ARAPAHOE COMMUNITY COLLEGE
DEMOSat
DESIGN DOCUMENT

High-precision Single-axis Fluxgate Magnetometer

DemoSat Team:
Justin Anderson, Cameron Bain, David Colclazier, Tasha Estein, Magdalena Franchois, Spencer Jackson, David Oelgoetz, Lawrence Perkins, and Bri Treffner
December 16 2016
Revision C
Table of Contents

1.0 Mission Overview
2.0 Requirements Flow Down
3.0 Design
4.0 Management
5.0 Budget
6.0 Test Plan and Results
7.0 Expected Results
8.0 Launch and Recovery
9.0 Results and Analysis
10.0 Ready for Flight
11.0 Conclusions and Lessons Learned
12.0 Message to Next Year
1.0 Mission Overview

Our mission was to send three different magnetometers, two of which were of the fluxgate design, into the upper atmosphere and measure the Earth’s magnetic field in reference to different altitudes. We also intended to measure the effects of temperature and movement on the magnetometer readings by comparing them to the known values of the magnetic field at different altitudes.

2.0 Requirements Flow Down

Objective Requirements

● Build a fluxgate magnetometer capable of detecting the Earth’s magnetic field.
● Measure how much the magnetometer readings can be affected by the varying temperature during flight.
● Measure to what extent motion can affect magnetometer readings.
● In order to measure a change in Earth’s magnetic field every 100 feet of ascent, the fluxgate magnetometer needs a precision of approximately $2.06 \times 10^{-1}$ nT.
● External components will need to withstand external environmental changes

System Requirements

● Must draw an acceptable amount of power, as well as being under our weight limit.
● Temperature changes must be tracked in order to see effect on data.
● Sensor updates from the custom magnetometer will need to be collected at a frequency of around 1.3kHz for proper post-analysis.
● A Bosch BNO055 sensor breakout developed by Adafruit will be included in payload to measure temperature, gravity, acceleration, angular velocity, orientation, linear acceleration as well as provide alternate magnetometer readings. The associated datasheet for this sensor is in the appendix (section 1.1). It does not have the capability of measuring changes in Earth’s field, and will be used exclusively for temperature fluctuation measurements.
● A Bosch BMP180 sensor breakout developed by Adafruit will also be included to measure atmospheric pressure, temperature, and altitude. The datasheet for this sensor is in the appendix (section 1.2).
● A commercial fluxgate magnetometer, the 113D from Applied Physics Systems, will be integrated to provide an alternative measurement of magnetic fields experienced during flight. It was specially selected because of its capabilities to measure the expected changes in magnetic field.
● At least one magnetometer needs to be thermally isolated from external temperatures and stay at around 27 degrees Celsius in order to provide accurate measurements.
Subsystem Requirements

- We require a battery that will last up to 1.5 hours while also providing power to our custom-designed magnetometer, which can take up to 8 watts of power.
- Three thermometers were included on payload, each integrated into its sensor - the BNO055, the BMP180, and the commercial magnetometer from Applied Physics System.
- The requirements of the payload software logic was as follows:
  - Keep track of payload runtime for timing sensor events.
  - Collect calibrated orientation, linear acceleration, gravitational field readings, barometric pressure, external temperature, and 3 sets of magnetometer data (two 3-axis, one 1-axis) coupled with their temperature readings.
  - Allow for the throttling of sensor events to increase or decrease collection frequency as required.
  - Log all collected data to SD card in the most efficient way possible, while ensuring that the SD card logging does not directly affect the periodicity of sensor collection events in ways that would negatively affect our analysis.
  - Display boot sequence and sensor calibration data for user-calibration prior to flight.
  - Monitor memory usage such that if the payload runs out of RAM, we pause sensor actions until the memory containing data waiting to be flushed to the SD card is cleared from RAM.

3.0 Design

A common theme in design choice for the structure and design of this years payload was based on our successes from previous payloads. Using a closed cell foam has proven to be beneficial to the overall weight of the payload simply because the foam has a lesser weight than other considered materials such as balsa wood, cardboard, and 3D printed plastic. One goal was to minimize interference from external sources and between our own components in order to provide the cleanest sensor data possible. To accomplish this, we attempted to maximize the distance between each component of the payload by placing most of our electronics as far apart as possible, some above and some below the structure, while maintaining the same axis orientation being measured by the various sensors.

The fluxgate magnetometer core was designed around a manganese-zinc toroidal core that measures 2.401” outer-diameter, 1.275” inner-diameter, and 0.945” in height. Using 30 gauge copper magnet wire, we manually wound 800 turns in several layers separated...
by electrical tape. This was then placed in a simple box-like enclosure which was then wound again with 344 turns orthogonal to the direction of the axis we intend to sense.

A single piece of closed cell foam 30cm in diameter was our only rigid structure. We used hot glue and silicone to secure the various components.
We placed the various batteries, microcontroller, and other temperature-sensitive electronics in black high-insulating light-weight foam, creating several containers that could be independently environmentally controlled. We experimented with several heating methods, from the most basic aspect of having a large ceramic resistor suspended within the box to a more intricate design with several resistors woven into flat pads designed for consistent heating each surface of the box. For final flight the padded heaters were removed due to their high energy cost on the system. The more simple ceramic design was far more power efficient.

There are a few complications that this payload design has introduced. A worry from early on was that the flight string would entangle itself around the external electronics enclosures during flight, possibly damaging the payload. Another worry that the single plane open payload design brings is the ability to stay connected to the flight string during the launch and then stay in place for the duration of the flight, as well as ensuring that no components fall off due to wind-shear or other force experienced during flight. To resolve this we simply extended the flight tube such that it was longer both above and below the payload. We used silicone to attach the washers to both the flight-tube and the payload, which ended up proving successful.
Several components of the custom fluxgate magnetometer were placed externally to the payload for lack of space and the necessity of not coming into contact with other components, or being near enough to them to affect them due to the necessarily high voltage running through parts of the device. In addition, the BNO055 sensor, shown in the picture below, was placed externally to allow us to measure an external temperature.

The final payload weighed in at around 1150 grams, slightly over the weight requirement provided by the COSGC. This proved to be a major disappointment to several members of our team; meeting the weight limit has been a consistent challenge for ACC for the past few DemoSat launches... this was no exception.
3.1 Electronics

The electronics team aimed to match the design specifications from similar off-the-shelf fluxgate magnetometers as well as provide electrical support for the entire payload. Our original design goals were to:

- Design a power supply and distribution method for the payload and all subsystems.
- Supply high voltage at low current to the excitation coil at 1-5 kHz.
- Amplify sense coil signal to flight computer.

It should be noted that our final design used a core that was substantially larger than similar designs. As a result, we were unable to drive this core into saturation at such high frequencies during testing. We found that the core responded optimally at 60Hz with no less than 30 volts AC. This changed our power requirements rather drastically.

Our design requires alternating current to drive the coil into saturation. To accomplish this task, the electronics team choose to use multiple operational amplifiers in a single package called an LM324 quad amp (link in appendix 2.2) to generate alternating current. The LM324 required a bipolar power supply to generate both sides of a sinusoidal waveform. To generate this supply, two nine volt batteries powered two separate five volt regulators which were arranged to provide both positive and negative voltages with a common rail between them. This generated an isolated bipolar power supply that could be used next to the main 5 volt power grid.

Bipolar Power Supply:

![Bipolar Power Supply Circuit Diagram](image)

The LM324 exclusively used the bipolar power supply to generate a sinusoidal waveform in three stages. The first stage created a square waveform with an amplitude of 5 volts. The second stage refined the waveform into a triangle waveform with an amplitude of 1.4
volts. The final stage further refined the waveform to a sinusoidal waveform with an amplitude of about 250 millivolts.

Signal Generator:

The resulting signal output from this stage was a stable sinusoidal waveform at 250 mv AC oscillating at 63.5 hertz.
After generating the proper AC signal, the amplitude (voltage) had to be boosted higher in order to drive the core into saturation. Normally a much lower voltage would be used; however, the core we ended up using required a much larger amount of power to drive into saturation. Many methods were used to amplify the signal but unfortunately, some designs were changing the shape of the signal to an extreme degree. Of the designs we tested with, we had the best results with using opposing transistors of the NPN type to amplify the signal while maintaining a suitable sinusoidal waveform. Unfortunately, this design uses quite a large amount of power to generate the boosted signal. A large portion of the power used was lost as heat in this circuit. The excitation amplifier required a DC voltage source greater than 30 Volts DC. After testing on a bench, it was determined that the core was entering further into saturation when using a supply capable of 60 volts DC. To achieve this two 5 volt DC to 30 volt DC power boosts called RRLM 2587 (link in appendix 2.4) were sourced and wired in series to provide a clean 60 volt DC supply for the excitation coil. Of the first versions of this design, we were easily exceeding the amount of heat the last transistor was able to dissipate. As a result, the transistor was replaced with an IRF450 (link in appendix 2.3) junction gate field-effect transistor (JFET) for the higher heat dissipation capacity and similar operational characteristics to an NPN transistor. To help limit the current, a 400 ohm 4 watt resistor was used in series with the excitation coil. These design modifications allowed for the use of higher currents and voltages without the risk of overheating and overloading of components. 

Signal Amplifier:

At this stage, the excitation coil was able to drive the core into saturation at the cost of heavy power consumption and excess heat generation. The heat would not go to waste as it was used to heat critical components during the flight.
The next goal was to amplify the signal coming from the sense coil to a voltage range that the flight computer could detect. To accomplish this a simple NPN transistor and operational amplifier combination was utilized to boost the sense coil output up to a 5V signal. The fourth operational amplifier could be utilized from the LM324 package using all four op amps in the package. It works by simply scaling the signal up by a factor of 50 to a point where the flight computer could read the signal. The sense coil amplifier is less powerful than the excitation coil amplifier due to the reduced current and voltage requirements.

Sense Amplifier:

The sense coil is wrapped around the entire core. It detects harmonics from the excitation coil when the core is in saturation. To demonstrate a magnet was passed over the core and the signal was read with an oscilloscope. The picture on the left is without the magnet modifying the field and the picture to the right is with a magnet passing within 3” of the core.
The design of the AMDAS (ACC Mag Driver/Amp Support) board was the last step in building the custom fluxgate magnetometer. This board allowed for the organization and structural support for the high voltage bus, the isolated bipolar power supply, the signal generator and amplifier, and the sense coil amplifier.
3.2 Software

Our original microchip of choice was an Arduino mini, simply because of its popularity and the myriad of sensor drivers and other possibly usable scripts available online. However, with a 16MHz crystal and 2kb SRAM, we knew it would be a challenge to get everything working properly and operating quickly enough. Upon finishing the first variant of our flight computer, it proved to be next to impossible, as the flight computer ran out of memory within a few seconds of being powered up (Appendix 3.1). To resolve this issue, we developed a barebones system that, without a lot of the advanced concurrency and memory management the other system offered, allowed sensor logging and data collection at a frequency of about 16 Hz. (Appendix 3.2)

It was later determined that the waveform from the custom magnetometer needed to be sampled at least 20-30 updates per cycle, at a cycle period of around 60 Hz. This requirement required a complete rework of the flight computer on a different microcontroller, as those speeds are simply unattainable on an Arduino also performing the other tasks required of our payload. We chose, with a bit less than 2 weeks to spare, the Netduino 3 Wifi, with 32Kb RAM and a 168MHz processor - more than 10 times faster than the Arduino. This microcontroller uses the .NET Micro Framework, one disadvantage of which is that existing libraries sensor drivers are practically non-existent. Because of this, we were forced to develop custom drivers for all of our sensors.

In order to achieve the high update frequency of the custom magnetometer, we needed the ability to slow down events that didn’t need as high an update frequency. To solve this, we chose to advantage of the advanced task scheduling capabilities of the .NET Micro Framework to allow for concurrently running processes on a single core. This allowed us some advanced functionality for controlling the flow and execution of our events by allowing slower events to pass cpu execution time to threads with a higher update frequency requirement.

Execution of updates and actions on the flight computer was handled using the concept of a thread-pool. One of the persistent threads running on the flight computer constantly monitors a queue of “work items”; executable code which performs some task on the flight computer: logging debug statements, updating sensors, writing to the SD card, etc. When a work item needs to be executed, it is added to the list of pending actions the thread-pool monitors. Once a work item appears in this list, it is executed in the order in which it arrived, much like a queue. Some work items, like sensor updates, are marked as repeatable such that they are added back to the thread-pool’s pending work items upon completion of its action. For sensor updates which require a much higher frequency than the thread-pool can offer (a repeatable action can only run once per every other repeatable action), we used a simple c-style loop to force execution time to remain inside the high-frequency update until it finished collecting a particular number of updates.
sequentially. By assuming that the period in between each update was constant based on the generally constant load of the processor, we simply collected a timestamp before the update started and after it finished and used them to interpolate the times for each update in between.

We also had to get creative with the data-logging portion of the software, our goal being to ensure it was both as efficient and quickly executing as possible. Each sensor-update related work item was designed to parse the data for the sensor into a binary packet to be more quickly and easily manageable by the data-logging system than the original UTF representation of the data. We were also able to achieve awesome compression for high-frequency update packets by removing the need for a 3-byte time stamp for each update within the packet. For an update packet containing 1200 updates of 1 byte each, this saves approximately 3.52 kb for each packet or around 210 kb/min, allowing more data to be stored on the SD card and thus a longer-lasting logging capability.

The structure for binary data packets is as follows:

<table>
<thead>
<tr>
<th>Start</th>
<th>Data Type</th>
<th>Data Size</th>
<th>Time-Stamp</th>
<th>Data</th>
<th>End Time-Stamp (for compression)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0xFF]</td>
<td>[0x00-0x04]</td>
<td>[1 byte]</td>
<td>[3 bytes]</td>
<td>[n bytes]</td>
<td>[3 bytes]</td>
</tr>
</tbody>
</table>

Some packets did not include the data size byte because they didn’t change their size for each update, whereas some did. Those that did required this in order to properly extract the data upon converting it back into readable data after the flight. In addition, sensor updates not requiring interpolation did not require an end time-stamp.

The flight computer logic uses the following logic after powering on:

- Upon initial power up, the payload enters a simple boot state, initializing communication channels and the status panel, as well the payload logger and internal timer.
- Once this is complete, the payload begins initializing and communicating with the different sensors attached to it.
- After all sensors are initialized, the boot sequence starts the associated work items for each sensor, causing them to be added to the flight computer’s execution queue.
- Once all sensors are updating and logging properly, the status panel switches to a runtime mode, displaying the calibration values on the orientation sensor, allowing us to calibrate it prior to launch.

A UML class diagram for the flight computer follows. In addition, the repository containing all of the flight computer code is available in the appendix (section 3.3).
Because of the nature of the flight data, our development team had to develop another piece of software capable of parsing this data into readable comma-separated-value tables for later analysis. The repository containing the source code for this software is available in Appendix section 3.5.

In addition, our lead developer is working on integrating new features into some software he’s developing called “GraphIt!” which allows for simple graphing and statistical analysis of sensor data. This began during the 2016 RockSat program and is still under active development, but supports automatic updates in case anyone wants to take advantage of the new features as they arrive. A link to download the software is available in Appendix 7.3.

4.0 Management

We initially had around 18 people sign up for DemoSat, presenting quite a challenge to the management team. The primary objective of the management team was to provide an experience for every member that fell in line with what they were wanting to learn, as well as ensuring that we made constant progress throughout the semester. This culminated in forming several different teams. As the project progressed, we unfortunately lost about half of our members - as of launch and the writing of this paper, we have 9 active participants.
We also implemented a rather rigid schedule in order to ensure we had enough time to complete construction and testing of all subsystems prior to flight. Sticking to the schedule proved to be difficult, in the beginning of the project the eagerness to fly and the general excitement of the group kept pushing us, however, when time was tight and things started to get complicated this time table had to be changed into a more organic structure where the completion of each section happened as they could be completed.
5.0 Budget

- **Cost**
  - We were able to complete the entire payload without going over-budget. In fact, we were around $300 under-budget.

- **Weight**
  - Mass was one of the more difficult constraints to conform to. In all of the preliminary testing they payload was set to weigh around 850 grams. There were some final additions made to the payload that were vital to any success, these additions unfortunately added more mass than anticipated and our final weight was 1100 grams. Many suggestions were made to save weight on the payload, everything was discussed from shaving or fluting the foam to scraping some of the circuitry all together. all of the options were found to be destructive to the nature of our payload or they caused a negligible weight difference so the final decision was made to fly overweight.

<table>
<thead>
<tr>
<th>Item Group</th>
<th>Name</th>
<th>Description</th>
<th>Unit Price</th>
<th>Selected for Payload</th>
<th>Quantity Ordered</th>
<th>Quantity On Hand</th>
<th>Power Consumption (mW)</th>
<th>Mass (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronics</td>
<td>Arduino Uno Rev 3</td>
<td>Arduino Uno Rev 3 (5V at 5mA, 16MHz, 10-pin, 5V/3.3V)</td>
<td>$5.85</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Electronics</td>
<td>Arduino Pro Trinket 5V</td>
<td>Arduino Pro Trinket 5V (5V at 5mA, 16MHz, 10-pin, 5V/3.3V)</td>
<td>$5.80</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Sensors</td>
<td>9-DOF Absolute Orientation IMU Fusion Breakout</td>
<td>SEN-1231 - provides absolute orientation IMU fusion</td>
<td>$14.95</td>
<td>Yes</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>Electronics</td>
<td>Image Sensor</td>
<td>Image Sensor</td>
<td>$15.95</td>
<td>Yes</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>Electronics</td>
<td>Lithium Polymer Battery 5V 3.5W 2200 mAh</td>
<td>Battery</td>
<td>$14.50</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

6.0 Test Plan and Results

A whip test was done to the structure of the payload without any of its electronic components. No issues were found. A cold test was done on two components that were not actually used in the final iteration of the payload. Several software tests were performed as the development occurred, and a final runtime test was performed prior to launch to ensure correct operation and runtime. The final changes that were made based on testing were covered in the design portion, they included changes to the flight tube do
to payload design, and the deletion of padded heaters do to their high energy requirements to provide optimal runtime for the payload.

7.0 Expected Results

Our main goal with the magnetometers was to detect changes in Earth’s magnetic field as the payload increased in altitude, and determine how temperature variations affected readings obtained from the sensors. These changes are incredibly small and hard to detect, providing the reason for the commercial magnetometer capable of reaching the necessary levels of precision. The custom magnetometer didn’t quite make it through the calibration process, so our main goal was to retrieve useable data we could perform later analysis on. In addition to our main goals, we planned to include sensors to collect a myriad of different environmental readings for later analysis and correlation with our main goals:

- Gravitational field readings
- Linear acceleration
- Gyroscope data
- Euler heading and quaternion rotation
- Atmospheric pressure and altitude

We tested data collection of all sensors successfully prior to flight. This involved following the procedures below for retrieving data from the payload. We unfortunately did not have time to perform proper analysis on the data for confirmation or calibration purposes, but have used many of the sensors in past projects with success. Unfortunately, the atmospheric pressure sensor failed prior to launch day and a replacement did not arrive in time. As such, even if our micro-SD card had not failed, we wouldn’t have been able to directly measure altitude or atmospheric pressure.

Procedure for retrieving data from payload:

1. Power on payload, ensure successful boot by monitoring messages and waiting for calibration screen to appear.
2. The magnetometer and gyroscope should calibrate almost immediately. There is a known bug with the payload that causes the magnetometer to stay at a calibration state of 0. This is unavoidable if encountered. To calibrate the accelerometer, hold the payload in 6 orthogonal positions for a few seconds, keeping the payload as still as possible at each position.
3. Data collection is handled automatically. When you are ready to retrieve the data, power off the payload and remove the micro-SD card from the microcontroller on board.
4. Power up a computer capable of reading the card and open the parsing application. Drag the binary data file into the same folder as the parsing application, and type the
name of the data file into the console window. From there, multiple .csv files are created and will be placed in the same folder as the binary file and application.

8.0 Launch and Recovery

Cameron Bain was put in charge of taking care of the payload after completion and until launch. Upon recovery of the payload, we passed the micro-SD card off to the software team so they could parse and analyze the data over the next few days after launch. After parsing the binary data file into comma separated value tables the data would have been made available to the rest of the team.

While attempting to recover the payload it was found draped over a barbed-wire fence after reportedly hanging from an electrical wire briefly. The first thing we noticed was that the custom magnetometer we made was no longer attached to the top of the payload. After a minor panic attack we located the custom magnetometer about 5 feet away from the rest of the payload. Two of the power boosters on the side of the payload had also detached from their original position but were still connected to the payload.

Upon retrieving the payload, we immediately noticed that the LCD panel was frozen on a particular update, displaying the current calibration level and a timestamp. Unfortunately we lost the picture containing the exact timestamp. This result was never experienced during our tests, but as we had several components exposed to the elements, occam's razor suggests that some environmental effect (temperature, humidity) was the same culprit, if not some bug that was not discovered prior to flight.

The payload booted successfully upon power-up prior to launch, but without initializing the custom magnetometer event. The only explanation for this is that the event was disabled during testing on the day prior to flight and was inadvertently not re-enabled.

9.0 Results, Analysis, and Conclusions

Upon analysing the different data files on the card, we discovered that at some point during the testing process, the SD card had a critical failure such that although it allowed successful read operations, no write operations (including formatting the card) were recognized. Thus, we did not get any data from the flight. The solution to this for future flights is simple - test the SD card the morning of the launch, and if it doesn’t work, have a spare card or two as a backup.
10.0 Ready for Flight

As with most Space Grant projects we’ve participated in or heard about in the past, we lost a lot of members due to other things - finals, falling behind in class, work, etc. Thus, we haven’t had the time we need to fully prepare the payload for a successful flight. Those on the team who plan on participating in the next BalloonSat flight will be pushing for this as a project. Structurally, the payload is ready to fly again; we shored up construction of external components by reinforcing them with hot glue instead of silicone. We also plan on moving all electronic components such that they are not externally exposed to the elements; this has not been completed yet. As there are no special storage requirements, we plan on keeping the payload in the ACC physics lab under lock and key until we need it again. All components on the payload don’t have a noticeable shelf-life; all components will still be in working order after 6 months.

11.0 Conclusions and Lessons Learned

There is a laundry list of things we might have done differently had we had a second chance. First, we might have revisited the open payload design. The open payload design is an Arapahoe Community College favorite, standing out from the crowd. However, it is not widely known for the best protection from outside variables and temperature control. Though temperature testing and effects on magnetometry was apart the objective and data result goals, it could be a part of the reason we saw some of the failures that we did.

We also would have chosen a smaller ring core. The ring core size we chose added an unnecessary variable in finding the proper number of wrappings of wire around the core. Plus, there would have been far less of a power requirement. In addition, it would have been a reduction of mass of just the ring core, and a reduction of mass from the amount of magnetic wire used to wrap the core.

From a software perspective, we would benefit from some automatic hardware checks during the payload boot in the form of a POST (Power On Self Test), such as performing a test-write and verification read on the SD card. In addition, we’d like to alter the code such that if a particular sensor or hardware component does in fact fail during flight, it does not cause a detrimental affect on the operation of other sensors. Although this didn’t present itself as a direct problem for us, it very much could have existed had the SD card worked to begin with.
12.0 Message to Next Year

Our message is simple: keep it simple, don’t fret, and don’t delay! One of the biggest recurring challenges of projects like this is that the time required to achieve perfect success is much longer than what you think on day 1. In addition, don’t be afraid to try something new! The physics and electronics work involved in this particular project is leaps and bounds beyond the skill level of every member of our team. This provided some frustration, but as you can see, we were able to achieve limited success! In addition, every single member of the team that is participating in classes at ACC next year wants to continue this project to, well, ‘get it right.’ The learning experiences in these projects are profound, and are definitely not subject to any prerequisites. Get involved, be involved, stay involved! That’s the key.

Appendix

1.0 Data Sheets
  1.1 - Bosch BNO055
  1.2 - Bosch BMP085
  1.3 - Applied Physics Systems 3-Axis Fluxgate magnetometer - 113D
  1.3 - LM324
  1.4 - IRF450
  1.5 - RRLM 2587

2.0 Schematic Repos
  2.1 - Electrical Schematics

3.0 Software Repos
  3.1 - Flight computer, rev 1 (C++, built for Arduino platform)
  3.2 - Flight computer, rev 2 (C++, barebones, built for Arduino platform)
  3.3 - Flight computer, rev 3 Final (C#, built for .Net Micro platform)
  3.4 - GraphIt! A sensor-data graphing and statistical analysis program
  3.5 - Flight Computer binary data processing software repo