University of Colorado
Department of Aerospace Engineering Sciences
Senior Projects - ASEN 4028

Thinsat Operational Management of Commands And Telemetry (TOMCAT)

Project Final Report

Monday 4th May, 2020

Information

0.1. Project Customers

Name: Mark Werremeyer Email: Mark.R.Werremeyer@Raytheon.com	Name: Sheldon Clark Email: Sheldon.C.Clark@Raytheon.com

0.2. Group Members

Name: Austin Scheck Email: Austin.Scheck@colorado.edu	Name: Cole Kenny Email: Nicholas.Kenny@colorado.edu
Name: Grant Novota Email: Grant.Novota@colorado.edu	Name: Jake Schroeder Email: Jacob.Schroeder@colorado.edu
Name: Katie Steward Email: Katie.Steward@colorado.edu	Name: Lara Buri Email: labu2151@colorado.edu
Name: Lauren De Moudt Email: Lauren.DeMoudt@colorado.edu	Name: Lucas Perry Email: Lupe6259@colorado.edu
Name: Manuel Lindo Email: Manuel.Lindo@colorado.edu	Name: Matthew McCallum Email: Matthew.McCallum@colorado.edu
Name: Srikanth Venkataraman Email: srve1799@colorado.edu	Name: Trevor Weschler Email: Trevor.Weschler@colorado.edu

Contents

List of Figures

List of Tables

Nomenclature

BPSK	Binary Phase Shift Keying
DDR	Double Data Rate
DL	Downlink
EEPROM	Electrically Erasable Programmable Read-Only Memory
eMMC	Embedded Multi-Media Card
I/O	Input/Output
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
ISM	Industrial, Scientific, and Medical
JTAG	Joint Test Action Group
LED	Light Emitting Diode
MCU	Micro-Controller Unit
MEMS	Micro-Electro-Mechanical System
OS	Operating System
OSD 3358 C-Sip	Octavo Systems Complete System-in-Package Surface Mount Device
PC	Personal Computer
PCB	Printed Circuit Board
SMD	Surface Mount Device
TOMCAT	Thinsat Operational Management of Commands And Telemetry
UL	Uplink
USB	Universal Serial Bus

1. Project Purpose

Cole

Expenditures to access space are slowly decreasing; however, development, testing, launch, and operating costs of spacecraft and small satellites still remain a huge barrier to entry for anyone looking to enter into the spaceflight industry. While these expenses make it harder for businesses to conduct activities in space, it also makes it difficult for primary schools and STEM organizations to expose their students to space-based research. To combat this, Virginia Space helped pioneer the ThinSat Program, which provides an opportunity for educational institutions to design lightweight, low-cost payloads for picosatellites.[5][6]

The ThinSat is a 1/7th slice of a 1U cubesat, with a fully functioning bus and payload area. The TOMCAT project expands on the ThinSat program, expanding its communication capabilities. By using a software defined radio (SDR), TOMCAT allows the satellite to communicate at any frequency that a company could want, while maintaining the small size of the ThinSat frame. The TOMCAT project is a 2/7th slice of a 1U cubesat, to provide a better, more versatile communication system and more battery power. By conforming to the ThinSat shape, the TOMCAT satellite can be launched into low Earth orbit for a very low price relative to other space launches.

The TOMCAT project can be expanded into infinite applications. Raytheon is specifically interested in two after the project is complete: an on-orbit ground station test asset and a spacecraft to characterized the upper atmosphere as the satellite burns up on re-entry.

2. Project Objectives and Functional Requirements

Katie, Cole

2.1. Project Objectives and Levels of Success

Raytheon has tasked the TOMCAT team to develop and test viable options for PicoSat telemetry and lead the way in developing and testing secondary communications on the ThinSat. Not only would this provide a redundancy to the satellite, an often-desired trait in space missions, but it would also allow for a low-cost option for testing ground-based communication. Following the standards laid out by the Consultative Committee for Space Data Systems (CCSDS)[7], the team will integrate command and telemetry, C&DH hardware and software first onto a FlatSat, then onto a ThinSat bus for verification and validation in the lab and on a high-altitude balloon. Communications will first be established during a bench test of two Software Defined Radios. These radios will then be modified to work at long range. The high-altitude balloon flight will act as a final validation test, in which the payload will be fully integrated with the ThinSat bus and the COMM system will be fully operational. Tests leading up to the balloon flight will also allow for the verification and validation of the wireless COMM system. The ThinSat payload will include a scientific sensor, which will take data and generate telemetry that will be sent through the COMM system to the COMM system on the ground. This will fulfill Raytheon's desire to have some kind of science data measured on the payload that can be sent as telemetry to further verify the capability of the COMM system.

The following table reflects the specific objectives for each of these three cornerstone components. Within each of these, objectives are subdivided into three levels of mission success, with Level I representing marginal, bottom-line performance, and Level III expressing high-tier achievement, implying the fulfillment of both Level I and II objectives in their completion.

Flight Software		
	Command	**Telemetry**

Level 1	Flight software should receive, parse, and execute commands from the SDR. Flight software should recognize and decode CCSDS-formatted command packets.	Flight software should generate state of payload health telemetry. Flight software should packetize telemetry it generates or receives from the science sensors in CCSDS format. Flight software should send the packetized telemetry to the SDR for downlink.
Level 2	Flight software should generate command counters to verify commands it received. Flight software shall receive and execute payload commands from the on-board bus computer. Flight software should generate command counters for commands it has sent to the science suite.	Flight software should store state of health and science telemetry that is generated so that it can be compared to the telemetry received on the ground.
Level 3	Flight software should generate command counters to verify commands have been recognized as acceptable commands for payload execution.	
Communications		
Level 1	The SDR should be able to receive commands from another SDR over a serial connection. The SDR should be able to transmit telemetry to another SDR over a serial connection.	
Level 2	The SDR should be able to receive commands from another SDR over a wireless connection at a range greater than 5m. The SDR should be able to transmit telemetry to another SDR over a wireless connection at a range greater than 5m.	
Level 3	The ground SDR should be able to receive commands from the flight SDR over a wireless connection during the balloon flight, with a range of 30km. The ground SDR should be able to transmit telemetry to the flight SDR over a wireless connection during the balloon flight, with a range of 30km.	
Science Sensor Suite		
Level 1	Sensor suite should generate science telemetry to send to the payload computer.	
Level 2	The sensor suite should execute commands received from the payload computer to take science data. The sensor suite should return science data to the payload computer. The sensor suite should generate and send state of health telemetry to the payload computer.	
Level 3		
Payload Structure		
Level 1	The payload should mount to the designated payload area on the ThinSat bus.	

Level 2	
Level 3	The payload should survive nominal (9 m/s) impacts from balloon launch and recovery. The SDR should be mechanically mounted to the bus for balloon flight.

2.2. System Operation and Deliverables

Raytheon would like to use TOMCAT as a relatively inexpensive on-orbit test asset that can verify the functionality of a ground station. TOMCAT will be launched in the low-earth orbit (LEO) where it can communicate with Raytheon's chosen ground station. TOMCAT will be focusing on beginning the development of long-range communications by using a software defined radio configured for the range of a balloon launch. Raytheon's mission is shown below on the right, and TOMCAT's mission in comparison is shown below on the left.

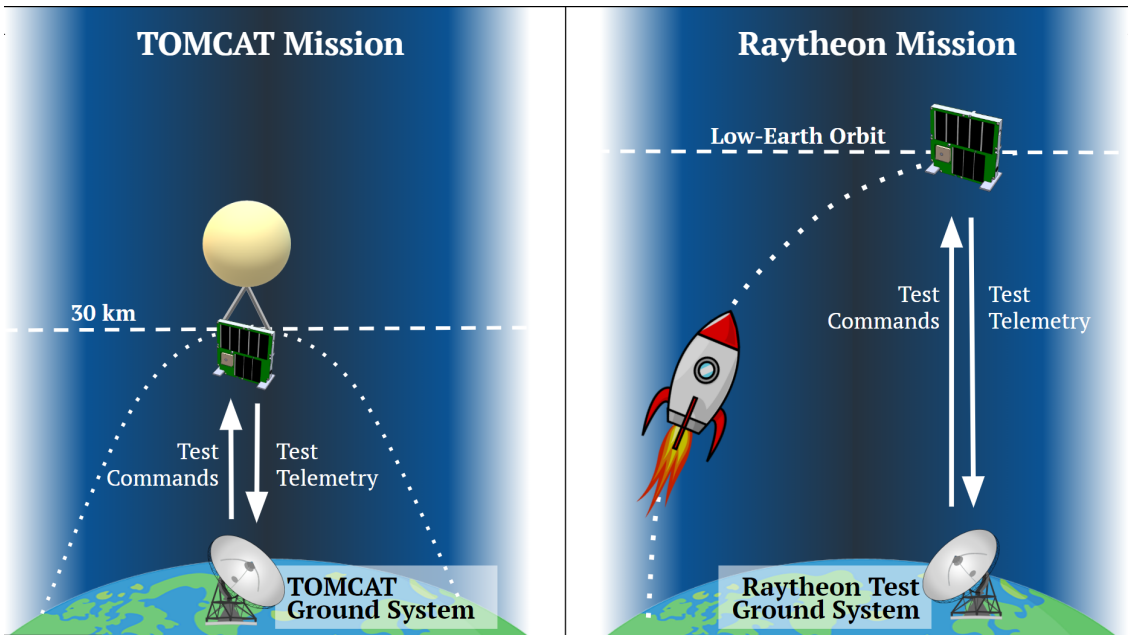


Figure 1. Raytheon's anticipated first mission with the ThinSat compared with TOMCAT's portion of the project development

A more detailed visual of the TOMCAT's end goal balloon flight is shown in the CONOPS below. TOMCAT will be launching on along on a balloon string hosted by the University of Colorado at Boulder's Gateway to Space class. The ground station will start by communicating with the ThinSat while it is on the ground so a two-way communication link can be verified. On-board the ThinSat, the ground commands will travel from software defined radio to the microcontroller to the inertial measurement unit, which will execute the command and send rate telemetry in return. Throughout the duration of the launch the ground system will be in constant communication with the ThinSat. At the peak altitude of the flight (around 30km), the balloon will automatically burst due to pressure, and a parachute will automatically deploy. During the descent, the ground station will continue to maintain two-way communication. The ThinSat is expected to land at a speed of around 9 m/s, where after it can be collected by the TOMCAT team.

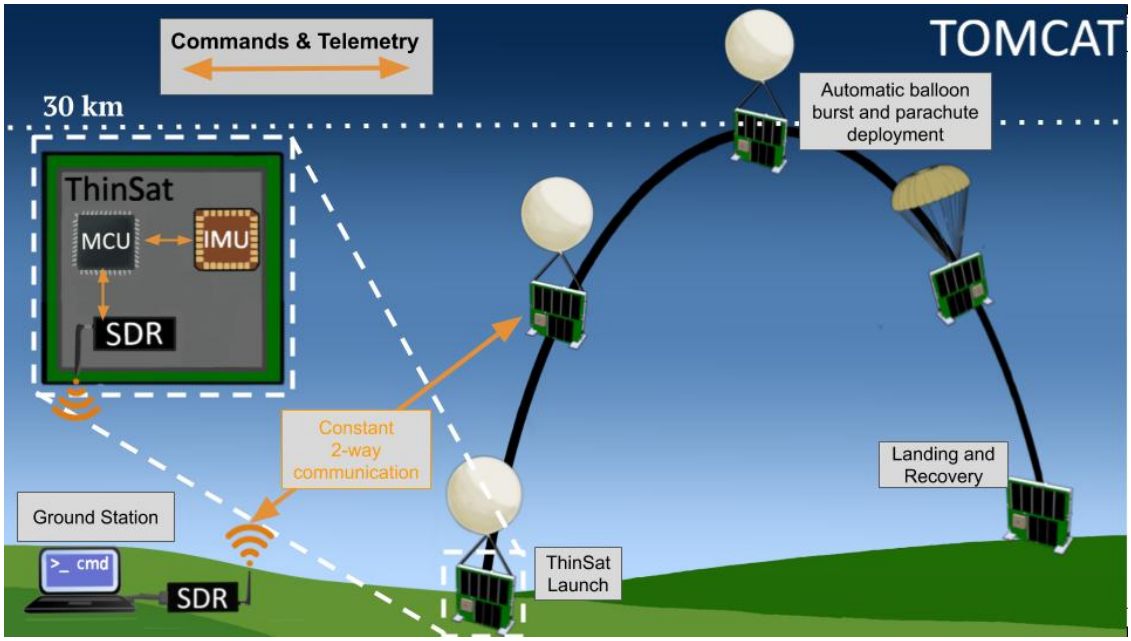


Figure 2. TOMCAT mission concept of operations

At the end of the project, the TOMCAT team will give Raytheon two systems – the TOMCAT ThinSat system and the TOMCAT ground system. The ThinSat system will consist of one purchased frame "slice", and one custom manufactured frame "slice", and all required internal hardware: communications hardware (radio, amplifiers), custom printed payload circuit board (containing an inertial measurement unit and an AM335x microcontroller), insulation, and battery. The software provided for this system will consist of custom flight software based on the framework of KubOS flight software and custom signal processing software based on the framework of GNURadio software. The ground system hardware will consist of a Yagi antenna, software defined radio, amplifiers, and a USB connection to a computer. The ground system software will consist of the same custom signal processing software as the ThinSat system package, as well as custom ground software built off the framework of COSMOS.

2.3. Functional Block Diagram

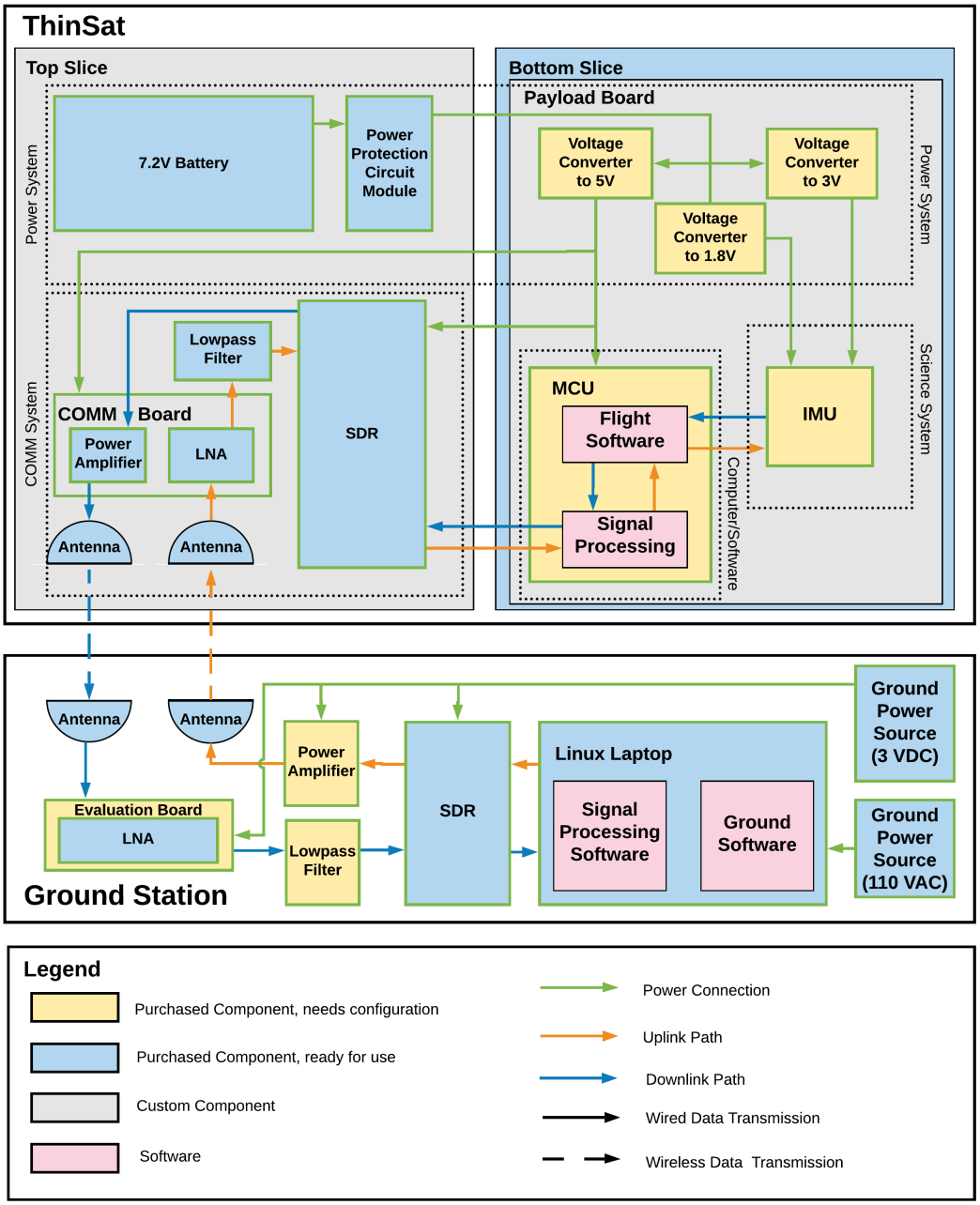


Figure 3. TOMCAT On-board and Ground System Functional Block Diagram

2.4. Project Functional Requirements

In early stages of the project, the TOMCAT team discussed the goals of the project with the Raytheon sponsor. In an effort to design the communication satellite that Raytheon is looking for, the team introduced the following high level functional requirements for the project to succeed.

1. The on-board payload shall interface with the ThinSat bus according to the ThinSat Interface Control Document (ICD)

2. The on-board payload shall be able to communicate with the ground system on a balloon-flight-like range test

3. The payload shall survive and operate in a balloon-flight-like environment for the duration of the proposed balloon flight (1.5 hrs)

4. The on-board electronics radio software, and flight software shall perform all software and hardware tasks necessary for communication on a balloon flight

Since Raytheon has requested that the TOMCAT project will build off of the base ThinSat created by NearSpace and Twigg's Space lab, the project needs to be designed to interface with the base. At a high level, this means that the electronics need to be connected, and all TOMCAT components need to fit within a ThinSat-shaped frame.

In order for the mission to be successful, the satellite needs to be able to communicate in both uplink and downlink. Raytheon has requested that the communications be designed to support a balloon flight, so the communications system needs to be designed to support the maximum range of the proposed balloon flight.

The payload is expected to go to space in the future, so in order to demonstrate a proof of concept that the satellite can withstand near-space conditions, the satellite needs to be designed to survive a balloon flight. Survivability includes working through the thermal and pressure conditions at high-altitude, as well as surviving the impact at the end of the balloon flight.

Other than the communications system, all other systems work to support the functionality or verification of the communication system. Thus, all other systems need to work together to perform all the tasks that are necessary to make the communications system work during flight. This includes everything from basic power requirements to signal processing software for the received signal.

All functional requirements give the design goals of the project, and give the basis for the design requirements developed for the project. The specific design requirements and how they are planning to be met are discussed in detail in a later section.

3. Design Process and Outcome

3.1. Systems Engineering

Katie

#	#.#	#.#.#	Requirement	Motivation	Validation
1			The payload shall interface with the bus according to the Thinsat Interface Control Document (ICD)	Customer has mandated the use of the ThinSat bus	System Demonstration
	1.1		The designed payload board shall fit within the ThinSat payload area as defined by the ICD	Customer has mandated that the SDR, comms hardware, and power system do not need to be in the payload area, but the other hardware components will	The components fit into the ThinSat bus
		1.1.1	A 0.25 mm tolerance shall be used when designing parts to fit within the ThinSat bus area	Conform to ThinSat ICD	The components fit into the ThinSat bus
	1.2		The designed payload board shall have a mass less than 55 grams	Conform to ThinSat ICD	Inspection of mass
	1.3		The designed payload shall interface with the bus computer via the provided payload connector	Conform to ThinSat ICD	The payload computer will interface with the bus computer via a flatsat test
		1.3.1	The payload shall be designed with a DF12(3.0)-20DS-0.5V (86) receptacle	Conform to ThinSat ICD	The components fit together with the provided ThinSat bus port
	1.4		The designed payload shall conform to the 'prohibited items' list as provided in the ICD	Conform to ThinSat ICD	Inspection of data sheets to ensure no component explodes or outgasses during flight
2			The payload shall communicate with the ground system during the balloon flight	Customer has mandated that the ThinSat must be commanded and must send telemetry to the ground during the balloon flight	System Demonstration
	2.1		The payload SDR shall be compatible with the ground SDR	The SDRs must be compatable in order to transfer commands and telemetry	Bench test with SDRs communicating through serial connection
		2.1.1	The ground SDR shall send and receive formatted data bits to/from the payload SDR over a wired connection	The wired connection allows for early C&DH testing	Bench test with SDRs communicating through serial connection
	2.2		The payload SDR shall wirelessly communicate with the ground SDR at a range > 30km	Verify that wireless communication works in preperation for the balloon flight	Long range wireless communication test
		2.2.1	The ground SDR shall wirelessly send and receive formatted data bits to/from the payload SDR	The SDRs must be able to communicate at a distance for the ThinSat to be commandable on a balloon	Bench test with SDRs communicating through wireless connection
		2.2.2	The payload SDR shall wirelessly send and receive formatted data bits to/from the ground SDR	The SDRs must be able to communicate at a distance for the ThinSat to be commandable on a balloon	Bench test with SDRs communicating through wireless connection
	2.3		The SDR shall operate at in ISM frequency band	Payload must use a carrier frequency that complies with the FCC, which is easiest in the amateur bands	Datasheet verification
	2.4		The SDR shall have an uplink rate such that the payload can receive at least one command packet every 10 seconds	The payload must be commandable. Since the data rate can only be determined from a full link budget analysis, the exact data rate is not chosen here	Bench test with SDRs communicating through wireless connection
	2.5		The SDR shall have an downlink rate such that the ground SDR can receive one telemetry packet every 10 seconds	The ground must receive telemetry. Since the data rate can only be determined from a full link budget analysis, the exact data rate is not chosen here	Bench test with SDRs communicating through wireless connection

#	#.#	#.#.#	Requirement	Motivation	Validation
3			The payload shall survive and operate in the environment during all stages of the balloon flight	Team needs to deliver completed product to the customer at the conclusion of the project	System Demonstration
	3.1		The payload shall be able to withstand the temperature range of flight, which includes the coldest point in the flight (-56° C), and a hot day on the ground (38°C)	The payload must work during testing and flight	Turn-on test in a dry-ice cooled cooler, turn-on test in the lab
	3.2		The payload shall withstand the impact from a balloon launch (estimated 9 m/s from vendor)	The payload must still operate after the balloon flight	Drop test from height that mimics 9m/s, followed by a power on test
	3.3		The designed payload shall operate between near-vacuum conditions of peak flight (~0.01 atm) and standard atmospheric pressure	The payload must survive and be usable after flight	Datasheet verification
	3.4		The IMU shall generate rate telemetry during ascent and descent	Telemetry must be generated so that it can be sent to the ground to prove that the C&DH system works	Lab test with IMU moving while on
		3.4.1	Science data shall be able to be collected and updated at least once every second	The telemetry must update fast enough that a model with reasonable resolution can be created	Software test
4			The On Board Computer (OBC), SDR Software, and Flight Software (FSW) shall perform all necessary software tasks for the balloon flight	Each part of the payload must be controlled to ensure that all parts work together to take and send science data during the balloon flight	System Demonstration
	4.1		The payload shall be commandable	Overall goal of the project	Bench test
		4.1.1	The SDR software shall send all command packets it receieves to the FSW	Received commands must go to the FSW for processing	Software test
		4.1.2	The FSW shall decode all commands received in CCSDS format	CCSDS format is mandated by the customer	Software test
		4.1.3	The FSW shall recognize errors in command formatting and return an error code	Formatting errors should be noted so that they can be fixed	Bench test with bad commands
		4.1.4	The FSW shall generate command counts for commands received in the correct format	Commands counts show that commands made it to the bus	Bench test with good commands
		4.1.5	The FSW shall generate command counts for commands with formatting errors	Command error counts show that commands made it to the bus, but that there was something wrong with them	Bench test with bad commands
		4.1.6	The FSW shall forward IMU commands to the IMU	Must be able to command the sensor suite	Bench test with sensor suite commands
	4.2		The payload shall generate telemetry	Overall goal of the project	Bench test, balloon flight
		4.2.1	The OBC shall receive raw telemetry from the IMU	The FSW should process the data that the sensor suite collects	Bench test
		4.2.2	The FSW shall packetize all (payload and health) telemetry in CCSDS format	CCSDS format is mandated by customer	Software test
		4.2.3	The FSW shall send packetized telemety to the SDR software	Telemetry must be sent to the SDR for downlink	Software test
	4.3		The OBC shall store all telemetry during flight	Allows the team to validate downlinked data with stored data	Balloon flight
		4.3.1	The OBC shall store IMU telemetry	Science data can be validated	Software test
		4.3.2	The OBC shall store payload health and safety telemetry	Safety data can be validated	Software test
		4.3.3	The OBC shall store command counts	Payload receipt of commands can be validated	Software test

#	#.#	#.#.#	Requirement	Motivation	Validation
5			Administrative requirements	Capture requirements that are non-technical in nature but still critical to the success of the project	Bookkeeping
	5.1		All purchased components must cost in total equal or less than the amount provided by the Aerospace Engineering Sciences department and other engineering funds obtained by the team	The project cannot cost more money than the team is given	The total cost at the end does not exceed the allocated funds
	5.2		The project must be FAA/FCC compliant	The project must be allowed to legally perform the balloon flight	Necessary permissions are given and limitations are met
	5.3		Any software used or generated by the project team shall be combatible with typical computer operating systems	The software components of the project need to be easy to manipulate and use	Software documentation review

3.1.1. Requirement 1.1

The primary ThinSat frame has a small area for electronics that should be filled with the payload board. The payload board contains the microcontroller and the inertial measurement unit. The other hardware components are too large to fit in this small area, so the customer decided that it was not mandatory for everything to fit in this area.

a. Requirement 1.1.1 The ThinSat ICD gives requirements for manufacturing components that the team will need to conform to.

3.1.2. Requirement 1.2

The ThinSat ICD recommends that the payload in the payload space should have a mass of less than 55 grams. Since only the payload board will be in this area, the other hardware components do not need to be contained within the 55g requirement.

3.1.3. Requirement 1.3

In mission development after the project, the TOMCAT payload should be able to send telemetry and receive commands from two communications systems– the TOMCAT system and the ThinSat system. In order to communicate with the ThinSat system, the TOMCAT payload needs to connect with the payload connector already on the ThinSat bus. Since the TOMCAT team will not have access to the ThinSat system in the project, the connection only needs to be established so that another team can make the software connection.

a. Requirement 1.3.1 The project needs to use the specific connector used on the primary ThinSat bus.

3.1.4. Requirement 1.4

For project safety, no part of the payload can explode or out-gas during flight, which is specified in the ICD.

3.1.5. Requirement 2.1

In order for the TOMCAT ground station to be able to communicate with the on-board communication system, both software defined radios in the systems need to be able to communicate with each other.

a. Requirement 2.1.1 At a base level, the SDRs need to be able to send data back and forth. The data that will be sent are commands, which at a basic form are just bits. If this can be performed over a bench test with the SDRs wired together, it shows that the system software works.

3.1.6. Requirement 2.2

The software defined radios need to be able to send data back and forth over a wireless connection in order to move towards a balloon flight. The maximum altitude of the balloon flight is 30km, so the radio system should be designed to function at least at that range.

a. Requirement 2.2.1 Over this wireless connection, the ground radio should be able to send commands or other formatted data files to the on-board radio.

b. Requirement 2.2.2 Over this wireless connection, the on-board radio should be able to send commands or other formatted data files to the ground radio.

3.1.7. Requirement 2.3

In an effort to avoid the necessity of a radio license, the team chose to operate in an open band that is available to everyone. Additionally, the team wanted to be at a frequency near the range used in space missions. These two requirements lead the team to operate in an ISM band.

3.1.8. Requirement 2.4

In order for the communication system to be successful, the radio system needs to be able to send commands from the ground to the satellite, even at a slow rate. The team decided that a reasonable, low end uplink rate should allow the system to send commands at least once every second. This allows the system to be successful at a low rate, but work to exceed the requirement.

3.1.9. Requirement 2.5

In order for the communication system to be successful, the radio system needs to be able to send telemetry packets from the satellite to the ground, even at a slow rate. The team decided that a reasonable, low end downlink rate should allow the system to send packets at least once every second. This allows the system to be successful at a low rate, but work to exceed the requirement.

3.1.10. Requirement 3.1

In order for the satellite to survive a balloon flight, it needs to withstand the temperature range of flight. This includes a high end for a hot day on the ground as well as a low end of the lowest temperature estimated on the 30km altitude balloon flight.

3.1.11. Requirement 3.2

In order for the satellite to be recoverable, it needs to survive the impact after the balloon flight.

3.1.12. Requirement 3.3

In order for the satellite to survive a balloon flight, it needs to function at the low pressure of the altitude-balloon flight.

3.1.13. Requirement 3.4

In order to verify the communication system, the satellite needs an instrument to generate verifiable data. Thus, the inertial measurement unit needs to generate telemetry during all parts of the balloon flight.

a. Requirement 3.4.1 In an effort to generate enough telemetry for a packet to be sent every 10 seconds, the team estimated that the IMU should generate telemetry at least once a second.

3.1.14. Requirement 4.1

In order to verify the communications system, the payload needs to be able to receive and execute commands.

a. Requirement 4.1.1 After the signal is received by the signal processing software and returned to packet format, it should forward all data it receives to the flight software to be decoded and further processed so that a command can be executed.

b. Requirement 4.1.2 By customer request, the team needs to use standard CCSDS packet protocol for all of its command and telemetry packets. Thus, the flight software needs to be able to decode one of these packets to understand what command has been sent from the ground.

c. Requirement 4.1.3 In an effort to have good awareness of if there are too many bit flips in uplink, the flight software should be able to recognized a poorly formatted CCSDS packet or poorly formatted command and give an error.

d. Requirement 4.1.4 In order to characterize the communications system, the flight software needs to generate a count of the number of good commands it receives and executes.

e. Requirement 4.1.5 In order to characterize the communications system, the flight software needs to generate a count of the number of bad commands it receives and rejects.

f. Requirement 4.1.6 The inertial measurement unit is the final destination for a command, so the flight software needs to be able to receive a command and format it such that the inertial measurement unit can understand it and react appropriately.

3.1.15. Requirement 4.2

In order to verify the communication system, the satellite needs to be able to generate telemetry that can be downlinked.

a. Requirement 4.2.1 The inertial measurement unit will generate telemetry, so the microcontroller needs to be able to receive that telemetry from the IMU.

b. Requirement 4.2.2 All telemetry that the flight software receives needs to be formatted in CCSDS format per customer request.

c. Requirement 4.2.3 The flight software needs to forward all telemetry packets to the signal processing software so that the packets can be modulated and downlinked.

3.1.16. Requirement 4.3

In the event that the communication system fails, the satellite should store all data so that when it is retrieved, the team can debug the problem. Additionally, if the data is stored, the team can validate the downlinked data with the stored data.

a. Requirement 4.3.1 To store all data, the microcontroller needs to store the inertial measurement unit science data.

b. Requirement 4.3.2 To store all data, the microcontroller needs to store all state of health data.

c. Requirement 4.3.3 To store all data, the microcontroller needs to store the command counters that the flight software generates.

3.1.17. Requirement 5.1

The project needs to fit within the budget. At a minimum, this can be the $5000 provided by the department. at a maximum, this can be the $5000 from the department, the $3000 from Raytheon for extra funding, and the $3000 from the Engineering Excellence Fund.

3.1.18. Requirement 5.2

In order for the project to be able to be successful, it needs to be legally able to perform the balloon flight, both in terms of the communications frequency and power and the balloon flight logistics.

3.1.19. Requirement 5.3

All software from the project should be able to run on someone's computer who would like to pick up the project after the team has completed it.

3.2. Top Level Design

Cole

The final design was a modified 2T ThinSat, two ThinSat slices stacked together, assembled in a way such that the primary communications and power storage were in the TOMCAT slice and the Power Distribution, C&DH, and Science equipment were in the ThinSat slice. Overall, the TOMCAT satellite was projected to weigh approximately 650g and conformed to the exterior profile of a standard cubesat, allowing it to be deployed from a typical cubesat canister like the one offered by Planetary Systems[8]. This can be seen in Fig. 4.

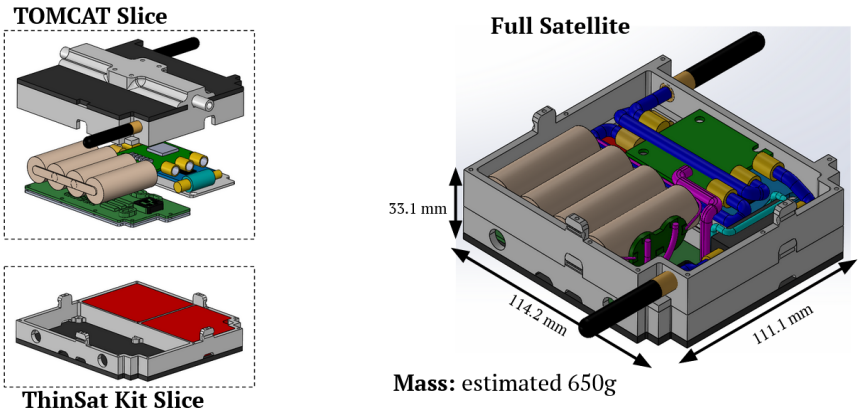


Figure 4. Top level design overview of the TOMCAT satellite

The major elements of the design will be discussed in detail in later sections, but include:

- Communications
- Flight Software
- Payload Board and Electronics
- Power Systems
- Science Payload

- Structures
- Thermal Systems

Additionally, while not located within the satellite itself, custom-configured Ground Station software and Communications equipment on the ground are critical to the project.

Some key design specifications are laid out in Tbl. 3.2 and further details are provided in each subsystem's respective section.

Table 2. Trade Topics and Weights for the MCU

Specification	Result	Margin
Link Margin	31.3 (DL), 15.53 (UL) [dB-Hz]	3.13 (DL), 1.55 (UL)
Data Rate	72,000 (DL), 128 (UL) [bps]	N/A
Battery Capacity	4046 [mAh]	1.12
Absolute Max Current	3 [A]	1.8

3.3. Communications

Lauren DeMoudt, Srikanth Venkataraman, Trevor Weschler

3.3.1. Conceptual Design

The customer has specified that the ThinSat shall use a software-defined radio (SDR) for communications with the payload. Software-defined radios differ from traditional radios in that the radio control typically associated with the hardware is done via software. This includes frequency adjustment, filtering, etc. SDRs are becoming more popular because the flexibility they provide over standard radios is highly desirable. SDRs allow the user to change transmission and receiving parameters with the use of software rather than modifying the hardware. For TOMCAT, the large amount of SDR options were narrowed down to four radios that are suitable for the mission. For future missions, the entire SDR subsystem will ideally fit onto a dedicated ThinSat slice. While this is not a requirement for TOMCAT, this desire does provide criteria to narrow down suitable options from the large amount of SDRs available. The main criteria used to select these radios were minimization of size, mass, and power draw. On a ThinSat, access to all of these criteria is constrained due the the small size of the spacecraft payload area.

a. SDR Trade Study The primary trade study conducted for Communications was with respect to the SDR model that would be selected. Further information on this can be found in the Communications Appendix, and ultimately, the LimeSDR Mini was selected. This SDR is the cheapest available option at $189.95 while having the lowest mass and size. Where this option separates itself is its large bandwidth capability at 96 MHz. This will allow the entire 902-928 MHz range to be usable if necessary for data transmission.

This SDR has a low max receiver gain compared to the other SDR options. An average max transmission gain will most likely mean that a separate RF amplifier will be necessary to ensure the communications system works at the distances expected on balloon flight. Additionally, the LimeSDR Mini doesn't have much MATLAB support. Since the TOMCAT team is experienced in using MATLAB, being able to interface with the ground SDR using MATLAB is preferable. This would have cut down in overall development time for the communication system since unfamiliar software would not have been used to interface with the ground SDR. On the flight SDR side, this radio has GNU Radio support. GNU Radio is the software that was used to interface with the flight SDR.

Additionally, the LimeSDR is rated to operate down to -40°C, but is normally operated at 25°C. The lowest temperature TOMCAT was expecting to see, atmospherically, was -56°C. This meant that there

would have to be some thermal design in order to ensure the radio continues to work throughout the balloon flight. The extent of this thermal control depended on modeling the thermal characteristics of the radio.

Pros	Cons
Cheapest viable option	Low receiving gain
Largest bandwidth capability	Average transmission gain
Small size	Limited MATLAB support
Low mass	Typical 25°C operating temperature
Down to -40°C operating temperature	

In addition to the formal trade study for SDR selection, the team looked at different options for transmission frequency and logistics, flight and ground antennas, and signal processing protocols. While it was determined that these decisions didn't require a full study, numerous options were considered for each.

b. Transmission Frequency and Duplex Communication An ISM frequency band was selected since that meant that no additional permitting was required to transmit as long as certain regulations were followed. This narrowed frequency selection between the 902-928 MHz frequencies (referred to as 900 MHz) and 2.4-2.5 GHz (referred to as S-Band). Initially, the S-Band was selected because the larger bandwidth (100 MHz overall) would allow for more data to be transmitted at a time. Additionally, a variant of the S-Band is often used for space-based communication. However, due to the better ranged performance of the 900 MHz band given our other design options, the S-Band was eventually dropped and switched for the 900 MHz frequency band. While its more limited bandwidth (26 MHz) would limit communication speed, it was still plenty large enough to accommodate project requirements. Additionally, the team opted to use a half-duplex communication scheme since that would allow for the full 26 MHz bandwidth to be used when communicating. The only limitation this would put on the system overall would be that telemetry and commands could not be sent simultaneously.

c. Antennas Given the ISM frequency selection, antenna choices were narrowed considerably. The main design considerations then were antenna style on the ground and in flight. An omni-directional antenna was preferred for the flight antenna, as even with it's overall decreased gain, it could still communicate as the payload was shaken around on the balloon. With the uncertainties in these vibrations, an omni-directional antenna allowed for extremely lax pointing requirements on the satellite. The ANT-916-CW-RAH, a quarter-wave monopole antenna from Linx Technologies[9] was ultimately selected for both antennas on the TOMCAT satellite.

To make up for the lost gain on the satellite antennas, a high-gain antenna was selected for the ground. Since the path of the balloon could be somewhat predicted and the ground station able to be positioned prior to the flight, an antenna with more strict pointing requirements could be selected to give a higher gain. Additionally, the strict mass, size, and power requirements on the satellite did not restrict the ground system. After considerable research, a four element Yagi Antenna (Model A09-Y8NF) from Arcadian Incorporated[10] was chosen.

d. Signal Processing Some method was necessary to convert the data from the flight or ground station computers into a modulated digital signal that the radios could transmit. This was accomplished through the signal processing software, of which there were two main options: MATLAB/Simulink or GNU Radio. Either option would have been acceptable, however given the LimeSDR Mini's poor compatibility with MATLAB, it was decided to move forward with GNU Radio. A choice of modulation technique was also required. Further details can be found in the Communication Appendix and in subsequent sections, however Binary Phase Shift Keying (BPSK) modulation was selected because it was easier to implement than alternatives while still meeting bandwidth and data rate requirements.

3.3.2. Resulting Communications Design

a. Link Budget The link budget is a critical model of the design that ensures that the communication between the ground station and ThinSat can be achieved. The goal is to account for all of the gains and losses in the system to determine if a positive link margin can be achieved. A positive link margin means that the signal is powerful enough to be received at the intended destination. To do this, the signal must overcome losses in propagation and remain a signal level higher than the noise encountered by the system. The first part of the link budget includes general parameters that were derived from design requirements or choices.

Table 3. TOMCAT Link Budget: General Parameters

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Frequency	0.915	0.915	GHz
Wavelength	0.328	0.328	m
Range	104	104	km

Table 3 shows the general parameters upon which the link budget is built on. As discussed previously, the chosen uplink and downlink frequency is 0.915 GHz. The reasoning is that since this frequency lies in an ISM band- we don't need to get a radio license to use this frequency. Additionally, it has a 26 MHz bandwidth which is more than necessary for TOMCAT purposes. Bandwidth is related to how much data can be encoded on the transmitted analog signal. Since TOMCAT plans to use the same frequency for both uplink and downlink, the communications system will be operated in half-duplex mode. This means that the system will not transmit and receive at the same time. The wavelength parameter is calculated from the chosen frequency.

The range parameter is determined by the maximum altitude that the balloon is expected to reach and the horizontal drift of the balloon. The maximum altitude expected is 30 km. This is a typical altitude of the Gateway to Space balloons. The maximum horizontal drift expected is 100 km. This value is based on historic flights and NOAA information on high altitude weather balloon drift. The worst case scenario for the link budget occurs when the balloon has drifted 100 km and is at the peak altitude of 30 km. The link budget will be computed at this worse case scenario for a range of 104 km (the hypotenuse of the triangle formed by 30 km height and 100 km drift).

Table 4. TOMCAT Link Budget: Data Parameters

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Bit Error Rate	10^{-6}	10^{-5}	[-]
Modulation Scheme	DPSK	DPSK	[-]
Required Eb/No	11.8	10.6	dB
Data Rate	256	144,000	bps
Pr/No	35.86	62.18	dB-Hz
Design Margin	3	3	dB
Minimum Pr/No	38.88	65.18	dB-Hz

Table 4 shows the data parameters of the link budget. These represented expectations of how TOMCAT would have handled the data and the requirements that fall out of those design choices. Since TOMCAT had no customer requirements for bit error rate, suggested values from SMAD were used. These values are 1 in every million bits on the uplink are in error while 1 in every 100,000 bits are in error on the downlink. This was a key parameter because it was one of the criteria that would have been used for testing during verification.

Modulation scheme is the method by which data is encoded onto an analog radio signal to be broadcast.

The modulation scheme that was chosen for TOMCAT is differential phase shift keying. The reasoning is that this scheme is fast and accurate. This choice drives the next parameter, required Eb/No, or ratio of received energy per bit to noise density. This value can be interpreted as how powerful the signal must be relative to the noise in order for the signal to be received. To determine the required Eb/No, SMAD can be used. Using fig. 5, the required Eb/No can be computed using the selected bit error rates and modulation scheme.

The data rate parameter was based on design choices for TOMCAT. Each command was expected to be a maximum of 64 bits long. A design choice was to be able to command at least twice per second. This means uplinking at a minimum of 128 bits per second. However, since the system operated in half-duplex mode, the uplink was multiplied by two again resulting in an uplink rate of 256 bits per second. The downlink was driven by how much data is produced by the IMU. In order to downlink all of the accelerometer data real time, a rate of 72,000 bits per second was needed. Once again, this value was multiplied by two to account for operating in half-duplex mode bringing the total downlink rate to 144,000 bits per second.

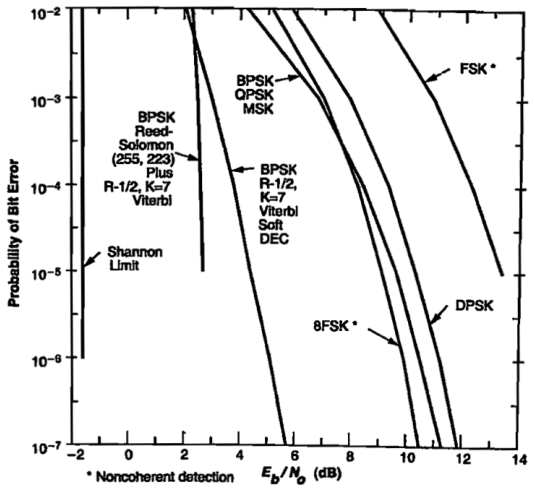


Figure 5. Required Eb/No SMAD (SMAD)

The next parameter Pr/No (carrier to noise ratio density) was the main parameter used to compute the link margin. Pr/No can be understood as the signal power divided by noise. The difference from Eb/No is that Pr/No deals with the signal instead of individual bit energy. It was found by multiplying the required Eb/No by the data rate. Performing this computation for both the uplink and the downlink provided the values in Tbl. 4. The design margin was required because real systems do not follow Fig. 5 perfectly due to system imperfections. Adding a design margin of 3 dB onto the Pr/No gives the minimum Pr/No. This value was the Pr/No that the system must be able to produce for the signal to reach the destination with enough power to be received.

Table 5. TOMCAT Link Budget: Noise Parameters

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Receiver Antenna Noise Temperature	873.03	97.7	K
Receiver Cable Loss	0.5	0.5	dB
Receiver Noise Figure	2	2	dB
Receiver Noise Factor	1.6	1.6	[-]
Receiver Noise Temperature	290	290	K
Receiver System Noise Temperature	1098.73	323.40	K
Receiver System Noise Power	-198.19	-203.5	dBW-Hz

Table 5 walks through the noise parameters that must be overcome by the link design. One common way to represent noise is in terms of degrees Kelvin. It is important to note that this is not the temperature of the item being discussed, but rather a measurement of the noise at that component. In general, the noise is usually divided into two parts: the antenna noise temperature and the receiver noise temperature.

The receiving antenna noise temperature was the noise encountered at the antenna. For the uplink, this is the antenna on board and for the downlink, this is the antenna on the ground. The downlink includes noise from the business area, galactic noise, black body noise, Earth noise, and PCB noise. The uplink includes

noise from the business area, galactic noise, and black body noise. Note the downlink noise is the most uncertain portion of the link budget and is part of the reason the link budget is so large.

The receiver noise temperature was further divided into two parts: the cable loss and then actual receiver noise. The receiver cable loss was computed based on the choice of cable that TOMCAT was using. Since the same cables are used for the uplink and downlink, the cable loss is the same. The receiver noise figure was a given value in the datasheet for the LimeSDR Mini. The receiver noise temperature was determined by multiplying the reference temperature by the noise figure minus 1. The typical reference temperature was 290 K. Since the noise figure is 2, the receiver noise temperature was also 290 K.

The final parameter in this section was the receiver system noise temperature. This was the combination of both the receiver antenna noise temperature and receiver system noise temperature. Once again, the downlink value was the most uncertain part of the budget. This was mostly due to the effects of the Sun on the noise. If the balloon drifts close to the sun, the noise becomes much greater. However, away from the Sun, the noise contribution was relatively modest (about 0.5 dB). Another uncertainty was how much noise will be encountered from other use in the band. This was taken into account with values recommended from SMAD, but more noise may have been encountered since the band was open. These uncertainties are mitigated by having a large link margin.

Table 6. TOMCAT Link Budget: Receiver and Propagation Parameters

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Receive Antenna Gain	2.2	8.5	dBi
Receiver Cable Loss	0.5	0.5	dB
Space Loss	132.04	132.04	dB
Atmospheric Attenuation	0.45	0.45	dB
Polarization Loss	3	3	dB

Table 6 shows the receiver and propagation parameters. These include parameters based on the hardware choices for the reception system and the losses due to propagation. Often times, the propagation loss is the greatest loss in the system, and that was true of the TOMCAT system. Effectively designing to overcome the propagation loss is mandatory for all communications systems.

The receiver parameters include the antenna gain and cable loss. The antenna gain is given in units of dBi. This was referenced to an isotropic radiator where all power is equally transmitted in all directions. In reality, such a radiator does not exist, nor is it necessarily beneficial. A real antenna has a beam width pattern which describes the direction in which the antenna radiates. Since the power was now confined to this area, there was a gain associated with this directionality. TOMCAT was using a monopole antenna on board. This antenna radiates in a hemisphere and has a modest gain of 2.2 dBi. The ground station used a Yagi antenna which was more directional. Due to the higher directionality, the gain was an increased value of 8.5 dBi. It should be noted that TOMCAT was a static pointing system, otherwise pointing considerations would have to be taken into account here.

The propagation parameters are given in Tbl. 6 as well. The propagation loss is how much power is lost in the signal over the intended transmission distance of 104 km. Equation 1 shows how this value is calculated based on wave length (λ) and distance (R). Additionally, atmospheric attenuation contributed to loss in the signal of 0.45 dB. This value can be found by referencing Fig. 6 taken from SMAD. Finally, polarization loss was estimated to be 0.2 dB (again from SMAD).

$$L_s = 10\log((\frac{\lambda}{4\pi R})^2) \tag{1}$$

Table 7. TOMCAT Link Budget: Transmitter Parameters

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Transmit Antenna Gain	8.5	2.2	dBi
Transmission Cable Loss	0.5	0.5	dB
Transmission Power	-2.2	3	dB
Power Transmitted	0.6	2	W
Effective Isotropic Radiated Power	5.78	4.71	dB

Table 7 gives the parameters associated with the transmission characteristics of the system. These include the gain of the transmission antenna, line loss, transmission power, and effective isotropic radiated power. In general, it is best to transmit with as much power as possible to reduce to necessity for amplifiers along the RF chain. Choosing a high gain antenna also helps remove the need for more amplifiers in the chain.

The first parameter is the transmit antenna gain. It can be noticed that essentially, the antenna gains have swapped places from the receiver antenna gain line. This is because on the uplink, the Yagi antenna was transmitting while the monopole antenna was transmitting on the downlink. The exact same Yagi and monopole antennas were being used, so the gain for each are still 8.5 dBi and 2.2 dBi respectively. The line loss was also 0.5 dB as the same cables were used.

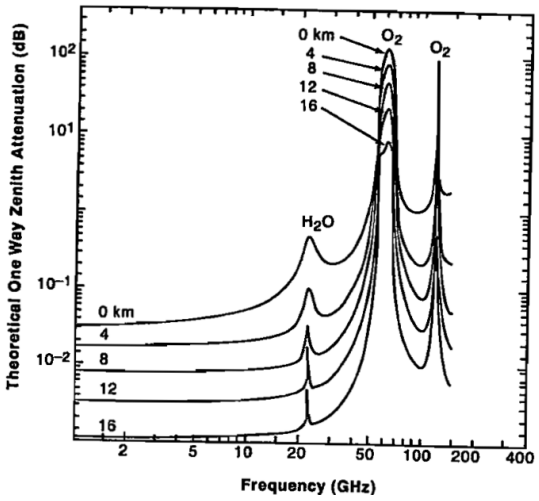


Figure 6. Expected Atmospheric Attenuation (SMAD)

The next parameter is power transmitted, which is subject to multiple constraints. The driving factor in this case is the legality of broadcasting power in this band. The effective isotropic radiated power could not have been above 6 dB in order to remain legal. Due to this constraint, the power could have either been made up in the power transmitted or the antenna gain. On the ground side, the antenna gain was high enough that the power transmitted can be lower at 0.6 W. This had the benefit of being less power intensive to operate. On board, the antenna gain was small, which meant the power transmitted must have made up more of the effective isotropic radiated power. Note that the downlink power transmitted had decreased by 0.6 W from FFR due to real world testing of the power amplifier. This is discussed in depth later in the report.

Finally, the effective isotropic radiated power is the sum of the transmitted power, antenna gain, and line loss. As mentioned earlier, this value had to be lower than 6 dB in order to remain legal. In order to meet this value, easily implementable amplifiers were used to boost the signal. In order to fine tune the power transmitted, the LimeSDR Mini has an onboard amplifier which can be programmed for up to 72 dB of gain. The team fine tuned this value in order to get as close to 6 dB of effective isotropic radiated power as possible. Estimates showed that 5.78 dB on the uplink and 4.71 dB on the downlink are reasonable. This is due to the step change of 1 dB on the transmitter gain. There wasn't a step there makes the effective isotropic radiated power exactly equal 6 dB.

Table 8. TOMCAT Link Budget: Summary

Parameter	Uplink (Command)	Downlink (Telemetry)	Units
Effective Isotropic Radiated Power	5.78	4.71	dB
Propagation Loss	132.69	135.49	dB
Receiver Gain	1.7	8	dBi
System Noise Power	198.19	203.5	dB-Hz
Pr/No	72.98	80.72	dB-Hz
Minimum Pr/No	35.87	65.18	dB-Hz
Link Margin	31.3	15.53	dB-Hz

In summary, Tbl. 8 shows key values that summarize the TOMCAT link budget. The new item that was introduced here is the Pr/No of the designed system. This value can be computed with Eq. 2 where EIRP is effective isotropic radiated power, L_s is space loss, L_a is atmospheric loss, G_r is receiver antenna gain, and T_s is system noise temperature. Note that this equation is in terms of decibels and that 228.6 comes from the logarithm of the Boltzmann constant.

$$\frac{P_r}{N_o} = \text{EIRP} - L_s - L_a + G_r - 10\log(T_s) + 228.6 \tag{2}$$

This Pr/No represents what the designed system was capable of producing. In order to get the final link margin, the minimum Pr/No was subtracted from the Pr/No of the system. This gave a 31.30 dB-Hz and 15.53 dB-Hz margin for the uplink and downlink, respectively. The margin was large in order to account for any modeling issues with the link budget such as an underestimation of noise. This also allowed more margin in case the balloon drifted further than expected.

b. RF Chain The link margin design ensured that the signal could reach the destination and be received. However, a signal must be conditioned before it processing by the software-defined radio. Essentially, filtering had to be employed to avoid aliasing in the signal. Despite the capabilities of the software-defined radio, it cannot remove signal aliasing through its software. To avoid signal aliasing, a low pass filter was implemented in the reception chain for both the ground and on-board systems. In order to simplify the design, the same chain was used for both the on-board and ground reception systems. Figure 7 shows the on-board receiver chain.

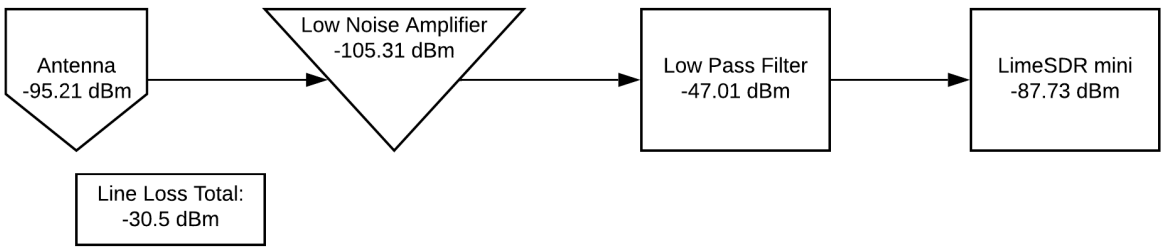


Figure 7. On-board Receiver RF Chain

It is important to note that the introduction of a low pass filter reduced the signal strength. In order to account for this, a low noise amplifier (PMA4-33GLN+) was introduced to the chain. This amplifier boosted the strength of the signal without adding too much noise. At each stage of Fig. 7, the signal strength was printed on the component. In order to be decoded by the radio, the signal had to arrive with at least -100

dBm of strength. As indicated, the signal arrived with -87.73 dBm, meaning that the signal should have been able to be decoded on-board. A similar analysis was done for the ground communications system discussed in the Appendix.

3.3.3. Ground Station Placement

As mentioned in the conceptual design section, the team chose a four element Yagi antenna for the ground station to maximize the gain without sacrificing too much beamwidth. However, the beamwidth is still limited to 100 degrees in the horizontal direction and 70 degrees in the vertical direction. The team was initially concerned that if the ground station was simply placed near the launch site, the balloon would rise outside of the vertical beamwidth. The team would have to respond to this issue by implementing a system to change the pointing of the antenna as the balloon ascended. To avoid the complexity involved with automated or manual pointing, the team chose to instead displace the ground station such that the balloon would still be within the vertical beamwidth of the antenna at its maximum altitude of 30 km. A visual representation of this concept is shown in Figure 8.

Figure 9 conveys the result of the geometric analysis implemented to determine the required offset. As can be noted from this image and recalling the 70 degree vertical beamwidth of the ground antenna, the minimum required offset for the ThinSat/Balloon configuration to always remain within view is 10 km.

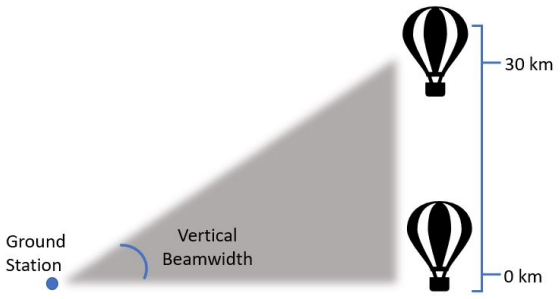


Figure 8. Ground Station Offset Concept Visualization

Another concern arose after deciding to offset the ground station. This concern was that the geography of the area surrounding the launch site could make it difficult to communicate with the ThinSat until it ascended a sufficient distance above the ground. Thus, the team decided to place a second, identical ground station at the launch site. The ground station at the launch site will be used to command the ThinSat until telemetry can be heard and understood by the offset ground station. At this point, commanding will be done from the offset ground station while the ground station at the launch site can still provide verification of the communications. The TOMCAT team will remain in constant contact with each other to ensure a smooth transition. With a ground station at the launch site, the team would also be able to verify ThinSat communications prior to the launch. This could also serve as an important troubleshooting tool if something is not working as planned on the day of the launch.

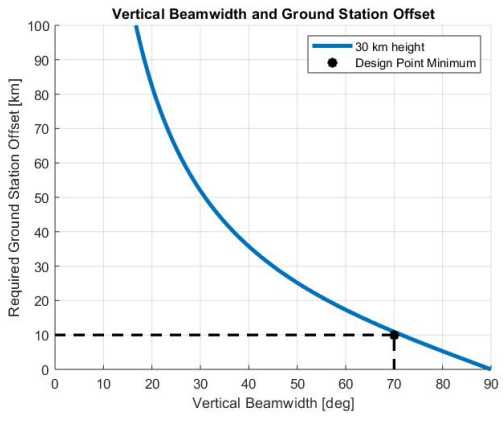


Figure 9. Required Ground Station Offset

After the discussion of how to adjust the design to account for the constrained vertical beamwidth, the question of the constrained horizontal beamwidth arises. The team consulted balloon flight data from the Gateway to Space balloon flights and it was determined that the balloon flight typically travels in one direction with lateral movement off that direction not approaching distances of any concern with an antenna horizontal beamwidth of 100 degrees.

a. Communications PCB The communications system on board will have its own PCB to hold the low noise amplifier and the power amplifier. The goal of the PCB is to reduced the mass and space required by the communication system. Without a custom board, each component would need to have

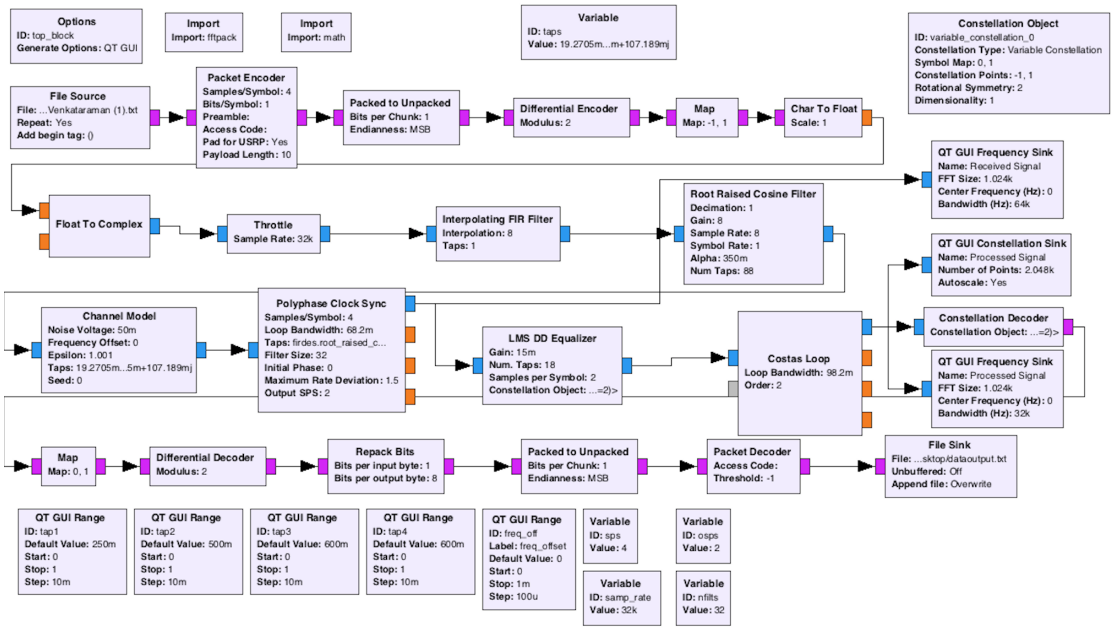


Figure 11. Full GNU Radio simulation block diagram

its own evaluation board, for which there is not room. The design is shown in Fig. 10. The ground has no such size requirements and an evaluation board will be used for the low noise amplifier on the ground.

b. Communications Software The communication software will provide an interface between ground/flight software and the SDRs. The communication software will run on both the ground station laptop and the OBC, and will be handling transmission and reception on both. A software development kit called GNU Radio was used to develop and test the communication software. The design of the transmission and reception signal processing chains will be described below. To verify the signal processing model would work as intended to modulate, process, and demodulate a signal without having to hardware test, a GNU Radio simulation was constructed. The full block diagram is pictured in Fig. 11.

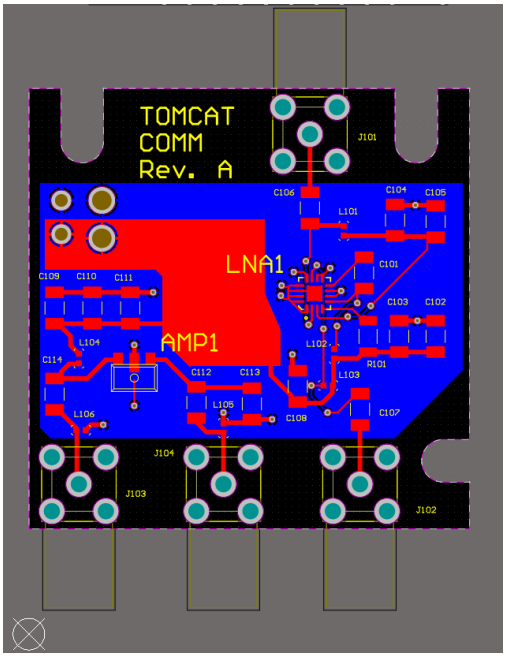


Figure 10. Custom Communications PCB Design

c. Transmission The purpose of the transmission side of the communication software is to take binary data from from ground and flight software over a UDP connection and turn it into a digital signal that can be transmitted by the SDR. The modulation scheme used is called differential binary phase shift keying (DBPSK). It differentially encodes data in phase shifts in a carrier wave. The carrier wave will shift between two phases offset by 180 degrees. A

phase shift in the carrier wave represents a one, while a lack of a phase shift represents a 0. Thus, a series of ones and zeros can be encoded in the wave.

Since the square wave generated by modulation has infinite bandwidth, the signal needs to go through a process called pulse shaping to constrain the frequency spectrum to a particular bandwidth. A root raised cosine filter was used to pulse shape the signal.

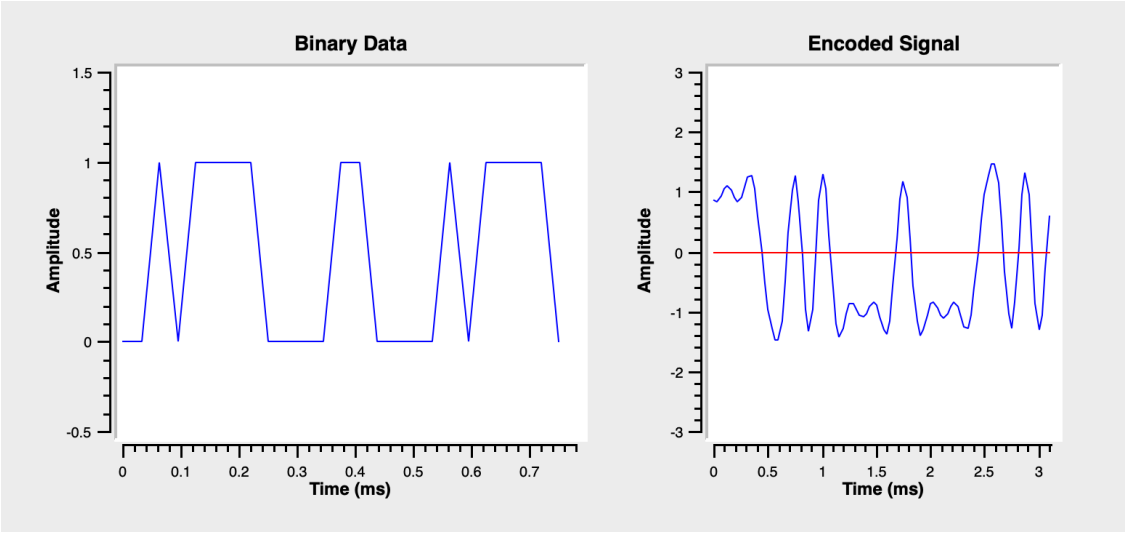


Figure 12. Binary data before and after modulation process

3.3.4. Reception

On the reception end, a digital signal will be taken from the SDR, processed to account for interference introduced during transmission, and decoded back into binary data to be sent to flight or ground software. Since the two SDRs' oscillators are not perfectly in sync, the receiving SDR will not sample the incoming signal at the optimal points, which results in a completely unintelligible signal in most practical situations. A polyphase clock synchronization algorithm was used to account for this. Another source of interference is multipath interference, where the transmitted signal reflects off of objects in the environment and arrives at the receiver multiple times. The results is that some frequencies along the bandwidth show increased gain, while others show reduced gain. A least means squared equalizer was implemented to undo the effects of this interference. This equalizer uses knowledge of what the incoming signal should look like to provide an error value for the LMS algorithm.

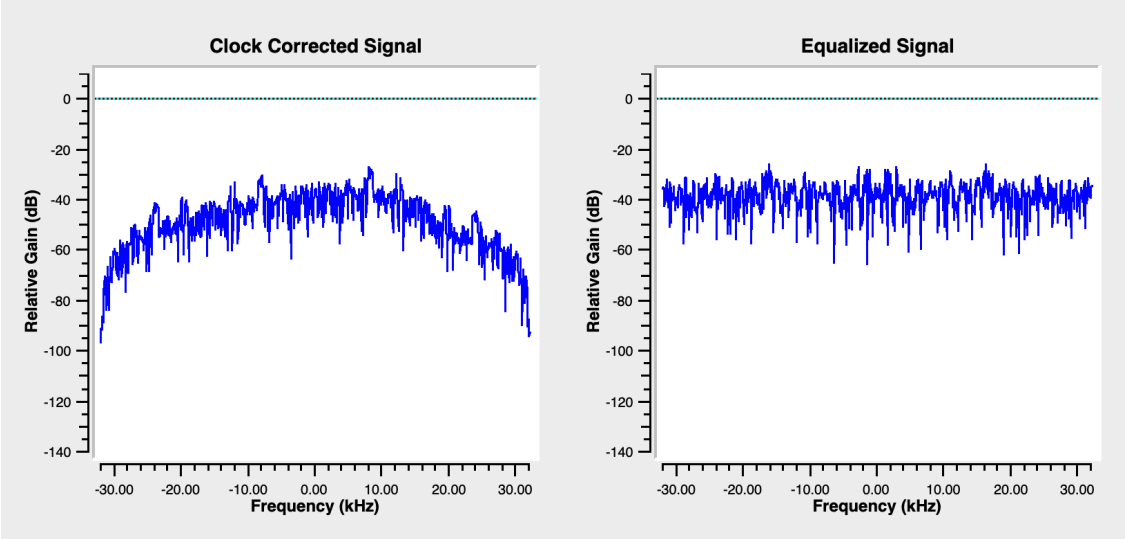


Figure 13. FFT Spectrum before and after equalizer

From this FFT plot, it can be seen that the equalizer is working as intended and boosting the power on lower power frequencies so all the frequencies have roughly the same gain. Since the oscillators of the two SDRs drift relative to each other, there will also be frequency drift in the carrier wave. A Costas loop algorithm was used to account for this drift. Finally, the signal can be demodulated by reversing the modulation process described above.

3.4. Flight Software

Manuel Lindo, Cole Kenny, Katie Steward

3.4.1. Conceptual Design

a. Software IDE Trade Since using KubOS, an off-the-shelf flight software framework, was not a requirement for the project, but instead a general request from the customer, the team decided early on to trade against multiple options. Three major IDE choices were considered, including KubOS, the Arduino IDE, and Eclipse. Details on this trade study can be found in the Appendix, however ultimately, KubOS was selected as the team's primary FSW base since many critical features were already designed and would only need to be configured by the team for use.

b. KubOS and Lightweight FSW In order to reduce overall project risk, it was determined that a second, more bare bones FSW package would be advantageous for testing purposes. This would allow the KubOS team to continue development of their flight software for the balloon flight while a second software team designed so-called 'lightweight' flight software for use in systems and integration tests. This parallel development allowed for fast iteration and the Lightweight FSW provided minimum viable functionality for the project per project requirements. The Lightweight FSW was written using Python. Since KubOS was still the primary choice for the team for operations, and given that neither software package had been fully tested at the time of the COVID-19 work stop, further discussion will be limited to KubOS.

3.4.2. Final Design

KubOS provides a number of "microservices" while fulfilling basic low-level flight software tasks. These include file transferring capability, monitor services for hardware and telemetry, a telemetry database built

on SQLite, shell services for uploading new code, and hardware and communications services, which can be configured for a number of different kinds of radios.

The primary advantage to using the KubOS software is that the team can focus on coding the mission software that specifically accomplishes the tasks that the team needs in order to communicate with the IMU and fulfill the project objectives. The built-in KubOS software provides a framework for the low-level services that the TOMCAT mission needs that can be fine-tuned to fully support the mission. KubOS is built off of a Linux system, so the low-level software focuses primarily on the required Linux services. These built-in services are summarized in the table below:

Table 9. KubOS Basic Services

Services	Basic Functionality
File Transfer	Focuses on moving command and telemetry files between different parts of the system. For the TOMCAT software, this will include between the IMU, the signal processing software, and to the telemetry storage on the microcontroller.
Monitor	Primarily works to ensure that all software processes are running as intended. Keeps track of software isues for easier debugging.
Telemetry Database	The telemetry database is stored in the microcontroller memory using a standard Linux SQLite database. This will allow telemetry to be stored for the duration of the balloon flight so the team can compare the downlinked data to the onboard stored data.
Shell	This service is primarily for externally communicating with KubOS software. A Linux shell prives a command line so that Linux operations can be performed on the software while the software in the development and testing.
Hardware	The hardware tools are low-level functions to perform low-level Linux tasks. This is primarily used to communicate with the microcontroller to execute hardware changes. An example of when this is necessary is when the software needs to query the database or use mutations.
Communications	The communications service will primarily communicate with the signal processing software. It will receive the command packets from the signal processing software and send telemetry packets back to it.

These services provide the low-level foundation for the TOMCAT flight software to be built upon. The following table shows the mission applications that need to be built/modified to support the TOMCAT mission:

Table 10. TOMCAT's Mission Applications

Mission Applications	Basic Functionality
Housekeeping	Reduces mission application complexity and increases fault tolerance by separating critical hardware components in software. This ensures that if some hardware system fails, they don't all fail with it.
Passive Telemetry Collection	Stores collected telemetry to the telemetry database. Polls hardware services periodically to collect data to store.
Deployment	Manages any tasks that need to occur after deployment from a launch vehicle but prior to normal system operation. TOMCAT's focus in this area will be limited due to project scope, but research will be provided to Raytheon.
Beacon	Downlinks critical telemetry (such as those related to system health and diagnostics) at a regular interval separate from the usual telemetry packets. While separate from regular data telemetry, the Beacon application still uses KubOS's communications service.
Operations	Defines the actual mission activities of the satellite.

Figure (14) shows the interactions between the custom designed mission applications and the core services KubOS provides.

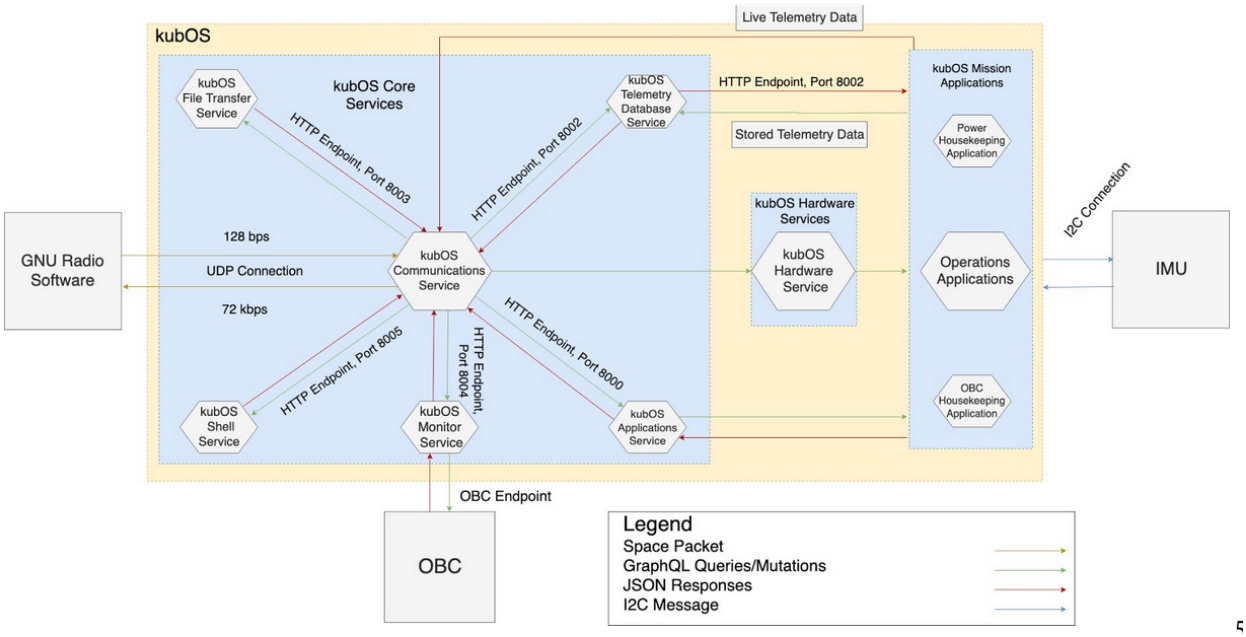


Figure 14. Flight Software Framework's Freebody Diagram

3.5. Ground Station Software

Lucas Perry

Numerous options were considered when selecting the Ground Station Software. Most notably, the final three options were KubOS, an in-house ground software, and Ball Aerospace's COSMOS. These software routes were weighed by their pros/cons, and ultimately COSMOS was the selected software to use.

Trade Study Results The first option considered was KubOS because it was already being designed as a flight software, and could send/receive CCSDS packetized data. Its pros include packetizing/depacketizing CCSDS packets and it is known to be able to receive and transmit data over a UDP port, and, most significantly, is already being used for the flight software, so minimal extra work would be required. This said, some of the downsides to KubOS are that it wasn't created to act as a ground software, therefore it has no graphical user interface. Besides, it would require another PCB for the ground system for it to be uploaded on, increasing complexity. In the end, KubOS did not fulfill all the requirements needed for a ground software as it lacked the interface and full scope of a ground system software program.

The second considered option was Creating an in-house software program. This was one of the options with the best ability to meet the needs of the ground software requirements. The code for an original program would have been written using C++ or Matlab. Some of the benefits of this method would be that our in-house software would have exactly what we needed to fulfill the mission requirements, no more and no less. This said, it was decided that this option would be extremely difficult and time-consuming and that it was out of the scope of the project, so the option was decided against.

The final option, which was also the selected one, was Ball Aerospace's COSMOS. As an open-source program with a lot of useful applications, COSMOS was able to satisfy all the requirements. It has a Command Sender application that was capable of sending commands in CCSDS format, it could receive/transmit data through a UDP socket, and its Telemetry Grapher and Extractor provided methods of graphing data with an interface and logging it. The single downside to COSMOS is that some work was required to learn the language of the configuration, which was formatted in Ruby.

In the end, COSMOS was chosen as the ground software because it is a very powerful program that easily fulfilled all the requirements. Many of the built-in applications were usable for the mission, so the design portion came in the form of writing configuration files for the tools, making a key for the CCSDS format, and creating specific commands to fulfill mission objectives. Additionally, COSMOS came recommended by the customer.

COSMOS Final Design After determining that COSMOS Ground Software and interface satisfied all the requirements, a plan was formed to assimilate COSMOS's features into the current mission. As stated previously, the main purpose of a ground software is to send commands, receive and store telemetry. More specifically, it must send commands packetized in CCSDS format over a UDP connection to the communications software, also running on the ground station computer. Then it must be able to receive telemetry data at a data rate of 72kbps in packetized CCSDS format, have the ability to plot the raw data, and then depacketize and store in text file format for later analysis. The major functional requirement that the ground software satisfies is:

FR2: The payload shall communicate with the ground system during the balloon flight.

Although COSMOS contains the foundation for everything that is needed for this mission, the design portion came, for the most part, from writing configuration files. Configuration files define how a tool in the COSMOS suite acts, and what it should be doing. It is crucial to the success of the project that the commands/telemetry packets are in a specific format of CCSDS. Because CCSDS format is a general term, the exact format of the header and the data included in the header was chosen and integrated into the configuration for the Command tool and the Telemetry tool suite. These edits to the configuration files were written in Ruby, and direct the application to know what format of file to expect for sending and receiving packets.

With these configuration files taken care of, the data can be transferred in the standard manner COSMOS uses. Firstly, the Command & Telemetry Server is started, and COSMOS reaches out to all it's predetermined target systems to make a successful connection. When this has been accomplished, the commands that were created previously were wrapped up in a space packet, and sent using the Command Sender tool

up to the ThinSat via the on-board computer. The flight software is in charge of querying the science data from the IMU, as well as health status data of the flight software, and sending this data back down in a space packet from the ThinSat. This telemetry data is sorted and plotted using the Telemetry Grapher, logged from the Command & Telemetry Server, and depacketized and transformed into a text file using the Telemetry Extractor.

The general functional block diagram for the workings of the ground software is shown below in Figure 15.

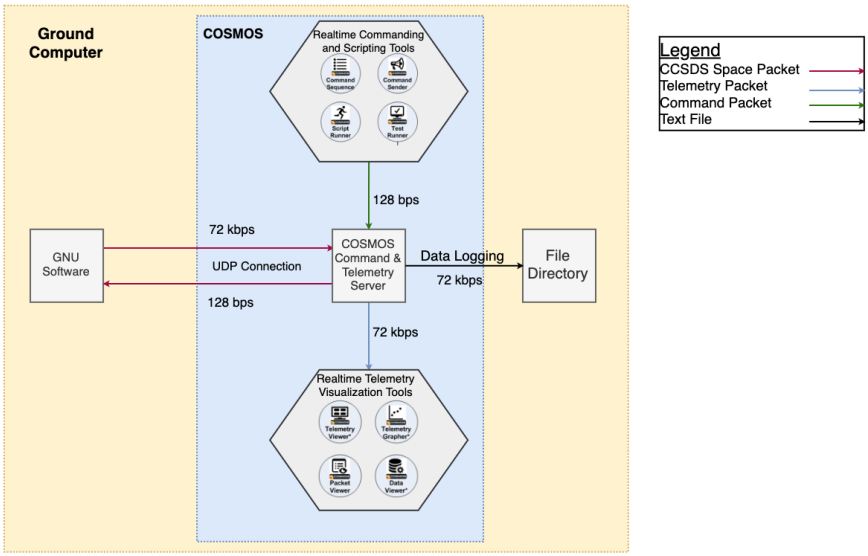


Figure 15. Functional Block Diagram for Ground Software

3.6. Payload Board and Electronics

Grant Novota

3.6.1. Conceptual Design

To meet the project objective of outfitting a ThinSat with an on-board communication system that receives commands and transmits data from the payload to the ground, the team needed to select a micro-controller unit (MCU) to provide all of the control functionality necessary for the operation of the ThinSat subsystems. The MCU needs to operate within the power budget set by the system's power supply and be able to drive various power outputs to the subsystems it is controlling, such as the SDR and IMU. It must also be capable of meeting all the real-time processing requirements that are set for the system to function as expected. Additionally, there must be a development kit available that fits within the project budget that provides the necessary functionality to develop and test both the software and hardware capabilities of the MCU before the final hardware is constructed. A trade study that considered these factors was done to select the MCU for the project.

3.6.2. MCU Trade Study

The following MCUs were considered for use in the ThinSat:

- ATmega328

- ATSAMD21G18
- AT91SAM3X8E
- AM3358

a. Trade Results The MCU trade study resulted in the selection of the AM3358, which has the highest computing power. The advantage of the AM3358's computing power is that it is capable of running a Linux kernel; something none of the other MCUs are capable of doing. This is a significant factor that lowers the difficulty of developing flight software because the operating system abstracts from the hardware and thus, allows the use of high-level and diverse programming. This advantage is well worth the increased power requirement that comes with the AM3358. Furthermore, this MCU is included in the Octavo Systems OSD3358 C-Sip (Complete System-in-Package) surface mount device, which integrates the MCU with many of the necessary components required to have a functioning embedded computer system. These components include DDR3 memory, power management, EEPROM, eMMC non-volatile storage, MEMS oscillator, and other passive components. Choosing this surface mount device for our payload board helps lower the number of discrete subsystems that need to be assembled and reduces the complexity of the team's custom PCB design.

3.6.3. Payload Board Subsystems

The payload board houses many subsystems that work together with the MCU to form TOMCAT's embedded computer system, which is designed to meet the functional requirement set by the team that the payload board shall perform all necessary hardware and software tasks for system operation. The major subsystems discussed in this paper are the IMU, power management, USB circuity, non-volatile storage, and debug circuitry.

i. Inertial Measurement Unit (IMU) The MCU interfaces with the IMU via an I^2C connection. The I^2C pins of the IMU are specified in its datasheet (pins 23 and 24) and are connected to the I^2C clock and data I/O of the MCU, as well as the 3.3V and 1.8V outputs of the power management system. Furthermore, pull-up resistors were included in the design to preserve the integrity of the signals traveling across the I^2C lines with both the IMU device and power management system connected to them. These components are shown in the schematic in Figure 16.

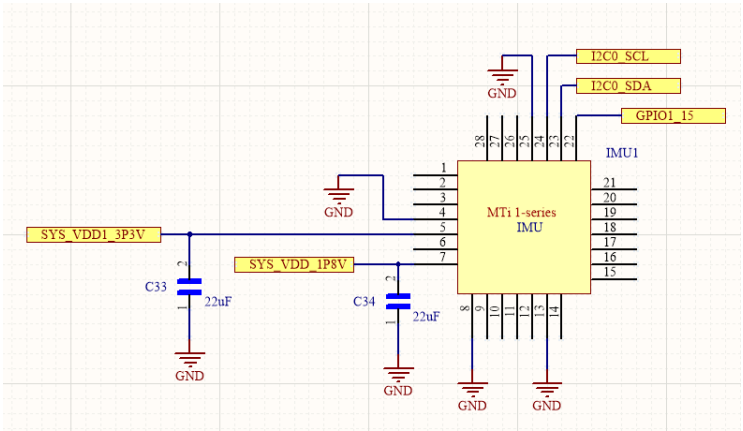


Figure 16. IMU Schematic in Altium Designer

The IMU is an Xsens MTi-3 Attitude Heading and Reference System, or AHRS. It acquires a variety of data including temperature, three-dimensional acceleration (free acceleration, high-rate acceleration, and delta-v), three-dimensional orientation (quaternion, euler angles, and the rotation matrix), three-dimensional angular velocity (rate of turn, delta-q, and high-rate of turn), and three-dimensional magnetic field. This sensor was selected for its expansive data set, ease of integration/configuration, and manageable mass and power requirements. Additionally, the sensor's nominal settings output data at a rate of 800 Hz, far above the team's requirement of 1 Hz. The data communicated to the signal processing software

and SDRs can be customized for a variety of missions and provides the team's sponsor with flexibility for future applications. As stated earlier, only acceleration, free acceleration, rotation speeds, and temperature will be communicated for TOMCAT's testing in order to reduce strain on the flight software and communication systems.

ii. Power Management Electronics The MCU is connected to a power management integrated circuit (TPS65217C) inside of the OSD3358 C-Sip via an I^2C interface. This produces the 5V, 3V, and 1.8V outputs needed by the various subsystems inside the ThinSat and meets the power distribution requirements for the project. Most of the functionality of this integrated circuit can be controlled programmatically through the I^2C connections, such as the internal buck and boost converter output voltage, linear regulator output voltage, power up/down sequences, and over current thresholds. In addition to the I^2C connections, more signal lines need to be connected between the MCU and power management so that the MCU can be used to regulate power. The team also added mechanical power and reset buttons to physically control the power on/off and resetting of the board through the power management integrated circuit.

iii. USB Circuitry This design includes two USB connectors; one for interfacing with a computer and another for interfacing with the SDR. These USB connections follow the same standard for all USB A connectors and fuses were added on the power line to protect the board from any power spike. It was necessary to design one of the USB connectors as a host port to allow a computer to supply power and transfer data the the MCU. The other USB port was designed in an on-the-go configuration, where the state could change between a host and client port depending on the data passing through the USB connector. Configuring the USB ports was done by following the USB configuration table below (Fig. 17), supplied by Octavo Systems in their design documentation for the OSD3358.

	Configuration		
	Through Hardware		Through Software*
USB Host	Ground USBx_ID	OR	Set IDDIG bit of USBxMODE register to 0
USB Client	Leave USBx_ID floating		Set IDDIG bit of USBxMODE register to 1
USB OTG	USBx_ID is directly controlled by USB Cable Connector		Set IDDIG bit to either 0 or 1 based on mode requirement

Figure 17. USB Configuration Table [1]

iv. Non-Volatile Storage KubOS is built on a Linux kernel, so in order for the PCB to be capable of running this operating system, non-volatile storage such as a micro-SD card slot and eMMC is required and included in the board design. Luckily, the OSD3358 SMD has a 4GB eMMC internally integrated, so an external eMMC does not need to be mounted to the PCB. This leaves the team with the need to design an external micro-SD card connection, which was done by utilizing the datasheets of both the micro-SD card part and the OSD3358, and by following good electrical engineering design practices, such as including decoupling capacitors in parallel to reduce the mounting inductance of the part. With both the eMMC and micro-SD memory components, KubOS is able to be flashed to the embedded computer system's memory and used as it's operating system.

v. Debug Circuitry The embedded computer system on the payload board must be designed with debugging mechanisms to assist with bringing up the board and troubleshooting any problems that may arise. Debugging circuitry on the board includes debug LEDs, a JTAG header, multiple probing points, and jumper pins. The debug LEDs are connected to various signal lines on the OSD3358 to indicate if the MCU hardware is following the expected behavior and probing points are placed on signal and power lines where expected voltages and currents are known, so that measurements may be taken with a digital multi-meter to compare the measured values with what is expected. Furthermore, a JTAG header is included because JTAG is an industry standard for verifying designs and testing PCBs after manufacturing and assembly. This

JTAG header is a common hardware interface that provides a PC with a way to communicate directly with the MCU and other SMDs on the payload board. The design also includes jumper pins, which allow signals like power and reset to be easily set by hand.

3.6.4. PCB Design

In order to conform to the unique shape and size of the ThinSat frame, a custom PCB needed to be designed by the team. The custom ThinSat payload PCB is a 4-layer board that integrates many electrical subsystems and was designed through the use of open source reference schematics and component datasheets. The main objectives in the design of the PCB were to design for connectivity, performance, manufacturing, assembly, bring-up, troubleshoot, and debugging. All of these subsystems work together to satisfy the following functional requirement:

FR4: The payload board shall perform all necessary hardware tasks for system operation.

The size and shape of the payload board is constrained by the payload space specified in the Thinsat design. The frame is split down the middle, dividing it into the payload area and an area for support hardware that comes with the frame. A CAD model of the frame was provided to the team by NearSpace Launch, which allowed for the shape of the payload board to be modeled. The outline of the payload board was chosen to take up as much of the payload area as possible to allow for the maximum amount of room on the board for the components. The outline of the board can be seen in Figure 18. The overall outline is the shape of the payload area, except for an area on the same end of the board as the text that is taken up by a connector for the hardware in the other half of the ThinSat. The semicircular indents are there to make room for the bolts that will hold the PCB to the plate at the bottom of the payload area. The notch at the same end of the board as the MCU is there to give space for some connectors to the PCB.

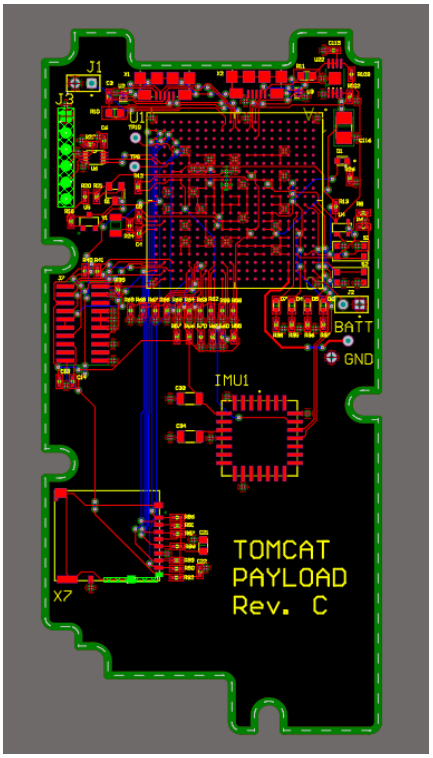


Figure 18. Payload PCB Layout in Altium Designer

The PCB board layout design, created in Altium, fulfills the connectivity requirement, where all of the connections between components specified in the schematics have copper traces making these connections. Furthermore, the copper trace widths are varied so that 6 mil traces are used for signals and 10 mil traces are used for power and ground connections. A lab experiment was done to test how much current a copper trace could handle before it began smoking and the results were that a 6 mil trace could handle 1.2 A of current and a 10 mil trace could handle over 5 A. The signal currents will remain well below half of this 1.2 A limit for a 6 mil trace and the power currents will also remain below half of the 5 A limit for a 10 mil trace. The layout design was also created with the goal of maintaining signal integrity. This was done by following good electrical engineering design practices such as keeping traces as short as possible, using large power and ground planes, and routing the traces to keep mutual loop inductance at a minimum. Mutual loop inductance is where the changing current in one copper trace induces a magnetic field, which then induces a voltage in adjacent traces and therefore creates unwanted current in the adjacent traces. Avoiding large current loops in the layout, as well as routing traces on the top layer of the board in opposite directions from the traces in the bottom of the board are some of the strategies used to mitigate mutual loop induction and keep the PCB signal noise at a minimum.

3.7. Power Systems

Jake Schroeder

3.7.1. Conceptual Design Alternatives

For a balloon satellite mission, the most important aspects of a battery considered were operational temperature, physical dimensions, power output, energy density, and feasibility in a vacuum environment. For optimal performance in high-altitude conditions, three types of battery chemistry were considered: Alkaline, Lithium Polymer, and Lithium Ion.

Alkaline batteries offer high density storage and performance at low temperatures, but are not rechargeable, making characterization of the batteries more difficult and expensive. Lithium Polymer batteries are similar in cost and performance to lithium ion batteries, but offer slightly less performance at lower temperatures and have less extensive research behind them compared to lithium ion. Therefore, Lithium Ion batteries were chosen by the TOMCAT team because of their large power capacity, their wide availability in a number of sizes from reputable sources shipping from within the United States, and the large body of research that gave the team confidence in their applicability and usefulness in a high altitude balloon flight. For more detail on the trade studies performed, see the Power Systems Appendix.

Pros	Cons
Rechargeable	Cylindrical Volume
High Energy Density	Less prone to external pressure damage
Large Body of Research	

Table 11. Lithium Ion Battery Pros and Cons

3.7.2. Battery Specifications

The Power Budget for the TOMCAT payload is shown below, and acted as a starting point from which to determine the required battery specifications and to model the power draw and thermal considerations for dissipated heat.

Component	Voltage Req. [V]	Max Current Draw [A]	Max Power Draw [W]	Energy Required (2hrs) [Wh]
XSens MTi-1 IMU	3V (3.3V Pin)	0.015	.045 W	0.09
	Science Total:	0.015	0.045	0.09
SDR	5	0.9	4.5	9
LNA	5	0.152	0.76	1.52
Power Amp	3	0.05	0.15	0.3
	Comms Total:	1.102	5.41	10.82
MCU	5	0.5	2.5	5
	FSW/OBC Total:	0.5	2.5	5
	Total:	1.617	7.955	15.91
	+50% Margin:	2.4255	11.9325	23.865

Table 12. ThinSat Power Budget

The most important factor in choosing a battery for the TOMCAT ThinSat was its physical volume such that it would fit inside the ThinSat payload, its charge capacity, voltage, and current output, even at low temperatures. These specifications are driven by **Functional Requirement 3**, which states that the ThinSat must be able to be fully functional throughout its high-altitude balloon flight. In order to calculate the required specifications of the battery, the power requirements of each subsystem at their maximum rated power draw were compiled

Accommodation	Additional Capacity
Low Temperatures	30%
DC/DC Converter Losses	10%
Other Efficiency Losses	10%
Total:	**50%**

Table 13. Calculation of Additional Battery Capacity Accommodations

to provide a picture of the peak demand of the system. The
battery was designed to these requirements in order to ensure performance even in the worst-case power draw scenario. Given that FR 3 requires the battery to power all systems operating at full capacity even in cold temperatures and with other sources of energy dissipation, such as power conversion losses and other unknowns, a margin of capacity is built-in to the battery. Because low temperatures have a known effect on lithium ion battery capacity, an additional 30% is built in to combat this. Losses due to power conversion are up to 10% and an additional 10% margin is built-in for any unaccounted losses or power draw.

3.7.3. Power System Design

Because the payload volume of the ThinSat is so small, considerable size constraints for the battery had to be taken into account. The lithium ion 18500 cell is a 3.7V, 2040 mAh battery with dimensions that allow for four batteries to be placed side by side within the payload.

The total size of the battery is of 7.5cm x 5cm x 2cm at 155g. It was chosen for its very high energy density compared to other lithium ion batteries and for its shape, which allows for it to be placed snugly within the payload. Using four batteries, the maximum rated current draw is 3A and the nominal capacity is 30.19 Wh, which is significantly above the 23.86 Wh required capacity for the two hour flight. With the batteries in a 2s2p configuration, stepping the voltage down from 7.4V is required instead of stepping the voltage up, allowing for improved power conversion efficiency and less current demand from the battery.
The power system used 22 AWG wires for all powered connections because of their small size and ability to carry over 3A. JST-RCY style connectors were also selected as the connector of choice because of their small 4.0mm width, rating to carry up to 3A, and a temperature range below the required -10°C. For more discussion on design selection of the power system, see the Power Systems Appendix.

Figure 19. Lithium Ion Battery

3.8. Science Payload

Austin Scheck

3.8.1. Conceptual Design

One of the core requirements provided to the team by their sponsor, Raytheon, specified that the satellite needed to acquire some sort of scientific data. As a result, the team conducted a thorough trade study to first determine what kind of scientific sensor would be optimal for the TOMCAT mission, and then another trade study to determine the specific model that would be integrated.

3.8.2. Trade Studies & Design Selection - Sensor Type

The first objective the team focused on was selecting a type of scientific sensor. The following alternatives were compared:

- Imaging Sensor/Camera
- Magnetometer

- Inertial Measurement Unit (IMU)
- Environmental Sensor Suite
- Radiation Sensor/Dosimeter

Each of these alternatives were compared upon the following criteria: power, size, survivability, cost, documentation, scientific merit, and data processing requirements. The greatest weighting was given to size, survivability, and documentation, as these qualities are most essential to mission success. The TOMCAT system is constrained on size more than anything else, having to fit in two-sevenths of a 1U cubesat, so that received the highest weighting of 0.17. Additionally, thermal survivability was critical, as the system would be exposed to near-space temperatures in the regime of -30 to -50°C. Documentation was also key because the team needed a component which could be easily configured to the unique TOMCAT system - having ample documentation on hardware, software, and electronic integration would be critical.

The lowest weighting was given to cost and scientific merit, as the team had ample funding to subsidize a more expensive component, and there was no pressure from the sponsor to acquire any sort of meaningful or cutting-edge scientific data.

Ultimately, the inertial measurement unit was selected due to its marginal size, impressive survivability, and thorough documentation. In addition, the IMU had the capacity to provide meaningful data to both an atmospheric balloon flight mission and one in the vacuum of space. Since Raytheon's ultimate goal involved launching a TOMCAT flight system into low-Earth orbit, this was classified as a necessity by the team. The full tables comprising the metrics and actual trade study can be found in the Science Appendix.

3.8.3. Trade Studies & Design Selection - Sensor Model

Once the IMU had been selected and accepted by all subsystems, a secondary trade study was conducted to determine which IMU would be most optimal for this mission.

- SparkFun IMU Breakout - MPU-9250
- SparkFun 9DoF IMU Breakout - ICM-20948
- Xsens MTi 1-Series

After analysis, which is further detailed in the Appendix, the MTi-1 Series, specifically the MTi-3 AHRS was selected.

3.8.4. Design Specifications

The MTi-3 model possessed the ability to take free acceleration data, which removes the effects of gravity, and a Kalman-filtering algorithm which refined measurements to an extent unparalleled by competing models. The following table provides its key parameters. Note that it outputs data at a rate of 800 Hz, far greater than required, has a mass of only 0.6 grams, a marginal power consumption of 100 mW, and slight dimensions at 12.1 x 12.1 x 2.6 mm.

Once all the operating specifications of the IMU were known, the team could plan for powering it, accommodating it in the ThinSat structure, and interfacing with it in an I^2C connection. For more specifications regarding its pin layout, data types, data communication, and physical design details, consult the Science Appendix.

3.9. Structures

Matthew McCallum

Interface		Min	Typ	Max	Unit
Size	Width/Length	12.0	12.1	12.2	mm
	Height	2.5	2.6	2.7	mm
Weight			0.6		gram
Temperature	Operating temperature	-40		+85	°C
Power consumption				100	mW
Timing accuracy			10^9		ppm
MTBF		225,000			hours
Output data rate				800	Hz

Figure 20. IMU Specifications

3.9.1. Structural Design

Figure 21 shows an external rendering of the CAD model of the full ThinSat. The ThinSat is 111.1mm by 114.2mm with heights of 45.21mm and 36.29mm with and without the gateway launch adapter respectively. The primary frame is the frame that was completely designed by NearSpace Launch Inc and is the lower frame in Figure 21. The secondary frame is the upper frame in Figure 21 and was created by TOMCAT by modifying the NearSpace frame. (The primary frame may also be referred to as the ThinSat kit frame/slice, and the secondary frame may also be referred to as the TOMCAT frame/slice.)

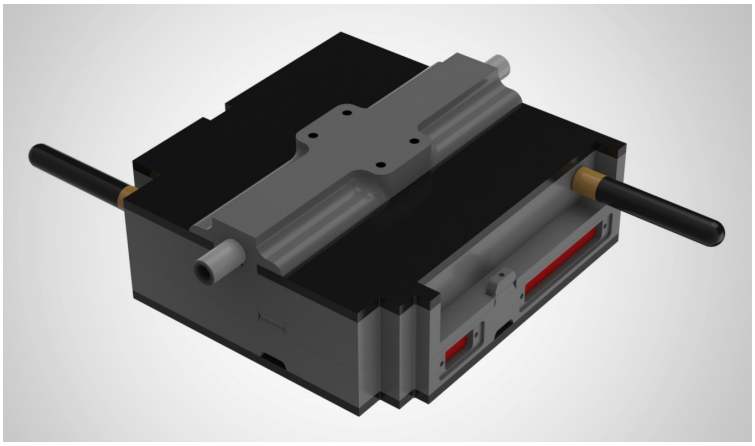


Figure 21. Rendering of the ThinSat model

There are three main modifications that were made to the secondary frame to make it fit our needs. The first modification was to extend the walls downwards towards the primary frame. The NearSpace design allows for a space between the ThinSats when they are stacked, which would have left a gap around the middle of the ThinSat if we did not make this modification. Removing the gap makes the ThinSat look like one unit as well as limit air circulation in and out of the ThinSat. The second modification was to create an indent on one side of the frame. The ultimate goal of this ThinSat is to be launched into space. During launch, the antennae cannot remain straight out like they are in Figure 21. There is internal space where the antenna on the left side could be retracted into, but there is no internal room for the antenna on the right to retract into. Therefore, the indent was added to give the antenna a place that it could be folded into. A note here is that we did not design mechanisms to perform the stowage and deployment of the antenna, but we left space for it to be possible to add this functionality. The third modification is the addition of the mounting adapter for the gateway to space balloon launch. The payloads are attached to the flight string of the balloon flight by threading the flight string through all of the payloads. Each payload must have a plastic tube that the string can be passed through, and the plastic tube must have a hole that a paperclip can be passed through in order to hold on to a knot in the flight string that keeps the payloads separated. This mounting adapter will only be on the balloon flight version of the ThinSat and will not be present on the ThinSat version that is sent into space.

Figure 22 is an exploded view of the ThinSat model which shows all of the internal components of the

ThinSat. This image shows the four other structures components in the ThinSat, the two covers and the two backplates. The two covers are the black panels on the left and right of Figure 22. These panels are made of acrylic and follow the outline of the frame where they are attached. They serve to seal the ThinSat, containing all of the internal components and restricting airflow in and out of the ThinSat. The top cover is broken into two pieces in Figure 22 in order to accommodate the mounting adapter for the balloon flight. This cover would be a single piece on the ThinSat that would be launched into space. The two backplates serve two main purposes. The first is to help attach the internal components to the frame. This is done by adding holes to the plates that the components can bolt to. The second is to act as a ground for the electronic components. The backplates are made out of aluminum and are the largest pieces of metal in the ThinSat that will go on the balloon flight. The version of the ThinSat that will be launched into space will have the frames made from aluminum, so there will be a better ground for that version.

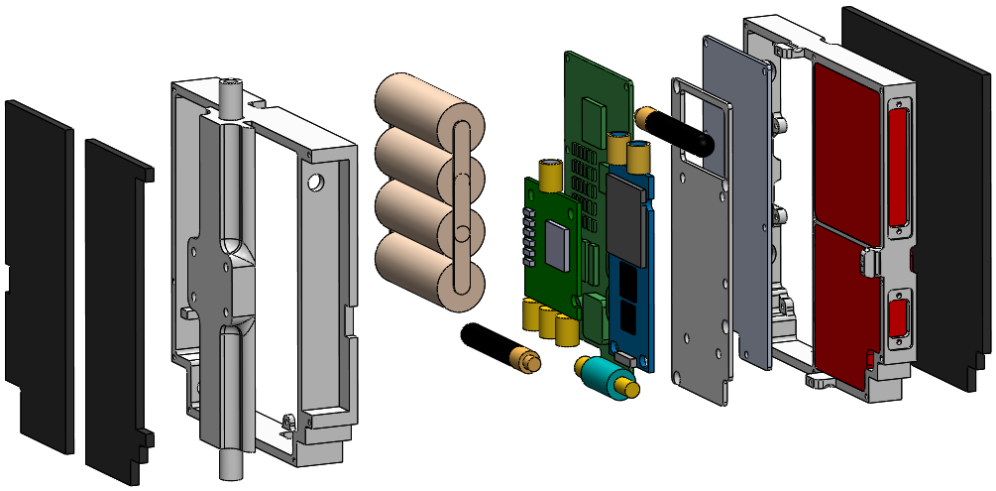


Figure 22. Exploded view of the ThinSat model

3.9.2. Trade Study

There were no trade studies performed related to the structure of the ThinSat. Our customer specified that we obtain a frame from NearSpace, which locked us into using 3D printed PLA as the frame material. Obtaining the frame from NearSpace also locked us into the shape of the ThinSat. NearSpace designed their frames to be able to stack together, so even when we added the second frame to our design we needed to conform to their design. However, if the project did not end early, a structural trade study would have needed to be performed. The drop test was performed on the last day of work on the project, and this test revealed that the ThinSat would require additional external padding in order to survive the impact with the ground at the end of the balloon flight. Therefore, a trade study would have been performed to determine what material and/or design to use for the padding.

3.10. Thermal Design

Lara Buri

Because the end goal of the TOMCAT mission was to send the ThinSat and payload on a high altitude balloon to 30km, the cold environment at this altitude needed to be accounted for. It was expected that the ThinSat would require some type of thermal control to maintain adequate temperatures inside the structure to ensure the battery and electronic components operate properly. Heritage data from Near Space Launch includes temperature data taken during high-altitude balloon flights for the same ThinSat frame design as TOMCAT. This data suggests that with no thermal protection or control, temperatures inside the ThinSat

reach around −40 degC. These temperatures are much too cold for a battery to maintain optimal operation and reach the threshold at which the electronic components will get too cold for operation.

3.10.1. Thermal Control System

The first option to maintain temperatures within the ThinSat and prevent the ThinSat from becoming too cold is to add a thermal control system. This would include a heater and a feedback control loop that would allow the heater to turn on if the ThinSat becomes too cold, and turn of if it becomes too warm. The following table presents the pros and cons of using an active thermal control system.

Pros	Cons
Allows for control of internal temperature	Complicated to model
Lowers risk of failure due to low temperatures	Requires more battery power
	Requires implementation of control law on OBC

3.10.2. Thermal Protection System

The second option to maintain temperatures within the ThinSat is to add thermal protection, or insulation. This option is a much simpler solution to keep the ThinSat electronics and battery warm and is more cost-effective in terms of both monetary cost and power. The following table presents the pros and cons of using a thermal protection system.

Pros	Cons
Easier to model	Little control of internal temperature
Does not use any power	Will change dimensions of ThinSat
Lower cost option	

3.10.3. Heat Removal System

After choosing the thermal system and building physical models of the ThinSat to test temperatures within the structure, it was determined that with insulation, cold temperatures were no longer as much of a concern as heat. The SDR and the payload computer had high heat concentrations that began to reach the upper bound of the component temperature limits. While the insulation provided needed protection to the ThinSat, the heat from the electronic components needed to be moved away from those components in order to prevent damage. It was determined that heat could easily be transferred using the concept of conduction. Research was done on heat pipes, which transfer heat using conduction coupled with phase transition of a liquid to a gas, and simply adding a conductive material like copper to transfer heat.

3.10.4. Thermal System Design

Due to a limited power budget and the fact that Near Space Launch has flown ThinSats with no thermal control or protection, it was decided the TOMCAT mission would use thermal protection (insulation) to maintain internal temperatures at the required levels. This is the simplest solution for thermal design and does not place more constraints on battery selection and complications on the payload computer design or software design. Figure 32, found in the Structures section of this paper, shows the exploded CAD view of the TOMCAT ThinSat with the insulation included on the front and back faces.

To reduce cost and complexity, a simple copper strip design was selected to transfer heat from the electronic components at risk of overheating. Adding this copper strip prevents the SDR from overheating and allows the team to find a simple solution to some components overheating while still keeping insulation to maintain internal temperatures, all to meet requirement 3. The following images show how the copper

strip would be attached to the SDR, the primary concern for heat build-up. The strip would be placed on top of the heat sink of the SDR and then wound around to the aluminum plate that the SDR sits on. Both the copper and the aluminum plate are good conductors, and would effectively remove heat build-up on the SDR.

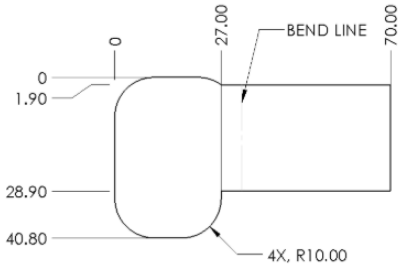


Figure 23. Copper Strip Drawing

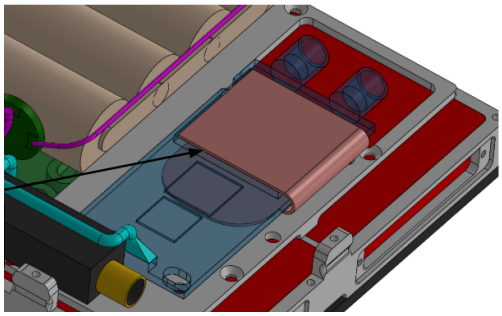


Figure 24. Placement of Copper Strip

4. Manufacturing

4.1. Overall Manufacturing Status

Cole

The manufacturing status of the project is described below in detail.

Overall, Nearly all Communications components were assembled on the flight side (with the exception being modifying the SDR to fit within the ThinSat frame). The ground station hardware was ready to be assembled and theoretically only required components to be plugged into the RF chain. The power components were assembled and pending final integration with the Payload Board, which contained the power regulation and distribution hardware. The board was delivered to the team shortly after the work hold, so it's assembly has not been verified and it still required manual assembly of many of the passive electrical components and ICs. The ThinSat structure itself had been manufactured for testing and was pending design revisions after the results of the drop test. Since the frame was 3D printed, manufacturing lead time for the ThinSat structure is low.

Software for the project was nearing the integration state, with all parts of the flight software nearing completion individually and aspects of the ground software only waiting on flight components to conduct final tests.

4.2. Communications

Trevor Weschler

The communications subsystem is comprised of four major sub-components with varying manufacturing needs. The RF chain is mostly assembled on-sight with a small need to manufacture power wires for one component. The communications PCB is printed by an external company but the components are soldered on by the team. The onboard SDR needs to be modified by removing the USB port and directly connecting to the SDR PCB. The SDR software was built using the GNU radio platform.

4.2.1. RF Chain

The major RF chain includes three low noise amplifiers, three power amplifiers, three low pass filters, three Lime-SDRS, two yagi-antennas, and one monopole antenna. All manufacturing for these components took

place in the Aerospace Electronics shop. The manufacturing requirement is to wire the components together. A particular challenge was the ground low noise amplifiers. These came on an evaluation board that had a unique power set up.

The low noise amplifier evaluation board required power through and SMA port. In order to power this board, a custom wiring solution was required. An adapter was order to convert a standard wire into SMA. This adapter was then soldered onto a wire from the Aerospace Electronics Shop. This allowed the low noise amplifiers to be successfully powered for testing and eventual integration into the TOMCAT Power System.

Additional manufacturing tasks included soldering on power leads to the power amplifiers and assembling the wire connection system. Wires from the electronics shop were soldered onto the power amplifiers to allow for them to powered for testing and integration with the power system. The chain was completed successfully by attaching all the components in the order specified in the FBD.

4.2.2. Communications Printed Circuit Board

The communications PCB was designed by the TOMCAT team and printed by PCBway and Oshpark. PCBway printed the first revision and Oshpark printed the second. The TOMCAT team was then responsible for soldering all of our components onto the board. This included resistors, capacitors, inductors, and integrated circuits. The team ended up needing to order two revisions of the PCB since the first one had a critical error.

The first version had the wrong chip set for our integrated circuits. Though a work around was possible for the power amplifier, it was impossible to properly attach the low noise amplifier to the board. The team returned to the design and redesigned the chip sets to be the proper size for both components. Revision B faced delays due to the pandemic crisis but did arrive in time to be manufactured. Team members spent several hours soldering two copies of the board to ensure redundancy as testing progressed on the board. Figure 25 shows one of these completed boards before it entered testing.

However, with revision B, there were issues encountered with the power amplifier that were unrelated to the chip size. It appeared that the amplifier was soldered on backwards for one copy which causing unexpected changes in the signal level through the amplifier. The other copy had an low noise amplifier that did not amplify. Project progress was halted before the power amplifier could be re-soldered in the correct direction. Had this step been completed, the communications PCB would have been complete.

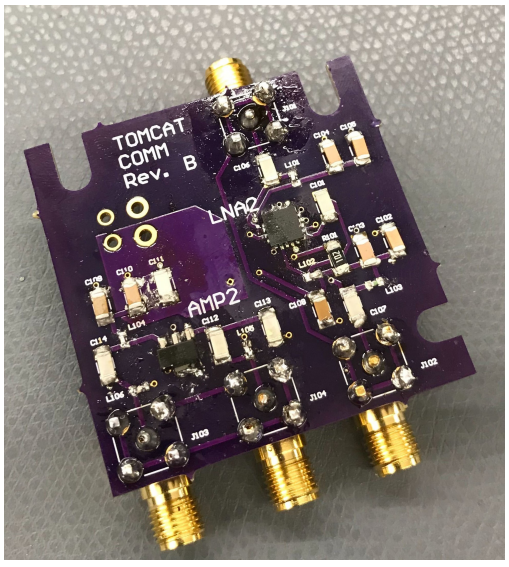


Figure 25. Revision 2 of the Communications PCB

4.2.3. Software Defined Radio

One of the three SDRs had to be modified to fit on-board the ThinSat. The choice for the modified radio was between removing the SMA ports or the USB ports. Given our team's limited knowledge of RF systems, it was decided that removing the SMA ports was more risky. This could cause more noise to be introduced into the system if done improperly. Since the team lacked a breadth of knowledge in RF systems, there was a high risk that SMA port removal and replacement with only wires would be done improperly. Any issue arising would be difficult to troubleshoot and fix. Therefore, the choice was made to remove the USB since the team had considerably more knowledge of that type of connection.

Removal of the USB was done by de-soldering the USB using a solder wick. Once enough solder was removed, the USB port was carefully separated from the board. With the pads exposed, the team was able to reference the SDR electrical diagrams to determine the layout of the USB pads. Wires from the electronics lab were then soldered onto the pads to service the various connections a USB has. This approach met limited success. The power was able to reach the SDR and turn it on. However, RF communication was impossible. In order to troubleshoot, the team soldered similar wires onto the appropriate USB pads of an unmodified radio. This allowed us to test the functionality of our wiring scheme against the working USB port. It was found that while the USB worked, the wiring scheme did not. Clearly there was not short circuit. The team decided the most likely cause was that the wires were not USB standard wires. This difference causes the USB data signal to decay too quickly and not reach the destination with enough power to be decoded. If further time was available, the next manufacturing task would have been to use USB standard wires instead of electronics shop wires to work the onboard SDR.

4.2.4. GNU Radio

The communications software was manufactured use the GNU Radio platform. GNU Radio is similar to simulink in that it allows blocks to be dragged to a flow diagram to create a signal processing model for the SDR being used. The individual parameters within these blocks were design dependent. The TOMCAT team software manufacturing process included the correct set up of the signal processing model and then setting the individual values relevant to our system. Design of the model was completed last semester and this semester served as a way to ensure our model was correct.

Significant time was spent tuning parameters within the model as part of the final manufacturing process. Parameters were tweaked during wired tests to ensure that the signal processing model was able to perform at the level required by our link budget. These parameters included bandwidth, power gain, and modulation parameters among others. The signal processing model was successfully manufactured by the time of ending project progress.

4.2.5. Communications Integration

Integration for the communications subsystem followed the test plan for the subsystem describe later in the report. Here, we will walk through this tests as stages in integration and leave the finer details of the test to last sections.

The first integration step was accomplished during the wired testing portion of the subsystem. Here, two SDRs were wired together with the full RF chain between them. Attenuation was used to ensure the receiving SDR was not overpowered. This was accomplished by connecting all the RF chain components in a line with the appropriate wires with power for amplifiers coming from the lab bench power supply. The integration achieved during this test is the completion of the RF chain as one functioning unit. This was almost completely accomplished before project progress was halted. The issues discussed above with the communications PCB prevented the completion of this step.

The second integration step was going to be accomplished during short and long range testing. At this stage, the ground communications system needed to be powered by the TOMCAT power system since range testing does not facilitate access to lab bench power supplies. Upon completion of this step, the power and communication systems would be integrated on the ground side. The flight side would have to wait until another step in the testing process. It is worth noting at this point that although short range testing was not able to be completed, the team was able to test out integration of the ground power subsystem components with the communications system. This yielded positive results for powering the RF chain with our laptops and USB breakouts though no official data was recorded.

The final step of integration is the full systems test. Here, the ground system remains the same, but the on-board system in integrated into the ThinSat and powered by the flight PCB. Numerous size measurements were made to ensure the on-board SDR and communications PCB would be able to fit into the ThinSat in

the allowable space. The connections to the flight PCB would be done via USB for the SDR and jumper wires for the communications PCB. Connections of these systems would occur before the test to ensure that power is received and the system is operational before doing a full long range test.

4.3. Flight Software

Manuel Lindo

Manufacturing Scope

The flight software development tasks can not be exactly divided between manufactured or purchased. However, these will sit in a spectrum, where one end holds tasks that were completely designed by our software team, and the other is tasks that use a pre-built KubOS software component. With this is mind, there were the following tasks:

Purchased (Assembled by Team)	Manufactured by Team
KubOS Telemetry Database	IMU Telemetry Collection App
OBC Housekeeping App	Health App
Inter-application Query Services	Communications Service

Table 14. Manufactured vs Purchased aspects of Flight Software

It is important to keep in mind KubOS is an open source software platform that comes with inclusive services built by its community of software developers, as well as guidelines and frameworks for custom applications/services. Therefore, the team only made minor compatibility modifications to the KubOS Telemetry Database, OBC Housekeeping App, and the Inter-application query services. The alterations ensured our "manufactured" software components could utilize the former throughout the different development stages. This report will detail the progress, components, and development environment of the "manufactured" software parts.

4.3.1. KubOS Telemetry Database

The parts assembled by the team were inherited from KubOS source code. Initially, the Telemetry database was tested by entering dummy data generate by a random number generator. Afterward, the relational database server, GRaphQL, was used to access it according to subsystem, parameter, and values. For example, a dummy data sample would look like "Subsystem: eps ; Parameter: Voltage ; Value: 5" as can be seen in figure 26. This was done in the development environment of laptops as well as the beaglebone. This was done through command line arguments that take in some dependencies and configurations files, and, consequently uses a Python API that connects our main python script with an accessible database. The database was accessed through a web browser interface when using laptop, and a command line argument when using the beaglebone black.

```
{
    "subsystem": "eps",
    "parameter": "voltage",
    "value": "3.5"
}
```

Figure 26. Telemetry Database example by KubOS [2]

4.3.2. OBC Housekeeping App

The purpose of this app is to take any necessary measures to correct any error or problem during our flight. It was going to be most important tool during communications system testing. Upon the inability to communicate with our flight software either commands or health beacons, this app was going to be used to reset our whole system. The only requirements for this app is that we have a functioning beaglebone. Its functionalities include cleaning the telemetry database, checking the log files, pinging different KubOS services,

as well as performing system reset. The cleaning the telemetry database option was going to be used if our available memory reached a critical threshold, value to be determined, and the health of our system was going to be in jeopardy. The IMU sampling rate was not relatively high for the duration of our flight, but the byproducts of flight, such as log files, might represent an unforeseen danger. Moreover, it is crucial to make sure the different communications, telemetry database, monitor services are functioning during flight. Finally, it is important to use the full system reset, if a critical error is found during flight, to make sure the continuity of the flight software during integration and balloon testing.

4.3.3. Inter-application Query Services

Under this category falls the software that executes the interactions between the applications and the relational database. These usually take the form of APIs - Application programmable interfaces, and they make sure mutations and queries can be exercised within our databases. They will not be altered for our purposes, but will still be used to ensure compatibility between our developed apps and the built-in telemetry database. It is an effective manner to maintain heritage for future Raytheon endeavors or CU Boulder senior designs building upon this project.

4.3.4. IMU Telemetry Collection App

The purpose of this application is to collect IMU acceleration and rotation data, through an I2C connection between the IMU XSens hardware and the Beaglebone Black micro-controller, and store it in our KubOS Telemetry Database. To accomplish this, the software components was split into two languages, C/C++ and Python. The former, a general-purpose and procedural computer programming language, decoded the XSens's proprietary Xbus packet into raw data. Some of the C/C++ methods were inherited from the XSens community of software developers, but the data parsing and processing algorithm was designed and implemented by the team. Data sampling was executed every 0.05 seconds, but the frequency was going to be decrease at a later stage to save database storage. The setup used to development this application was the following figure 27. *Show picture of successful storage in database*

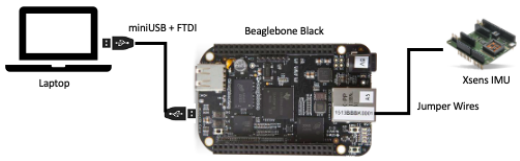


Figure 27. IMU Collection App Development Setup

4.3.5. Health App

This app was not concluded by the COVID-19 deadline, but its components, requirements analysis, and test were set out. The purpose of this application was to downlink health flags throughout our flight depending on the specific requirements. The first health flag to be considered would consist of a confirmation/ping that all KubOS builtin services and custom apps were running correctly. This "ping" would be sent periodically throughout

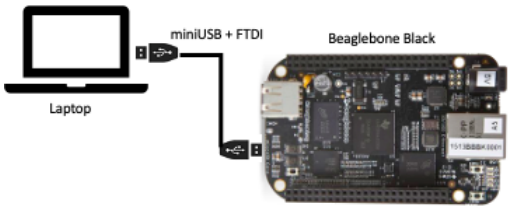


Figure 28. Health App Development Setup

flight, independently of external conditions. Secondly, it would detect improperly formatted commands and send a bit error flag response to ground station, as the request to the database raises a system error. This would prevent runtime errors of waiting indefinitely for a database response and mitigating risk of full system collapse/freeze as a result of error. Ultimately, this application would reduce uncertainty and impact of bit errors in commands received from the ground, allowing the ground team to understand if it's a problem with Flight Software or Signal Processing/Communications System.The setup used to development this application was the following Figure 28.

4.3.6. Communications Service

Lucas Perry The Communications Service is a network of structures, functions, and scripts in the language Rust that held the responsibility of communicating both within the KubOS subsystems, but also between KubOS and external systems. It's role in completing the mission objectives included receiving commands from the ground system in CCSDS format, depacketizing them, and transforming them into queries that were forwarded to the service or application that was being commanded within KubOS. The Comms Service duties also included receiving data collected by the IMU Telemetry Collection App and Health App in a JSON response file, packetizing them in CCSDS format, and forwarding them down to the ground.
A partial framework for the communications service was included in the initial KubOS install, including methods of testing components and the ability to start up the client. However, the majority of the functionality was manufactured by the software team. The manufacturing and testing process for the Communications Service had been partially completed at the time of the COVID-19 cutoff, but the tasks that remained were well-structured out and were anticipated to be completed soon.
The steps in manufacturing the Communications Service included creating a system that satisfy the mission tests. The first of these test is that KubOS could send a UDP packet across a Beaglebone to the communications software. On the flip side, it had to be able receive commands over a UDP socket from the communications software. And finally, the system had to pull health and IMU data from their respective services and send this to the communications software. The first of these tests had been completed, and the other two were partially completed, with a test plan in place. The setup for this service can also be seen in Figure 28.

4.3.7. Integration

IMU Hardware and KubOS database The purpose of this integration was to fulfil the functional requirements 4.2 and 4.3 - "The payload shall generate telemetry" and "The OBC shall store all telemetry during flight. This integration was concluded before the COVID-19 deadline This integration procedure was conducted to ensure IMU data from our scientific hardware was collected, parsed correctly, and easily passed on to other software modules by our KubOS flight application. The process included a laptop, the inertial measurement unit, and the beaglebone black. Before developing the application, an I2C connection was established with jumper wires between the beaglebone and IMU. Afterward, the application development process started and some C/C++ project's modules from XSens (IMU brand) were used in the collection algorithm for heritage purposes. The C/C++ executable read data from the measurement pipe, extracted acceleration and rotation data, and opened a socket to send the data to the python API. Inside the latter, the telemetry dictionary was filled with the data and passed on to other software subsystems.

Communications Service and KubOS database and Health Service The purpose of this integration was to ensure the fulfilment of the function requirement 4.1: "The payload shall be commandable". This integration was not completed by the COVID-19 deadline, but it was in progress and soon to be tackled. The procedure was going to involve integrating the commands received by the signal processing software and retrieving the appropriate data from the KubOS Telemetry database. In more detail, the test would start with a Laptop connected over a wire to a Beaglebone with KubOS and the signal processing software installed.

The Laptop, simulating the ground station, would send a "Collect" command (unfinished at the time of the deadline). This command would be received and depacketized by the Communications Service in KubOS, which would use the CCSDS primary header to figure out what kind of command it was, then send it off to the proper service or application within KubOS. For this test, that would be the IMU service. This command would query a ping and receive a response, which would be sent back down and displayed on the laptop. The criteria of this integration test would be firstly receiving the command within KubOS and the response on the "ground station", but also to verify the bit precision that the downlink and uplink had. The integration between the Communications Service and the Health Service would detect improperly formatted commands and send a flag response to ground station, showing the database request raised a system error. Finally, we would look for full system errors resulting from the commands sent. These would indicate that the Flight Software did not have the level of commandability required yet. The system would first be rebooted to see if the issue resolved, but if not, it would be picked apart to fix the error.

Full System Integration This integration will bring together the integration between IMU Hardware and KubOS database, as well as between the Communications Service and KubOS database and Health service to approximate a full flight condition from a flight software point of view. It will make sure all of the functionalities of the former integration can be done while an active IMU hardware samples euler angles with appropriate timestamps. Moreover, it would have been integrated before the communications hardware and signal processing software were tested with our flight software.

4.4. Ground Station Software

Lucas Perry

The Ground Station Software was split into a few parts in terms of manufacturing. The solid open source framework for COSMOS was set in place when installed, coming with useful built-in applications and documentation. This said, the manufacturing part came in the form of writing the configuration files of each COSMOS tool that was being used for the mission, making a key of bit numbers for the CCSDS packet header, creating commands, and planning integration testing. Because the configuration file writing took the vast majority of the work, and encompasses most of the other manufacturing items, this was focused on for the purposes of describing the manufacturing process. The divide in manufacturing vs purchased parts can be seen below in Table 15.

Purchased (Assembled by Team)	Manufactured by Team
COSMOS Command & Telemetry Server	COSMOS Configuration Files
COSMOS Command Sender	COSMOS Command writing
COSMOS Telemetry Grapher	COSMOS Integration Test planning
COSMOS Telemetry Extractor	COSMOS CCSDS Packet Header formatted

Table 15. Manufactured vs Purchased aspects of Ground Station Software

COSMOS Configuration Files

Written in Ruby, the COSMOS configuration files took the majority of the work when manufacturing the Ground Station Software. This was done by reading the documentation for Ruby and the documentation on COSMOS's website. This was helpful because it showed a lot of the different ways tools could be configured, but not how to really implement it in the tools' file directories, or how to test. Writing the configuration files for each tool took a lot of experimentation, but the end result was a program that was tailored well to satisfy all our mission objectives. The configuration files were mostly broken into three categories: target, data, and tool configuration.

Target Configuration Target configuration included creating a virtual target system (that would later become the on-board Beaglebone Black board) to connect to in the Command & Telemetry Server. This was the bulkiest of the configuration tasks as it required setting up the target system's interface, possible commands, connection status, and what tools can be used on it, among other things.

Data Configuration The data configuration included customizing the commands and telemetry packets going in and out of COSMOS to do so in CCSDS format. Included in here was a key for what the CCSDS Packet Header should look like, and how many bits each section is. Our packet header created in the configuration file is shown below in Figure 29.

```
PARAMETER    CCSDSVER         0   3  UINT  0     0   0 "CCSDS primary header version number"
PARAMETER    CCSDSTYPE        3   1  UINT  1     1   1 "CCSDS primary header packet type"
PARAMETER    CCSDSSHF         4   1  UINT  0     0   0 "CCSDS primary header secondary header flag"
ID_PARAMETER CCSDSAPID        5  11  UINT  0  2047 999 "CCSDS primary header application id"
PARAMETER    CCSDSSEQFLAGS   16   2  UINT  3     3   3 "CCSDS primary header sequence flags"
PARAMETER    CCSDSSEQCNT     18  14  UINT  0 16383   0 "CCSDS primary header sequence count"
  OVERFLOW TRUNCATE
PARAMETER    CCSDSLENGTH     32  16  UINT  MIN MAX  12 "CCSDS primary header packet length"
ID_PARAMETER PKTID           48  16  UINT  MIN MAX  <%= id %> "Packet id"
```

Figure 29. Ground Software CCSDS configuration

Shown in this figure is the breakdown of the packet header before data is entered. This way, when a packet is uplinked or downlinked through COSMOS, it would know what is contained, where it is going, and what to do with it.

Tool Configuration The tool configuration was where the tools had to be told specifically what to do. This is where the commands for collecting telemetry and health data from the Thinsat was placed, as well as how the data should be plotted and stored. Much of the information required here was found by combing through the long documentation archives on COSMOS's website, and applying the functions and different options for tool settings to COSMOS on the ground station computer.

4.4.1. Integration

When the configuration files for all the different subsystems of COSMOS were in place, the integration testing was the next step. Unfortunately, only some of this testing was able to be completed at the time when the project ended. The system was able to send and receive packets of CCSDS data over the proper UDP socket, as well as store the data correctly. What was not tested yet was interacting with the specific Beaglebone Black MCU loaded with the flight software and on-board communications software. In theory, these tests should have proceeded fairly smoothly, with COSMOS being able to send it's collect command to the flight software, collect IMU data, and receive it within a margin of error required for the mission objective.

The Ground Software Integration Test would have been setup using a wired connection between a Laptop with COSMOS installed and a Beaglebone Black with KubOS installed. This test would be solely to confirm the integration of the Ground and Flight Softwares, cutting out the communication software and hardware components. The full system including the comms components would later have been tested in the Full System Integration Test.

Because each of the components of the ground software system were tested thoroughly individually, the integration of each of these would happen simultaneously. First, the connection between COSMOS and KubOS would be confirmed in COSMOS's Command & Telemetry Server interface. This is an important step because it simulates the UDP socket between the COSMOS and the ground station signal processing software that would have taken place during the full system tests. Then the created ping command would be wrapped up in a small CCSDS packet using the format established in the configuration files, and the

bits of data would be sent over to KubOS. The first stage of test verification would be that the command was successfully sent over the socket. COSMOS would show a confirmation that "command(TOMCAT Ping...) sent." The second stage of confirmation would be seeing the response from the Flight Software in the COSMOS data viewer to confirm it can successfully fulfill all mission requirements. As mentioned previously, any further testing on the Ground Software system would be done at the Full System level.

4.5. Payload Board and Electronics

Grant Novota

4.5.1. Manufacturing

The payload board PCB was manufactured by PCBWay. They also did some of the assembly for parts that are difficult to hand solder. These parts include the MCU packaging (OSD355x C-Sip), the micro-USB ports, the micro-SD card reader, IMU socket, and other parts where the soldering pads are smaller than the 0402 size. All other surface mount devices were purchased from Digikey and were to be hand-soldered to the PCB once it arrived.

PCB manufacturing delays were experienced when working on this project and PCBWay was one of the companies that were having difficulty processing orders, so we began to identify other potential vendors for manufacturing the PCB that were not having this problem, such as Oshpark (which was used to build the second communications PCB revision) and Advanced Circuits.

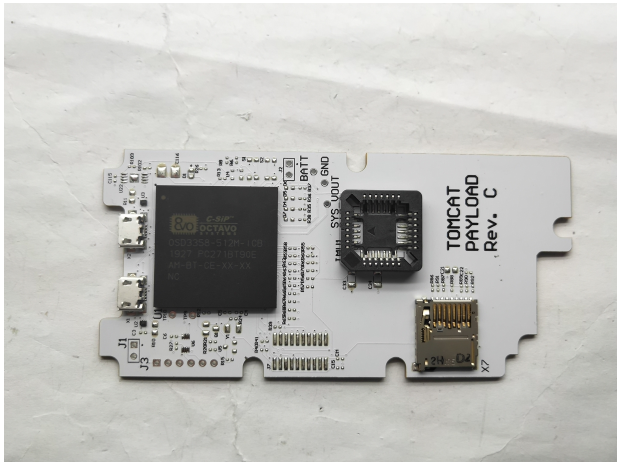


Figure 30. Partially Populated Payload Board PCB

4.5.2. Integration

After completing the board assembly and testing to confirm that the embedded computer system functions as expected outside of the ThinSat, the plan for integrating the payload board was to connect it to the battery via thru-hole connections on the PCB and place the board into the ThinSat frame. More tests would be conducted that are targeted at testing the various power transmission lines in the payload board design.

4.6. Power Systems

Jake Schroeder

4.6.1. Battery

The battery was custom designed by the TOMCAT team and manufactured by BatterySpace in California. It consists of four lithium-ion 18500 cells in 2s2p configuration to provide a nominal 4080mAh at 7.4V. It also contains a manufacturer-provided protection circuit to prevent over-voltage charging, under-voltage discharging, and fuse to prevent over-current above 3A. The battery and protection circuit came assembled and shrink-wrap sealed requiring no additional assembly by the TOMCAT team except to crimp the battery leads and add on the JST-RCY style connectors. JST-RCY style connectors allow for current transmission of up to 3A and were added to all wires connecting payload components to the battery or power distribution

system in the PCB. Each wire is 24 AWG, meaning that 3A can be safely carried by the wires. Adding the connectors was a fairly straightforward process that involved stripping and crimping one end of the positive and ground wires and fixing either the male or female JST-RCY style connector over the crimped ends. To ensure a strong connection during the balloon flight, a small amount of solder would have been added to the crimped wires used in the flight hardware.

4.6.2. Integration

All power management and distribution is performed on the TOMCAT payload board, requiring integration between the battery, payload board, and every powered component. The planned integration of each system with the power system, including all cables and connectors, is shown in Figure 31.

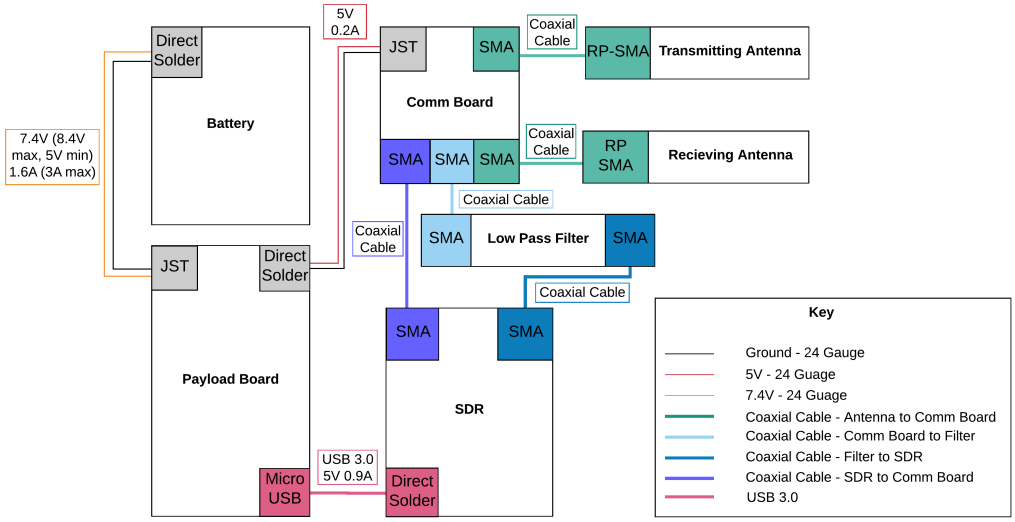


Figure 31. Payload Power System Integration

4.7. Science Payload

Austin Scheck

The strictly science portion of manufacturing consisted of both the actual Xsens IMU sensor and the Python algorithm script that it was to be run on. The manufacturing status of both are given below.

4.7.1. Manufacturing: Xsens IMU Sensor

The Xsens MTi-3 Attitude Heading and Reference System was manufactured by Xsens and acquired through the electronics retail website Mouser. In particular, the team ordered the MTi-3 AHRS Development Kit, which came with the sensor integrated onto an Xsens-proprietary microcontroller board. This development kit provided the team with a physical testing platform to test flight software packages on. The development kit additionally provided access to the MT Development Suite, a software interface which allowed a computer to interact with the IMU and the microcontroller through a USB connection.

Mechanically, there was no manufacturing associated with the IMU. The full development kit included the sensor and a mount for which to integrate it onto the team's payload board. While the provided microcontroller board was not necessary for the final TOMCAT system, it did provide a much-needed testing

platform for the flight software team to experiment running CCSDS packet transfer scripts on, among other things.

Electrically, the IMU had to be integrated into the finalized payload board, which did not arrive before all development was halted. All electrical connections to the IMU were accounted for in the design of the payload board, and thus, there was no electrical manufacturing necessary.

4.7.2. Manufacturing: IMU Mission Code

While the mechanical and electrical manufacturing portions of science were very straightforward, the software aspects were undoubtedly the most involved, if still relatively basic overall. The IMU had to provide data which could be interpreted using a script to determine what phase of flight the satellite was in at any given stage: pre-flight, ascent, descent, or post-flight. This was the basic "mission" that the whole TOMCAT system was supporting, though the true engineering design resided in the communication system.

The code was first developed using MATLAB, and was completely developed from scratch by the science team. Ultimately, the plan was to transfer this script into Python, however project development was halted before this could be completed.

The MATLAB script was developed based upon data provided by previous Gateway to Space flight data. Because the TOMCAT system was to be tested using a Gateway to Space flight, the team decided referencing historic data from nearly-identical flights would be an apt method to characterize the flight and derive the code. The team finished with a functional MATLAB script that could accurately determine which of the four flight phases was occurring based on three-axis acceleration alone.

There were additionally plans to generate an alternative script which evaluated flight phase based upon rotation data in addition to acceleration data, however development was ceased before this activity could be completed.

The team was confident in its ability to successfully port the script into another language, as MATLAB and Python are notably quite similar. The MATLAB script developed by the team can be found in the Science Appendix.

4.7.3. Integration

The Xsens IMU was to be integrated on the payload board, attached using the mount provided in the Xsens development kit. All electrical connections were to be handled by the payload board, which was designed with the IMU's requirements in mind.

The mission code software was to be ran on the microcontroller unit on the payload board, written in Python, which would interface with the physical IMU through the wired connections described earlier. The code was to be one of a multitude of scripts the team ran on the satellite, but was not anticipated to require significant computing power due to its simplicity.

4.8. Structures

Matthew McCallum

The structural components of the ThinSat were mostly manufactured at the aerospace building. We received a 3D printed frame from NearSpace Launch Inc. that we would have used on the balloon flight, but all other structural components were manufactured by us. These components manufactured by us are the frames, acrylic panels, mounting plates, and mass simulants.

The frames were printed from PLA in printers in the aerospace building, using the data from the CAD models, and had additional work done on them in the machine shop. The PLA printing filament was ordered through Amazon. The low cost of printer material allowed us to perform several test prints in order to tune the printer settings to those that produced the best results. Once the prints were coming out well, a few frames were printed to be used for thermal testing and the drop test, while a secondary frame would have

been printed to be used on the balloon flight (along with the primary frame provided by NearSpace). After the frames were printed, they had their support material removed mostly with pliers, however some of the support material required some drilling in the machine shop to break it loose. The frames for the drop test also needed holes drilled through the frames to allow for the bolts that held to two frames together to pass through.

The acrylic panels were laser cut using the ones in the aerospace building. Two 6" x 12" x 1/8" acrylic sheets were ordered from McMaster-Carr. The drawings used to model the acrylic sheets in CAD were used to create .DXF files of the shapes of the sheets. These files were then used to cut the acrylic panels out of the sheets in a laser cutter. Next, after being cut, the panels had holes drilled in them that allowed for the bolts that squeeze the structure together to pass through them. This was done in the aerospace machine shop. These panels were printed for the thermal and drop tests, and more would have been printed for them balloon flight.

The mounting plates were manufactured in the machine shop in the aerospace building. A 2.5" x 36" x 1/16" 6061 aluminum sheet was ordered from McMaster-Carr, which was cut into shorter pieces after it was delivered to us. These shorter pieces were then hand milled into the final shapes using drawings made from the CAD model for reference. After the milling was done, through holes for #2 bolts were drilled into the plates to allow for the components to be attached to the plates and for the plates to be attached to the frame.

The mass simulants were manufactured in the machine shop in the aerospace building. Pieces of plastic and steel rods were taken from the scrap material area of the machine shop to be used to make them. The plastic pieces were originally cut to size using a band saw, but this proved to be too inaccurate for what we needed. The pieces were then cut to size using the mills to get the required accuracy. The plastic ended up being a good weight for the mass simulants of the PCB-based components, but was too light to simulate the battery. Therefore, holes were drilled into the battery mass simulant and sections of steel rod were put in the holes to increase its weight to the correct amount.

Figure 32 shows an exploded view of the CAD model of the ThinSat. The order of the components in the exploded view is the same as they would be when the ThinSat is fully assembled. The ThinSat is assembled by first bolting the payload board and a mounting plate to the primary frame in the payload area. Then the communications board, low pass filter, and SDR are bolted to the other mounting plate. This mounting plate is then bolted onto the secondary frame. The battery is then placed in its location above the payload board, and the two frames are brought together. Next, the antennae are attached to the outside of the secondary frame. Then the acrylic panels are placed on the outside of the two frames. Finally, the entire stack is bolted together with bolts passing through both frames and both acrylic panels.

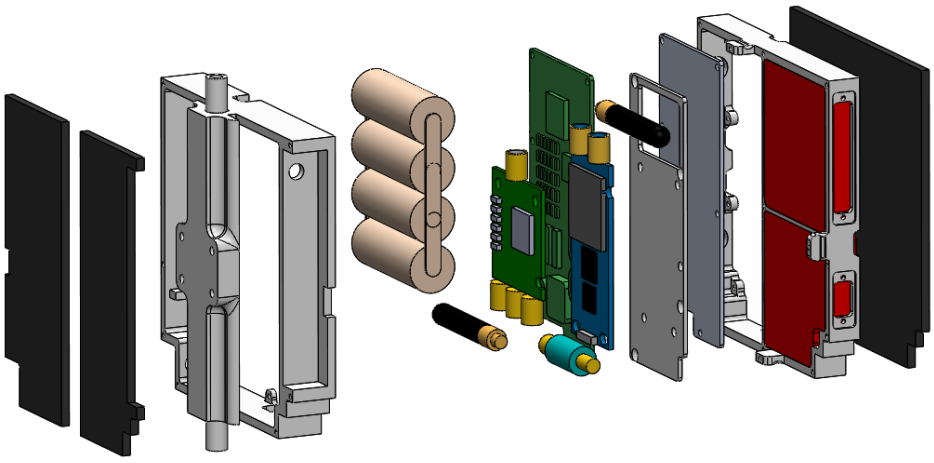


Figure 32. Exploded view of the CAD model

4.9. Thermal Design

Lara Buri

Manufacturing of the passive thermal system was simple and did not require extra machinery or supplies. Extruded polystyrene foam core purchased at Home Depot was simply cut to the shape of the frame of the ThinSat. For final manufacturing, the insulation would be attached using hot glue or something similar. Figure 32 in the Structures section shows where the insulation would attach to the ThinSat structure.

The copper strip used to transfer heat from the SDR would be cut in the shape shown in Figure 24 and placed on the SDR using a conductive thermal paste. The copper was purchased along with the thermal paste and the only manufacturing required would be cutting the copper into the desired shape.

5. Verification and Validation

5.1. Communications

Trevor Weschler, Lauren DeMoudt

Verification for the communications system is done in four stages. The first stage is wired testing which focuses on component functionality and expected performance. The second stage is power where an absolute measure of output power is measured for the system. The third stage is short range testing and is the first wireless test of the system. The criteria for the short range test is data rate and bit error rate. The fourth and final stage is long range testing and the criteria remain the same as the short range test.

5.1.1. Wired Testing

Wired testing serves as the primary way to verify component functionality by isolating the system from the complexities of wireless transmission. The wired testing serves two purposes. The first is component functionality. This deals with ensuring no communications components arrived damaged or inoperable. After ensuring the components are functional, testing is performed to determine characteristics for each component in the RF chain. The second purpose is that wired testing allows the verification of our software model. This is important to ensure there are no errors in signal processing in our software before introducing wireless noise and other effects to the signal.

The test was set up in the Aerospace Electronics Lab and involved two computers, two SDRs, attenuation, and the component being tested. Figure 33 shows a diagram of the test setup for each component tested. It is important to note that we used an SDR to receive the transmission. This receiver SDR was being run by a program that worked as a spectrum analyzer. This allowed us to see signals that were more powerful than others. However, the drawback to these programs is they are for amateur use to explore the radio spectrum around them. Therefore, they do not give exact power measurements of signals. The programs only give power measurements relative to the lowest power being detected at the time. This meant that the TOMCAT team could measure relative performance value like gain or attenuation, but could not measure absolute performance values like power output. This prompted the need for a power test which will be discussed later in this section.

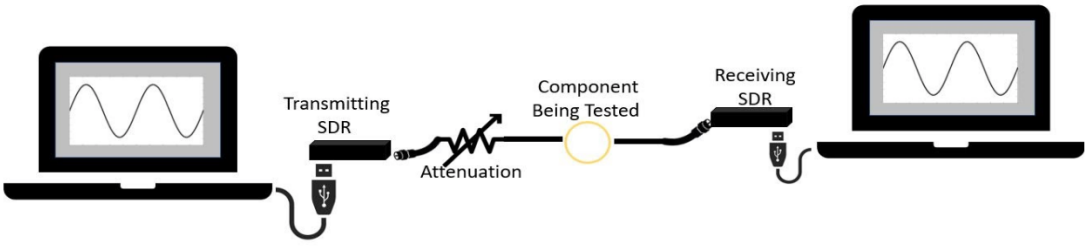


Figure 33. Communications Wired Test Setup

Functionality was established on all communications hardware. That is, no components were faulty, and all were able to be turned on and pass a signal through. Additionally, the software model was able to control the transmitting SDR to send data to the receiving SDR. With these ground-level functionality tests out of the way, the team began testing for how well each component met our model using the setup in Figure 33.

a. Low Pass Filter The first component to be tested was the low pass filter. The low pass filter is a passive electronic component that attenuates signals at frequencies higher than the cutoff frequency. The cutoff frequency of the low pass filter for our design is 1000 MHz. Above this frequency, we expect the filter to follow the manufacturer's data for attenuation. At our transmission frequency, 915 MHz, we expect 1 dB of attenuation. Figure 34 shows the results of TOMCAT's test against the manufacturer's data.

TOMCAT data stops before the manufacturer's data because at the next data point, the signal is too weak for the receiving radio to pick up. This is expected since we are transmitting with less power than the manufacturer was for their test. As can be seen with the results, our data is slightly better than the manufacturer's data. This could be due to power difference in our tests. These results are acceptable as the data showing the performance in our power regime is important to classifying how our system will perform. The important point is that our data is within family of the manufacturer's data.

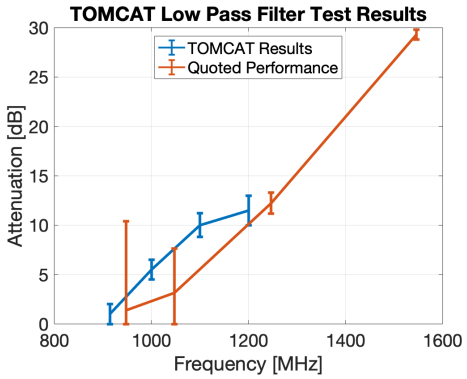


Figure 34. Low Pass Filter Test Results

b. Low Noise Amplifier The second component to be tested was the low noise amplifier. It is important to note that in this test we are testing the ground low noise amplifier on the evaluation board, not the on-board low noise amplifier on the communications PCB. The expected value from the manufacturer that the amplifier should produce is 38.5 dB of gain. Figure 35 shows the results of the low noise amplifier test.

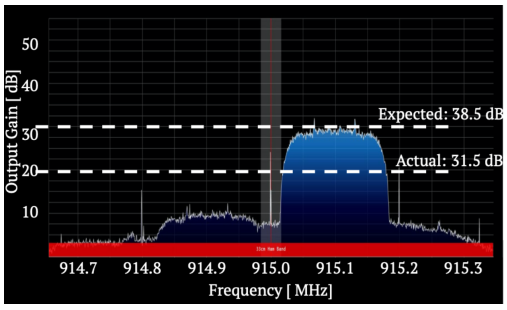


Figure 35. Ground Low Noise Amplifier Test Results

The data shows that in this case, our low noise amplifier significantly under performs compared to the manufacturer's data. The reason is that the team had to hand solder the SMA power connection which input more noise into the system. Running this updated gain value through the link budget still allows for an acceptable link margin though. As such, the team is satisfied with the performance of the low noise amplifier.

c. Power Amplifier The third component to be tested was the power amplifier. It is necessary to note that this test refers to the ground power amplifier, not the one on the communications PCB. The expected gain from the amplifier is 20 dB. Figure 36 shows the test results for the power amplifier.

In this case, the test results are within 10% of the expected value. The team plugged this actual value back into the link model to verify that this was an acceptable value. The link model still returned a sufficient margin for the team to accept the performance of the ground amplifier.

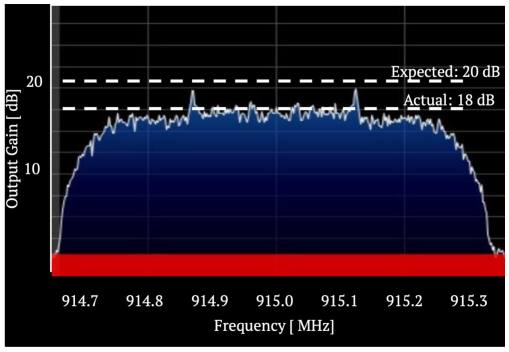


Figure 36. Ground Power Amplifier Test Results

d. Communications PCB The final component that was to be tested was the communications PCB. Due to an error on the first revision and issues with soldering the second revision, this test was not able to be completed before project progress was halted. The design of this test was the same as the above tests for the other amplifiers. The only difference was that the low noise amplifier and power amplifier were on the communications PCB. This test would have verified the gain for both the low noise amplifier and power amplifier for the on-board system. The expected gain values were 38.5 dB and 17.5 dB for the low noise amplifier and power amplifier respectively.

e. Wired Test Impacts The wired test was key to reducing the uncertainty in our link budget model. Understanding how our actual components perform allowed the team to make a realistic assessment of the risk for the link margin. This also serves to isolate component level issues from other issues that can occur in an RF system. Most notably, the uncertainty with broadcasting wireless. There are so many variables that generate noise or signal attenuation that having an accurate understanding of component performance is essential. With the knowledge of exactly how our components perform we are better able to model how much the signal will be effected by wireless transmission.

5.1.2. Power Testing

The power test is similar to the wired test except for two important differences. First, this test is a measure of absolute power instead of relative. This is important to know because there is a legal power transmission limit that is applicable in our broadcast spectrum. That limit is measured as 6 dB effective radiated isotropic power. Second, this test must be conducted in Dr. Scott Palo's research lab. The Aerospace Electronics Lab does not have equipment capable of measuring output power in our frequency range. Due to the change of location and measurement, this test was separated from the wired testing despite having all components still wired.

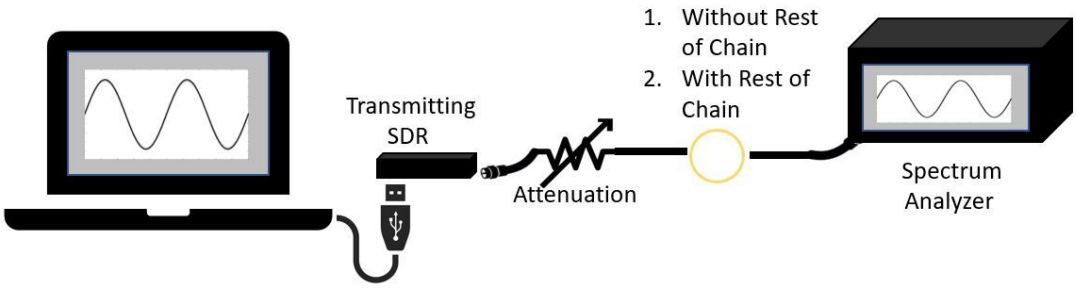


Figure 37. Communications Power Test Setup

Figure 37 shows the test set up for the power test. Two tests will be conducted. The first is a test of just the power output of the SDR's. Research indicates that the power output of the LimeSDR mini can range based on the transmission frequency. It is important to use the spectrum analyzer to know what power output our hardware is producing. The second will include the rest of the transmission chain. Both the ground on-board transmission chains will be tested. Expected value for only SDR transmission is 0 dBm. For the ground and on-board transmission chains, the expected power values are 33 dbM and 27 dBm respectively.

This test was not able to be completed before project progress was halted. The team was finishing up paperwork in order to use Dr. Palo's lab when progress was halted. That said, this test provides two important details about the design that reduce the risk. First, the total power output of both chains. Understanding how well the real world design meets our power model is essential to ensuring proper performance over wireless testing. Without a power reading, we run the risk of not transmitting with enough power to cover the necessary distance or transmitting with too much power and breaking the legal limits. Having power results that confirm our model also provide confidence in the link model as we move to wireless testing. Second, this test serves as an additional verification of the earlier testing done in the wired testing section. If there is a discrepancy between the wired tests and the power tests, that would point to an error in one or both of the tests.

5.1.3. Short Range Testing

Short range testing is the first time in the testing process where we move to wireless transmission. At this point, individual component performance and link performance should have been characterized by the two previous tests. At this point, the system will have to overcome real-world noise and other effects in the intended frequency range. Figure 38 shows the setup for each of these tests. Note the use of the full RF chain. At this point, integration with the power subsystem is necessary to provide power for the amplifiers in the RF chain. Short range wireless testing has three defined sub-tests for TOMCAT.

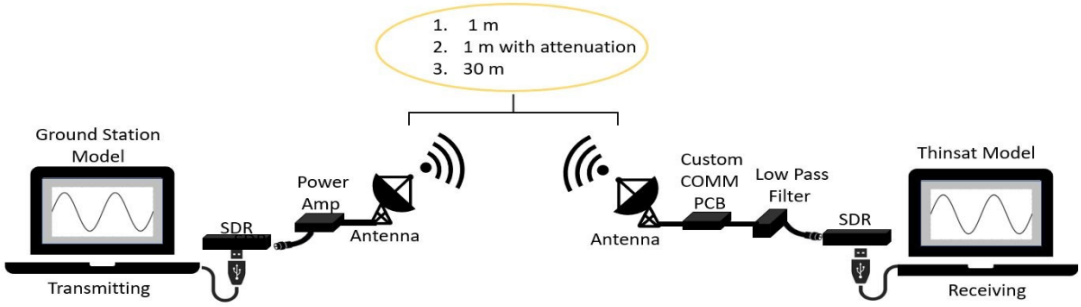


Figure 38. Communications Short Range Test Setup

The first test is the one meter distance test. We will transmit our signal wirelessly from one SDR to another over a distance of 1 meter in the Aerospace Electronics Lab. Beyond just a simple it worked or did not work, the team has expected values for the data rate and bit error rate for the test. Those values are 144 kbps and $10^{-}5$ bit error rate. If these values are achieved, there is confidence for the team's noise model over the short range.

Upon completion of the 1 meter test, attenuation will be added to simulate transmitting wirelessly over a longer distance. The difference in the losses associated with wireless transmission over long distances is mostly in space loss. Space loss is how much power is lost over the broadcast distance. This test serves as a way to evaluate how the system performs in a real-world environment with a weaker transmission signal due to simulated space loss. This test will be evaluated the same as the one meter test with data rate and bit error rate. We will maintain the same expectations for data rate and bit error rate as the one meter test. Note that these values are required values and expected for the balloon flight. For all short range tests, the system should be able to over-perform on these metrics.

The last short range test will be conducted in the Biotech field across a distance of 30 meters without attenuation. The evaluation metrics will be the same as the previous two tests. For this test we are looking at an increase of physical distance along with an outdoor environment. The balloon flight will be in an outdoor environment so it is important to move outside and begin understanding the noise environment outside.

Completion of these tests signal the readiness to move on to the long range test. At this point we would have more confidence in our link budget model and are ready to demonstrate long range communication capabilities.

5.1.4. Long Range Testing

The final communications test is the long range test. The test will take place with one station on the US-36 overlook and the other station on the roof of the Laboratory for Atmospheric and Space Physics. This is a distance of 7 kilometers. For this test, the goal is to encompass all the possible noise effects that were not covered in the short range test. This test set up is what is used by Dr. Palo and graduate projects teams for long range communications tests. TOMCAT's long range test include two sub-tests.

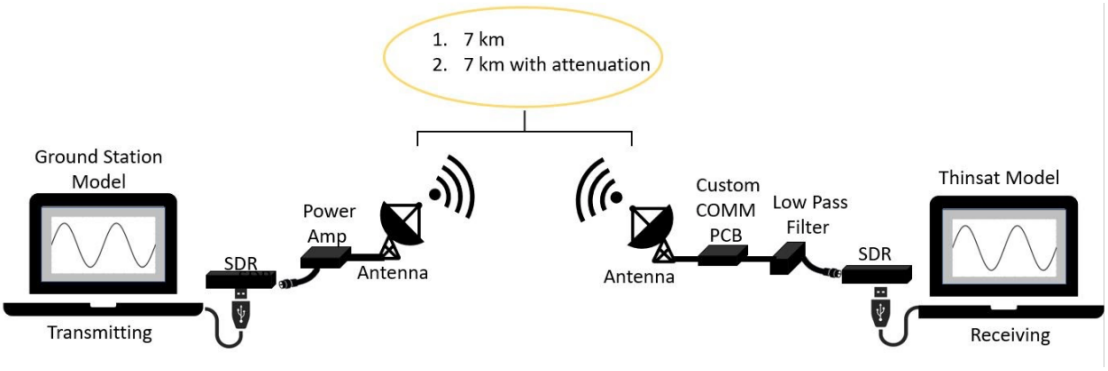


Figure 39. Communications Long Range Test Setup

The first test is an un-attenuated test over the 7 kilometer distance. The metrics will again be the data rate and bit error rate of 144 kbps and $10^{-}5$ bit error rate. Since the model is made for a maximum range of 104 kilometers, we expect that the system will over-perform at this range as well. The important difference between this test and the short range test is the noise environment. From the US 36 overlook, all of the noise produced by Boulder should be able to interact with the signal. Completion of this test will give confidence in our noise modeling. The next step is to add attenuation to simulate a longer distance.

As discussed before attenuation simulates a longer transmission distance by decreasing the power of the transmitted signal. For this second long range test, we will use enough attenuation to simulate a distance of 104 km. This is the maximum distance we anticipate the balloon traveling during flight. The metrics will be the same as the non-attenuated test. However, we expect to be much closer to our model predictions in terms of performance. Successful completion of this test confirms our model of noise and link margin overall. However, upon consulting with Dr. Palo and Dr. Rainville, they warned us that this test may not be successful. Below a certain elevation angle above the ground, noise and multipath effects are greatly increased. This could cause the test to be unsuccessful since the transmission will be pointing down into Boulder. They stressed that even if this test is unsuccessful, it does not necessarily mean the balloon flight will be or that our model is bad for the flight.

5.1.5. Requirement and Success Validation

All of the communication tests work to verify models developed in pursuit of requirements and levels of success. It is important to also ensure that the system validates the requirements and levels of success.

The major requirement that effects the communication subsystem is functional requirement 2: The payload shall communicate with the ground system during the balloon flight. In additional, the levels of success for the communications system is broken into three levels. Level 1 is that the communications system shall send and receive commands and telemetry over a wired connection. Level 2 says the communications system shall send and receive commands wirelessly over at least five meters of distance. Level 3 establishes that the communication system shall send and receive commands wirelessly over balloon flight with a range of 30 kilometers.

In order to validate functional requirement 2, a demonstration balloon flight was planned to showcase the capability to communicate during the balloon flight. The levels of success are not directly validated by the development of the communications system. Rather, the communications system development provides a base to validate levels of success. Once the system is integrated with the ground software and flight software, levels 1 and 2 of success can be demonstrated both over a wire and wirelessly. Level 3 is validated after the full system integration and balloon demonstration flight.

5.2. Flight Software

Manuel Lindo

The flight software design was going to be validated and verified through three tests: communications testing, IMU testing, and integration testing. Each test is outlined in the following subsections.

5.2.1. Communications Testing

The purpose of communications testing was to ensure the commandability and health status of our flight software and payload, confirming the validation of Functional Requirement 4.1: The payload shall be commandable. This test was meant to validate commandability through a ground station command coming from an SDR and error handling by the system.

The test would make sure the communications system could depacketize a CCSDS formatted command into readable information in KubOS. This testing was already complete within a hex-type packet being sent across a wired connection between a laptop and beagle-bone, as shown in Figure 69. However, the standard command to retrieve data from the KubOS telemetry database was not established yet. Nonetheless, it was confirmed that with a correctly formatted ground station command, the flight software system could retrieve the true data. To validate this confirmation, the C/C++ log files must be accessed, and then compared with the unpacketized response. Bit flips and error handling need to be measured and tested as well. To test these qualities, improperly formatted commands would be simulated through the SDR. Error handling would prevent runtime errors of waiting indefinitely for a database response and would mitigate risk of a full system collapse/freeze as a result of an error. To confirm the work-ability of the error handling following a test or balloon launch, the team would be able to measure how many commands were incorrectly understood by KubOS, and if the data suffered any alterations as it was being downlinked. The main tools used in this comparison would be the on-board log files, downlinked responses, and health beacons.

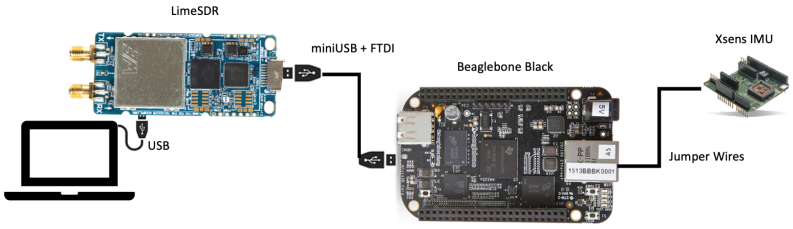


Figure 40. Communications Testing Setup

5.2.2. IMU Testing

The purpose of IMU testing was to ensure the collection of IMU data and validation of the functional requirements 4.2 and 4.3: The payload shall generate telemetry and the OBC shall store all telemetry during flight. It would verify that the IMU data, which has already been parsed and filled into the telemetry dictionary, could be stored into KubOS telemetry database. The test consisted of two steps: verifying the actual data being outputted by the IMU, and confirming the data is stored in the KubOS telemetry database. The test setup is shown in Figure 41.

As the IMU data generates data every 0.045 seconds, the data is logged into a separate text file within the C/C++ execution program before the socket is opened. This means the data had the least "interference" possible and was the closest to the true data. This data set was the benchmark for the IMU Testing. After the IMU data is stored into our database, the received telemetry is compared to the true, stored data. The timestamp and euler angles both provide verification and validation. This would be used to later confirm the scientific model, and measure the error rate of our COMM system. Moreover, a test would be needed to

ensure that the flight software could react to a sudden stop of telemetry collection, measured by the number of new entries in our telemetry database, and restart the payload board.

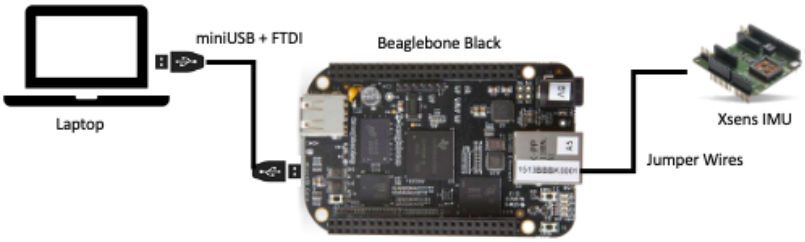


Figure 41. IMU Testing Setup

5.2.3. Integration Testing

Integration testing would ensure that full flight conditions could be simulated with an SDR, GNU Radio, KubOS Software, and IMU Hardware in order to fulfill functional requirement 4: The On-Board Computer, SDR, and FSW shall perform all necessary software tasks for the balloon flight. The test would need to encompass all of the functionalities of real flight such as commandability of the telemetry database under correct and incorrect ground station commands, periodic health application pings and error handling response, collection of raw IMU data from the Beaglebone, and finally integrating all of the subsystems into a working flight software system.

This test would take place over a long range wireless connection to ensure the replication of noise and other types of communication interference. However, a laptop would be plugged into the Beagle-Bone to more accurately measure feedback from our software system. There would be a record for incoming commands verified against the true commands, health beacon status and error handling verification, and IMU data validation. In addition, if there was a non-organic full system collapse, the system would need to be restarted using KubOS's built-in housekeeping app and the system behavior would need to be verified that it returned to normal after the event. This would be specially important for keeping a correct reference for our data timestamp. It is necessary to note this would not replicate a full system corruption during flight. Since we would not be able to communicate with the flight system to tell it to restart, we would have to rely on an automated system response to restart the flight software. For example, if there was no command received for 30 minutes or any data entry in our KubOS telemetry database system, then the payload board would need to be restarted.

5.3. Payload Board and Electronics

Grant Novota

The plan to verify if the payload board functions properly and fulfills functional requirement 4 was to perform three different tests: a boot test, a software test, and an integration test.

5.3.1. Boot Test

After the PCB assembly is complete, a boot test would take place to see if the computer system boots up. This test involves connecting the PCB to power and inspecting the indicator LEDs to see if the SMDs are active. KubOS would be flashed to the MCU and the presence of shell terminal access would indicate that the MCU is working as expected. The goal of this test is to determine if the MCU hardware design and assembly was done correctly.

5.3.2. Software Test

Once it is confirmed that KubOS is running on the payload board, the FSW would be loaded onto the MCU and the same software defined tests that were done to verify the FSW on the development boards would be used to verify that the payload board is functioning as expected. The goal of this test is to learn if hardware connections to the IMU and SDR were designed correctly.

5.3.3. Integration Test

The final test to verify the payload board is to connect it to the battery power and place it in the ThinSat frame. Voltage measurements would be taken for all of the power outputs (1.1V, 3.3V, 5V) and the battery input to verify that the power electronics were designed correctly. The goal of this test is to determine if the payload board was designed correctly for integration within the ThinSat and the power system.

5.4. Power Systems

Jake Schroeder

This section outlines the verification and validation of the power subsystem. The tests that were performed were aimed at validating the battery performance and comparing that to the manufacturer specifications, and verifying that the battery met the following requirement:
Functional Requirement 3: The payload shall survive and operate in the environment during all stages of the balloon flight.

5.4.1. Battery Cold Testing and Model Validation

Cold Tests 2 and 3 were performed in conjunction with the Thermal Subsystem, and the test setup and temperature results are given in Section 5.7. The objective of both cold tests was to verify that the battery was able to discharge in the low temperatures expected in the ThinSat during the balloon flight, even in the worst case scenario that the internal temperature dropped down to the outside temperature of −45°C. These tests reduced projected risks by demonstrating that the battery was in fact capable of powering all components in the low temperatures expected in flight.

In these tests, the battery was connected to a circuit that acted as an analog to the TOMCAT subsystems in terms of its power draw. In Cold Test 2, the battery was placed inside the ThinSat frame along with resistive heaters that gave off the same amount of heat as would be expected from the ThinSat components. The ThinSat body was then placed in the cold test environment. This test acted to give the team a clearer idea of what the battery performance would look like in the real-world balloon flight. In Cold Test 3, the battery was placed directly in the cold test environment so that the team could determine the performance of the battery in the worst case scenario. In both tests, the circuit setup was the same and is shown in figure 44. The voltage and current draw were measured throughout the test and are plotted in figures 42 and 43. Modeled lithium ion battery discharge capacity at varying temperatures is given in both 43 and 71 and discussed more in-depth in the Power Systems section of the Appendix. These figures show how similar the modeled battery capacity was to the actual capacity, and was very close at standard room temperature. The models differed more from the test at lower temperatures, allowing the team to reassess the model and verify the power budget's validity at the lower temperatures.

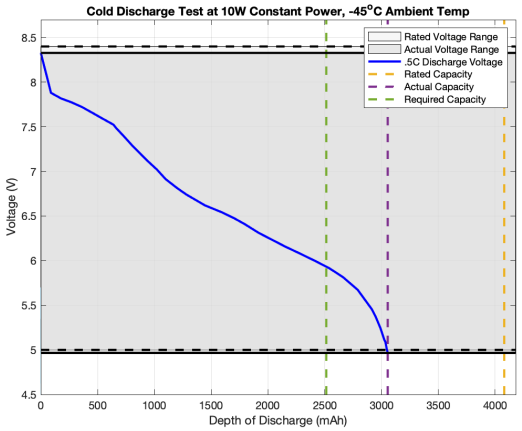


Figure 42. Cold Test 3 Results

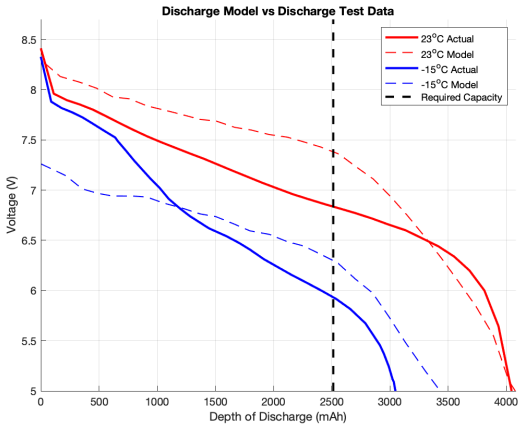


Figure 43. Battery Discharge Modeled vs Actual Results

Results: *Full Success* - Battery discharged over 3000mAh, with 2500 mAh being the minimum required.

5.4.2. Battery Characterization Testing and Model Validation

Battery characterization testing consisted of three tests: the battery capacity test, the voltage range test, and the current draw test. These tests were performed by constructing the circuit below, which served to provide the maximum power draw and strain on the battery that could be expected of the system given every component's power draw specification. The testing circuit used to conduct the tests is given in figure 44. The characterization testing was performed in the same way as in the cold tests, but in a room temperature environment, and served as a way to verify the battery specifications from the manufacturer in an ideal environment. This was taken into account when reassessing the power budget model and helped validate the accuracy of the battery discharge model shown in figure 43.

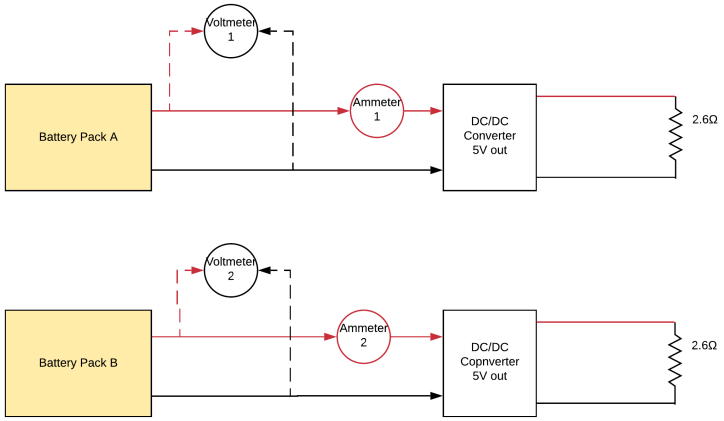


Figure 44. Battery Characterization Test Circuit Diagram

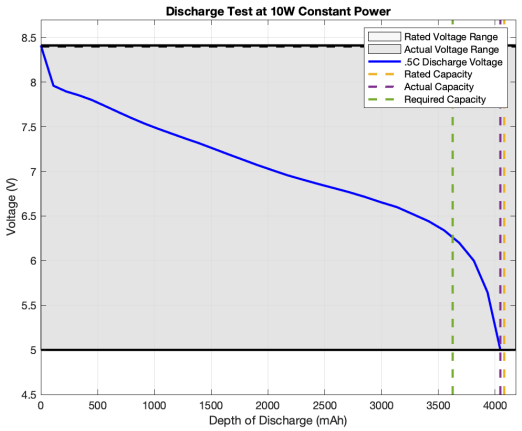


Figure 45. Discharge Test Voltage vs Depth of Discharge

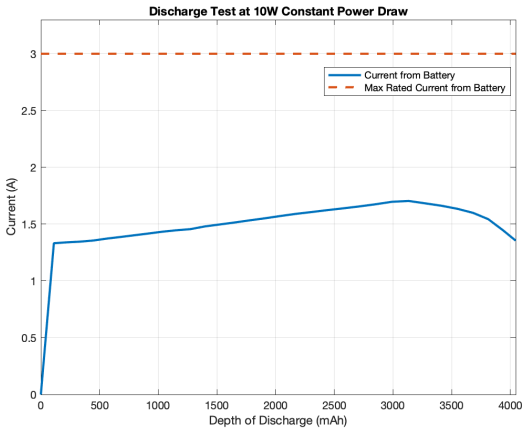


Figure 46. Current Draw vs Depth of Discharge

Voltage Range Test Results: *Full Success* - Actual Voltage Range: 4.99V to 8.41V

Battery Capacity Test Results: *Full Success* - Actual Capacity: 4046 mAh

Current Draw Test Results: *Full Success* - Max Current Draw: 1.704 A

5.4.3. Integration Testing and Power Budget Validation

Because of the halt in development due to COVID-19, no integration testing with the power system was able to be completed. There were to be two tests integrating the battery with the Payload Board: the PCB with Power Supply test and the PCB with Battery test.

There were to be three Full System Integration Tests: the ThinSat with Power Supply test, the ThinSat with Battery test, and the Full System Cold Test.

The ThinSat with Power Supply test's objective was to verify that the PCB converted power from the battery to 5V and 3.3V, and to verify that the battery current draw requirements are met. In order to power every component via the PCB, the current requirements for each power output from the PCB are as follows: 2.0A from 5V out, .08A from 3.3V out, and .08A from 1.5V out. This test would have helped to validate the Component Level Power Budget by determining if the PCB was able to supply the current draws predicted by the power budget. By meeting the following success criteria, this test would have reduced risk by demonstrating that the PCB was able to supply adequate power to all subsystems from battery. *Full Success:* Greater than 2.0A and .08A available from 5V and 3.3V ports respectively. *Partial Success:* Greater than 1.6A and .065A available from 5V and 3.3V ports respectively.

The ThinSat with Battery Test's objective was to verify that the battery could fully power the ThinSat with all systems running per functional requirement 3. The same process would have been followed in the ThinSat with Power Supply test except for replacing the power supply with the flight battery. This test would have reduced risk by demonstrating that the battery is capable of fully powering the ThinSat. The test success criteria is as follows: *Full Success:* All subsystems are fully operational throughout the test. Both ThinSat power tests would help to validate and help to improve the power budget model in table 12 by revealing the actual power draw of the system.

Full System Integration testing and cold testing would have been the final tests before the balloon flight to ensure all hardware and software worked together. The objective for the battery in terms of the full system integration was to verify that the ThinSat is fully operational in the Cold Test environment with all subsystems operating as they will during the Balloon Launch. This test would fully verify Functional Requirement

3, which required that the payload shall be operational throughout the duration of the balloon flight. These final tests would have reduced risks in terms of the power system by demonstrating that the ThinSat can fully satisfy Functional Requirement 3 and validating that the ThinSat can fully operate in a near space environment. The success criteria is as follows: *Full Success:* All subsystems are fully operational throughout the test.

5.5. Science Payload

Austin Scheck

The science subsystem of this project was verified using a combination of testing and simulation using past test data. This decision was made because the science mission revolved around characterizing a balloon flight, something which is rather difficult to accomplish using available ground-based testing equipment.

The requirements that the science team aimed to validate via testing are as follows:

- DR 3.4 - The IMU shall generate rate telemetry during ascent and descent
- DR 3.4.1 - Science data shall be able to be collected and updated at least once every second.
- DR 4.1.6 - The FSW shall forward IMU commands to the IMU
- DR 4.2 - The payload shall generate telemetry
- DR 4.2.1 - The OBC shall receive raw telemetry from the IMU

From the IMU's data sheet, the team could validate DR 3.4.1, as the IMU had a maximum data rate of 800 Hz in the I^2C format. That said, the team planned to also verify that rate through integrated system testing.

5.5.1. Sample IMU Ground Testing

To begin testing, the team borrowed a sample SparkFun Breakout IMU from the electronics lab at CU Boulder, where it was soldered in connection with an Arduino Due, also borrowed from the electronics lab. The goal of this test was to verify DR 4.2 and 4.2.1 in the absence of the proper Xsens IMU. The team sent the IMU a mock algorithm requesting that the IMU "downlink" its acceleration and rotations in three dimensions.

After some brief code manipulation in Arduino, the test was performed. The team attempted to characterize ground motion in the pre-flight phase of the mission by moving the IMU around in a series of gentle and more forceful translations and rotations.

Once roughly three minutes of data was acquired, the team dismantled the test setup and utilized the data output to inform decisions about how a pre-flight phase of flight looks in terms of acceleration and rotation. It was ultimately decided that maximum ground accelerations would reach 2 gs maximum, far lower than the anticipated spike of 5 gs when takeoff would occur. This data was promising for the reliability of the mission algorithm.

Another ground test, using the flight-ready Xsens model, was planned during the month of March, but was not performed because of the cease-development orders given by the department.

5.5.2. Mission Algorithm Modeling

Due to the nature of balloon flights, where the satellite is expected to incur sudden spikes of high-g loading in all directions for brief periods of time, practical testing of a mission algorithm was unfeasible. A simple drop test or throw test would not be able to accurately recreate the sensation of a balloon bursting, so the

team relied on previous ASEN 1400 Gateway to Space data to inform the creation and validation of the mission algorithm. As the final test flight was to be the exact same type of flight as these previous missions, the team agreed that this would lead to the most accurate algorithm.

Initially with three full-flight data sets to observe, the team came to a conclusion that the phases of flight could be accurately determined using two axes of acceleration. The following plot shows a sample output of flight phase over the course of a full mission. Note that a zero indicates pre/post flight, positive one indicates ascent, and negative one indicates descent.

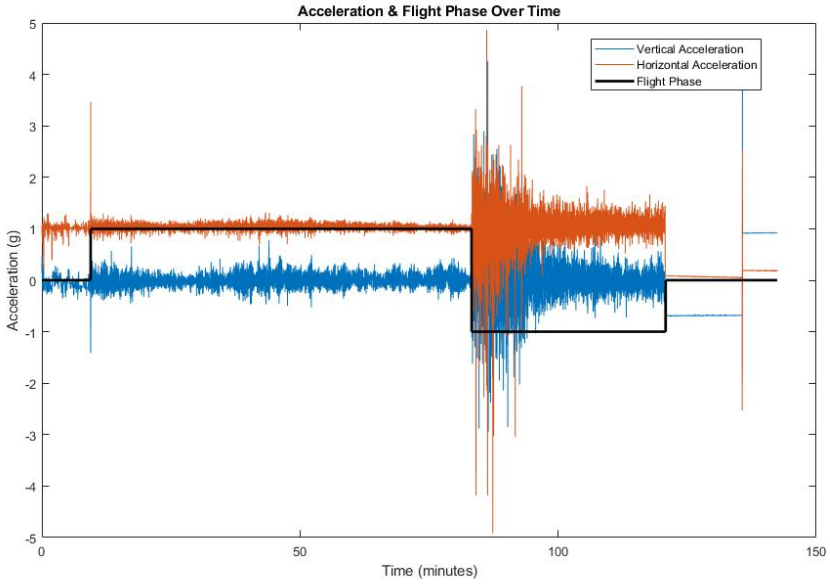


Figure 47. Flight Phase Using Mission Algorithm

The algorithm flowchart which outputs this plot is given in the Science Appendix. Before more previous balloon flights could be analyzed and characterized using the MATLAB algorithm, all development was halted. The team was confident, however, that requirement DR 4.1.6 could be validated. The script would have taken the input from a simple 'take data' command and outputted a simple value which represented the flight phase. This would have induced only marginal strain in the communications budget, and would successfully complete a science mission, as requested by the sponsor.

5.5.3. Full System Integrated Test

This test was planned for April, and would have seen the IMU and science algorithm working in tandem with the full TOMCAT flight system, all integrated into the ThinSat frame.

With a simulated ground station placed multiple kilometers away from the flight system, the team would have been able to validate DR 3.4, 3.4.1, and 4.2 in a live test. The IMU would have been powered and commanded to take data once every second, where it would communicate to the payload board, and be sent to the ground station via a CCSDS telemetry packet. This test would have shown the team that the full science system is operating effectively - accepting commands, generating data, deciphering flight phase, and downlinking telemetry. As with many of the other tests, this was not able to be completed before development ceased in March.

5.6. Structures

Matthew McCallum

This section outlines the verification and validation of the structures subsystem, starting with modeling and then discussing how testing verified the models and validate the project requirements. For the structures subsystem, the main requirements to be verified and validated were the following:
Functional Requirement 3: The payload shall survive and operate in the environment during all stages of the balloon flight
Design Requirement 3.2: The payload shall withstand the impact at the end of a balloon flight.

5.6.1. Structures Models

A few methods were used to attempt to predict the forces applied to the connections holding the components to the frame. These models were compared to the data from NearSpace that stated the number of connections needed for the payload area in order to determine the model accuracy. The first method used bar theory and a simplified structure to calculate a maximum acceleration that the ThinSat would experience. This method assumed that the mass of the ThinSat was in a bar on top of two bars that simulated the walls of the ThinSat. This method resulted in an equation for the maximum acceleration of the ThinSat, however the forces calculated from this predicted needed many more supports than were known to be needed for the payload area. The next method used a model predicting the oscillatory motion of the same two bar one mass model from the previous method to determine an equation for the position of the mass over time. This equation was then integrated twice to get an equation for acceleration over time of the mass and a peak acceleration was then gained from the equation. This maximum acceleration was then used to calculate a maximum force on the components, but this too predicted forces that required many more supports than were known to be necessary. Finally, the simplified system was modelled in MATLAB using ode45 to predict the motion of the ThinSat. This provided a maximum acceleration of the ThinSat that was used to calculated the forces on the components, but this model predicted needing many more supports than were known to be needed. After none of these models were found to be able to accurately predict the forces of impact, it was determined that a drop test would be necessary to verify the design. A model that is known to provide accurate results is the analysis of joints. The equations in Figure 48 were used to model the joints that connect the components to the ThinSat frame. This analysis showed that the joint is most likely to fail via edge shearing of the member (3D printed frame) at 57 Newtons of force. This provided the team with the strength of the connections being used, but it does not provide any information about what loads the components will be under. In order to gain more insight into this before the drop test could be performed, it was decided to interpolate the amount of mass that each joint could support in the impact from what was known to be able to survive from the vendor. The documentation from NearSpace limits the mass of the payload area to 55 grams. The payload area is supported by 7 joints, so it can be assumed that each joint should be able to support 7.857 grams of mass through the ground impact. This allowed the calculation that the SDR and communications board should be able to be supported by the 4 joints that were already designed for support, but that the battery would need 19 of these joints to support it. 19 of these joints would be far too many to use to support the battery, so a different system was designed using beams printed into the frame to support the battery. However, there was no way of knowing what forces the

Bearing in Bolt

$$F = \frac{tdS_p}{n_d}$$

Shear of Bolt

$$F = \frac{\pi d^2 S_p}{4\sqrt{3}\, n_d}$$

Edge Shearing of Member

$$F = \frac{2atS_y}{\sqrt{3}\, n_d}$$

Bearing in Member

$$F = \frac{tdS_y}{n_d}$$

Tensile Yielding of Member

$$F = \frac{(w-d)tS_y}{n_d}$$

Figure 48. The equations used to calculated the joint strength

battery would be imparting to its supports during the impact until after the drop test, so these beams were made mostly off of what little intuition the team had.

5.6.2. Drop Test

The failure of the models to predict the survivability of the ThinSat on impact with the ground at the end of the balloon flight warranted the use of a drop test to verify this requirement. The plan for the drop test was to drop the ThinSat onto a hard flooring to simulate a worst case impact with the ground and film the impact with a high speed camera. If the ThinSat received minimal damage from the impact, then the design would be verified to be able to survive the impact at the end of the balloon flight. If the ThinSat was severely damaged from the impact, then it would show that we would have needed to add additional padding to the ThinSat to ensure it could survive the impact. The high speed camera filmed the impact in order to measure the impact time and to verify the impact velocity.

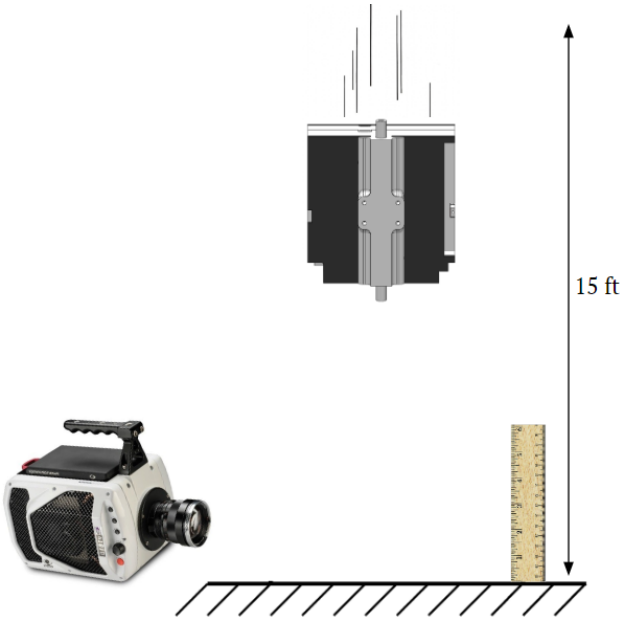


Figure 49. Diagram of the drop test setup

The first task that needed to be performed to be able to perform the drop test was to find a device that could be used to measure the duration of the impact. It was recommended that the team use a high speed camera from the ITLL. The other major task before the test could be performed was the creation of mass simulants to be used for the test. Dummy parts of the same mass and similar dimensions needed to be created to take the place of the actual components. The balcony level of the ITLL was about 15 feet above the main floor of the ITLL, and this is where the test was executed. Once everything was set up, a few test drops were performed with some small boxes to ensure that the ThinSat could be dropped accurately and that the high speed camera was working correctly. The actual test was performed by dropping the ThinSat with the mass simulants inside, recording the impact with the high speed camera, and evaluating the damage that the ThinSat sustained.

The main purpose of the drop test was to attempt to validate that the ThinSat could survive the impact with the ground at the end of the balloon flight. This test showed that without additional padding, the ThinSat was likely to sustain serious damage and possibly break components upon impact. This was shown in the test by the battery breaking loose, causing a portion of the frame to break off from the rest of the frame. The other components appeared to withstand the landing well, but the battery being free to move could cause serious damage to other components. Because of this, more drop tests would have needed to have been performed to validate that the ThinSat could survive the impact. These tests would have been performed with varying amounts of padding added to the outside of the ThinSat to determine how much padding would be needed to sufficiently protect the ThinSat from damage. The purpose of using the high speed camera during the test was to determine the duration of the impact. The high speed camera has a trade off of lower resolution for higher frame rates. In order to get a high enough frame rate the resolution needed to be reduced to a point where it became a little difficult to determine when the ThinSat impacted the ground. The information from the video could still be gathered with sufficient accuracy, but a higher resolution and higher frame rate would have helped make the process easier and more accurate. Determining the duration of the impact allowed for the forces involved in the impact to be calculated more accurately. These calculations predicted requiring a number of joints that is close to the number of joints calculated by

interpolating the information from NearSpace. The calculations from the high speed camera also provided information about the forces involved in the impact, which would have been used to better secure the battery.

5.7. Thermal Design

Lara Buri

This section outlines the verification and validation of the thermal subsystem, starting with modeling and showing how testing would verify the models and validate the project requirements. For the thermal system, the main requirements to be verified and validated were the following:

Functional Requirement 3: The payload shall survive and operate in the environment during all stages of the balloon flight

Design Requirement 3.1: The payload shall be able to withstand the temperature range of flight (-56°C to 38°C).

5.7.1. Preliminary Thermal Model

In order to characterize the performance of a passive thermal system, the team created a preliminary 1D, steady state heat transfer model that attempted to model the effects of the changing temperature with altitude throughout the high altitude balloon flight.

For this model, heat transfer by conduction, convection, and radiation were accounted for but sun loading or Earth's albedo were not accounted for. The model used concepts of thermal resistance networks to solve for the heat transfer from the component in question to the surrounding environment.

For brevity, the full results of this model can be found in the Thermal Appendix at the end of this report. Figure 50 shows the results of this model for the payload area of the ThinSat. The main takeaway from these results is that adding insulation will greatly increase the temperature inside of the ThinSat and the thicker the insulation, the higher the temperature will be. However, these results have some downfalls, mainly in the fact that the temperatures become unrealistically high according to the model. This means that the assumptions made to compute the temperatures are not adequate for the system, and that higher fidelity models need to be developed to yield more accurate results. These high-fidelity, 3D, transient models require computational softwares that the team did not have access to. For this reason, the team decided to build physical models of the ThinSat to characterize the temperature distributions of the system.

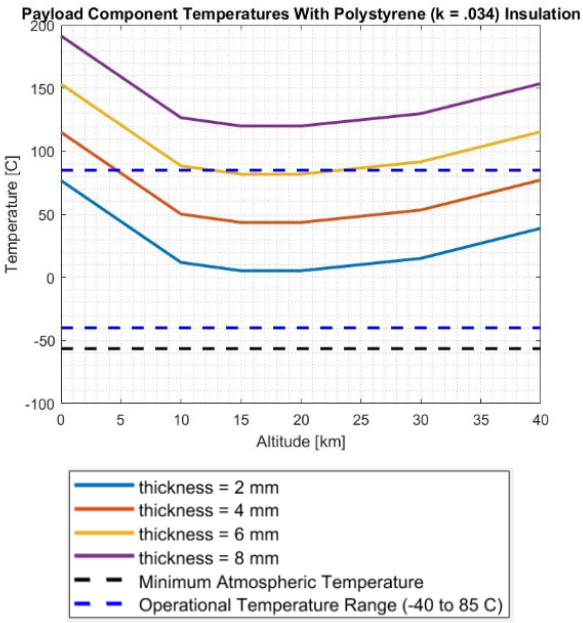


Figure 50. Model Results For Payload Space

5.7.2. Baseline Physical Model

To better characterize the temperature within the ThinSat, the team built a physical model of the TOMCAT system that would provide a simulation of the thermal conditions. The baseline model included two ThinSat frames placed together in flight configuration, two power resistors that simulated the heat coming from the

electronics boards (namely, the payload board and SDR), acrylic sheets to attach to the frames, and insulation. Thermocouples were placed in areas of interest to measure the temperature distribution inside the test specimen. An image of the test setup can be seen in the Thermal Appendix. The test setup was placed in a cooler with dry ice such that the ambient temperature reached -60°C, similar to the conditions of near space. Two tests were run, one without insulation to provide control data, and one with extruded polystyrene insulation.

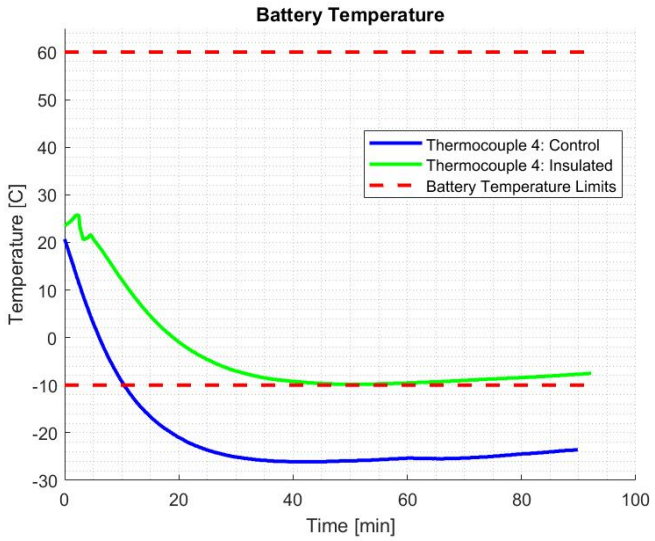


Figure 51. Baseline Physical Model Results for Battery Area

Figure 51 shows the results of the test for the area that houses the battery. The battery is of great concern because the performance of Lithium-Ion batteries degrades significantly if temperatures become too cold.

These results show that while the battery area does not go below the limit of -10°C, it reaches the limit. This is not ideal as the baseline physical model likely has errors due to the model not being set-up exactly how it would be in flight. Because there is no margin on the temperature, any error could cause the temperature to go too low, adding risk of the battery becoming too cold for proper performance. This led the team to build upon the physical model in an attempt to have it more accurately represent the flight configuration of the ThinSat.

5.7.3. Cold Testing

In order to increase the fidelity of the physical model, another cold test was performed with an updated test specimen. The test procedure was very similar to that of the baseline physical model, only in this test, the ThinSat was filled with electronics board simulators in order to more accurately fill the space in the ThinSat. These simulators included 3D printed models of the payload board, the COMM board, and the SDR. A test battery pack was utilized such that a platform was built for battery discharge tests. The results of the improved physical model are shown in figure 52. Images of the test configuration can be found in the Thermal Appendix. The main takeaway from these results is that components becoming too cold is no longer a concern, rather, components becoming too hot was identified as a new risk. The SDR simulated temperature became very hot, and began to approach the temperature limit of the component. This led the team to add a copper 'heat pipe' to the thermal system design that would conduct heat away from the SDR to a cooler area on the ThinSat. This addition was

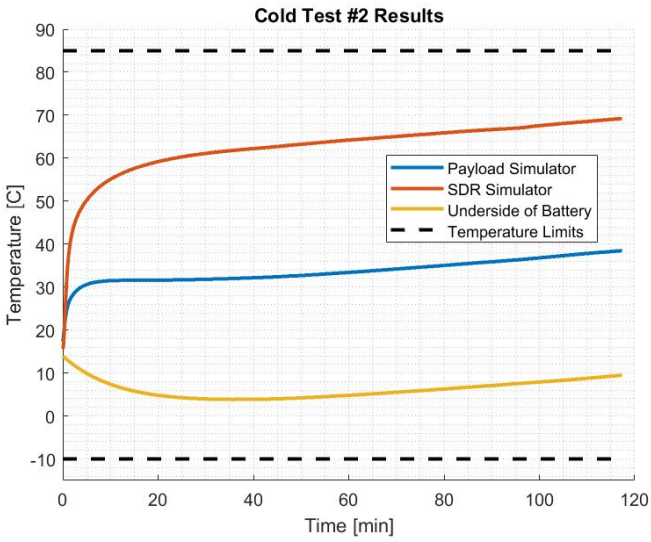


Figure 52. Results from Cold Test

planned to be tested during the integrated
cold test.

5.7.4. Integrated Cold Test

The integrated cold test was scheduled to occur on April 8th, but due to the COVID-19 pandemic, had to be canceled. This test would have allowed for a full integration test of the TOMCAT system from a thermal perspective. This test would look very similar to the cold tests outlined in the previous section, but the test configuration would be a full flight configuration, with all electronic components installed and power being supplied by the battery. This test would allow for thermal day in the life testing, battery state of charge testing, and component thermal modeling in an environment simulating near space. The goals of this integrated test were to validate the physical model and verify requirements 3 and 3.1. The physical models provide predictive results that could be compared with the integration test to characterize the performance of the overall system in the cold environment of near space. The integrated test results should look similar to the results from the previous cold tests, and any differences in results could be attributed to assumptions made in the physical model. A successful integration test would reduce risk for the balloon flight and allow the team to reach level 3 success for the thermal subsystem.

5.7.5. Model Assumptions and Results

There were a few key assumptions made in the physical modeling of the thermal system that would likely cause error/differences in the integrated test results.

- It was assumed that the resistive strips used in the physical models to mimic the electronic components accurately represented the heat distribution from the component. In reality, the components dissipate heat out of different chips on the boards, not uniformly as a resistive strip would. This could change the integration test results as the areas of heat concentration may be smaller than predicted.
- It was assumed there was little to no convection and convection was not forced in the physical models. This would not necessarily change the integration test results but could have an effect on the balloon flight thermal results if there was more or less convection present.
- It was assumed that all electronic components draw maximum power, which translates to the amount of heat dissipated by the components. In reality, the components may draw less power, impacting the amount of heat dissipated. This change would be evident in the integrated cold test by lower temperatures than predicted.

6. Risk Assessment and Mitigation

Cole

The team approached risk tracking with the mindset that risks aren't simply addressed and mitigated, but that project risk evolves over time. A spreadsheet to track project risks was created at the start of the project and was used to populated a risk matrix like the one shown in Fig. 53, which also shows how risks changed between CDR and now. Note that only the most critical risks are shown for brevity.

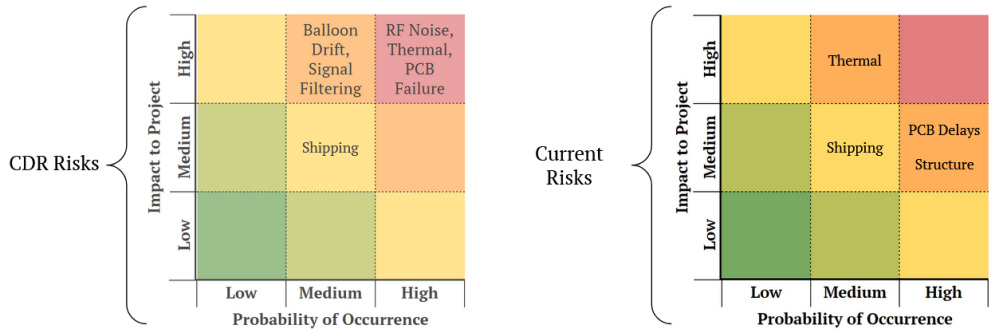


Figure 53. Risk evolution from Fall to Spring Semester

When mitigating risk, the goal is to move as many risks as possible to the lower left corner, where their chance of occurring is small and/or their impact to project success and requirements fulfillment is minimal.

6.1. Risk Management: Balloon Drift

Such was the case with risks like 'Balloon Drift' that were identified at CDR. The team selected an antenna of sufficient beamwidth to account for uncertainties. That risk, while still having a high impact to the project (loss of communication with the payload), has an extremely low probability of occurrence because of the mitigation plan, and was therefore removed from the critical risk matrix.

6.2. Risk Evolution: Thermal and the PCB

Risks like Thermal and PCB matters have remained on the critical risk matrix throughout the Spring semester, but are no longer the same as they were at CDR. These risks changed and evolved through several mitigation strategies, but remain on the team's radar.

6.2.1. Thermal

At CDR the team was concerned with the satellite payload dropping below the safe operating temperature specified by many of the components that make up the TOMCAT system. Much of this risk came from the uncertainty in the team's thermal models. Through additional modeling efforts and research into insulation materials, the team was able to reduce the probability of occurrence on 'Cold Thermal' to being extremely low (as described in Section 5.7). However in doing so, the team identified that the probability of certain components getting too hot as a new thermal risk. This was in the process of being mitigated using heat pipes and required further testing before the team was comfortable moving Thermal off of the critical risk matrix.

6.2.2. PCB

Issues related to the team's PCB were also identified early on as posing a significant risk to the project. At CDR, the concern was mostly on ensuring we received working components and ensuring we were able to properly debug any issues with the board. These risks were quickly mitigated through intelligent board design which allowed for easy debugging of components. If components were faulty, the team simply replaced them, as spare parts were part of nearly every purchase on the electrical systems. However, the PCB risk transitioned to a logistical problem, as the COVID-19 situation in China was interrupting our vendor's ability to deliver our manufactured board. This because a problem in late-January and we began looking for alternate vendors at that time. Little did we know, the same root cause of our PCB risk would eventually go on to impact the rest of the project.

6.3. Risk Discovery: Structures

In addition to existing risks evolving, new risks are constantly being identified and managed. One such risk that cropped up fairly late for our team was structural concerns. Because of largely-delayed testing of the structure of the ThinSat, we weren't able to fully recognize this risk until mid-March. In essence, drop testing of mass models of the TOMCAT system identified that the structure was likely to break apart on impact at the conclusion of the balloon flight. While this would not have jeopardized the operation of the communication systems of the project, it would make it impossible to fulfill all survivability requirements laid out by the team and defining Level 3 success. The team was in the process of mitigating this risk through a more robust mounting assembly for the battery as well as structural padding to cushion the impact.

7. Project Planning

7.1. Organizational Chart

Cole

The organizational chart shown in Fig. (54) shows the leadership roles of all team members. In addition to the technical leadership roles, all team members also serve as technical members of at least one subteam. For example, Trevor is the Communications Hardware Lead, and both Lauren and Srikanth contribute to the communications hardware team in addition to leading the Safety/Test and Communications Software teams, respectively.

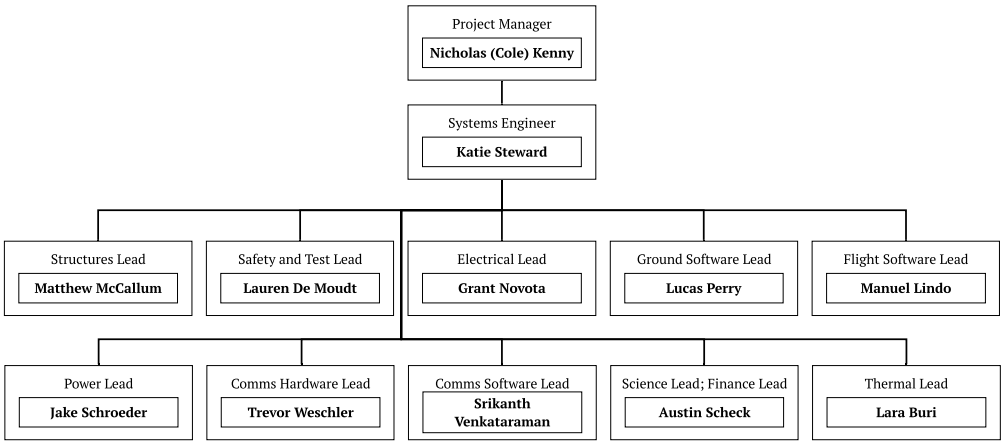


Figure 54. TOMCAT Org Chart

7.2. Work Breakdown Structure

Cole

The Work Breakdown Structure (WBS) in Fig. (55) shows an overview of the major work tasks that remain broken down by subteam. Each of these tasks have a number of subtasks which are available in the full Work Plan.

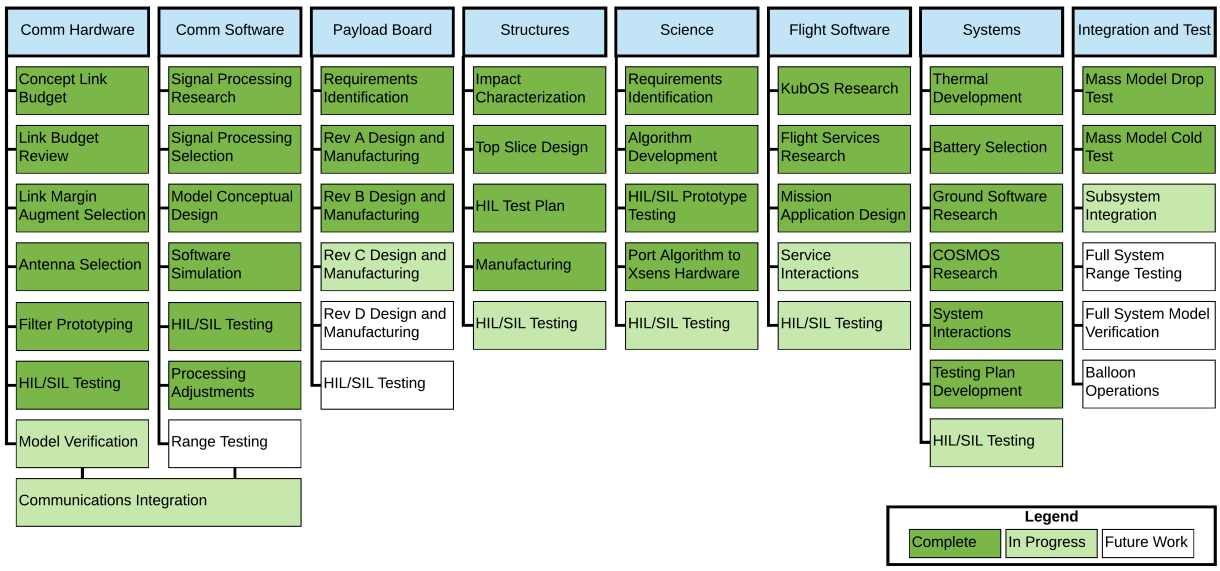


Figure 55. TOMCAT Work Breakdown Structure - Updated 04/20/20

These work tasks were determined from the gantt chart in Fig. 56 and serve as an easier way to visualize the remaining tasks in the project.

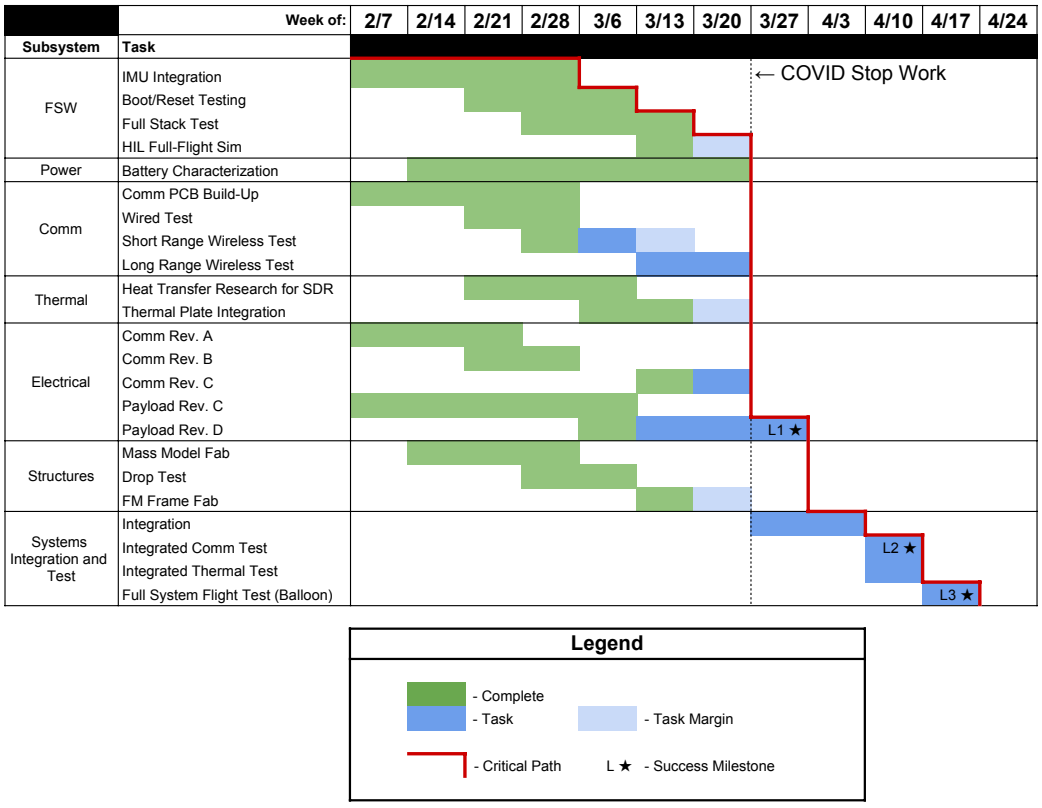


Figure 56. TOMCAT Gantt Chart - Spring

7.2.1. Critical Path Analysis

The critical path is defined as the continuous string of tasks which collectively have zero slack. By definition, the last task in a project will always be on the critical path. Working backwards, the critical path includes the balloon operations and the highest level system tests, including the integrated communications test, the integrated cold test. Since these tasks require the use of the TOMCAT satellite, they cannot be completed concurrently, but they can be conducted in a single week as scheduled since each test only takes a day to conduct.

Flowing back, the final payload board revision is on the critical path, mostly due to shipping and vendor delays at this point. As the payload board is required to integrate all other parts of the system, it is needed in order to do final top level testing. Additionally, all tasks in the Flight Software subteam are on the critical path, as this flight software is also needed before final testing can begin.

To help mitigate risk in the Flight Software, the team decided to split design effort, with one group focusing on using the KubOS code base and another group designing a bear bones, Lightweight FSW that would fulfil only the critical tasks required by the project. In doing so, we were able to mitigate the risks here and were on track to complete project objectives by the end of the semester.

Margins in the schedule were automatically calculated based on the task duration and predecessors. By defining the work week to match that of the senior projects schedule and allocating time to defined team members, the Microsoft Project scheduling features produce a reasonable and accurate schedule to work

from that doesn't overdraw from any particular teams (assuming the tasks were allocated properly when created). Additionally, engineering and management judgement was used when determining task duration.

7.3. Cost Plan

Austin Scheck

While the standard AES Senior Project budget is $5,000, TOMCAT was provided an additional $3,000 from their sponsor, Raytheon, for assistance in purchasing a required component of the TOMCAT system, the ThinSat Education Kit. This mandatory component had an initial projected cost of $3,000.

TOMCAT thus worked with a final budget of $8,000, of which a total of $5,961.01 was spent throughout the course of project development. The TOMCAT team was left with a substantial margin of $2,038.99 at the department-mandated development shut down in March.

7.3.1. Critical Project Expense Budget

In this section, the expenses relating to the critical projects elements will be noted, however a fully-comprehensive financial budget will be provided in the Appendix.

Subsystem	Component Name	Model Name	Qty.	Component Unit Price	Component Total Price	Subsystem Price
Payload Board	Microcontroller Unit (MCU)	BeagleBone Black Wireless	4	$76.13	$304.52	**$1,096.41**
	Inertial Measurement Unit (IMU)	MTi-3 AHRS IMU Dev Kit	1	$437.89	$437.89	
	Payload Printed Circuit Board (PCB)	n/a	1	$354.00	$354.00	
Communications Hardware	Software-Defined Radio (SDR)	LimeSDR Mini	4	$159.00	$636.00	**$1,066.70**
	Ground Antenna	Yagi Antenna	4	$92.70	$370.80	
	Flight Antenna	RPSMA	2	$3.00	$6.00	
	Communcations Printed Circuit Board (PCB) Rev. A	COMM PCB Rev A	5	$3.60	$18.00	
	Communcations Printed Circuit Board (PCB) Rev. B	COMM PCB Rev B	3	$11.97	$35.90	
Power	Battery	PR-CU-R782	2	$64.00	$128.00	**$184.95**
	Battery Charger	CH-L7405	1	$19.95	$19.95	
	Class 9 Hazard Handling Fee	HZ-S&H-Ground	1	$37.00	$37.00	
Structures	Aluminum	Multipurpose 6061 Aluminum	1	$5.89	$5.89	**$16.63**
	Acrylic	UV-Resistant Cast Acrylic Sheet	2	$5.37	$10.74	
Systems	Acer Chromebook Laptops	CB3-431-C5FM	2	$208.83	$417.66	**$1,417.66**
	ThinSat Education Kit	n/a	1	$1,000.00	$1,000.00	

Figure 57. TOMCAT Critical Project Expenses

This budget only comprises the most critical elements of the project - please note that the team incurred many more expenses on components for testing, integration, and modification of these aforementioned items. A chart breaking down the overall expenses by category is given below:

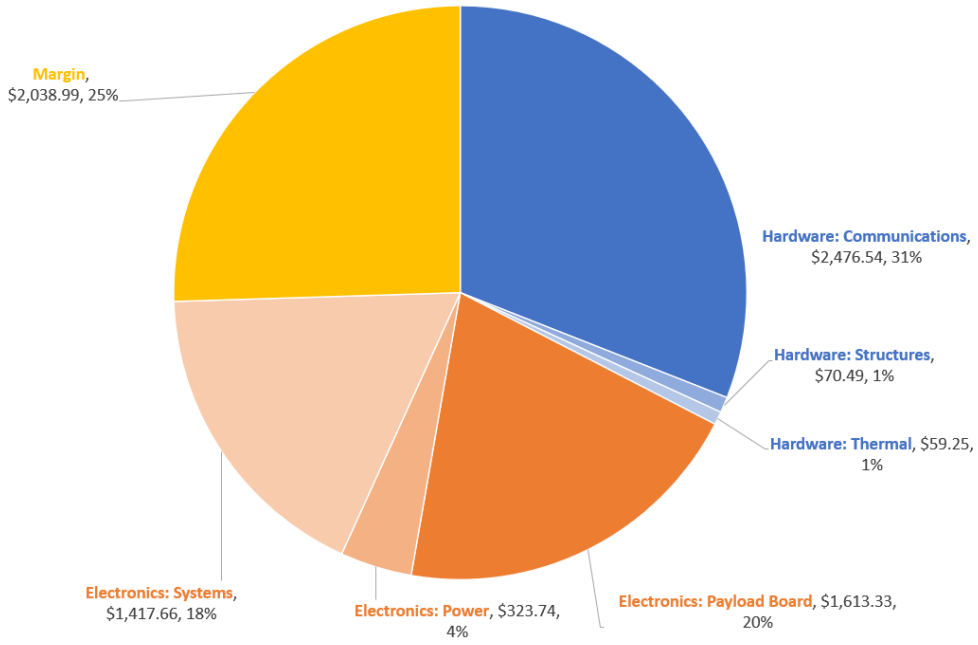


Figure 58. TOMCAT Overall Expenses

7.3.2. Uncertainties

The team was fortunate to have relatively few uncertainties regarding the procurement of components, save for a single item - the payload printed circuit board (PCB). This critical aspect of the project, serving as the command & data handling module for the TOMCAT flight system, was designed in January and early February by Grant Novota, the electronics lead. In February, it was ordered through PCBWay, a company highly-regarded and widely used within CU Boulder's electrical engineering department. They were selected on account of their low cost, rapid production speed, and trusted heritage producing quality components for previous electrical engineering projects. PCBWay is headquartered in China, and has offices across the globe, giving them an international presence.

Unfortunately, the order was sent over Chinese New Year, which initially delayed the design revision process of production. Shortly after New Year's celebrations had concluded, the novel coronavirus struck China and slowed PCBWay's production to a crawl.

The order was eventually finished in late March, more than 3 weeks after the initially-quoted finish date. If the CU Senior Projects curriculum was continuing on a nominal trajectory, this would have been a major setback for TOMCAT, as it would have delayed critical testing activities by multiple weeks.

7.4. Test Plan

Lauren

In general, for each subsystem associated with this project, a series of tests were scheduled. If these subsystems obtained any consumer off the shelf products, their preliminary tests involved ensuring that the functionality of these products matched the datasheet within a specified tolerance. After this, the team progressed to testing any custom hardware or software developed for the project subsystem in order to ensure that it functioned as intended and as predicted by any associated models. After these functionality tests, the different subsystems progressed to individual integration tests to ensure that all the different aspects of the subsystem integrated as expected and all the interfaces were compatible. Once the subsystem integration was completed, the subsystems were scheduled to focus on integration testing with other subsystems and

eventually the entire system as a whole. The tests were scheduled according to the time required for the completion of all the tests leading up to a given test. The test flow is shown in Figure 59.

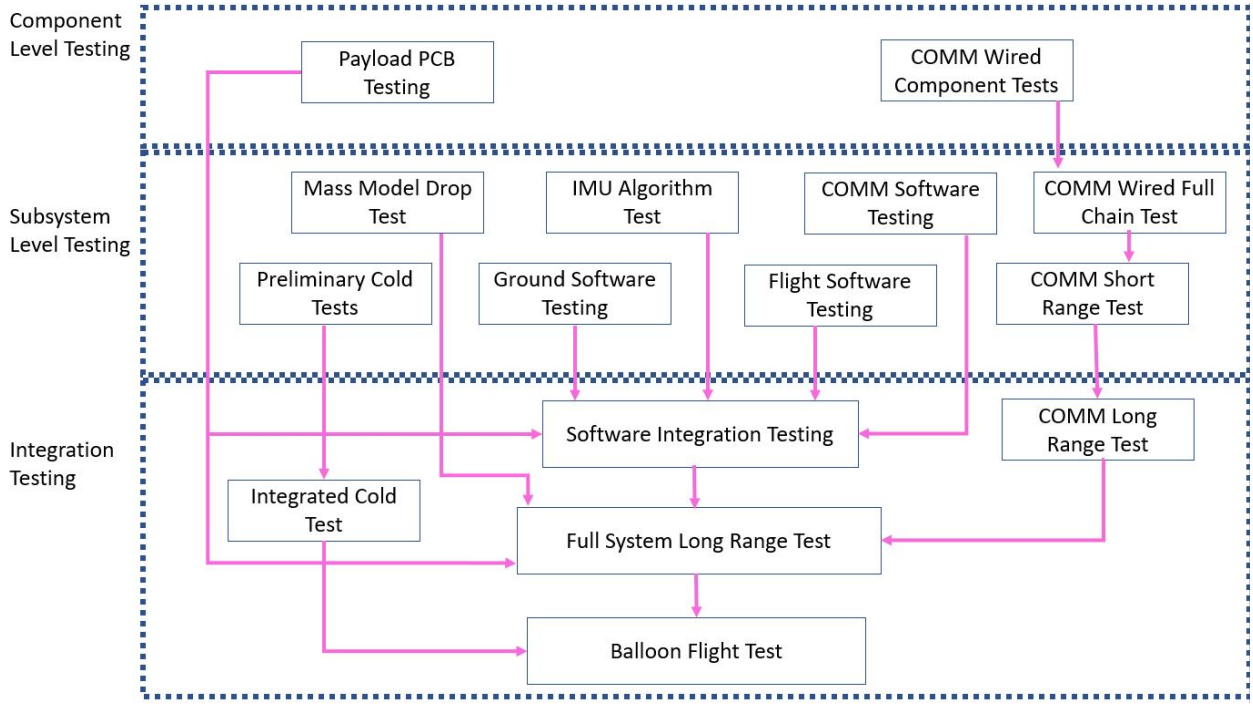


Figure 59. Test Flow and Dependencies

In addition to planning out this test flow, the team also had to plan out gaining access to specialized test equipment and facilities in order to conduct these tests. The cold test would have required access to a freezer. The team intended to coordinate with either LASP or BioServe to gain access to these facilities. In addition to this, for the communications testing, the team needed access to both the electronics lab (for lower level testing) and the RF lab in the aerospace engineering building on campus in order to sufficiently test all of the hardware received for the wired communications testing and the power testing. Access to both of these labs was coordinated through communication with Dr. Palo and Prof. Schwartz.

8. Lessons Learned

Cole

There are a number of technical, logistical, and administrative topics which, given prior knowledge, could have allowed the project to progress smoother. Some detail will be given for major lessons learned, and additional lessons will simply be listed at the end of this section for brevity.

8.1. Technical: Communications Design

A major focus on the TOMCAT project was that of communications. Unfortunately, the 2 weeks of communications instruction in the Aerospace Electronics course did not prepare us for the technical depth that was required for this project. We leaned heavily on both online resources and additional faculty members in the department, including Dr. Nick Rainville, Dr. Scott Palo, and Trudy Schwartz.

8.2. Logistical: Shipping Delays and the Cost of Waiting

While our biggest shipping delays were mostly the result of the COVID-19 situation internationally, which was unavoidable, there were a few circumstances where the team was held up waiting for parts to arrive due to just slow shipping. In hindsight, we had plenty of leftover budget, and it would have been beneficial to simply pay for expedited shipping in the first place. In some situations, the team definitely could have benefited from actually just buying the part again and paying for expedited shipping rather than waiting for the first order to arrive. While this would have resulted in double-ordering some items, it would have saved the team a lot of time in the long run and having spares isn't really such a bad thing.

8.3. Administrative: Learn What Gets People Excited and Push That

Specifically from a Project Manager's perspective, it's important to understand how to motivate people. Most of my management experience came from leading CU Hyperloop and in that environment, it was easy to push people hard to get things done. Everyone was there because they found the work to be challenging yet rewarding. Hyperloop was a passion and got people excited, so working for hours and hours on hyperloop tasks was easy and fun for everyone. That's not always the case with Senior Projects, or any school project in general, and my naive understanding of getting people to put in work was absolutely detrimental to the team (especially in the beginning). As a manager, it's important to understand that everyone works in different ways and that in order to have an effective team, you need to push people in ways that get them excited to do work on their own.

8.4. Uncatagorized Lessons Learned

- Talking to everyone (or as many as possible) on the PAB before presentations is critical. It gives you time to answer questions before the presentation even starts and allows the individual PAB members to better understand your project ahead of time rather than having to learn it all in the condensed presentation period.
- Big meetings are a waste of time, but sometime you have to just bite the bullet and do them so that people don't feel left out.
- Define project requirements early, and whenever you make a decision, check it against your requirements. Decisions made in other ways lead to scope creep.

9. Individual Report Contributions

- *Cole Kenny*: Purpose, some of Obj+Requirements sections. Process - Top Level Design section. Some reorganization of Communications, Science, and Flight Software sections. Risk, and some Planning sections (Org chart, WBS, Gantt chart). Lessons learned section. Creation of template. General management of section and proofreading assignments.
- *Katie Steward*: Project Objectives Functional Requirements, Design Process Outcome; Systems Engineering, proofreading assignments.
- *Austin Scheck*: Science Design, Science Manufacturing, Science V&V, Cost Plan, Science & Finance Appendices
- *Grant Novota*: Payload Board and Electronics Design, Payload Board and Electronics Manufacturing, Payload Board and Electronics V&V, Payload Board and Electronics Appendix
- *Jake Schroeder*: Power Systems Design, Power Systems Manufacturing and Integration, Verification and Validation of the Power System
- *Lara Buri*: Process: Thermal Design; Manufacturing: Thermal Design; V&V: Thermal Design; Thermal Appendix; proofreading
- *Lauren De Moudt*:Test Plan in planning section, COMM HW testing diagrams
- *Lucas Perry*: Ground Software: Design process, Manufacturing and Integration, Appendix
  Flight Software: Manufacturing-Communications Service, Manufacturing-Integration, proofreading
- *Manuel Lindo*: Flight Software: Design Process, Manufacturing and Integration V&V Communications IMU, Testing, Integration, and Process Flight Software
- *Matthew McCallum*: Structures Design, Structures Manufacturing, Structures V&V, Structures Appendix
- *Srikanth Venkataraman*: Comm SW Design, Comm SW v& V, Comm SW Appendix
- *Trevor Weschler*: Communictions Design, Communications Manufacturing, Communications V&V, Communications Appendix

References

[1] *OSD335x System-in-Package A New Era of Integration and Flexibility based on AM335x*, 2020. https://octavosystems.com/octavo_products/osd335x/.

[2] "Fetching Telemetry Data," , 2020. URL https://docs.kubos.com/1.18.0/tutorials/querying-telemetry.html?highlight=telemetry.

[3] Chen, S., Wan, C., and Wang, Y., "Thermal analysis of lithium-ion batteries," , 2005. , URL http://www.sciencedirect.com/science/article/pii/S0378775304008596.

[4] Lijun Gao, Shengyi Liu, and Dougal, R. A., "Dynamic lithium-ion battery model for system simulation," *IEEE Transactions on Components and Packaging Technologies*, Vol. 25, No. 3, 2002, pp. 495–505. .

[5] Twiggs, R., et al., "The ThinSat Program: Flight Opportunities for Education, Research and Industry," *AIAA/USU Conference on Small Satellites*, 2018, p. all.

[6] Tristancho, J., "Implementation of a femto-satellite and a mini-launcher," Master's thesis, Universitat Polit'ecnica de Catalunya, 2010.

[7] "CCSDS Protocol," , 2019. URL https://public.ccsds.org/about/default.aspx.

[8] "CANISTERIZED SATELLITE DISPENSER(CSD) DATA SHEET - Planetary Systems," , 2018. URL https://www.planetarysystemscorp.com/wp-content/uploads/2018/08/2002337F-CSD-Data-Sheet.pdf.

[9] "ANT-916-CW-RAH Data Sheet," , 2013. URL https://linxtechnologies.com/wp/wp-content/uploads/ant-916-cw-rah.pdf.

[10] "A09-Y8NF: RFMAX 4 Element 900 MHz Yagi Antenna," , 2016. URL https://www.arcantenna.com/a09-y8nf-rfmax-4-element-900-mhz-yagi-antenna-8-1-dbi-with-n-female-connector.html.

[11] Jackson, J., "Fall Final Report (FFR)," Tech. rep., University of Colorado Boulder, 2019.

[12] Kenny, N., Steward, K., et al., "Thinsat Operational Management of Commands And Telemetry Fall Final Report," Tech. rep., University of Colorado Boulder, 2019.

[13] "Rover and Air Visual Environment Navigation," , 2018. URL https://www.colorado.edu/aerospace/current-students/undergraduates/senior-design-projects/past-senior-projects/2017-2018/rover-and-air.

[14] "Europa Lander for Science Acquisition," , 2016. URL https://www.colorado.edu/aerospace/current-students/undergraduates/senior-design-projects/past-senior-projects/2015-2016/elsa.

10. Appendices

Communications Appendix

Trade Studies

a. Software Defined Radio

i. Metrics and Weighting The metrics that the team decided upon as determinants of an SDR's compatibility with the scope of the project are conveyed in Table 16 along with the rationale for these choices and associated driving requirements. In addition to this, a weighting scheme was developed to allow the aspects that were deemed more important to have a greater influence on the overall scoring. These metrics and their corresponding weights reflect the needs and priorities of the team as well as the customer.

Table 16. Trade Topics and Weights for the Software Defined Radio

Metric	Weight	Driving Requirement	Rationale
Max Power Draw	0.1	N/A	Power draw should be minimized to enable the smallest possible power supply. Due to the minimal power supplied from the ThinSat bus, an alternate power source will be used to power the SDR and associated components.
Size	0.1	N/A	The size of the SDR must be minimized. While the SDR is not required to fit within the payload space, it will need to attach to the exterior of the bus.
Mass	0.1	N/A	The mass of the SDR must be minimized.
Operating Temperature	0.15	3.1	The SDR must be capable of TX/RX over the expected temperature range.
Cost	0.05	5.1	Cost must fit within project budget. While cost is an important factor to consider, the other components that the team is required to purchase will only take a small portion of the budget. Thus, there is more room for flexibility in this category.
Bandwidth	0.1	2.4,2.5,2.6	The bandwidth is related to the quantity of data that can be transmitted, received, and handled. It is essential for the radio to be able to effectively transmit and receive important command and telemetry data.
Max TX Gain	0.1	2.2,2.4	Gain will drive the maximum transmission distance across which the ground SDR can communicate with the onboard SDR, which will be critical during the balloon flight.
Max RX Gain	0.1	2.2,2.4	Many SDRs have differing TX and RX gain, but the rationale for this metric is the same as above.
Configurability/Compatibility	0.2	2.3	Ease of configurability/compatibility will diminish complexity and time consumption involved with configuring the SDR interface with the Raytheon ground system and OBC.

ii. Metric Quantification The ranges across which the metrics were scored were dependant on the available SDRs on the market as well as the project requirements. Because of the much higher power consumption of the SDR over other systems, the range over which this metric is scored is much larger than in

other studies. Similar to other studies though, operating temperature scores also serve as a proxy for scoring the complexity of the necessary thermal system that we will have to implement after further analysis. Specifically with regards to configurability and complexity, which is the only metric not supported quantitatively, higher scores were given to SDRs which are supported in MATLAB, as the team has considerable experience working with it.

Table 17. Explanation of Analytical Scale for the Software Defined Radio

Metric	0	1	2	3	4
Max Power Draw	≤ 1000 mA	≤ 750 mA	≤ 500 mA	≤ 250 mA	≤ 100 mA
Size	≥ 90 cm^2	≥ 72.9 cm^2	≥ 55.8 cm^2	≥ 38.7 cm^2	≥ 21.6 cm^2
Mass	≥ 100 g	≥ 80 g	≥ 60 g	≥ 40 g	≥ 20 g
Operating Temperature	untested	25 °	0 °	-25 °	-50 °
Cost	≥ $759	≥ $616.74	≥ $474.50	≥ $332.32	≥ $189.95
Bandwidth	≤ 28 MHz	≤ 45 MHz	≤ 62 MHz	≤ 79 MHz	≤ 96 MHz
Max TX gain	≥ 45 dB	≥ 60 dB	≥ 70 dB	≥ 80 dB	≥ 90 dB
Max RX gain	≥ 65.5 dB	≥ 70 dB	≥ 80 dB	≥ 80 dB	≥ 90 dB
Configurability / Compatibility	3rd Party Configuration SW	Major Configuration SW	Major Configuration SW, limited 3rd party MATLAB support	Major Configuration SW, 3rd Party MATLAB Interface	Major Configuration SW, Mathworks Supported

iii. Trade Results Once the metric scale was developed, the various attributes of the four SDRs were rated accordingly and the results are illustrated numerically and visually (color coded) in Table 18. As can be deduced from these results, the leading model choices were the LimeSDR Mini and the USRP B200mini. Although the LimeSDR lost out in the configurability/compatibility and TX gain categories, it made up for it with its superior bandwidth capabilities, modest cost, lower minimum operating temperature, and minute size. The low TX gain can be compensated for by implementing a form of RF amplification in order to achieve the desired communication range. The LimeSDR primarily lost out in the configurability category due to the minimal MATLAB support, which is what the TOMCAT team has the most experience implementing. Thus, the configuration time and complexity on the ground SDR side will be increased slightly but not to an extent that is seen as unreasonable. However, the LimeSDR was noted to also have GNU Radio support. The team anticipates this to be advantageous for interfacing with the on-board SDR.

As a general note, this trade study resulted in an overall under-performance of the selected SDR models, with final scores ranging in the 1s and 2s as opposed to the 2s and 3s like in other subsystem trades. The reason for this is not because of a poor selection of SDR choices, but because of the very specific constraints of the projects resulting in very few SDRs that fulfill all objectives.

Table 18. Trade Study Results for the Software Defined Radio

Metric	Weight	Lime SDR Mini	BladeRF 2.0 Micro	HackRF One	USRP B200mini
Max Power Draw	0.1	0	0	2	0
Size	0.1	4	1	0	3
Mass	0.1	4	1	0	4
Operating Temperature	0.15	3	2	0	2
Cost	0.05	4	2	4	0
Bandwidth	0.1	4	1	0	1
Max TX Gain	0.1	2	4	0	4
Max RX Gain	0.1	1	2	4	0
Configurability/Compatibility	0.2	2	2	3	4
Total	1	2.55	1.7	1.4	2.3

Discussion on Design Options Considered

b. LimeSDR Mini The first suitable option is the LimeSDR mini available from SparkFun. This SDR is the cheapest available option at $189.95 while having the lowest mass and size. Where this option separates itself is its large bandwidth capability at 96 MHz. This will allow the entire 902-928 MHz range to be usable if necessary for data transmission.

This SDR has a lower max receiver gain of the SDR options. Along with an average max transmission gain will most likely mean that a separate RF amplifier will be necessary to ensure the communications system works at the distances expected on balloon flight. Additionally, the LimeSDR mini doesn't have much MATLAB support. Since the TOMCAT team is experienced in using MATLAB, being able to interface with the ground SDR with MATLAB is preferable. This will cut down in overall development time for the communication system since unfamiliar software will not have to be used to interface with the ground SDR. On the flight SDR side, this radio has GNU Radio support. GNU Radio is the software that will be used to interface with the flight SDR.

Additionally, the LimeSDR is rated to operate down to -40°C, but is normally operated at 25°C. The lowest temperature TOMCAT is expected to see is -56°C. This means that there will have to be some thermal design in order to ensure the radio continues to work throughout the balloon flight. The extent of this thermal control will depend on modeling the thermal characteristics of the radio.

Pros	Cons
Cheapest viable option	Low receiving gain
Largest bandwidth capability	Average transmission gain
Small size	Limited MATLAB support
Low mass	Typical 25°C operating temperature
Down to -40°C operating temperature	

c. BladeRF 2.0 Micro xA4 The second suitable option is the BladeRF 2.0 Micro xA4. This SDR has high transmission gain and reasonable reception gain. While investment in a RF amplifier will probably still be necessary, it may be possible to get a cheaper amplifier since the signal will already be amplified. In addition, the BladeRF has more MATLAB compatibility than the LimeSDR. 3rd parties have been developing MATLAB software for this radio since it was released around a year ago. With additional GNU radio software, the team will be less pressed to develop custom software to support the BladeRF's operations.

This radio has the second highest mass and size of the options up for consideration. As future iterations of this project will launch into space, both of these characteristics need to be minimized. The BladeRF is also the 2nd highest cost option. At $480.00, purchasing two of these radios will consume almost 20% of

the total budget allocated for the project. When adding in any external elements such as antennas and RF amplifiers, this option could become a significant portion of the total budget.

Finally, the BladeRF only has a 0°C minimum operating temperature. It requires thermal protection in order to ensure communications can be maintained over the entire balloon flight. This SDR needs more thermal control than the LimeSDR or other radios that are rated to operate to lower temperatures.

Pros	Cons
High transmission gain	Large mass
MATLAB compatibility	Large size
Decent receiver gain	High cost
	0°C operating minimum

d. HackRF One The third suitable option is the HackRF One. This SDR has the highest receiver gain than any of the other options at 94 dB. Additionally, an entire 3rd party interface has been developed for MATLAB to control the HackRF One. This would make software development for the ground SDR must simpler. This combined with the GNU support for the flight SDR would streamline the software development process.

Additionally, the HackRF One has a lower power draw than any of the other SDR option. A USB 2.0 interface is used instead of a USB 3.0. The total power consumed is 2.5 W generated by using 500 mA at 5V. The Hack RF One is relatively cheap at $299.95. This could free up budget for any necessary external elements such as antennas and RF amplifiers.

As the "tinkerers" radio, the HackRF One also has several drawbacks. Despite the high receiver gain, the transmission gain and bandwidth is the lowest of all the options. This reduces the maximum achievable data rate. Additionally, the radio has the highest mass and largest size of the available options.

Perhaps the most concerning characteristic of the HackRF One is that it has no documentation on operating temperature. If selected, much more modeling would be necessary to determine the minimum operation temperature and design a suitable thermal control system. Such a process could complicate the thermal control system which will most likely be essential for other components of the ThinSat.

Pros	Cons
High receiver gain	Low transmission gain
3rd Party MATLAB interface	Narrow bandwidth
Low power draw	Undocumented operating temperature range
Cheap cost	High mass

e. USRP B200mini The final suitable option is the USRP B200mini. This SDR is similar to the LimeSDR in terms of minimizing total footprint. It has the second smallest size and mass of the options. Additionally, Mathworks has developed USRP support in the MATLAB Communications Toolbox. It provides radio-in-the-loop design and modeling, allowing the user to design and verify SDR systems. Coupled with GNU radio support, the software support for the B200 mini is unmatched by any of the other options. In addition to low mass, small size, and great software support, this radio also has a high transmission gain at 90 dB.

One major downside is that the SDR costs $759.00 per radio. In addition, the need for amplifiers on both the flight and ground SDRs will increase the overall setup cost further. Finally, the B200mini does not meet the operating temperature requirements out of the box. At -20°C, a thermal control system will need to be developed to keep the SDR operational.

Pros	Cons
High transmission gain	High cost
Mathworks MATLAB and GNU radio support	Low receiver gain
Small size	-20°C operating minimum
Low mass	

f. Communication Band When determining which frequency band to communicate within, the main concerns of the team were accessibility and the effects on communication capabilities between the ground and onboard systems. The initial concern of the team was that many frequency bands require the user to possess some kind of license in order to broadcast within the band. Due to the complexity often involved with obtaining a radio license, the team determined that it would be best to operate within a band designated for amateur radio services that does not require a license. The team consulted the customer and they recommended transmitting in the ISM bands. The ISM bands are designated for industrial, medical, and scientific applications. The bands designated for amateur, unlicensed services considered by the team were the 2.4-2.5 GHz band and the 902-928 MHz band. The higher frequency band allowed for the team to transmit across a wider bandwidth. This would allow for more transmitted data at once. However, transmitting at higher frequencies means that the signal would be less powerful when it reached the onboard system and thus more difficult to receive and decode. Due to the ISM bands being open for amateur radio services, the noise within these bands is unpredictable. Since the amount of telemetry the team intends to downlink will be small enough that the narrower bandwidth will still be sufficient, the team determined that the bigger concern was the ability to deal with unanticipated noise. Due to this, the team chose to transmit within the lower frequency band because it results in a larger link margin and thus more room for unexpected noise, despite the sacrificed bandwidth. The link margin and budget will be discussed in more detail later in this document.

g. Antenna After conducting research on antenna functionality and design available for SDR applications, the team was able to fairly quickly determine what was desirable without the need to do a full trade study. The most important factor was the frequency range of the antenna. For this project, the team intends to utilize the ISM band frequencies allocated for amateur satellite services. As discussed above, the chosen frequency band encompasses 902 MHz through 928 MHz. For the onboard system, the team noted a preference for omni-directional antennas. Despite the diminished gain, and thus decreased range, the capability to receive and radiate energy in all directions was deemed more important due to the lack of on-board pointing capabilities for the ThinSat. In addition to this, the team needed to minimize the size and weight for the onboard antennas in order to minimize the mass contribution to the system and allow the antennas to fit within the ThinSat dimensions. Due to these constraints, the team determined that a quarter-wave monopole antenna would work best for this application. With quarter-wave antennas, the antenna is a fourth of the length of the wavelength of the radio waves. This allows for the antenna to be shorter than other options that broadcast at the same frequency, thus making this model ideal for this application.

After taking these design preferences into consideration, the specific antenna model chosen for the onboard system for both the receiving and transmitting antennas was the ANT-916-CW-RAH, a quarter-wave monopole antenna, from Linx Technologies shown in Figure **??**.This antenna has a length of 47 mm which is small enough to fit within the design space. It possesses an omni-directional radiation pattern as desired and a center frequency in the middle of the ISM band chosen for operation. While there were other antenna choices with similar radiation patterns and roughly the same size, this antenna was primarily chosen due to its relatively high gain of 2.2dBi and the lower bound of the operational temperature extending to -40 degrees Celsius.

The requirements for the antennas for the ground system varied slightly from those of the onboard system. The team initially considered an omni-directional antenna on the ground as well but after noting that there would never be a need for the ground antenna radiate in all directions, the team turned the research

efforts towards antennas with higher gain but a more restricted radiation pattern. In addition to this, the mass and size constraints placed on the onboard system are not an issue with the ground system. Thus, the team could focus more on finding an antenna with a large amount of gain and a desired radiation pattern without having to think too much about size or weight. With this in mind, the team decided on a four element Yagi Antenna (Model A09-Y8NF) from Arcadian Incorporated, shown in Figure **??**. Its operational range falls within the 896-980 MHz which encompasses the desired operational zone for the project. The primary justification for choosing this specific model was that it provided a large amount of gain without sacrificing too much beamwidth. To be more specific, this model provides 8.5dBi gain, a 100 degree horizontal beamwidth, and a 70 degree vertical beamwidth. Despite the constrained vertical and horizontal beamwidth compared with an omni-directional antenna, the team is confident that they can adjust the design to compensate for this while still taking advantage of the high gain provided by the Yagi antenna. The specific design choices that account for the beamwidth constraints will be discussed in more detail later in this report.

h. Half-Duplex Communication In addition to the above considerations, the team also put some thought into how to communicate within the ISM band of choice. Since the bandwidth of this communications band is relatively narrow compared with other bands, 26 MHz, the team expressed a preference to communicate using the entire band rather than using half of the band for commanding and the other for telemetry. The negative side of this design choice is that the team will not be able to send commands and receive telemetry at the same time. In order to account for this, the team will implement a system to ensure that command uplink and telemetry downlink are offset. This will be discussed in more detail later in the report.

i. SDR Software After doing research on the various SDR software options available, two options were identified as the most viable: MATLAB/Simulink and GNU Radio. MATLAB is a closed source proprietary programming language and development environment that has a robust signal processing toolbox. GNU Radio, is an open source project that is popular among academics, hobbyists, and institutions. Three main criteria were used when evaluating the strengths and weaknesses of these two options: compatibility with the SDRs, performance, and ease of use. All of the SDRs considered in the trade study are well supported by GNU radio, with each having vendor-supplied plugins to use with their hardware. The MATLAB support for each SDR varied from having Mathworks supported libraries complete with example code to unofficial 3rd party support. While hardware specific support isn't necessary to interface with the SDRs, it does make it a lot easier. With the exception of the USRP B200mini, the GNU Radio support was equivalent or superior to the MATLAB support. When it comes to ease of use, most of the TOMCAT team have considerable MATLAB experience and limited GNU radio experience. The documentation and online resources for both are good and there they both have active help forums. In addition, both MATLAB and GNU Radio feature graphical programming interfaces that allow for more rapid design work. MATLAB supports standalone C++ code generation to a variety of targets. This means that the software design can be done in MATLAB code or graphically through Simulink, then turned into high performance C++ code that can be run on the OBC. While GNU Radio supports Python and C++, it only supports Python code generation from graphical designs. If the Python code is too slow, then a C++ version has to be written manually. In addition, GNU Radio code still needs GNU Radio to be installed on the system to run, as opposed to MATLAB generated code which can run standalone. Overall, both options were deemed viable, and the final decision came with the result of the SDR trade. Since the LimeSDR had the worst MATLAB support out of all the options, GNU Radio was selected.

j. Modulation Technique To encode binary data in a transmittable wave, the data must be modulated in some way. The two major forms of modulation considered were phase shift keying (PSK) and frequency shift keying (FSK). Phase shift keying encodes binary data in a carrier wave though phase shifts, while frequency shift keying encodes data in frequency shifts. Though FSK provides a higher signal to noise ratio in general, it was decided that PSK would be used since it is more bandwidth efficient, more power

efficient, and provides better bit error rates in noisy environments. Next, a decision was made on whether to use binary phase shift keying (BPSK) or quadratic phase shift keying (QPSK). BPSK uses two phases to encode data, while QPSK uses four. Though they have identical noise performance characteristics and QPSK is more bandwidth efficient, the decision was made to go with BPSK since the implementation is simpler and it still fulfills our data rate requirements. Finally, it was decided to use differential encoding (DBPSK). With differential encoding, the reference phase is the phase of the last sampled signal instead of the carrier reference phase. Since the carrier wave can be distorted during transmission, the received signal can be quite noisy. Differential encoding remedies this at the cost of slightly worse noise performance.

10.0.1. Trade Studies and Design Selection

The only trade study that was conducted for the communications subsystem was for the SDR. Detailed breakdown of how the trade study was conducted can be found in the communications appendix. There were many considerations for the SDR trade study including power draw, size, mass, operating temperature, cost, bandwidth, max gain, and configurability. The highest weight was configurability since the team did not want to loose time configuring a radio when one serves the project purposes off the shelve. The lowest weight was cost because the team expected a large margin in budget. All although metrics were tied in weight corresponding to their equal importance to operating on a ThinSat. In general, the SDR that required the least work to become compatible with the ThinSat mission would be selected.

a. Software Defined Radio Based on the results of the SDR trade study, the Lime SDR Mini was chosen. While the Lime SDR Mini and the USRP B200mini had similar scores, several key factors made the Lime SDR Mini the best choice. For one, it has a much higher bandwidth than the USRP B200mini. This allows for the potential for higher data rates when receiving telemetry if paired with an amplifier. Also, the Lime SDR has a minimum operating temperature of -40°C, compared to 0°C with the USRP b200mini. Since the minimum temperature expected during the balloon flight is -56°C, the Lime SDR will require a less robust thermal control system, which simplifies the design. The cost of the USRP B200mini also poses a challenge. The cost of the 2 SDRs alone would take up 30% of the budget, and that doesn't include additional necessary hardware like antennas or amplifiers. While one of the strengths of the USRP B200mini over the Lime SDR is its transmission gain, the use of an amplifier can mitigate this shortcoming. Since each of these options would require an amplifier anyways, the effect of the lower gain will be minimal. Finally, while the USRP B200mini has better integration with MATLAB and can also communicate using protocols defined in GNU Radio (compared to only GNU Radio protocols being supported on the LimeSDR), the extensive documentation of GNU Radio will help mitigate the learning curve associated with using the unfamiliar software. Overall, since the LimeSDR performs better in nearly every other category, it's selection over the USRP B200mini was obvious. Further, since GNU Radio will be implemented on the flight SDR anyway given the more powerful OBC, implementing it on the ground station SDR should present minimal difficulty.

Ground System

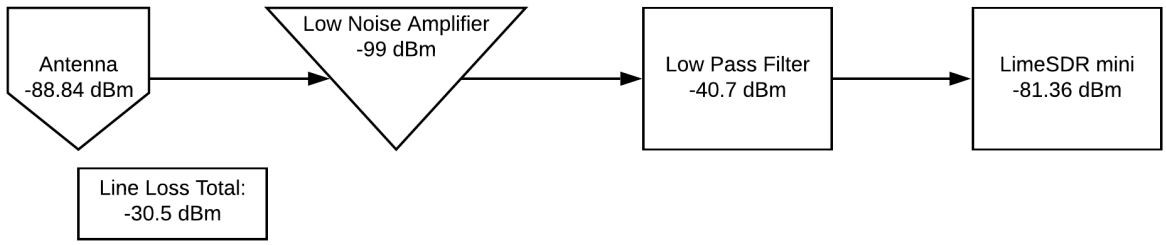


Figure 60. Ground Receiver RF Chain

Figure 60 shows the ground side receiver chain. The same hardware components are used. Note the low pass filter meant to prevent signal aliasing. In addition the signal power at the radio is above -100 dBm ensuring that the signal can be decoded. The difference between the two is that the ground receiving antenna has a larger gain meaning the signal arrives in the chain with more power than the on board version.

The transmission chain is designed to transmit as much power as possible. The layout includes the radio, a power amplifier, and then the antenna. In the transmission case, there is a difference between the on board and ground chains. On the ground, the power amplifier is weaker than on board. This is due to the antenna gain on the ground being much higher gain than the antenna on-board. The ground side power amplifier gives 20 dB of gain and the on-board power amplifier gives 22 dB of gain. The associated effective isotropic radiated powers are listed in the link budget section.

10.1. Communication Software Verification and Validation

The communication software verification and validation has three main phases: pure software tests, laptop tests with the SDRs, and flight computer tests with the SDRs. The purpose of the pure software test was to ensure the functionality of the signal processing system. As there was no hardware in the loop and it would be difficult to simulate the noise and interference expected during a hardware test, the only metric used to judge the success of the test was a successful transmission of data through the full signal processing chain. This test was performed successfully in the fall. The next phase of the comm software testing was a wired test with a laptop and two SDRs connected by a wire. The purpose of this test was to ensure that the signal processing software worked with the hardware and that the performance was within the required specifications. The two metrics of interest were data rate and bit error rate. This test was carried out by sending a text file between the two SDRs, measuring how long it took and checking for errors. The resulting data rate from this test was 218 kb/s, which is higher than the needed 144 kb/s. The bit error rate was zero. The final communication software test would be a repeat of the previous test but with the payload computer instead of a laptop. Since the main limiting factor for data rate is the processing speed of the computer, it is important that the software be tested on the much weaker payload computer. This test was not carried out due to the early end of testing. However, it was expected that this test was going to be successful because most of the intensive data processing is done on the receiving end, and the payload computer would be transmitting most of the time.

Flight Software Appendix

Trade Studies

a. Metrics and Weighting As discussed previously, the IDE selected plays a large role in providing a good environment for the flight software development. The selected metrics of importance are displayed below. The weighting of each metric is also an important factor of the trade study process, and so these were taken into account based on what was needed most to have a successful IDE to fulfill the requirements of the project as defined by the project requirements.

Table 19. Trade Topics for the IDE

Metric	Weight	Driving Requirement	Rationale
Complexity	0.15	N/A	There is a steep learning curve for some of the more robust IDEs. It can be difficult to understand all of the IDE's capabilities and use them to drive the FSW development forward.
MCU Compatibility	0.3	4.1	FSW developed in the IDE and MCU should be compatible for the payload to be commandable, generate telemetry, and store all telemetry during flight.
OS Compatibility	0.1	5.3	It should support the conventional operating systems to ensure every collaborator can access and modify the FSW.
Popularity	0.2	N/A	Community help and resources are very important in high-level development
Programming Language Support	0.05	N/A	It should support conventional programming languages
Additional Features	0.2	N/A	It should provide a convenient developing environment for code compilation, debugging, and team collaboration

b. Metric Quantification The scoring criteria are described in the following table. In general, metrics which were directly related to project requirements were weighted higher. The compatibility metric scores were determined by comparing the IDE's ability to communicate with either the MCU or a computer operating system during development. This is also how the program language support category was scored.

Complexity and Popularity were scored using relative comparisons, with complex and unpopular IDEs being unknown to not just the team, but also the community at large. Conversely, less complex, popular solutions had support from a wide community online and at least one team member of TOMCAT was familiar with the IDE.

Table 20. Explanation of weights for IDE

Metric	0	1	2	3	4
Complexity	Unfamiliar to all team members/very unintuitive with minimal documentation	Unfamiliar to all team members/very intuitive and step-by-step documentation.	Previously used by or familiar to at least one team member/ very intuitive with step-by-step documentation	Previously used by or familiar to >1 team members/very intuitive and step-by-step documentation	At least one team member has expertise/extremely intuitive with step-by-step documentation
MCU Compatibility	The MCU is unable to communicate with the IDE.	The MCU can partially communicate with IDE. None of the IDE's features are compatible with MCU	The MCU can partially communicate with IDE. Some of IDE's features are compatible with MCU	The MCU can fully communicate with IDE. All of IDE's features are very compatible with the MCU.	The MCU can fully communicate with IDE. All of IDE's features are very compatible with the MCU. There is extensive MCU compatible libraries/plug-ins and, payload and hardware integration.
OS Compatibility	The IDE is not compatible with readily available operating systems	Linux only	Linux and either Mac OS or Windows	Mac OS, Windows, Linux	Mac OS, Windows, Linux, Web Browser
Popularity	The IDE is widely unknown.	The IDE is known in specific community, not very well rated by users.	The IDE is widely known, not very well rated by users.	The IDE is widely known, very well rated by users.	The IDE is industry practice and has outstanding user ratings, with minimal criticism.
Programming Language Support	C,Python/Micr and Arduino not supported				C/C++ or Python/Micro Python or Arduino fully supported
Additional Features	Only code compilation	Code compilation, and simple debugging	Code Compilation, complex Debugging, and Help highlighting	Code Compilation, Run Code, Debugging, Help highlighting	Code Compilation, Run Code, Debugging, Help highlighting, Team Collaboration

c. Trade Results The results of the trade study using the above metrics to judge three options for IDEs is shown below. The narrowed down options for IDEs were Eclipse C++ for Arduino, KubOS, and the Arduino IDE. All of these options were described more in detail in Section 4.

Table 21. Trade Study Results for the Flight Software IDE

Metric	Weight	Eclipse C++ Arduino	KubOS	Arduino IDE
Complexity	0.15	3	1	4
MCU compatibility	0.3	2	4	2
OS Compatibility	0.1	3	3	4
Popularity	0.2	2	4	3
Programming Languages Support	0.05	4	4	4
Features	0.2	3	3	1
Total	1	2.55	3.25	2.6

After using the decided upon criteria to score the Eclipse, Arduino, and KubOS IDEs, it was concluded that the best choice was KubOS. The KubOS operating system scored well on all metrics considered, aside from complexity. As was discussed in the informational paragraph about KubOS, its biggest downside is the unfamiliarity of the program to all the members of the group. KubOS was scored a 1 in complexity because it is believed that although the team has no experience to start with in KubOS, it was intuitive and would be learned swiftly. The KubOS IDE received a 4 in the MCU compatibility metric due to its extensive libraries and plug-ins accessible by the MCU, along with all the communication compatibility requirements. Continuing on KubOS received a 3 in OS compatibility because it works with Mac OS, Windows, or Linux, which are available operating systems usable to the team. The KubOS IDE got a 4 in popularity due to the customer wanting it to be used, and only good ratings were found from those who have used KubOS. KubOS also got a 4 in the Programming languages metric as it simply supports the desired language to create the software development with. Finally, KubOS received a 3 in Additional Features because it surpassed the level 3 requirements but unfortunately did not have a team collaboration component to it, failing to fulfill that requirement. Overall KubOS was found through this trade study that it will work the best for the purposes of this project. Although complex, it has a large amount of resources, great compatibility and features that will make it ideal for this project.

Ground Software Appendix

Trade Studies

There were a few options considered for the ground software.

d. KubOS Because it was already being designed as a flight software to receive CCSDS packetized data, KubOS was considered as a possible ground software. Some of the pros and cons of KubOS are displayed below.

Pros	Cons
Built-in packetizing/depacketizing CCSDS formatted data	Not created to act as a ground software
Can receive and transmit data through a UDP/TCP port	No graphical user interface
Already being used for FSW so minimal extra work involved	Would require another PCB Board on the ground side

Table 22: Pros and Cons of KubOS as a ground software

In the end KubOS did not fulfill all the requirements needed for a ground software as it lacked the interface and full scope of a ground system software program. KubOS was decided against as the ground software.

e. Creating an original program One of the options with the best ability to meet the needs of the ground software requirements was to write the code for an original program using C++ or Matlab. Some of the pros and cons considered for this option are shown below.

Pros	Cons
Could be written specifically to fulfill the requirements of the project	Would be extremely difficult and time consuming
Would be very interesting and educational	Out of the scope of the purpose of the project
	Not guaranteed to work

Table 23: Pros and Cons of creating an original ground software

Because creating an original ground software program was not a requirement and would be a huge amount of extra and unnecessary work, this option was decided against.

f. COSMOS As an open source program with a lot of useful applications, COSMOS seemed to satisfy all the necessary requirements. Some pros and cons are shown below for COSMOS.

Pros	Cons
Command Sender application sends command packets to a specific target	Configuration code is written in Ruby so extra work required
Can receive and transmit data through a UDP/TCP port	
Telemetry Grapher and Extractor applications can graph data on a GUI and store it	

Table 24: Pros and Cons of COSMOS as a ground software

g. Trade Study Results In the end COSMOS was chosen as the ground software because it is a very powerful program that easily fulfilled all the necessary requirements. Many of the built in applications were usable for the mission, so the design portion comes in the form of writing configuration files for the tools and creating specific commands to be tested.

Payload Board Appendix

Trade Studies

h. Microcontroller Unit The MCU must be able to interface with and provide all of the control functionality necessary for the operation of the subsystems. It needs to operate within the power budget set by the system's power supply and be able to drive additional power to the subsystems it is controlling. In addition, it must be capable of meeting all of the real-time processing requirements that are set for the system to function properly as a whole. Another important factor to consider when choosing a MCU is the development process. There must be a development kit available that fits within the budget and provides the necessary functionality to develop and test both the software and hardware capabilities of the MCU before the final hardware is constructed.

i. ATmega328 The ATmega is an 8-bit AVR architecture MCU that operates at 20MHz and is commonly found in low profile micro-controller boards. This includes many Arduino-type boards, which have the benefit of being a familiar prototyping tool that is easy to learn and use. It provides 23 GPIO pins. This option also provides the opportunity to develop software using the Arduino IDE and libraries, which greatly simplify the embedded programming process by abstracting from the low-level code needed for the MCU to operate. The power requirement for this MCU is 1.8V - 5.5V, 25mA.

Pros	Cons
Low level of complexity	Minimal computing power
Access to Arduino's IDE and libraries	Low number of GPIO pins
Good documentation	Low efficiency

ii. ATSAMD21G18 The ATSAMD21G18 is a 32-bit MCU based on the ARM architecture that operates at 48MHz. The advantages of using an ARM processor are that they are cheaper and more power efficient than other processors based on other architectures. The high availability and applications support for ARM processors make them a popular choice for embedded computing. This processor is available on an Arduino board, which comes with the prototyping benefits associated with using Arduinos. It also provides 38 GPIO pins. The power requirement for this MCU is 1.62V - 3.6V, 25mA.

Pros	Cons
High efficiency	Medium number of GPIO pins
Good documentation	Medium computing power
Access to Arduino's IDE and libraries	Medium level of software development complexity
	Medium level of hardware complexity

iii. AT91SAM3X8E The AT91SAM3X8E is another 32-bit ARM MCU that provides a step up in performance from the ATSAMD21G18 with a higher clock speed at 84 MHz and more than double the number of GPIO pins at 103. This MCU is also available on an Arduino board. The power requirement for this MCU is 1.62V - 3.6V, 25mA.

Pros	Cons
High efficiency	Medium level of hardware complexity
High number of GPIO pins	Medium level of software development complexity
Access to Arduino's IDE and libraries	
Good documentation	
High computing power	

iv. AM3358 The AM3358 as a 32-bit ARM MCU with enough processing power to support high-level operating systems such as Linux. It is enhanced with graphics processing peripherals and industrial interface options. It also contains subsystems separate from the ARM core that provide more flexibility in implementing fast, real-time responses, specialized data handling operations, custom peripheral interfaces, and in allocating tasks to different processor cores. This MCU's processing capabilities come at the cost of increased complexity and power requirements. The power requirement for this MCU is 3.7V - 5V, 700mA.

Pros	Cons
Ability to run high-level operating systems	Large power requirement
Good documentation	High level of hardware complexity
High efficiency	
Low software development complexity	
Very high computing power	
High number of GPIO pins	

i. Metrics and Weighting The metrics used to conduct the trade study below are described in Table 8. They were also given weight values according to their importance as defined by requirements. These measure the most significant features of the MCUs that are needed to make comparisons in order to select the optimal MCU for the system. We chose not to include other metrics like size, operating temperatures, and cost because all of the MCUs are nearly identical in these areas.

Table 25. Trade Topics and Weights for the MCU

Metric	Weight	Driving Requirement	Rationale
Architecture/Performance	0.4	4.1, 4.2, 4.3	The MCU must be able to meet all of the real-time processing requirements for our operation. It must have the architecture and performance capabilities to execute the required instructions before their deadlines.
Power Requirement	0.1	N/A	The MCU must operate within the power budget.
GPIO Pin Count	0.2	4.1, 4.2.1, 4.2.3	The MCU must be able to interface with all required subsystems
Development Kit	0.3	N/A	The MCU must be compatible with software developed before the final microcontroller hardware is finished with the help of a development board that is reasonably easy to use and within the project budget.

j. Metric Quantification Table 9 includes the details of how each metric is scored. The metric carrying the greatest significance in the trade study is the architecture and performance of the MCU. This is what will determine how capable an MCU is at meeting the real-time processing requirements of the system we will be designing. An MCU based on 8-bit architecture is significantly weaker and less efficient than an MCU based on 32-bit ARM architecture. A 32-bit ARM architecture will have increased processor register memory width and include five stage pipe-lining to further increase the speed that instructions are executed beyond the clock speed of the processor. Speed increases of up to 5 times are attainable over a comparable 8-bit architecture. The MCU's clock speed will determine the likelyhood that software instructions are executed by their deadlines, which are set by the requirements of the system.
The next metric considered is the power needed for the MCU to operate. This will influence the amount

of power the system needs to be supplied. An MCU with more efficient architecture lowers the power requirement, and increasing clock rates bring increasing power requirements.
GPIO pin count influences how many subsystems are able to be controlled by the MCU. More GPIO pins provide a less constrained control system that is able to interface with more subsystems.
The last metric considered was the development kit for the MCU. The availability of a development board to be purchased early in the development process is crucial in order to have the time to design the flight software and explore the capabilities of the MCU before a final hardware design is constructed. The cost of the development board needed to fit within the budget constraints of the project.

Table 26. Explanation of Analytical Scale for the MCU

Metric	0	1	2	3	4
Architecture/ Performance	8-bit architecture with a clock speed ≤ 20 MHz	8-bit architecture with a clock speed > 20 MHz	32-bit ARM architecture with a clock speed > 30 MHz	32-Bit ARM architecture with a clock speed > 80MHz	32-Bit ARM architecture with extra processor cores and a clock speed ≥ 1GHz
Power Requirement	Operating requirements greater than 5V, 200mA	Operating requirements greater than 5V, 100mA	Operating requirements greater than 3.3V, 50mA	Operating requirements greater than 3.3V, 40mA	Operating requirements less than 3.3V, 40mA
GPIO Pin Count	< 20 pins	< 30 pins	< 40 pins	< 50 pins	< 60 pins
Development Kit	A prototyping/development board is not available to purchase	A prototyping/development board costs more than $200	A prototyping/development board costs less than $200	A prototyping/development board costs less than $100	A prototyping/development board costs less than $70

Table 27. Trade Study Results for MCU Selection

Metric	Weight	ATmega328	ATSAMD21G18	AT91SAM3X8E	AM3358
Architecture/Performance	0.5	0	2	3	4
Power Requirement	0.1	0	4	4	0
GPIO Pin Count	0.2	1	3	4	4
Development Kit	0.2	4	4	4	4
Total	1	1.0	2.8	3.5	3.6

k. Trade Results The MCU trade study results show equal scores for the AT91SAM3X8E and the AM3358. Both of these MCUs have development kits available to be purchased at low cost and they both have a large number of GPIO pins available to use for interfacing with the various subsystems that require controls. The differences between these two MCUs are that the AM3358 is significantly more powerful and complex than the AT91SAM3X8E and as a result, it requires more power. The advantage of the AM3358's computing power is that it is capable of running a Linux kernel; something none of the other MCUs are capable of. This is a significant factor that lowers the difficulty of developing flight software because the operating system abstracts from the hardware and thus, allows the use of high-level and diverse programming. This advantage is well worth the increased power requirement that comes with the AM3358.

Payload Board PCB Schematics

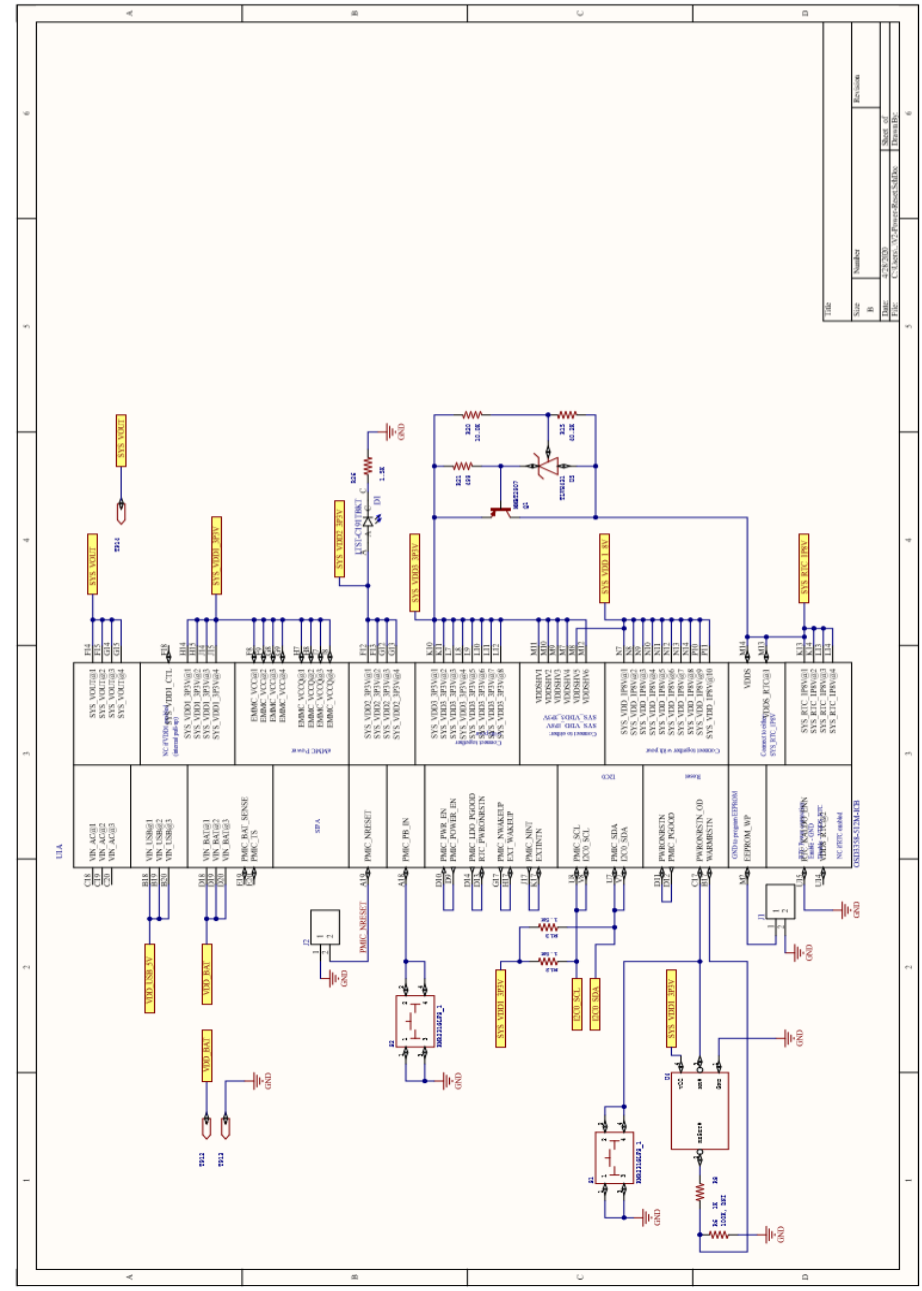


Figure 61. Power Electronics

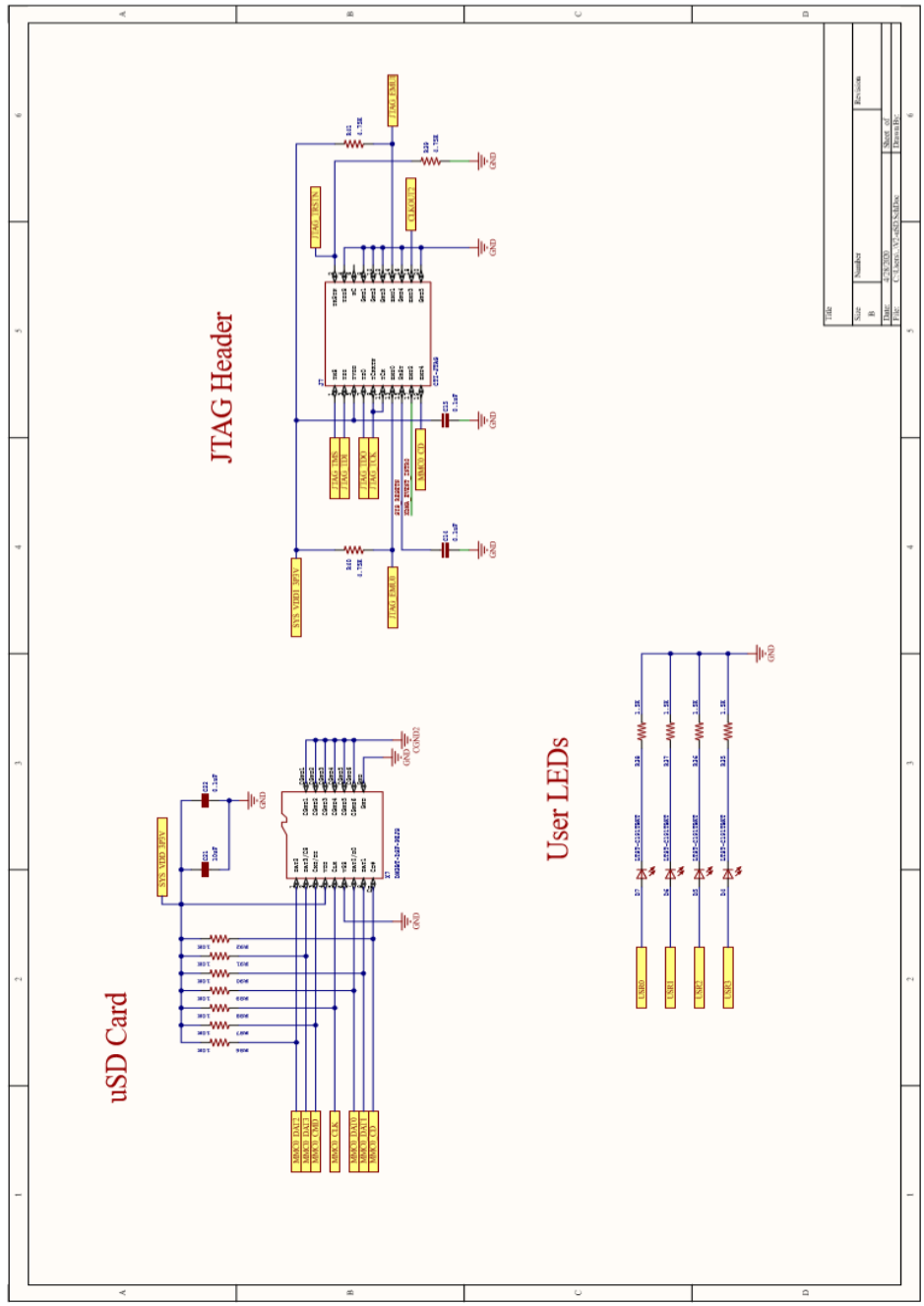


Figure 62. I/O and Debug Electronics

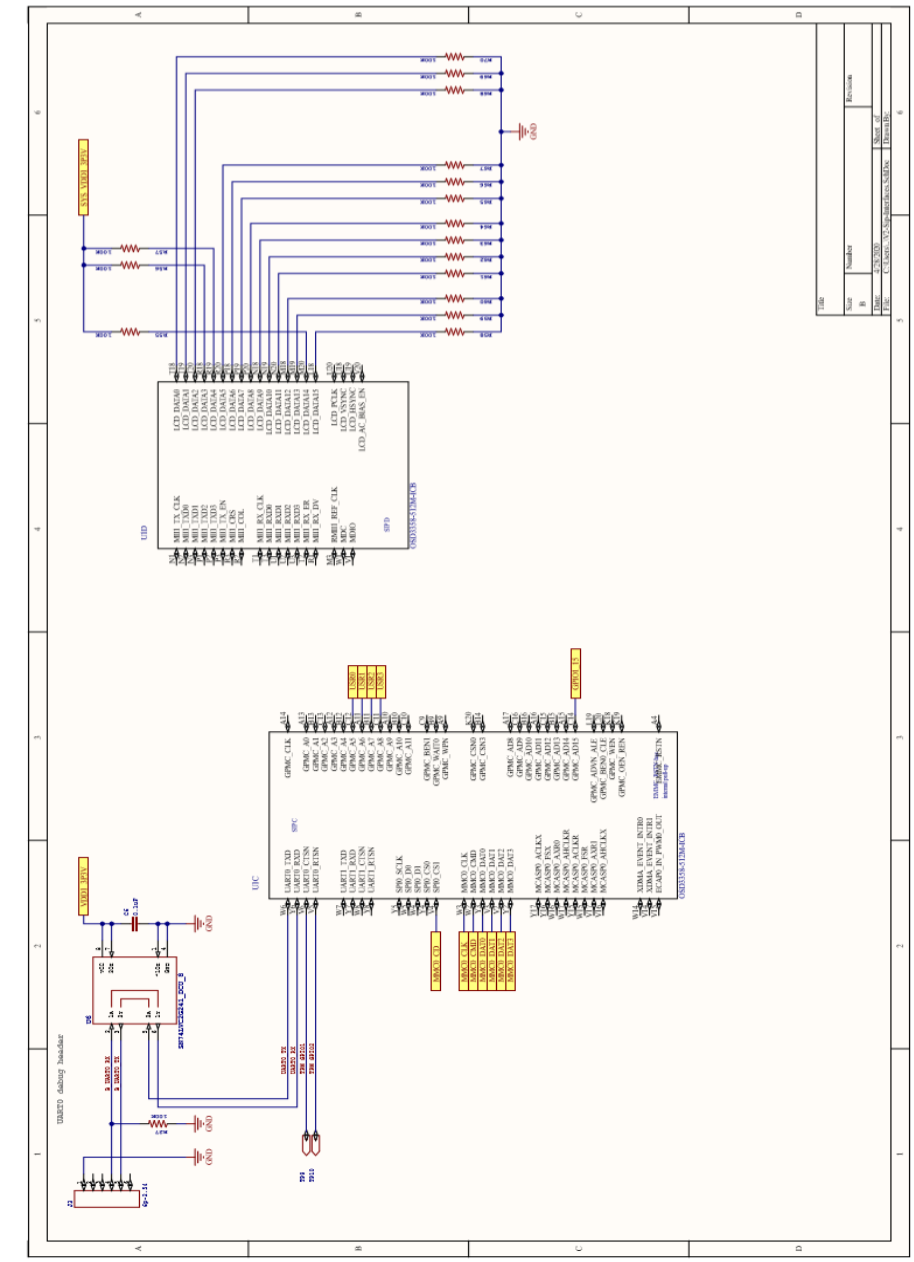


Figure 63. UART and Boot Configuration

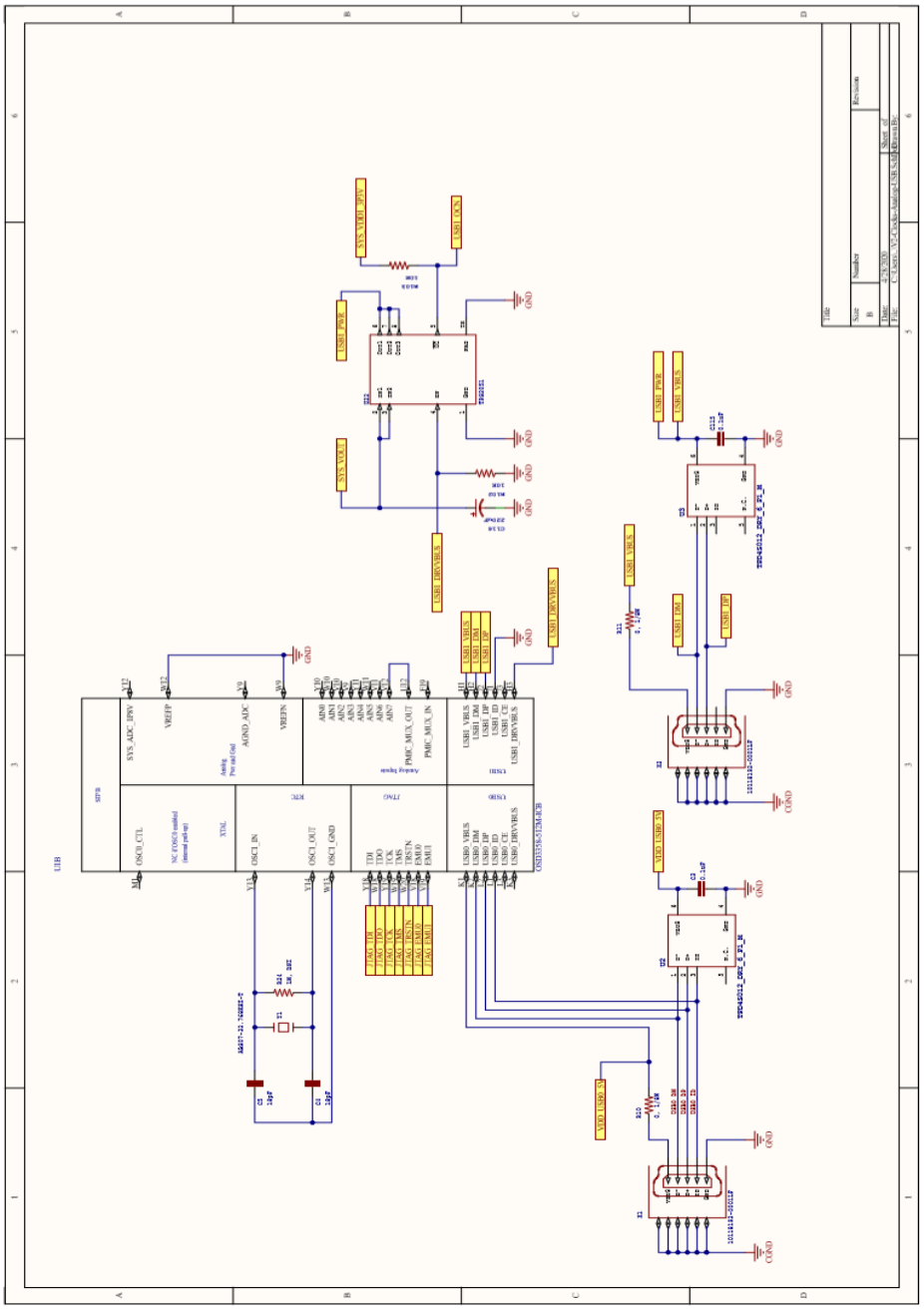


Figure 64. USB and Clock

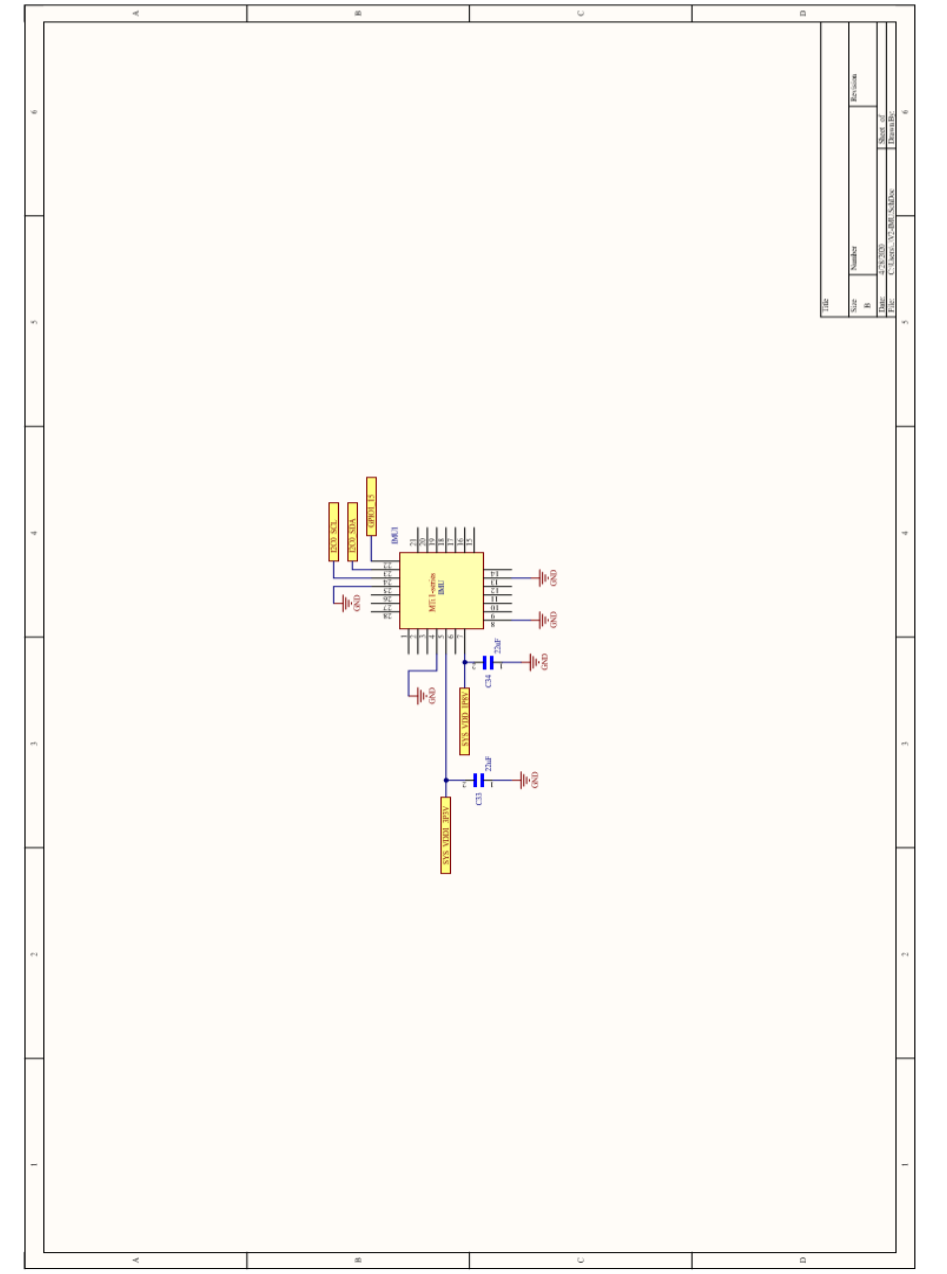


Figure 65. IMU

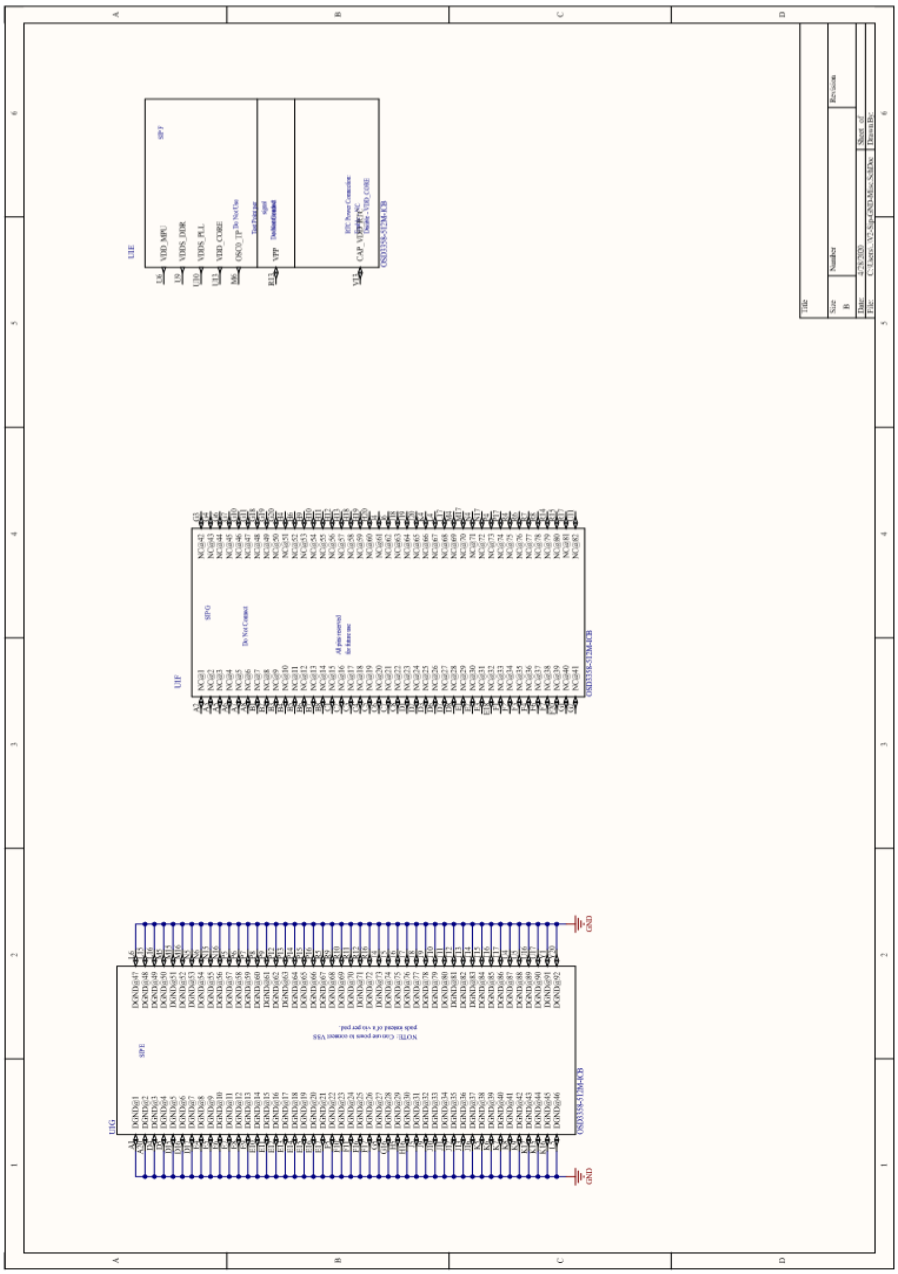


Figure 66. Ground Connections

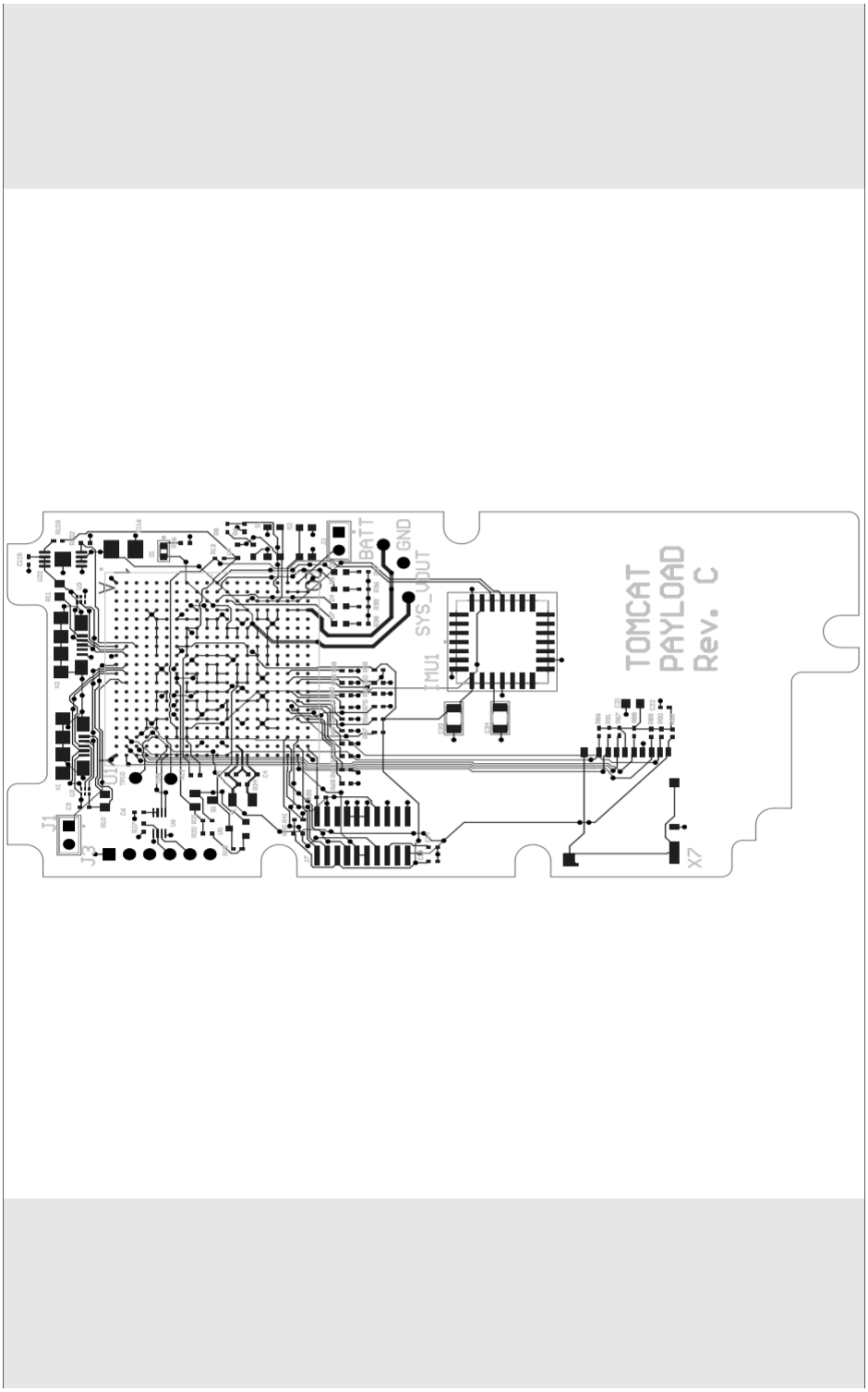


Figure 67. Payload Board PCB Layout

10.2. Comm PCB Schematics

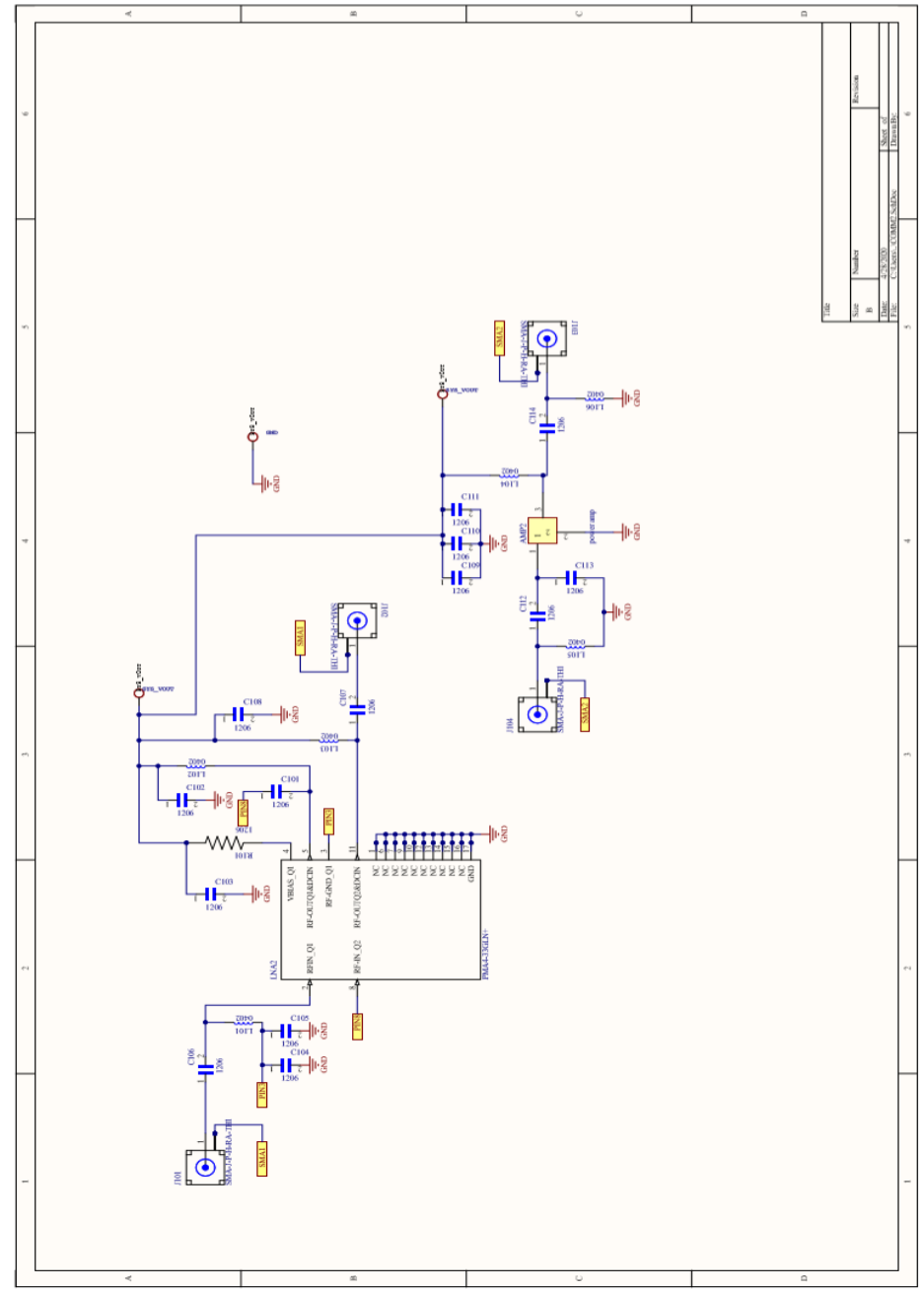


Figure 68. Comm PCB Schematic

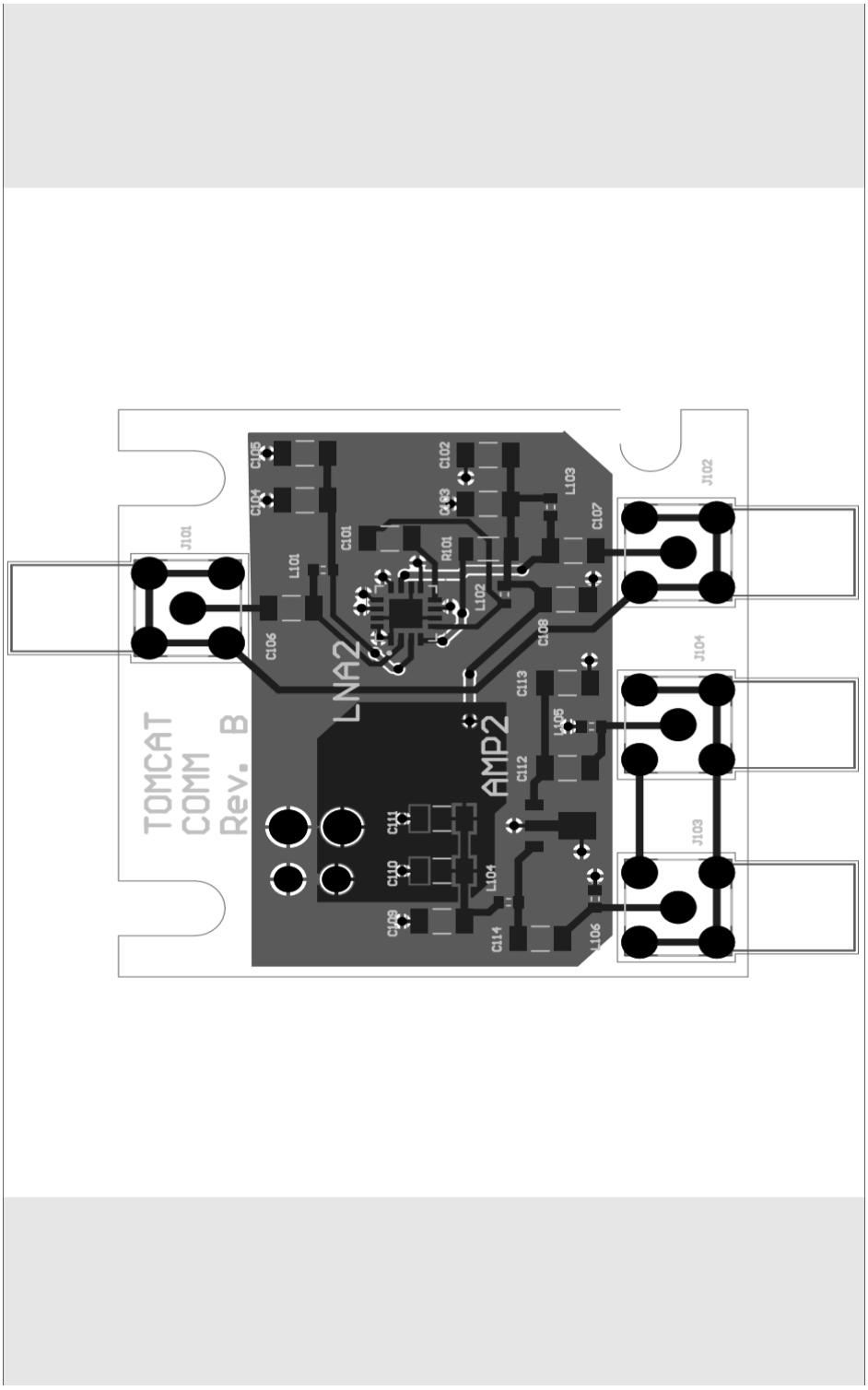


Figure 69. Comm PCB Layout

Power Appendix

Jake Schroeder

Trade Studies

While a power supply is available from the ThinSat Bus, the power required from the SDR alone is several times that which is supplied. A secondary or completely standalone power supply enables the ability to power on-board computers that are able to take advantage of more powerful flight software, such as those built off Linux like KubOS. The power supply, like the SDR, does not fall within the weight and size constraints of the original ThinSat payload, but does fall within the new mass requirements specified for the Gateway to Space Launch and the size requirements for the double ThinSat design being used for the high altitude balloon flight. The battery was chosen based on its feasibility for use in a near-space environment and its ability to fulfill power requirements of each of the subsystems for a two hour high altitude balloon flight.

a. Power Supply Types When determining the type of power supply required, it was important to distinguish between the scope of the project to launch on a balloon satellite and the ultimate goal of launching into Low Earth Orbit. One commonly used power supply option for satellites of any size in orbit is a combination of solar panels and batteries since they allow for missions to be recharged almost indefinitely. While solar panels may be useful in the future for use on the payload of ThinSats when they are launched into orbit, in order to operate on a high altitude balloon flight for a maximum of two hours, no long-term power supplies are required. Even for ThinSats launched into extreme low Earth orbit, the typical lifespan is only around five days, meaning batteries could still potentially be useful without the help of generating power from the sun. While there are other energy storage options, nearly all high altitude balloon payloads use some form of battery due to their high energy density, low cost, simplicity, ease of use, and reliability. There are several types of chemical batteries that were considered. For a balloon satellite mission, the most important aspects of a battery considered were operational temperature, physical dimensions, power output, energy density, and feasibility in a vacuum environment.

i. Alkaline Household alkaline batteries are proven to be flight tested and reliable, with a large temperature range from -20°C - 55°C. As with all chemical batteries, power output decreases as temperature decreases. These batteries are inexpensive and are widely available at general stores. Their primary drawback is their inability to be recharged. For single missions this is acceptable, but becomes more difficult when the payload needs to be tested since every test will require a new set of batteries. These batteries are also less power dense with respect to volume compared to Lithium Ion and Lithium Polymer batteries, so physical dimensions were important to take into account when considering Alkaline batteries.

Pros	Cons
High energy:weight ratio	Not rechargeable
Large temperature range	New batteries required for every test
Flight Heritage	Lower energy:volume ratio

ii. Lithium Polymer Lithium polymer batteries are generally very similar in cost and performance to lithium ion batteries, and are commonly used in rechargeable portable electronics. Lithium polymer batteries offer similar life cycles, charging/discharging safety, and performance at lower temperatures. In the past, Lithium Polymer batteries have been used on a number of CubeSat flights as well as previous CU Senior design projects such as RAVEN [13] and ELSA [14], so several resources for similar missions could be used for reference.

Pros	Cons
Rechargeable	Smaller temperature range
Lightweight, flexible casing	Slightly more susceptible to damage from external pressure
High energy density	

iii. Lithium Ion Lithium ion batteries are most commonly found in hand-held electronics such as smart phones and are relatively inexpensive. They have a large operational temperature range from -20°C - 60°C, but decrease in effectiveness as temperature decreases. Lithium ion batteries have slightly higher power densities compared to Lithium Polymer batteries, and while they have similar operating temperature ranges, Li-Ion batteries tend to remain only 55%-66% effective at -20°C compared to 70% effective for lithium polymer batteries. Most commonly, Lithium Ion batteries come in cylindrical cells, which are more power dense, but utilize the available volume inside the ThinSat less efficiently. Additionally, there is an extensive body of research on Lithium Ion batteries detailing the effects of temperature, discharge rate, and other variables because of their wide use in consumer electronics.

Pros	Cons
Rechargeable	Cylindrical Volume
High Energy Density	Less prone to external pressure damage
Large Body of Research	

10.3. Further Discussion on Battery Selection

10.3.1. Margin and Built in Capacity

It is important to note that the battery capacity required by the components alone is 15.91 Wh, but the power requirements for the system as a whole at high altitude conditions may be more. Due to power conversion inefficiencies, small unaccounted for power draws, and losses in battery capacity due to low temperatures, built in additional battery capacity is required.

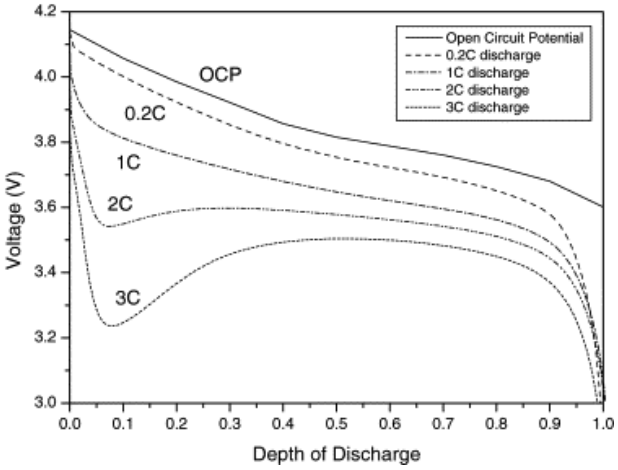


Figure 70. Voltage vs Depth of Discharge at Varying Discharge Rates [3]

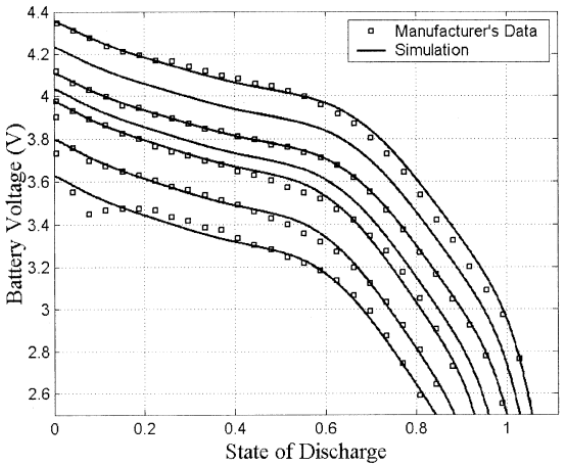


Figure 71. Voltage vs Depth of Discharge at Varying Temperatures [4]

Figures 70 and 71 plot the voltage of a lithium ion battery over the course of its depth of discharge. Depth (or state) of discharge describes the percentage of charge that has been removed from the battery,

meaning at depth of discharge of 1.0, the battery has been fully discharged. Both of these plots show that as the battery is discharged, voltage decreases, which requires the use of DC/DC converters to keep the voltage to each of the components constant.

Figure 70 plots the voltage output of the lithium ion battery at different rates of discharge (C rates), where the C rate is defined as the percentage of the battery discharged per hour. Given that the TOMCAT flight is 2 hours, and assuming that the battery is completely discharged in that amount of time, the C rate of the TOMCAT battery is a maximum of 0.5. As shown in the figure, lower C rates allow for the battery to output at higher voltages, and therefore allow for a greater power output. With a relatively low C rate of less than 0.5, the battery for the TOMCAT mission does not need to add built in capacity for high discharge rates.

Figure 71 shows the voltage output of a lithium ion battery versus state of discharge at varying temperatures. From top to bottom, the temperatures plotted are: 45°C, 34°C, 23°C, 10°C, 0°C, -10°C, and -20°C., at a discharge rate of 0.7 C. In this plot, the battery is discharged in each test until it reaches 2.5 V. From this state of discharge plot, the effective depths of discharge for a lithium ion battery discharged to 3.0 V at varying temperatures can be gathered. Table 28 below takes the data from figure 71 and shows the effective battery capacity of a lithium ion at decreasing temperatures.

Temperature	Battery Capacity
10°C	83%
0°C	80%
-10°C	70%
-20°C	65%

Table 28. Effect of Temperature on Battery Capacity [4]

From the first thermal test discussed above, it was shown that the temperature of the battery could reach as low as -10°C with insulation. Since the mission will not be employing the use of active thermal control due to its increased complexity, the battery must be able to power the mission even at very low temperatures. At -10°C, the battery is still within its operational range, but at a cost to its effective battery capacity. From table 28, it is expected that at this temperature, the battery is capable of discharging to 70% its specified capacity. Therefore, an additional 30% capacity will be built into the power budget.

Since power converters will be employed to maintain a constant voltage for each of the ThinSat components, a small power loss due to DC/DC conversion must be accounted for. The efficiency of a DC/DC converter is typically around 95%, but depends on the voltage received from the battery, which changes over time. To account for conversion losses, a 10% capacity will be built into the power budget. Furthermore, an additional 10% capacity will be built into the power budget to account for any other power losses, such as increased demand from individual components, and power draw from the payload board. These additional battery capacity accommodations are shown in the table below.

Accommodation	Required Additional Capacity
Low Temperatures	30%
DC/DC Converter Efficiency Losses	>10%
Other Efficiency Losses	10%
Total:	**50%**

Table 29. Calculation of Additional Battery Capacity Accommodations

10.3.2. Power Distribution

Figure 72 shows the connections and power requirements of each of the on-board and ground station components.

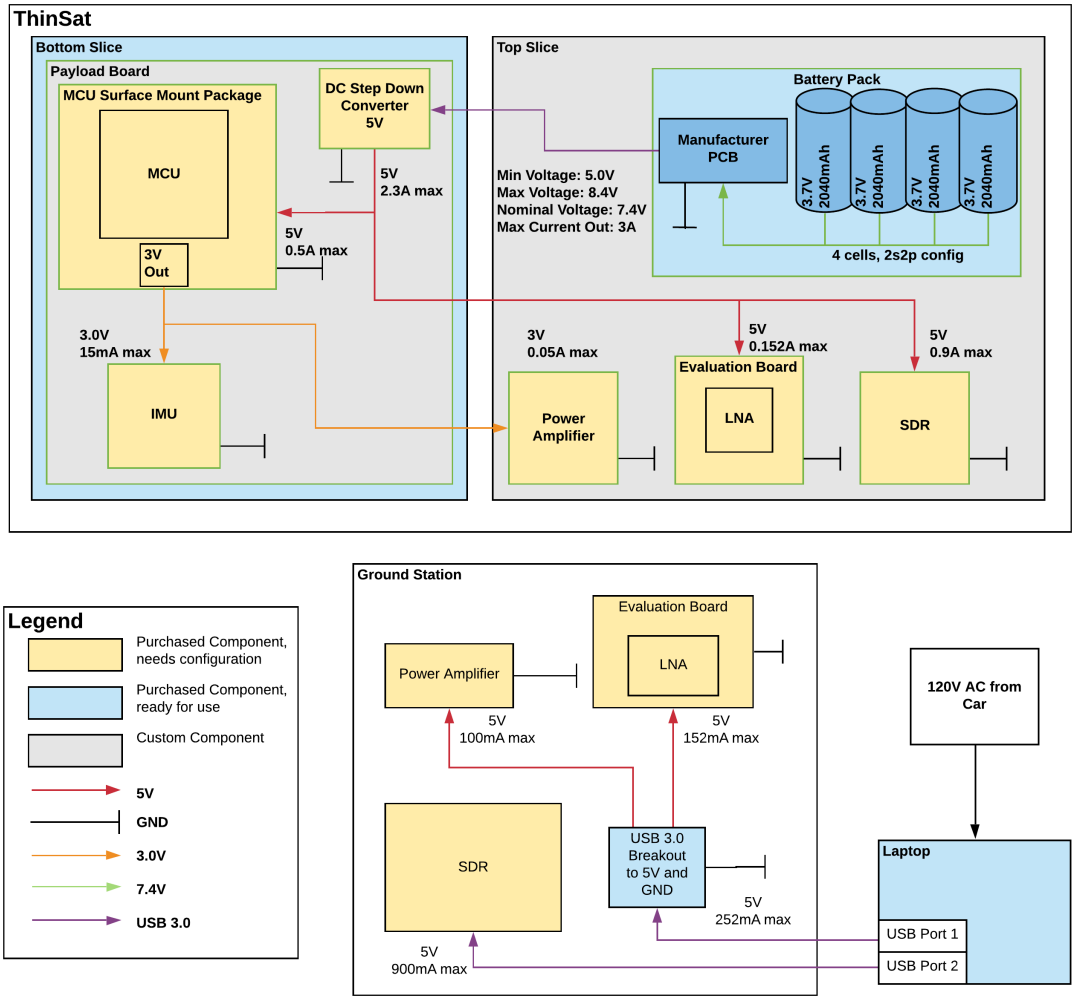


Figure 72. ThinSat and Ground System Power System FBD

a. Manufacturer Provided PCB To prevent damage to the payload components and the battery itself, a PCB provided by the battery manufacturer will be included attached to the battery. The PCB allows for the batteries to operate in a two series, two parallel configuration, increasing the output voltage of the battery to 7.4V. The PCB allows for safe discharging, with a fuse that disconnects the battery above 3A, which is much greater than the current expected to be drawn at any time during operation. The PCB also regulates battery voltage, preventing the pack as a whole from discharging below 5V, which keeps each individual cell from decreasing below 2.5V, and it allows for safe charging and discharging of the pack in its 2s2p configuration without needing to separate the batteries during charging. It was important to find a PCB that was included with the battery as designed by the manufacturer, rather than to design a fully custom PCB for the battery, which would have an increased likelihood of possibly damaging the battery or other components.

b. Payload Board Power Regulation The payload board, which houses the microcontroller unit and inertial measurement unit, as well as all the power distribution connections, will connect the battery to a 5V DC/DC step down converter capable of supplying 2.3A to the microcontroller surface mount package, the low noise amplifier evaluation board, and the software defined radio. There is a built in 3V out pin on the microcontroller surface mount package, which will be used to provide power to the inertial measurement unit and the power amplifier. The DC/DC step down converters ensure that each component is delivered the proper voltage, and allow for the required current draws. Since the battery is regulated by its own PCB to disconnect once the pack reaches 5V, no step up converter is necessary, which improves the overall efficiency of the power distribution system.

Science Appendix

Trade Studies

Science Sensor Type Selection

Metrics and Weighting The science sensor selection portion of this endeavor required a two-step process in order to determine the baseline design. First, it was necessary to compare the capabilities of different types of sensors considered in Section 4.3. The sensors were juxtaposed using the metrics and weights provided below, in Table 30.

Table 30. Trade Metrics for the Science Sensor Selection

Metric	Weight	Driving Requirement	Rationale
Power	0.14	1.3	Power will be coming from an external source chosen by the team. Though this allows for some room in terms of power usage, it is still best practice to use as little power as possible, in the event that future missions will be using power from the ThinSat bus. Because power is less of an issue, it has not been weighted the highest.
Size	0.17	1.1, 1.1.1	The ThinSat payload area is very small ($54.8 cm^2$), and the team must be able to fit the payload computer, science sensor, and any thermal protection and structural support within the small area. This size constraint merits a higher weight to ensure that the sensor(s) do not violate the payload size requirement.
Survivability	0.17	3.1, 3.3	In order to have a successful balloon flight, the science payload must be able to withstand the extreme environments of near space, this means very low temperatures for a prolonged amount of time, and very low pressure. In order for mission success, the sensors need to be taking data and operating properly at all points of flight. It is very important that the science sensor is operational throughout the entire mission, as the data sent from the sensor will be sent as telemetry, verifying the COMM system capability. This metric is weighted higher because of this.
Cost	0.12	5.1	In order to allow as much room as possible for the COMM system and the payload computer, the lower the cost of the science sensor, the better. The weight is given based off of the fact that many sensors are not of concern when it comes to cost, though more expensive options may allow for better data resolution, lower power consumption, smaller size, etc.

Documentation	0.15	N/A	Documentation will be key in understanding how the sensors work and how they can be integrated with other boards. This will come in handy when writing software for the sensor that allows it to interface with the payload computer. Documentation is rated high due to its importance in understanding how the science sensor will operate.
Scientific Merit	0.12	N/A	Raytheon has tasked the team with developing ideas for a mission that can be translated to a future space mission. This means choosing science that can be done on a balloon and in space. Though scientific merit is desired, the most important aspect of the mission is to generate data that can be sent as telemetry in order to verify and validate the COMM system, therefore, scientific merit is weighted lower.
Data Processing Requirements	0.13	2.3, 4.2.1, 4.2.2	The amount of data processing must be taken into account when selecting a sensor. Some types of sensors require more extensive on-board processing before data can be sent and received by the ground. The payload computer and the SDR must be able to handle the data rates from the sensor. The lower the data rates and the less onboard processing required, the simpler the scientific aspect of the mission becomes, and more focus can be placed on developing the payload computer and COMM system. This metric is moderately rated since most sensors did not have problematically high data rates, though some types required more data processing than others.

With the parameters for the trade study defined, it was necessary to determine the scoring subdivisions for each metric through an objective lens. The following table details what attributes constitute what scores for each of the metrics. The size category works around a maximum value of 12.5 cm^2, equal to half the total payload size. For realistic operations, this size acted as a feasibility ceiling value. Anything larger potentially compromises the space requirements of the thermal insulation and on-board computer. The power category is limited by the ThinSat bus payload maximum, and thus, a "zero" score in that category represents anything outside the capabilities offered by the bus.

Table 32. Weights for Science Sensor Selection

Metric	0	1	2	3	4
Power	>100mA and/or greater than 5V	≤ 50 at 3.3V or 5V	≤ 10mA 3.3V or 5V	≤ 5mA 3.3V or 5V	<1mA 3.3V or 5V
Size	$> 12.5cm^2$	$\leq 12.5cm^2$	$\leq 9cm^2$	$\leq 6cm^2$	$\leq 3cm^2$
Survivability	Sensor can survive temperatures 0 Celsius and above	Sensor can survive temperatures -5 and above	Sensor can survive temperatures -10 Celsius and above	Sensor can survive temperatures -30 Celsius and above	Sensor can survive temperatures -40 Celsius and above
Cost	>$300	≤ $300	≤ $200	≤ $100	< $50
Documentation	Little to no documentation or product page only	Bare minimum details provided, such as size, mass, and power requirements	Datasheet available with at least absolute maximum operating values and schematic.	Detailed datasheet available	Highly detailed datasheets, software documentation, external resources available
Scientific Merit	Does not provide data that can be applicable in a space mission and/or is not relevant for a balloon flight		Data is applicable for balloon flight, may be able to be translated to a space mission		Data is directly translatable to a space mission and also gives valuable results for a balloon flight.
Data Processing Requirements	Requires onboard processor running at 200 MHz or greater and/or data requires extensive post-processing	Requires onboard processor running at 160 MHz or greater and/or data requires extensive post-processing	Requires onboard processor running at 130 MHz or above and/or requires extensive post-processing	Requires onboard processor running at 95 MHz or above and/or data may require some post-processing	Requires onboard processor running at 60 MHz or less and/or data requires minimal post-processing

Trade Results - Sensor Type In conducting the trade study, both the imaging sensor and dosimeter received low scores, mostly due to their higher size and power requirements, both relatively highly weighted metrics. The environmental sensor came in very close to the magnetometer and and IMU, however it's applicability in space is limited and the sensor type was therefor not selected. Both the IMU and the magnetometer scored very close to each other, and since many IMUs can be purchased with a magnetometer built in, it was selected as our primary science instrument. It only scored poorly on cost, mostly because the price range of IMUs that were considered varied widely. Additional information of specific IMU models is discussed in Section b.

Table 33. Trade Study Results for Science Sensor Selection

Metric	Weight	Imaging Sensor/-Camera	Magnetometer	IMU	Environmenta Sensor	Radiation Sensor/-Dosimeter
Power	0.14	0	4	3	4	2
Size	0.17	1	3	4	2	0
Survivability	0.17	3	4	4	4	2
Cost	0.12	3	2	2	4	1
Documentation	0.15	2	4	4	4	0
Scientific Merit	0.12	4	3	3	1	4
Data Processing	0.13	0	3	3	4	3
Total	1	1.82	3.34	3.37	3.30	1.61

Science Sensor Model Selection

From the science mission trade study, it was decided that an IMU would be used as the science sensor. This decision was made based on the fact that IMUs have magnetometers built in, and more data on the ThinSat's balloon flight could be taken using an IMU. This decision is explored further in Section 6. As a result, a secondary science sensor trade is conducted to choose an IMU model. The three sensors under consideration for this trade study are summarized below:

Table 34. Characteristics of Each IMU Sensor

	SparkFun IMU Breakout - MPU-9250	SparkFun 9DoF IMU Breakout - ICM-20948 (Qwiic)	Xsens MTi 1-Series
Current Draw	12 mA	3 mA	30 mA
Size	5 cm^2	5.88 cm^2	1.46 cm^2
Temperature Range	-40 to 85 C	-40 to 85 C	-40 to 85 C
Price	$14.95	$16.95	~$200
Documentation	Good	Good	Good
Magnetometer Resolution	15000 nT	150 nT	25 nT

1. **SparkFun IMU Breakout - MPU-9250**
   The SparkFun MPU-9250 contains a 9-axis MEMS sensor. The breakout board houses two chips: the MPU-6500, containing a 3-axis gyroscope and a 3-axis accelerometer, and the AK8963, containing a 3-axis magnetometer. The MPU-9250 breakout houses 11 plated through-hole pins around the border of the PCB. These pins include the power interface and the I^2C interface, which allows interface between this board and other boards, such as our payload computer. The MPU-9250 uses 16-bit Analog-to-Digital Converters (ADCs) for digitizing all nine axes. [**?**]

2. **SparkFun 9DoF IMU Breakout - ICM-20948 (Qwiic)**
   The ICM-20948 features a 3-axis gyroscope and a 3-axis accelerometer with four selectable ranges, and a 3-axis magnetometer. The breakout board contains a logic shifter and broken out GPIO pins. The chip itself is very low-powered with an I^2C interface. The chip also uses a digital motion processor that offloads the computation of motion sensing algorithms from the detectors.

3. **MTi 1-series**
   The MTi 1-series IMU features a 3-axis gyroscope, a 3-axis magnetometer, and a 3-axis accelerom-

eter. It contains an on-board 'sensor fusion' algorithm that provides rudimentary data filtering. All sensors are calibrated and tested prior to shipment. These sensors are small in size, low in power consumption, and have I^2C interfacing. The sensor can be bought with a development kit that is Arduino compatible and comes with a free download of the MT software suite, which can be used to program the MTi 1-series.

Table 35. Trade Metrics for the Sensor Model Selection

Metric	Weight	Driving Requirement	Rationale
Power	0.15	1.3	Though the IMU sensors chosen met the trade study metrics for the previous trade study, power is kept as a metric in order to emphasize that the lower power, the better.
Size	0.14	1.1, 1.1.1	The sensors chosen are all very small in size, however this metric is kept to emphasize the importance of having small sensors for such a small payload area.
Survivability	0.14	3.1, 3.3	The sensors chosen must be able to survive the low temperatures at 30km altitude.
Cost	0.17	5.1	In order to allow as much room as possible for the COMM system and the payload computer, the lower the cost of the science sensor, the better. Cost is given a higher weight due to the fact that the team wants to give as much room as possible for purchasing larger-impact items such as the SDR and payload computer supplies.
Documentation	0.2	N/A	Documentation will be key in understanding how the sensors work and how they can be integrated with other boards. This will come in handy when writing software for the sensor that allows it to interface with the payload computer. Documentation is rated high due to its importance in understanding how the IMU sensor will operate. Though there are many options for IMUs, some may have better documentation than others, and that should be considered when purchasing.
Magnetometer Resolution	0.2	N/A	The resolution of the IMU is important to consider when choosing the sensor. The magnetometer resolution was chosen as the metric due to the fact that each datasheet for each sensor under consideration had consistent ratings for resolution. A higher resolution sensor will give more accurate results and therefore more valuable results. The highest weight was placed on resolution due to the fact that most of the IMU sensors already fit the criteria set in the previous trade study, and though higher resolution comes with a higher price tag, it is something to heavily consider.

Table 36. Weights for Sensor Model Selection

Metric	0	1	2	3	4
Power	≥ 100mA and/or greater than 5V	≤ 50A at 3.3V or 5V	≤ 10mA 3.3V or 5V	≤ 5mA 3.3V or 5V	<1mA 3.3V or 5V
Size	$> 12.5cm^2$ (half the payload area)	$\leq 12.5cm^2$	$\leq 9cm^2$	$\leq 6cm^2$	$\leq 3cm^2$
Survivability	-10 Celsius and above	Sensor can survive temperatures -20 and above	Sensor can survive temperatures -30 Celsius and above	Sensor can survive temperatures -40 Celsius and above	Sensor can survive temperatures -50 Celsius and above
Cost	>$300	≤ $300	≤$200	≤ $100	<$50
Documentation	Little to no documentation or product page only	Bare minimum details provided, such as size, mass, and power requirements	Datasheet available with at least absolute maximum operating values and schematic.	Detailed datasheet available	Highly detailed datasheets, software documentation, external resources available
Magnetometer Resolution	>77.5 nT	<77.5 nT	<55 nT	<32.5 nT	<10 nT

Metrics and Weighting

Table 37. Trade Study Results for Sensor Model Selection

Metric	Weight	SparkFun IMU Breakout - MPU-9250	SparkFun 9DoF IMU Breakout - ICM-20948 (Qwiic)	Xsens MTi 1-Series
Power	0.15	2	3	1
Size	0.14	4	4	4
Survivability	0.14	3	3	3
Cost	0.17	4	4	2
Documentation	0.2	4	4	4
Magnetometer Resolution	0.2	0	0	3
Total	1	2.76	2.91	2.87

Trade Results For the first metric, power, the Xsens MTi 1-Series IMU scored the lowest, this is due to the fact that the sensor can draw up to 30 mA, which is on the high end of our scale. The SparkFun MPU-9250 received a score of two as it can draw up to 12 mA. The SparkFun ICM-20948 received a score of three as it draws the least current of the three sensors. All three IMUs considered received a 4 in size as they are all very small ($< 5cm^2$) and can easily fit on a small PCB that would not take up too much payload space. All three IMUs considered received a 4 for documentation. Each sensor has a very thorough datasheet and all come with some kind of code documentation as well.

The resolution of the sensor is very important if accurate results are sought. Neither SparkFun sensor had a resolution anywhere near as good as the Xsens IMU. The resolution on the Xsens magnetometer was

hundreds of times better than the magnetometers on the SparkFun IMUs. Therefore, both SparkFun sensors received a zero in this category while the Xsens IMU received a 3.

Design Specifications

Xsens IMU Data Types

The following table, provided in the Xsens Communications Manual, denotes all the data that the Xsens IMU 1-Series are capable of acquiring. In this particular case, the team integrated an MTi-3 AHRS module, which represents the third column. Note that this model has the added functionality of orientation data expressed in 3 formats and free acceleration data, making it a more desirable model as compared to the MTi-1. The Xsens IMUs also have a temperature sensor, which would have played an integral role in monitoring the health of the payload board and many nearby internal components of the TOMCAT system.

Group Name	Type Name	XDA type name	Unit	Hex Value	Valid for MTi product										Max frequency[5]
					1	2	3	7	10	20	30	100	200	300	710	
Temperature		**XDI_TemperatureGroup**		**08x0**	•	•	•	•	•	•	•	•	•	•	•	1 Hz
	Temperature	XDI_Temperature	°C	081y	•	•	•	•	•	•	•	•	•	•	•	
Timestamp		**XDI_TimestampGroup**		**10x0**	•	•	•	•	•	•	•	•	•	•	•	See note[6]
	UTC Time	XDI_UtcTime	N/A	1010	•	•	•	•	•	•	•	•	•	•	•	
	Packet Counter	XDI_PacketCounter	N/A	1020	•	•	•	•	•	•	•	•	•	•	•	
	Sample Time Fine	XDI_SampleTimeFine	N/A	1060	•	•	•	•	•	•	•	•	•	•	•	
	Sample Time Coarse	XDI_SampleTimeCoarse	s	1070	•	•	•	•	•	•	•	•	•	•	•	
Orientation Data		**XDI_OrientationGroup**		**20xy**		•	•	•		•	•		•	•	•	400 Hz MTi-2/3: 100 Hz
	Quaternion	XDI_Quaternion	N/A	201y		•	•	•		•	•		•	•	•	
	Rotation Matrix	XDI_RotationMatrix	N/A	202y		•	•	•		•	•		•	•	•	
	Euler Angles	XDI_EulerAngles	deg	203y		•	•	•		•	•		•	•	•	
Pressure		**XDI_PressureGroup**		**30xy**				•				•	•	•	•	50 Hz
	Baro Pressure	XDI_BaroPressure	Pa	301y				•				•	•	•	•	
Acceleration		**XDI_AccelerationGroup**		**40xy**	•	•	•	•	•	•	•	•	•	•	•	2000 Hz[7] MTi-1/2/3: 100 Hz
	Delta V	XDI_DeltaV	m/s	401y	•	•	•	•	•	•	•	•	•	•	•	
	Acceleration	XDI_Acceleration	m/s^2	402y	•	•	•	•	•	•	•	•	•	•	•	
	Free Acceleration	XDI_FreeAcceleration	m/s^2	403y		•	•	•		•	•		•	•	•	
	AccelerationHR	XDI_AccelerationHR	m/s^2	404y	•	•	•	•	•	•	•	•	•	•	•	~800-1000 Hz[8]
Position		**XDI_PositionGroup**		**50xy**				•							•	400 Hz
	Altitude Ellipsoid	XDI_AltitudeEllipsoid	m	502y				•							•	
	Position ECEF	XDI_PositionEcef	m	503y				•							•	
	LatLon	XDI_LatLon	deg	504y				•							•	
GNSS		**XDI_GnssGroup**		**70x0**				•							•	4 Hz
	GNSS PVT data	XDI_GnssPvtData	N/A	7010				•							•	
	GNSS satellites info	XDI_GnssSatInfo	N/A	7020											•	
Angular Velocity		**XDI_AngularVelocityGroup**		**80xy**	•	•	•	•	•	•	•	•	•	•	•	2000 Hz[9] MTi-1/2/3: 100 Hz
	Rate of Turn	XDI_RateOfTurn	rad/s	802y	•	•	•	•	•	•	•	•	•	•	•	

Xsens IMU Data Communication

The Xsens IMU uses a proprietary Xbus communication protocol which is broken down using the following table, as provided in the Xsens Communications Manual.

This packet format was to be decoded and transferred to a CCSDS packet by the flight software subsystem. This task had made some progress before all work was halted.

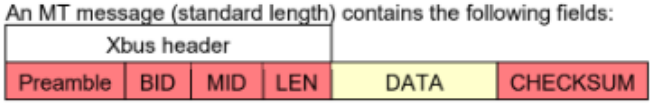
Xsens IMU Hardware Integration

The Xsens IMU is physically integrated to a printed circuit board using a plastic mounting system an electrically connected using a selection of its 28 pins. A pin diagram is presented below, as provided by the

	Delta Q	XDI_DeltaQ	N/A	803y	•	•	•	•	•	•	•	•	•	•	
	RateOfTurnHR	XDI_RateOfTurnHR	rad/s	804y	•	•	•	•	•	•	•	•	•	•	~800-1000 Hz[10]
Sensor Component Readout (SCR)		**XDI_RawSensorGroup**		**A0x0**					•	•	•	•	•	•	2000 Hz
	ACC, GYR, MAG, temperature	XDI_RawAccGyrMagTemp	N/A	A010					•	•	•	•	•	•	
	Gyro temperatures	XDI_RawGyroTemp	°C	A020					•	•	•	•	•	•	
Magnetic		**XDI_MagneticGroup**		**C0xy**	•	•	•	•	•	•	•	•	•	•	100 Hz
	Magnetic Field	XDI_MagneticField	a.u.	C02y	•	•	•	•	•	•	•	•	•	•	
Velocity		**XDI_VelocityGroup**		**D0xy**				•						•	400 Hz
	Velocity XYZ	XDI_VelocityXYZ	m/s	D01y				•						•	
Status		**XDI_StatusGroup**		**E0x0**	•	•	•	•	•	•	•	•	•	•	See note[11]
	Status Byte	XDI_StatusByte	N/A	E010	•	•	•	•	•	•	•	•	•	•	
	Status Word	XDI_StatusWord	N/A	E020	•	•	•	•	•	•	•	•	•	•	
	Device ID	XDI_DeviceId	N/A	E080					•	•	•	•	•	•	
	Location ID	XDI_LocationId	N/A	E090					•	•	•	•	•	•	

Figure 73. Xsens IMU Data Types

An MT message (standard length) contains the following fields:



An MT message (extended length) contains these fields:

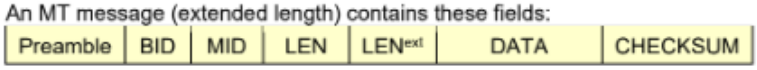


Table 3: Construction of an Xbus message

Field	Field width	Description
Preamble	1 byte	Indicator of start of packet → 250 (0xFA)
BID	1 byte	Bus identifier or Address → 255 (0xFF)
MID	1 byte	Message identifier
LEN	1 byte	For standard length message: Value equals number of bytes in DATA field. Maximum value is 254 (0xFE) For extended length message: Field value is always 255 (0xFF)
EXT LEN	2 bytes	16 bit value representing the number of data bytes for extended length messages. Maximum value is 2048 (0x0800)
IND ID	1 byte	The type of indication received
DATA (standard length)	0 – 254 bytes	Data bytes (optional)
DATA (extended length)	255 – 2048 bytes	Data bytes
Checksum	1 byte	Checksum of message

Figure 74. Xsens Xbus Packet Structure

Xsens Hardware Integration Manual. Note that the team selected to use the I^2C protocol for transferring information over the SPI and UART methods.

The pins most integral for the successful operation of the IMU were pins 5, 7, and 8 for power; pins 14 and 15 for setting the communication protocol to I^2C; and pins 22, 23, and 24 for communicating the clock and science data through the I^2C format.

The footprint of the Xsens IMU is given below, provided in the Xsens Hardware Integration Manual. Dimensions are in millimeters.

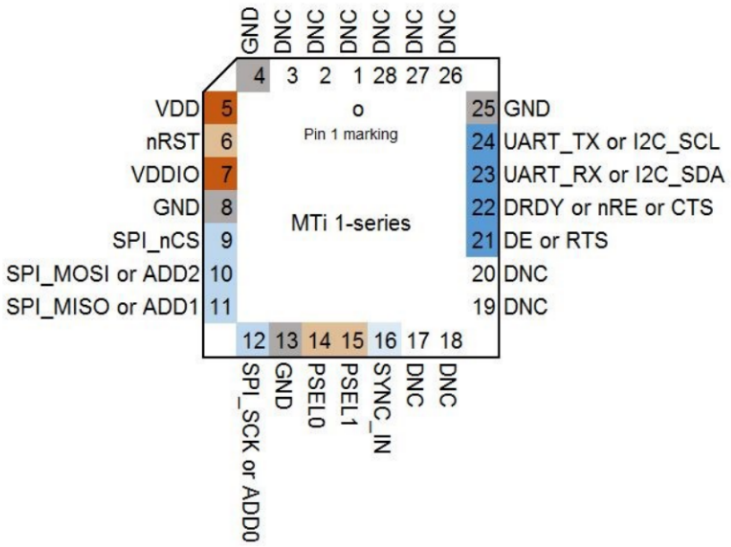


Figure 75. Xsens IMU Pin Layout

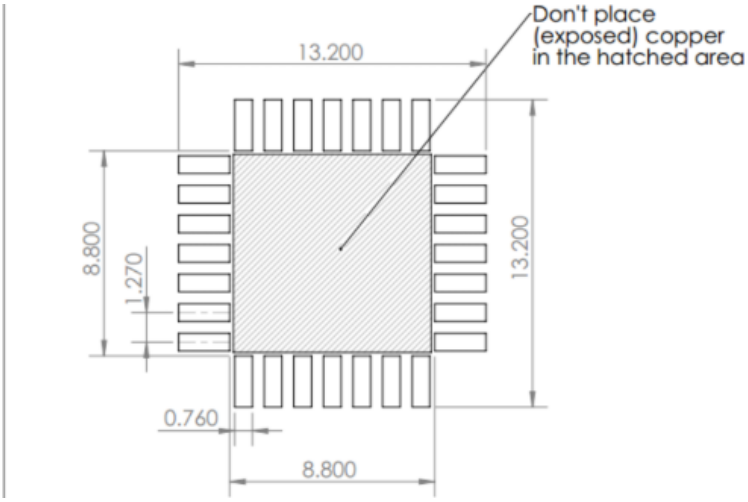


Figure 76. Xsens IMU Footprint

Manufacturing - Mission Code

```
% Austin Scheck - TOMCAT Science Lead
% ASEN 4028 Senior Projects
% Created January 15th, 2020
% Last Modified: February 13th, 2020
clear; close all; clc;

counter = 1;
fid = fopen('asen1400.txt');
tline = fgetl(fid);

%% DETERMINING AVERAGES FOR PRE-FLIGHT PHASE (0)
while ischar(tline)

    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if counter == 10
        meanground = mean(accx);
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = 0;
    tline = fgetl(fid);
    counter = counter+1;

end

%% DETERMINING ASCENT PHASE (1)
while ischar(tline)
    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if mean(abs(accx(counter-2:counter))) > 2*meanground
        phase(counter) = 1;
        disp(counter)
        % standard loop end procedure
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = 0;
    tline = fgetl(fid);
    counter = counter+1;
end

start_ascent = counter(end);

%% DETERMINING AVERAGES FOR ASCENT PHASE (1)

while ischar(tline)
```

```
    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if counter == start_ascent + 1000
        meanascent = mean(abs(accx(start_ascent:start_ascent+100))); % WATCH
OUT WITH ABSOLUTE VALUES GAMER
        phase(counter) = 1;
        % standard loop end procedure
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = 1;
    tline = fgetl(fid);
    counter = counter+1;

end

%% DETERMINING DESCENT PHASE

while ischar(tline)
    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if mean(abs(accx(counter-2:counter))) > 4*abs(meanascent)
        phase(counter) = -1;
        disp(counter)
        % standard loop end procedure
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = 1;
    tline = fgetl(fid);
    counter = counter+1;
end

start_descent = counter(end);

%% DETERMINING POST-FLIGHT PHASE
while ischar(tline)

    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if mean(abs(accz(counter-10:counter))) < 0.1

        phase(counter) = 0;
```

```
        % standard loop end procedure
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = -1;
    tline = fgetl(fid);
    counter = counter+1;

end
%% GROUND PHASE
while ischar(tline)
    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);

    if mean(abs(accx(counter-2:counter))) > 4*abs(meanascent)
        phase(counter) = -1;
        disp(counter)
        % standard loop end procedure
        tline = fgetl(fid);
        counter = counter+1;
        break;
    end
    phase(counter) = 1;
    tline = fgetl(fid);
    counter = counter+1;
end

start_descent = counter(end);

%% DETERMINING POST-FLIGHT PHASE
while ischar(tline)

    data = str2num(tline);
    time(counter) = data(8)/60;
    accx(counter) = data(6);
    accz(counter) = data(7);
    phase(counter) = 0;
     tline = fgetl(fid);
    counter = counter+1;
end


%% PLOTTING RESULTS
plot(time,accx)
hold on;
plot(time, accz)
plot(time,phase,'k','LineWidth',2)
xlabel('Time (minutes)')
ylabel('Acceleration (g)')
title('Acceleration & Flight Phase Over Time')
legend('Vertical Acceleration','Horizontal Acceleration','Flight Phase')
```

Figure 77. Mission Code MATLAB

Verification - Mission Algorithm

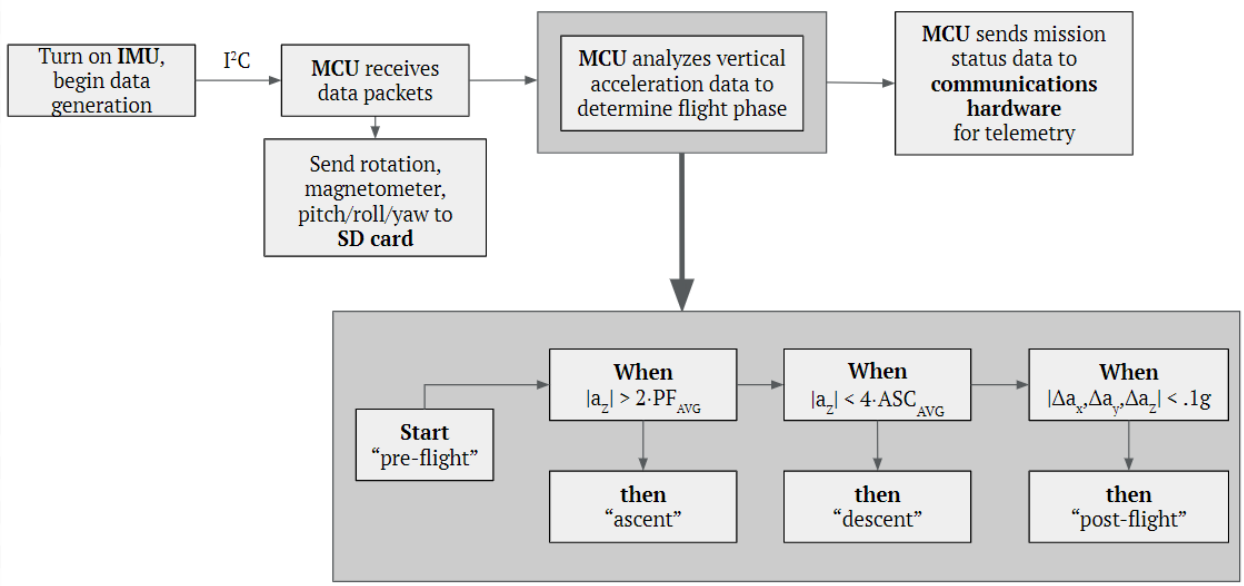


Figure 78. Mission Algorithm Flowchart

Structures Appendix

Hole Locations and Tolerances Drawings

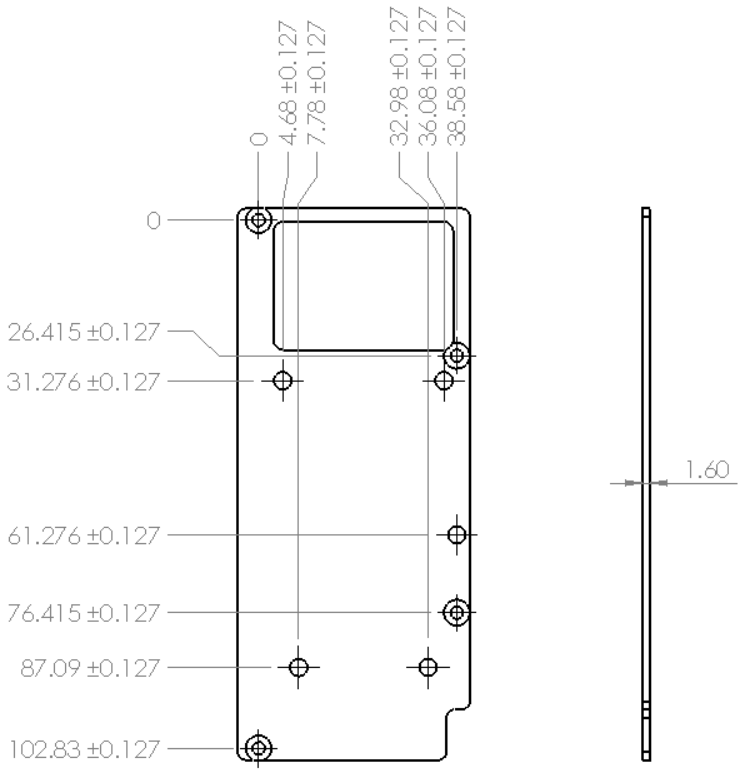


Figure 79. Communications Hardware Backplate

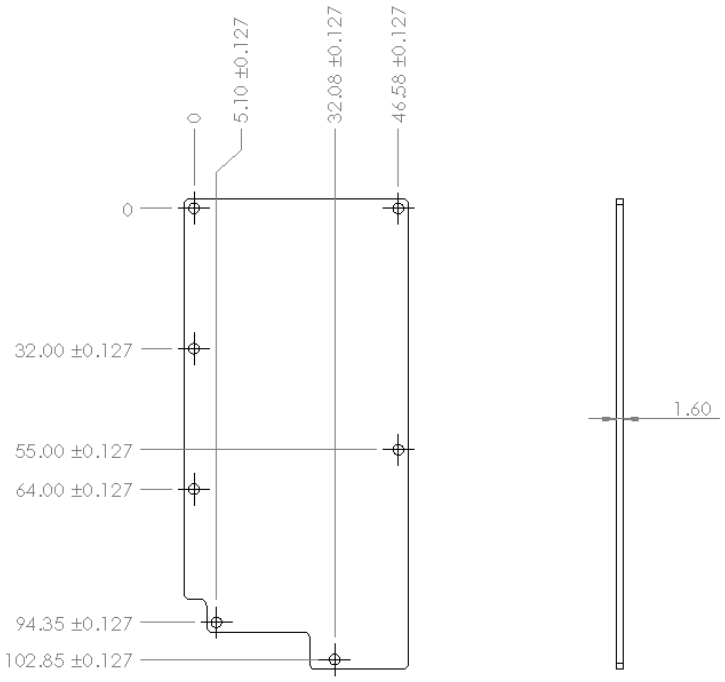


Figure 80. Payload Board Backplate

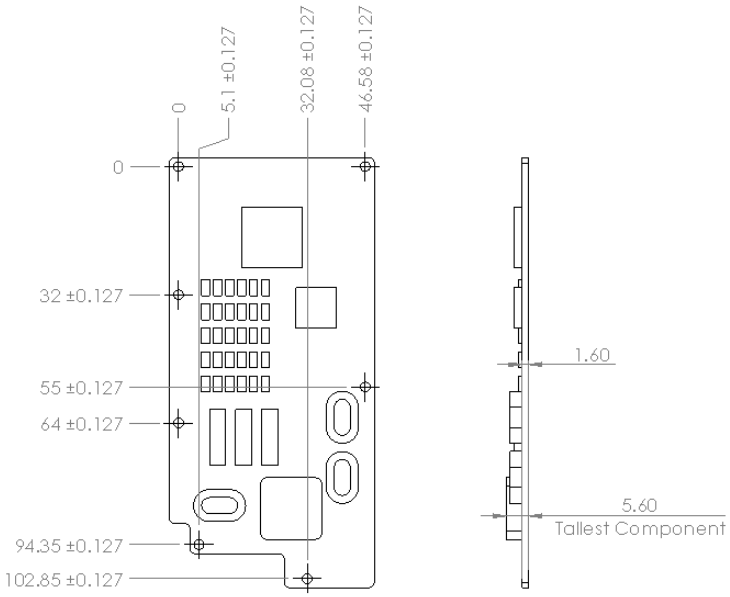


Figure 81. Payload Board

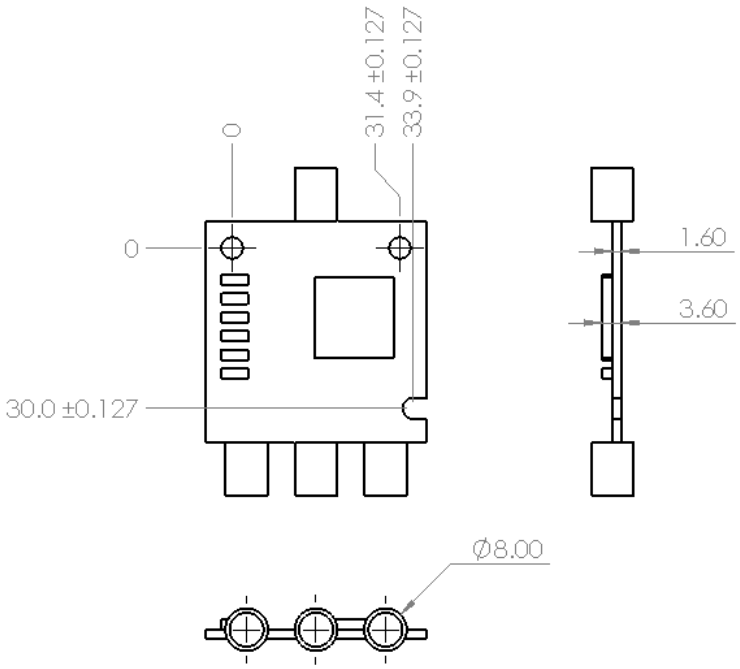


Figure 82. Communications Board

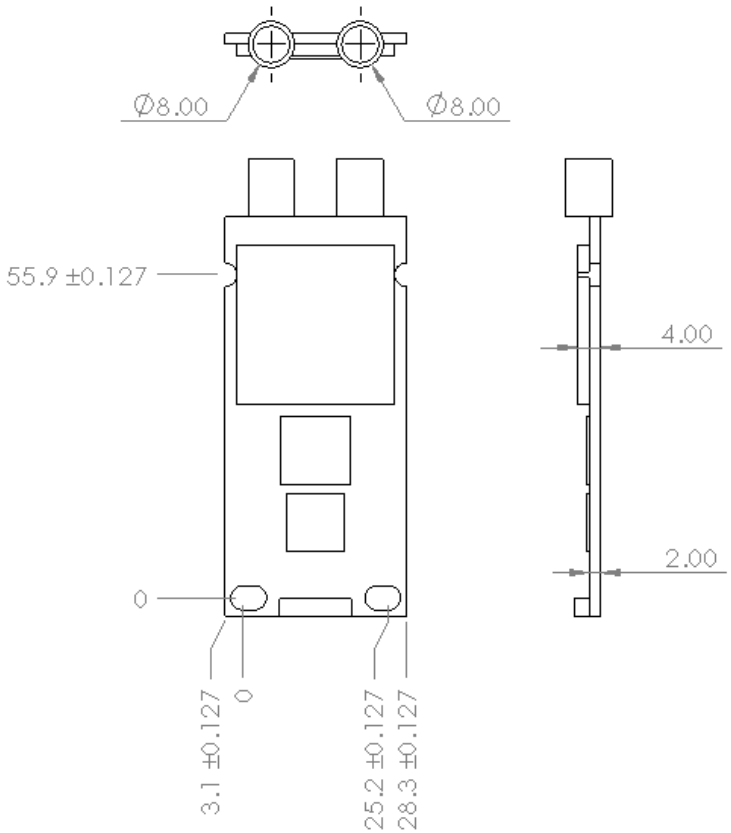


Figure 83. Lime SDR

Thermal Appendix

Preliminary Model Full Results

Figure 84 shows the results of the preliminary model when no insulation is added. These results show that some kind of thermal protection is needed or else the components inside the ThinSat reach temperatures below their operating temperature, resulting in possible failure of electronic systems during flight.

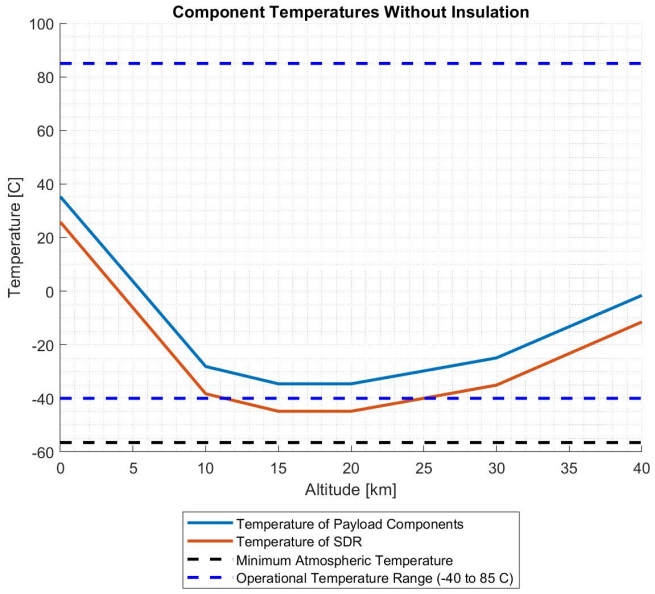


Figure 84. Model Results with No Insulation

Figures 85 and 86 show the results of the model when extruded polystyrene insulation is added.

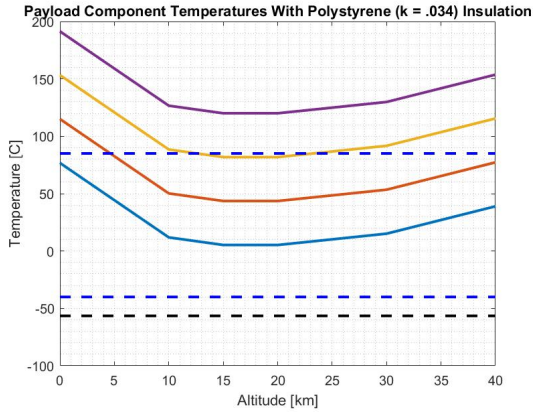


Figure 85. Model Results for Payload Area with Polystyrene Insulation

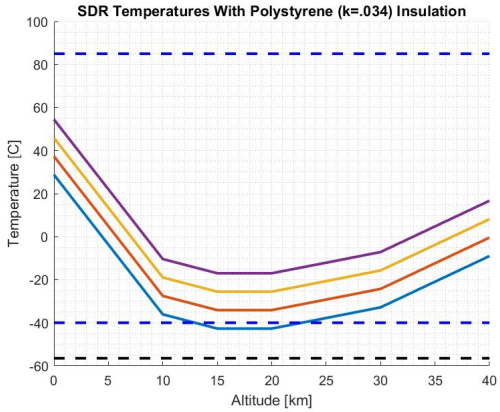


Figure 86. Model Results for SDR Area with Polystyrene Insulation

Figures 87 and 88 show the model results when aerogel insulation is added. It was determined that polystyrene insulation would be used due to its lower cost while still being an effective insulator for the purpose of this project.

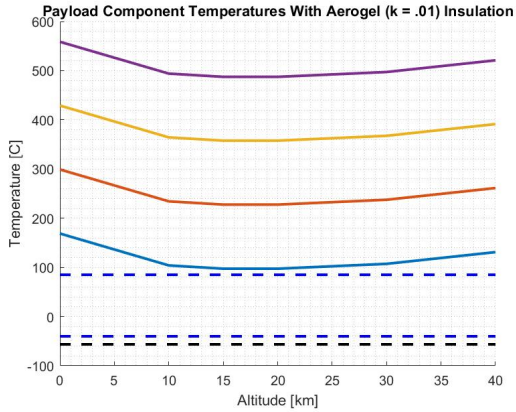


Figure 87. Model Results for Payload Area with Aerogel Insulation

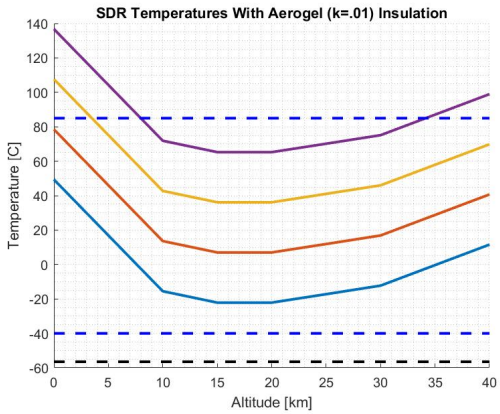


Figure 88. Model Results for SDR Area with Aerogel Insulation

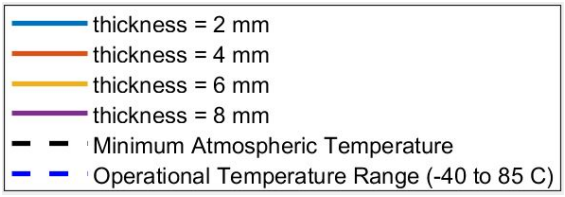


Figure 89. Legend for Model Results

10.4. Baseline Physical Model Full Results

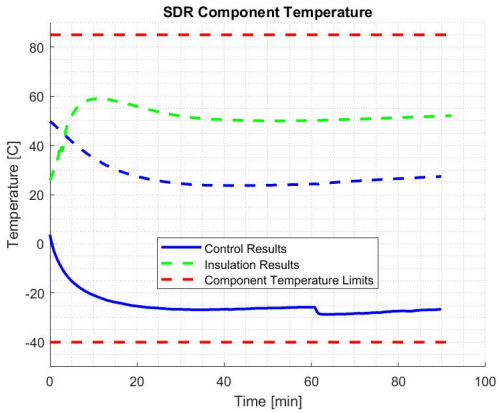


Figure 90. Baseline Physical Model Results for SDR Area

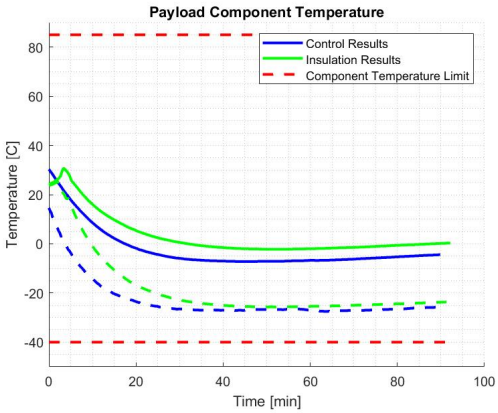


Figure 91. Baseline Physical Model Results for Payload Area

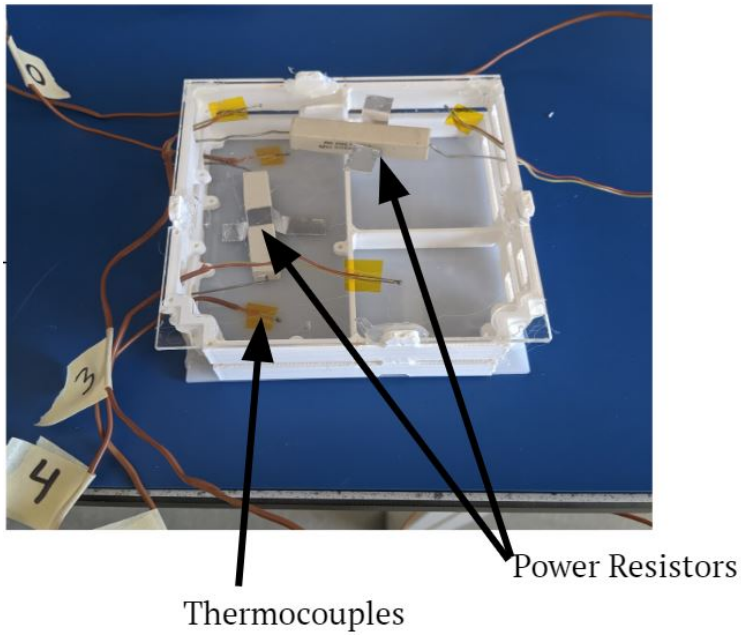


Figure 92. Baseline Physical Model Configuration

10.5. Cold Test Configuration Images

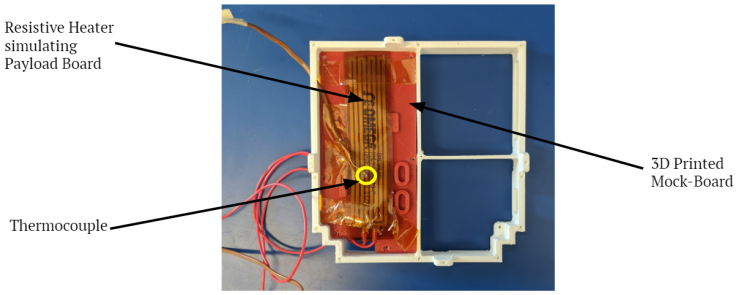


Figure 93. Cold Test Configuration

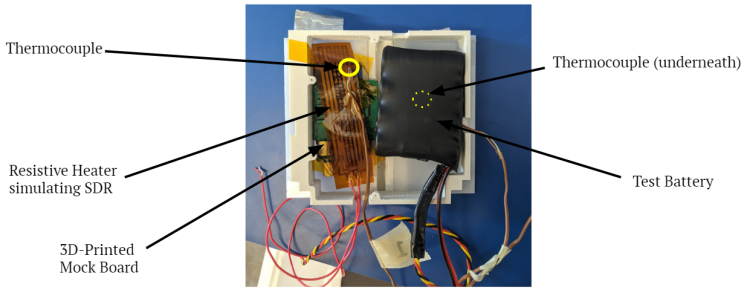


Figure 94. Cold Test Configuration

Finance Appendix

FINANCIAL OVERVIEW - *Total Purchasing*									
Order No.	**Component Name**	**Component ID**	**Subsystem**	**Date Ordered**	**Qty.**	**Component Unit Price**	**Component Total Price**	**Order Subtotal Price**	**Full Total Price**
1.1	BeagleBone Black Wireless	BBB-WL-SC-562-ND BBB-WL-SC-562	Payload Board	10/2/2019	1	$76.13	$76.13	**$89.45**	**$108.15**
1.2	AC/DC Wall Mount Adapter	102-3424-ND SWI12-5-N-P5	Payload Board	10/2/2019	1	$13.32	$13.32		
2.1	BeagleBone Black Wireless	BBB-WL-SC-562-ND BBB-WL-SC-562	Payload Board	10/7/2019	1	$76.13	$76.13	**$89.45**	**$108.15**
2.2	AC/DC Wall Mount Adapter	102-3424-ND SWI12-5-N-P5	Payload Board	10/7/2019	1	$13.32	$13.32		
3.1	Software Defined Radio	LimeSDR Mini	Comm HW	11/6/2019	2	$159.00	$318.00	**$350.76**	**$347.52**
3.2	SMA Cables	ACX1731-ND	Comm HW	11/6/2019	2	$16.38	$32.76		
4.1	Styrofoam Cooler	n/a	Thermal	11/7/2019	1	$5.99	$5.99	**$13.98**	**$15.22**
4.2	Extruded Polystyrene	Black Foamboard	Thermal	11/7/2019	1	$7.99	$7.99		
5.1	Dry Ice	n/a	Thermal	11/9/2019	1	$15.29	$15.29	**$20.28**	**$22.07**
5.2	26QT Cooler	n/a	Thermal	11/9/2019	1	$4.99	$4.99		
6.1	Attenuation Test Wire 2	CSA-SMAM-216-SAFB-ND	Comm HW	1/21/2020	2	$12.85	$25.70	**$25.70**	**$30.69**
7.1	Onboard Power Amp	SKY65009-70LF	Comm HW	1/21/2020	5	$6.67	$33.35	**$228.76**	**$241.13**
7.2	Attenuation Test Wire 1	744-1427-ND	Comm HW	1/21/2020	2	$10.68	$21.36		
7.3	M-RP-SMA to M-SMA cable	CSA-SMAM-216-	Comm HW	1/21/2020	4	$12.73	$50.92		
7.4	M-SMA to M-SMA cable	744-1427-ND	Comm HW	1/21/2020	6	$9.92	$59.52		
7.5	F-SMA to M-SMA cable	CSA-SMAM-216-	Comm HW	1/21/2020	2	$12.85	$25.70		
7.6	LNA 1 kOhm resistor	311-1.0KERCT-ND	Comm HW	1/21/2020	4	$0.10	$0.40		
7.7	LNA 5.6 nH inductor	712-1462-1-ND	Comm HW	1/21/2020	4	$0.10	$0.40		
7.8	LNA 15 nH inductor	587-1521-1-ND	Comm HW	1/21/2020	10	$0.03	$0.29		
7.9	LNA 82 nH inductor	490-2634-1-ND	Comm HW	1/21/2020	4	$0.10	$0.40		
7.10	LNA 56 pF capacitor	311-1159-1-ND	Comm HW	1/21/2020	4	$0.23	$0.92		
7.11	LNA 82 pF capacitor	311-4424-1-ND	Comm HW	1/21/2020	4	$0.25	$1.00		
7.12	LNA 68 pF capacitor	311-1160-1-ND	Comm HW	1/21/2020	4	$0.22	$0.88		
7.13	LNA 1000 pF capacitor	399-8131-1-ND	Comm HW	1/21/2020	4	$0.19	$0.76		
7.14	LNA 0.1 µF capacitor	399-C1206C104K5RAC7800CT-ND	Comm HW	1/21/2020	12	$0.09	$1.06		

7.15	Power Amp 0 Ohm Resistor	RMCF1206ZT0R00CT-ND	Comm HW	1/21/2020	10	$0.03	$0.30		
7.16	Power Amp 1 µF capacitor	399-8150-1-ND	Comm HW	1/21/2020	4	$0.19	$0.76		
7.17	Power Amp 1000 pF capacitor	399-8131-1-ND	Comm HW	1/21/2020	4	$0.19	$0.76		
7.18	Power Amp 68 pF capacitor	311-1160-1-ND	Comm HW	1/21/2020	4	$0.22	$0.88		
7.19	Power Amp 4.7 pF capacitor	311-1218-1-ND	Comm HW	1/21/2020	10	$0.17	$1.71		
7.20	Power Amp 6.8 pF capacitor	311-1220-1-ND	Comm HW	1/21/2020	4	$0.25	$1.00		
7.21	Power Amp 8.2 nH inductor	587-1517-1-ND	Comm HW	1/21/2020	10	$0.03	$0.29		
7.22	Power Amp 12 nH inductor	587-1520-1-ND	Comm HW	1/21/2020	10	$0.03	$0.29		
7.23	Power Amp 47 nH inductor	587-1527-1-ND	Comm HW	1/21/2020	10	$0.03	$0.29		
7.24	SMA Connectors	SAM8856-ND	Comm HW	1/21/2020	4	$6.38	$25.52		
8.1	N-type Male to SMA Female adapter	PE9081	Comm HW	1/22/2020	2	$16.35	$32.70	**$32.70**	**$44.95**
9.1	Yagi Antenna	A09-Y8NF	Comm HW	1/21/2020	2	$92.70	$185.40	**$185.40**	**$185.40**
10.1	Ground Power Amp	Walfront15mgeafc7z	Comm HW	1/21/2020	1	$17.89	$17.89	**$17.89**	**$17.89**
11.1	LNA	PMA4-33GLN+	Comm HW	1/21/2020	20	$6.95	$139.00		
11.2	LNA Eval Board	TB-754+	Comm HW	1/21/2020	1	$104.95	$104.95	**$341.60**	**$353.60**
11.3	Attenuator	VAT-20+	Comm HW	1/21/2020	7	$13.95	$97.65		
12.1	Low Pass filter	744-1433-ND	Comm HW	1/23/2020	2	$24.09	$48.18	**$48.18**	**$48.18**
13.1	Onboard Antenna	RPSMA	Comm HW	1/23/2020	2	$3.00	$6.00	**$6.00**	**$15.00**
14.1	Battery	PR-CU-R782	Power	1/22/2020	1	$64.00	$64.00	**$64.00**	**$74.75**
15.1	Battery	PR-CU-R782	Power	1/27/2020	1	$64.00	$64.00		
15.2	Battery Charger	CH-L7405: 5.2 x21.mm	Power	1/27/2020	1	$19.95	$19.95	**$120.95**	**$135.74**
15.3	Class 9 Hazard Handling Fee	HZ-S&H-Ground	Power	1/27/2020	1	$37.00	$37.00		
16.1	Dry Ice	n/a	Thermal	1/24/2020	1	$6.90	$6.90	**$6.90**	**$6.90**
17.1	Acer Chromebook Laptops	CB3-431-C5FM	Systems	2/3/2020	2	$208.83	$417.66	**$417.66**	**$417.66**
18.1	ThinSat Education Kit	n/a	Systems	2/19/2020	1	$1,000.00	$1,000.00	**$1,000.00**	**$1,000.00**
19.1	Software-defined radio	LimeSDR Mini	Comm HW	1/27/2020	2	$159.00	$318.00	**$318.00**	**$318.00**
20.1	Inertial measurement unit	MTi-3-DK	Science	1/27/2020	1	$437.89	$437.89	**$437.89**	**$449.48**
21.1	COMM PCB	n/a	Comm HW	1/27/2020	5	$3.60	$18.00	**$18.00**	**$38.00**
22.1	General Purpose USB to GPIO+SPI+I2C	FT232H	Payload Board	1/29/2020	1	$14.95	$14.95	**$14.95**	**$23.25**
23.1	Aluminum	Multipurpose 6061 Aluminum	Structures	1/29/2020	1	$5.89	$5.89		

23.2	Acrylic	Clear Scratch- and UV-Resistant Cast Acrylic Sheet	Structures	1/29/2020	2	$5.37	$10.74	**$17.63**	**$35.35**
24.1	Ground Power Amp	Walfront15mgeafc7z	Comm HW	2/4/2020	1	$17.89	$17.89	**$17.89**	**$17.89**
25.1	Logic Level Converter - Bi-Directional	BOB-12009	Payload Board	2/5/2020	2	$2.95	$5.90	**$8.40**	**$8.40**
25.2	JST RCY Connector - M/F	PRT-10501	Power	2/5/2020	5	$0.50	$2.50		
26.1	USB Type A Female Breakout	BOB-12700	Power	2/10/2020	2	$4.50	$9.00	**$16.90**	**$24.04**
26.2	USB 6 FT extension	CAB-00517	Power	2/10/2020	2	$3.95	$7.90		
27.1	Payload PCB	n/a	Payload Board	2/12/2020	1	$354.00	$354.00	**$354.00**	**$354.00**
28.1	BeagleBone Black Wireless	BBBWL-SC-562-ND	Payload Board	2/12/2020	3	$76.13	$228.39	**$425.34**	**$458.95**
28.2	AC/DC Wall Mount Adapter	102-4136-ND	Payload Board	2/12/2020	3	$8.50	$25.50		
28.3	USB A to Mini B 0.8m Black	WM14083-ND	Payload Board	2/12/2020	4	$2.67	$10.68		
28.4	Memory Card Mi	1582-1012-ND	Payload Board	2/12/2020	3	$35.59	$106.77		
28.5	Cable USB to UA	768-1319-ND	Payload Board	2/12/2020	3	$18.00	$54.00		
29.1	Digi-Key BOM	n/a	Payload Board	2/12/2020	1	$102.95	$102.95	**$102.95**	**$102.95**
30.1	Ground Power A	YWBL-WHfbeqg	Comm HW	2/12/2020	1	$23.49	$23.49	**$46.48**	**$46.48**
30.2	PLA Spool	n/a	Structures	2/12/2020	1	$22.99	$22.99		
31.1	JST RCY Connect	10501	Power	2/14/2020	10	$0.50	$5.00	**$5.00**	**$12.14**
32.2	Buck-Boost Con	15208	Power	2/14/2020	1	$9.95	$9.95	**$9.95**	**$17.09**
33.1	SMA to BNC Kits	n/a	Comm HW	2/17/2020	1	$6.39	$6.39	**$13.98**	**$13.98**
33.2	USB 3.0 A to A M	n/a	Power	2/17/2020	1	$7.59	$7.59		
34.1	Low Noise Ampli	PMA4-33GLN+	Comm HW	2/17/2020	1	$29.00	$29.00	**$29.00**	**$29.00**
35.1	McMaster-Carr B	n/a	Structures	2/19/2020	1	$20.28	$20.28	**$20.28**	**$35.14**
36.1	SMA Connectors	SAM8856-ND	Comm HW	2/20/2020	6	$6.38	$38.28	**$38.28**	**$43.27**
37.1	SparkFun DC/DC	BOB-09370	Power	2/23/2020	1	$37.08	$37.08	**$37.08**	**$37.08**
38.1	Comm PCB Rev.	n/a	Comm HW	2/24/2020	3	$11.97	$35.90	**$35.90**	**$42.60**
39.1	LNA Evaluation I	TB-754+	Comm HW	2/24/2020	2	$104.95	$209.90	**$209.90**	**$222.73**
40.1	Power Amplifier Module	n/a	Comm HW	3/9/2020	2	$23.49	$46.98	**$44.63**	**$44.63**
41.1	SPARKFUN ORD	BOB-12700	Power	3/9/2020	2	$4.50	$9.00	**$15.75**	**$22.90**
41.2	Heat Sink Compo	PRT-09599	Thermal		3	$2.25	$6.75		
42.1	USB A Male to A	1175-1075-ND	Power	3/9/2020	4	$4.50	$18.00	**$137.02**	**$143.97**
42.2	RF Filter Low Pa	744-1433-ND	Comm HW		1	$24.09	$24.09		
42.3	CBL ASSY SMA P	744-1427-ND	Comm HW		1	$10.68	$10.68		
42.4	CBL ASSY SMA P	J10379-ND	Comm HW		2	$32.34	$64.68		
42.5	SMA Bulkhead Ja	24-415-0205-MM	Comm HW		1	$19.57	$19.57		
43.1	Multipurpose 11	8963K304	Thermal	3/9/2020	2	$7.53	$15.06	**$15.06**	**$15.06**
44.1	Yagi Antenna	A09-Y8NF	Comm HW	3/9/2020	2	$92.70	$185.40	**$185.40**	**$185.40**
45.1	SMA Female to N	PE9081	Comm HW	3/9/2020	2	$16.99	$33.98	**$33.98**	**$46.23**

Figure 95. TOMCAT Full Expenditure