TigerCub: An Accessible, Open-Source PocketQube Platform

Candace Do, '24

Submitted to the
Department of Mechanical and Aerospace Engineering
Princeton University
in partial fulfillment of the requirements of
Undergraduate Independent Work.

Final Report

April 24, 2024

Michael Galvin
Ryne Beeson
MAE 442
200 pages
File Copy

This report represents my own work in accordance with University regulations.

Abstract

The past few decades have seen an explosion of nanosatellite development, with the proposal of the CubeSat in 1999 followed by the PocketQube in 2009. CubeSats have seen wide use across educational, industry, and scientific applications. PocketQubes, a smaller 5-cm cubical form factor, have seen growing popularity in the past few years but still lack significant development and regulation compared to CubeSats. PocketQubes have a significant cost and time-of-development advantage over larger satellites due to their smaller size, and they have been launched by both commercial companies and educational institutions such as Alba Orbital, G.A.U.S.S., and T.U. Delft. One component of PocketQubes poses a significant challenge to first-time developers: the communications module. Many student-developed small satellite missions fail due to the loss of downlinking ability after launch; most radio modules are also difficult to test from the ground. A promising but yet-unproven option for PocketQube developers is the Iridium satellite network, which has been successfully used in CubeSat missions to provide communication from satellites to Iridium's ground stations. Leveraging the Iridium satellite network will simplify PocketQube development and provide an opportunity for developers to perform conservative downlinking tests and prove link budgets from ground on Earth prior to launch. We propose TigerCub, a modular, mission-agnostic PocketQube platform designed for student development that features SparkFun's RockBLOCK 9603, an Iridium-based communications module. The TigerCub platform aims to be an open-source, low-cost, accessible option for students. Besides the Iridium communications module, the TigerCub features an Arduino-based flight computer, an electrical power system derived from BHDynamic's open-source DynOSSAT EDU, a sheet-metal structure, and allocated, interface-controlled space and power for a payload. We conduct testing and analyses to confirm the downlinking abilities of the Iridium module and power positivity for the spacecraft. Finally, we create a user's guide for developers who wish to use TigerCub. As a new PocketQube platform, TigerCub will contribute to the growing number of open-source platforms available and help establish best practices for future PocketQube development.

Acknowledgements

This endeavour would not have been possible without my thesis adviser, Michael Galvin. Mike, thank you for being an incredible mentor and for your invaluable advice and feedback over the past year and a half. Working in the TigerSats Lab has put me on a path I'm excited to pursue, and I couldn't have done it without your support. I'm also grateful to the many TigerSats alums who paved the way for my work.

Funding for this project was generously provided by the Department of Mechanical and Aerospace Engineering, the School of Engineering and Applied Sciences, and the Council of Science and Technology.

To Mom, Dad, and Ethan–thank you for your unwavering support. None of this would have been possible without you. To the members of Brown Co-op and Club Badminton–thank you for being my home on campus. To Nathan–thank you for being my biggest cheerleader and for always believing in me. To Ari, Manali, and Dave–thank you for the MAE camaraderie, and I can't wait to see what you all achieve. And to Angel, Riri, Zoe, Ethan–thank you for the love, the laughs, and the late nights. I'll treasure them forever.

Per aspera ad astra!

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

α	absorptance
ϵ	emissivity
ADS	Attitude Determination System
ASCII	American Standard Code for Information Interchange
AT	Attention
BOL	Beginning-Of-Life
CAD	Computer-Aided Design
CNC	Computer Numerical Control
COTS	Commercial-Off-the-Shelf
CVCM	Collected Volatile Condensable Material
DoF	Degrees of Freedom
EMISM	Electromagnetic Interference Safety Margin
EOL	End-Of-Life
EPS	Electrical Power System
FTDI	Future Technology Devices International
g	Gravitational acceleration on Earth, 9.81 m/s^2
GND	Ground
GPIO	General-Purpose Input/Output
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IMU	Inertial Measurement Unit

ISS	International Space Station
LED	Light-Emitting Diode
LEO	Low Earth Orbit
MCU	Microcontroller Unit
MO	Mobile Originated
MPPT	Maximum Power Point Tracking
MT	Mobile Terminated
OBC	On-Board Computer
PCB	Printed Circuit Board
PCBA	Printed Circuit Board Assembly
PQ	PocketQube
RXD	Received Data
SBD	Short Burst Data
SCL	Serial Clock
SD	Secure Digital
SDA	Serial Data
SEE	Single Event Effect
SEL	Single Event Latchup
SEU	Single Event Upset
SHE	Single Hard Error
SMA	SubMiniature version A
SMAD	*Space Mission Analysis and Design*
SPI	Serial Peripheral Interface
TID	Total Ionizing Dose
TML	Total Mass Loss
TXD	Transmitted Data
USB	Universal Serial Bus
USD	United States Dollar

Chapter 1

Introduction

1.1 Background

The past few decades have seen an explosion of small satellite development. Small satellites, generally defined as satellites with a mass under 180 kg, have overtaken larger spacecraft as the primary vehicles for a wide range of applications, from scientific research by governmental and educational institutions to technology demonstrations by commercial companies [1]. As launch costs become cheaper as a result of the competitive launch vehicle industry, small satellites have proven to be a cost-efficient method for bringing payloads to space.

One common small satellite form factor is the CubeSat. Originally proposed by Professors Jordi Puig-Suari and Bob Twiggs in 1999, the CubeSat was meant to increase accessibility to space by reducing the cost and time associated with developing satellites. In 2009, Twiggs proposed an even smaller form factor, the PocketQube, for even faster development and lower costs [2]. A PocketQube is a picosatellite (a satellite with a mass between 0.01 and 1 kg) composed of 5-cm cubical units; a 1-unit PocketQube is designated as 1p, similar to the 1U unit classification for CubeSats [1]. In 2018, the PocketQube Standard, a document describing several requirements for standard PocketQubes (similar to the CubeSat Design Specification) was published [3]. As of 2023, over 70 PocketQubes have been launched to orbit [4]. Though PocketQubes still lag behind CubeSats in terms of number of satellites launched per year, their low cost and time of development, coupled with the increasing availability of miniature commercial-off-the-shelf electronic components, makes them a promising option for educational institutions hoping to send something to space on a budget.

1.2 Related Work

The TigerSats Lab, labs at other universities, and commercial institutions have made strides towards satellite miniaturization, including the design of PocketQubes.

The TigerSats Lab has been developing student-accessible small satellite subsystems for the past five years. This project will build upon TigerSats' previous experience in small satellites, such as Kyle Ikuma '23's EduSat, a CubeSat platform for up to five educational payloads (such as the MaxIQ xChips). In particular, Kyle's flight computer design, which included both commercial-off-the-shelf (COTS) components and custom designs, inspired some of the flight computer design in this project.

In addition to these TigerSats projects, other universities have also built PocketQube platforms such as PyCubed-Mini [5], based off the popular PyCubed CubeSat framework [6]. PyCubed-Mini is an open-source 1p PocketQube research platform [5]. Though PyCubed-Mini is designed to be affordable and accessible, it is still a research platform, and some of its more complex components might deter a first-time builders with little satellite experience. For example, its radio module requires a custom-built antenna and long range (LoRa) communication knowledge [5].

Several companies manufacture and sell PocketQube platforms and dispensers. For instance, Alba Orbital, based on Scotland, offers their Unicorn-2 PocketQube platform and the Albapod PocketQube deployer [7]. Alba Orbital is also launching their own PocketQube constellation, named Nightlights, to image Earth at night [8]. However, buying these COTS PocketQube platforms is prohibitively expensive for small university labs; pricing for the Unicorn-2 platform (not including launch) starts at 149,000 euros (about 162,000 USD) [9].

1.3 Motivation

The TigerSats Lab at Princeton is dedicated to developing nanosatellites, emphasizing methods that promote accessibility (in terms of cost, time, and experience) to such projects. The TigerSats Lab has previously focused on building CubeSats, ThinSats, and qualification test facilities. However, due to student turnover, designing and building entire satellite systems has thus far been difficult for the lab to achieve. Since PocketQubes are smaller systems, and thus require fewer resources to design and build, the TigerSats Lab hopes to gain experience in PocketQubes to provide future students with cradle-to-grave satellite development opportunities. In this project, we aim to create a mission-agnostic PocketQube platform for students and other first-

time PocketQube developers to further develop for their own picosatellite projects.

In addition to platform development, we aim to create a PocketQube that is easy to build and use for students who may have little experience with satellites. One significant challenge for student-developed nanosatellites has historically been satellite communication to ground; about 40% of hobbyist-class CubeSat missions (which most university-developed missions fall under) were dead-on-arrival to orbit, with one of the main failures being to loss of communication [10]. To this end, we hope to demonstrate that the RockBLOCK 9603 satellite communication module, built to interface with the Iridium satellite network, is a viable radio module for the PocketQube. Using this COTS component (rather than a bespoke solution) will simplify future PocketQube development.

1.4 Objectives

TigerCub is a modular, mission-agnostic PocketQube platform for future student projects. This project can be split into four primary objectives:

1. Design and prototype a PocketQube according to the PocketQube Standard,
2. Demonstrate power positivity under orbital solar charging conditions,
3. Demonstrate downlinking abilities through the Iridium radio network, and
4. Create a PocketQube Development Guide for future student reference.

To keep costs low for future students, we aim for the total cost of components, manufacturing, and assembly to be less than 2000 USD.

Since TigerCub is mission-agnostic, it is not designed for any particular payload. Instead, we focused on creating a platform that can support a variety of payloads by providing power, volume, and mass allowances. Further discussion of payload interfaces and sample payloads is provided in Chapter 5.

1.5 Requirements

The following tables outline the project requirements. Rationales are derived from the PocketQube Standard, PQ9 Standard, project objectives, or as footnoted.

#	Requirement	Rationale
Functional Requirements		
F-01	At least two kill switches shall be used to keep the satellite offline while in the deployer.	PQ-Mech-12
F-02	The kill switches shall electrically disconnect power from powered functions.	[1]
F-03	The PocketQube should conform to the PocketQube Standard. See Table 1.2.	Obj. 1
F-04	The PocketQube should conform to the PQ9 Standard. See Table 1.3 and Section 3.2 for details.	[2]
Performance Requirements		
P-01	The PocketQube shall demonstrate power positivity under orbital solar charging conditions.	Obj. 2
P-02	The PocketQube shall provide payload interfaces and budgets.	[3]
P-03	The PocketQube shall be able to communicate with a ground station.	Obj. 3
P-04	The PocketQube shall provide on-board computation, data storage, and attitude determination.	[4]
Constraints		
C-01	The external dimensions shall be at most $50 \times 50 \times 50$ mm.	PQ-Mech-01
C-02	The sliding backplate dimensions shall be $58 \times 64 \times 1.6$ mm.	PQ-Mech-01
C-03	The maximum component envelope shall be 7 mm.	PQ-Mech-08
C-04	The maximum mass shall be 250 grams.	PQ-Mass-01
C-05	The center of mass must be within 10 mm of the geometric centroid.	PQ-Mass-04
C-06	The PocketQube should cost less than 2000 USD.	[5]
Environmental Requirements		
E-01	The total mass loss shall be at most 1.0%.	PQ-Gen-04
E-02	The collected volatile condensable materials from outgassing shall be at most 0.1%.	PQ-Gen-04
E-03	The PocketQube shall withstand the launch vehicle load and vibration environment.	[6]
E-04	The PocketQube shall withstand the radiation environment in low-earth orbit.	[7]
E-05	The PocketQube shall withstand the charge environment in low-earth orbit.	[8]
E-06	The PocketQube shall operate in the thermal environment of low-earth orbit.	[9]

Table 1.1: TigerCub requirements.

[1] Kill switches must completely disconnect spacecraft power while the satellite is in the deployer.

[2] PQ9 is a common PocketQube electrical standard.

[3] The PocketQube should provide mass, volume, data, and power budgets and mechanical and electrical interfaces for potential payloads.

[4] These are crucial subsystems for any satellite.

[5] The platform should be affordable for student teams and university labs.

[6] The satellite must survive the launch environment to conduct the mission in orbit.

[7] The satellite must protect internal components from the effects of space radiation.

[8] The satellite must protect components from the effects of charging.

[9] The PocketQube's thermal control should ensure that components are within operating temperature ranges.

#	Requirement	Source
F-03-01	Components shall not be deliberately detached during the mission lifetime.	PQ-Gen-01
F-03-02	Pyrotechnics shall not be used.	PQ-Gen-02
F-03-03	Toxic, hazardous, and flammable materials shall not be used	PQ-Gen-03
F-03-04	Metallic materials in contact with the deployer must be hard-anodized.	PQ-Mat-03

Table 1.2: Additional requirements drawn from the PocketQube Standard [3].

#	Requirement	Source
F-04-01	The board dimensions shall be $42 \times 42 \times 1.6$ mm.	PQ9 §3.1
F-04-02	Four mounting holes shall be provided with size and positions as shown in Figure 3.2.	PQ9 §3.1
F-04-03	The PQ10 bus connector shall be centered horizontally and 1 mm from the top edge.	PQ9 §3.1
F-04-04	Board-to-board connectors shall be of the Samtec SQT-1XX-XX-L-S series.	PQ9 §3.2
F-04-05	There must be a 1 mm margin between PCBs.	PQ9 §3.2

Table 1.3: Additional requirements drawn from the PQ9 Standard [11]. See Section 3.2 for additional details.

1.6 System Architecture

The TigerCub spacecraft is split into five main subsystems: the electrical power system (EPS), the on-board computer (OBC) and attitude determination system (ADS), the radio communication system, the payload, and the structure. Figure 1.1 shows a high-level overview of how these subsystems interface with each other.

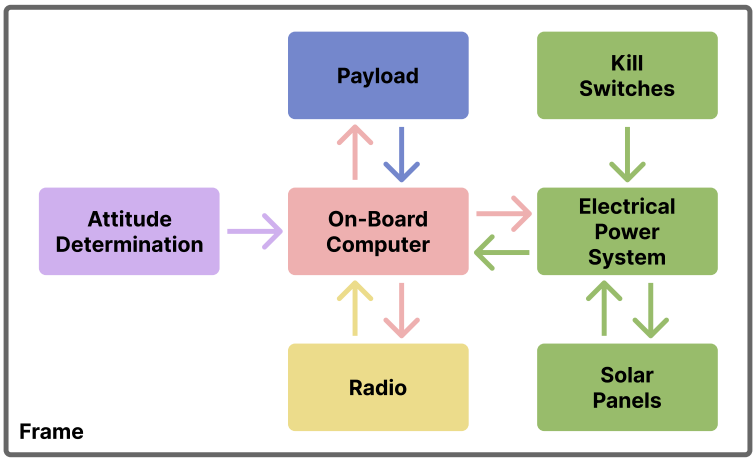


Figure 1.1: System block diagram.

The PocketQube Standard limits the mechanical architecture as described in the constraints in Table 1.1. For a 1p PocketQube, the external dimensions of the spacecraft should be $50 \times 50 \times 50$ mm, and a "sliding backplate" provides an interface with the PocketQube deployer [3]. The sliding backplate has dimensions of $58 \times 64 \times 1.6$ mm, suggesting that the backplate is typically a printed circuit board (PCB), which are commonly 1.6 mm thick.

There are few restrictions on the shape and structure of the main spacecraft frame. As described in Chapter 6, we chose to design a sheet-metal aluminum frame for inexpensive manufacturing, easy assembly, and weight reduction. The solar panels will be fastened to the aluminum frame, which will be fastened to the sliding backplate. Four screws through the sliding backplate will locate and fasten the internal PCB stack. The radio module will also attach to the aluminum frame.

The PCB stack is shown in Figure 1.2. The boards are connected via a custom PQ10 bus (using board-to-board connectors), as described in Section 3.4.

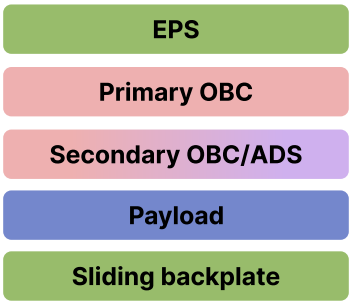


Figure 1.2: Diagram of PCB stack.

Locating the payload board near the bottom of the spacecraft allows potential payloads to use additional space on the sliding backplate and/or a cutout in the sliding backplate. Next, the primary and secondary OBC/ADS boards are located in the middle of the stack for easy access to both the payload and EPS boards. The OBC/ADS subsystems are split into two separate boards due to volume constraints. Finally, the EPS board, which uses the custom PQBH60 bus (as described in Section 4.2), connects to the primary OBC/ADS board.

Finally, the radio module is located at the top of the stack but is connected to the primary OBC/ADS board via a wire harness rather than a board-to-board connector. The radio module's patch antenna is exposed through a cutout in the aluminum frame.

Figure 1.3 shows a CAD model of TigerCub with labelled subsystems and components.

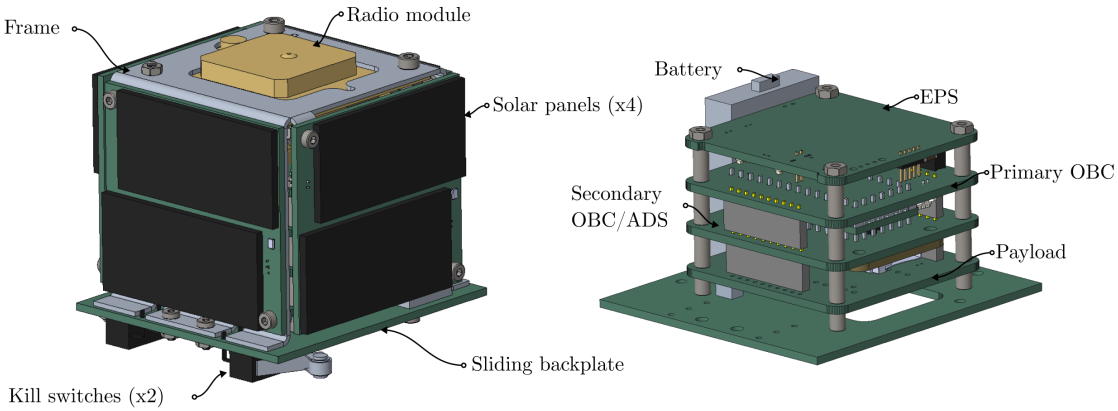


Figure 1.3: CAD model of TigerCub with external (left) and internal (right) views.

The design and analysis (and testing, if applicable) of each subsystem is detailed in the following chapters.

Chapter 2

Radio Communication System

The radio communication system is responsible for transmitting data to ground stations and receiving data from ground. We chose to use the RockBLOCK 9603 module from Ground Control for TigerCub's radio. Demonstrating the feasibility of the Iridium-based RockBLOCK module in a PocketQube form factor and as an student-accessible radio module is a primary motivation for TigerCub's development and drives the design of the remaining subsystems. This chapter discusses the RockBLOCK 9603 specifications, usage, and ground testing.

2.1 RockBLOCK 9603 Specifications

As mentioned in Section 1.3, the primary motivation for using the RockBLOCK 9603 module is that it communicates with the Iridium's 66-satellite network in low earth orbit (as opposed to communicating directly with a ground station) [12]. The reliable coverage of Iridium satellites and the ability to test the module from ground suggests viability in space. The radio's performance will actually improve in space in comparison to its performance on the ground because the link from the radio's modem to the Iridium constellation will not have to suffer atmospheric losses and the module will be physically closer to the Iridium satellites. The main communications challenge in orbit will be establishing a connection while the satellite is tumbling since TigerCub does not have an attitude control system. However, previous missions have demonstrated that Iridium transceivers in low-earth orbit can communicate even with a high tipoff rate of 20 degrees/sec at the time of deployment [13].

The RockBLOCK 9603 module features the Iridium 9603 modem, which boasts

"the smallest form factor of any commercial satellite transceiver available" [12]. Though we could incorporate the bare 9603 modem into our own radio module, the RockBLOCK module has the advantage of coming pre-built and is ready to use out of the box, which which simplifies assembly for students. Other pre-built Iridium modules exist, such as the NearSpace Launch EyeStar S4 radio previously used in the TigerSats Lab, but they are too large to fit in the PocketQube form factor [14].

Users communicate with the RockBLOCK over a serial connection, commonly used for other COTS devices. The module has a patch antenna along with the required power supply routing [15]. It also has a SMA connector for users to attach an external antenna [15]. TigerCub will only use the patch antenna since an external antenna would violate the component envelope, and the SMA connector also poses interference issues with the solar panels. The process of removing the SMA connector is detailed in Section 7.4.1.

Overall, the RockBLOCK 9603 was designed to be a lightweight module for volume-constrained applications [15]. Most importantly, using this pre-built module allows students to focus on their mission payload instead of designing a complex communications system.

Some specifications of the RockBlOCK 9603 module are listed Table 2.1.

Specification	**Value**
Dimensions	$45 \times 45 \times 16$ mm
Mass	39 g
Temperature range	-40°C to $+85$°C
Frequency range	Microwave L-band (requires line-of-sight)
Header connector	10-pin Molex Picoblade 1.25mm pitch

Table 2.1: RockBLOCK 9603 specifications [16, 17, 18].

The small dimensions and low mass help the RockBLOCK 9603 meet the PocketQube's volume and mass constraints. The large operational temperature range also makes the 9603 module suitable for use in the low earth orbit (LEO) environment.

2.1.1 Power Consumption

Along with the mass and volume budgets, the RockBLOCK also helps meet the stringent power budget. The RockBLOCK 9603 has four modes of operation [19]:

- Charging mode: During startup or after a long period of being powered off, the module must charge its supercapacitor. Since the TigerCub will attempt transmissions at least once per orbit, supercapacitor charging only needs to occur at the beginning of the mission.
- Idle mode: The module is powered on but not executing any commands.
- Active mode: The module is actively transmitting or executing a command.
- Sleep mode: A low-power mode to conserve energy. The module will be in this mode for the majority of the mission.

The following table describes the module's current consumption in each mode. The RockBLOCK always operates at a voltage of 5 V.

Mode	**Current Consumption (mA)**
Capacitor charging	500
Idle	40-50
Active	45-65 (average)
Sleep	0.1

Table 2.2: RockBLOCK 9603 current consumption [19].

Since the RockBLOCK will be asleep for over 90% of a typical mission, it will consume very little power on average. Radio duty cycles and power budgeting are outlined in Section 4.6.

2.1.2 Wiring

The RockBLOCK 9603 uses a 10-pin Molex Picoblade 1.25 mm header connector to interface with other components. Table 2.3 shows the connector pinout.

We communicate with the RockBLOCK over serial, connecting pins 1 and 6 to serial ports on the OBC. We power the RockBLOCK at the operating voltage of 5V, and control sleep mode using the OnOff pin. Thus, we need five pins to be connected to the OBC: pins 1, 6, 7, 8, and 10. We will connect the RockBLOCK module directly to the primary OBC board (OBC1) using a custom wire harness. One end of the harness has the mating connector to the RockBLOCK's 10-in Molex

Picoblade header: the Molex Picoblade 51021-1000. The other end has a five-pin Molex Picoblade connector, the 51021-0500. The harness connections are shown in Figure 2.1. We use Molex Picoblade pre-crimped wires, such as the Molex 79758-0006 wires.

Pin	Name	Description
1	RXD	Iridium 9603N RX (output from RockBLOCK)
2	CTS	Iridium 9603N CTS
3	RTS	Iridium 9603N RTS
4	NetAv	Network available signal
5	RI	Ring indicator signal
6	TXD	Iridium 9603N TX (input to RockBLOCK)
7	OnOff	Sleep control (pull to GND to switch off)
8	5V In	5V power supply
9	Li-Ion	3.7V power supply
10	GND	Ground

Table 2.3: RockBLOCK 9603 pinout [20].

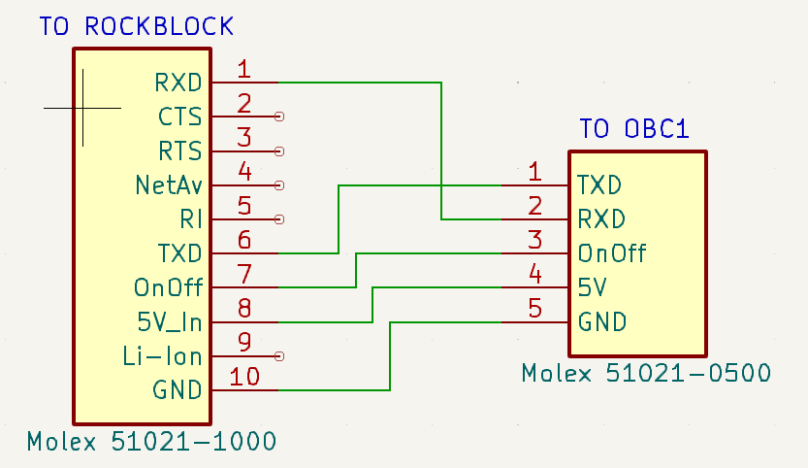


Figure 2.1: RockBLOCK to OBC1 harness diagram.

Connections from the 5-pin header on the OBC1 board to the microcontroller are described in Section 3.5.

2.2 Transmitting and Receiving

The Iridium 9603 modem can transmit and receive ASCII or binary messages. Transmitted packets can be up to 340 bytes and received packets can be up to 270 bytes [21]. Since binary messages are smaller than ASCII messages, converting messages to binary allows users to send more data per transmission. Messages sent from the RockBLOCK are designated Mobile Originated (MO) and messages sent to the RockBLOCK are Mobile Terminated (MT) [21].

To send a message, the message must first be transferred from the microcontroller to the RockBLOCK's MO buffer via the serial connection. Then, the Iridium modem sends the message through a short burst data (SBD) transmission and receives the next MT message, along with the status of the MO transmission. Finally, the MT message must be moved from the MT buffer on the RockBLOCK to the microcontroller [21].

RockBLOCK transmissions use a set of AT (attention) commands as described in the ISU AT Command Reference [22]. Commands should begin with `AT` and end with a carriage return `\r`. Some common commands are displayed in the following table.

Command	Description
`+SBDIX`	Start a SBD session.
`+SBDWT`	Write an ASCII message into the MO buffer.
`+SBDWB`	Write a binary message into the MO buffer.
`+SBDRT`	Read an ASCII message from the MT buffer.
`+SBDRB`	Read a binary message from the MT buffer.
`+SBDSX`	Get status.

Table 2.4: Common AT commands for RockBLOCK [23].

The `+SBDSX` command retrieves the status, which is a sequence of six numbers that describes the following [23]:

- MO status: status of the MO transmission. Any number from 0 to 2 denotes a successful transmission. 32 denotes that there is no network service (i.e. the RockBLOCK cannot communicate with the network).
- MOMSN: outbound sequence number; cycles between 0 and 65535.

- MT status: status of the MT transmission. 0 denotes that there are no new messages to be received. 1 denotes that there is a new message. 2 denotes that there was an error.
- MTMSN: inbound sequence number; cycles between 0 and 65535.
- MT length: number of bytes in the received MT message.
- MT queued: number of messages waiting to be received.

For instance, a status sequence of `0, 0, 0, 0, 0, 0` means that the MO message was successfully transmitted, no new messages were received, and there are no new messages in the queue [24].

A sample series of transmissions is as follows.

Command	**Response**
`AT\r`	
	`OK\r`
`AT+SBDWT=Hello World\r`	
	`OK\r`
`AT+SBDIX\r`	
	`0, 1, 0, 0, 0, 0\r`

The first command checks that the RockBLOCK is awake, which is confirmed when the RockBLOCK sends the `OK` response. The next command sends an ASCII message `Hello World`. The final command retrieves the status, which reports that the `Hello World` message was successfully transmitted and no new messages were received.

To view messages sent from the RockBLOCK, Rock7 provides an online admin portal at `rockblock.rock7.com`. From the admin portal, users can connect their RockBLOCK devices, view and send messages, and buy credits for future messages [25]. A monthly line rental costs 13 Euros (14.22 USD), and each credit (each successful transmission uses one credit) costs 0.145 Euros (0.16 USD).

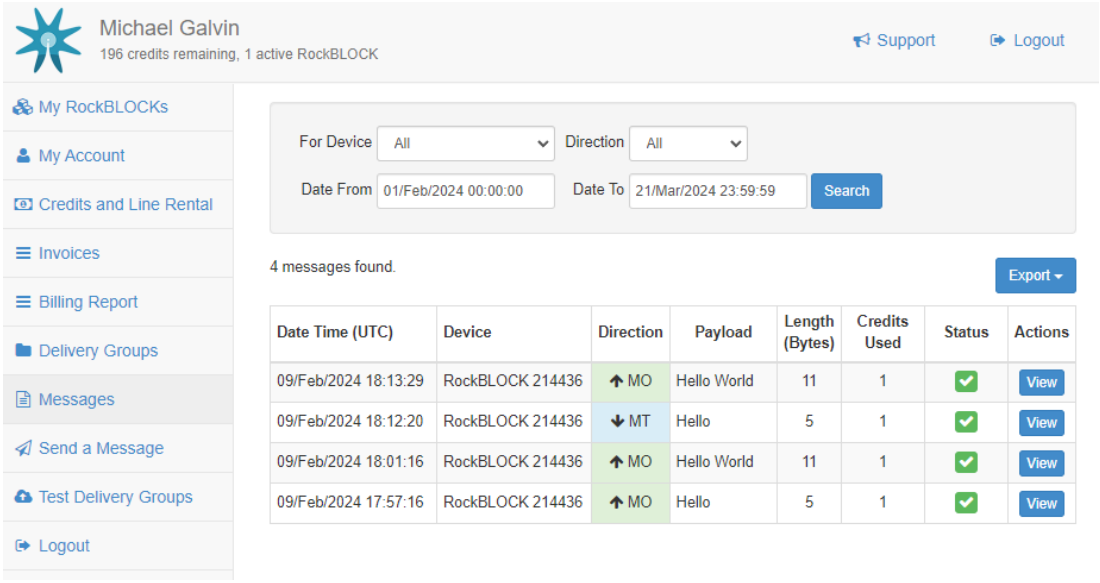


Figure 2.2: Screenshot of messages page in Rock7 portal.

The RockBLOCK-Iridium system is shown in the diagram below, from Adafruit.

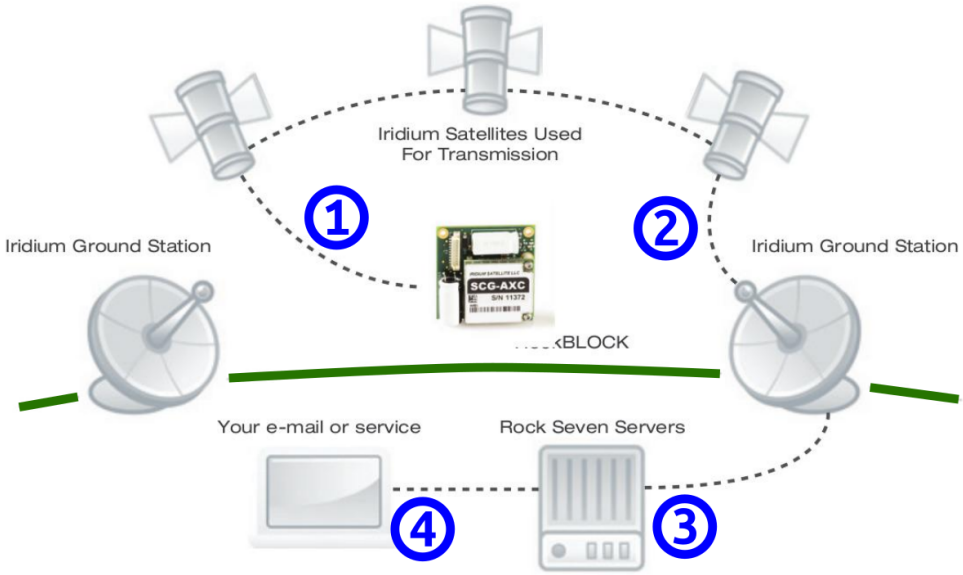


Figure 2.3: Transmission sequence from RockBLOCK to Rock7 portal [23].

2.3 Ground Testing

2.3.1 Transmitting and Receiving

We conducted ground testing with the RockBLOCK 9603 module to confirm that we could transmit and receive messages using the Iridium network from Earth.

As a proof of concept, we first attempted transmitting using a USB serial connection, connecting the RockBLOCK module to a laptop computer via the provided FTDI USB cable. The RockBLOCK module was set up indoors next to a large window, and the patch antenna had a clear view of the sky as shown in Figure 2.4.

Figure 2.4: Setup for transmitting and receiving ground test.

The following Python script sends an ASCII message to the RockBLOCK via the Iridium network and receives the status of the message.

```
import serial
import io

# set up serial connection
ser = serial.serial_for_url('COM4', timeout=1)
sio = io.TextIOWrapper(io.BufferedRWPair(ser, ser), newline="\r\n")
ser.baudrate = 19200

# check that RockBLOCK is awake
```

```
ser.write("AT\r".encode())
print(ser.readline()) # should receive 'AT'
print(ser.readline()) # should receive 'OK'

# send ASCII message
ser.write("AT+SBDWT=Hello\r".encode())
print(ser.readline()) # should receive 'AT+SBDWT=Hello'
print(ser.readline()) # should receive 'OK'

# get status
ser.write("AT+SBDIX\r".encode())
print(ser.readline()) # should receive 'AT+SBDIX'
print(ser.readline()) # should receive status sequence of 6 numbers

# close out serial connection
ser.close()
```

This script successfully transmitted the `Hello` message through Iridium, as depicted in the earliest message in Figure 2.2. We were also able to receive a message sent from the online portal to the RockBLOCK.

Then, after a successful transmission using USB serial, we next wired the RockBLOCK up to TigerCub's primary microcontroller, a Teensy 4.0 from PJRC. We powered the Teensy and RockBLOCK via an external 5V power supply, then created pin-to-pin connections using a custom harness with the Molex 51021-1000 and jumper wires on the other. The pin-to-pin connections between the Teensy and the RockBLOCK are displayed in the table below. Note that the OnOff pin on the RockBLOCK was not connected for this test since we were not testing sleep control. Figure 2.5 displays the test setup.

Pin on Teensy	**Pin on RockBLOCK**	**Power Supply**
GND	GND	GND
–	5V	5V
TX2	TXD	–
RX2	RXD	–

Table 2.5: Pin-to-pin connections from Teensy to RockBLOCK.

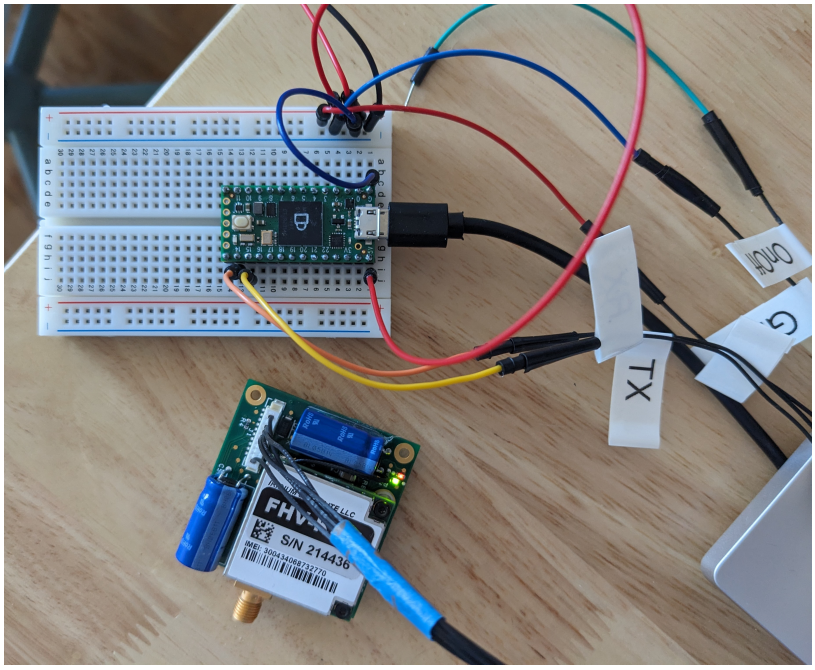


Figure 2.5: Test setup for Teensy 4.0 and RockBLOCK module.

The `IridiumSBD` Arduino library provides functions for an Arduino microcontroller to interface with a RockBLOCK module [26]. The `BasicSend.ino` example demonstrates startup and transmission operations. The full script is provided in Appendix B.1.

Using the `BasicSend.ino` script, we successfully transmitted a `Hello World` message through the Iridium network and received it through the online portal as shown in the most recent message in Figure 2.2.

We have thus demonstrated that it is possible to transmit and receive using the RockBLOCK 9603 module from ground. As previously mentioned, since it is more difficult to transmit from ground than from low earth orbit, we suggest that it is possible to communicate to a ground station from low earth orbit using the RockBLOCK module, satisfying Requirement P-03 from Table 1.1.

2.3.2 Sleep Control

Due to the power restrictions of a fixed-panel solar array as discussed in Section 4.6, we need the RockBLOCK to operate in a low-power mode for the majority of the mission. This is accomplished by putting the RockBLOCK in Sleep mode, in which the RockBLOCK only consumes 0.1 mA (on average) [19]. To put the RockBLOCK in sleep mode, we pull its `OnOff` pin to ground [17]. The RockBLOCK's `OnOff` pin

is connected to the Teensy's digital pin 6. An example of turning sleep mode on and off is shown below:

```
#include <IridiumSBD.h>
#define IridiumSerial Serial3

IridiumSBD modem(IridiumSerial);
const int onOffPin = 6;

void setup() {
  int err;

  // start the console serial
  Serial.begin(115200);
  while (!Serial);

  // start the RockBLOCK serial
  IridiumSerial.begin(19200);

  // set Teensy on/off pin as output
  pinMode(onOffPin, OUTPUT)

  // begin modem operation
  // code block from BasicSend example by SparkFun
  Serial.println(F("starting modem..."));
  err = modem.begin();
  if (err != ISBD_SUCCESS)
  {
    Serial.print(F("Begin failed: error "));
    Serial.println(err);
    if (err == ISBD_NO_MODEM_DETECTED)
      Serial.println(F("No modem detected: check wiring."));
    return;
  }
}

void loop() {
```

```
35   // put RockBLOCK in sleep mode
36   digitalWrite(ledPin, LOW);
37   // wait 60 seconds
38   delay(60000);
39
40   // activate RockBLOCK
41   digitalWrite(ledPin, HIGH);
42   delay(60000);
43 }
```

We ran this script and confirmed that the RockBLOCK's current consumption drops from about 40mA (while the RockBLOCK is active/idle) to less than 10 mA after just a few minutes.

2.4 Data Budget

As previously mentioned in Section 2.2, the RockBLOCK can transmit packets that are at most 340 bytes and receive packets that are at most 270 bytes [21].

ASCII messages consume one byte per character, so any ASCII messages transmitted from RockBLOCK (assuming one successful packet per transmission attempt) can be at most 340 characters long. However, most data sent from the PocketQube will be either integers or floating point numbers, which can be efficiently compressed. For instance, floating point numbers can be converted to their byte representations, which consumes only 4 bytes (for single-precision) [23].

The RockBLOCK must transmit telemetry data (including satellite orientation, microcontroller temperature, and battery status) as well as payload data to ground. Assuming satellite orientation data uses 9 floating-point numbers, temperature data uses 1 float, and battery status uses 1 float, telemetry data will consume $(9+1+1)4 = 44$ bytes total. This leaves 296 bytes for payload data. Developers may want to send multiple telemetry data points for each RockBLOCK transmission, which would consume more bytes. Depending on the data format (e.g. if the payload needs to send images back to Earth), future developers may have to compress their data.

The RockBLOCK receives commands from ground. Most PocketQube missions will likely run autonomously, but any commands sent from ground should be less than 270 bytes.

Chapter 3

On-Board Computer and Attitude Determination System

This chapter discusses the design of the on-board computer (OBC) and attitude determination system (ADS). In particular, we discuss the PQ9 electrical standard, the selection of commercial-off-the-shelf (COTS) components, the design and test of the PQ10 bus and OBC boards, as well as fault detection and recovery methods.

3.1 Subsystem Overview

The OBC is the heart of the satellite. It issues commands to other subsystems and is responsible for handling data between subsystems. The OBC and ADS are comprised of a primary microcontroller for command and data handling, an inertial measurement unit (IMU) for attitude determination, and a Micro SD card for external memory and data storage. The OBC is also connected to the electrical power system (EPS) and the payload via a custom PQ10 bus. The RockBLOCK interfaces with the OBC via a wire harness as shown in Figure 2.1. A high-level overview of the system is shown in Figure 3.1.

The OBC is separated into two printed circuit board assemblies (PCBAs) due to volume constraints. The primary OBC board, named OBC1, contains the Teensy as well as connectors to the RockBLOCK module, kill switches, and EPS. The secondary

OBC board, OBC2, contains the IMU breakout board and the MicroSD breakout board. Specific design considerations are described in Sections 3.5 and 3.6.

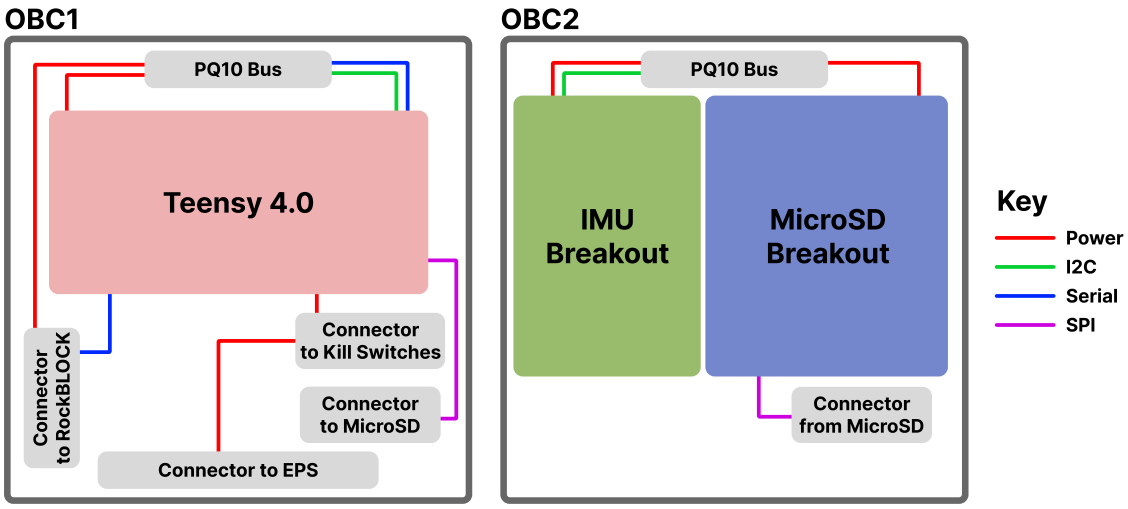


Figure 3.1: OBC/ADS block diagram.

3.2 PQ9 Standard

The PocketQube Standard (Issue 1) does not have any electrical requirements; they are to be written in the second issue, which has not yet been released [3]. PocketQube developers commonly use the PQ9/CS14 Standard instead, which is an electrical and mechanical interface standard written for PocketQubes (PQ9) and CubeSats (CS14) [11]. We will be conforming to the PQ9 mechanical requirements for our PCBAs. However, because PQ9 uses the RS-485 serial interface but most of our components use I2C or serial, we will not using the same electrical requirements. That being said, our design will be inspired by PQ9 and the electrical design in the previously-mentioned PyCubed-Mini PocketQube.

The additional mechanical requirements from PQ9 were described in Table 1.3 in Section 1.1. Figure 3.2 shows the PQ9 PCB outline referenced in Requirement F-04-02.

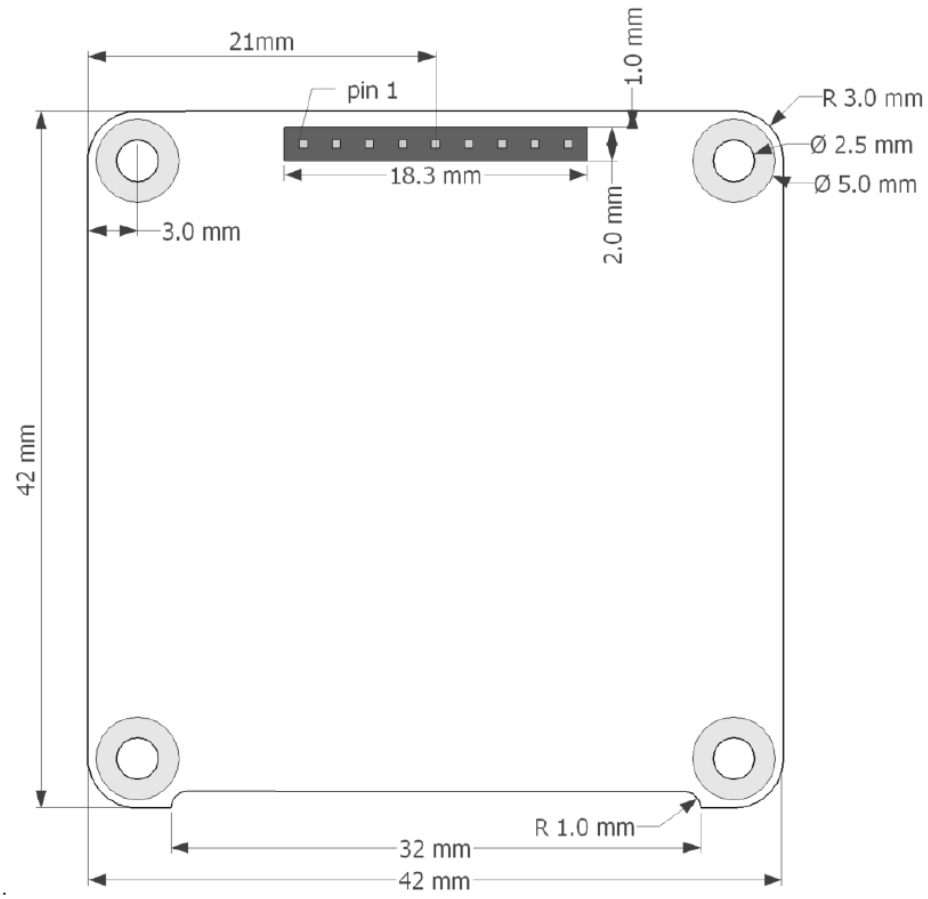


Figure 3.2: PQ9 printed board outline as referenced in Requirement F-04-02 [11].

3.3 COTS Components

Due to my limited circuit design experience and the objective of creating a simple OBC for future student use, we sourced COTS breakout board components for the primary microcontroller, attitude determination system, and on-board data storage. Though these consume more volume than discrete components, the pre-built breakout boards simplify circuit design and assembly for future student use. This section details the selection of the COTS components and their specifications.

3.3.1 Primary Microcontroller

Microcontroller Trade Study

According to the PQ9 interface (Requirement F-04-01), the primary microcontroller must be at most 42 mm in any dimension, which constrains us to a few commercial-off-the-shelf COTS microcontroller development board options. Some other considerations for our primary microcontroller are as follows:

- Dimensions: The microcontroller board should be as small as possible to leave room for other components.
- Communication: The microcontroller board should have at least one I2C line and at least two serial lines to interface with the EPS, ADS, radio, and payload subsystems. The EPS and ADS both use I2C, the radio uses serial, and the payload should have the option of either I2C or serial.
- Power: The microcontroller board should have a low-power mode to conserve power during periods of low usage.
- Availability: The microcontroller board should be produced and distributed by reputable vendors and should be in stock for the foreseeable future ($\approx$5 years).

With these considerations, we picked the following boards to compare: the PJRC Teensy 4.0, the PJRC Teensy 3.2, the Adafruit Trinket M0, and the SparkFun Pro Micro. These are all popular microcontroller development boards; the TigerSats Lab has experience with the Teensy 4.0 and Teensy 3.2 in particular. They all use similar microchips (the ATSAMD21 or ATmega32U4). Table 3.1 displays a trade study of these microcontroller boards based on the factors we identified.

Microcontroller	Dimensions	Communication	Power	Availability
PJRC Teensy 3.2	30 × 18 mm	2 I2C, 3 Serial, 1 SPI	3.3V operating voltage low power mode available	Out of stock
PJRC Teensy 4.0	36 × 18 mm	3 I2C, 7 Serial, 3 SPI	3.3V operating voltage low power mode available	Available
Adafruit Trinket M0	27 × 15.3 mm	1 I2C, 1 Serial, 1 SPI	3.3V operating voltage low power mode available	Available
SparkFun Pro Micro	33 × 18 mm	1 I2C, 1 Serial, 1 SPI	5V operating voltage low power mode available	Available

Table 3.1: Microcontroller board trade study [27, 28, 29, 30].

Out of these four, the trade study suggests that the Teensy 4.0 is the best option since it meets the data channel requirements and is in stock for the foreseeable future. It is the largest board out of the four we compared, but still meets the dimension requirements.

Teensy 4.0 Specifications

The Teensy 4.0 development board has the following specifications.

Specification	Value
Processor	ARM Cortex-M7 at 600 MHz
Memory	1984K flash, 1024K RAM, 1K EEPROM
Pins	40 digital I/O, 31 PWM output, 14 analog input, 1 LED
Communication	USB, 7 serial, 3 SPI, 3 I2C
Timing	24 MHz crystal for system clock, 32.768 kHz crystal for real time clock, 3 watchdog timers
Power	5V input, 3.3V operating voltage, 100 mA current consumption at 600 MHz
Dimensions	36.26 × 17.78 × 3.64 mm

Table 3.2: Teensy 4.0 specifications [28].

Other than these specifications, it is important to note that the Teensy 4.0 does not have a hardware reset signal; instead, resets are done through software [28]. The reset procedure is detailed in Section 3.8.

Also, it is possible to lower the Teensy's power consumption by clocking down the processor. Users have pushed the current consumption as low as 36 mA at 16 MHz [31, 32]. For this project, we will assume that we can clock down the processor to operate around 50 mA for typical operations. We will further reduce the power consumption by using the `Snooze` library to bring the current consumption during sleep mode to 20 mA [33]. For additional power savings, it is also possible to turn off the 3.3V output using the OnOff pin.

3.3.2 Attitude Determination

For attitude determination, we use an inertial measurement unit (IMU), common in nanosatellites. As with the microcontroller board, our primary driver for the IMU is size; it must fit on a PCBA along with the on-board data storage solution. The IMU must also have a 9 degrees of freedom (9-DoF) sensor (as opposed to a 6 DoF sensor), which typically includes an accelerometer, a gyroscope, and a magnetometer. The gyroscope measures the angular speed and can be used to determine tumble speed and attitude [34]. The magnetometer can detect where the strongest magnetic force comes from (typically magnetic north); it helps provide an absolute orientation reference for the satellite's attitude [34]. Finally, accelerometers can detect spacecraft deceleration (typically due to atmospheric drag).

Adafruit provides a few small IMU breakout boards, based on sensors such as the BNO055, the ICM-20948, and the LSM6DS3TR-C + LIS3MDL. These breakout boards provide an easy interface with the IMU sensors. Adafruit's boards are functionally very similar for our purposes; they all take 3-5V input and use I2C to transfer data. We chose the smallest of the three, which is the the Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU. Its relevant pinouts are listed in the table below.

Pin Name	Description
VIN	3-5VDC input
3Vo	3.3V output, up to 100 mA
GND	Ground
SCL	I2C clock pin (with 10KΩpullup)
SDA	I2C data pin (with 10KΩpullup)

Table 3.3: Adafruit LSM6DS3TR-C + LIS3MDL pinout [35]. Additional unused pins are not included.

Of these pins, we use VIN and GND to provide power and the SCL/SDA I2C pins to provide attitude determination data to the Teensy.

To use the Adafruit LSM6DS3TR-C + LIS3MDL breakout board with the Teensy, we installed the Adafruit LIS3MDL library and the Adafruit LSM6DS libraries in the Arduino integrated development environment (IDE). To use the breakout board as-is, we must change the `LSM6DS3_CHIP_ID` definition from `0x69` to `0x6A` in the library's `Adafruit_LSM6Ds3.h` file. This file can typically be found in the users's `Documents\Arduino\libraries\Adafruit_LSM6DS` folder. Then, to test the IMU, we can use sample code provided from Adafruit as shown in Appendix B.2.

3.3.3 On-board Data Storage

Similar to the IMU, our on-board data storage solution is primarily driven by size. We chose to use a microSD card for extra data storage. Adafruit also has several microSD breakout boards available, and their smallest is the Adafruit Micro SD SPI or SDIO Card Breakout Board. This breakout board and the IMU breakout board fit on one PCBA. The pinout is listed below.

Pin Name	Description
3V	3.3V input
GND	Ground
DET	True when a microSD card is inserted, False otherwise
CLK	SPI clock
SO	SPI serial out
SI	SPI serial in
CS	SPI chip select; drop low to start SPI transaction

Table 3.4: Adafruit MicroSD SPI or SDIO Card Breakout Board pinout [36].

To test the breakout board, we use the `SD` Arduino library, which comes preinstalled. We can run the `ReadWrite.ino` example sketch; the script is included in the Examples folder in the Arudino IDE and also in Appendix B.3.

3.4 PQ10 Interface

CubeSats and PocketQubes with an internal board stack often use a standard connector interface to run power and signal lines between boards; examples are the PC104 and CS14 standards used in CubeSats and the PQ60 and PQ9 standards for PocketQubes. Though the PQ60 standard is more popular, its 60-pin connector has more pins than we need and is difficult to solder by hand, making it less accessible for students [37]. Instead, TigerCub will use a custom 10-pin connector interface on its OBC boards and payload board. This interface, modeled after the PQ9 standard, will be called PQ10.

The PQ9 connector pinout is as follows:

1	2	3	4	5	6	7	8	9
RST	485A	485B	GND	V_1	V_2	V_3	V_4	GND

Table 3.5: PQ9 connector pinout [11].

The PQ10 connector pinout is:

1	2	3	4	5	6	7	8	9	10
RST	SCL0	SDA1	GND	5V	3V3	SCL1/RX1	SDA1/TX1	RX2	TX2

Table 3.6: PQ10 connector pinout.

Two key differences between the two interfaces are explained below:

- PQ9 uses the RS485 standard while PQ10 uses I2C and serial communication. This decision was made because the EPS and Adafruit LSM6DS3TR-C + LIS3MDL breakout board both use I2C, while the RockBLOCK 9603 uses serial. We also wanted to allow the payload to use I2C or serial communication. No components on the satellite use RS485, and we do not forsee using RS485 communication in the future.

- PQ9 has four voltage lines and two ground lines while PQ10 has two voltage lines and one ground line. All other subsystems on TigerCub use either 5V or 3.3V input power (which are common logic levels for most hobby-class components), so there is no need for four voltage lines. We are also maintaining a common ground for all subsystems, so only one ground line is necessary.

To comply with Requirement F-04-04, we use the Samtec SQT-110-03-L-S and Samtec SQT-110-01-L-S connectors. Both are 10-pin connectors with a height of 6.35 mm. The Samtec SQT-110-01-L-S connector has a shorter pin length than the Samtec SQT-110-03-L-S and is used on OBC1 to maintain clearances above the EPS board.

3.5 Primary Board Design

The primary OBC board, named OBC1, contains the Teensy 4.0, the PQ10 bus connector, and breakout connectors to the RockBLOCK module, the MicroSD breakout board, and the EPS. The board schematic is shown in Figure 3.3 and the PCB design is shown in Figure 3.4. Both OBC1 and OBC2 were fully designed in KiCAD, an

open-source schematic capture and PCB design software. KiCAD is easy to pick up for beginners have little to no PCB design experience as well as more experienced designers who have used other software such as Altium Designer or DipTrace.

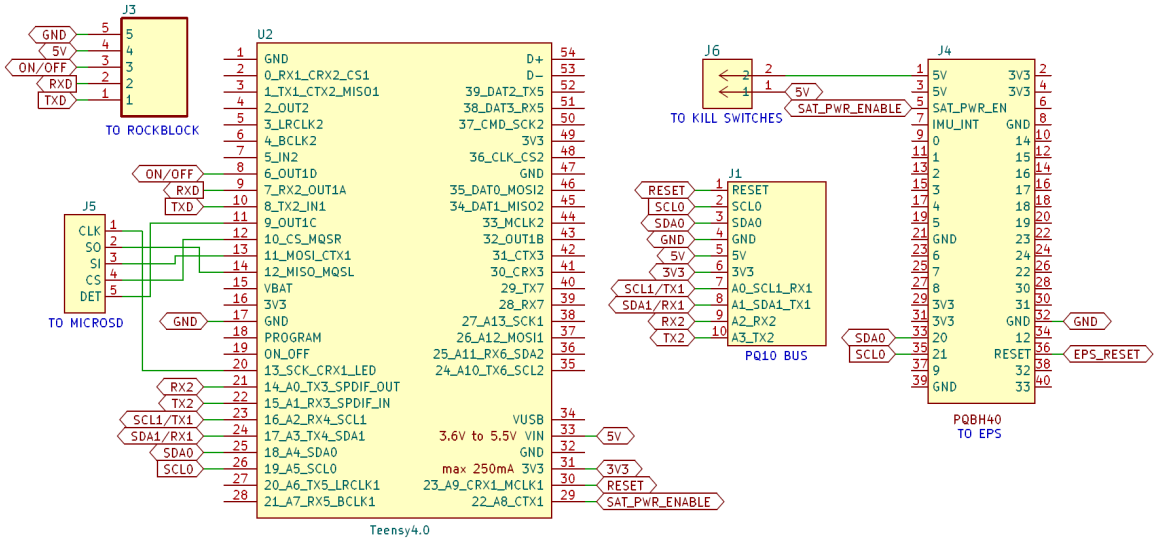


Figure 3.3: OBC1 board schematic.

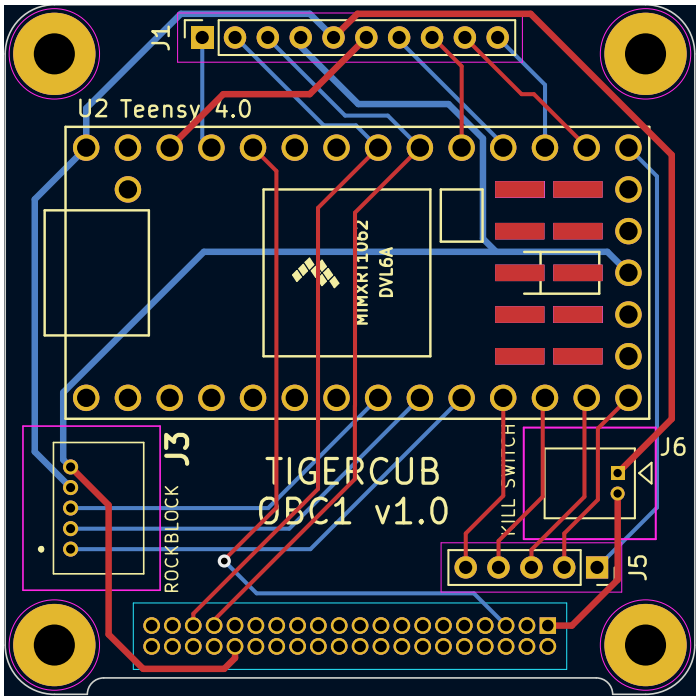


Figure 3.4: OBC1 PCB design.

3.5.1 Power

There are two voltage lines on the TigerCub: 5V and 3.3V.

5V power is routed from the EPS, through the kill switches, to the PQ10 bus (pin 5) and Teensy 4.0. Routing 5V power through the kill switches ensures that if the kill switches are closed (i.e. the TigerCub is still in its deployer), the 5V power line will be disconnected and the satellite will be unpowered; this functionality is derived from Requirements F-01 and F-02 (see Table 1.1). Further discussion of the kill switch logic is included in Section 4.3. The RockBLOCK module is also powered by 5V power from the PQ10 bus.

3.3V power is routed from the Teensy's 3.3V power output to the PQ10 bus (pin 6). Components on a 3.3V logic level (such as the MicroSD breakout board and the IMU) are powered by this line. Note that the Teensy can provide at most 250 mA of power through its output pin, so this limits the current consumption of the payload; however, due to the small power budget, the payload cannot draw this much current anyways.

The `SAT_PWR_ENABLE` signal runs from the EPS to the Teensy 4.0. This signal, which is controlled by the EPS's own microcontroller, must be turned on in order for 5V power to flow from the EPS. This is further discussed in Section 4.2.

Finally, all components share a common ground, set through the EPS.

3.5.2 Communication

The PQ10 bus has two I2C lines and two serial lines; one I2C line and one serial line share the same pins, so effectively there are either two I2C lines and one serial line or one I2C line and two serial lines available as shown in Table 3.6. Developers can choose which to implement. The I2C and/or serial communication protocols are compatible with the large majority of hobby-class electronics and sensors.

The first I2C line (pins 2 and 3 on the PQ10 bus) are connected to the first I2C line on the Teensy (pins 18 and 19). An I2C line runs from the EPS to this I2C line as well. The second I2C line/first serial line (pins 7 and 8 on the PQ10

bus) are connected to pins 16 and 17 on the Teensy, which can handle serial or I2C communication. Finally, the second serial line on the PQ10 bus (pins 9 and 10) are connected to pins 14 and 15 on the Teensy.

Note that the RX and TX labels refer to the payload input and output, respectively. The payload's RX is connected to the Teensy's TX and vice versa.

3.5.3 RockBLOCK and MicroSD Connections

Two 5-pin connectors provide harness connections from the Teensy to the RockBLOCK and MicroSD breakout board. The RockBLOCK connector is a 5-pin Molex Picoblade connector, and the MicroSD connector is a 5-pin stackthrough connector of the Samtec SQT-105 series.

The RockBLOCK serial lines run to pins 7 and 8 on the Teensy. Note that the RX and TX pin names refer to the output and input from the RockBLOCK, respectively; this is flipped from the PQ10 pin denominations. Thus, the RX from the RockBLOCK is connected to the RX of the Teensy, and the TX from the RockBLOCK is connected to the TX of the Teensy. The RockBLOCK also has an `ON/OFF` signal which allows the Teensy to put the RockBLOCK in sleep mode, as discussed in Section 2.3.2.

The MicroSD breakout board's `CLK`, `SO`, `SI`, `CS`, and `DET` signals run to pins 13, 12, 11, 10, and 9 on the Teensy 4.0 respectively. Pins 10-13 on the Teensy are set aside for MicroSD SPI transactions.

3.6 Secondary Board Design

The design of the secondary OBC board, named OBC2, is considerably simpler than that of OBC1. OBC2 contains the MicroSD breakout board, the MicroSD breakout connector, the IMU breakout board, and the PQ10 bus connector as depicted in Figure 3.1. The schematic for OBC2 is shown in Figure 3.5 and the PCB is shown in Figure 3.6. Note that the breakout boards contain their own pullup resistors for the I2C pins.

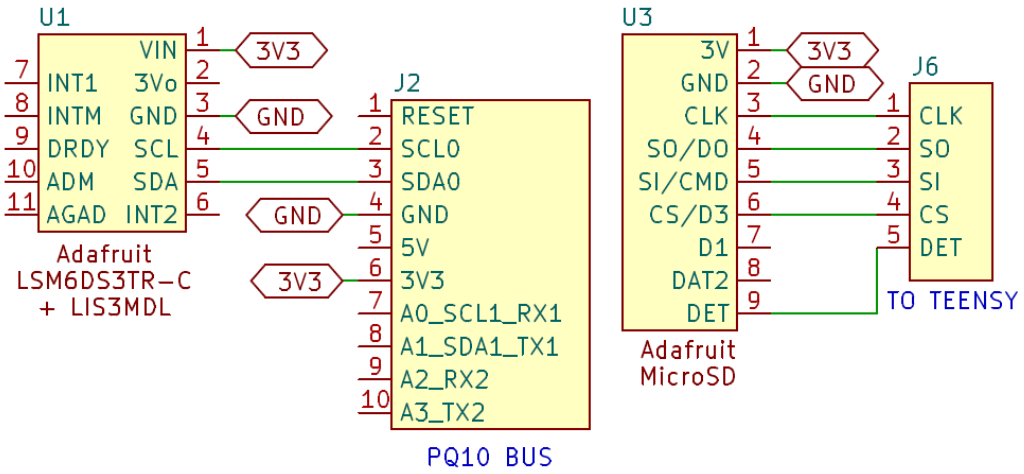


Figure 3.5: OBC2 board schematic.

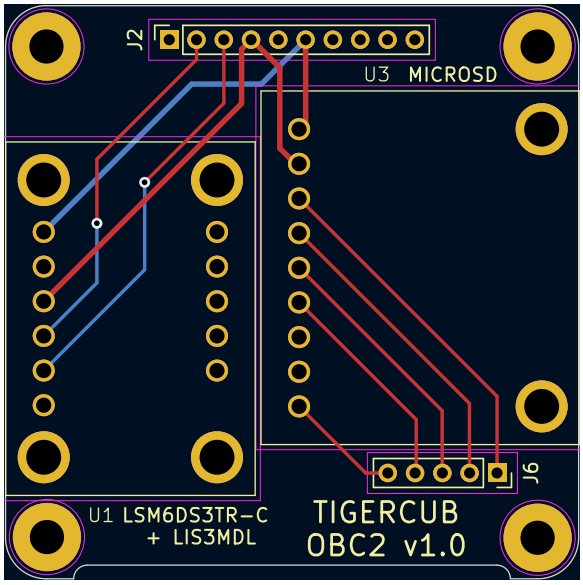


Figure 3.6: OBC2 PCB design.

Power flows from the 3.3V line on the PQ10 bus to both the MicroSD and IMU breakout boards. The IMU provides data to the Teensy through the first I2C line on the PQ10 bus. The MicroSD card has its own 5-pin breakout connector that connects to the Teensy as previously described in Section 3.5.3.

OBC manufacturing and assembly are described in Chapter 7.

3.7 Testing

Once OBC1 and OBC2 were manufactured and assembled, we conducted verification tests to demonstrate that components performed as expected. We stacked the two boards together using the stack-through connectors and powered them through the Teensy's USB connector as shown in Figure 3.7.

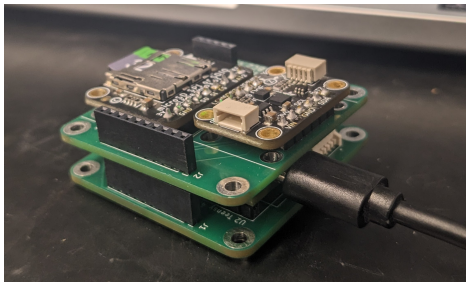


Figure 3.7: OBC1 and OBC2 test setup.

We ran the `IMU_Demo.ino` and `MicroSD.ino` scripts (included in Appendix B) and successfully received IMU measurements and performed a MicroSD read/write transaction as shown in Figures 3.8 and 3.9.

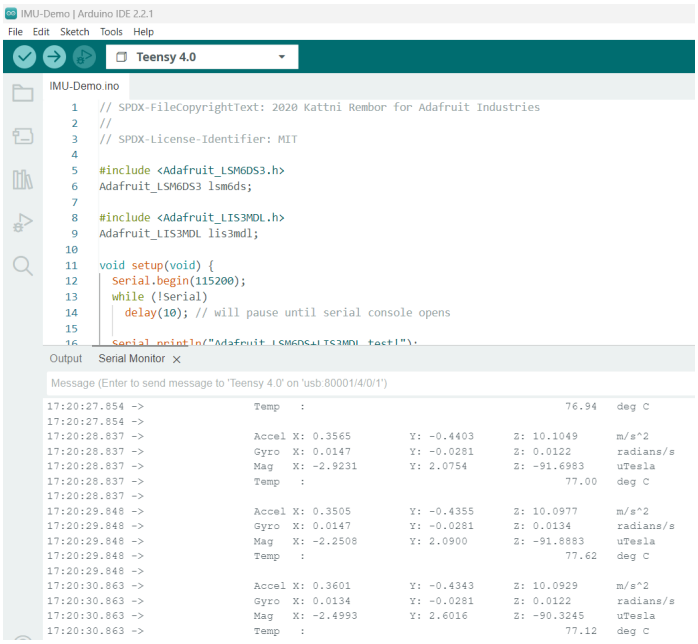


Figure 3.8: Screenshot of successful IMU measurements using OBC stack. Teensy reported accel/gyro/mag measurements.

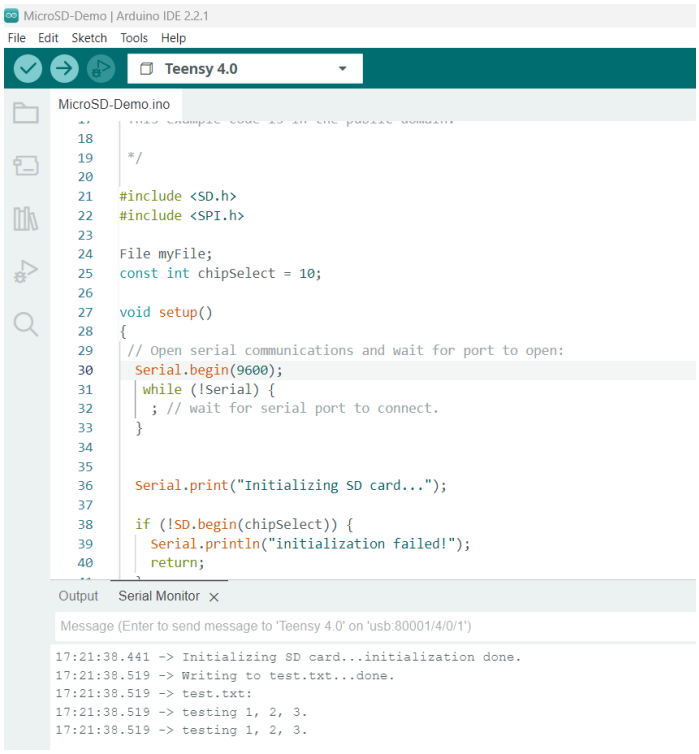


Figure 3.9: Screenshot of successful MicroSD demonstration. Teensy successfully read and wrote from the inserted MicroSD card.

3.8 Fault Detection and Recovery

In the space environment, radiation can lead to single event effects (SEEs) in electronic components, typically caused by cosmic rays and high energy protons; COTS components are particularly susceptible to these effects [38].

Some common SEEs are single event upsets (SEUs), single event latchups (SELs) and single hard errors (SHEs). SEUs are typically caused by a single ion and leads to a change of state or transient in digital, analog, and optical components such as the OBC components; a reset of the device typically corrects the effect [39]. SELs create a short-circuit in the component that may or may not cause permanent damage, and can sometimes be corrected by power cycling [39]. SHEs cause a permanent change to the device's functionality and cannot be reversed [39].

Radiation over long periods of time build up to a total ionizing dose (TID), which typically causes electronic components to fail [40]. SHEs and failures due to TID are

permanent failures and cannot be corrected. In the context of OBC design, we will focus on correcting the effects from other SEEs. We address the effects of TID in Section 6.5.2.

Microcontrollers typically use "watchdogs" to look for abnormal functionality. A watchdog monitors a signal (typically called a "heartbeat" signal in spacecraft) and if it does not receive the signal for a set period of time (sometimes called "feeding the watchdog"), it triggers a reset of the system [41]. The TigerCub has two types of resets: a software reset via the microcontrollers, or a hardware reset via power cycling by the EPS.

Most components in TigerCub can be reset by the Teensy. The RockBLOCK module can be turned off (i.e. put in sleep mode) through pulling its `OnOff` pin to ground as detailed in Section 2.3.2. The MicroSD and IMU breakout boards can be power cycled by turning the 3.3V output of the Teensy off and on. The Teensy itself has an internal software reset available through the `Watchdog` Arduino library [42]. The `Watchdog` library contains four different types of watchdogs, each with different functionality. Developers should use `WDOG1` for the Teensy watchdog, which performs a internal reset if the watchdog is not fed after a specified time interval [43].

The payload reset is controlled through the PQ10 bus's `RESET` line (connected to pin 23 on the Teensy), allowing developers to drive a reset signal to their payload; the payload should implement its own heartbeat signal using the bus serial or I2C lines. The payload watchdog should use `WDOG2` from the `Watchdog` library, which performs a reset on a specific GPIO pin instead of the entire Teensy.

Demonstration scripts for `WDOG1` and `WDOG2` are provided in Appendix B.4.

In the case that the Teensy is not able to reset itself, the EPS can power cycle the entire stack by cutting off the 5V power line via the `SAT_PWR_ENABLE` pin.

Finally, the microcontroller on the EPS (the Atmel SAMD21) has its own watchdog timer, which will automatically reset the EPS when necessary [44]. In the case that the EPS fails due to an SEE and its internal watchdog also fails to reset the EPS, this would constitute mission failure; however, the risk of this are low due to the mitigations described above.

Figure 3.10 depicts the watchdog operations in the internal stack, where an arrow from one component to another denotes that the first component is the watchdog for the second component.

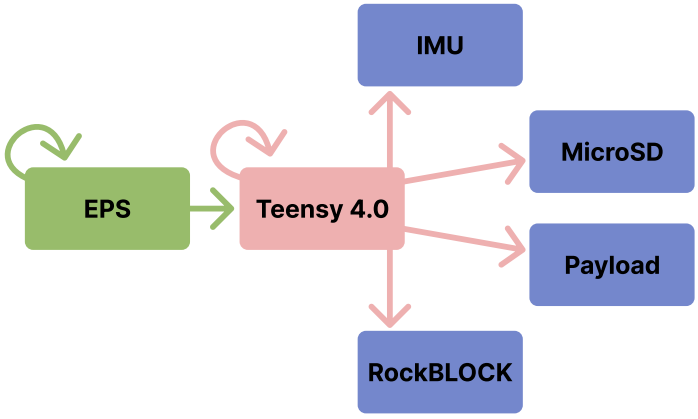


Figure 3.10: Diagram of watchdog components in internal stack.

Chapter 4

Electrical Power System

This chapter discusses the design of the electrical power system (EPS), including the DynOSSAT-EDU board, kill switches, battery, and solar panels, as well as analysis of the satellite's power budget.

4.1 Subsystem Overview

The EPS generates and distributes power to the spacecraft. TigerCub's EPS is composed of the the DynOSSAT-EDU board, two kill switches, a lithium ion battery, and four solar panels. A diagram of the EPS architecture is shown in Figure 4.1.

The four solar panels provide power to the DynOSSAT-EDU, which takes power from the solar panels using a maximum power point tracking algorithm, charges the lithium ion battery, and distributes power to the PQ10 bus. The two kill switches enable power from the EPS to the PQ10 bus via OBC1, as stipulated in the PocketQube Standard.

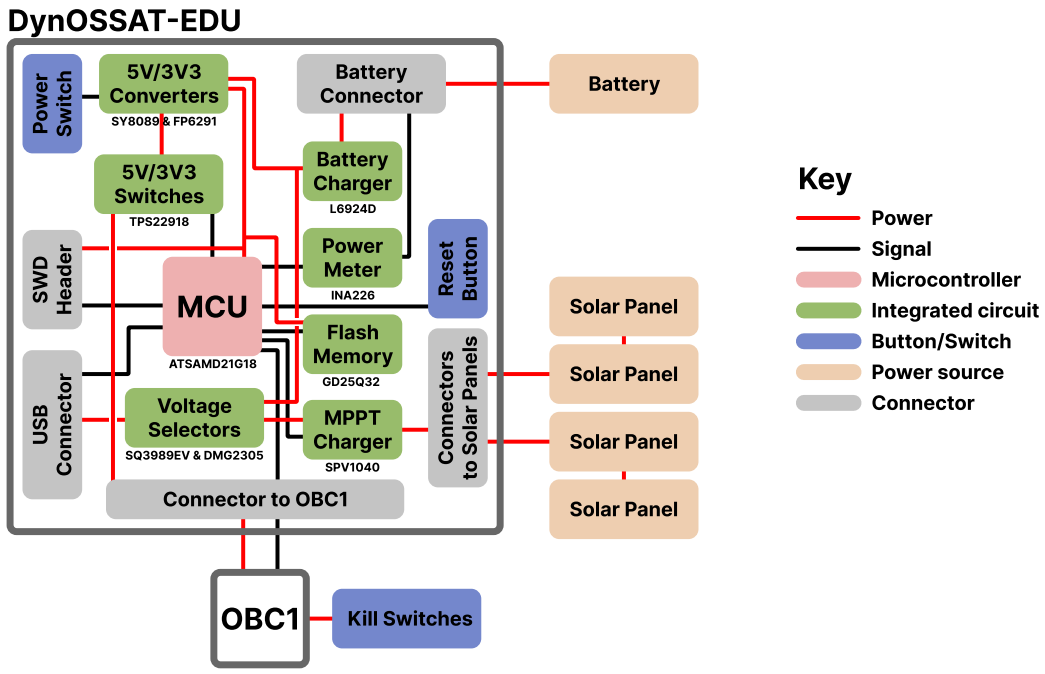


Figure 4.1: EPS block diagram.

4.2 DynOSSAT-EDU EPS

Designing an entirely new EPS board (especially without a background in electrical engineering) would have been beyond the scope of this project. In lieu of designing a new EPS, we decided to use the open-source DynOSSAT-EDU EPS, an EPS module built for PocketQubes [45].

There are few open-source PocketQube EPS modules available; as of the time of writing, we could only find the PQ60-EPS and the DynOSSAT-EDU. However, the PQ60-EPS is untested and there is very little explanation of its design and usage [46]. On the other hand, the DynOSSAT-EDU has been ground tested (at least by the creator), has several flight-proven components, and has an extensive Wiki page detailing design decisions as well as a User's Guide, though it has no verifiable flight heritage [45].

Some features of the DynOSSAT-EPS are as follows [45]:

- Atmel ATSAMD21E18 microcontroller unit with 256 KB flash memory and 32 KB random-access memory

- 32 MB SPI flash memory for storing code
- Lithium battery charger integrated circuit (IC) to charge the on-board battery using solar power
- Flight-proven maximum power point tracking (MPPT) IC for solar cell power optimization and management
- 5V and 3.3V power rails to provide power to other subsystems and components
- Flight-proven battery measurement with a digital meter
- Flight-proven back-power protection and load-switching
- USB-C connector for power and data logging
- Header connector for programming using a J-Link

These features help satellite developers properly manage, monitor, and control their power system without needing experience in detailed circuit design. The DynOSSAT-EDU EPS schematic is included in Appendix A.2.

The DynOSSAT-EDU EPS uses a similar mechanical interface as in the PQ9 standard, so it fits into the internal stack as the top board. To power the EPS module, plug the battery into the battery connector on the bottom of the board, and plug the two solar panel harnesses into the two solar panel connectors on the top side of the board.

The DynOSSAT-EDU stack was previously available for sale on Tindie, but has been out of stock since 2020 [47]. To build the EPS, we bought all the components and had them assembled by JLCPCB as discussed in Chapter 7. This reduced the price of the EPS module from 325 USD to about 125 USD. The full list of components is in Appendix C.2. However, this method means that the EPS does not come pre-programmed. Users must bootload software and code onto the SAMD21 MCU themselves using the J-Link and USB connectors. Though we successfully bootloaded

CircuitPython onto the SAMD21, we did not create software due to time and programming experience limitations. Future EPS software is further discussed in Section 9.3.1.

Once the EPS is assembled and programmed, it interfaces with the rest of the internal stack through its PQBH40 connector, a custom interface designed by BH-Dynamics. The `SAT_PWR_ENABLE` pin, controlled by the SAMD21 MCU, turns on (or off) the two 5V and 3.3V power rails, allowing the EPS to power cycle the entire spacecraft if necessary. The OBC1 board also pulls power and data from this connector, feeding it to the Teensy and the PQ10 bus.

For further details on the EPS module and the PQBH40 interface, refer to the DynOSSAT-EDU User's Guide [48].

4.3 Kill Switches

TigerCub has two kill switches as required by the PocketQube Standard (Requirement PQ-Mech-12). They are located on the bottom side of the sliding backplate PCB since they must contact the deployment rails as specified in Requirements PQ-Mech-13 and PQ-Mech-14 of the PocketQube Standard [3]:

> **PQ-Mech-13**: The PocketQube kill switches shall make contact with the deployer rail or with another PocketQube.
> **PQ-Mech-14**: Kill switches shall be located only on the z-axis. There are two different possible placement areas:
>
> a) In the lateral side of the satellite within 20 mm from the Z- faces and touching the deployment rails (lateral green area in Figure 4.2);
>
> b) Aligned with the sliding backplate in the Z- face and in contact with the PocketQube below the pusher plate (blue area in Figure 4.2).

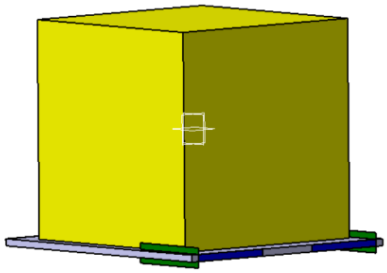


Figure 4.2: Possible kill switch locations (in green and blue). Referenced from Figure 6 in the PocketQube Standard [3].

We decided to place the kill switches so that they contact the deployment rails (i.e. the green area in Figure 4.2). This avoids overlapping the kill switches with the sheet metal frame mounting points (see Section 6.1.1 for details).

There are multiple types of limit switches that could be used as kill switches, such as whisker switches, level switches, roller switches, and plunger switches [49]. The roller switch is most suitable for our purposes since it allows for low-friction contact with the deployment rail during deployment. Alba Orbital also recommends this type of switch in their AlbaPod deployer's interface control document; they suggest using the Omron D2F-L2-A and D2F-L2-A1 switches (part of Omron's Ultra Subminiature Basic Switch line) [50]. The D2F-L2-A1 is a mirrored version of the D2F-L2-A, and one is used on each side of the TigerCub. A depiction of a similar switch, the D2F-L2, is shown in Figure 4.3.

Figure 4.3: Depiction of D2F-L2-A switch from datasheet [51].

These switches are both rated for 3 A of current and 1.75 N of operating force, suitable for our needs; they can also operate at temperatures from -40°C to $+85$°C [51]. The mechanical drawings for the D2F-L2-A are included in Appendix A. The D2F-L2-A and D2F-L2-A1 are single pole, double throw (SPDT) switches. That

is, they have one common ground (COM), one normally-open line (NO), and one normally-closed line (NC) as shown in Figure 4.4.

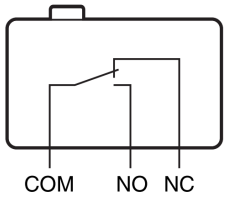


Figure 4.4: SPDT configuration for limit switches [51].

In the free configuration, COM and NC are connected, and when the roller hinge is pushed down, the COM and NO lines are connected. When the PocketQube is in the deployer, the hinge will be pushed down. Thus, the satellite should be powered off when the COM and NO lines are connected (i.e. the hinge is pushed down), and the satellite should power on when the COM and NC lines are connected after deployment (i.e. the hinge returns to its normal free position).

The schematic and PCB design for this circuit are shown in Figures 4.5 and 4.6.

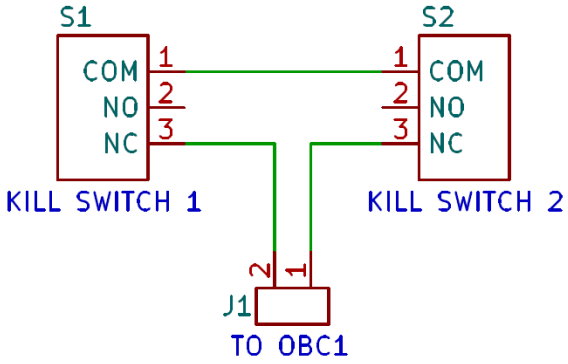


Figure 4.5: Sliding backplate schematic, including kill switches.

Connector J1 on the sliding backplate is a 2-pin Molex Picoblade connector. A wire harness runs between this connector and connector J3 on OBC1 (see Figure 3.3). After the satellite is deployed, the switch will change from the NO position to the NC position, completing the circuit that runs between the kill switches to OBC1 and powering the satellite on. The design of the sliding backplate, where the kill switches are mounted, is discussed in Section 6.1.2.

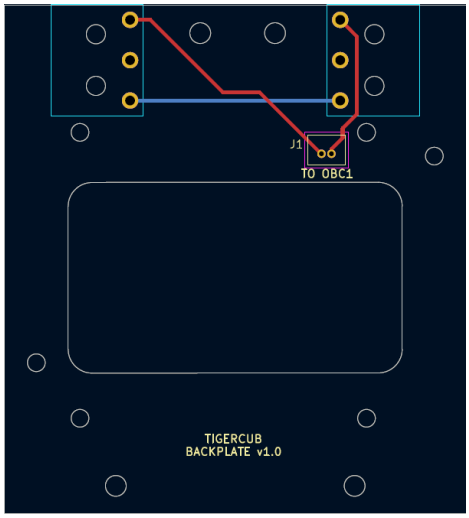
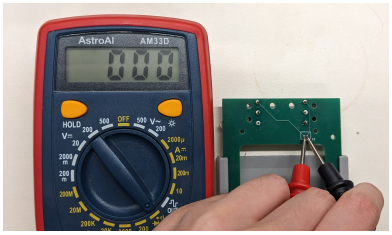
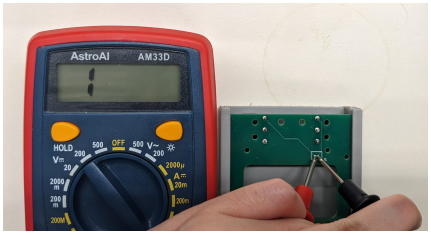


Figure 4.6: Sliding backplate PCB design, including kill switches.

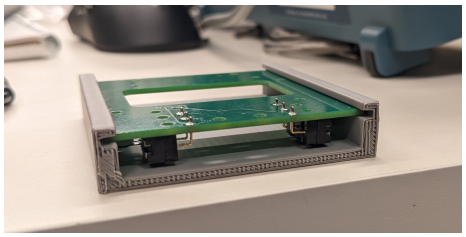
Once the sliding backplate PCB was manufactured and assembled, we tested for continuity in the Molex Picoblade connector while the backplate was not in a 3D-printed, to-scale deployment rail, and verified that there was no continuity when the backplate was in the deployment rail (i.e. when the switches were pressed down). Images of this verification process are shown in Figure 4.7.



(a) Continuity after release from rail.



(b) No continuity while in rail.



(c) Side view of sliding backplate in deployment rail.

Figure 4.7: Verification of kill switch logic via continuity test while sliding backplate is/is not in the deployment rail.

4.4 Solar Panels

The solar panels generate power for the satellite. TigerCub has four solar panels, each with two solar cells.

We chose to use fixed solar panels as opposed to deployable solar panels. Deployable solar panels are common in nanosatellites because the small surface area of these satellites limits the number of solar cells that can be placed; additionally, maneuverable solar panels can be turned to face the sun for maximum power generation. However, fixed solar panels are less complex since they do not require any mechanisms, and we currently do not have enough mass or volume margin to accommodate solar array drive assemblies. Ultimately, we decided to implement fixed panels because TigerCub is designed to be easily accessible to first-time PocketQube builders. Future developers may want to switch to using deployable panels for more power-intensive operations; suggestions are discussed in Section 9.3.5.

The solar panel design is also constrained by the DynOSSAT-EDU EPS architecture. On the EPS module, there are two solar panel connectors available, which combines the power generated by all the solar panels. Since there are only two solar panel connectors on the EPS and four solar panels on TigerCub, two panels route to each solar panel connector on the EPS as shown in Figure 4.8. The cells on each panel are connected in series, and the two panels are connected in parallel. The parallel connection between panels mitigates some of the effects of shadowing, in which some of the cells are in shadow while others are exposed to sunlight as the satellite is tumbling; if the panels were to be connected in series, then both panels would not be able to generate power if the first panel in the series is in shadow [52].

Note that there are two solar panel variants, PV1 and PV2, as described in Section 4.4.2.

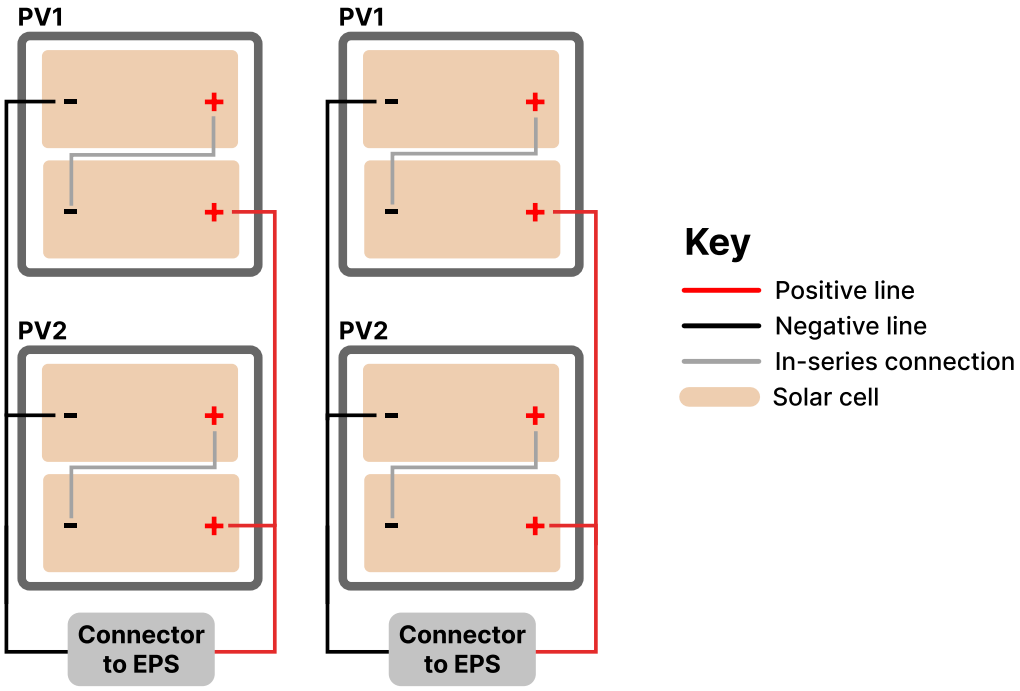


Figure 4.8: Solar panel block diagram.

4.4.1 Solar Cell Selection

The primary constraint on the solar cells is available surface area on the external surfaces of the satellite. TigerCub's four lateral surfaces are dedicated to solar panels; the top side houses the RockBLOCK patch antenna, and the bottom side contains a cutout window to offer an external view to the payload.

To size the solar cells, we maximized the power output of the cells given the available surface area for the cells on the satellite. This is a non-traditional method to size solar cells; *Space Mission Analysis and Design* (SMAD) suggests deriving the cell size from the power budget [52]. However, because of the initial decision to implement fixed solar panels, TigerCub already has very constrained power generation capabilities, so maximizing solar panel area (rather than designing to the power budget) was a known initial design goal.

We chose to compare cells in ANYSOLAR's IXOLAR line, which has previously been used in the TigerSats' CubeSat kit and has flight heritage on Cal Poly Pomona's PROVES CubeSat kit [53, 54]. These are lightweight cells that are ideal for mass-

constrained applications [55], and are much more affordable and accessible than typical space-grade gallium arsenite triple-junction solar cells.

DigiKey has over 20 different IXOLAR cells available. The maximum size of each solar panel is 50×50 mm (from Requirement C-01). We calculated the maximum power we could generate using each type of solar cell on a single solar panel, accounting for the number of cells we could fit on the solar panel. The five cells that can produce the most power are listed in Table 4.1 below.

Part Number	Cell Characteristics			Power Calculations	
	Current (mA)	Voltage (V)	Dimensions (mm)	Cells per Panel	Power (mW)
SM940K09L	36.6	5.02	44.5 × 21.5	2	367.5
KXOB101K08TF-TR	40.2	4.46	23 × 42	2	358.6
KXOB201K04TF-TR	78.7	2.23	23 × 42	2	351.0
SM141K05TF	58.6	2.79	22 × 35	2	327.0
SM101K07TF	39.3	3.91	22 × 35	2	307.3

Table 4.1: Solar cell comparison, listed in order of maximum power generated [55].

The cell that generates the most power given the area constraints is the ANYSOLAR SM940K09L; its characteristics are outlined in the Table 4.2 below.

Characteristic	Value
Typical voltage at MPP (V)	5.02
Typical current at MPP (mA)	36.6
Open circuit voltage (V)	6.22
Short circuit current (mA)	39.0
Maximum power at MPP (mW)	183.7
Open circuit voltage temp. coefficient (V/K)	−0.02
Short circuit voltage temp. coefficient (mA/K)	0.18

Table 4.2: ANYSOLAR SM940K09L solar cell electrical characteristics [56]. Values measured at standard condition from ground (1 sun = 1000 W/m^2, air mass = 1.5, temperature = 25°C).

The DynOSSAT-EDU module performs maximum power point tracking (MPPT)

to extract the maximum possible power out of the solar cells, as described in Section 4.2. Though ANYSOLAR does not provide current-voltage data for specific solar cells, the operating current vs. voltage curve for typical ANYSOLAR SolarMD cells is shown in Figure 4.9.

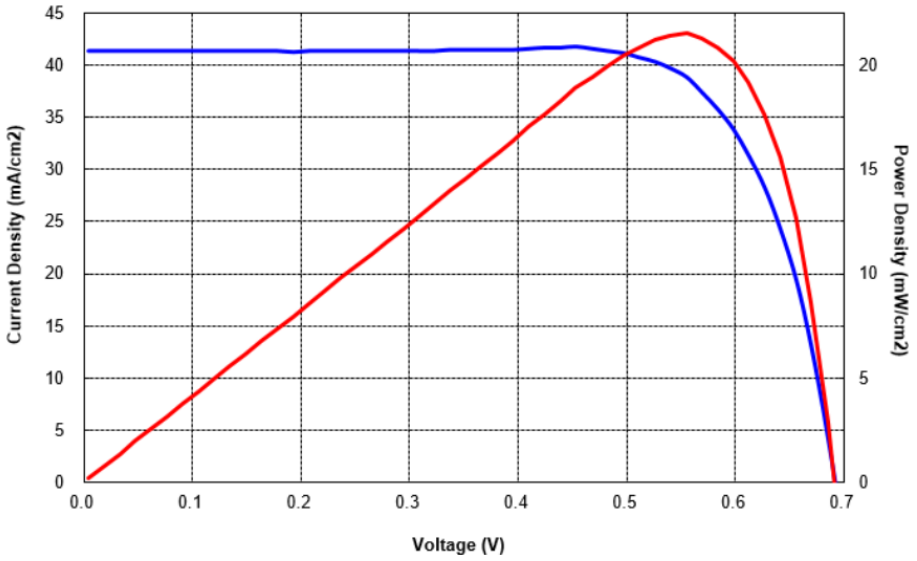


Figure 4.9: Typical operating current vs. voltage curve for ANYSOLAR SolarMD cells, referenced from the datasheet [56].

The values given by ANYSOLAR are measured at standard condition on the ground, where there is less solar flux than in space due to atmospheric attenuation. Additionally, manufacturing and assembly errors and temperature fluctuations during orbit will degrade the performance of these cells. We will account for these discrepancies from the maximum power point in the power budget in Section 4.6.

Note that this prototype does not include cover glass for the solar cells. Cover glass is typically used to protect solar cells from the effects of space radiation, and can sometimes even increase the performance of solar cells [57]. However, installing cover glass is non-trivial, especially for first-time PocketQube builders. We discuss the possibility of adding cover glass in Section 9.3.4.

4.4.2 Panel Design

As previously mentioned, each solar panel has two SM940K09L cells each. This effectively creates two strings of solar cells. Their combined power generation capabilities are discussed in Section 4.6.1. There are two design variants for the solar panels: PV1 and PV2, as shown in Figure 4.8.

PCB designs for the two panel variants are shown in Figure 4.10. Note that each panel contains a small Schottky diode. This serves as a blocking diode, which prevents electrical current from flowing back into the solar cells; blocking diodes are typically used in solar arrays with two or more parallel strings that might be shaded differently during satellite rotation [58].

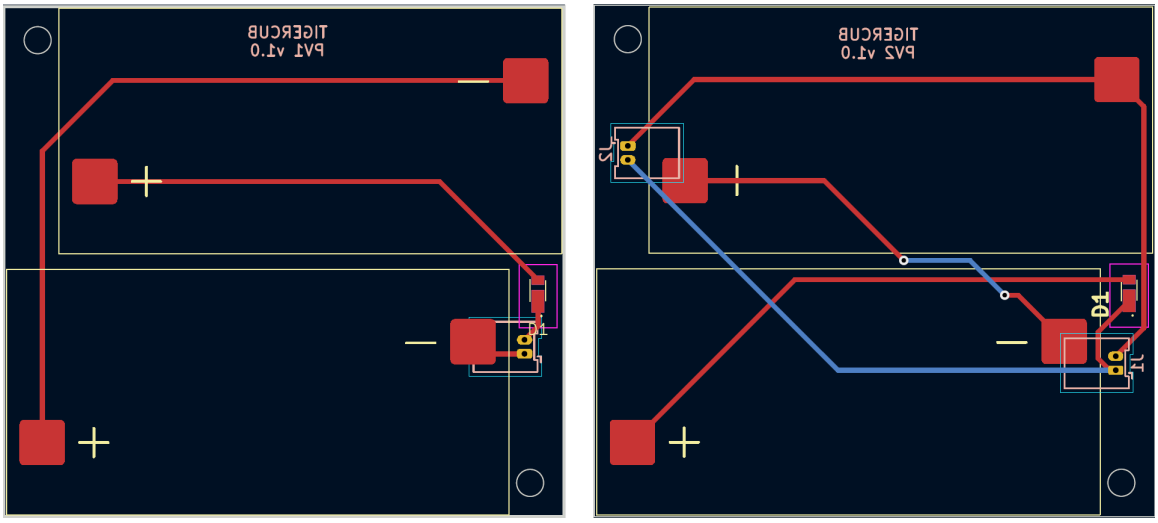


Figure 4.10: Solar panel PCB designs: PV1 (left) and PV2 (right).

4.5 Battery

The battery stores energy for the power system and is the satellite's power source during the eclipse period of the orbit when the solar cells cannot generate power. Mass and volume constraints limit us to a small battery, and the DynOSSAT-EDU recommends a 503040 Lithium-ion battery with an operating voltage of 3.7V and a capacity of 600 mAh [45]. Rechargeable lithium-ion batteries offer advantages in mass and volume savings as well as greater energy density as compared to more traditional

nickel cadmium and nickel hydrogen rechargeable batteries [59].

We opted for the slightly smaller 703525 lithium-ion battery which has the same operating voltage and capacity of 3.7V and 600 mAh, respectively [60]. For the purposes of this project, we sourced our battery from Amazon, but future developers may want to contact a battery supplie.

Our battery has the following characteristics:

Specification	**Value**
Model	702535
Material	Lithium polymer
Operating voltage	3.7V
Capacity	600 mAh
Maximum charge voltage	4.25V
Maximum charge current	300 mA
Dimensions	$7 \times 25 \times 35$ mm

Table 4.3: Characteristics of 3.7V 702535 600 mAh battery [60].

Space Mission Analysis and Design recommends using a battery-to-load transmission efficiency of $n = 0.9$ [59], so for our power budget analysis we will use a battery capacity of $600 \cdot 0.9 = 540$ mAh.

4.6 Power Budget Analysis

The solar cells and battery must power the internal stack (which includes the OBC, EPS, and payload) as well as the RockBLOCK. In the power budget, we calculate how much power the solar cells can provide and compare it to how much power the internal components consume. We also calculate the predicted mission lifetime based on how long it takes for the mission to become power-negative (consuming more power than it generates) due to the degradation of the solar cells and battery.

4.6.1 Power Generation

Since the solar panels are fixed, they cannot be oriented to face the sun and thus not all solar cells will be exposed to sunlight at any given time during the mission. As the satellite tumbles, some solar cells will be exposed to sunlight and some will not. To determine our power budget, we first calculate the orbit-averaged power that TigerCub can produce; that is, how much power is available during a typical orbit.

There is some literature that discusses how much power can be generated by a tumbling satellite with fixed solar panels. SMAD suggests an "array's reduction in output power per total surface area would be approximately...[a factor of] 4 for body-mounted cells on a cubic-shaped spacecraft that does not employ active tracking" [52]. Testing conducted in the TigerSats Lab of a CubeSat spinning in a tilted configuration with four fixed solar panels in a solar simulator suggested that the power generation was reduced by a factor of 2 as compared with a stationary (non-spinning) CubeSat in the same configuration [61]. Finally, Clyde Space suggests that a typical spinning four-panel CubeSat in low-earth orbit can generate the equivalent of 60% of one panel's power output, averaged over the entire orbit [62].

Considering these, we will assume, as a conservative estimate, that the orbit-averaged power for TigerCub is equal to 12.5% of the power generated by all cells. This is equivalent to half the power from one of the four solar panels (derived from assuming that only one panel faces the sun at a time and that the satellite is tumbling).

Each solar cell produces 5.02 V at 36.6 mA at the maximum power point, yielding a power output per cell of 183.7 mW. Eight cells yield approximately 1469.9 mW of power, and so the orbit-averaged power is $1469.9 \times 12.5\% = 183.7$ mW.

The solar cells are rated for conditions at ground, where the power density of the sun is approximately 1000 W/m^2 and the air mass value (representing how much air the radiation must pass through) is 1.5 [63]. However, in low earth orbit, the air mass is 0 (since the solar radiation does not pass through any air), and the solar power density is approximately 1360 W/m^2 [63]. To find the power generation capabilities of

our solar cells in LEO, we scale the power generated by the new solar power density:

$$\text{Orbit-averaged power} = 183.7 \text{ mW} \cdot \frac{1360 \text{ W/m}^2}{1000 \text{ W/m}^2} = 249.8 \text{ mW}.$$

Solar cells also face some degradation losses due to the space environment and manufacturing defects. These losses include:

- Assembly loss: Defects in the assembly process, such as cell mismatches when soldering solar cells onto panels, may cause a 1% loss in the short-circuit current value [64].

- Manufacturing loss: Defects in the manufacturing process may cause a 1% loss from the maximum power generated [64].

- MPPT: The maximum power point tracking IC, which is routed in series with the panels and consumes power itself, causes a loss of about 5% [65].

- Temperature: Changes in solar cell temperature cause the short-circuit current and open-circuit voltage to change at the rates described in Table 4.2. In Section 6.4, we determine the temperature range that the PocketQube encounters in 1) a low-earth orbit similar to that of the International Space Station (ISS) and 2) in a sun-synchronous "dawn-dusk" orbit that has no eclipse period (i.e. the satellite is always in sunlight). See Section 6.4 for additional description of these orbits. The temperature range in the ISS orbit is -25 to -15°C, and the range in the sun-synchronous orbit is 17 to 29°C. To find the temperature losses, we first extrapolated the current-voltage characteristics curve given the parameters in Table 4.2 for the SM940K09L solar cell at each of four temperatures: -25°C, -15°C, 17°C, and 29°C. These curves are shown in Appendix A.3. We then found the power generated at the maximum power point (given that the cells must operate at the same current) and compared the power values to those at the standard temperature of 25°C. For each of the two orbits, we took the greater loss as a conservative estimate.

The losses from each of these factors is shown in Table 4.4.

Source of degradation	Loss Percentage
Assembly	1%
Manufacturing	1%
MPPT	5%
Temperature (ISS orbit)	5%
Temperature (sun-synchronous orbit)	22%

Table 4.4: Solar array degradation losses.

For an ISS orbit, the total loss percentage is 28%, and for the sun-synchronous orbit, the total loss is 12%. This yields an orbit-averaged power of 180.0 mW for an ISS orbit and 219.9 mW for a sun-synchronous orbit. However, this value assumes the satellite is in a typical LEO orbit, in which a satellite spends typically a third to a half of the orbit in eclipse; in the sun-synchronous orbit we are considering, the satellite spends almost no time in eclipse. To account for this, we multiply the sun-synchronous orbit-averaged power by $\frac{4}{3}$ to obtain 293.2 mW.

4.6.2 Power Consumption

Typical Operations

Each electronic component consumes some power during typical operations. Table 4.5 describes the operating voltage and current during various modes for each component, each mode's duty cycle during typical operations, and the resulting power consumption for each mode, where the power consumption was calculated using the following equation:

$$\text{Power consumption} = \text{operating voltage} \cdot \text{operating current} \cdot \text{duty cycle}.$$

One duty cycle is one orbit of the satellite. Note that the duty cycle accounts for both sunlight and eclipse operations; that is, these components operate through an

entire orbit.

Component	Operating Mode	Voltage (V)	Current (mA)	Duty cycle (%)	Power (mW)
Teensy 4.0	Active	5	50	20	50
	Sleep	5	10	80	45
IMU board	Active	3.3	1.2	100	3.96
MicroSD board	Active	3.3	100	0	0
RockBLOCK	Charging	5	470	0	0
	Active	5	60	2	6
	Idle	5	50	2	5
	Sleep	5	0.5	95	2.4
DynOSSAT-EDU EPS	Active	3	10	100	30
Payload	–	–	–	–	40
				Total power	**177.36**

Table 4.5: Orbit-averaged power consumption of TigerCub components.

Justification of each component's power consumption is given below:

- **Teensy 4.0:**
  - *Active mode:* The Teensy 4.0 operates at 5V and normally operates at a clock rate of 600 MHz, consuming approximately 100 mA of current [28]. However, the Teensy can be clocked down to a lower clock rate, which also lowers the average current consumption; testing has shown that the Teensy operates at 44 mA when clocked down to 72 MHz, and at 53 mA when clocked down to 96 MHz [31]. We will operate the Teensy at 72 MHz, but will assume that the average current consumption is 50 mA as previously mentioned in Section 3.3.1. The Teensy only has to operate when reading and writing data from the payload and IMU and sending data through the RockBLOCK. We will assume that these operations will occur for about one-fifth of each orbit, so the duty cycle is 20%.
  - *Sleep mode:* The Teensy additionally has a low-power mode, called deep sleep mode, which can be implemented using the `Snooze` library. In this mode, the Teensy operates at 10 mA while preserving the real-time clock

function [33]. The Teensy operates in deep sleep mode for the remainder of the duty cycle, or about 80% of the duty cycle.

- **IMU:** The IMU breakout board contains the LSM6DS3TR-C accelerometer and gyroscope as well as the LIS3MDL magnetometer. The breakout board operates at 3.3V [35]. The LSM6DS3TR-C has a maximum current consumption of 0.90 mA [66], and the LIS3MDL has a maximum current consumption of 0.27 mA [67]. As an estimate, we will assume the IMU breakout board has an average current consumption of 1.2 mA. We will assume that the IMU is collecting data for 100% of the duty cycle.

- **MicroSD:** The MicroSD breakout board operates at 3.3V [36]. MicroSD read-/write operations can have a maximum current consumption of up to 100 mA, but since these read/write operations will occur infrequently (data will be transmitted directly to the RockBLOCK and/or stored in the Teensy), we assume an average duty cycle of close to 0% and thus the average power required is 0 mW.

- **RockBLOCK:** As outlined in Table 2.2, the current consumptions while charging, in active mode, when idle, and in sleep mode are 500 mA, 45-65 mA, 40-50 mA, and 0.073 mA, respectively. For budgeting purposes, we will assume that the average current consumptions are 500 mA, 60 mA, 50 mA, and 0.1 mA, respectively.

  - *Capacitor charging mode:* Charging only occurs once per mission, when the satellite must charge the RockBLOCK's supercapacitor at the beginning of the mission. Since charging does not occur during a typical orbit, its duty cycle is 0% and we do not account for charging in our nominal power consumption.

  - *Active mode:* The RockBLOCK is active when actively transmitting or executing a command. We will attempt a transmission once per orbit to relay data to the ground station. Rock7 estimates that this takes approx-

imately 1 minute [19], or as a conservative estimate, about 2% of a duty cycle.

  - *Idle mode:* The RockBLOCK is idle when it is powered on (i.e. not in sleep mode) but not actively executing commands. We assume that the RockBLOCK is idle for about 30 seconds before and after Active mode, or for about 2% of the duty cycle.

  - *Sleep mode:* The RockBLOCK sleep mode is a low-power mode to conserve energy. The RockBLOCK is in this mode for the remainder of the duty cycle, or about 96% of the duty cycle. Since we are waking up the RockBLOCK at least once per orbit to transmit data, we do not expect the RockBLOCK to lower its current consumption all the way to 0.1 mA (as advertised in the RockBLOCK specifications). Instead, we will assume the current is lowered to 0.5 mA.

- **DynOSSAT-EDU EPS:** Power on the DynOSSAT-EDU EPS is mainly consumed by the ATSAMD21 microcontroller unit, which typically consumes 3 mA but can consume up to 9 mA [68]. All other components on the EPS do not consume much power, so we will assume the overall current consumption is 10 mA at the 3V operating voltage of the SAMD21. The EPS is always in operation, so it has a duty cycle of 100%.

- **Payload:** We will allocate 40 mW of power during a typical duty cycle to the payload. This can result in a variety of operating voltage, current, and duty cycle combinations, such as 5V, 40 mA, and 20%, respectively. Developers can choose a payload that meets these power constraints. Note that if the duty cycle exceeds 20%, developers may have to increase the duty cycle of the Teensy's active mode if using the Teensy as the payload's microcontroller. The 40 mW orbit-averaged power includes all operating modes.

From Table 4.5 we find that the power consumption in a typical duty cycle is approximately 177.4 mW. Given our power generation capabilities as as calculated in

Section 4.6.1, our margin on power consumption in an ISS orbit is

$$\text{Power margin}_{\text{ISS}} = \frac{180.0 \text{ mW} - 177.4 \text{ mW}}{177.4 \text{ mW}} = 1.5\%$$

and the margin in a no-eclipse "dawn-dusk" sun-synchronous orbit is

$$\text{Power margin}_{\text{sun-synchronous}} = \frac{293.2 \text{ mW} - 177.4 \text{ mW}}{177.4 \text{ mW}} = 65.3\%$$

We have a relatively safe margin of 65% in the "dawn-dusk" orbit, but a very small (but still positive) margin in an ISS orbit. Note that this is the margin at the beginning of the mission. As the solar cells continue to degrade due to radiation and other effects (especially without cover glass), they will not be able to generate as much power, and eventually the power margin will become negative, which will happen sooner in the ISS orbit. This is one of the drivers of the mission lifetime, as described in Section 4.6.3.

Battery Depth of Discharge during Nominal Operations in ISS Orbit

The battery depth of discharge is the percentage of battery capacity used during a discharge period [69] and can be calculated using the following equation [64]:

$$\text{Depth of discharge} = \frac{\text{load power} \cdot \text{discharge time}}{\text{discharge voltage} \cdot \text{battery capacity}} \tag{4.6.1}$$

The discharge time is the time spent in eclipse. For a given orbital altitude, we can calculate the orbital period as well as the eclipse time [64]:

$$\text{Orbital period} = \frac{84.48 \cdot \left(\frac{\text{orbital altitude+Earth radius}}{\text{Earth radius}}\right)^{1.5}}{60}$$

$$\text{Fractional sun time} = \frac{0.5 + \frac{1}{\pi} \cdot \arcsin\sqrt{1 - \left(\frac{\text{Earth radius}}{\text{Earth radius+orbital altitude}}\right)^2}}{\sin(\text{inclination})}.$$

For an ISS orbit (which has an average orbital altitude of 400 km and an inclination of 51.6 degrees), we have an orbital period of 1.54 hours and a fractional sun

time of 0.64. Approximately 0.99 hours are spent in sunlight during each orbit, and approximately 0.55 hours are spent in eclipse.

Using our battery specifications, our depth of discharge is

$$\text{Depth of discharge} = \frac{177.4 \text{ mW} \cdot 0.55 \text{ hr}}{3.7\text{V} \cdot 540 \text{ mAh}} = 4.9\%.$$

This relatively low depth of discharge implies that our battery will have a longer cycle life [59].

Battery Recharge during Nominal Operations in ISS Orbit

The required battery recharge current is

$$\text{Battery recharge current} = \frac{\text{depth of discharge} \cdot \text{battery capacity}}{\text{time of recharge}} \tag{4.6.2}$$

which, for our power budget, is

$$\text{Battery recharge current} = \frac{4.9\% \cdot 600 \text{ mAh}}{0.99 \text{ hr}} = 26.2 \text{ mA}.$$

This is below our battery's maximum charge current of 300 mA, so the battery will be able to recover the discharged energy during the charging period. The battery charge margin is well above 100%.

Maximum Power Consumption

In Table 4.5, two components have a large current consumption but a low duty cycle: the MicroSD board and the RockBLOCK module in charging mode. Of these two, the RockBLOCK charge mode requires the most power (470 mA at 5V). We must ensure that the battery will not be depleted when the RockBLOCK powers up.

Charging the RockBLOCK's supercapacitor takes about 30 seconds and, as a conservative estimate, consumes 500 mA at the operating voltage of 5V. This consumes

$$500 \text{ mA} \cdot 5\text{V} \cdot \frac{30 \text{ sec}}{3600 \text{ sec/hr}} \div 3.7\text{V} = 7.1 \text{ mAh}.$$

This is much less than the battery capacity of 600 mAh. Since RockBLOCK charging will occur at the beginning of the mission after the battery fully charges, this will not deplete the battery.

Battery Usage in Sun-Synchronous Orbit

In the "dawn-dusk" sun-synchronous orbit, we assume there is no eclipse time and therefore very little battery discharge since the solar panels have adequate margin to power the satellite.

4.6.3 Mission Lifetime

All of our previous calculations used beginning-of-life (BOL) assumptions. However, components degrade over time in the space environment, mostly due to usage and the effects of space radiation. The two power components whose characteristics may change the most during the mission are the solar cells and the battery.

Solar cells degrade as a result of manufacturing inaccuracies, radiation, solar intensity, and temperature fluctuations [64]. For the monocrystalline cells that TigerCub uses, SMAD estimates that their performance degrades by up to 3.75% per year for a satellite in LEO [69]. We can calculate the number of years it would take for our solar cells to degrade enough to make the power margins negative in an ISS orbit (considering only the effects of solar cell degradation):

$$\begin{aligned} 177.4 &= 180(1 - 0.0375)^t \\ t &= \log_{0.9625} \tfrac{177.4}{180} \\ t &= 0.38 \text{ years} = 4.6 \text{ months.} \end{aligned}$$

The battery capacity decreases with cycle life for lithium-ion batteries [52]. Though the battery we used does not have end-of-life (EOL) specifications, similar batteries state that they are meant for 500-800 charge-discharge cycles [70, 71]. Since each charge-discharge cycle is one orbital period, 500 cycles would be equivalent to a mis-

sion lifetime of approximately

$$1.54 \text{ hrs} \cdot 500 \text{ cycles} = 770 \text{ hrs} \approx 32 \text{ days}.$$

However, it is likely that the battery will last for much longer than 500 cycles; even after 600 cycles, the battery capacity remains well above 90% of its BOL specification, as shown in Figure 4.11. Further testing must be conducted to determine when the battery capacity will drop below the required capacity for power positivity. At this time, we will restrict the mission lifetime in ISS orbit to 32 days to comply with the battery specification.

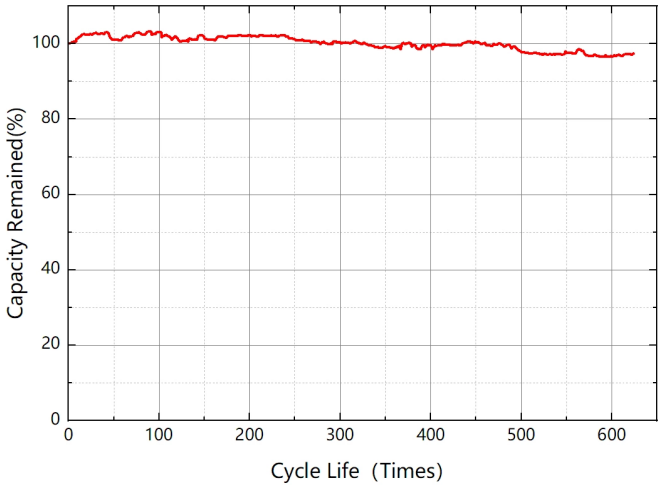


Figure 4.11: Battery capacity vs. cycle life for similar battery from UFine [70].

Completing a similar solar cell degradation calculation for the sun-synchronous orbit yields a (theoretical) mission lifetime of 13 years before the solar panels degrade enough to make the power margin negative. However, it is very unlikely for a PocketQube to actually survive for 13 years in orbit due to environmental factors such as aerodynamic drag and radiation (discussed in Section 6.5.2). Bouwmeester et al. suggest that a typical 1p PocketQube with body-mounted (i.e. fixed) solar panels survives for 2.5 years at a 500 km circular orbit similar to that of our proposed sun-synchronous orbit before it deorbits due to atmospheric drag [72]. For these reasons, we do not recommend planning for a mission lifetime of more than 1 year in the "dawn-dusk" sun-synchronous orbit.

Chapter 5

Payload

As a mission-agnostic PocketQube platform, TigerCub does not have a designed payload. Instead, it provides mechanical and electrical interfaces for potential payloads. This chapter describes these interfaces and provides some example payloads that could be flown on the TigerCub.

5.1 Mechanical Interface

Payloads will be located in the bottom section of the satellite, between OBC2 and the sliding backplate. If the payload requires a PCB, developers can install it as the bottom board in the internal stack; otherwise, that volume can be used for other payload components. The location of the dedicated payload PCB is shown in Figure 5.1. The payload volume parameters are outlined in Table 5.1.

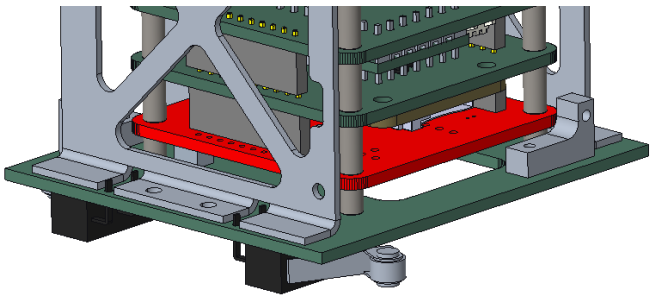


Figure 5.1: Payload PCB location highlighted in red.

Parameter	With PCB	Without PCB
Maximum dimensions	$36 \times 36 \times 4$ mm[1]	$36 \times 36 \times 8$ mm
Extended dimensions[2]	$48 \times 48 \times 4$ mm	$48 \times 48 \times 8$ mm
Maximum component height, top side	2 mm	–
Maximum component height, bottom side	4 mm	–

Table 5.1: Payload volume parameters.

If a payload PCB is used, it should conform to the PQ9 mechanical interface as described in Section 3.2 and use the PQ10 Samtec SQT-110-03-L-S connector. The dimensions of the PCB are shown in Figure 5.2. The four mounting holes allow the payload PCB to use the threaded rods of the internal stack.

Developers can additionally use the sliding backplate as an additional payload PCB (or the sole payload PCB). Developers who choose this method can can design a wire harness to connect to the PQ10 connector on OBC2 or use stacking connectors (such as the Samtec MTMM-110-02-F-S-080 along with two stacked SQT-110-03-L-S connectors); surface-mount solutions (for builders with more soldering experience) also exist.

Payloads can extend below the sliding backplate through a cutout in the backplate PCB. This relaxes the maximum volume parameters. Per Requirement PQ-Mech-08 in the PocketQube Standard, components should not extend more than 7 mm below the sliding backplate [3].

Developers can choose not to include the cutout if no components need it. If the cutout is included, developers should be aware of any potential thermal and radiation effects on their components.

[1] Assumes no components on bottom side of PCB.

[2] Payload components that require more volume than the minimum volume may use extra space outside of the internal stack. These payload components must not interfere with the threaded rod spacers or the kill switch components on the sliding backplate. Developers are encouraged to check for interferences using the CAD model.

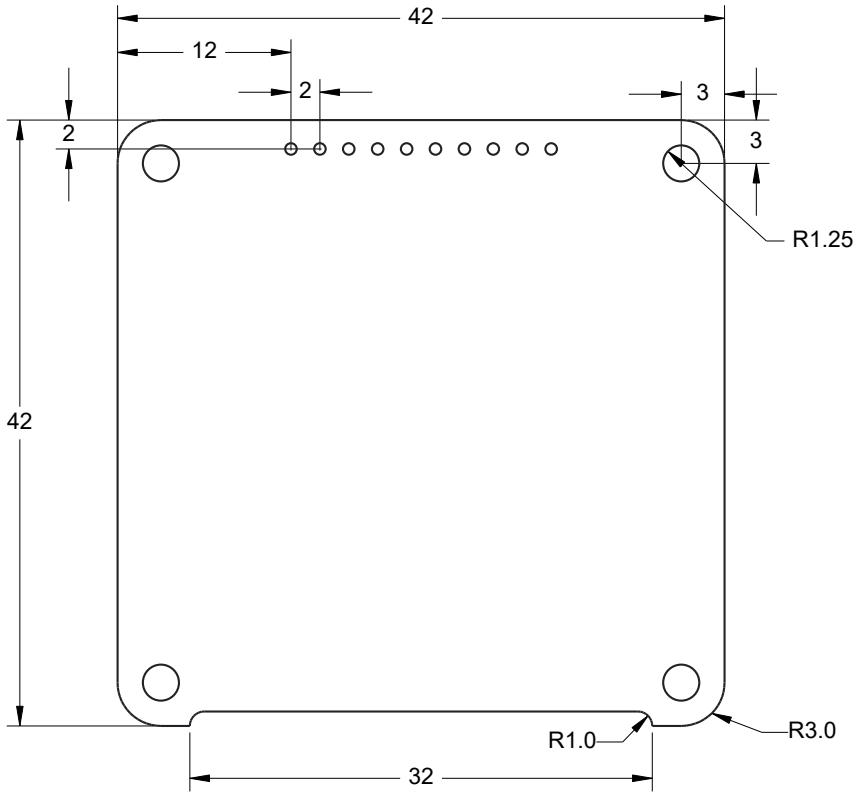


Figure 5.2: Payload PCB outline with PQ10 connector footprint. All dimensions in millimeters.

The mass of all payload components should not exceed 30 grams. A breakdown of TigerCub's mass budget is provided in Section 6.3. Finally, developers should ensure that their payload does not violate any other mechanical requirements in the PocketQube Standard.

5.2 Electrical Interface and Power Consumption

The payload board should use the PQ10 electrical interface as described in Section 3.4 to access power and the Teensy microcontroller. Developers can choose to use pins 7 and 8 as either an I2C line or a serial line. Developers also have an additional I2C line (on pins 2 and 3) and a serial line (on pins 9 and 10) available. Payloads can be reset using pin 1 of the PQ10 bus, which is connected to a digital pin on the

Teensy. Payloads can use either 5V power (pin 5) or 3V3 power (pin 6), but should operate at a 3.3V logic level to communicate with the Teensy 4.0.

Payloads are restricted to 40 mW of orbit-averaged power as described in the power budget in Section 4.6. At a 20% duty cycle for typical operations (and assuming the payload consumes 0 mW of power during the other 80%), a payload could consume 40 mA at 5V and 60 mA at 3.3V.

Payloads that require more power could consider adding an additional solar panel to the sliding backplate, which could contain at least two SM940K09L solar cells if developers remove the cutout window. This would increase the power generation capabilities by about 20% in a typical low-earth orbit.

5.3 Payload Environments

Most launch vehicle providers require payloads to undergo flight unit qualification tests to verify that the payload will survive the flight environment, which is a much harsher environment than that of orbit. In this section, we outline the qualification tests necessary to fly as a payload on a SpaceX Transporter launch, referencing the Falcon 9 Rideshare Payload User's Guide. Most PocketQubes in the near future will likely fly on a Falcon 9, but most launch vehicle providers publish their own guidelines for payload qualification testing.

Table 6-1 in the Falcon 9 Rideshare Payload User's Guide provides required and advised (optional) tests for payload assemblies, and SpaceX prefers that flight units are tested to the protoflight qualification levels [73]. SpaceX requires the following tests [73]:

- Quasi-static load: 1.25 times the limit load in each of the 3 axes. According to Table 4-1 in the Rideshare Payload User's Guide, the highest load a payload will experience is 17g, so payloads should be qualified to at least 21.25g.
- Random vibration: 3 db above the maximum predicted environment (MPE) spectrum (see Table 4-6 in the Falcon User's Guide) for 1 minute in each of 3

axes. Random vibration must be a standalone test.

- Sine sweep: Payloads must be subjected to a low-level sine sweep to ensure that all elastic natural frequencies are above 40 Hz.

- Pressure system and leak tests: Contact SpaceX for testing requirements. The majority of (non-propulsion) PocketQubes will not have any pressure vessels or components, in which case pressure testing is not necessary.

The following tests are recommended (but not required) by SpaceX [73]:

- Sine vibration: 1.25 times limit levels (see Table 4-2 in the Falcon User's Guide), four octave/minute sweep rate in each of 3 axes

- Acoustic: 3 dB above maximum environment spectrum (see Tables 4-3 and 4-4) for 1 minute

- Shock: 3 dB above maximum environment (see Table 4-5), 2 times in each of 3 orthogonal axes

- Thermal vacuum and thermal cycle: ± 5°C beyond the envelope of maximum temperature (see Table 4-11) and minimum range (-24 to 61°C) for 20 cycles

Of these recommended tests, acoustic and shock tests are not typically performed for nanosatellites (including CubeSats and PocketQubes) due to their small size. Sine vibration is a useful test to perform even as just a worksmanship test. Thermal vacuum cycling is also useful to ensure that COTS components on the spacecraft can survive the temperature range in the vacuum of space. More details on these tests from the Rideshare Payload User's Guide are provided in Appendix A.4.

Note that the minimum temperature range envelops the on-orbit temperature range that TigerCub is expected to see in either an ISS orbit or in a "dawn-dusk" sun-synchronous orbit (as calculated in Section 6.4), though the PocketQube (and its payload) will be powered off during launch and flight. Developers should ensure that their payload has an operating range of at least ± 10°C past the temperature ranges

described in Section 6.4; that is, 7 to 39°C for a "dawn-dusk" sun-synchronous orbit and −35 to 35°C for an ISS orbit.

In addition to these qualification tests, developers should perform a thermal vacuum bakeout on the flight unit to remove potential outgassing materials; this is required by the AlbaPod PocketQube deployer as well [50].

The TigerCub platform (without the payload) should also undergo these tests to qualify the platform for flight. This work is left to a future student, as described in Section 9.3.3.

5.4 Sample Payload Concepts

Previous PocketQube missions have included technology demonstrations of novel communications and propulsion technology (e.g. WREN and TRSI), measurements of the space environment (e.g. SMOG-1), earth observation (e.g. Alba Orbital's Unicorns), and more [74, 75, 76, 77]. Note that any payload that requires data to be sent back to Earth is allotted at most 296 bytes per transmission as calculated in Section 2.4.

5.4.1 Imaging

A popular PocketQube mission is taking images from space and sending image data back to Earth. Due to mass, volume, and power constraints, an imaging payload must use a very small camera and must have low-power capabilities. OmniVision's small imaging sensors, particularly the 2-5 megapixel sensors, are used in several PocketQube camera modules, such as in the PyCubed-Mini (which uses the OV5640) and the Pikocamera (which uses the OV2648) [5, 78].

For instance, the OV02C 1080p image sensor was originally made for laptop and tablet video cameras, but its small size, low power requirements, and high resolution make it a good candidate for an Earth imaging sensor [79]. Some of its specifications are listed in Table 5.2. Note that the sensor operates at a nominal power input of 2.8V; developers would need to include a buck converter to step-down the 3.3V rail from the PQ10 bus. The 115.4 mW power consumption during active mode means

that developers could operate the sensor for about 40% of the duty cycle, assuming that the other 60% is spent in standby mode. The 2-wire serial interface is compatible with I2C and can be used on the second I2C line on the PQ10 bus [80].

Along with the imaging sensor, the payload may include an imaging lens (some popular ones are from ArduCam), dedicated flash memory for image storage, a microcontroller to process images, and thermal control for the camera. Since the TigerCub has a limited data budget (as described in Section 2.4, the payload may have to packetize image data before sending it to ground through the RockBLOCK.

Specification	**Value**
Dimensions	$4.5 \times 3.2 \times 2.6$ mm
Power input	2.8V nominal
Power consumption	115.4 mW (active), 0.25 mA (standby)
Data output	8-bit in 2-wire serial interface
Temperature range	-30°C to $+85$°C
Resolution	2MP at 60 fps
Pixel size	1.116×1.116 μm

Table 5.2: OmniVision OV02C imaging sensor specifications [79].

5.4.2 Spectroscopy

Another popular payload for CubeSats and PocketQubes is spectrometers, which are devices that measure the effect of some portion of the electromagnetic spectrum (typically light) on some object. For instance, the SMOG-1 PocketQube, built by students from the Budapest University of Technology and Economics and launched in 2021, measures electromagnetic waste from Earth in the 430-860 MHz range (typically used by TV stations) [81].

A potential spectrometer for a PocketQube mission is ams OSRAM's AS7263, a digital 6-channel spectrometer that detects near IR (NIR) light wavelengths. The AS7263 integrates well with PocketQube; it operates at 3.3V and 5 mA, uses the I2C

communication protocol, has dimensions of $4.5 \times 4.7 \times 2.5$ mm, and has a operating temperature range of -40°C to 85°C [82]. These specifications all conform to the TigerCub interface. Payload integrators could design a breakout board for the AS7263 sensor as the payload board on the TigerCub, perhaps modeled after SparkFun's breakout board for this sensor [83].

5.4.3 Novel Technology

Since PocketQubes are small, inexpensive, and fast to develop, they are an ideal platform for on-orbit technology demonstrations. Advanced developers could implement novel technology demonstrations with TigerCub. Up-and-coming smallsat concepts include technology to support satellite constellations, amateur radio demonstrations, and RF sensing [84].

Chapter 6

Structure, Thermal, and Materials

The external structure encapsulates all subsystems and serves to protect the spacecraft from the space environment. This chapter discusses the design and analysis of the aluminum frame and the PCB sliding backplate, as well as the mass budget, thermal analysis and control, and material outgassing considerations.

6.1 Structural Design

6.1.1 Frame

The frame provides structural support for the solar panels and radio module. It also provides shielding for the internal stack.

Small satellite frames are typically machined aluminum. For instance, the CubeSat Development Specification specifies that CubeSat structures should be constructed of an aluminum alloy [85]. Additional constraints on CubeSat structural design means that CNC machining is the usually the most cost-effective way to manufacture CubeSat structures.

On the other hand, the PocketQube Specification does not have any specific requirements relating to structural design or choice of materials. This flexibility allows us to choose an alternative manufacturing method: sheet metal fabrication. While CNC machining requires a specialized machinist and expensive machines that can

create tight tolerances and complex features, sheet metal fabrication bends sheets of malleable metal into shape and is generally much less expensive than CNC machining [86]. Since TigerCub's structure does not need complex features or tight tolerances, sheet metal fabrication is an ideal choice to reduce manufacturing costs and promote rapid prototyping.

We chose to use Aluminum 5052 for the sheet metal frame. Aluminum 5052 is one of the most common aluminum alloys for sheet metal applications due to its relatively high tensile strength, good workability and formability, low density, and high availability in sheet-metal fabrication shops [87].

Though sheet-metal parts are much less expensive than CNC-machined parts, one limitation of sheet-metal parts is that tapped holes cannot be formed since the material is thin. This forces us to use bolts and nuts for mounting the radio module and solar panels as well as attaching the frame to the sliding backplate. This does not pose a problem for the satellite layout, but does make assembly more tedious. The assembly process is detailed in Section 7.4.

A model of the sheet-metal frame is shown in Figure 6.1 below.

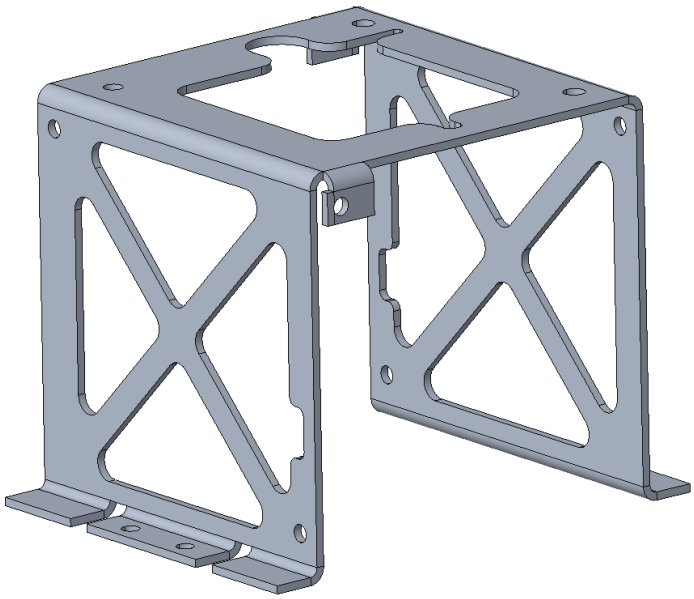
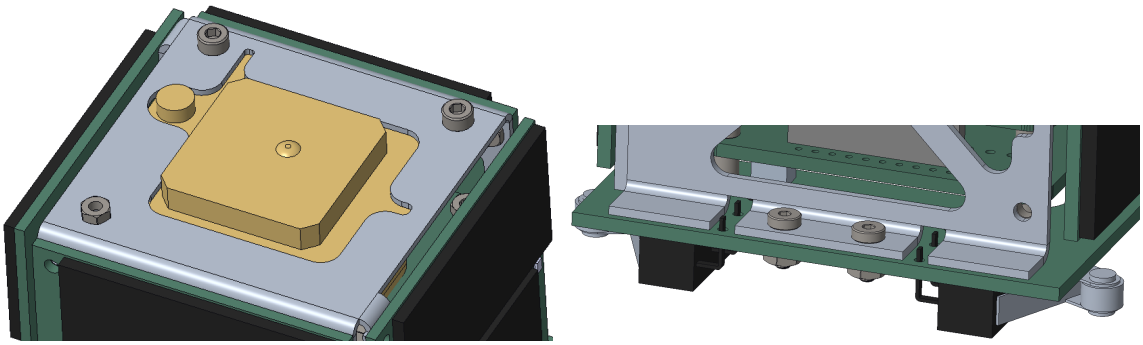


Figure 6.1: CAD model of sheet metal frame.

This model was created with Creo's sheet metal part design software. Creo allows

users to set a material thickness and bend radius for their sheet metal parts. The frame has a thickness of 1 mm and an external bend radius of 2 mm; most sheet metal shops recommend that the bend radius is at least the thickness of the material [88].

The frame has several cutouts to avoid interference with other components, shown in Figure 6.2. Besides these cutouts, the frame also has triangular cutouts on the front and back sides for light-weighting. These cutouts also allow solar panel harnesses to connect to the EPS board.

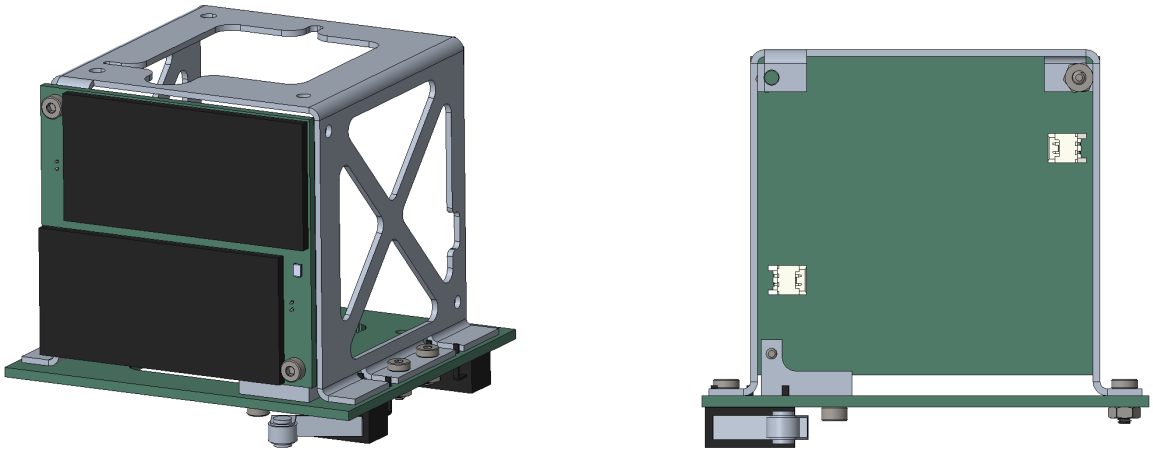


(a) Cutout for RockBLOCK module. **(b)** Cutout for kill switch pins.

Figure 6.2: Frame cutouts.

The RockBLOCK module is attached with two M2.5 fasteners and one M2 fastener as shown in Figure 6.2a above.

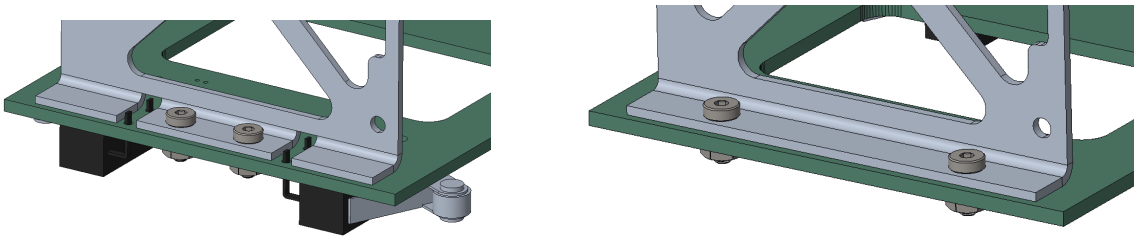
Two solar panels mount directly to the sheet-metal frame with fasteners. The other two solar panels are mounted to tabs on the sheet-metal frame and brackets that are fastened to the sliding backplate. The mounting locations for these two solar panels are shown in Figure 6.3. The small L-shaped mounting bracket is a CNC-machined part since it includes tapped holes. Dimensioned drawings of the frame and L-bracket are included in Appendix A.1. These L-brackets are Aluminum 6061, another common aluminum alloy for spacecraft.



(a) Trimetric view. **(b)** View from behind panel.

Figure 6.3: Mounting locations for right solar panel.

The frame fastens to the sliding backplate with four sets of bolts and nuts. The mounting points are shown in Figure 6.4.



(a) Front view. **(b)** Back view.

Figure 6.4: Mounting locations for frame to sliding backplate.

6.1.2 Sliding Backplate

The sliding backplate serves to constrain the PocketQube within its deployer and provide attachment points for internal components and the frame. Its dimensions are governed by the PocketQube Standard (see Requirement C-02 in Table 1.1); it must have dimensions of $58 \times 64 \times 1.6$ mm. In addition, Requirement PQ-Mech-07 in the PocketQube Standard states that "the rail clamping dimensions shall be 2 mm on each side of the sliding backplate," so we must leave clearance for the rail to pass through, as seen in Figure 6.5 [3]. Dimensioned drawings of the sliding backplate and a model of the deployment rail are included in Appendix A.1.

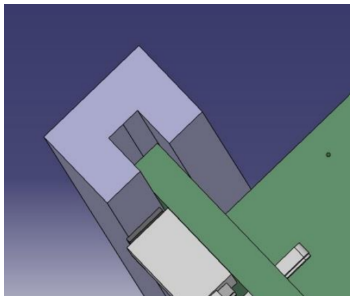


Figure 6.5: Deployment rail position with respect to sliding backplate. The rail constrains the sliding backplate in two axes. Referenced from Figure 4 in the PocketQube Standard [3].

The frame attaches to the sliding backplate via four M2 fasteners (shown in Figure 6.4), and the internal stack has four M2 attachment points. The L-brackets each have one M2 attachment point on the backplate. The backplate PCB also contains the kill switch circuit as described in Section 4.3. No components are located in the 2 mm deployment rail zone. A CAD model of the sliding backplate is shown in Figure 6.6.

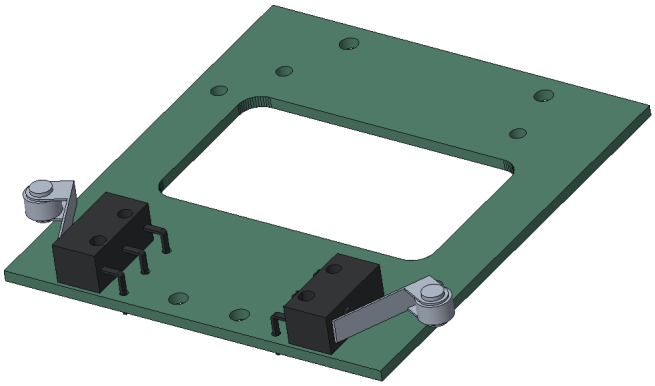


Figure 6.6: CAD model of sliding backplate with kill switches.

6.1.3 Layout

Internal components include the internal stack of PCBAs (the EPS, OBC1, OBC2, and payload boards) and the battery. The internal stack is constrained to the sliding backplate via a set of four 35 mm M2 screws through the four mounting holes on each board and stainless steel spacers between boards. The internal stack is slightly offset from the geometric center of the satellite to accommodate the battery and kill switches, as shown in Figure 6.7 below.

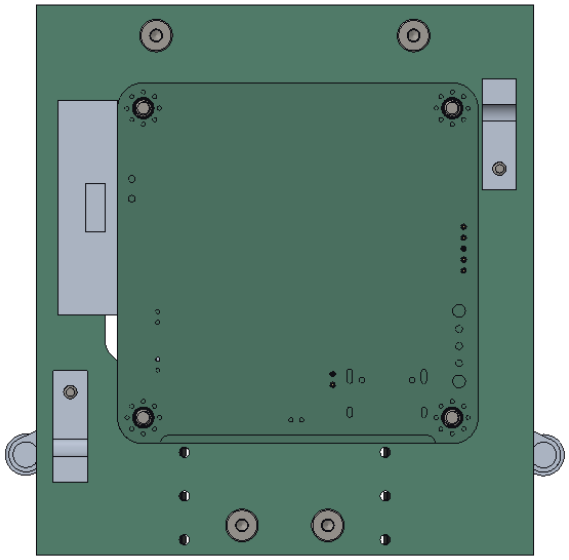


Figure 6.7: Top view of satellite showing internal stack and battery.

For consistency, we used M2 screws throughout the satellite, with the exception of two M2.5 screws used to mount the RockBLOCK module to the frame. All screws except the 35 mm M2 screws for the internal stack are 5 mm in length. The four screws fastening the frame to the sliding backplate are low-profile screws and the two nuts at the top-left corners of the solar panels on the front and back of the satellite are thin nuts to prevent interference between components. These fasteners are widely available through suppliers such as McMaster-Carr.

6.2 Structural Analysis

TigerCub must withstand intense the intense load, vibration, shock, and acoustic environments of launch and flight. To validate that TigerCub can survive these environments, we conducted load and modal analyses in Creo Simulate and compared the results to published information about launch vehicle flight environments.

6.2.1 Flight Environments

As a small satellite, TigerCub will most likely launch as a secondary payload on a launch vehicle such as SpaceX's Falcon 9, Firefly Aerospace's Alpha, or Rocket Lab's

Electron. Of these, the Falcon 9 has the harshest flight environments. The maximum load factors are shown in Table 6.1.

Load Factor	Falcon 9	Alpha	Electron
Axial (g)	10	8	7.5
Lateral (g)	17	2.4	2

Table 6.1: Maximum quasi-static load factors for the Falcon 9, Alpha, and Electron launch vehicles [73, 89, 90].

Additionally, minimum payload fundamental frequencies as required by each launch vehicle provider are listed in Table 6.2.

	Falcon 9	Alpha	Electron
Min. fundamental frequency (Hz)	40	25	–[1]

Table 6.2: Minimum payload fundamental frequencies for the Falcon 9, Alpha, and Electron launch vehicles [73, 89, 90].

Note that we will not simulate the effects of the vibration, shock, and acoustic environments. Instead, developers should test TigerCub under these environments if required by the launch provider as described in Section 9.3.3.

6.2.2 Simulation Setup

Before running any analyses, we simplified the CAD model to reduce FEA mesh time and avoid meshing errors. We removed electronic components from the PCBAs (such as solar cells, connectors, passive components, and microcontrollers) and increased the density of the corresponding PCBs so that the mass of each PCB was equal to the mass of the actual PCBA. We also removed holes from the PCBs to simplify the meshing process.

[1]Rocket Lab does not provide a required minimum fundamental frequency for payloads [90]. We will assume it is not higher than that of Falcon 9.

Creo Simulate automatically bonds connected surfaces during meshing, which also allows us to remove fasteners from the simplified model. Though this method is not as accurate as simulating bolted joint connections, it is sufficient for the analysis of a simple satellite.

Each component was assigned to one of four materials: Aluminum 5052, Aluminum 6061, 18-8 Stainless Steel, or FR-4. For simplification, we assigned all PCBAs and the radio module to the FR-4 (circuit board) material, since the FR-4 PCB takes most of the load. The simplified model for analysis is shown in Figure 6.8 below, with Aluminium 5052 and 6061 components in gray, stainless steel components in black, and FR-4 components in green. One of the solar panels is removed in the image to show the internal stack.

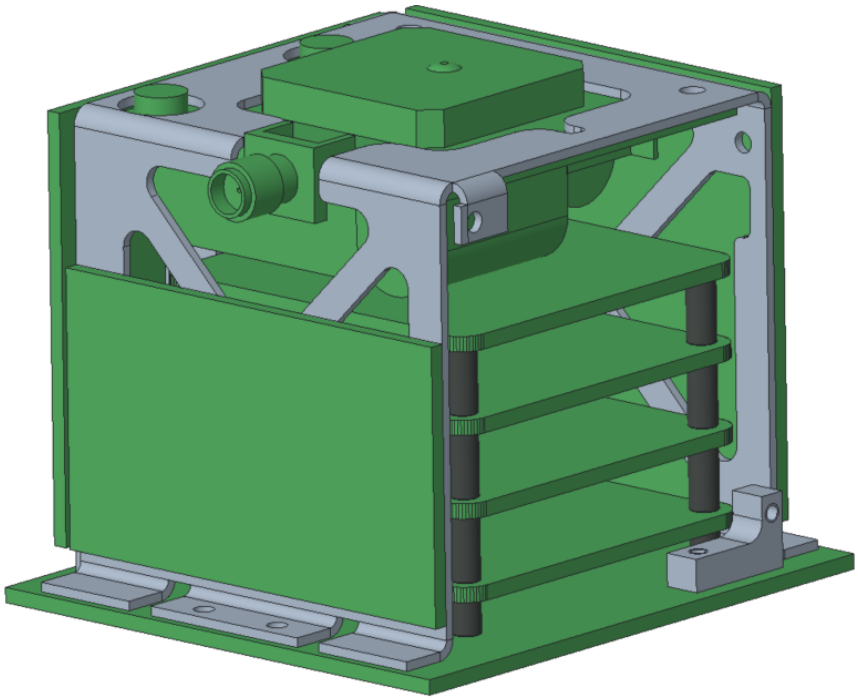


Figure 6.8: Simplified CAD model for analysis. Note that structural analysis was completed before the last design iteration, in which some structurally-minor changes were made.

The material properties for Aluminum 5052, Aluminum 6061, 18-8 Stainless Steel, and FR-4 are described in Table 6.3 below.

Property	Al-5052	Al-6061	18-8 Stainless Steel	FR-4
Density (g/cm^3)	2.68	2.71	7.74	1.85
Poisson's ratio	0.33	0.33	0.29	0.12
Young's modulus (GPa)	70.3	68.9	276	21
Yield strength (MPa)	193	276	215	241
Ultimate strength (MPa)	228	310	505	–[2]

Table 6.3: Material properties used for analysis [87, 91, 92, 93].

Note that FR-4 is actually an anisotropic material, but we will treat it as isotropic to simplify analysis. As a conservative measure, we will use the lower (cross-wise) tensile strength in our calculations.

TigerCub will be constrained within its deployer via the deployment rail, which clamps the sliding backplate as mentioned in Section 6.1.2 and as seen in Figure 6.5. To simulate this, we created planar constraints where the rail clamps the sliding backplate. We also added a planar constraint on the back face of the sliding backplate where TigerCub would be constrained by another PocketQube in the deployer. The constraints are shown in Figure 6.9 below.

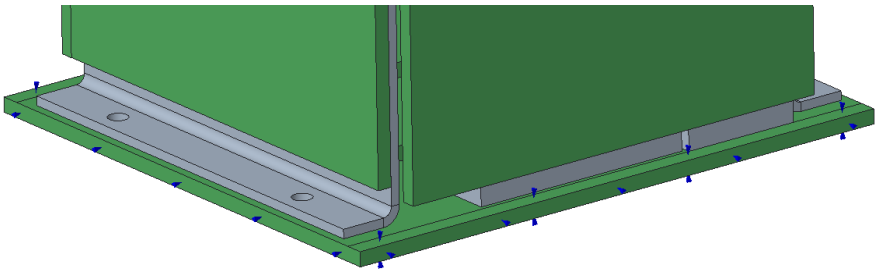


Figure 6.9: Deployment rail constraints (blue arrows) for analyses.

Finally, note that the spacecraft axes are defined as in Figure 6.10. We take the positive Z-axis (i.e. the front of the spacecraft) to be pointing towards the kill switches.

[2]We were unable to find a value for the ultimate strength of FR-4.

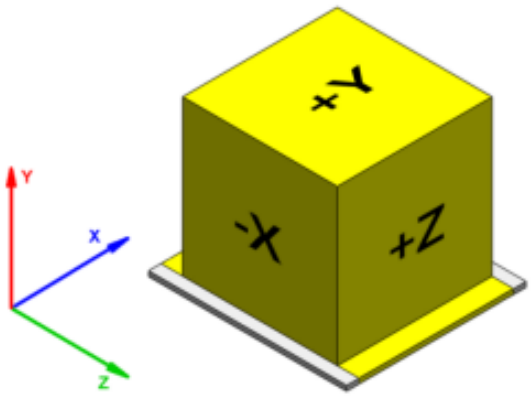


Figure 6.10: Definition of TigerCub axes. Referenced from Figure 3 in the PocketQube Standard [3].

6.2.3 Static Load Analysis

We conducted a static load analysis using the largest load factor from Table 6.1 in the three axial directions (X, Y, and Z). Each static load analysis was computed using multi-pass adaptive convergence with a convergence limit of 10% and maximum polynomial order of 6. Using the previously-defined constraints and a gravitational load of 17 g, we found the following:

Result	X-direction	Y-direction	Z-direction
Max. von Mises Stress (MPa)	12	16	28
Max. displacement (mm)	0.06	0.14	0.15

Table 6.4: Static load analysis results for a gravitational load of 17 g.

Note that the maximum stress and displacement were not taken directly from the "maximum" values from the static load analysis. For instance, the "maximum" stress in the X-direction was actually 67.4 MPa, but taking a closer look at the results shows that this stress occurred only at a small portion of the edge between the frame and the sliding backplate, as seen in Figure 6.11 below. This singularity indicates a location, most often between two elements of dissimilar materials or geometries, where the analysis diverged; however, singularities do not accurately represent what

the real stress value would be.

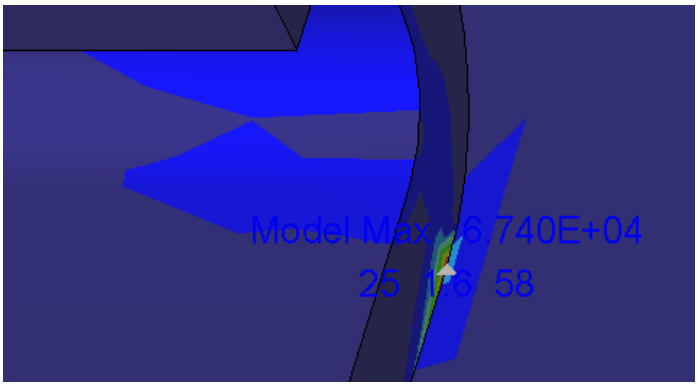


Figure 6.11: Singularity in static load analysis.

Instead, we consider the satellite as whole and look at regions where the average stress is higher, as seen in the circled regions in Figure 6.12. As we might expect, these mostly occur in regions where the sheet-metal frame bends.

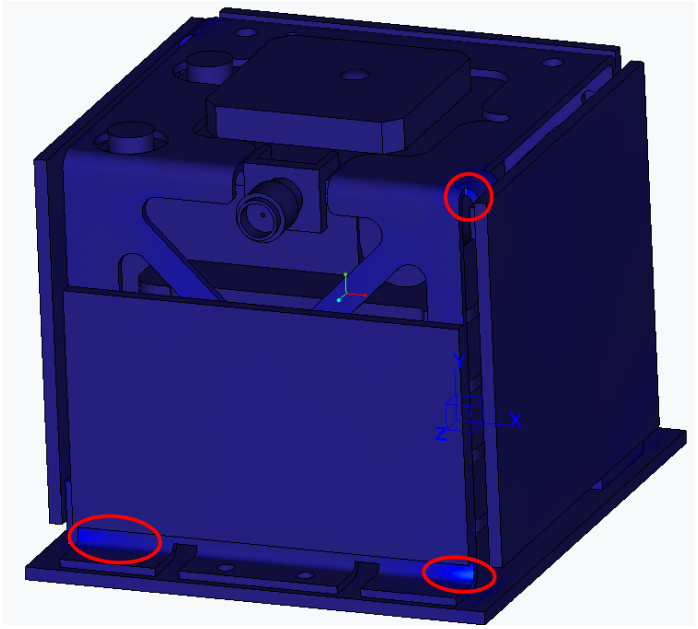


Figure 6.12: High-stress regions from static load analysis.

Inspecting the stress values at these points suggests that the maximum stress value should be about 12 MPa.

From Table 6.4, the highest von Mises stress in any direction is 28 MPa. This

gives the following yield stress margins, calculated using the equation

$$\text{Stress margin} = \frac{\text{strength of material} - \text{max. von Mises stress}}{\text{max. von Mises stress} \cdot \text{safety factor}} \tag{6.2.1}$$

The NASA General Environmental Verification Standard (GEVS) recommends in Section 2.4.1.3 that spacecraft use a yield safety factor of 2 and ultimate safety factor of 2.6 during structural load analysis to account for inadvertently-high loads during flight and manufacturing defects [94].

Material	**Yield Stress Margin**	**Ultimate Stress Margin**
Aluminum 5052	2.94	2.74
Aluminum 6061	4.43	3.87
18-8 Stainless Steel	3.34	6.55
FR-4	3.81	–

Table 6.5: Stress margins for static load analysis.

These stress margins are all positive, so we have verified through analysis that TigerCub will survive the static load flight environment (Requirement E-03). Launch providers may require further testing, described in Section 9.3.3.

6.2.4 Modal Analysis

To verify that we meet the minimum fundamental frequencies as listed in Table 6.2, we conducted a modal analysis of the satellite. Using the same constraints as before (shown in Figure 6.9), we conducted a multi-pass adaptive modal analysis to search for the first four fundamental frequencies. Along with the frequencies, we also found the effective masses, which describe how much mass of the satellite is "participating" in each fundamental mode; if the effective mass is relatively high, the corresponding mode is easily excited [95]. These are the modes that we will use in Miles' equation in the next section.

Table 6.6 shows the first four modes found from the Creo modal analysis, along with the highest effective mass value and its corresponding direction.

Frequency (Hz)	Highest Effective Mass (%)	Direction of Effective Mass
216	12.5	translation in Y
291	1.3	translation in Y
310	0.0	N/A
564	6.3	translation in Z

Table 6.6: Fundamental modes from modal analysis.

We see that 216 Hz is the first mode (i.e. fundamental frequency) of the satellite. This is over 400% higher than any of the minimum fundamental frequencies listed in Table 6.2, so we have verified that we can meet this requirement from the launch providers (Requirement E-03 from Table 1.1). In addition, no fundamental frequencies will be excited during sine vibration testing, which typically tests payloads to frequencies up to 100 Hz.

6.2.5 Random Vibration Analysis Using Miles' Equation

Spacecraft qualification for random vibration is typically done through physical testing on shaker tables. In addition to these tests (discussed in Section 9.3.3), we can use Miles' equation to approximate the effects of random vibration during launch on the spacecraft. Miles' equation, shown below, converts an acceleration spectral density (which describes the random vibration) and natural frequency to a quasi-static load.

$$G_{\text{RMS}} = 3\sqrt{\frac{\pi}{2} f_n Q[\text{ASD}_{\text{input}}]} \tag{6.2.2}$$

In this equation, G_{RMS} is the equivalent quasi-static load in terms of g, f_n is the fundamental frequency in Hz, Q is the transmission factor (typically taken to be the square root of the natural frequency), and $\text{ASD}_{\text{input}}$ is the acceleration spectral density in g^2/Hz, given by the launch provider [96, 97]. The additional factor of 3 represents the conservative "3-sigma" estimation of the quasi-static load [96].

In Table 6.6, we found that the fundamental frequency of 216 Hz has the highest effective mass factor. We will use this fundamental frequency in the Miles' equation. At 216 Hz, the random vibration ASD for Falcon 9 is 0.015 g^2/Hz [73]. This is the highest ASD at this frequency for any of the three vehicles.

The equivalent quasi-static load is

$$G_{\text{RMS}} = 3\sqrt{\frac{\pi}{2} \cdot 216\sqrt{216} \cdot 0.015} = 25.9 \text{ g}.$$

To verify that TigerCub can survive this quasi-static load, we conducted another set of static load analyses in the three axial directions using a gravitational load of 26 g. The results are shown below.

Result	**X-direction**	**Y-direction**	**Z-direction**
Max. von Mises Stress (MPa)	16	20	40
Max. displacement (mm)	0.09	0.22	0.22

Table 6.7: Static load analysis results for gravitational load of 26 g.

The maximum stress is 40 MPa, which gives a yield stress margin of 1.9 for the Aluminum 5052 material. These results suggest that the satellite will survive the random vibration environment, but random vibration testing should still be conducted before launch.

6.3 Mass Budget

The maximum mass of TigerCub is 250 grams (see Requirement C-04 in Table 1.1). To verify that we meet this mass requirement, we weighed each component with a digital scale. Each PCBA was weighed with all its electronic components soldered on. The mass of each component is shown in Table 6.8.

Component	Unit Mass (g)	Quantity	Total Mass (g)
RockBLOCK	39.2	1	39.2
OBC1	12.3	1	12.3
OBC2	12.3	1	12.3
EPS board	9.5	1	9.5
Solar panel	13.7	4	54.8
Battery	11.7	1	11.7
Frame	10.7	1	10.7
L-bracket	0.6	2	1.2
Sliding backplate	10.1	1	10.1
9/32" spacer	2.1	12	25.2
5/32" spacer	1.2	4	4.8
M2 SHCS [3] (5 mm)	0.2	12	2.4
M2.5 SHCS	0.4	2	0.4
M2 nut	0.1	14	1.4
M2.5 nut	0.2	2	0.4
M2 SHCS (35 mm)	0.8	4	3.2
Wire harnesses	–	–	10
Payload	–	–	30
		Total mass	239.6
		Total allowable mass	250
		Mass margin	4.34%

Table 6.8: Mass budget.

The payload is allocated 30 grams of mass, as discussed in Chapter 5. The positive mass margin indicates that we have satisfied Requirement C-04.

The center of mass, as reported by the CAD model in Creo, is $(-3, 1.1, 0.3)$ mm, with respect to the geometric centroid of the satellite. The center of mass is less than 1 cm from the geometric centroid, satisfying Requirement C-05.

6.4 Thermal Analysis and Design

The thermal control system maintains the satellite's temperature within a desired range, typically the operating range(s) of the satellite components. For this analysis,

[3]SHCS is the abbreviation for "socket head cap screw."

we determine if the baseline configuration of the spacecraft (without any additional thermal control elements) is sufficient to maintain the spacecraft in the desired temperature range in a typical low-earth orbit similar to that of the ISS and in a no-eclipse sun-synchronous "dawn-dusk" orbit. This sun-synchronous orbit, in which the satellite is constantly at dawn or dusk, is typically used by picosatellites (such as Alba Orbital's Unicorn PocketQubes) since the satellites are never in full eclipse and can therefore generate more power (and, for our purposes, stay warm) [98, 99]. The orbital parameters for both orbits are defined in Table 6.9. Since we find that the spacecraft tends to run cold in the ISS orbit, we also discuss some thermal control configurations to increase the steady-state temperature.

Orbit	**Semi-Major Axis (km)**	**Inclination (°)**	**Longitude of Ascending Node (°)**
ISS	6771	51.6	0
Sun-synchronous	6871	97.5	90

Table 6.9: Orbital parameters for ISS orbit and no-eclipse sun-synchronous orbit [98, 100].

All other orbital parameters are defined to be 0. Note that the longitude of the ascending node of the sun-synchronous orbit was chosen such that the spacecraft is in a dawn-dusk orbit. Developers who wish to use such an orbit should confirm their orbital parameters before launch.

For simplicity, we will assume that the spacecraft is isothermal; that is, the entire spacecraft is at one temperature. We also assume the spacecraft is a single node, so the entire spacecraft has the same thermal properties. The following sections discuss each step of the thermal analysis.

6.4.1 Operating Temperature Range

We first determine the operating range for the (assumed) isothermal spacecraft body. Table 6.10 shows operating temperature ranges for TigerCub's electronic components. Of these, the narrowest range is that of the battery, from -20 to $+60$°C. We will attempt to design the entire spacecraft to operate in this temperature range.

While the battery's stated operating range is between −20 to +60°C, typically this is the *discharge* operating range for lithium-ion batteries; to prevent damage to the batteries, they should be *charged* at a temperature above 0°C [101]. Battery charging occurs during sunlight periods in the orbit when the solar cells are generating power to recharge the battery.

Component	**Operating Range (°C)**
RockBLOCK	−40 to +85
Teensy 4.0	–[4]
LSM6DS3TR-C accel/gyro	−40 to +85
LIS3MDL magnetometer	−40 to +85
SAMD21 EPS MCU	−40 to +85
Solar cells	−40 to +90
Battery	−20 to +60

Table 6.10: Operating temperature ranges for TigerCub components [56, 66, 67, 70, 102].

6.4.2 Thermal Environment

As the satellite orbits, its primary external heat source is the Sun, which provides, on average, 1360 W/m^2 of solar flux in LEO; however, during eclipse periods the spacecraft does not have access to solar heating and its primary heat source is blackbody radiation from the Earth [103]. Another source of energy is the Earth's albedo, which is the fraction of sunlight reflected off the Earth. We assume that satellites in the "dawn-dusk" orbit are not affected by albedo since they thread the terminator.

In addition to these external sources, spacecraft components also give off heat as they consume power. We assume that any power consumed will be released as heat.

[4]We could not find an operating temperature specification for the Teensy 4.0. We will assume it has an operating range of at least −40 to +85°C since this is a typical operating range for electronic components.

Table 11-48A in *Space Mission Analysis and Design* outlines the worst-case hot and cold conditions in Earth orbit [103]. Conditions for the typical ISS low-earth orbit (inclination between 30–60°) and sun-synchronous orbit (inclination between 60–90°) are summarized in Table 6.11. Note that the cold case for the LEO orbit is during eclipse (when there is no solar flux) but the sun-synchronous "dawn-dusk" orbit we are considering has no eclipse period.

Parameter	**ISS**		**Sun-Synchronous**	
	Hot	**Cold**	**Hot**	**Cold**
Solar constant (W/m^2)	1420	0	1414	1322
Albedo	0.57	0	0	0
Earth blackbody radiation (W/m^2)	257	218	257	218
Internal power dissipation (W)	0.48	0.12	0.48	0.12

Table 6.11: Worst-case thermal parameters for Earth orbits [103].

For the internal power dissipation, we assume that all components are in low-power mode in the cold case and all components are in their maximum-power modes in the hot case. See Table 4.5 for component power consumption data.

We must ensure that the spacecraft does not exceed 60°C during the hot case (maximum power consumption and maximum solar, albedo, and IR fluxes) and does not go below −20°C during the cold case (minimum power consumption and minimum solar, albedo, and IR fluxes).

6.4.3 Spacecraft Radiation Properties

In the vacuum of space, the primary mode of heat transfer for spacecraft is radiation. In terms of thermal analysis, all materials have two properties that affect how they transfer heat radiatively: absorptance (α), which describes how a material absorbs radiation (generally in the visible light spectrum), and emissivity (ϵ), which describes how a material emits radiation (generally in the infrared radiation spectrum) [104].

A material's absorptance governs how much solar radiation (including albedo) it absorbs, and its emissivity determines how much blackbody radiation it absorbs and how much of its own heat it radiates.

Since we are treating the spacecraft as a single node, we must calculate the absorptance and emissivity of the satellite as a whole. This involves combining the surface area fraction of each material with its known absorptance and emissivity. The absorptance/emissivity properties and surface area fractions (derived from the CAD model) are shown in Table 6.12 below.

Material	**Absorptance (α)**	**Emissivity (ϵ)**	**Area fraction**
Bare aluminum	0.17	0.03	0.12
White paint[5]	0.18	0.85	0.06
Solar cells[6]	0.8	0.8	0.54
FR-4 (PCB)	0.8	0.8	0.28

Table 6.12: Absorptance, emissivity, and surface area fractions of TigerCub external materials [105, 106, 107].

We calculate the overall absorptance and emissivity of the spacecraft using the following formulas:

$$\alpha = \sum_i \alpha_i f_i, \quad \epsilon = \sum_i \epsilon_i f_i \tag{6.4.1}$$

where f_i denotes the surface area fraction of the material. For the baseline configuration of the spacecraft, we obtain $\alpha_{\text{baseline}} = 0.69$ and $\epsilon_{\text{baseline}} = 0.72$.

6.4.4 Single-Node Isothermal Steady-State Analysis

Now that we have the spacecraft thermal environment and radiation properties, we can calculate the steady-state temperature of the satellite in the worst-case hot and cold conditions. Since this is a first-order analysis, the worst-case conditions envelop

[5]We assume that the patch antenna (which is white) has approximately the same properties as white paint.

[6]We took the average values for the solar cells listed in the "Solar Cells" table in [106].

all possible conditions the spacecraft could be in, and is therefore a conservative estimate of the steady-state temperatures.

We begin with the first law of thermodynamics, which states that

$$Q_{\text{in}} - Q_{\text{out}} = \frac{dU}{dt}. \tag{6.4.2}$$

In the steady state, there is no change in the temperature of the spacecraft so $\frac{dU}{dt} = 0$ and the equation becomes

$$Q_{\text{in}} = Q_{\text{out}}. \tag{6.4.3}$$

As previously defined, the spacecraft gains energy through sunlight, Earth albedo, Earth blackbody radiation, and internal power dissipation. The spacecraft loses energy through its own radiation to the vacuum of space. We define expressions for each of these.

First, we must define the view factor from the spacecraft to the Earth. A view factor (F) describes the fraction of energy from one surface (i.e. Earth) that directly transmits onto another surface (i.e. the external surface of the spacecraft); this factor depends only on the geometry and distance between the two bodies [108]. For the view factor calculation, we will assume one face of the satellite is directly facing Earth (and four faces are level/perpendicular to the Earth). Then, the view factor is

$$F_{\text{Earth}} = \frac{1}{h^2} + 4\left[\frac{1}{\pi}\left(\arctan\frac{1}{x} - \frac{x}{h^2}\right)\right] \tag{6.4.4}$$

where h is the ratio between the spacecraft's semi-major axis and the radius of the central body, and $x = \sqrt{h^2 - 1}$ [108].

For a spacecraft at an altitude of $h = 400$ km in orbit around Earth (which has a radius of $R_E = 6378$ km), we have

$$h = \frac{h + R_E}{R_E} = 1.063$$

and $x = 0.360$. From these, we obtain the view factor $F_{\text{Earth}} = 0.35$.

For the solar view factor, the value of the solar flux already takes into account the spherical geometry of the Sun and the flux value is provided as a planar constant. Thus, the view factor of a cubical spacecraft is simply the area of the cube's shadow on a plane; for a tumbling spacecraft, the average shadow area (as a fraction of the total surface area) is approximately $F_{\text{Sun}} = 0.25$ [109].

We can now define the energies from each heat source. $A = 0.015$ m^2 is the surface area of the satellite (approximated as a 5-cm cube).

$$Q_{\text{solar}} = (\text{solar flux}) F_{\text{Sun}} A \alpha \tag{6.4.5}$$

$$Q_{\text{albedo}} = (\text{albedo})(\text{solar flux}) F_{\text{Earth}} A \alpha \tag{6.4.6}$$

$$Q_{\text{Earth IR}} = (\text{Earth IR flux}) F_{\text{Earth}} A \epsilon. \tag{6.4.7}$$

These, along with the internal power dissipation, add up to the input energy:

$$Q_{\text{in}} = Q_{\text{solar}} + Q_{\text{albedo}} + Q_{\text{Earth IR}} + Q_{\text{internal}}. \tag{6.4.8}$$

The energy emitted from the spacecraft can be calculated using the Stefan-Boltzmann Law, which describes the radiative output of a body at a given temperature:

$$Q_{\text{out}} = \sigma \epsilon A T^2 \tag{6.4.9}$$

where $\sigma = 5.67 \times 10^{-8}$ W/m^{2}K^4 is the Stefan-Boltzmann constant and T is the temperature of the spacecraft in Kelvin. Setting Q_{in} equal to Q_{out} yields the equation

$$Q_{\text{solar}} + Q_{\text{albedo}} + Q_{\text{Earth IR}} + Q_{\text{internal}} = \sigma \epsilon A T^2. \tag{6.4.10}$$

At this point, the only unknown is the spacecraft temperature T. We can solve for T in the steady-state hot and cold cases to obtain the results in Table 6.13.

Orbit	Hot Case	Cold Case
ISS	65.5°C	−75.0°C
Sun-synchronous	29.1°C	17.4°C

Table 6.13: Steady-state temperature results for ISS and no-eclipse sun-synchronous orbits from first-order analysis.

These results suggest that if we assume a steady-state in the worst hot and cold conditions, we will exceed the operating temperature range (on both ends) in an ISS orbit. However, the no-eclipse sun-synchronous orbit stays within the desired temperature range. This suggests that in the baseline configuration, TigerCub can operate safely in a no-eclipse "dawn-dusk" sun-synchronous orbit. However, the "dawn-dusk" orbit limits the potential missions that TigerCub can perform. This motivates some additional analysis and thermal control design so that TigerCub can also operate in an ISS low-earth orbit.

6.4.5 Single-Node Isothermal Transient Analysis

We perform a higher-fidelity transient analysis (as opposed to steady-state) for the ISS orbit. The transient analysis simulates the changing input energy conditions as the spacecraft orbits the Earth, and calculates the resulting spacecraft temperature. We will use the same spacecraft model as before, assuming that the spacecraft is isothermal and operates as a single node.

We use the proprietary CubeSat Toolbox thermal simulators from Princeton Satellite Systems (provided to us for educational purposes) to perform transient analyses for the ISS orbit. The orbital parameters were defined in Table 6.9. For all analyses, we define the initial temperature to be 300 K (around room temperature).

The transient analysis is similar to the steady-state analysis described above except that the temperature is propagated over time using the equation

$$\frac{dQ}{dt} = mc_p \frac{dT}{dt} \tag{6.4.11}$$

where m is the mass of the satellite and c_p is its specific heat. We approximated TigerCub's specific heat by taking the weighted average of the specific heats of each component, and we obtained a value of $c_p = 1068$ J/kg·K.

The results of the transient analysis is displayed in Figure 6.13 below.

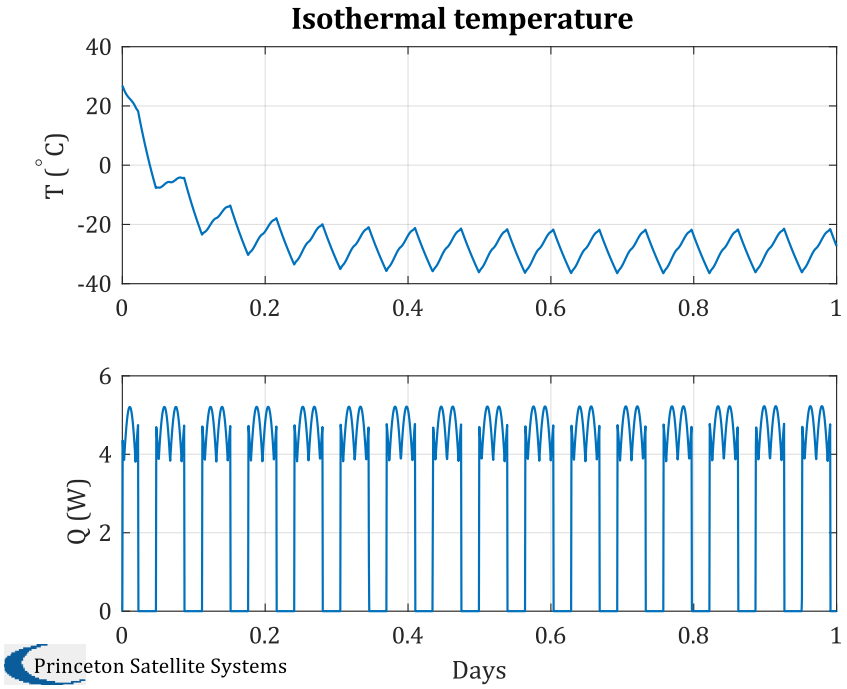


Figure 6.13: Temperature and Q_{in} results from CubeSat Toolbox transient analysis for spacecraft baseline configuration in an ISS orbit.[7]

We can see that in the ISS orbit, the spacecraft tends to run cold and is almost entirely out of the operating range of the battery (but in the operating range of all other components). This suggests that the baseline configuration of the satellite is not sufficient to keep the battery warm enough to operate.

6.4.6 Thermal Control

Spacecraft typically use a combination of passive thermal control methods (e.g. thermal coatings and surface finishes) and active control methods (e.g. powered heaters and coolers) [105]. Due to the limited power budget, TigerCub will only use passive thermal control. This typically consists of surface finishes (e.g. white or black paint,

[7] Q does not include internal power dissipation.

silvered Teflon, and anodized surfaces) and/or insulation (e.g. multi-layer insulation and single-layer radiation shields) [105].

To change the absorptance and emissivity of the spacecraft, we can add aluminum tape, which has the low absorptance ($\alpha = 0.20$) and emissivity ($\epsilon = 0.03$) characteristic of metals, to exposed outer surfaces of the spacecraft. Consider the case in which we add aluminum tape to all exposed PCB surfaces, which includes exposed PCB on the solar panels and sliding backplate. From Table 6.12 and Equation 6.4.1, this reduces the overall absorptance and emissivity of the spacecraft to $\alpha_{\text{tape}} = 0.48$ and $\epsilon_{\text{tape}} = 0.45$. Re-running the transient analysis for a spacecraft in the ISS orbit yields the results in Figure 6.14.

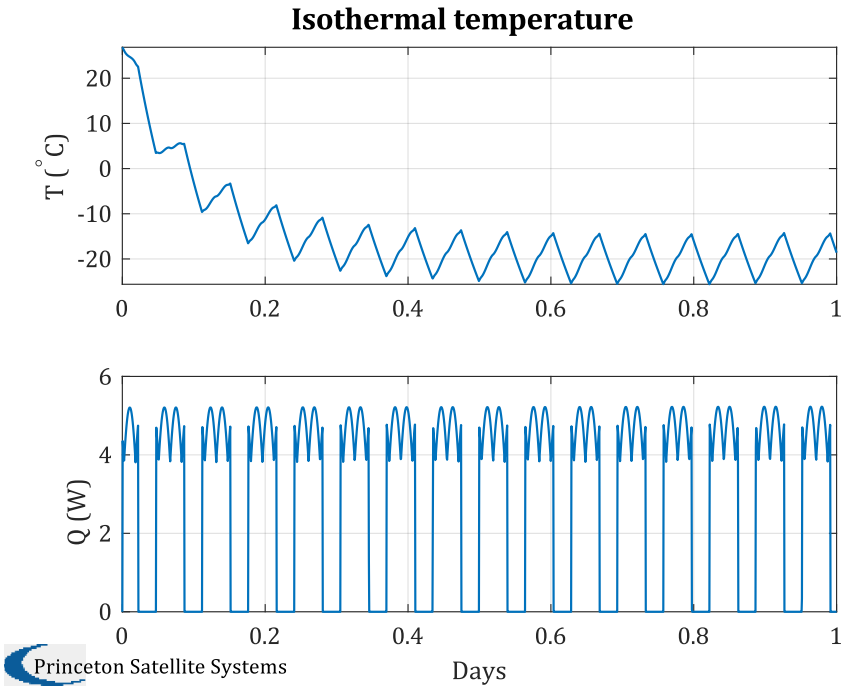


Figure 6.14: Temperature and Q_{in} results from CubeSat Toolbox transient analysis for aluminum tape configuration.

These results suggest that aluminum tape helps increase the satellite's temperature but is not sufficient to keep the spacecraft in the battery's operating temperature range; the spacecraft is colder than $-20°\text{C}$ for about half the time. TigerCub developers can choose to risk operating the battery below its specified temperatures, which may result in suboptimal battery performance and a shorter cycle life.

Future developers could choose to insulate the battery in addition to adding aluminum tape if they wish to operate in an ISS orbit. Common spacecraft insulators are Kapton or Teflon film. There is very little space within the satellite to add insulation around the battery. We perform a 1D steady-state heat conduction analysis to estimate the temperature increase from a 0.5 mm-thick Kapton film around the battery.

The thermal conductivity of Kapton film is 0.120 W/mK. We model the Kapton layer as a flat plate, and the one-dimensional heat conduction equation through a flat plate is

$$\frac{\Delta Q}{A} = k\frac{\Delta T}{L} \tag{6.4.12}$$

where ΔQ is the amount of heat transferred, A is the area of the plate, k is the thermal conductivity of the material, ΔT is the temperature difference, and L is the thickness of the plate.

Lithium-ion batteries are very efficient and do not dissipate very much power as heat. We were unable to find specific data on the efficiency of our battery, but as a first-order assumption, we set Q = 0.01 W (about 5% of the average power consumption of the satellite). We assume that the battery radiates through its largest face, which has a surface area of about 8.75×10^{-4} m^2. The thermal conductivity of Kapton is approximately 0.120 W/mK. The "length" of the plate is the thickness of the Kapton layer, 0.5 mm.

Solving this equation for ΔT yields $\Delta T = 0.05$ K, which is a very small temperature difference (due to the low heat dissipation of the battery and the thin layer of Kapton). This suggests that developers may have to find alternate battery insulation or heating solutions, discussed in Section 9.3.4.

6.5 Materials in the Space Environment

6.5.1 Outgassing

Material outgassing occurs when materials lose some portion of their mass spontaneously to the outside environment, mainly due to the vacuum environment of space. Outgassing is quantified as total mass loss (TML) which is the mass of material lost to outgassing as a percentage of the original mass, and percent collected volatile condensable material (CVCM), which is the percentage of mass that outgasses and then condenses on a collector plate (i.e. the mass that might condense on optical or thermal control surfaces) [110].

From Requirements E-01 and E-02, the TML must be at most 1.0% and the CVCM must be at most 0.1%; these requirements were derived from the PocketQube Standard as shown in Table 1.1.

Using NASA's Material Outgassing Database, which provides TML and CVCM information for many common spacecraft materials, we verified the outgassing properties of several of our materials, shown in Table 6.14. Non-anodized aluminum and stainless steel are commonly used in spacecraft due to their low outgassing properties, so they already meet outgassing requirements.

Material	Data Ref. Number	TML (%)	CVCM (%)
FR-4 PCB	GSFC17818	0.35	0.01
Wire insulation	GSFC20585	1.34	0.37
Plastic in electronic components	GSFC22915	0.13	0.02

Table 6.14: Outgassing properties of TigerCub materials [111].

Note that the rubber insulation of most commercial wiring (including the ones used in this prototype) do not meet outgassing requirements. As described in Section 9.3.4, developers should replace these wires with Teflon (polytetrafluoroethylene)-insulated wiring, which has a TML of 0.01% and CVCM of 0% (data reference number: GSFC12601) [111].

We were unable to find outgassing data for most of the other components on TigerCub, either because outgassing data was unavailable or because information about the specific materials in COTS components was unavailable. However, some components (e.g. the solar cells and kill switches) have flight heritage, and we do not expect these to pose outgassing concerns.

To ensure that material outgassing does not contaminate the spacecraft, developers should perform a thermal vacuum bakeout of the entire satellite, as described in Section 9.3.3 (and as required by the commonly-used AlbaPod PocketQube deployer [50]).

6.5.2 Radiation

As previously discussed in Section 3.8, spacecraft are exposed to radiation in space, which can result in damage to electronic components if they are not properly shielded.

NASA has published a guide to using COTS electronic components for NASA missions, which states that Class D missions (high-risk, low-significance missions, typically with experimental payloads) should use established COTS parts that are manufactured by industry leaders; Class D missions are not required to use MIL-SPEC components or radiation-tested components [112, 113]. Most PocketQube missions would be classified under this category.

Satellites in LEO orbit with inclinations between 20–85° see approximately 1–10 krad/year [114]. Typical COTS parts can withstand a total radiation dose of 2–10 krad [114]. For these reasons, space radiation is likely one of the limiting factors of mission lifetime in the "dawn-dusk" no-eclipse sun-synchronous orbit as discussed in Section 4.6.3.

6.5.3 Spacecraft Charging

In the ionosphere, which typically extends from an altitude of 80 km to thousands of kilometers, the atmosphere contains ions and free electrons (which form from solar radiation) [115]. The presence of these charged particles leads to a region around the

spacecraft in which positive ions outnumber negatively-charged particles [115].

Since the region around the spacecraft is charged, the spacecraft itself must ensure that there are not large potential differences between spacecraft components, which could cause damaging electrical arcing [115].

The current TigerCub prototype has some protections against the effects of spacecraft charging. All components in the internal stack share the same ground, which helps prevent large potential differences across components in the stack. For better protection, the internal stack should also be grounded to the aluminum frame; this is further discussed in Section 9.3.4.

Chapter 7

Manufacturing and Assembly

This chapter details the manufacturing and assembly processes for the TigerCub prototype. A full bill of materials is included in Appendix C.2.

7.1 Outsourced PCB Manufacturing

The printed circuit boards (PCBs), once designed, need to be manufactured in order to be assembled and tested. Though the TigerSats Lab has some PCB manufacturing capabilities (via the Voltera PCB printer and Bantam PCB mill), we opted to outsource PCB manufacturing for the TigerCub prototype to save on costs and time.

We manufactured PCBs for the two OBC boards, solar panels, and sliding backplate through PCBWay, a custom PCB prototype service based in China [116]. PCBWay is very affordable compared to US-based PCB houses. We bought 5 boards for each PCB design, using a HASL lead-free finish, for 5.00 USD per design with an additional 21.75 USD for shipping and online payment fees. Build time is 24 hours and shipping takes 2-4 days with DHL. We typically received PCBs within a week of ordering them.

To order PCBs from PCBWay, export Gerber and NC Drill files from KiCAD and upload them to the PCB Instant Order website; PCBWay provides instructions on how to do this [117]. From there, developers can choose specific options (such as the HASL lead-free surface finish).

The EPS module was manufactured and assembled by JLCPCB, another custom PCB service based in China [118]. Due to the large number and small size of surface-mounted device (SMD) components, we outsourced the assembly as well as the manufacturing to JLCPCB. Though assembly could potentially be done in-house (using a solder paste stencil, reflow machine, and steady hands), we do not recommend it for students without extensive PCB assembly experience.

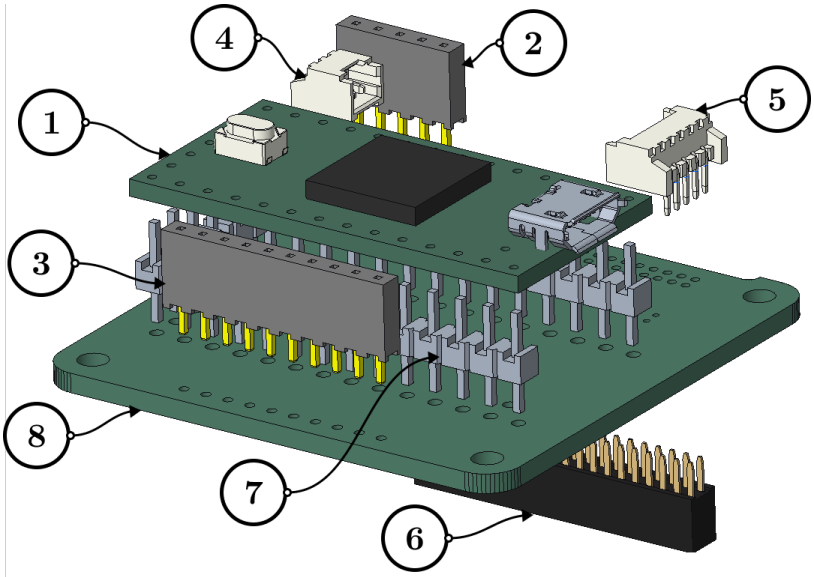
JLCPCB manufactured the PCBs, sourced most of the PCB components, and assembled the PCBAs. We provided JLCPCB with a bill of materials (BOM) and a component placement list (CPL); JLCPCB also has instructions for these procedures on their website [119]. We had JLCPCB manufacture 5 PCBs and assemble 2 PCBAs for a total cost of 177.04 USD, including shipping. There were a few parts that JLCPCB was unable to source. Fortunately, these parts were easy to add to the assembly in-house, as discussed in the next section.

7.2 In-house PCB Assembly

The majority of components on our PCBs are through-hole components that can be soldered directly to the PCBs using a soldering iron kit. Most of these components have a standard 2 mm or 2.54 mm (0.1 in) pitch, which are easily solderable with basic soldering training. Some additional assembly details are included in the following sections.

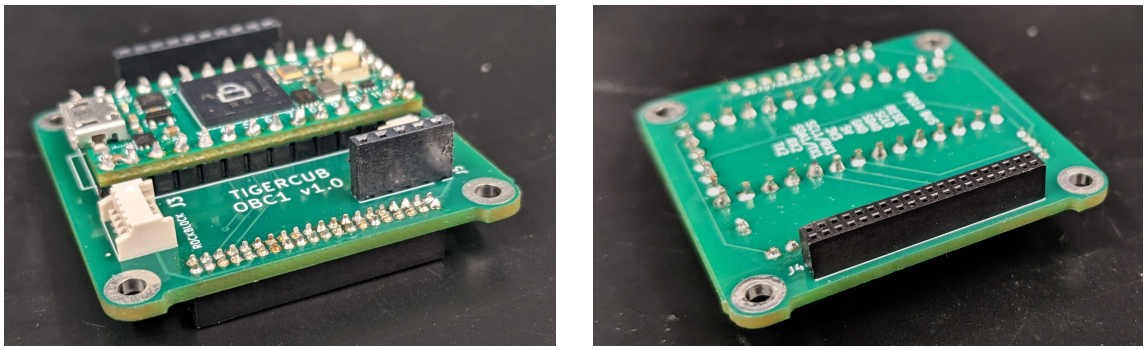
7.2.1 OBC1 Assembly

OBC1 contains the Teensy 4.0 development board and connectors to the EPS, OBC2, sliding backplate, and RockBLOCK as shown in Figure 7.1. The most difficult component to solder on these boards is the 2×20 1.27-mm pitch header that connects to the EPS, shown in Figure 7.2. However, since this is a through-hole connector and there are only two rows of pins, it is still easily solderable with some practice. Make sure that the connector is soldered onto the bottom side of the board as shown in the figure.



#	Component	Part Number	Qty
1	Teensy 4.0	TEENSY40	1
2	5-pin header	SQT-105-01-F-S	1
3	10-pin header	SQT-110-01-F-S	1
4	Picoblade 2-pin	53048-0210	1
5	Picoblade 5-pin	53048-0510	1
6	2x20 connector	M50-3002045	1
7	Header pins	N/A	33
8	OBC1 PCB	N/A	1

Figure 7.1: Exploded view of OBC1 with components list.

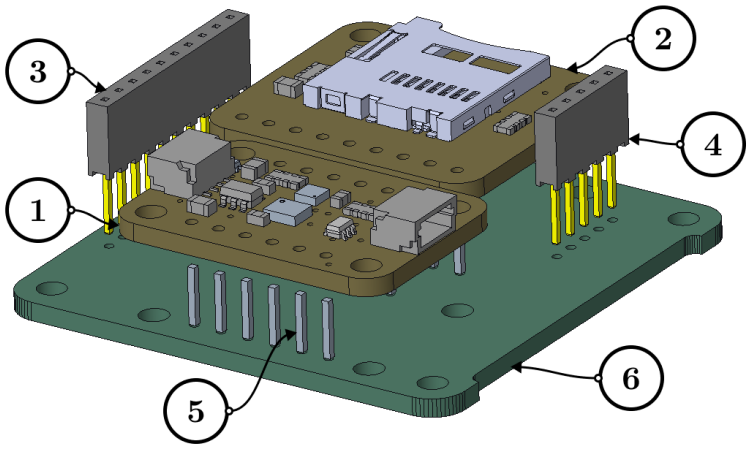


(a) Top side view. **(b)** Bottom side view.

Figure 7.2: 2×20 connector soldered onto OBC1.

7.2.2 OBC2 Assembly

OBC2 contains the IMU breakout board, MicroSD breakout board, PQ10 connector, and 5-pin connector to OBC1 as shown in Figure 7.3. These components all have through-hole pins and are easily solderable.



#	Component	Part Number	Qty
1	IMU breakout	Adafruit 5543	1
2	MicroSD breakout	Adafruit 4682	1
3	10-pin header	SQT-110-03-F-S	1
4	5-pin header	SQT-105-03-F-S	1
5	Pins	N/A	19
6	OBC2 PCB	N/A	1

Figure 7.3: Exploded view of OBC2 with components list.

Note that the IMU breakout board should be soldered without a header. Adding a header would increase the total component height and cause interference issues with the adjacent board. Instead, the IMU should be soldered directly onto the OBC2 PCB using bare pins as shown in Figure 7.4. Developers have the option of using header pins with the MicroSD breakout board, but it can also be soldered directly to the OBC2 PCB.

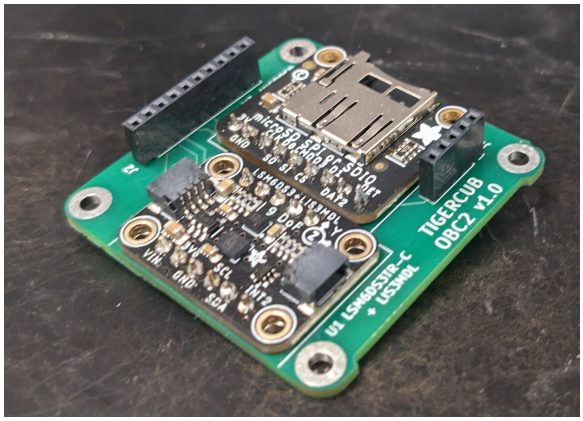
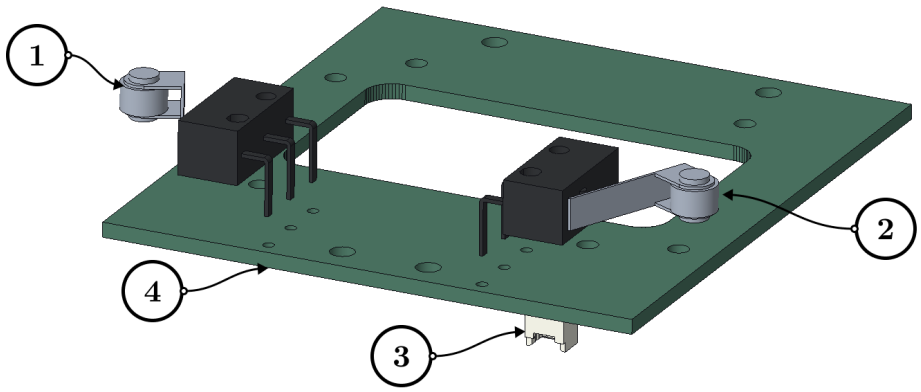


Figure 7.4: Breakout boards soldered onto OBC2 without headers.

7.2.3 Sliding Backplate Assembly

The only components on the sliding backplate are the two kill switches and the Molex Picoblade two-pin connector, all of which have through-hole pins, as shown in Figure 7.5.



#	Component	Part Number	Qty
1	Roller switch, RA	D2F-L2-A1	1
2	Roller switch, LA	D2F-L2-A	1
3	Picoblade 2-pin	53048-0210	1
4	Sliding backplate PCB	N/A	1

Figure 7.5: Exploded view of sliding backplate with components list.

During assembly, ensure that the kill switches are facing the correct direction. The kill switch hinges should be closest to the edge of the board as shown in Figure 7.6.

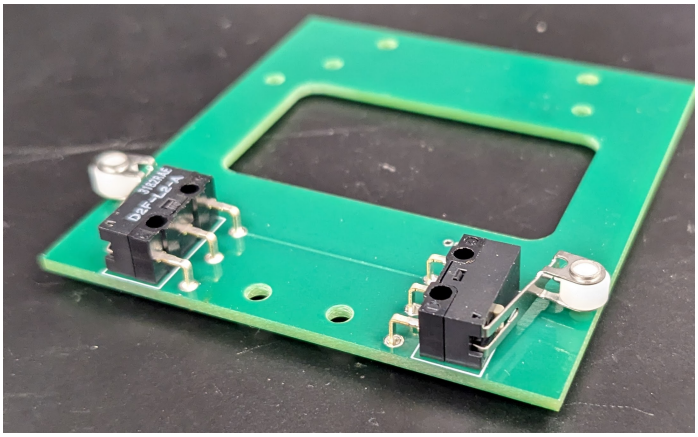
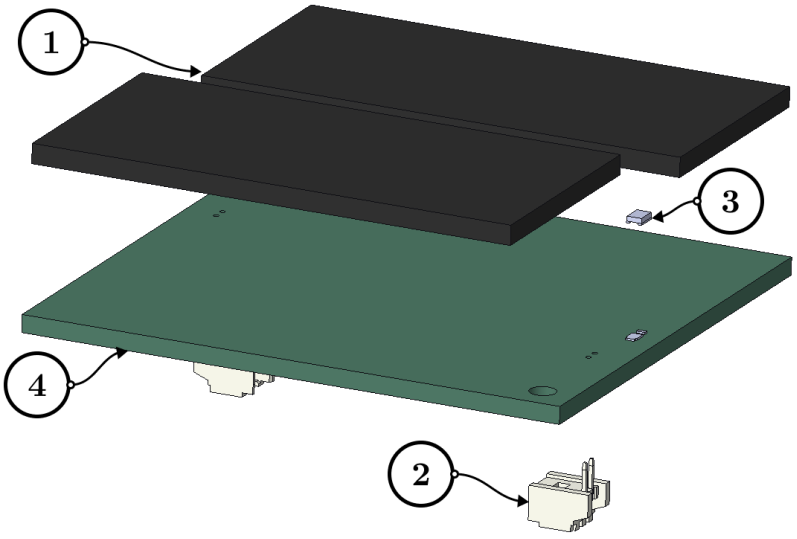


Figure 7.6: Kill switches soldered onto sliding backplate.

7.2.4 Solar Panel Assembly

The solar panel PCBs have several surface-mount components: the solar cells and diodes. A sample solar panel breakdown is shown in Figure 7.7; note that the second variant of the solar panels only has one Picoblade connector.



#	Component	Part Number	Qty
1	Solar cell	SM940K09L	2
2	Picoblade 2-pin	53048-0210	2
3	Schottky diode	CUHS20S40,H3F	1
4	Solar panel PCB	N/A	1

Figure 7.7: Exploded view of solar panel with components list.

The solar cells and diode should be assembled by applying solder paste onto the solder pads then using a reflow machine such as the Voltera V-One PCB printer [120] as shown in Figure 7.8. If a reflow machine is not available, integrators can (carefully) use a heat gun instead, but should take extra care to verify that components are fully soldered onto the board.

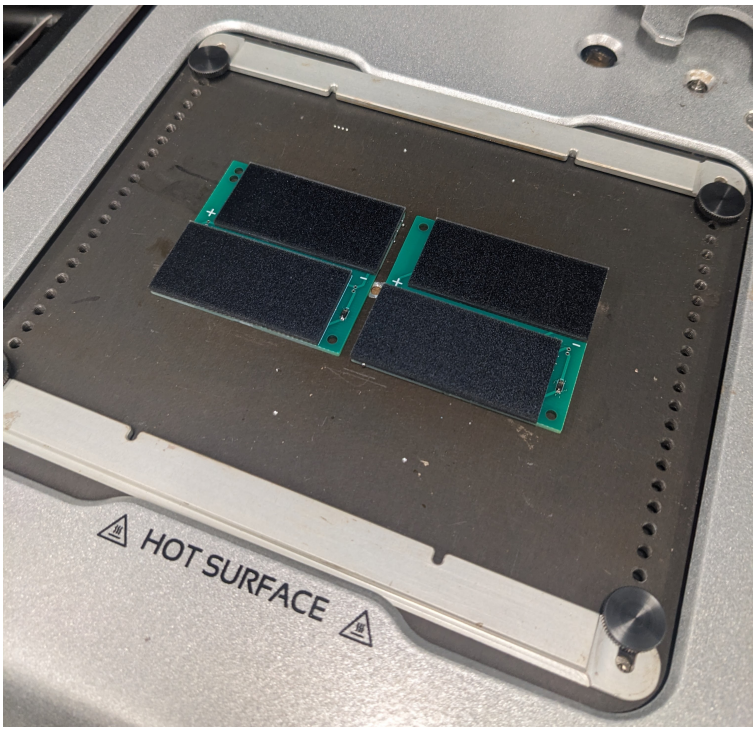


Figure 7.8: Solar panel assembly on Voltera reflow machine.

The final component on the solar panels is the Molex Picoblade 2-pin connector(s), which have through-hole pins and can be attached using a soldering iron.

7.2.5 EPS Assembly

As described in Section 7.1, the DynOSSAT-EDU EPS module was mostly assembled by JLCPCB, but there were a few components we needed to integrate ourselves. We soldered the slide switch, Molex Picoblade connectors, and LED onto the EPS; these components are circled in Figure 7.9. There is an additional Picoblade connector on the bottom side of the PCB that also needs to be assembled. With the exception of the LED, these are easy to solder. The LED is a surface-mount component with four

small pads; builders should be careful during the soldering process. The LED simply serves as an indicator that the board is powered, so developers can leave it off the board if necessary.

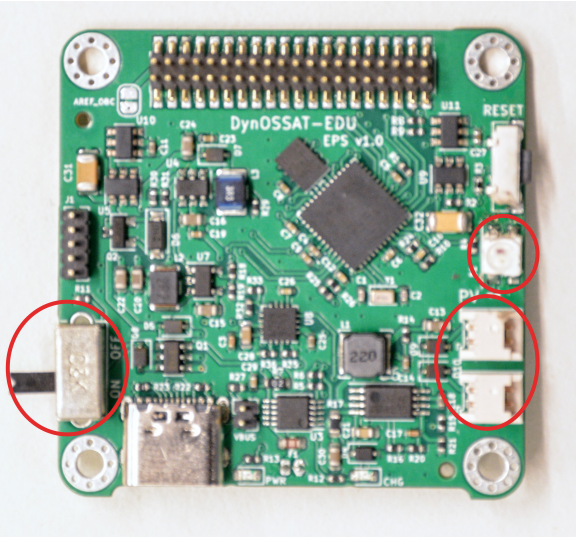


Figure 7.9: Components integrated onto the EPS board in-house.

7.3 Frame Manufacturing

For ease of iteration, we manufactured the sheet-metal frame through PCBWay since PCBWay provides relatively inexpensive and fast sheet-metal manufacturing. We purchased two Al-5052 sheet-metal frames with no surface finish through PCBWay for 53.89 USD (excluding shipping). Sheet metal products have a build time of 6 to 11 days.

The frames arrived mostly free of defects. There were some small jagged edges from punch-outs during the sheet metal forming process, but none of these defects interfered with the structural function of the frame. Nonetheless, developers may want to have the sheet metal frame fabricated through a manufacturer with greater quality control such as Xometry or Protolabs. Xometry quoted a single Al-5052 sheet-metal frame for 167.74 USD with a build time of 10 business days.

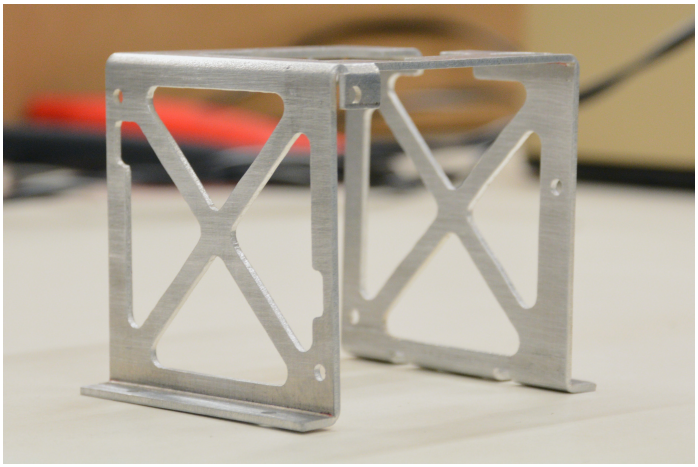


Figure 7.10: Sheet-metal frame manufactured by PCBWay.

We also had the small L-brackets manufactured through PCBWay, which cost 137.20 USD for five brackets (excluding shipping costs). Developers can alternatively choose to manufacture the brackets through another machining shop. Xometry quoted two hard-anodized brackets for 101.59 USD each with a lead time of 9 business days.

7.4 Satellite Assembly and Integration

Once all subsystem components have been manufactured and assembled, we can integrate them into the PocketQube. Due to the small components and layout of the spacecraft, it is important to assemble the satellite in a specific order, as follows:

1. Integrate the internal stack onto the sliding backplate.
2. Fasten the L-brackets onto the sliding backplate.
3. Integrate the RockBLOCK module onto the sheet-metal frame.
4. Fasten all solar panels onto the sheet-metal frame.
5. Integrate the sliding backplate with the sheet-metal frame.

Additional assembly details for each step are discussed in the following sections. In accordance with NASA Directive 540-PG-8072.1.2B (Mechanical Fastener Torque Guidelines), all fasteners should be torqued to 5.7 in-lb (0.64 N-m) [121].

7.4.1 RockBLOCK Modifications

Before integration can occur, we need to remove the SMA connector (circled in Figure 7.11) from the RockBLOCK module.

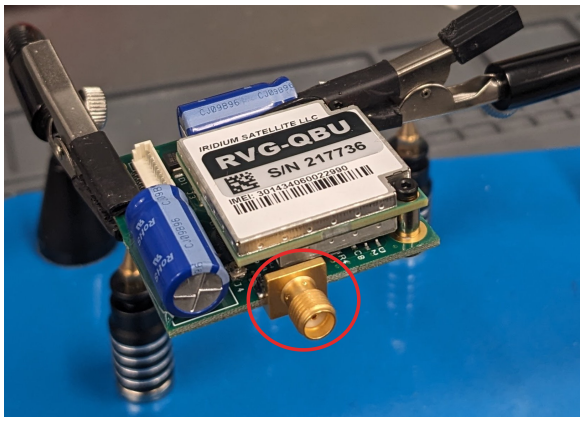


Figure 7.11: RockBLOCK module before removing the SMA connector (circled).

The pins of the SMA connector are located partially below the 9603 modem, which we removed for ease of desoldering. The modem is fastened to the module with two small screws and a header connector. We removed the two screws using a 1.5 mm hex driver and carefully wiggled the modem out of its header connector. We left the modem connected via its UFL cable.

Figure 7.12: RockBLOCK module after removing the Iridium 9603 modem.

The SMA connector has three pins on the top side of the PCB; they are slightly

visible in Figure 7.13. Using a soldering iron and a copper desoldering braid, we removed as much of the solder from these pins as possible. We then used wire cutters to snip off the three pins, at which point the SMA connector can be easily removed.

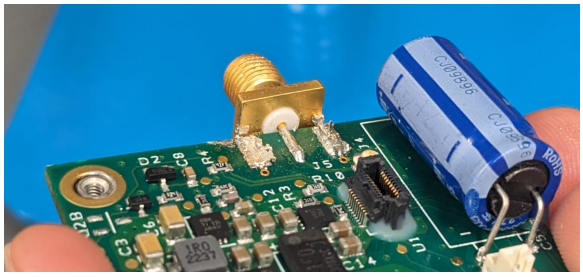


Figure 7.13: Removing solder from the SMA connector pins.

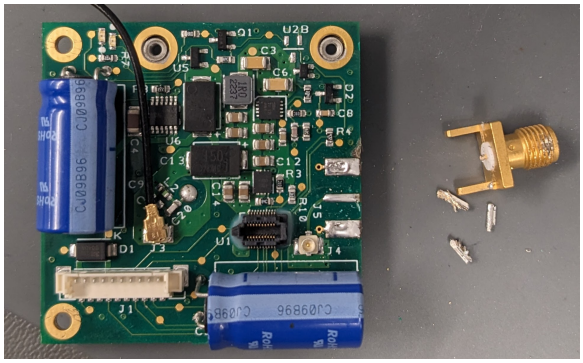


Figure 7.14: RockBLOCK module after removing the SMA connector.

We also need to remove one of the soldered nuts on the RockBLOCK, which allows us to fasten the RockBLOCK to the aluminum frame at a third mounting point to prevent cantilevering, shown in Figure 7.15.

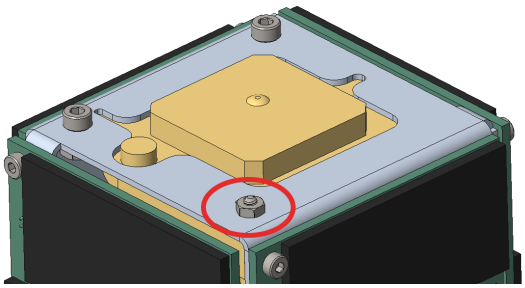
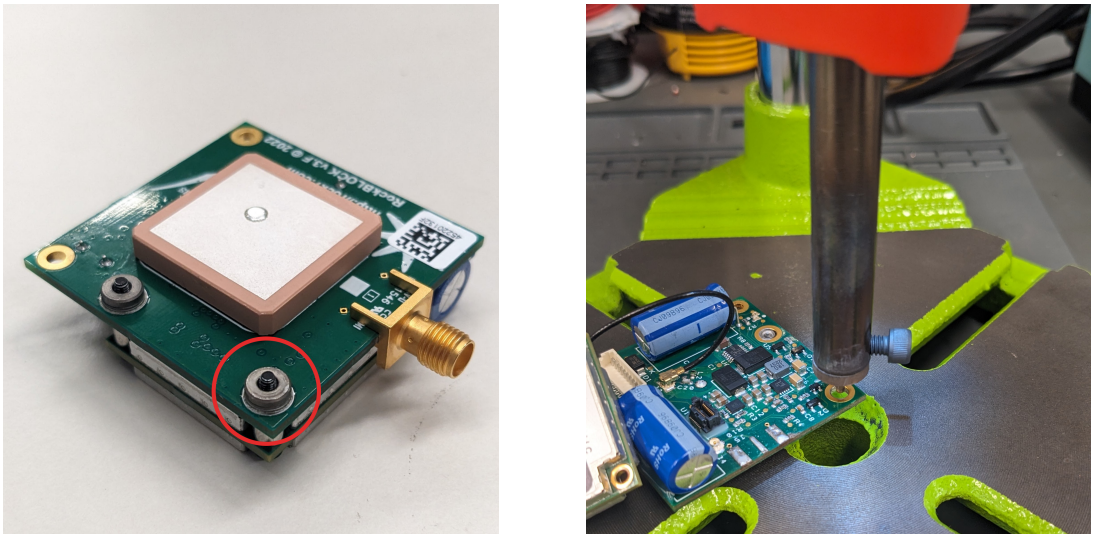


Figure 7.15: Third mounting point for RockBLOCK to frame.

The soldered nut to be removed is shown in Figure 7.16. To remove the nut, we

heated it up using a soldering iron until it released and fell off.



(a) Soldered nut to remove. **(b)** Soldering iron setup.

Figure 7.16: Removing soldered nut from RockBLOCK.

Once the SMA connector and soldered nut had been removed, we re-connected the modem and replaced its two screws. We then verified that the module had not been damaged during the process by successfully transmitting and receiving a message using the RockBLOCK.

7.4.2 RockBLOCK and Solar Panel Integration

The RockBLOCK and solar panels are integrated directly onto the sheet-metal frame. First, fasten the RockBLOCK to the frame using two pairs of M2.5 screws and nuts as shown in Figure 7.17. Then, for the RockBLOCK's third mounting point, fasten the original M2 screw through the RockBLOCK and frame into a M2 nut.

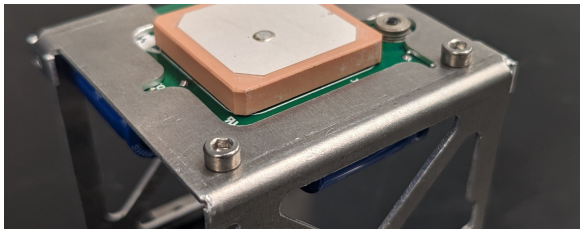


Figure 7.17: RockBLOCK attachment points.

Fasten the two single-connector solar panels onto the front and back of the frame, using M2 screws and nuts. For later ease of assembly, we added the M2 screws that fasten the frame to the sliding backplate before we integrated the front and back solar panels, as shown in Figure 7.18.

Fasten the two double-cell solar panels onto the left and right sides of the frame using a single M2 screw and nut. At this time, the solar panels will be attached at one point each, at the top left corner. The bottom right corners will be fastened to the L-brackets in the final integration step.

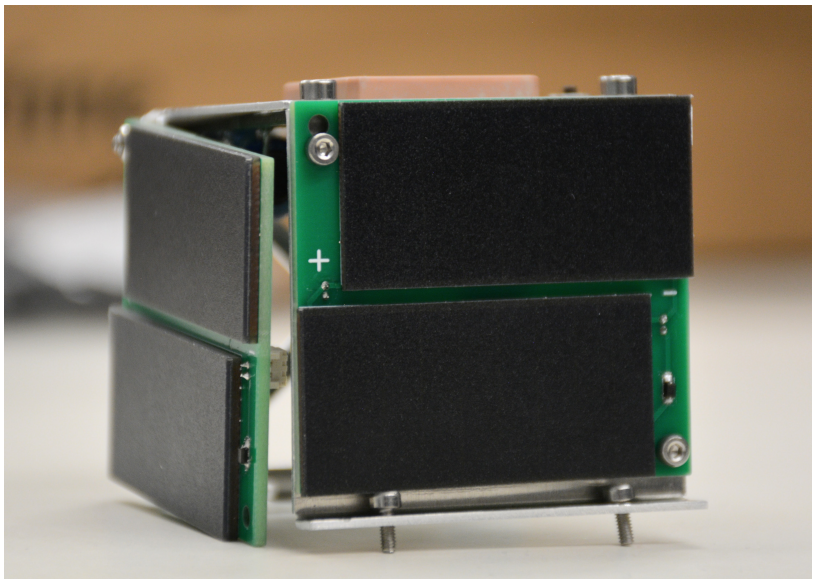


Figure 7.18: Two solar panels integrated onto frame.

Plug in the four solar panel wire harnesses (2-pin Picoblade cables). Two of these cables should go from one solar panel to another, and the other two cables go from a solar panel to the EPS.

We also need to construct the wire harness from OBC1 to the RockBLOCK. The pinout for this harness is shown in Figure 2.1. Using pre-crimped wires, connect the relevant pins on a 10-pin Picoblade socket connector to the pins on a 5-pin Picoblade socket connector as seen in Figure 7.19.

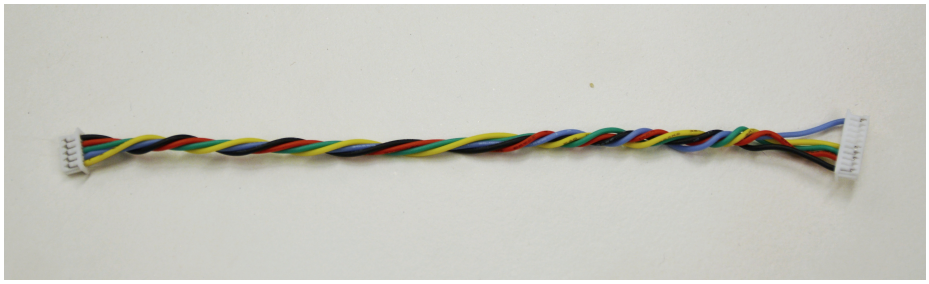


Figure 7.19: Wire harness from RockBLOCK to OBC1.

After assembling the harness, plug the 10-position connector into the RockBLOCK.

7.4.3 Internal Stack Assembly

From bottom to top, the order of the internal stack is payload PCBA (if using), OBC2, OBC1, and finally the EPS; see Figure 1.2.

To start, plug the kill switch harness (a 2-pin Picoblade cable) into its Picoblade connector on the sliding backplate. fasten the two L-brackets onto the sliding backplate using M2 screws. Then, slide the four M2×35 screws onto the sliding backplate as shown in Figure 7.20. If using the payload PCB, slide one 5/32" spacer onto each of the screws, followed by the PCB then one 9/32" spacer on each screw; if not, use one 1/2" spacer on each screw.

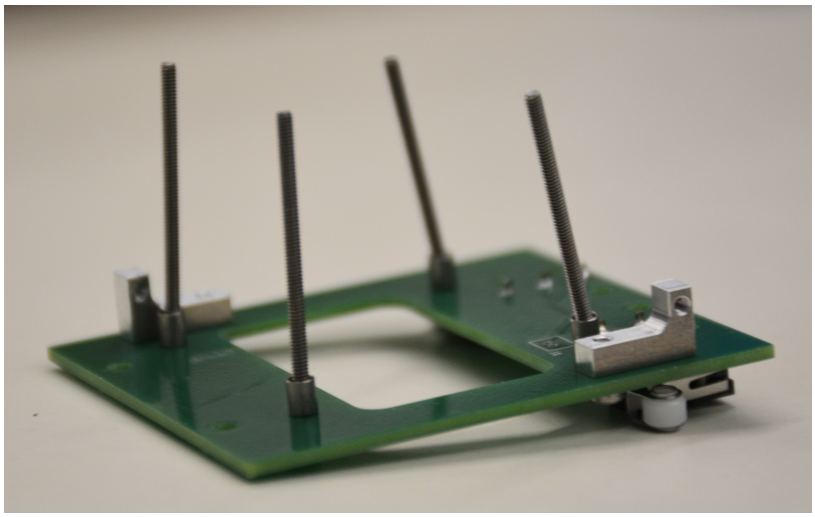


Figure 7.20: 35-mm M2 screws in sliding backplate.

Next, slide the OBC2 PCBA onto the stack. Ensure that the PQ10 connector

from the payload board (if using) attaches securely to the PQ10 connector on the OBC2 board. Once OBC2 is secure, slide another 9/32" spacer onto each of the four screws.

Before adding the OBC1 PCBA, connect the kill switch harness and the RockBLOCK harness to their respective connectors on OBC1. Then, slide the OBC1 PCBA onto the stack. Again, ensure that the PQ10 connector and 5-pin connector are securely connected.

Plug in the two solar panel cable connectors and the battery connector into the respective EPS connectors. Slide the EPS module onto the stack. Be careful with the 2×20 connectors on OBC1 and the EPS. Once the connectors are securely attached, fasten a M2 nut onto the end of each of the four screws. The internal stack is now complete.

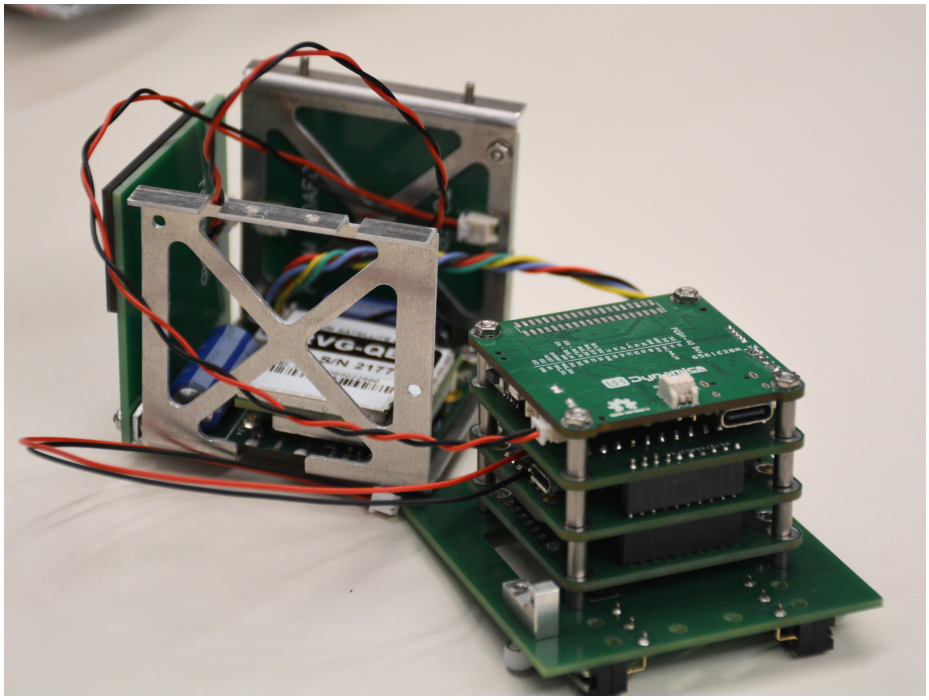


Figure 7.21: Assembled internal stack.

7.4.4 Final Integration

We must now integrate the frame (with all its components) onto the sliding backplate (which now includes the internal stack).

First, place the sheet metal frame on top of the sliding backplate and ensure that there are no inferences. Fasten the two double-cell solar panels to the L-brackets using the M2 screws. Finally, fasten the frame to the sliding backplate using four M2 screws and nuts (two on the front and two on the back).

TigerCub integration should now be complete.

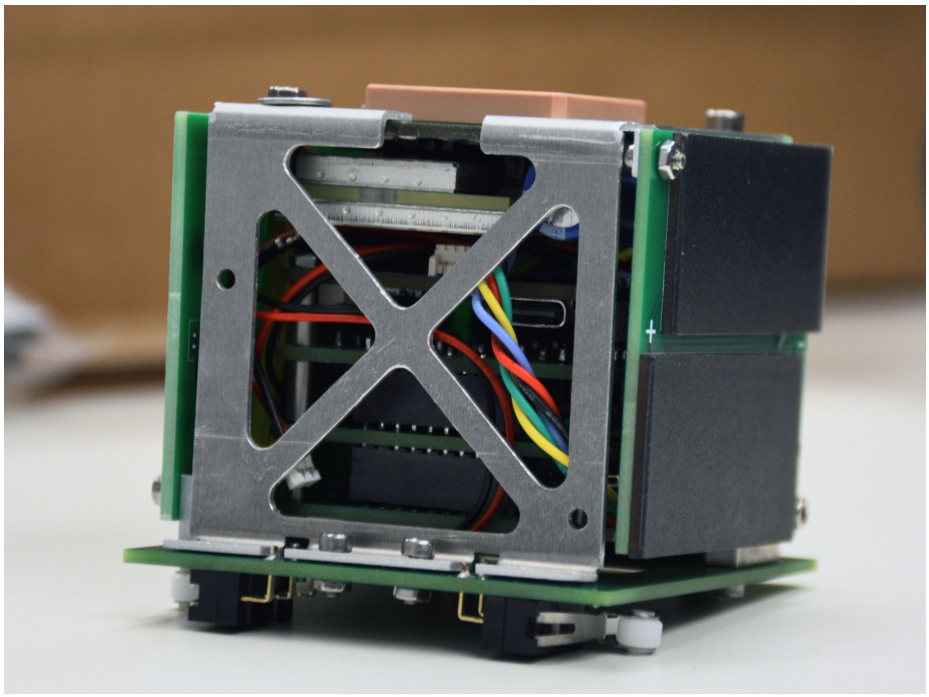


Figure 7.22: TigerCub assembly. As of the time of writing, the final frame and solar panel prototypes had not arrived yet, so this assembly used some previous prototypes. Because of this, we were unable to integrate the fourth solar panel for this build.

Developers may also 3D-print or machine a model of the PocketQube deployment rail to check for rail interference. The PocketQube should slide smoothly through the rail.

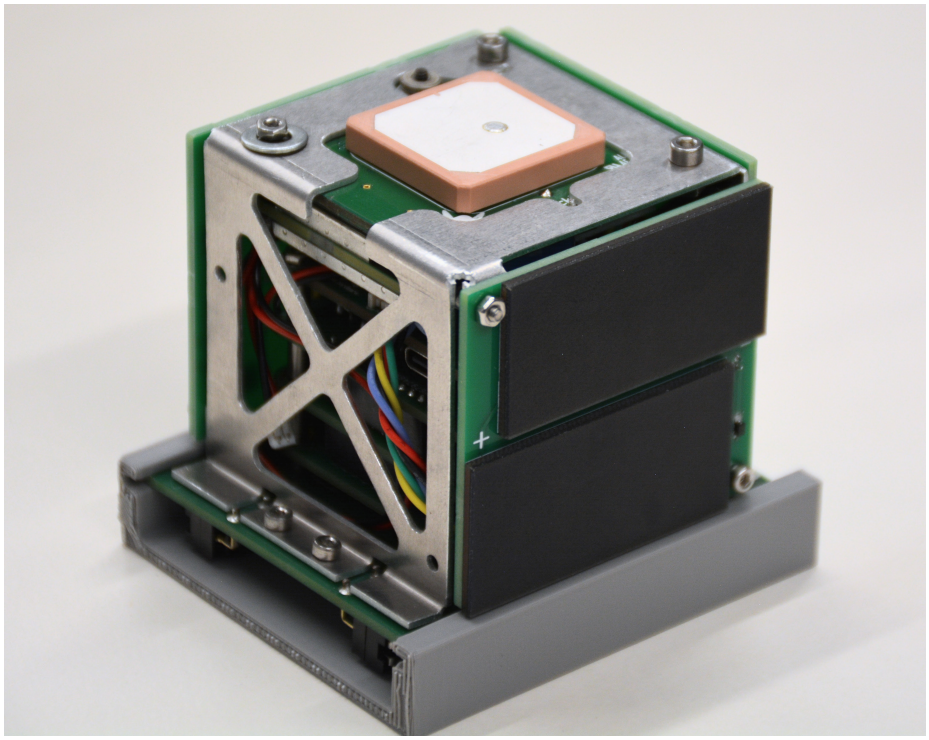


Figure 7.23: TigerCub assembly in deployment rail.

Chapter 8

TigerCub User's Guide

To aid future PocketQube development using the TigerCub platform, we have created a User's Guide for future TigerCub developers and mission planners. The User's Guide is designed to help developers build a TigerCub without needing to know all the details of its internal design.

Some sections in this chapter reference other sections in this report (rather than repeating the information), but a stand-alone User's Guide will be available in the TigerCub GitHub repository at `https://github.com/candacedo/tigercub-pq`, along with all design files. As of the time of the publication of this report, this repository has not been published yet due to export control concerns (and users are not able to access it), but it will be made available to the public in late 2024.

8.1 Hardware Overview

TigerCub is composed of five main subsystems as shown in Figure 1.1: the radio communication system, the on-board computer and attitude determination system, the electrical power system, the payload, and the structure. We briefly summarize the functions of each subsystem in this section.

The radio communication system, described in Chapter 2, transmits and receives messages between TigerCub and ground control via the Iridium satellite network. TigerCub uses the pre-built RockBLOCK 9603 module, which is built around the

Iridium 9603 modem. This small but powerful radio module can be conservatively tested from ground, which serves as reassurance that TigerCub can downlink (and uplink) from orbit, mitigating a common nanosatellite failure point. The RockBLOCK also has the advantage of having a low-power "sleep" mode, which helps conserve power in a power-constrained mission.

The on-board computer (OBC) and attitude determination system (ADS), discussed in Chapter 3, are comprised of the Teensy 4.0 development board (for on-board computation), the Adafruit LSM6DS3TR-C + LIS3MDL inertial measurement unit (IMU) breakout board (for attitude determination), and the Adafruit Micro SD SPI or SDIO Card Breakout Board (for on-board data storage). These pre-built breakout boards simplify circuit design, manufacturing, assembly, and usage for first-time PocketQube developers. The OBC interfaces with the payload through a custom PQ10 bus, which includes 5V and 3.3V power rails and both I2C and serial communication capabilities.

The electrical power system (EPS), discussed in Chapter 4, is composed of the open-source DynOSSAT-EDU EPS module, four solar panels, battery, and kill switches. The DynOSSAT-EDU EPS, an open-source EPS designed for PocketQubes, takes in power from the solar panels and battery and distributes it to the rest of the system via a 5V rail; it also includes key functions such as battery measurement and maximum power point tracking. The four body-mounted solar panels provide power directly to the EPS, which charges the battery, and the EPS sources power from the battery during eclipse periods. Finally, the two kill switches keep the satellite powered off while in the deployer, and power the satellite on after deployment.

The outer structure (described in Chapter 6) is comprised of the aluminum sheet-metal frame, two L-brackets, and sliding backplate board. The structure shields the internal components from the space environment and ensures that the satellite will survive the flight environment.

As a mission-agnostic platform, TigerCub does not provide a payload; instead, developers will integrate their own payloads. Payload interfaces and design are described in Section 8.3 and in Chapter 5.

8.2 Procurement, Assembly, and Integration

For component procurement, reference the Bill of Materials in Appendix C.2. STEP, Gerber, and NC Drill files for the frame, L-brackets, and PCBs are available in the GitHub repository. If any changes to the hardware are made, be sure to re-export these files.

For manufacturing, reference Sections 7.1 and 7.3. For assembly and integration, reference Sections 7.2 and 7.4. We do not include instructions on assembling potential payloads; developers must create their own procedures.

To verify that the internal stack has been built correctly, developers should run the IMU and MicroSD sample scripts included in Appendix B. Developers should also verify that the RockBLOCK is installed correctly by running the RockBLOCK sample transmission and receiving script in Appendix B.

8.3 Payload Design and Mission Planning

For mechanical interfaces and budgets (e.g mass, volume, and layout), refer to Section 5.1. Note that payloads can use the sliding backplate as an additional PCB if necessary. For electrical interfaces and budgets (including maximum power consumption), refer to Section 5.2. For flight and orbit environments as well as qualification testing recommendations, refer to Section 5.3. Some sample payload concepts are provided in Section 5.4. If developers find that they need to exceed the provided mass, volume, and power allowances, they must re-verify that they meet the mass and volume requirements and that the PocketQube can have a positive power margin in the desired orbit.

Developers should ensure that the payload does not violate the PocketQube Standard, especially the requirements laid out in Table 1.2. For instance, the payload may not contain any pyrotechnics or toxic, hazardous, or flammable materials (Requirements F-03-02 and F-03-03). No components can be detached from the PocketQube during the mission lifetime (Requirement F-03-01).

In this report, we discussed and analyzed two potential orbits that mission planners can use: a low-earth orbit similar to that of the International Space Station, and a no-eclipse "dawn-dusk" sun-synchronous orbit. The orbital parameters were described in Table 6.9. If mission planners would like to use alternative orbits, they may have to conduct additional analyses and/or testing to ensure that the PocketQube can generate enough power and stay within operating temperature ranges.

Our power and thermal analyses (Sections 4.6 and 6.4, respectively) showed that the "dawn-dusk" sun-synchronous orbit provides the PocketQube with a safe positive power margin and a temperature range within the operating ranges of all its components. We expect PocketQubes to have a mission lifetime of at most one year in this orbit. The ISS orbit is more risky for a PocketQube mission, with very little power margin and a temperature range that is out of the operating range of the battery (but in the operating range of all other components). In this orbit, PocketQubes have a lifetime of about one month. If mission planners would like to operate TigerCub in an ISS orbit, they are encouraged to perform their own power analysis and test as well as thermal tests to ensure that flight units will survive in the ISS orbit for the desired mission lifetime.

As of the time of writing, the only active commercial PocketQube deployer integrator is Alba Orbital. Mission planners should contact Alba Orbital early in the development process if they plan to use an AlbaPod deployer.

There are additionally some open-source PocketQube deployers available, such as Libre Space Foundations's PICOBUS. If mission planners would like to use one of these, the mission planners will be responsible for deployer integration and integration onto the launch vehicle.

8.4 Communication Protocols

TigerCub uses the serial and I2C communication protocols across its internal stack. The MicroSD and IMU breakout boards communicate with the Teensy over I2C, and the RockBLOCK communicates over serial. As described in Section 3.5.2, payloads

have the option of using serial and/or I2C over the PQ10 bus. This section describes both communication protocols and provides some tips for using them.

8.4.1 Serial Communication

Serial protocol is a common type of asynchronous data transfer (i.e. it does not require a synchronized clock signal between the two devices). Serial communication protocol entails data bits, synchronization bits, parity bits, and the Baud rate; it is important to ensure that both devices are using the same protocol [122].

When data is sent, each packet of data is composed of the data, synchronization bits, and parity bits. There are typically 8 bits of data, the start and stop bits (for synchronization), and the optional parity bits, which help perform error checking. Finally, the Baud rate determines how fast the data is sent. Standard Baud rates are 1200, 2400, 4800, 19200, 38400, 57600, and 115200 [122].

As an example, the RockBLOCK uses a 19200 8N1 protocol [123]. This means the Baud rate is 19200, and there are 8 data bits, no parity bits, and 1 stop bit.

A serial bus has two wires: the transmitter, TX, and the receiver, RX. The receiving end of one device should be connected to the transmitting end of the other; that is, RX always goes to TX (and vice versa).

Most serial buses operate using transistor-transistor logic (TTL) [122]. These signals are between 0V and the microcontroller's supply voltage, typically 3.3V or 5V. The RockBLOCK and Teensy 4.0 both operate on a 3.3V logic level.

Finally, be aware that a serial bus is only built for communication between two devices (unlike I2C). Do not attempt to have more than one pair of devices communicating on a single serial bus.

For an example of serial communication in Arduino, reference the RockBLOCK transmission script in Appendix B.1.

8.4.2 I2C Communication

The Inter-Integrated Circuit (I2C) protocol is used for multiple peripheral devices to communicate with a controller device [124]. As opposed to the serial protocol, I2C is synchronous and also allows for more than two devices to communicate over a single bus.

I2C buses have two signals: serial data (SDA) and serial clock (SCL). The clock signal is controlled by the controller device. Each of these signal lines has a pull-up resistor to make the signal high when a device is not pulling it low [124]. Most COTS breakout boards, such as the IMU and MicroSD breakout boards from Adafruit, have pull-up resistors built-in.

Each message sent through an I2C bus has two components: the address frame, which indicates the device the message is sent to, and the data frame(s), which include 8-bit data messages. Data is sent through the SDA line after the SCL line goes low, and data is sampled from the SDA line after the SCL line goes high [124]. There are some additional details about the I2C data transmission procedure, but most devices take care of the internal workings. For more information, developers can refer to the I2C Primer [125].

In terms of logic levels, devices with different logic levels can operate on the same I2C bus, but it is safer to have devices that operate on the same logic level to avoid inadvertent damage [124]. If necessary, a developers can add a level shifter to their devices to bring them up or down to the 3.3V logic level of the Teensy.

For examples of I2C communication in Arduino, reference the IMU and MicroSD breakout board sample code in Appendices B.2 and B.3.

8.5 Qualification Testing

All payloads should undergo qualification testing as described in Section 5.3. We have provided sample test levels from the Falcon 9 Rideshare Payload User's Guide. The PocketQube must be tested after it is fully integrated (i.e. after the payload is assembled into the satellite), but it can be useful to test the payload components

separately first. At minimum, payloads should undergo thermal vacuum bakeout prior to integration to ensure that all organic compounds have outgassed. Developers may also want to perform a sine sweep (especially if the PocketQube has any deployables) as a worksmanship test.

Chapter 9

Conclusions

This chapter begins with a brief project summary. We then verify that we have met the project requirements. Finally, we discuss some future work and project risks.

9.1 Summary

TigerCub is a mission-agnostic PocketQube platform geared towards students and first-time PocketQube developers. Its modular design and pre-built COTS components make it accessible to builders who have little to no experience with picosatellites. In particular, the use of the RockBLOCK 9603 radio module, which communicates with the Iridium satellite network, precludes students from having to design their own complex radio system and allows developers to verify connectivity before launch from the ground.

In this project, we have successfully met our four objectives from Section 1.4. We designed and prototyped a PocketQube following the PocketQube and PQ9 standards, calculated a positive power budget margin, transmitted and received messages using the RockBLOCK, and created a User's Guide for future student reference. As a new PocketQube platform, TigerCub also serves to further the TigerSats Lab's goals of democratizing small satellites.

Some potential payload applications were included in Section 5.4. As Princeton's first PocketQube (including the first-time development of many subsystems on

a PocketQube-scale at Princeton) and as one of the few open-source PocketQube platforms available, we hope TigerCub will allow more students to develop their own missions and join the small community of PocketQube builders.

9.2 Verification of Requirements

The following tables describe TigerCub's compliance with the project requirements.

#	Met?	Rationale	Section
F-01	Yes	Sliding backplate contains two kill switches.	4.3
F-02	Yes	Kill switches interrupt 5V power from EPS to OBC1.	4.3
F-03	Yes	See Table 9.2.	–
F-04	Yes	See Table 9.3.	–
P-01	Yes	Positive power margins. [1]	4.6
P-02	Yes	Payload interfaces and budgets provided.	5.1, 5.2
P-03	Yes	Communication through RockBLOCK 9603.	2.2
P-04	Yes	Provided by Teensy, IMU, and MicroSD.	3.3
C-01	Yes	Frame is $50 \times 50 \times 50$ mm.	6.1.1
C-02	Yes	Sliding backplate is $58 \times 64 \times 1.6$ mm.	6.1.2
C-03	Yes	Maximum component extension (solar panels) is less than 7 mm.[2]	6.1.3
C-04	Yes	Total mass is 239.6 grams.	6.3
C-05	Yes	Center of mass is $(-3, 1.1, 0.3)$ mm from the geometric centroid.	6.3
C-06	Yes	Total cost of materials is 1184.62 USD.	C.2
E-01	Yes	TML of all components is less than 1.0%.	6.5.1
E-02	Yes	CVCM of all components is less than 0.1%.	6.5.1
E-03	Yes	Positive stress margin in load and vibration analysis.	6.2
E-04	Yes	COTS components will be sufficient for a short mission in LEO.[3]	6.5.2
E-05	Partially	Internal stack has a common ground but is not grounded to frame.	6.5.3
E-06	Yes	Satellite operates within temperature range in a sun-synchronous orbit.	6.4

Table 9.1: Project-level requirements verification. Refer to Table 1.1 for the list of requirements.

#	Met?	Rationale	Section
F-03-01	Yes	No components detach from the satellite.	–
F-03-02	Yes	There are no pyrotechnics on the satellite.	–
F-03-03	Yes	There are no toxic, hazardous, or flammable materials.	–
F-03-04	Yes	There are no metallic materials in contact with the deployer.	6.1.2

Table 9.2: PocketQube Standard requirements verification. Refer to Table 1.2 for the list of requirements.

#	Met?	Rationale	Section
F-04-01	Yes	All boards are $42 \times 42 \times 1.6$ mm.	3.5, 3.6, 5.1
F-04-02	Yes	Mounting holes provided on all boards.	3.5, 3.6, 5.1
F-04-03	Yes	PQ10 connector positioned as required.	3.5, 3.6, 5.1
F-04-04	Yes	All stacking connectors are in the Samtec SQT-1XX-XX-L-S series.	C.2
F-04-05	Yes	There is 1 mm margin between PCBs.	6.1.3

Table 9.3: PQ9 Standard requirements verification. Refer to Table 1.3 for the list of requirements.

We have satisfied all requirements except Requirement E-05 through the design, and analysis, and test of TigerCub's systems. We will discuss spacecraft charging mitigations in Section 9.3.4.

9.3 Future Work

Some work still needs to be completed for TigerCub to be a full-fledged PocketQube. This section outlines five areas of future work.

[1]Additional verification of solar panel power generation capabilities should be completed. Refer to Section 9.3.2 for details.

[2]We measured the maximum component extension to be 5.5 mm.

[3]Though not all components are radiation-hardened, our components provide sufficient radiation shielding for a high-risk, low-significance, and short mission in LEO as described in 6.5.2.

9.3.1 Software Development

This project has mostly focused on the hardware design and verification of PocketQube subsystems. Due to a lack of experience in low-level programming and hardware-in-the-loop (HITL) software, we have left most of the software development to a future student.

First, software needs to be developed for the EPS module. As previously described in Section 4.2, the DynOSSAT-EDU EPS does not come pre-loaded with software since we did not buy it directly from the developer and the developer has not released sample code for the EPS. To use the EPS, we must write and upload code to the SAMD21 MCU through the SWD header on the board. The software must perform many of the key functions (for which hardware already exists on the EPS board):

- enable and disable power to the rest of the internal stack,
- manage charging and discharging from the solar cells and battery,
- provide telemetry on battery state of charge, and
- serve as a watchdog for the EPS and the Teensy.

Software will also need to be developed for on-orbit operations, including startup and nominal operating modes. Some of this code is included in this report and its appendices, but the code must be aggregated into a series of scripts for the Teensy 4.0 to run once in orbit.

Once software has been written, developers can perform verification tests of the satellite power budget, including the solar cells' power generation capabilities and the satellite's power consumption. DynOSSAT has provided some examples of code for the EPS on their website [126].

9.3.2 Verification Testing

After software development occurs, some components and subsystems should be tested to verify that they function and perform as expected. In particular, the following should be verified:

- Verify that the EPS can provide 5V and 3.3V power throughout the internal stack from USB input, the battery, and the solar cells.

- Verify the power consumption of individual components when powered by the EPS during nominal operations.

- Verify solar panel power generation. Developers can use a sun simulator, such as the TigerSats Lab's solar charging testbed [127]. Developers can also determine how much power the solar panels can generate in a typical LEO orbit as opposed to the no-eclipse sun synchronous orbit proposed in Section 6.4.

9.3.3 Qualification Testing

Before launch, payload integrators should conduct qualification testing to ensure that the PocketQube will survive launch and operating environments. The NASA General Environmental Verification Standard describes the tests that spacecraft should undergo prior to launch [94]. Picosatellites are generally exempt from some of these tests due to their small size and (typically) high-risk missions. However, to verify the analysis in this report, the following tests, at minimum, should be conducted:

- Sine sweep vibration testing: Sine sweep testing qualifies satellites for the low-frequency sine environments during flight and serves as a worksmanship test (to ensure that bolts or wiring harnesses are not loose) [94]. The PocketQube should be constrained in the same configuration as it will be during launch, using the tabs on the sliding backplate.

- Random vibration testing: Satellites are subjected to random vibration (non-deterministic motion) during lift-off, typically from the firing of the launch vehicle engines and aerodynamic turbulence. Spacecraft must verify they can survive such an environment through random vibration tests, in which the spacecraft is subjected to random vibration simulating what the satellite might see during launch [94]. As with the sine sweep, payloads should be tested in the same configuration that they will launch in.

- Thermal vacuum cycling: Thermal cycling tests ensure that the satellite can operate in temperature ranges similar to those they will experience in orbit. Spacecraft should undergo thermal cycling in a vacuum chamber and should be performance-tested before and after the test to ensure that there are no adverse effects [94].

- Thermal vacuum bakeout: Flight units undergo thermal vacuum bakeout to remove as many outgassing compounds as possible before launch. Subsystems should be heated to the highest temperature possible without damaging any components. [94].

9.3.4 Making TigerCub Space-Worthy

Before TigerCub is ready for launch, there are a few additional design changes that could make TigerCub more space-worthy. These will mitigate some of the risks of flying COTS components.

Replacing RockBLOCK Supercapacitors

The RockBLOCK has two large electrolytic supercapacitors [128] that hold enough charge for the RockBLOCK to make transmissions without drawing a high current from the battery. However, electrolytic capacitors are generally not recommended for space applications due to outgassing concerns. Instead, ceramic or tantalum capacitors are typically used.

There is some evidence that this particular line of supercapacitors could be space-qualified [129]. However, to mitigate the risks of using electrolytic capacitors, future work should include replacing the supercapacitors on the RockBLOCK with several flight-worthy capacitors of equivalent capacitance.

Adding Solar Cell Cover Glass

As mentioned in Chapter 4, cover glass is commonly used to cover solar cells on larger spacecraft. Considering TigerCub's limited power budget, cover glass could slow the

degradation of solar cells and potentially give TigerCub a longer mission lifetime.

Cover glass is typically custom-manufactured by vendors such as Excelitas [130]. Future developers can contact Excelitas for a quote on cover glass for TigerCub's solar cells.

Securing and Insulating the Battery

In the current design, the battery is not secured inside the satellite, though it is jammed tightly between the internal stack and a solar panel. For greater security, the battery should be held down with a cable tie; the cable tie can be secured around the spacers in the internal stack.

Most commercial cable ties violate spacecraft outgassing requirements. However, there are a few cable ties that are designed for harsh environments (radiation, UV light, extreme temperatures, etc.) that are suitable for space and have flight heritage. The Pan-Ty PLT1M-C71 cable ties are made of polyether ether ketone (PEEK), which is a cable tie material commonly used by commercial spacecraft to secure wire harnesses [131]. Similarly, the Pan-Ty PLT1M-C76 cable ties are made of Tefzel and will be used on the upcoming Interstellar Mapping and Acceleration Probe (IMAP) mission that the Space Physics at Princeton Group is currently working on [132].

Future developers may also want to add additional insulation to the battery. This would protect the battery against the cold temperatures in an ISS orbit and increase the battery's cycle life in such an orbit, as discussed in Section 6.4. There is currently not much space inside the spacecraft for additional insulation. Future developers could replace the battery with a smaller one to provide extra volume for insulation. Developers could also consider adding a small heater to the battery, though this would consume additional power.

Protecting Against Spacecraft Charging

In addition to these component modifications, additional changes need to be made to make TigerCub more resistant to the effects of spacecraft charging as mentioned in Section 6.5.3. A relatively simple grounding scheme (for a relatively simple spacecraft)

is to connect all signal and power grounds to a single common ground plane, usually the aluminum honeycomb structure for larger spacecraft (and in TigerCub's case, the aluminum frame). For this method, the ground on all boards in the internal stack should additionally be grounded to the four mounting holes. The sliding backplate should include references to ground at the four mounting holes for the internal stack, which should then also connect to the four mounting holes for the aluminum frame.

9.3.5 Generating More Power

Potential missions are primarily limited by TigerCub's limited power budget. To expand TigerCub's power generation capabilities, future developers could integrate deployable, actuate-able solar panels. Constantly pointing the solar panels towards the sun would at least double TigerCub's power generation; however, there may be a difficult trade between power generation versus mass and volume.

As an alternative, future developers could upgrade TigerCub's solar cells to more efficient ones, such as space-grade gallium arsenide triple-junction cells from TrisolX [133], which have previously been used in the TigerSats Lab on a CubeSat solar panel. These cells are slightly more efficient than the monocrystalline silicon cells currently used on TigerCub, but are much more difficult to assemble for the first-time PocketQube developer.

9.4 Project Risks

There are some inherent project risks due to untested components that could cause potential missions to fail. First, there are no redundant battery cells. If the single battery fails, the PocketQube would likely be unable to survive an eclipse period in an ISS orbit, though the PocketQube could survive in a sun-synchronous orbit. In addition, the DynOSSAT-EDU EPS and RockBLOCK 9603 both have no flight heritage and are unproven in the space environment. Without flight time, it is difficult to know if these components will survive in space.

That being said, TigerCub has the potential to prove that the RockBLOCK 9603

and other COTS components can be used to build an accessible, low-cost PocketQube. As one of the first open-source educational PocketQube platforms, we hope TigerCub will help open up this class of picosatellites to first-time satellite developers.

Bibliography

[1] Jamie Groh. *What are SmallSats and CubeSats?* NASA. 2024. URL: https://www.nasa.gov/what-are-smallsats-and-cubesats.

[2] Bob Twiggs. *Making it Small*. Cal Poly Developer's Workshop. 2009. URL: https://web.archive.org/web/20160303185449/http://mstl.atl.calpoly.edu/~bklofas/Presentations/DevelopersWorkshop2009/7_CubeSat_Alt/1_Twiggs-PocketQub.pdf.

[3] S. Radu et al. *PocketQube Standard*. 2018. URL: https://static1.squarespace.com/static/53d7dcdce4b07a1cdbbc08a4/t/5b34c395352f5303fcec6f45/1530184648111/PocketQube+Standard+issue+1+-+Published.pdf.

[4] *PocketQube (PocketQub)*. Gunter's Space Page. URL: https://space.skyrocket.de/doc_sat/pocketqub.htm.

[5] Neil Khera. "PyCubed-Mini: A Low-Cost, Open-Source Satellite Research Platform". MA thesis. Pittsburgh, PA: Carnegie Mellon University, 2023.

[6] *PyCubed*. URL: https://pycubed.org/.

[7] *Alba Orbital*. URL: https://www.albaorbital.com/.

[8] *Nightlights Early Access*. Alba Orbital. URL: https://www.albaorbital.com/nightlights-early-access.

[9] *Pricing*. Alba Orbital. URL: https://www.albaorbital.com/unicorn-pricing.

[10] *CubeSat Mission Assurance Trends*. NEPP Electronics Technology Workshop. Saint Louis College. 2020. URL: https://nepp.nasa.gov/docs/etw/2020/18-JUN-THU/1300-Swartwout-NEPP-ETW-CubeSat-v2.pdf.

[11] Jasper Bowmeester. *PQ9 and CS14 Electrical and Mechanical Subsystem Interface Standard for PocketQubes and CubeSats*. 2018. URL: https://doi.org/10.34894/6MVBCZ.

[12] *Iridium 9603*. SBDN9603. Iridium Satellite LLC. 2020.

[13] Vincent J. Riot, Lance M. Simms, and Darrell Carter. "Lessons Learned Using Iridium to Communicate with a CubeSat in Low Earth Orbit". In: *Journal of Small Satellites* 10 (1 2021), pp. 995–1006.

[14] *EyeStar-S4: Iridium Enabled Sat-to-Sat Radio*. NearSpace Launch. URL: https://nearspacelaunch.com/eye-star/.

[15] *RockBLOCK 9603N - Iridium SatComm Module*. SparkFun. URL: https://www.sparkfun.com/products/14498.

[16] *Dimensions*. Rock7. URL: https://docs.rockblock.rock7.com/docs/dimensions.

[17] *Connectors*. Rock7. URL: https://docs.rockblock.rock7.com/docs/connectors.

[18] *Environmental Specification*. Rock7. URL: https://docs.rockblock.rock7.com/docs/environmental-specification.

[19] *RockBLOCK 9603 - Power Consumption*. Rock7. URL: https://docs.rockblock.rock7.com/docs/power-consumption-guidance.

[20] *Connectors/Wiring*. Rock7. URL: https://docs.rockblock.rock7.com/docs/connectors.

[21] *The RockBLOCK data cycle*. Rock7. URL: https://docs.rockblock.rock7.com/docs/the-rockblock-data-cycle.

[22] Jeremy Lavine and Peter Laird. *ISU AT Command Reference*. MAN0009. Version 2. Iridium Satellite LLC. 2023. URL: https://www.groundcontrol.com/us/wp-content/uploads/sites/4/2022/02/IRDM_ISU_ATCommandReferenceMAN0009_Rev2.0_ATCOMM_Oct2012.pdf.

[23] Carter Nelson. *Using the RockBLOCK Iridium Modem*. Adafruit Industries. 2024. URL: https://learn.adafruit.com/using-the-rockblock-iridium-modem?view=all.

[24] *Transmit ASCII data*. Rock7. URL: https://docs.rockblock.rock7.com/docs/transmit-ascii-data.

[25] *RockBLOCK*. Rock7. URL: https://rockblock.rock7.com/.

[26] Paul Clark and Mikal Hart. *SparkFun IridiumSBD I2C Arduino Library*. GitHub. 2023. URL: https://github.com/sparkfun/SparkFun_Iridium SBD_I2C_Arduino_Library.

[27] *Teensy 3.2 Development Board*. PJRC. URL: https://www.pjrc.com/store/teensy32.html.

[28] *Teensy 4.0 Development Board*. PJRC. URL: https://www.pjrc.com/store/teensy40.html.

[29] *Adafruit Trinket M0 - for use with CircuitPython and Arduino IDE*. Adafruit Industries. URL: https://www.adafruit.com/product/3500.

[30] *Pro Micro - 5V/16Mhz*. SparkFun. URL: https://www.sparkfun.com/products/12640.

[31] *Disappointing and inexplicable Teensy 4.0 24mhz high power consumption compared to 3x*. PJRC Forum. 2019. URL: https://forum.pjrc.com/index.php?threads/disappointing-and-inexplicable-teensy-4-0-24mhz-high-power-consumption-compared-to-3x.57450/#post-214313.

[32] *How to change clock speed on Teensy 4.0?* PJRC Forum. 2019. URL: https://forum.pjrc.com/index.php?threads/how-to-change-clock-speed-on-teensy-4-0.57444/.

[33] *Teensy 4.0 Low Power*. PJRC Forum. 2020. URL: https://forum.pjrc.com/index.php?threads/teensy-4-0-low-power.57885/.

[34] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Attitude Determination and Control. Microcosm, 2005. Chap. 11.1, pp. 354–380.

[35] Liz Clark. *Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU*. Adafruit Industries. 2024. URL: https://learn.adafruit.com/adafruit-lsm6ds3tr-c-lis3mdl-precision-9-dof-imu/lsm6ds3tr-c-lis3mdl-pinouts.

[36] Jeff Epler. *Adafruit MicroSD SPI or SDIO Card Breakout Board*. Adafruit Industries. 2024. URL: https://learn.adafruit.com/adafruit-microsd-spi-sdio/pinouts.

[37] Eric Becnel et al. *PQ60 Standard Document*. Version 1. 2015. URL: https://nanopdf.com/download/pq-60-standard-document-version-release-date-11_pdf.

[38] Martha O'Bryan. *Single Event Effects*. NASA GSFC. 2021. URL: https://radhome.gsfc.nasa.gov/radhome/see.htm.

[39] *Single Event Effects Specification (Draft)*. NASA GSFC. URL: https://nepp.nasa.gov/DocUploads/074CCC9D-51FB-4323-AB81F47E8F06FFDB/DraftSingleEventEffectSpecification.pdf.

[40] Martha O'Bryan. *Natural Space Radiation Effects on Technology*. NASA GSFC. 2021. URL: https://radhome.gsfc.nasa.gov/radhome/Nat_Space_Rad_Tech.htm.

[41] Kelvin Odom. *What is a Watchdog Timer and Why is it Important?* Texas Instruments. URL: https://www.ti.com/document-viewer/lit/html/SSZTAH7.

[42] Antonio Brewer and Sam Nobs. *WDT_T4. Watchdog for Teensy 4*. GitHub repository. URL: https://github.com/tonton81/WDT_T4.

[43] *WDT_T4 - Watchdog Library for Teensy 4*. PJRC Forum. 2020. URL: https://forum.pjrc.com/index.php?threads/wdt_t4-watchdog-library-for-teensy-4.59257/.

[44] Eugenio Pace. *Using a Watchdog to fix all issues*. 2020. URL: https://eugeniopace.org/arduino/resiliency/2020/05/11/Using-a-Watchdog-to-fix-all-issues.html.

[45] Enrique Casado. *DynOSSAT-EDU EPS*. GitHub repository. 2020. URL: https://github.com/BHDynamics/dynossat-edu-eps.

[46] Robert K. *PQ60 - EPS*. Hackaday. 2015. URL: https://hackaday.io/project/4591/logs.

[47] *DynoSSAT-EDU*. Tindie. URL: https://www.tindie.com/products/bhdynamics/dynossat-edu/.

[48] Enrique Casado. *DynOSSAT-EDU User Guide*. Github. URL: https://github.com/BHDynamics/dynossat-edu-eps/wiki/User-Guide.

[49] Ted Mortenson. *Limit Switch Explained — Working Principles*. RealPars. 2020. URL: https://www.realpars.com/blog/limit-switch.

[50] *AlbaPod Interface Control Document*. Alba Orbital. URL: https://static1.squarespace.com/static/53d7dcdce4b07a1cdbbc08a4/t/607eb95a8453342db546143f/1618917746906/AlbaPod+ICD+v.02+.

[51] *D2F Ultra Subminiature Basic Switch*. B036-E1-12. Omron. 2021.

[52] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Power Sources. Microcosm, 2005. Chap. 11.4.1, pp. 409–418.

[53] *Princeton Cubesat Kit*. TigerSats Lab. URL: https://tigersats.princeton.edu/cubesatkit.

[54] *Pleiades Rapid Orbital Verification Experimental System (PROVES)*. Bronco Space. URL: https://www.broncospace.com/proves.

[55] *IXOLAR Thin 1.2 mm, 25% Efficiency Solar Cells*. DigiKey. URL: https://www.digikey.com/en/product-highlight/a/anysolar/ixolar-thin-1-2mm-25-efficiency-solar-cells.

[56] *IXOLAR High Efficiency SolarMD*. SM940K09L. ANYSOLAR LTD. 2020.

[57] *Anti-Reflection Coated Cover Glasses*. European Space Agency. 2012. URL: https://connectivity.esa.int/projects/antireflection-coated-cover-glasses.

[58] *Bypass Diodes in Solar Panels*. Electronics Tutorials. URL: https://www.electronics-tutorials.ws/diode/bypass-diodes.html.

[59] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Energy Storage. Microcosm, 2005. Chap. 11.4.2, pp. 418–422.

[60] *3.7V 702535 600mAh Lithium Polymer ion Battery Rechargeable Polymer Battery Pack with JST 2.0mm Connector*. Amazon. URL: https://www.amazon.com/702535-Lithium-Polymer-Rechargeable-Connector/dp/B0C2KQJDWP.

[61] Anuja Magdum and Michael Galvin. *A Novel Solar Charging Testbed for CubeSats*. Tech. rep. Princeton University TigerSats Lab, 2023.

[62] Craig Clark and Ritchie Logan. *Power Budgets for Mission Success*. Clyde Space. 2011.

[63] *Introduction to Solar Radiation*. Newport Corporation. URL: https://www.newport.com/t/introduction-to-solar-radiation.

[64] Joe Troutman. *Satellite Electrical Power System Design*. Forge Nano. 2024.

[65] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Power Regulation and Control. Microcosm, 2005. Chap. 11.4.4.

[66] *iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope*. LSM6-DS3TR-C. Version 3. STMicroelectronics. 2017.

[67] *Digital output magnetic sensor: ultralow-power, high-performance 3-axis magnetometer*. LIS3MDL. Version 7. STMicroelectronics. 2023.

[68] *SAMD21 power consumption too high in ACTIVE mode*. 2020. URL: https://forum.arduino.cc/t/samd21-power-consumption-too-high-in-active-mode/668966.

[69] Wiley J. Larson and James R. Wertz. *Space Mission Analysis and Design*. 3rd ed. Microcosm, 2005.

[70] *3.7 V 600mAh Lithium Ion Battery 702535*. UFine Battery. URL: https://www.ufinebattery.com/products/3-7-v-600mah-lithium-ion-battery-702535/.

[71] *702535 3.7v 600mah rechargeable battery pack*. Vats Battery. URL: https://www.vatsbattery.com/product/702535-3-7v-600mah-rechargeable-battery-pack/.

[72] J. Bouwmeester et al. "Utility and constraints of PocketQubes". In: *CEAS Space Journal* 12 (2020), pp. 573–586.

[73] *Rideshare Payload User's Guide*. Version 9. SpaceX. 2023. URL: https://storage.googleapis.com/rideshare-static/Rideshare_Payload_Users_Guide.pdf.

[74] Gunter D. Krebs. *Wren*. Gunter's Space Page. 2023. URL: https://space.skyrocket.de/doc_sdat/wren.htm.

[75] Gunter D. Krebs. *TRSI-Sat (TRSI 1), TRSI 2, 3*. Gunter's Space Page. 2023. URL: https://space.skyrocket.de/doc_sdat/trsi-sat.htm.

[76] *SMOG-1*. BME-GND. URL: https://gnd.bme.hu/smog.

[77] *Alba Orbital's First Earth Observation pico-satellite phones home*. Alba Orbital. 2022. URL: https://www.albaorbital.com/unicorn2-phone-home.

[78] *Pikocamera Camera for PocketQube*. Orion Space. URL: https://www.tindie.com/products/orionspace/pikocamera-camera-for-pocketqube/.

[79] OV02C. Version 1.1. OmniVision Technologies. 2021. URL: https://www.ovt.com/wp-content/uploads/2022/11/OV02C-PB-v1.1-WEB.pdf.

[80] *What is TWI? How to Configure the TWI for I2C Communication*. Microchip Technology Inc. 2018. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/90003181A.pdf.

[81] *PocketQube Satellite Scans the Atmosphere for Electrosmog Pollution*. MathWorks. URL: https://www.mathworks.com/company/mathworks-stories/pocketqube-satellite-scans-atmosphere-for-electrosmog-pollution.html.

[82] *6-Channel NIR Spectral_ID Device with Electronic Shutter and Smart Interface*. AS7263. Version 4. ams OSRAM Group. 2022.

[83] *SparkFun Spectral Sensor Breakout - AS7263 NIR (Qwiic)*. SparkFun. URL: https://www.sparkfun.com/products/14351.

[84] *Six space-proven PocketCube mission ideas*. SatNews. 2023. URL: http://www.satmagazine.com/story.php?number=1780991041.

[85] Alicia Johnstone. *CubeSat Development Specification*. CP-CDS-R14.1. Version 14.1. California Polytechnic – San Luis Obispo. 2022. URL: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779/CDS+REV14_1+2022-02-09.pdf.

[86] *Sheet Metal Fabrication vs. CNC Machining*. Sweeney Metal Fabricators. 2022. URL: https://www.sweeneymetal.com/blog/sheet-metal-fabrication-vs-cnc-machining.

[87] *Aluminum 5052-H32*. MatWeb. URL: https://www.matweb.com/search/DataSheet.aspx?MatGUID=96d768abc51e4157a1b8f95856c49028&ckck=1.

[88] *Sheet Metal Bending: The Basics*. Xometry. URL: https://www.xometry.com/resources/machining/the-basics-of-bending-sheet-metal.

[89] *Payload User's Guide*. Firefly Aerospace. 2023. URL: https://fireflyspace.com/wp-content/uploads/2023/08/Alpha-PUG-4.0.pdf.

[90] *Payload User's Guide*. Version 7.0. Rocket Lab. 2022. URL: https://www.rocketlabusa.com/assets/Uploads/Electron-Payload-User-Guide-7.0.pdf.

[91] *Aluminum 6061-T6; 6061-T651*. MatWeb. URL: https://www.matweb.com/search/DataSheet.aspx?MatGUID=b8d536e0b9b54bd7b69e4124d8f1d20a&ckck=1.

[92] *304 Stainless Steel*. MatWeb. URL: https://www.matweb.com/search/DataSheet.aspx?MatGUID=abc4415b0f8b490387e3c922237098da.

[93] *Glass Epoxy (G10, FR4) Characteristics*. Dielectric Manufacturing. URL: https://dielectricmfg.com/resources/knowledge-base/glass-epoxy/.

[94] *General Environmental Verification Standard (GEVS) for GSFC Flight Programs and Projects*. NASA Goddard Space Flight Center. 2021. URL: https://standards.nasa.gov/sites/default/files/standards/GSFC/B/0/gsfc-std-7000b_signature_cycle_04_28_2021_fixed_links.pdf.

[95] *Mode Participation Factor and Effective Mass. Modal Analysis – Lesson 4*. Ansys. URL: https://courses.ansys.com/wp-content/uploads/

2020/10/Lesson4_ModeParticipationFactorAndEffectiveMass.pdf.

[96] Ryan Simmons. *Miles' Equation*. NASA Goddard Space Flight Center. 2001. URL: https://femci.gsfc.nasa.gov/random/MilesEqn.html.

[97] *Miles' Equation for Vibration*. Engineers Edge. URL: https://www.engineersedge.com/material_science/vibration_miles_equation.htm.

[98] *Types of orbits*. European Space Agency. 2020. URL: https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits.

[99] *Alba Orbital Unicorn PocketQubes*. eoPortal. 2023. URL: https://www.eoportal.org/satellite-missions/alba-orbital#launches.

[100] *ISS: International Space Station*. European Space Agency. URL: https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/International_Space_Station/ISS_International_Space_Station.

[101] *Why you should not charge a Lithium battery below 32 degrees*. REDARC. URL: https://www.redarcelectronics.com/us/resources/chargers-isolators-faqs/do-not-charge-lithium-battery-below-32-degrees.

[102] *SAM D21/DA1 Family*. DS40001882F. Microchip Technology. 2020.

[103] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Spacecraft Thermal Environment. Microcosm, 2005. Chap. 11.5.1, pp. 431–434.

[104] Vincent L. Pisacane. In: *Fundamentals of Space Systems*. 2nd ed. Radiation Properties. Oxford University Press, 2005. Chap. 7.9.1, pp. 432–434.

[105] Wiley J. Larson and James R. Wertz. In: *Space Mission Analysis and Design*. 3rd ed. Thermal Control Components. Microcosm, 2005. Chap. 11.5.2, pp. 434–446.

[106] J.H. Henninger. *Solar absorptance and thermal emittance of some common spacecraft thermal-control coatings*. Tech. rep. NASA-RP-1121. NASA Goddard Space Flight Center, 1984.

[107] Jukka Rantala. *Emissivity in Practical Temperature Measurements*. Electronics Cooling. 2003. URL: https://www.electronics-cooling.com/2003/08/e-missivity-in-ppractical-temperature-measurements/.

[108] Isidoro Martinez. *Radiative View Factors*. URL: http://imartinez.etsiae.upm.es/~isidoro/tc3/Radiation%20View%20factors.pdf.

[109] *Average area of the shadow of a convex shape*. Math StackExchange. 2019. URL: https://math.stackexchange.com/questions/3222317/average-area-of-the-shadow-of-a-convex-shape.

[110] Vincent L. Pisacane. In: *The Space Environment and Its Effects on Space Systems*. 2nd ed. Material Outgassing. American Institute of Aeronautics and Astronautics, 2016. Chap. 10.2, pp. 296–299.

[111] *Outgassing Data for Selecting Spacecraft Materials*. NASA Goddard Space Flight Center. URL: https://outgassing.nasa.gov/outgassing-data-table.

[112] Robert F. Hodson et al. *Recommendations on the Use of Commercial-Off-The-Shelf (COTS) Electrical, Electronic, and Electromechanical (EEE) Parts for NASA Missions*. Tech. rep. NESC-RP-19-01490. NASA Engineering and Safety Center, 2022.

[113] Gregory Robinson. *Class-D Spacecraft Risk Classifications Streamlining*. Tech. rep. NASA Science Mission Directorate, 2017.

[114] *Space Radiation Effects on Electronic Components in Low-Earth Orbit*. Tech. rep. NASA Johnson Space Center, 1999. URL: https://llis.nasa.gov/lesson/824.

[115] Vincent L. Pisacane. In: *Fundamentals of Space Systems*. 2nd ed. The Ionosphere. Oxford University Press, 2005. Chap. 2.4, pp. 62–74.

[116] *PCB Prototype the Easy Way*. PCBWay. URL: https://www.pcbway.com/.

[117] *How to Generate Gerber and Drill Files in KiCad 7.0*. PCBWay. 2023. URL: https://www.pcbway.com/blog/help_center/How_to_Generate_Gerber_and_Drill_Files_in_KiCad_7_0_ab0d12bb.html.

[118] *PCB Manufacturing and Assembly Capabilities*. JLCPCB. URL: https://jlcpcb.com/capabilities/pcb-capabilities.

[119] *PCBA Files Preparation*. JLCPCB. URL: https://jlcpcb.com/help/catalog/190-PCBA-Files-Preparation.

[120] *V-One*. Voltera. URL: https://www.voltera.io/v-one.

[121] *Mechanical Fastener Torque Guidelines*. Directive 540-PG-8072.1.2B. NASA Mechanical Systems Division. 2020.

[122] *Serial Communication*. SparkFun. URL: https://learn.sparkfun.com/tutorials/serial-communication/all.

[123] *Serial Interface*. Rock7. URL: https://docs.rockblock.rock7.com/docs/serial-interface.

[124] *I2C*. SparkFun. URL: https://learn.sparkfun.com/tutorials/i2c.

[125] *I2C Primer*. I2C Bus. URL: https://www.i2c-bus.org/i2c-primer/.

[126] *Code Examples*. BHDynamics. URL: https://sites.google.com/bhdyn.com/dynossat/examples?authuser=0.

[127] *Testbeds*. Princeton University TigerSats Lab. URL: https://tigersats.princeton.edu/simulation/testbeds/solar-charging-testbed.

[128] *Datasheet - 2.7V / 5F Cell*. ESHSR-0005C0-002R7. Version 14. Nesscap Ultracapacitors. 2014.

[129] Géraldine Palissat, Leo Farhat, and Joaquín José Jiménez Carreira. *Supercapacitors for space applications: trends and opportunities*. Tech. rep. European Space Agency, 2023.

[130] *Space-Qualifed Cover Glass*. Excelitas Technologies. URL: https://www.excelitas.com/product/space-qualified-cover-glass.

[131] *Pan-Ty PLT1M-C71 Cable Tie, Brown, PEEK, Radiation, High Heat, 4"L, 35lb, PK100*. URL: https://www.panduit.com/en/products/wire-routing-management-protection/cable-wire-ties-mounts-straps/cable-wire-ties/plt1m-c71.html.

[132] *Pan-Ty PLT1M-C76 Cable Tie, Aqua, Tefzel, UV, High Heat, 4"L, 18lb, PK100*. Panduit. URL: https://www.panduit.com/en/products/wire-routing-management-protection/cable-wire-ties-mounts-straps/cable-wire-ties/plt1m-c76.html.

[133] *Services*. TrisolX. URL: https://trisolx.com/services.html.

[134] David A. Mellis and Tom Igoe. *ReadWrite.ino*. 2020. URL: https://github.com/arduino-libraries/SD/blob/master/examples/ReadWrite/ReadWrite.ino.

[135] *DuPont Kapton 500VN Polyimide Film*. MatWeb. URL: https://www.matweb.com/search/datasheet_print.aspx?matguid=338573ad1bdf4586aa17fab95f3a57d7.

[136] *PICOBUS*. Libre Space Foundation. URL: https://libre.space/projects/picobus/.

Appendix A

Reference Material

A.1 Mechanical Drawings

For reference, we include dimensioned drawings of custom-designed parts as well as dimensioned drawings of the kill switches.

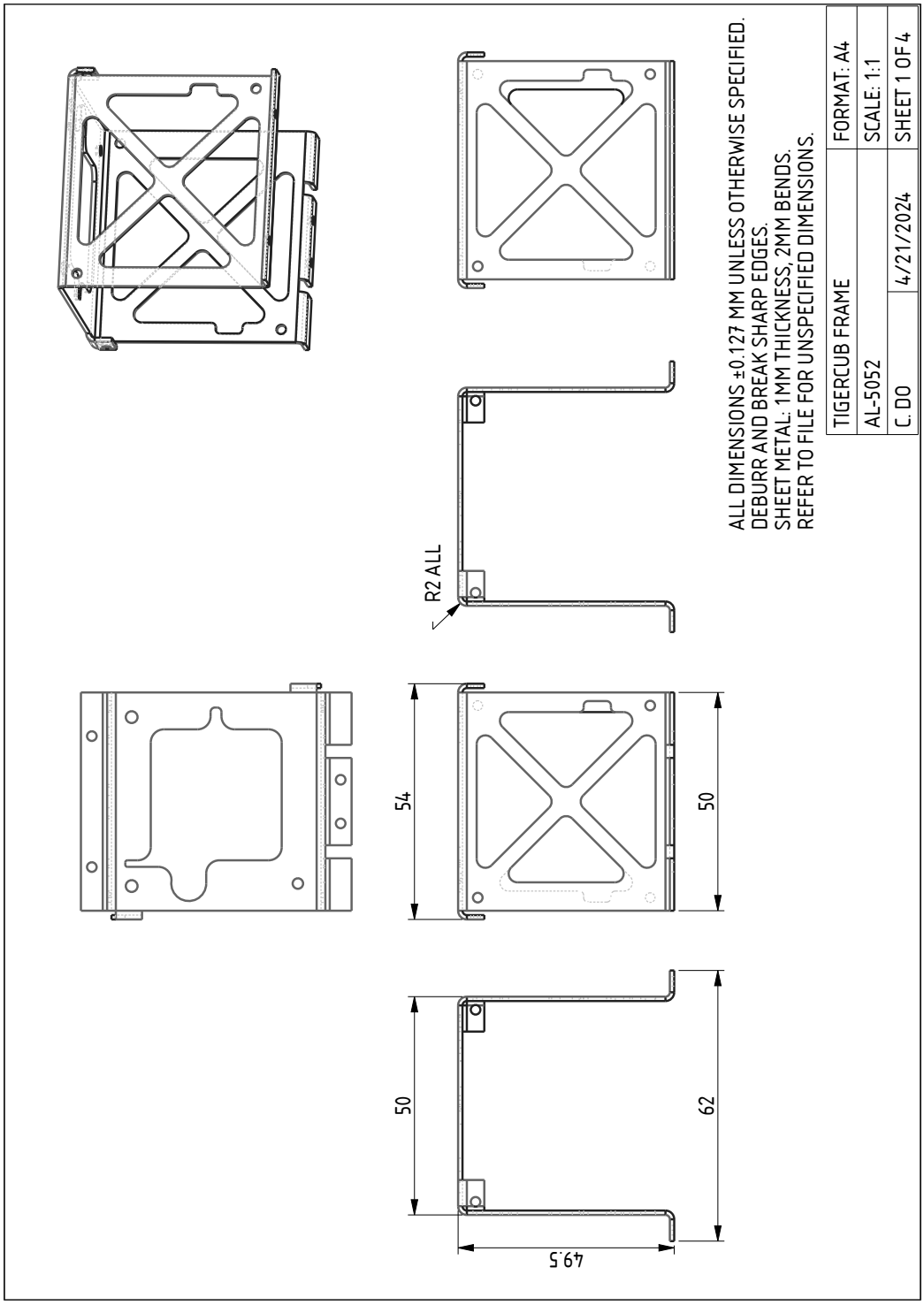


Figure A.1: Dimensioned drawing of sheet-metal frame, sheet 1 of 4.

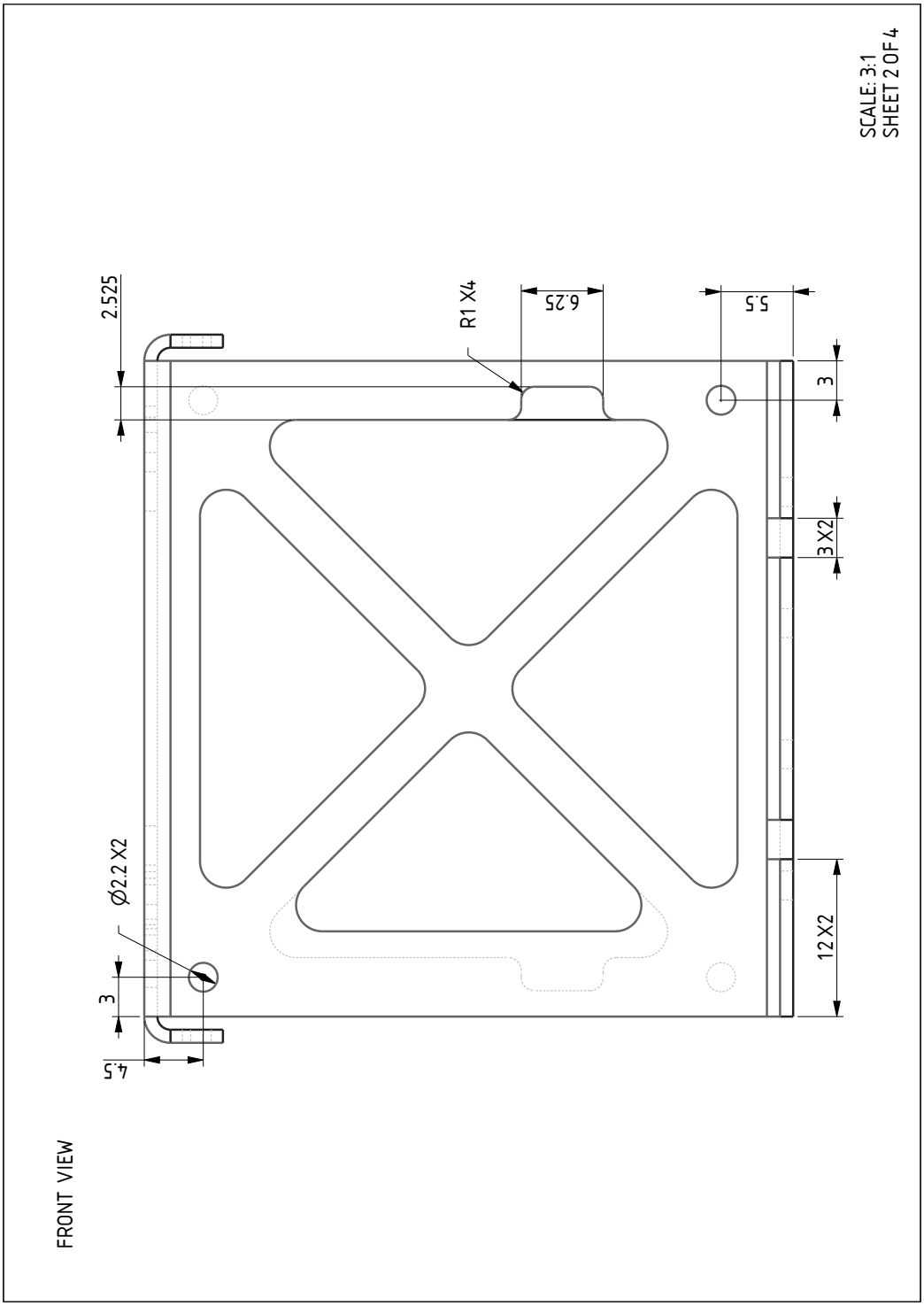


Figure A.2: Dimensioned drawing of sheet-metal frame, sheet 2 of 4.

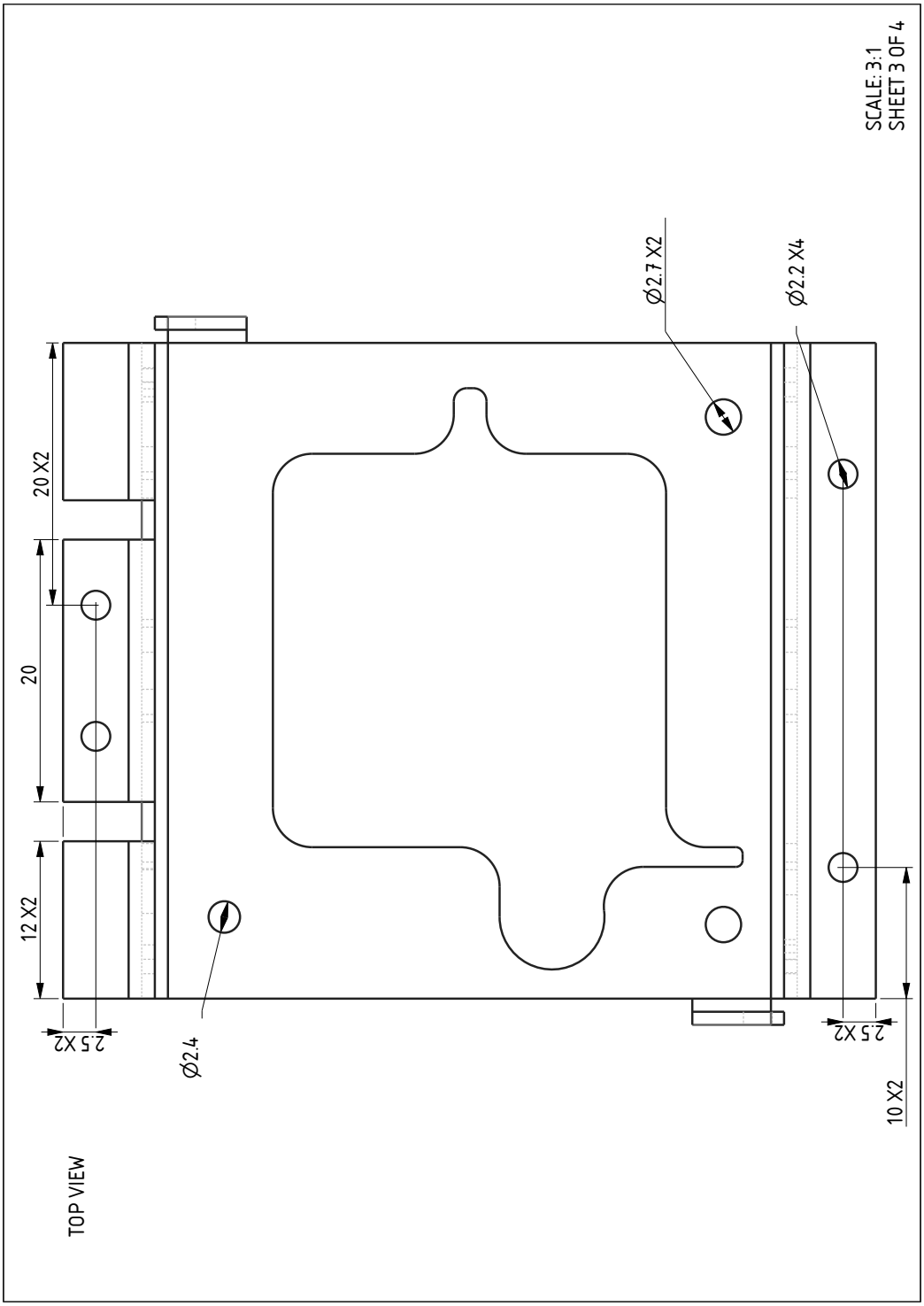


Figure A.3: Dimensioned drawing of sheet-metal frame, sheet 3 of 4.

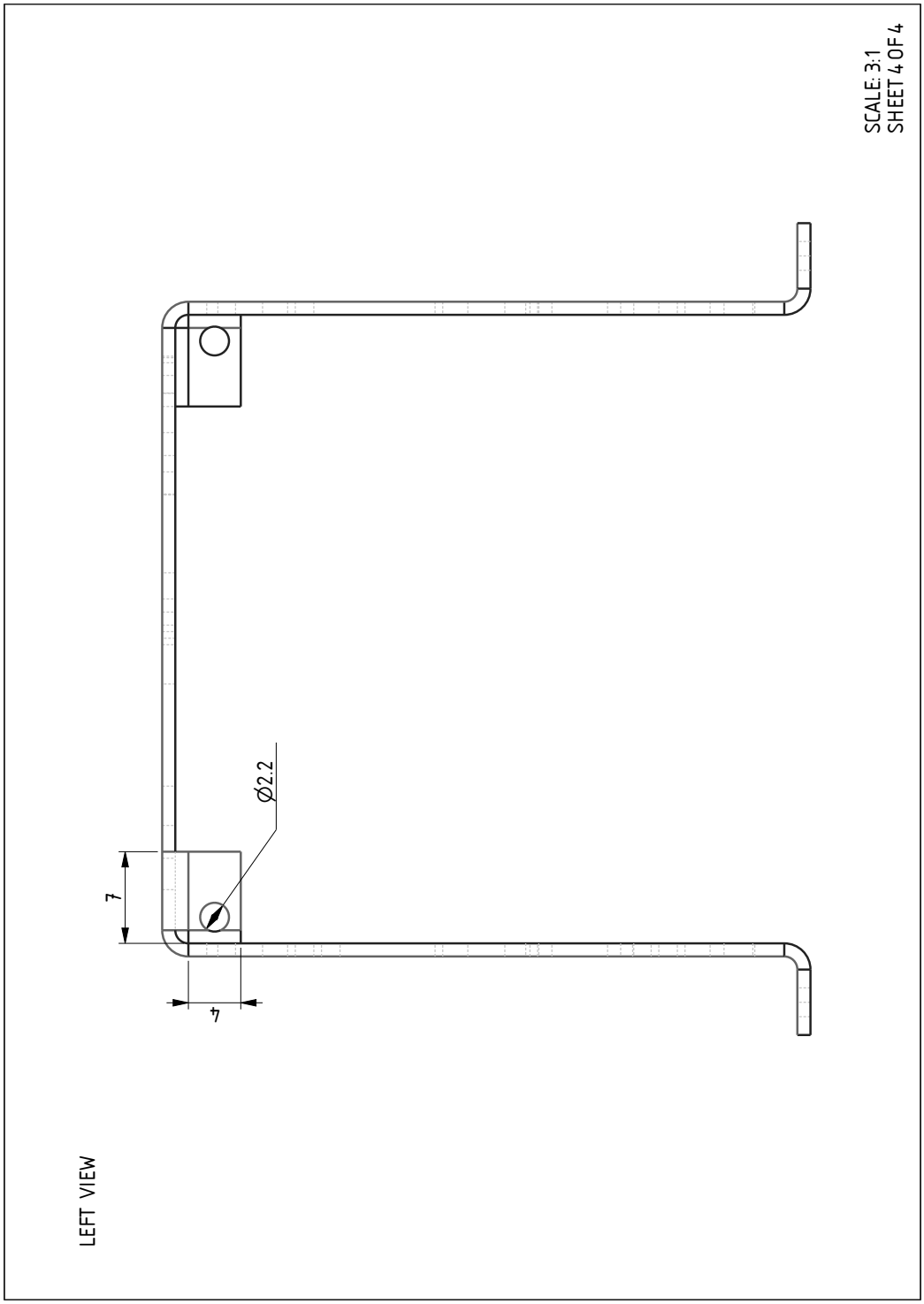


Figure A.4: Dimensioned drawing of sheet-metal frame, sheet 4 of 4.

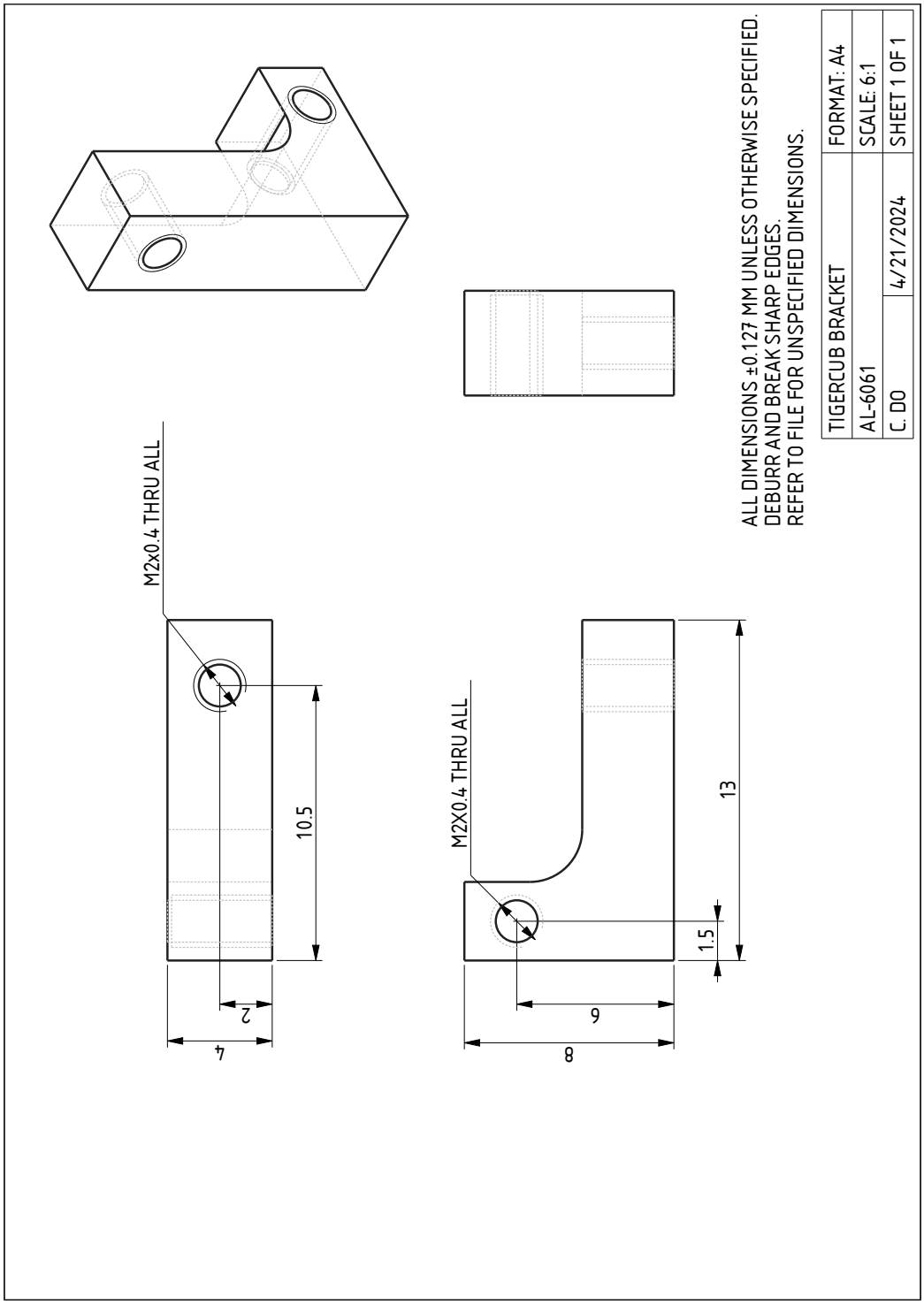


Figure A.5: Dimensioned drawing of L-bracket.

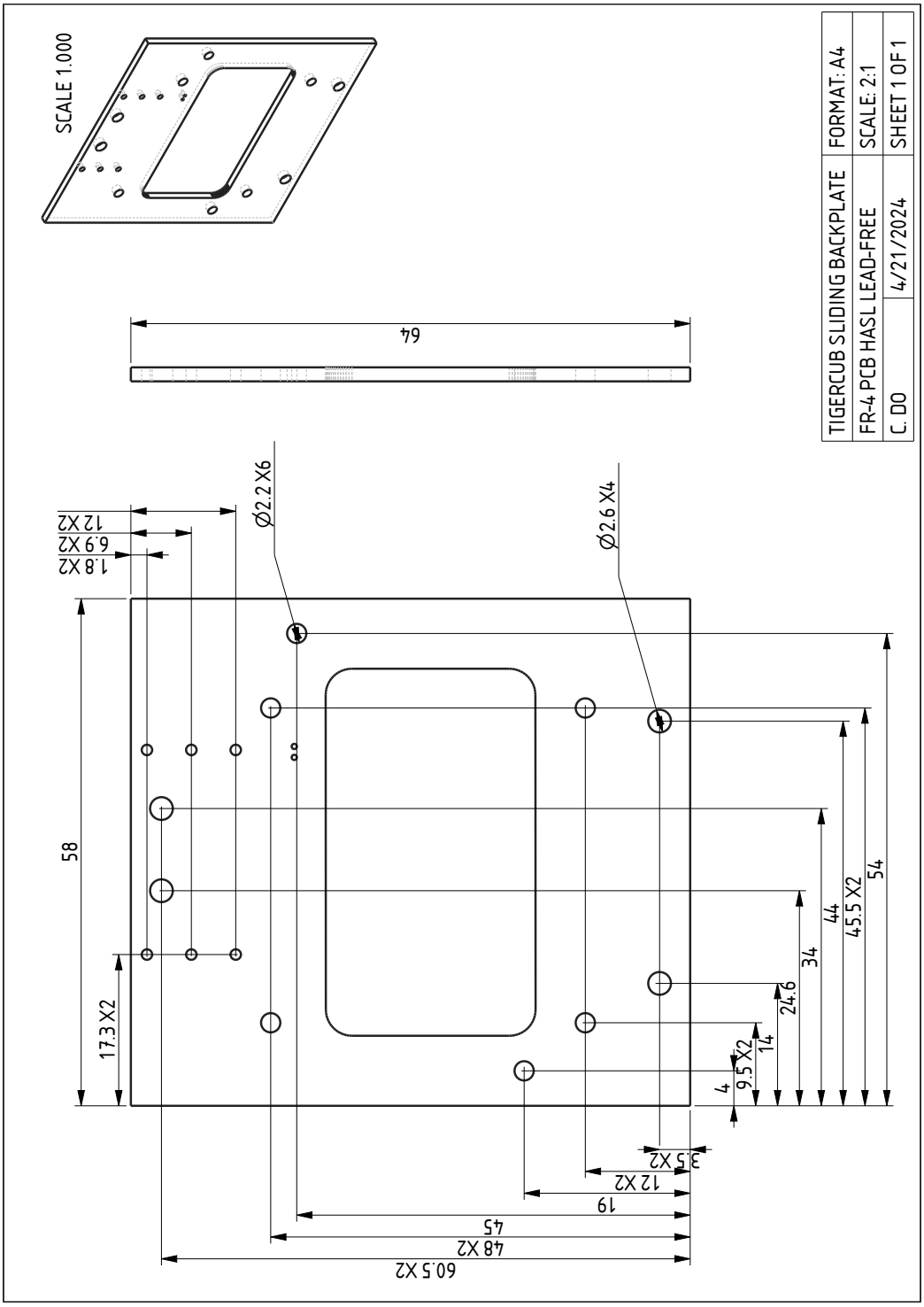


Figure A.6: Dimensioned drawing of sliding backplate.

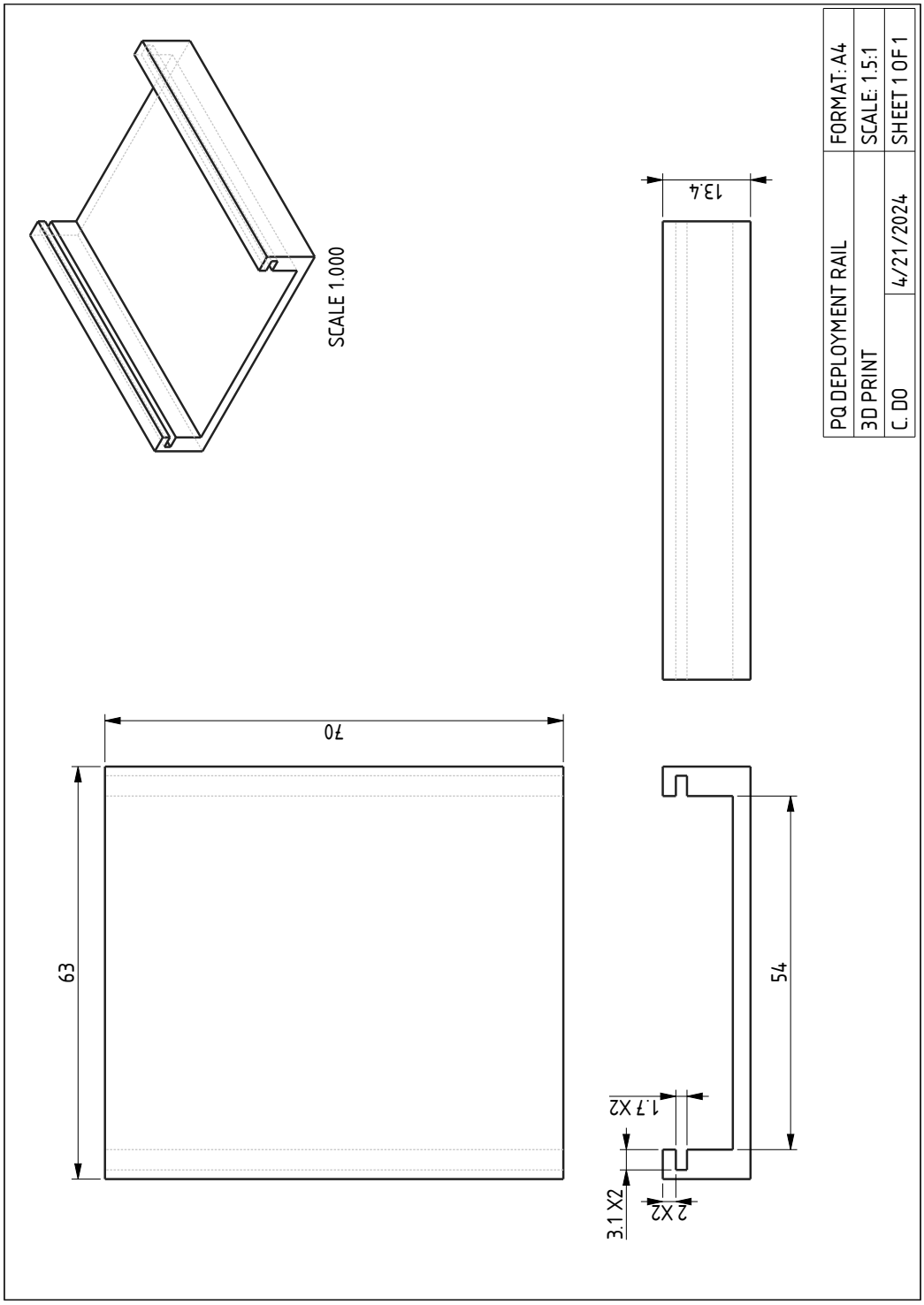


Figure A.7: Dimensioned drawing for 3D-printed deployment rail model.

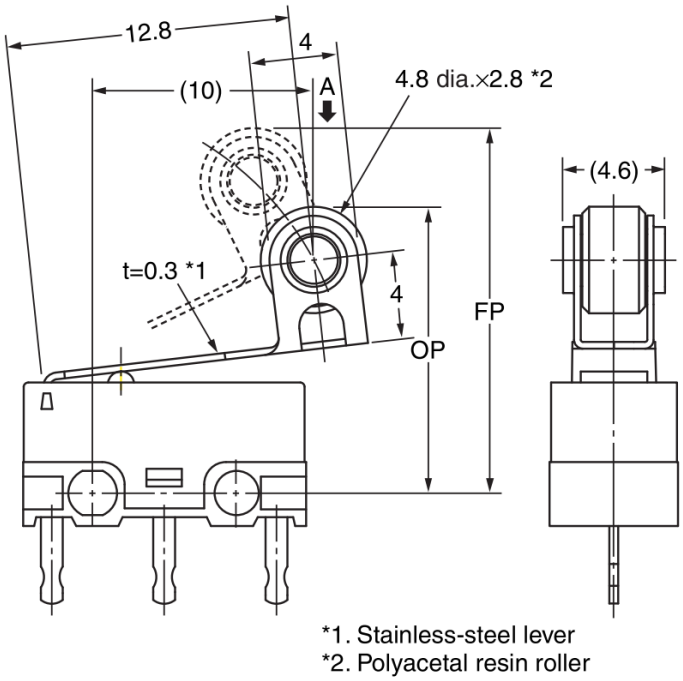


Figure A.8: Dimensioned drawing of roller hinge for D2F-L2-A and D2f-L2-A1 kill switches. Referenced from D2F datasheet [51].

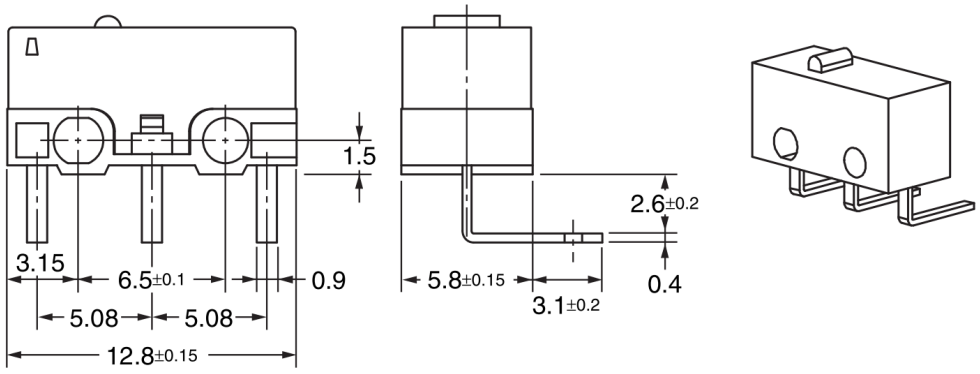


Figure A.9: Dimensioned drawing of PCB terminals for right-angled switch (D2F-L2-A). Referenced from D2F datasheet [51].

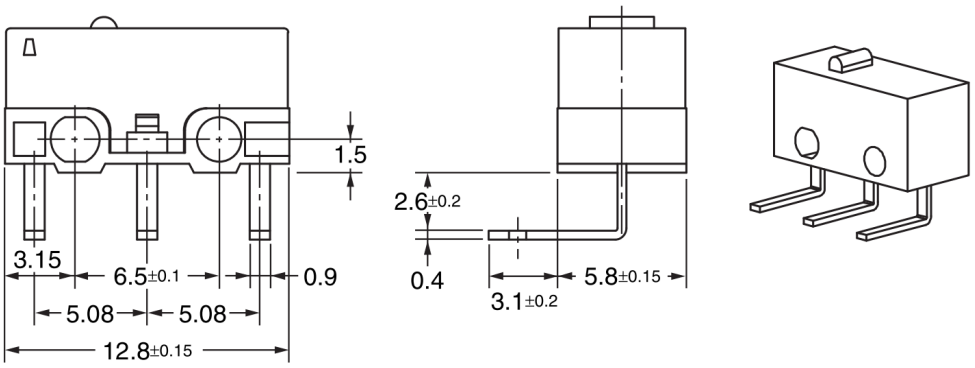


Figure A.10: Dimensioned drawing of PCB terminals for left-angled switch (D2F-L2-A1). Referenced from D2F datasheet [51].

A.2 Electrical Drawings

For reference, we include schematics for the DynOSSAT-EDU EPS.

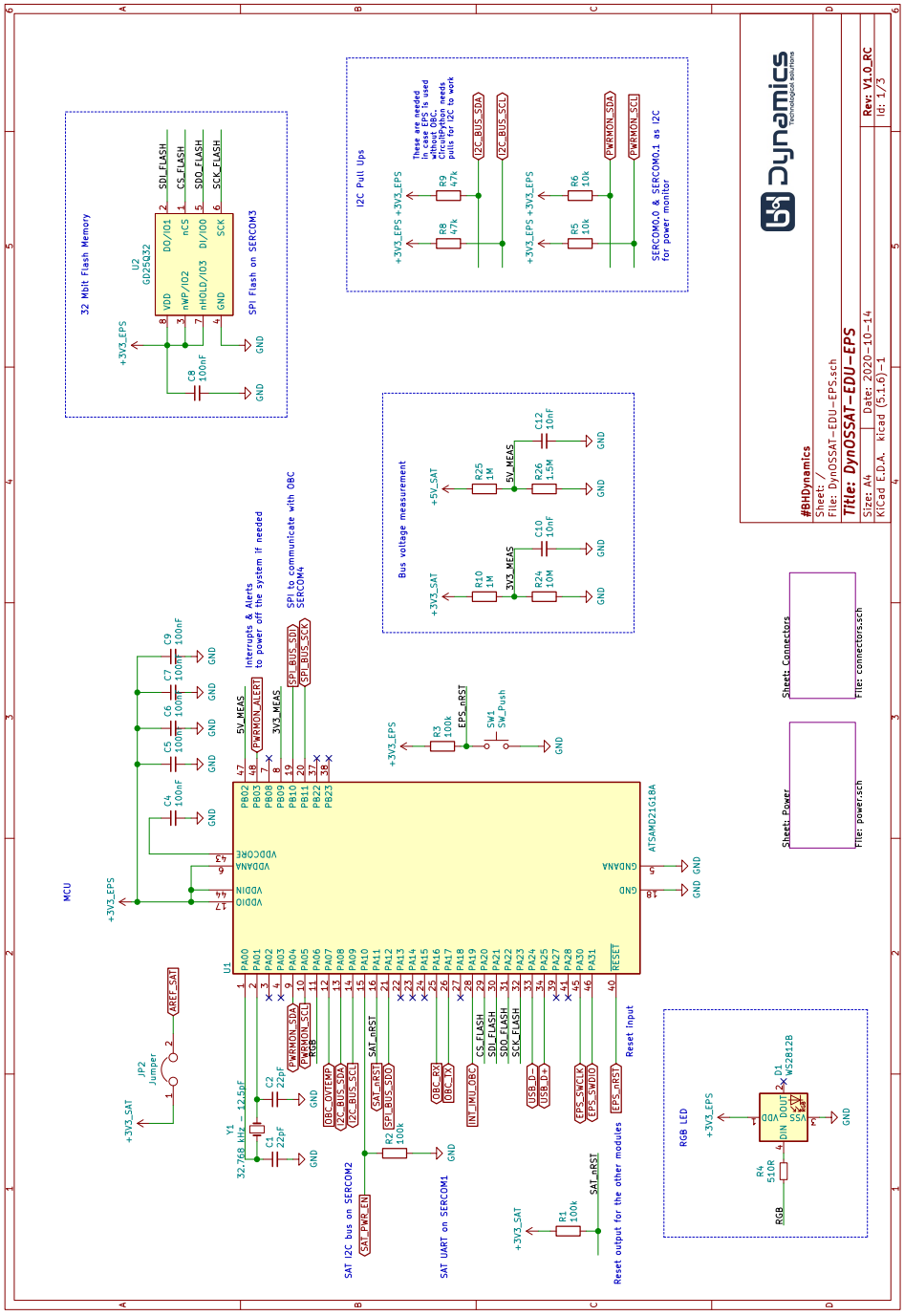


Figure A.11: DynOSSAT-EDU EPS schematic, root page. Referenced from DynOSSAT-EDU GitHub [45].

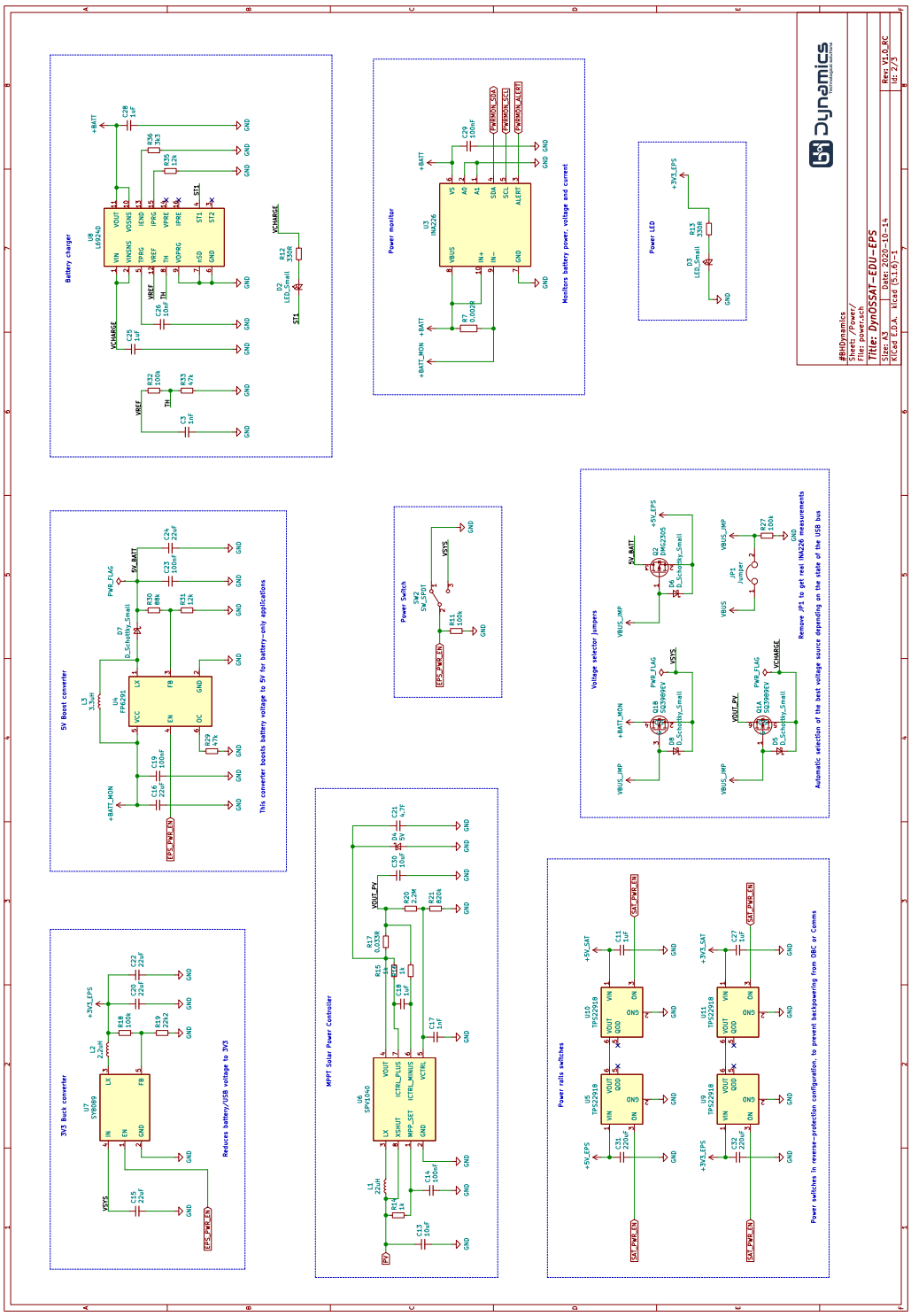


Figure A.12: DynOSSAT-EDU EPS schematic, power connections. Referenced from DynOSSAT-EDU GitHub [45].

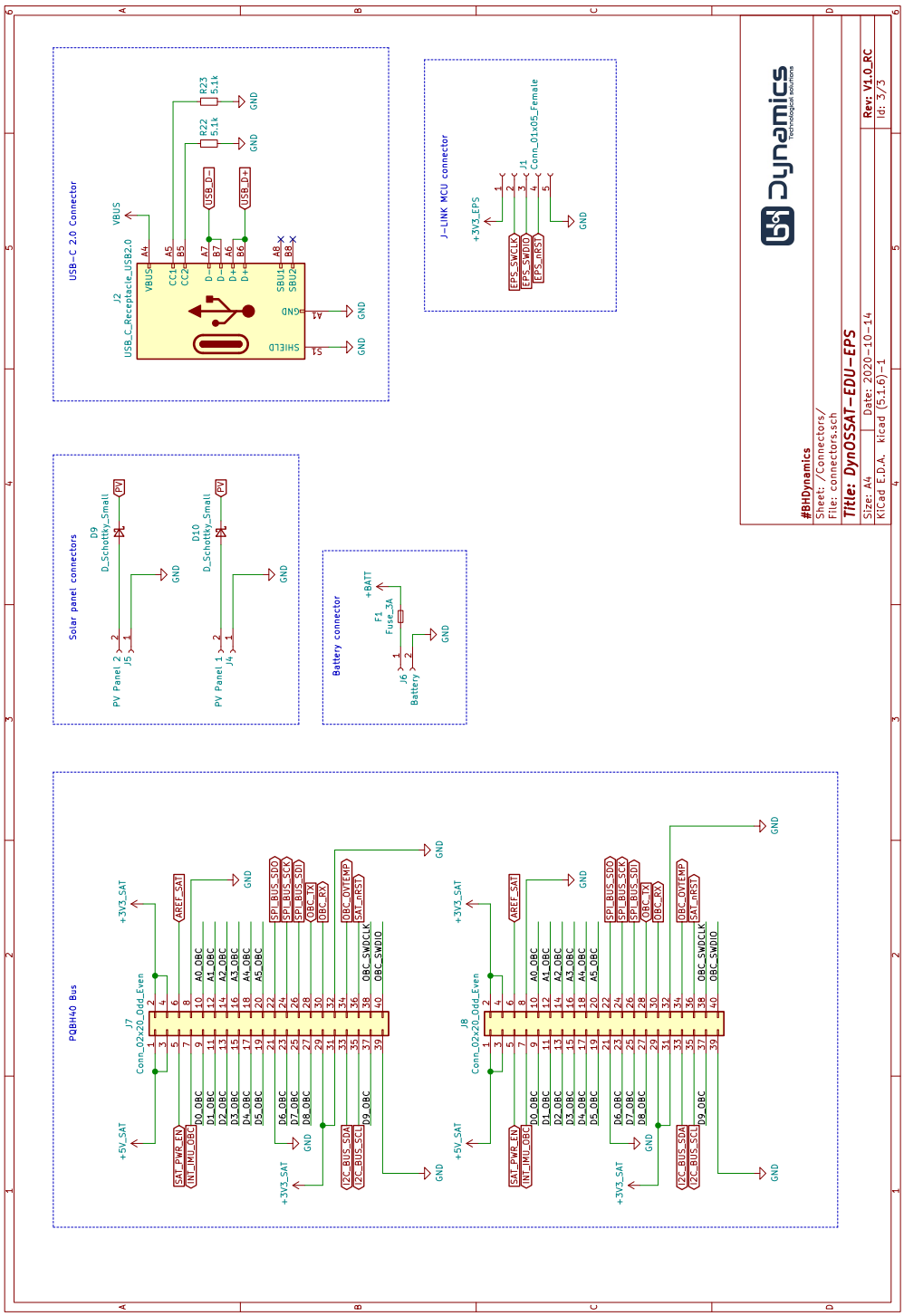


Figure A.13: DynOSSAT-EDU EPS schematic, connectors. Referenced from DynOSSAT-EDU GitHub [45].

A.3 Solar Cell Performance Data

The following data was extrapolated from the solar cell datasheet using the typical SolarMD current-voltage curve (shown in Figure 4.9) and the parameters in Table 4.2 [56].

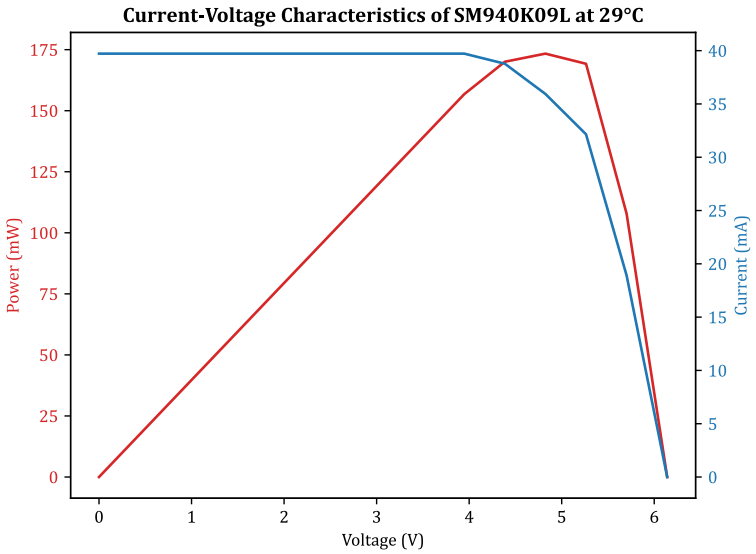


Figure A.14: Solar cell current-voltage curve at 29°C.

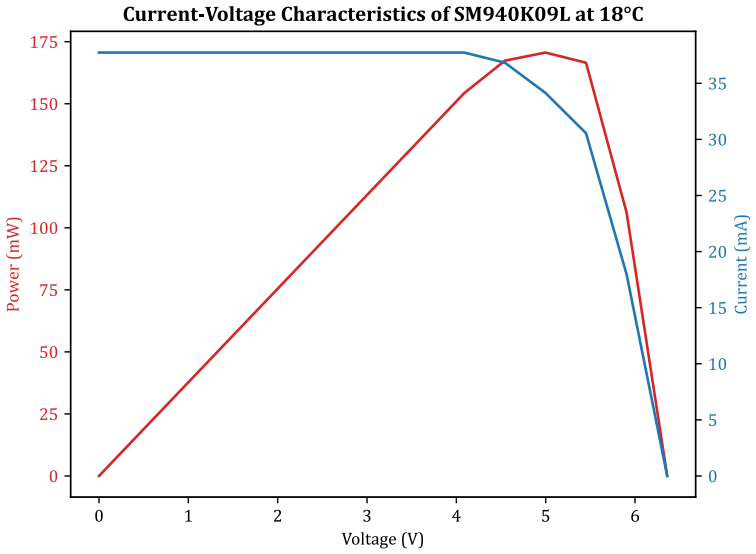


Figure A.15: Solar cell current-voltage curve at 17°C.

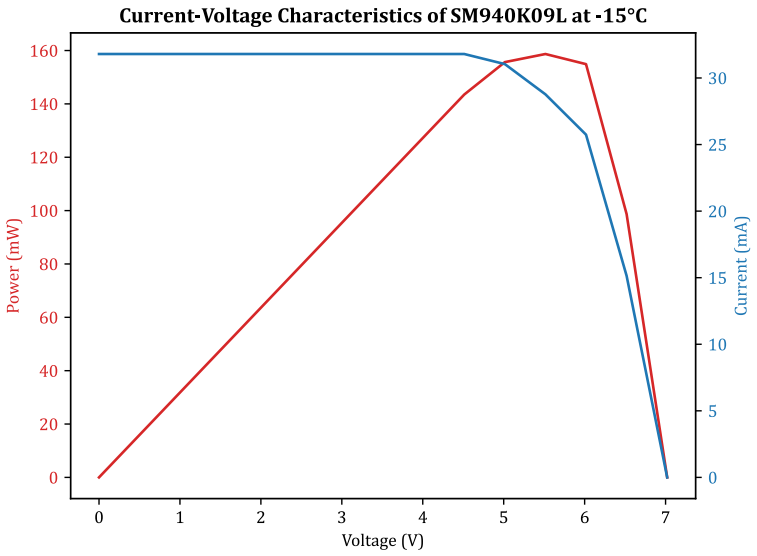


Figure A.16: Solar cell current-voltage curve at $-15°C$.

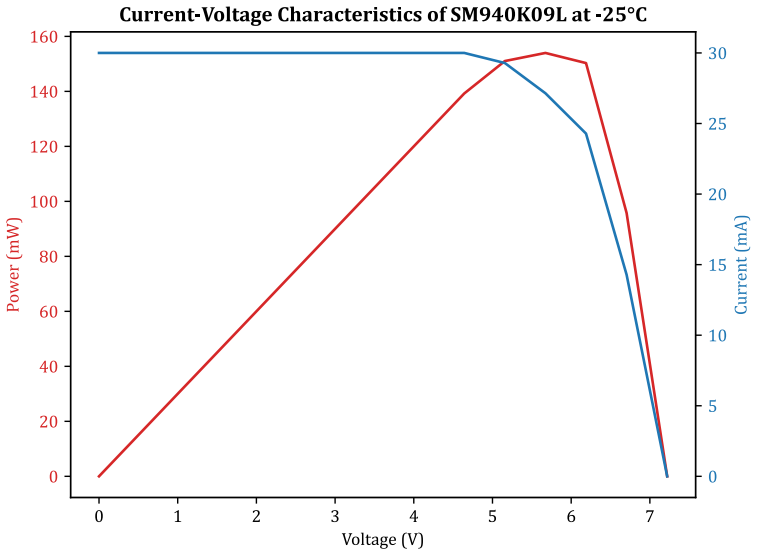


Figure A.17: Solar cell current-voltage curve at $-25°C$.

A.4 Qualification Test Requirements

The following tables from the Falcon 9 Rideshare Payload User's Guide describe test requirements, levels, and success criteria for payload assemblies.

Table 6-1: Payload Unit Test Levels and Durations

Test	Required/Advised	Unit/Fleet Qualification and Acceptance Approach		Flight Unit
		Qualification	Acceptance Must be performed on fully integrated Payload assembly	Protoflight Qualification Must be performed on each fully integrated Payload assembly
		Unit Not flown	Unit Flown	Unit Flown
Quasi Static Load [1]	REQUIRED	1.25 times the limit load in each of 3 axes	1.1 times the limit load in each of 3 axes	1.25 times the limit load in each of 3 axes
Sine Vibration	Advised	1.25 times limit levels, two octave/minute sweep rate in each of 3 axes	1.0 times limit levels, four octave/minute sweep rate in each of 3 axes	1.25 times limit levels, four octave/minute sweep rate in each of 3 axes
Acoustic	Advised	6 dB above acceptance for 2 minutes	MPE spectrum for 1 minute	3 dB above acceptance for 1 minute
Shock	Advised	6 dB above MPE, 3 times in each of 3 orthogonal axes	Not Required	3 dB above MPE, 2 times in each of 3 orthogonal axes
Random Vibration [2]	REQUIRED	6 dB above acceptance for 2 minutes in each of 3 axes	MPE spectrum for 1 minute in each of 3 axes	3 dB above acceptance for 1 minutes in each of 3 axes
Electromagnetic Compatibility [3]	REQUIRED	6 dB EMISM by Test or 12 dB EMISM by Analysis	Not Required	6 dB EMISM by Test or 12 dB EMISM by Analysis
Combined Thermal Vacuum and Thermal Cycle [4]	Advised	±10 °C beyond acceptance for 27 cycles total	Envelope of MPT and minimum range (-24 to 61 °C) for 14 cycles total	±5 °C beyond acceptance for 20 cycles total
Pressure System [5,6]	REQUIRED	Pressures as specified in Table 6.3.12-2 of SMC-S-016 following acceptance proof pressure test, duration sufficient to collect data. Minimum 2.0 times MEOP.	1.5 times ground MEOP for pressure vessels and pressure components. Other metallic pressurized hardware items per References 4 and 5 from SMC-S-016	See Note 6
System-Level Pressure Leak Test [7]	REQUIRED	Not Required	Full Pressure System MEOP Leak Test per Section 6.8.4	Full Pressure System MEOP Leak Test per Section 6.8.4
Pressure Vessel Leak Test [7]	REQUIRED	Not Required	Pressure Vessel Level MEOP Leak Test per Section 6.8.4	Pressure Vessel Level MEOP Leak Test per Section 6.8.4

Figure A.18: Payload unit test levels and durations for payload assemblies on Falcon 9 Rideshare launches. Referenced from Table 6-1 in the Rideshare Payload User's Guide [73].

Table 6-3: Low-Level Sine Sweep Acceptance Criteria

Low-Level Sine Sweep	
Level	0.1 g (minimum)
Sweep rate	3 oct/min or slower
Test axes [1]	X_{PL}, Y_{PL}, Z_{PL}
Success criteria	• All modes above 40 Hz • < 10% shift in first mode frequency • < 30% shift in first mode response

Figure A.19: Payload low-level sine sweep levels and success criteria for for payload assemblies on Falcon 9 Rideshare launches. Referenced from Table 6-3 in the Rideshare Payload User's Guide [73].

Table 6-4: Rideshare Plate Payload Minimum Test Load Factors

Payload Mass	Axial Load Factor (g) X_{PL}	Lateral Load Factor (g) See Table 6-5 for required test axes	
Applicability	All Interfaces	Circular interfaces, CubeSats and 4-points with footprint ratio a/b ≥ 0.6	4-point interfaces with footprint ratio a/b < 0.6
CubeSat Dispenser	10.0	17.0	N/A
20-60 kg	10.0	16.0	18.4
61-150 kg	10.0	13.0	15.0
151-250 kg	8.0	12.0	13.8
251-450 kg	5.0	11.0	12.7
451 kg +	4.5	10.5	12.1

Figure A.20: Quasi-static load factors for payload assemblies on Falcon 9 Rideshare launches. Testing can be done through a sine burst test or a sine sweep test. Referenced from Table 6-4 in the Rideshare Payload User's Guide [73].

Table 6-5: Quasi Static Load Verification Test Requirements

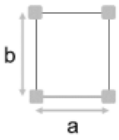
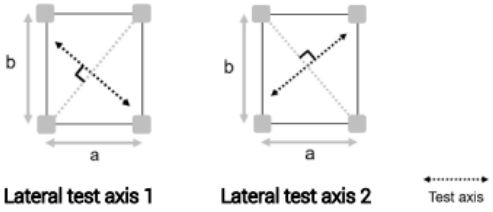
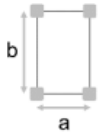
Criteria		Sine Burst Test	Sine Sweep Test
Test parameters		• Sine burst input frequency < 2/3 of Payload first mode frequency • 5 cycles at full level minimum	Sweep rates as specified in Table 6-1 for sine vibration
CG difference between flight and test		X_{PL} direction: Max 10% [1] Y_{PL}, Z_{PL} directions: Max 20%	
Test axes	Circular interfaces & CubeSat Dispensers	Three orthogonal axes: • Axial payload direction: X_{PL} • Lateral payload directions: Y_{PL}, Z_{PL}	
	4-point interface with footprint ratio *a/b* between 0.6 and 1.0 [2] 	Three axes: • Axial payload direction: X_{PL} • Lateral payload directions: axes orthogonal to diagonal axes as shown [3] 	
	4-point interface with footprint ratio *a/b* less than 0.6 [2] 	Three orthogonal axes: • Axial payload direction: X_{PL} • Lateral payload directions: Y_{PL}, Z_{PL} **Note**: this interface aspect ratio requires higher test levels, as shown in Table 6-4	
Accelerometer placement		• At CG (preferred), **OR** • Customer to determine accelerometer placement	
Success criteria		Measured or calculated acceleration at CG achieves target acceleration	

Figure A.21: Quasi-static load verification requirements for payload assemblies on Falcon 9 Rideshare launches. Referenced from Table 6-5 in the Rideshare Payload User's Guide [73].

Refer to to the Rideshare Payload User's Guide for further details on qualification testing.

Appendix B

Code

B.1 RockBLOCK Transmission Sample Code

The following script, provided by SparkFun, performs startup and a basic transmission operation on a RockBLOCK [26]. Install the `IridiumSBDi2c` library through the Arduino library manager before running this script.

```
1  #include <IridiumSBD.h> // Click here to get the library: http://
       librarymanager/All#IridiumSBDI2C
2
3  /*
4   * BasicSend
5   *
6   * This sketch sends a "Hello, world!" message from the satellite modem.
7   * If you have activated your account and have credits, this message
8   * should arrive at the endpoints (delivery group) you have configured
9   * (email address or HTTP POST).
10  *
11  * Assumptions
12  *
13  * The sketch assumes an Arduino Mega or other Arduino-like device with
14  * multiple HardwareSerial ports.  It assumes the satellite modem is
15  * connected to Serial1.  Change this as needed.  SoftwareSerial on an
       Uno
16  * works fine as well.
```

```
 */

#define IridiumSerial Serial1
#define DIAGNOSTICS false // Change this to see diagnostics

// Declare the IridiumSBD object
IridiumSBD modem(IridiumSerial);

void setup()
{
  int signalQuality = -1;
  int err;

  // Start the console serial port
  Serial.begin(115200);
  while (!Serial);

  // Start the serial port connected to the satellite modem
  IridiumSerial.begin(19200);

  // Begin satellite modem operation
  Serial.println(F("Starting modem..."));
  err = modem.begin();
  if (err != ISBD_SUCCESS)
  {
    Serial.print(F("Begin failed: error "));
    Serial.println(err);
    if (err == ISBD_NO_MODEM_DETECTED)
      Serial.println(F("No modem detected: check wiring."));
    return;
  }

  // Example: Print the firmware revision
  char version[12];
  err = modem.getFirmwareVersion(version, sizeof(version));
  if (err != ISBD_SUCCESS)
```

```
53  {
54     Serial.print(F("FirmwareVersion failed: error "));
55     Serial.println(err);
56     return;
57  }
58  Serial.print(F("Firmware Version is "));
59  Serial.print(version);
60  Serial.println(F("."));
61
62  // Example: Print the IMEI
63  char IMEI[16];
64  err = modem.getIMEI(IMEI, sizeof(IMEI));
65  if (err != ISBD_SUCCESS)
66  {
67     Serial.print(F("getIMEI failed: error "));
68     Serial.println(err);
69     return;
70  }
71  Serial.print(F("IMEI is "));
72  Serial.print(IMEI);
73  Serial.println(F("."));
74
75  // Example: Test the signal quality.
76  // This returns a number between 0 and 5.
77  // 2 or better is preferred.
78  err = modem.getSignalQuality(signalQuality);
79  if (err != ISBD_SUCCESS)
80  {
81    Serial.print(F("SignalQuality failed: error "));
82    Serial.println(err);
83    return;
84  }
85
86  Serial.print(F("On a scale of 0 to 5, signal quality is currently "));
87  Serial.print(signalQuality);
88  Serial.println(F("."));
```

```
89
90    // Send the message
91    Serial.println(F("Trying to send the message.  This might take several
        minutes."));
92    err = modem.sendSBDText("Hello, world!");
93    if (err != ISBD_SUCCESS)
94    {
95      Serial.print(F("sendSBDText failed: error "));
96      Serial.println(err);
97      if (err == ISBD_SENDRECEIVE_TIMEOUT)
98        Serial.println(F("Try again with a better view of the sky."));
99    }
100
101   else
102   {
103     Serial.println(F("Hey, it worked!"));
104   }
105
106   // Clear the Mobile Originated message buffer
107   Serial.println(F("Clearing the MO buffer."));
108   err = modem.clearBuffers(ISBD_CLEAR_MO); // Clear MO buffer
109   if (err != ISBD_SUCCESS)
110   {
111     Serial.print(F("clearBuffers failed: error "));
112     Serial.println(err);
113   }
114
115   Serial.println(F("Done!"));
116 }
117
118 void loop()
119 {
120 }
121
122 #if DIAGNOSTICS
123 void ISBDConsoleCallback(IridiumSBD *device, char c)
```

```
124 {
125   Serial.write(c);
126 }
127 
128 void ISBDDiagsCallback(IridiumSBD *device, char c)
129 {
130   Serial.write(c);
131 }
132 #endif
```

B.2 Adafruit LSM6DS3TR-C + LIS3MDL Sample Code

The following script, modified from the script provided by Kattni Rembor for Adafruit Industries, demonstrates accelerometer, gyroscope, and magnetometer readings for the Adafruit LSM6DS3TR-C + LIS3MDL breakout board [35].

Install the `Adafruit_LIS3MDL` and `Adafruit_LSM6DS` libraries before running this script.

```
// SPDX-FileCopyrightText: 2020 Kattni Rembor for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_LSM6DS3.h>
Adafruit_LSM6DS3 lsm6ds;

#include <Adafruit_LIS3MDL.h>
Adafruit_LIS3MDL lis3mdl;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause until serial console opens

  Serial.println("Adafruit LSM6DS+LIS3MDL test!");

  bool lsm6ds_success, lis3mdl_success;

  // hardware I2C mode, can pass in address & alt Wire

  lsm6ds_success = lsm6ds.begin_I2C();
  lis3mdl_success = lis3mdl.begin_I2C();

  if (!lsm6ds_success){
    Serial.println("Failed to find LSM6DS chip");
```

```
  }
  if (!lis3mdl_success){
    Serial.println("Failed to find LIS3MDL chip");
  }
  if (!(lsm6ds_success && lis3mdl_success)) {
    while (1) {
      delay(10);
    }
  }

  Serial.println("LSM6DS and LIS3MDL Found!");

  // lsm6ds.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
  Serial.print("Accelerometer range set to: ");
  switch (lsm6ds.getAccelRange()) {
  case LSM6DS_ACCEL_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case LSM6DS_ACCEL_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case LSM6DS_ACCEL_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case LSM6DS_ACCEL_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }

  // lsm6ds.setAccelDataRate(LSM6DS_RATE_12_5_HZ);
  Serial.print("Accelerometer data rate set to: ");
  switch (lsm6ds.getAccelDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
    Serial.println("0 Hz");
    break;
  case LSM6DS_RATE_12_5_HZ:
```

```
    Serial.println("12.5 Hz");
    break;
  case LSM6DS_RATE_26_HZ:
    Serial.println("26 Hz");
    break;
  case LSM6DS_RATE_52_HZ:
    Serial.println("52 Hz");
    break;
  case LSM6DS_RATE_104_HZ:
    Serial.println("104 Hz");
    break;
  case LSM6DS_RATE_208_HZ:
    Serial.println("208 Hz");
    break;
  case LSM6DS_RATE_416_HZ:
    Serial.println("416 Hz");
    break;
  case LSM6DS_RATE_833_HZ:
    Serial.println("833 Hz");
    break;
  case LSM6DS_RATE_1_66K_HZ:
    Serial.println("1.66 KHz");
    break;
  case LSM6DS_RATE_3_33K_HZ:
    Serial.println("3.33 KHz");
    break;
  case LSM6DS_RATE_6_66K_HZ:
    Serial.println("6.66 KHz");
    break;
  }

  // lsm6ds.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS );
  Serial.print("Gyro range set to: ");
  switch (lsm6ds.getGyroRange()) {
  case LSM6DS_GYRO_RANGE_125_DPS:
    Serial.println("125 degrees/s");
```

```
    break;
  case LSM6DS_GYRO_RANGE_250_DPS:
    Serial.println("250 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_500_DPS:
    Serial.println("500 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_1000_DPS:
    Serial.println("1000 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_2000_DPS:
    Serial.println("2000 degrees/s");
    break;
  case ISM330DHCX_GYRO_RANGE_4000_DPS:
    Serial.println("4000 degrees/s");
    break;
  }
  // lsm6ds.setGyroDataRate(LSM6DS_RATE_12_5_HZ);
  Serial.print("Gyro data rate set to: ");
  switch (lsm6ds.getGyroDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
    Serial.println("0 Hz");
    break;
  case LSM6DS_RATE_12_5_HZ:
    Serial.println("12.5 Hz");
    break;
  case LSM6DS_RATE_26_HZ:
    Serial.println("26 Hz");
    break;
  case LSM6DS_RATE_52_HZ:
    Serial.println("52 Hz");
    break;
  case LSM6DS_RATE_104_HZ:
    Serial.println("104 Hz");
    break;
  case LSM6DS_RATE_208_HZ:
```

```
    Serial.println("208 Hz");
    break;
  case LSM6DS_RATE_416_HZ:
    Serial.println("416 Hz");
    break;
  case LSM6DS_RATE_833_HZ:
    Serial.println("833 Hz");
    break;
  case LSM6DS_RATE_1_66K_HZ:
    Serial.println("1.66 KHz");
    break;
  case LSM6DS_RATE_3_33K_HZ:
    Serial.println("3.33 KHz");
    break;
  case LSM6DS_RATE_6_66K_HZ:
    Serial.println("6.66 KHz");
    break;
  }

  lis3mdl.setDataRate(LIS3MDL_DATARATE_155_HZ);
  // You can check the datarate by looking at the frequency of the DRDY
   pin
  Serial.print("Magnetometer data rate set to: ");
  switch (lis3mdl.getDataRate()) {
    case LIS3MDL_DATARATE_0_625_HZ: Serial.println("0.625 Hz"); break;
    case LIS3MDL_DATARATE_1_25_HZ: Serial.println("1.25 Hz"); break;
    case LIS3MDL_DATARATE_2_5_HZ: Serial.println("2.5 Hz"); break;
    case LIS3MDL_DATARATE_5_HZ: Serial.println("5 Hz"); break;
    case LIS3MDL_DATARATE_10_HZ: Serial.println("10 Hz"); break;
    case LIS3MDL_DATARATE_20_HZ: Serial.println("20 Hz"); break;
    case LIS3MDL_DATARATE_40_HZ: Serial.println("40 Hz"); break;
    case LIS3MDL_DATARATE_80_HZ: Serial.println("80 Hz"); break;
    case LIS3MDL_DATARATE_155_HZ: Serial.println("155 Hz"); break;
    case LIS3MDL_DATARATE_300_HZ: Serial.println("300 Hz"); break;
    case LIS3MDL_DATARATE_560_HZ: Serial.println("560 Hz"); break;
    case LIS3MDL_DATARATE_1000_HZ: Serial.println("1000 Hz"); break;
```

```
  }

  lis3mdl.setRange(LIS3MDL_RANGE_4_GAUSS);
  Serial.print("Range set to: ");
  switch (lis3mdl.getRange()) {
    case LIS3MDL_RANGE_4_GAUSS: Serial.println("+-4 gauss"); break;
    case LIS3MDL_RANGE_8_GAUSS: Serial.println("+-8 gauss"); break;
    case LIS3MDL_RANGE_12_GAUSS: Serial.println("+-12 gauss"); break;
    case LIS3MDL_RANGE_16_GAUSS: Serial.println("+-16 gauss"); break;
  }

  lis3mdl.setPerformanceMode(LIS3MDL_MEDIUMMODE);
  Serial.print("Magnetometer performance mode set to: ");
  switch (lis3mdl.getPerformanceMode()) {
    case LIS3MDL_LOWPOWERMODE: Serial.println("Low"); break;
    case LIS3MDL_MEDIUMMODE: Serial.println("Medium"); break;
    case LIS3MDL_HIGHMODE: Serial.println("High"); break;
    case LIS3MDL_ULTRAHIGHMODE: Serial.println("Ultra-High"); break;
  }

  lis3mdl.setOperationMode(LIS3MDL_CONTINUOUSMODE);
  Serial.print("Magnetometer operation mode set to: ");
  // Single shot mode will complete conversion and go into power down
  switch (lis3mdl.getOperationMode()) {
    case LIS3MDL_CONTINUOUSMODE: Serial.println("Continuous"); break;
    case LIS3MDL_SINGLEMODE: Serial.println("Single mode"); break;
    case LIS3MDL_POWERDOWNMODE: Serial.println("Power-down"); break;
  }

  lis3mdl.setIntThreshold(500);
  lis3mdl.configInterrupt(false, false, true, // enable z axis
                          true, // polarity
                          false, // don't latch
                          true); // enabled!

}
```

```
206
207 void loop() {
208
209   sensors_event_t accel, gyro, mag, temp;
210
211   //  /* Get new normalized sensor events */
212   lsm6ds.getEvent(&accel, &gyro, &temp);
213   lis3mdl.getEvent(&mag);
214
215   /* Display the results (acceleration is measured in m/s^2) */
216   Serial.print("\t\tAccel X: ");
217   Serial.print(accel.acceleration.x, 4);
218   Serial.print(" \tY: ");
219   Serial.print(accel.acceleration.y, 4);
220   Serial.print(" \tZ: ");
221   Serial.print(accel.acceleration.z, 4);
222   Serial.println(" \tm/s^2 ");
223
224   /* Display the results (rotation is measured in rad/s) */
225   Serial.print("\t\tGyro  X: ");
226   Serial.print(gyro.gyro.x, 4);
227   Serial.print(" \tY: ");
228   Serial.print(gyro.gyro.y, 4);
229   Serial.print(" \tZ: ");
230   Serial.print(gyro.gyro.z, 4);
231   Serial.println(" \tradians/s ");
232
233   /* Display the results (magnetic field is measured in uTesla) */
234   Serial.print(" \t\tMag   X: ");
235   Serial.print(mag.magnetic.x, 4);
236   Serial.print(" \tY: ");
237   Serial.print(mag.magnetic.y, 4);
238   Serial.print(" \tZ: ");
239   Serial.print(mag.magnetic.z, 4);
240   Serial.println(" \tuTesla ");
241
```

```
242   Serial.print("\t\tTemp    :\t\t\t\t\t");
243   Serial.print(temp.temperature);
244   Serial.println(" \tdeg C");
245   Serial.println();
246   delay(1000);
247 
248 }
```

B.3 Adafruit MicroSD Sample Code

The following script, modified from the script provided by David A. Mellis, demonstrates accelerometer, gyroscope, and magnetometer readings for the Adafruit MicroSD breakout board [134].

```
/*
  SD card read/write

 This example shows how to read and write data to and from an SD card
    file
 The circuit:
 * SD card attached to SPI bus as follows:
 ** SI - Teensy 4.0 pin 11
 ** SO - Teensy 4.0 pin 12
 ** CLK - Teensy 4.0 pin 13
 ** CS - Teensy 4.0 pin 10

 created   Nov 2010
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe

 This example code is in the public domain.

 */

#include <SD.h>
#include <SPI.h>

File myFile;
const int chipSelect = 10;

void setup()
{
 // Open serial communications and wait for port to open:
```

```
30  Serial.begin(9600);
31   while (!Serial) {
32    ; // wait for serial port to connect.
33  }
34
35
36  Serial.print("Initializing SD card...");
37
38  if (!SD.begin(chipSelect)) {
39    Serial.println("initialization failed!");
40    return;
41  }
42  Serial.println("initialization done.");
43
44  // open the file.
45  myFile = SD.open("test.txt", FILE_WRITE);
46
47  // if the file opened okay, write to it:
48  if (myFile) {
49    Serial.print("Writing to test.txt...");
50    myFile.println("testing 1, 2, 3.");
51  // close the file:
52    myFile.close();
53    Serial.println("done.");
54  } else {
55    // if the file didn't open, print an error:
56    Serial.println("error opening test.txt");
57  }
58
59  // re-open the file for reading:
60  myFile = SD.open("test.txt");
61  if (myFile) {
62    Serial.println("test.txt:");
63
64    // read from the file until there's nothing else in it:
65    while (myFile.available()) {
```

```
66        Serial.write(myFile.read());
67      }
68      // close the file:
69      myFile.close();
70    } else {
71      // if the file didn't open, print an error:
72      Serial.println("error opening test.txt");
73    }
74 }
75
76 void loop()
77 {
78   // nothing happens after setup
79 }
```

B.4 Teensy Watchdog Sample Code

The following script, referenced from the `watchdog1_demo.ino` by Antonio Brewer, demonstrates `WDOG1` (Teensy internal reset) [42]. Users should install the `WDT_T4` library before running this script.

```
#include "Watchdog_t4.h"
WDT_T4<WDT1> wdt;

void myCallback() {
  Serial.println("FEED THE DOG SOON, OR RESET!");
}

void setup() {
  Serial.begin(1);
  delay(600);
  Serial.println("Begin......"); // reset

  WDT_timings_t config;
  config.trigger = 5; // time before the callback triggers; 0->128 sec
  config.timeout = 10; // time before the reset triggers (without
    feeding); 0->128 sec
  config.callback = myCallback;
  wdt.begin(config);
  pinMode(13, OUTPUT);
}

void loop() {
  static uint32_t blinkLed = millis();
  if ( millis() - blinkLed > 50 ) {
    blinkled = millis();
    digitalWriteFast(13, !digitalReadFast(13));
  }

  static uint32_t callback_test = millis();
  if ( millis() - callback_test > 5500 ) {
    callback_test = millis();
```

```
    wdt.feed(); // feeds the watchdog
  }
}
```

The following script, referenced from the `watchdog2_demo.ino` by Antonio Brewer, demonstrates `WDOG2` (GPIO pin reset) [42].

```
#include "Watchdog_t4.h"
WDT_T4<WDT2> wdt;

void myCallback() {
  Serial.println("FEED THE DOG SOON, OR RESET!");
}

void setup() {
  Serial.begin(1);
  delay(600);
  Serial.println("Begin......");

  WDT_timings_t config;
  config.trigger = 5; // time before the callback triggers; 0->128 sec
  config.timeout = 10; // time before the reset triggers (without
    feeding); 0->128 sec
  config.pin = 23; // payload reset GPIO pin
  config.callback = myCallback;
  wdt.begin(config);
}

void loop() {
  static uint32_t feed = millis();

  // feed the dog every 11 sec to exceed the 10 sec timeout
  if ( millis() - feed > 11000 ) {
    feed = millis();
    wdt.feed();
  }
}
```

Appendix C

Budget and Bill of Materials

C.1 Project Budget

The following items were purchased for this project. All prices are in USD.

Item	Vendor	Part Number	Unit Price	Qty	Total Price	Date
EPS electronic components	JLCPCB	–	35.00	1	35.00	Sep-18
EPS PCB	JLCPCB	–	142.04	1	142.04	Oct-20
EPS Molex Picoblade 2-pin header	DigiKey	530480210	0.36	5	1.80	Oct-30
EPS 2x20 SMD connector	DigiKey	204632-E	4.90	3	14.70	Oct-30
EPS slide switch	DigiKey	OS102011MA1QN1	0.49	5	2.45	Oct-30
Picoblade 2-pin cables	Adafruit	4922	0.68	10	6.80	Nov-20
Battery	Amazon	–	8.99	1	8.99	Nov-20
Teensy 4.0	Amazon	–	23.46	2	46.92	Nov-20
Molex Picoblade 2-pin header	Mouser	53048-0210	0.216	20	4.32	Nov-25
18-8 SS M2x0.4 SHCS, 5 mm, pack of 100	McMaster	91292A005	14.26	1	14.26	Nov-25
18-8 SS M2.5x0.45 SHCS, 5mm, pack of 100	McMaster	91292A009	6.00	1.00	6.00	Nov-25
18-8 SS M2x0.4 SHCS, 35 mm, pack of 5	McMaster	91292a337	20.18	1	20.18	Nov-25
18-8 SS M2x0.4nuts, pack of 100	McMaster	91828A111	6.14	1	6.14	Nov-25
18-8 SS M2.5x0.45 nuts, pack of 100	McMaster	91828A113	6.45	1	6.45	Nov-25
Schotty diodes for solar panels	Mouser	CUS10S30,H3F	0.243	15	3.65	Nov-25
Rollder switch, left	Mouser	D2F-L2-A	5.06	5	25.30	Nov-25
Roller switch, right	Mouser	D2F-L2-A1	5.12	5	25.60	Nov-25
Solar cells	DigiKey	SM940K09L	6.026	15	90.39	Nov-25
Solar panel PCBs	JLCPCB	–	–	–	27.80	Nov-25
Frame	PCBWay	–	–	–	85.05	Dec-3
Solar panel PCBs	JLCPCB	–	–	–	29.53	Dec-4
Micro SD SPI or SDIO Card Breakout Board	Adafruit	4682	3.50	2	7.00	Dec-7
LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU	Adafruit	5543	19.95	4	79.80	Dec-7
Molex Picoblade 10-pin header	Mouser	51021-1000	0.36	5	1.80	Feb-5
RockBLOCK monthly line rental	Ground Control	–	13.96	3	41.88	Feb-5
RockBLOCK 200 credits	Ground Control	–	0.13	200	26.00	Feb-5
Molex Picoblade pre-crimped 6-in wires	Mouser	79758-0006	0.773	20	15.46	Feb-9
Molex Picoblade pre-crimped 12-in wires	Mouser	79758-0014	1.04	5	5.20	Feb-9
Molex Picoblade 5-pin header	DigiKey	530480510	0.49	5	2.45	Mar-12
2x20 1.27mm pitch header	DigiKey	M50-3002045	4.91	3	14.73	Mar-12
Samtec 5-pin header (shortpins)	DigiKey	SQT-105-01-F-S	1.83	3	5.49	Mar-12
Samtec 5-pin header (long pins)	DigiKey	SQT-105-03-F-S	1.91	3	5.73	Mar-12
Samtec 10-pin header (long pins)	DigiKey	SQT-110-03-F-S	2.73	5	13.65	Mar-12
OBC, sliding backplate PCBs	PCBWay	–	–	–	37.04	Mar-12
18-8 SSNo. 2 unthreaded spacer, 5/32”	McMaster	92320A453	1.34	10	13.40	Mar-14
18-8 SS No. 2 unthreaded spacer, 9/32”	McMaster	92320A455	1.35	30	40.50	Mar-14
Frame and L-brackets	PCBWay	–	–	–	231.28	Mar-16
OBC2, solar panel PCBs	PCBWay	–	–	–	31.87	Mar-21
Battery	Amazon		8.52	2	17.04	Mar-22
Sliding backplate PCB	PCBWay	–	–	–	26.70	Mar-22
Sliding backplate PCB	PCBWay	–	–	–	26.70	Mar-25
Molex Picoblade 5-pin header, vertical	DigiKey	51021-0500	0.26	10	2.60	Mar-27
Molex Picoblade 2-pin header, vertical	DigiKey	51021-0200	0.14	20	2.80	Mar-29
RockBLOCK 9603	Mouser	WRL-14498	345.07	1	345.07	Apr-5
Solar panels and frame	PCBWay	–	–	–	78.53	Apr-6
18-8 SS thin hex nut M2x0.4, pack of 50	McMaster	90710A020	8.52	1	8.52	Apr-12
Solar cells	DigiKey	SM101K12TF	9.45	4	37.80	Apr-12
18-8 SS low-profile SHCS, M2x0.4, 5 mm, pack of 10	McMaster	92855A837	19.06	1	19.06	Apr-16
Solar panels and sliding backplate PCB	PCBWay	–	–	–	36.77	Apr-18
Frame	PCBWay	–	–	–	133.99	Apr-20
				Total	**1912.23**	

Table C.1: Project budget.

This project was generously funded by the following organizations.

Organization	Amount
Lidow Independent Work/Senior Thesis Fund	1200
Morgan W. McKinzie '93 Senior Thesis Prize Fund	1400
Department of Mechanical and Aerospace Engineering	400
Council of Science and Technology	400
Total	**3400**
Remaining	**1487.77**

Table C.2: Funding sources.

C.2 TigerCub Bill of Materials

The following tables contain the bill of materials for all of TigerCub's subsytems, followed by a cost summary. All prices are in USD. The prices for all machined parts (the sheet-metal frame and L-brackets) are for "Economy" speed (8-11 business days), manufactured internationally. All prices are subject to change.

Item Description	Supplier	Manufacturer	Part Number	Unit Price	Quantity	Total Price
RockBLOCK 9603	Mouser	Ground Control	RockBLOCK 9603	334.38	1	334.38
Pre-crimped Picoblade wires	Mouser	Molex	79758-0006	0.89	5	4.45
Picoblade 5-position receptacle	DigiKey	Molex	51021-0500	0.26	1	0.26
Picoblade 10-position receptacle	DigiKey	Molex	51021-1000	0.41	1	0.41

Table C.3: Bill of materials for radio communication system.

Item Description	Supplier	Manufacturer	Part Number	Unit Price	Quantity	Total Price
10-position header with short pins	DigiKey	Samtec	SQT-110-01-F-S	2.24	1	2.24
5-position header with short pins	DigiKey	Samtec	SQT-105-01-F-S	1.68	1	1.68
10-position header with long pins	DigiKey	Samtec	SQT-110-03-F-S	2.35	1	2.35
5-position header with long pins	DigiKey	Samtec	SQT-105-03-F-S	1.76	1	1.76
Picoblade 2-position right-angle header	Mouser	Molex	53048-0210	0.38	1	0.38
Teensy 4.0 development board	PJRC	PJRC	TEENSY40	23.80	1	23.80
40-position receptacle, 1.27mm pitch	DigiKey	Harwin	M50-3002045	4.91	1	4.91
Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU	Adafruit	Adafruit	5543	19.95	1	19.95
Adafruit Micro SD SPI or SDIO Card Breakout Board	Adafruit	Adafruit	4682	3.50	1	3.50
PCB (OBC1), pack of 5	PCBWay	PCBWay	–	5.00	1	5.00
PCB (OBC2), pack of 5	PCBWay	PCBWay	–	5.00	1	5.00

Table C.4: Bill of materials for on-board computer (OBC) and attitude determination system (ADS).

Item Description	Supplier	Manufacturer	Part Number	Unit Price	Quantity	Total Price
SM940K09L solar cells	DigiKey	ANYSOLAR	SM940K09L	7.12	8	56.96
Schottky diode, 2A 40V 290pF	Mouser	Toshiba	CUHS20S40,H3F	0.36	4	1.44
Picoblade 2-position right-angle header	Mouser	Molex	53048-0210	0.38	6	2.28
Picoblade 2-position cable, 20cm	Adafruit	Adafruit	4922	0.75	4	3.00
PCB (PV1), pack of 5	PCBWay	PCBWay	–	5.00	1	5.00
PCB (PV2), pack of 5	PCBWay	PCBWay	–	5.00	1	5.00

Table C.5: Bill of materials for solar panels.

Item Description	Supplier	Manufacturer	Part Number	Unit Price	Quantity	Total Price
Capacitor 22 pF 0402	DigiKey	Murata Electronics	GRM1555C1H220JA01D	0.10	2	0.20
Capacitor 10 nF 0402	Mouser	Vishay	VJ0402Y103KXAAC	0.22	3	0.66
Capacitor 1 uF 0402	DigiKey	Samsung	CL05B104KP5NNNC	0.10	4	0.40
Capacitor 10 uF 0603	DigiKey	Samsung	CL10A106MQ8NNNC	0.11	2	0.22
Capacitor 100 nF 0402	DigiKey	KYOCERA AVX	04023C104KAT2A-62	0.37	10	3.70
Capacitor 22 uF 0603	DigiKey	Murata Electronics	GRM188R60J226MEA0J	0.04	5	0.18
Capacitor 1 nF 0402	DigiKey	YAGEO	AC0402FRNPO9BN102	0.21	2	0.42
Capacitor 4.7 uF 0603	DigiKey	Samsung	CL10A475KP8NNNC	0.10	1	0.10
Capacitor 1 uF 0603	DigiKey	Samsung	CL10B105KP8NNNC	0.10	1	0.10
Capacitor 220 uF 1206	DigiKey	Murata Electronics	GRM31CC80E227ME11K	0.92	1	0.92
NeoPixel LED, pack of 5	Adafruit	Adafruit	2659	4.95	1	4.95
LED 0603, red	DigiKey	Oriential Technology	BND0603JKRS001	0.07	1	0.07
LED 0603, green	DigiKey	Oriential Technology	BND0603JKGS001	0.07	1	0.07
5V TVS Diode SOD-32	DigiKey	Littelfuse Inc.	SD05-01FTG	0.40	1	0.40
Schottky diode	DigiKey	Toshiba	CUS10S30,H3F	0.35	5	1.75
Schottky diode	DigiKey	Diodes Incorporated	1N5819HWQ-7-F	0.40	1	0.40
Fuse 3A 0603	Newark	AEM	T0603FF3000TM	0.89	10	8.89
Socket header connector 1x05 1.27mm pitch vertical	DigiKey	Samtec	TMS-105-01-G-S	1.19	1	1.19
Picoblade 2-position right-angle header	Mouser	Molex	53048-0210	0.38	3	1.14
Pin header connector 2x20 1.27mm pitch SMD	DigiKey	Harwin	M50-3602042	4.19	1	4.19
Pin header connector 1x02 1.27mm pitch thru-hole	DigiKey	Harwin	M50-3530242	0.15	1	0.15
Inductor 22 uH	DigiKey	ECS	ECS-MPI4040R4-220-R	1.40	1	1.40
Power inductor 2.2 uH 1210	DigiKey	Murata Electronics	1277AS-H-2R2M=P2	0.40	1	0.40
Power inductor 3.3 uH 1210	DigiKey	TDK	NLV32T-3R3J-PF	0.15	1	0.15
Dual P-channel MOSFET	DigiKey	Vishay Siliconix	SQ3989EV-T1_GE3	0.57	1	0.57
Power P-channel MOSFET	DigiKey	Diodes Incorporated	DMG2305UX-7	0.39	1	0.39
Resistor 100k 0402 1%	DigiKey	YAGEO	RC0402FR-07100KL	0.10	7	0.70
Resistor 510R 0402 1%	DigiKey	Stackpole	RMCF0402FT510R	0.10	2	0.20
Resistor 10k 0402 1%	DigiKey	Samsung	RC1005F103CS	0.10	1	0.10
Current shunt 0.033R 0603	DigiKey	Vishay Dale	WSL0603R0330FEA	1.09	1	1.09
Current shunt 0.002R 0603	DigiKey	Ohmite	LVT06R0020HER	0.71	1	0.71
Resistor 1M 0402	DigiKey	Panasonic	ERJ-2RKF1004X	0.10	2	0.20
Resistor 330R 0402	DigiKey	YAGEO	RC0402FR-07330RL	0.10	2	0.20
Resistor 1k 0402	DigiKey	YAGEO	RC0402FR-071KL	0.10	3	0.30
Resistor 22.2k 0402	DigiKey	NIC	NRC04J222TRF	0.10	1	0.10
Resistor 2.2M 0402	DigiKey	YAGEO	RC0402FR-072M2L	0.10	1	0.10
Resistor 820k 0402	DigiKey	YAGEO	RC0402FR-07820KL	0.10	1	0.10
Resistor 5.1k 0402	DigiKey	Panasonic	ERA-2AEB512X	0.17	2	0.34
Resistor 10M 0402	DigiKey	Stackpole	RMCF0402FT10M0	0.10	1	0.10
Resistor 1.5M 0402	DigiKey	YAGEO	RC0402JR-071M5L	0.10	1	0.10
Resistor 47k 0402	DigiKey	Panasonic	ERJ-2RKF4702X	0.10	4	0.40
Resistor 88k 0402	DigiKey	Stackpole	RMCF0402FT88K7	0.10	1	0.10
Resistor 12k 0402	DigiKey	Panasonic	ERA-2AEB123X	0.17	2	0.34
Resistor 3.3k 0402	DigiKey	Panasonic	ERA-2AEB332X	0.17	1	0.17
Push Button side mount	JLCPCB	HYP	1TS002G-1800-3500-CT	0.07	1	0.07
SPDT switch	DigiKey	C&K	OS102011MA1QN1	0.61	1	0.61
MCU	DigiKey	Microchip Technology	ATSAMD21G18A-MUT	4.16	1	4.16
32 Mbit flash memory SPI	DigiKey	GigaDevice Semiconductor (HK) Limited	GD25Q32CNIGR	1.11	1	1.11
Current/Voltage monitoring IC	DigiKey	Texas Instruments	INA226AIDGSR	2.88	1	2.88
Step-Up 2A DC-DC Boost Converter	JLCPCB	Advanced Analog Technology	FP6291LR-G1	0.12	1	0.12
Load Switch	DigiKey	Texas Instruments	TPS22918DBVR	0.60	4	2.40
Solar Power Harvester	DigiKey	STMicroelectronics	SPV1040T	4.03	1	4.03
Step-Down 2A DC-DC Buck Converter	JLCPCB	Silergy	SY8089AAC	0.06	1	0.06
Low Vin Battery Charger IC	Mouser	STMicroelectronics	L6924D	2.25	1	2.25
2012 32768 kHz Crystal 12.5pF	JLCPCB	YXC	X201232768KGD2SI	0.26	1	0.26
PCB	JLCPCB	–	–	2.00	5	10.00
PCB Assembly	JLCPCB	–	–	65.79	1	65.79

Table C.6: Bill of materials for DynOSSAT-EDU EPS. Passive components are generic, but sample part numbers and their prices are provided.

Item Description	Supplier	Manufacturer	Part Number	Unit Price	Quantity	Total Price
Sliding backplate PCB	PCBWay	PCBWay	–	5.00	1	5.00
Kill switch (left)	DigiKey	Omron	D2F-L2-A	5.12	1	5.12
Kill switch (right)	DigiKey	Omron	D2F-L2-A1	5.12	1	5.12
Picoblade 2-pin header	Mouser	Molex	53048-0210	0.38	1	0.38
Sheet-metal frame	Xometry	Xometry	–	167.74	1	167.74
CNC-machined L-brackets	Xometry	Xometry	–	101.59	2	203.18
M2x5 SHCS, pack of 100	McMaster	–	91292A005	14.26	1	14.26
M2x0.4, 5 mm low-profile SHCS, pack of 10	McMaster	–	92855A837	19.06	1	19.06
M2x5 hex nut, pack of 100	McMaster	–	91828A111	6.14	1	6.14
M2x0.4 thin hex nut, pack of 50	McMaster	–	90710A020	8.52	1	8.52
M2.5x0.44, 5 mm SHCS, pack of 100	McMaster	–	91292A009	6.00	1	6.00
M2.5x0.45 nut, pack of 100	McMaster	–	91828A113	6.45	1	6.45
M2x0.4, 35 mm SHCS, pack of 5	McMaster	–	91292A337	20.18	1	20.18
No. 3 9/32” unthreaded spacer	McMaster	–	92320A455	1.61	12	19.32
No. 3 5/32” unthreaded spacer	McMaster	–	92320A453	1.58	4	6.32
No. 3 1/2” unthreaded spacer	McMaster	–	92320A286	2.16	4	8.64

Table C.7: Bill of materials for structure (including sliding backplate) and fasteners.

Component	Total Cost
Radio communication system	339.50
OBC and ADS	70.57
Solar panels	73.68
DynOSSAT-EDU EPS	132.10
Structure and fasteners	501.43
PCBWay shipping	20.00
JLCPCB shipping	18.00
RockBLOCK monthly line rental	13.87
RockBLOCK 100 credits	15.47
Total	**1184.62**

Table C.8: TigerCub total cost summary. Includes cost of shipping PCB components and RockBLOCK monthly line rental and 100 credits.