Final Scientific Report
Maple Leaf Particle Detector

Andreas Buttenschön
Quinton Farr
Wyatt Johnson
Cory Hodgson, Ian R. Mann
Laura Mazzino,
and Jonathan Rae

University of Alberta
Department of Physics
Edmonton AB T6G 2G7
Canada

1.6

December 16, 2011

Table of Contents

List of Figures

1 Acronyms

Glossary

ADC	Analog Digital Converter. 11, 12
ASCII	American Standard Code for Information Interchange. 14
CPP	Center for Particle Physics. 10
CPU	Central Processing unit. 11, 12
CSA	Canadian Space Agency. 6, 12
CSBF	Columbia Scientific Balloon Facility. 12, 14, 15
CUPC	Canadian Undergraduate Physics Conference. 6
DFL	David Florida Laboratories. 10, 12
EDAC	Electronic Design Automation Companies. 11
EEPROM	Electrically Erasable Programmable Read-Only Memory. 9, 10
GM	Geiger Müller tubes. 9
HASP	High Altitude Student Platform. 6, 7, 9–12, 14, 15
HV	High Voltage. 10, 11
I2C	Inter-Integrated Circuit. 11
MLI	Multi Layer Insulating Tape. 15
MOSFET	metal-oxide-semiconductor field-effect transistor. 7
PCB	Printed Circuit Board. 9–11
PSIP	Payload Specification Integration Plan. 14
RS232	Recommended Standard 232. 9
SRIM	Stopping and Range of Ions in Matter. 7
UA-HAB	University of Alberta High Altitude Balloon. 5, 11, 15

2 Participants

Name	Position	Sex	Ethnicity	Nationality
Ian R. Mann	Project Supervisor	Male	Caucasian	British
I. Jonathan Rae	Project Supervisor	Male	Caucasian	British
Laura Mazzino	Team Leader	Female	Latina	Italian
Quinn Farr	Project Manager	Male	Caucasian	Canadian
Andreas Buttenschön	Software programmer	Male	Caucasian	German
Cory Hodgson	Systems Manager	Male	Metis	Canadian
Wyatt Johnson	Analyst	Male	Caucasian	Canadian

3 Presentations

The UA-HAB team presented at a number of local, national and international conferences. The goal of these presentations was to:

- Give the undergraduates presenting and public speaking experience
- Showcase possible undergraduate research opportunities
- Raise awareness about CSA funded initiatives
- Raise awareness about international opportunities such as HASP to undergraduate students

Conferences (Oral Presentations):

University of Alberta Astronomy and Space Physics Student Symposium (Local) This was a small symposium held by space physics and astronomy research groups to give summer students a chance to present on their summer research. 15 minute presentations were given to an audience of about 15 people

Undergraduate Physics and Engineering Physics Symposium (Local) This was a small symposium held by the Undergraduate Physics Society and the Engineering Physics Association. 15 minute presentations were given to an audience of about 15 people.

Canadian Undergraduate Physics Conference (National) CUPC is a national conference where over 150 physics students come to present their summer research. 15 minute presentations were given to audiences and we received second place in the space and astrophysics category.

Canadian Space Society Annual Summit (National) This is a meeting industry, academia, military and civilians connected to and interested in space and space science in Canada. A 12 minute presentation was given to an audience of about 30 people.

American Geophysical Union Fall Meeting 2011 (International) The AGU Fall Meeting is the largest worldwide conference in the geophysical sciences, attracting nearly 20,000 Earth and space scientists, educators, students, and policy makers. This meeting showcases current scientific theory focused on discoveries that will benefit humanity and ensure a sustainable future for our planet.

Conferences (Poster Presentations):

University of Alberta Students Union Research Symposium (Local) The Students Union's Research Symposium is a campus wide research symposium showcasing undergraduates research.

American Geophysical Union Fall Meeting 2011 (International) The AGU Fall Meeting is the largest worldwide conference in the geophysical sciences, attracting nearly 20,000 Earth and space scientists, educators, students, and policy makers. This meeting showcases current scientific theory focused on discoveries that will benefit humanity and ensure a sustainable future for our planet.

4 Science Introduction

In January 31 1958, James Van Allen and collaborators launched the Explorer 1, the first Earth satellite of the United States, as part of the International Geophysical Year, celebrated in 1957 and extended to 1958. Explorer 1 was equipped with an Anton 314 omnidirectional Geiger-Muller tube Geiger counter to measure any space radiation. sensitive to measure protons with energies larger than 30MeV and electrons with energies greater than 3MeV. Van Allen hoped to measure galactic cosmic rays that were too slow not energetic enough to penetrate Earth's atmosphere. The results were puzzling, as a colleague of Van Allen's exclaimed "My God, space is radioactive! ". The Geiger-Muller produced some expected readings at the low points of the satellite's elliptical orbit and dropped to zero in the higher points of the orbit. Further testing, done with the instrumentation at ground laboratories, after Explorer 3 produced similar results, indicated that the Geiger-Muller detector saturated when producing zero readings. The discovery revealed the existence of the now known as the Van Allen Radiation Belts, doughnut-shaped regions surrounding Earth, and within Earth's magnetic bubble in Earth's magnetosphere where the where highly energetic electrons and protons are trapped.

The region where the Van Allen radiation belts reside is also the region where many satellites orbit the Earth, including telecommunications satellites, and the Global Positioning System (GPS) satellites. Understanding the variability of energetic particles that can damage and destroy the sensitive electronics within these satellites is of paramount importance to exploiting near-Earth space.

The University of Alberta-High Altitude Balloon (UA-HAB) project was to design, build, test and fly an instrument to study the energetic radiation environment of the Radiation Belts. By building an instrument that is able to discriminate both incoming particle energy and time, enabling the UA-HAB project team to study the dynamics of the Radiation Belts over a long period of time, which is a research topic at the forefront of both NASA and Canadian Space Agency priorities.

The Proton Radiation Belt exhibits little variation due to solar activity and it is pretty stable. The main source of the protons in the inner belt is the Cosmic Ray Albedo Neutron Decay, CRAND: Cosmic rays interact with constituents of the earth's atmosphere produce neutrons which are emitted into the trapped region. Protons are produced by beta decay of the neutrons, that become trapped by Earth's magnetic field Figure 4 (right). The inner proton belt exhibits high fluxes during solar minimum and low fluxes during solar maximum: solar activity during solar maximum produces a shielding effect and cosmic ray flux reaching the earth is low compared to solar minimum.

5 Design

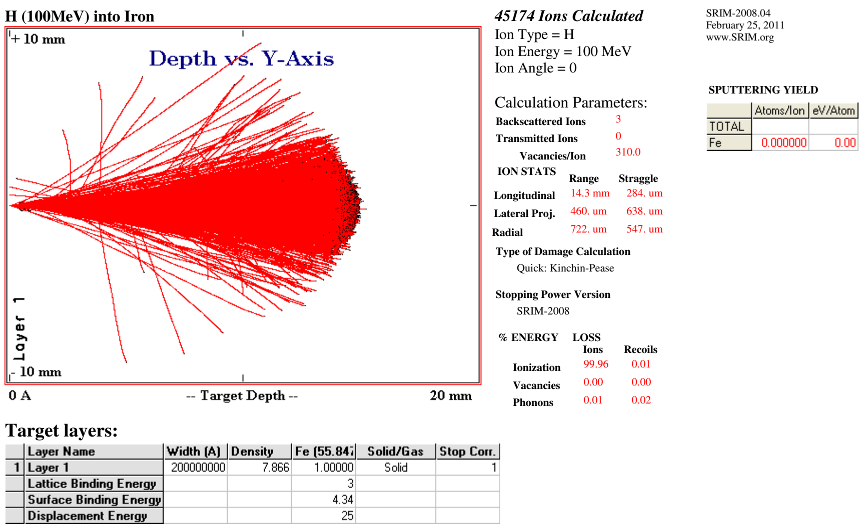
5.1 Principle of operation

The detector is composed of 3 LND-712 Geiger-Müller counters, encased in 1018 (low carbon) steel to provide minimum energy detection thresholds. The purpose of the payload is to measure the particle precipitation from the inner Van Allen radiation belts and cosmic rays from intergalactic sources such as supernovae. It is designed to measures particle flux as a function of altitude.

5.2 Energy Range

The steel surrounding the Geiger counters discriminates the energies of particles detected by them. Any count that registers in one of the Geiger counters must have a minimum energy to penetrate the surrounding steel shielding. Each successive Geiger counter is encased by a thicker amount of steel (the top having the thinnest, the bottom the thickest), which leads to a higher minimum energy threshold going down the detector. The successive energy minimums of our detector are 50MeV, 100MeV and 150MeV.

These energy barriers were determined by SRIM software, which uses probabilistic Monte Carlo methods to calculate penetration depth of ions in different materials. The following are results in iron (a good approximation to low-carbon steel) for 50MeV, 100MeV and 150MeV, which correspond to the minimum energy detection thresholds.



45174 Ions Calculated
Ion Type = H
Ion Energy = 100 MeV
Ion Angle = 0

SRIM-2008.04
February 25, 2011
www.SRIM.org

Calculation Parameters:

Backscattered Ions	3
Transmitted Ions	0
Vacancies/Ion	310.0

ION STATS	Range	Straggle
Longitudinal	14.3 mm	284. um
Lateral Proj.	460. um	638. um
Radial	722. um	547. um

Type of Damage Calculation
Quick: Kinchin-Pease

Stopping Power Version
SRIM-2008

% ENERGY LOSS	Ions	Recoils
Ionization	99.96	0.01
Vacancies	0.00	0.00
Phonons	0.01	0.02

SPUTTERING YIELD

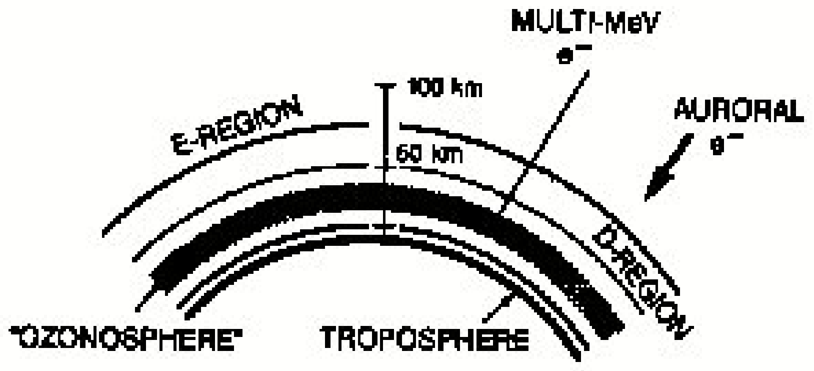
	Atoms/Ion	eV/Atom
TOTAL		
Fe	0.000000	0.00

Target layers:

	Layer Name	Width (A)	Density	Fe (55.847	Solid/Gas	Stop Corr.
1	Layer 1	200000000	7.866	1.00000	Solid	1
	Lattice Binding Energy			3		
	Surface Binding Energy			4.34		
	Displacement Energy			25		

Figure 1: SRIM simulation for iron, 50MeV

The particles detected by the Geiger tubes are not at their original energies, as some energy is absorbed by the upper atmosphere and the ionosphere which exists between the HASP platform and the mirror points. Energy deposition will occur within these regions as per the following graph (see Figure 2).



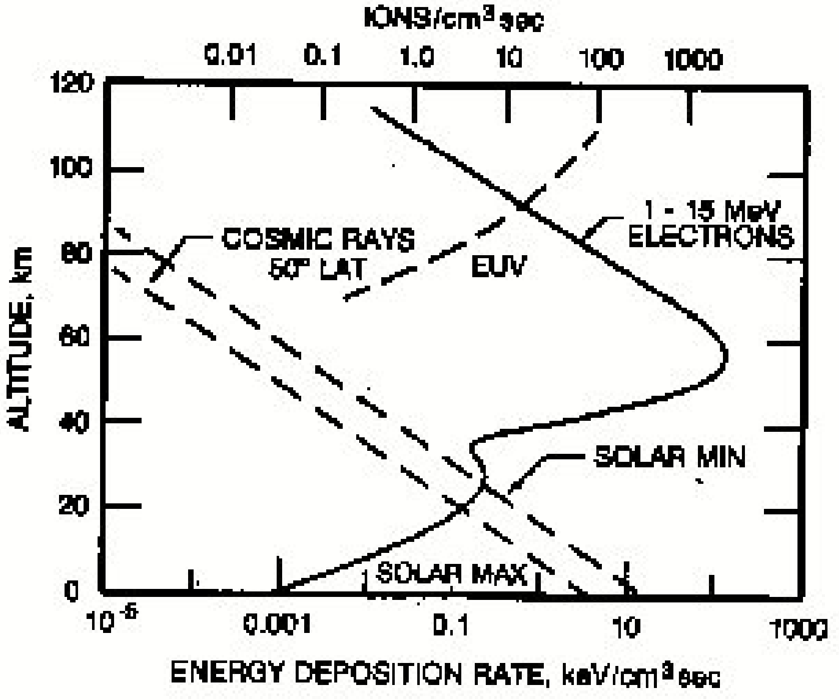


Figure 2: (Top) A diagram illustrating the much greater depth to which precipitating relativistic electrons can penetrate into the Earth's atmosphere as compared to characteristically lower-energy auroral electrons.
(Bottom) Multi-MeV electrons, when present, represent the dominant ionization source in the middle atmosphere (40-80 km altitude). The figure shows the expected energy deposition versus altitude if an energetic (1-15 MeV) electron event observed at geostationary orbit were to precipitate into the atmosphere. This is contrasted with solar extreme ultraviolet (EUV) ionization at high altitude and the effect of galactic cosmic rays at low altitude. http://www.srl.caltech.edu/sampex/pet.html

5.3 Electronics

The Geiger tubes had the option to be ordered with a simple circuit which would count events, and store the information to a computer connected via USB. The team decided to order the Geiger tubes independently, and consolidate what would otherwise be three counting boards with USB output into one PCB with serial output. The design of the electronics board was outsourced to the physics department's electronic shop. The undergraduates did all the board populating, testing and troubleshooting. Five boards in total were constructed. We decided to have five boards was 2 for testing, 1 prototype, 1 flight board, and one extra in case any were damaged. The basic design consisted of power running into a 5V regulator to establish a 5V line. To power the Geiger Muller tubes a high voltage (between 450Vand 600V) line was need for each channel. This was done by using a MOSFET chip, and a 1:10 transformer on each channel. See Figure 4 for a power diagram. See Figure 3 for a basic electronic schematic of the payload.

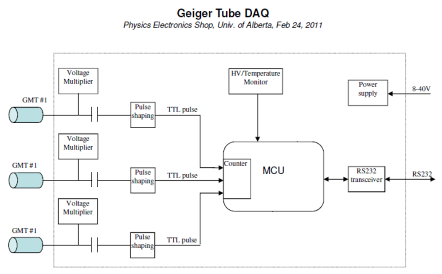


Figure 3: Outline of Electronics Schematics

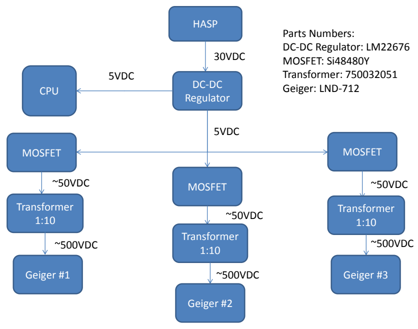


Figure 4: Power distribution diagram

Communication to the PCBs was done through an RS232 serial jack. A temperature sensor was also included on the PCBs, but due to undetermined causes, it was not reliable. The PCB recorded counts by measuring the spikes in voltage produced by the GM tubes every time an "event" was registered.

An addition was later made to the PCBs specifically for the weather balloon testing in subsection 8.3. These add-ons consisted of a pressure sensor and a 1MB EEPROM storage chip to store data collected over the duration of the weather balloon flight. They simply plugged into the PCBs through a 6 pin header connection. These add-ons were not included during the HASP flight. All 5 boards are numbered one to five and were referred to as board 1 through board 5 for the duration of the project. All boards were operational except for board 5 due to communication problems. Uses and performances of boards one through four will be addressed in subsection 8.3.

5.4 Mechanical Drawings

The steel is in the shape of a square pyramid, to create the different energy thresholds required. Four holes are drilled in each corner of the steel pyramid, through the HASP mounting plate, and bind the stack to itself and the mounting plate via steel nuts. Long metal screws bind a foam enclosure to the HASP mounting plate via holes drilled in the mounting plate, and serve to hold the motherboard to the platform through the use of more steel nuts. The entire payload is then encased in a foam board/aluminium housing which acts as a thermal and weather barrier (since steel alone rusts, and absorbs lots of heat from sunlight) and provides an extra layer of protection to the electronics board. See the Mechanical Drawings Figure 23 in the Appendix for an exploded schematic.

5.5 PCB Enclosure

On the HASP mounting plate, the PCB board was enclosed in a simple foam box, glued together with epoxy and bound to the platform with screws drilled through the mounting plate. The box had 5 holes, one for the serial port connection, one for the power connection, and one for each set of wires to and from the Geiger counters. The function of the box is to shelter the board from small collisions, and to bind it to the mounting plate see Figure 28.

5.6 Heat Shield

A heat shield was constructed to surround the payload and PCB enclosure (see section 9). It was made from foam board, aluminium foil (heat reflection), aluminium tape, and epoxy (for stability). There was a small gap in the box to allow heat exchange between the environment and the inside of the payload (so that heat from the New Mexico sun wouldn't be trapped inside (see Figures 28 30 29).

5.7 Timepix

Near the beginning of the project, the team attempted to requisition "Timepix" chips used at CERN for energy profiling to complement the Geiger component of the detector. These chips would have given energy profiles of incident particles, along with orientation and temporal information. The Timepix chip has a 256×256 pixel detection area which records information as particles pass through the pixels. Initially, the steel enclosure was designed to accommodate these additional detector components with rough dimensional information given by the supplier. The exact dimensions were later given on a data sheet, and the design was changed accordingly (they were found to be larger than anticipated, so the shielding thickness had to be reduced to compensate). As a result, the already-ordered steel had to be cut into the form of the new design. Later, when the Timepix were known to be unavailable the design was changed again, to seal off the Geiger tube chambers (see subsection 6.1).

6 Manufacturing

6.1 Machining

Manufacturing the steel enclosure took longer than anticipated due to a change in design to accommodate detector components (see subsection 5.7). As a result, the steel pieces we ordered first had to be cut by a large hack-saw or band-saw to their approximate size. This was made more difficult by the machine shop moving to the new physics building from late April to the beginning of June, which delayed cutting. When all the steel had been cut, it was moved to the university's heavy machine lab, wherein old CPP (center for particle physics) machines resided. In the heavy lab, the mill and drill press were used to both machine all the individual pieces to exact dimension, and drill holes through which the connecting rods would bind them.

First, the steel enclosure was cut to allow for additional detector components, and this model was shipped to DFL for testing. Rough cuts had been made on steel pieces sufficient for the assembly of three of these detectors. The design was then changed when it was discovered these components would be unavailable, see subsection 5.7 . Parts were taken from the other two enclosures to seal off one, final enclosure. This sealed enclosure had holes filed in the steel components to allow the Geiger wires to reach the containers. Finally, all the parts were labelled so that they could be easily assembled on the HASP mounting plate, bound by the steel rods. Detailed specifications of the final design can be found in subsection 5.4

6.2 Electronics

A total of five boards were manufactured. The first board was constructed independently, in the old electronics shop, to be made ready for DFL testing. It was tested by the electronics shop supervisor, and then given a conformal coating for DFL testing. Later, a team member realized a resistor on two of the three HV channels of this board were different from the corresponding resistor on other boards (see subsection 8.2). The second board was manufactured immediately after the first, in two phases. First, the majority of the components were added with the pick and place machine in the old electronics shop. At this point, the electronics shop began to move (May-June) and further manufacturing was delayed. Finally, in the new electronics shop, the surface mounted pieces were attached and the second board was complete. Then, after a shipping delay, the parts for the final three PCB boards arrived, and were assembled together. This soldering was done by hand due to the a vent being installed improperly, making the pick and place machine unusable. Six resistors and three capacitors were changed on the final three boards compared to the first two, the resistors due to component availability, and the capacitors to reduce the charge time of HV channels. Finally, for weather balloon flights (see subsection 8.3), an EEPROM chip was soldered to a blank breadboard, with a pressure and temperature sensor. This circuit was integrated to the board's 5V line, and this allowed data to be stored locally on our PCB during flights, and recovered later for analysis.

6.3 Integration to HASP Mounting Plate

After rigorous testing (see subsection 8.2), board 4 was chosen to be integrated to the HASP mounting plate. First, the serial cables were soldered on to an F/F serial point, which was in turn connected to the serial port on the board. Next, all cables from the EDAC connector were tied aside, except for the power, and ground lines, which were soldered together in series to a single wire, both of which were inserted in a pin-header, to attach/detach quickly from the circuit board. At integration, it was suggested that the Geigers be soldered directly to the board, as well as the power cable. These were all attached directly and applied Corona Doping to prevent HV discharge. On the mounting plate, the electronics were encased in a protective Styrofoam casing, screwed directly in to the mounting plate (see Figure 21). As outlined in the (Figure 20), the steel housing was mounted directly to the mounting plate via the connecting rods.

7 Firmware

The UA-HAB PCB has all the measuring tools required to measure temperature, pressure, and counts of three Geiger-Mueller tubes. All these sensors are controlled by a ATMEL-328P (CPU), which is run at 3.57MHz. The CPU, runs the UA-HAB firmware. The firmware code is divided into different subsystems, each representing a physical subsystem of the CPU. The different subsystems are described in the firmware documentation in the Appendix. UA-HAB is interested at obtaining the flux of protons at altitudes of 30km, additionally atmospheric data is also obtained (temperature & pressure), the data acquisitions are outlined in subsection 7.1.

7.1 Data Acquisition

There are three types of data, that detector is interested in, counts, temperature and pressure. The flux of particles are measured using three Geiger-Mueller tubes each connected to one of the available External-Interrupts of the CPU. The external interrupt is triggered by a voltage change in the counting circuit (reference description of this circuit here). For more details see section 13 for more details. The temperature is measured using a T101 chip, manufactured by Texas Instruments, which is connected to the I2C bus of the CPU see the firmware documentation section 13 for more details. The pressure is measured using a MPXA6115 pressured sensor, which is connected to one of the ADC (analog-digital-converter) pins of the CPU see section 13 for more detail.

7.2 Recognized Issues

In section 12 it was found that data packets were send at time intervals longer then the anticipated 10s. The firmware controls the time interval at which packets are send, thus this discrepancy in the time interval is a result of a malfunction in the firmware (see the section section 12). The time interval was measured between packets, during a fixed 120s time interval 9 packets were received, therefore the average time interval between packets is approximately 13s. The time interval is approximately 33% higher then expected. A This agrees with the calculated time interval from the received packets (see section 12).

Potential reasons for the longer time interval were investigated for their potential of producing the observed. The design of the timer and the high voltage circuit are most likely responsible in the increased time interval between packets and the drop in high voltage during periods of high counts. The timer and the high voltage control logic of the payload are implemented as interrupts, which are called at fixed frequencies. The assembler code generated by the compiler was analyzed to determine the number of CPU cycles required to serve a single interrupt. The code serving the external interrupt, requires 25 CPU cycles per execution, meaning each observed count requires 25 CPU cycles. Each, execution of the high voltage control requires approximately 350 CPU cycles. The timer interrupt has to be called every 224 cycles to reset the counter. However, servicing the high voltage control requires more cycles then 224. Further, interrupts cannot be interrupted therefore the timer will not be reset and the clock will not be incremented. The high voltage control code, requires $28,125,000$ per second in the worst case scenario. The timer requires $1,600,000$ CPU cycles per second. The CPU only has 3.57 million cycles per second available. The the interrupts have fixed priorities assigned to them by the manufacturer. The external interrupts are the highest, followed by the timer and the high voltage control (ADC). This means, if there were to many counts occurring, the CPU would fall behind in servicing the timer and the high voltage control. Falling behind on the timer means that time was stopped being counted. Consequently, packets were send less frequently. Furthermore, this also means that the high voltage circuit was not charged as regularly as required. Lowering the high voltage across all channels, during times of high flux.

8 Testing

8.1 David Florida Laboratories

The team flew to Ottawa, Ontario on May 15th to conduct pressure and thermal testing at the Canadian Space Agency's (CSA) David Florida Laboratories (DFL). The testing followed the temperature pressure profile approved by HASP.

Testing began May 16th. The team was first given a tour of the facilities and began thermal testing at approx. 1:30pm. The purpose of the test was to evaluate the performance of the electronics and the mechanical components in a formal testing environment prior to integration at CSBF. We tested our first prototype board. The test lasted 4 hours, with the temperature first being dropped 3°C per minute to −50°C, held for 2 hours. Following, the temperature was increased at the same rate to 50°C, held for 30 minutes. After which the payload cooled to room temperature, concluding the test. See Figure 5 for the collected data. The detector malfunctioned at a temperature of ≈ 40°C (see Figure 5). The issue of performance drop at large temperature variances was corrected during the CSBF Integration in Palestine TX. The gaps in the data in Figure 5 were a result of a recording hardware malfunction, resulting in temporary data loss.

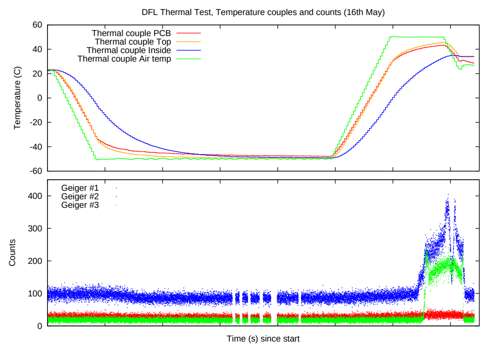


Figure 5: David Florida Laboratories Temperature Test - Counts and Temperature. During this test, the temperature was reduced to −50°C and held there for a period of time before heating up to ≈ +50°C. Errors occurred with the board near the end of the test as a result of the sharp change in temperature. This was later corrected with the addition of thermal shielding.

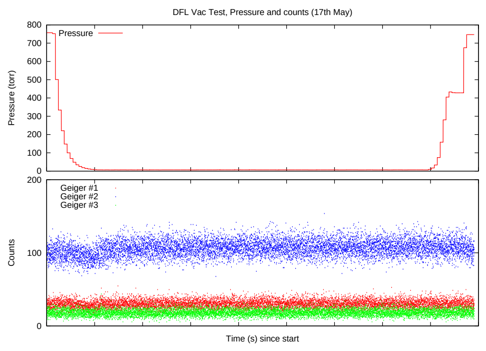


Figure 6: David Florida Laboratories Altitude Test - Counts and Temperature. During this test, the pressure of the test chamber was reduced, then held at a constant low temperature, where after it was raised again to atmospheric temperature. No main errors were detected during this test.

Altitude testing of the payload was performed the following day. The test was conducted using the pressure profile outlined in Figure 5. The payload was first cooled to −50°C, where then the pressure was reduced from 760 torr down to 6.5 torr in a period of ≈ 20 minutes. The payload was held at this pressure for 2 hours, and then returned back to atmospheric pressure. There was no malfunction observed.

8.2 Custom Thermal Testing

Due to the issues with the count rate during high temperatures on board 1, thermal testing was done on boards 2, 3 and 4 as well to verify they worked correctly at high temperatures. Thermal testing consisted of using an oven belonging to the Center for Particle Physics. Our testing procedure was to bring our payload up to 50°C and hold it there for approx. 1 hour. The results of this testing can be seen below in Figure 7 and Figure 8.

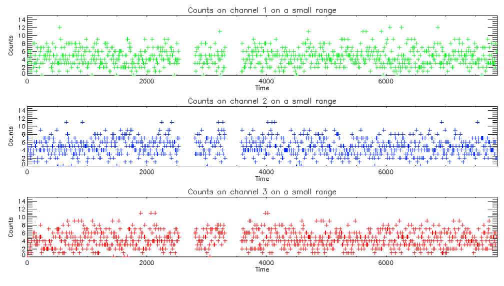


Figure 7: This graph shows the count rate during the custom thermal testing.

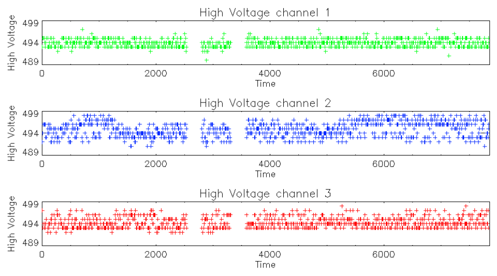


Figure 8: This graph shows the high voltage during the custom thermal testing.

However boards 3 and 4 didn't have the issues boards 1 and 2 experienced as you can see from figure Figure 7 and Figure 8. We believe this is because of the resistors used in the construction of boards 1 and 2. They were thermally sensitive and thus caused the counting errors at high temperatures. The same resistors were not used in boards 3 and 4 and thus the issue was not present. The gaps in the data in Figure 7 and Figure 8 are from some sort of issue with the serial readout program we used to record the data. It stamped the data taken at those points at with an incorrect time for about a period of 300s. During these two tests was the only time this issue was encountered and the cause for it was never determined, but it has never repeated the issue.

8.3 Weather Balloon Launches

The weather balloon launches conducted by the team provided us with the opportunity to launch and test our electronics in a near similar environment to flight conditions. The process of launching a weather balloon and conducting a successful weather balloon mission, prompted us to generate a set of documentation, see section 13 on Page 57. This document covers the procedures for a successful weather balloon launch and recovery. It includes checklists for materials and safe operating tips.

With the use of the weather balloons, we were able to launch our electronics into the atmosphere, running on a set of batteries, and collect some unshielded data from the three Geiger tubes attached. The data generated from our first flight, as seen in Figure 31 on Page 58. This demonstrates our data collected for this phase of the project. What this was able to demonstrate was that our board was able to function in the extreme weather conditions on a weather balloon high in the atmosphere.

9 Integration

Upon arrival at CSBF in Texas, the team unpacked the already shipped payload, and mounted its components on the HASP mounting plate as detailed in the PSIP document. Bringing the mounting plate with payload for integration certification, the team was notified of various issues with the payload.

1. Having two bolts on the rods below the mounting plate made the screws extend too far downward and prevented successful integration with the HASP platform
2. The payload lacked proper heat protection, and needed to be shielded to prevent overheating while the payload was waiting to be launched in the hot New Mexico sun

3. The data format of the payload was in binary, when it should have been formatted to output ASCII code

The above issues were resolved by the following:

1. Cutting excess bolt off till it would fit on the platform

2. Designing, and building a head shield out of aluminium foil and foam board

3. Rewriting the data output of the firmware to output in ASCII

After integration, a large round of stress testing on one of our electronics boards with identical hardware to the one we shipped to New Mexico for the launch. These tests consist of running the board for long durations of time, while uploading the data automatically to the web though a bash script intended to graph the data so the team can track whether the payload can run successfully for the entire duration of the flight and beyond.

The payload failed the first thermal vacuum integration test. It was hypothesized that the payload's electronics were discharging against the payload housing when it was under a certain pressure. This was corrected by utilizing electrical insulation tape called MLI. We additionally applied high voltage insulation on all junctions and exposed electrical components to prevent accidental discharge. This resolved the high voltage discharge issues, and allowed the board to maintain the minimum operating voltage of $\approx 500V$ for the Geiger tubes to function. HASP personnel gave the team the opportunity of a second vacuum test at the end of the week (Friday).

10 Launch

Upon arrival to CSBF in Fort Sumner, NM, team members unpacked the UA-HAB payload. The payload arrived in tact, in full integration form. The first task was to dissemble the aluminium foil / foam board box, and re-enforce it with silicon to weather-proof it. The box was cut back into its five faces, and reassembled, with a small hole near the bottom for thermal reasons. The entire payload, assembled, was then mounted to the HASP platform to test power and data transmission. No issues were detected with the payload, and it was left on the platform to collect a radiation background profile for Fort Sumner. Launch day then arrived, and the payload was launched successfully with the HASP platform.

The team had to wait for a few days before the weather conditions permitted a safe launch. Once the safe launch conditions were attained, the team was sent to watch the procedures carried out by the HASP personnel. This process only took a few hours from the team being notified that they were launching, before the HASP personnel had the platform in the air.

During flight, the payload reached an altitude of approximately 36 kilometres above sea level. The flight lasted ~ 20 hours. A few hours into the flight, the payload's high voltage began to fluctuate wildly, and counts escalated far above normal levels. For more information and for results, see sections 13, 11, 12, and 13.

11 Data Analysis

The first objective of data analysis was to verify measured results and account for possible environmental factors. The two factors investigated were the high voltage of each channel and the pressure experienced by the payload. We did not investigate the relationship between temperature and count rate because it was verified in early testing (see subsection 8.2) that the flight board (board 4) had no issues with temperature affecting the count rate.

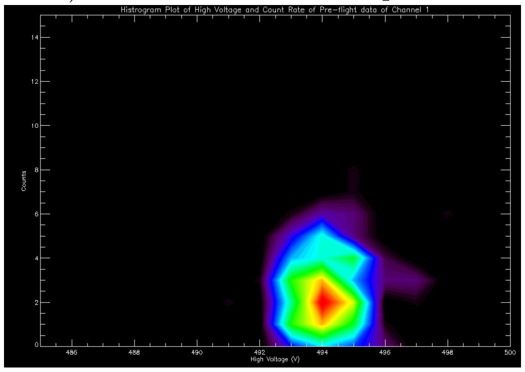


Figure 9: This graph was made using data from when the payload was sitting on the ground before launch. The colour scale is in arbitraty units with red being a higher frequency and blue being a lower frequency.

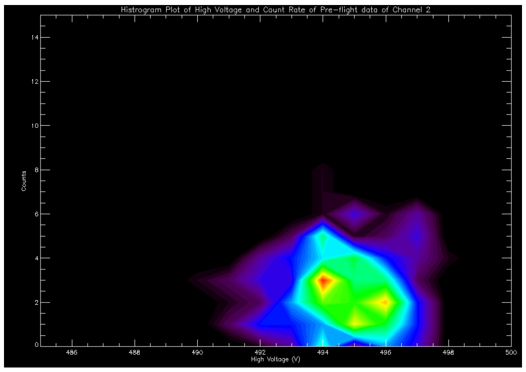


Figure 10: This graph was made using data from when the payload was sitting on the ground before launch. The colour scale is in arbitraty units with red being a higher frequency and blue being a lower frequency.

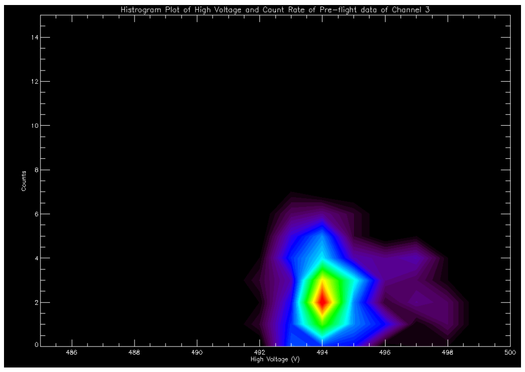


Figure 11: This graph was made using data from when the payload was sitting on the ground before launch. The colour scale is in arbitraty units with red being a higher frequency and blue being a lower frequency.

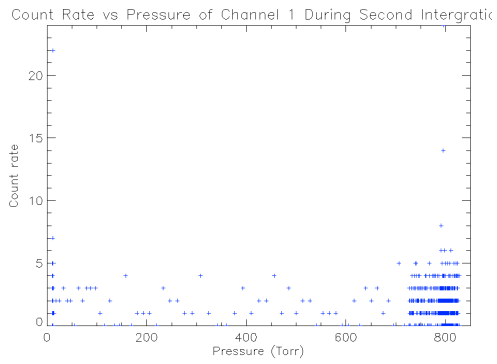
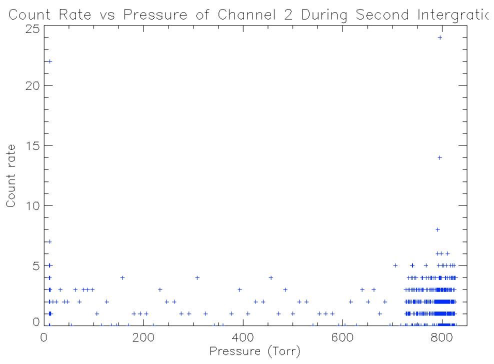


Figure 12: This graph was made using data from the second intergration before the payload malfunction.

A relationship between the high voltage of a channel and its count rate was investigated by determining the frequency of a count range at a specific high voltage. This was done over multiple data sets, such as preflight data and integration data. Essentially a 3-D histogram was plotted to display the various count rates at a specific high voltage value. There were no abnormal count rates corresponding to high or low high voltage values as can be seen from Figures 9, 10 and 11 and thus we concluded there is no relationship.



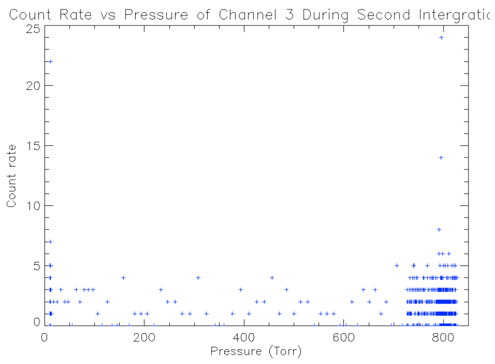


Figure 13: This graph was made using data from the second intergration before the payload malfunction.

Figure 14: This drawing details the full integration with the mounting platform from a different angle.

To investigate the possible relationship between pressure and count rate, we plotted the count rate versus the pressure during our second attempt at integration. As can be seen in figures 12, 13 and 14, there is no correlation between the two.

11.1 Data Collection Notes

When the team went to investigate the data that was recovered from the payload during the flight in New Mexico, we noticed that there was some data missing. Using data reconstruction, we were able to identify the amount of data lost during the flight that was due to transmission errors or from data writing errors during the payload's flight.

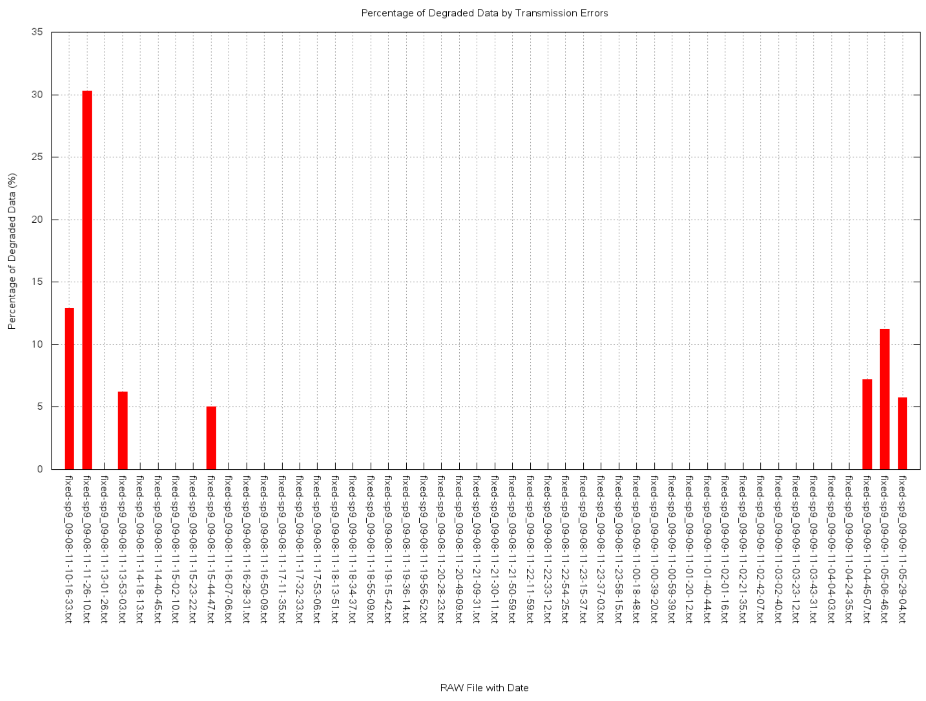


Figure 15: This illustrates the percentage of data that was lost due to transmission or data write errors.

Which properly documents the loss of data as a percentage for each of the data files that were collected as the amount of the file that was lost. The highest of these being $\approx 30\%$. Because of the nature of the data, we were not able to extrapolate the missing data, instead the gaps in the data were noted during the analysis.

12 Timing Issues

We can infer from the available data, and its associated analysis, that there was an issue in either the software or the hardware pertaining to the ability of our board to maintain the pre-determined rate of packet transmission.

12.1 Scientific Problem

By looking at the relationship between the data collected and the time elapsed for the whole time collection run, we initially calculated a difference in the two times. This prompted an investigation into the reliability of the timing data that was outputted by the payload's firmware during data collection.

Two theories were proposed. The first was that the HASP Platform encountered transmission errors during the payload flight, and therefore, lost data due to those errors. The second was that the board itself was having timing issues, and was not transmitting the packets at the pre-defined rate of once per every ten seconds of data collection.

12.2 A First Investigation, Flight Data

As the data is collected from the payload during the data collection phase, the HASP platform outputs the data into files with a pre-determined size. As the file fills up with output data from the payload, the platform cuts off the current data stream, stamps the output file (with the current file creation time), and opens a new data buffer. If we made the assumption that the time it takes for the HASP platform to switch to the new file after the previous one was filled up, then we can make the following inferences:

- The time in-between two files timestamps is the same amount of time that that file was left open to write data to, and therefore is equivalent to the time that the data file was collecting data
- During the time that the data is collected into the file, we can count the number of data packets received by comparing the initial and the final packet ID within the file to determine the amount of packets collected during the designated time period
- From the amount of time that the payload was able to output data to a specified file, and the amount of collected packets from within the file, we can make an estimate on the amount of time it takes to transmit each packet as a function of file

One additional consideration must be taken into account however. The HASP payload at some points did lose data due to what we are calling ”Transmission Errors”. Packets are considered to be ”In error” if they meet any of the following criteria:

- The Board ID, which is a known integer value at the beginning of each packet, is missing or incomplete

- The packet contained more data fields than was designed (e.g., the packet contained 14 fields of data while we only have 12)
- A packet ID number is missed entirely (e.g., the first packet has ID#12, the second has ID#14, hence the packet associated with ID#13 is missing)

To account and compare our theories, we reconstructed the missing data through computational means. The intent of reconstructing the data was not related to the actual data that was being reconstructed, but rather the existence of the packet was identified. The following logical flow proceeded the detection of missing packets and packet existence reconstruction on Page 29 as Listing 1. In essence, this code counts, when fed the data via *stdin*, was able to count the number of packets there were per file including those transmission errors.

To compare this reconstructed code with code that has been excluded via the logical flow detailed on Page 29 as Listing 2. We need to use the following command as seen on Page 29 as Listing 3. To count the remaining lines of data within the file, who has had its invalid data removed. Upon comparing this data with the time it took to generate these files, we were able to generate the Graph 19 in the Appendix Page 30. This detailed the transmission rates for the individual packet transmissions for the flight data.

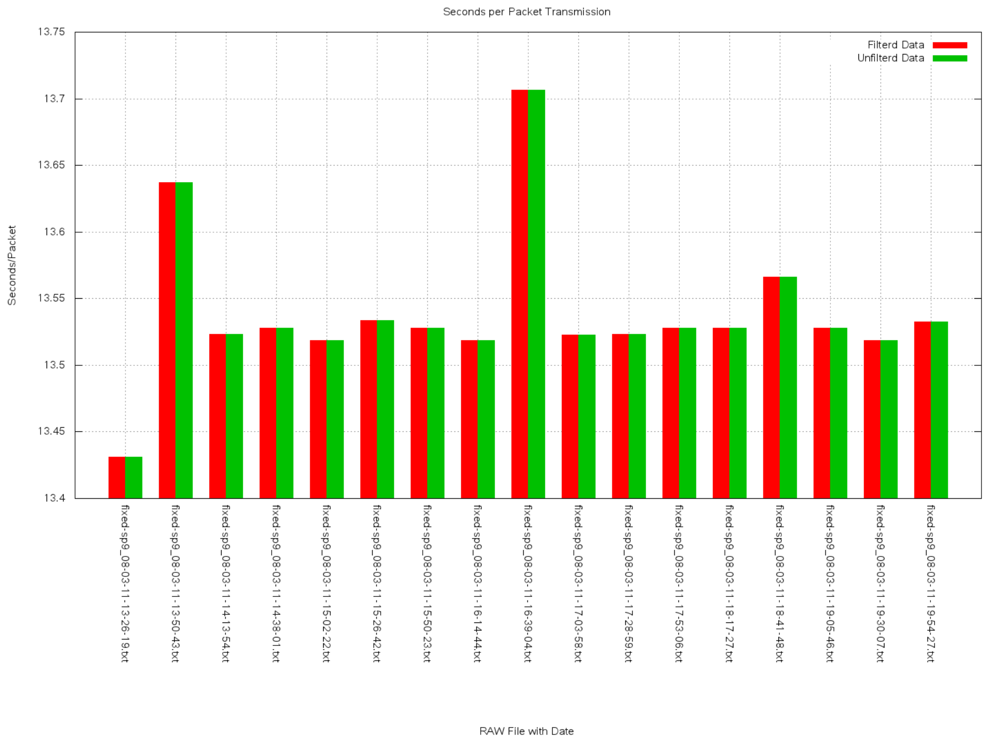


Figure 16: Histogram of Seconds per Packet Transmission Per file: First Integration

12.3 Conclusion of Timing Issues

What we determined from our data is that instead of the expected ten seconds per packet transmission, that we have approx 13 seconds per packet transmission. (see Figure 16 which shows the average time interval between packets). For instance, in 1260s 93 records were send from the payload, thus one record each 13.5s. The same board was used in the integration, thus to verify our results we performed the same analysis on the data collected during integration. This data also indicated an average of approximately 13s per record transmission. These results are completely consistent with the results discussed in subsection 7.2. Where the time interval between records was directly measured.

13 Scientific Results

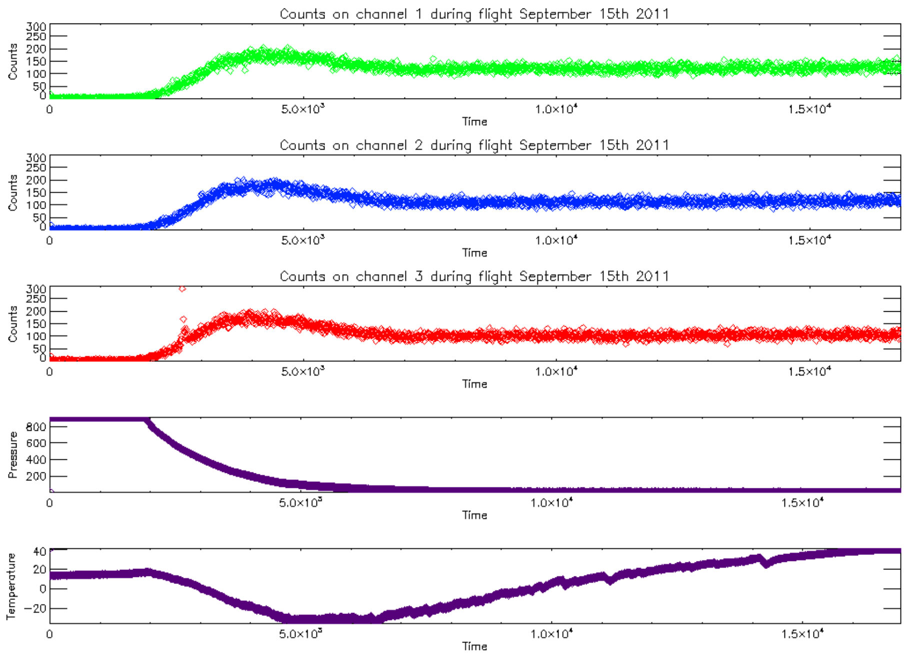


Figure 17: Data during flight before payload malfunction. Channel one, two and three correspond to the 50MeV, 100meV and 150MeV channels respectively

After our data from the flight was determined to be valid due to the analysis done in section 11, we were able to analyze it for scientific results. As can be seen from the flight data graphed in Figure 17, the count rates increase as the payload rises because of the increased level of radiation experienced. The fact that there is no discrimination between any one the channels shows that the radiation energy level is above our maximum energy threshold. There appears to be a small spike in the count rate on channel one while the payload is rising. We were not able to determine whether or not this was an event because it did not appear on the other channels and was of such small duration. It's likely it is glitch or random event.

The rise then decrease of the count rate seen at $\approx$ 400s to $\approx$ 600s was something we did not expect to see. It seems this occurred from approximately 40 000 feet to about 100 000 feet, with the peak being at 60 000 feet as seen in Figure 18. Radiation levels are predicted to increase linearly with altitude, so a peak at 60 000 feet is not something we are currently able to explain. While we will continue to attempt to explain this rise in counts, it may be worthy of future research, particularly because this close to the region commercial aviation uses.

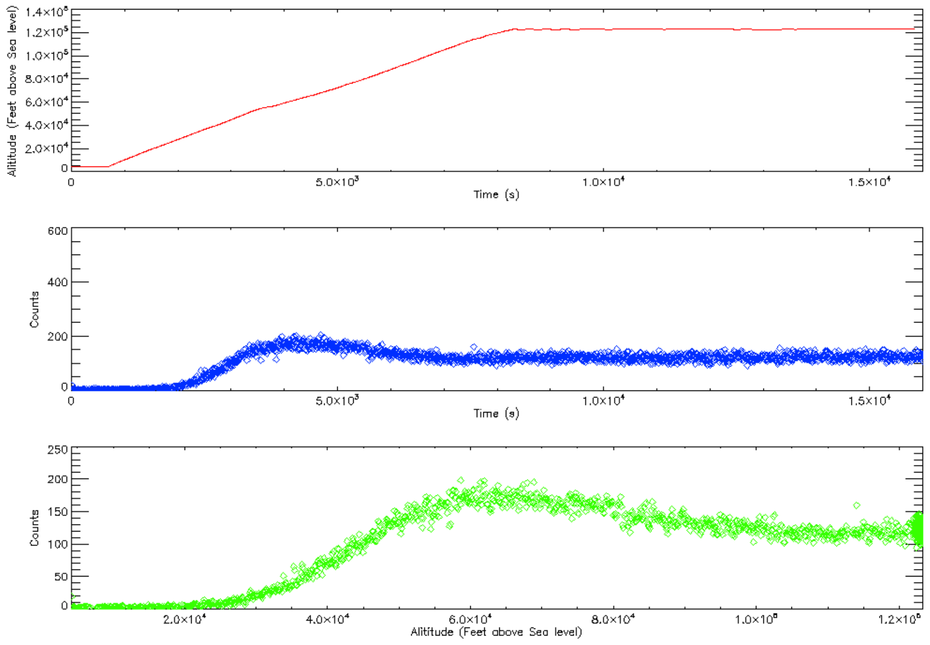


Figure 18: The first graph is the altitude over the flight, the second graph is the count data from channel 1, and the third graph is a plot of counts vs altitude. The third graph was created by interpolating the count data over the altitude data.

University of Alberta Non-Conformance Review

Review Panel:

Prof. Ian Mann

Dr. I. Jonathan Rae

Mr. David Miles

Attendees:

Andreas Buttenschoen (via teleconference)

Cory Hodgson

Laura Mazzino

Quinton Farr

Wyatt Johnson

Known Faults and Issues

Major Fault 1: Loss of two of three Geiger Channels

The major fault which triggered this review was the loss of two Geiger channels during integration testing and the subsequent failure of the integration test. The review board accepts the proposal of the UAHAB team that the technical cause of this fault was:

1) High-voltage leakage into the metal housing (this not having been tested before integration)
2) Caused a reduction in the high-voltage supplies
3) Which caused anomalous Geiger tube behaviour when the HV supply dropped below the operating threshold.

This explanation is found to be consistent with the observed behaviour and with the field repair where exposed surfaces were insulated with insulating tape and corona dope. The review board accepts that this field repair has a reasonable chance of prevent a re-occurrence of the major fault during flight. Finally, since physical access to the payload is apparently no-longer possible the board accepts that no further hardware modification is possible.

Major Fault 2: Ghost Counts (Escalated by Review Board)

"Ghost Counts" is a term used by the UAHAB team to refer to periods when the one or more channels suddenly. Potentially anomalous behaviour is observable in Channel 2 during the first thermal test. When the count rate becomes very high, the reporting interval for the CPU becomes longer.

See recommendation later.

Major Fault 3: HV Droop During Thermal Test 2 (Escalated by Review Board)

Team to re-examine cause of HV droop and produce short email.

Major Fault 4: Unknown board makeup (Escalated by Review Board)

Informal at this point. Boards 3 and 4 were assembled from parts from the same batch. Team to document any differences between 3 and 4 and only test on these boards.

Minor Fault 1: Incorrect Data Format (Binary vs Ascii)

Either data forms are compliant. Ascii was preferred. Potentially increases the load on the CPU. No change recommended.

Minor Fault 2: Payload Potentially Vulnerable to Solar Heating

No shorting or interference is believed to be possible. Insulating tape was rigourously applied. Acts as insulation and may keep payload at a higher temperature.

Note: The current parallel test board is not housed in a metal casing or in a thermal tent.

Suggest that the parallel universe test board be placed in a crude foam box which is representative of the flight configuration? Suggest that metal blocks be placed around at least one Geiger channel.

Minor Fault 3: Payload Volume Violation (Mounting Rods too long)

Modified mounting believed solid. The interface specification and instrument design are correct. However, the as-built instrument did not meet these requirements. A design review should have caught the issue earlier. An internal integration test against the HASP documentation would likely have caught this issue.

Minor Fault 4: Instrument damage during integration

Monday August 1: Bolts stuck on mounting rods. While team members were at store procuring lubricant a team member attempted to release the rods by striking them with a hammer which damaged the rods.

Tuesday August 2: Power cable ripped off, repaired using soldering tools borrowed from other teams.

The power cable was sheared so damage to any other part of the payload is unlikely. The payload had been internally powered up till this point. It is possible the cable had existing damage but was never used and so never detected. A full internal integration test would likely have diagnosed such an issue. Power cable was multimeter tested the weekend before shipping for integration. The repaired cable was also tested with a multimeter (power only).

Recommendation: Budget for and create a field repair kit which should be created and should travel with the team.

Contributing Circumstances

Late Entry Into Proposal and Payload Concept

Short timeframe led to photographic plates which wasn't the final design choice.

Late Entry into Design and Build Phase

Consideration of multiple instrument types and topologies delayed the start of major design work.

Schedule Rush Before Integration

The housing was not completed till ~day shipping cut-off

No Test Integration

Payload had never been operated, even on a bench, fully assembled.

No Test Mounting or Insufficient Test Mounting

Payload had never been test mounted to UAHAB flight plate. How was the envelope violation missed.

Unknown Status and Composition of flight Payload

Is there a recording of what the machine housing is actually shaped / constructed?

Inside surfaces are well checked. External surfaces are uneven but issue is only cosmetic. Internal surfaces match the design so there is no need for an "as-built"

Is there a recording of what surfaces were taped?

All internal surfaces and all contacting surfaces are believed to be taped. External surfaces are not necessarily all taped. Photographs document this process.

Is there an accurate record of how each PCB is assembled?

Board 3 and board 4 are identically assembled.

Unclear Leadership Structure and Roles and Responsibilities

Training: Project Management course. Consider teambuilding classes or activities? Lay out expectations for how decisions are made, how communication occurs (ie. Wait for team meetings, email, knock on a door). Also, what questions should be made by the individual, by the team or raised up the chain.

No clear tracking process for the status of major tasks

Poor communication within the working team and up the management chain

Weekly meeting among undergraduates was very useful. Suggest a weekly meeting of the working team so that all members are aware of the status of the project, the major tasks, and upcoming tasks and milestones.

Recommendations for UAHAB

Major: Change Geiger Counting to Polled from Interrupt Driven

The current software appears to be vulnerable to CPU time over-runs due to high count rates on the three interrupts/input pins used to capture pulses from the Geiger tubes. High pulse rates could potentially be caused by HV supplies below the Geiger threshold, fast transients on the HV due to poor HV regulation and a poorly designed pulse shaping circuit. It is no-longer feasible to modify the hardware; however, it should not be possible for high edge rates on these inputs to starve CPU time from other essential functions such as data transmission (eg. The extended 10second data report cadence during high count rates) and the potential loss of HV regulation.)

The reported count rates on the order of 2x10^4 on three channels would leave approximately 3686400 (microcontroller clock rate) / (3 x 2x10^4) = ~60 cycles per interrupt. Given the overhead of an ISR (pushing and popping all registers, stack pointers, etc) this suggests that the microcontroller would be entirely occupied servicing interrupts and would have no time to execute any other tasks.

I suggest disabling all three interrupts and treating the pins as flagged interrupts. Create a timer compare interrupt at, for example, 1000 interrupts per second. Have this interrupt test and clear the edge flag for each input and increment the three timers as appropriate.

The benefits of this are:

1) No longer possible for high edge rates to starve the microcontroller of CPU time as the interrupt rate is fixed.
2) Sampling period no longer dependant on count rate.
3) Fixed and obvious maximum count rate to tell when the instrument has hard saturated.

The disadvanages are:

1) Limited maximum count rate.
2) Multiple pulses within 1 ms will be undercounted.

Major: Document how human in the loop intervention will occur if the payload enters a non-operating state.

How is the decision made. What is the process and who is consulted.

Recommendations For Future Projects

Major: Creation of Internal formal Reviews

Kick-Off, Design Review, Interim Review, Pre-Ship/Flight Readiness

Major: Re-structuring of Team Layout with Clear Roles and Lines of Responsibility

Major: Establish Triggers to Escalate Issues to Team Supervisors

Major: Instigate Buddy/Mentor Reviews for Major Design Changes

Major: Create and Maintain a bug/task tracker system

Minor: Dedicate a central workspace and storage space for the project

Minor: Creation and Maintenance of "As-Built" Schematics

Minor: Creation and Maintenance of Changelog Sheets for Individual Instruments

Minor: Each Team Member should maintain a project journal / lab book,

Minor: Technical and Professional Skill Development for Participants

Techical skills, management skills, team skills etc.

Minor: Limit the technical role of the Project Manager and/or Team Leader to ensure sufficient time and focus.

- Consider combining the roles of Project Manager and Team Leader

Major: Implement Watchdog Circuit

Determine the minimum activities needed for a "functioning" instrument and use the watchdog to trigger a software reset if these activities do not occur for a predetermined time period.

Appendix

Timing Issue

```
uNEW = packetid;

 if(  (uOLD+1) != uNEW ){
          if (uOLD>uNEW) {
                    if (uNEW == 1) {
                              truePID++;
                              }
                    else if (uNEW != 1) {
                              truePID += uNEW;
                              }
                              else {
                                        printf("Failure - Packetid(%li) - uOLD(%li) - uNEW(%
                                            li)\n",packetid,uOLD,uNEW);
                                        return 1;
                                        }
                                        }
                    else {
                              truePID += uNEW - uOLD;
                              }
                    }
                    else
                              truePID++;
          uOLD = packetid;
```

Listing 1: Packet Reconstruction Code

```
//  Verify nominal result
               if(boardid != 3091) continue;
               else linenum++;

               // Turn corrections on or off
               if (false) {

                    if (linenum >=10) {
                         // Process ID corrections
                         if (packetid < 10) {
                              RPID = OLDPID;
                         }
                         //  Process time corrections
                         if (time < 100) {
                              RTIME = OLDTIME;
                         }
                    }
                    time = time + RTIME;
                    packetid = packetid + RPID;
                    OLDPID = packetid;
                    OLDTIME = time;
```

Listing 2: Packet Cleaning Code

```
lines=`cat ${filename} | wc -l`
```

Listing 3: Packet Counting Code (BASH)

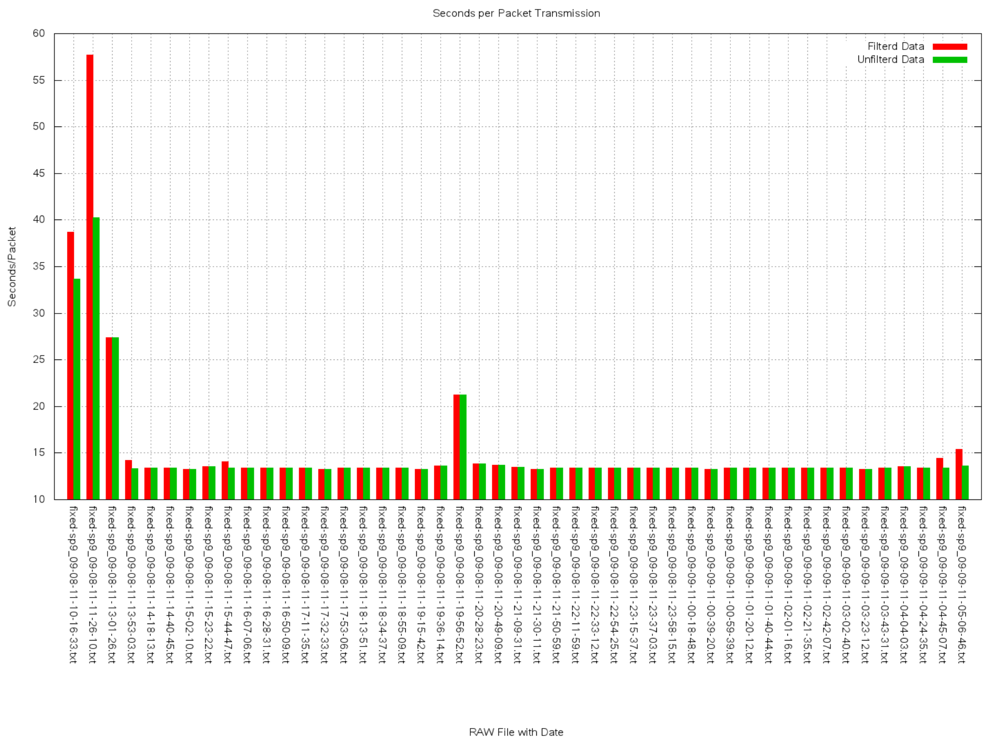


Figure 19: Histogram of Seconds per Packet Transmission Per file: Flight

Mechanical Design

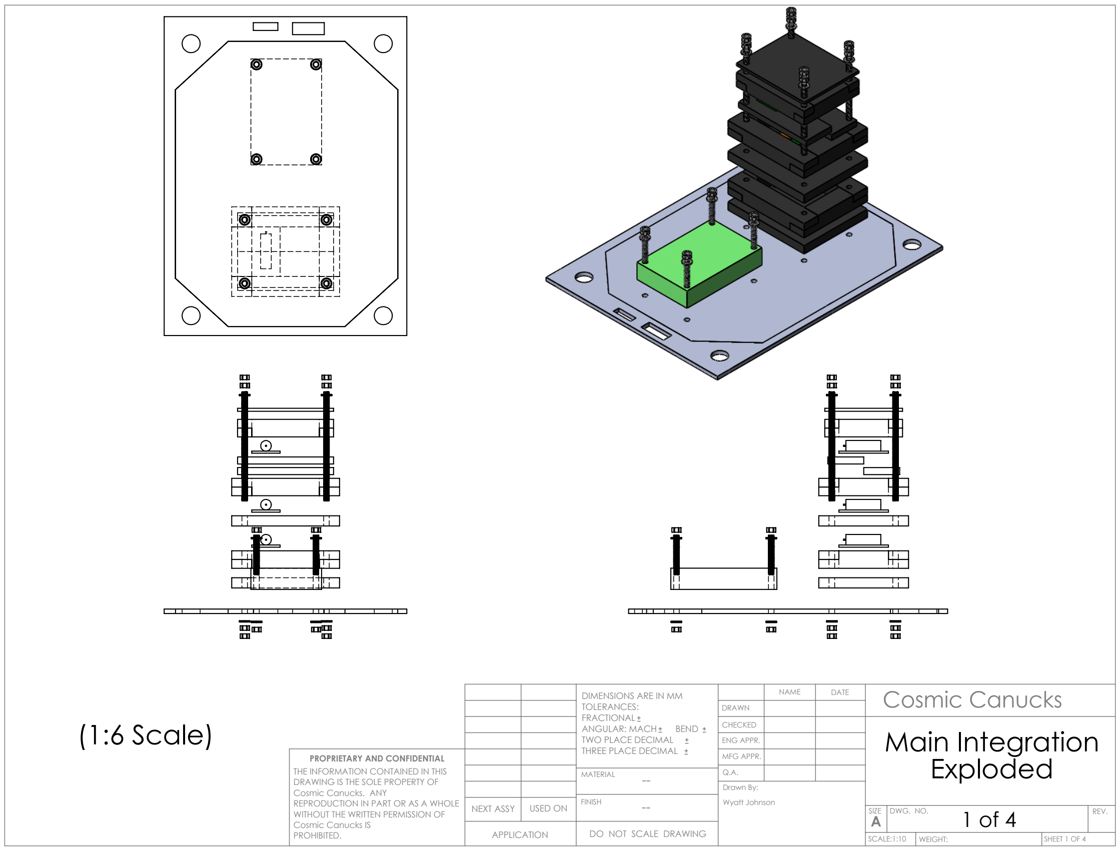


Figure 20: The main integration drawings feature the payload as it was mounted on the HASP mounting plate. Attached with bolts and nuts, it ensured that the payload was properly secured during the flight.

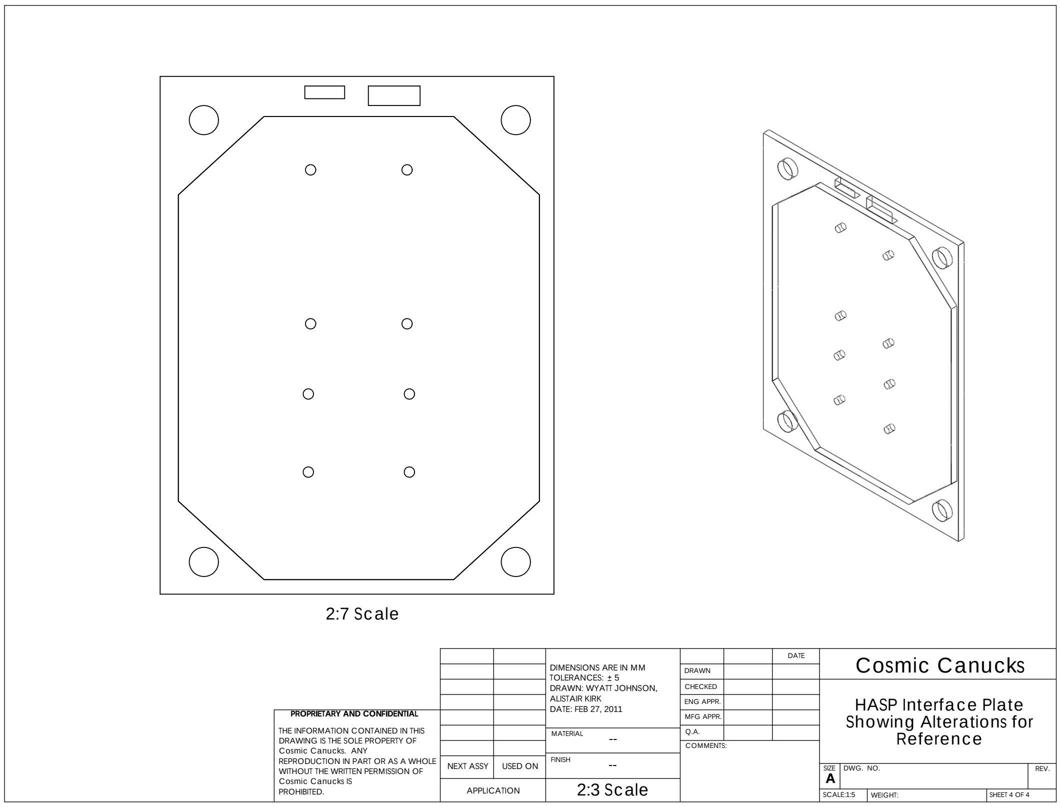

2:7 Scale
DIMENSIONS ARE IN MM
TOLERANCES: ± 5
DRAWN: WYATT JOHNSON, ALISTAIR KIRK
DATE: FEB 27, 2011
MATERIAL --
FINISH --
2:3 Scale
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF Cosmic Canucks. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF Cosmic Canucks IS PROHIBITED.
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
Cosmic Canucks
HASP Interface Plate Showing Alterations for Reference
SIZE A
DWG. NO.
REV.
SCALE:1:5
WEIGHT:
SHEET 4 OF 4

Figure 21: The HASP Mounting plate, with holes for our payload, is the connection in-between the platform and our payload for both electronic and power connections.

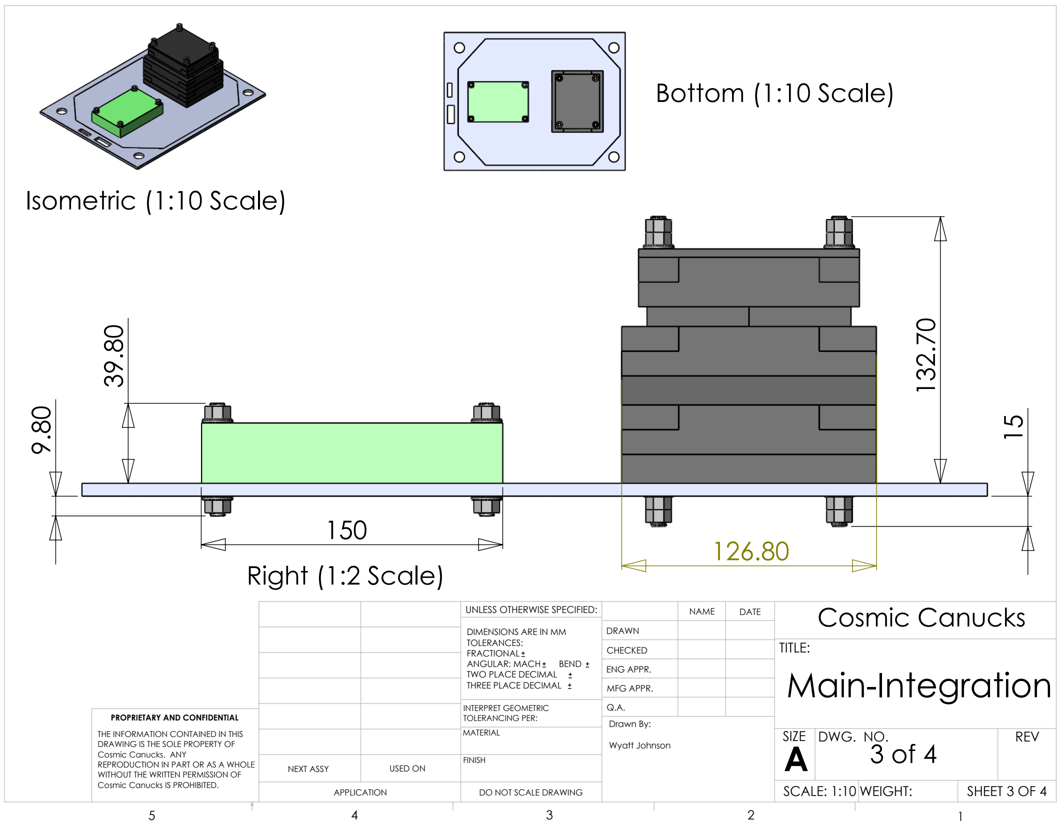
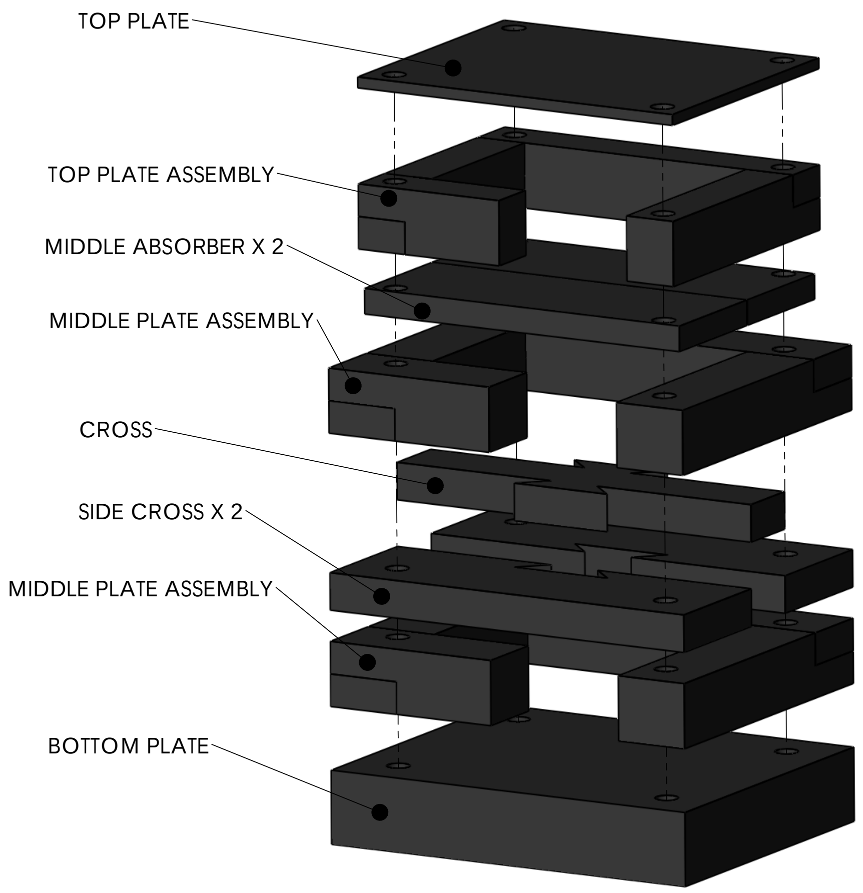


Figure 22: This drawing details the full integration with the mounting platform from a different angle.



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

COSMIC CANUCKS

DEBUR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

REVISION

DRAWN BY: WYATT JOHNSON

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

Main Exploded

MATERIAL:

DWG NO.

9 of 9

A4

WEIGHT:

SCALE:1:5

SHEET 9 OF 9

Figure 23: This drawing is the main steel housing in an exploded view.

Additional Graphs

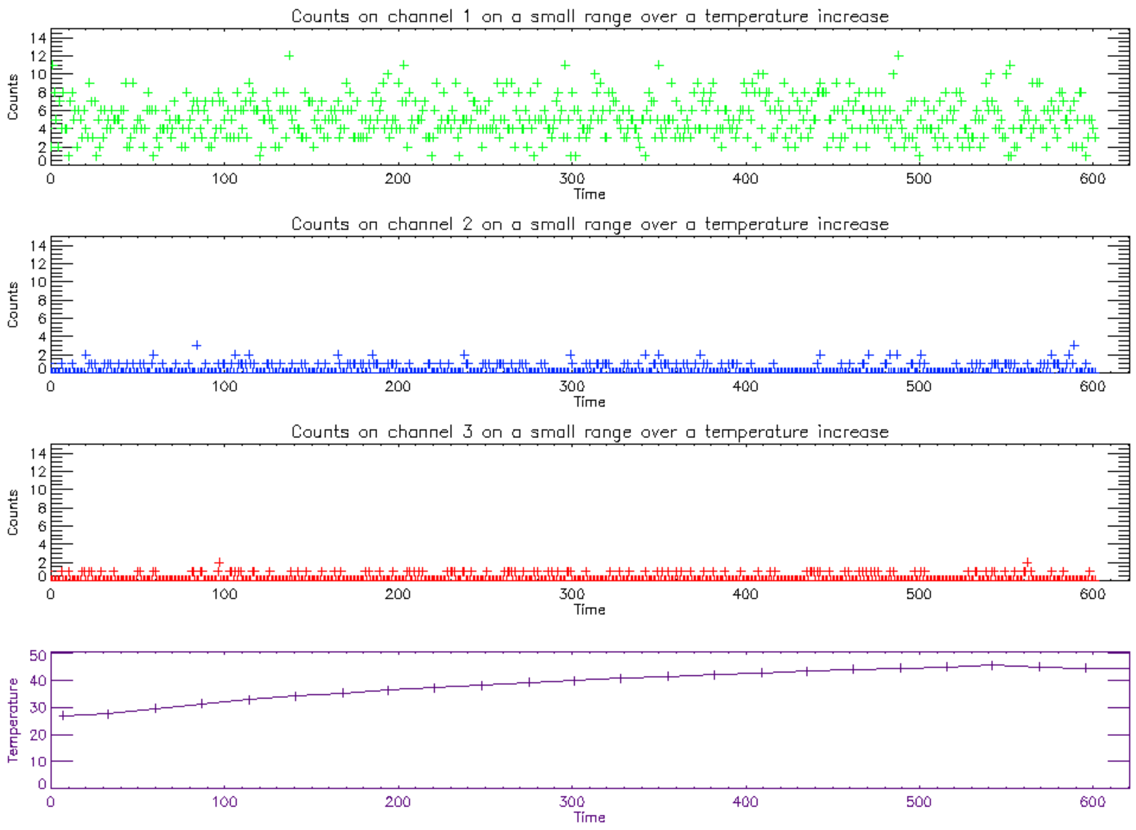


Figure 24: This graph shows the count rate during the custom thermal testing. As can be seen on the graph, board 2 experinced issues at temperatures similar to board 1

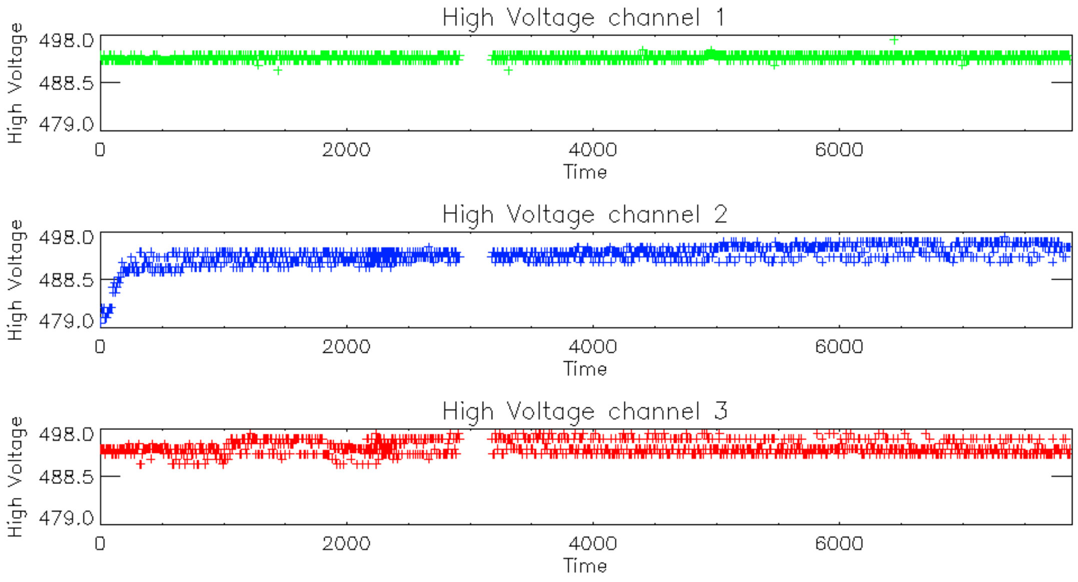


Figure 25: This graph shows the high voltage during the custom thermal testing.

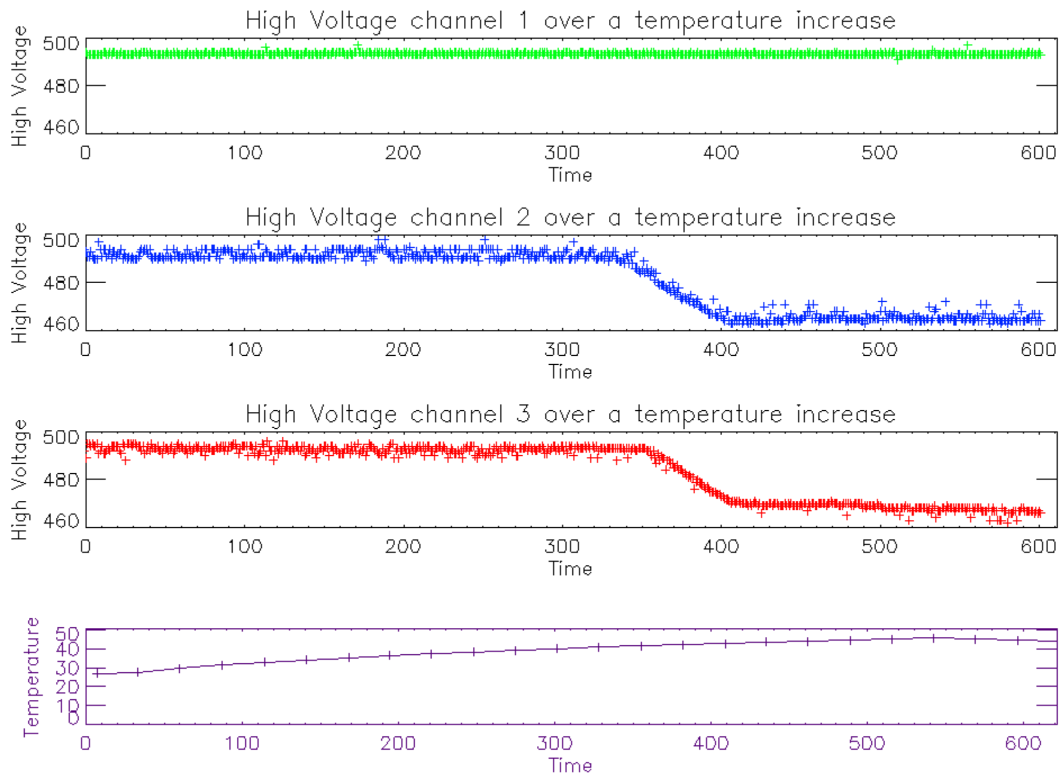


Figure 26: This graph shows the high voltage during the custom thermal testing. As can be seen on the graph, board 2 experinced issues at temperatures similar to board 1

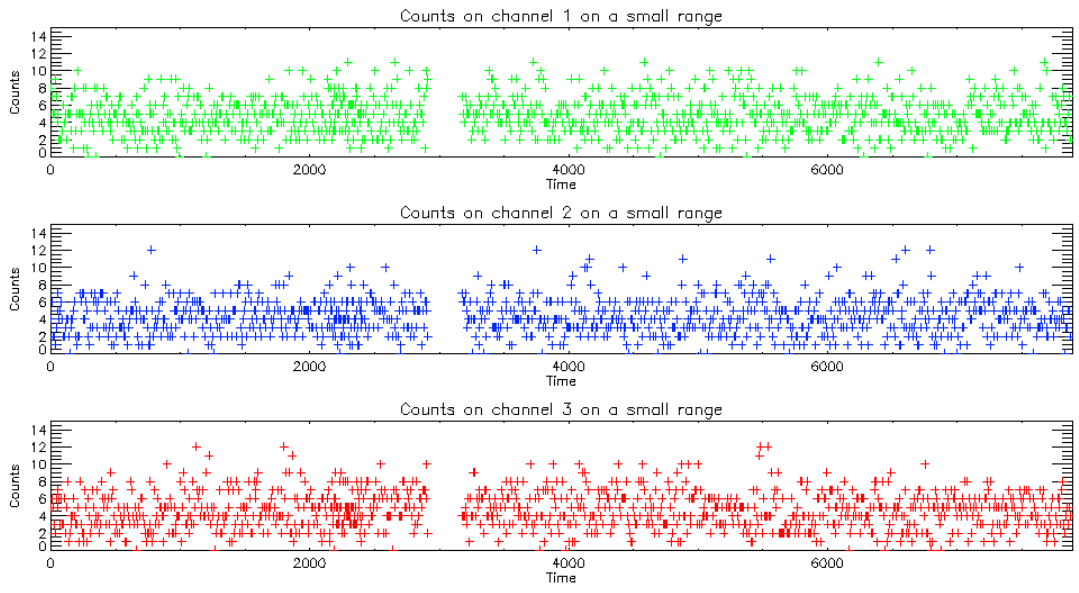


Figure 27: This graph shows the count rate during the custom thermal testing.

Design Appendix

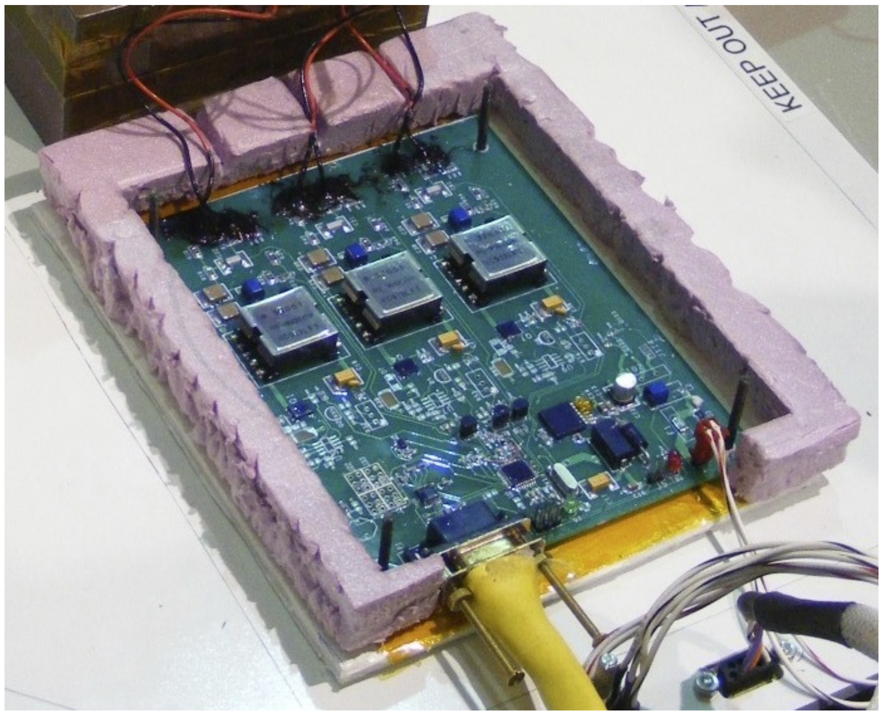


Figure 28: Foam PCB enclosure without top

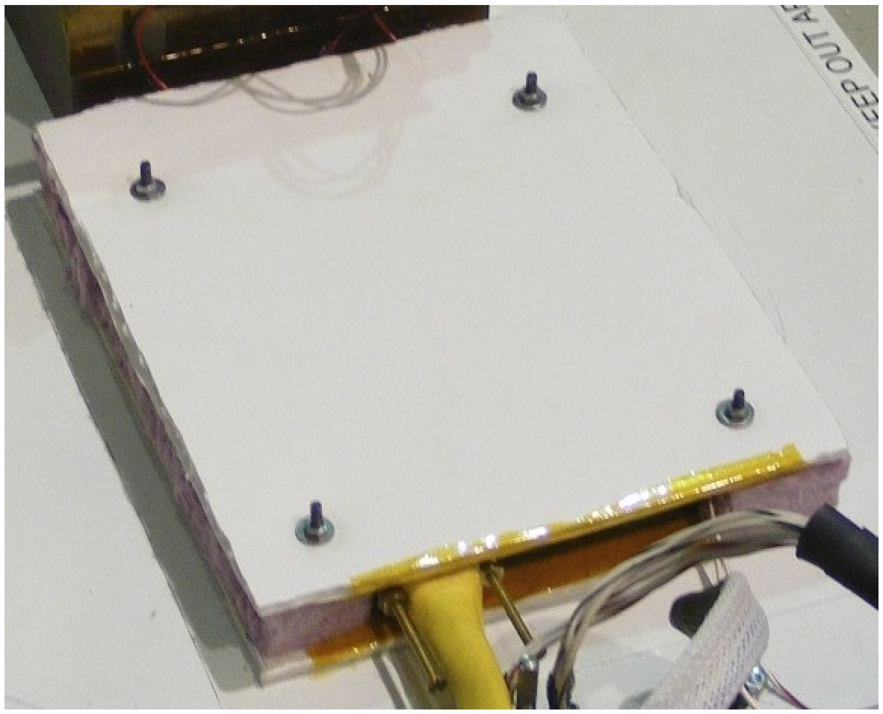


Figure 29: Aluminum Heat Shield

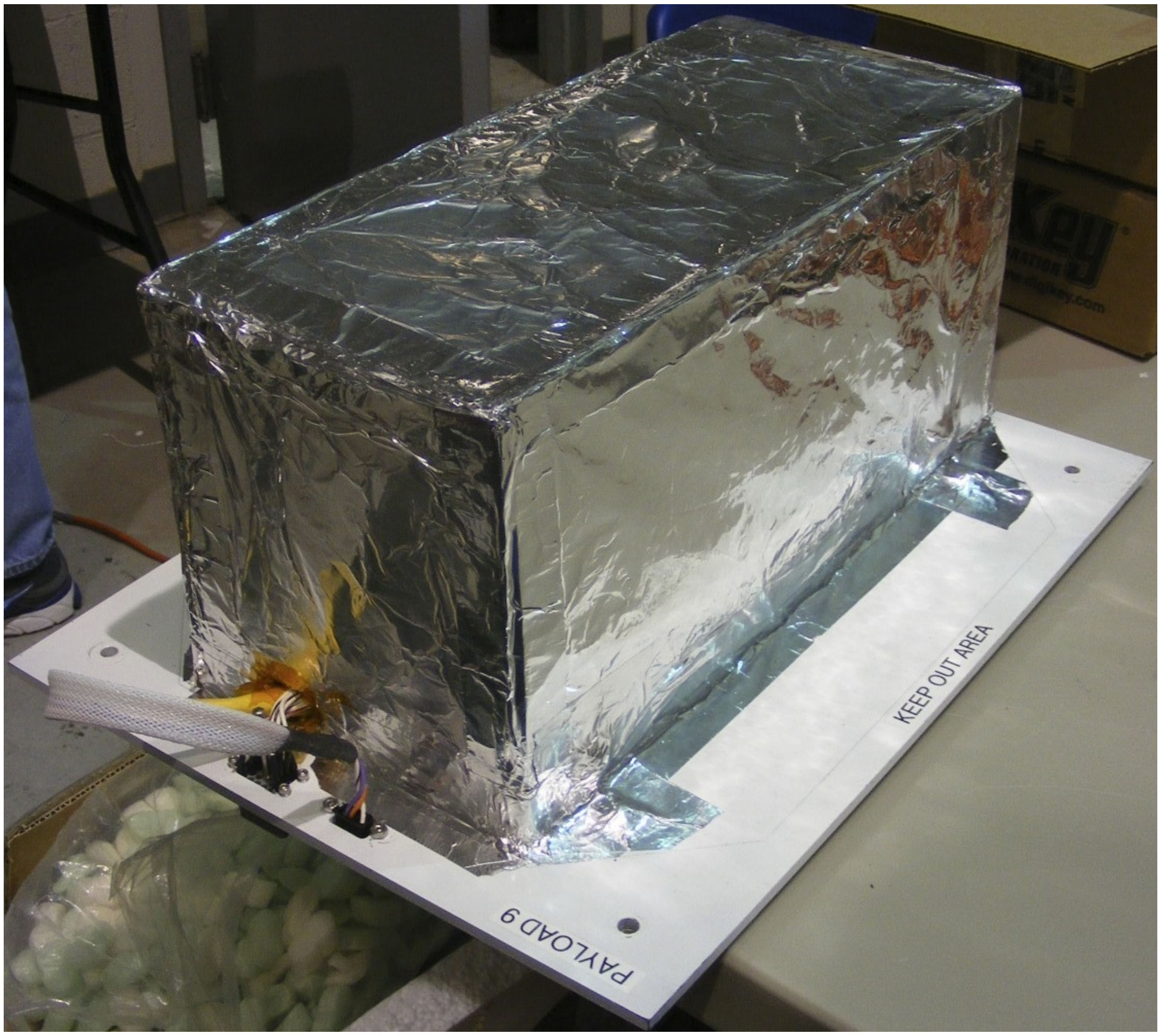


Figure 30: Closed foam PCB enclosure

Firmware Documentation

Code Documentation UA-HAB

Maple Leaf Particle Detector

Version: 1.6

Andreas Buttenschön
Cory Hodgson, Laura Mazzino, Wyatt Johnson
and Quinn Farr

Department of Physics
University of Alberta
Edmonton AB T6G 2G7
Canada

July 7, 2011

Revision: 0.1

Table of Contents

1 Introduction

The UA-HAB PCB has all the measuring tools required to measure temperature, pressure, and counts of three Geiger-Mueller tubes. All these sensors are controlled by a ATMEL-328P (CPU). The CPU, runs the UA-HAB firmware. The firmware code is divided into different subsystems, each representing a physical subsystem of the CPU. The different subsystems are described in section 5. UA-HAB is interested at obtaining the flux of protons at altitudes of 30km, additionally atmospheric data is also obtained (temperature & pressure), the data acquisitions are outlined in section 2.

2 Data Acquisition

There are three types of data, that UA-HAB is interested in, counts, temperature and pressure. The counts are measured using three Geiger-Mueller tubes each connected to one of the available External-Interrupts of the CPU see section subsubsection 5.5.5 for more detail. The temperature is measured using a T101, manufactured by Texas Instruments, which is connected to the I2C bus of the CPU see sections subsection 5.4 and subsection 5.6 for more details. The pressure is measured using a MPXA6115 pressured sensor, which is connected to one of the ADC (analog-digital-converter) pins of the CPU see **??** for more detail.

3 Source code organization

The firmware is divided into the following source code files.

Filename	Path	Description
`main.c`	`/main/`	The main program (see 5.0.1)
`main.h`	`/main/`	(see 5.0.1)
`general.h`	`/main/`	General program constants (see 5.0.2)
`interrupts.c`	`/main/`	Interrupts
`interrupts.h`	`/main/`	Interrupts header
`rx.c`	`/main/`	RX-code
`rx.h`	`/main/`	RX-code header
`util.c`	`/main/`	Firmware Utilities
`util.h`	`/main/`	Firmware Utilities header
`packet.h`	`/main/`	Firmware packet definitions
`time.c`	`/main/`	Timer code (see 5.1)
`time.h`	`/main/`	Timer code constants and function headers (see 5.1.2 & 5.1.4)
`uart.c`	`/main/`	USART code (see 5.2)
`uart.h`	`/main/`	USART code constants and function headers (see 5.2.2 & 5.2.3)
`twi.c`	`/main/`	TWI code (see 5.4)
`twi.h`	`/main/`	TWI code constants and function headers (see 5.4.2 & 5.4.4)
`temperature.c`	`/main/`	TMP101 driver code (see 5.6)
`temperature.h`	`/main/`	TMP101 code constants and function headers (see 5.6)
`mpxa6115a.h`	`/main/`	Pressure sensors header
`m24m01r.c`	`/main/`	Driver for `M24M01` eeprom
`m24m01r.h`	`/main/`	Header for `M24M01` driver
`serial.c`	`/tools/`	Utility of `UNIX` like OS for writing/reading to payload

Table 1: Program Files

4 Packet formats

There are two packet formats, one for sending data the other for receiving data. Both formats are defined as structures in `packet.h`.

4.1 Tx packet format

The current data format of the firmware (version 1.6) is,

Byte	Bits	Type	Description
1-2	16	`uint16_t`	Board ID
3-4	16	`uint16_t`	Packet ID
5-6	16	`uint16_t`	HV1
7-8	16	`uint16_t`	HV2
9-10	16	`uint16_t`	HV3
11-12	16	`uint16_t`	T1
13-14	16	`uint16_t`	T2
15-16	16	`uint16_t`	T3
18-21	32	`uint32_t`	seconds since sys start
22-23	16	`uint16_t`	msecs since last second
24-25	16	`uint16_t`	temperature
26-27	16	`uint16_t`	pressure
28-32	16	3× `uint16_t`	reserved
33-34	16	`uint16_t`	CRC checksum
35	8	`uint8_t`	`NL`
total 35 bytes.			

Table 2: Tx Data Format (version 1.6)

The RX-packet is defined in `packet.h` as follows.

```
typedef struct {
        uint16_t id;
        uint16_t packet_id;
        uint16_t HV1, HV2, HV3;
        uint16_t T1, T2, T3;
        uint32_t tr_sec;
        uint16_t tr_msec;
        int16_t temperature;
        uint16_t pressure;
} packet_t;
```

4.2 Rx (receive) Data

In versions 1.6receiving support is implemented, and packet format is according to HASP suggestion & specifications.

Byte	Bits	Type	Description
1	8	`uint8_t`	Start of heading (`SOH`)
2	8	`uint8_t`	Start of Text (`STX`)
3-4	16	`uint16_t`	Command
5	8	`uint8_t`	End of Text (`ETX`)
6	8	`uint8_t`	Carriage Return (`CR`)
7	8	`uint8_t`	Line Feed (`LF`)
total 7 bytes.			

Table 3: Rx Data Format (version 1.6)

The TX-packet is defined in `packet.h` as follows.

```
typedef struct {
        uint8_t start;
        uint8_t start_status;
        uint16_t command;
        uint8_t end_text;
        uint8_t CR;
        uint8_t NF;
} uplink_cmd_t;
```

5 Subsystems

The following subsection contain a detailed descriptions of all the subsystems in the UA-HAB firmware.

5.0.1 Subsystem Overview

The firmware starts running in `main.c`, which initializes all the required subsystems and variables. Following this the firmware enters a endless loop, that controls the creation and sending of packets.

5.0.2 General Constants

All the following are defined in `general.h` or `main.c` (for `main.c` usage only).

`FOSC`	frequency external quartz
`F_CPU`	CPU frequency
`LED`	LED on/off
`HEATER`	heater on/off
`MAIN_FAILURE`	failure in `main.c`
`MAIN_OK`	everything ok in `main.c`
`PAYLOAD_ID`	Payload ID `0x09`
`TRUE`	boolean constant true (`0x01`)
`FALSE`	boolean constant false (`0x00`)

5.0.3 Data storage

There are several external variables, defined in `general.h`. These three external variables are used as a status register of the packet build and send process (see table **??** for usage details).

The external variables are defined in `general.h` as follows.

```
extern uint8_t send_packet;
extern uint8_t prepare_packet;
extern uint8_t packet_ready;
extern uint8_t rx_data_available;
extern volatile unsigned int ADC0_value;
extern volatile unsigned int ADC1_value;
extern volatile unsigned int ADC2_value;
extern volatile uint16_t T1;
extern volatile uint16_t T2;
extern volatile uint16_t T3;
extern volatile uint16_t pressure;
```

Note that they have to be declared in each file they are used, for example.

```
uint8_t send_packet; // redeclare external variable
```

Details External variables

Name	Data-type	Initial value	Description
`send_packet`	`uint8_t`	`FALSE`	if `TRUE` send next packet
`prepare_packet`	`uint8_t`	`FALSE`	if `TRUE` prepare next packet
`packet_ready`	`uint8_t`	`FALSE`	if `TRUE` packet ready for send
`rx_data_available`	`uint8_t`	`FALSE`	if `TRUE` rx packet available
`ADC0_value`	`volatile uint16_t`	`0`	HV of channel 1
`ADC1_value`	`volatile uint16_t`	`0`	HV of channel 2
`ADC2_value`	`volatile uint16_t`	`0`	HV of channel 3
`T1`	`volatile uint16_t`	`0`	Counts channel 1
`T2`	`volatile uint16_t`	`0`	Counts channel 1
`T3`	`volatile uint16_t`	`0`	Counts channel 1
`pressure`	`volatile uint16_t`	`0`	pressure measurement

Table 4: External variables

5.1 Timer0 (8bit-Timer)

5.1.1 Overview

The current timer (version 1.1), previous versions did not feature any timer support, runs at a 500µs resolution with and error of 6×10^{-7}s. A data structure associated with the timer keeps track of the current time (see section 5.1.3).

5.1.2 Constants

The following constants are defined in `time.h`.

`TIMER_STEPS` Timer0 resolution

5.1.3 Data storage

The time structure is to keep track of time since system boot, it is defined in `time.h` as follows (see table 5 for a detailed description).

```
struct time {
        uint16_t msec;
        uint32_t seconds;
        uint8_t timeout_ticks;
        uint32_t ticks;
};
```

Details Time Structure

Name	Data-type	Initial value	Description
`msec`	`uint16_t`	0	ms since last second
`seconds`	`uint32_t`	0	seconds since sys boot
`ticks`	`uint32_t`	0	ticks since sys boot
`timeout_ticks`	`uint8_t`	0	ticks since timeout call

Table 5: Time structure

5.1.4 Functions & Macros

The following function headers and macro definitions can be found in `time.h`.

`TICKS_TO_MS(ticks)` return `ticks` in ms

`MS_TO_TICKS(ms)` returns `ms` in timer ticks

`TIMER_GET_TICKCOUNT` return timer ticks since sys boot

`TIMER_GET_TICKCOUNT_8` returns timer ticks since sys boot in `uint8_t`

`TIMER_GET_MSEC` return ms since last second

`TIMER_GET_SEC` return seconds since sys boot

`void Timer0_init(void)` Initialize Timer0

`uint32_t timer_get_tick(void)` Returns number of ticks since system start

`uint16_t timer_get_msec(void)` Return ms since last full second

`uint32_t timer_get_seconds(void)` Return seconds since system start

`uint8_t timeout_ms_passed(uint8_t * ticks, uint16_t ms)` Return `TRUE` when `ms` have passed, `FALSE` otherwise. If `TRUE` then writes current `ticks` to `* old_ticks`

`uint8_t timer_ms_passed(uint32_t * ticks, uint16_t ms)` Return `TRUE` when `ms` have passed, `FALSE` otherwise. If `TRUE` then writes current `ticks` to `* old_ticks`

`void reset_timeout(void)` Reset the timeout (`timeout_ticks = 0`)

`uint8_t timeout_get_ticks(void)` Return `timeout_ticks`

5.1.5 Interrupt Routine

The Timer0 causes a system interrupt each time the 8-bit tick counter overflows. Each interrupt will increment the `ticks` and `msec` variables of the time structure `t0` (see Table 5). Every 2000 ticks, the `msec` variable is reseted to 0, and the `seconds` variable of the time structure is incremented by 1 (see Table 5).

In C code the Timer0 overflow interrupt looks as follows.

```
ISR(TIMER0_OVF_vect)
{
        ++t0.msec;
        ++t0.ticks;

        if (t0.msec == 2000) { // after 1 second
                ++t0.seconds;
                t0.msec = 0; // reset
        }
}
```

5.2 USART - Universal Synchronous and Asynchronous serial Receiver and Transmitter

5.2.1 Overview

The USART, as of version 1.0 provides the facilities of transmitting data bytes over a serial connection. Currently, the USART sends data at a baud rate of 9600, in asynchronous mode, with no parity, 1 stop bit and a char size of 8.

5.2.2 Constants

The following constants are defined in `usart.h`.

`BAUD` serial port baud rate

`MYUBBR` `UBRR` (USART Baud Rate Register) sets USART clock

5.2.3 Functions & Macros

The following function headers and macros are defined in `usart.h`.

`void USARTInit(void)` initialize USART

`static unsigned char USARTReadChar(void)` returns next character from `UDR0`

`void USARTWriteChar(unsigned char data)` write character `data` to `UDR0`

`uint8_t USARTWrite(uint8_t * data, uint8_t txSize, uint8_t reserved)` write `*data` to USART

5.3 USART Interrupts

The receiving portion of the USART port is implemented as a interrupt, thus it only disturbs the program if there is new data available in the USART buffer of the CPU `UDR0`. The RX code uses the following external variables defined in `general.h` and `packet.h`.

Name	Data-type	Initial value	Description
`rx_packet`	`uplink_cmd_t`	`undef`	uplink packet structure (def. `packet.h`)
`rx_data_available`	`uint8_t`	`FLASE`	`TRUE` if 7-bytes received
`* rx`	`static uint8_t * &rx_packet`	`NULL`	pointer to uplink packet structure
`rxn`	`static uint8_t`	0	number of bytes received

Table 6: USART RX data structure

The interrupt routine is implemented as follows

```
ISR(USART_RX_vect)
{
        if (rxn == RX_PACKET_SIZE) // reset
                rxn = 0;

        // read character
        rx[rxn++] = UDR0;

        if (rxn == RX_PACKET_SIZE)
                rx_data_available = TRUE;
}
```

5.4 I2C - Two wire interface

5.4.1 Overview

The I^{2}C interface, is a interface developed by Phillips, for communication between different electronic components [1]. The I^{2}C interface will be called `TWI`, standing for `Two-wire interface`. The TWI interface is used to interface with the temperature sensor (TMP101) and the storage EEPROM (24M01).

5.4.2 Constants

The following constants are defined in `twi.h`. Please note the TWI code relies heavily on constants defined in `util/twi.h`, which is part of the `avr-libc`.

`MYTWBR` TWI Bit rate register

`MYTWSR` *prescaler* for TWI frequency

`F_SCL` TWI interface frequency

`TWI_BUS_MAX_RETRIES` number of retries before giving up

`TWI_BUS_TIMEOUT` twi bus timeout in ms

[1]`I2C_phillipse_docs.pdf`

`TWI_START` if set TWI will send START condition

`TWI_DATA` if set TWI will send DATA with NACK

`TWI_DATA_ACK` if set TWI will send DATA with ACK

`TWI_STOP` if set TWI will send STOP condition

`TWI_BUS_SUCCESS` *deprecated* use `TWI_OK`

`TWI_OK` TWI command was successful

`TWI_BUS_FAILURE` *deprecated* use `TW_BUS_ERROR` from `util/twi.h`

`ACK` *deprecated*

`NACK` *deprecated*

`TWI_GENERAL_CALL` TWI bus general call (`0x00`)

5.4.3 Data storage

The TWI subsystem does not save any data internally, all functions that handle data require a pointer to be passed to them, furthermore the code requires sufficient storage to be allocated.

5.4.4 Functions & Macros

The following function headers and macros are defined in `twi.h`.

`void twi_init(uint8_t twbr_value)` TWI interface init

`uint8_t twi_transmit(uint8_t operation)` TWI transmit data on the bus

`uint8_t twi_writebyte(uint8_t reg_addr, uint8_t dev_id, uint8_t dev_addr, uint8_t * pTxData)`
TWI write a single byte to a device

`uint8_t twi_writebytes(uint8_t reg_addr,  uint8_t dev_id, uint8_t dev_addr, uint8_t * pTxDat`
TWI write several bytes to a device

`void twi_stop(void)` *deprecated* send STOP condition

`uint8_t twi_readbyte(uint8_t reg_addr, uint8_t dev_id, uint8_t dev_addr, uint8_t * pRxData)`
TWI read a byte from a device

`uint8_t twi_readbytes(uint8_t reg_addr, uint8_t dev_id, uint8_t dev_addr, uint8_t * pRxData`
TWI read several bytes from a device

Details TWI function arguments

5.5 Interrupts (ADC & External INT)

5.5.1 ADC Overview

The analog-digital converter is used to transform a analog signal ($0 - 5$V) to a 10-bit number. On the UA-HAB PCB four components are connected to four of the ADC pins of the CPU, the three HV (high voltage) readings of the Geiger tubes and the pressure sensor.

Name	Data-type	Initial value	Description
`reg_addr`	`uint8_t`	`N/A`	buffer
`dev_id`	`uint8_t`	`N/A`	buffer index pointing to next free index
`dev_addr`	`uint8_t`	`N/A`	
`* pRxData`	`* uint8_t`	`N/A`	pointer to a sufficiently large array of bytes for Rx-data
`rxSize`	`uint8_t`	`N/A`	number of bytes to be received
`* pTxData`	`* uint8_t`	`N/A`	pointer to a sufficiently large array of bytes for Tx-data
`txSize`	`uint8_t`	`N/A`	number of bytes to be send

Table 7: TWI function arguments

5.5.2 Constants

`PPM1` HV1 control

`PPM2` HV2 control

`PPM3` HV3 control

`ADC0` HV1

`ADC1` HV2

`ADC2` HV3

`ADC7` pressure channel

`ADC8` A328P internal temperature channel

`GMT_HV` Geiger tube voltage

`ADC_HV` Geiger tube voltage measured

`PPM_PERIOD` HV control frequency

5.5.3 Data storage

The measurement of the voltage is stored in a external variables defined in `general.h` (see subsubsection 5.0.3) called `ADC#_value` (where `#` is the number of the channel). These variables are redeclared in `interrupts.c`

5.5.4 Functions & Macros

The following function headers are found in `interrupts.h`

`void ADC_init(void)` Initialize ADC

`void ADC_stop(void)` Stop ADC

5.5.5 ADC Interrupts

The interrupt of the ADC subsystem is called periodically. The frequency of calling can be adjusted with the ADC frequency. The ADC interrupt routine is defined in `main.c`.

```
ISR(ADC_vect)
{
        dummy;
}
```

5.5.6 External Interrupts Overview

The external interrupts are used to register and count the voltage changes that indicate a count by the Geiger Mueller tubes.

5.5.7 Data storage

The number of counts are stored in a external variables defined in `general.h` (see subsubsection 5.0.3) called `T#` (where `#` is the number of the channel). These variables are redeclared in `interrupts.c`

5.5.8 Functions & Macros

The following function headers are found in `interrupts.h`.

`void Ext_interrupt_init(void)`	Initialize external interrupts
`void Ext_interrupt_stop(void)`	Stop external interrupts

5.5.9 Interrupts

The interrupt of the external interrupt is always called if there is a voltage change on the input pin. The following interrupt routine is defined in `main.c`.

```
ISR(PCINT0_vect)
{
       T1++;
}
```

5.6 Temperature T101

Weather Balloon Documentation

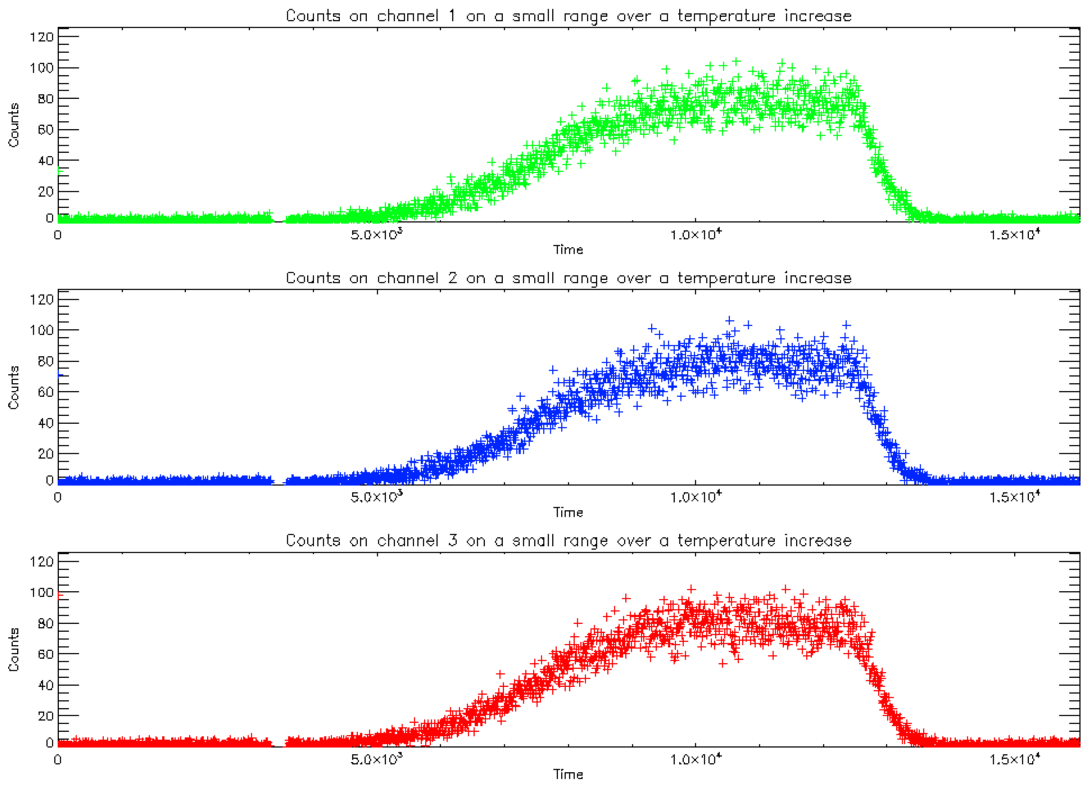


Figure 31: This balloon flight was conducted July 5, 2011, and collected counts as a function of time.

Weather Balloon Preparation and Launch Documentation
Maple Leaf Particle Detector

Wyatt Johnson
Andreas Buttenschön, Cory Hodgson, Laura Mazzino,
and Quinn Farr

Department of Physics
University of Alberta
Edmonton AB T6G 2G7
Canada

July 25, 2011

Revision: 1.0

Table of Contents

1 Preparation Details

During the preparation for the weather balloon launch, the following is a procedure for the safe and efficient operation of the mission. The following details are designed to be printed off and checked as the various tasks are completed.

1.1 Preparation Checklist

This is a procedure and checklist for the preparation stage of the gondolas as well as the materials required for a successful launch. Times are relative to launch date/time.

T = -2 Days `Prepare the Gondolas for payload` - The payload must remain secure whilst riding in the Gondola, appropriate pieces of styrofoam must be cut to secure the payload components in place. Care must be taken to insure that none of the components of the payload touch another as this could cause harm on impact.

☐ Measure, cut, and glue styrofoam used to construct the main Gondola housing

☐ Measure and cut styrofoam inserts to be inserted into main Gondola housing for the placement of the tracker and the payload that you are launching

☐ Verify that string is not in direct contact with styrofoam block, ensure instead that there is a layer of tape around edge, this prevents the styrofoam from getting extremely stressed at these contact points

☐ Prepare and attach a recovery sign, including information such as project name, phone number if found, etc

☐ Verify that all seams are taped sufficiently to resist water with the exception of the top of the box so that the payload may be inserted

T = -2 Days `Verify Payload Operation` - Verify that the payload operates on battery power before the expected launch date.

- ☐ Verify proper operation of payload according to expected payload operation

T = -2 Days `Acquire Data for Predictions` - To run predictions, you need to weigh the total payload (Including payload, gondola, trackers, animals, etc). Use the co-ordinates of the launch site (i.e. Lister Field 53.5252N, -113.5317W)

- ☐ Total weight of payload assessed
- ☐ Accent rate calculated via web-app on: `http://www.isset.ualberta.ca/students/balloontools` or manually in Section 4.1.1 on page 7

T = -2 Days `Run First Prediction on Predicted Landing Location` - Run the prediction on the prediction webpage: `http://habhub.org/predict/`. Information will be required such as the weight of the balloon, the accent rate of the balloon as calculated via the web-app on: `http://www.isset.ualberta.ca/students/balloontools` or manually in Section 4.1.1 on page 7, and the exact time of the launch.

- ☐ Prediction Ran
- ☐ Initial decision made regarding location and date of flight as per the prediction results

T = -2 Days `Prepare Helium` - Acquire and prepare the helium tanks, verify that they meet the requirements outlined in Section 4.2.1

- ☐ Acquire helium tanks from the department
- ☐ Verify that there is the minimum pressure/number as outlined in Section 4.2.1
- ☐ Store the tanks secured with a chain in a location that will be accessible before the expected launch time

T = -2 Days `Prepare Launch Material/Tools Package` - Purchase and/or verify all materials on the *Preparation & Materials Checklist* in Section 4.2 on Page 8.

- ☐ Verify that all batteries are fully charged or have sufficient charge to last the flight and tracking
- ☐ Items are purchased and/or verified on the list located in Section 4.2 on Page 8, check off those boxes as these are completed
- ☐ Prepare, verify integrity, and package all materials for Pre-Launch listed in Section 4.2.1 on Page 8 required for launch into a box or other vessel

T = -1 Days `Run 24 hour Prediction on Predicted Landing Location` - Run the prediction on the prediction webpage: `http://habhub.org/predict/`. Information will be required such as the weight of the balloon, the accent rate of the balloon as calculated via the web-app on: `http://www.isset.ualberta.ca/students/balloontools` or manually in Section 4.1.1 on page 7, and the exact time of the launch.

- ☐ Prediction Ran
- ☐ Verify decision made regarding location and date of flight as per the prediction results

T = -1 Day `Prepare Tracking Materials Package` - Verify and prepare all items required for tracking and post-launch as outlined on Page 9 in Section 4.2.2.

- ☐ Charge the HAM Radio Equipment and laptops
- ☐ Verify that the correct call signs are inputed into the HAM Radios
- ☐ Prepare and pack all necessary materials required and listed on Page 9 in Section 4.2.2
- ☐ Prepare the counterweight as listed in Section 4.2.1 on Page 8 from the web-app on: `http://www.isset.ualberta.ca/students/balloontools` or manually in Section 4.1.2 on page 7

T = -1 Day `Prepare Map Tiles` - Based on the most recent prediction data, collect and cache the appropriate map tiles on the APRSIS32 software.

- ☐ Download APRSIS32 Software: `http://aprsisce.wikidot.com/downloads`
- ☐ Ensure that your computers have the appropriate drivers to interface with the radios [CP210x USB to UART Bridge VCP Drivers] available here: `http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx`
- ☐ Download map tiles surrounding and along predicted path after disabling the map tile cache purge option

T = -12 Hours `Run 12 hour Prediction on Predicted Landing Location` - Run the prediction on the prediction webpage: `http://habhub.org/predict/`. Information will be required such as the weight of the balloon, the accent rate of the balloon as calculated via the web-app on: `http://www.isset.ualberta.ca/students/balloontools` or manually in Section 4.1.1 on page 7, and the exact time of the launch.

- ☐ Prediction Ran
- ☐ Verify decision made regarding location and date of flight as per the prediction results

2 Pre-Launch

2.1 Pre-Launch Checklist/Procedure

T = -1 Hour `Arrive at Designated Launch Site` - Arrive at the site location that was determined via the most recent prediction data. Re-run the predictions to ensure highest success rate. Verify that all materials are available. Additionally the counter weight is prepared.

- ☐ Inventory check completed
- ☐ Prediction Ran
- ☐ Verify decision made regarding location and date of flight as per the prediction results
- ☐ Prepare counterweight as per counterweight calculations

T = -1 Hour `Prepare Helium Tanks` - This is where the helium tanks are prepared for filling the balloon.

- ☐ Attach helium regulator to tanks
- ☐ Verify Correct Pressure as outlined in Section 4.2.1

T = -45 Min `Prepare Launch Site` - This involves preparing the launch site for the weather ballon. The site must be cleaned before you can lay out the balloon or other equipment.

- ☐ Lay out tarp/plastic in an area that contains little/no debris that provides some (If any) shelter from the wind

T = -30 Min `Prepare Balloon` - Get the weather balloon for launch/filling.

- ☐ Remove balloon from packaging and lay it out onto the tarp/plastic
- ☐ Attach balloon filling tube in preparation for filling the balloon
- ☐ Attach the opening of the balloon to the helium tanks filling tube and secure it using the screwdriver

T = -30 Min `Final Verification/Preparation of Gondolas` - This stage prepares the gondola and attached hardware/equipment for flight. All electronics equipment are additionally prepared.

- ☐ Verify that the gondolas are secure
- ☐ Insert the gondola inserts with payloads and power the electronic equipment as you insert
- ☐ Make a note of the time that the electronic devices are powered for tracking purposes
- ☐ Seal and attach all gondolas in order (Tracker $\rightarrow$ Payload $\rightarrow$ Parachute)
- ☐ Verify that all components of the gondola and attached are flight ready

3 Launch

3.1 Launch Checklist/Procedure

T = -15 Min `Verify tracker functionality` - Verify that the trackers are operating correctly to prevent payload loss.

- ☐ Verify GPS data using HAM Radios
- ☐ Verify packet data on APRS via: `http://www.aprs.fi/`

T = -10 Min `Fill Balloon` - This is the most critical stage in the launch. The balloon must be kept safe from contaminants and prevented from taking off prematurely.

- ☐ Ensure that the balloon is securely attached to the fill hose via the metal clamp
- ☐ Attach counter weight (Already prepared) to the bottom of the balloon fill hose
- ☐ Fill balloon until counterweight balances the lift of the balloon such that the counterweight appears weightless (Begins to lift off)
- ☐ Tie the balloon off from the helium nozzle with the string, using many double knots; fold the balloon end over itself once, then continue tying knots around this as well as through the newly created loop from the bent over end of the balloon end

T = -10 Min `Final Payload/Weather Balloon Validation` - This is the final verification stage of the payload and balloon before it is launched

- ☐ Attach the payload package (Tracker $\rightarrow$ Payload $\rightarrow$ Parachute) to the bottom of the weather balloon via the strings that are attached to close/tie the balloon
- ☐ Tug on all connections to verify tough enough knots to withstand flight

T = 0 Min `Launch` - Balloon is literally launched.

- ☐ Move to clear area of launch site
- ☐ Release balloon along sting, hand over hand in order following the string starting with the balloon

4 Appendix

4.1 Calculations

4.1.1 Accent Calculations

The balloons accent rate is calculated using the weight of the payload mass (This is including the payload gondola, the tracker + gondola, parachute), and the weigh of the helium tank fill nozzle. The calculations are provided most accurately on: `http://www.isset.ualberta.ca/students/balloontools`. If a computer is not available, the following computation can be used:

$$\text{Accent Rate [m/s]} = \frac{52.9306\sqrt{p}}{\sqrt[3]{b + 1501.35p}} \tag{1}$$

Where p is the payload weight in kg including the gondola, the parachute, and tracker. b is the latex ballon's weight in g.

4.1.2 Counterweight Calculations

The counterweight is usually a jug of water that is filled such that its weight will perfectly balance the balloon when it is able to provide the correct lift to correspond with the desired landing zone as per the predictions that are numerously ran. The calculations are provided most accurately on: `http://www.isset.ualberta.ca/students/balloontools`. If a computer is not available, the following computation can be used:

$$\text{Weight of Counterweight [kg]} = p \times 1.5 - n \tag{2}$$

Where p is the payload weight in kg including the gondola, the parachute, and tracker. And n is the weight of the helium nozzle that is attached to the balloon during the inflation.

4.2 Inventory Checklists

4.2.1 Pre-Launch Equipment

Preparation Equipment Checklist		
Yes	No	
☐	☐	Gondolas
☐	☐	Tape (Masking/Scotch/Packing)
☐	☐	Rope/String
☐	☐	Trackers **2X**
☐	☐	→ GPS Module
☐	☐	→ Electronic controller
☐	☐	→ Antenna
☐	☐	→ Battery pack
☐	☐	→ 4 AA Lithium batteries
☐	☐	Helium Regulator
☐	☐	→ Regulator
☐	☐	→ Air hose
☐	☐	→ Balloon nozzle
☐	☐	Helium Tanks **2X** - These tanks each have 2100psi of pressure
☐	☐	Chains - For Helium Tanks
☐	☐	Latex Gloves
☐	☐	Counterweight - Usually 2L of water
☐	☐	Small weight scale
☐	☐	Parachute
☐	☐	Scissors
☐	☐	Paper and Marker - For recovery note
☐	☐	Plastic Sheet - For covering ground where balloon is filled from
☐	☐	Flathead Screwdriver
☐	☐	Crescent wrench
☐	☐	Bug Spray
☐	☐	Outdoor footware

4.2.2 Tracking Equipment

Tracking Equipment Checklist		
Yes	No	
☐	☐	HAM Radio Equipment **MIN 3X**
☐	☐	→ HAM Radio
☐	☐	→ Car battery charger
☐	☐	→ Wall battery charger
☐	☐	→ HAM car extender antenna
☐	☐	Laptop's **MIN 3X**
☐	☐	Vehicles **MIN 3X**
☐	☐	Car laptop power adapters
☐	☐	Camera(s)
☐	☐	Axe
☐	☐	Scuba Gear
☐	☐	Ladder - *Optional*