eclipse. To save weight and space, a GoPro Hero action camera was used. To further reduce its size and mass, the camera was taken apart and only the essentials were kept. Critical plastic structural components were retained and implemented into the payload design to secure the individual camera components.

Unstabilized weather balloon payloads continually rotate and sway in flight. This limits the amount of view time the camera has to record the solar eclipse. To remedy this issue, a turret mechanism was developed. A stem was integrated on top of a plate which also acted as the top cover for the CubeSat. The payload unit could rotate around this stem using a continuous servo-motor. Commands would be sent from the ground station to the payload to activate the servo-motor in order to rotate the cameras to a certain orientation.

During full sun and partial eclipse, a biaxially-oriented polyethylene terephthalate (BoPET) film was used to protect the camera sensors. (A brand name for BoPET is Mylar.)

During total eclipse, a servo-motor was used to move the filter away from the camera lens to permit more light to enter the camera.

Wires that carried the signals for controlling the servo, camera control and filter passed from the service module into the payload module through the stem. The wires unfortunately limited the rotation of the payload module to only 200 degrees. Figure 3.6 shows the payload mounted on Prototype Two.

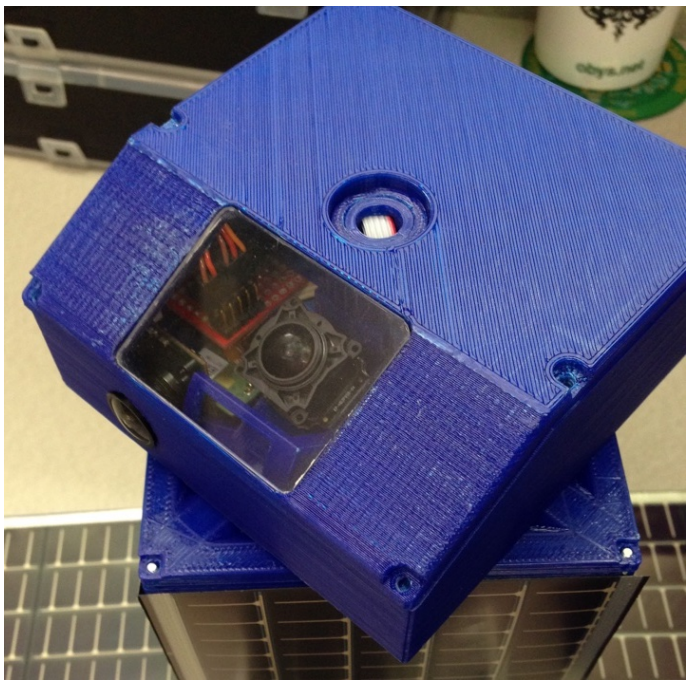

Figure 3.6 The payload attached on top of Prototype Two

3.5 Flight and Recovery

In October 2016, Prototype Two was flown at Mississippi State University North Farm using a 1200-gram weather balloon. Communication was lost at around an altitude of 7600 meters and never re-established. Two months later the system was found in a tree in Alabama. Figure 3.7 shows Prototype Two fully recovered. The onboard SD card gave hints to what happened during flight and why communication was lost. The excessive

amount of current going through the 5V regulator heated it to shutdown temperatures in order to protect itself. This occurred multiple times during flight which caused intermittent data storage which made it hard to understand the maximum altitude it reached and the path that it took. Since communication is also powered by the same regulator, it did not receive enough power to transmit data.

The structure was damaged during landing where it could no longer be assembled. There was extensive water damage inside the electronics and payload. Although corrosion was visible on the flight computer, it was still functional. The camera inside the payload would power on but would not record vide or capture pictures. One battery cell also would not maintain charge anymore.

Figure 3.7 Prototype Two Recovered

CHAPTER IV

FINAL VERSION

4.1 Design

The final version of the eclipse imager vehicle was redesigned from scratch with the dimensions returning to CubeSat standards. Two issues with Prototype Two were the difficulty in modifying the internal components as the design developed and the difficulty in assembling it before flight. The final version addressed both of those issues by employing a foldable structure which was inserted into a box. Figure 4.1 shows the final version. The image was taken after it was recovered from its eclipse flight.

The system is divided into two parts, the service bus and the payload. The service bus contains the flight computer, wireless communications, batteries, heatpad, cut down mechanism and Spot tracking unit. It can be unfolded in order to modify or replace the components inside with ease. Once it is ready, it is then folded, inserted and secured into its enclosure which adds another layer of thermal and structural support. The service bus has shelves where modules can slide in and out and are secured in using screws. Figure 4.2 displays the service bus unfolded.

The payload is attached onto the service bus using two internal clamps. In order to detach the payload from the service bus, two corners of the payload plate are pushed in. The left image in Figure 4.3 shows the payload clamps which attach to the circular opening on the service bus.

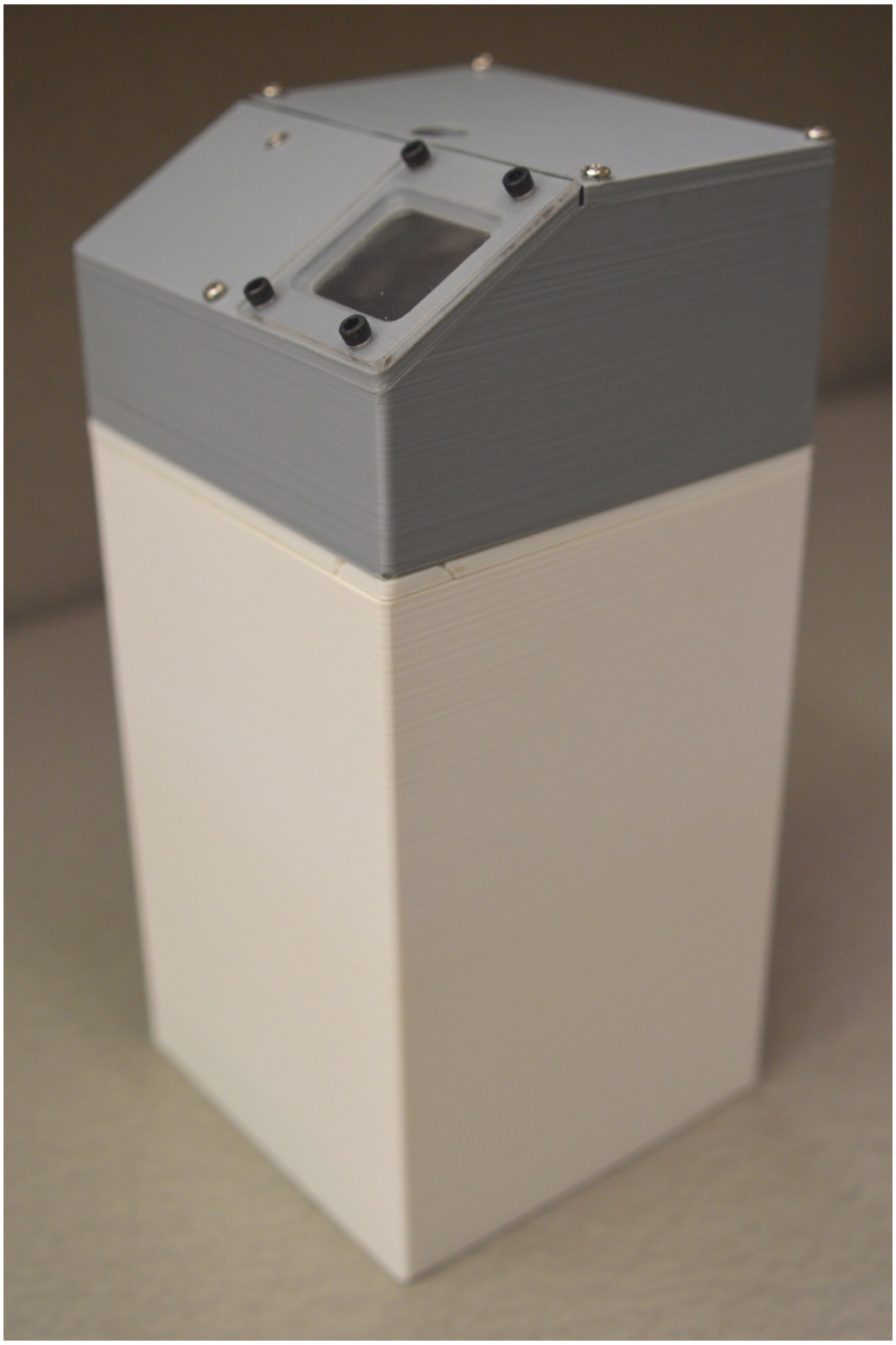

Figure 4.1 Final Version of the Solar Eclipse Photographing System (SEPS)

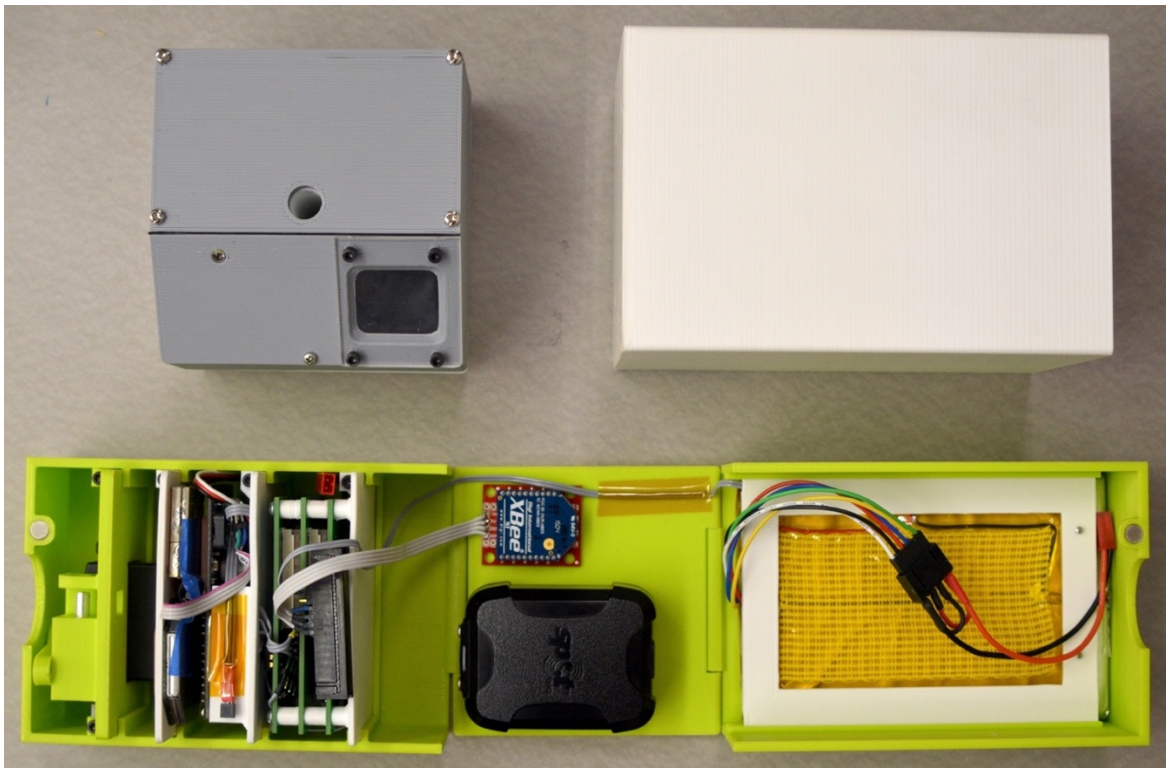

Figure 4.2 Service bus unfolded - bottom. Service Bus enclosure - top right. Payload - top left

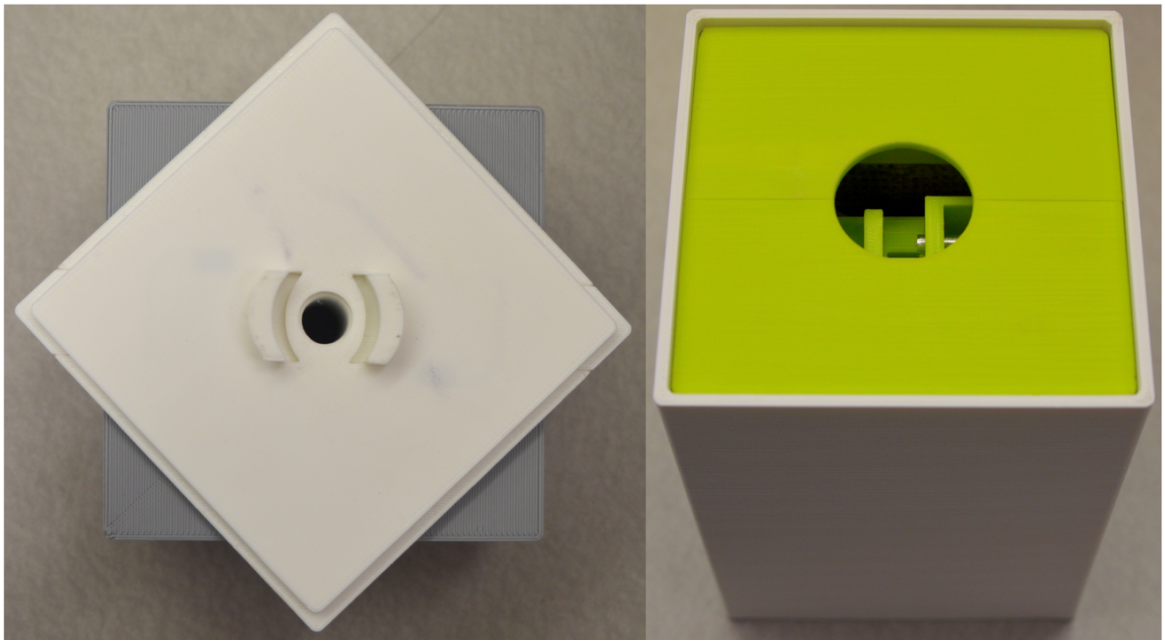

Figure 4.3 Payload on the left, service bus on the right

4.2 Electronics

4.2.1 Flight Computer

4.2.1.1 Circuit Design

To reduce the size of the flight computer, it was designed using Surface Mount Device (SMD) components mounted to Printed Circuit Boards (PCBs). The Atmel ATmega 328p processor used in the Arduino Uno and Pro mini demonstrated some shortcomings in flash storage and SRAM. The flash storage is where the code is stored and on processor one of Prototype Two flight computer, it had already reached 80%. This prevents future software upgrades to be added. This processor is also limited on SRAM and when most of it is used, the processor starts to exhibit lag and freezes. Furthermore, it only has one UART port.

Since the board is being designed from the grounds up, a new processor was selected. Table 4.1 compares three common Atmel processors. Although they have different hardware specifications, they are programmed the same way, which makes the transition, far easier. Atmel ATmega 2560 has four UART ports, 256Kbytes of storage and four times the amount of SRAM as Atmel 328. However, it is a larger chip and takes more space on the board. The Atmel ATmega 1284 has half of the storage of the 2560, but double the amount of SRAM and has two UART ports at a smaller size.

Table 4.1 Atmel Processor comparison [12], [13], [14]

Processors	Speed	Flash Storage	SRAM	Total Pins
Atmel 328	16MHz	32Kbytes	2Kbytes	28
Atmel 2560	16MHz	256KBytes	8Kbytes	100
Atmel 1284	16MHz	128KBytes	16Kbytes	44

The final flight computer had six Atmel 1284 processors on a single board. Processor one was dedicated to collecting data from other processors, communications in and out of the service bus. It also managed data storage on the onboard SD card. All 6 processors communicate with each other using I2C.

A chip is used to upload code onto the processors using a USB cable. On Arduino Uno, this chip is installed onboard. On the Pro Mini, this chip was removed to reduce the size of the board. A special adapter which contains the chip was used to program each individual processor which made the task of programing each of them a bit of hassle. To improve on this, the new flight computer includes this chip and uses multiplexers and a dip switch to easily switch between the processor being programmed.

Most COTS GPS which are available to civilians have a limit which prevents them from tracking location once the device is traveling at higher speeds of 1,000 knots or altitudes of 18,000 meters. Depending on the manufacturer of the unit, some limit the tracking if either situation is met or both and some do not limit at all. The service bus had two GPS units, one that limits tracking and one that did not.

Figure 4.4 shows the block diagram of the electronics inside the service bus. The flight computer is labeled as CPU. The wireless communication units are on the right. The peripherals that are connected to the CPU are on the left with labels that indicate the type of port they use. The payload block diagram is discussed in detail later in this chapter.

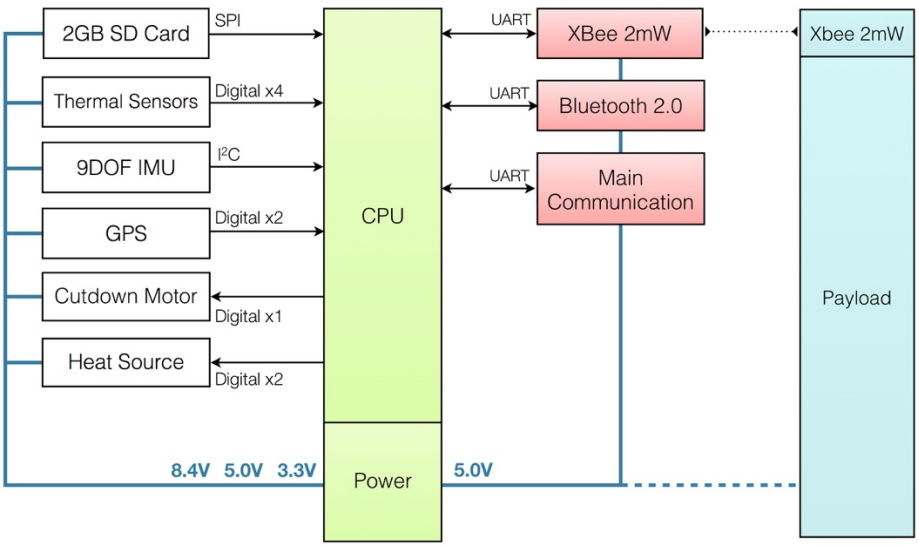

Figure 4.4 Block Diagram for service bus

4.2.1.2 Printed Circuit Board

For the PCBs developed in this thesis, EAGLE PCB Design software was used. First, the schematic of the drawing was laid out in the software. The schematic is a drawing of how the components are wired to each other. An example of this is presented in Figure 4.5 which shows part of the flight computer schematic for the final version. The example shows two processors that are labeled as core. The capacitors and resistors are required for the processor to run at the desired speed. The boxes with text inside are called labels and two boxes with the same name represent a wire connection. This is useful when connecting two pins that are far apart and not have wires going around, which makes the schematic more legible.

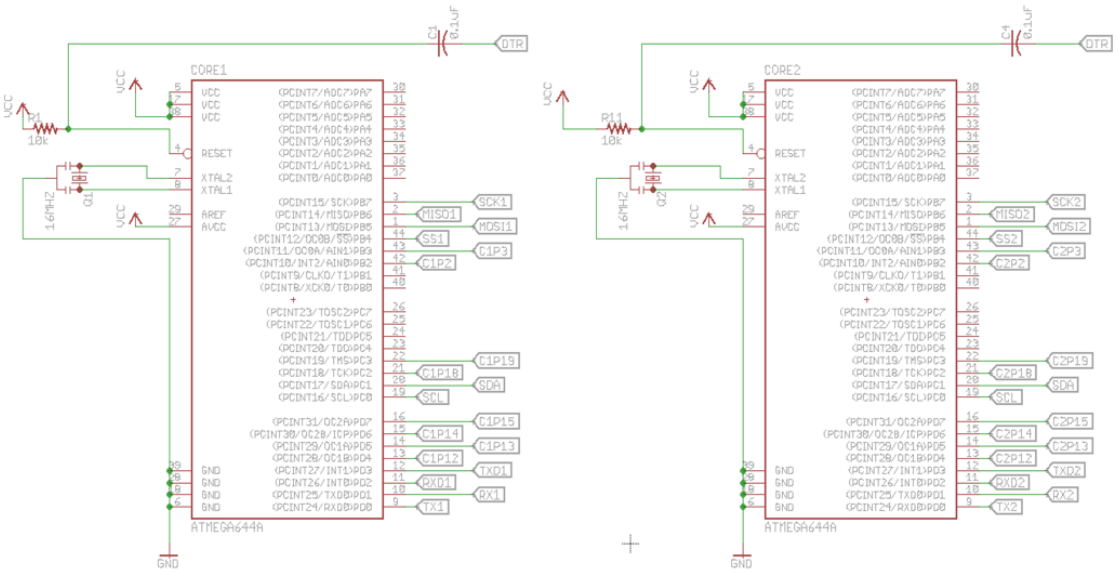

Figure 4.5 A section of the wire schematic in EAGLE Software

Once the schematic of the desired board is done, Eagle can produce a set of files that can then be sent to a company that manufactures the board. These files contain information on locations where holes must be drilled, Trace (wire connection between components) layout of each layer and solder pads where components will be installed. The software can automatically generate the traces and vias (wire connection between the different layers) however, the user must place the components. The location of the components is very critical in order for the software to be able to trace all the connections. The position of the processors, FTDI chip, multiplexer and all the resistors and capacitors were moved and oriented at different orientations until an optimum arrangement was obtained. Due to the density of components in the flight computer, a two-layer board did not suffice, therefore the design was implemented on a four-layer board. Figure 4.6 shows a section of a three layers board. The traces are the green lines on top and the vias are the gold connections.

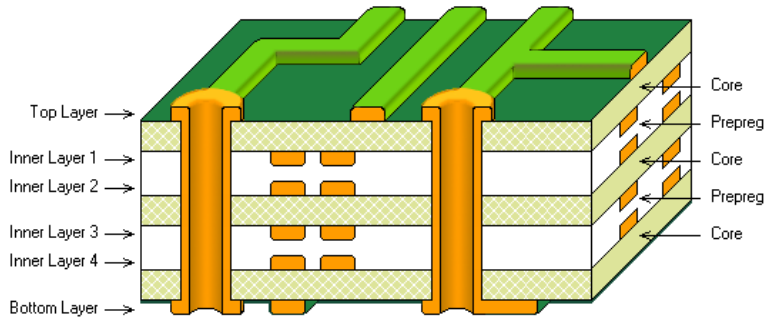

Figure 4.6 Traces and vias in a multilayer PCB [15]

Figure 4.7 shows the tracing that EAGLE has generated. Blue lines represent bottom layer traces, red is top, pink and light blue are middle. The very small dots are vias. The bottom left section is the flight computer. The bottom right is the power supply board. The top right is the SD card and Bluetooth board. The top left is not part of the CubeSat and it is only used to test the processors before soldering them onto the board.

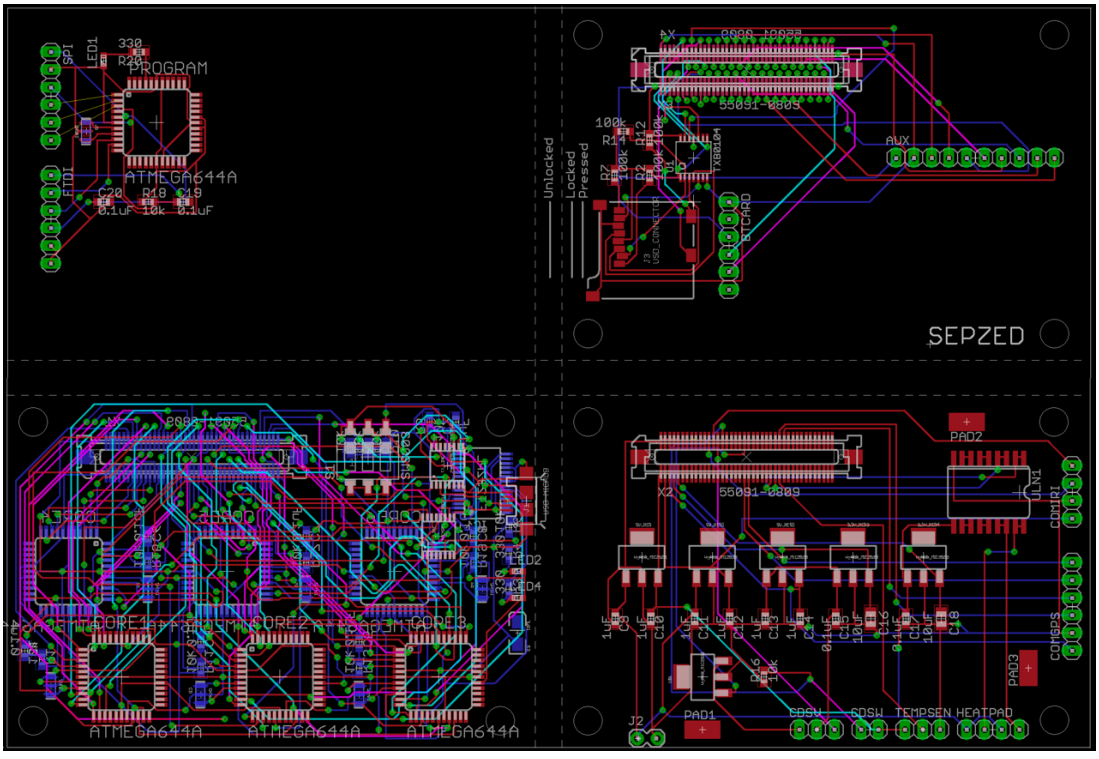

Figure 4.7 Board Design of Flight computer

Figure 4.8 shows each board separately after each component has been soldered on. The board image in each quadrant matches with the quadrant in Figure 4.7 except the top left image, which is the three boards assembled together to create the flight computer.

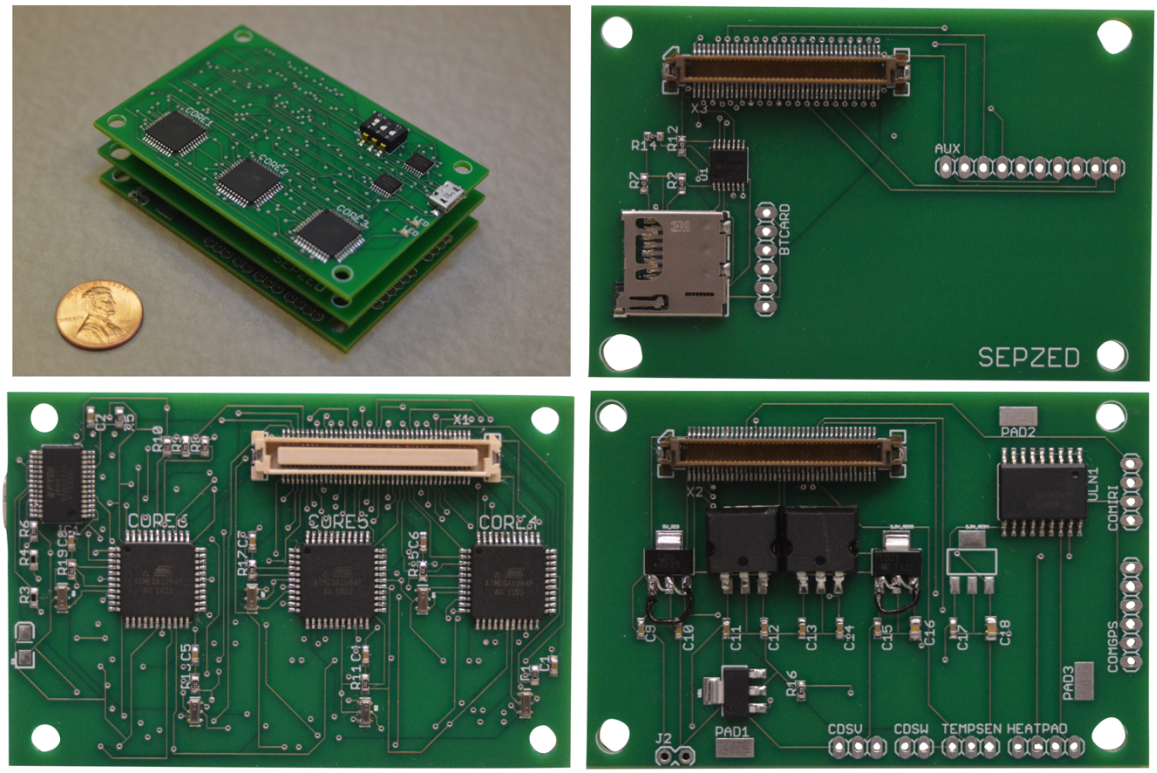

Figure 4.8 Flight computer in separate and combined format.

4.2.2 Power Management

4.2.2.1 Battery

The battery capacity was increased over that of Prototype Two to accommodate additional devices. A four cell 14V, 2800mAh battery was purchased and then split into two cells in parallel and two in series. By splitting it into two, the battery capacity was doubled and voltage was halved, creating a 7.4Volt battery with a capacity of 5600mAh that better suited the electronic system. A heat pad is installed on top of the battery pack.

The flight computer uses a temperature sensor to monitor the temperature inside and controls the amount of heat generated in order to maintain an optimum temperature inside the payload. This ensures that the battery can deliver its maximum capacity at all times.

4.2.2.2 Voltage Regulator

The single regulator in Prototype Two resulted in overheating issues and caused power failure. To remedy this, a separate power supply board was designed with five linear regulators; Three 5V regulators and two 3.3V regulators. One 5V regulator provided power to the flight computer itself, and the other two 5V regulators powered the communications and other accessories. This method ensured that if the current draw through one regulator exceeded its limit, causing the regulator to cut power, the flight computer could continue to operate. 3.3V regulators provided the lower required voltage of the SD card and sensors.

4.2.3 Communications

4.2.3.1 Internal

The CubeSat bus was equipped with a wireless transceiver in order to minimize wiring between devices that are far from each other or when multiple wires were required. It was primarily used for transmitting data between the bus and the payload. However, if additional systems were added to the same system, they could join the network and exchange information. An Xbee 1mW Series 1 transceiver in Figure 4.0 was used for the internal communications. It used 50mA during transmission at a frequency of 2.4GHz. It used UART as the primary method of data input and output.

4.2.3.2 External

The XTend 900MHz transceiver performed poorly on the first two prototypes, due to the line of sight requirement of the antennas. As a result, both prototypes were lost (although both were later found by members of the public). Satellite communication on the other hand does not rely on line of sight and distance which means the CubeSat can land very far from launch zone and ground station could know its location. The Iridium constellation currently has 66 active satellites in orbit and the module in Figure 4.9 uses the network to send and receive short messages from anywhere on earth. Messages can be transmitted and received using UART com port which makes it easy to connect with the flight computer.

Figure 4.9 RockBLOCK Mk2 – Iridium Satcomm Module [16]

A Bluetooth module was added to the flight computer in order to facilitate the testing process of the CubeSat. Most personal computers sold today come with a Bluetooth chip installed, therefore it was easy to connect the CubeSat to any computer wirelessly and communicate. This feature was very useful during the testing of the Lock on Sun Turret

(LOST) since the CubeSat was on a rotating platform to mimic the rotations of the weather balloon.

4.3 Release Mechanism

The release mechanism carries the same design from Prototype Two with a slight redesign in order to have a smaller size and weight profile. Additionally, the power is cut to the servo after it has moved the pin. This prevents from any electrical noise activating the release mechanism. Figure 4.10 shows the updated release mechanism.

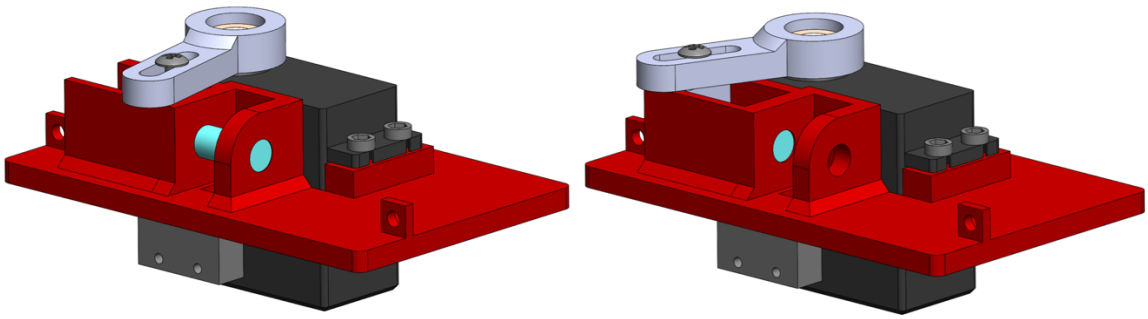

Figure 4.10 Release Mechanism on prototype two. Left image – locked, right image – released.

4.4 Payload

The design of the payload continues to use the turret architecture. However, to improve it from the previous version, the wires connecting the service bus to the payload were removed. This improves the assembly process and allows the turret to spin continuously. To accomplish this task, a separate electronics and power system had to be

designed specifically for the turret tasks. A dual processor computer was used to control the status of the camera, the orientation of the turret and the position of the Mylar filter.

Two cameras were placed inside the payload. A GoPro Session 5 was positioned on the rear of the payload to record the horizon and a GoPro Hero 3 was placed at the front to record the solar eclipse. The main goal from the beginning was to determine the location of the sun and constantly maintain the sun in the field of view (FOV) of the Hero 3 camera. The location of the sun can be determined in terms of Elevation and Azimuth by obtaining the time and location of the observer which can be seen in Figure 4.11.

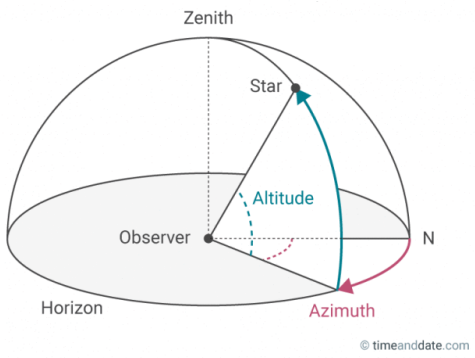

Figure 4.11 Elevation and Azimuth of a star in respect to an observer [17]

Kentucky was one of the states in the path of totality and the closest state to Mississippi, therefore it was chosen as the launch site. In Kentucky, total eclipse occurred at noon local time. Using the location and time, the position of the sun was determined using the Matlab script in Appendix A. It was found that the elevation would be at 60 degrees and therefore the camera was fixed inside the payload at that angle. The vertical component of the camera's field of view was 37.2 degrees which was sufficient enough to maintain the sun in the camera's view during the two-hour flight time. For this reason and

the limited space inside the payload, the elevation angle of the camera was not actively controlled. However, to maintain a lock on the sun, the azimuth must be actively controlled. To achieve this, the flight computer uses the location and time data of the GPS to determine the current azimuth of the sun, then it sends that value to the Payload. The processors inside the payload use a 6 Degrees of Freedom (DOF) IMU to determine the direction at which the turret is pointing at. It then uses a PID control law to calculate the amount of power it should give to the turret's servo motor in order to spin it and point at the sun. This process is repeated constantly in order to maintain the sun in the cameras view.

A PID (Proportional, Integral and Derivative) is the most commonly used control law in the industry due to its robustness and simplicity. The concept is to stabilize a naturally unstable system by reading its current state using a sensor, then computing the error between the desired state and current state. The error is then used to determine the proportional, integral and derivative responses and summing these three components produces an output. The output is what drives the servo motor of the turret. Each component of the PID has different effect on the output. The proportional component has the strongest effect and it only depends on the error. Integral component sums the error over time. The derivative component deals with the rate at which the error changes. Equation 4.1 is the PID equation. Kp is proportional gain, Ki is integral gain and Kd is the derivative gain. The values of these gains affect the strength of each response and must be properly tuned. [18]

$$Output = K_p e(t) + K_I \int e(t)\, dt + K_d \frac{d}{dt} e(t) \tag{4.1}$$

$$e(t) = Setpoint - Input$$

A 6 DOF IMU is a sensor that has an accelerometer and gyroscope in a single chip. In situations where yaw is being determined like in this project, a 9DOF IMU is preferred because it has a compass which corrects for yaw drifts. However, the servo and stepper motor inside the payload will cause magnetic interference on the compass which can cause inaccurate results. This was the reason behind the use of a 6 DOF IMU, where the yaw drift was manually corrected over time by the processor. Although this practice is not recommended, it did produce acceptable results for a two-hour flight.

The GoPro Hero 3 would not fit at an angle inside the payload, therefore it was taken apart from its enclosure and only the necessary components were kept. Also wires were connected to the three buttons that control shutter, menu and Wi-Fi allowing the camera's mode to be changed even during flight. The horizon camera was not modified.

To power all the components inside the payload, two Li-ion batteries rated at 1850mAh were used. Figure 4.12 shows the block diagram of the system inside the payload. The CPU is the dual processor board. The peripherals that are connected to the CPU are on the left with labels that indicate the type of port they use. The Payload communicates to the service bus using the Xbee 2mW wireless module.

Since in Prototype Two the Mylar filter didn't cover the entire view window, light would leak in and reflect onto the filter and ruin the image. To resolve this issue, the mechanism was changed to a linear movement. Rails allow the filter to move back and forth and also prevent light from leaking inside. The actuator was a very small stepper motor with a lead screw. An L293 motor driver IC received the signal from the flight computer and applied the voltages to the coils of the stepper motor to move the filter. Figure 4.13 shows the filter mechanism; the dark film is the Mylar filter.

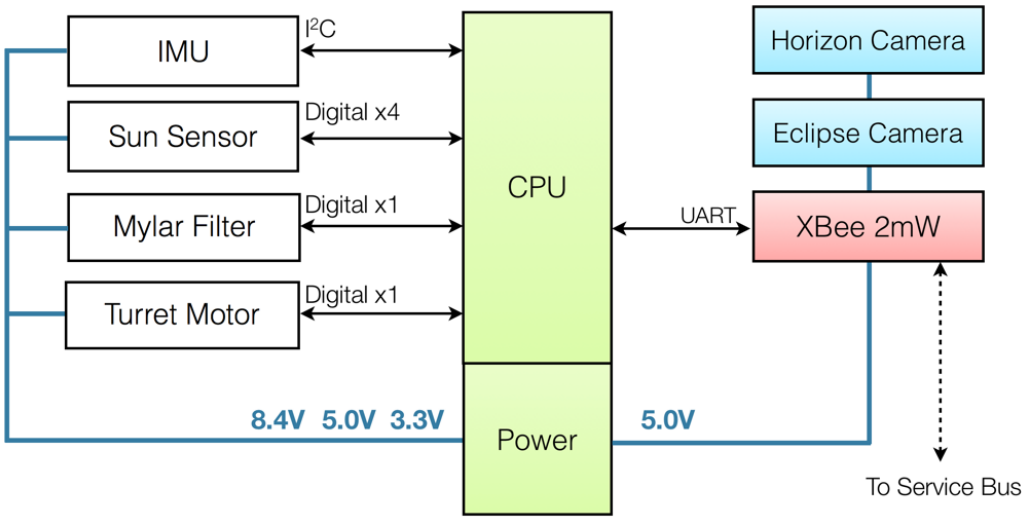

Figure 4.12 Payload Block Diagram

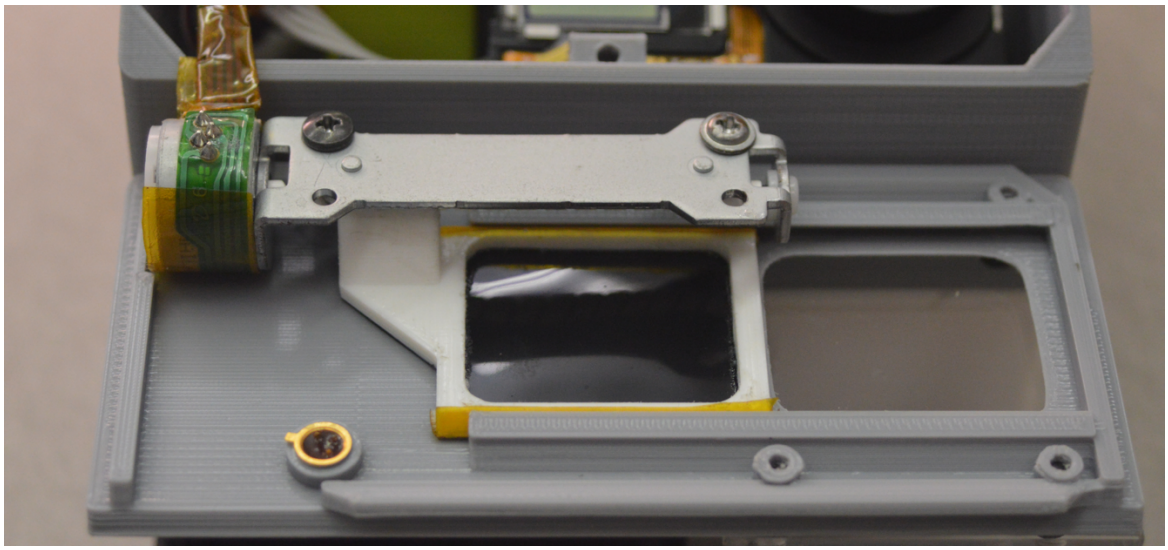

Figure 4.13 The Mylar filter mechanism.

Figure 4.14 shows a top view of the payload with its top cover removed. The horizon camera is in the bottom right. Batteries and processors are on top right. The Servo motor is hidden in the middle which drives the white gear. The lime green gear is secured on the stem. The eclipse camera is in the bottom left and the IMU is placed in top left corner. Figure 4.15 demonstrates all the components that go inside the payload.

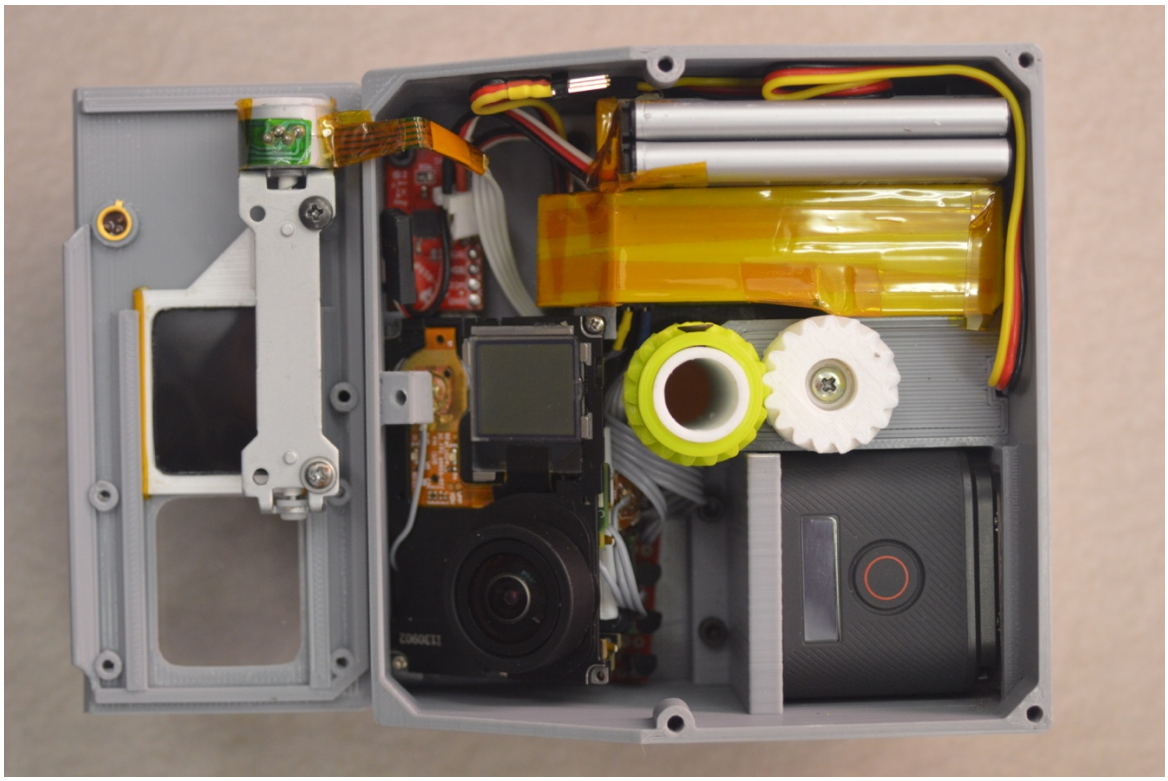

Figure 4.14 A top view of the internal components of the payload.

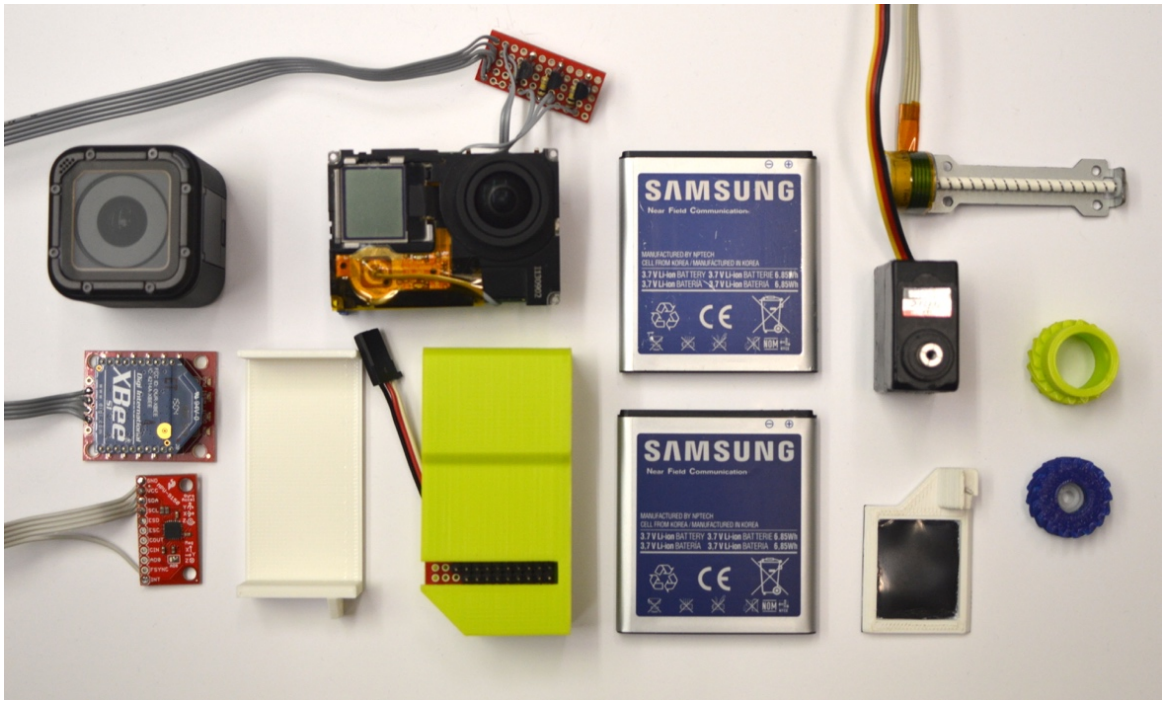

Figure 4.15 All the components inside the Payload

4.5 Flight Computer Processor Codes

The six processors inside the flight computer of the service bus and the two inside the payload will not perform any task on their own. A set of instructions must be given to them. These instructions come in the form of a code which is then uploaded to them. Each processor has a different set of tasks and therefore a different code. In this section, each code is explained and a simplified flow chart is provided.

4.5.1 Processor 1 code

The code starts with loading the I2C, SPI, Servo, SD Card Libraries. (Libraries are files that contain a separate set of code that facilitates the process of reading a type of sensor, transceiver or a peripheral on the processor). Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. Most of these variables are assigned a value of zero while constants are created with a predefined value. The setup routine is executed next and only once after the processor is powered up. The setup routine initiates the I2C communication and sets Processor 1 to master since it will be controlling the other five processors (Processors 2 – 6). UART ports one and two are activated with a baud rate of 9600 bps (bits per second). The Release Mechanism Servo is assigned a hardware pin. The processor uses the SD card and SPI library to communicate with the onboard SD card. The code checks proper connection with the card and then creates an empty text file on there where the processor will later write data to the file. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The main routine reads the data being sent from Processors (2 – 6) and stores it inside the memories created at first. It then proceeds to send data to Processors (2 and 3). The same data is also stored on the SD card in a text string

format. Then data is sent and received from the payload using the wireless module connected to Processor 1 UART two port. This is followed by the release mechanism function where it checks if cut down command has been received. If so, it will activate the servo mechanism. If not, it will then proceed with determining the processor speed which is the amount of time in milliseconds (ms) it took to run the entire main routine. This is only for optimization. Figure 4.16 shows a simplified flow chart of this code.

4.5.2 Processor 2 code

The code starts with loading the I2C and Iridium Libraries. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. The setup routine is executed next and only once after the processor is powered up. The setup routine initiates the I2C communication and sets Processor 2 to slave. UART ports one and two are activated with a baud rate of 115200 bps and 19200 bps respectively. The Iridium module is activated and a signal quality test is performed. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The main routine reads the data being received from Processor 1 and stores it inside the memories created at first. Then the data is transferred to the Iridium module using the UART two port. The data includes location and time obtained from the two GPS modules. The Iridium module has a ring pin which changes from one to zero (binary) when a message has been received by the module. When the ring pin is zero, the code initiates a function which reads the message and compares it to a look up table of functions to determine which task to activate. Figure 4.17 shows a simplified flow chart of this code.

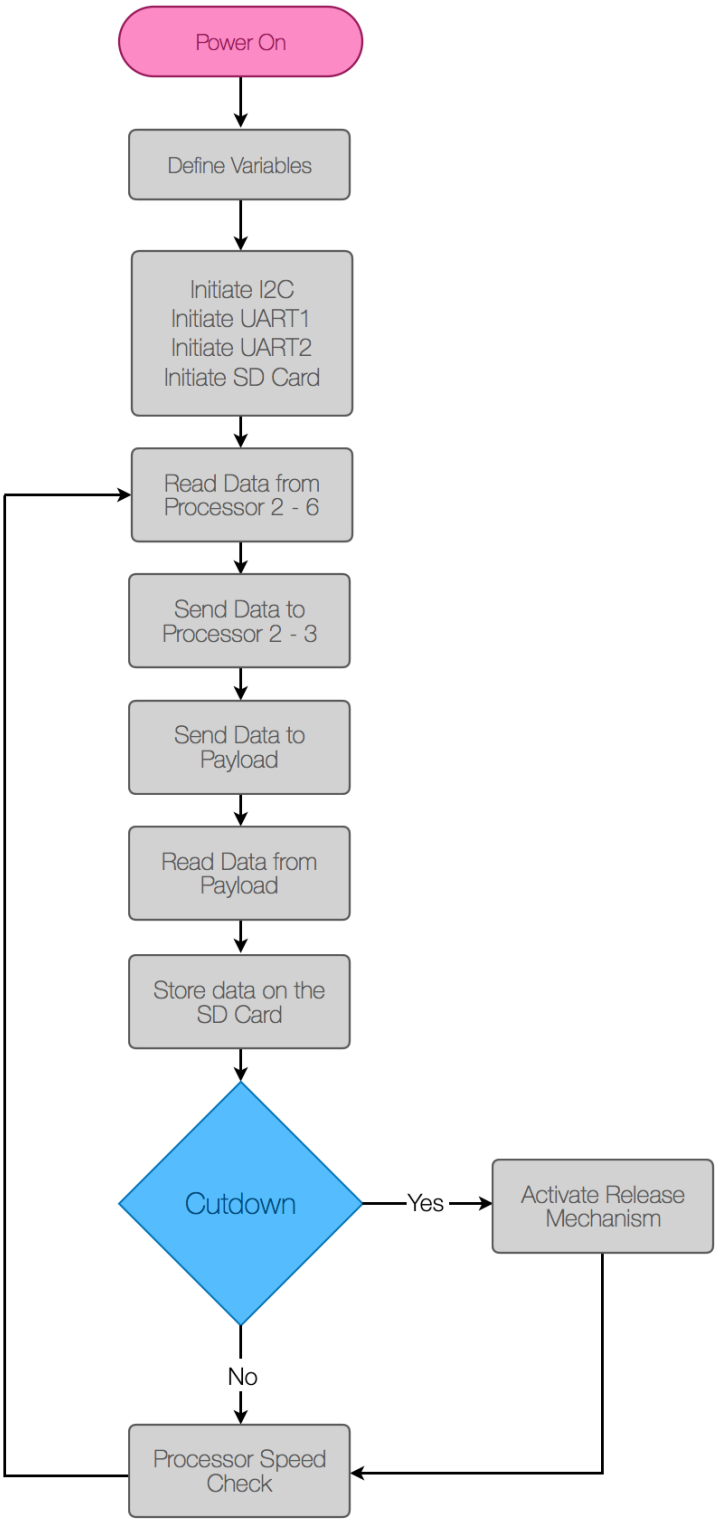

Figure 4.16 Simplified flow chart of the code for Processor 1

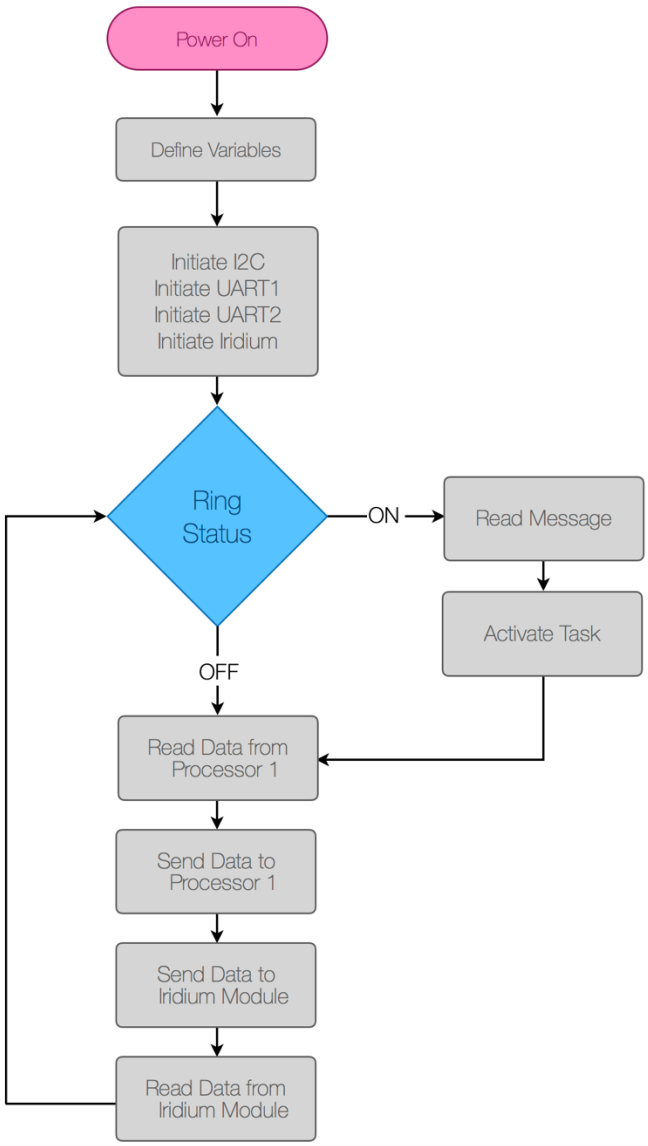

Figure 4.17 Simplified flow chart of the code for Processor 2

4.5.3 Processor 3 code

The code starts with loading the I2C Library. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 3 to slave. UART ports one and two are activated with a baud rate of 9600 bps and 115200 bps respectively. The Bluetooth module is then initiated and connected to

UART two port. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The main routine reads the data being received from Processor 1 and stores it inside the memories created at first. Then the data is sent through the Bluetooth module. A PC (Personal Computer) can connect to this Bluetooth module and view the data being sent using a serial data viewer program. The PC can also send messages to the flight computer and the code reads the message and compares it to a look up table of functions to determine which task to activate. Figure 4.18 shows a simplified flow chart of this code.

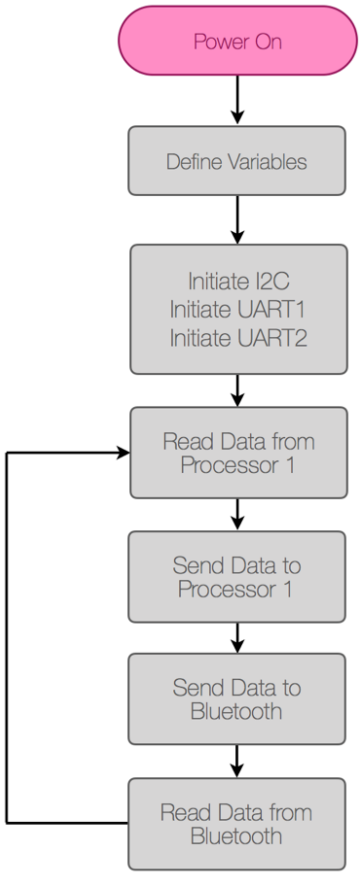

Figure 4.18 Simplified flow chart of the code for Processor 3

4.5.4 Processor 4 code

The code starts with loading the I2C Library. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 3 to slave. UART port one is activated with a baud rate of 9600 bps. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The code reads the data output of the GPS Module 1 and stores it in the variables created at the beginning. This data includes Latitude, Longitude, Altitude, Speed, Number of Satellites in View and Time. Once the data is stored, it is then sent to Processor 1. Figure 4.19 shows a simplified flow chart of this code.

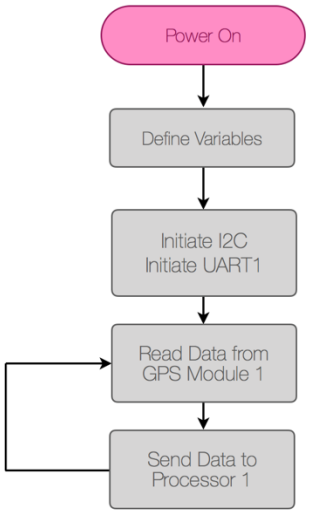

Figure 4.19 Simplified flow chart of the code for Processor 4

4.5.5 Processor 5 code

The code starts with loading the I2C Library. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. Most of these variables are assigned a value of zero while constants are created with a predefined

value. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 3 to slave. UART port one is activated with a baud rate of 9600 bps. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The code reads the data output of the GPS Module 2 and stores it in the variables created at the beginning. This data includes Latitude, Longitude, Altitude, Speed, Number of Satellites in View and Time. It then uses the data from the GPS to calculate the solar position (Zenith and Azimuth). Once the data is stored, it is then sent to Processor 1. Figure 4.20 shows a simplified flow chart of this code.

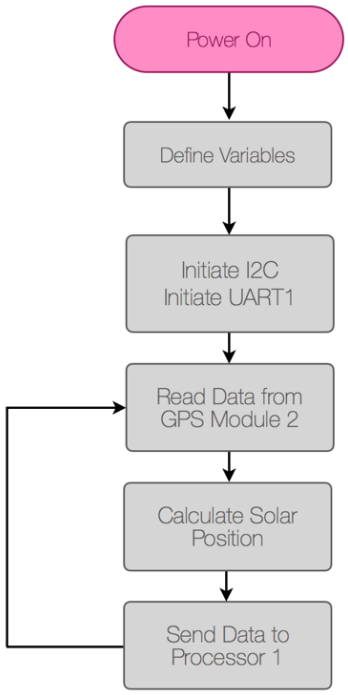

Figure 4.20 Simplified flow chart of the code for Processor 5

4.5.6 Processor 6 code

The code starts with loading the I2C Library. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. Most of these variables are assigned a value of zero while constants are created with a predefined value. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 6 to slave. UART port one is activated with a baud rate of 9600 bps. Once the setup routine is complete, the main routine is started and is executed continuously until the processor is powered down. The code starts by reading the value from the temperature sensor and compared to a predefined set value and if it is below, the heat pad is turned on. If the temperature is above, the heat pad stays off. However, if the heat pad was on and the temperature rises, then it is then turned off. Finally, the status of the heat pad and temperature sensor value is sent to Processor 1 for data recording. Figure 4.21 shows a simplified flow chart of this code.

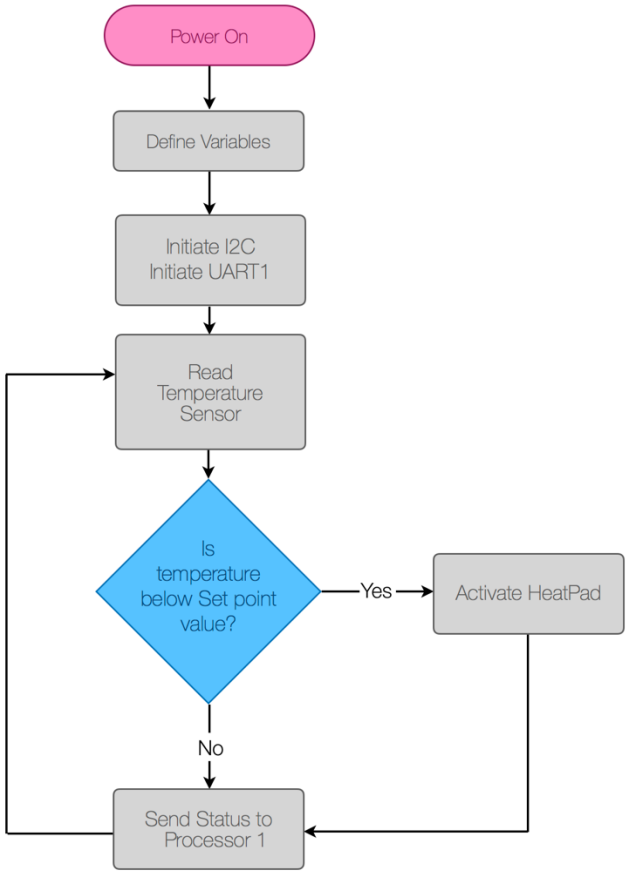

Figure 4.21 Simplified flow chart of the code for Processor 6

4.5.7 Payload Processor 1 code

The code starts with loading the I2C, Servo and MPU6050 Sensor Libraries. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 1 to master. UART port one is activated with a baud rate of 115200 bps. The MPU6050 IMU is initiated using the library. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The code starts by obtaining the Euler Angles in degrees from the IMU. It then waits for the yaw value to stabilize and then activates the servo motor. This ensures that the servo does not rotate the turret while the IMU is calibrating itself. The program

then reads the azimuth angle being received from Processor 1 of the payload and performs a PID control to position the turret in the direction of the sun. Figure 4.22 shows a simplified flow chart of this code.

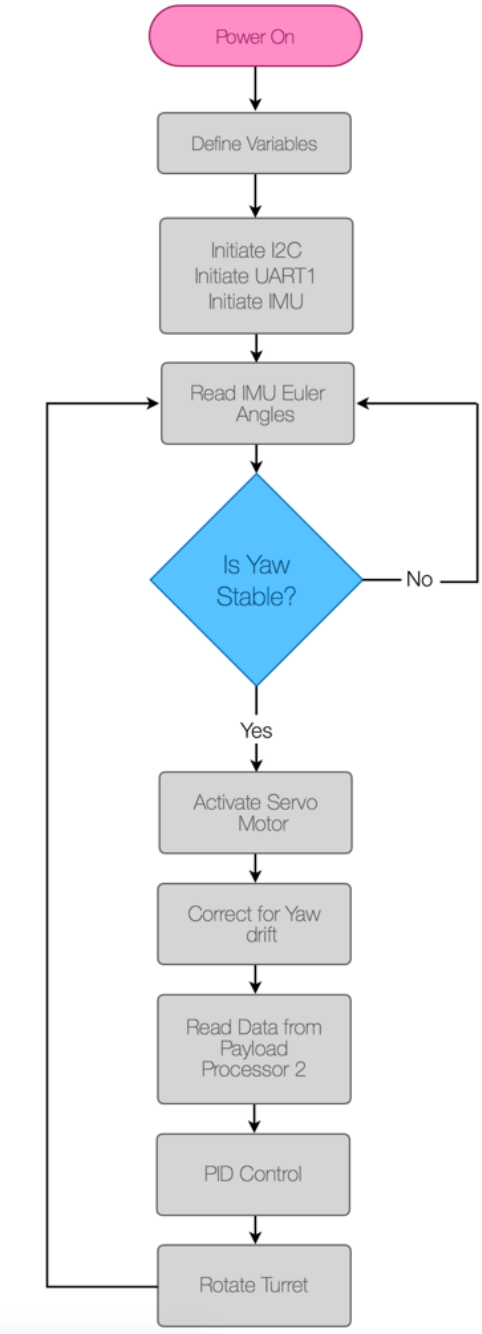

Figure 4.22 Simplified flow chart for the code for Payload Processor 1

4.5.8 Payload Processor 2 code

The code starts with loading the I2C and Stepper Libraries. Then the code proceeds with setting aside memory locations for variables and constants that are used by the program. The setup routine is executed next. The setup routine initiates the I2C communication and sets Processor 6 to slave. UART port one is activated with a baud rate of 9600 bps. Once the setup routine is completed, the main routine is started and is executed continuously until the processor is powered down. The code starts by reading the data being received from the Processor 1 in the Service Bus. It also sends data to the service bus and to the payload processor 1. It will also control the position of the Mylar filter and the status of the camera. Figure 4.23 shows a simplified flow chart of this code.

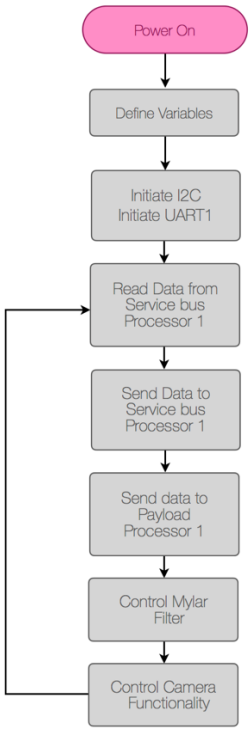

Figure 4.23 Simplified flow chart of the code for Payload Processor 2

CHAPTER V

FLIGHT RESULTS

The final version of the eclipse imager was launched on August 21st, 2017 at Russellville High School in Kentucky using a 3000-gram weather balloon. The Iridium module was transmitting data properly while on ground, however ascend speeds made it difficult to obtain a lock on satellite in order to transmit its location. The Spot tracking unit also did not provide any information during ascend. However, at 7,800 meters, both the Iridium Module and Spot unit began to transmit data to ground station. The Spot tracking also began showing its location on the map. Nearly two hours after launch, the payload was recovered without any damage. No water marks were observed inside the payload or service bus. The onboard SD card demonstrated that the flight computer performed nominally and none of the regulators shut down. Also, the design of the service bus proved to insulate the interior better than expected. The heat pads were never activated during flight and the temperature inside did not go below 30 degrees Celsius.

Although the flight was a success, during the launch, the turret would not stay locked on the sun. It just continuously spun and therefore the servo was deactivated. The culprit of the issue is still unknown since the system worked perfectly during a test that was performed the night before and three hours before launch. Deactivating the servo did not affect other systems; only the camera recordings were not stabilized.

Figure 5.1 is a picture taken at peak altitude of 20,912 meters using the Horizon camera. Figure 5.2 is a picture taken during totality by the same camera at totality.

Figure 5.1 Photo taken by Horizon Camera at peak altitude

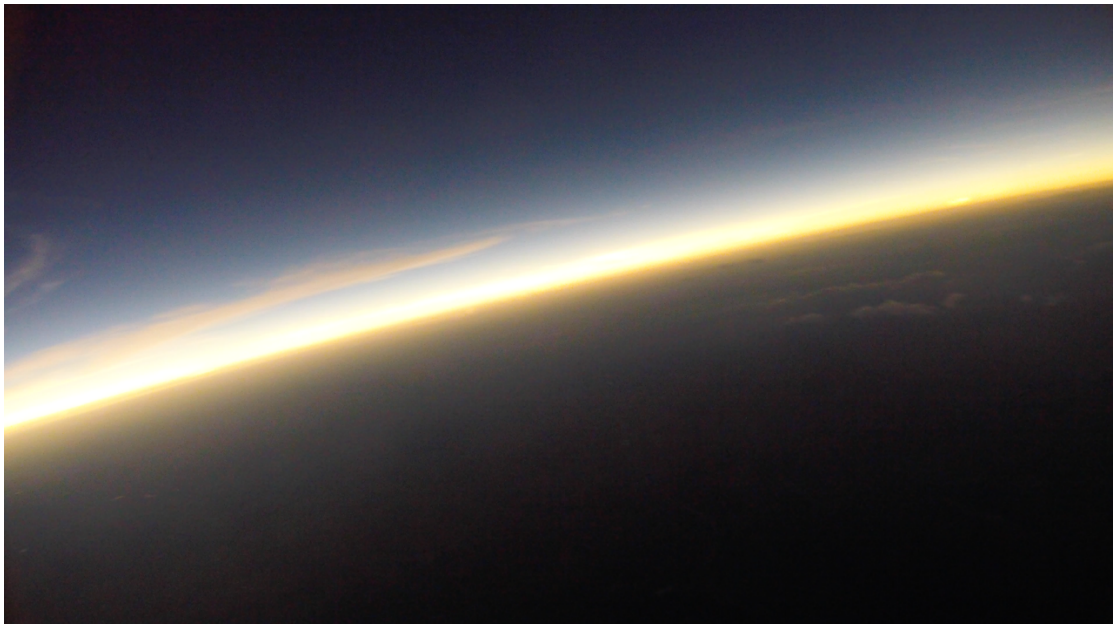

Figure 5.2 Photo taken by Horizon Camera during totality

Figure 5.3 is a picture taken by eclipse camera during partial eclipse.

Figure 5.3 Photo taken by the Eclipse Camera during partial eclipse

Figure 5.4 shows the altitude of the CubeSat over time. CubeSat was powered on at 17:50 Greenwich Mean Time (GMT). Data recording stopped at 20:56 GMT or 190 minutes after being turned on, this is when it was recovered. The CubeSat began ascending 32 minutes after being powered on and landed 130 minutes later. The altitude data is obtained from GPS module 2. At 65 min mark, the GPS stopped obtaining location. Because the ascend speeds were very high, it became difficult for the GPS to get a good lock signal. This was also experienced with the Iridium Module and Spot Tracker. However, at 108 min mark, GPS starts responding again. Figure 5.5 shows the internal temperature of the CubeSat over time.

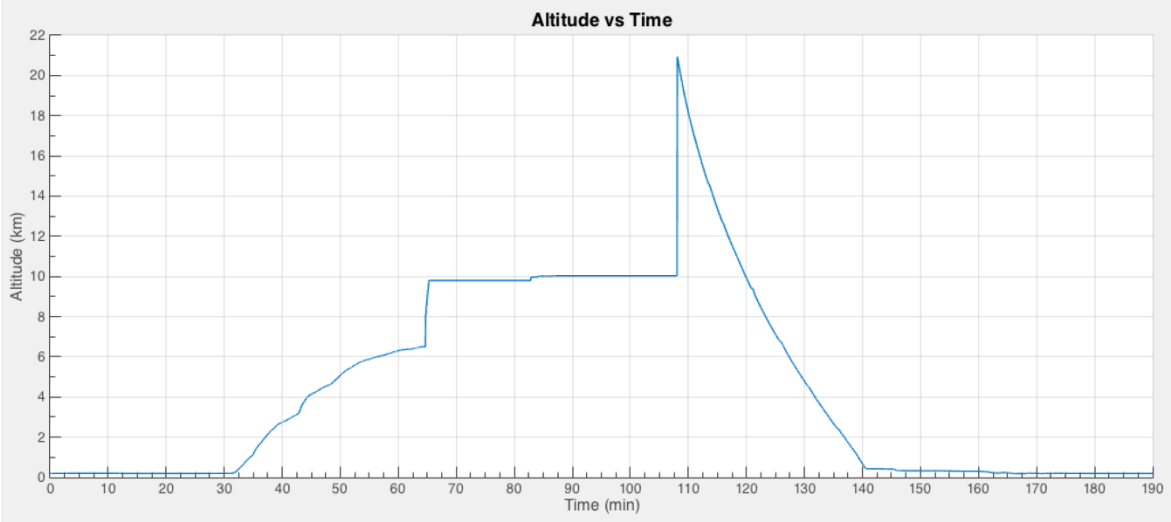

Figure 5.4 Altitude vs time graph

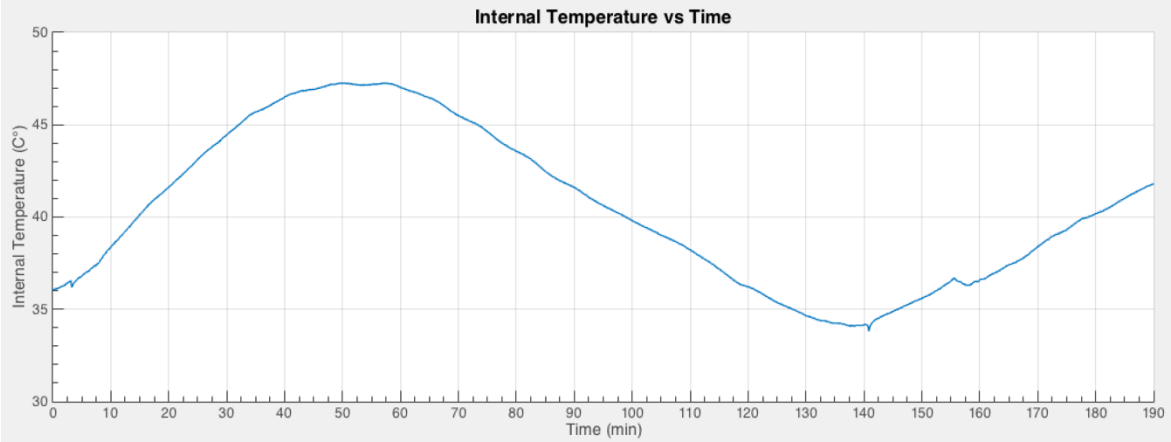

Figure 5.5 Internal temperature vs time graph

REFERENCES

[1] Total solar eclipse of Aug 21, 2017. (n.d.). *20th century*. [online] Available at: https://www.greatamericaneclipse.com/20th-century/. [Accessed 17 Mar. 2017].

[2] Lakind, S. (n.d.). *The Great American Eclipse And Its Effect On Retail Traffic*. [online] Forbes. Available at: https://www.forbes.com/sites/forbescommunicationscouncil/2017/09/12/the-great-american-eclipse-and-its-effect-on-retail-traffic/#737d64b044b0 [Accessed 20 Dec. 2017].

[3] NASA. (n.d.). *NASA's Space Grant Ballooning Project*. [online] Available at: https://eclipse2017.nasa.gov/nasas-space-grant-ballooning-project [Accessed 20 Dec. 2017].

[4] Eclipse Ballooning Project. (n.d.). *About the Eclipse Ballooning Project*. [online] Available at: http://eclipse.montana.edu/about/ [Accessed 20 Dec. 2017].

[5] NASA. (n.d.). *CubeSats Overview*. [online] Available at: https://www.nasa.gov/mission_pages/cubesats/overview [Accessed 20 Dec. 2017].

[6] CubeSat Organization. (n.d.). *6U CubeSat Design Specification*. [online] Available at: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf [Accessed 20 Dec. 2017].

[7] The CubeSat Program. (n.d.). *About - CubeSat*. [online] Available at: http://www.cubesat.org/about/ [Accessed 20 Dec. 2017].

[8] Schweber, B. (2017). *Understanding Linear Regulator Advantages*. [online] DigiKey. Available at: https://www.digikey.com/en/articles/techzone/2017/sep/understanding-

the-advantages-and-disadvantages-of-linear-regulators [Accessed 25 Jul. 2016].

[9] uA7800 Series Positive-Voltage Regulators. (2003). [ebook] Texas Instruments, pp.1 - 8. Available at: https://www.sparkfun.com/datasheets/Components/LM7805.pdf [Accessed 25 Jul. 2016].

[10] Government Publishing Office. (2018). *eCFR - Code of Federal Regulations*. [online] Available at: https://www.ecfr.gov/cgi-bin/text-idx?rgn=div5&node=14:2.0.1.3.15#se14.2.101_13 [Accessed 10 Apr. 2016].

[11] RF Wireless World. (n.d.). *UART vs SPI vs I2C*. [online] Available at: http://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html [Accessed 20 Dec. 2016].

[12] ATmega 328 Datasheet. (n.d.). [ebook] Atmel, pp.8-23. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf [Accessed 24 Nov. 2016].

[13] ATmega 2560 Datasheet. (n.d.). [ebook] Atmel, pp.8-23. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf [Accessed 24 Nov. 2016].

[14] ATmega 1284 Datasheet. (n.d.). [ebook] Atmel, pp.8-23. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42718-ATmega1284_Datasheet.pdf [Accessed 24 Nov. 2016].

[15] Howie, J. (2017). *The Board*. [online] Altium. Available at: http://www.altium.com/documentation/18.0/display/ADES/((The+Board))_AD [Accessed 23 Feb. 2018]

[16] Spakrfun. (n.d.). *RockBLOCK Mk2*. [online] Available at: https://www.sparkfun.com/products/13745 [Accessed 16 Feb. 2018].

[17] timeanddate.com. (n.d.). *The Horizontal Coordinate System*. [online] Available at: https://www.timeanddate.com/astronomy/horizontal-coordinate-system.html [Accessed 14 Feb. 2018].

[18] National Instruments. (2011). *PID Theory Explained*. [online] Available at: http://www.ni.com/white-paper/3782/en/ [Accessed 23 Jan. 2017].

APPENDIX A

MATLAB CODE FOR AZIMUTH AND ELEVATION ANGLE CALCULATION

```
clc
format long g

%The matlab can either use computer clock time or a user input time
%Computer clock time
c = clock;
Y = c(1);
M = c(2);
D = c(3);
H = c(4)+5;
m = c(5);
s = c(6);

%User Input time
H = 18;
m = 0;
s = 0;

%Location is manually entered
lat = 33.445302;
lon = -88.792988;

%This calculates the UT time in hour decimal
UT = ((H * 3600)+(m * 60) + s)/3600;

%This determines the current Julian date
%Julian dates (abbreviated JD) are simply a continuous count of days
 and
%fractions since noon Universal Time on January 1, 4713 BC
%(on the Julian calendar).

G = fix(7*(Y+fix((M+9)/12))/4);
K = fix(3*(fix((Y+(M-9)/7)/100)+1)/4);
J = fix((275*M)/9);
JD = (367*Y)- G - K + J + D + 1721028.5 + (UT/24);
fprintf('Current Julian Date is: %f \n',JD);

%Compute fractional year in radians
gamma = ((2*pi)/365)*(JD);

%Derive equation of time (deltaT) in minutes

%The equation of time represents the difference between apparent solar
%time and mean solar time, which can be as large as 16 min. It is due
%to the obliquity of the ecliptic and the elliptical form of the earth
%orbit.

deltaT = 229.18*(0.000075 + 0.001868*cos(gamma)-0.032077*sin(gamma)...
- 0.014615*(cos(gamma)) - 0.040849*(sin(gamma))^2);

%From the fractional year, we also get the solar declination (in
 radians)
```

```
%The declination is the equivalent of the latitude on the celestial
%sphere.

del = 0.006918 - 0.399912*cos(gamma) + 0.070257*sin(gamma)...
- 0.006758*cos(2*gamma) + 0.000907*sin(2*gamma) -
 0.002697*cos(3*gamma)...
+ 0.00148*sin(3*gamma);

%The time ffset (in minutes) between the UTC time and the true solar
 time
%deoends on the longitude (in degrees East) and is:

Toff = deltaT +(4*lon);

%True solar time (in minutes)

tst = (H*60) + m + (s/60) + Toff;

%The solar hour angle (in degrees)

ha = (tst/4)-180;

% For a given latitude, the hour angle and the declination, a
 horizontal
%coordinates (zenith and azimuth angles) can be determined

zenith = acosd((sind(lat)*sin(del))+(cosd(lat)*cos(del)*cosd(ha)));
azimuth = - atan2d(sind(ha)*cos(del),(cosd(ha)*sind(lat)*cos(del)-
cosd(lat)*sin(del)));

theta = acosd(-((sind(lat)*cosd(zenith))-sin(del))/
(cosd(lat)*sind(zenith)));
```

Current Julian Date is: 2458173.250000

Published with MATLAB® R2015a