Flight Computer Board for IlliniSat-2

Design Document

Team 23 – Adam Newhouse and Dillon Hammond
ECE 445 Design Document – Spring 2020
TA: Chi Zhang

Contents

1 Introduction

1.1 Background

This project was originally proposed by Alex Ghosh for the CubeSat team. The Illinisat-2 is a scalable CubeSat satellite bus developed at the University of Illinois. The problem statement was to design a carrier board that both mounts the flight computer and interfaces with other components of the satellite, including the power system, payload, and radio connections. The carrier must be built to flight electronic specifications using high reliability parts, leaded construction to prevent tin whiskering, and conformally coated. The board must also conform exactly to the Illinisat-2 mechanical component outline in order to properly fit in the satellite. Developing CubeSat busses is an active area of commercial investment and by providing a system with a Linux based microcontroller for ease of software development, providing Attitude Determination and Control Systems (ADCS) capability as described, reliable storage, and providing multiple methods of payload communication, our design will be attractive to investors.

1.2 Objective

The original solution utilized a commercially available MitySOM module from CriticalLink as the primary flight computer. This module was then mated via a SODIMM connector to a complex carrier board that included flash storage, interface drivers, and connectors. The payload connections (five RS422, one RS485, two USB-UART, and one UART) [1] were designed specifically for the original mission that this bus was intended to be used for: Lower Atmosphere/Ionosphere Coupling Experiment (LAICE). Designing for the LAICE mission was an understandable first step, however the use case for the IlliniSat-2 flight board was quickly extended past the LAICE missions. As a result, the original design was not flexible enough to be conveniently used in later missions.

Our new solution removes the need for an expensive computer module and reduces the complexity of the carrier board, leading to an inexpensive and passive two-layer BIB. This redesign is focused on correcting design decisions and oversights made in the original project, as well as extending it for better performance in a wider array of CubeSat missions, and reducing the overall cost and size of not only the flight computer board but the IlliniSat-2 bus as a whole.

We will integrate communication to a wide range of payloads on the same board as the flight computer including specific hardware for (ADCS) that the original solution neglected to specifically address, despite it being a critical component of most CubeSat operations. This is important because at a minimum, ADCS operation requires what is known as “detumbling” which is the initial process after satellite launch that puts the vehicle in an initial known orientation. This can be accomplished with the IMU and magnetometer that we provide.

This solution will be a significant improvement over the original design for any user of the bus. Our flight computer board will utilize a backbone connector to attach to a stakeholder designed bus interface board (BIB) which provides the physical and electrical interfaces to a mission’s payloads and power system. Having the BIB allows for convenient swap in/out of hardware. The design of the BIB is beyond the scope of this project and is left to the end user create. However, the BIB design itself is simple because all it is required to do is adapt from the flight computer backbone to the mission specific physical and electrical design. This should allow compatibility with almost any bus standard and reduce the mission development time significantly.

1.3 Visual Aid

Shown in **Error! Reference source not found.** is a render of the completed board. The main integrated circuit is the SiP from Octavo. The microSD card slot and real time clock battery backup are also visible. The higher-level connection diagram for the proposed solution is show in Figure 2.

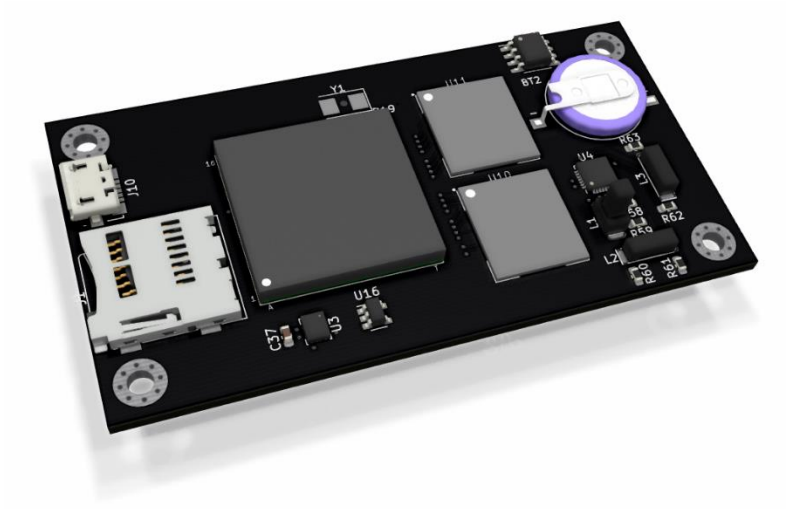

Figure 1 - Flight Computer PCB Render

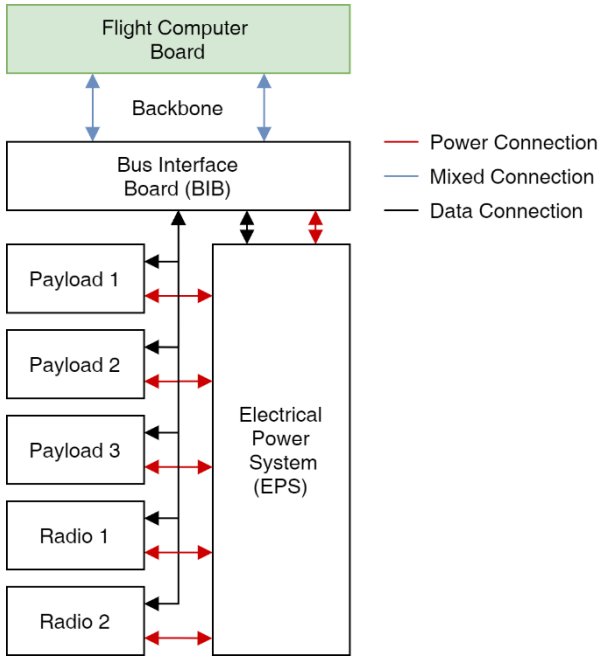

Figure 2 – System Diagram

1.4 High Level Requirements

- Successfully boot Linux and communicate with all the interfaces described: RS422, CAN, UART, USB 2.0, I2C, and read and write persistent data to the eMMCs.
- The attitude determination sensors must be sensitive enough to allow for detumbling from a maximum of 15 deg/sec to less than 5.0 deg/sec.
- The flight board must be able to support at least two radios and three scientific payloads for the duration of a mission.

2 Design

2.1 Block Diagram

The two primary components of this flight computer board are the Octavo SiP and the backbone connector. The flight computer board contains the minimum hardware necessary for the flight board operation (besides power) via the sensors and storage groups. Generally the SiP will communicate with the sensors and storage to monitor flight status and record any desired data. Many interfaces are listed as crossing from the SiP to the backbone connector. This allows the stakeholder to create their own BIB which connects to the backbone and provides hardware using these interfaces. The uSD card, micro USB, status LEDs, and debug connector are all used primarily for laboratory testing. The reset supervisor is crucial in preventing system lockups in the harsh environment of low Earth orbit.

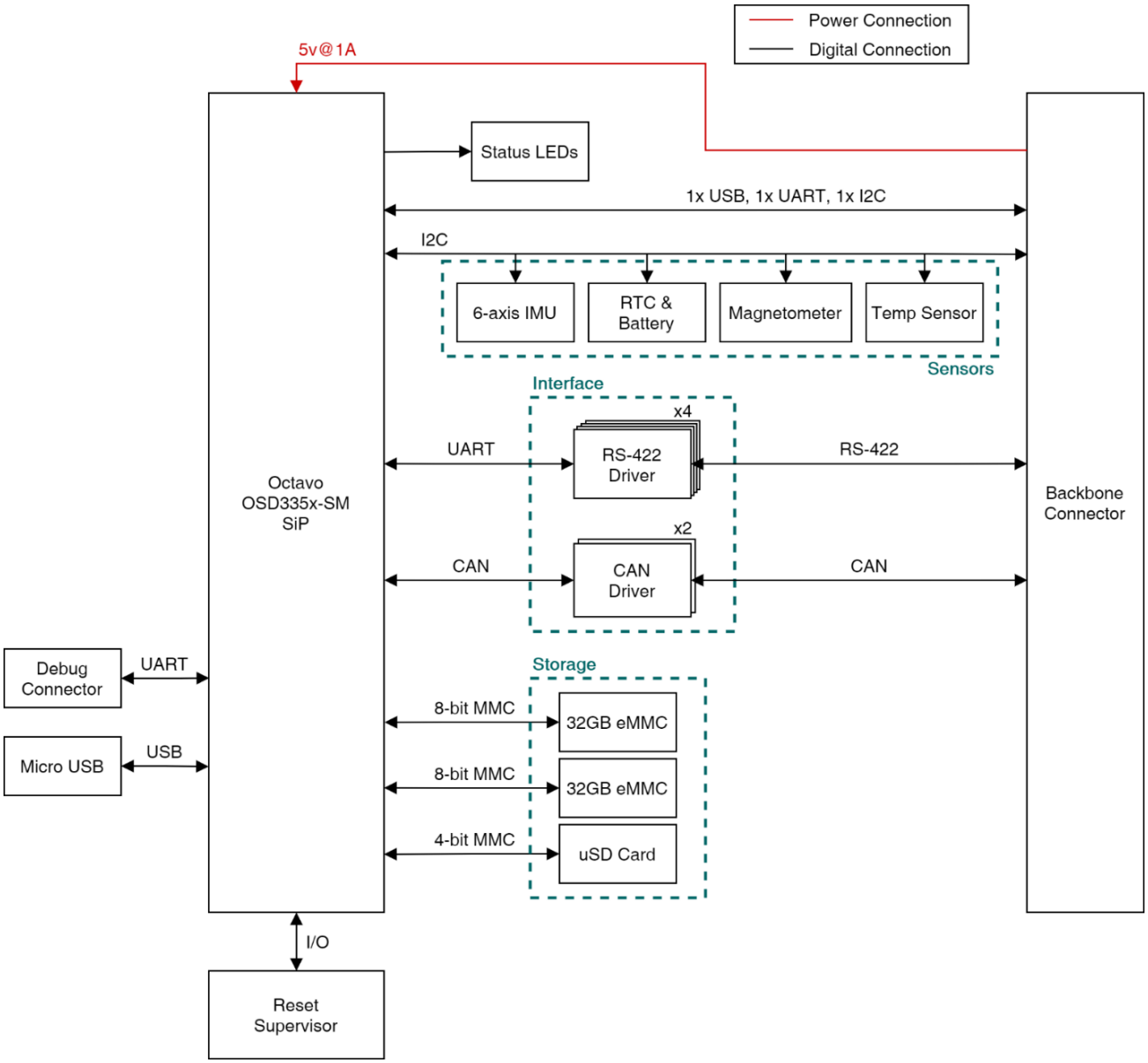

Figure 3 - Block Diagram

2.2 Physical Design

The flight computer board (shown in green) is small enough to allow two side-by-side flight boards to co-exist on a BIB designed for the dimensions of a typical CubeSat (90x90mm). The board mounts to the BIB via four M2.5 screws and has a backbone connector on both short edges to improve physical ruggedness. Rigid mounting is needed to survive the vibration and shock conditions of testing and eventual launch on an orbital vehicle. The flight computer board will be a standard FR4 four-layer board with 0.005-inch clearance and trace width. By minimizing the size of the high-density flight computer board, manufacturing cost is reduced.

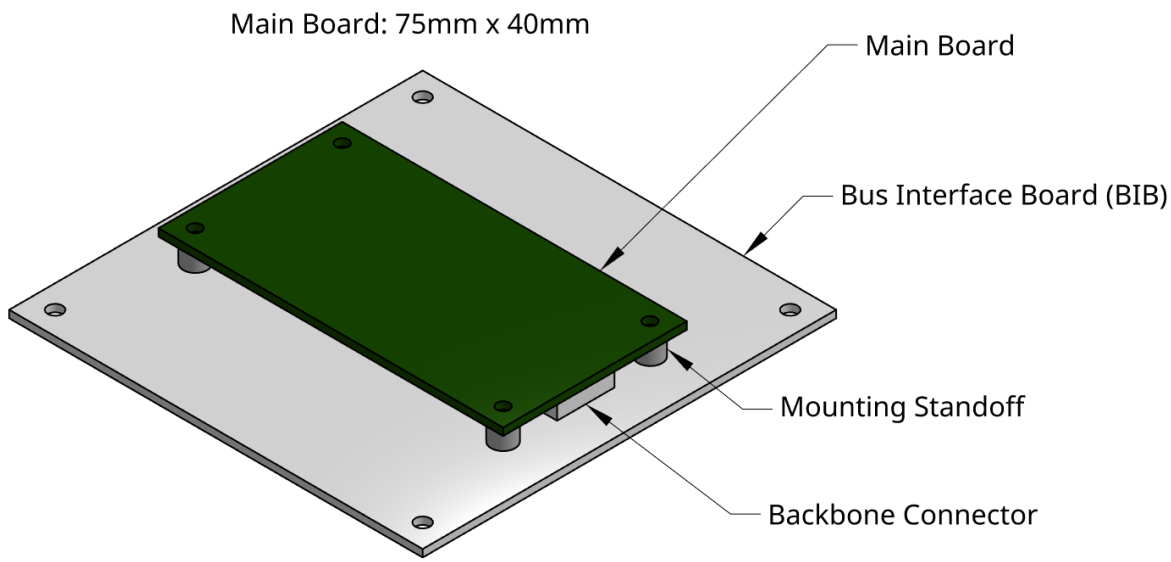

Figure 4 - Physical Design

2.3 Schematic Design

The full schematic of the proposed design is shown in 7.1. Figure 5, shows the SiP connections required for this design.

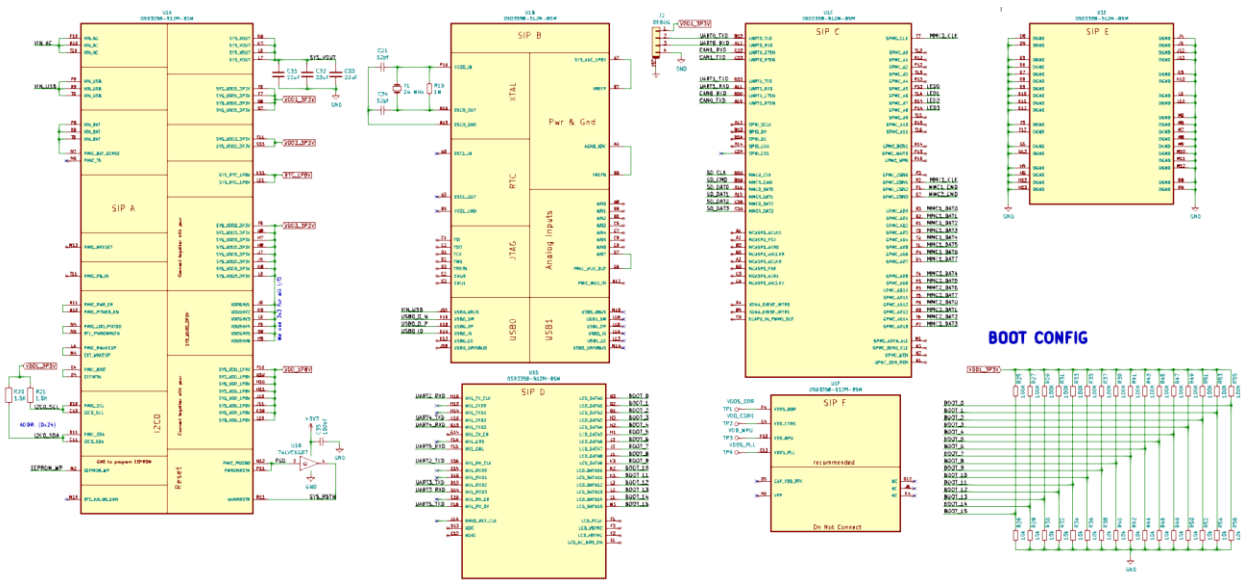

Figure 5 - SiP Connections

2.4 Power Consumption

In typical use cases, the micro SD card is not used and does not factor into our power consumption estimate. Since datasheets are not typically available for eMMCs, we estimate about 500mW.

Device	**Typical Power (mW)**	**Max Power (mW)**
ISM330DLC	1.65	2.475
RM3100	0.4455	0.858
TMP112	0.0495	0.2805
TCAN332	0.0396	0.0858
Micro SD	0	330
eMMC	500	500
AM3358	990	1,500
TOTAL	1,492	2,334

Table 1 - Power Consumption

2.5 Software Design

To satisfy the first high level requirement, it is necessary for us to construct a Linux kernel image that is capable of being booted on the AM335X, the chip underlying the Octavo OSD335-x SiP that we are using on our board.

2.5.1 Cross Compiling Toolchain

To be able to compile many of the later steps of this process, a proper cross compiling toolchain must be established. Texas Instruments provides their own custom Arago line of modified GCC compilers and C libraries which were used in the original project. However, this is an old version of GCC and as a result does not support many flags that we would like to rely on. Instead an open source tool called crosstool-

NG provides an easy and scriptable interface to build a custom GCC compiler and C standard library specifically for the intended target. This allows us to use a modern compiler and still ensure that the compiler and libraries are appropriate for our target.

2.5.2 Das U-Boot

Das U-boot is a commonly used boot loader for embedded applications that loads and begins execution of the Linux kernel. It must be configured and built specifically for the relevant target. The BeagleBone Black, a popular commercial embedded board that also uses the AM335X, freely distributes a precompiled distribution of U-boot for this target, so we will use this.

2.5.3 Linux Kernel and Modules

The original project utilized a custom build of a Linux 3.2 kernel, which was updated in 2012. Unfortunately this kernel version is old enough that it did not directly support some of the hardware utilized on the bus: both RS485 support and USB-UART support had to be manually patched in and the method of manually configuring pin muxing in the kernel was very difficult. For this new project we intend to target Linux 4.14 which is a Long-Term Support (LTS) kernel until 2024. It also has several AM335X specific patches included by Texas Instruments. The process of building the kernel requires initially manually selecting the set of compile time options. Most kernel options will be unrelated to our project and should bed disabled to save build time and space. The kernel is then compiled into a zImage with the tool chain created from crosstool-NG. Any external kernel modules that we need will also be built here in a separate command.

2.5.4 Device Trees

Device trees are the modern Linux kernel's way of specifying interfaces to internal and external hardware. Texas Instruments has already patched the default device trees necessary for building an AM335X kernel. We will have to add in additional device trees that specify the pin mapping of our custom board design. These are then built externally to the kernel into a single binary file.

2.5.5 Root Filesystem

Once the kernel is properly configured, the user space root filesystem can be set up. Here we will utilize crosstool-NG to compile any libraries and programs that we intend to always have on the system. Another open source tool, Build Root, makes it very easy to choose packages to build into the root filesystem and it leverages the crosstool-NG toolchain to compile these packages from source. The resulting root filesystem is loaded by the kernel on boot.

2.6 Requirements and Verification

2.6.1 Micro USB

The Micro USB connector is provided to allow easy powering of the flight board when no BIB is connected, as well as debugging over USB.

Requirement	Verification
1. *The Micro USB interface can power the flight board by providing 4.3 v – 5.8 v at 1 A.* 2. *The flight computer can communicate with USB devices via the Micro USB connector.*	1. a. Configure a power supply with a Micro USB connector to provide between 4.3 v – 5.8 v at 1 A. b. The board should become powered and the status LEDs will light up. 2. a. Attach an ethernet to micro USB adapter to the Micro USB connector. b. Power on the board with 5 v at 1 A. c. While running, the Linux system will display the new network interface after running the command `ip a`

2.6.2 Debug Connector

The debug connector provides a simple UART interface to the flight computer. Typically, this used as a serial console for Linux.

Requirement	Verification
1. *The debug connector shall provide a serial interface at a minimum of 115.2 kbaud.*	1. a. Connect a visible and controllable UART interface to the debug connector. b. Configure the interface to run at 115.2 kbaud. c. Power on the board with 5 v at 1 A. d. After powering the board, a Linux shell prompt will be displayed.

2.6.3 Reset Supervisor

The reset supervisor acts as a watchdog for the flight computer. If the flight computer ceases to message the reset supervisor or if the voltage supplied to the reset supervisor drops below a threshold, it resets the flight computer.

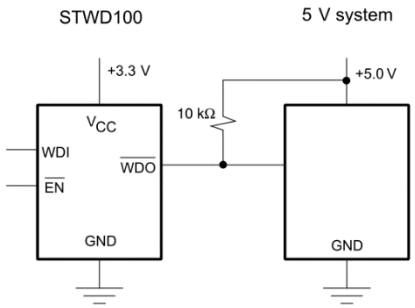

Figure 6 - STWD100 Watchdog Circuit [7]

Requirement	Verification
1. *In event of a software lockup, the reset supervisor will reset the flight computer within 60 seconds.* 2. *In event of a single event upset (SEU), the reset supervisor will reset the flight computer within 100 uS.*	1. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. Run the provided Linux program which will enable the reset supervisor but not ping it. d. Within 60 seconds, observe on the debug interface that the Linux system reboots and displays a prompt again. 2. a. Attach oscilloscope probes to the reset supervisor I/O line and ground and configure for a one-shot change. b. Attach oscilloscope probes to the power line and ground and configure for a one-shot change. c. Power on the board with 5 v at 1 A. d. Reconfigure the power supply to provide 4 v at 1 A. e. Observe that the flight computer is no longer responsive. f. Measure the time between power dropping to 4 v and the reset line changing on the oscilloscope. This should be within 100 us. g. Reconfigure the power supply to provide 5 v at 1 A. h. Observe that the flight computer boots normally and provides a prompt.

2.6.4 Flash Storage

Non-volatile storage is critical for IlliniSat-2 CubeSat operations as nearly all missions require storing data for extended periods of times (at least days, if not years) which is either eventually transmitted to the ground or is used for in-flight operations of the satellite. Given that the data stored is critical to science missions and satellite operations, it is important to provide redundancy in case of an in-flight data storage failure.

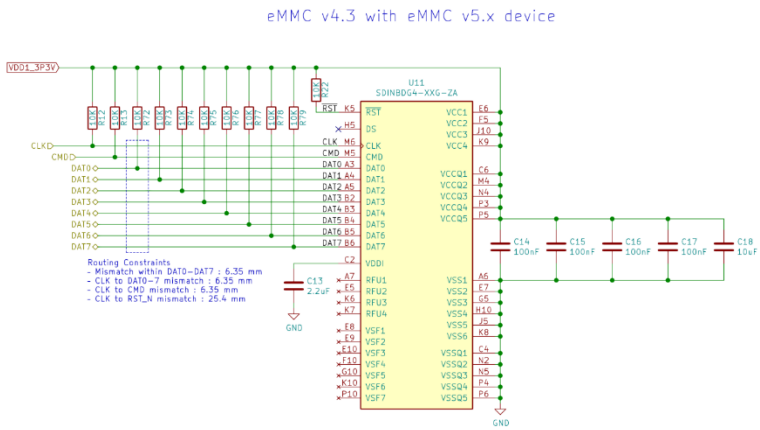

Figure 7 - eMMC Schematic

Requirement	Verification
1. *At least 32 GB of non-volatile flash storage is available to the flight computer.* 2. *The non-volatile flash storage shall provide a sustained write speed of at least 1 MB / s.* 3. *The non-volatile flash storage shall provide a sustained read speed of at least 1 MB / s.*	1. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. Run `lsblk -f`. d. Verify that at least one eMMC device appears and lists at least 32 GB of storage. 2. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. Use `mount` to mount the eMMC. d. Run `dd if=/dev/zero of=<file on eMMC> bs=8k count=100k`. e. Observe that the previous command lists a write speed of at least 1 MB/s 3. a. Perform the previous write speed verification first. b. Run `sync` c. Run `dd if=<file on eMMC> of=/dev/null bs=8k count=100k`.

	d. Observe that the previous command lists a read speed of at least 1 MB/s.

2.6.5 Micro SD Card Slot

The micro SD card slot can be used for easily extracting information during debugging as well as flashing the Linux image and root filesystem to the flight computer.

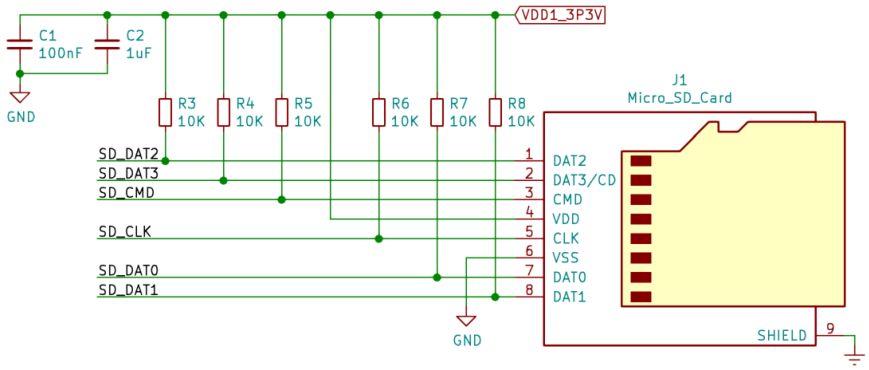

Figure 8 - Micro SD Card Schematic

Requirement	Verification
1. *The flight computer can be flashed from the micro SD card.* 2. *The flight computer can read or write data arbitrarily to the micro SD card.*	1. a. Set the boot config to boot from the micro SD card rather than the internal flash. b. Connect a visible and controllable UART interface to the debug connector. c. Power on the board with 5 v at 1 A. d. From the presented U-boot prompt, boot the Linux kernel on the micro SD card. e. From Linux, copy the following items to the internal flash. i. MLO ii. U-boot image iii. U-boot environment variables iv. zImage (Linux kernel) v. Device tree binary blob vi. Root filesystem f. Reboot the system g. Use U-boot to boot from the internal flash. h. Verify that the system boots and reaches a Linux prompt. 2.

	a. Set the boot config boot from the internal flash. b. Insert a micro SD card. c. Connect a visible and controllable UART interface to the debug connector. d. Power on the board with 5 v at 1 A. e. Use `lsblk -f` to verify that the micro SD card is listed as a block device. f. Use `mkfs` to create a filesystem on the micro SD card. g. Use `mount` to mount the filesystem h. Run `dd if=/dev/zero of=<file on eMMC> bs=8k count=100k` and observer that the command succeeds. i. Run `sync`. j. Run `dd if=<file on eMMC> of=/dev/null bs=8k count=100k` and observer that the command succeeds.

2.6.6 Flight Computer

The flight computer controls the operations of the satellite including communication via radio, interfacing with payloads, and collecting science data. It must be capable of utilizing all the desired interfaces for this board. For simplicity of programming, this chip will run Linux. Refer to 2.3 for the schematic.

Requirement	Verification
1. *The flight computer shall be capable of booting a Linux operating system.* 2. *The flight computer shall be capable of communicating over all interfaces specified in 2.6.9.* 3. *The flight computer shall provide hardware floating point calculation capability.*	1. Verified with the Micro SD card's first verification procedure, 2.6.5. 2. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. RS422 i. Connect a device with at least one known command and response. ii. Communication will utilize the Linux built-in serial driver. iii. Send the known command. iv. Verify that the intended response is received. d. CAN i. Connect two devices each with at least one known command and response. ii. Communication will utilize the Linux built-in CAN driver.

	iii. Send the known command to the first device. iv. Verify that the intended response is received. v. Send the known command to the second device. vi. Verify that the intended response is received. e. UART i. Verified with the Debug Connector procedure, 2.6.2. f. USB 2.0 i. Verified with the Micro USB procedure, 2.6.1. g. I2C i. Connect each of the devices specified in the block diagram as on the same I2C bus. ii. Communication will utilize the Linux built-in I2C driver. iii. For each device. 1. Send the command(s) to generate data. 2. Receive this data and verify that it is correct. 3. Observe that the data sheet for the AM335X lists the VFPv3 Floating Point Unit.

2.6.7 Real-Time Clock (RTC)

Keeping accurate time in orbit is very important for attitude determination and control. A time error of just one second can lead to a positional error of over 8 km. Relatively precise positional data is used to

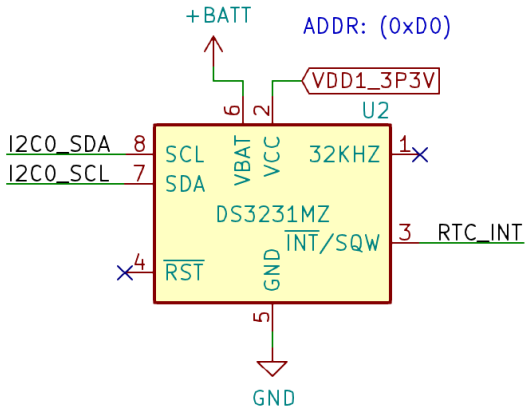

Figure 9 - Real Time Clock Schematic

propagate and predict our current orbital position for attitude control. However, the clock can be calibrated in orbit to cancel out any accumulated drift.

Requirement	Verification
1. *The RTC shall not drift by more than 1 second / day when running on backup battery power.* 2. *The RTC shall communicate with the flight computer over I2C.*	1. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. Use `hwclock -r` to show the current RTC time and record it externally. d. Wait 24 hours. e. Use `hwclock -r` and compare the results to the previous attempt. If the difference is less than one second, the RTC meets requirements. 2. Verified in the flight computer I2C test, 2.6.6.

2.6.8 RTC Battery Backup

A good CubeSat can keep accurate time even when fully powered off. The backup battery should rechargeable which means it never needs to be replaced. This is important because some missions must wait more than a year to get launched. The selected RTC IC typically draws 2.0uA of current in standby mode. With the 17mAh battery selected in this design, the expected battery life is 350 days. Any time the satellite is powered on, this battery is automatically recharged.

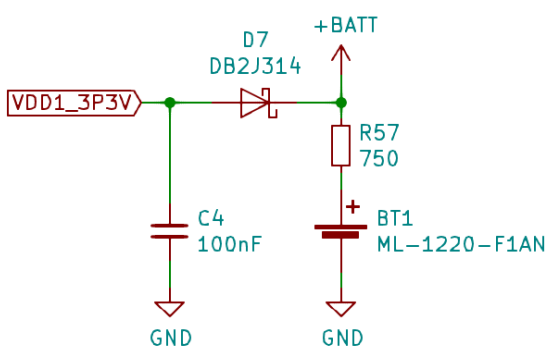

Figure 10 - RTC Battery Charging Circuit

Requirement	Verification
1. *The RTC's backup battery shall be rechargeable.*	1. Verify that the battery is chargeable from the manufacturer datasheet.

2.6.9 Interfaces

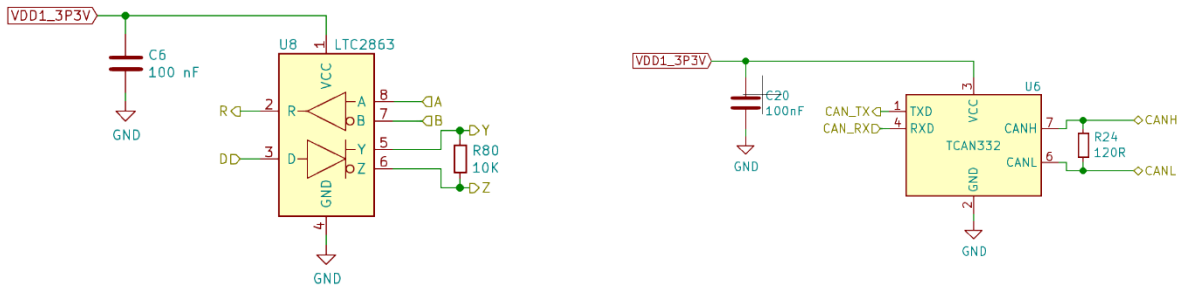

Figure 11 - CAN and RS422 Interface Schematic

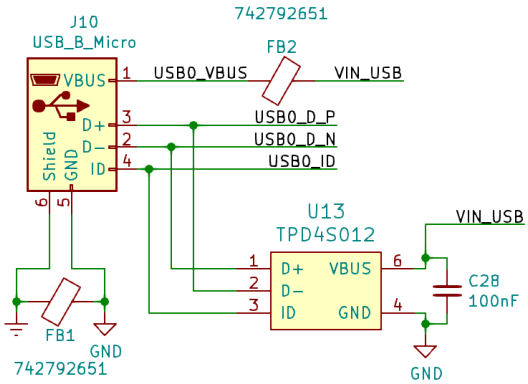

Figure 12 - USB Interface Schematic

Having a wide range of interfaces allows the flight computer to communicate with a variety of on-board hardware and off board payloads. At a minimum, this project should provide the same number of interfaces as the original project (four RS422 and one UART).

Requirement	Verification
1. *There shall be at least four RS422 interfaces to the flight computer.* 2. *There shall be at least two CAN busses interfaced to the flight computer.* 3. *There shall be at least one UART interface to the flight computer.* 4. *There shall be at least one USB 2.0 interface to the flight computer.* 5. *There shall be at least two I2C busses interfaced to the flight computer.*	1. Refer to full board schematic 7.1. 2. Refer to full board schematic 7.1. 3. Refer to full board schematic 7.1. 4. Refer to full board schematic 7.1. 5. Refer to full board schematic 7.1.

2.6.10 6-Axis Inertial Measurement Unit (IMU)

Since every CubeSat should have basic attitude determination, an IMU should be included in the flight board instead of attached externally as in the original design.

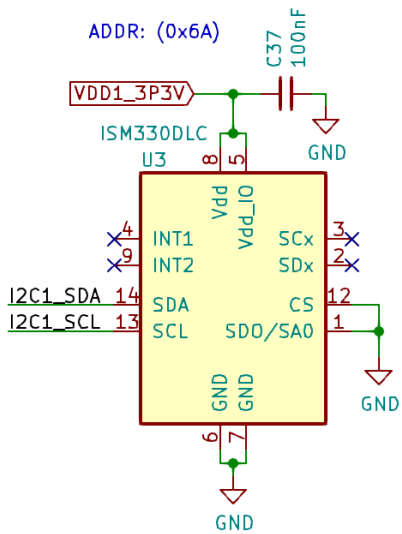

Figure 13 - 6-axis IMU Schematic

Requirement	Verification
1. *The IMU shall provide a resolution of at least 50 milli-degrees / s of angular rate with a full scale of at least +/- 30 degrees / s.* 2. *The IMU shall sample at a minimum of 100 Hz.* 3. *The IMU shall communicate with the flight computer over I2C.*	1. Verify that these specifications are achievable with the manufacturer's data sheet. 2. Verify that these specifications are achievable with the manufacturer's data sheet. 3. Verified in the flight computer I2C test, 2.6.6.

2.6.11 3-Axis Magnetometer

The magnetometer uses inductive coil pickups to measure magnetic fields with no temperature-based drift. This is important for an active ADCS algorithm as it can sense Earth's natural magnetic field (~50 uT) and then perform attitude adjustments with this information.

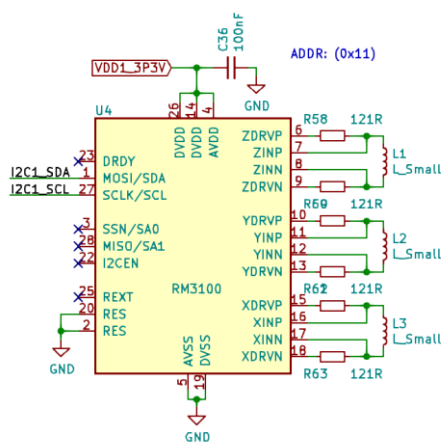

Figure 14 - Magnetometer Schematic

Requirement	Verification
1. *The magnetometer shall provide a resolution of at least 50 nT with a full scale of at least 100 uT.* 2. *The magnetometer shall sample at a minimum of 100 Hz.* 3. *The magnetometer shall communicate with the flight computer over I2C.*	1. Verify that these specifications are achievable with the manufacturer's data sheet. 2. Verify that these specifications are achievable with the manufacturer's data sheet. 3. Verified in the flight computer I2C test, 2.6.6.

2.6.12 Temperature Sensor

The AM335x (the processor we will use) includes an onboard die temperature sensor, but due to silicon errata it does not function correctly. A local digital temperature sensor was added to replace the built in one.

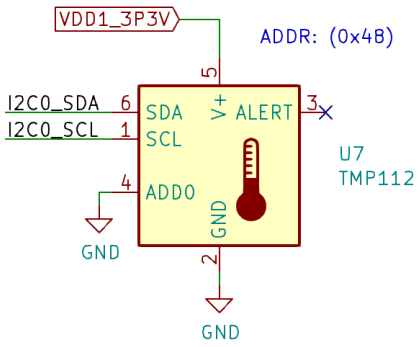

Figure 15 - Temperature Sensor

Requirement	Verification
1. *The temperature sensor shall have an accuracy of +/- 1 deg Celsius on the local sensor.* 2. *The temperature sensor shall have a resolution of at least 1 deg Celsius on the local sensor.* 3. *The temperature sensor shall communicate with the flight computer over I2C.*	1. Verify that these specifications are achievable with the manufacturer's data sheet. 2. Verify that these specifications are achievable with the manufacturer's data sheet. 3. Verified in the flight computer I2C test, 2.6.6.

2.6.13 Backbone Connector

The backbone connector is used to connect the flight board that this project is designing to a stakeholder's BIB.

Requirement	Verification
1. *Any interface (2.6.9) not already used for flight board hardware shall be available through the backbone connector.* 2. *The backbone connector shall provide an interface for the BIB to power the flight board.*	1. Refer to full board schematic 7.1. 2. Refer to full board schematic 7.1

2.6.14 Status LEDs

These are user programmable for conveying any desired information, primarily during testing.

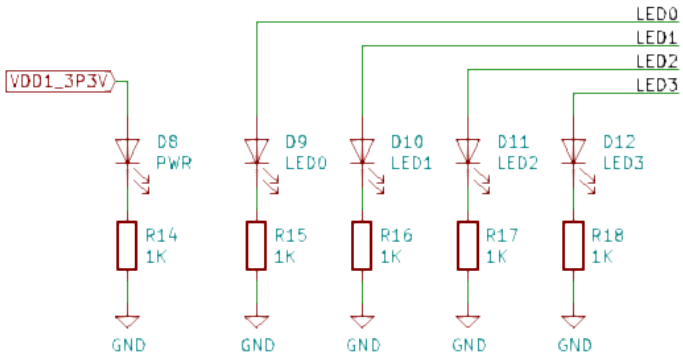

Figure 16 - Status LED Schematic

Requirement	Verification
1. *The flight computer shall be able to individually control the status LEDs.*	1. a. Connect a visible and controllable UART interface to the debug connector. b. Power on the board with 5 v at 1 A. c. Controlling the LEDs will utilize the built in Linux GPIO driver. d. Using `echo`, individually set the state of each of the four GPIO pins to on and then off. Observe that each LED lights up when its corresponding GPIO pin is enabled.

2.7 Tolerance Analysis

One of the most important aspect of our project is the built-in sensor suite for inertial measurement. This is composed of a low noise magnetometer and a compact combination accelerometer and gyroscope sensor. Between the two chips, there are nine axes of sensitivity. These sensors inform the higher-level control algorithms what the vehicle is physically doing in orbit.

One of the first operations a satellite will attempt after deployment is damp its rotation rates. This allows the solar panels to be more efficient and makes ground communication more reliable. The primary attitude control method that is available on almost every nano satellite are magnetorquers. These work by creating a magnetic moment orthogonal to the Earth's magnetic field leading to a torque. This method only works in non-equatorial orbits because it depends on a varying magnetic field over a single orbit.

To determine what magnetic moment commands must be sent to the three axis magnetorquer it is necessary to measure the ambient magnetic field from the body frame of the satellite. While detumbling, it is also important to measure and the vehicle's angular rate to ensure the detumbling process is functioning. Measuring angular rate is also important for precise pointing algorithms used after detumbling is complete.

The magnetometer we have selected is the PNI Corp. RM3100 which employs a novel induction pickup system to measure magnetic fields. Compared to typical magneto-resistive or hall effect sensors, the RM3100 delivers significantly higher sensitivity and lower noise.

Sensitivity	Field Range	Noise	Sampling Rate (3-axis)
13 nT	±1100 µT	15 nT	147 Hz

Table 2 – RM3100 Specifications

The specific detumbling algorithm used is called B-Dot control. This simple algorithm is based on premise that the measured field is sinusoidal in nature and the average of sinusoid is zero. Throughout an orbit, the vehicle will sample the magnetic field and then take the time derivative of the measured field in all three axes. This derivative is then multiplied by a constant factor and is then fed as a magnetic moment command to the three axis magnetorquer. This process takes place periodically when the magnetorquers are disabled as they would interfere with the field reading if they were enabled. This is shown in Equation 1, where M_i is the magnetic moment, and k_i is a derivative gain. Conveniently, the derivative of the field is always orthogonal to the field itself, so the magnetic moment is guaranteed to provide a torque.

$$M_i = -k_i \dot{B}_i$$

Equation 1 – B-Dot Control Law

Our required minimum rotation rate is ±5.0 degrees per second and our maximum rotation rate is ±15 degrees per second. To ensure stability, the sample rate must be chosen so that we do not travel too far between samples. A sample rate of 1 Hz has been selected because it satisfies this requirement and allows for activating the magnetorquer between samples.

At a tumble rate of 5 degrees per second, we will have traveled 5 degrees between samples. This corresponds to a magnetometer reading difference of 801.91 nT. Since the sensitivity of the magnetometer is 13 nT, we can expect a change in readings of about 61.69 counts. Therefore, with the magnetometer we have selected, detumbling to less than or equal to 5.0 degrees per second is possible. It should be noted that when the measured sinusoid reaches a local maxima or minima, the derivative is zero and therefore, the magnetorquer is off. For the purposes of this analysis, we have assumed the measured field derivative is halfway between the maximum and zero.

3 Project Differences

3.1 Overview

The original project's stated goal was to build a "scalable CubeSat satellite bus" [2]. However, by the project proposal, the high-level requirements had been simplified to the following (emphasis mine) [1]:

- Develop a board that has **only the necessary peripherals for the MitySOM**
- Develop a board that has the correct physical characteristics for items such as connectors and IC's
- Develop a board that will function in space

Of note is that the original team decided to restrict their work and innovation to supporting specifically the MitySOM. The original team also did not greatly exceed their performance requirements (a selection is shown) [1]:

- The C&DH Carrier Board shall provide all necessary electrical connections between the daughter board and the power board, necessary for the processor to function properly.
- The C&DH Carrier Board shall provide all peripheral ports to communicate with other systems of the satellite – including circuitry to support USB to UART, RS485, and RS422 transceivers.
- The C&DH Carrier Board shall provide a secondary flash memory storage with size of no less than 1GB to supplement the NAND flash on the daughter board. The specifications of this storage are TBD.

The types of interfaces listed here are exactly what the original team included. They did improve on the minimum secondary flash storage requirement with one 4 GB and two 8 GB eMMCs, although the traces were too long leading to degradation in read/write speed. Our project's overarching focus is to innovate and expand upon the work of the original group, while still achieving their minimum requirements.

In order to achieve the original "scalable" design goal, we found it necessary to drop the initial MitySOM board that the group had intended to use. Opting instead for an Octavo SiP that contains the same processor but provides us flexibility in the board design and placement process while also being significantly cheaper to obtain.

The original design displayed tunnel vision in a focus on the three payloads and two radios that the LAICE mission was going to require and as a result provided limited forms of serial communication. Our design takes into the account the advantages of addressed bus communication over I2C and CAN and having a separate BIB, meaning that we can have a very large number of payloads (and radios) subject to power and size constraints. This provides an incredible amount of flexibility to stakeholders.

We believe that the original design made a mistake in not intentionally considering the support of ADCS hardware. The requirements listed in the project proposal [1] included RS485 support specifically for ADCS hardware. However, given that nearly all CubeSat missions will require basic ADCS capability, we decided to incorporate an IMU and magnetometer directly onto our flight board, sitting on the same I2C bus as most of the included components. As a result, we eliminated the RS485 interface, which in practice does not limit the ease of payload interface with the system.

The external storage requirement was also easy to improve upon and we did so with an extra focus on redundancy, providing two 32 GB eMMCs. Thirty-two GB was chosen as a number that was sufficiently

big enough that the second eMMC could be used purely as a replication of the first eMMC's data to guard against data corruption or component failure.

However, the extreme modularity and in-house nature of our new design does create engineering tradeoffs. Our solution now has many components that are purchased individually. With the original design many of the components are integrated as modules which simplified purchase and assembly. By leaving a COTS computer module (the MitySOM) we also lose out on the personal support from companies like CriticalLink which became very valuable to users of the original design.

Creating a new, replacement design for a proven working piece of technology is always a risky endeavor. The original design has been in place for several years and as a result has all its behavior, including quirks, well understood and documented by the engineers using it. A new design is likely to have initial flaws and require board revisions and careful attention to ensure that it works as intended.

3.2 Analysis

Our solution is more flexible and can support more payloads than the original solution at a fraction of the cost. We believe solving this scalability problem for little cost is a core aspect of the improved project.

According to our bill of materials (4.2), the cost of a single fully assembled (with leaded solder) flight board is $266.11. The total lead time would be approximately two weeks. Contrast that with the original solution where the MitySOM module from CriticalLink had a minimum order size of five (despite the fact that the mission only needed one and possible a second for backup), cost $3,000 for the set of five, and had a lead time of six weeks for a leaded solder module. Additionally, the original project still had the carrier board (which was the focus of their design) that had to be printed and assembled for around $3,000 for an order size of ten. This is a massive improvement in cost and lead time which is a critical improvement for any stakeholder, particularly the cash-strapped environment of a university CubeSat program.

4 Cost and Schedule

4.1 Labor

We estimate our fixed development costs at $45/hour for 10 hours/week over 15 weeks (one of the weeks in the semester is spring break), assuming this project would normally be completed in over an entire semester. We then multiply by a factor of 2.5 to account for other overhead in a real engineering organization, such as administrative costs.

$$2 \times \frac{\$45}{\text{hour}} \times \frac{10\text{ hr}}{\text{week}} \times 15\text{ week} \times 2.5 = \$33{,}750$$

Equation 2 - Estimated Cost of Labor

4.2 Bill of Materials

Item	Description	Cost	Quantity	Total Price
OSD3358-512M-ISM	IC MODULE CORTEX-A8 1GHZ 512MB	$52.80	1	$52.80
RM3100	3-axis magnetic sensor suite	$15.50	1	$15.50
DS3231MZ+	IC RTC CLK/CALENDAR I2C 8-SOIC	$7.69	1	$7.69
ISM330DLCTR	INEMO INERTIAL MODULE	$6.02	1	$6.02
LTC2863IDD	RS422 TRANCEIVER	$4.17	4	$16.68
SDINBDG4-32G	eMMC 32GB	$26.70	2	$53.40
TCAN332DR	CAN TRANSCEIVER 1/1 8SOIC	$2.01	2	$4.02
Passives	Miscellaneous Passive Components	$50.00	1	$50.00
PCB Assembly	Automated PCB Assembly	$30.00	1	$30.00
PCB	4 Layer 5mil/5mil 75x40mm	$30.00	1	$30.00
			TOTAL	**$266.11**

Table 3 - Bill of Materials

4.3 Cost Analysis

In quantities of five units at a time, we expect the total bill of materials cost to be about $266 per unit. This estimate is based on PCBWay quotes, so it is likely cheaper than it would truly cost for a flight qualified board. We estimate that a flight ready board would cost about $1,000 based on the more extensive testing and higher quality PCB assembly. This is still far cheaper than the current design which costs roughly $6,000. Our labor estimate comes to $33,750.

4.4 Schedule

Week of	Description	Responsibility
3/15/20	Brainstorm new project ideas	Both
3/22/20	Have new project idea approved	Both
3/29/20	Perform in-depth research and designs to improve project	Both
	Write new project proposal	Both
4/05/20	Begin work on design document	Both
	Improve explanation of project and differences	Dillon
	Begin work on schematics	Adam
4/12/20	Finish schematic	Adam
	Finish design document	Both
4/19/20	Present design document at review session	Both
	Attend peer design reviews	Both
	Begin work on final report	Both
4/26/20	Work on final report – original project section	Dillon
	Work on final project – new project section	Adam
5/03/20	Merge and complete final report	Both

Table 4 - Work Schedule

5 Safety and Ethics

The primary ethical concerns with our project involve ensuring the stakeholder has an accurate understanding of the product we are delivering. In order to satisfy section 7.8.3 of the IEEE code of ethics [3] and section 1.3 of the ACM code of ethics [4], we present a clear description of our project, in particular emphasizing that our flight board on its own is not enough to create a useful CubeSat. A BIB is necessary for a flight configuration of a CubeSat as at a minimum it provides the power to the flight board for operation. A stakeholder would additionally likely want to purchase a radio for communication with the ground.

There are several safety factors relevant to our flight board. The most immediate is the presence of the backup battery for the RTC. This battery is a Manganese Lithium coin cell. It contains very little actual Lithium and has a small capacity (around several mAh) is therefore an extremely safe choice [5].

The remaining safety factors are interactions between our flight board (and the CubeSat enclosing it) and the surrounding environment (launch vehicle, deployer pod, International Space Station (ISS)). NanoRacks, who sells CubeSat deploying services, provides a list of many requirements. Some of the most relevant safety requirements are space debris, battery failure, and structural failure. Our flight board satisfies the section 4.4.6 requirement of not producing any debris [6]. Our RTC battery, as discussed, is very safe. Additionally, section 4.4.7.10 classifies our battery as a “Button Cell” which means acceptance testing is not necessary [6]. To ensure that our flight board does not suffer from any outgassing issues, we will ensure that all materials used satisfy the section 4.4.10.3 requirements [6]. Finally, no components that we utilize shall have issues passing a random vibration test as specified in 4.3.2-1 [6].

6 References

[1] D. Brackmann, M. Mahowald and A. Pasricha, "Flight Computer for IlliniSat-2 - Project Proposal," Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, Urbana-Champaign, 2014.

[2] D. Brackmann, M. Mahowald and A. Pasricha, "ECE 445 Web Board - RFA," September 2014. [Online]. Available: https://courses.engr.illinois.edu/ece445/pace/view-topic.asp?id=8975. [Accessed March 2020].

[3] IEEE, "IEEE Code of Ethics," IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 2 April 2020].

[4] ACM, "ACM Code of Ethics," ACM, 2018. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed 2 April 2020].

[5] Panasonic, "Manganese Lithium Coin Batteries (ML series): Individual Specifications," 2014.

[6] NanoRacks, "NanoRacks CubeSat Deployer (NRCSD) Interface Definition Document (IDD)," 2018.

[7] ST, "Watchdog Timer Circuit," 2017.

7 Appendix

7.1 Full Board Schematic

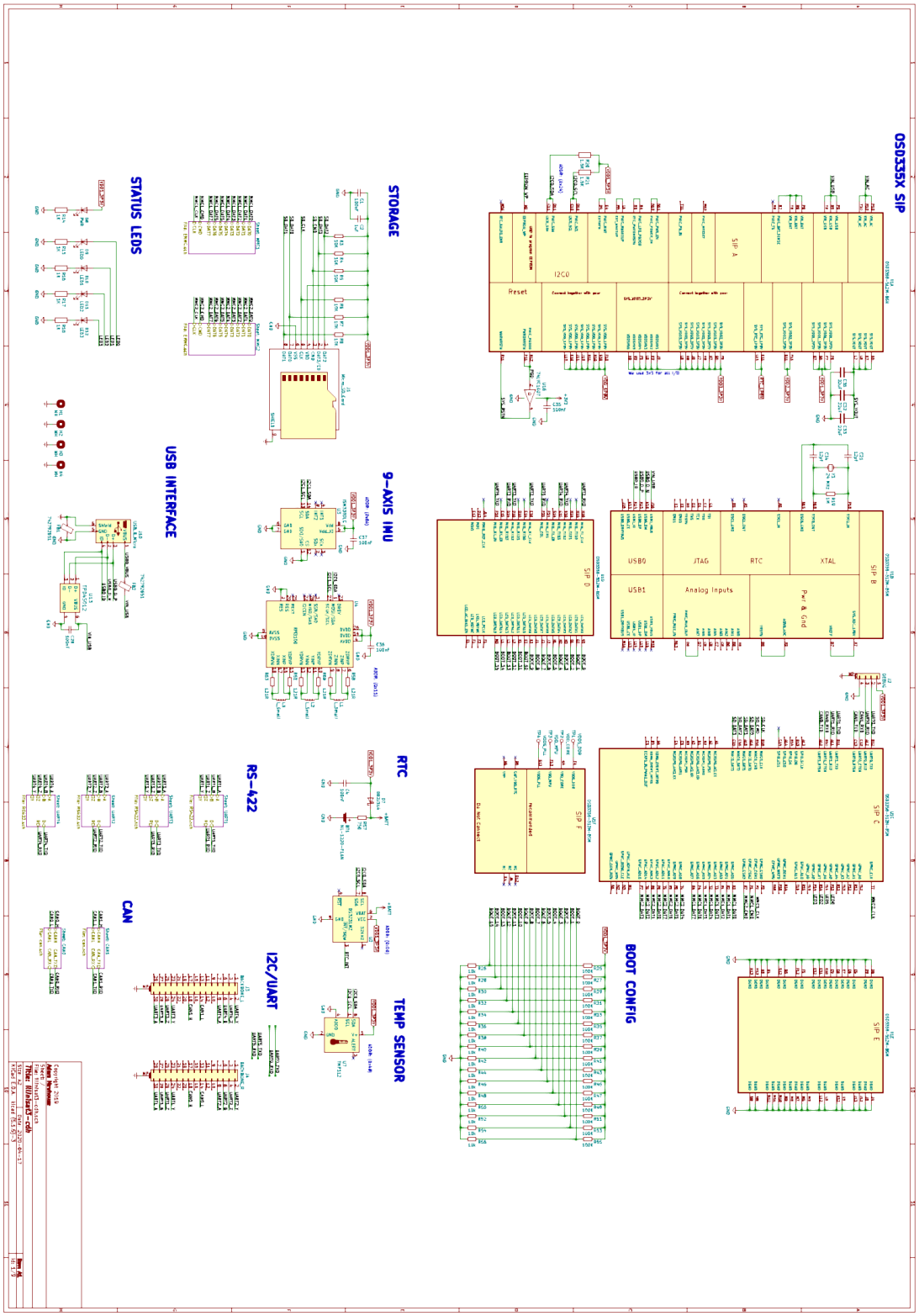

Figure 17 - Full Design Schematic