DemoSat Design Document

Team Apogee

Andrew Mezich
Jack Hetherington
Jonathon Davidson
Melinda Bradley
Josh Connolly
Wes Hileman
Matt Hileman
Peter Garate
Will Van Noordt
Cory Ross

27 April 2015

Revision Log

Revision	Description	Date
A	Critical Design Review	27 March 2015
B	Final Report	27 April 2015

Contents

1 Mission Overview

The DemoSat program is a statewide engineering project in which teams of college students construct payloads for launch into the lower stratosphere on a high-altitude balloon. The payload built by Pikes Peak Community College's Apogee team is a *spacecraft* that embodies sensors, actuators, and their supporting subsystems. The spacecraft imitates a cube satellite (CubeSat) and, as our mission statement indicates, is designed to exploit the atmosphere's weather, radiation, and perspective conditions:

A multitude of unique conditions exist in the lower atmosphere such as increased radiation-intensity, widened perspective, and weather variation. One goal of this mission is to test the effectiveness of magnetic radiation-shielding in the atmosphere. Also, low-power radio systems have the greatest potential to function in wide-perspective environments; thus, another goal of this mission is to establish a low-power radio downlink. This mission's final goal is to collect information pertaining to the kinetic and weather conditions encountered by the payload during flight, as this information is useful to future teams.

We've developed several primary and secondary objectives from this mission statement. Our primary objectives include (1) to establish a radio downlink during flight and (2) to test the effectiveness of magnetic fields as ionizing-radiation shielding. Our secondary objectives are (1) to collect and store weather information, (2) to collect and store flight information, and (3) to collect and store electromagnetic-radiation intensity information.

By completing this mission, we aim to learn the effectiveness of magnetic radiation-shielding, determine the feasibility of low-power radio systems at high-altitude, and characterize the flight of a payload on a high-altitude balloon.

2 Requirements Flow Down

At the topmost level, three requirements are imposed on the spacecraft. The spacecraft must (1) collect mission data, (2) operate for the mission's duration and be recovered, and (3) comply with sponsor constraints to fulfill the mission objectives. From these broad requirements, a number of lower-level requirements are derived. Table 1 summarizes level 0 (toplevel) and level 1 (lower-level) requirements.

Requirement ID	Name	Detail
R0.0	Data Collection	Accurate collection and storage of relevant data.
R0.1	Survivability & Recoverability	Full spacecraft operation is maintained during flight and the spacecraft is recovered upon mission completion.
R0.2	Sponsor Constraints	All sponsor constraints are satisfied.

Table 1: Mission requirements.

3 Design

Six elements form the mission design, or mission architecture, of the Apogee project. Table 2 on the next page summarizes these six elements, including the mission concept, subject, payload, spacecraft bus, launch system, and ground system. A complete mission design encompasses a set of designs for each element [4].

A number of the elements are marked as non-tradable, indicating the designs of those elements are fixed by some constraining factor. For example, the subject is defined by the mission statement and the sponsor sets the launch and ground systems. Contrary to non-tradable elements are tradable elements, whose designs are open to change. The mission concept, payload, and spacecraft bus are tradable elements. In the sections that follow, the design of each tradable element is discussed in detail.

Figure 1 on the following page shows the spacecraft's electrical schematic, Figure 2 on page 6 depicts a block diagram of the spacecraft, and Figure FIG lists all of the hardware used in the spacecraft's construction. The components identified in the diagrams are examined in following sections.

Element	Definition	Can be Traded	Reason
Mission Concept	The fundamental statement of how the system will work (data delivery, tasking, communications, timeline)	yes	Open to alternate approaches
Subject	The items sensed by the payload	no	Subject is passive and well defined in the mission statement
Payload	Hardware and software that sense the subject	yes	Can select method of sensing
Spacecraft Bus	Houses the payload and supporting subsystems such as power, structure and rigidity, temperature control, and data handling	yes	Can choose structure and subsystem configuration
Launch System	Launch vehicle, interfaces, and ground-support equipment. Constrains size, shape, and mass of spacecraft	no	Fixed by sponsor (COSGC)
Ground System	The hardware and software governing ground-based command and control of the launch vehicle and spacecraft	no	Fixed by communications architecture

Table 2: Mission elements.

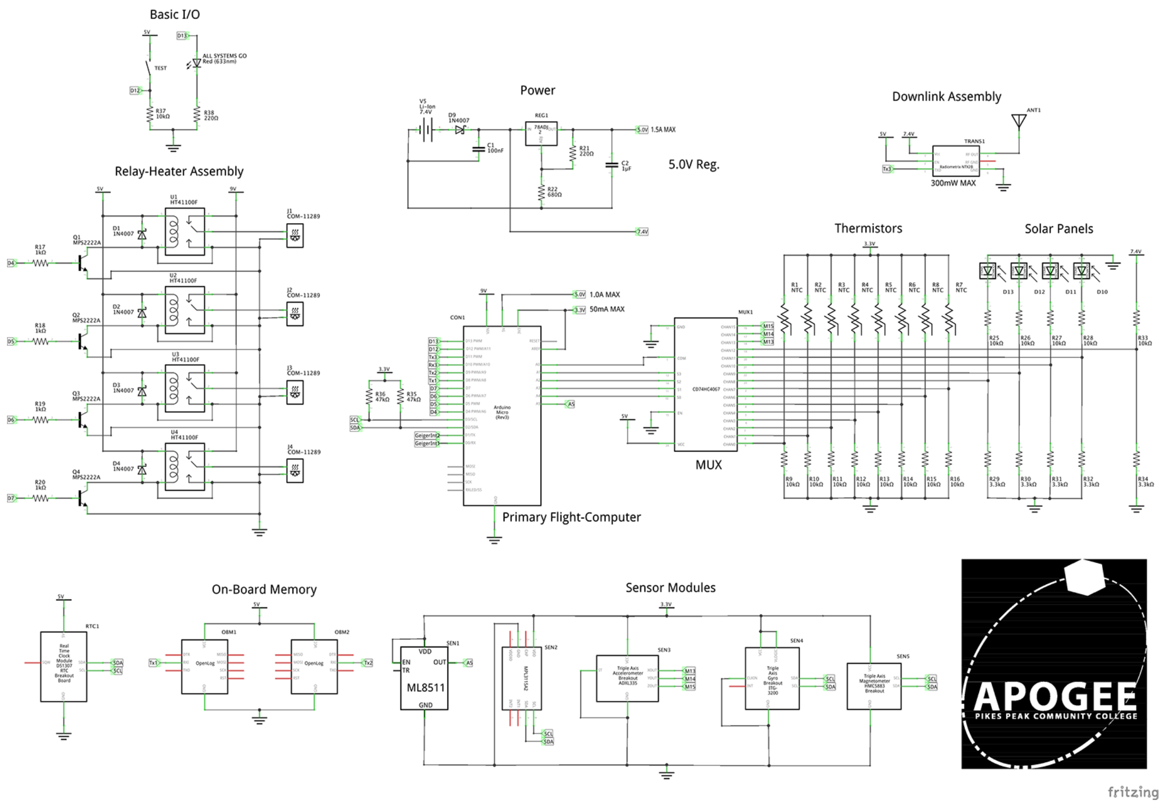


Figure 1: Spacecraft electrical schematic.

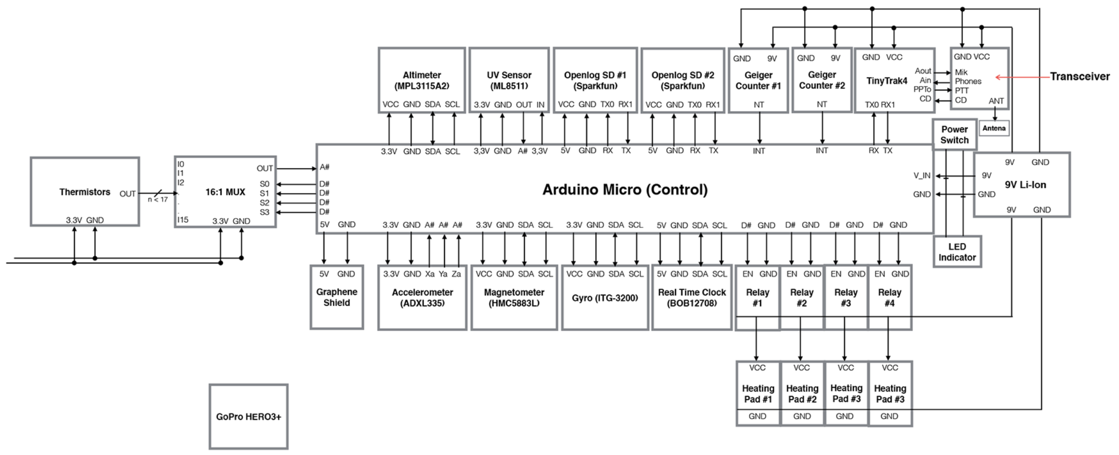


Figure 2: Spacecraft block diagram.

3.1 Mission Concept

The *mission concept* defines the elementary workings of the mission and is subdivided into the areas of data delivery and tasking [4]. Each area is discussed in the following sections.

3.1.1 Data Delivery

Data delivery addresses data generation, collection, and ultimate distribution to the Apogee team. Figure 3 on the following page summarizes the high-level data flows of the mission. As shown, data is produced by sensors, processed into storable formats, and stored in onboard memory for later retrieval. Telemetry is also sent via radio downlink, providing a level of redundancy.

The data generated by the payload and transmitted by downlink is *mission data* that characterizes the subject directly. *Housekeeping data* is collected by sensors in the spacecraft bus and either influences the spacecraft's operation (e.g. heater control) or is used in analysis of mission data (e.g. altitude data) [4].

3.1.2 Tasking

Tasking decides when and in what order operations are preformed by the spacecraft [4]. Tasking for the Apogee spacecraft be accomplished autonomously by an onboard computer. The computer is an *Arduino Micro*, a breakout of the ATmega32u4 microcontroller (see section 3.2.4 on page 9 for more details). Figure 4 on the following page depicts the state diagram defining the computer's operation at any instant. See appendix section 13 on page 22 for the flight software.

3.2 Spacecraft Bus

The *spacecraft bus* accommodates the payload's supporting subsystems, including structural, power, thermal, structural, command and data handling (C&DH), and communications. Each subsystem is discussed in the next sections.

3.2.1 Structural

The structure is made from six pieces of rigid sheets of $\frac{3}{4}$"-thick expanded polystyrene Mylar-foil-lined foam, cut and assembled to form the shape of a hollow cube. Each piece is fastened together using 100% silicon. An approximately 1" thick layer of *Great Stuff* insulating spray foam covers the outside of the cube. The inside dimensions of the structure are 6.5" x 6.5" x 6.5" (274.625 in^3).

The rigid foam core gives the spacecraft a uniform interior with straight walls and a level floor and ceiling, which allows each payload component to fit neatly inside. To facilitate component placement and organization, a system of shelves is used. This shelving system is comprised of two 6.5" x 6.5" shelves made of foam-core board, placed horizontally, and held in place at each corner by hollow copper rods. The

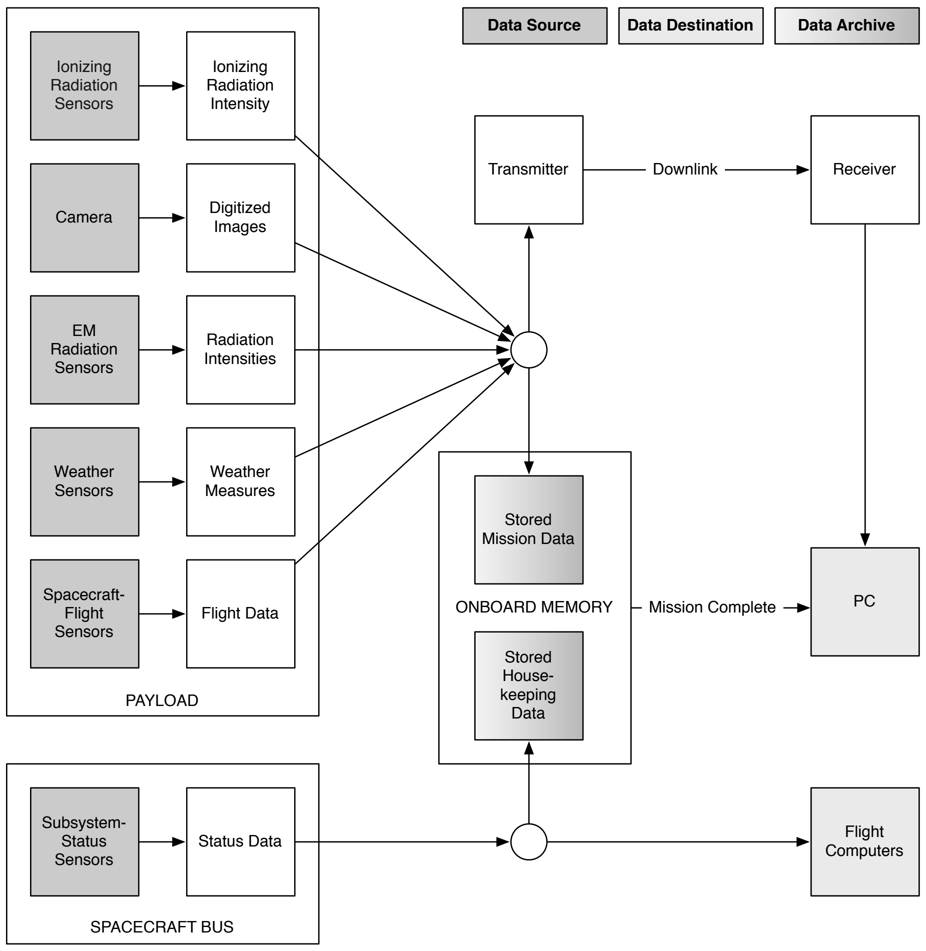


Figure 3: Toplevel data flows.

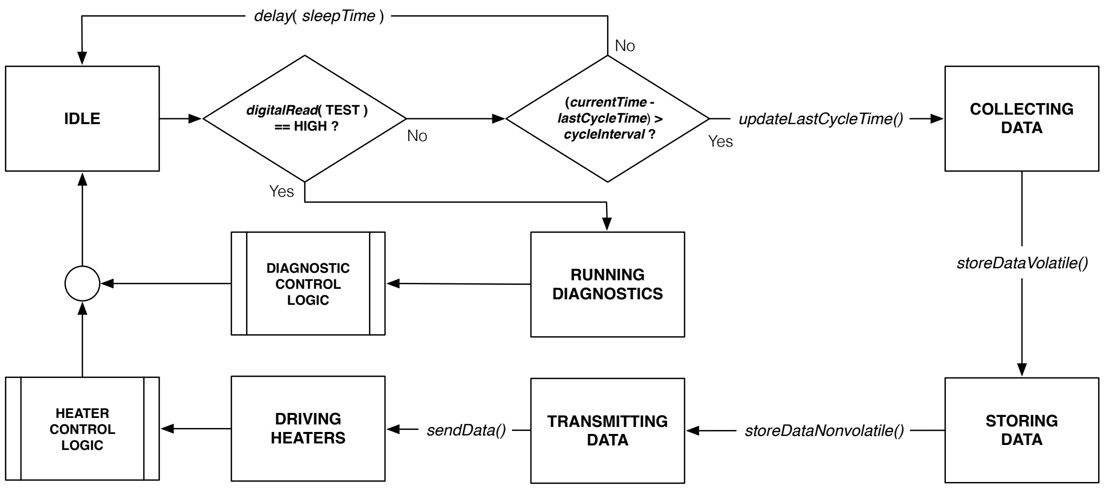


Figure 4: State diagram that determines the flight-computer's programming.

shelving system divides the interior of the spacecraft into three compartments and is easily removed from the spacecraft, allowing the crew to quickly make repairs or modifications to interior components.

To satisfy the secondary function, insulation from extreme cold, the structural materials were carefully chosen. The $\frac{3}{4}$" expanded polystyrene foam lined with Mylar foil has an R-value of 2.9. The Great Stuff insulating spray foam has an R-value of approximately 6 at a thickness of 1". These two materials were chosen for their thermal insulating characteristics, their superior durability and strength, and for their low weight.

The copper rods assist in each of the three structural functions. They are each anchored into the floor and ceiling of the spacecraft, supporting the two shelves and adding strength to the spacecraft structure as a whole. The rods are also connected to the power supply, which allows them to serve as power rails to the electrical and thermal components of the spacecraft. This satisfies the tertiary goals of the spacecraft structure.

The spacecraft's electrical components are secured and connected in circuits with printed circuit boards (PCBs) to reduce wiring and promote intelligibly. For ease of removability, external components are wired to the boards with screw headers. Figure 5 portrays the circuit board created for the thermal subsystem.

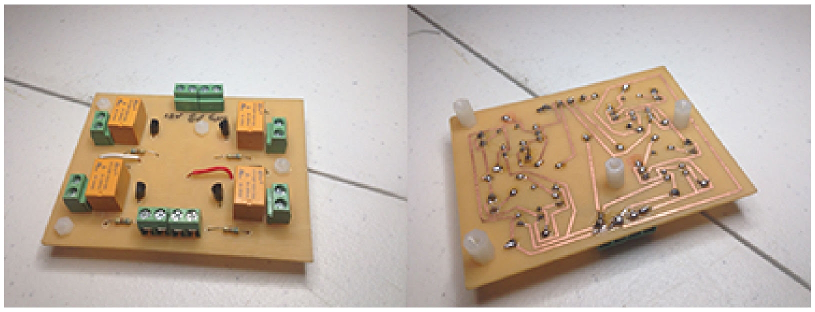

(a) Front (b) Back

Figure 5: Thermal-subsystem PCB.

3.2.2 Power

Eight *Efest* Purple 18650 3100mAh 3.7V 20A batteries supply spacecraft power. The power system provides, via the onboard voltage regulator, 3.3V and 5.0V electric potentials to the electrical components and 7.4V to the thermal components (heating pads) of the spacecraft. Each of these potentials is supplied individually through three of the four copper tube power rails used simultaneously as structural support, while the fourth copper tube goes to ground.

The extreme cold of the upper troposphere and lower stratosphere has the potential to significantly reduce battery life and threaten the success of the spacecraft. To protect the batteries from the cold, a single heating pad is used in the power subsystem compartment to keep the batteries at a temperature of 10° C with a variance of approximately ± 5°C. A thermistor in the power compartment detects the current temperature and relays that information to the flight computer, which controls the status of the heating pads.

By a wide margin, the power subsystem is the heaviest of the spacecraft subsystems. To keep the center of gravity as low as possible, the batteries are secured in plastic single- or double-battery holders and fastened to the floor of the structure.

3.2.3 Thermal

The thermal subsystem consists of four heating pads powered at 7.4 volts. Each heating pad is controlled by an electrical relay system managed by the ATmega32u4 microcontroller on the Arduino Micro board. The heating pads consist of a mesh of polyester and conductive metal. When supplied with a small current (~1.0 Amp), the materials act as a resistor and produce heat. The electrical relay system is assembled on a custom printed circuit board.

Thermistors inside each of the three compartments of the spacecraft detect the local temperature and send that information to the ATmega32u4 microcontroller. The microcontroller is programmed to activate each heater individually if the local temperature falls below 10°C. Temperature regulation of the power subsystem is the top priority of the thermal subsystem.

3.2.4 Command and Data Handling

Spacecraft timekeeping, FC health monitoring, and data collection, processing, and storage are encompassed by the command and data handling (C&DH) subsystem [4].

The SparkFun Real-Time Clock (RTC) Module realizes timekeeping. The device is a breakout of the DS1307 microchip, a RTC that provides I2C-accessible date (year, month, day) and time (hour, minute, second) data. The chip updates the stored date and time data with the output of an on-board crystal oscillator. A small lithium battery placed on the underside of the breakout runs the RTC in the event of power failure.

Data collection and processing is accomplished with a ATmega32u4 microcontroller (*Arduino Micro* board). The microcontroller implements a 10-bit ADC that is utilized in obtaining readings from analog sensors (such as the UV sensor and thermistors discussed in following sections). Data is obtained from digital sensors (such as the MPL3115A2 pressure sensor) with the supported I2C two-wire serial protocol. Last, the microcontrollers' digital pins are employed to collect data from the Geiger counter. The FC health-monitoring implement is the ATmega32u4's built-in watchdog timer; in the event the chip fails to reset the timer in a certain amount of time (e.g. due to a lock-up), the computer is reset.

3.2.5 Communications

The communication subsystem is responsible for passing sensor data to a ground station for analysis via a UHF radio transmitter. The transmitter chosen for the project is a Radiometrix NTX2 70cm band transmitter. Its operational frequency is 434.650MHz. The data mode chosen for encoding the data is audio Morse code to be modulated by an Arduino Uno microcontroller. A twin lead J-pole has been fashioned to increase the likelihood of the low power transmitter being detected by the ground station while in flight.

The ground station is comprised of a laptop to display and record the decoded data, a Coastal Chipworks TNC-X terminal node controller to demodulate the data, a Baofeng UV5R handheld UHF/VHF dual band radio to detect the radio transmission, and a custom built, high gain, directional antenna to increase the likelihood of detecting the transmitter while in flight. If possible, an omnidirectional antenna will be used while mobile.

3.3 Sensor Array

The hardware and software that sense the mission's subject comprise the *payload* [4]. Seven devices form this mission's payload: (1) two Geiger counters, (2) an ultraviolet (UV) light sensor, (3) solar panels, (4) thermistors, (5) a pressure and altitude sensor, (6) a triple-axis magnetometer, and (7) a triple-axis accelerometer. These items are discussed below.

3.3.1 Geiger Counters

The GCK-01-01 analog Geiger counter is responsible for measuring radioactivity. The radioactivity collected is alpha (above 3.0 MeV), beta (above 50 KeV), and gamma (above 7 Ke). The Geiger counter clicks and blinks an LED with each detection along with an output to a data chart. Two counters will be on the payload, a shielded device along with a control to measure the difference in incoming particles. They may be powered with a DC or AC voltage between 6 and 12 volts.

3.3.2 Ultraviolet-Light Sensor

Our payload is equipped with a ML8511 microchip breakout from Sparkfun. The ML8511 is a UV sensor capable of reading UV-A and UV-B radiation and is most effective at reading 280-390nm light. The sensor sends data by way of an analog output that can be read with an analog-to-digital converter (ADC). The potential across the output and ground is generated by an internal amplifier that converts photo-current to voltage.

The UV sensor is utilized to read UV radiation from the sun. To obtain an accurate reading, the sensor must be in direct contact with the sun or the radiation. For this reason, the UV sensor was attached to the outside of the payload. Also, depending on temperature, the UV intensity received from the sensor will vary slightly, as shown in Figure 3 8. Since the data received from the UV sensor is a voltage reading, it is necessary to convert the readings to intensity units ($^{mW}/_{cm^2}$ in this case).

3.3.3 Solar Panels

Attached to the exterior of our payload are PowerFilm® MPT6-75 flexible solar panel cells. We have one panel attached to each exterior side of our payload, excluding the top and bottom, and thus having four in total. Each MPT6-75 solar panel produces approximately 6V 50mA in standard conditions. Portrayed in figure 6 is the MPT6-75 solar panels.

Although solar panels can be used as a power source, the payload has an insufficient amount to charge the batteries properly. The payload instead utilizes the solar panels as sun sensors. When connected to an analog input, the voltage from the panels can be determined. As providing four analog pins directly from our micro-controller is impossible, we have instead put the panels onto the same 16:1 multiplexer the thermistors are on.

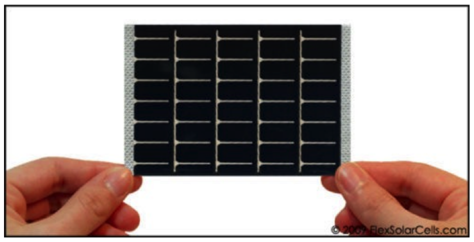


Figure 6: PowerFilm® MPT-75 flexible solar panel.

By having a solar panel on each side of our payload, we will be able to determine the rate of revolution, direction the payload is spinning, which direction the payload is facing at any given point in time, and the amount of energy being produced from the solar panels from solar energy. Although we have included other sensors that determine revolution, the solar panels will help verify these results.

3.3.4 Thermistors

External and internal temperature measurement is accomplished with a set of *thermistors*—ceramic- or polymer-based resistors whose resistances change considerably with temperature [3]. Due to their low cost, precision, and predictable temperature-response, thermistors are commonly applied as temperature sensors [2]. Figure 7 shows one of the eight negative temperature coefficient (NTC) thermistors placed in our payload.

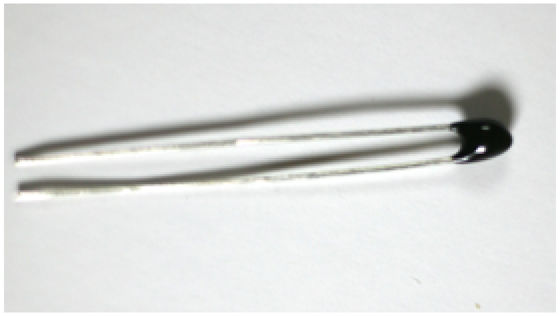


Figure 7: NTC Thermistor (10kΩ at 25°C with 3% error).

The resistance of a NTC thermistor decreases nonlinearly as temperature increases, roughly according to the B-parameter equation

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right) \tag{1}$$

where T is the thermistor's temperature in kelvin, R is the thermistor's resistance in ohms, B is a constant that depends on the thermistor, and R_0 is the thermistor's resistance at a reference temperature T_0 [2].

The thermistors utilized in this mission have a B-value of $B = 3950$ K. Table 3 displays the thermistors' resistances R_0, as measured by a multimeter, at a reference temperature of $T_0 = 292.1$ K.

The final unknown in the preceding equation (1), R, is determined with a 10-bit analog-to-digital converter (ADC) and a voltage divider; each thermistor is placed in series with a 10kΩ resistor across a 3.3V potential and the voltage drop across the resistor is measured with the 10-bit ADC of the flight-computer (an ATmega32u4 microcontroller). It can be shown that

$$R = (\frac{1023}{ADC_{out}} - 1)R_s \tag{2}$$

where R_s is the *measured* resistance of the series resistor (this value may differ from 10kΩ due to tolerances) and ADC_{out} is the digital output of the 10-bit ADC—an integer between 0 and 1023 ($= 2^{10} - 1$) that is directly proportional to a voltage between zero and the ADC's analog reference (AREF) voltage. Due to the stability of the 3.3V power supply on the *Arduino Micro* board [2], that supply is used as the AREF voltage. Also shown in table 3 are the measured series resistance values corresponding to each thermistor.

Thermistor ID	R_0 (kΩ)	R_s (kΩ)
T1	13.63	9.65
T2	13.63	9.99
T3	13.43	9.88
T4	13.25	9.90
T5	13.57	9.94
T6	13.28	9.95
T7	13.35	9.96
T8	13.37	10.04

Table 3: Thermistor parameters ($T_0 = 292.1$ K).

3.3.5 Pressure & Altitude Sensor

Portrayed in figure 8 is the implement for measuring pressure and altitude. The device is a SparkFun breakout of the MPL3115A2 I2C Precision Altimeter, a microchip utilizing a pressure sensor and a 24-bit ADC to generate serial-accessible pressure and altitude data. The chip also provides 16-bit temperature data via an internal temperature sensor.

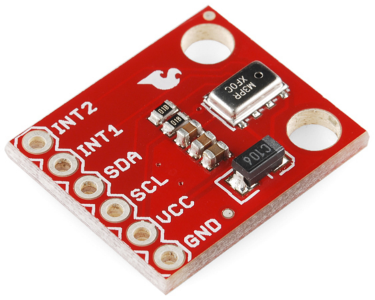


Figure 8: SparkFun MPL3115A2 Breakout

The MPL3115A2 derives altitude from the pressure measurement and the 1976 U.S. Standard Atmosphere. The relation

$$h = 44330.77(1 - (\frac{P}{P_0})^{0.1902632}) \tag{3}$$

results in the altitude h (in meters) given by the chip where P is the measured pressure (in Pascals) and $P_0 = 101326$ Pa is sea-level pressure [1]. Communication with the chip is handled with a C++ class written for import into the Arduino software. The chip is rated to a minimum operational pressure of 20kPa (which corresponds to an altitude of 12km). Thus, readings below that pressure value may be inaccurate.

3.3.6 Triple-Axis Magnetometer

Direction measurement is accomplished with a digital compass, specifically a SparkFun breakout of the HMC5883L triple-axis magnetometer. The device measures an ambient magnetic field's strength in three mutually-orthogonal x, y, and z directions and converts the measurements to I2C-accessible digital data with a 12-bit ADC. The magnetometer's gain is programmable between $0.73{}^{mG}\!/_{output\ unit}$ and $4.35{}^{mG}\!/_{output\ unit}$ to allow measurement of both relatively weak and strong magnetic fields. A gain of $0.73{}^{mG}\!/_{output\ unit}$ is used in our application to sense the earth's magnetic field.

Given magnetic-field strengths in the x, y, and z directions, m_x, m_y, and m_z, respectively, the angle the overall magnetic field vector B makes in the xy-plane is

$$\theta = \begin{cases} \arctan(\frac{m_y}{m_x}) & m_x > 0 \\ \pi + \arctan(\frac{m_y}{m_x}) & m_x < 0 \end{cases} \tag{4}$$

The angle the vector makes with the z-axis is

$$\phi = \begin{cases} \arctan(\frac{\sqrt{m_x^2+m_y^2}}{m_z}) & m_z > 0 \\ \pi + \arctan(\frac{\sqrt{m_x^2+m_y^2}}{m_z}) & m_z < 0 \end{cases} \tag{5}$$

Finally, the intensity of the magnetic field is

$$|B| = \sqrt{m_x^2 + m_y^2 + m_z^2} \tag{6}$$

These calculated quantities determine the direction of the spacecraft relative to the earth's magnetic field given the readings from the magnetometer.

3.3.7 Triple-Axis Accelerometer

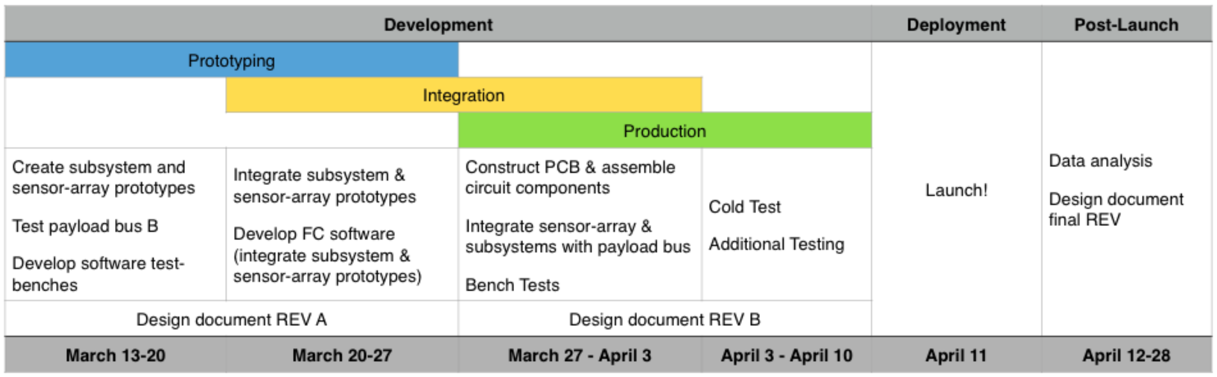
The implement for measuring acceleration is a SparkFun breakout of the ADXL335 triple-axis accelerometer. The device produces three voltage-outputs between zero and 3.3V that are directly proportional to the acceleration experienced in three mutually-orthogonal x, y, and z directions. The flight-computer's ADC is utilized to convert the voltage readings to digital form for processing.

4 Management

The challenge of management occurs in every team, the return of previous demosat participants were invaluable in the construction and the design of the payload. Another positive component of the team is the number of students. This helped in dividing up responsibilities and reducing the pressure from individuals and to allow decisions to be made as a group. Our limitations are generally time related, with deadlines and the PPCC lab being only available on Fridays for team meetings and scheduled work.

Spacecraft Bus	**0.8 kg**
Structure	0.3 kg
Power	0.3 kg
Thermal	0.1 kg
C&DH	0.1 kg
Communications	0.1 kg
Payload	**0.3 kg**
Shielding Expirement	0.2 kg
Sensor Array	0.1 kg
TOTAL	**1.1 kg**

Table 4: Mass Budget



Development				**Deployment**	**Post-Launch**
Prototyping				Launch!	Data analysis; Design document final REV
	Integration				
		Production			
Create subsystem and sensor-array prototypes; Test payload bus B; Develop software test-benches	Integrate subsystem & sensor-array prototypes; Develop FC software (integrate subsystem & sensor-array prototypes)	Construct PCB & assemble circuit components; Integrate sensor-array & subsystems with payload bus; Bench Tests	Cold Test; Additional Testing		
Design document REV A		Design document REV B			
March 13-20	**March 20-27**	**March 27 - April 3**	**April 3 - April 10**	**April 11**	**April 12-28**

Figure 9: Mission schedule.

5 Budget

Shown in figure 4 is the mass allocations for the spacecraft. Figure 10 on the following page shows the components' actual mass.

6 Test Plans and Results

Three types of testing have been conducted: Structural, Battery, and Power/Thermal.

The structural tests are meant to simulate takeoff, flight, and landing conditions. To simulate takeoff and flight conditions, the spacecraft and payload are spun overhead with a violent change in direction. To simulate landing conditions, the spacecraft and payload are dropped onto concrete from a height of 20ft as well as pitched down several flights of stairs. In each of the structural tests, the spacecraft performed well – sustaining only light scratches to the exterior and only minor interior damage. This interior damage was noted and adjustments were made to the payload securement plans.

Battery testing involves the comparison of three types of batteries in controlled conditions. The three types of batteries tested are all type 18650 3.7V batteries: Ultrafire 5000mAh, Efest 2500 mAh, and Efest 3100 mAh. Two batteries of each type are connected in parallel and the resulting voltage is applied to one of the 5cm x 10cm heating pads used in the thermal subsystem. Each test is run at room temperature (~22°C) until either the battery power is exhausted or the test reaches approximately three hours in length. The test results are in favor of the Efest 3100mAh batteries, which sustains a sufficient voltage over nearly three hours of testing.

The power/thermal test involves testing the Efest 3100mAh batteries powering the same heating pad in a cold environment, approximately 0°C. Again, the batteries sustained a voltage sufficient to power the heating pads over the course of nearly 2.5 hours.

Mass Breakout

Item(s)	Single Item Mass (g)	Quantity (#)	TOTALS (g)
Prototype A	740	0	0
Prototype B	310	1	310
Geiger Counter A	70.273	1	70.273
Geiger Counter B	59.253	1	59.253
E Fest IMR 3.7V Li-Ion	44.968	8	359.744
Copper Clad (~)	40.00	1	40
Solar Panels (Flexible)	3.665	4	14.66
ADXL335	1.595	1	1.595
MPL3115A2	1.635	1	1.635
MMA8452	1.620	1	1.62
RTC Module	3.075	1	3.075
NTX2B (RadioMetrix) (~)	30.0	1	30
ITG-3200	1.936	1	1.936
Mux 16-1	4.50	1	4.5
Heating Pads (L)	5.688	2	11.376
Heating Pads (S)	4.297	2	8.594
Battery Holder (E Fest)	13.381	4	53.524
Arduino Micro	6.720	1	6.72
Extra Wiring (~)	170	1	170
Fasteners (~)	150	1	150
TOTAL MASS	**1298.505**		

Figure 10: Mass of various spacecraft-components.

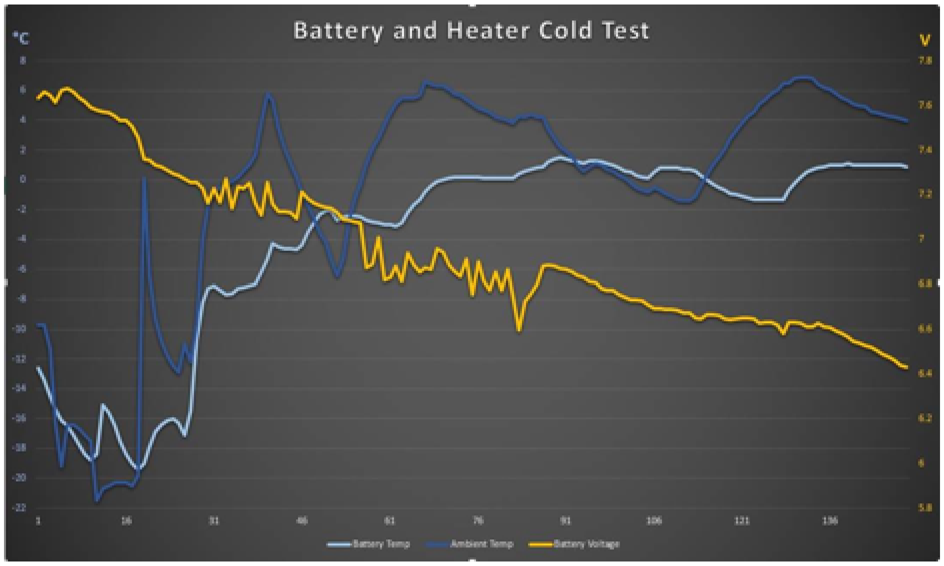


Figure 11: Battery and heater cold test.

7 Expected Results

The expected results of our flight vary with the different sensors the payload contained. The hypothesis is these sensors will ether increase or decrease in altitude. Examples of this is the pressure decreasing, UV and the solar panel voltage increasing with altitude. The expected results for the Geiger counter are that the shielded Geiger counter would receive a lower dosage of radiation compared to the control counter.

8 Launch and Recovery

Our payload was secured on the third launch balloon. We launched at 7:30 AM in Eaton, Colorado. The payload handler was Pete Garate who tested system functionality with the LCD pre-launch device, turned on power to satellite, and double checked payload integrity. At a first attempt to launch the payload, the launch balloon had insufficient hydrogen gas for lift, thus resulting in the launch interface landing in a upon take off. The launch interface was successfully re-obtained and relaunched with the correct amount of hydrogen, resulting in a successful mission to around 28 km at apex. After launch, we tracked the balloon via provided GPS system. Upon landing, the COSGC officials obtained permission of the land owner to intrude on the property. We then hiked out to get the payload a mile out from a dirt road. The launch interface was found intact and our payload was found in exceptional condition. After taking our payload, we hiked back to our vehicles and left the landing sight. We recovered data from our payload with SD Open Logs from SparkFun. Only system failed was one geiger counter's wire came loose during flight, thus providing inaccurate data for our geiger counter related experiments. Batteries were still powerful and able to run the payload for many hours after retrieval.

9 Results, Analysis, and Conclusions

9.1 Geiger Counters

Our radiation shielding experiment used 2 Geiger counters, one shielded and one unshielded. Figure 12 on the next page portrays plots of the two counters' readings with flight time.

The shielded counter is on top while the unshielded on is on the bottom. The shielded counter shows slightly higher readings than the unshielded on indicating either a higher sensitivity, or the shielding redirected the radioactive particles towards the sensor. Unfortunately, the shielded counter malfunctioned during the flight so we were unable to see the results as it reached the higher altitudes.

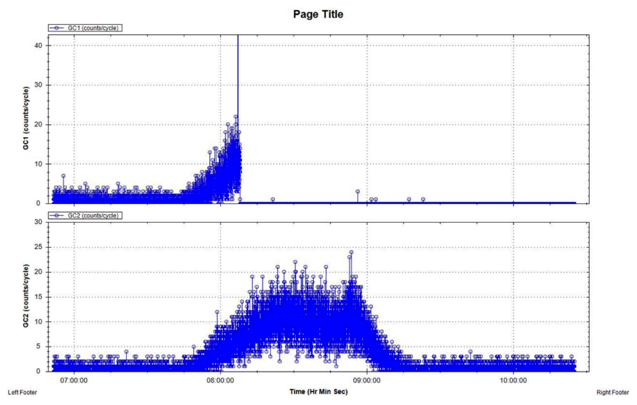


Figure 12: Particle-frequency with time.

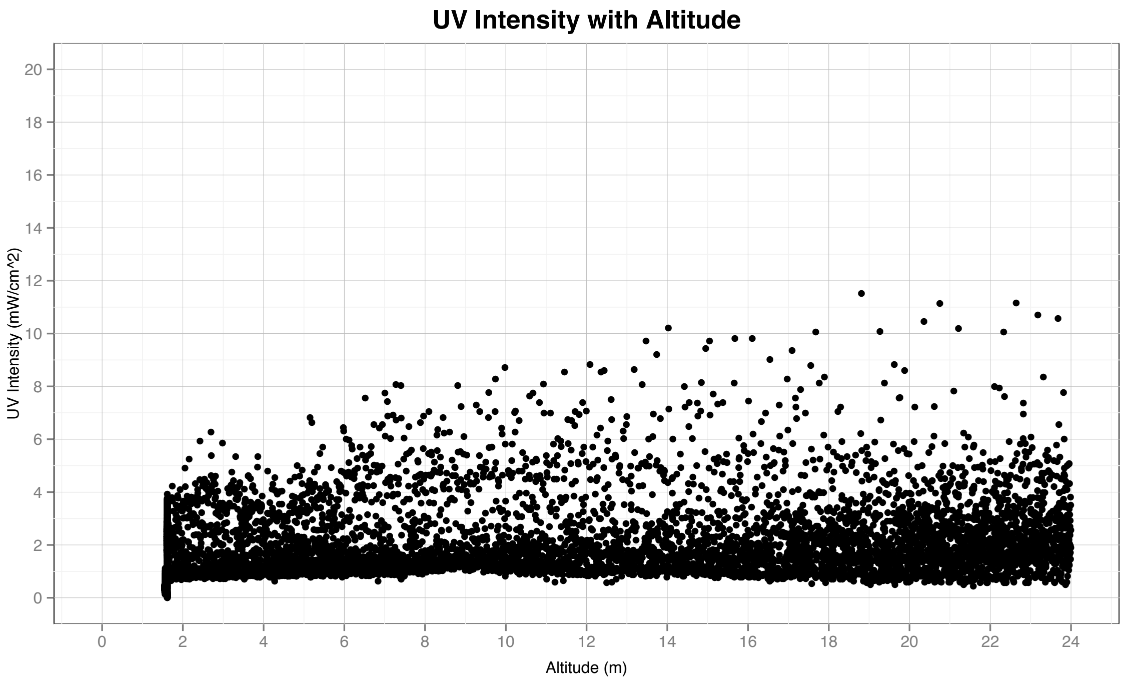


Figure 13: Ultraviolet-light intensity with altitude.

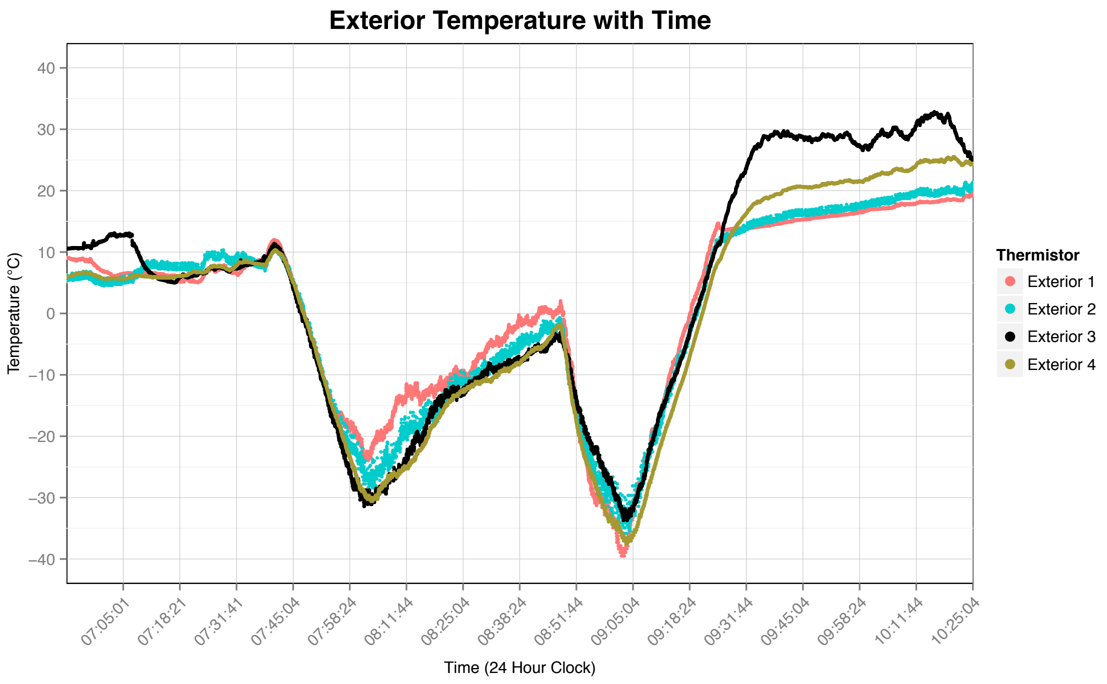
9.2 UV Sensor

The UV sensor showed sporadic readings, and the intensity general increased with altitude. The data plot of the UV readings are seen in figure 13.

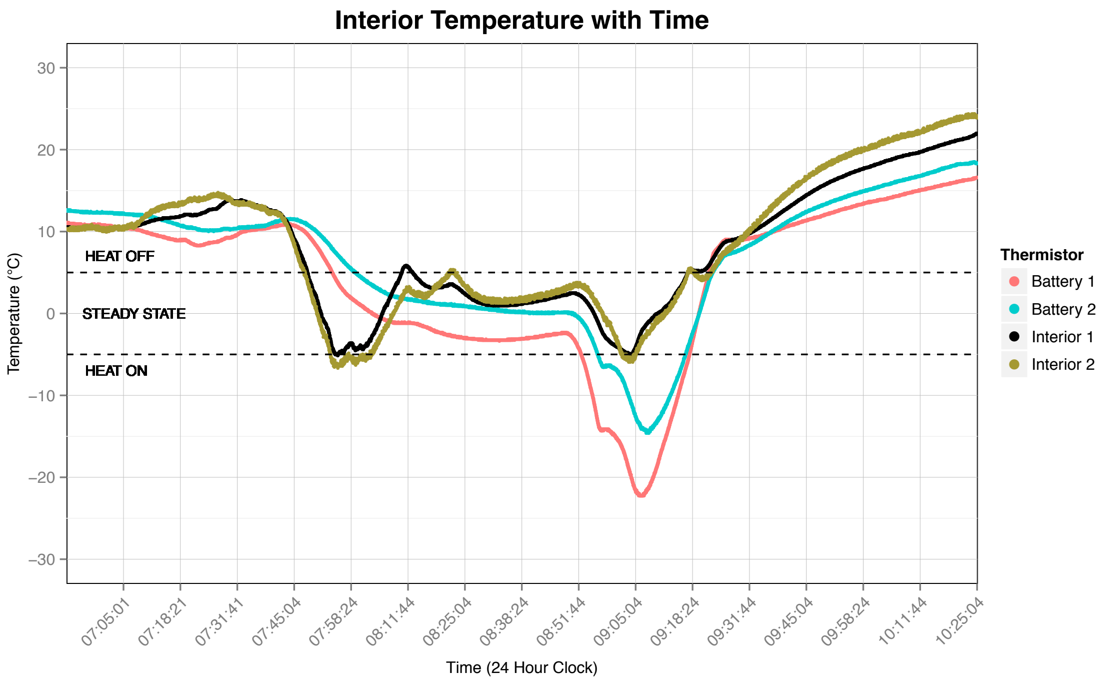
9.3 Temperature Sensors

We used four thermistors to record exterior temperatures the payload encountered. There are two minimum points recorded, each around -40 ℃. The relative maximum temperatures were recorded at takeoff and landing. Upon landing, the exterior reached around 20 ℃. Refer to figure 14a for exterior temperature plot.

We also used four thermistors to record interior temperatures, two near our batteries, and two near our payload bus. Two minimums can be observed with the two payload bus thermistors, sporting a minimum -5 ℃. -5 ℃ was the temperature minimum programmed into our thermal system. Our battery heaters malfunctioned, and a strange slope can be observed where only one minimum can be observed, and that reached around -22 ℃. The plot of interior temperature is shown in figure 14b.



(a) External thermistors.

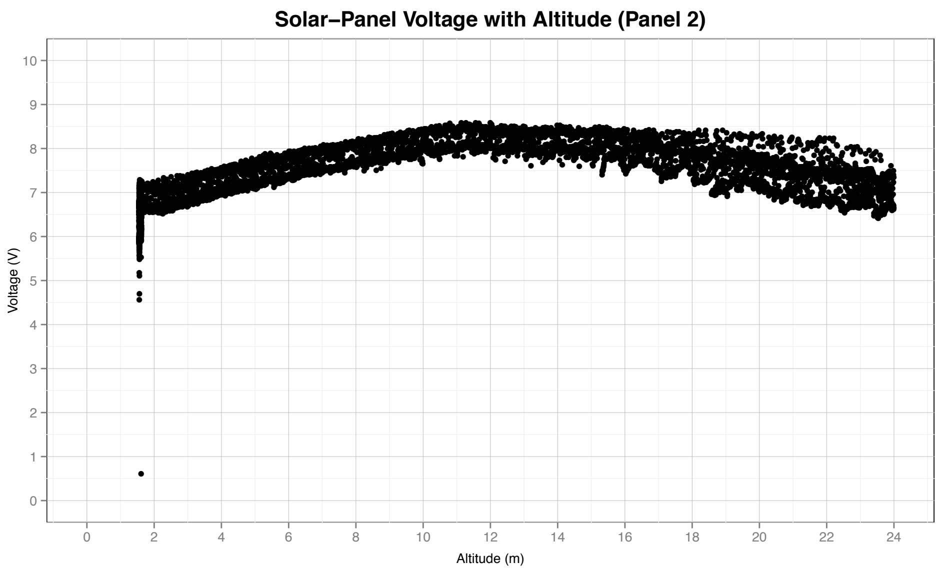


(b) Internal thermistors.

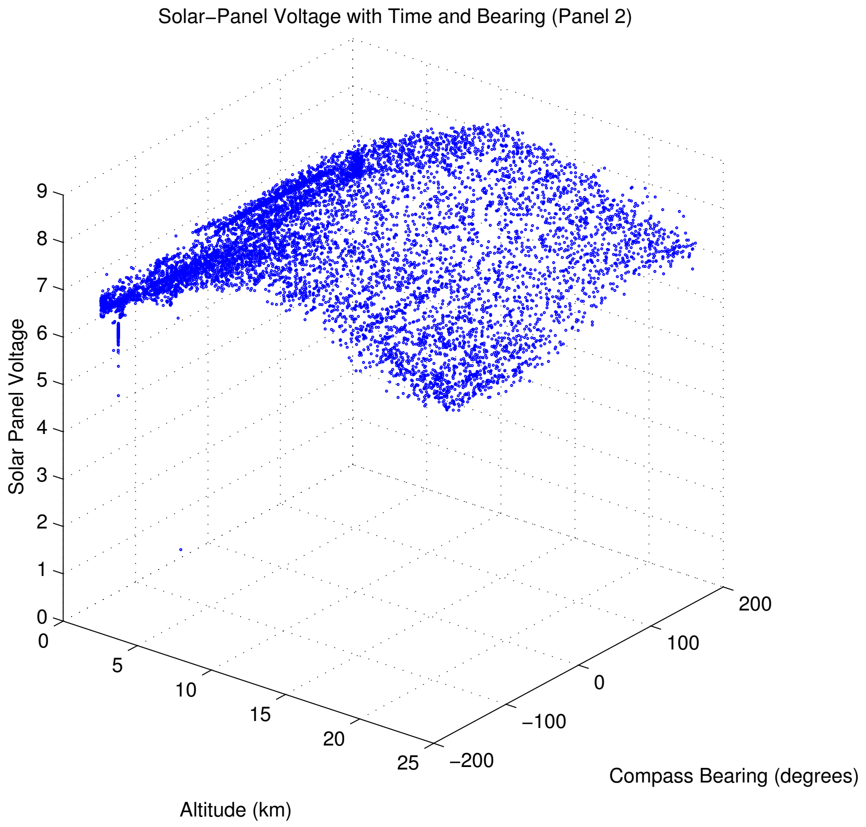
Figure 14: Temperature with time.

9.4 Solar Panels

Figure 15 on the next page shows the potential difference generated by solar panel two with altitude. Due to rotation of the spacecraft during flight, the data points plotted in figure 15a are encompassed by a wide envelope. The compass-bearing dimension is added in figure 15b and shows the variation of the panel's voltage with the payload's angular position about the flight cord.



(a) With altitude.



(b) With altitude and compass bearing.

Figure 15: Solar panel voltage (panel 2).

A plot of solar-panel voltages with mission time is depicted in figure 16 on the following page for compass-bearings between 0 and 90 degrees and compass elevation between 20 and 160 degrees (90 degrees is horizontal). The data points are tighter, most likely to the bearing and elevation restrictions.

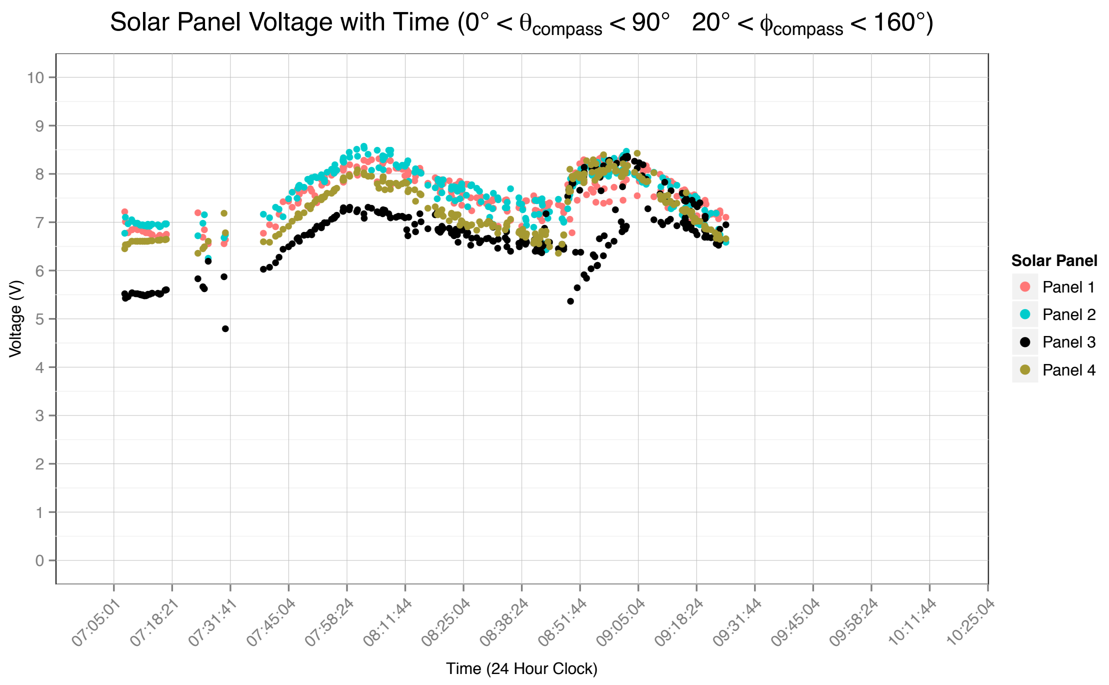


Figure 16: Solar-panel voltage with time (limited compass bearing).

9.5 Housekeeping Data

Shown in figure 17 is the battery voltage of our power system over the flight time. Upon launch, the battery read 8.1V. When the payload landed, the battery voltage was 7.5V. From figure 14b on page 17, we can observe a temperature increase where the batteries began powering the heaters. In this time frame, we also see a decrease in battery voltage.

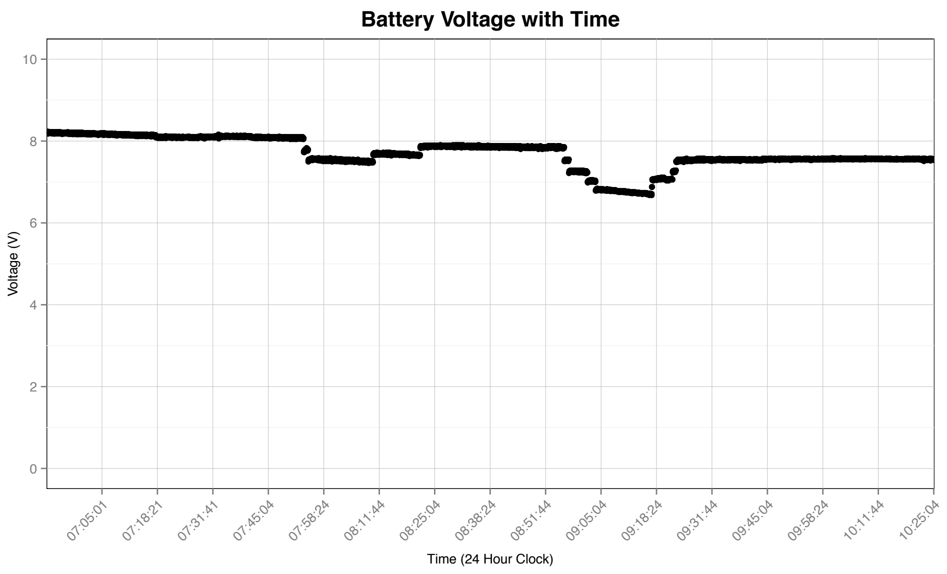


Figure 17: Potential difference between battery terminals with time.

10 Ready for Flight

To prepare the payload for another flight, Team Apogee must do the following:

10.1 Primary Adjustments

- Revise the radiation shielding experiment by researching and testing the magnetic fields created by the permanent magnets, adjusting position of Geiger counters to avoid interference, and shielding each Geiger counter from internal interference.
- Reconnect and securely solder the disconnected ground wire that may have led to Geiger counter failure on the initial flight.
- Adjust the position of power system heaters to properly heat batteries during ascent and descent.
- Adjust position of UV sensor to avoid extreme cold while maintaining an effective orientation.

10.2 Maintenance

- Recharge or replace each battery used during the initial flight.
- Check each sensor, wire, power, and data connection.
- Power-on payload and test each system for functionality. Team Apogee's payload must be sealed at the top and bottom access panels in preparation for flight. It is activated for flight by flipping the toggle switch on the outside of the payload. Once activated, the payload operates autonomously until it is deactivated post-flight.

11 Conclusions and Lessons Learned

Based on the data presented in the previous section, our objectives have been satisfied. Primary objectives were partially achieved. We can conclude that ionization radiation increases with altitude and that temperature varies with altitude. All systems performed as expected except anonymous heater failures and one geiger counter. We learned significant information on creating a payload with printed circuit boards and other high end electrical technics. We also expect the skills learned throughout teamwork and this mission's design will aid our future endeavors, regardless of our separate career or major choices.

12 Message to Next Year

We offer these last words of advice to future DemoSat teams based on our experiences.

12.1 General Suggestions

- Read the DemoSat User Guide and Website. The user guide contains testing and design review guidelines needed to pass design reviews and to be prepared for a successful launch. The website contains key deadlines and design review and document templates. Each of these things is crucial to meeting theDemoSat requirements for launch.
- Determine All Tasks and Delegate Each Task. It is important to determine each task to be done and to delegate each and every task to a particular team member(s). This gives each team member responsibility and accountability, spreads the workload, and avoids losing track of important tasks. It may be of benefit to the team to determine group leaders. Additionally, do as much early research and planning as possible. This will give the team a greater understanding of the scope of the project and the number of tasks to be accomplished.
- Be Aware of Key Deadlines and Plan Each Week Accordingly. Important deadlines include design reviews and paper submissions. Because the team meets only once a week, these deadlines approach quickly. Create a work plan for each week that moves the team toward accomplishing the tasks needed to meet each deadline.
- Communicate Effectively. Communication is one of the most important aspects of any team project. Find a method of communication (email, phone, etc.) that works well for each team member and make use of it.

- Test Early and Often. Testing each phase of the project is crucial to success during the flight. Check the DEMOSAT user guide for testing guidelines and complete each test as early as possible. This allows the team to uncover weaknesses in the design of the payload and to make adjustments in a timely manner. It is likely that the team will need to re-test several times a particular piece of the payload design.
- Take Inventory and Stay Organized. It is important for the team to know what items are available for it to use. Early in the project, take an inventory of each item and note what may need to be purchased, replaced, or replenished. Keeping the inventory organized saves the team precious time, its most valuable commodity.
- Work Outside of Sessions. Depending on the scope of the project, the team may need to work outside of each weekly session – especially before critical deadlines such as design reviews and launch. Communication and forward-thinking are important to the success of the work done outside of weekly sessions.
- Attend Symposiums. There are countless benefits to presenting at the available symposiums. Presenting the team project at a symposium gives the team valuable experience in presentations, public speaking, and building education and industry contacts. The symposiums also allow the team to compete for cash prizes and to be inspired by exposure to the space and engineering industries.

12.2 Specific Suggestions

- Use #slack. For communication purposes, Team Apogee used an app called #slack for PC, Mac, Android, and IOS mobile devices. #slack is a simple chat program that allows quick and simple communication, the creation of individual chat channels for each piece of the project, and the ability to upload files (images, video, text documents, etc.) for sharing with the team. The importance of #slack to Team Apogee simply cannot be overstated - it was critical to their success.
- Use Trello. For task management and delegation purposes, Trello is an excellent piece of software for PC, Mac, and mobile devices. Team Apogee did not make proper use of Trello, and their effectiveness suffered because of it. Learn from Team Apogee's mistakes!
- Use Great Stuff. This insulating spray foam is aptly named. It is light: density of 0.0211 g/cm^3. It is durable: during Team Apogee's drop- and stair-tests, the payload bounced off the concrete, completely undamaged. It has excellent thermal insulating properties: an R-value of 6.6 at a thickness of one inch.
- Use Rechargeable Batteries. Using rechargeable batteries saves money and the environment. It also makes payload power-system testing easier by allowing the team to recharge batteries before every test, rather than guess at which disposable batteries have enough power for each test and/or launch. Remember, though, that recharging takes time. So, charge them every session and keep them charged before important tests and the launch date.

References

[1] Freescale Semiconductor. (2012, July). I2C Precision Altimeter. Retrieved June 23, 2014, from SparkFun Electronics: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Pressure/MPL3115A2.pdf

[2] Fried, L. (2012, July 29). Themistor. Retrieved July 22, 2014, from Adafriut: https://learn.adafruit.com/thermistor

[3] Recktenwald, G. (2010). Thermistor Calibration: T = F(R). Retrieved July 22, 2014, from http://web.cecs.pdx.edu/~eas199/B/howto/thermistorCalibration/thermistorResistanceCalibration.pdf

[4] Wertz, J. R., & Larson, W. J. (Eds.). (1999). Space Mission Analysis and Design (3rd ed.). Torrance, California: Microcosm Press.

13 Appendix: Flight-Computer Code

```
// ApogeePayloadFinal.ino
//
// Written by Wes Hileman on 22 March 2015
//
// Purpose: To control the Apogee payload during flight.

// INCLUDES

// Spacecraft-bus libraries
#include <avr/wdt.h>                  // watchdog timer
#include <RTC.h>                      // real-time clock
#include <MUX.h>                      // multiplexer
#include <Heater.h>                   // heating pads
#include <SendOnlySoftwareSerial.h>   // serial Tx for OpenLogs and LCD
#include <VoltageDivider.h>           // voltage monitor and solar panels

// Sensor libraries
#include <Wire.h>        // I2C communication library
#include <MPL3115A2.h>   // altimeter
#include <HMC5883L.h>    // triple-axis magnetometer
#include <ML8511.h>      // uv sensor
#include <ADXL335.h>     // triple-axis accelerometer
#include <Thermistor.h>  // thermistor temperature-sensors

// CONSTANTS

// Aliases for multiplexer pins
enum MuxPin { M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15 };

#define ANALOG_REF 3.3
#define WATCHDOG_TIMEOUT WDTO_8S
#define LOG_BAUD 9600
#define LCD_BAUD 9600

// Temperature control
#define BATT_CENTER_TEMP 0
#define BATT_EPSILON 5
#define INT_CENTER_TEMP 0
#define INT_EPSILON 5

// Timing
#define CYCLE_TIMEOUT 1000  // milliseconds
#define SLEEP_TIME 10       // microseconds
#define DIAGNOSTIC_TIME 100 // milliseconds

#define ALTIMETER_OVERSAMPLE_RATE 7
#define COMPASS_AV_SAMPLES AV8
#define COMPASS_SR F3HZ
#define COMPASS_GAIN G1370
#define COMPASS_MODE CONTINUOUS

// ========== PIN MAPPING ==========

#define MUX_IO_PIN A1
#define DATA_LOG_TX_PIN 10
#define BATT_HEAT1_PIN 4
#define BATT_HEAT2_PIN 5
#define INT_HEAT1_PIN 6
#define INT_HEAT2_PIN 7
#define TEST_BUTTON_PIN 12
#define LCD_PIN 8
#define VOLT_METER_PIN M4

#define NUM_MUX_SELECT_PINS 4
// Multiplexer select pins
int muxSelectPins[ NUM_MUX_SELECT_PINS ] = { A5, A4, A3, A2 };

#define GEIGER1_INTERRUPT 2
```

```
#define GEIGER2_INTERRUPT 3
#define UV_PIN A0
#define ACCEL_X_PIN 15
#define ACCEL_Y_PIN 14
#define ACCEL_Z_PIN 13
#define BATT_THERM1_PIN M12
#define BATT_THERM2_PIN M11
#define INT_THERM1_PIN M10
#define INT_THERM2_PIN M9
#define EXT_THERM1_PIN M8
#define EXT_THERM2_PIN M7
#define EXT_THERM3_PIN M6
#define EXT_THERM4_PIN M5
#define SOLAR_PANEL1_PIN M0
#define SOLAR_PANEL2_PIN M1
#define SOLAR_PANEL3_PIN M2
#define SOLAR_PANEL4_PIN M3

// ======== END PIN MAPPING ========

// OBJECTS

// Spacecraft-bus objects
RTC clock;
MUX mux( muxSelectPins, NUM_MUX_SELECT_PINS );
SendOnlySoftwareSerial datalog( DATA_LOG_TX_PIN );
SendOnlySoftwareSerial lcd( LCD_PIN );
// Thermistors that control heaters
Thermistor battTherm1( MUX_IO_PIN, 10040.0, 3950.0, 293.2, 12900.0, &mux, BATT_THERM1_PIN
   );
Thermistor battTherm2( MUX_IO_PIN, 9960.0, 3950.0, 292.1, 13350.0, &mux, BATT_THERM2_PIN
   );
Thermistor intTherm1( MUX_IO_PIN, 9950.0, 3950.0, 292.1, 13280.0, &mux, INT_THERM1_PIN );
Thermistor intTherm2( MUX_IO_PIN, 9940.0, 3950.0, 292.1, 13570.0, &mux, INT_THERM2_PIN );
// Heater objects (require thermistor objects)
Heater battHeater1( BATT_HEAT1_PIN, BATT_CENTER_TEMP, BATT_EPSILON, &battTherm1 );
Heater battHeater2( BATT_HEAT2_PIN, BATT_CENTER_TEMP, BATT_EPSILON, &battTherm2 );
Heater intHeater1( INT_HEAT1_PIN, INT_CENTER_TEMP, INT_EPSILON, &intTherm1 );
Heater intHeater2( INT_HEAT2_PIN, INT_CENTER_TEMP, INT_EPSILON, &intTherm2 );
// Battery-voltage monitor
VoltageDivider battMeter( VOLT_METER_PIN, 9910, 4620, ANALOG_REF, &mux, MUX_IO_PIN );

// Sensor objects
MPL3115A2 alt;
HMC5883L compass( COMPASS_AV_SAMPLES, COMPASS_SR, COMPASS_GAIN, COMPASS_MODE );
ML8511 uvSen( UV_PIN, ANALOG_REF );
ADXL335 accel( ANALOG_REF, ACCEL_X_PIN, ACCEL_Y_PIN, ACCEL_Z_PIN, &mux, MUX_IO_PIN );
// Thermistors
Thermistor extTherm1( MUX_IO_PIN, 9650.0, 3950.0, 292.1, 13630.0, &mux, EXT_THERM1_PIN );
Thermistor extTherm2( MUX_IO_PIN, 9990.0, 3950.0, 292.1, 13630.0, &mux, EXT_THERM2_PIN );
Thermistor extTherm3( MUX_IO_PIN, 9880.0, 3950.0, 292.1, 13430.0, &mux, EXT_THERM3_PIN );
Thermistor extTherm4( MUX_IO_PIN, 9900.0, 3950.0, 292.1, 13250.0, &mux, EXT_THERM4_PIN );
// Solar panels
VoltageDivider solar1( SOLAR_PANEL1_PIN, 9920, 4590, ANALOG_REF, &mux, MUX_IO_PIN );
VoltageDivider solar2( SOLAR_PANEL2_PIN, 9870, 4570, ANALOG_REF, &mux, MUX_IO_PIN );
VoltageDivider solar3( SOLAR_PANEL3_PIN, 9870, 4650, ANALOG_REF, &mux, MUX_IO_PIN );
VoltageDivider solar4( SOLAR_PANEL4_PIN, 9940, 4620, ANALOG_REF, &mux, MUX_IO_PIN );

// Geiger-counter counts
unsigned long geigerCount1 = 0, geigerCount2 = 0;

void countGeiger1( void )
{
  geigerCount1++;
} // end countGeiger1

void countGeiger2( void )
{
  geigerCount2++;
} // end countGeiger1

```

```
void setup()
{
  // Initialize I2C communication
  Wire.begin();

  // ========== SPACECRAFT BUS ==========

  // Initialize real-time clock
  clock.begin();
  // Initialize MUX I/O pin as input
  pinMode( MUX_IO_PIN, INPUT );
  // Initialize data logging
  datalog.begin( LOG_BAUD );
  // Initialize heaters
  battHeater1.begin();
  battHeater2.begin();
  intHeater1.begin();
  intHeater2.begin();
  // Initialize test-button input
  pinMode( TEST_BUTTON_PIN, INPUT );
  // Initialize lcd
  lcd.begin( LCD_BAUD );
  // Send calibration code to lcd
  lcd.print("?f?c0?G420?f");
  // Initialize battery voltage-monitor
  battMeter.begin();

  // Set analog reference voltage to external AREF (3.3V regulator)
  analogReference( EXTERNAL );
  // Enable watchdog timer
  wdt_enable( WATCHDOG_TIMEOUT );

  // Print data-file header
  datalog.println( "Time (ms), Timestamp, Pressure (Pa), UV Int (mW/cm^2), Comp Theta (
    deg), Comp Phi (deg), Comp Mag (G), X Accel (g), Y Accel (g), Z Accel (g), Batt T1 (C
    ), Batt T2 (C), INT T1 (C), INT T2 (C), EXT T1 (C), EXT T2 (C), EXT T3 (C), EXT T4 (C
    ), Heat1, Heat2, Heat3, Heat4, GC1 (counts/cycle), GC2 (counts/cycle), Batt V, Solar1
     (V), Solar2 (V), Solar3 (V), Solar4 (V)" );

  // ============ PAYLOAD ============

  // Initialize altimter
  alt.begin( ALTIMETER_OVERSAMPLE_RATE );
  alt.barometer();
  // Initialize magnetometer
  compass.begin();
  // Initialize UV sensor
  uvSen.begin();
  // Initialize accelerometer
  accel.begin();
  // Initialize Geiger-counter interrupts
  attachInterrupt( GEIGER1_INTERRUPT, countGeiger1, RISING );
  attachInterrupt( GEIGER2_INTERRUPT, countGeiger2, RISING );
  // Initialize solar panels
  solar1.begin();
  solar2.begin();
  solar3.begin();
  solar4.begin();
} // end setup

void loop()
{
  // Update the start time of the latest cycle
  unsigned long lastCycleTime = millis();
  // String to store timestamp data
  char timestamp[ TIMESTAMP_LENGTH ] = "";
  // Variables to store sensor data
  float pressure, uv, compassTheta, compassPhi, compassMag, accelX, accelY, accelZ;
  float battTemp1, battTemp2, intTemp1, intTemp2, extTemp1, extTemp2, extTemp3, extTemp4;
  float battV, solar1V, solar2V, solar3V, solar4V;
  unsigned long gCount1, gCount2;
  bool heat1State, heat2State, heat3State, heat4State;

  // Reset watchdog timer
  wdt_reset();

  // Collect data
  clock.timestamp( &timestamp[ 0 ] );
  pressure = alt.pressure();
  uv = uvSen.intensity();
  compass.data( &compassTheta, &compassPhi, &compassMag );
  accelX = accel.accelerationX();
  accelY = accel.accelerationY();
  accelZ = accel.accelerationZ();
  battTemp1 = battTherm1.temp();
  battTemp2 = battTherm2.temp();
  intTemp1 = intTherm1.temp();
  intTemp2 = intTherm2.temp();
  extTemp1 = extTherm1.temp();
  extTemp2 = extTherm2.temp();
  extTemp3 = extTherm3.temp();
  extTemp4 = extTherm4.temp();
  gCount1 = readGeiger1();
  gCount2 = readGeiger2();
  solar1V = solar1.voltage();
  solar2V = solar2.voltage();
  solar3V = solar3.voltage();
  solar4V = solar4.voltage();

  battV = battMeter.voltage();

  // Drive heaters
  heat1State = battHeater1.cycle();
  heat2State = battHeater2.cycle();
  heat3State = intHeater1.cycle();
  heat4State =  intHeater2.cycle();

  datalog.print( lastCycleTime );
  datalog.print( ", " );
  datalog.print( timestamp );
  datalog.print( ", " );
  datalog.print( pressure );
  datalog.print( ", " );
  datalog.print( uv );
  datalog.print( ", " );
  datalog.print( compassTheta );
  datalog.print( ", " );
  datalog.print( compassPhi );
  datalog.print( ", " );
  datalog.print( compassMag );
  datalog.print( ", " );
  datalog.print( accelX );
  datalog.print( ", " );
  datalog.print( accelY );
  datalog.print( ", " );
  datalog.print( accelZ );
  datalog.print( ", " );
  datalog.print( battTemp1 );
  datalog.print( ", " );
  datalog.print( battTemp2 );
  datalog.print( ", " );
  datalog.print( intTemp1 );
  datalog.print( ", " );
  datalog.print( intTemp2 );
  datalog.print( ", " );
  datalog.print( extTemp1 );
  datalog.print( ", " );
  datalog.print( extTemp2 );
  datalog.print( ", " );
  datalog.print( extTemp3 );
  datalog.print( ", " );
  datalog.print( extTemp4 );
  datalog.print( ", " );
  datalog.print( heat1State ? "ON" : "OFF" );
  datalog.print( ", " );
  datalog.print( heat2State ? "ON" : "OFF" );
  datalog.print( ", " );
  datalog.print( heat3State ? "ON" : "OFF" );
  datalog.print( ", " );
  datalog.print( heat4State ? "ON" : "OFF" );
  datalog.print( ", " );
  datalog.print( gCount1 );
  datalog.print( ", " );
  datalog.print( gCount2 );
  datalog.print( ", " );
  datalog.print( battV );
  datalog.print( ", " );
  datalog.print( solar1V );
  datalog.print( ", " );
  datalog.print( solar2V );
  datalog.print( ", " );
  datalog.print( solar3V );
  datalog.print( ", " );
  datalog.println( solar4V );

  // Wait until the cycle times out
  while( millis() - lastCycleTime < CYCLE_TIMEOUT )
  {

    if( digitalRead( TEST_BUTTON_PIN ) == HIGH )
    {
      runDiagnostics();
    } // end if

    delayMicroseconds( SLEEP_TIME );
  } // end while

} // end loop

void runDiagnostics()
{
  char timestamp[ TIMESTAMP_LENGTH ] = "";
  float compassTheta, compassPhi, compassMag;
  clock.timestamp( &timestamp[ 0 ] );
  compass.data( &compassTheta, &compassPhi, &compassMag );

  // Clear screen
  lcd.print( "?f" );
  // Output diagnostic data
  lcd.print( "B1T " );
  lcd.print( battTherm1.temp() );
  lcd.print( " B2T " );
  lcd.print( battTherm1.temp() );
  lcd.print( "?n" );
  lcd.print( "Ax" );
  lcd.print( accel.accelerationX() );
  lcd.print( "Mg" );
  lcd.print( compassMag );
  lcd.print( " S" );
  lcd.print( solar1.voltage() );
  lcd.print( "?n" );
  lcd.print( "Alt" );
  lcd.print( alt.isRunning() ? "OK" : "NK" );
  lcd.print( " UV" );
  lcd.print( uvSen.intensity() );
  lcd.print( "B" );
  lcd.print( battMeter.voltage() );
  lcd.print( "V" );
  lcd.print( "?n" );
  lcd.print( timestamp );

  delay( DIAGNOSTIC_TIME );
} // end runDiagnostics
```

```

unsigned long readGeiger1()
{
  unsigned long gc1 = geigerCount1;
  geigerCount1 = 0;

  return gc1;
} // end readGeiger1

unsigned long readGeiger2()
{
  unsigned long gc2 = geigerCount2;
  geigerCount2 = 0;

  return gc2;
} // end readGeiger2
```