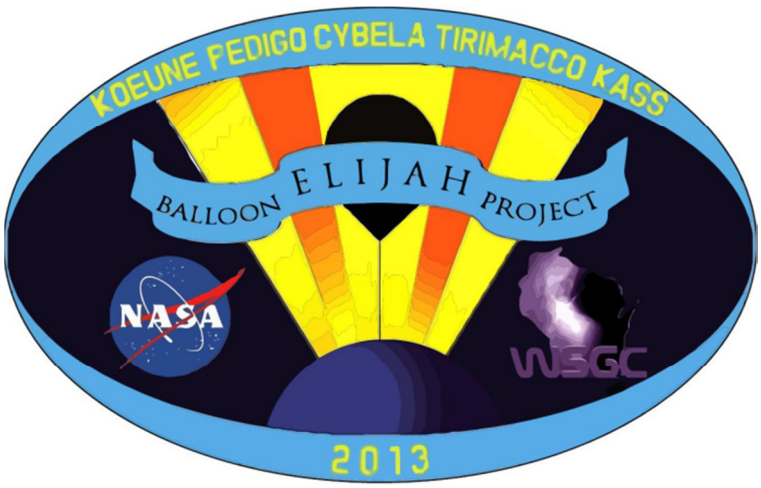
2013 Payload Team Report



2013 Elijah High Altitude Balloon Payload Team

Trent Cybela, Daniel Kass, Amber Koeune, Nathaniel Pedigo, and Alana Tirimacco

Faculty Advisor: Dr. William Farrow

Wisconsin Space Grant Consortium

August 16, 2013

Executive Summary

Over the past ten weeks, the payload team for the Elijah High Altitude Balloon Project has researched, designed, and constructed a scientific payload that would fly to the edges of space and gather data advancing the team's knowledge of Earth sciences, aerospace sciences, and aerospace engineering. The payload packages consisted of a sensor suite, telemetry system, Geiger counter, and camera. The telemetry system was able to provide a live data feed that interacted with existing networks to record data acting as a proof of the concept that a similar system would be viable and even suggested for future teams. The sensor suite was designed to gather data that would provide tracking data and finalize the battery experiment. While the tracking sensors were unsuccessful the thermistors and voltmeter reported data that confirmed that the various electronic components onboard were kept within their operating temperatures and gave data on the battery performance inflight. The Geiger counter data that was gathered was not able to be directly correlated with an altitude, it was able to be correlated with time, and as all data points received were on the ascent of the balloon it is possible to state that there was a positive correlation between radiation levels and altitude. The camera system that was included in the payload was able to record the entirety of the two flights and provided imagery from the edge of space. A large amount of the project was also devoted to the development of a rugged and reusable capsule; this task was accomplished by the creation of a layered capsule wall that insulated the capsule and gained increased rigidity due to the geometry of the capsule and an internal rib cage that supported plates for the electronics.

Table of Contents

List of Figures and Tables

Introduction

The 2013 Elijah High Altitude Balloon Project Payload Team was composed of five engineering students of varying disciplines; as such, the team was interested in creating a well-designed capsule and program that would resolve existing problems and act as a stepping stone for future teams. Previous team's reports were examined and the problems from their designs were determined; the two main problems found were a weak capsule design resulting in payload failures and a lack of redundant data recording systems

To address the total capsule failures the capsule design became a prominent focus of the team as past teams frequently had issues with the capsule cracking or failing entirely. Research performed also indicated that the capsules of previous teams may not have been well enough insulated to protect the on-board electronics from freezing, which prompted the team to include insulation into the designs of the capsule.

The team also attempted to create a redundant data collection system that would not require the recovery of an intact payload to gather data, as half of the previous teams had been unable to gather or recover any data. A telemetry system should be included in the payload that would allow the team to transmit data from the payload to ground via HAM radio.

Other portions of the payload included a camera system, a sensor suite, and a Geiger counter; these components were included to satisfy the current team's inquiries. It was decided that a camera should be included in the payload to record images of the payload, balloon, and Earth during the flight; this is a traditional package that all previous teams have included, however it was to be modified to pan during flight. It was decided that another package should consist of a series of sensors that would be useful for determining the efficacy of the thermal insulation and the tracking of the payload; this package would communicate to with the ham radio and its data would be broadcasted periodically. A Geiger counter was also to be included as a package to gather data about the relationship between altitude and radiation levels in counts per minute.

Ground Tests

Several experiments were performed prior to the balloon launch to choose between the many options for various components of the payload. These tests were based upon preliminary research and were designed to target the differences between products.

Battery Tests

Overview

The battery experiment was designed to help future high altitude balloons teams with their design process. This experiment was designed to show how sub-zero temperatures and low

pressures will affect lithium-ion batteries. Preliminary ground experiments were performed in order to decide which batteries would be ideal for the final experiment. The purpose of the tests was to see if battery voltage would be affected by the decreased temperature and pressure experienced inflight; the voltage must be maintained above certain levels in order to power the microcontrollers and sensors on board.

Dry ice was used to simulate the sub-zero conditions the batteries would experience during flight, and a vacuum bell jar was used to simulate the decreased pressure. Energizer lithium AA and Samsung Aviator cell phone batteries were to be used for the tests. AA batteries were chosen because they have been used in past balloon flights and have a high current to weight ratio. Since most previous teams have used alkaline AA batteries, it was decided that the team would test how lithium batteries would work in the extreme conditions. The Samsung Aviator battery was chosen because it was the only smart phone battery that could be removed and charged using a team member's smartphone and had a relatively high amount of power. Both batteries chosen were lithium-ion batteries because of their known properties, including high battery life, low weight, and ability to produce heat. Different brands of batteries were not tested and compared due to the time and budget constraints of the project. After preliminary experiments were conducted, the Samsung Aviator cell phone batteries were eliminated from further testing because they did not produce enough current to power the Arduino.

A digital reading of the surface temperature for the batteries was taken during the middle of the dry ice experiment and at the end of the experiment. The way the batteries reacted to the temperature and pressure changes provided information on how the batteries might perform inflight. The data was recorded on the Arduino during the experiments and extracted from the Arduino when the experiment had completed.

During the balloon flight, the voltage was recorded and graphed to track the battery voltage throughout the trip.

Procedure

The first set of batteries underwent the test in room temperature and pressure to provide a control data set. The second set was tested in a vacuum bell jar which provided a decreased pressure. The final set was tested in a cooler along with dry ice. A set of batteries consisted five Energizer AA lithium-ion batteries connected in series.

For each of the three sets of batteries, the same procedure was used. The only changes were the environment the test took place in. The set of batteries was used to power the Arduino board, with a voltage divider circuit connected to monitor the voltage of the battery set throughout the test. The Arduino requires between six and twelve volts for operation, so five AA batteries were arranged in series to meet this constraint. Due to the limited amount of storage on the Arduino,

the program was set to record data for eight hours and take a voltage reading every 30 seconds. After the program had been uploaded to the board, the Arduino setup was left to run for eight hours. For the vacuum bell jar test, the same setup, with fresh batteries, was placed in a vacuum bell jar. The next day, the test was replicated using dry ice and fresh batteries. The setup was placed in a cooler with dry ice, recording data in the sub-zero temperatures.

During flight, the battery experiment was again replicated, to yield data about battery performance when both temperature and pressure are decreased simultaneously. The voltage was monitored using the same code as the ground experiments. Five lithium AA batteries were used to power all of the electronics except for the GoPro camera and the handheld radio, both of which had self-contained batteries.

Working with the Arduino

The Arduino was used to take readings and store data, which would later be extracted for analysis. A voltage divider was created to divide the battery voltage to a value below five volts which is the maximum voltage the Arduino can handle as an input reading. The voltage dividing method takes advantage of circuit properties as the voltage was divided between two resistors, as shown in the figure below.

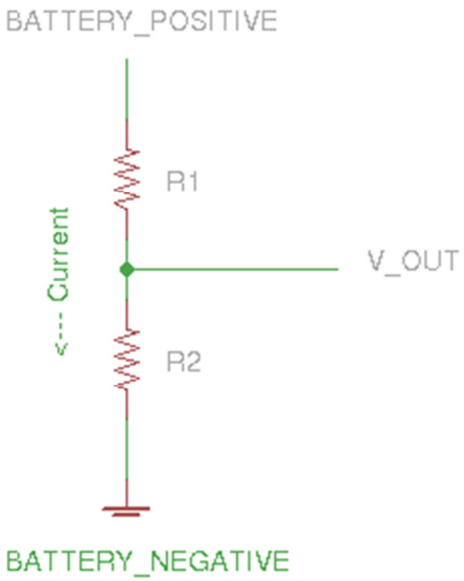


Figure 1: The Voltage Divider Circuit.

The formula used to determine how much voltage is divided between each resistor is:

$$V_{R_2} = R_1R_2 + R_2V_{in}$$

V_{R_2} is the voltage dropped across R_2 or the amount of voltage that the R_2 resistor divides. R_2 and

R_1 are resistors in Ohms and V_{in} is the battery voltage. The amount of voltage across R_2 is determined by the value of the ratio of R_1 and R_2 combined. If the resistors were the same value then the voltage drop on R_2 will be equal to R_1. As the battery voltage varies, the output voltage will vary proportionately, and this is the property of the voltage divider that was used to safely read the voltage on the batteries.

Results

The below graphs were generated from the data gathered in the various battery tests performed for the project.

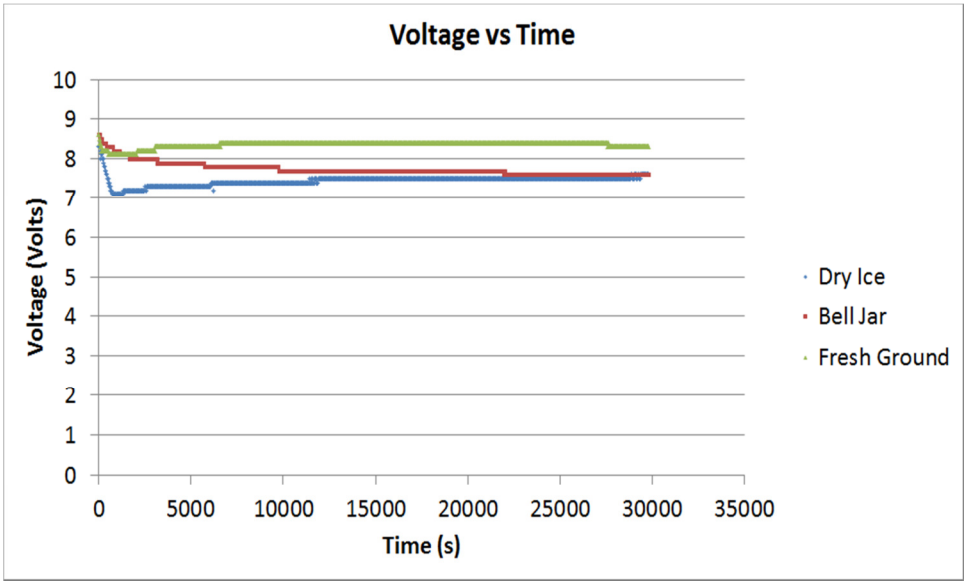


Figure 2: Voltage vs. Time Graphs for all Ground Tests.

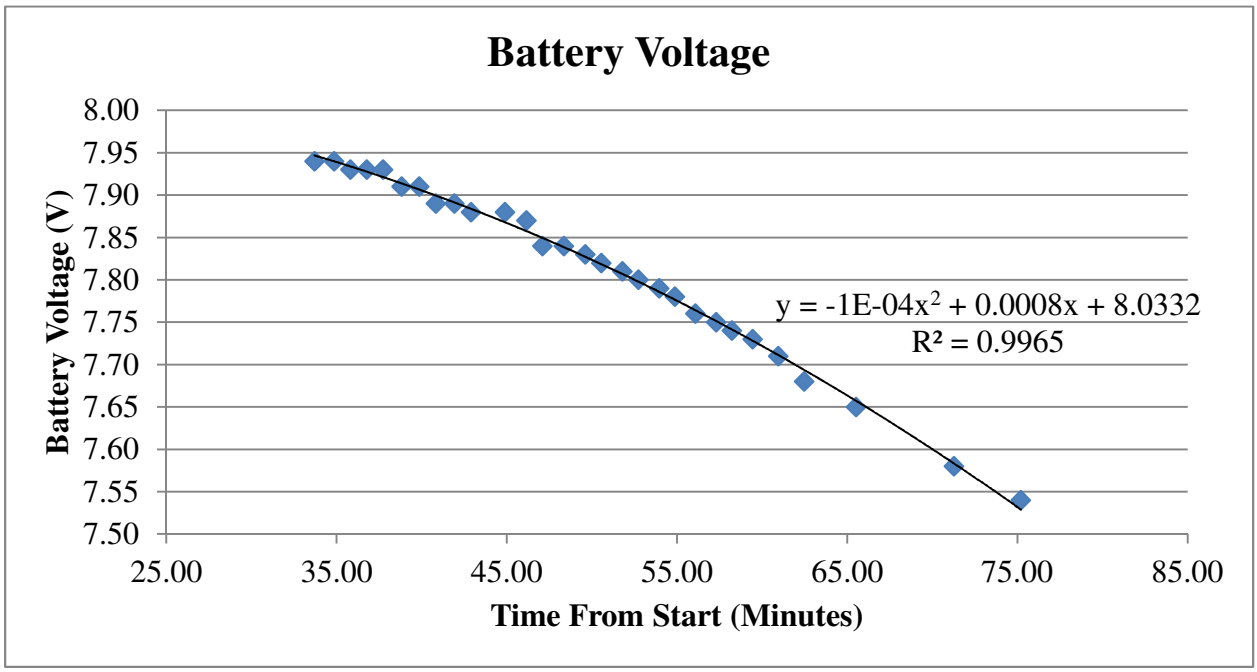


Figure 3: Battery Voltage vs. Time. The above graph shows how the battery voltage decreased during the duration of the flight. The voltage changes are modeled by a second-degree polynomial function.

Conclusion

The battery voltage was also monitored in flight. All data from the experiment and balloon flight will be documented in such a way that it will be beneficial to future teams who are looking for a way to power their payloads. The room temperature and pressure test was the control for this experiment. The data and results from the vacuum bell jar and dry ice experiments will be later compared to the control once testing is completed.

Composite Material Tests

Overview

It was decided that composite materials were to be used in the construction of the capsule body, so it was decided that several varieties of composite materials would be tested and assessed to determine the proper material for this application. Because composite cloths come in many arrangements of weight and weaving patterns, deciding on one type proved to be difficult. Twelve different types of composite materials (including two types of carbon fiber) were tested to determining what the exterior of the capsule would be laminated in. The materials were judged against one another to determine which cloth material would be strong enough, yet still lightweight enough to be used in application. The samples were provided courtesy of the Milwaukee School of Engineering, but upon selection of a particular material more stock would have to be ordered for the project.

Procedure

Mounting surfaces needed to be created for each sample. For this, 6x6" squares of extruded polystyrene (EPS) foam were cut. The EPS was chosen as substrate because it is same material which would make up the capsule body core. The twelve squares of EPS were each marked with a letter for recording purposes. With swatches of fiber cloth cut, EZ-Lam brand 30 minute cure epoxy was used to apply the material to the foam squares. For this procedure the team also made use of paintbrushes, shop towels, drop cloth, latex and nitrile gloves, safety glasses, mixing cups, acetone, and a slop bucket. Once the panels were saturated with epoxy, the samples were left to fully cure and dry overnight.

Once curing had completed, the team proceeded to investigate the finished samples. Data was recorded giving information such as texture, weight, the ease and feasibility of sanding, hardness, and thickness.

Another important factor was how difficult it was to "lay-up" the swatches onto the foam squares when applying epoxy. This was an important factor to the team as the team would need to apply the chosen composite material without the use of specialty machinery.

Conclusion

No hardness or impact testing was performed on the test samples, though tests that explored these properties may be useful to future teams. Ultimately, a four ounce (medium weight) plain weave fiberglass cloth was chosen. Although not the hardest of the test samples, its ease of application combined with low weight would provide more benefit than the heavier but stronger samples. More information regarding fiberglass materials can be found under the capsule design section.

Camera Tests

After the GoPro Hero 3 Silver Edition was chosen as the camera that would be flown on the capsule it remained to decide what settings should be used. In order to make this decision it was decided that recording time tests would be performed for each of the settings considered: 960p (960 progressive) without Protune activated, 960p with Protune activated, 1080p without Protune activated, and 1080p with Protune activated. Viewing sample videos recorded prior to testing allowed us to eliminate the 960p without Protune activated which yielded an almost incomprehensible video. Each setting configuration recorded 30 frames per second, the lowest frame rate supported by the GoPro Hero 3 series of cameras. Each test was initiated with an empty 64 GB class 10 micro SD card and a full charge on the battery. The camera would then be turned on and recording would be started immediately, the camera would then be left to run until it turned itself off due to lack of battery life or memory space. After the camera turned off the memory card would be removed from the camera and the length of the video to be recorded. Ten tests were to be performed in each setting configuration after which an average and

uncertainty in the average were determined for each setting configuration. The below table indicates the major results of the experiments; a full table can be found in Appendix 1.

GoPro Hero 3 Silver Edition Camera Recording Time Testing

Trial	Recording Time at 960p With Protune	Recording Time at 1080p Without Protune	Recording Time at 1080p With Protune
Average (hours)	3.6291	4.0382	3.7375
Uncertainty (hours)	0.0248	0.0397	0.0174

Table 1: Results of the Ground Testing of the GoPro Hero 3

Capsule Design

The capsule design – including body shape, material, and selected fasteners – was based on the fulfillment of both primary and secondary mission objectives which were decided by the team as important goals for creating a well-engineered capsule.

Objectives

The objectives set by the team could be divided into primary objectives that involved the utility of the payload itself and secondary objectives that would increase the ease of use of the payload. Primary objectives included: the launch and recover the capsule safely, with minimal damage to both the vessel and it's payload; and the use of HAM radio to successfully send/receive telemetry data from the capsule, and by doing so monitor the capsule's status in real-time. Secondary objectives included: successfully communicate with all payload sensors simultaneously; use LED indicator lights to confirm payload functionality; record payload data locally by use of microprocessor and SD card storage device; and to fly secondary payload of commemorative coins for dispersal to team members and other persons of interest.

Combining both the team's objectives for the capsule with constraints assigned in section 101 of FAA regulations, work on the design could begin. As part of the primary condition of launching and returning the capsule safely, an area of major importance was the determination of how much impact force the capsule would have to endure both during flight and landing. Given that the capsule would need to return to earth in an airworthy condition, it was decided that the strength of the capsule body would need to be great enough to support an unaided fall from apogee – meaning a fall without parachutes. An algebraic simplification was used to approximate terminal velocity:

$$v_t = \sqrt{\frac{2mg}{\rho A C_d}}$$

Values considered for this calculation included mass, gravity, fluid density, surface area of capsule, and the relating coefficient of drag. Using this methodology, terminal velocity was

found to be approximately 37 miles per hour. Using the same methodology, maximum velocity for a parachute-aided fall was found to be 12 mph. The impulse-momentum equations then gave an approximation for max impact force:

$$m_1 v_1 + \int_0^t F dt = m_2 v_2$$

Assuming an impact time less than a tenth of one second (t0.1sec) max impact force was estimated at 102 lbs. It was clear from these approximations that if the team was to protect both the capsule and payload under the most drastic conditions, new avenues of design, materials selection, and fabrication would have to be explored.

Early Concepts

Choosing a strong geometric shape was a key factor for the development a payload capsule which would be able to effectively withstand and absorb the stresses predicted to occur during flight and landing. Taking inspiration from various sources, a spherical capsule was initially decided upon for its ability to more effectively distribute compressive forces relative to designs which included edges, and therefore more stress risers. The added benefit to a sphere is its ability to function well as a pressure vessel. Early discussions also included the possibility of creating a hermetic vessel – a capsule which could cope with the extremely low external pressures while containing a ground atmosphere within. The justification behind such a design was the added insulation benefits of retaining air within the capsule body. By doing so the team believed that the decrease of internal temperature could be reduced, thereby protecting the power supply from voltage drops due to low temperatures (which were expected to reach -50º C). This “stretch goal” was dropped however, so that focus could be put on more immediate project objectives.

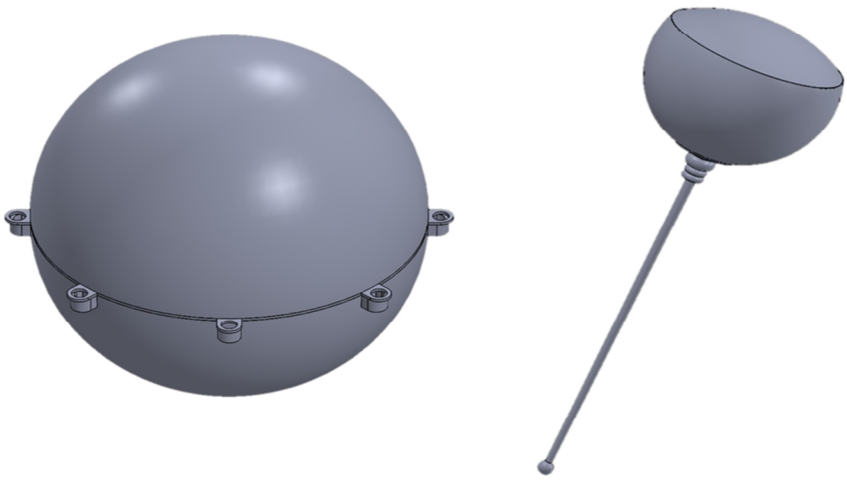


Figure 4: Early Capsule Design Concepts

After several revisions a final design was set on which we believed would be conducive to the accomplishment of the specified objectives. The final design was composed of a hemispherical body with a convex lid to increase the volume of the capsule.

Capsule Sections

The capsule body can be divided into exterior, core, and interior layers. The core, or base material, consisted of extruded polystyrene (EPS) foam. This pink, rigid foam was purchased in thick, 4x8 foot sheets, and though many methods for shaping this foam exist it was decided that a "build up" method for construction was most suitable. By this method of construction, multiple sections of the board would be cut using a CNC (computer numerical control) router. Once the sections were cut to size they would be stacked and bonded together to create the full shape. The EPS proved to be an excellent core material, given its low thermal conductivity, high strength to weight ratio, availability, and its machinability. A prototype, scaled-down version of the core was built first as a proof of concept for the design. Access to the CNC router was gained later through a partnership with the Milwaukee Makerspace. With the help of Makerspace we were able to cut the foam to an amazing level of accuracy.

Unlike earlier iterations, the final body profile was not a full sphere. Reasoning behind this was the lack of necessity for so much internal volume. Also from a practical standpoint, having a full sphere of electronics - with components residing both in the top and bottom hemispheres of the capsule - would have been very difficult to pack together and assemble on launch day. The bowl approach was much easier to handle, and a convex lid was added to give a slightly larger amount of headroom for any of the taller electronics or related wiring. The mating surface between body

and lid was beveled in order to ensure a snug, no-slip, and well centered fit between these two components.

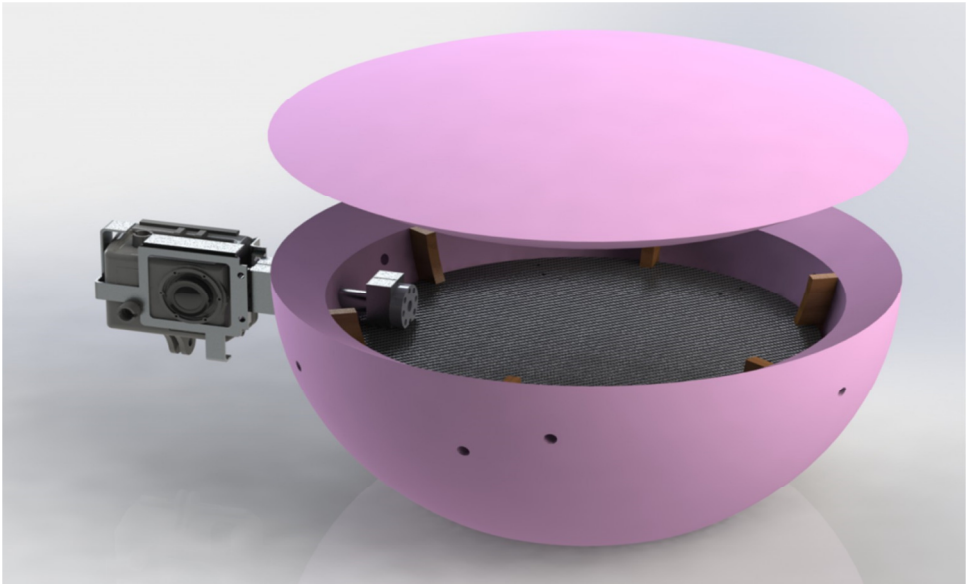


Figure 5: Capsule Design

As stated previously, the foam core touted a high strength to weight ratio, and because it's initial purpose was intended for home insulation it was believed that it would serve well as a non-conductive thermal barrier (R value approximately 5) between the sensitive electronics and the harsh -50^0F environment outside the capsule. Despite these benefits however, the team felt that more could be done to make the body impact resistant. This was due mainly to conversations with past Elijah team members, who found that EPS alone was impressionable and could crack easier than expected. Adding an additional layer of rigid strength was deemed necessary, and for this purpose composite materials were explored for use as an exterior layer.

A combination of 4oz S-2 unidirectional plain weave fiberglass and epoxy resin was used to laminate the exterior surface of the foam capsule. After a series of testing involving 12 different samples of fiberglass material - each with its own weave and weight characteristics - the 4oz fiber was decided upon for its ability to provide appropriate hardness and scratch resistance to the exterior while adding very minimal weight to the capsule. Heavier weaved fibers were effective as well, but were also more difficult to lay up over the curved capsule body. A layer of reflective Mylar and foil tape was added to surround the fiberglass hull. This wrapping was used for the purpose of retaining heat inside the capsule, which otherwise would have been leaked gradually to the surrounding atmosphere in the form of infrared radiation.

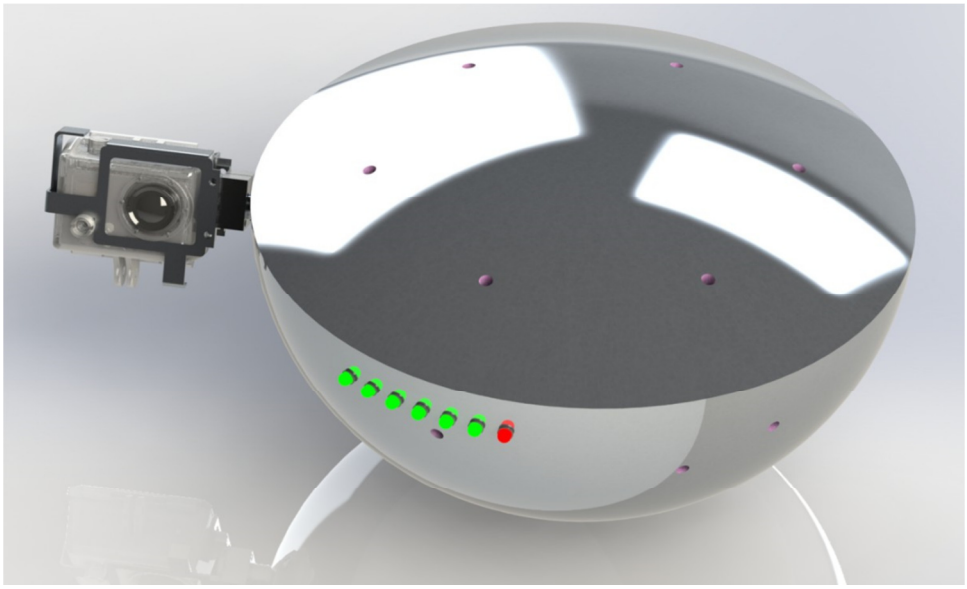


Figure 6: Capsule Rendering Shown with Aluminized Mylar Layer

A very important design consideration made was the method of mounting the microcontroller, radio, and other peripheral components to the interior of the capsule. Initial designs called for either heavy threaded studs to suspend mounting platforms within the body, or alternatively to machine ledges into the foam itself to provide a seat for those same platforms to rest. Either for reasons relating to weight restrictions or to concerns regarding the robustness of the design, an alternative method of mounting was developed to provide two stable mounting platforms supported by a hemispherical "ribcage-like" framework. In the final design, the actual electronic components were mounted to circular plates of rigid fiberglass FR4 board (similar to G10 board). These circular plates were cut using a similar CNC machine to that used for the EPS foam core. The disks were not cut as perfect circles, but instead included holes for threading screws as well as carefully placed slots which would allow the disks to conform around the six-rib array. The array - which would support the payload - in combination with Thinsulate fabric comprised the inner layer of the capsule. Basswood which was used for the 6 vertical ribs, was chosen because of its high strength-to-weight ratio. While balsa wood performs better in this category, higher porosity and lower machinability of the material was a serious concern. The rings which bound together the rib array were Baltic birch plywood. Because these rings created the ledge onto which the electronics plates (avionics plates, or AV-plates) would be mounted, a less compressible/higher strength wood was needed which could more readily accept fastening hardware.

Figure 7: Ribcage Design

The nine components which comprised the finished ribcage were cut by use of a 60 Watt CNC laser. The cutter provided a precision which our team would not have been able to replicate by use of hand tools. The assembly was completed using a combination of wood glue and epoxy adhesives, and the wood was then further treated with a sealant to prevent moisture from entering the material. By use of additional epoxy and steel u-pins, the frame was then lowered and anchored into the interior of the capsule. The telemetry radio and power supply were attached to the lower AV plate, while the remaining components attached to the top. Though the radio was the largest of our airborne components, and therefore not the best candidate for the smaller of the two compartments, it was believed that by placing it low we would ensure that the center of gravity would remain below the midline of the capsule. This would be an important consideration in the event of a parachute failure, where the capsule would impact the ground in a free fall condition.

Electronics were fastened to the AV plates by use of machine screws, washers, nylon standoffs, zip ties, and nuts. T-nuts were pressed into the birch wood seating rings to receive the machine screws from the AV plates. Out-holes were drilled through the capsule walls and lid to provide exits for externally mounted components (including the camera, temperature, and altitude sensors) as well as the Dacron cord which attaches to the rest of the flight train.

In past years many teams have experienced failures related to capsule detachment from the rest of the flight train. To avoid capsule separation, six Dacron (polyester) leader cords were used in the connection to the launch train. Each of these six cords were routed through all three sections of the capsule: internal, external, and core. If the cords were to “zipper” through one or more of the capsule segments, the redundancy in the cord routing would ensure the best possibility of retaining a solid connection between those two components. Another creative feature was the

design of brass, tubular fittings through which the Dacron is routed. These flared metal components give the cord a smooth surface to ride against when leaving the capsule body, thereby eliminating much of the worry associated with fatigue failure of the cord. A combination of high strength swivels and quick-links (ranging from 150-1500 lb. test) were used to connect all six leaders to the upper portions of the launch train.

Materials

Research was done to determine the best materials available to insulate and protect the payload during flight. The below listed materials were chosen by balancing cost and availability with beneficial materials properties and ease of fabrication.

Woven fiberglass is a material consisting of weaved and interlocking strands of glass (GFK glass fiber) reinforced by an epoxy resin. The resin acts as both an adhesive for whatever surface the fiberglass is laminated to, as well as a binder for the glass fibers. Fiberglass is a popular solution for many applications where having a high strength to low weight ratio is critical, such as aerospace engineering pursuits. Common uses for fiberglass include glider aircraft, boats, piping, and body/shell structure for various high stress applications.

Aluminized Mylar, the material used in space blankets, was used to reflect back radiated heat from inside the capsule to prevent radiated heat from escaping. The Aluminized Mylar was very flexible and easily punctured and therefore required support to be an effective component of the capsule, so it was bonded to the exterior of the fiberglass shell.

In order to further address the temperature concerns of the capsule air trapping insulators were also used, these insulators were extruded polystyrene foam and Thinsulate. Air pockets are essential to the function of these insulators since they both prevent warm air from escaping the capsule. The R value is a measure of thermal resistance, the higher the R value the greater the material resists the flow of heat from one side of the material to the other. Thermal conductivity is a measure of how well a material conducts heat. If a material has a low thermal conductivity then it does not easily allow for heat to travel from one side to the other and in turn creating a better insulator.

Polystyrene, also known as extruded polystyrene (EPS), is a commonly used insulation foam that works by trapping air inside small pockets formed during its manufacturing process. EPS is a rigid, yet easily breakable housing insulation foam, which can typically be purchased as large board/panels from a hardware or home improvement store. EPS has been used many times in the past for ballooning capsules because it is relatively easy cut and form. Further, because it is lightweight, has low thermal conductivity, and is rigid, it is a very cost effective “all in one” solution for capsule construction - provided the capsule is not required to endure high stresses during flight.

Thinsulate is a relatively new product developed by 3M. It is an extremely lightweight insulation fabric commonly used in winter jackets, hats, and gloves. This fabric is known for its ability to breathe, thus in order to prevent the air from leaving the microfibers, a layer of material will be placed on each side of the fabric. Fabric from a t-shirt was used as the inside layer and polystyrene on the outer side layer attached to the Thinsulate. Thinsulate has a high R value and low thermal conductivity and worked very well for our purposes. It added very little weight to the capsule and was easy to cut and sew into gores. 3M 79 spray foam worked well as an adhesive between the Thinsulate and polystyrene.

Packages

Packages are those components of the scientific payload that were not structural and were used to gather, process, or transmit data; the packages included the Arduino Mega 2560 (which process and controlled various sensors and systems), the telemetry system, the Geiger counter, the sensor suite and the camera. The full code that controlled the sensors and electronics can be viewed in Appendix 2.

Telemetry

Purpose

One of the objectives the team had for the launch was to receive a live data feed from all of the sensors aboard the payload. To achieve this objective, one of our team members needed to obtain an amateur radio operator's license (HAM license) to allow the use of the frequency band necessary for this type of data transfer. Dan Kass was able to pass the test and receive this license, KC9ZGW.

System Components

The next step was to determine how the data would actually be sent. With the help of another HAM Radio Operator, it was determined that the use of the Automatic Packet Reporting System (APRS) should be used on the frequency 144.390 MHz. 144.390 MHz is the standard ARPS frequency, meaning that on this frequency there are many repeaters. More repeaters will increase the likelihood of keeping a strong signal at great distances. One other advantage of using the standard APRS frequency was the digipeter which is similar to a repeater but instead of rebroadcasting the single it uploads it to the internet for access anywhere in the world. This was how data was eventually acquired for our second launch.

The radio that chosen for the balloon was the Kenwood TH-D7A which was chosen because the WSGC already owned two and that it included a built in Terminal Node Controller (TNC). The TNC is needed to convert text into tones that the radio can transmit

System Communications

The communication done between the radio and the Arduino was done using Serial Communication. The Arduino mimicked the commands that can be sent from a personal computer that can control the radio. By doing it that way the communications was relatively simple. The Arduino would perform a Serial print with the command and parameterize the information as the text. An example of the code is:

```
Serial.print("BT This is an Arduino on the radio");
```

The "BT" is the command for the radio that stands for Beacon Text and the text "This is an Arduino on the radio" is the text that the radio will broadcast at an interval of time that is set with other command.

There was originally going to be a radio on the ground connected to a laptop to receive the data packets coming in, but due to time constraints there was no time to create an interface to display the information nicely so it was determined to use the digipeters and the internet to collect the data from the balloon. The information was accessed from http://aprs.fi; a Packet of information looked like this:

```
2013-08-08, 11:34:23, CDT:KC9ZGW-7>GPS,qAR,N9QIP:
2021969,35.05,20.77,7.94,2.00,0.00,0.00,0.00000,0.00000,0.00,0.00,0
```

Everything after the colon was the information the Arduino sent, which is the data from the sensors in a pre-set order.

Results

The receiving radio in the chase car was plagued with slight issues at the connection points between the antenna and radio, and radio and computer; information was recovered from a website that receives and reports information sent over the APRS frequencies. With the collection of data from the radio we proved that information can be sent from an Arduino in the High Altitude Balloon and received on the ground before the balloon lands.

Geiger Counter

Purpose

A Geiger counter was included as one of the sensors to be included in the payload to observe the galactic cosmic rays (GCR) and solar radiation that are constantly bombarding the Earth. The Geiger counter would measure the amount of radiation in counts per minute, from which a dosage equivalency can be estimated and the effects that being exposed to those levels of radiation can be generalized. To most accurately fulfill this purpose the Geiger-Muller is to be mounted so that it receives no shielding form the capsule.

Information Concerning Radiation

Two main types of radiation are constantly bombarding the Earth: galactic cosmic rays and solar rays. Both of these types of radiation are highly variable; constantly changing and interacting with the upper atmosphere. The solar radiation levels are relatively stable compared to GCR and can be predicted with a fair degree of accuracy while the GCR is entirely unpredictable. Most of the solar radiation is gamma rays however alpha and beta radiation are also constantly being thrown off of the sun while GCR is mostly alpha and beta rays with some gamma rays also being present. The Earth's magnetosphere tends to deflect most of the alpha and beta radiation towards the poles, as both alpha and beta rays are charged particles, creating the borealis. The gamma radiation can pass through the magnetosphere and interact with the atmosphere more extensively, often penetrating the atmosphere until the atmosphere becomes dense enough to dissipate the radiation. The interaction of the incoming radiation and the atmosphere produces secondary radiation, typically in the form of positrons, muons, pi mesons (pions), alpha rays, and beta rays. The positrons, muons, and pions are typically very reactive and have very little penetrating power, consequently these types of radiation interact with the atmosphere and can generate different types of radiation, i.e. alpha and beta rays.

Justification of Geiger Counter

Interfacing with Arduino

Many options exist for a digital Geiger counter however relatively few of these are open source, meaning that we would be able to access codes for them, and only two of these were compatible with an Arduino board. The Geiger counter manufactured by Sparkfun Electronics was able to be integrated into the main capsule system as it could, by use of the serial peripheral interface (SPI), communicate with and be controlled by the Arduino that would be controlling all sensors in flight. The Geiger counter manufactured by Libellium also could communicate with an Arduino board via the SPI but this board was encumbered by a large amount of features that detracted from its usefulness for recording data, chief amongst these was a LCD display board.

Geiger – Muller Tube Specifications

The Geiger counter chosen included a more sensitive tube than was offered by other manufacturers, meaning that it would detect more of the radiation incident upon the tube. The original intent of the Geiger counter was to observe the radiation levels of the upper atmosphere where most of the radiation is in the form of alpha, beta, and gamma rays. For this reason it was important to use Geiger-Muller tube that could measure each of these forms of radiation, and the only Geiger counter that featured such a tube and was Arduino compatible was the one manufactured by Sparkfun Electronics, the one manufactured by Libellium only measured gamma and beta rays.

Expected Results

The GCR and solar radiation levels are expected to rise significantly with altitude as the radiation will not have had as much of a chance to interact with the air. The secondary radiation levels are likely to peak at approximately 45,000 ft. as this is the altitude at which the atmosphere is sufficiently thick enough to interact heavily with incoming radiation, while not yet being thick enough to dissipate most of the energy. As a consequence of the design changes that necessitated that the window of the Geiger-Muller tube not be exposed to the open air it is likely that most of the secondary radiation will be blocked by the capsule and so not be read by the Geiger counter. Because all alpha and most of the beta particles will be blocked by the capsule it is likely that low levels of radiation will be read until approximately 45,000 ft. after which the radiation levels should rise significantly and have a positive correlation with altitude.

Results

The first flight provided no data on the radiation levels at any altitude while the second flight provided data from only a portion of the flight. The second flight data did indicate that the radiation increased with altitude as all data points recorded were taken prior to the balloon reaching its apogee and increased exponentially with time as indicated in the figure below.

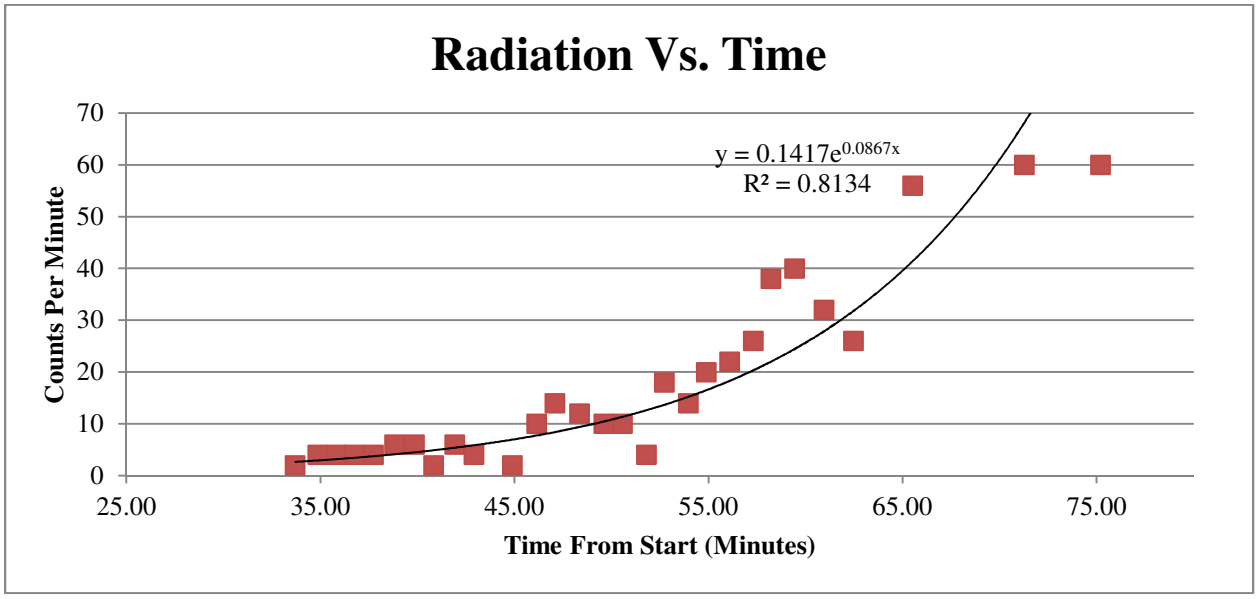


Figure 8: Gamma Radiation Vs. Time

The exponential line in the figure above is the best fit line for a plot of the radiation versus the time inflight generated by Microsoft Excel. The R^2 value reported indicates that there is a fairly strong correlation between the radiation levels and time inflight; as the times that the radiation levels were reported the balloon was still on its ascent to its apogee this also indicates that there is a correlation between the radiation levels and altitude.

Sensor Suite

Altimeter

Purpose

An Altimeter was included in the sensor suite to give a more accurate figure for the altitude of the balloon as GPS systems tend to have poor accuracy in altitude calculations. This information would be used as a reference point for other data gathered so that the data could be correlated with altitude.

Altitude Calculations

The altimeter used both a temperature sensor and a barometric sensor to calculate the altitude, by use of known relationships between temperature and pressure in the atmosphere. The sensor used had been tested to be accurate to an altitude of 120,000 ft. and was thus well suited for the purposes of the team.

Results

The altimeter was not used successfully, likely due to a wiring issue, and no data was recorded during the flight. As no data was recorded inflight it was not possible to fulfill the purposes of the altimeter, however approximations could be made using the altitudes reported by the primary and secondary tracking payloads.

Thermistor

Purpose

Thermistors were included in the sensor suite because this would allow for the evaluation of the efficacy of the insulation that was used in the construction of the capsule. The thermistors also allowed for the team to determine whether or not other sensors and electronics were working properly as most of the electronics were rated to -40^{o} C while others are rated to -20^{o} C. The thermistors were also to be used to verify the altitudes reported by the altimeter, which would also report the temperature and pressure that it used to calculate the altitude this would be useful as inaccurate temperature readings in the altimeter would result in inaccurate altitude data.

Placement

One thermistor was placed on the outside of the payload to record the external temperature as the payload flew. The other thermistor was placed inside of the capsule to record the internal temperature of the capsule. From these two data sets it was possible to evaluate the efficacy of the insulation of the capsule.

Process

Thermistors use the resistance of a piece of metal to determine the ambient temperature of its surroundings. The resistance of the metal varies as the temperature changes, as the metal

expands; the resistance of the thermistor is recorded and the temperature then calculated from this information relatively simply.

Results

Both thermistors worked well in flight, reporting temperatures that were within the temperature range anticipated during the flight, with a minimum external temperature of -50° C. The data gathered by the thermistors also indicated that the insulation included in the capsule was highly effective as at an external temperature of -50° C the internal temperature was 4.5° C; the temperature gap between internal and external temperatures was evident at all points in the recorded data, with the external temperature decreasing at a greater rate than the internal temperatures as indicated by the figure below.

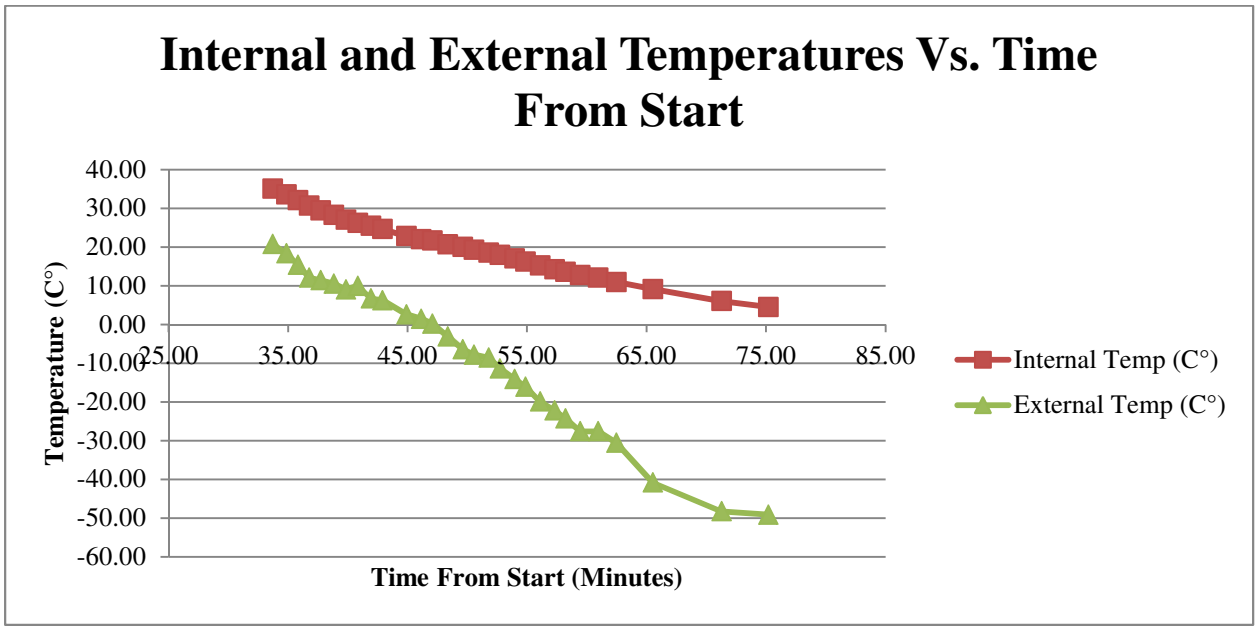


Figure 9: Graph Depicting the Effectiveness of the Capsule Insulation

Global Positioning System

Purpose

A GPS or Global Positioning System device was chosen to go into the payload to get information on latitude, longitude, altitude, speed, and the number of satellites it was connected to. This GPS was a further redundancy on the tracking systems for the payload train and was intended to give an alternate method of accessing the GPS location of the payload inflight.

Accuracy

The GPS was fairly accurate for latitude and longitude readings but was fairly inn accurate in altitude readings which necessitated the use of an altimeter to gain altitude data.

Results

Unfortunately while testing the device it stopped working the manufacture thought that the device had burnt out, and this was too close to launch day to get a replacement in so the device was not flown. Latitude, Longitude, and Altitude, however inaccurate, data was however collected by both the main and secondary tracking payloads that the launch team was responsible for.

Camera

The team decided that one of the items to be included was some form of camera to record imagery of the Earth during the flight; this portion payload was later expanded to incorporate a mount for the camera that would pan the camera in flight, reducing the number of shots that will contain essentially the same view.

Camera Selection

An initial decision was made to focus our camera searches on action cameras (action cams) which are by necessity of purpose rugged, lightweight cameras that are typically small in size.

Camera Selected

The camera that was selected is a GoPro Hero 3 Silver Edition, this action cam is renowned for its durability, and is easily modified to better suit our specific needs. We will be using the camera with the Battery BacPac which essentially doubles the battery life of the camera and the water proof housing with Battery BacPac back door, an expansion of the GoPro water proof housing that allows for the use of the Battery BacPac.

Justification

The camera selection process was done by the research of a myriad of action cam and action camera manufactures. It was found that a vast majority of the action cams available had poor image quality and battery life, leaving relatively few possibilities to pursue more closely. Of the remaining cameras, the forerunners of which were the GoPro Hero 3 line of action cams and the JVC GC-XA1 ADIXXION, all of which could be modified to increase battery life to the desired level, approximately three hours. The JVC ADIXXION and the White and Silver editions of the GoPro Hero 3 action cams were comparable in price while the Black Edition of the GoPro Hero 3 was considerably above this price range. The GoPro Hero 3 line of action cams have an optional auxiliary battery which doubles the battery life of the camera in almost all modes and attaches securely to the back of the camera. The JVC ADIXXION action cam had no options for extending battery life; however it could be modified to charge inflight and was fairly shock resistant. Due to the ease with which the action cams could be modified we decided to pursue the GoPro Hero 3 line of action cams, comparing the different editions looking for a balance of battery life, image quality, and price. The White Edition was fairly basic and did not include high quality software, so its image quality was often poor. The Black Edition was overwrought

with features such as a Wi-Fi remote that drastically reduced battery life and would be unusable for our purposes. The Silver Edition included better software that significantly increases the image quality of the camera; and while this camera too had many superfluous features, they could be suppressed to increase the battery life of the camera.

Camera Settings

Settings Selected

The GoPro Hero 3 Silver Edition was flown in the 1080p wide frame mode with Protune turned on, which allowed for high definition (HD) recording with minimal compression artifacts and an acceptable battery life.

Protune

Protune is a proprietary software package manufactured by GoPro for its action cams that reduces the number and frequency of compression artifacts by altering the algorithm by which the images are saved. The software that is standard on the GoPro Hero 3 line of action cams is not fast enough to process and compress HD footage at 30+ frames per second which leads to large compression artifacts in the video and ghosting, when images are placed in the wrong frame.

Ground Test Results

Ground testing indicated that all modes tested had a battery life that would last the expected three hours of flight. The recording life of the GoPro Hero 3 Silver Edition in all tested settings was adequate for the payloads flight, which was anticipated to be no longer than 3 hours. With the matter of recording time being a non-issue the setting selection process became an issue of image quality, and as 1080p with Protune activated clearly had the best image quality this was the setting configuration chosen for the flight.

Camera Mount

It was decided by the team that the camera should be mounted on a device that would allow for camera to be panned in flight, thus reducing the amount of time that the camera would capture essentially the same image.

Mount Selected

Two main options were presented to the team for the purposes of achieving the camera panning: the use of a simple mechanical device or a servo motor system. The simple mechanical suggested was a Scotch Yoke mechanism which changes rotational motion into alternating linear motion. This mechanism would then be attached to a bar extending from the GoPro housing, using the alternating linear motion to rotate the camera. The use of a servo motor to pan the camera would involve the programming of a servo motor to rotate the camera a certain number of degrees at set intervals.

The servo motor method of panning the camera involved fewer exposed moving parts and could be completed without the need to manufacture many custom parts or the calibration of motors. The servo motor mount would also be more flexible in terms of placement within the capsule, not requiring near perfect alignment like the Scotch Yoke mechanism. The servo motor mount consisted of three main portions: the servo motor, the arm assembly, and the camera holster.

Servo Motor

Servo Motor Selected – The servo selected for the mount is the Hitec HS-765HB Sail Arm, a servo motor intended for use on small remote controlled sail boats that allowed for 140^{o} of rotation and was relatively light. This servo motor weighs 3.6 ounces, which is lighter than other servo motors considered, and did not rotate an excessive amount, some servo motors considered rotated 1260^{o}.

Controlling the Servo Motor – The servo motor was controlled by the Arduino board that also communicated with the sensors and radio in the payload. The program written that controlled the sensors, radio, and servo motor used rotated the servo motor five degrees approximately once a minute, using the sensors and radio to control the timing of the program. The full program can be found in Appendix 2.

Results – On the first flight the servo motor appears to have been ineffectual, due to the program not running as anticipated, however, while the program was running the servo portion of the program ran smoothly. Additionally the reason that the servo mount was included on the payload, to change the view from the camera, was shown to be un-based as the capsule was constantly in motion, swaying and rotating, preventing the camera from capturing the similar images for more than a few frames, approximately a tenth of a second.

Arm Assembly

Components – The arm of the mount assembly was constructed using pre-fabricated parts and intended to allow the camera to be outside of the capsule while the servo could remain inside of the capsule. The am assembly consisted of a hub horn, two bore clamps, a tube, and a 90^{o} quad hub mount; all of these components were made of 6061 aluminum. The hub horn attached directly to the servo motor; attached to this was one of the bore clamps, next was the tube, then another bore clamp, and finally the 90^{o} quad hub mount.

Results – The assembly described above was successful as it allowed the servo motor to remain inside of the capsule and the camera to be rotated outside of the capsule. This assembly was also very strong, surviving the impact of the capsule as it returned to ground and despite the teams' best efforts could not be pulled apart without unscrewing the bore clamps.

Camera Holster

Design – The camera holster was designed to fit around and firmly grip the waterproof housing of the GoPro camera, while being flexible enough to remove the GoPro and housing, and remaining fairly light weight. The mount also had to be designed to be compatible with the 90^{o} quad hub mount. The design process began by taking the dimensions of the GoPro housing and recording the placement of the tapped holes on the 90^{o} quad hub mount. Next an initial mockup of the holster was created in SolidWorks that conveyed the general concepts to be explored for the holster design. Further iterations were created that reflected necessary revisions and strengthened the design until the mount seemed as though it would be reasonably strong. The model was then converted into a flat pattern for fabrication using SolidWorks sheet metal tools. A model of the arm assembly and the camera holster can be viewed in the figure below.

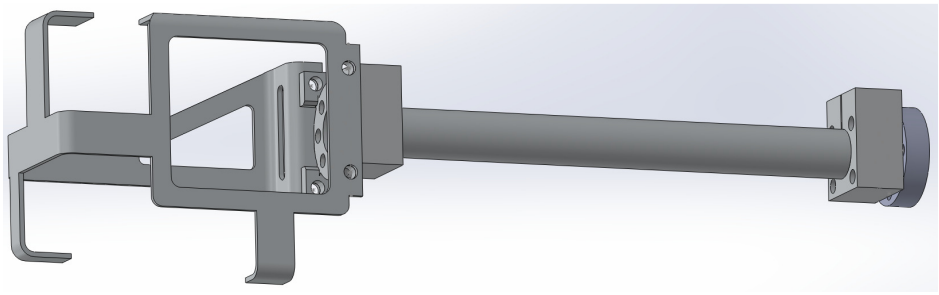


Figure 10: Camera Holster on Arm Assembly

Fabrication – A piece of sheet aluminum was cut by a computer numerical control (CNC) router using the flat pattern to control the milling of the metal. After the sheet metal had been cut to shape it had to be bent and formed to create the holster. The first bend made was a hard bend that would allow the holster to attach to the 90o quad hub mount. All other bends were made using the housing as a form and pressing the sheet metal into shape, the tabs were further bent using jewelry pliers to give a firmer fit.

Results – The camera holster fit the housing exceptionally well, requiring a great amount of force to be applied to the holster to either insert or remove the housing from servo motor mount. Additionally, while the holster was not designed to do so, the sheet metal acted as a spring and pressed in on the housing; this resulted in a firmer grip on the housing than was anticipated.

Conclusions

Throughout the duration of this project, the team achieved their goals of flying a successful payload and learning more about science and engineering in the process. The capsule performed well, sustaining no visible damage on either of the two flights, and the methods used to fabricate the capsule body while not readily available at the Milwaukee School of Engineering would be available to future teams if needed through the Milwaukee Makerspace. The telemetry system created performed well on the second flight but was somewhat temperamental; nevertheless this

system could be expanded upon by future teams increasing the utility system. The Geiger counter gathered evidence that supported the hypothesis that the radiation levels would increase with altitude, but was unable to determine an exact correlation. The sensor suite performed in part, the GPS and altimeter did not report data, and provided information that was used to complete the battery tests and confirm that the electronics onboard were kept within their operating temperature ranges.

Acknowledgements

The 2013 Elijah High Altitude Balloon Internship Payload Team would like to extend a special thanks to the following organizations and individuals:

The Wisconsin Space Grant Consortium
The Milwaukee School of Engineering
Dr. Matthew Panhans
Professor Jeff Korn
Vivian Mickelson
Jim Yauch
Roger Hajny
Richard Phillips
Valery Meyer
Jeananne Bargholz
Gary Bargholz
Marcus Reed Jr.
Jeff Hanson
Matt Neesley
Steve Pilone
Annya Ertel

Appendix 1: Camera Testing Table of Results

GoPro Camera Testing With 64 GB Micro SD Card			
Trial	Recording Time at 960p With Protune	Recording Time at 1080p Without Protune	Recording Time at 1080p With Protune
1	12975	14796	13517
2	13388	13555	13603
3	13330	14905	13462
4	13274	14641	13067
5	13002	14645	13434
6	13222	13904	13779
7	13125	14993	13296
8	12457	14629	13635
9	12772	14613	13358
10	13102	14693	13398
Average (s)	13064.7	14537.4	13454.9
Sample Standard Deviation (s)	281.944	451.726	197.793
Uncertainty (s)	89.2	142.8	62.5
Average (converted to minutes)	217.745	242.290	224.248
Uncertainty (converted to minutes)	1.486	2.381	1.042
Average (converted to hours)	3.62908	4.03817	3.73747
Uncertainty (converted to hours)	0.02477	0.03968	0.01737

Appendix 2: Full Code Used for Flights

```
/*************************************************************
 * Filename: payload.ino
 * Date: 8/7/2013
 * Author: Dan Kass (kassd@msoe.edu)
 *      2013 WSGC High Altitude Balloon Payload Team
 * Version: 1.6
 * Micro-controller: Arduino Mega
 * Provides: This is the main code that will run on the
 *    Arduino Mega during flight
 *************************************************************/

/**************************
 *********Includes*********
 **************************/

#include <Wire.h>          //I2C Library (Altimeter)
#include <TinyGPS.h>       //GPS Library
#include <Radio.h>         //Radio Library
#include <GeigerCounter.h> //Geiger Counter Library
#include <Servo.h>         //Servo Library
#include <SD.h>            //SD Card Library
#include <math.h>          //Math Library for Thermistors
#include <stdlib.h>        //Stard Library

/**************************
 ********Variables*********
 **************************/
 //LED Pin Locations
 int LED1 = 44; //Thermister LED  gray
 int LED2 = 45; //Geiger Counter  blue
 int LED3 = 46; //Altimeter  green
 int LED4 = 47; //GPS  brown
 int LED5 = 48; //Servo  purple
 int LED6 = 49; //SD Card  white
 int LED7 = A15; //Power LED

 int delayLED = 350; //The delay between delay Flashes

 double externTemp; //external temperature
 double internTemp; //internal temperature
 int externTherm = 0; //Location of Pin
 int internTherm = 1; //Location of Pin

 int geiger = 13; //Location of pin
 int cpm; //counts per minute
 GeigerCounter geigerCounter(geiger);

 Servo camera; //creates servo object
 int camMax = 160; //max degree of Servo
 int camMin = 20; //min degree of Servo
 int cameraLocation = camMin; //start of the servo
 int camDirection = 1; // 1 is up 0 is down
 int servo = 7; //Location of Pin

 const int SENSORADDRESS = 0x60; // address specific to the MPL3115A1 used in I2C
```

```
Radio radio;  //creates radio object

TinyGPS gps; //creates the GPS Object

//Variables from the GPS
float latitude = 0;
float longitude = 0;
float gpsAlt = 0;
float gpsSpeed = 0;
int gpsSatellites = 0;

//Voltage Divider Variables
int val = 0;
float pinVoltage = 0;
float batteryVoltage = 0;
float ratio = .423;

File data; //used for SD Card
int slaveSelect = 53; //must be set to output for SD card to work

//Set up Code
void setup(){
//Start up code initialize devices.

        //LED Initialization
        //Must write low so that are off to start
        pinMode(LED1, OUTPUT);
        digitalWrite(LED1, LOW);
        pinMode(LED2, OUTPUT);
        digitalWrite(LED2, LOW);
        pinMode(LED3, OUTPUT);
        digitalWrite(LED3, LOW);
        pinMode(LED4, OUTPUT);
        digitalWrite(LED4, LOW);
        pinMode(LED5, OUTPUT);
        digitalWrite(LED5, LOW);
        pinMode(LED6, OUTPUT);
        digitalWrite(LED6, LOW);
        pinMode(LED7, OUTPUT);
        digitalWrite(LED7, LOW);

        //Start up lights run across to the power light
        digitalWrite(LED1, HIGH);
        delay(delayLED);
        digitalWrite(LED1, LOW);
        delay(delayLED);
        digitalWrite(LED2, HIGH);
        delay(delayLED);
        digitalWrite(LED2, LOW);
        delay(delayLED);
        digitalWrite(LED3, HIGH);
        delay(delayLED);
        digitalWrite(LED3, LOW);
        delay(delayLED);
        digitalWrite(LED4, HIGH);
        delay(delayLED);
        digitalWrite(LED4, LOW);
        delay(delayLED);
```

```
	digitalWrite(LED5, HIGH);
	delay(delayLED);
	digitalWrite(LED5, LOW);
	delay(delayLED);
	digitalWrite(LED6, HIGH);
	delay(delayLED);
	digitalWrite(LED6, LOW);
	delay(delayLED);
	digitalWrite(LED7, HIGH); //want to keep the power LED on

	//Serial
	Serial.begin(9600);
	//GPS Serial Setup
	Serial1.begin(9600);

	radio.mycall("KC9ZGW-7");

	//setup I2C
	Wire.begin();
	if(IIC_Read(0x0C) == 196){ //checks who_am_i bit for basic I2C handshake test
		digitalWrite(LED3, HIGH);
	}
	else {
		digitalWrite(LED3, LOW);
	}
	//These are the sensor configuration values used in the sample code
	//they work so don't want to mess with it.

	// CTRL_REG1 (0x26): enable sensor, oversampling, altimeter mode
	IIC_Write(0x26, 0xB9);
	// CTRL_REG4 (0x29): Data ready interrupt enabled
	IIC_Write(0x29, 0x80);
	// PT_DATA_CFG (0x13): enable both pressure and temp event flags
	IIC_Write(0x13, 0x07);
	// This configuration option calibrates the sensor according to
	// the sea level pressure for the measurement location
	// BAR_IN_MSB (0x14):
	IIC_Write(0x14, 0xC6);
	// BAR_IN_LSB (0x15):
	IIC_Write(0x15, 0x5B);

	//SD Card Setup
	pinMode(slaveSelect, OUTPUT);
	if(SD.begin()) {
		digitalWrite(LED6, HIGH); //turn on LED SD Card initialized
	}

	data = SD.open("datalog.csv", FILE_WRITE);
	if(data) {
		//writing the header to the csv file, column titles
		data.println("Time From Start, Internal Temperature, External Temperature, Battery Voltage, Counts
Per Minute, Altitude, Altimeter Temperature, Latitude, Longitude, GPS Altitude, Speed, Satellites");
		data.close(); //close the file
	}
	else {
		digitalWrite(LED6, LOW); //There was an error opening the file turn off the LED
	}

	//Servo
```

```
	//to get it to move camera.write(20 through 160)
	camera.attach(servo);
	camera.write(cameraLocation);
	if(camera.attached()) {
		digitalWrite(LED5, HIGH);
	}


	//Radio set up beacon and location time
	radio.beaconText("WSGC High Alt Balloon. DataOrder: Time, Intern Temp, Extern Temp, Bat Volt, Counts Per
Minute, Alt, Alt Temp, Lat, Long, GPS Alt, Speed, Sats");
	radio.beacon(300); //beacons every 300 seconds (5 minutes)
	radio.location(60); //location sends out every 60 seconds

}


//Main running loop
void loop(){

	//get the Internal and External Temperatures;
	externTemp = getTemp(analogRead(externTherm));
	internTemp = getTemp(analogRead(internTherm));

	//The best way to check to see  if they are working properly is
	//to check their values against them self
	//might have to be changed for launch
	if((externTemp < internTemp + 2) && (externTemp > internTemp - 2))
	{
		digitalWrite(LED1, HIGH);
	}
	else
	{
		digitalWrite(LED1, LOW);
	}

	//getting GPS Data
	bool newData = false;
	// For one second we parse GPS data and report some key values
	for (unsigned long start = millis(); millis() - start < 1000;)
	{
		while (Serial1.available())
		{
			char c = Serial1.read();
			// Serial.write(c); // uncomment this line if you want to see the GPS data flowing
			if (gps.encode(c)){ // Did a new valid sentence come in?
				newData = true;
			}
		}
	}

	if (newData)
	{
		digitalWrite(LED4, HIGH);
		unsigned long age;
		gps.f_get_position(&latitude, &longitude, &age);
		gpsSatellites = gps.satellites();
		gpsSpeed = gps.f_speed_mph();
		gpsAlt = gps.f_altitude();
```

```
}
else
{
        digitalWrite(LED4, LOW);
}

//getting Altimeter data

// This function reads the altitude and temperature registers, then
// concatenates the data together, and prints in values of
// meters for altitude and degrees C for temperature.

// variables for the calculations
int m_altitude, m_temp, c_altitude;
// these must be floats since there is a fractional calculation
float l_altitude, l_temp;
float altitude, temperature;

// read registers 0x01 through 0x05
m_altitude = IIC_Read(0x01);
c_altitude = IIC_Read(0x02);
// the least significant bytes l_altitude and l_temp are 4-bit,
// fractional values, so you must cast the calculation in (float),
// shift the value over 4 spots to the right and divide by 16 (since
// there are 16 values in 4-bits).
l_altitude = (float)(IIC_Read(0x03)>>4)/16.0;
m_temp = IIC_Read(0x04); //temp, degrees
l_temp = (float)(IIC_Read(0x05)>>4)/16.0; //temp, fraction of a degree

// here is where we calculate the altitude and temperature
altitude = (float)((m_altitude << 8)|c_altitude) + l_altitude;
temperature = (float)(m_temp + l_temp);

long time = millis() + 2000; //current time plus 2 seconds
// wait here for new data
while(check_new() == false)
{
        //had some problems with data retrial if problem with altimeter so put
        //a break if takes longer than 2 seconds
        if(millis() > time){
        digitalWrite(LED3, LOW);
        break;
        }
}

//move the servo
if(camDirection == 1) {
        //if direction is up add 5 degrees
        cameraLocation += 5;
}
else {
        //other wise subtract 5 degrees
        cameraLocation -= 5;
}

if(cameraLocation == camMax) {
        //when location is max set label to go down
        camDirection = 0;
}
```

```
if(cameraLocation == camMin) {
        //when location is min set label to go up
        camDirection = 1;
}
camera.write(cameraLocation);

//getting Geiger Counter data
cpm = geigerCounter.read(30); //going to get a 30 second sample
//That is background radiation so if it is reading that we want the light on
if((cpm > 1) && (cpm < 30)) {
        digitalWrite(LED2, HIGH);
}
else {
        //otherwise we want it off
        digitalWrite(LED2, LOW);
}
//Battery Monitor
val = analogRead(batMonPin);
pinVoltage = val * 0.00488; //calculate the voltage on the a/d pin

batteryVoltage = pinVoltage / ratio; //used with voltage divider

//create data string

//need to convert all of the floats/doubles to strings
//dtostrf(value, width, precision, output);
char temp[10];//temporary string

//gets the time from start
dtostrf(millis(),1,0,temp);
String information = String(temp) + ",";

dtostrf(internTemp,1,2,temp);
information = information + String(temp) + ",";

dtostrf(externTemp,1,2,temp);
information = information + String(temp) + ",";

dtostrf(batteryVoltage,1,2,temp);
information = information + String(temp) + ",";

dtostrf(cpm,1,2,temp);
information = information + String(temp) + ",";

dtostrf(altitude,1,2,temp);
information = information + String(temp) + ",";

dtostrf(temperature,1,2,temp);
information = information + String(temp) + ",";

dtostrf(latitude,1,5,temp);
information = information + String(temp) + ",";

dtostrf(longitude,1,5,temp);
information = information + String(temp) + ",";

dtostrf(gpsAlt,1,2,temp);
information = information + String(temp) + ",";
```

```
        dtostrf(gpsSpeed,1,2,temp);
        information = information + String(temp) + ",";

        dtostrf(gpsSatellites,1,0,temp);
        information = information + String(temp);

        //send data string over radio
        radio.locationText(information);

        //save data to SD card
        data = SD.open("datalog.csv", FILE_WRITE);
        if(data) {
                //writing the header to the csv file, column titles
                data.println(information);
                data.close(); //close the file
        }
        else {
                digitalWrite(LED6, LOW); //There was an error opening the file turn off the LED
        }

        delay(10000); //just wait 10 seconds don't need to go to fast
}


/**************************
 ********Functions*********
 **************************/

/***********************************
 * Function: getTemp
 * Author: Mark McComb, hacktronics LLC
 * Parameters: Raw ADC Value from the
 *    Arduino
 * Returns: Temperature in C
 * Provides: The temperature reading
 *    from the thermistor
 ***********************************/
double getTemp(int RawADC) {
        double Temp;
        // See http://en.wikipedia.org/wiki/Thermistor for explanation of formula
        Temp = log(((10240000/RawADC) - 10000));
        Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp));
        Temp = Temp - 273.15;          // Convert Kelvin to Celsius
        return Temp;
}

/***********************************
 * Function: IIC_read
 * Author: SparkFun Electronics, A.Weiss, 7/17/2012
 * Parameters: register Address
 * Returns: byte of data
 * Provides: The I2C read functionality
 *    for the altimeter
 ***********************************/
byte IIC_Read(byte regAddr)
{
        // This function reads one byte over IIC
        Wire.beginTransmission(SENSORADDRESS);
        Wire.write(regAddr);  // Address of CTRL_REG1
```

```
	Wire.endTransmission(false); // Send data to I2C dev with option
                                                                    //  for a repeated start. THIS IS
                                                                    //  NECESSARY and not supported before
                                                                    //  Arduino V1.0.1!!!!!!!!!
	Wire.requestFrom(SENSORADDRESS, 1); // Request the data...
	return Wire.read();
}

/**********************************
 * Function: IIC_Write
 * Author: SparkFun Electronics, A.Weiss, 7/17/2012
 * Parameters: register Address, and value
 * Returns: none
 * Provides: The I2C write functionality
 *    for the altimeter
 **********************************/
void IIC_Write(byte regAddr, byte value)
{
	// This function writes one byte over IIC
	Wire.beginTransmission(SENSORADDRESS);
	Wire.write(regAddr);
	Wire.write(value);
	Wire.endTransmission(true);
}

/**********************************
 * Function: check_new
 * Author: SparkFun Electronics, A.Weiss, 7/17/2012
 * Parameters: none
 * Returns: boolean true if there new data
 * Provides: The test for new date
 *    for the altimeter
 **********************************/
boolean check_new()
{
	// This function check to see if there is new data.
	// You can call this function and it will return TRUE if there is
	// new data and FALSE if there is no new data.

	// If INT_SOURCE (0x12) register's DRDY flag is enabled, return
	if(IIC_Read(0x12) == 0x80) // check INT_SOURCE register on
                                                                    // new data ready (SRC_DRDY)
	{
		return true;
	}
	else return false;
}
```