Flight Computer Board for IlliniSat-2

Final Report

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

Abstract

The IlliniSat-2 small-satellite (CubeSat) bus was developed in the fall of 2014 as a scalable bus for the University of Illinois's CubeSat mission: Lower Atmosphere/Ionosphere Coupling Experiment (LAICE). The problem statement was to design a carrier board that mounts the flight computer and interfaces with the power system, control system, scientific payloads, and radio connections. The use of the IlliniSat-2 flight board was quickly extended past the LAICE mission, but the original design was not flexible enough to be effectively used in later missions. Our new solution integrates a cheaper compute module with common bus components and simplifies the carrier board to a simple, passive two-layer Bus Interface Board (BIB). This redesign is focused on correcting design decisions and oversights made in the original project, reducing overall cost and size of the IlliniSat-2 bus, and providing better performance in a wider array of CubeSat missions.

Contents

1 Project Motivation

Using small-satellites (particularly CubeSats) as a platform to conduct Low Earth Orbit (LEO) experiments is an idea that has grown significantly in popularity since the 1990s. Traditional satellite programs are characterized by multi-million dollar contracts, a development time-span of several years, and complicated collaboration among multiple vendors working on separate but integrated components of the overall system. While this model can be generally acceptable for large-scale NASA or ESA funded space programs with mission lifetimes extending to decades, it is incredibly prohibitive to the larger scientific community whose goals are simpler, on shorter time spans, and financially constrained with less than million dollar budgets. Providing a laboratory that can produce full-fledged CubeSats in a year's time and on hundred-thousand dollar budgets would remove the "biggest impediment to reestablishing a vigorous small scientific satellite program" [1].

The University of Illinois is uniquely positioned to become such a CubeSat lab under the direction of the Laboratory for Advanced Space Systems at Illinois (LASSI) in the Aerospace department. The effort to meet the kind of timing and budget requirements in large part rested on the development of a flexible, scalable, and cost-efficient CubeSat bus: IlliniSat-2. The original design from fall 2014 was an admirable start but was hampered from a tunnel vision bias due to the needs of the LAICE mission and so this original vision was lost. With our improvements upon the IllniSat-2 bus, we demonstrate the clear need for a new understanding of the baseline responsibilities and necessary capabilities of a reimagined CubeSat bus system and provide a new solution, an IlliniSat-3 bus, that can allow LASSI and the University of Illinois to become a leader in CubeSat production for scientific research.

1.1 Problem Statement

This project was originally proposed by Dr. Alex Ghosh for the University of Illinois CubeSat team (later, LASSI). The Illinisat-2 was intended to be 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 to properly fit in the satellite [2]. Developing CubeSat busses is an active area of commercial investment particularly due to the value functional CubeSat programs would bring to the scientific community and others [1]. By providing a system with a Linux based microprocessor, Attitude Determination and Control Systems (ADCS) capability, reliable storage, and multiple methods of payload communication, our design will be attractive to investors, The Laboratory of Advanced Space Systems at Illinois, and the scientific community at large.

1.2 Solution

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) [3] were designed specifically for the original mission, LAICE, that this bus was intended to be used for. 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 have integrated 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.

There are existing companies operating in the same market as we are targeting. Pumpkin Space Systems is a well-known and reliable company. However, they provide a set of convenient and pre-made CubeSat systems. While convenient, their prices exceed our own while often providing less interfaces and a less powerful processor [4]. Our goal is to target a modular design that is low cost but flexible and our bus system would be an alternative to Pumpkin Space System’s inventory of computer modules.

1.3 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.

1.4 Visual Aid

Shown in Figure 1 is a render of the completed board. The main integrated circuit is the System in Package (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.

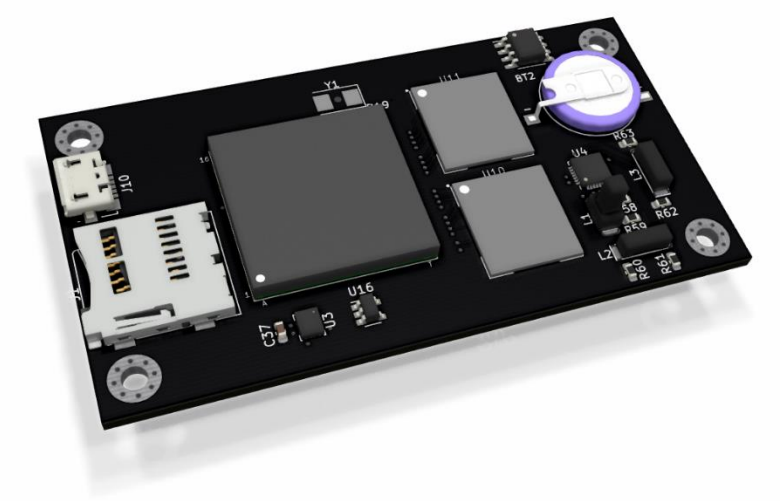

Figure 1 - Flight Computer PCB Render

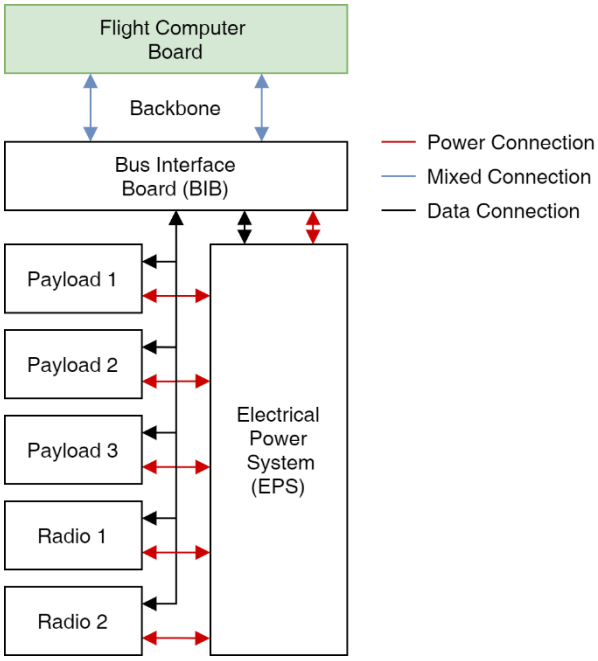

Figure 2 – System Diagram

1.5 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.

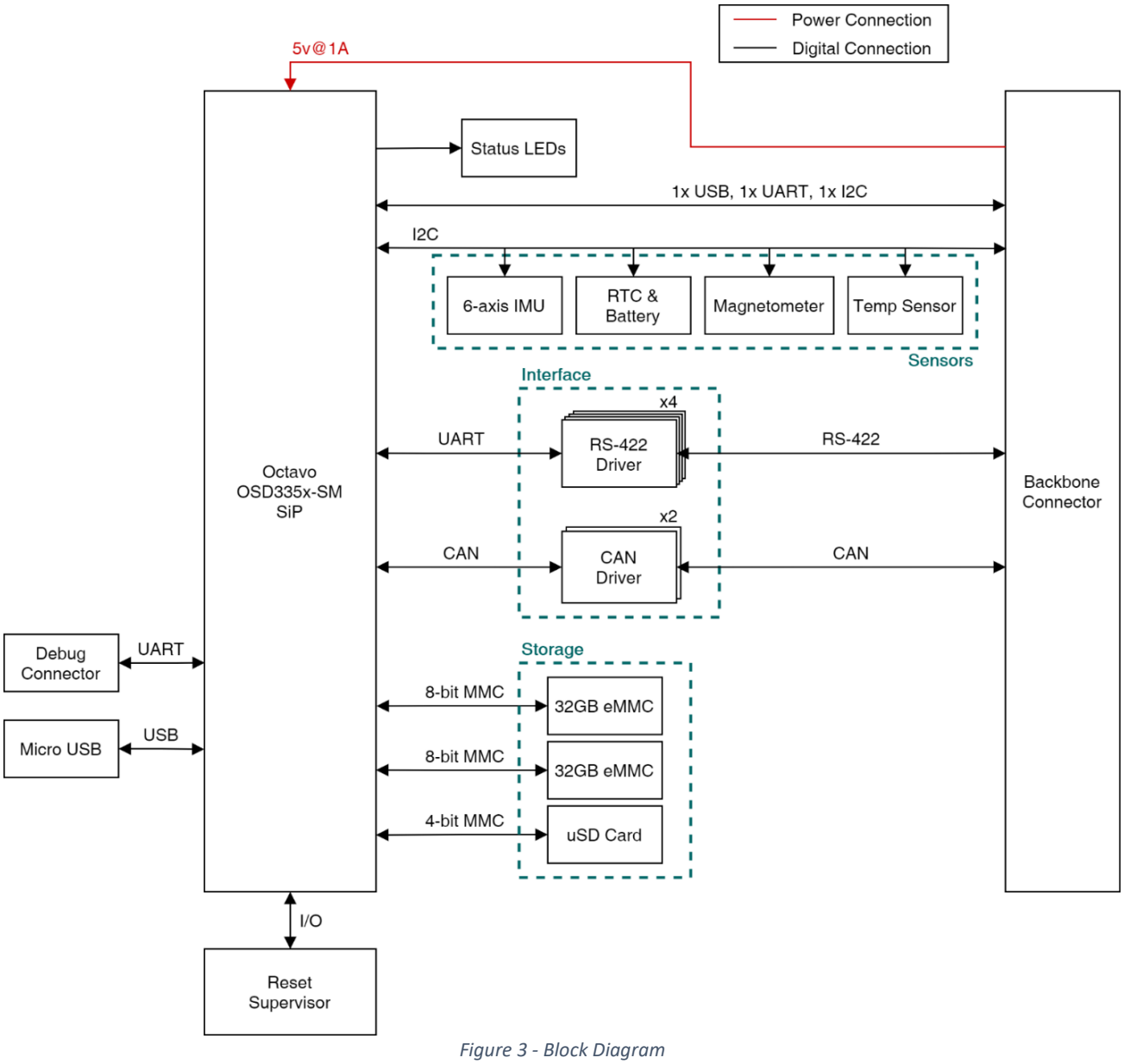

Figure 3 - Block Diagram

2 Implementation

We designed our own flight computer board that has a similar form factor to the original IlliniSat-2 CriticalLink MitySOM module but includes more components and capabilities for less cost.

2.1 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.

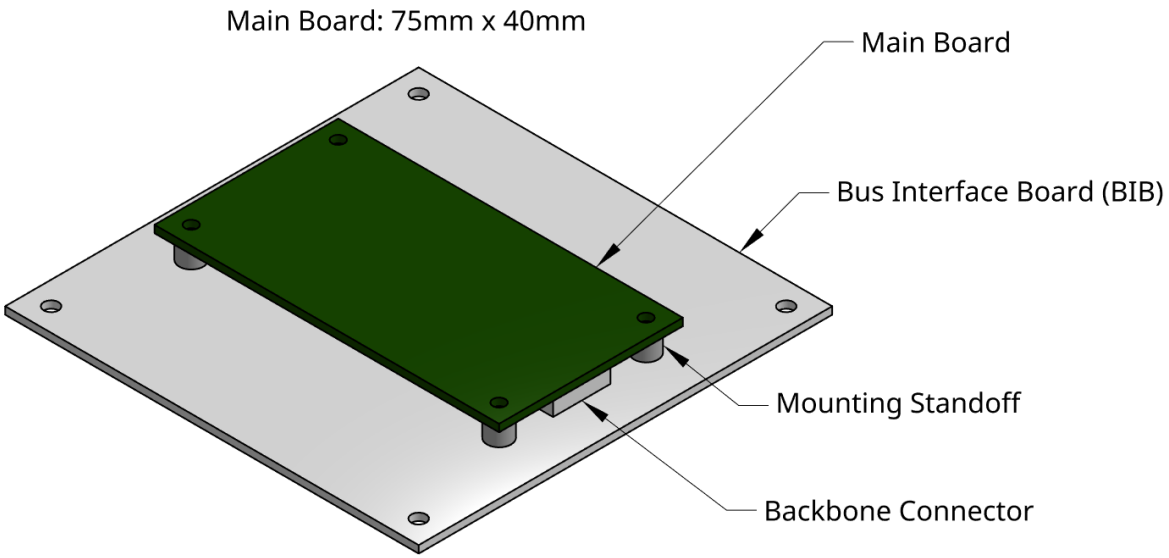

Figure 4 - Physical Design

2.2 Schematic Design

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

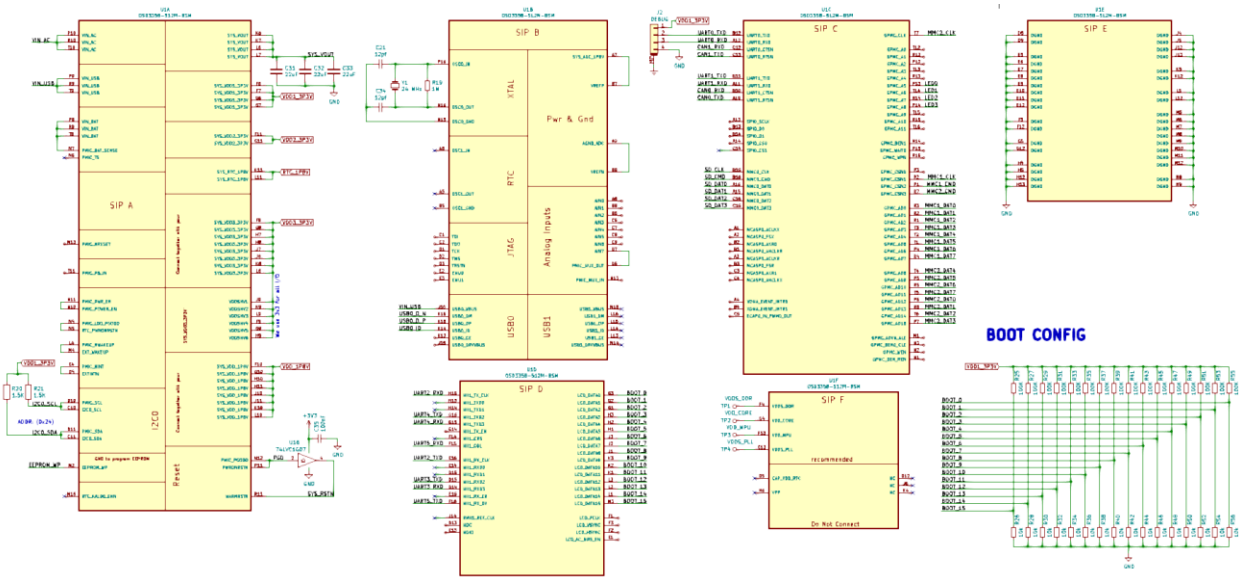

Figure 5 - SiP Connections

2.3 PCB Design

Due to limited implementation time, a full PCB design has not been completed. However, the overall component layout and placement has been finished. This proves that the physical size of the newly designed computer module is adequate for all components chosen. See Figure 1 for a 3D render of the computer module with its associated components.

2.4 Software Toolchain

The most involved software portion of this project was proving that a convenient and reliable toolchain could be established to both boot Linux on the AM335X processor as while as cross compile software to target the ARM Cortex A8 architecture. As the toolchain was developed, it was tested on an AM335X processor, but not the Octavo OSD335-x SiP intended for this project due to resource and time constraints. The differences between the test environment and the SiP, however, are relatively minor related mostly to changing pin mappings between the processor and physical components as well as possible adjusting timing and size values for peripherals such as DRAM or the crystal oscillator. The general procedure and core work is described below, and a more detailed instruction set is provided here https://github.com/dchammond/uiuc-ECE445-Documents/blob/master/updating_linux.txt.

2.4.1 Cross Compiling Toolchain

Software projects benefit greatly from easily accessed build tools and a primary goal for this project was to avoid any vendor specific compiling tools. By creating a toolchain from scratch, it was not necessary to rely on any prebuilt components (such as a specialized version of libc) which meant that it would be possible to simply use GCC as a cross compiler. However, given that build environments may differ (for

example, different Linux distributions) it was important to utilize a consistent and reproducible version of GCC. To achieve this, we used “crosstool-ng” a software suite designed to build C runtimes and a GCC style compiler for specific targets [5]. The AM335X is an ARM Cortex A8 chip with hardware floating point (VFPv3) support. The full config file is provided here https://github.com/dchammond/uiuc-ECE445-Documents/blob/master/crosstool-config.txt.

2.4.2 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 used this.

2.4.3 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 were able to build and boot Linux 4.14.90 which is a Long-Term Support (LTS) kernel until 2024. It also has several AM335X specific patches included by Texas Instruments [6]. 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. The full Linux kernel config is provided here https://github.com/dchammond/uiuc-ECE445-Documents/blob/master/linux-config.txt.

2.4.4 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 also added in an additional device tree that specify the pin mapping of our custom board design. These are then built externally to the kernel into a single binary file. This device tree is provided here https://github.com/dchammond/uiuc-ECE445-Documents/blob/master/am335x-mitysom-illinisat2.dts.

2.4.5 Root Filesystem

Once the kernel is properly configured, the user space root filesystem can be set up. Here we will utilize crosstool-NG again to compile any libraries and programs that we intend to always have on the system. Another open source tool, Buildroot, 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 [7]. The resulting root filesystem is loaded by the kernel on boot. Our buildroot config file is provided here https://github.com/dchammond/uiuc-ECE445-Documents/blob/master/buildroot-config.txt.

2.5 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 1 – 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 (1.0), 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.

$$M_i = -k_i \dot{B}_i \tag{1.0}$$

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 Conclusions

Despite being unable to produce a physical result of our planned design, the design work still presents a significant step forward in the reengineering of an inexpensive, scalable CubeSat flight computer. If implemented in its entirety, this new design would be a significant improvement while not significantly contributing to the overhead of small CubeSat projects in both cost and additionally safety requirements.

3.1 Implementation Summary

Due to the constraints of this semester and no longer having access to normal lab materials, we were unable to build or test most of our hardware in a physical setting. However, we did make significant progress in the software stack and hardware design.

Dillon was able to achieve, using a spare AM335X, proof of concept of a software toolchain to construct the Linux kernel, root filesystem, and GCC toolchain to compile software for the AM335X. This process is a significant amount of work as it typically involves invasive changes to the Linux kernel, as well as a full mapping of processor pins to the appropriate peripherals as described in the device tree. If we were able to order and assemble the board, this software stack and set of instructions would allow testing to quickly commence via programs written to utilize the Linux drivers made available. This means that validation of all the peripherals would be achievable.

Adam was able to complete the entire compute module schematic, including the backbone connector pinout, and all component selection. These are visible in Appendix A. Additionally, the physical design of the solution and the PCB component placement was completed.

3.2 Unknowns, Uncertainties, and Testing Needed.

The most significant blocker is having the equipment to properly test and debug the board. While buying and assembling this board is doable with basic soldering equipment, it represented a financial burden intended to be covered as part of the senior design class. Additionally, testing and debugging the functionality of the board would be nearly impossible without the expensive equipment (such as oscilloscopes and logic analyzers) provided by the senior design lab. If we still had access to these resources, then much of the testing would commence by writing software programs to utilize peripherals like an I2C bus which as hardware components like a real-time clock attached. If the software was unable to successfully complete a test, then we would utilize tools such as a logic analyzer to try and debug the board and find any potential PCB routing or other design mistakes.

3.3 Ethics and Safety

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 [8] and section 1.3 of the ACM code of ethics [9], 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 [10].

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 [11]. 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 [11]. 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 [11]. Finally, no components that we utilize shall have issues passing a random vibration test as specified in 4.3.2-1 [11].

3.4 Project Improvements

The most important piece of future work that we would complete given more time would be to manufacture and test our new design. This would entail ordering boards and then soldering components to them. Once assembled, we would be able to attempt to boot Linux and begin testing board functionality. Through this testing and debugging phase, we would then identify changes and fixes to be included in a second PCB revision. At this point, the second revision could start undergoing testing for flight use. This means that we would begin using it for active missions and run it through a selection of environmental tests.

One additional feature that we would attempt to add to the new solution given more time, is the inclusion of three axis magnetorquer driver. The circuit needed to drive a magnetorquer is a simple h-bridge with inline current sensing. Three of these circuits would drive the corresponding three coils. Based on the current supplied to each coil, and the measured Earth's field, the exact amount of torque

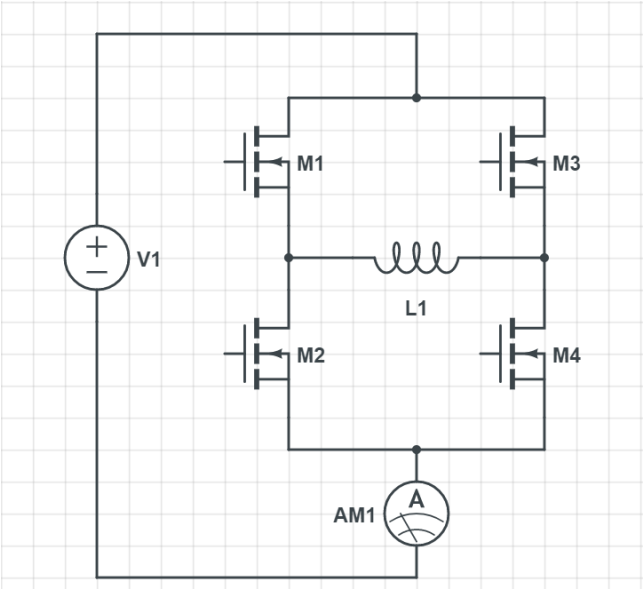

Figure 6 - Magnetorquer Driver

can be controlled and quantified. This is a useful inclusion because almost all CubeSats require magnetorquers for initial detumbling and attitude control.

Given that we intend for this project to be used in future CubeSat missions, another useful inclusion given more time would be to design a simple Bus Interface Board for benchtop testing and debugging. This would also be useful as an example for stakeholders that intend to use our flight computer module. Having an existing BiB makes designing a custom version simpler.

Given significantly more time for our design, it is unlikely that we would entertain alternative designs. The design presented in this report is already the result of personally dealing with the fall 2014 IlliniSat2 design for several years in the LASSI lab. The most likely design change would be to switch out the OSD335-x SiP for a simpler microcontroller based design, possibly a variant of an STM32F. We would consider this design if feedback from possible stakeholders and proposed research products displayed a need for a less power hungry and generally lower power microcontroller.

4 Progress Made on First Project

Since the design review of our first project – an insulin pump for dogs, we have made progress on the software, hardware, and structural components of the project.

Dillon focused on the software components and was able to create a Firebase database that communicated with an Android app which ran on Adam's phone. The Android app was additionally able to search for Bluetooth Low Energy (BLE) devices and print out information about the nearby devices. See Appendix B for a screenshot of the database and Appendix C for a screenshot of the Android app.

Adam was responsible for the hardware components and was also able to 3D print a possible design for the insulin pump. He created a render of what would have been a test version of our PCB which intentionally was in a more convenient form factor and has many test points. This render is available in Appendix D and a fully manufactured board is pictured in Appendix E.

References

[1] D. N. Baker and S. P. Worden, "The Large Benefits of Small-Satellite Missions," *EOS,* vol. 89, no. 33, pp. 301 - 312, 12 August 2008.

[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] 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.

[4] Pumpkin Space Systems, "Motherboard Module (MBM)," May 2020. [Online]. Available: https://www.pumpkinspace.com/store/p49/Motherboard_Module_%28MBM%29.html. [Accessed 6 May 2020].

[5] crosstool-NG, "crosstool-NG Documentation," [Online]. Available: http://crosstool-ng.github.io/docs/. [Accessed 4 2020].

[6] CriticalLink, "Linux MityArm 335x Git," 29 January 2020. [Online]. Available: https://support.criticallink.com/gitweb/?p=linux-mityarm-335x.git;a=summary. [Accessed April 2020].

[7] Buildroot, "Buildroot," 2020. [Online]. Available: https://www.buildroot.org/. [Accessed 4 2020].

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

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

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

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

Appendix A

Full Board Schematic

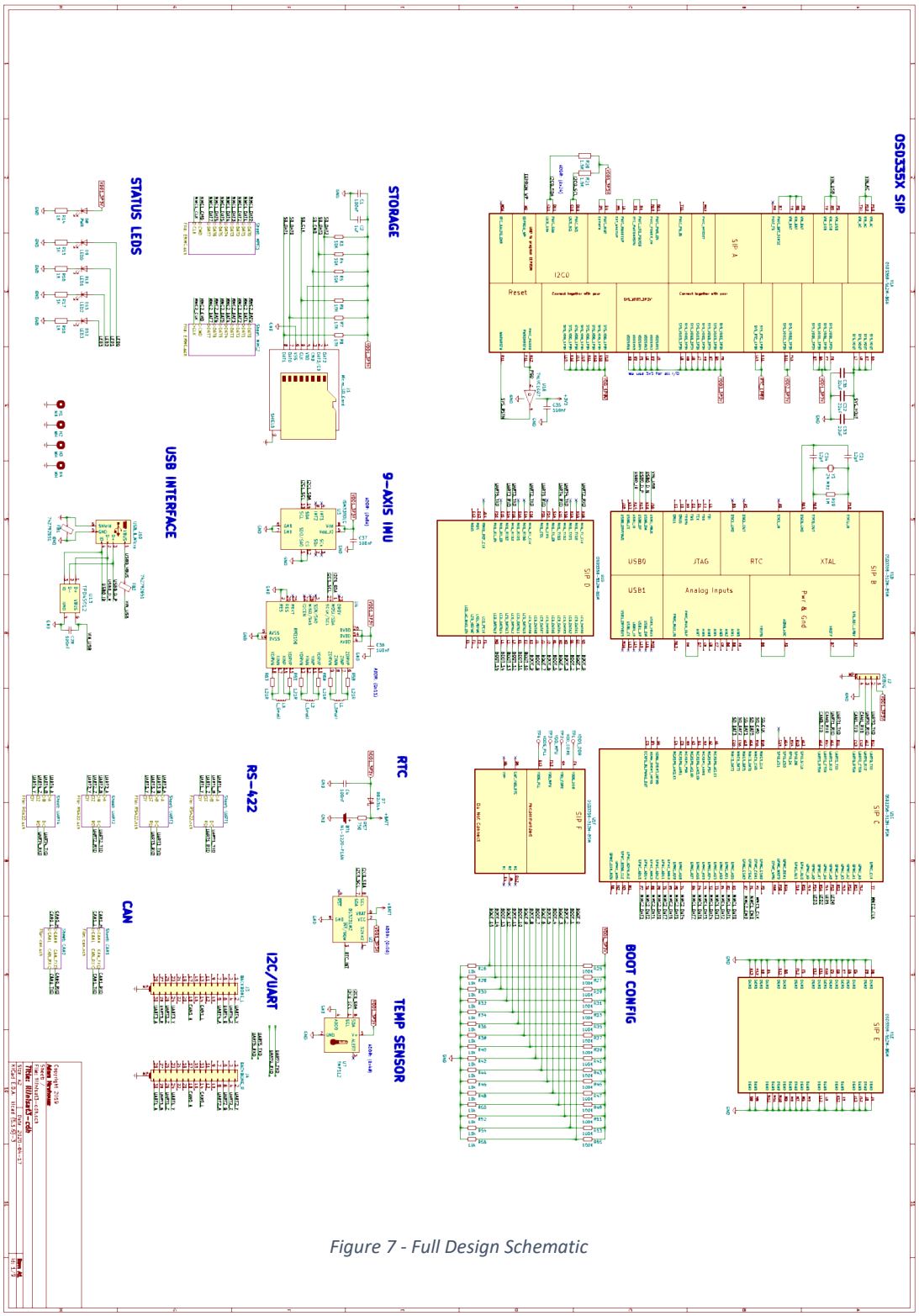

Figure 7 - Full Design Schematic

Appendix B

Firebase Database

Figure 8 - Snapshot of Firebase Database

Appendix C

DogPod Android App

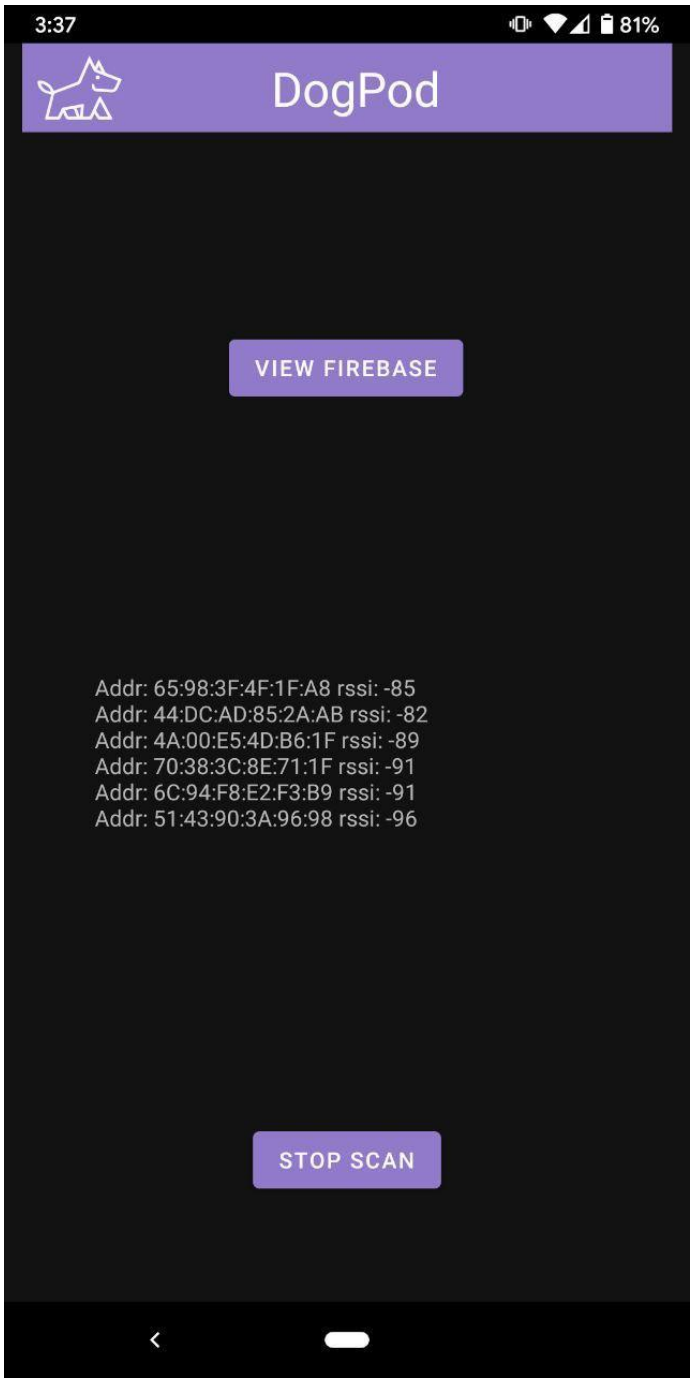

Figure 9 - Screenshot of DogPod Android App

Appendix D

DogPod Insulin Pump Test PCB Render

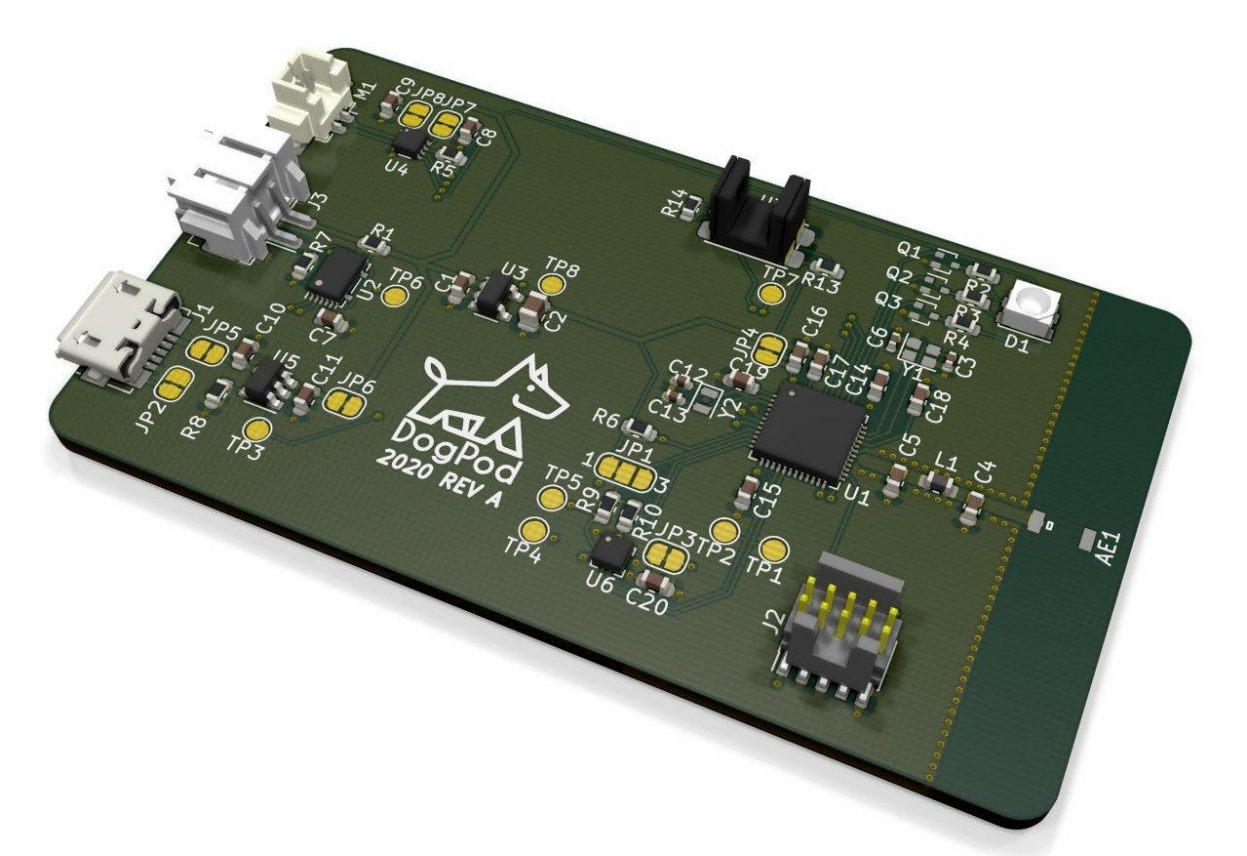

Figure 10 - DogPod Test PCB Render

Appendix E

DogPod Insulin Pump PCB

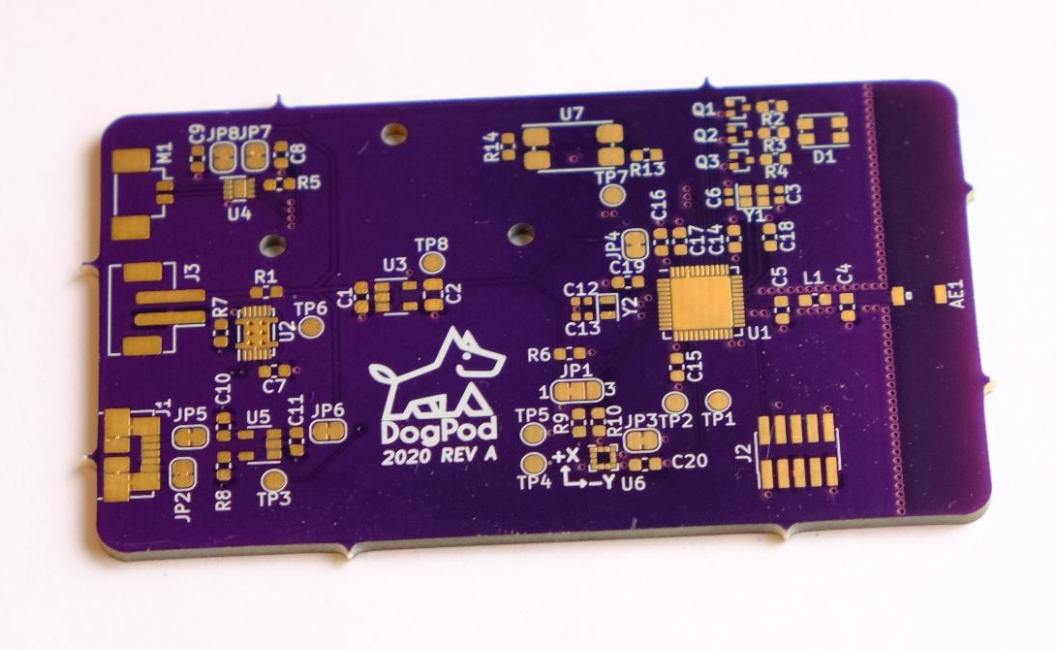

Figure 11 - DogPod PCB

Appendix F

DogPod Insulin Pump 3D Printed Shell

Figure 12 - DogPod Insulin Pump 3D Print

Appendix G

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 2 - Bill of Materials