

DESIGN, DEVELOPMENT, AND IMPLEMENTATION OF THE 2018 ROCKSAT-C DATA MANAGEMENT SYSTEM

A SENIOR PROJECT SUBMITTED TO THE
DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING TECHNOLOGY
OF THE SCHOOL OF ENGINEERING, TECHNOLOGY, AND MANAGEMENT AT THE
OREGON INSTITUTE OF TECHNOLOGY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE

Steven Reeves

© June 2018

SENIOR PROJECT APPROVAL PAGE

The senior project of Steven Reeves for the Bachelor of Science degree was accepted by the evaluation committee and the Department of Computer Systems Engineering Technology at the Oregon Institute of Technology.

COMMITTEE APPROVALS:

Sherry Yang, Wilsonville Program Director (Oregon Institute of Technology)

Reviewer 2 (Oregon Institute of Technology)

Reviewer 3

Reviewer 4

ABSTRACT

RockSat-C is a program for students from all over the U.S. to design, build, and launch a sounding rocket payload out of NASA's Wallops Flight Facility. A team of 10 undergraduates here at Oregon Tech designed and proposed this project in fall 2017. Oregon Tech's proposal was accepted in January 2018 as one of 9 schools across the nation approved for launch. The primary mission is to design a complex, multi-experiment payload emulating the methodology of a satellite or other space vehicle. Mission success is defined as all systems and modular payloads functioning and collecting data.

This project is one of the experiments onboard the Oregon Tech RockSat-C mission. The objective of this project is to implement communication and data storage protocols for other modular experiments onboard. Four modular experiments will be connected to the Data Management System and data will be written to an SD card for each individual line of communication.

The second, off-board part of this project is a user interface to allow further analyzing of the collected data after launch. This program allows the user the ability to open and view different file types via a standalone Windows application.

Mission success of the Data Management System is defined as data being recorded and written to a local SD card during flight time and displayed with the user interface after recovery. Multiple full mission simulations have confirmed this functionality. Unfortunately, launch is scheduled for June 21, 2018, so this document will need to be updated with flight data.

ACKNOWLEDGEMENTS

I would like to acknowledge all the faculty at Oregon Tech for providing an environment that leads to such creative and technical advancement. In particular, I would like to thank Sherry Yang for the guidance as I navigated my way through such an intense project. I'd also like to thank the other students that were a part of RockSat-C; Krystal Cruz, Wilson Davenport, Diego Garrido-Mendoza, Zach Hofmann, Andrew Horn, Caleb Ives, Chris Love, Jean-Luce Nabors, Thomas Pearce, and Jack Thomas. Additionally, I'm incredibly grateful to the faculty that stepped up to make sure RockSat-C could happen. Without the dedication of Andria Fultz and Dawn LoweWincentzen, we wouldn't have had the means to make such a project happen. Finally, I want to thank everyone at the NASA Wallops Flight Facility for offering such an incredible opportunity for students all over the country. Emily Logan, Chris Koehler, and Audrey Viland deserve a special recognition for all the work they've done to connect NASA with students.

LIST OF ACRONYMS

DMS	Data Management System
EDL	Environmental Data Logger
FOG	Fiber Optic Gyroscope
IC	Integrated Circuit
MCU	Microcontroller Unit
MEH	Mesa Energy Harvester
OS	Operating System
PCB	Printed Circuit Board
REH	Rocket Energy Harvester
RSE	Radiation Shielding Experiment
SD	Secure Digital
UI	User Interface
USB	Universal Serial Bus
WFF	Wallops Flight Facility

SUMMARY TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	8
CHAPTER 2 BACKGROUND.....	11
CHAPTER 3 FUNCTIONAL DESCRIPTION	13
CHAPTER 4 DETAILED DESCRIPTION	17
CHAPTER 5 TEST RESULTS & VALIDATION.....	22
CHAPTER 6 ECONOMIC AND IP ANALYSIS	32
CHAPTER 7 SUMMARY	34

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION	8
OVERVIEW	8
PRODUCT DESCRIPTION.....	8
EXISTING PRODUCTS	9
SUMMARY	9
REPORT OUTLINE	10
CHAPTER 2 BACKGROUND.....	11
OVERVIEW	11
BACKGROUND INFORMATION.....	11
STATE OF THE ART	12
SUMMARY	12
CHAPTER 3 FUNCTIONAL DESCRIPTION	13
OVERVIEW	13
FUNCTIONAL DESCRIPTION	13
BLOCK DIAGRAM	13
SUMMARY	16
CHAPTER 4 DETAILED DESCRIPTION	17
OVERVIEW	17
DETAILED DESCRIPTION.....	17
ARCHITECTURE	17
SUMMARY	21
CHAPTER 5 TEST RESULTS & VALIDATION.....	22
OVERVIEW	22
TEST PLAN	22
TEST RESULTS.....	25
DISCUSSION.....	30
SUMMARY	31
CHAPTER 6 ECONOMIC AND IP ANALYSIS	32
OVERVIEW	32
ENGINEERING ECONOMIC ANALYSIS.....	32
INTELLECTUAL PROPERTY ANALYSIS	33
SUMMARY	33
CHAPTER 7 SUMMARY	34
OVERVIEW	34
PROJECT SUMMARY	34
FUTURE DIRECTION.....	34
CONCLUDING REMARKS.....	35

Chapter 1 Introduction

Overview

This chapter defines the problems to be addressed with this project. A project description, comparison to existing projects, and a summary are also included. Finally, an outline of the rest of the report is included.

Product Description

The idea for this project is to make a data management system for an on-board module. This module will be connected to four other modular experiments. The entire system will be launched on a Sounding Rocket from the NASA Wallops Flight Facility in June 2018. The data management system will be responsible for monitoring four experiments. Furthermore, the data from each module will be written on an SD card to be examined later. Extensive error/bounds checking will be a big part of this data management system as well.

The second part of this project is an off-board user interface to allow users to view data after launch/retrieval. This user interface will be completely separate from the main system and will operate independently.

The project is responsible for the following tasks:

1. Module connections
 - a. Recognize and connect to modules via USB
 - b. Recognize and reconnect to disconnected modules
2. Data input streams
 - a. Monitor data from different modules
 - b. Error/bound check all data
 - c. Organize and record data to SD card
 - d. Shut off collection if one should fail
3. UI
 - a. Display data for each experiment
 - b. Display .csv and .txt files
4. Initiation
 - a. Initiate all data collection on startup
 - b. Log errors
 - c. Timestamp log messages
 - d. End data collection after allotted flight time
5. Standby mode

- a. Wait for module connections

Existing Products

A similar system was launched on the 2017 RockSat-C mission. An excerpt from the “RockSat-C 2017 Final Report” is below:

“Although a Raspberry Pi had initially been considered as the primary controller for the payload, the Arduino Mega was eventually chosen. This was due to its relative simplicity, ease of use, and number of available I/O pins. Additionally, previous experience working with the Arduino platform would allow for less of a learning curve. Additionally, the large catalog of off the shelf components specifically designed to work with the Arduino platform made initial prototyping far easier. ...At its core, this payload is essentially a large data logger. Additionally, many of the subsystems have readily available libraries and example code for reference.”

The main difference between last year’s data management and this year’s is the number of modules collecting data. RockSat-C 2018 has a total of seven modules recording data. The microcontroller for this project is the Raspberry Pi Zero W, which is different than the MCU flown last year.

Unfortunately, due to the nature of this project, there aren’t existing products to compare to.

Summary

RockSat-C 2018 has a goal of creating a universal mounting system for student capstone projects that would benefit from hostile environment testing. This modular system will help maximize the availability of functional experiments within the payload volume.

Onboard experiments include:

Environmental Data Logger – Gathers data from the surrounding environment such as pressure, humidity, and temperature

Fiber Optic Gyroscope – Gathers gyroscopic data using a fiber optic cable

Mesa Energy Harvester – Uses piezo elements to produce electrical current

Rocket Energy Harvester – Uses piezo elements in different formations to produce electrical current

Radiation Shielding Experiment - Compares three types of shielding on Geiger–Müller tubes to test effectiveness of radiation deterrent

Optic Window Camera – Gathers images during flight

The DMS will act as a data redundancy mechanism for the four connected experiments on board. The data from these experiments is very valuable not only scientifically, but monetarily as well. Furthermore, Oregon Tech capstone projects are being flown on this mission, and the integrity of the data they return is valuable for the both the student and school itself.

Report Outline

The following report is outlined as such:

- Background
 - Background information on project
 - State of the Art
- Functional Description
 - Functional requirements of the system
 - High level descriptions of how to fulfill requirements
- Detailed Description
 - Expanded look into high level requirements
 - Details on how requirements at this level were achieved
 - Reasoning of choices made
- Testing and Validation
 - Test plan and methodologies used during implementation
 - Test results and descriptions
 - Discussion on testing methodologies
- Economic and IP Analysis
 - Engineering economic analysis on cost of development
 - Intellectual property analysis on copyright effects
- Final Summary and Conclusion

Chapter 2 Background

Overview

This chapter provides background information necessary to understand the project from both a value and technical perspective. State-of-the-art of the project area is discussed along with the current technologies being used in the industry.

Background Information

RockSat-C is a follow-up program to the RockOn Workshop where customers design their own sounding rocket payload and compete for a spot on the launch vehicle. The RockOn Workshop teaches participants how to build a sounding rocket payload in three days. The goal is that participants will take what they learn during the workshop and return the next year with an original payload to fly with RockSat-C or RockSat-X programs.

The RockSat-C program is designed to provide students with access to low-cost flight opportunities and further develop their engineering skills. The RockSat-C flight is made possible through significant cost sharing provided by Wallops Flight Facility (WFF) and the launch fees paid by RockSat-C teams. A portion of the RockSat-C launch fees are invested back into the RockOn Workshop. The RockSat-C and RockOn Workshop canisters fly on the same rocket each year.

The program uses a modular canister system to allow for simple integration to the WFF Sub-SEM ring assembly. This standardized approach simplifies final integration and allows for more focus on the design of the payload. The organizers of the RockSat-C program guide the RockSat-C customers through the design process in the fall with multiple design reviews, leading projects to a Critical Design Review level design in December. Based on available space in the rocket, the most developed and capable projects are selected for flight in January. These projects then make their first payment and begin building. The projects continue to have subsystem and system testing reviews with the RockSat-C program manager through May. Any special requirements that arise for payloads are communicated to WFF through the RockSat program manager. The program culminates in June when the teams travel to WFF in Virginia for inspection, integration to the rocket, launch and recovery. The intent of the RockSat-C program is to provide hands-on experiences to students and faculty advisors to better equip them for supporting the future technical workforce needs of the United States and/or helping those students and faculty advisors become principal investigators on future NASA science missions. Therefore, RockSat-C is limited to U.S. educational institutions; only payloads from U.S. educational institutions are eligible to participate in the RockSat-C program. For the purpose of RockSat-C, 'educational institution' is defined broadly and includes, but is not limited to, the following: universities, colleges, technical schools, public

and private high school, middle school and grade school, science museums, etc. Organizations which are not included in the above listing, are encouraged to contact Colorado Space Grant Consortium (COSGC) to clarify their eligibility in the program. In addition, U.S. entities (e.g. industry, research institutions, etc.) that fall outside of the eligibility conditions listed above, but that are interested in participating in the program, are encouraged to team with an eligible U.S. educational institution. Teaming between educational institutions and industry or other interests is allowed and encouraged.

State of the Art

The RockSat program has been running for 10 years and has launched many different individual experiments. Between RockOn, RockSat-C, and Rocksat-X there have been numerous experiments sent and received by the WFF itself. The technology used to encapsulate experiments has evolved over the years and has been tried and true at this point. Being that the RockSat program has had some time to evolve, the documentation and guidance provided (while minimal) was effective.

Sounding rockets themselves have been in use since as early as 1962 by the Andøya Space Center in Norway. NASA has been using sounding rockets for over 40 years, primarily for low gravity and material based research. This sounding rocket program is primarily used by universities for upper atmospheric research. The lower cost makes sounding rockets an attractive alternative as they do not need expensive boosters or extended telemetry. As a result, mission costs are substantially less than those required for orbiter missions. The sounding rocket program takes advantage of a high degree of commonality and in many cases, only the experiment provided by the designer is changed. In some cases (such as almost all astronomy, planetary, solar, and microgravity missions), the payloads are recovered which means the costs of the experiment and sub-systems are spread out over many missions.

Not only are sounding rocket missions carried out at very low cost, but also the payload can be developed in a very short time frame, sometimes as quickly as 3 months. This rapid response enables scientists to react quickly to new phenomena and to incorporate the latest, most up-to-date technology in their experiments.

The DMS is very particular to the modules it's connected to, so systems like it are only vaguely similar. The individual components of the system have had quite some time to advance and have a lot of documentation and implementation examples. MCUs, USB ICs, and SD cards have been around for a while and are still evolving today.

Summary

This chapter provides some background of the RockSat program and canister system used for launch. The state-of-the-art in regards to the RockSat program, sounding rockets, and materials to build the DMS are discussed as well.

Chapter 3 Functional Description

Overview

This chapter provides a functional description of both parts of the DMS. Included in this description are block diagrams of how each section of the DMS works at a high level. System requirements for both parts of the project are addressed as well.

Functional Description

The main part of the DMS is the on-board electronics. This system is responsible for collecting data from 4 modules, separating the data and recording it to local storage in separate files. A Raspberry Pi Zero W is used for the MCU, while a USB hub is used to port all the connections. A 5v power regulator is used to maintain voltage from the power line while the Pi's SD card stores the data. This system must treat each module independently and have the ability to reconnect if needed. This system is to run for a total of 25 minutes, with the first 3 minutes having no connection with any module. This is due to the flight schedule shown in Figure 1. System requirements for the Rasbian distribution of Linux that runs on the MCU are minimal.

The secondary part of the DMS is the user interface. This small executable file will search its working directory and give the user the ability to display .txt and .csv files. As the program implements fundamental functionality, system requirements are very low. Figure 4 shows the basic navigation flow through windows.

Block Diagram

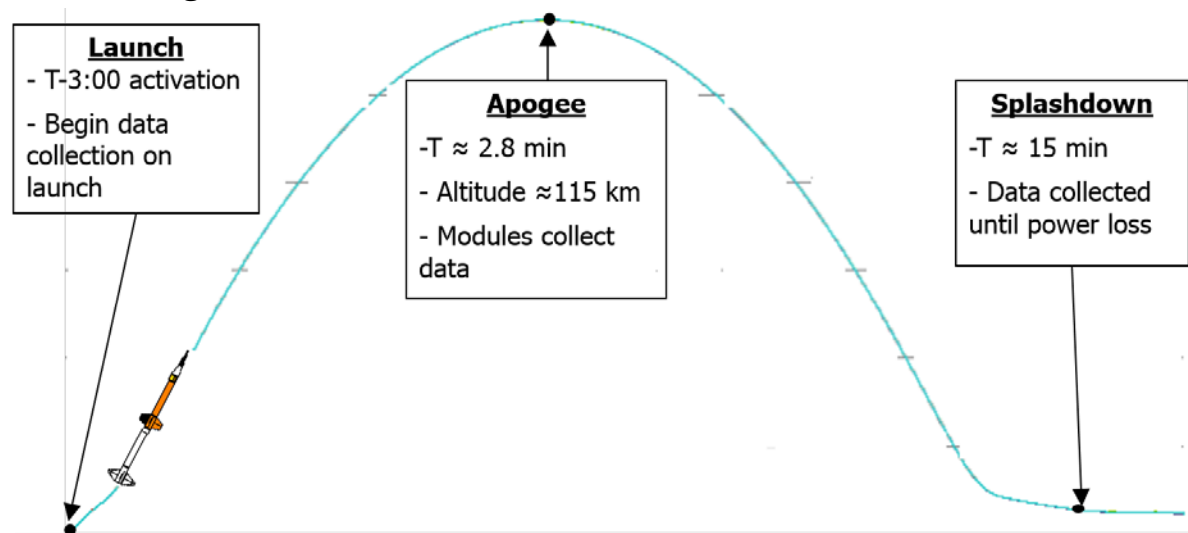


Figure 1: Concept of operations

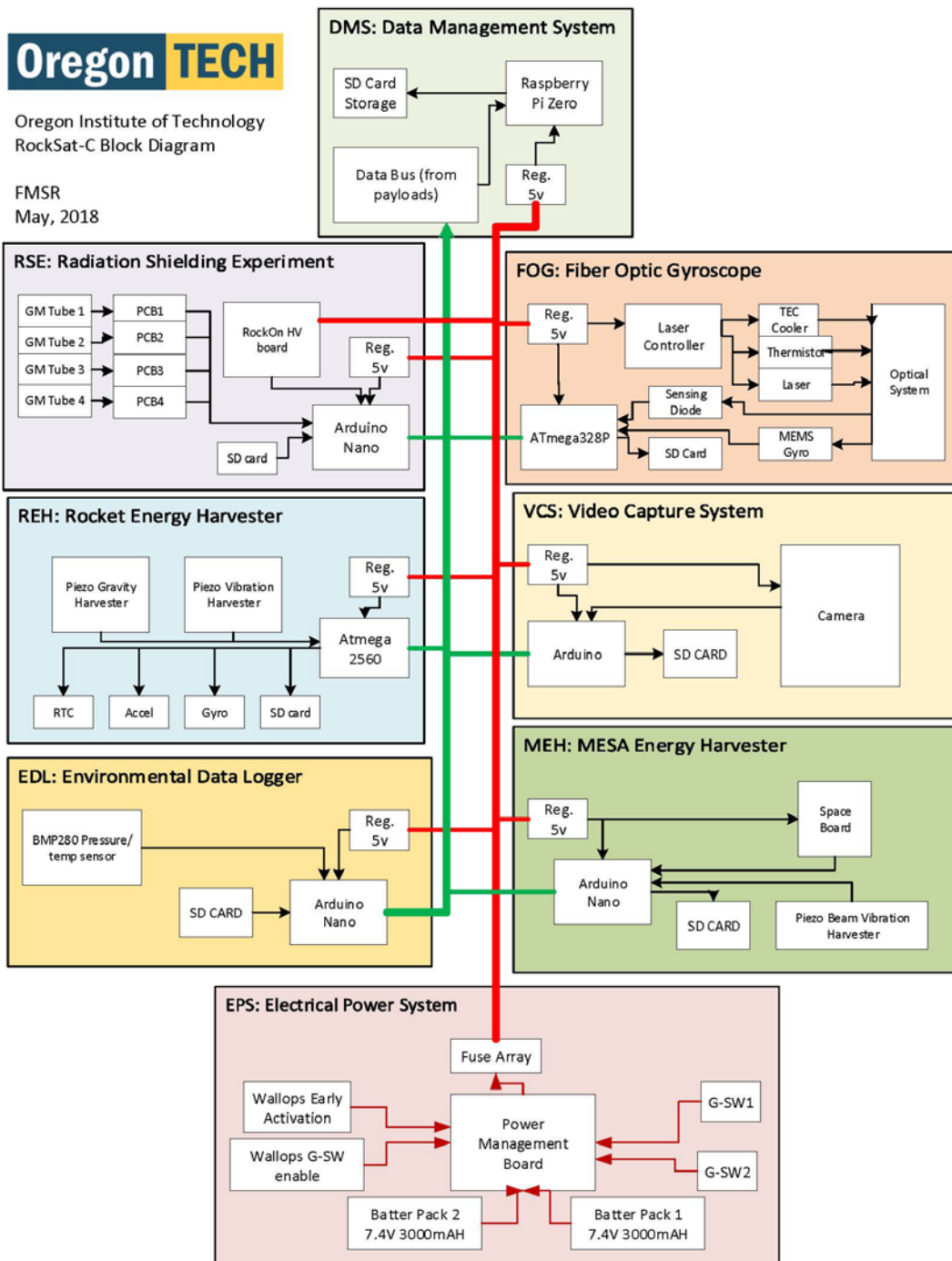


Figure 2: RockSat-C 2018 block diagram

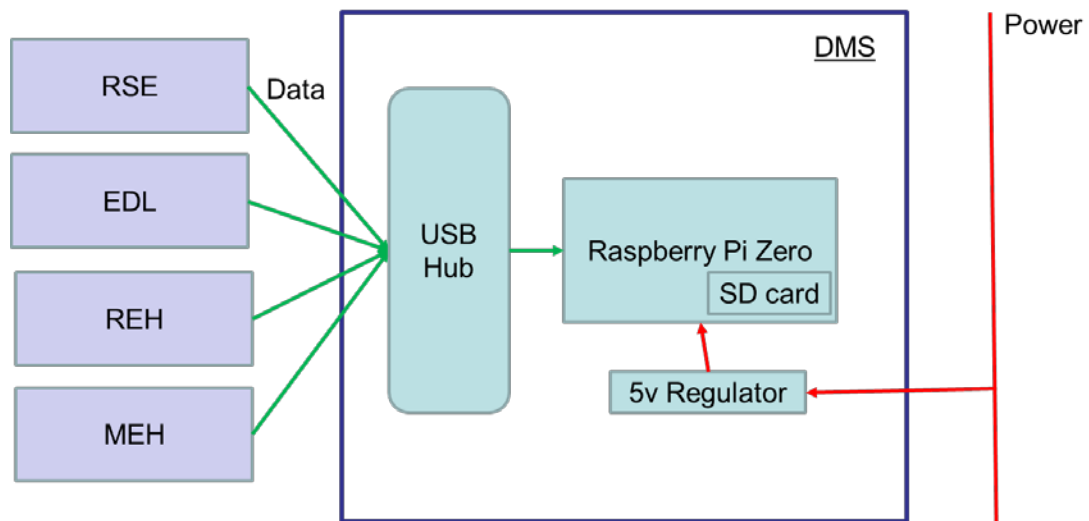


Figure 3: DMS block diagram

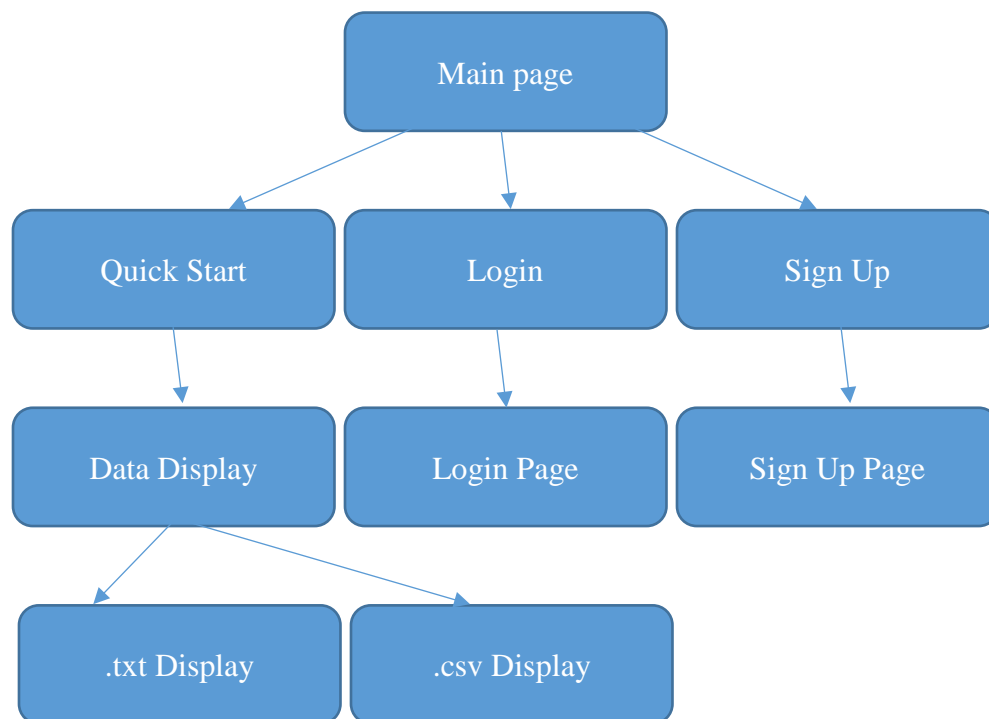


Figure 4: DMS UI block diagram

Summary

This chapter summarized the functionality of the DMS at a high level. Block diagrams show the general flow of information and organization of components for both the on-board and user interface sections. An overall block diagram of the entire RockSat-C mission shows the DMS's place. System requirements, while minimal, are addressed and a basic timeline of launch is described.

Chapter 4 Detailed Description

Overview

This chapter provides explicit information about the requirements for the project and how they were met. Major components and relationships are shown in this section. Data flow through specific sections are shown in figures below. Navigation of the UI is explained in this chapter as well. Technologies, protocols, error handling, initiation, and closure are all explained in this section.

Detailed Description and Architecture

The data flow for the DMS begins at each modules serial out functionality. As each module collects data, it sends it out via the USB port on the MCU. Data then flows over the RX/TX lines in a USB cable until it gets to the USB hub. There the IC on the USB allocates a port to the line of communication and sends that data through another USB connection to the DMS MCU itself. There, the operating system (Raspbian Linux distribution) recognizes the USB connection. A Python program “main.py” will be running on the DMS’s MCU that will recognize this connection through the Pyserial library. A thread is allocated to each serial connection and data is parsed and written to individual .txt files.

The second part of data flow in this project is the user moving the SD card to another machine for inspection via the DMS UI. At this point, the DMS UI will open, allow the user to navigate to a New Data Display page and pick the file to open. Once chosen, the UI will load the data from the .txt or .csv file and display it in a new window.

Module MCU: Each MCU will differ based on its own functionality. The way it sends data out over USB is the part that pertains to the DMS.

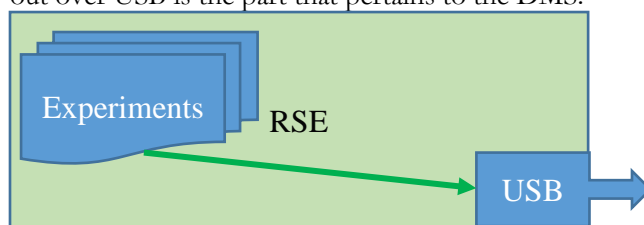


Figure 5: Data flow diagram of module MCU

USB hub: The USB hub uses an IC to route multiple connections to the DMS MCU via one USB port. An FE1.1s USB 2.0 High Speed 4-Port Hub Controller is used for this.

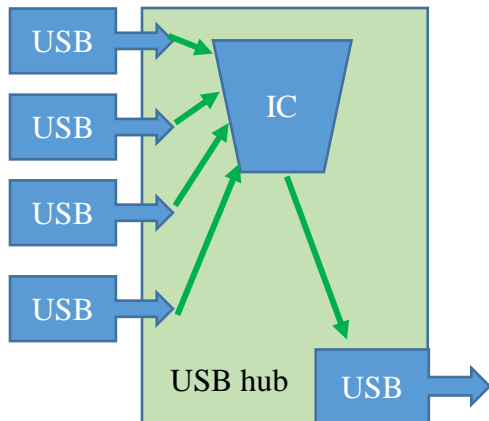


Figure 6: Data flow diagram of USB hub

DMS MCU: The DMS uses the integrated parts of a Raspberry Pi Zero W and the Rasbian distribution of Linux to receive the information coming in via USB. A python program runs within the Linux environment that deals with reading and writing accordingly.

Main.py: This program runs at startup of the operating system. It uses the DataThread function and starts a new function for 4 threads based on the port given by the USB hub and regulated baudrate (9600).

DataThread.py: This function waits for a serial connection to be successfully made, then converts the bytes into ascii characters to be written to a .txt file. This is where errors with connection are caught, and exceptions are dealt with. An additional thread is initiated to check all other thread's health. If a module disconnects, the thread tries to keep connecting until successful. The handshake protocol is encapsulated inside the Pyserial library. A timer is kept in this function for 1500 seconds or 25 minutes. After the timer is over, threads are closed and the program exits.

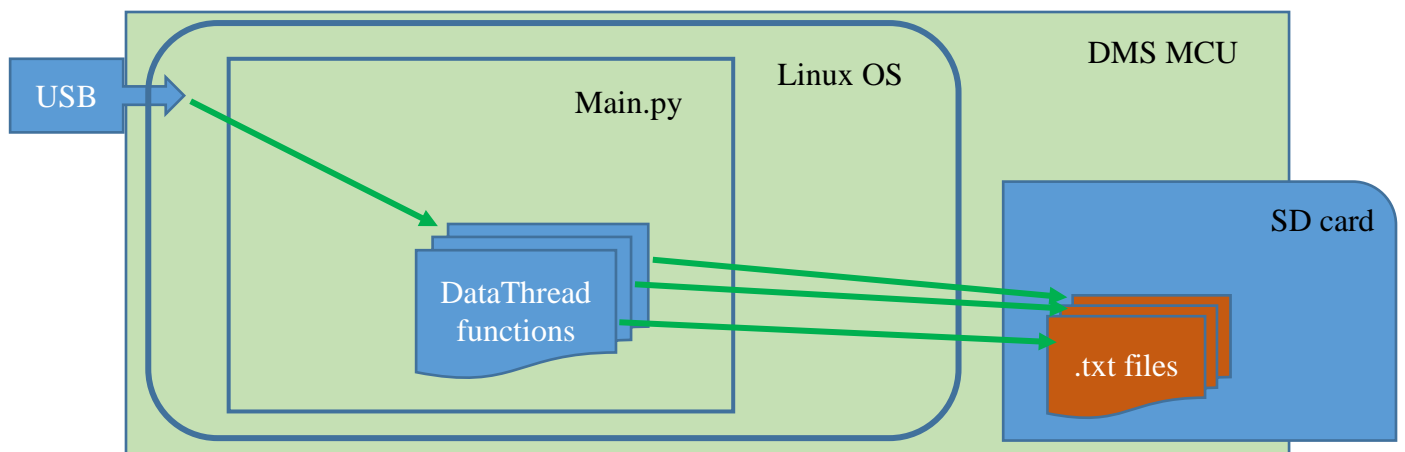


Figure 7: Data flow diagram of the DMS MCU

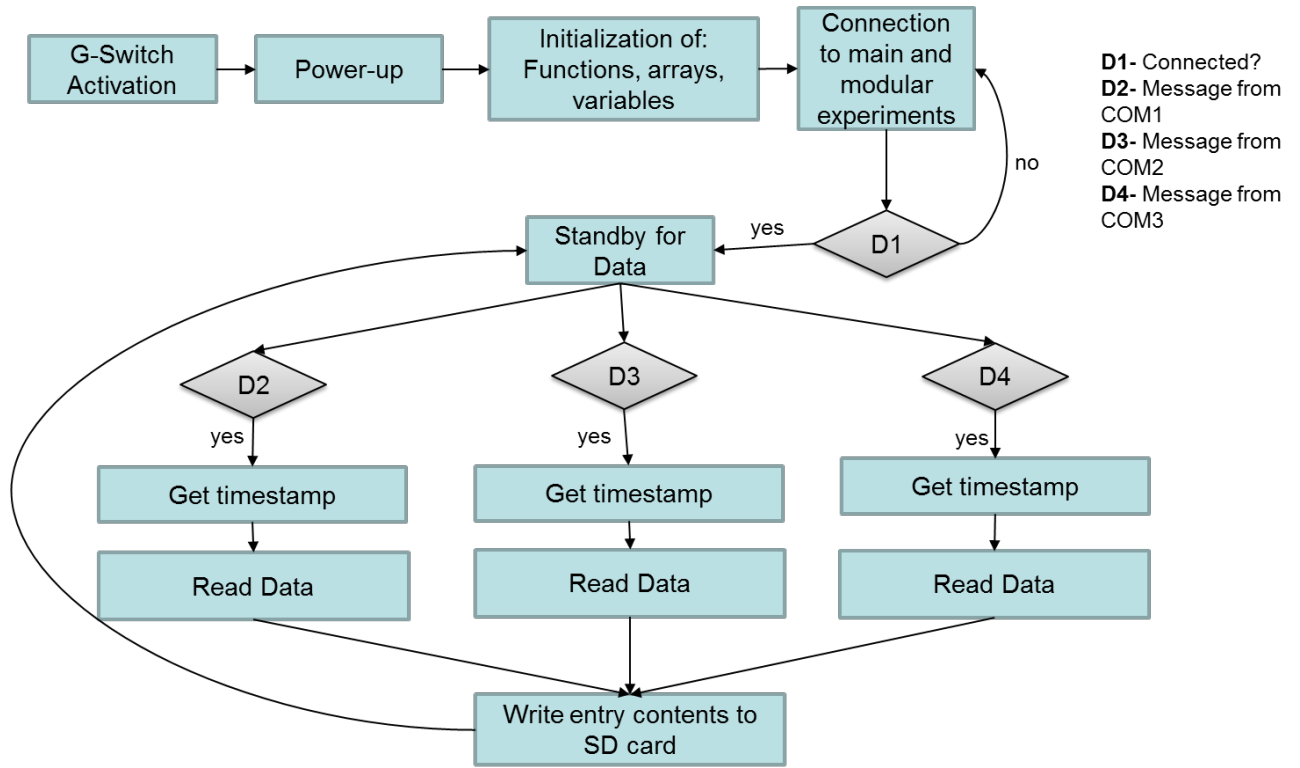


Figure 8: Software flow diagram for 3 modules

DMS UI: The UI on this project was built with C# using a .NET framework. Windows forms are used to display choices and data as shown in the following figure. Basic user prompts and error checking are implemented at this level.

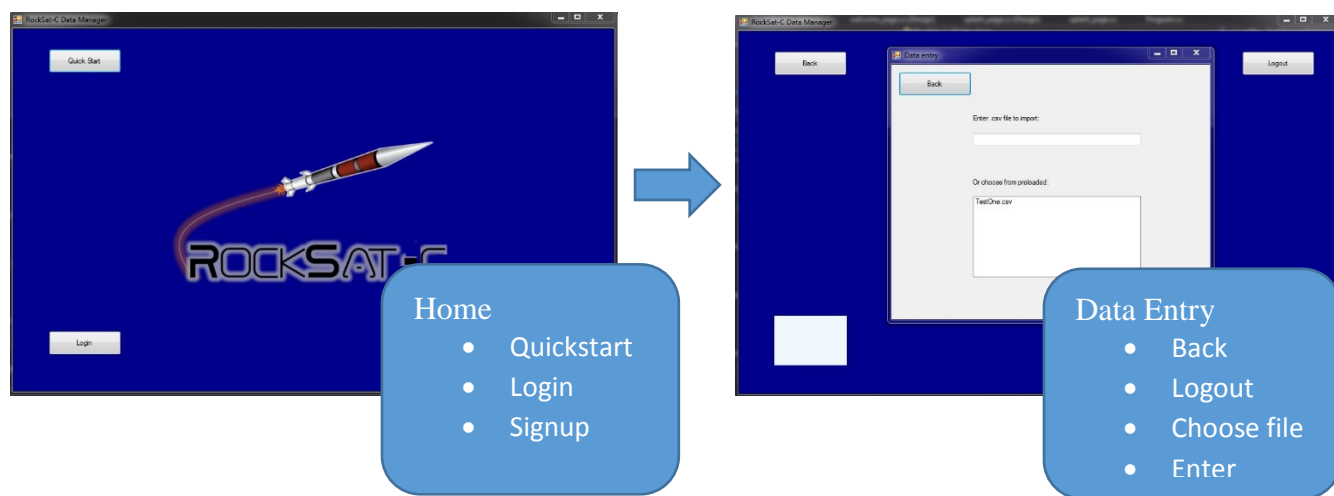


Figure 9: DMS UI basic navigation

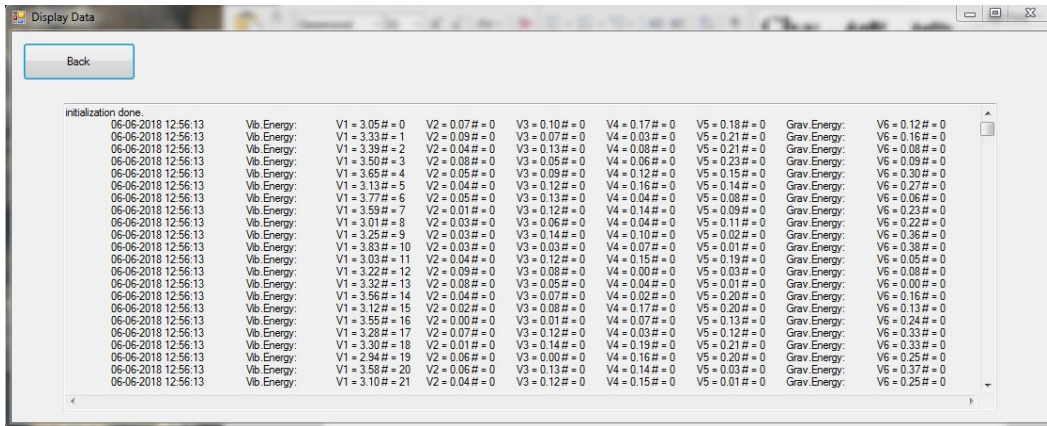


Figure 10: DMS UI .txt display

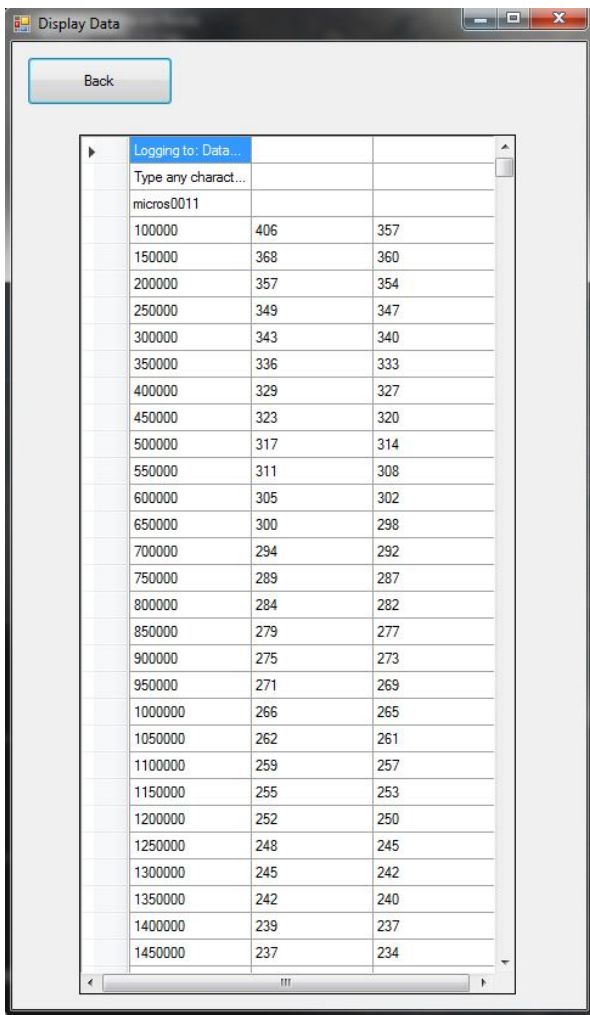


Figure 11: DMS UI .csv display

Overall, the element that the two sections share is the SD card for data. Due to the nature of this project, the UI wasn't able to be integrated directly into the main.py program. Once the canister is recovered after launch, the SD card will need to be physically removed and placed into a different computer. An overall architecture of the entire system is included below.

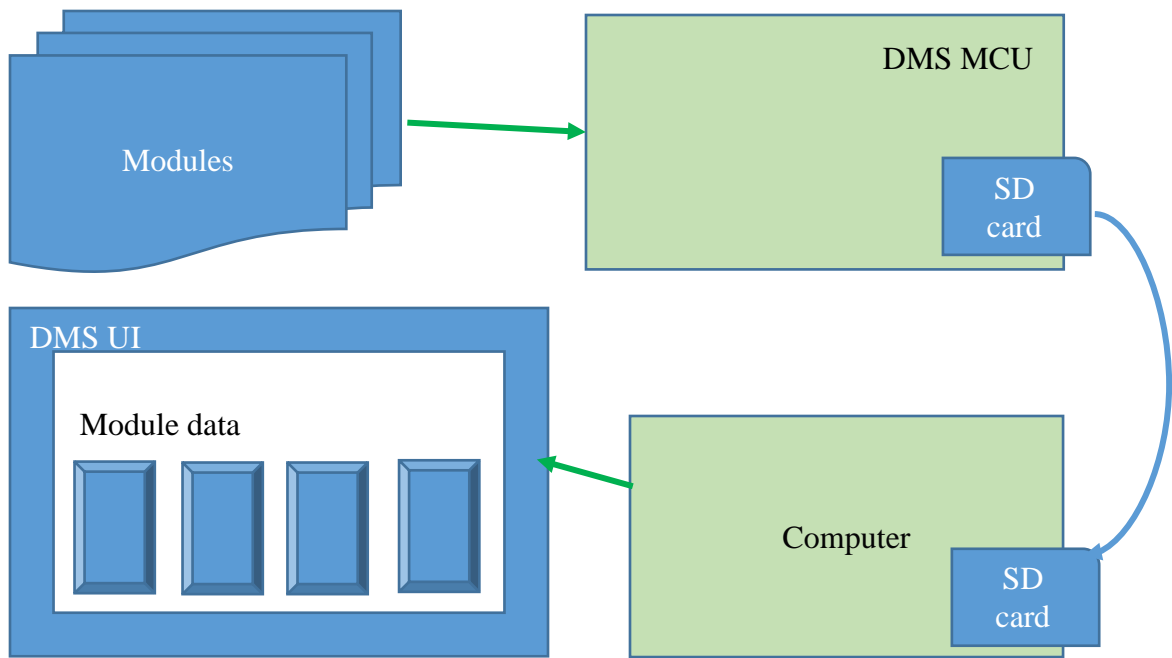


Figure 12: Overall Architecture of both DMS elements

Summary

This chapter provides detailed information on how each element of both DMS systems works to implement the required functionality. Major components and their relationships are shown, as well as minor components. Data flow is shown in detail and manipulation techniques are explained. Example of the UI and how a user might navigate around it is shown and technologies used are mentioned. Additional technologies and methods on closure, initiation, error handling, and communication protocols are explained as well.

Chapter 5 Test Results & Validation

Overview

This chapter describes the test plan and validation techniques used in this project. Launch for the RockSat-C project is planned for June 21, 2018. All the test results in this section were created using mock code and prototype MCUs.

Test Plan

The test plan for this project and tentative deadlines were as follows:

UI design, prototype, implementation, and testing
Sept - May
Data Management System/ Redundancy design
Nov – Dec 15
Data Management System/ Redundancy prototyping
Dec 15 – Jan 15
Data Management System integration
Jan 15 – April 1
Refinement and margin for debugging
April 1– May 1

Video conferences with WFF were frequent and subsystem testing was a point of conversation every meeting. Examples of test results are included in the Test Results section of this chapter.

Validation of functionality was tested using MCUs with actual flight code running. Flight code for each module was edited to produce mock output, as actual sensors weren't attached. Test driven development was used throughout development, to make sure each block of code was functioning as needed. The DMS UI was user tested by various people for basic navigation and functionality.

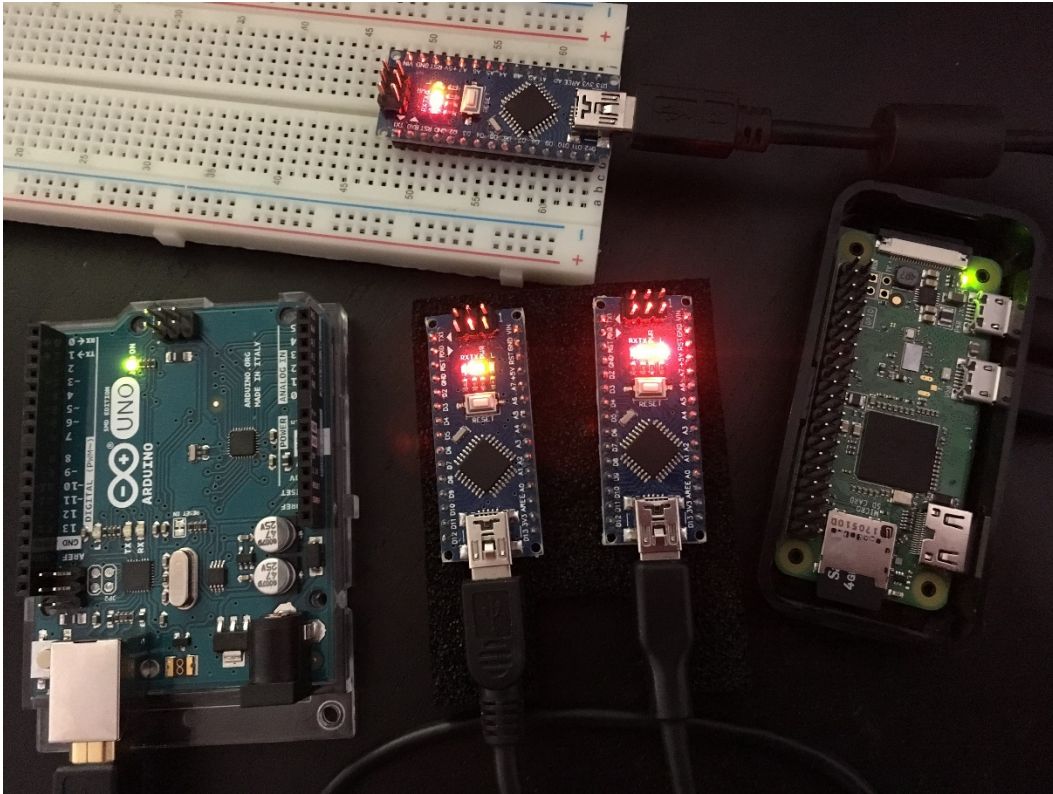


Figure 13: MCUs used for testing


```
94 //myFile = SD.open("test.txt", FILE_WRITE);
95
96 // if the file opened okay, write to it:
97 if (true) {
98
99     // When the file is open, write the values of PZ Cap Voltage and Count
100    // and the Spaceboard Sensor Values
101    Serial.println();
102    Serial.print(millis()/1000);
103    Serial.print(", ");
104    Serial.print(voltage);
105    Serial.print(", ");
106    Serial.print(countA0);
107    Serial.print(", ");
108
109    //ambient.read();
110    Serial.print(random(1,10));
111    Serial.print(", ");
112    //infrared.read();
113    Serial.print(random(16,100));
114    Serial.print(", ");
115
116    //lum.read();
117    Serial.print(random(1,100));
118    Serial.print(", ");
119    //uv.read();
120    Serial.print(random(1,100));
121    Serial.print(", ");
122
123    //accel.read();
124    Serial.print(random(1,100));
125    Serial.print(", ");
126    Serial.print(random(1,400));
127    Serial.print(", ");
128    Serial.print(random(1,600));
129    Serial.print(", ");
130
131    //mag.read();
132    Serial.print(random(1,10));
133    Serial.print(", ");
134    Serial.print(random(1,70));
135    Serial.print(", ");
136    Serial.print(random(1,100));
```

Figure 14: Mock code set up for MEH module

Test Results

Test results that were presented to WFF during teleconferences are provided below.



DMS: Status

Progress:  20%

- Status
 - Tested
 - Log files, Pi/Arduino environments, connectivity
 - Not tested
 - Multiple experiments, real time data ingestion, failure scenarios, watchdog system

```
log.write("*****Initating log*****\n")
log.write(str(datetime.now())+ "\n")
log.write("Looping to create data in two dimensional array\n")
myData = [{"Time", "Count", "Col 3", "Col 4"}]

for x in range(1,10):
    myData.append([str(datetime.now()),x,"filler","example"])
    log.write("Data Written!\n")
    time.sleep(1)
```



RockSat-C 2018

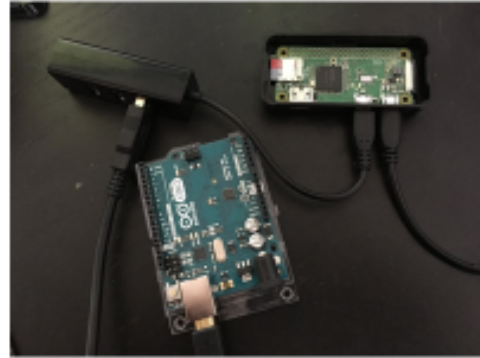
STR



Figure 15: Subsystem Testing Review, presented 2/14/2018

DMS: Completed Tests

- Completed Tests
 - Environment setup
 - Pi script on startup
 - Log file written with no I/O
 - .csv creation from data structure
 - Physical connection Arduino to Pi
 - Initial serial communication



RockSat-C 2018

STR



Figure 16: Subsystem Testing Review, presented 2/14/2018

DMS: Results

- Results
 - Environments, log files, .csv, data structure pass
 - Connectivity fail (serial connection)

```
log.txt
1 *****Initiating log*****
2 2018-02-07 01:13:59.541992
3 Looping to create data in two dimensional array
4 Data Written!
5 Data Written!
6 Data Written!
7 Data Written!
8 Data Written!
9 Data Written!
10 Data Written!
11 Data Written!
12 Data Written!
13 Fake data initialised
14 This is where to start a .csv file
15 Data written to .csv!
16
```

	A	B	C	D
1	Time	Count	Col 3	Col 4
2	13:59.5		1 filler	example
3	14:00.6		2 filler	example
4	14:01.6		3 filler	example
5	14:02.6		4 filler	example
6	14:03.6		5 filler	example
7	14:04.6		6 filler	example
8	14:05.6		7 filler	example
9	14:06.6		8 filler	example
10	14:07.6		9 filler	example



RockSat-C 2018

lochnessproject.org



STR

Figure 17: Subsystem Testing Review, presented 2/14/2018

DMS Testing Status

Successful Tests:

- Data transfer over UART
- Threading supported on Pi OS
- Timing for data collection on bootup
-

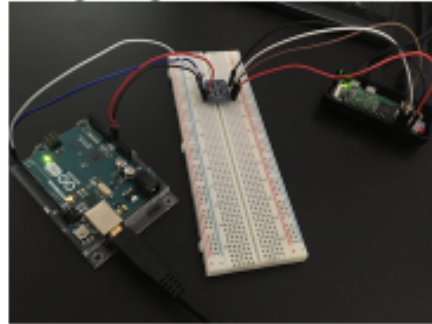
Progress:  40%

Failed tests:

- Direct connection (need logic level shifter)

Pending tests:

- Threaded UART communication with multiple experiments
- Unit testing
- Failed experiment scenarios



RockSat-C 2018

ISTR




Figure 18: Integrated Subsystem Testing Review, presented 3/27/2018

DMS Testing Status

Progress:

98%

Successful Tests:

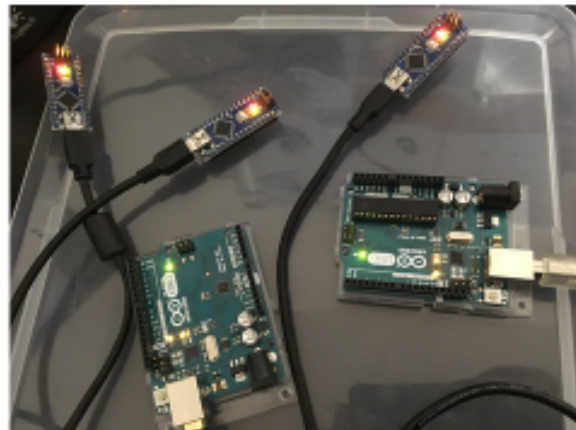
- Data Transfer with 5 modules for full flight time (mock code)
 - Reconnection of modules
 - Data output integrity
 - Delay times on modules
 - Timer cutoff
- 

Failed tests:

- Sending power to modules via RPi

Pending tests:

- DMS power loss
- Unit testing
- Modules with flight code



RockSat-C 2018

FMSR



Figure 19: Full Mission Simulation Review, presented 5/4/2018

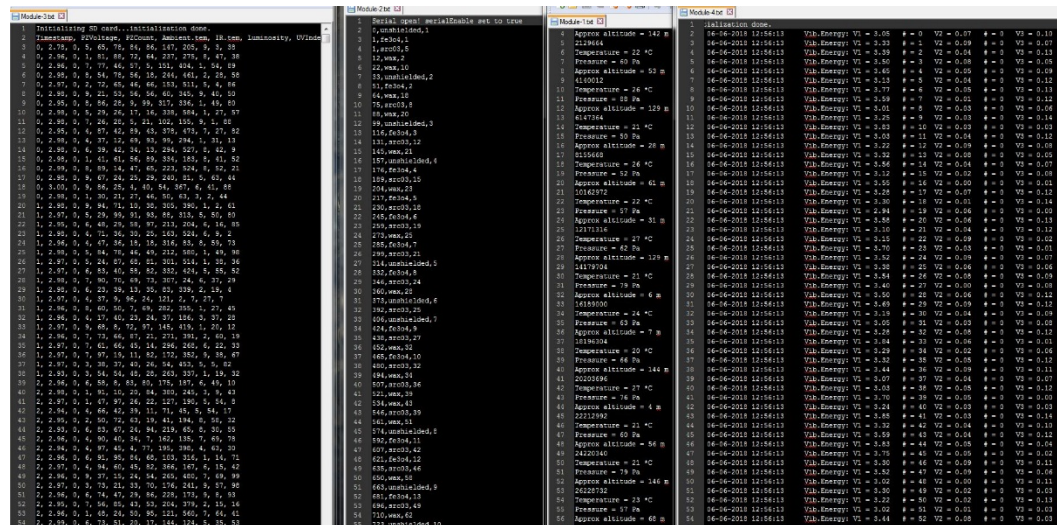


Figure 20: Final output using mock flight code

Usability of the UI was also tested during the Oregon Tech Project Symposium 6/6/2018. Multiple full mission simulations with 4 mock modules running were also run at this time.

Discussion

Test results are described below per each function requirement.

1. Module connections - Passed
 - a. Recognize and connect to modules via USB
 - b. Recognize and reconnect to disconnected modules

```
605 30150000,211,209
606 30200000,211,209
607 30250000,211,208
608 30300000,212,209
609 30350000,212,209
610 30400000,212,209
611 30450000,212,208
612 30500000,212,209
613 30550000,212,209
614 30600000,211,209
615 30650000,211,208
616 30700000,211,208
617 30750000,212,208
618 30800000,212,208
619 30850000,211,208
620 30900000,211,208
621 30950000,212,208
622 Logging to: Data00.csv
623 Type any character to stop
624 micros0011
625 100000,354,350
626 150000,346,344
627 200000,339,338
628 250000,333,331
629 300000,327,325
630 350000,321,319
631 400000,315,314
632 450000,310,308
633 500000,305,303
634 550000,299,298
635 600000,294,293
636 650000,290,288
637 700000,285,283
638 750000,281,278
639 800000,276,275
640 850000,272,270
641 900000,268,267
642 950000,265,263
643 1000000,261,259
```

Figure 21: .csv file showing reconnection of module at line 622

2. Data input streams – Passed, shown in Figure 20
 - a. Monitor data from different modules

- b. Error/bound check all data
 - c. Organize and record data to SD card
 - d. Shut off collection if one should fail
- 3. UI – Passed, shown in Figures 10 and 11
 - a. Display data for each experiment
 - b. Display .csv and .txt files
- 4. Initiation – Passed, shown in successful full mission simulations
 - a. Initiate all data collection on startup
 - b. Log errors
 - c. Timestamp log messages
 - d. End data collection after allotted flight time
- 5. Standby mode – Passed, shown in full mission simulations
 - a. Wait for module connections

Overall, all the specifications for the experiment were met. A few off-ramps had to be made, but the main functionality of the project is intact and working correctly.

Summary

This chapter explains the test plan used for this project and the results found. Examples of testing reports delivered to WFF are included as well as an overall discussion on the test results versus the specifications of the project. Being that launch is scheduled after this paper is finished, no actual launch data is included.

Overview

Engineering Economic Analysis

[illegible]

The DMS itself had relatively low prototyping costs (around \$200), as the parts needed are quite common and no custom PCBs needed to be printed (aside from the power regulator). The hours spent developing this project are innumerable and thus difficult to estimate financially.

32

process decreases engineering time will become less costly. Abstracting away the integration process from the end user saves them a lot of R&D time as well.

Intellectual Property Analysis

Most of the tools used in this project are open source, so the only intellectual property that would be owned by anyone would be the logic in both `main.py` and `datathread.py`. Due to the nature of this project, the only part that would be used by anyone else would be the DMS UI. There may be concern about the data privacy, but no database connectivity is used. All the RockSat-C program information is public IP, being that it's published by a federal agency. No concerns with Copyright, Trademark, Patent, or Trade Secret issues arise from the development and implementation of this project. There are no intellectual property protections or licensing agreements in place for this project.

Summary

This chapter describes the overall cost of the project and analyses the intellectual property of the developed systems. Considerations regarding issues such as; Copyrights, Trademarks, Patents, and Trade Secrets are discussed in this section as well.

Chapter 7 Summary

Overview

This chapter provides an overall summary of the project. Future directions after mission completion are mentioned in this section.

Project Summary

RockSat-C is a follow-up program to the RockOn Workshop where customers design their own sounding rocket payload and compete for a spot on the launch vehicle. The RockOn Workshop teaches participants how to build a sounding rocket payload in three days. The goal is that participants will take what they learn during the workshop and return the next year with an original payload to fly with RockSat-C or RockSat-X programs.

The idea for this project is to make data management system for an on-board microcontroller. This microcontroller will be connected to four modular experiments. The entire system will be launched on a Sounding Rocket from the NASA Wallops Flight Facility in June 2018. This system is responsible for collecting data from 4 modules, separating the data and recording it to local storage in separate files. A Raspberry Pi Zero W is used for the MCU, while a USB hub is used to port all the connections. A 5v power regulator is used to maintain voltage from the power line while the Pi's SD card stores the data. This system must treat each module independently and have the ability to reconnect if needed. This system is to run for a total of 25 minutes, with the first 3 minutes having no connection with any module.

The second part of this project is an off-board user interface to allow users to view data after launch/retrieval. This user interface will be completely separate from the main system and will operate independently. This small executable file will search its working directory and give the user the ability to display .txt and .csv files.

Future Direction

In regards to RockSat-C as a whole, the natural progression is to aim for the RockSat-X program. This program chooses even less participants and uses a different rocket with a removable fairing. This introduces many more opportunities for research, as well as engineering challenges. As the 2018 RockSat-C team has learned to work together so well and develop such a technical payload, RockSat-X seems like the right direction. The development of such a modular system could definitely be improved upon and used for another RockSat-C mission as well.

As for the DMS itself, this module could definitely be re-purposed and used again for a different payload. The logic was written in such a way that adding new modules is as easy as changing the baudrate in main.py. Using USB abstracts away a lot of the communication

protocol challenges. Being that there are no licensing or copyright implementations in place, anyone is welcome to use the source code.

As for me, this whole process has really put me in an embedded systems direction. System integration is also a new field I'd like to learn more about. Engineering management is also a new role I'm very interested in. Leading a project like this would definitely be something I would like to do in the future. I hope to take all the skills learned in this project and apply them to bigger and more complex missions.

Concluding Remarks

In conclusion, I definitely got in over my head in regards to the electrical/embedded aspects of this project. I feel like this project was much more an exercise in engineering in general than any one discipline. Being that the entire RockSat-C project was student run, we all had to take on roles we didn't initially sign up for. This is very similar to a real world example of an engineering team. Learning to communicate with each other in a way to effectively reach solutions is something that can't be taught. Practice is the only way to get used to it. From a technical standpoint, I learned a lot. Countless hours of independent research helped me design and implement a system I had no knowledge about previously. The Data Management System in particular was a great exercise in all the skills I've learned from the Software Engineering program here at Oregon Tech. RockSat-C is an amazing opportunity for students to get a multi-disciplined, real world experience in engineering. I'm glad to have been a part of it and apply my senior project to such an amazing opportunity.

References

- Jenner, Lynn. “Sounding Rockets”. Internet:
https://www.nasa.gov/mission_pages/sounding-rockets/index.html, Aug.
3, 2017 [June 6, 2018].
- RockSat-C Payload Canister User’s Guide*, 1st ed., Colorado Space Grant
Consortium., Boulder, CO, USA, 2018.

Appendix

.