The Virginia Tech FlatSat: A platform for small satellite testing

Richard F. Gibbons III

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Jonathan T. Black, Chair
Gregory D. Earle
Samantha P. Kenyon

November 14, 2023
Blacksburg, Virginia

Keywords: FlatSat Platform, CubeSat, Testing etc.
Copyright 2024, Richard F. Gibbons III
The Virginia Tech FlatSat: A platform for small satellite testing

Richard F. Gibbons III

ABSTRACT

The small satellite community understands that small satellite missions need to produce two models of the satellite. The flight model is the final model of the spacecraft, which will fly in the space environment. The other, an engineering model, also known as a "Flat-Sat," is a lower fidelity prototype of the flight model. For most mission classes, many design alternatives exist for engineering models and tools to build them. However, this is different for Low Earth Orbit (LEO) CubeSats. Some engineering models are cost-prohibitive for these missions, while others need to offer more features to troubleshoot hardware/software-related issues effectively. This thesis presents a design for a FlatSat motherboard, a cost-effective alternative that allows engineers to build and test their design with flight hardware for LEO satellite missions. First, this thesis will cover the motherboard’s schematic capture and PCB design. Then this thesis will cover functional testing of the motherboard itself. Finally, testing will cover the payload control module, a flight computer to manage all of the payloads on the UtProSat-1 CubeSat mission the payload control module (PCM) the payload computer flying on UtProSat-1. Space@VT expects a dramatic increase in reliably testing all satellite features with this motherboard while drastically reducing the cost for future satellite missions at Virginia Tech.
The Virginia Tech FlatSat: A platform for small satellite testing

Richard F. Gibbons III

GENERAL AUDIENCE ABSTRACT

The small satellite community understands that small satellite missions need to produce two models of the satellite. The flight model is the final model of the spacecraft, which will fly in the space environment. The other, an engineering model, also known as a "Flat-Sat," is a lower fidelity prototype of the flight model. For most mission classes, many design alternatives exist for engineering models and tools to build them. However, this is different for Low Earth Orbit (LEO) CubeSats. Some engineering models are cost-prohibitive for these missions, while others need to offer more features to troubleshoot hardware/software-related issues effectively. This thesis presents a design for a FlatSat platform, that allows engineers to build and test their design with flight hardware for LEO satellite missions. First, this thesis will cover the motherboard’s schematic capture and PCB design. Then this thesis will cover functional testing of the motherboard itself. Finally, this thesis will cover the testing of the payload control module, a flight computer to manage all of the payloads on the UtProSat-1 CubeSat mission the payload control module (PCM) the payload computer flying on UtProSat-1.
Dedicated to Virginia Tech.
Acknowledgments

I would like to thank Dr. Jonathan Black for advising as well as providing me with support throughout my graduate experience and for letting me take on this work.

I would like to thank Dr. Leon Harding for providing the funding and the informal advice-ment on this work and how to write this thesis.

I would like to thank Dr. Samantha Parry Kenyon for providing her infinite wisdom and technical expertise on the hardware development for this project.

I would like to thank Dr. Earle for providing informal supervision on this work as well as his wisdom when it came to navigating my graduate experience.

I would also like to thank Toby Meadows for being a valuable mentor and for working with me in the major design experience at Virginia Tech, and teaching me how to lead with the heart of a teacher.

I would also like to thank Dr. Jaime De La Ree and Dr. Scott Ransbottom for instilling the value of teaching and what it means to help my fellow engineers.

I would also like to thank all of my friends in graduate school for supporting me, listening when I needed to talk, and helping me navigate my graduate experience.

And Most importantly, I would like to thank my family. My sisters, Meghan and Anna for always keeping me humble and my parents, Cynthia and Richard, for supporting me throughout my graduate experience through our weekly Sunday calls.
Contents

List of Figures ix

List of Tables xii

List of Abbreviations xiii

1 Introduction 1

1.1 Problem Background ........................................... 1

1.2 Design Criteria ................................................... 5

2 Review of Literature 7

2.1 Prior FlatSat Designs ............................................ 7

2.2 Validation and Verification methods .............................. 15

2.3 CubeSat Flight Software Frameworks ............................ 19

3 System Design 21

3.1 Hardware Design ................................................. 21

3.1.1 Overview .......................................................... 21

3.1.2 Experiment Management Computer Selection .............. 23

3.1.3 CAN transceiver selection ..................................... 24
3.1.4 Current Sense Design .......................................... 26
3.1.5 Power Connections design ...................................... 33

3.2 Software Design ..................................................... 37
3.2.1 Overview .......................................................... 37
3.2.2 Current Sensing Software ........................................ 37
3.2.3 Remote OpenOCD Debug Configuration ....................... 41

4 Results ................................................................. 43

4.1 Virginia Tech FlatSat Functional Testing: Power Connections ........ 43
4.1.1 Objectives ........................................................ 43
4.1.2 Setup .............................................................. 44
4.1.3 Procedures ....................................................... 45
4.1.4 Results & Discussion ............................................ 45

4.2 Virginia Tech FlatSat Functional Testing: Current Sensing ............. 48
4.2.1 Objectives ........................................................ 48
4.2.2 Setup .............................................................. 48
4.2.3 Procedure ....................................................... 49
4.2.4 Inrush Current Detection Procedure ............................ 51
4.2.5 Results & Discussion ............................................ 52

4.3 UtProSat-1 Payload Control Module Testing .......................... 53
## List of Figures

2.1 Generic Dock FlatSat platform ........................................... 9

2.2 An overview of criteria of previous university FlatSat designs. Virginia Tech FlatSat is the proposed new design. ........................................... 15

3.1 System Architecture of Virginia Tech FlatSat Platform ...................... 21

3.2 CAN transceiver Operation ....................................................... 25

3.3 Current Sense Operation ......................................................... 27

3.4 Current sensing architecture for each slot. Blue and Green differentiate different chip packages. ......................................................... 27

3.5 Percent Current Sense Error versus current to be measured .................. 31

3.6 Percent Current Sense Error versus current to be measured. Every source of error except for quantization canceled out. ......................................... 32

3.7 Simplified Circuit diagram of 3.3 Volt and 5 Volt Bus power [22] ........... 34

3.8 Simplified circuit diagram of power supply for battery bus and unregulated voltages. [22] ......................................................... 34

3.9 Simplified Circuit diagram of Power Supply for Current Sensing circuitry ... 35

3.10 Flowchart of the python program used for real-time plotting of inrush current events. The parallel structure indicates the use of multithreading. For complete Python code, see appendix C. ......................................... 38
3.11 Flow charts of current sensing program. (Left) Program for reading ADC and logging data. (Right) Program for filtering and plotting collected data. For complete Python code, see appendix C.

3.12 Remote SSH connection with Virginia Tech FlatSat Platform

4.1 Schematic of test setup for 3.3V and 5 volt power connections. “O-scope” denotes oscilloscope. The dotted line indicates external equipment.

4.2 Schematic of test setup for unregulated and battery bus voltage connections. “O-scope” denotes oscilloscope.

4.3 Normal Operations with bench Power supply (left) and Under Reverse Polarity (Right)

4.4 Step Response of each power channel (Right) and Turn Off of each power channel (left)

4.5 3.3 Volt Bus Turn On to Reverse Polarity (Right) and 5 Volt Bus Turn on to reverse polarity (Left)

4.6 Unregulated Bus Turn On to Reverse Polarity (Right) and Battery Bus Turn on to reverse polarity (Left)

4.7 Circuit Diagram for Setup for testing overall current sense

4.8 Resistor Setup for testing overall current sense

4.9 The location of the power channel pins on a PC/104 slot. The pinout is the same for all four slots.

4.10 Inrush current detection testing setup with 470 uF capacitor
4.11 Example graph showing the current pulses for the 5V bus, turning on to 5.6-ohms, 56-ohms, and 560-ohms

4.12 Inrush currents versus time. Switched eight times, the system only detected seven inrush current events. This was done on channel 0 for slot PC1.

4.13 System block Diagram of the PCM.

4.14 PCM testing setup

4.15 OpenOCD failing to Halt the PCM (Left) OpenOCD successfully halting and flash programming the PCM (Right)

4.16 PCM white wire fix on back of the board

5.1 Original system architecture (Left) Proposed system architecture (Right)

5.2 Testing setup for inrush current detection with a resistor

5.3 Testing setup for hard turn-on of the PCM
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Specifications for the Experiment Management Computer Selection [35]</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>INA4180 Electrical characteristics [5]</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>ADS8344 Electrical characteristics [4]</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Sources of error in the current sensing circuit design [4], [5]</td>
<td>30</td>
</tr>
<tr>
<td>3.5</td>
<td>Electrical characteristics of the LM338T</td>
<td>36</td>
</tr>
</tbody>
</table>
List of Abbreviations

ADC  Analog-to-Digital Converter
AIV  Assembly, Integration, and Verification
CAN  Controller Area Network
CPU  Central Processing Unit
DIP  Dual In-line Package
DMA  Direct Memory Access
DUT  Device Under Test
EPS  Electric Power Supply
FPGA Field Programmable Gate Array
GaN  Gallium Nitride
GPIO General Purpose Input Output
I2C  Inter-Integrated Communication
IDE  Integrated Development Environment
JTAG Joint Test Action Group
LDO  Low Dropout Voltage Regulator
LEO  Low Earth Orbit
MCU Microcontroller Unit

OBC On-board Controller

OCD On-Chip Debugger

OS Operating System

PCB Printed Circuit Board

PCM Payload Control Module

RBF Ready Before Flight

SAR Successive-Approximation

SPI Serial Peripheral Interface

SSH Secure Shell

SWD Serial Wire Debug

TVS Transient Voltage suppression

UART Universal Asynchronous Receiver Transmit

USART Universal Synchronous Asynchronous Receiver Transmit

USB Universal Serial Bus
Chapter 1

Introduction

1.1 Problem Background

CubeSats are small satellites that come in units (U) of 10-centimeter cubes. CubeSats range from 1U to 12U to suit individual mission needs. A CubeSat’s small form factor and low weight offer rapid development and deployment. This makes them favorable for technology demonstration missions where new technology must be tested in space before being adopted for larger-scale missions. Although CubeSats are small, the technology they comprise is very complex. Each CubeSat will typically have several subsystems to accomplish mission objectives. These subsystems are essential for a satellite’s operation and performance, playing specific roles in communication, power, navigation, and more.

As of 2018, 855 CubeSats have launched into Low Earth Orbit (LEO), with 21% failing to complete their mission [40]. To make more of these missions successful, more rigorous testing of the embedded systems on the spacecraft must be conducted before hand-off to the launch provider. As the CubeSat specification states, “It is always cheaper to build two models instead of one and then realize that two were needed [23].” Decker [16] also finds that small satellite missions for which an engineering model was constructed beforehand had a greater probability of mission success than those without. There is a greater probability for mission success because an engineering model of a spacecraft allows engineering teams to capture errors in the hardware and software design before building the flight model. An engineering
model will enable designers to validate that software functions correctly at the component or system level. It allows designers to troubleshoot hardware as though it were a part of the spacecraft. The ultimate objective is to validate that the subsystems will work before integration with the chassis. Therefore, a practical engineering model is needed to improve the reliability and performance of the spacecraft. Over the years, Virginia Tech has been developing CubeSats to demonstrate new technologies in the space environment.

The Virginia CubeSat Constellation mission was a three-satellite constellation aiming to collect data on properties of the atmosphere as well as collect data on the atmospheric drag profile of CubeSats in Low Earth Orbit (LEO) [18]. One of the three satellites, known as Ceres, was produced by undergraduate students at Virginia Tech. The satellites were launched in 2019 from the International Space Station. In Gargioni et al. [18], the authors discuss the technical challenges of the mission and the lessons learned during the development of the satellite. Some of their recommendations were that proper maintenance of the Electric Power Supply (EPS) is critical to ensure that the satellite has sufficient charge before launch and that space environment testing must be conducted on the flight model to ensure no subsystems fail during launch or orbit. A key recommendation they make is to assemble the spacecraft only after the flatsat testing is successful. The authors clearly state that a proper flatsat is essential as software and hardware issues necessitated the disassembly of the Ceres CubeSat.

The Virginia Tech ThickSat mission aimed to test the passive deployment of a tape spring boom in the space environment. Confirmation of this would be accomplished by downlinking an image of the fully deployed boom down to earth [19]. The satellite was delivered to Northrup Grumman in October 2020. Gargioni et al. [19], discuss the challenges in the development of the deployer mechanism as well as testing the subsystems on the CubeSat as a whole. One key technical challenge lay in the power budget for the satellite. The authors
1.1. Problem Background

did not realize that their original main computer, the Raspberry Pi 3, had a steady state current draw of 500mA. This meant that their satellite bus could not support the power draw and that they would have to go back and design a custom low-power computer. Ensuring that the satellite bus can support the power requirements of all subsystems is an important engineering challenge for small satellite teams.

The Ut ProSat-1 mission aims to test three different technologies in the space environment. The first was the deployment of a tape spring boom from NASA Langley in the space environment. The second is the test of the radiation resiliency of phase change memory modules. The third is to successfully demonstrate S-band communications with the VT ground station [43]. At the time of this writing, the satellite will launch in August of 2024.

Following the lessons from prior Virginia Tech (VT) CubeSat missions, the development of the VT FlatSat platform was motivated to address key challenges in power budget validation and subsystem testing for future CubeSat missions out of VT. A proof-of-concept would be developed to test all subsystems on the Ut ProSat-1 mission.

Previous approaches to building an engineering model involve using a motherboard to house industry-standard CubeSat subsystems. These motherboards, also called “FlatSat Platforms” in this thesis, come with some supporting electronics to power and troubleshoot data bus communication on CubeSat hardware modules. The significant drawbacks to those FlatSat platforms are that they are cost-prohibitive for most university satellite programs. On the other hand, some FlatSats are more cost-effective. However, they have fewer features. For example, FlatSat Platforms must allow for sequential testing of embedded systems on Spacecraft. This sequential testing means that every new hardware module is tested before being connected with the rest of the Spacecraft and tested again. Sequential testing allows the designer to isolate whether or not the error comes from the synthesis of two components or if the error came from an individual component.
Another primary concern to the designer is power management on a spacecraft. If the dynamic power draw on the Spacecraft is not managed correctly, the electric power supply will not be able to meet the power requirements of the rest of the electronics. To mitigate this risk, designers create a power budget that lists the power draw of all the electronics on the Spacecraft. The challenge is that one can only estimate the hardware’s worst-case max power draw for custom or commercial hardware. Trying to estimate the dynamic power draw based on the concept of operations proves to be even more difficult [19]. Current sensing on the power channels of the hardware is necessary to model and test the dynamic power draw accurately. None of the previous FlatSat designs known to the author have supporting electronics to accomplish this.

The networked debugger is another feature that has yet to be added to previous approaches known to the author. A networked debugger allows multiple software engineers to test and debug software systems on the Spacecraft’s hardware without moving the hardware from one place to another. Most past approaches require the software engineer to be close to the hardware to test software systems. With a networked debugger, the software engineer can remotely connect to the FlatSat and troubleshoot as needed.

With several satellites being dead on arrival and looking at the engineering challenges faced by the VT CubeSat team, motivation for a FlatSat testing approach was found. The VT FlatSat will incorporate supporting hardware to address the key challenges of accurately measuring power draw and remote connection capabilities. The next section will cover assumptions and further problem definition in more technical detail.
1.2 Design Criteria

Based on the gaps discussed in Section 1.1, the requirements for an improved design are enumerated here. The system design section of this thesis will cover more technical implementation details.

1. The design shall accommodate multiple PC/104 slots to connect multiple hardware components.

2. The design shall include remote Secure Shell (SSH) capability.

3. The design shall communicate on the common PC/104 data channels as if it were another piece of hardware on the spacecraft. These common data channels are two SPI buses, two I2C buses, one UART bus, one USART bus, and two CAN busses.

4. The design shall include over-voltage and over-current protection.

5. The design shall include integrated remove-before-flight (RBF) and kill switches to test power on and off the spacecraft.

6. The design shall facilitate power monitoring of individual channels on each slot to allow for the construction of improved power-usage profiles.

Given the time constraints of this research work, the following assumptions are imposed to reduce the scope of the work.

1. The design shall have a maximum of four PC/104 slots. This number handles most 1U-3U CubeSat hardware stacks. Any further additions will increase the real estate of the motherboard.
2. The design shall have power connectors so that an external bench power supply can be connected to supply power to the main bus. The power supply would be used to substitute the Flight EPS when testing other subsystems. It also uses the safety features built into the bench power supplies, such as soft turn-on.

3. The scope shall be restricted to functional testing of power and command and data handling. Much literature exists on testing attitude determination and control systems (ADCs). Adequate testing of power supplies requires additional hardware specifically designed to emulate solar panels.

The literature review of this thesis will cover prior flatsat designs and the methods of how flatsats are used as tools for the rapid development of CubeSats. This is to ensure that this design for a FlatSat builds on what has already been done rather than recreating an existing design.
Chapter 2

Review of Literature

2.1 Prior FlatSat Designs

A FlatSat is a familiar idea in the CubeSat community. Multiple universities, companies, and organizations have developed FlatSats for more mature spacecraft testing. Before building a new FlatSat system, it is essential to review the initial designs to see what did and did not work or prove not to be helpful to CubeSat teams. This component of the literature review determines what feature set the Virginia Tech FlatSat platform would offer to CubeSat developers.

It is essential to distinguish between a “FlatSat” and a “FlatSat platform.” A FlatSat platform is a tool used in building a FlatSat, which is an engineering model of the spacecraft. It houses the individual components that comprise a CubeSat. The term “FlatSat platform” is overloaded in the satellite community as a FlatSat platform design varies from one CubeSat mission to another. Barcellos et al. [7] review all the literature that use this term to describe this kind of ground support equipment used for CubeSat missions. The authors provide a helpful classification of the different kinds of FlatSat platforms used in the CubeSat community.

In their classification, Barcellos et al. [7] notice several characteristics for what they call “back-plane” FlatSats. The first characteristic is passive versus active FlatSats. Passive
FlatSats offer only the connectors and little additional circuitry, whereas active FlatSats offer additional circuitry for voltage regulation or measuring purposes [7]. The other characteristics ask specifically what features the FlatSat offers. Does the FlatSat offer sensors to measure power consumption or temperature? Does the FlatSat offer connectors to allow for the use of an external power supply? Are there switches or jumpers for other applications, such as triggering faults? Does the FlatSat offer a complex programmable logic device (CPLD) to monitor the main bus? The CPLD characteristic suggested that it could be further classified to the use of microcontroller units (MCUs) or field programmable gate arrays (FPGAs) to monitor their main bus.

The literature review of this thesis will consider active FlatSat platforms falling under the category “Docks.” Docks interface the individual components of a CubeSat through a printed circuit board (PCB) backplane, and the FlatSat platform houses those components [7] as shown in Figure 2.1. The drawback of these platforms is that they cannot be divided into several parts like the modular FlatSat platforms. However, they offer an advantage over Raw FlatSat platforms, where individual components of the spacecraft are connected via wires since the setup of raw platforms is prone to user error.

Reilly et al. [36] present the design of a dock FlatSat platform, which they call EIRFLAT-1, to capture errors and unexpected behaviors in individual components for the EIRSAT-1 Mission. They provide a brief overview of additional pieces of ground support equipment and an overview of their test campaign. They describe using a pseudo-electrical power system to test other flight hardware for the spacecraft. Using an engineering unit like this removes the risk of over-cycling and damaging the power supply with untested components [36] (i.e., the payload board shorts to the ground). The authors propose a solution where they design custom PC/104 (PC/104 refers to the interconnect standard used to connect hardware within CubeSats) hardware to plug into the dock FlatSat to emulate actual hardware components
Morea Font \cite{30}, conducts a validation and verification program of the flight software that will go on the onboard controller (OBC) for the 3Cat-4 CubeSat mission. They use their FlatSat platform to power on and flash program the processor in a thermal vacuum chamber. A significant insight of Morea Font \cite{30} is having the supporting power and debug circuitry built outside the thermal vacuum chamber and connected to the FlatSat inside. While this is outside the immediate scope of this research, this understanding could prove helpful for future iterations of the FlatSat platform. What the author aims to accomplish is to validate the control software. However, the FlatSat docks could have been made with additional PC-104 slots to accommodate more hardware in the thermal vacuum chamber. A nice feature that could also prove helpful for future iterations of the FlatSat is the dashboard visualization running on another computer. The ground support software provides visuals of all significant signals on the ADCS System. This dashboard is viewed through another
de Souza et al. [15] conduct functional testing of an onboard controller using a mini-thermal vacuum chamber. They then use a FlatSat to conduct integration testing with other satellite components, which allows them to explore the environmental testing of individual components using the mini thermal vacuum chamber in greater detail. The authors could design a dock FlatSat platform with a single PC/104 slot and a connected cable outside the thermal chamber. The FlatSat used in this integration testing was a passive backplane with a linear array of PC/104 slots. This layout offers more direct routing as traces do not have to bend or curve from one PC/104 slot to another. However, there are no active components on the FlatSat to aid in debugging software. Most of that is off-board. Overall the authors did have an excellent testing plan to bring up the board and to validate expected behavior.

Praks et al. [33] try to solve the functional and integration testing of their custom hardware for the Alto-1 CubeSat mission. The paper presents everything from the mission requirements down to the design specifics of each component. In section 7 of the paper, they present their FlatSat platform. Praks et al. [33] propose a FlatSat implementation for their mission. The FlatSat model they designed has six PC104 slots, just enough to accommodate all the hardware components used on their specific CubeSat. It is a passive FlatSat platform with power connections and Dual in-line package (DIP) switches. What the DIP switches do remains to be clarified, as the platform was not the focus of this paper.

Haber et al. [20] design and build a CubeSat to test quantum key distribution experiments in space where they use attenuated light pulses to communicate encrypted information down to a ground station. They used custom hardware to build the satellite bus for this CubeSat. In the laboratory-based testing section of the paper, they discuss the UNISEC board, which is a custom-designed FlatSat platform dock. Its design accommodates the satellite bus from the UWE-3 mission [13]. Haber et al. [20] design is different as it has a small form factor, so
it only takes up part of the lab bench for testing. The only drawback is that the boards are mounted perpendicular to the work surface, so it is challenging to probe different voltage points on any device under test.

Eugene et al. [17] build a small satellite for the Glassia Mission to host two experiments in LEO: A payload to record the total electron content in the ionosphere and a secondary payload to test quantum communication links where the payload generates photon pairs to establish a communication link between to sites [17]. The team designed a FlatSat platform to construct the engineering model in the satellite assembly, integration, and testing. It has a port to supply power to the engineering model and a series of ports to connect to the data bus. The FlatSat platform is just a passive backplane connecting the boards. If testing requires additional functionality, the authors design another PC/104 board to plug into the Engineering Model to provide feedback.

Tumenjargal et al. [38] propose using a complex programmable logic device (CLPD) or a Field Programmable Gate Array (FPGA) to program new interfaces in software rather than redesigning the backplane PCB for every test as the interfaces for different components on a spacecraft can change from one satellite to another. Some significant insights from this paper are that it highlights the disadvantages of the PC/104 bus originally proposed by Pumpkin Inc. [2]. They note that a smaller connector with fewer pins is more desirable than the larger PC/104 connector. In addition, using an FPGA in the backplane to remap pin assignments would save time and energy in developing a CubeSat.

Busch et al. [13] provides coverage of the lessons learned from developing their custom satellite bus. Most of the paper focuses on the bus architecture’s performance in the space environment and the residual magnetic dipole in the satellite. Busch et al. [13] mentioned the need for more elaborate bus testing in the introduction, which would have captured many errors in the development process. The UWE-3 bus architecture uses a backplane to
connect all of the individual components of the spacecraft, so it is a dock FlatSat platform. The advantage of this is the reduced amount of space on the lab bench, which already serves as part of the spacecraft. However, it is still only a backplane for connecting components.

Monteiro et al. [29] conducts a testing program on their educational satellite platform from the ISTSat-1 CubeSat Mission. Where they conduct radiation testing on all of their components for the spacecraft, this paper offers several insights, such as introducing the “Vee” model for spacecraft development. The model comes from the systems engineering field that describes the steps of development from the user requirements all the way down to acceptance testing. They also present the system architecture for their FlatSat platform and electrical ground support equipment [29]. Their platform offers a wide range of capabilities, from network access to the FlatSat to Electric Power Supply (EPS) emulation [29]. This system architecture is what informed many of the objectives for the Virginia Tech FlatSat platform. In addition, their FlatSat is modular, allowing the user or the developer to add or remove any modules from the engineering model. While it is not a dock FlatSat, this design is included in the literature review as it offers plenty of capabilities.

Kiesbye et al. [24] develop a FlatSat platform to do hardware-in-the-loop simulation for validating their CubeSat design for the MOVE-II mission from the Technical University of Munich. They develop several subsystems to model various sensors and solar arrays as they perform in the space environment. The authors also develop a universal interface node using what looks like three Nucleo™ development boards from ST-Microelectronics™ to communicate with any of the subsystems on the spacecraft. This work gave the idea of peripheral expansion for the Virginia Tech FlatSat platform. Instead of three development boards, the platform can use just one to communicate with all the PC/104 interface data buses.

Corpino and Stesina [14] attempt to validate and verify all subsystems on the E-ST@R
satellite. They present an overview of the verification program and its platform design. Some insights from this paper are that the authors present the system architecture of all of their components, from the EPS to the ADCS system. This system architecture offers an alternative design to satellite buses that has the potential to emulate the system behaviors of various components (e.g., an engineering radio model that can emulate multiple radios on the market). The platform (or, as they refer to it, the simulator) is effective as it uses a power pack to power on the device under test and a ttyUSB interface to communicate with any given device under test (DUT). This drastic reduction in complexity for a platform makes it easier for the user or the developer to test various commercial satellite hardware.

Radu et al. [34] from Delft aim to develop a platform for development in three key areas: education, technology demonstration, and further innovation. They use a different form factor than the traditional PC/104, which is much smaller. The smaller pico satellite form factor [25] allows the board to house 6-7 satellite components within a small area. Their FlatSat platform would also lead to cheaper satellite development as they are targeting the development of smaller satellites. However, this means that they are incompatible with the industry standard components as the PC/104 specification requires subsystems to be 10-by-10 centimeters and not 50-by-50 millimeters. Regarding its feature set, the board offers external power connections and slots to add additional hardware besides the satellite components.

Sabogal et al. [37] develop a custom FlatSat to run super-computing experiments for video and image processing on spacecraft. The engineering model they develop uses the CubeSat space processor by Wilson and George [44] along with a daughter board to run a Gallium Nitride (GaN) power converter experiment. The platform they develop uses connectors for external cameras. The platform does not follow the industry standard use of PC/104 connectors, and they use some other connectors only compatible with their custom hardware.
The board uses an FPGA to remap the pin assignments for data bus communication from one satellite mission to another. While the platform they develop suits their custom hardware well, it must still be compatible with commercial hardware. Using an FPGA to remap physical interfaces proves helpful when changing from one satellite mission to another.

In this review of the literature for prior FlatSat designs, several consistent design features are evident. First, prior designs use basic power electronics to generate fixed voltages from a bench power supply or use several bench power supplies to power all of the power input channels to the devices under test. Second, these FlatSat platforms use the PC/104 back connector. While [11] found that participants from their survey thought the PC/104 standard is too large, but it still is the industry standard. For CubeSat teams to use commercial hardware in their designs, their platforms must support this hardware. The key differences across these designs are the number of programmable logic devices used. Some are passive backplanes with few active components, while other FlatSat platforms host MCUs for modeling and simulation.

Several university and commercial designs have been reviewed following the criteria proposed in Barcellos et al. [7] with an additional criterion of remote SSH program capability. Table 2.2 shows some key criteria that highlight differences in these platforms, along with the new design in this thesis.

The Virginia Tech FlatSat Platform offers a more comprehensive set of these features for a FlatSat platform. It offers unique remote SSH capability at a cost similar to the other platforms.
Validation and Verification methods refer to the tests conducted on CubeSat engineering models to qualify them to work in space. Reilly et al. [36] summarize the testing conducted in one of these validation and verification campaigns. They start with functional testing of components of the CubeSat (i.e., Radio, EPS, and OBC), where they verify the functionality of that individual component before integration with other components. Integration testing is where individual components are connected to each other and tested again. Environmental testing takes this flight simulation to a more rigorous level in which the user gathers data while the engineering model is subjected to stress testing that simulates the space environment. These conditions include vacuum levels consistent with the nominal flight altitude range, radiation levels commensurate with those expected for the nominal orbit and expected solar flux levels, and thermal cycling in vacuum over the expected operational temperature range. A complete understanding of the problems FlatSat testing platforms aim to address is formed by examining how satellite developers use them in these campaigns.
Bouwmeester et al. [12] investigate whether the reliability of CubeSats over their lifetime can improve through testing and by creating designs with more subsystem redundancy. This paper argues that improved testing wins over redundancy when it comes to improving the reliability of CubeSats over their lifetime. This claim is based on the data collected from their survey of entries in the CubeSat Failure Database. While adding redundancy reduces risk by eliminating single points of failure in the design, it can add to the monetary cost of the design as well as take additional time to design. [12]. They use a Kaplan-Meier estimator to model the reliability for CubeSats and subsystems [12]. The flow graphs for their reliability modeling simulations could be an exciting starting point for more reliable modeling of CubeSat hardware failure points.

Morris et al. [31] use a satellite mission, STF-1, to demonstrate the capabilities of a digital twin, or a software emulation, of their hardware developed by the NASA independent test and capability team. The problem they are trying to solve is how to build digital twins of the hardware to lessen the reliance on commercial CubeSat hardware for software testing. The critical insight of this paper is that a satellite developer can use NOS³, the software developed by the NASA team, to build a software abstraction layer for more unique protocols such as universal asynchronous receiver transmit (UART) and inter-integrated communication (I2C). This way, software developers can design in parallel to the hardware development. While it has been adapted to many missions (STF-1 and James Webb), building the software on a Raspberry Pi could be complex and time-consuming as a Raspberry Pi has limited processing power. Furthermore, troubleshooting software dependencies will be problematic if the correct environment variables are not set.

Walsh et al. [41] aims to build a 2U CubeSat to test three novel payloads: the gamma-ray module, a thermal coating management experiment, and a novel attitude control module. To accomplish this, they adopt a prototype design philosophy, which means building two
models of their spacecraft. They outline the assembly and verification process used to test all subsystems on the spacecraft. The outcome of this work is the assembly, integration, and verification process to confirm that the satellite will work as intended in the space environment. Before performing environmental testing, they undergo two test campaigns: a functional test campaign and a mission test campaign. The functional test would test each payload and subsystem (i.e., flashing code to the system and testing communication). The Mission test would test all the components together under different scenarios (i.e., Nominal operation, hard reset of the spacecraft, etc). The authors plan a rigorous testing campaign to capture any errors made in the design of the spacecraft. They start with a round of functional testing where the complete function of the satellite is verified. This includes testing electrical, mechanical, and optical functions on each subsystem [41]. Then, they move on to mission testing, where the authors verify that the satellite can carry out mission operations. They do this by running mission operations as though in-flight with ground support equipment. Finally, the authors complete a round of environmental testing which involves vibrations and thermal vacuum testing of the engineering model. They are consistent with the other literature in advocating for building two spacecraft models: An engineering model and a flight model.

Mariano et al. [27] aims to cover the development process for their open-source CubeSat bus solution. They adopt a gradual engineering process where they build multiple spacecraft models. Between each model, they conduct a major environmental test. The critical insight of this paper is that for a mature satellite design to be possible, the developer should take smaller, more gradual steps in the engineering development process. This way, each phase of the development is subject to fewer errors. The authors propose an excellent solution for building mature flight hardware. However, the university-class missions have short lifetimes and small budgets incompatible with this engineering process as they typically do not have
the required resources and facilities. The authors performed a sounding rocket experiment to
test one of their engineering models, which is also cost-prohibitive for a university CubeSat
program.

Decker [16] look at multiple approaches to building and testing CubeSats to determine how
a group spends the least money on building a small satellite that works in the space environ-
ment. Decker [16] determines that a satellite mission building two spacecraft models is more
successful than one, which justifies having an engineering model of the final flight configura-
tion. Another insight from this thesis is that it covers what it calls “beta build strategies”. 
These are built approaches aimed at new satellite developers to yield greater success with
getting their CubeSats fully operational [16]. Strategy three in the thesis involves prolonged
testing of the spacecraft on the ground before moving to the flight model. The author pro-
vides a cost analysis to show that strategy three is the next cheapest alternative besides
building just one satellite and having it be a total success.

Borowicz [10] conducts a failure modes, effects, and criticality analysis (FMECA) of the
SeaLion mission from Old Dominion University. It adopts MIL_STD-1629A for developing
the FMECA. The SeaLion mission had several custom hardware pieces on the spacecraft, so
the author maps out where and when failures could occur. This mapping was valuable insight
as the author conducts FMECA on several data buses and components common to CubeSat
hardware, both custom and commercial. Finally, the author proposes a good solution as they
map out the severity, likelihood, and criticality of a standard satellite component failing on
the spacecraft.

Each validation and verification campaign is unique because the steps to a final model vary
drastically between different missions. Some follow the simple approach of functional testing,
integration testing, and environmental testing, while others perform environmental testing
of each component before integration testing. There are notable similarities between these
approaches. One is that in each campaign, the developers build multiple spacecraft models, and more elaborate testing methods to execute at the hardware and software levels. The statistical [12] and cost analyses [16] justify this approach. For all of these campaigns, a FlatSat platform of some form was used and proved to be instrumental in improving the reliability of the spacecraft.

2.3 CubeSat Flight Software Frameworks

When designing the Virginia Tech FlatSat Platform it was essential to consider flight software frameworks used on CubeSats. This way, the hardware could meet the appropriate processing and memory requirements. The following are some frameworks commonly used to troubleshoot flight hardware.

COSMOS is a set of applications to test whole integrated embedded systems [6]. While initially designed to test embedded systems on spacecraft, developers have applied it to more general embedded development. It even has a version developed to run on a Raspberry Pi which is currently used as the command and control software for the ground station at Virginia Tech.

NOS3 or the NASA Operational Simulator for small satellites is the only software test bed for developing flight software [3]. It was initially developed for the West Virginia STF-1 Mission to demonstrate how software validation and verification can occur with or without hardware in the loop. It uses COSMOS, core flight system (cFS, a generic flight software architecture developed by NASA), and vagrant (a software package for building and maintaining virtual environments) to offer various development and visualization capabilities. Unfortunately, while it has flight heritage [31] and is compatible with multiple platforms, it is too large and complex to comprehend and use intuitively within the short time frame over which CubeSats
are developed.

Each framework addresses a different aspect of embedded software testing, from generating inputs for custom modules to providing an intuitive structure for software development. For these frameworks, both have worked on Raspberry Pi processors in the past. This is another reason for selecting the Raspberry Pi as the experiment manager for the Virginia Tech FlatSat platform.

The main topics covered in this literature review started with a review of prior FlatSat designs with a comparison to the proposed VT FlatSat. Then, CubeSat missions’ Validation and verification methods were covered to build a working knowledge of how FlatSat platforms can be utilized to improve CubeSat reliability. Finally, A review of CubeSat flight software frameworks was conducted to determine the processing requirements of the VT FlatSat. The following chapter will cover the design of the FlatSat in greater detail.
Chapter 3

System Design

3.1 Hardware Design

3.1.1 Overview

As shown in Figure 3.1, the design as a whole involves four PC/104 slots (indicated by DUT #1, DUT #2, etc.) to physically connect the boards to the platform. External power circuitry is present to provide power to the boards when an EPS is not plugged into the platform. A microcontroller unit (MCU) is present to communicate as though it is the satellite’s on-board controller (OBC). The Raspberry Pi acts as the experiment manager.

Figure 3.1: System Architecture of Virginia Tech FlatSat Platform

As shown in Figure 3.1, the design as a whole involves four PC/104 slots (indicated by DUT #1, DUT #2, etc.) to physically connect the boards to the platform. External power circuitry is present to provide power to the boards when an EPS is not plugged into the platform. A microcontroller unit (MCU) is present to communicate as though it is the satellite’s on-board controller (OBC). The Raspberry Pi acts as the experiment manager.
where it monitors the power consumption of each power channel of each DUT and controls the MCU for testing communication. The STM32f4 discovery board was selected to provide USB peripheral expansion, enabling the Raspberry Pi to convert USB communication into CAN, I2C, UART, USART, and SPI. ST Microelectronics™ offers an integrated development environment that auto-configures the firmware with the correct GPIO. It is also the IDE that the software engineering team on UtProSat-1 uses to program all of the hardware.

For each PC/104 slot there are eight input power channels. Each power channel has current sensing implemented to record where and when power anomalies occur and the load characteristics for each power channel under different operating modes. This analog output data feeds into an 8-channel ADC that converts that data to be sent over SPI to the Raspberry Pi. A JTAG Hat is a circuit board that connects to the GPIO on the Raspberry Pi. Mechanically, it sits on top of the raspberry pi hence why it is called a hat. The JTAG Hat on the Raspberry Pi provides robust JTAG communication to enable flash programming a DUT.

One of the primary design considerations for the PCB design of this board is its physical size, as larger boards cost more. Ease of assembly was another factor as it should be hand solder-able to reduce cost, which meant avoiding ball grid array packages or component packages where all of the electrical contacts cannot be heated using a soldering iron. If the board had to get populated out-of-house, that would drastically increase the cost.

The PCB design has the following requirements to minimize cross-talk between data lines and mitigate signal and power integrity issues.

1. The layer stack up is as follows: Top and bottom layers serve as the primary signal layers. The second inner layer is the solid ground plane, and the third is the power plane.
2. Power polygons on the third layer must have a 3.55mm width to handle the power connections’ max current output.

3. Top and bottom layers must have a ground fill in the empty space between lines.

4. Ground vias must be distributed throughout the board to couple all ground planes properly.

5. The width of the signal lines shall be 0.2030 mm at a minimum.

6. The board shall not have any right angle corners on any traces.

No constraints were imposed on the distance between signal lines, or the length over which such lines are parallel. To mitigate cross-talk on signal lines, ground vias are placed in between signal lines to provide shielding. The signal frequency is kept below one megahertz to mitigate transmission line effects for long traces.

### 3.1.2 Experiment Management Computer Selection

The experiment management computer is the computer that hosts any software for testing and programming the subsystems plugged into the platform. For example, if the user needs to test a communication channel on a particular subsystem, they would write a simple script on the computer to do so. This works for any communication protocol on the PC/104 interface by configuring an MCU to be the computer’s target to translate to and from a USB port. Following Figure 3.1, the software on the Raspberry Pi would talk to the MCU via USB. Then, the MCU translates that communication to the dedicated data bus on the platform.

The experiment manager is not necessarily a do-everything computer for the user. There
will always be a unique test case that the current test programs on the computer cannot accommodate. It is up to the user to write a new test program to cover that use case.

The computer must have a Linux operating system to perform these tasks, as most CubeSat flight software frameworks are built in a Linux environment. The computer must also have a wide range of connectivity, meaning multiple USB ports and GPIO to connect peripheral devices. Finally, the computer must be portable so that the platform overall does not take up space on the lab bench when testing. The Raspberry Pi is selected to be the experiment manager as it is powerful enough to run most flight software packages [6], [3] and offers external GPIO to run the current sensing and JTAG programming. See table ?? for performance specifications of the Raspberry Pi 3 model B.

### Experiment Management Computer Specifications

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Raspberry Pi 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Raspbian OS</td>
</tr>
<tr>
<td>CPU</td>
<td>BCM2837@1.2 GHz</td>
</tr>
<tr>
<td>RAM (GB)</td>
<td>1</td>
</tr>
<tr>
<td>Max Power Draw (A)</td>
<td>2.5</td>
</tr>
<tr>
<td>Connectivity</td>
<td>4 USB Ports</td>
</tr>
<tr>
<td></td>
<td>1 HDMI Port</td>
</tr>
</tbody>
</table>

Table 3.1: Specifications for the Experiment Management Computer Selection [35]

### 3.1.3 CAN transceiver selection

Controller Area Network (CAN) is a communication protocol used by the standard PC/104 interface. The automotive industry originally developed the CAN bus to communicate between embedded computers inside cars, and it was later adapted for applications on spacecraft. The differential signal that the bus uses is more resilient to external noise sources than other communication protocols. CAN transceivers translate the differential signal from the CAN bus into packets that can be transmitted and received by the CAN controller on the
The 120-ohm resistors serve as the termination resistors for the ends of the CAN bus. They reduce signal rebound on the transmission lines of the bus, thus making it less likely to corrupt the transmission of data. For these resistors to work correctly, they are placed on the physical ends of the bus. So the communication with the MCU acting as a peripheral expansion for the experiment management computer will only sometimes be on the end of the CAN bus. Other devices under test may already have their terminating resistors for communication on CAN. The 100pF capacitors (C22, C23, C24, and C25) serve as decoupling on the line to filter out any high-frequency noise. R41 and R42 serve as 120-ohm resistors on the end of the bus close to PC/104 slot “PC1” [1].

Figure 3.2: CAN transceiver Operation
3.1.4 Current Sense Design

Background

Monitoring the power draw on a given device under test is essential to record the dynamic load overall for a spacecraft system. If a user can build a dynamic power profile for all subsystems on the spacecraft, a power-balancing software design can be validated with hardware in the loop. Each power channel on each DUT is monitored so that the user can detect system faults within each DUT.

The two main current sensing techniques considered were High-side current sensing, where the shunt resistor is placed between the supply and the load, and Low-side current sensing, where the shunt resistor is placed between the load and ground. The high-side current sensing was selected for this application because it can detect system shorts. This capability is essential if custom-designed hardware is plugged into the platform and, for some unseen reason, is shorted.

Figure 3.3 shows a simplified circuit diagram of the current sensing deployed per power channel. A small shunt resistor, $R_{shunt}$, is connected in series between the main bus power supply and the measured load. The differential input to the op-amp is the voltage drop across $R_{shunt}$. The op-amp amplifies the differential input, and the ADC reads the output voltage. Finally, the ADC sends the data to the Raspberry Pi over the SPI. The current draw is then calculated in the current sense program.

Figure 3.4 shows a circuit topology allowing concurrent current sensing of eight power channels, covering all the power channels on a PC/104 connection. Each power channel has its own shunt resistor with an instrumentation op-amp connected to an input channel on a multi-channel ADC. An input channel is connected to the internal SAR ADC via an eight-
to-one multiplexor inside the multichannel ADC package. Reading output data and selecting a channel to be read is controlled by the SPI protocol via control circuitry internal to the chip.

![Diagram of Current Sense Operation](image1)

Figure 3.3: Current Sense Operation

![Diagram of Current sensing architecture for each slot](image2)

Figure 3.4: Current sensing architecture for each slot. Blue and Green differentiate different chip packages.
Component Selection

High-side current sensing requires that the instrumentation op-amp have a high common-mode voltage. The op-amp must be powered via 3.3V to match the dynamic voltage range of the ADC. The op-amp should have a large bandwidth to detect inrush current events, which requires a high sampling rate. An instrumentation op-amp is necessary as the voltage drop across the shunt resistor is small (in the millivolt range), which necessitates a high input impedance so as not to draw too much current from the shunt resistor.

The INA4180 chip was selected as it was designed for current sensing applications. The chip houses four op-amps, each set to a gain of 200 V/V via the internal gain setting resistors. This saves space on the board, allowing for more flexibility with the routing and layout of the circuit. The component specifications are shown in table 3.2.

<table>
<thead>
<tr>
<th>Current Sense Amplifier Specifications</th>
<th>INA4180</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical Characteristics</td>
<td>INA4180</td>
</tr>
<tr>
<td>RMS Output Noise</td>
<td>0.0003%</td>
</tr>
<tr>
<td>Current Limit (A)</td>
<td>8</td>
</tr>
<tr>
<td>Minimum Load Current (mA)</td>
<td>10</td>
</tr>
<tr>
<td>Ripple Rejection Ratio (dB)</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 3.2: INA4180 Electrical characteristics [5]

For gathering the output voltages of 32 instrumentation op-amps, a multi-channel ADC was selected to reduce space taken on the PCB. Its system voltage must also match the 3.3V system voltage of the GPIO on the Raspberry Pi. Preference is given to lower noise ADCs for better output signal quality. Preference is also given to ADCs communicating on SPI as it is a faster data bus than UART or I2C. A higher sampling rate is not required, as this system is intended to measure current draws at a steady state.

The required resolution to build a dynamic power profile for each DUT is 1mA. The max current to be measured is five amps for a given power channel. This number was selected
to ensure a wide enough dynamic range to monitor the current draw on most hardware and matches the upper current limit on the LDOs selected for the power circuitry.

\[
\frac{5A - 0A}{2^n} \leq 1mA
\]  \hspace{1cm} (3.1)

Following equation 3.1, where n is the number of required bits, the number of required bits must be at least 13 bits. Since no 13-bit ADCs could be found, a 16-bit ADC was selected. For these reasons, the ADS8344EB™ from Texas Instruments™ is selected. Given this ADC’s higher sampling rate of 100KHz, some more advanced features, such as inrush current detection, can be implemented [4].

<table>
<thead>
<tr>
<th>ADC Specifications</th>
<th>ADS8344</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise ($\mu V_{RMS}$)</td>
<td>20</td>
</tr>
<tr>
<td>Offset Error (mV)</td>
<td>$\pm 0.5$</td>
</tr>
<tr>
<td>Gain Error</td>
<td>$\pm 0.024%$</td>
</tr>
<tr>
<td>Conversion Time (CLK cycles)</td>
<td>16</td>
</tr>
<tr>
<td>Acquisition Time (CLK cycles)</td>
<td>4.5</td>
</tr>
<tr>
<td>Throughput Rate (kHz)</td>
<td>100</td>
</tr>
<tr>
<td>External Clock Frequency (MHz)</td>
<td>2.4</td>
</tr>
<tr>
<td>$+V_{cc}$ Supply Voltage (V)</td>
<td>2.7 - 3.6</td>
</tr>
<tr>
<td>Quiescent Current (mA)</td>
<td>1.85</td>
</tr>
</tbody>
</table>

Table 3.3: ADS8344 Electrical characteristics [4]

**Analysis**

The targeted, dynamic current range to measure on a power channel is zero to five amps. This was chosen to match the upper current limit of the LDOs in the power supply circuitry. A small shunt resistor is selected following equation 3.2 from the datasheet [5] to measure a
high current without the amplifier saturating.

\[ I_{\text{max}} \times R_{\text{shunt}} \times Gain < V_{sp} \]  \hspace{1cm} (3.2)

Where \( I_{\text{max}} \) is the maximum current to sense, \( R_{\text{shunt}} \) is the resistance of the current sense resistor, and \( V_{sp} \) is the positive rail of the amplifier. Let \( V_{sp} = 3.3 \) as the positive rail is 3.3V, \( I_{\text{max}} = 5 \text{Amps} \), and the Gain equal 200 V/V. The maximum value \( R_{\text{shunt}} \) can be is then 3.30 milliohms. Since a 3.30 milliohm resistor was unavailable, a 3 milliohm resistor was selected instead. This means that the amplifier’s output voltage when 5 Amps flows through the current sense resistor is 3V.

The lowest current the amplifier can detect with this current sense resistor is worth noting. The minimum current that can be sensed is 0.833mA following equation 3.3

\[ I_{\text{min}} \times R_{\text{shunt}} \times Gain > V_{sn} \]  \hspace{1cm} (3.3)

<table>
<thead>
<tr>
<th>Sources of Error</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC Noise (( \mu V_{RMS} ))</td>
<td>20</td>
</tr>
<tr>
<td>ADC Quantization Error (LSB)</td>
<td>2</td>
</tr>
<tr>
<td>ADC Offset Error (mV)</td>
<td>±0.5</td>
</tr>
<tr>
<td>ADC Gain Error</td>
<td>±0.024%</td>
</tr>
<tr>
<td>Amplifier Offset Error (mV)</td>
<td>0.5</td>
</tr>
<tr>
<td>Amplifier Gain Error</td>
<td>1%</td>
</tr>
</tbody>
</table>

Table 3.4: Sources of error in the current sensing circuit design \[4, 5\]

To determine the accuracy of a given current sensing channel, an error budget is conducted
to determine the effect of sources of error on the overall measurement. Table ?? lists the sources of error and values from the datasheets of the selected components.

The total error is calculated following equation 3.4 where Q is the ADC quantization error, R is the ADC noise in $V_{RMS}$, S is the ADC offset error in volts, T is the ADC gain error, U is the Amplifier offset error in volts, and V is the Amplifier gain error [32].

$$\text{Total Error} = \sqrt{(Q + R)^2 + (S)^2 + (T)^2 + (U)^2 + (V)^2}$$  \hspace{1cm} (3.4)

In Figure 3.5, the percent error as a function of the current being measured is shown. The amplifier gain error is the source of error that mainly contributes to the total error. Even though the percentage error in the graph is seemingly high for the current measuring range, the gain and offset errors introduced by the ADC and the instrumentation op-amp are
only static. This means that the errors do not change with time or frequency and can be factored into the measurement post-processing of the data. The graph in Figure 3.6 shows the measurement’s accuracy when these other error sources are filtered out. These error sources are filtered out by calibrating the software with a known DC load. More on this procedure will be covered in chapter four.

Figure 3.6: Percent Current Sense Error versus current to be measured. Every source of error except for quantization canceled out.

Even though the gain and offset errors of the ADC and Op-amp are filtered out, the quantization error remains as it is a dynamic error source. Not much can be done to cancel this out, as the quantization error is inherent to the ADC. This impacts currents under 20mA and cannot be accurately sensed, following Figure 3.6.
3.1.5 Power Connections design

Background

Most CubeSat subsystems have at least two power input channels for five and 3.3V. Those channels power any processors and logic devices on the board and come from the PC/104 standard. Therefore, embedded power electronics on the board to output 3.3V and 5V reduces the number of bench power supplies required to turn on the device under test.

An adjustable low-dropout voltage regulator was selected for the following reasons. First, since the board ultimately gets its power from the wall rather than a battery pack, high efficiency is not a priority. Since high efficiency is unnecessary, it eliminates the need for DC-to-DC converters, which are more complex than other design alternatives. AC/DC conversion is unnecessary as a bench power supply will supply a DC voltage to the board, thus eliminating the need for AD/DC rectification. In addition, using a bench power supply in this design reduces the complexity of the circuit design as it implements essential safety features.

The circuit for both these power channels is shown in Figure 3.7, where C1, C2, and C3 act as decoupling capacitors to reduce noise on the input, feedback, and output lines, respectively. R1 and R2 set the output voltage of the adjustable LDO. D1, D2, and D3 act as protection diodes for the LDO. D1 protects against reverse polarity if the user plugs the power cables from the power supply into the FlatSat backward. D2 protects the LDO from in-rush currents if the power supply shorts. D3 protects the LDO from inrush current from the capacitor C2 when the LDO turns off.

“Bat” and “Unreg” power channels come from interfaces on standard electric power supplies. Bat on an electric power supply denotes a direct connection to the battery bus voltage. Un-
reg denotes an unregulated power channel for higher current draws. The voltages can vary drastically depending on several complications on the spacecraft. For example, depleting the batteries of charge lowers the voltage level. A simple circuit with just the banana jack connection and a diode to protect against reverse polarity from the power supply accommodates this, as shown in Figure 3.8. Any unique DC voltage will come from the bench power supply. Pin headers on these voltage rails are added to break out the voltage to multiple platform parts [26].

Another LDO circuit, shown in Figure 3.9, is added to the power connection circuitry to
power the ADCs and Op amps in the current sensing circuitry separate from the main bus. This way, power is not being drawn from the main bus, and if there were a system short in a DUT, the op-amps and ADCs would not get shorted out either. Diode D1 serves as protection from reverse polarity. D2 protects the LDO from an in-rush current from the output decoupling capacitor, C2. C1 and C2 filter the input and output voltages of the LDO for high-frequency noise.

Figure 3.9: Simplified Circuit diagram of Power Supply for Current Sensing circuitry.

**Component Selection**

The LDOs need to have a high current limit so that most DUTs on the board can be powered simultaneously. Finding fixed LDOs that have a high current limit (greater than 1A) is difficult. Making them adjustable allows for more options. Preference is given to LDOs with a high ripple rejection ratio, offering more stability. For these reasons, the LM338T Adjustable Regulators™ from Texas Instruments are selected for the power circuitry. See Table 3.5 for its component specifications.

The secondary LDO did not have a high current limit requirement, so a fixed LDO was selected to reduce the number of components used to build the circuit and reduce cost.
Adjustable LDO Component Specifications

<table>
<thead>
<tr>
<th>Electrical Characteristics</th>
<th>LM338T</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS Output Noise</td>
<td>0.0003%</td>
</tr>
<tr>
<td>Current Limit (A)</td>
<td>8</td>
</tr>
<tr>
<td>Minimum Load Current (mA)</td>
<td>10</td>
</tr>
<tr>
<td>Ripple Rejection Ratio (dB)</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 3.5: Electrical characteristics of the LM338T

Analysis

The voltage output of the adjustable voltage regulator follows this equation:

\[ V_{out} = 1.25 \times \left( 1 + \frac{R_2}{R_1} \right) \]  \hspace{1cm} (1)

Where \( I_{Adj} \) is the current through the adjust pin, \( R_1 \) and \( R_2 \) are the feedback resistors, and 1.25 is the required reference voltage.

Following 1, if \( V_{out} = 3.3 \text{V} \) and \( R_1 \) is chosen to be 240Ω solve for \( R_2 \)

\[ R_2 = \frac{V_{out} - 1.25}{\frac{1.25}{R_1} + I_{Adj}} \approx 390\Omega \]  \hspace{1cm} (2)

Resistance values were chosen to be 1% tolerance so that the output voltage has low tolerance [22]. The same design procedure is followed for the 5V adjustable voltage regulator. The feedback resistor values are 120Ω and 360Ω for R2 and R1, respectively.

This section covered the technical design of the circuitry in the VT FlatSat. Beginning with an overview of the system as a whole and then moving into more detail with the design aspects of the power connections, current sensing, and component selection of supporting hardware, such as the experiment computer selection. With the physical hardware design
challenges addressed, software systems must be implemented to complete the VT FlatSat. The next section will cover the software systems used and the design challenges encountered.

3.2 Software Design

3.2.1 Overview

The software systems covered in this section address several key problems that come up in the development of CubeSats:

1. How to program multiple processors across the individual subsystems.
2. The ability to connect remotely to the platform.
3. Tracking the power consumption of each component on each of their power input channels.
4. Validating communication channels for command and data handling.

The Virginia Tech FlatSat Platform also has the capacity to run other test programs to fit the user’s needs. The following sections describe the proposed basic software functionality of the Virginia Tech FlatSat platform.

3.2.2 Current Sensing Software

The current sensing software thus far implemented consists of three separate Python programs. One program reads incoming samples from any of the ADCs on the board and stores that data in a .csv file. The second program reads data from the .csv file, filters it, and
plots it for the user to see. The third program detects inrush current events, plots them in real-time, and also stores the data in a .csv file, following Figure 3.10.

Figure 3.10: Flowchart of the python program used for real-time plotting of inrush current events. The parallel structure indicates the use of multithreading. For complete Python code, see appendix C.

The current logging program’s primary purpose is to sample data coming from each ADC on the Virginia Tech FlatSat platform and log that data in a .csv file for post-processing. Following the left flow chart in Figure 3.11, the program outputs a status message to the terminal indicating that the program is collecting data. For each iteration, it records the
time of the sample to be acquired. Then, it reads each channel of the ADC and stores the corresponding channel value in an eight-element array for temporary storage. Then, each recorded sample is written to a .csv file already initialized by the program. The program will then wait for the sample period time before entering the next iteration.

![Flowcharts of current sensing program](image)

**Figure 3.11:** Flow charts of current sensing program. (Left) Program for reading ADC and logging data. (Right) Program for filtering and plotting collected data. For complete Python code, see appendix C.

To read a sample from a specific ADC on the board, the chip select line for that ADC is pulled low to allow for data transfer on the SPI bus. A command byte is then sent to the ADC, indicating what channel to read, to operate the ADC in single-ended mode, and to operate the analog-to-digital conversion off of an external clock rather than the built-in
clock of the ADC. The output sample is delayed for two clock cycles to allow the ADC to finish processing the command byte and begin the analog-to-digital conversion [4]. It takes 24 clock cycles for the ADC to complete the conversion. However, because the bits for the sample are delayed, the sample read is off by one clock cycle, so it takes an extra command byte to get the least significant bit (LSB) of the sample [4]. Altogether, it takes 32 clock cycles to read in one sample from one channel of one ADC channel. This, combined with the fact that the ADC has a maximum clock frequency of 2.4 MHz (see table ??), means that the system has a maximum sampling rate overall.

This is acceptable for recording the current draw at steady-state over longer periods of time. However, due to this limitation, the inrush current detection program can only detect inrush current events on one channel. Since inrush current events happen quickly, the system can only get about four to five samples of this event.

Inrush current events occur when digital subsystems are turned on. Since these subsystems use several decoupling capacitors to reduce noise on input voltages, they begin to behave like capacitors. When there is a fast change in the voltage (also known as hard turn-on), the amount of current increases. This sudden inrush of current can damage sensitive electronics. These events can occur quickly, on the order of milliseconds, as the change in voltage goes to zero.

The plotting program filters the data collected by the current logging program and displays it to the user in a readable graph. A four-point moving average filter filters out higher-frequency noise from the recorded data. This proved to be advantageous over using a digital low-pass filter as it smooths the data without filtering out sharp transitions in the current measurement. A moving average filter works by averaging samples of input data in a fixed range to produce a single averaged sample in the output data. The equation for the moving
average follows equation 3.5 [28].

\[ y[n] = \frac{1}{W} \sum_{k=0}^{W-1} x[n-k] \]  

(3.5)

\(y[n]\) is the output data, \(x[n]\) is the input signal, \(W\) is the window size, and \(n\) is the output and input data index. The user would have to go into the program and change the window size variable to change the window size.

### 3.2.3 Remote OpenOCD Debug Configuration

OpenOCD [21], or open on-chip debugger, is an open-source hardware debugger designed to flash programs to multiple processors. It can turn any JTAG or SWD connection into the equivalent of the flash programmer sold commercially. This way, multiple boards with different microcontrollers can be programmed from the same IDE. It also has compatibility with multiple IDEs such as STM32cubeMX™, Eclipse™, Visual Studio™, and Code Composer Studio™.

A remote SSH connection allows users to connect remotely to the platform, as shown in Figure 3.12. This way, the software developers do not have to put on personal protective equipment whenever they want to program the hardware or perform diagnostic checks on a system stored in a clean room. The remote SSH connection also eliminates the risk of hardware getting damaged in transit, as developers would no longer require physical proximity to program the hardware.

The JTAG hat is used to allow users to remotely flash program the target hardware. The JTAG Hat from Blinkin Labs [9] is a PCB that turns the Raspberry Pi computer into a general-purpose networked debugger via a JTAG ribbon external to the FlatSat PCB. This
hardware allows the Raspberry Pi to replace any other hardware debugger for flash programming any microcontroller unit. Furthermore, if the satellite utilizes different processors from one board to another, there is no need to purchase additional hardware to program all of these processors.

This chapter covers the details of design choices made for both the hardware and software aspects of the VT FlatSat. The hardware features of the FlatSat include a Raspberry Pi for monitoring the power consumption of individual subsystems and a debugger for flash programming multiple MCUs. The next chapter will discuss testing procedures and what parts of the VT FlatSat work.
Chapter 4

Results

This chapter will cover the testing and results of different features of the VT FlatSat Platform. Each test will follow objectives, setup, and procedure, and conclude with results and discussion. This chapter will cover functional testing of the power connection circuitry, current sensing for steady-state and inrush detection, and partial functional testing of the Payload Control module for the Ut ProSat-1 Mission.

4.1 Virginia Tech FlatSat Functional Testing: Power Connections

4.1.1 Objectives

Two tests are performed for functional testing of the power connections on the VT FlatSat. The first is a step input test to 10V to see if the embedded LDOs turn on as expected and if there is no significant overshoot greater than ten millivolts. This test will also examine the steady state output voltage to see if the output noise is acceptable within five millivolts root-mean-squared (RMS). The second test is a reverse polarity test where the cables from the external power supply are intentionally plugged in backward. This is to see if the power connections will stay off when the power is turned on.
4.1.2 Setup

The setup for this test requires an external bench power supply, banana cables, an oscilloscope, and oscilloscope probes. The oscilloscope is set in trigger mode with the oscilloscope probes attached to test point 9 (TP9) for the 3.3V channel or to TP10 for the 5V channel. Following the schematic in figure 4.1, the banana cables are connected to connector 'J9' where red refers to the red terminal of the connector, and black refers to the ground terminal. For the battery bus voltage (Vbat) power channel, the banana cables are connected to the J8 connector and the oscilloscope probe is connected to the cathode of diode D1. For the unregulated power channel, the banana cables are connected to connector J11 and the oscilloscope probe is connected to the cathode of diode D7, following figure 4.2.

![Schematic of test setup for 3.3V and 5 volt power connections. "O-scope" denotes oscilloscope. The dotted line indicates external equipment.](image)

The setup for the reverse polarity test is the same as before, except the banana cables from the external power supply are plugged in reverse to the power connector, following figure 4.3.
4.1.3 Procedures

A bench power supply’s output is connected to each power channel on the board, as shown in figure 4.3. Oscilloscope probes are attached to each power channel’s input and output test points. The oscilloscope is set to the trigger mode on a rising edge in order to capture the turn-on event. The output channel of the power supply is turned on. Once the power channel is fully on, a basic voltage check with a multi-meter is performed across all test points on the platform. Once the standard output is observed across all test points, the power supply is turned off, and the cables are flipped at the platform’s input power channels. Then, the oscilloscope is set to trigger on a falling edge. The power supply output is turned on, and another voltage check with a multimeter is performed across all test points on the platform to see if there is an observed voltage.

4.1.4 Results & Discussion

All tests passed with the power channels. The output voltage during the nominal test of the LDOs was observed to be 3.30V and 5.00V. Figure 4.4 shows the step response of all the power channels to a step input of zero to 10V. Unregulated and battery bus power channels
Figure 4.3: Normal Operations with bench Power supply (left) and Under Reverse Polarity (Right)

show a slight overshoot but level out to approximately 9.4V. This is expected due to the diode’s voltage drop on the upper power channels.

Figure 4.4: Step Response of each power channel (Right) and Turn Off of each power channel (left)

Figure 4.5 shows the 3.3V and 5V power channels turned on to a reverse polarity voltage. As expected, the output voltage, shown in green, stays at 0V with no transients on the line.

On the other hand, figure 4.6 shows a slightly negative voltage transient on both the outputs of the Battery Bus and Unregulated channels. To remove the transient event, a transient
voltage suppression (TVS) diode should be used on the output of these power channels so that no electronics are damaged.

While a test for over-voltage and over-current was not conducted, the power connections behaved as expected. The voltage check showed that power was distributed throughout the whole FlatSat. However, there could be an induced voltage drop if a CubeSat subsystem were plugged into the board, drawing a substantial amount of current.

Figure 4.5: 3.3 Volt Bus Turn On to Reverse Polarity (Right) and 5 Volt Bus Turn on to reverse polarity (Left)

Figure 4.6: Unregulated Bus Turn On to Reverse Polarity (Right) and Battery Bus Turn on to reverse polarity (Left)
4.2 Virginia Tech FlatSat Functional Testing: Current Sensing

4.2.1 Objectives

A voltage check on the power pins for the op-amps and ADCs ensures power is distributed to all of the current sense circuitry. An overall test of the system ensures that all current sensing channels are accurate, linear, and repeatable.

Accuracy is determined by connecting a load that draws a known amount of current. For example, a 3.3 ohm resistor connected to a 3.3V voltage source will have 1A of current flowing through it. Linearity refers to when there is a linear change in the current draw. If so, the computer should be reading the same change as well. Repeatability is when the same measurement can be repeated multiple times, assuming nothing about what is being measured changes.

4.2.2 Setup

The required equipment for this test is an external bench power supply, banana cables, a multimeter, Through-hole resistors, and peripherals to connect to the Raspberry Pi (i.e., USB keyboard, mouse, HDMI monitor, etc.)

For the overall current sense test, two resistors are connected between the power channel output pins and ground to form the load, as shown in figure 4.8. A test program running on the Raspberry Pi records samples from the ADC following the schematic in Figure 4.7.
4.2. Virginia Tech FlatSat Functional Testing: Current Sensing

4.2.3 Procedure

A multimeter checks each power pin on the op-amps and ADCs to see if 3.3V are being supplied. While the program is running, the enable switch for that power channel is turned on and off four times, manually, to ensure that the measurements are repeatable. Once this is done, the program is terminated, and a separate program to plot the collected data is
run. The output graph should show data only on the channels that are drawing current and should show four pulses. Each pulse should match the expected current draw. Once this is done, the collected data is stored in a .csv file, and the resistors are moved to the next PC/104 slot on the Virginia Tech FlatSat platform. The same procedure is repeated until all four PC/104 slots have been tested with the resistors. The same procedure is repeated for three sets of resistances. Once the three rounds have been completed, new resistor values are used for the next set of channels, and the whole process starts over again. Figure 4.9 shows the equivalent schematic for testing each current draw channel of the whole platform.

Figure 4.9: The location of the power channel pins on a PC/104 slot. The pinout is the same for all four slots.
4.2.4 Inrush Current Detection Procedure

A 470 uF capacitor was connected to slot PC4 on the 3.3V channel to test the inrush current detection program, as shown in figure 4.10. The 3.3 V power channel enable switch is switched on to trigger an inrush current event. After turning the switch off, the indicator LED for the 3.3V channel should turn off completely before turning the switch back on to trigger another event.

Figure 4.10: Inrush current detection testing setup with 470 uF capacitor
4.2.5 Results & Discussion

All 32 current sensing channels work as expected following the design. Figure 4.9 shows an example graph for the current measurement data. Where descending subplots show turn-ons to higher resistances, lowering the target current to be measured. For a target current of 10mA, the quantization noise begins to take over the precision of the measurement. This is expected following the graph in Figure 3.6 where the percent error exponentially increases below 50mA. Also, notice that the data returns to the expected value for each turn-on, indicating that the current sensing system measurements are repeatable. The system is accurate to ± 5% because 5% tolerance resistors were used in testing. Current sensing data for the other 30 current sensing channels can be found in Appendix IV.

![Figure 4.11: Example graph showing the current pulses for the 5V bus, turning on to 5.6-ohms, 56-ohms, and 560-ohms](image)

For inrush current detection, however, the current inrush detection program proved unreliable as the enable switch for this power channel was switched on and off eight times, which
is supposed to trigger an inrush current event eight times. This system only detected seven inrush current events. This was repeated several times to see if this performance was repeatable, and it was not. There were rounds of testing where the system got all of the events and others where it detected none. This is because it does not have the required time resolution to make these measurements accurately. Figure 4.12 shows the recorded current measurement with respect to time. Each impulse on the graph shows an inrush current event detected by the inrush current program.

Figure 4.12: Inrush currents versus time. Switched eight times, the system only detected seven inrush current events. This was done on channel 0 for slot PC1.

### 4.3 UtProSat-1 Payload Control Module Testing

The UtProSat-1 CubeSat has the primary mission objective of successfully deploying and retracting a tape spring boom developed by NASA Langely [43]. This requires the control of a stepper motor and two servos in the boom deployment mechanism. There is also a small
proxy MCU in the deployer for collecting data on the vibrations of the boom deployment and retraction. A custom-designed subsystem was needed as outsourcing the design is cost-prohibitive for the mission budget. This custom computer is the Payload Control Module (PCM).

4.3.1 Background UtProSat-1 Payload Control Module

![System block Diagram of the PCM.](image)

The PCM is the computer designed to manage all the payloads on the UtProSat-1 satellite mission. Per the system block diagram in figure 4.13, the PCM has outputs for stepper motors and servos to actuate the deployment mechanism. In addition, it interfaces with a more miniature embedded computer to record data on the vibration of the boom from the...
inertial measurement unit on the tip of the boom. It was also designed to communicate with an optane memory module to perform space qualification testing. Other features of the PCM include a microSD card to store data in-between passes, IMUs to track the orientation of the CubeSat, and a camera to capture an image of the boom fully deployed in space.

### 4.3.2 Objectives

A power-on test ensures that power is reliably delivered to the board. Subsequent voltage checks also ensure that all logic devices on the board receive the correct voltage. Finally, a basic JTAG communication test is needed to ensure that the Raspberry Pi on the Virginia Tech FlatSat platform can program the PCM.

### 4.3.3 Setup

This setup requires an engineering model of the PCM, an external bench power supply, two banana cables, an IDC ribbon cable for JTAG, and peripherals to connect to the Raspberry Pi (i.e. USB keyboard, mouse, HDMI monitor, etc.) to view output messages.

The PCM should be connected to slot PC1 on the VT FlatSat following figure 4.14. Two jumper wires were required to connect the power channels on the Virginia Tech FlatSat platform to the 3.3V and 5V input channels on the Payload control module. The payload control module is powered through different pins on the PC/104 Connector interface. The external bench power supply should be connected to connect J9 on the VT FlatSat via the banana cables. The external power supply’s output should be set to 10V.
4.3.4 Procedure

Once the setup is complete, the output of the external power supply is turned on and the 3.3V and 5V LEDs should turn on. In the desktop environment on the Raspberry Pi, a terminal is opened and the command “Sudo OpenOCD” is run. The output message in the terminal window should match the left figure in Figure 4.15.

4.3.5 Results & Discussion

The PCM behaves as expected. When turned on both the 3.3V and 5V indicator LEDs turned on. For the flash programming test, it was discovered in the program testing of the board that the nRST pin of the JTAG input was never connected to the microcontroller unit. This was a routing mistake in the PCM PCB itself. The nRST pin or the reset pin is
4.3. UtProSat-1 Payload Control Module Testing

Figure 4.15: OpenOCD failing to Halt the PCM (Left) OpenOCD successfully halting and flash programming the PCM (Right)

critical for programming the board with openOCD, as the software cannot reset the device to halt its operation in the debugging session. The output in the following Figure 4.15 shows the OpenOCD session failing to halt the PCM’s microcontroller unit.

Once this was discovered, a white wire fix connecting the reset pin on the connector to the microcontroller was warranted. A thin gauge wire was soldered onto the back of the payload control module, forming the connection as shown in Figure 4.16. Once completed, the board was powered on again to ensure nothing was shorted in soldering the wire onto the PCM. The successful output in figure 4.15 session was obtained with the same setup as before.

In this chapter, the power connections, current sensing, and flash programming of were tested for functionality. The PCM from UtProSat-1 was tested on the VT FlatSat, and errors in its design were discovered as a result. While the VT FlatSat behaves as expected, more tests are needed to fully cover the functionality offered by the VT FlatSat. Future revisions in the design are also needed to expand the capabilities of the VT FlatSat further.
Figure 4.16: PCM white wire fix on back of the board
Chapter 5

Conclusions and Future Work

5.1 Future Work

5.1.1 Hardware Changes

Future revisions of the Virginia Tech FlatSat platform will embed most of the electronics into one PCB to reduce the total area the board takes up on the lab bench. Embedding the electronics will also eliminate any unnecessary I/O that comes with off-the-shelf parts. Switching from the Raspberry Pi 3 to the Raspberry Pi Compute module will aid this effort because the compute module offers a smaller form factor and additional I/O to support new peripherals. Another design alternative for future revisions would be adopting a new interface besides PC/104. The study by Bouwmeester et al. [11] concluded that the PC/104 interface was not ideal, given its large size. In addition, using an interface with a smaller connector would allow users to mount more hardware in the satellite.

Another design alternative is presented in figure 5.1, where the ‘experiment management computer’ becomes any host computer available to the user. A more powerful MCU controls the SPI bus communicating with the ADCs instead of the Raspberry Pi.

An additional feature to add to the platform would be to add more LDOs to emulate the I/V characteristic curves of commercial solar panels as though they were in the space en-
Figure 5.1: Original system architecture (Left) Proposed system architecture (Right)

environment. Adding this circuitry would allow for complete power balancing testing of the engineering model where the EPS design is validated with hardware-in-the-loop simulation.

5.1.2 Software Changes

The platform can still be improved by expanding the capabilities of its software systems. Integration with COSMOS will be the next step of this effort, as it is used to control and operate the radio ground station at Virginia Tech. For example, the platform could store GNU radio flow graphs of how data are processed at the ground station so that communications testing of the transceiver on a given spacecraft can occur.

Another potential avenue is integration with SatCat5. SatCat5 [39] is a mixed-media Ethernet switch developed by the Aerospace Corporation to handle multiple payloads using any communication interface. This way, software defines hardware changes and can swiftly change from one satellite mission to another. This technology already has flight heritage on SlingShot-1 [42] and can be adapted for future CubeSat missions at Virginia Tech by adding an FPGA.

Finally, Direct Memory Access (DMA) should be used to improve the current sensing software. When the Raspberry Pi CPU is used for other high-priority tasks, the result can be
5.1. Future Work

A non-uniform sampling of the collected data and reduced sampling rates [8]. DMA allows uniform sampling by bypassing the CPU in the program’s execution. However, a major drawback of this method is the direct reading and writing to registers on the CPU. If not done properly, this can lead to the operating system crashing as the OS relies on those registers to execute other tasks.

5.1.3 Further Testing

Further testing of the inrush current detection involves using a 10-ohm resistor connected in parallel to the capacitor for the test load following Figure 5.2. This would be more accurate to subsystems plugged into the FlatSat as there is a Thevenin equivalent resistance looking into the power terminals. The resistor would be ohms so that the resistor does not overheat from the power dissipated. The setup and procedure would follow the testing procedure for inrush current detection covered in section 4.2.4. The inrush current event is expected to rise quickly at roughly 10A/ps. The current will then decrease at roughly 6A/us, since the capacitor discharges through the resistor. Eventually, the current would settle to 0A as the circuit would be disconnected from the source.

Figure 5.2: Testing setup for inrush current detection with a resistor.
Further testing of the PCM needs to be conducted before it can be integrated into the flight model for UtProSat-1. Functional testing would include a test of streaming vibrational data and data from other payloads coming from the PCM, a test of multiple power-up cycles showing that the system boots up properly, and testing for power sequencing the 3.3-volt, 5-volt, and the battery bus voltage to see if there is unexpected behavior coming from the PCM.

Inrush current testing of the PCM is also needed as a hard turn-on from the EPS in the CubeSat could cause damage to the board. The PCM never included the supporting circuitry in the design to mitigate this potential issue. From hard turn-on, the current starts at 0A and then rises roughly at 1A/ns. Then, the current should overshoot by 10mA and settle at 20mA, steady-state. The setup for the test follows Figure 5.3, where an external bench power supply is connected to connector J9 on the VT FlatSat. The PCM is plugged into slot PC1 on the VT FlatSat. The enable switches on the output power channels need to be in the off position. Then, the external power supply output channel is set to 10V. The inrush current detection program should be running in the terminal on the Raspberry Pi at this point. The procedure then is to switch the enable switch to the on position and let the inrush detection program measure the inrush current. If this becomes an issue where the magnitude of the inrush current is too high (greater than 5 Amps), a soft-start circuit on the 3.3 and 5-volt input channels on the PCM will be added.

5.2 Summary

Following the VT CubeSats missions and the engineering problems they faced, the development of a FlatSat Platform is proposed to mitigate those problems. A literature review of relevant FlatSat designs was conducted to develop a FlatSat with more features. The design
of the VT FlatSat offers multichannel current sensing to monitor multiple power channels on CubeSat Subsystems to build dynamic power models. This addressed a major problem faced in past VT CubeSat Missions with power balancing. The design also allows connecting subsystems on the FlatSat to the internet for remote monitoring and measurement. Then, functional testing of the FlatSat was conducted. The FlatSat behaved as expected. However, more testing is required to ensure that the FlatSat works completely. The VT FlatSat, as it stands, proved helpful in troubleshooting errors encountered during partial testing of the PCM.

Most of the design requirements for the FlatSat have been met, with some needing further testing to be conclusive. The design accommodates multiple PC/104 slots, the RBF switches work, and the current sensing works. Minor changes to the current sensing can be made to sense smaller currents less than 20 milliamps, such as increasing the resistance of the sense resistor. The inrush current detection, however, requires multiple changes in the software stack to work completely. This will be covered in greater detail in the software changes section.

As for the remote SSH capability, it works when the IP address of the Raspberry Pi is known. When restarting the Raspberry Pi and connecting it to the Eduroam network, the IP address
will change because the Eduroam network will assign a new IP address to the Raspberry Pi each time it reconnects. Remote users must know the Raspberry Pi’s IP address to establish a remote connection to the FlatSat. The Raspberry Pi also has limited processing power compared to larger personal computers.

For the power connections, it has been shown that they can take power from an external bench power supply. The enable switches on the outputs of the power connection circuitry to the FlatSat allow the FlatSat to simulate hard turn-on events as though the hardware is connected in the CubeSat. However, over-voltage and overcurrent precautions were not implemented in the circuit design. Additional circuitry, such as poly resettable fuses and zener diodes, would meet these requirements. Other limitations of the power connections include the inability to power subsystems that require a floating ground. This instrument could make measurements at 100V above the Earth’s ground.

It was never tested for data communication on the common data busses in the PC/104 connection. The supporting PCB routing and hardware are there, but the software at the time of this writing is not functional, as a full test of communication would require the implementation of interrupt service routines in FreeRTOS. The STM32f4 MCU selected is limited for this application. A more powerful MCU, such as an STM32H5 or STM32H7, is needed.
Appendices
Appendix A

Appendices I: Current Sensing test results data
APPENDIX A. APPENDICES I: CURRENT SENSING TEST RESULTS DATA

### Turn on testing for slot PC3

**Turn On to 5.6 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

**Turn On to 56 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

**Turn On to 560 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

### Turn on testing for slot PC4

**Turn On to 5.6 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

**Turn On to 56 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

**Turn On to 560 Ohms**

<table>
<thead>
<tr>
<th>Channel 0</th>
<th>Channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image6" alt="Graph" /></td>
<td></td>
</tr>
</tbody>
</table>

**Time (s)**

**Current (A)**
APPENDIX A. APPENDICES I: CURRENT SENSING TEST RESULTS DATA

**Turn on testing for slot PC3**

**Turn On to 4.7 Ohms**

**Turn On to 47 Ohms**

**Turn On to 470 Ohms**

**Turn on testing for slot PC4**

**Turn On to 4.7 Ohms**

**Turn On to 47 Ohms**

**Turn On to 470 Ohms**
Turn on testing for slot PC1

Turn On to 4.7 Ohms

Turn On to 47 Ohms

Turn On to 470 Ohms

Current (A)

Time (s)

Turn on testing for slot PC2

Turn On to 4.7 Ohms

Turn On to 47 Ohms

Turn On to 470 Ohms

Current (A)

Time (s)
Appendix A. Appendices I: Current Sensing Test Results Data

Turn on testing for slot PC3
Turn On to 4.7 Ohms

Turn On to 47 Ohms

Turn On to 470 Ohms

Turn on testing for slot PC4
Turn On to 4.7 Ohms

Turn On to 47 Ohms

Turn On to 470 Ohms

Current (A)

Time (s)


[39] Alexander Utter, Mark Zakrzewski, Andrew Keene, Samuel Dietrich, Sammy Lin, Eric


