Portfolio of Design Evidence

ES410 Group Project

Sebastian Fieldhouse (1922280), Harvey Holliday-Williams (1903833),

Zitai Jin (1922672), Adam Rich (1839791),

Pritesh Shah (1918930) and Joshua Wong (1802336)

Supervisor: Dr. William Crofts

School of Engineering

University of Warwick

Abstract

This project aimed to look at the aspects of designing a first iteration of a CubeSat, housing a biological payload. Overall a model was created, which adheres to all laid out requirements and system budgets. Though the project was successful there were many questions laid out at the end as to whether the design is optimal, as such it is left to the next WUSAT team to further refine the design.

Keywords: CubeSat, Biological, Concurrent Engineering

Abbreviations

Attitude Determination and Control System	ADCS
Concept Of Operations	CONOPS
Commercial Off-The-Shelf	COTS
CocoonSat	COC
CubeSat Design Specification	CDS
Direct Energy Transfer	DET
End-Of-Life	EOL
Electrical Power System	EPS
European Space Adgency	ESA
Fluorescent Deep Space Petri-Pod	FDSPP
Finite Element Analysis	FEA
International Telecommunication Union	ITU
Low Earth Orbit	LEO
Launch and Early Orbit Phase	LEOP
Maximum Power Point Tracker	MPPT
NanoRacks CubeSat Deployer Interface Definition Document	NRCSD IDD
North American Aerospace Defence Command	NORAD
On-Board-Computer	OBC
Power Conditioning and Distribution Unit	PCDU
Radio Frequency	RF
Right Ascension of the Ascending Node	RAAN
Solar Radiation Pressure	SRP
Space Park Leicester	SPL
University Of Exceter	UOE
Ultra High Frequency	UHF
United Kingdom Launch Services Ltd	UKLSL
Warwick University Satellite Team	WUSAT

Contents

List of Figures

List of Tables

1 Introduction

CubeSats are a class of nanosatellite that are built to standard sizes and form factors, known as units (U), and can range from 1U to 12U [5]. They are developed by two professors, Prof. Jordi Puig-Suari from California Polytechnic State University, and Prof. Bob Twiggs at Stanford University [6]. CubeSats are typically flown as a secondary payload of other planned missions [1]. To interface between the Launch Vehicle of the mission and the CubeSat, a CubeSat dispenser known as Poly Picosatellite Orbital Deployer (P-POD) is used [6]. All the above means a reduction in cost and development time, making CubeSat more accessible to university students and amateurs. As of mid-2018, more than 2100 CubeSats and nanosatellites have been launched, and the team aims to do the same with the WUSAT-4 project [7].

All design features and justification of the WUSAT-4 preliminary design are detailed in this design portfolio. The portfolio is divided into subsystems, namely; Systems Engineering, Onboard Computer, Electrical Power Supply, Communications, and Chassis Design. This report will outline evidence of design work carried out by the team, and highlight the deliverables produced during the year.

1.1 Scope

This project aims to work with Space Park Leicester (SPL) to support their payload of a biological experiment or Fluorescent Deep Space Petri-Pod (FDSPP), as it is launched into low earth orbit (LEO). The main mission of the FDSPP is to take images of the specimens held inside and transmit them back to earth. The WUSAT team aims to aid in this endeavour through the use of a CubeSat, thus being able to provide power and communication capabilities to the FDSPP. Utilising the lessons learned from past WUSAT teams, WUSAT-4 (COCOONSat) will help take another step towards human space travel.

1.2 Collaboration with Industrial Partners

Throughout the duration of the WUSAT-4 project, the team maintained close collaboration with various industrial partners and organisations. These are essential to the progress made to the project and they will be discussed in further details in this section.

1.3 Space Park Leicester (SPL) and University of Exeter (UoE)

The aim of the WUSAT-4 project is to integrate and launch a biological experimental payload, known as the Fluorescent Deep Space Petri Pod (FDSPP). FDSPP is the primary payload of the WUSAT-4 project and it is co-developed by Space Park Leicester and the University of Exeter. Therefore, throughout the project, the team has liaised and worked closely with both SPL and UoE to ensure a smooth integration of the payload into the designed CubeSat chassis.

For example, review meetings were conducted with the responsible engineers from SPL and UoE understand the payload requirements, which includes:

- Power consumptions
- Communication requirements
- Onboard data handling requirements
- Thermal requirements
- Ensuring dimensions of the FDSPP envelop is compatible with the designed CubeSat

The team also presented design and progress updates to SPL and UoE representatives during review meetings and raise any concerns or issues regarding to payload integration. Moreover, the WUSAT team conducted a day visit to Space Park Leicester meet with the SPL engineer who is working on the project. This visit was essential as it allowed the team to observe the physical model of the payload (FDSPP). This improved the team's physical understanding of the payload, and how it can be integrated into the chassis. An image of the physical FDSPP model is shown in Figure 1 below. During the visit to SPL, the WUSAT-4 team also presented the progress made on the satellite design to SPL engineers, and the following topics were discussed:

- WUSAT-4 systems engineer conducted a review of requirements and concepts of operations
- WUSAT-4 mechanical engineers presented CAD model of the existing design, addressed any potential interfacing issues. Ensure compatibility of dimensions between the payload and chassis.
- Discussion on data handling methods, including requirements for image transmission and housekeeping data.

SPL engineers provided feedback and technical support for the team to work on and pointed out potential areas for improvement. This was beneficial as the team was able to focus on areas that requires further work and queries regarding the payload were also answered. The team was also able to gain industrial knowledge through SPL engineers, which was extremely valuable to the team as this is not readily accessible through university studies.

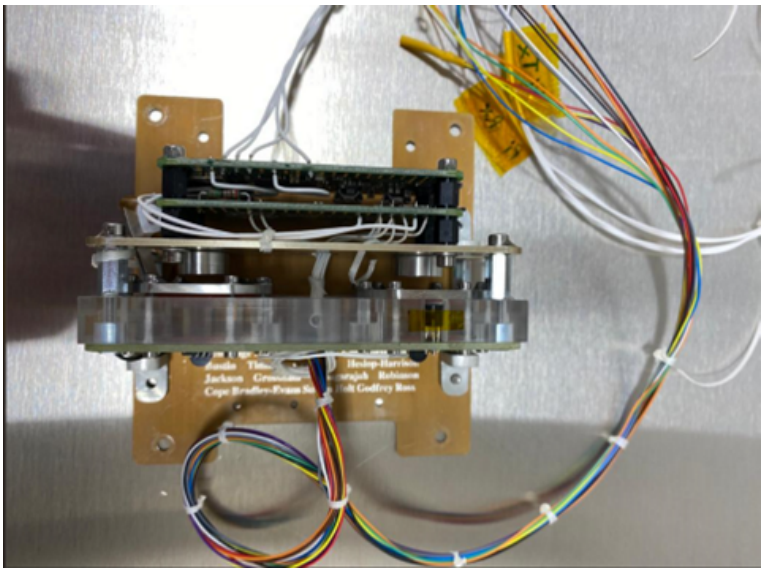


Figure 1.1: Image of the physical model of the FDSPP

1.4 Airbus Defence and Space

The WUSAT-4 team also worked closely with Airbus Defence and Space throughout the duration of the project, engineers from Airbus provided technical support to work of the team. For example, Airbus engineers reviewed design work completed by the WUSAT-4 team such as the link analysis and project requirements. Feedbacks were provided to the WUSAT-4 team by Airbus, allowing the team to improve on the quality of work completed, ensuring that it is technically accurate, and is compliant to industrial practices. The team also conducted fortnightly progress meetings, where updates and progress made were presented to the project supervisor (Dr Bill Croft), also raising any technical problems faced. These meetings were attended by Airbus engineers, again providing the team with technical support and industrial expertise. In November 2022, the team also conducted a site visit to the Airbus and Defence and Space located in Stevenage, UK, as shown in Figure 2. During the visit, the team visited testing and assembly facilities used for satellite production, such as clean room and thermal vacuum testing chambers. The team also presented the WUSAT-4 project to various Airbus engineers and received feedbacks. Most importantly, the team participated in a system engineering workshop, where the team was required to design a satellite using commercial off the shelf (COTS) products based on mission requirements set by Airbus. This was beneficial as the team gained experience and knowledge in using COTS products for CubeSat satellite design process, which can be directly applied to the WUSAT-4 project. Finally, the team also attended a lecture on thermal modelling presented by an Airbus thermal modelling specialist. This was again advantageous to the team as the knowledge gained in this area was applied when conducting thermal analysis during the design process.

1.5 European Space Agency (ESA) [N]

In February, 4 members of the WUSAT-4 team attended ESA's weeklong CubeSat Concurrent Engineering Workshop 2023, held in Belgium. Throughout the workshop, WUSAT members worked under time pressure in different subsystem teams along with other university students to design a CubeSat mission. The mission was a technology demonstration for close proximity active debris removal in low earth orbit. This was achieved by

Figure 1.2: WUSAT Team Members at Airbus

designing two 12U CubeSats, a chaser satellite and a target satellite, to demonstrate how the target can be captured by the chaser. Throughout the design process, WUSAT-4 members worked in a concurrent engineering environment, in different subsystem teams, namely configuration, communication, thermal, and power systems. Team members work collaboratively to resolve design issues and developed different subsystems of the satellite simultaneously which fulfilled the mission requirements. The knowledge and experience gained through this workshop was applied to the WUSAT-4 project.

As shown in Figure 3, the team also had the opportunity to present the WUSAT-4 project to ESA engineers during the workshop to receive feedback.

Team members also attended lectures on various topics related to CubeSat engineering, these are listed as follows:

- Concurrent Engineering methodologies
- Reliability, Availability, Maintainability and Safety (RAMS) Engineering. Relevant RAMs tools such as Failure Mode Effect Analysis (FMEA); Fault, Detection, Isolation, Recovery (FDIR) Analysis; and Hardware and Software Interaction Analysis (HSIA)
- CubeSat Architectures and Technologies
- Fly Your satellite (FYS) Programme

The knowledge gained in these lectures were applied to this project when conducting risk analysis and mitigation for the WUSAT-4 mission. The team also gained understanding of the FYS programme which would allow the team to better prepare for the upcoming application to be accepted onto the programme, further advancing the chance of launching the WUSAT-4 satellite with the support of ESA.

1.6 UK Launch Services Limited (UKLSL)

As detailed in the week 10 project brief, the original plan of the WUSAT-4 project was to integrate and launch two experimental payloads. Firstly, the FDSPP developed by Space Park Leicester and University of Exeter. Secondly, the BEAP (Beacon for Evaluation of

Figure 1.3: Team members presenting the WUSAT-4 project

Attitude and Position) developed by the UK Launch Services Limited. However, after further communication with UKLSL, it was discovered that the technology readiness level of BEAP is not advanced enough to be incorporated into the WUSAT-4 satellite. Therefore, it was decided by the team that the FDSPP will be the only payload integrated of the WUSAT-4 project.

2 Design Overview

As per the classical approach to system design the problem should first be fully specified. For this mission the main considerations include the CubeSat Design Specification (CDS) [6], the NanoRacks CubeSat Deployer Interface Definition Document (NRCSD IDD) [8] and the Fluorescent Deep Space Petri-Pod (FDSPP) datasheet, "Design Portfolio Submission - 6.FDSPP - FDSPP Data Sheet Summary (005).pdf". From each of these sources requirements need to be derived to ensure that the CubeSat adheres to both ESA and NanoRacks launchers, thus it can be placed on most launch vehicles.

For clarity in the listing and use of requirements, the following sections were used:

- Mode, describes the general state the satellite is in where only specific functions are performed.
- Subsystem, describes the relevant parts needed to adhere to this requirement.
- Tags, describes any of the teams which need to know about the requirement, allowing for easier sorting.
- Source, describes where the requirement came from.
- Requirement Number, used to specify the requirement uniquely.
- Related Requirements, lists any requirements with similar content.
- Comments, states any notes to clarify the requirement.
- Requirement, an express statement which describes the need for the mission.
- Justification, describes why the requirement is needed.
- Verification Method, explains a method to ensure the requirement has been met.

These sections easily allow for traceability of where a requirement came from and simple sorting using the in-built excel filtering feature. This better helps other groups only read sections relevant to them, and not waste time by reading every requirement. The full requirements table can be seen in "Design Portfolio Submission - 9.Subsystems - 9.6 Systems Engineering - Project Requirements-221109 WUSAT-4 Requirements.xlsx"[C].

From the documentation, the CDS and NRCSD IDD requirements were explicitly stated, though only the potentially relevant requirements towards this mission were chosen to be added to the table.

Table 2.1: Section of the Requirements Document

Requirement No	Requirement	Justification	Verification Method
COC-2	The payload shall receive 6V and above, and 5V DC from the supply.	As required by FDSPP Data Sheet	Comprehensive inspection of voltages delivered to payload.
COC-8	The CubeSat shall send parts of packeted images to the ground station.	Prevents full images from having to be sent, which could not be received without error.	Test software can send a packet through the antenna to a receiver.
COC-15	The payload will not exceed 37 degrees and fall below -80 degrees.	To prevent biological payload death.	Perform a thermal vacuum bakeout test.

The FDSPP requirements were inferred through the datasheet and discussion with members of Space Park Leicester (SPL). A section of the requirements can be seen in Table 2.1, illustrating some of the most influential requirements imposed on the mission.

Additionally, a requirement from the Internation Telecommunication Union (ITU) stating bandwidth limitations was also included as this body will be used to licence a frequency, thus it must also be considered.

Once the requirements were defined a requirements review was done with SPL and each requirement was confirmed and checked with them to ensure they met their expectations.

2.1 Concept of Operations (CONOPS)[J]

Utilising the requirements laid out in "Design Portfolio Submission - 9.Subsystems - 9.6 Systems Engineering - Project Requirements - 221109 WUSAT-4 Requirements.xlsx" an idea of the phases that will be used for the mission can be devised.

Pre-Launch Phase This is the point before the satellite is launched, allowing for various inspections to be done to ensure it adheres to any requirements specified. All functions of the satellite will be usable during this phase.

Launch Phase When launched the satellite will have all powered systems off and not be communicating with the ground. When being deployed it should utilise a deployment mechanism to separate from other CubeSats.

Early Operations Phase After deployment, the satellite will begin orbiting the earth, where it will engage its solar panels, processor, and heater, and then deploy the antenna. The passive hysteresis system will start detumbling the satellite. At this point, a connection should be established to the ground station.

Commissioning Phase Once the ground station has been communicated with the payload will be commissioned and general housekeeping data should be sent to the ground station.

Processing Phase Once the payload has finished its operation an image will be requested from it, either at a specific time or the midpoint and either high or low resolution. This image will then be packaged and prepared to be transmitted. Whilst this is happening housekeeping data will be sent to the ground station.

Additionally, the On-Board Computer (OBC) will process any stored unhandled telecommands at this point.

Transmit Phase The packeted image data will then be sent, along with housekeeping data to the ground station, upon the next successful heartbeat signal.

Receive Phase After 400s of transmission, the transceiver will begin receiving signals and thus send any sent telecommands to the OBC to be stored.

End-Of-Life (EOL) Phase At the end of the mission, the CubeSat will receive the 'Kill Command' to kill the worms. After sending back images of this it will deplete its batteries and then fall back to earth within 5 years.

Safe Mode In the case of an emergency, or unexpected operation the telecommand to enter Safe Mode can be sent. Whilst in this mode the payload will be placed in standby, though housekeeping data and telecommands will still be sent and received.

These nine modes make up the method to perform, they are visualised in Appendix F.

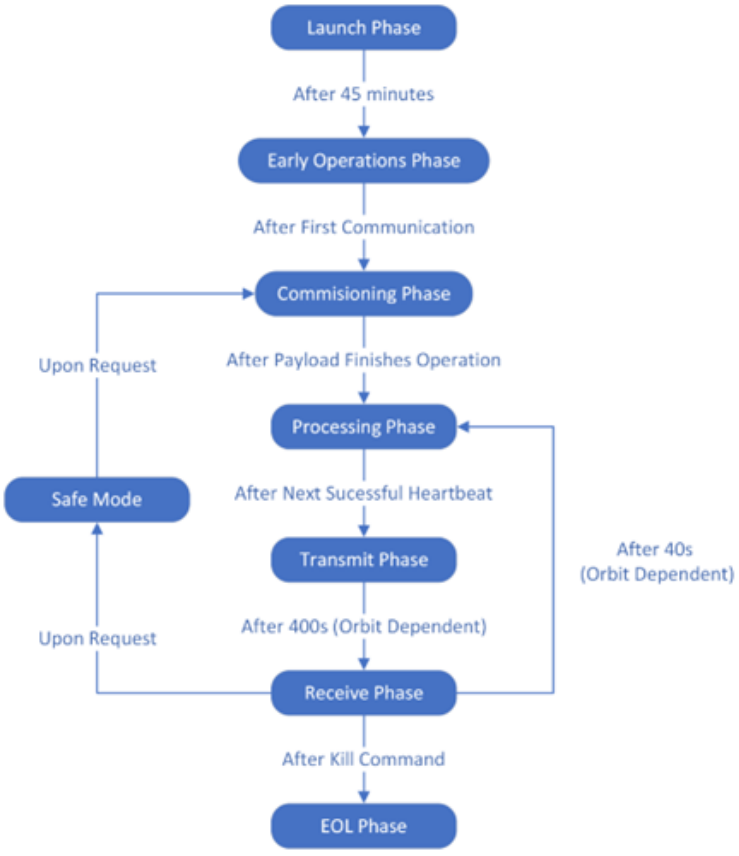


Figure 2.1: Phase Transition Diagram

When put together the links between them can be seen in the Phase Flow Diagram in Fig **??**.

The transition between the Launch Phase and Early Operations Phase, of 45 minutes, was chosen due to the CDS requirement number 2.4.5 of zero transmission or generation of a signal earlier than 45 minutes after on-orbit deployment.

To move from the Early Operations Phase to Commissioning Phase initial contact with the ground station was chosen as the cause. From [9] it can be seen that most CubeSat missions failed due to lack of contact, to mitigate this the first contact will be mission-critical to ensure it occurs. Additionally without commissioning the payload the chance of error, from other systems is reduced.

After commissioning is finished the satellite waits for the payload to finish its 30-minute procedure. The end of this procedure is thus the start of the processing phase. As there could be several orbits without access, after processing the images, transmission should not begin until a connection is established. This acts to also not waste energy transmitting into empty space.

The access time for the satellite is dependent on its orbit, thus it must be calculated accordingly. For the assumed orbit, stated later in the Trajectory Analysis section, this came to a mean time of 440s (including margin), thus 400s of this time is used for transmission.

Similarly, the receive time is defined by the access time of the satellite and uses the remaining time of 40s. As after the receive phase there could be a telecommand needed to be processed the processing phase is entered, after receiving telecommands. The Safe Mode is entered after the 'Safe Mode' command is received and only exited after the 'Normal Mode' command is received.

2.2 User Diagram

Utilising the ideas presented in the requirements, CONOPS and phase transitions a user diagram can be created to show how each 'actor' or user of the system can interact with the satellite. This is illustrated in Fig 2.2.

From the top of the diagram: the ground station can send a command, 'Safe Mode', which would cause a transition into the Safe Mode Phase. In this case, the payload would receive a 'StandBy' command from the OBC.

Once the Safe Mode has been entered, sending a command 'Normal Mode' would cause the satellite to enter the commissioning phase. Thus, the payload would receive a 'Normal' command from the OBC.

The ground station can also send a command 'Kill Worms', which causes the payload to receive the same command from the OBC. This causes the OBC to terminate the worms and thus the satellite would move to the end-of-life (EOL) phase.

Additionally, the ground station can send a command 'Send High/Low Resolution Picture at xx:xx:xx'. Where high or low is chosen and indicates the image resolution of 4608x2592 or 256x256 respectively. 'xx:xx:xx' indicates a timestamp for a point during the mission. This command is sent to the payload, from the OBC, and causes the image at that time to be sent to be stored and then processed in OBC. After it will be transmitted.

During the Early Operation phase, the ground station needs to be able to receive a connection from the satellite if it is lost for a longer period of time. If a heartbeat signal is picked up from the satellite a connection needs to be established where the housekeeping data of the satellite and payload is sent to ensure correct operation.

For the payload, it needs to be able to send housekeeping and picture data to the OBC computer both automatically and by request.

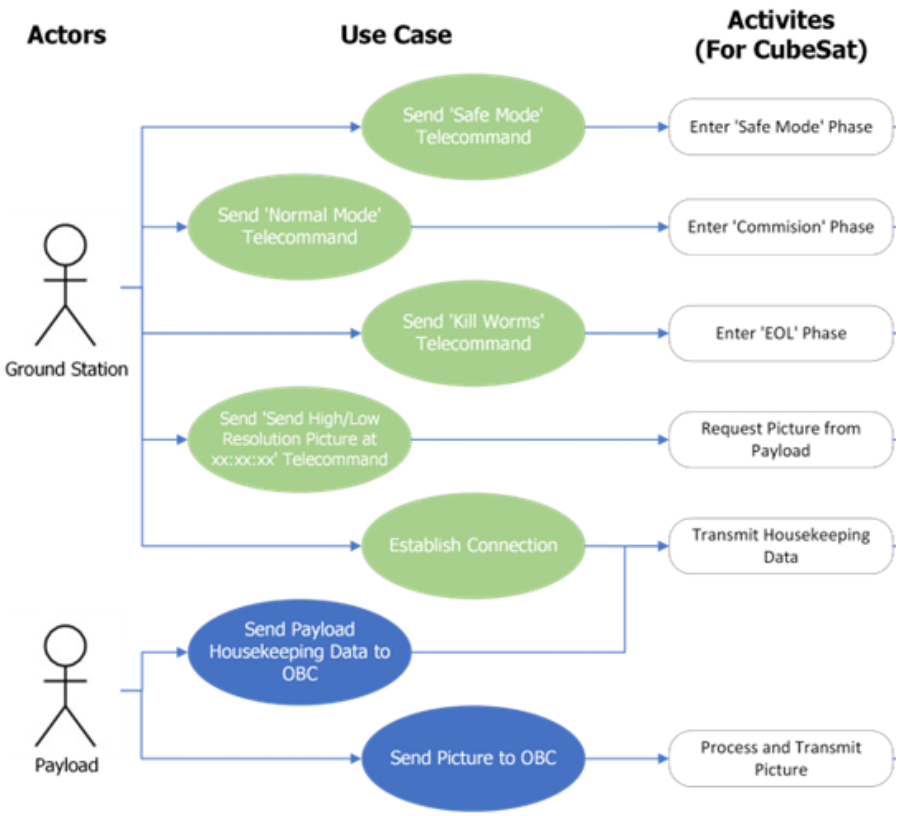


Figure 2.2: Phase Transition Diagram

An extended version of Fig 2.2 can be found in G, which further describes the effects of each of the actor's actions.

2.3 System Level Architecture

Having now devised the need for the system, the users, and the modes of operation a diagram of subsystems can be created. The full diagram can be seen in H, though the general overview will be discussed here. As more was known about each subsystem and its interconnections, more information was added.

The main subsystem is the payload, as it contains the biological experiment. This can both send and receive data from the On-Board Computer (OBC). It also requires power from the Electrical Power System (EPS), used to power its Raspberry Pis and the heaters.

The OBC receives, stores and processes images into packets. The OBC also receives commands and sends telemetry data to the transceiver. Similarly, it sends commands and receives images and housekeeping data from the payload. It also receives housekeeping data from the EPS.

The EPS provides power to other subsystems within the satellite, which are the payload, OBC, communications system and thermal system. It also sends temperature sensor readings and battery charge states to the OBC.

The Communication System is used to communicate with the ground station, consisting of the transceiver and antenna. It uses packeted data sent to it by the OBC and sends this out, in the form of a radio frequency (RF) wave, to the ground station. It also receives telecommands from the ground station, sending these to the OBC for processing.

The Ground Station receives RF waves from the satellite, these contain modulated information about the housekeeping and packeted image data.

The Thermal System acts to ensure the satellite maintains an operating temperature, only requiring power from the EPS.

The Attitude Determination and Control System (ADCS) acts to prevent the satellite from spinning violently upon launch. The use of a passive hysteresis rod achieves this by aligning the satellite with the earth's magnetic field. This subsystem does not require any inputs, as it is passive.

Excluded from this diagram is the Chassis, as it encloses the whole system, though it should still be considered as it must adhere to all the requirements laid out above. It does not require any data or power, though must be physically connected to each system.

3 Trajectory Analysis

Understanding the dynamics of the satellite whilst in orbit is pivotal. If the orbit of the satellite cannot be predicted devising whether a connection can be established between the ground station and the satellite is impossible.

Looking at the worst-case scenario the expected orbit height will be 600km, taken from data on the Exolaunch website, [10] . Thus, the characteristics of an orbit can be calculated, using this height.

3.1 Parameter Definitions

The semimajor axis is defined as the largest distance from the centre of the orbit. In this case, it would be the earth's radius add 600km.

Eccentricity defines a measure of how elliptical the orbit is, calculated as the division of the distance from the centre to foci divided by the length of the semimajor axis. In this case, the orbit is assumed to be circular, thus the value is very small.

The inclination is an angle, comparing the orbit path to a plane through the earth's equator. An assumption is made that a ground station would be constructed at the University of Warwick, to allow communication. This is assumed as it could be feasibly constructed during a future project, as opposed to other ground stations which access would be less certain. Thus, a large inclination is required to pass over the ground station.

The Right Ascension of the Ascending Node (RAAN), as the orbit is a geocentric orbit defined as the Longitude of the Ascending Node. Thus, this value is defined as the starting longitude of the satellite.

Table 3.1: Assumed Parameters for Trajectory Calculation

Parameter	**Value**	**Justification**
Semimajor Axis	6978.14 km	Yields a 600 km orbit.
Eccentricity	1.8731e-16	Assumes a near circular orbit.
Inclination	82 degrees	Ensures the orbit passes over the University of Warwick.
Right Ascension of the Ascending Node (RAAN)	310 degrees	Starts the orbit at a point near an access point.

3.2 Structure and Mission Definitions

To model the orbit of the satellite the, Satellite Communications Toolbox on MATLAB was used, this provided precise enough information for a first iteration estimate.

The ground station was modelled as an isotropic antenna placed on the Engineering Building within the University of Warwick. This has an estimated altitude of 930 meters above sea level. This is shown in Fig 3.1.

The satellite antenna is modelled according to specifications from the Communications subsystem. The main parameter was a maximum viewing angle of 157.25 degrees.

A start time of 1st March 2023 at 12:00:00 and a stop time of 1st June 2023 at 12:00:00 was used. This simulates a three-month-long mission, as this is expected to be enough time to finish the mission, though this length is heavily subject to change.

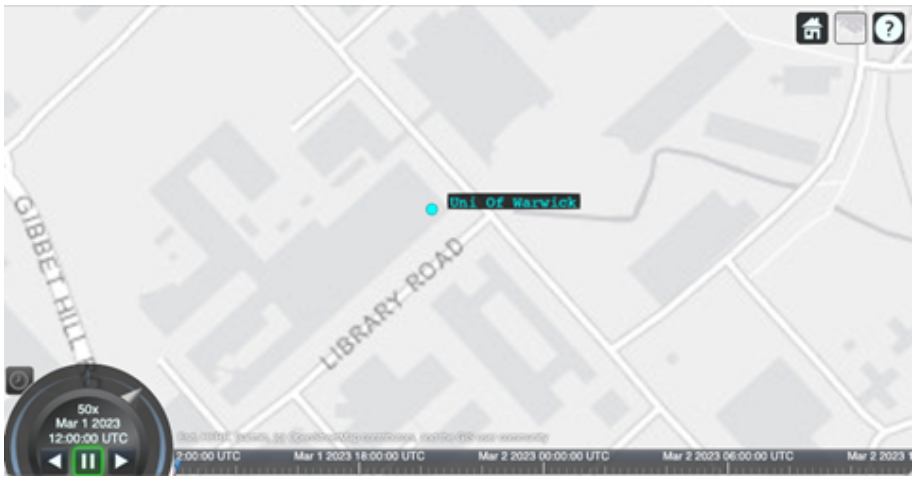


Figure 3.1: Ground Station Location

3.3 Simulation

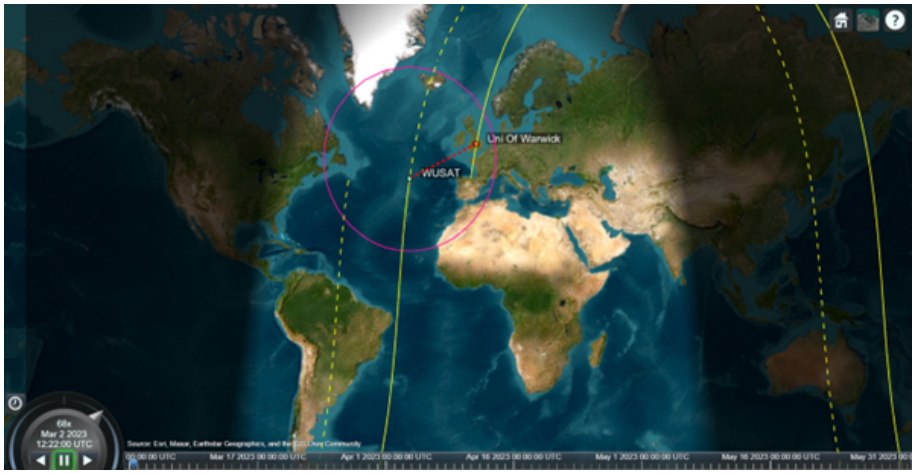


Figure 3.2: Predicted Trajectory of Satellite Compared to Ground Station Location

Utilising the MATLAB code specified in I the simulation was run. The result can be seen in Fig **??**. The solid yellow line represents the past orbit of the satellite, and the dotted yellow line the future orbit of the satellite. The pink circle represents the field of view of the transmitter and the dotted red line is the connection between the ground station and the satellite.

Having run the simulation 3.3, showing a section of the access summary of the satellite, was calculated. The duration is defined by when the time when the ground station comes into range of the satellite, and then exits the range. This result shows that the duration of the access time varies dramatically between each orbit.

Table 3.2: Mean and Total Access Time Over 3 Month Period

Method	**Time (s)**
Total (with 20% margin)	293424
Mean (with 20% margin)	443.24
Minimum	60

From the mission each duration of access was recorded, and the total access time was

Source	Target	IntervalNumber	StartTime	EndTime	Duration	StartOrbit	EndOrbit
"WUSAT TX"	"Uni Of Warwick"	1	01-Mar-2023 12:10:00	01-Mar-2023 12:21:00	660	1	1
"WUSAT TX"	"Uni Of Warwick"	2	01-Mar-2023 13:53:00	01-Mar-2023 13:56:00	180	2	2
"WUSAT TX"	"Uni Of Warwick"	3	01-Mar-2023 18:55:00	01-Mar-2023 19:01:00	360	5	5
"WUSAT TX"	"Uni Of Warwick"	4	01-Mar-2023 20:31:00	01-Mar-2023 20:43:00	720	6	6
"WUSAT TX"	"Uni Of Warwick"	5	01-Mar-2023 22:09:00	01-Mar-2023 22:20:00	660	7	7
"WUSAT TX"	"Uni Of Warwick"	6	01-Mar-2023 23:48:00	01-Mar-2023 23:53:00	300	8	8
"WUSAT TX"	"Uni Of Warwick"	7	02-Mar-2023 09:06:00	02-Mar-2023 09:16:00	600	14	14
"WUSAT TX"	"Uni Of Warwick"	8	02-Mar-2023 10:42:00	02-Mar-2023 10:54:00	720	15	15
"WUSAT TX"	"Uni Of Warwick"	9	02-Mar-2023 12:21:00	02-Mar-2023 12:31:00	600	16	16
"WUSAT TX"	"Uni Of Warwick"	10	02-Mar-2023 19:05:00	02-Mar-2023 19:13:00	480	20	20
"WUSAT TX"	"Uni Of Warwick"	11	02-Mar-2023 20:41:00	02-Mar-2023 20:53:00	720	21	21
"WUSAT TX"	"Uni Of Warwick"	12	02-Mar-2023 22:19:00	02-Mar-2023 22:30:00	660	22	22
"WUSAT TX"	"Uni Of Warwick"	13	03-Mar-2023 00:00:00	03-Mar-2023 00:02:00	120	23	23
"WUSAT TX"	"Uni Of Warwick"	14	03-Mar-2023 09:16:00	03-Mar-2023 09:26:00	600	29	29
"WUSAT TX"	"Uni Of Warwick"	15	03-Mar-2023 10:52:00	03-Mar-2023 11:04:00	720	30	30

Figure 3.3: MATLAB Access Time Variable

calculated. Over the three months, this gave a total, mean and minimum access time shown in Table 3.2. It should be noted that the simulation predicts the longest time with no contact with the ground station to be 38760s, which should be considered before commissioning the payload if the next predicted orbit has this large gap.

3.4 Analysis

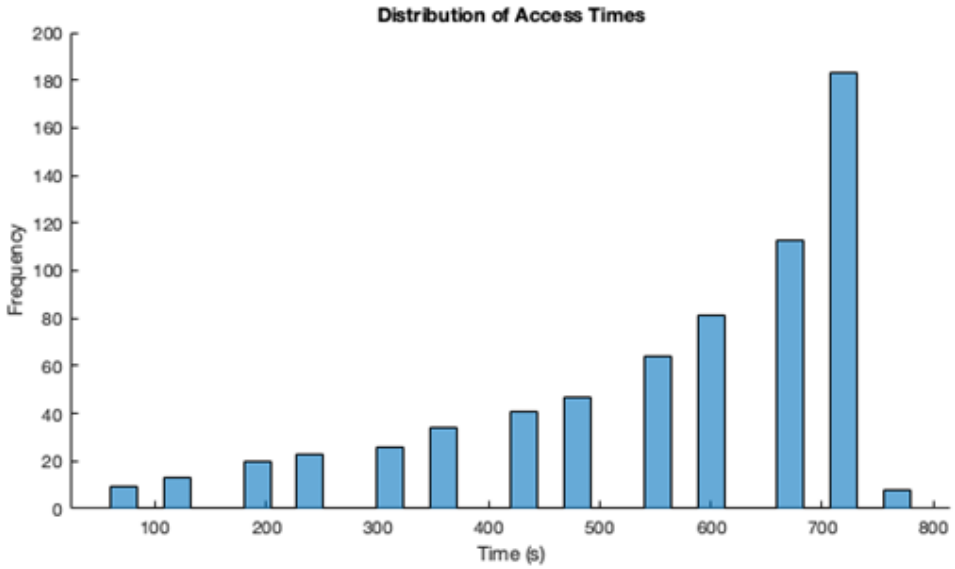


Figure 3.4: Access Time Frequency

Using the results from the simulation, the access times can be seen in Fig 3.4. On average there are many more longer access times, which would allow larger quantities of data to be transferred. This brings to question as to the length of the transmit period being a set time of 400s, if a longer period could be used many more images could be sent.

Table 3.3: Calculations for Number of Sent Images

Method	Resolution	Number of Images Sent
Total	High	479
	Low	87755
Mean	High	0.72
	Low	132

Using the assumptions that the data rate of 19.6 kbps can be established, taken from the communication system, a low-resolution image is defined as 65536 bits and a high

resolution as 12 Mbits. Using these assumptions Table 3.3 was created. It can be seen that many low-resolution images can be sent, though few high resolutions images. This suggests that a large amount of compression may be needed, or cropping, to reduce the high-quality image size in order to make it more feasible for transmission.

3.5 Conclusion

This inspection into the expected worst-case orbit has shown that communication of a ground station based atop the School of Engineering building in the University of Warwick provides enough access time for the orbit. Additionally, it shows that a UHF antenna is suitable for connection in this range. Questions should be taken to the payload as to the resolution and size of the images, as to whether they can be reduced in size, in order to provide a better outcome of the satellite mission.

4 Telemetry Data

Telemetry data for this mission can be split up into two sections, housekeeping data and payload image data. Looking at the sensors within different subsystems an estimate for the size of housekeeping data can be established.

- Solar Temperature Sensor – 32bit at 0.2 Hz
- Battery Capacity – 32bit at 0.2 Hz
- Payload Status – 1 Kbit at 0.2 Hz
- IMU Readings – 32 bit at 1 Hz

This gives a total of 244.8 bits generated per second.
Considering the data generated by the FDSPP camera and image is assumed to be taken at a low resolution and a high resolution. Looking at [11], the high resolution would 4608x2592 pixels and from suggestions by SPL the low would be 256x256 pixels. This generates images of 12Mbits and 65.5Kbits. Only a small number of photos would be taken whilst the payload is in operation.

5 Attitude Control[U]

At the very start of the WUSAT-4 project it was decided that the use of Attitude control should be avoided. Attitude control systems tend to be complicated and require a lot of technical expertise to get right. As a team of undergraduate students, it would be a very difficult task to design an attitude control system for the WUSAT-4 CubeSat. Furthermore, the WUSAT-3 mission was discontinued mainly due to unsurmountable difficulties arising from the design and development of the attitude control system. However, a satellite which has no attitude control system whatsoever has nothing stopping it from tumbling out of control. Large angular velocities would cause internal forces within the satellite. The forces arising from spinning might have a negative effect on the biological payload of the WUSAT-4 mission. There are many possible ways in which uncontrolled tumbling could negatively affect the WUSAT-4 CubeSat. As such, it is the duty of the engineers designing the CubeSat to do due diligence on whether it is safe and appropriate for the CubeSat to have no attitude control system. To begin with, it should be established what angular velocities the CubeSat can be expected to be subject to during its operation. The highest angular velocity will most likely be immediately after ejection from the launch vehicle, as the spring release mechanism will impart momentum upon the CubeSat. The other event during the satellite mission which could cause the satellite to spin is if an antenna (like a turnstile antenna) has to be extended. Focusing on post-ejection angular velocities, Ahmed Khan et al. [12] estimate that a 2U CubeSat that had just been launched from a launch vehicle could have an indicative angular acceleration of around 1.2 rad/s. Similarly, Tumisang Ramodimo et al. [13] use 0.8 rad/s around each axis as an estimate for the 2U CubeSat in their simulations. Alternatively, the Aalborg University CubeSat team – one of the first teams to ever launch a CubeSat – estimated a maximum angular velocity of 0.1 rad/s for their 1U CubeSat [14]. Based on these sources, it will be assumed that the WUSAT-4 CubeSat will not have an initial angular velocity of more than 2 rad/s. Thus, it should be established what internal forces this angular velocity could create within the CubeSat.

5.1 Methods for effecting attitude control

Attitude control systems can be split into two broad categories, active attitude control and passive attitude control. Generally speaking, active attitude control requires an input of energy from the satellite in order for it to function, whereas passive control systems do not. As such, active attitude control comes along with a host of extra requirements; it must be paired with sensor data and a controller in order to operate. This also implies that if there are any faults within the software that enacts the control algorithm, the sensors which read the pointing direction of the satellite or the systems which actuate the active attitude control (such as thrusters for example), the whole attitude control system will fail. On the other hand, passive control does not require a controller or coordination with attitude determination systems in order to function – this means that it is it is much less complex, and this is what makes it an appealing choice for a university CubeSat project. The main attitude control systems available for nanosatellites are as follows:

5.2 Passive Control Systems

- **Gravity Gradient Stabilization:**this method uses the satellite's mass distribution and the earth's gravitational field to stabilize the attitude of the satellite. The gravitational field strength of the Earth decreases along with the square of the distance from the Earth's centre of mass. As such, for a non-cube satellite, the "lower" part

of the satellite will have a slightly greater gravitational force acting upon it than the "upper" part, causing the satellite to align its axis of minimum moment of inertia with the direction of the Earth's gravitational field lines.

- **Solar Radiation Pressure Stabilization:** this uses the pressure exerted by the sun's rays to control the orientation of a CubeSat in low Earth orbit. Solar radiation pressure is proportional to the surface area of the satellite exposed to the sun and the solar flux, and results in a force that acts on the CubeSat. The orientation of the CubeSat can be controlled by changing the surface area exposed to the sun through adjusting the solar panels or other appendages. However, it is important to note that SRP is only effective at controlling orientation for CubeSats in low Earth orbit, where the solar flux is relatively constant, and may not be as effective in higher orbits or for satellites with irregular shapes. This is not an issue for WUSAT-4, however, since this CubeSat is intended for LEO operation.
- **Passive magnetic stabilization:**this relies on the interaction between a permanent magnet and the Earth's magnetic field. The satellite's magnetization and the orientation of the magnetic dipole relative to the Earth's magnetic field create a torque on the satellite, which will tend to align the satellite's magnetic dipole with the Earth's magnetic field lines. The satellite's magnetic dipole will naturally tend to align with the magnetic field lines, which will cause the satellite to maintain a stable orientation relative to the Earth's magnetic field. The magnitude of the torque will depend on the strength of the satellite's magnetic dipole and the orientation of the magnetic dipole relative to the Earth's magnetic field, so passive magnetic stabilization can be used to help maintain a stable orientation for the satellite in LEO. This method is relatively simple and low cost, but it has limited accuracy and may not be suitable for all missions. However, this method can also cause EMI issues, for instance it inhibits the use of magnetometers as an attitude determination system.

5.3 Active Control Systems

In the case that the satellite is found to require an active control system, it is important to be aware of the possible options available. Furthermore, it is important for the ADCS engineers to be aware of all possible attitude control systems as due diligence on the design of the ADCS.

- **Reaction Wheels:** these consist of a rotating wheel that can be spun in one direction or the other, depending on the desired change in the satellite's orientation. When the wheel rotates in one direction, the resulting angular momentum produces a torque that changes the orientation of the satellite in the opposite direction. By controlling the direction and speed of the wheel's rotation, the satellite's orientation can be precisely controlled. The reaction wheels are mounted on gimbals, which allow them to be rotated independently of the satellite body. This allows for precise control of the satellite's attitude, making reaction wheels an effective active attitude control method for CubeSats.
- **Control Moment Gyroscopes:**these operate on a similar principle to reaction wheels. Where reaction wheels control systems use a change in relative spin of the reaction wheel to create a torque on the satellite, control moment gyroscopes use a change in the axis of angular momentum in a gyroscope to produce a torque on the satellite.

- **Thrusters:** These work by expelling a small amount of propellant in a specific direction, creating a force that changes the satellite's velocity. This change in velocity causes the satellite to rotate and alter its attitude. Thrusters are typically used in combination with other attitude control systems to provide precise and efficient control of the CubeSat's orientation. The advantages of using thrusters include their ability to provide a large amount of torque quickly and their flexibility in terms of the direction in which they can apply force. The main disadvantage is that they consume a limited supply of propellant, so they need to be used judiciously. They are more appropriate for large satellites which have much greater moments of inertia than

- **Magnetorquers:** these are devices used to control the attitude of a satellite by generating magnetic torques on the satellite body. They are typically composed of a coil of wire surrounded by a ferromagnetic material and a power source. When a current is passed through the coil, it generates a magnetic field that interacts with the Earth's magnetic field to generate a torque on the satellite. This torque can be used to control the orientation of the satellite and keep it pointing in a desired direction. By controlling the current through the coil, the magnitude and direction of the magnetic torque can be adjusted to achieve the desired attitude. They are cost effective, and the small forces they generate are appropriate to control the pointing of an object with a small mass like a CubeSat

6 On Board Computer

The aim of this document is to cover the thought process that went into the current design for the microcontroller, its functionality, and how it will be tested before deployment. The document is intended to be continually updated as the design process goes on and other teams introduce their requirements to show a full pathway on the design system as well as inform future teams as to how we arrived at our current decisions.

6.1 The Controller [D]

6.1.1 ARMCortex M3

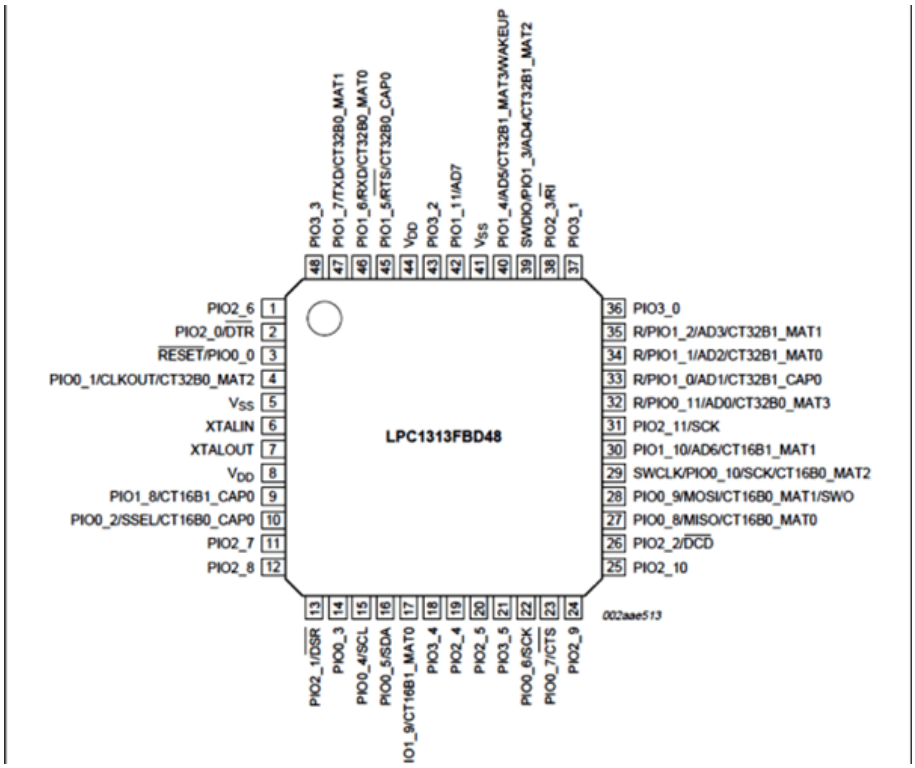


Figure 6.1: Pin diagram for ARMCoretex M3 processor [1]

The first step to figuring out how to get a functioning controller in space is, of course, picking the controller. The ideal controller needs to be able to weather the harsh environment of space and its temperature fluctuations as well as its high levels of radiate, use minimal power and have enough pins to connect to the multiple sensors arrayed in the satellite – ideally it should also be inexpensive. All of these are not ever found together so a trade off must be done between controllers to decide on one for this project.

Initially when discussing the microcontrollers, we were partnered with an industry partner who wanted to track position, telemetry and speed. Given this extra complication to the project and that it would be a 2U CubeSat, the ARM Cortex M3 was chosen for its excellent functionality by way of the fact that it has an inbuilt transceiver, telemetry device and clock. Having all these built into the device would increase the robustness of it as well as possibly increase its functionality and speed. Not only this, but the processing power is much higher than an Arduino as well as having a better memory. One common use of the ARM Cortex M3 is that a standard use is biological telemetry [15], proving it has the legacy ability to optimally measure the telemetry of a CubeSat. The controller

also has a much larger memory (4GB) as opposed to the Arduino (256kB).

The problem with the ARM Cortex M3 is the functionality is extremely high and so is the price. The price of the ARM Cortex makes it inappropriate to be a development board before proper tests and decisions are made.

6.1.2 Arduino Leonardo

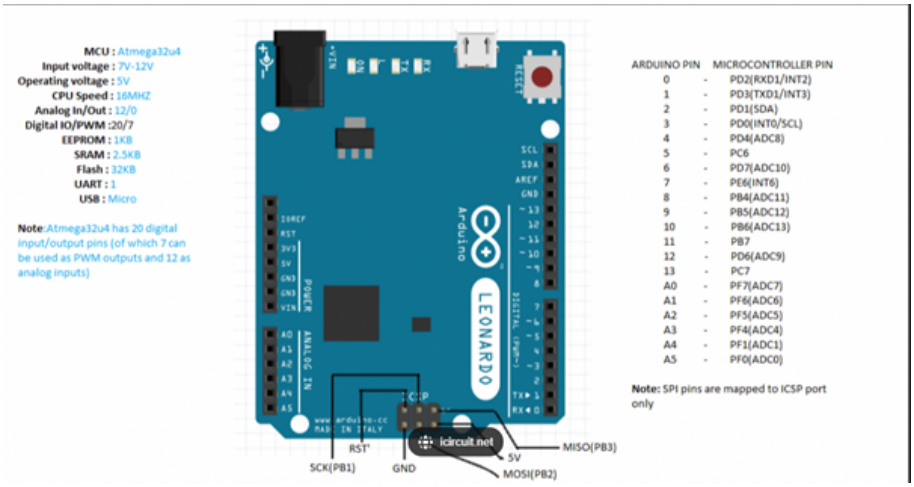


Figure 6.2: Pin diagram for Arduino Leonardo [2]

The Arduino Leonardo is a small but powerful microcontroller which is widely available yet relatively cheap for the performance. The reason it is a good candidate for this project is that it not only is integrable with almost any system, but it also has a wide level of support and community that a student project would find it easy to get any help required., but the in-built libraries and functions, such as the sleep function, allow for the best performance with an inexperienced developer. The size and power usage are minimal making it ideal for space as well as having a wide operating temperature and input voltage. It does not, however, have the radiation resistance of the other options. As well as the ease of integrating and programming such a device, and the low cost, there are other options than the Leonardo that could be changed to with almost zero effort or cost if the requirements change. For example, if the size and power output requirements change, as is likely with the continuation of this project, something like the Arduino Nano which has a smaller power budget, weight and size but less functionality and a higher cost. The drawbacks to having an Arduino are that it doesn't have the space legacy or previous tests that the other microcontrollers possess, as well as lower functionalities, like not having any radiation hardening. The Arduino also has no inbuilt functions unlike the ARM M3 meaning that the microcontroller would have to have manufactured accessories to function, which inherently makes it more susceptible to the vibrations and the damage of initial take off.

6.1.3 RCA1802

The RCA 1802 is a radiation hardened microcontroller which has been used on space missions such as the Galileo spacecraft or the plasma wave analyser instrument on the ESA's Ulysses spacecraft. Given its impressive usage throughout the history of space technologies we would know it has the functionality and ability to cope in space for much more

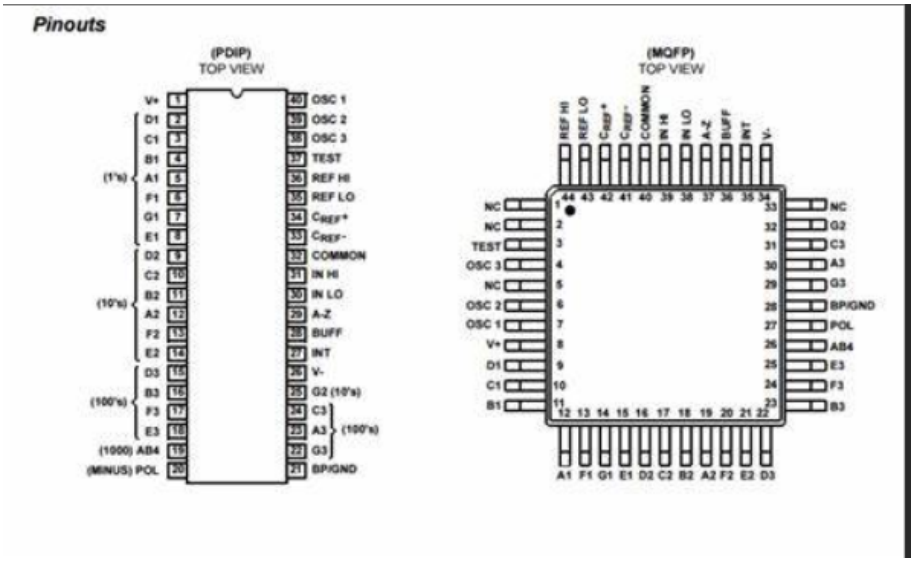


Figure 6.3: RCA1802 pin diagram [3]

complex missions than would be used in WUSAT-4[16]. Not only this but the controller also has a graphics companion which would allow for the complex computation needed to adjust pictures coming to and from the FDSPP biological sample. Because the RCA 1802 also has a wide range of operating voltages it meant it would have more versatility for the beginning of the project when a power supply requirement was still undefined. As with all space related microcontrollers, it was vital that the power consumption remains low especially whilst in modes where most functionality would not be used. The draw backs to the RCA 1802 are that it could be considered too functional for the purposes of this project, meaning that the wide range of abilities and high processing speeds may not be necessary for the aims of the project. There is also a factor of cost and due to the space legacy of the controller it would not only be a more expensive option, but the guarantee of working would be less than optimal. With the technology being older, it also means the compatibility and coding on the system would have to be learned by members of the team - whilst it can be coded in C which some members know -, the integrability of it is simply less than for other devices.

Table 6.1: Pugh Matrix for Controller decision

Key	
1	Above average
0	Average
-1	Below Average

		Processor		
		Arduino	**ARM Cortex M3**	**RCA 1802**
Reasoning	Weighting			
Cost	10	1	-1	0
Functionality	9	0	1	-1
Compatibility	8	1	0	-1
Ease of use	7	1	0	-1
Total		25	-1	-24

The above, 6.1, shows a Pugh Matrix which helped with the decision of which processor should be chosen within this process. Ideally, this will be revisited and updated as the project changes and further requirements are implemented. As this is the beginning of the

project the belief on getting a functional prototype and testing rig was important which helped to decide on the most utilisable and cheapest controller to implement. Not only this, but because the controller will need to integrate with a Raspberry Pi on the FDSPP rig, it is believed that an Arduino will have the best chance at communicating between the two devices (other than a Raspberry Pi which comes at a higher cost without much extra functionality).

6.2 I2C [L]

I2C communication is a protocol by which a master and a slave interface (possibly bi-directionally) to transfer data using two pins, the SDA and SCL pin. An Arduino Leonardo naturally has 2 dedicated I2C communication pins but any 2 of its 13 other digital pins can act to use I2C communication methods [17]. The reason I2C communication is so vital in the project is almost every sensor off the shelf for CubeSats use I2C communication so any device such as the communication unit or the IMU in the solar panels all require the use of I2C. The protocol relies on having the most and least significant byte (MSB and LSB respectively) which help to denote the start and ending transmissions states for the pin whilst reading or writing to a master or slave.

To use I2C in Arduino is fairly simple as there is a library which can create most of what is needed when communicating this way. By using the line *#include ¡Wire.h¿* at the start of the code it will include the public domain library that handles all of the I2C communication methods.

Once this is done the only thing left to do is to set up the pins which are used to communicate using the function Wire.begin(int) where int is the pin numbers you want to use. Other functions like emphWire.read() or emphWire.write() are then self-explanatory to use when you want to read or write for a bi-directional system. Examples of this can be seen in the test 3 and 4 files within the code scripts.

6.3 External Storage

6.3.1 SD Card

For Arduinos, the most common usage is an SD card to be used as they are cheap, reliable, long lasting and have been used for countless projects, such as a “lab ina box project” [18] so have a lot of support. An SD card is often used in for storage devices with cameras and computers as they can hold large amounts of data for a relatively small device. They do however suffer from limited read and write cycles but for the purposes of this project that is not relevant as the number of cycles far exceeds the number required for this project. To use an SD card with an Arduino it would need not only SD cards but an SD card reader also. These are fairly bulky units in terms of a CubeSat taking up 32x24mm if area and having to be connected using 4-5 pins from the Arduino, this is an issue as an Arduino has only 20 pins for connection and will need to be connected to no less than 5 different accessories. There would also have to be some kind of system to ensure the SD card remained securing in the SD card reader through the intense vibration of take off when launching the satellite.

6.3.2 EEPROM

On the other hand, EEPROMs have reduced usability and lifetime as well as being more complicated to use within the Arduino. They do, however, have the advantage of being much smaller in mass and dimension as well as having reduced power usage and having

the ability to be used within a PCB. Given these distinct advantages over the SD card and the fact that the lifetime of an EEPROM is approximately 1000000 cycles or 10 years this is easily long enough for this project. A popular EEPROM that is used in combination with the Arduino Leonardo is the AT24CM02 which is extra useful as the EEPROM of this kind naturally saves each "page" of 256kb of data which means we will automatically have small packages of data that can be sent by the Arduino at a time and thus make it much easier to pause for part of the sending cycle when the satellite moves out of range. Also, the size of an EEPROM is 4x5mm making it extremely small and can be added to a PCB manufactured in the future which will integrate better with the Arduino and make a much sturdier connection.

6.3.3 Memory Storage decision

Given the pros for the EEPROM in a system where weight, power usage ad rigidity of the connections are vital for the success of this project it seems like the most logical choice to choose an EEPROM which will work for our system. Given the size of the images taken by the FDSPP system it is likely that we will need two as EEPROMS tend to only come in 2Mb sizes at their maximum. A good choice of EEPROM would be the AT24CM02 model as it is reliable and has good functionality with the Arduino. Luckily when discussing EEPROMs given their I2C communication functionality it can be recommended that these are "daisy chained" to each other thereby only taking up essentially what would be 1 Arduino slot.

6.4 Other Accessories

Given the OBC will have to communicate with every sensor on board the satellite it is vital that the consistent communication of I2C is maintained throughout so as to ensure ease within both the coding and working generally of the OBC. However, a lot of the measurement systems will be decided by other factors, for instance, the IMU on board will actually be in built to the COTS solar panel which is procured. Given this fact planning for every eventuality and system is impossible but based on the requirements document we know the OBC will need to communicate with an IMU, power measurement system, the FDSPP payload, the external storage and the communication unit. It is likely that the options of these accessories will continue to change as the project moves on so research into each part and how they connect with the OBC will be vital as the other subteams come to their decision. Luckily almost every device is set up to communicate using I2C communications as well as having a 5V or 3.3V input meaning the Arduino is extremely capable of running these systems.

6.5 Printing a PCB

For the OBC, a rigid connection will need to be established to many different systems, as discussed previously, and the connection will need to be extremely robust. Given this, it is thought that printing a PCB for all the connections to be set up would be the best way to assure no connections come loose. Creating a PCB with the External hard drives connected and soldered into position this will reduce the risk of it being shaken apart greatly. Along with this the PCB can be rigidly connected to the Arduino and extend the ports within the Arduino, that would be used for different things, and extend them in separate directions so cable management within the satellite as well as ease of use to someone who didn't design the system would be much more obvious. Given the support from the engineering department at the University of Warwick who have offered to help

design and manufacture PCBs for this project in conjunction with the team means the benefits of the PCB can be brought to the team at zero budget cost. The design for this PCB has not begun yet as it is still unclear as to what components will need to be connected, but once this is confirmed design of the PCB can begin.

6.6 Thermal Concerns

According to the Arduino data sheet (see repository) the OBC function at -400C to 85oC which is a temperature range that will not naturally work in space. Given this it is vital that the thermal team work out a way of ensuring the satellite is insulted against possible thermal issue, most electrical components will also have issues outside of this temperature range.

6.7 Modus Operandi [K]

6.7.1 First orbit

The idea for the first orbit is the Arduino will wait half an hour after release to begin transmitting a "heartbeat signal", which will consist of the Arduino sending a signal every 2.5 minutes (this can be adjusted with the project) so at some point the ground station can be informed that the Arduino is functioning and working as expected. Between each signal the Arduino can enter rest mode as this will be the time that the Arduino could be on its lowest power of the mission, and therefore have the highest chance to fail. From here the ground station will receive an approximate time at which the Arduino is contactable.

6.7.2 Second orbit

During the second orbit a consistent signal will be sent from the ground station to the CubeSat with the Arduino going between sleep and receive modes, this should also be every two and a half minutes. Once the Arduino first receives a signal it will then remain on receive until it no longer receives a signal. After this it will enter a sleep phase for approximately 50 minutes (estimated on random orbit while trying to conserve power, this may need to be looked at) before turning on to it's receive phase. The Arduino will then save when it next receives a signal as well as when it no longer receives a signal once more. Once all this information is combined the Arduino will have the information for the time period it is in communication with the ground station and the length of the orbit as a whole, which can be the first transmission to the ground station, so it has the same information.

6.7.3 Standard orbit

The send/receive portion of the orbit is the only real concern for the Arduino as it will remain in rest mode during other times. In this time, it has been debated whether to send a "start transceiving" and "stop transceiving" signal to the ground station and it was decided this isn't needed but as the project continues this may need to be revisited. The Arduino will automatically switch to a receive function so any information from the ground station can be sent to the FDSPP package. Once a signal is received the Arduino will read it and follow the instruction expected, for instance if the instruction is to ask for an image, the Arduino will ask the Raspberry Pi for the image. The only time it will not switch to receive is if it is interrupted mid communication due to its exiting of the communication range. In this case, instead of being in receive mode it will send a signal saying "ready" once it believes it is in range again, then wait for a signal from the

ground station before continuing with the transmission. When receiving an instruction, the Arduino could be asked to retrieve a file from the FDSPP to send back down to the ground station. In this instance it would be expected that the FDSPP would receive the instruction from the Arduino before transferring the file to the storage on the Arduino where the Arduino can then take the data in discrete packages and send the back down to the ground station. This way the Arduino doesn't need to have a storage the size of the FDSPP and communication methods should become easier. After the information is transmitted the Arduino could then either wipe the data from the storage or tell the FDSPP to do this.

6.7.4 Possible further options

As extras which could be easily added into the already existing code it may be possible for a "recalibrate" function to be added which repeats the first and second orbit routine in case the orbit of the CubeSat changes in some way over time. However, any extra code must be carefully considered as the power draw and failure points will increase with the number of lines of code used within the Arduino so while having many functions may be useful, they have to be weighed up against the risk of failure that each line represents. This risk can be mitigated with appropriate testing but the exponential increase of use case errors means the likelihood of failure could go from low to moderate quickly.

6.8 Testing

Before sending the controller system to space there are some tests that will need to be performed which are controller specific. The main tests to be performed are the code, the hardware, durability and the communications.

6.8.1 The hardware

As a test of the hardware functionality, many mechanical tests can be performed. While each part comes with a data sheet telling us how capable they are of handling heat and radiation, this must be tested in their current set up. Unfortunately, as we know the Arduino as it cannot handle the temperatures of space, other members of the team will have to be further along with their designs to test the protection to get a better estimation of what temperatures will be experiences by the Arduino. A testing method could be using the heat cyclers in the electronic labs which stress test electronics at extreme temperatures to test their performance. Another test that can be done to test the hardware if to run the full code while connected to a battery and oscilloscope, by doing this we can measure the actual power output of the Arduino to be expected in the first set of orbits as well as a standard loop. By doing this a much better estimation of power consumption can be estimated. This can also be used to estimate the efficiency of the sleep mode compared to regular modes to test whether the extra memory is worth the power consumption. Once the rig is complete and manufacturing is decided, a vibrational test will also need to be done to check whether the hardware is likely to break up upon launch. Along with a space radiation check to see if the Arduino can handle to harsh environment of space. By performing these tests, we can see how well we are currently performing and the level to which we are progressing. Updates will be added as the code is confirmed and the tests are passed or failed.

6.9 Next Steps

As the project continues to develop the functioning code can also develop as well as the code testing phases. An important start will be quickly deciding on the instruction protocol for how the ground station will tell the Arduino to do things on board the satellite. After this, buying the components and testing that each one behaves as expected as well as ideally getting a prototype FDSPP to communicate with will be vital for ensuring the success of a launch mission.

6.10 Testing

Within this test the aim is to check that two Arduino Leonardos can communicate with each other over a radio frequency to check that the Arduinos function as we expect them to, the code also verifies the time measurement system of the inbuilt function millis() within Arduino. The transceiver being used is an NRF24L01 which operates at a 2.4Ghz frequency with a range of 1000m.

6.10.1 Set up

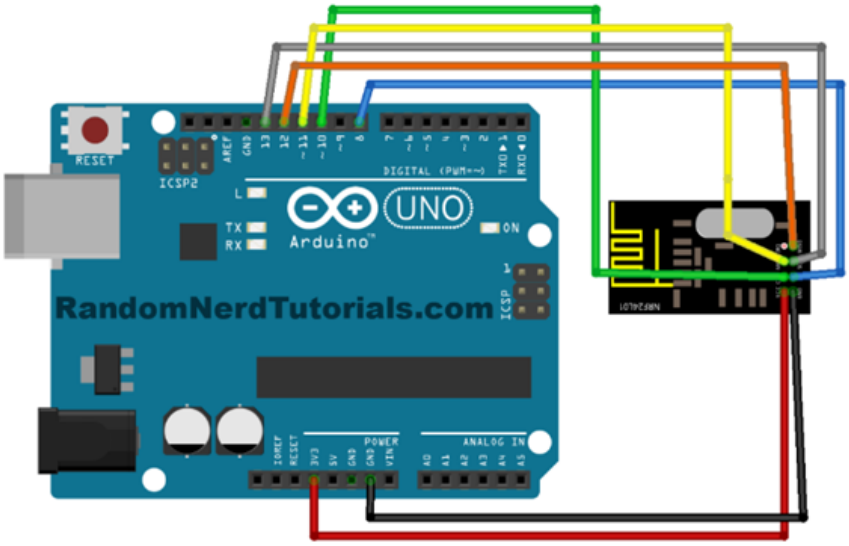


Figure 6.4: Set up for arduino communication test [**?**]

The set up of the Arduino will be as shown in the above figure. The pins will be the same with the Leonardo as the Uno. It is important that the voltage for the transceiver doesn't exceed 3.6V or below 1.9V or the circuit will be fried. Once this set up is tested a test code was run to ensure that there was some functionality to the code that was being written. The code (found in test code 1 and 2 in the microcontroller folder) sets up a system where two Arduinos would communicate in a similar way to how it would within the satellite, with one Arduino acting as the satellite and the other acting as the ground station would. Unfortunately, this will not be the exact communication method used on the satellite as that would use an external communication unit which talks to the Arduino using I2C communication methods.

6.10.2 Code

The full code can be accessed at: A B

Both codes begin with the same set up code:

```
3   #include <SPI.h>
4   #include <RH_NRF24.h>
5
6   RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin
7
8   long double starttime = millis();
9   void setup()
10  {
11    // standard comms speed
12    Serial.begin(9600);
13    //check the nrf24 is connected correctly
14    while(!Serial)
15      delay(1000); //waiting for board
16    if (!nrf24.init())
17      Serial.println("init failed");
18    // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
19    if (!nrf24.setChannel(1))
20      Serial.println("setChannel failed");
21    if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::TransmitPower0dBm))
22      Serial.println("setRF failed");
23  }
```

Figure 6.5: Set up code to begin arduino test

This set up code is used firstly to import the required libraries of SPI.h and RH_NRF24.h. The SPI.h is freely available for all Arduinos and the RH_NRF24.h library can be found (here in reference) or is already loaded into the code files. The RH_NRF24.h library was chosen as it has many of the functions built around the hardware used for this test meaning integration would be as easy as possible. Then a global variable of "starttime" is created which checks the time when the Arduino is switch on and saves it to a global variable in the form long double. After this the command on line 12 starts the communication between the board and Arduino IDE of where "9600" is the bits per second of transferred data. The lines of code between 12-22 are designed to test that the transceiver used within this test is functioning as expected and will therefore be able to communicate. It checks the transceiver is connected, the channel is correct, and the communications settings are as expected for the model being used.

6.10.3 Test code 1 Satellite

The first code is designed to be the satellite function within this test instance. The code begins by sending a message to the serial port saying "Sending to ground station" to the serial port so it is easy to track where in the code the Arduino is. It then loads an "Are you there" message into uint_8 format which is what the transceiver needs to send information between the two transceivers. The code (line 31) then attempts to send the message using the function built into the nrf24 library and waits till the transceiver sends the information. From there the code registers the information of the buffer size of the transceiver and its length so it can check if the transceiver receives anything, the function for this requires these two inputs. After this an if statement is made to see if any information is received by the transceiver; if information is received the serial port should read "Reply:" followed by the received message, alternatively a message will print that no message was received. If no message at all is received after a set town the message will read "We appear to have lost the satellite" before the code waits 1 second and repeats.

6.10.4 Test code 2 - Ground station

Within the second test code , the object imitates what the ground station would function as well as check the clocking of the Arduinos. The code starts by checking if there is a nrf24 available at all. The code then proceeds to fill variables for buffer size and length, as was done in test code 1, before setting the transceiver to receive. After this is an if statement so that if a message is received within 5 seconds of the Arduino turning on it will respond with "Too early!" to the satellite ground station after printing the received message to the serial port. Alternatively, if a message is received after 5 seconds, the ground station will send a message back to the satellite of "Howdy". If none of these happen a message will be printed to the serial port saying "receive failed".

6.10.5 Results

A successful test will have the Arduino outputs as follows: This test makes sure the two

```
Sending to ground station
Reply:
Too early!
Sending to ground station
Reply:
Too early!
Sending to ground station
Reply:
Too early!
Sending to ground station
Reply:
Too early!
Sending to ground station
Reply:
Howdy
```

```
got request
Are you there?
Sent a reply
got request
Are you there?
Sent a reply
got request
Are you there?
Sent a reply
got request
Are you there?
Sent a reply
got request
Are you there?
Sent a reply
```

Figure 6.6: Set up code to begin arduino test

Arduinos are functioning as expected and that important pins work as well as checking the timing system that will be used on the satellite works. Along with this the test just shows a successful communication method over radio waves as well as providing a good introduction to the Arduinos and how code will run.

6.11 Testing

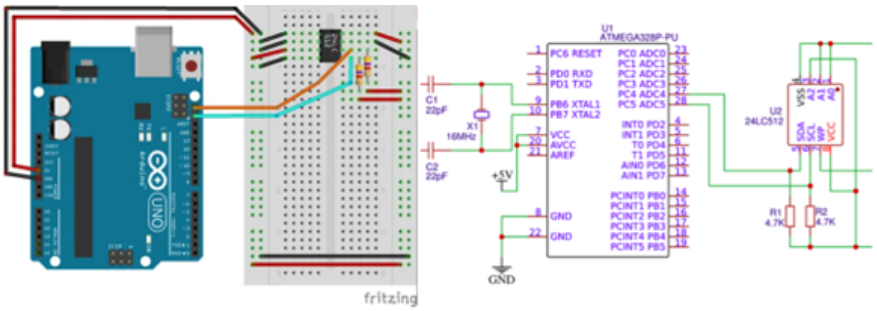


Figure 6.7: Set up for Arduino external storage test [4]

The set up for test 2 is slightly more complicated than test 1 as the above diagram would need to be completed as well as a connection method seen in test 1. Test 2 takes test 1 and extends it as new iterations to the on-board computer were added. This treats the arduinos in the satellite-ground station simulation and must write the information sent by one Arduino to an external memory and later reads that memory to send back to the "ground station". This will be a vital procedure as during the satellite flight this will be how the Arduino sends messages and receives information to pass onto the FDSPP system. Unfortunately, there was not time to complete this test as the external storage decision was done very late in the year so the time to order the parts was not available, but an example output from the code is shown below.

6.11.1 Code

The code for test codes 3 and 4 are the same as they have the same requirements and set up, they can be found here: C and D. **??** shows how the set up is similar to test 1 but has

```
3   #include <SPI.h>
4   #include <RH_NRF24.h>
5   #include <Wire.h>
6   #define eeprom 0x50 //defines the base address of the EEPROM this will
7                       //change depending on how the arduino is wired!!!!
8   long double starttime = millis();
9   RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin
10
11
12  void setup()
13  {
14
15    Wire.begin(); //creates a Wire object
16    Serial.begin(9600);//standard communication speed for board
17    while (!Serial)
18      delay(1000);; // wait for serial port to connect. Needed for Leonardo only
19    if (!nrf24.init())
20      Serial.println("init failed");
21    // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
22    if (!nrf24.setChannel(1))
23      Serial.println("setChannel failed");
24    if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::TransmitPower0dBm))
25      Serial.println("setRF failed");
26  }
```

Figure 6.8: Code for set up of Arduino

the addition of the library for Wire.h which is the library used when I2C communication is used within Arduino as it simplifies the functions needed for communication. The

definition for “eeprom” states that the base address for the EEPROM is at 0x50 (this is due to the shown hardware set up and will change if the set up changes), these can be found in the Arduino datasheet to help define base address depending on connections. The final change is the addition of “Wire.begin()” which opens the communication method using the wire library for the EEPROM.

6.11.2 Test code 3 - Satellite

For test code 3 the code is supposed to simulate the satellite so will be the Arduino reading and writing to an EEPROM (technically any I2C code is redundant in test code 4 but adding it allows for flexibility in how it is used) and will be then sending that to the ground station. Initially a function has to be written that will write the information received by a transmission to an EEPROM. The function will not have to return any data so will be a “void” function and the inputs required will be the address of the EEPROM (“adress”), the address of the information within the EEPROM (“promadress”) and the “data” being written in the form of a byte. From there the inbuilt wire library function “Wire.beginTransmission(address)” opens the communication between the device registered at that pin to begin communication. From there the MSB and LSB (see I2C section) need to be written to the EEPROM addresses before the data that is wanted is written. Once the data is written it is important the communication pin is then closed using “Wire.endTransmission()”. After this a read function must also be put in place, this will obviously return data so must be a “byte” function. The input for this function still needs the address of the EEPROM and the address of where you need to write the data to. This function then starts the same as the write function as it begins transmission to the storage begore writing the MSB and LSB of the data. After this the Arduino will “Wire.requestFrom(adress,1)” which requests the information from the from the specific EEPROM address and states the quantity wanted, before filling the “datafill” with the byte of information contained at that address, this is then returned from the function. Once these are in place the standard operating loop can be entered for the test code. The receiving code is the same as for test 1 except when the information is further analysed to see what function the Arduino should perform. Once a form of data is received the code checks whether the first bit of the transmission is a 0 or 1. The code is set up so that a 0 causes the Arduino to send the rest of the data to the EEPROM whilst printing “writing data” to the serial port. If the received data starts with a 1 the code reads the data from the transceiver (having converted it to an int) then reads the data in the addresses from 0 to the second number in the transmitted signal and sends it via the transceiver. Otherwise, the code should return the same failed message function as previously.

6.11.3 Test code 4 Ground station

Test 4 code has the redundant code for reading and writing to the external storage. Initially the void loop starts by sending the data “131415” to the satellite Arduino. This data should cause the other Arduino to write the data “31415” to the external storage After this the code begins the process for receiving as seen before (buf and lengths of the nrf24) and delaying to give the satellite Arduino time to write the data. After this new data is created of “04” which is then sent to the Arduino with the aim that satellite Arduino will read the data at storage address 0-4. The Arduino then going into receive mode with the hope it will receive a response of “31415” (this may be on separate lines and have “No message receive” in between depending on lag times). The exit statuses are the same as the previous test if the Arduino fails to receive a signal.

6.11.4 Results

As the EEPROM was not ordered in time there was no physical model made this year, so currently there are no results from this test. However, a successful result for this test would have one port outputting that it has received data and sent a reply with the other port saying it received a reply and printing "31415" as the reply received as this would show the satellite Arduino received the correct information, wrote it to the EEPROM and then took information from the EEPROM to send back to the ground station.

7 Electrical Power System

The main tasks of the electrical power system (EPS) are power generation, energy storage and power distribution. It is the role of the EPS to provide continuous power to all the subsystems of the satellite, including during eclipse when no power can be generated through solar panels.This includes the backup power required in certain parts of the orbit where the satellite fails to generate any power from the solar panels, and the protection under any fault conditions. The EPS should be able to manage the power input from solar panels, charging and discharging the onboard battery, and the distribution of energy to each subsystem at required voltage levels. The operation status, monitoring, and processing should be able to communicate to the onboard computer.

It is the orbital dynamics and mission constraints that determine the amount of energy that is available to the CubeSat. This research aims to bring orbital dynamics and onboard mission constraints together and to answer two of the most common design questions based on our specific mission:

- Is the solar array sized appropriately for the payloads?
- Can the EPS fulfil the power requirements of all the subsystems?

With assumptions that the orbit data is in the worst-case sun conditions in LEO, independent of the orbital precession, and that specific output voltages are regulated to avoid damage and fatigue of electrical components.

7.1 Design topology – MPPT versus DET

Most CubeSats use one of two power distribution topologies – Maximum Power Point Tracker(MPPT)[F] versus Direct Energy Transfer (DET). Both have specific benefits and drawbacks, so the context of the satellite within which they will be used must be considered to make an informed decision on which topology to use. [19]

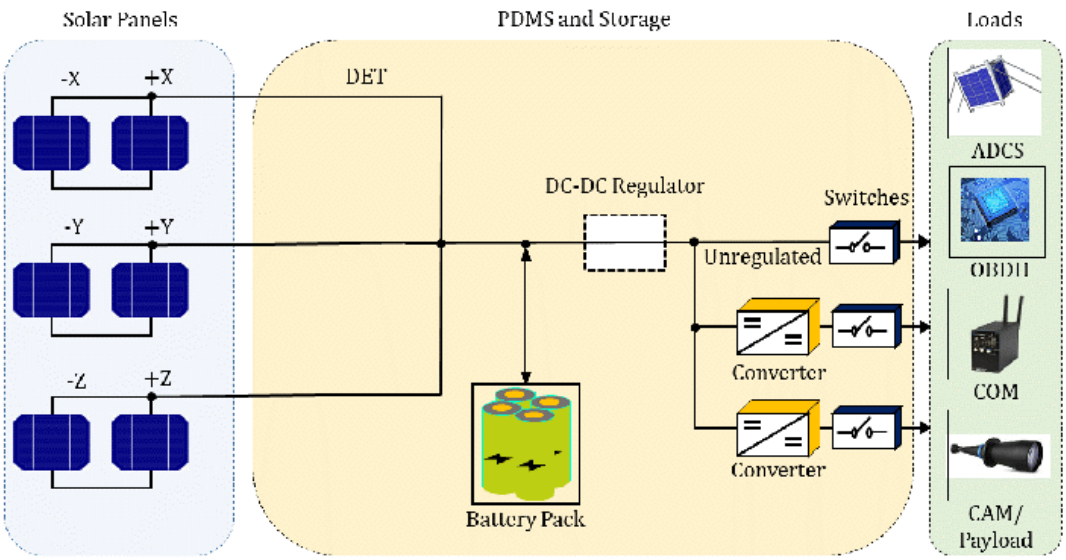
The DET architecture delivers the required power to the loads in a regulated or unregulated form and shunts extra power while operating at a fixed voltage point on the I-V characteristics. In certain circumstances a DET could be higher efficiency because there is no DC-DC conversion between the solar array and the battery. This converting process is easy but effective. Most applications using DET designs have power budgets under 100W[19]. However, because the PV I-V curve of the solar cells is a direct consequence of temperature, irradiance, and deterioration, DET systems will not make the most of the solar energy that the solar arrays are harvesting.

The MPPT-based architecture is designed to vary the voltage output of solar array to always be set at the value where power transfer from array to battery is always at the maximum possible value, regardless of the solar cell temperature and degree of degradation.

A comparison of the peak power tracking EPS architectures is performed in Edpuganti's work[20], which demonstrates that EPS architecture with series connected MPPT converters and regulated dc-bus has the highest orbital efficiency for all the operating modes, lower component count, higher reliability, and greater life of expectancy for battery cells due to lower depth-of-discharge.

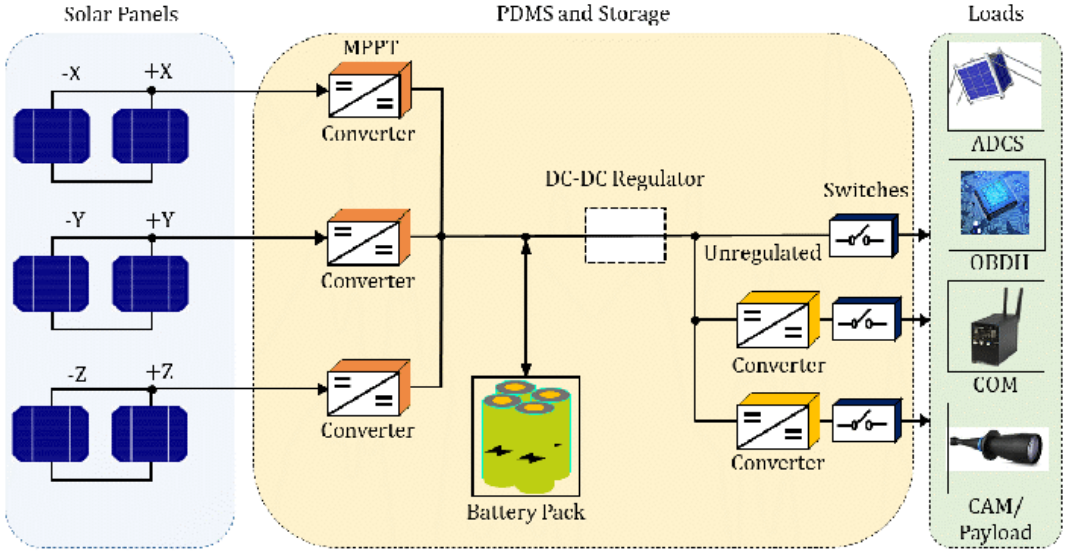
7.2 Power Budget and subsystem requirements

The power budget is defined according to the worst-case scenario which is the simultaneous operation of the loads and not according to the installed load capacity. For this project,



(a)

Figure 7.1: (a)DET solar power conversion



(b)

Figure 7.2: (b) MPPT solar power conversion topologies

because it is still in a preliminary stage, there are several commercial payloads that are not strictly defined, thus the power bus elements would be:

- Power in: ten solar panels dhvtechnology 2U DHV-CS-10
- Power out: EPS iEPS Electrical Power System (A), Antenna Nano-Avionics UHF 2U, Transceiver Endurosat UHF-II, Arduino Leonardo, Payload FDSPP

For the battery of the system, the requirement of power is calculated and shown as: The power budget is calculated and results of different phases are shown as:

Table 7.1: Battery requirement calculations

BATTERY CALCULATIONS	
Time in Eclipse (s)	2150
System Power Required During Eclipse (W)	5.982
Maximum Acceptable Depth of Discharge	40.00%
Battery to Load Power Efficiency	0.9
Required Battery Capacity (W-hr)	9.923842593

Table 7.2: Power budget calculations

POWER BUDGET							
Phase	EQUIPMENT LIST	Endurosat UHF-II	Nano-Avionics UHF 2U Antenna	FDSP P	ARDUIN O	Total	Total + 20%
	Required Voltage (V)	3.3	3.3	5, 7.2	5		
LEOP	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.1	0	0.3	0.5		
	Avg Pwr (W)	0.5	0	0.84	0.145	**1.485**	**1.782**
COM	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.1	0	0.2	0.5		
	Avg Pwr (W)	0.5	0	0.56	0.145	**1.205**	**1.446**
PRO	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.1	0	0.7	0.5		
	Avg Pwr (W)	0.5	0	1.96	0.145	**2.605**	**3.126**
TRANS	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.8	0	0.3	0.5		
	Avg Pwr (W)	4	0	0.84	0.145	**4.985**	**5.982**
REC	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.5	0	0.3	0.5		
	Avg Pwr (W)	2.5	0	0.84	0.145	**3.485**	**4.182**
SAFE	**Pk Pwr (W)**	5	0	2.8	0.29		
	Duty Cycle	0.1	0	0.3	0.5		
	Avg Pwr (W)	0.5	0	0.84	0.145	**1.485**	**1.782**
EOL	**Pk Pwr (W)**	5	0	0	0.29		
	Duty Cycle	0.01	0	0	0.01		
	Avg Pwr (W)	0.05	0	0	0.0029	**0.0529**	**0.06348**

For each subsystem, according to its own system limitations of current and voltage, the electrical system diagram could be represented as:

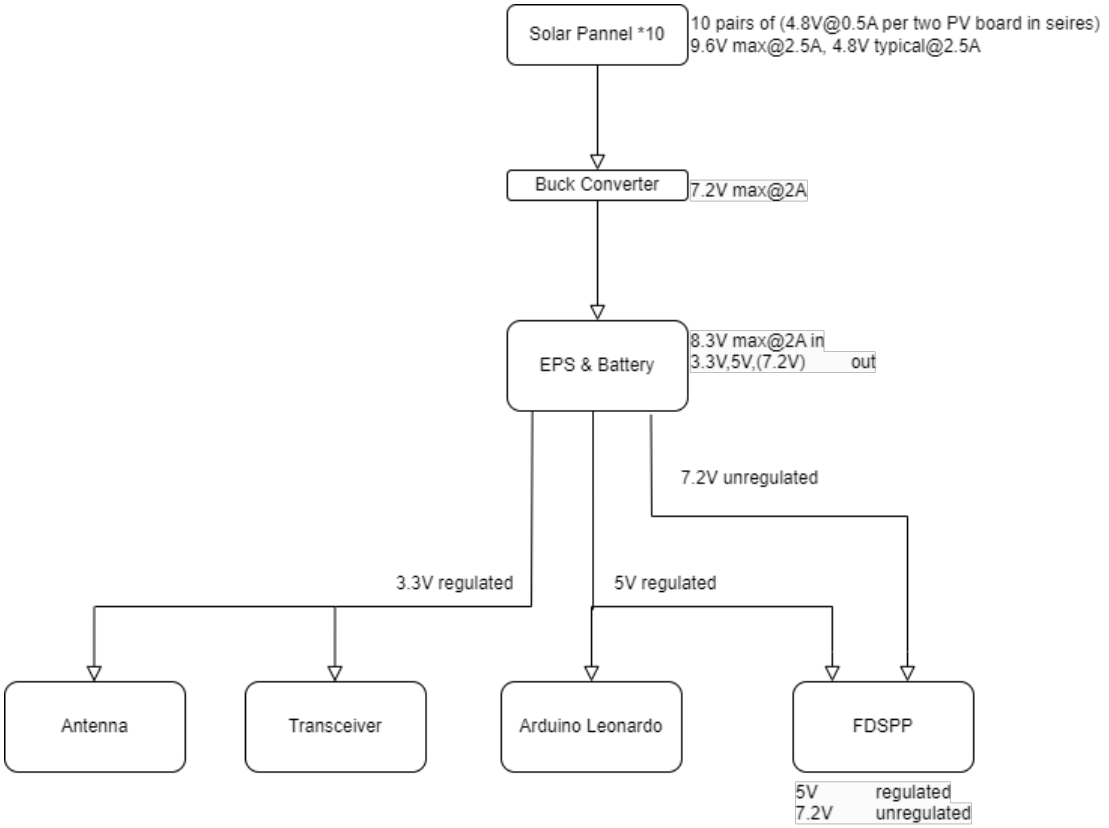


Figure 7.3: CubeSat power electrical system diagram

7.3 Design topology – Solar panel selection

For the purpose of selecting components, a decision matrix is completed. This guarantees all factors are concerned and it is helpful to make final decision. An example process of selecting the optimal solar panel would be shown as:

Table 7.3: General factors that will affect final decision with quantized ranking and significance

Factor	Justification	Significance	Points
Efficiency	Maximum efficiency and power output. This is affecting the quantity and cost.	2	4
Cost	The total cost of the solar panels will be one of the biggest expenditures. This will be different for different manufacturers.	1	5
Operating temperature	All models should be space-rated anyway.	3	3
Weight	Depends on mass budget.	4	2
Others	Additional sensors and components	5	1

Table 7.4: Panel efficiency and power capacity

Manufacturer	**Model**	**Max. Efficiency (%)**	**Power Generation Capacity [mW/cm^2]**	**Power per unit [W]**	**Rating**	
GOMSpace	2U NnPwr P110	30.2	38.10	2.3	3	4
NanoAvionics	2U GaAsSP	28.7	36.85		6	1
EnduroSat	2U SP X/Y	29.5		2.4	2	5
ISIS	2U	30	37.94	2.3	4	3
NPC Spacemind	SM-SP1Z00 2U	29.5		2.4	5	2
dhvtechnology	2U DHV-CS-10	30		2.42	1	6

Table 7.5: Price per panel and price per Watt

Manufacturer	**Model**	**Price per unit**	**Rating**	
GOMSpace	2U NnPwr P110	€2500	3	4
NanoAvionics	2U GaAsSP		4	3
EnduroSat	2U SP X/Y	€2500	3	4
ISIS	2U	€2500	3	4
NPC Spacemind	SM-SP1Z00 2U	€1750	2	5
dhvtechnology	2U DHV-CS-10	€1,450	1	6

Table 7.6: Panel Weight

Manufacturer	**Model**	**Weight[g]**	**Total Weight[g]**	**Rating**	
GOMSpace	2U NnPwr P110	29, 26	29*2+26*8	1	6
NanoAvionics	2U GaAsSP			5	2
EnduroSat	2U SP X/Y	44	44*10	3	4
ISIS	2U	50,100	50*2+100*4	4	3
NPC Spacemind	SM-SP1Z00 2U	35g	35*n	2	5
dhvtechnology	2U DHV-CS-10	50g	50*nd	4	3

Table 7.7: Panel operating temperature

Manufacturer	Model	Min. Temp (°C)	Max. Temp (°C)	Rating	
GOMSpace	2U NnPwr P110	-40	85	5	2
NanoAvionics	2U GaAsSP	-40	85	5	2
EnduroSat	2U SP X/Y	-40	105	4	3
ISIS	2U	-40	125	3	4
NPC Spacemind	SM-SP1Z00 2U	-50	130	2	5

Table 7.8: All other factors and addition components

Manufacturer	**Model**	**Other info**	**Rating**	
GOMSpace	2U NnPwr P110	Coarse sun sensor • Temperature sensor	2	4
NanoAvionics	2U GaAsSP	Integrated cover glass for improved durability and radiation protection Integrated by-pass diode Surface-mounted wire cutters Integrated magnetometers Attitude estimation sensors	1	5
EnduroSat	2U SP X/Y	Integrated sun sensors and gyroscope, optional magnetorquer	2	4
ISIS	2U	Includes Coarse sun sensors and temperature sensors	2	4
NPC Spacemind	SM-SP1Z00 2U	SM-SP are equipped with a standard sensor package including: • Accelerometer, Gyroscope, Magnetometer, Temperature sensor: • Sun sensor: • External Temperature sensor • IMU	1	5
dhvtechnology	2U DHV-CS-10	Magnetometer, sun sensor and temperature sensor	2	4

Table 7.9: The final decision matrix

Manufacturer	**Model**	**Efficiency (4)**	**Cost (5)**	**Operating temperature (3)**	**Weight (2)**	**Additional features (1)**	**Total**
GOMSpace	2U NnPwr P110	4	4	2	6	4	58
NanoAvionics	2U GaAsSP	1	3	2	2	5	34
EnduroSat	2U SP X/Y	5	4	3	4	4	61
ISIS	2U	3	4	4	3	4	54
NPC Spacemind	SM-SP1Z00 2U	2	5	5	5	5	63
dhvtechnology	2U DHV-CS-10	6	6	6	3	4	82

Therefore, for the solar panel, it was chosen to be DHV technology's 2U DHV-CS-10 as it has the highest ranking among all. According to DHV Technology, the output data would be:

$$P_{max} = 2.42W, V_{typical} = 4.8V, I_{typical} = 0.5A$$

7.4 Design topology – Solar energy generation with rotational effects

7.4.1 Introduction

Power generation for a CubeSat is one of the most important factors in satellite mission success, without suitable generation the satellite would eventually fail as the battery package has limited energy. To this end, simulations should be done to check that enough power is generated, whilst in orbit. A MATLAB based model is therefore designed which takes a 1U, 1.5U and 2U CubeSat and finds the lower bound mean power generation over various ratios of rotational velocity. The model assumes that each face contains a solar panel.

7.4.2 Model explanation

A 1U CubeSat can be approximated by six planes, in three parallel pairs. Each pair always has one side facing the sun, and the other away from the sun. Thus, to find the power generation, only three faces need to be inspected.

$$c_1 = (-n, 1, 1)$$

$$c_2 = (n, -1, 1)$$

$$c_3 = (n, 1, -1)$$

$$c_4 = (n, 1, 1)$$

(1) where n is the CubeSat size.

The three faces can be defined by four points, at positions shown in Eq 1. The four points describes an arbitrary initial position, which was chosen for ease of changing CubeSat size.

$$\overrightarrow{n_{sun}} = [1, 0, 0]$$

(2) To calculate the power generation for a face, the angle between that face and the sun needed to be found. The sun was approximated by a plane, with normal vector shown in Eq 2. The normal vector of each of the planes was then found and the angle between them, and then sun normal, was found. This angle describes the angle of incidence of sun rays on the solar panel. To relate angle of incidence and power generation an equation was fitted to the data in [2]. Multiple types of fitting were tried, though a polynomial fit was found to be the best. As solar panel values are likely to vary, the order of the polynomial fit is calculated dynamically, with a threshold of 0.001 for the difference between the predicted points and actual points.

$$a_1 = -8.01 * 10^{-11}$$

$$a_2 = 1.74 * 10^{-8}$$

$$a_3 = -1.44 * 10^{-6}$$

$$a_4 = 5.38 * 10^{-5}$$

$$a_5 = -9.73 * 10^{-4}$$

$$a_6 = 4.68 * 10^{-3}$$

$$a_7 = 2.29$$

(3)

$$y = a_1x^6 + a_2x^5 + ... + a_6x + a_7$$

(4)

For a maximum power of 2.3W the coefficients in Eq 3 were found. This polynomial function, Eq 4, was then used to find the power generated on a single solar panel at a given angle.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$R_x(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & \\ sin(\theta) & cos(\theta) & 00 & 0 & 1 \end{bmatrix}$$

(5)

To simulate satellite tumbling of the CubeSat through space the rotation matrices were used, as shown in Eq 5.

$$c = \begin{bmatrix} -n & n & n & n \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}$$

(6)

$$c_{rot} = R_{x,y,z}(\theta) \times c$$

Defining the points of the cube as the columns of a matrix, Eq 6, allows for remultiplication by the rotation matrix, Eq 7, creating a new set of points, rotated around an axis by

$$\theta$$

radians.

In space the CubeSat will have a constant angular velocity, which rotates the cube around the x, y and z axes. In this case a full rotation will generate the same power as the previous, thus is periodic.

7.4.3 Code explanation

- CubeRotation.m – Creates a graphical plot of the cube rotation, over 360 degrees, with the power generation values, with the input of a given configuration, rotation ratio and maximum power generated by solar panel.

- CubeRotationAll.m – Iterates over an array containing configurations of 1, 1.5 and 2U sizes and calculates the average power generation over a large range of ratios. Uses more efficient processes than CubeRotation.m to produce the same result. Takes input of maximum power generated by solar panel. Saves the average power arrays in the current directory.
- PowerEq.m – Used in the above files, calculates the power at a given angle, assuming proportional relation to [2].
- Plotting.m – Plots the graphs like that shown in figure 3. Takes input of the three saved files from CubeRotationAll.m.

7.4.4 Power Comparison

For different ratios of x, y and z rotation different amount of power are generated, due to more faces being in the sun at one point. To gain good insight into how different rotations affect power generation, an array from 0 to 1, with a step size of 0.01, was used. Every unique combination of this array was then used, to get a systematic sample of the effect of ratios on power generation and to see which is the most likely level of generation. As the satellite is likely to fully rotate multiple times per orbit, the mean power, over 360 points around one rotation, is taken.

7.4.5 Simulation results

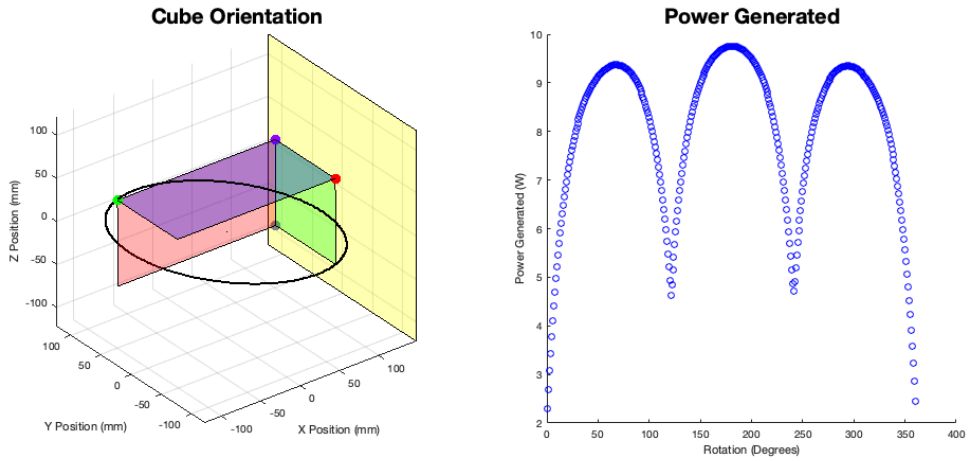


Figure 7.4: (a)2U Cube Rotation Path, (b) Power Generation at Degree of Rotation

The plots above shows the three faces of a 2U CubeSat, in red, green and blue, and a yellow plane representing the sun. The four points are rotated around the x, y and z axes, such that the motion of the green point follows the black rotation path line. This rotation creates the graph on the right, with differing amounts of power being generated at different angles. For this test a ratio of 1:1:1 degrees of rotation around the x, y and z axes were used.

Similar to the 2U rotation the 1U rotates around a path and produces a periodic power graph. As the area of each side is the same it will produce a different set of power generation graphs. In the case in Fig 2, a 1:1:0 ratio of x, y and z rotations were used to produce these results. It is evident that many different ratios will need to be investigated to get an adequate view of power generation[H].

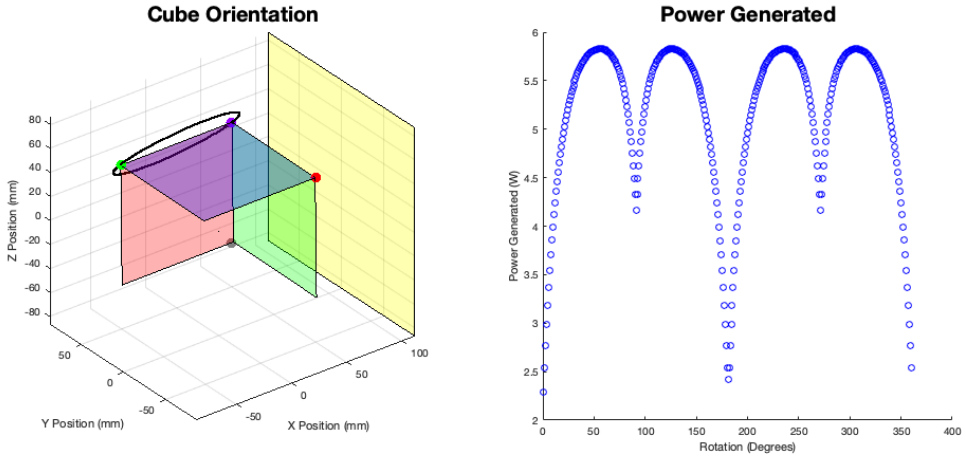


Figure 7.5: (a)1U Cube Rotation Path, (b) Power Generation at Degree of Rotation

7.4.6 Simulation Analysis

The above figure shows the distribution of power generation for each type of CubeSat, where the black cross on each shows the minimum expected power. The minimum expected power is calculated as one standard deviation from the mean, which, in this case, coincided with the drop-off point of generation. Power values lower than this are highly improbable, thus will not be expected. The failure probability is the sum of the probabilities below the threshold of 4.5W, which was the minimum supply which the satellite can take. Comparing each of the plots the 1U CubeSat just barely passes this threshold, with 4.67W. Considering the closeness of this, if an error margin of 20The 1.5U and 2U configurations both work adequately well and produce more than enough power to supply the satellite. Though consideration should be made as to whether this power requirement would increase as development goes on.

7.4.7 Conclusion

The current power requirement lies at 4.5W of generation, with this in mind a 1U CubeSat is unfeasible, though 1.5U and 2U are. Considering an increased power draw the suggestion of a 2U CubeSat would be made over the 1.5U[I].

7.5 Design topology – Solar array construction

Due to the natural geometry of 2U CubeSat, it would be assumed that two solar panels facing opposite directions would be operating in 180° out of phase, which would therefore half the total peak power. Wiring in parallel allows the CubeSat to have more solar panels that produce energy without exceeding the operating voltage limits of the EPS inverter. Whereas it also has amperage limitations, which this could be achieved by wiring panels in parallel. Due to such topology, two wiring strategies are designed and shown as following.

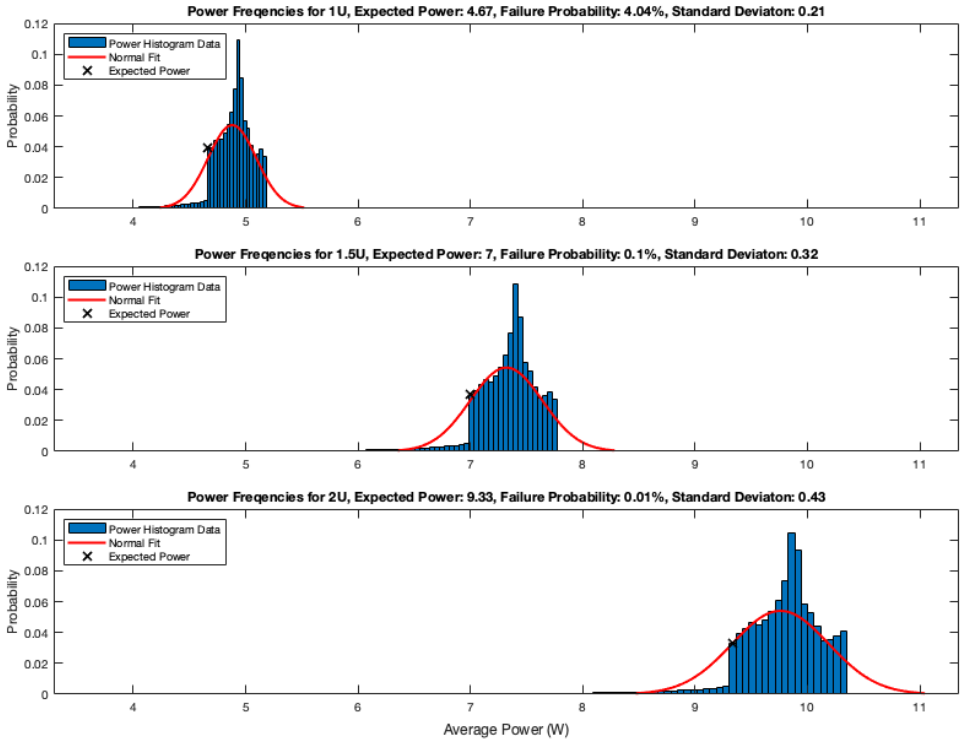


Figure 7.6: Power frequencies for 1U, 1.5U and 2U

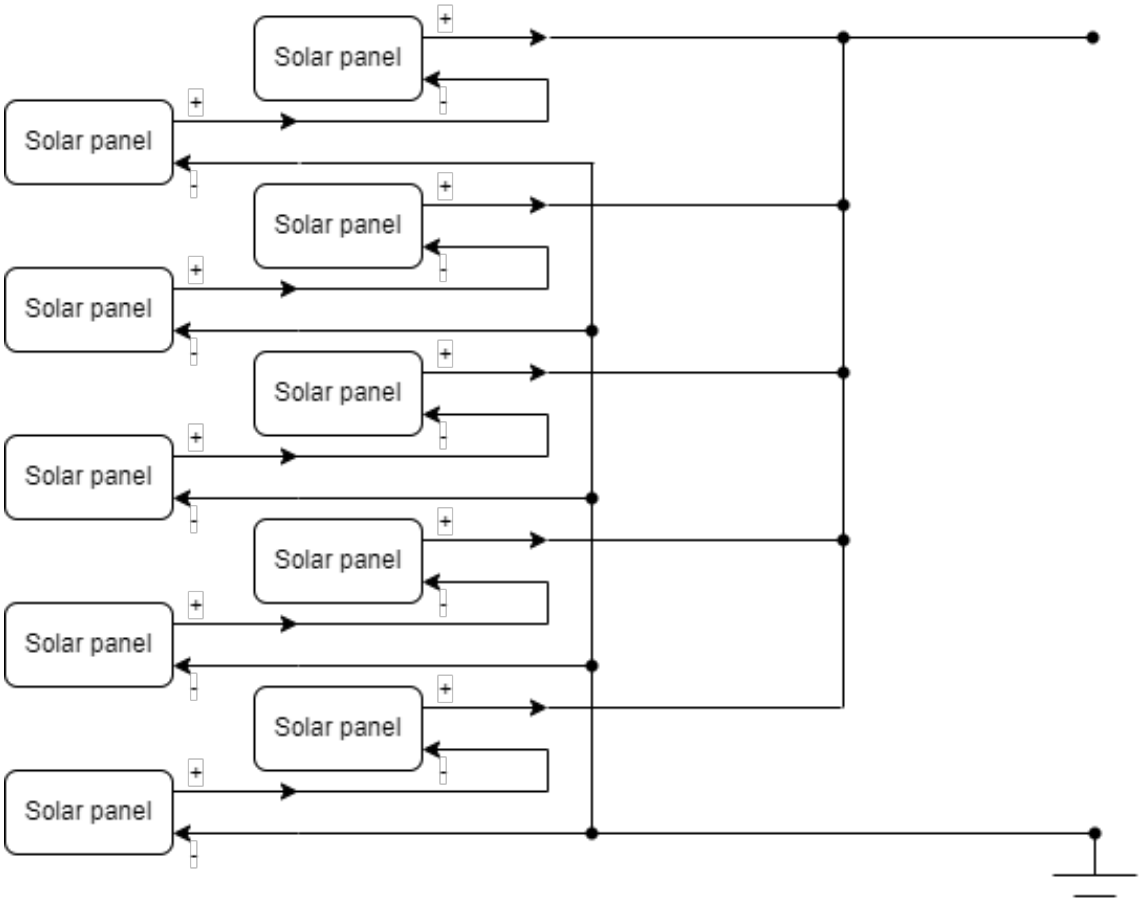


Figure 7.7: (a) connect two opposite panels in series and five pairs in parallel, 2 by 5 solar array

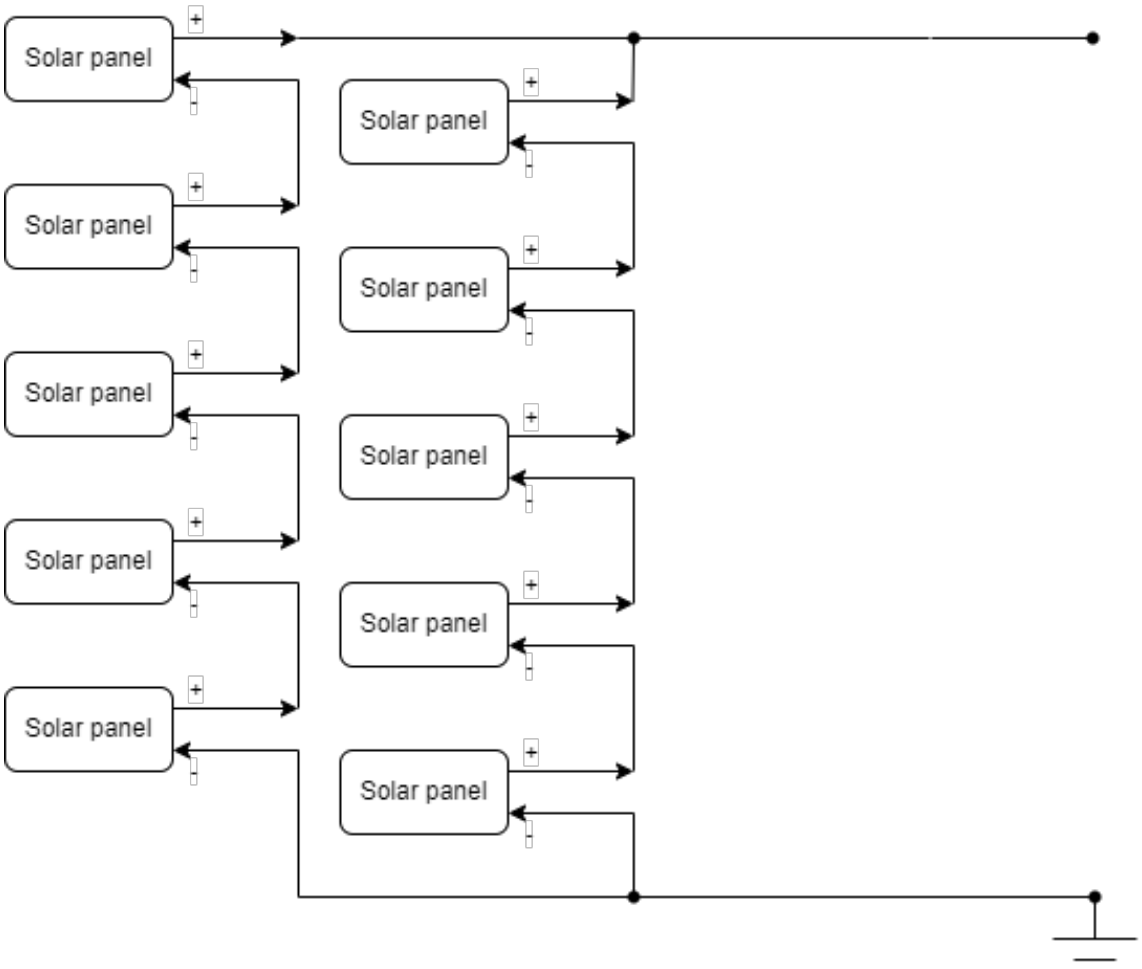


Figure 7.8: (b) connect five panels in series and two groups in parallel, 5 by 2 solar array

Due to strategy (a), the theoretical maximum output of the entire solar array system would be

$$P_(theory, max) = 2.42W10, V_theory = 4.8V, I_theory = 0.5A5$$

whereas for (b), those of the solar array system would be

$$P_(theory, max) = 2.42W10, V_theory = 4.8V5, I_theory = 0.5A$$

Although in practice these calculations are vague, and has insufficient assumptions, this value is however useful in estimating the peak power value, as in reality the solar panel opposite to the sunlight would still generate power from the Albedo effect of earth reflection.

For other unstated connecting strategies, considered that one solar panel breakdown, in (a) there would be no side effect on generating power from other solar panels, whereas in (b) a circuit line is open and half of the solar panels will stop working. Thus, the strategy of connecting multiple panels in series has a significantly high risk of failure.

According to ISIS, the EPS management system chosen has a maximum input voltage for battery to be 8.3 V,2A. Because the voltage generated in strategy (b) would be considered significantly higher than that of (a), and during individual failure, the risk of open circuit half of the solar panels would be considered overwhelmingly high, thus strategy (a) would be used for wiring the solar panels. A buck converter would be used for interface conversion from solar arrays to EPS module to lower the maximum current from 2.5A to 2.0A to fit the current limitations.

7.6 Battery OTC analysis

Because of the natural complexity of designing process of a battery control system, in this mission stage it would be ideal to select a fully functionalized and well-tuned battery system that is already used in other CubeSat missions rather than designing the system by the team, because there are still power characteristics that needed to be further adjusted, and the power electronics need to be adjusted to suit different payloads. This decision would mainly affect on the MPPT control, DC-DC regulator, converter, and digital switches in Figure 1 and 2 shown above. Although the actual cost of purchasing versus manufacturing would be significantly higher, after comparing all opportunity cost of developing and verification period, it would be reasonable to choose commercial digital controlled power systems rather than analogue control systems.

Therefore, for the interfaces between solar array system and battery, due to the evidence that MPPT-based EPS systems are significantly more efficient in terms of power generating, the Power Conditioning and Distribution Unit (PCDU) system for CubeSat missions with MPPTs are gathered and analysed in the table shown below.

Same process of decision matrix was performed. The battery capacity per mass per volume, which would also represent the overall efficiency, in addition that the mission requirements that 3.3V, 5V and 7.3V are needed for the CubeSat to operate are taken into consideration, and it is concluded that the iEPS Electrical Power System (A) manufactured by ISIS Space Agency are chosen for this CubeSat mission [21].

7.7 Conclusion

In this EPS design, it is evident that the solar panels would provide more than sufficient energy for the CubeSat to operate under normal condition. Though it is not feasible to perform any testing and optimization, the power bus is able to obtain different voltage

Table 7.10: PCDU battery pack analysis and trade-offs

BATTERY PACK AND PCDU				
NAME	**iEPS Electrical Power System (A)**	**NanoPower P31u**	**EPS I**	**Optimus**
BRAND	Isispace	Gomspace	Endurosat	AAC Clyde Space
BATTERY CAPACITY(Whr)	23.04	20	10.2/20.4	30
Charging Voltage	8.27V typical	5V Charging interface		8.4V max
BATTERY VOLTAGE	7.2	8		8.26-6.2
OUTPUT CHANNELS	3.3V, 5V and battery raw	3.3V and 5V	3.3V, 5V, and battery raw	3.3V and 5V
INPUT CHANNELS	3(3.5V to 7.5V)	3	3(up to 5.5V and 1.8A)	
DIMENSIONS (XYZ)	96x92x26.5	89.3x92.9x25.6		95.89x90.17x21.55
MASS (g)	184	200	193	268

and current limitations for different subsystems, which could be considered as a successful preliminary engineering model for team's future use.

7.8 Further work and self-critique

For the CubeSat EPS system, the most crucial component would be the battery. In the commercial applications, LiPo batteries and Li ion batteries are so far the most used. A battery analysis is presented below to show a selection of all suitable batteries that could be purchased if the power control and regulation system could be developed.

Table 7.11: Component List of commercial CubeSat batteries with dimension less than 95*95*50 mm and weight lighter than 750 g on the global market[ST]

Component ID	**Type**	**length(mm**	**width(mm)**	**height**	**capacity**	**voltage**	**watt*hour**	**charge rate**	**max discharge rate**	**mass (g)**	**Wh/g**	**Wh/g per height**
25 Whr High Energy Density LiPo Battery Array	Li-poly	95	89	7	6Ah	3.7 V - 4.2V@full	22.2	4.2 V-4.10 V; 0.2 A - 5.0 A		125	0.1776	0.025371
P1U Vasik Power Supply	Li-ion	90	96	12.2	3.8 Ah	3.7 V typical	14.6		3.3 V, 5 V, 12 V constant	80	0.1825	0.014959
14V Modular SmallSat Battery	Li-ion	94.5	84.1	23.1	45 Wh	13.2 - 16.8 V	45	1.6 A	6.5 A	375	0.12	0.005195
OPTIMUS-30	Li-poly	95.89	90.17	21.55	30 Wh	6.2 V typical	30	8.4 V, 1.95 A	6.2 V, 1.95 A full	268	0.11194	0.005194
LP 33330 6Ah Space Cell	Li-ion	71	113.1	25.3	6Ah	3.0 V - 4.1 V	23.4	6A (0.5C) to 4.1V 4.10V to 0.24A (C/50)	24 A const. 48 A pulse	222	0.105405	0.004166
LP 32975 12Ah Space Cell	Li-ion	91.36	49.9	22.2	12 Ah	3.0 V - 4.1 V	30	6A (0.5C) to 4.1V 4.10V to 0.24A (C/50)	96 A const. 180 A pulse	465	0.064516	0.002906
OPTIMUS-80	Li-poly	95.89	90.17	56.94	80 Wh	6.2 V typical	80	8.4 V, 5.2 A	6.2 V, 5.2 A full	670	0.119403	0.002097

To control the battery as already discussed in the sessions above, the design of such power electrical components including MPPTs, power converters, DC-DC converters and digital switches would be designed and manufactured by the team to lower the cost of purchasing, as the main process would be PCB printing, which would then reduce the cost significantly.

8 Communications

The communication system for this project is primarily discussed in the technical report for this project. Though additional justifications as to specific choices, which were not able to be gone into detail are discussed below.

8.1 Frequency Band Considerations[O]

Three bandwidths are available for amateur radio operator, VHF, UHF and S-Band. Parameters of each have been considered. In terms of data rates the UHF and S-Bands bolster typically much higher data rates that VHF, [22] [23] [24], thus VHF was not considered viable to send the large data volume required for this mission.

As S-Band is tailored towards high data rates most models only offer modulation types to support this, though in turn this means that it suffers more heavily from noise, leading to greater bit error rates. With higher bit error rates more power is required to close the link margin, thus rendering the S-Band transceivers unsuitable for our application.

Additionally, many other CubeSat projects utilise the UHF band for communications meaning many technology ready COTS components are available.

This topic was also discussed during the ESA concurrent engineering course, where it was strongly recommended that a UHF antenna be used for omnidirectional applications.

8.2 Component Tradeoffs[P]

Utilising the ideas proposed in the technical paper for this project a set of trade-off analysis was conducted on components for the communications system.

Parameter	Weight	TRX-U	UHF-II	SatCOM UHF
Output Power	0.2	5	3	4
Input Power	0.4	2	5	3
Data Rate	0.3	5	5	3
Mass	0.1	3	3	5
Total		3.6	**4.4**	3.4

Table 8.1: Transceiver Tradeoffs

Parameter	Weight	Alen Space	Dhruva Space	ISIS Space
Peak Gain	1	1	16.3	15.5
Total		1	**16.3**	15.5

Table 8.2: Ground Station Tradeoffs

Parameter	Weight	CubeSat Antenna	Antenna 2U	UHF 1U Antenna	UHF 2U Antenna
Gain (dB)	0.2	3	3	1	4
Half-Power Bandwidth (dB)	0.5	4	4	5	3
Link Margin (dB)	0.2	4	4	4	4
Mass	0.1	2	1	5	4
Total		3.6	3.5	**4**	3.5

Table 8.3: Antenna Tradeoffs

The full specifications for 8.1, 8.2 and 8.3 are listed in J.

For the transceiver, in Table 8.1, the heaviest weighting was towards the magnitude of the input power. This is due to the nature of this project, minimising the power drawn by the communication system is very important. This led to the EnduroSat UHF-II to be chosen as the transceiver to be used.

The ground station was simply chosen as to the one with the largest gain, though one of the choices had to be removed as it did not operate within the expected amateur bands. This caused the Dhruva Space UHF antenna to be the most suitable, as shown in Table 8.2

Utilising the parameters from the transceiver and the ground station a link margin was calculated for each antenna option. The highest weight, in 8.3 was attributed to the Half-Power Bandwidth as this correlated with the omnidirectionality of the antenna, thus a more omnidirectional antenna is superior. This led to the Nano Avionics UHF 1U Antenna to be chosen as the optimal decision.

9 Mechanical

9.1 Chassis Overview

The chassis is an essential part of any space vehicle, including a CubeSat. Its primary function is to provide the CubeSat with mechanical integrity to withstand loads during launch and orbit operations [25]. The chassis structure also acts as a platform to secure the internal components of the CubeSat. An ideal chassis can be structurally strong whilst also being as light as possible. Therefore, design choices have to be made to utilise minimum material. Autodesk Fusion 360 was the CAD software used for the chassis design process. The chassis structure of WUSAT-4 was designed to comply with the CubeSat Design Specification REV 14.1, FYS Design Specification, FDSPP payload requirements and the launcher specifications. All of these specifications were compared to analyse where the requirements were the same and where there were clashes in which the worst-case scenario was picked. For the launcher specification, there are 2 launchers which the CubeSat can use, the P-POD [6] and the NanoRacks [8]. Both requirements are similar in most respects, but there are some instances in which they differ. Such as for the P-POD the minimum extension from the end of the rail and the +/-Z faces is 0.5 mm [6], while the minimum extension for the NanoRacks is 2 mm [8]. Furthermore, for a 2U CubeSat, the maximum mass allowable for the P-POD is 4 kg [6], while the maximum mass requirement for the NanoRacks is 3.6 kg [8]. In all of these such cases, the worst-case scenario is used, so the CubeSat will be compliant with both launchers. The dimensions for the CubeSat do not change for each launcher, and so for a 2U CubeSat the width shall be 100.00 mm and the height from the end of each face of the rail shall be 227.00 mm. The design of the rail is also an important part of the CubeSat and so the dimensions for it have to be exact, as it will be the only part of the CubeSat touching the launcher and other CubeSats, and any inaccuracies can lead to either the CubeSat not fitting into the launcher so additional vibrational loads onto the spacecraft created from the gap between the rail and the launcher. The specification lay out precise requirements for the design of the rail [6]:

- 2.2.5 - Rails shall have a minimum width of 8.5mm measured from the edge of the rail to the first protrusion on each face
- 2.2.7 - The edges of the rails should be rounded to a radius of at least 1 mm.
- 2.2.8 - The ends of the rails on the +/- Z face shall have a minimum surface area of 6.5 mm x 6.5 mm contact area with neighbouring CubeSat rails.

Whilst designing the CubeSat as well as ensuring everything complies with all the requirements it is also essential to make sure that the components designed are easy and affordable to manufacture. This involves making sure that the components are kept as simple as possible, as any added difficulties would lead to complications during manufacturing. To ensure that the design complied with this, meetings with the School of Engineering workshop technicians were conducted, where the design was shown and the ease of manufacturing was discussed. To allow for components within the CubeSat to be accessible while everything has been assembled, the inclusion of an access panel is necessary. This is a removable panel which does not affect the rest of the structure and can be attached and detached with ease. Screw holes are also needed to be placed along the chassis so that the structure can be assembled, whilst also ensuring that components can be attached to the spacecraft. If a stack configuration is used for the CubeSat then attachment points are needed on the top and bottom panel for the rod through which

each component within the CubeSat will be held in place. Furthermore, screw holes are also needed for the placement of the solar panels on each face. Screw holes were made to be the same size as the solar panel holes so that the chassis panel and solar panel could be attached with the same screw.

9.2 Chassis Research

Before starting to design the new chassis, research was conducted on other CubeSat chassis designs from other universities and companies to further understand the CubeSat structure and help generate ideas on how the WUSAT-4 chassis could be designed. The advantages and disadvantages of different options were discussed, such as their ease of manufacturing and ease of assembly. Before work was started on Autodesk Fusion 360, hand-drawn sketches were made to portray different ideas that were generated. This allowed for a discussion in which the best ideas were put forward and features from different ideas were merged. This then allowed for the first iteration of the design to be drawn on CAD.

9.3 Chassis Iterations

9.3.1 Iteration 1

For the first iteration, it was thought to design one piece which could be manufactured 4 times to assemble the CubeSat, as shown in 9.1 and figure 9.2 This was due to the belief it would simplify the design and manufacturing process, while also saving on material. To achieve this, one component was designed where the rail and the panel are joined. The panels are joined together by screwing the side of the panel into the rail. The top and bottom panels are attached to the excess material left on the top. The thickness of the panel was made to be 2 mm, as it was thought to be a good starting point which could be reduced in the future.

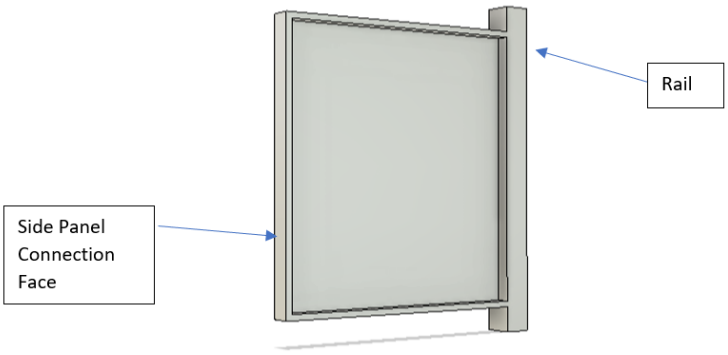


Figure 9.1: Iteration 1 Side Panel Inside

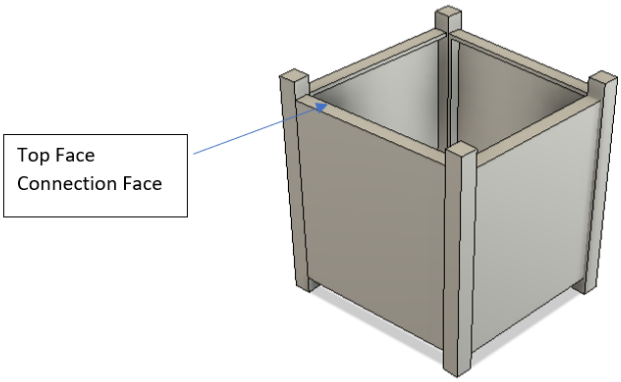


Figure 9.2: Iteration 1 Full Assembly

9.3.2 Iteration 2

For the second iteration, material was removed from the area where the panels are joined together, this is to reduce unnecessary material and reduce weight. The material was also removed from the inner section of the rail where it was not needed, as shown in 9.3 The outer section of the rail was kept to comply with the requirements, where the outer dimension should be 100 mm by 100 mm. Holes were also added to the design to show where the panels would be attached. However, after reviewing this iteration it was realised that it did not comply with our need for an access panel. Once a panel is removed the CubeSat loses all its balance and so, a new design is required which solves this issue.

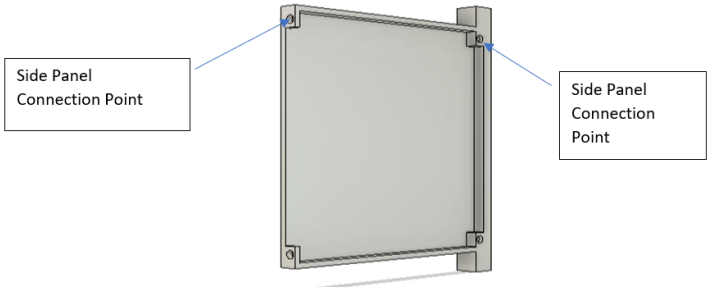


Figure 9.3: Iteration 2 Side Panel

9.3.3 Iteration 3

To account for the access panel, the side panel was changed so that the rails were designed on both sides of the panel, which were incorporated into the connection points, as shown in 9.4 This allows for the access panel to be removed and all 4 legs allow for the CubeSat to be stable. The access panel is a flat sheet of material which can easily be attached and removed, as shown in 9.5 It was also deemed to be easily manufacturable and so was an ideal design for an access panel. Some material was also removed at the top of the main side panel, where the top and bottom panel is attached. However, it was still not deemed to be fully optimised, with the potential for further material to be removed.

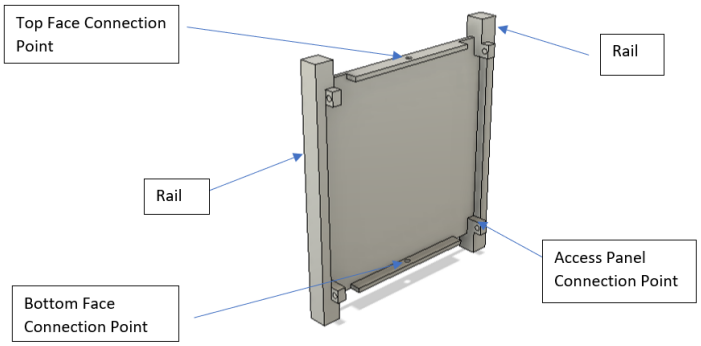


Figure 9.4: Iteration 3 Side Panel

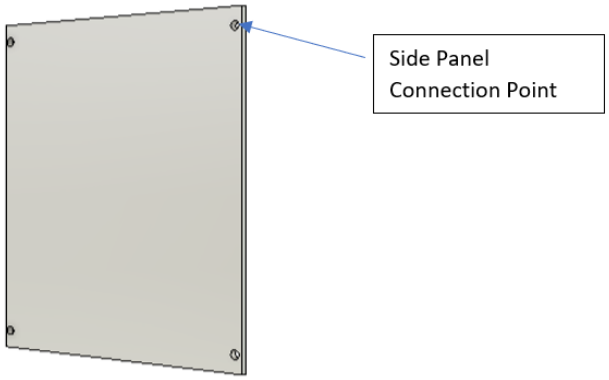


Figure 9.5: Iteration 3 Access Panel

9.3.4 Iteration 4

To reduce unnecessary material, the screw holes for the top and bottom panels were moved to the side of the panel rather than the middle as shown in 9.6 and 9.7 There was uncertainty over whether the manufacturing of such a panel would be feasibility. However, after meeting with the workshop technicians to discuss this issue, it was decided that there were no problems with manufacturing such a design and that from their experience from previous years of the WUSAT project, they believed that this design was a suitable solution for the chassis design.

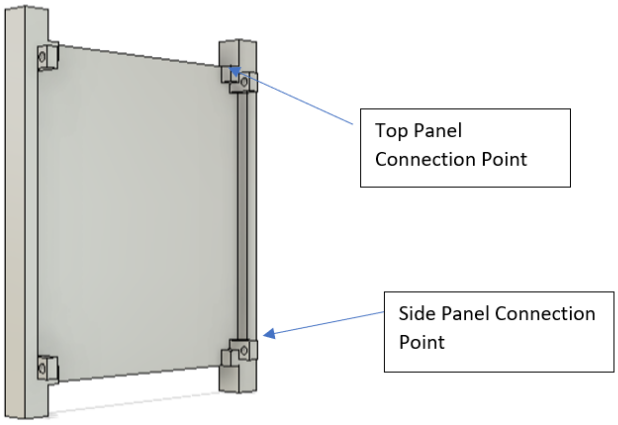


Figure 9.6: Iteration 4 Side Panel

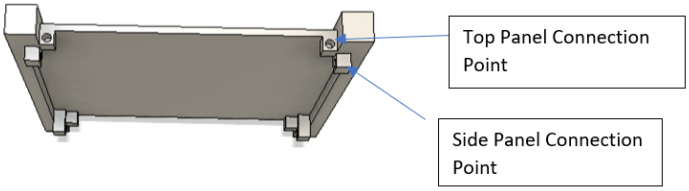


Figure 9.7: Iteration 4 Side Panel Top View

9.3.5 Iteration 5

It was decided by the whole team that to incorporate everything into the CubeSat the chassis would have to be made into a 2U CubeSat, as shown in 9.8 and 9.10 This changes the height of the CubeSat from 113.5 mm to 227 mm. From the CubeSat specification requirement number 2.2.3, " No components shall protrude farther than 6.5 mm normal to the surface from the plane of the rail."[6], it was also found that an indent is not needed for the solar panels. Therefore, the connection points were extruded out by 2 mm so that the panels were flush on the outside, as shown in 9.9 Holes were made in the top and bottom panels to insert the configuration rods, as shown in 9.11. To comply with requirement 2.2.8, " The ends of the rails on the +/- Z face shall have a minimum surface area of 6.5 mm x 6.5 mm contact area with neighbouring CubeSat rails", the material was removed from the top and bottom faces of the rail, as can be seen in 9.12.

The chassis consists of the following components:

- Top panel
- Bottom Panel
- 2 Side panels with in-built rails
- Front and rear access panels

Each component will be manufactured separately and then assembled.

After a material for which the chassis could be manufactured was chosen, this iteration was then analysed using finite element methods where the stress and natural frequency could be analysed, and further improvements could be made to the design.

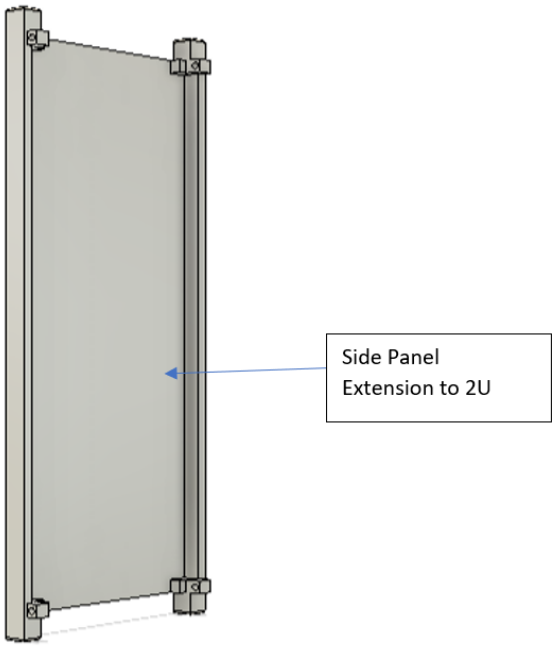


Figure 9.8: Iteration 5 Side Panel Inside

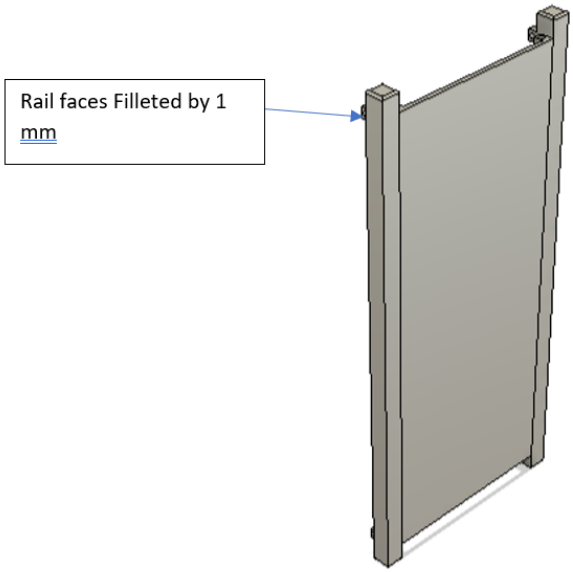


Figure 9.9: Iteration 5 Side Panel Outside

Figure 9.10: Iteration 5 Access Panel

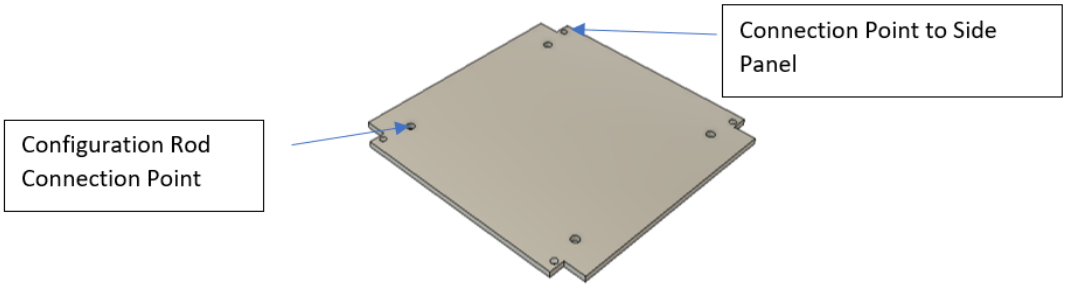


Figure 9.11: Iteration 5 Top and Bottom Panel

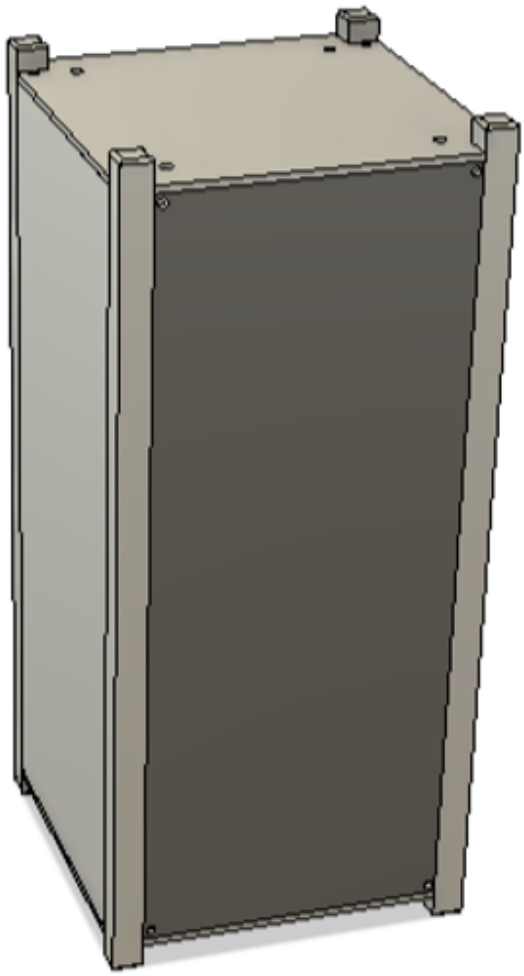


Figure 9.12: Iteration 5 Full Assembly

9.4 Material Research [E]

The material chosen for the CubeSat can greatly affect the behaviour of the structure from its manufacture to end-of-life. Therefore, it is a vital part of the design that has to be chosen carefully. The choice of materials has to be an aluminium alloy as specified in the ESA Fly Your Satellite specification: 4.1.21 - Aluminium 7075, 6061, 5005 or 5052 shall be used for both the main CubeSat structure and the rails Aluminium alloys are an ideal material for the structure due to having low density, high strength, and good manufacturability [26]. Therefore, these materials were compared to identify the best choice for WUSAT-4. When considering the material of the chassis it is important to consider some key factors [25]. These can be listed as:

- specific strength and stiffness
- fracture toughness
- fatigue resistance

- stress corrosion
- resistance
- thermal resistance
- sublimation
- erosion
- manufacturability

During a CubeSat's life, the structure experiences many different stresses. Thus, to be able to cope with this, the strength and stiffness of the structure have to be to a suitable level [25]. The material is a major factor in this. To compare the strength and stiffness of each material, the modulus of elasticity of each material is compared to each other. The yield strength of each material is also compared, the higher the yield strength the better the material is at allowing for higher stress to be applied without changing the shape or deforming the structure. Secondly, the mass of the structure is of vital importance, each gram saved can lead to keeping the satellite light, allowing for mass to be spent in other subsystems. Therefore, material density is another key factor. The chassis must be able to withstand the harsh climate of space, from high temperatures when it is in sight of the sun, to the other end of the spectrum when the satellite is behind the Earth. To aid in this a material must not only be able to withstand both these extremes but also protect the equipment inside this. To judge this the thermal conductivity of each material was considered. The material chosen for the chassis must be able to be easy to machine and manufacture to allow for the precise shape of the chassis to be formed. Different manufacturing techniques must be able to be applied to the material to allow for a large range of options when the chassis has to be manufactured. To judge this each material is researched to determine its suitability to different manufacturing options. These different properties were put into a table as shown to allow them to be compared.

Table 9.1: Material Table

Material	Density (g/cm3)	Youngs Modulus (GPa)	Yield Strength (MPa)	Thermal Conductivity (W/m.K)	Manufacturability
Al-5052	2.68	70	193	138	Good welding characteristics. Difficult machinability for high-precision applications.
Al-5005	2.70	69.5	197	201	Good welding characteristics. Difficult to machine for high precision. Mostly suitable for anodising.
Al-7075	2.81	72	503	166	High yield strength means it is not suitable for bending, leading to risk of additional stresses during manufacturing. Not ideal for high-precision cutting.
Al-6061	2.70	70	276	134-160	Suitable yield strength which can tolerate levels of bending during manufacture. Allows for high-precision cutting. Good welding characteristics.

All the materials that were compared have similar densities and so the mass of the structure will be similar for all the alloys. From the table, Al-6061 and Al-7075 can be highlighted due to their superior yield strength. However, the high yield stress of Al-7075 makes it harder to bend and so may lead to stresses being generated during manufacturing. This is where Al-6061 is preferable due to the lower yield stress, making it ideal for the manufacturing process, while still being high enough for the CubeSat structure throughout its life. Al-6061 is also a suitable material to weld. The wide variety of manufacturing processes that Al-6061 can deal with, whilst also having a suitable yield strength and thermal conductivity is why it was chosen as the ideal material for the CubeSat.

9.5 Testing and Simulation[M]

Using Abaqus, Finite Element Analysis (FEA) was carried out to model the loads experienced by the CubeSat during the launch process and its operations. FEA analysis was carried out for two main reasons. Firstly, it was to simulate the launch and operational environment of the satellite and ensuring the designed CubeSat structure can withstand the induced loads. Secondly, the FEA results can be used to further optimise the chassis structure design as it provided information about the stress distribution of the primary chassis structure under load. This allows high and low stress areas to be identified, which informed the process of lightweighting and design optimisation. In this section, background theory of loads experienced by a spacecraft will be considered. FEA results, design iterations, and the optimisation process will also be discussed in further detail.

9.5.1 Theory and Background Information

The spacecraft structure experiences different types of loads during its operations. Therefore, it is important to take into account all of these loads during the design process to ensure the CubeSat chassis is able to withstand the launch environment. In general, loads experienced by the spacecraft can be separated into quasi-static loads and dynamic loads. Quasi-static loads represent the slow temporal evolution of loads, it can be defined as the combination of steady-state acceleration and low-frequency vibration[27]. In contrast, dynamic loads can be defined as the temporal evolutions of the loads for different frequency bands[28]. The spacecraft structure would experience the highest dynamic loads during its launch when it serves as a payload of the launch vehicle[29]. In the case of a CubeSat, it is designed to be carried by a CubeSat Dispenser, the Poly Picosatellite Orbital Deployer (P-POD) shown in 9.13 below, which serves as the interface between the CubeSat and the Launch Vehicle [6].

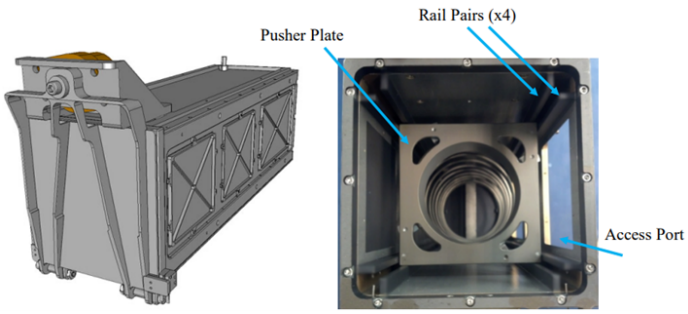


Figure 9.13: Poly Picosatellite Orbital Deployer (P-POD) and cross section utilizing the rail system

During launch, the launch vehicle generates a large amount of acceleration and dynamic loads, all of which are experienced by the CubeSat[30]. The mechanical dynamic loads can be divided into the following categories:

- Acoustic loads: Acoustic loads are caused by the noise generated from the launch vehicle engines, such as exhaust noise. It is also caused by aerodynamic forces and turbulence, for example, when flow separation occurs along launch vehicle[29].
- Random vibrations: Random vibrations are high frequency vibrations caused by the launch vehicle's motion during lift-off and aerodynamic forces. Acoustic loads also contribute to the generation of random vibrations[30].
- Shock loads: These are caused by different events during the launch phase of the spacecraft, including the separation of stages and when the payload is deployed from the P-POD. All these events induce loads onto the CubeSat structure for a very short duration[29].
- Sinusoidal loads: Sinusoidal vibrations are low frequency vibrations mainly caused by the combustion and running of the launch vehicle engine, and the build-up of thrust during lift-off[29].

The "Fly Your Satellite" (FYS) Design Specification outlines a set of environmental requirements that the designed satellite would be required to comply with, this includes testing requirements for various dynamic and quasi-static loads mentioned above. However, some test requirements such as tests for random vibrations are usually conducted as physical testing, which is not applicable at this stage of the chassis design. Therefore, at this stage of the design process, focus will be placed on the following environmental requirements detailed in the FYS Design Specification[**?**].

- 4.10.1. – The first natural frequency of the CubeSat, when computed by analysis, shall be no less than 130 Hz, on the condition that the four rail ends on +/-Z are rigidly fixed. Compliance can be also demonstrated by test, in which case the first natural frequency shall be no less than 115 Hz.
- 4.10.2.1. – Quasi-Static Load Analysis – If qualification is not demonstrated by test, a qualification factor of 2 is requested with respect to the design limit. Quasi-static load analysis shall thus consider a load of ± 20 g in any direction.

9.5.2 FEA Simulation Setup

Table 1 below shows the material properties of Aluminium 6061, which is the material selected to be used to manufacture the CubeSat chassis. In order to ensure there is sufficient safety margins, it was aimed to have a yield stress safety of factor of 2 for the chassis design. This means that the maximum stress of the chassis cannot exceed a design limit of 138MPa. The FEA simulation was only carried on the primary chassis structure, meaning that no internal components and supporting structures are included in the analysis.

Table 9.2: Material Properties of Aluminium 6061

Material Properties	Value
Density (tonne.mm^{-3})	2.7 x 10^{-9}
Young's Modulus (MPa)	70000
Poisson Ratio	0.33
Yield Stress (MPa)	276
Design Limit (MPa)	138

After establishing the material properties of the chassis, the load and boundary conditions were established. As mentioned in the section above, the FYS requirements requests testing of an acceleration of 20g in any direction, namely x, y, and z, for quasi-static load analysis. Since the maximum mass of a 2U CubeSat is 4kg, this was used to calculate the gravitational loads that acts on the structure:

$$F = ma = 4kg \times 20 \times 9.81 = 784.8N \tag{9.1}$$

On Abaqus, the load calculated in equation 1 was modelled as a pressure force on a surface. The rails are the only part of the CubeSat that will be in contact and interfacing with the P-POD deployer, as shown in 9.13 above. Therefore, the loads and boundary conditions applied on the FEA model in the X, Y, and Z direction were placed on the rail surfaces, as shown in 9.14 and 9.15 below. Firstly, how the load and the boundary conditions for X direction are applied to the rails are shown in 9.14 (left). It shows that a pressure force is applied to the rails on one side of the of the panel in the X direction, with the opposing rails being constrained to limit displacement and rotation in all directions. The area of the rails in the X-direction were found to be:

$$A_x = 3825mm^2 \tag{9.2}$$

Using the area of the rails along with the magnitude of the load calculated in equation 1, the pressure force applied to the rails in the X-direction were then calculated as follows:

$$P_x = \frac{F}{A_x} = \frac{784.8N}{3825mm^2} = 0.205MPa \tag{9.3}$$

The load and boundary conditions were applied in the same manner for the Y direction, which is demonstrated by the 9.14(right). Since the area of the rails are the same for both the X and Y direction, the magnitude of the pressure force applied is also identical:

$$P_y = P_x = 0.205MPa \tag{9.4}$$

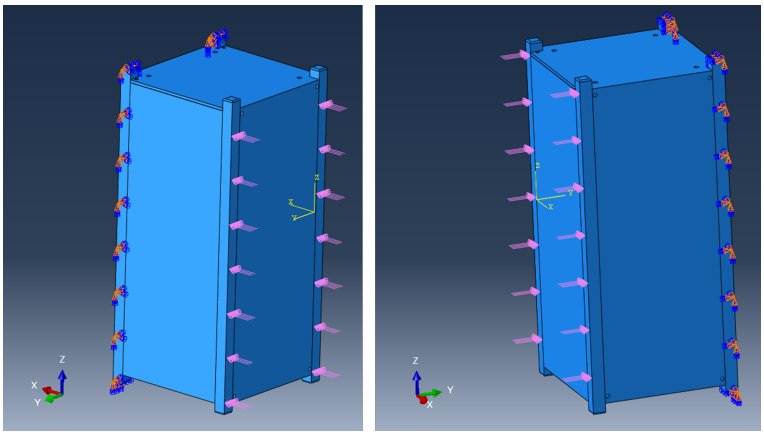


Figure 9.14: Load and boundary conditions applied on the FEA model in X (left) and Y (right) direction.

Finally, the load and boundary conditions applied for the Z direction is shown in 9.15 below. In contrast with the X and Y directions, the area of contact is much smaller for the Z direction as only the ends of the rails are in contact with the launcher. Similar to the other directions, the pressure force was applied to one side of the rails whilst the rails in the opposing ends were constrained to limit displacement and rotation. The area of the end of the rails were found to be:

$$A_z = 289mm^2 \tag{9.5}$$

This means the pressure applied has a magnitude of:

$$P_z = \frac{F}{A_z} = \frac{784.8N}{289mm^2} = 2.72MPa \tag{9.6}$$

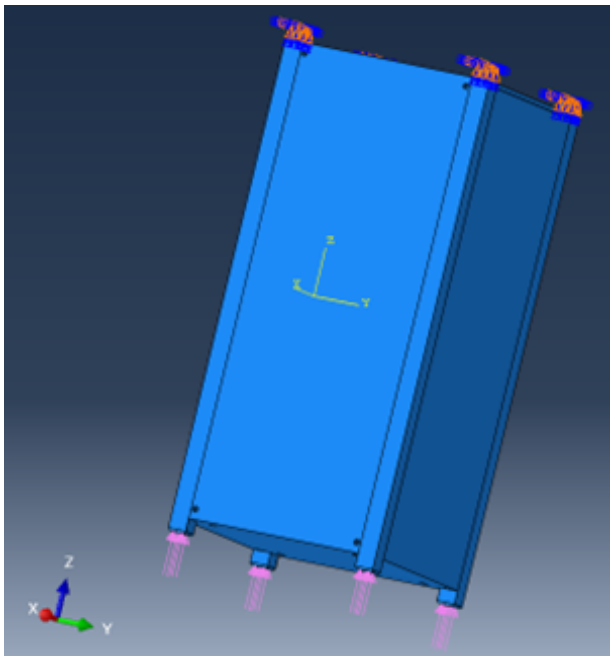


Figure 9.15: Load and boundary conditions applied on the FEA model in the Z direction

Using the load and boundary conditions established for the X, Y, And Z direction, FEA analysis of the chassis structure will consist of simulating the load case in each of

the three directions. The results obtained will be used to inform the iterative design optimisation process which aims to reduce the mass of the chassis whilst maintaining structural integrity.

9.5.3 Grid Independence Test

After defining the materials, loads, and boundary conditions of the FEA model, a grid independence test was conducted to investigate the optimum seed size for the mesh that will provide accurate results with a reasonable simulation time. Firstly, due to the complex geometry of the chassis, a tetrahedral mesh was used for the stress analysis. In the grid independence test, a total of 8 tests with seed sizes varying from 3mm to 11mm was conducted. Table 2 below shows a summary of the results from the grid independence test. The resultant Von Mises stress and displacement values were plotted against the number of elements and the total CPU time required to compute the simulation, this is shown in 9.16 and 9.17 respectively.

The plots shows that the resultant stress and displacements begins to converge with seed size smaller 5mm, which is 52797 elements. Therefore, it was decided that a seed size of 4mm (59949 elements) will be used as it is deemed the optimum mesh. This is because further reducing the seed size will significantly increase the time of the simulation but with only minimal benefits on the accuracy of the results.

Table 9.3: Summary of Grid Independence Test Results

Test Number	Seed Size (mm)	No. of Elements	Von Mises Stress (MPa)	Displacement (mm)	CPU Time (s)
1	11	19480	59.82	0.02732	5.1
2	8	26717	15.94	0.02987	6.7
3	7	33740	29.79	0.03019	8.8
4	6	40657	24.16	0.03039	10.9
5	5	52797	18.87	0.03119	15.3
6	**4**	**59949**	**18.15**	**0.03147**	**17.5**
7	3.5	74237	14.88	0.0316	22.6
8	3	99544	19.88	0.03179	32.3

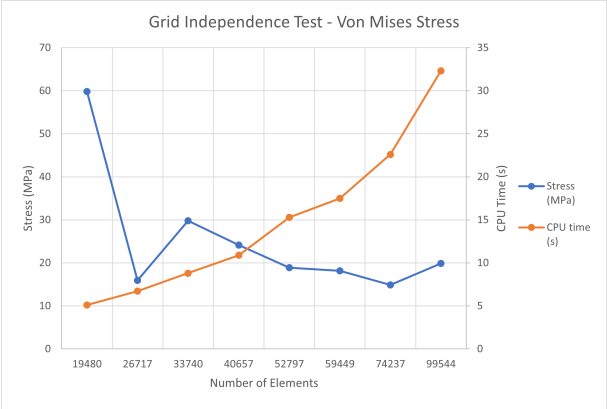


Figure 9.16: Grid Independence Test measuring Von Mises Stress

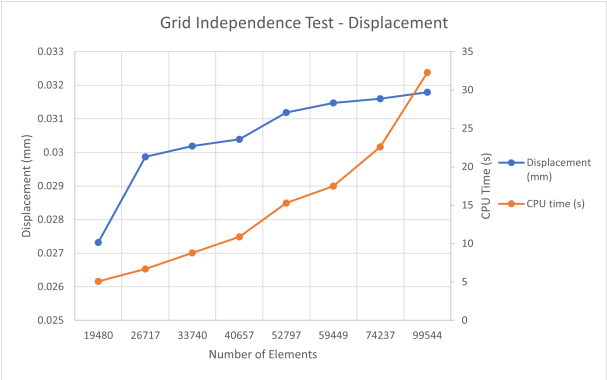


Figure 9.17: Grid Independence Test measuring displacement

9.5.4 Initial FEA Simulation

After carrying out a grid independence test, an initial FEA was conducted on the existing chassis structure design. This aims to identify the stress and displacement levels experienced by the chassis under the FYS specified load cases, which would allow excess materials from areas of the chassis that experiences low stress to be removed, minimising the mass of the chassis. The current mass of the chassis is 603g, reducing this will not only help to ensure the designed CubeSat remain compliant with the 4kg mass limit from as specified by the CubeSat specification[6], but it will also allow more mass margins to be allocated to the payload and other satellite internal components. Moreover, the results of the analysis will also highlight areas of the chassis with high stress concentration that has a greater probability of failure which may require further redesigning. The core body structure of a spacecraft typically accounts for 10% of the spacecraft's dry weight[26]. Therefore, the optimisation process should aim to achieve a maximum target weight of 400g (10% of 4kg).

Three separate simulations were conducted to simulate the loads in each of the X, Y, and Z direction. The Von Mises stress distribution and displacement plots with the load applied in the X axis is shown in 9.18. The resultant stress and was found to be 22.7MPa, which is significantly below the yield strength of 276MPa. The maximum displacement was found to be 0.032mm, this is acceptable as there is a sufficient gap between internal component and the chassis . A very similar result can be observed for the Y direction (9.19), with the maximum stress found to be 8.39MPa with a maximum deflection of 0.0276mm. Finally, 9.20 shows the FEA results of the Z direction, which shows the maximum stress and deflection to be 23.4MPa and 0.00029mm. The stress contour plots of all 3 load cases also shows that there are no high stress areas on any of the panel surfaces and the stress levels are comfortably below the design limit, magnitudes of displacement are also very low. This suggests that the current chassis design is rather conservative and have capacity for design optimisation. Therefore, the following sections will follow an iterative approach to remove materials that are not stress critical.

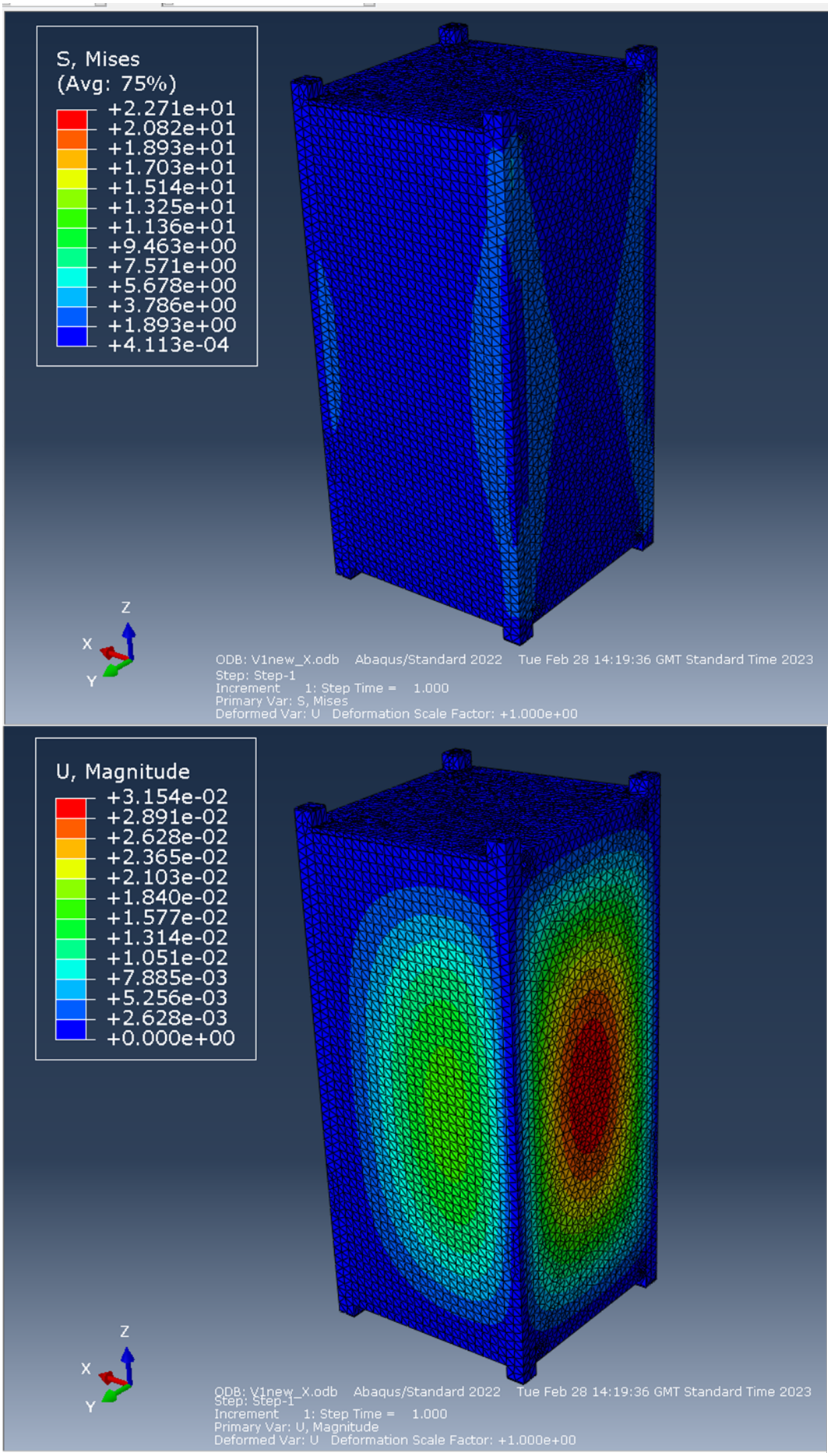


Figure 9.18: Von Mises stress contour plot (top) and displacement plot (bottom) of initial chassis design with load applied in the X direction

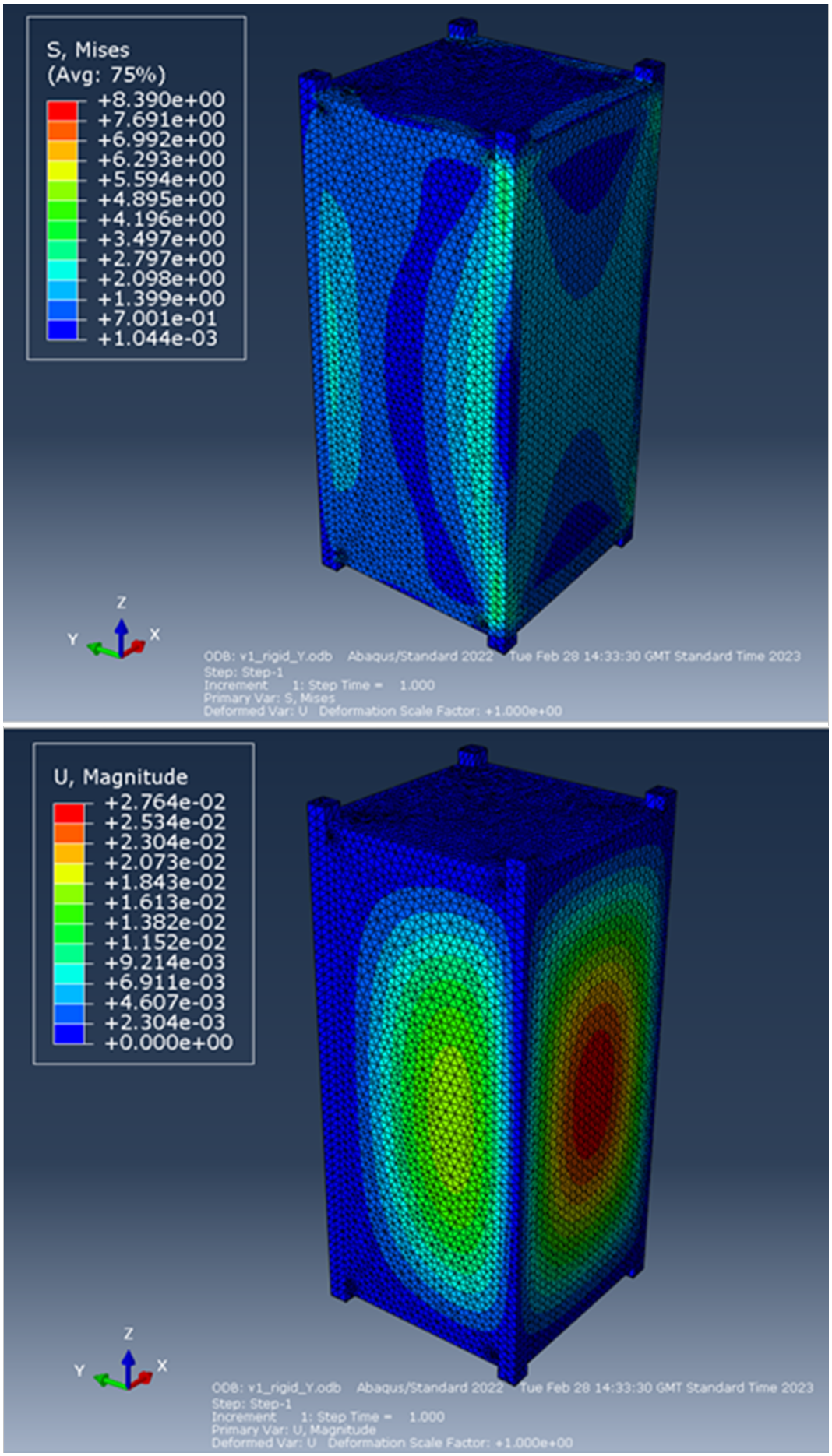


Figure 9.19: Von Mises stress contour plot (top) and displacement plot (bottom) of initial chassis design with load applied in the Y direction

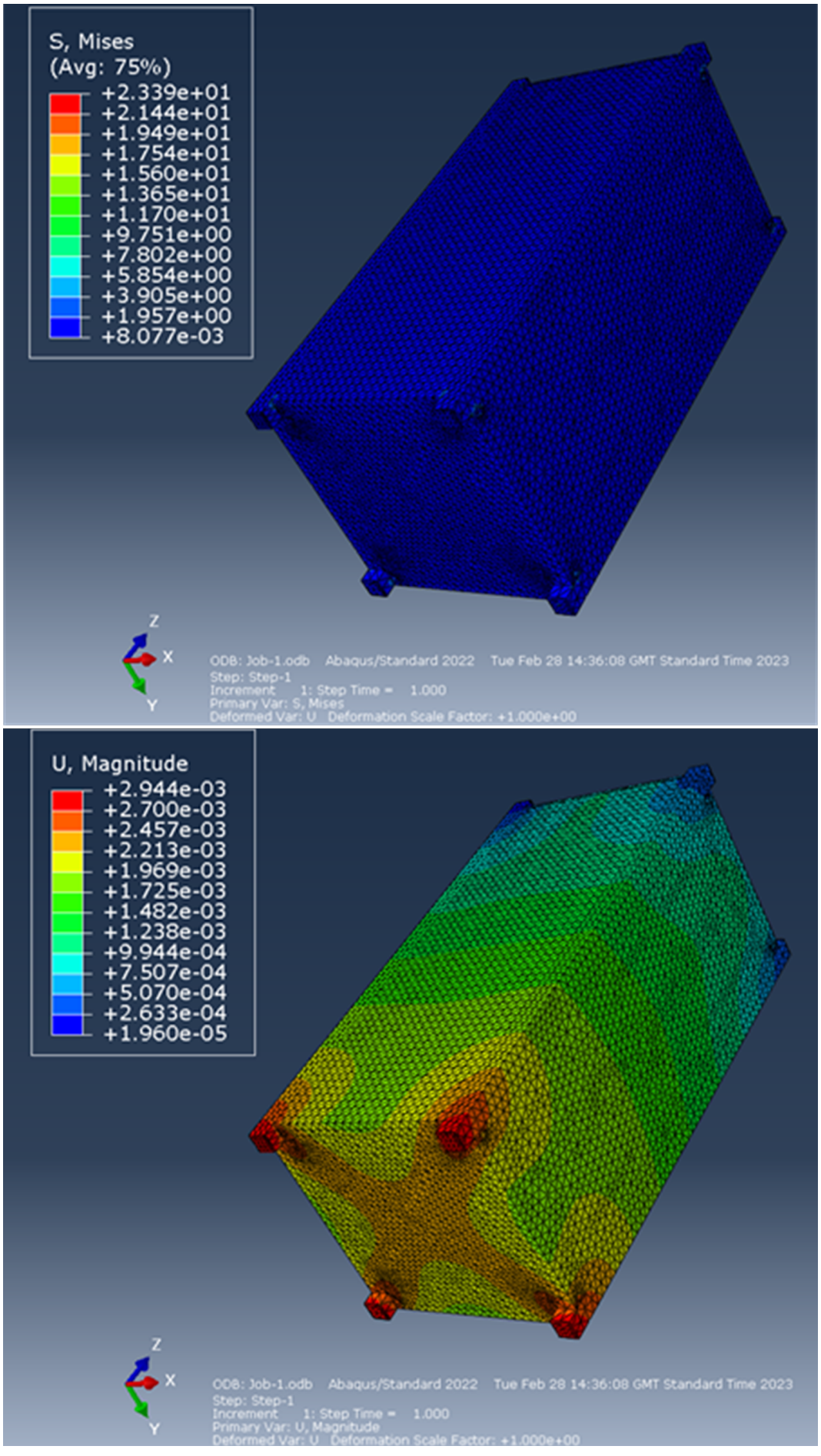


Figure 9.20: Von Mises stress contour plot (top) and displacement plot (bottom) of initial chassis design with load applied in the Z direction

9.5.5 1^{st} Iteration

In this first iteration, circular extrusions of 75mm diameter were made to the side, top, and bottom panels. This resulted with a chassis mass of 364g, which is a percentage decrease of 39.6%. 9.21 shows a maximum stress and displacement of 12.3MPa and 0.065mm in the X direction, respectively.

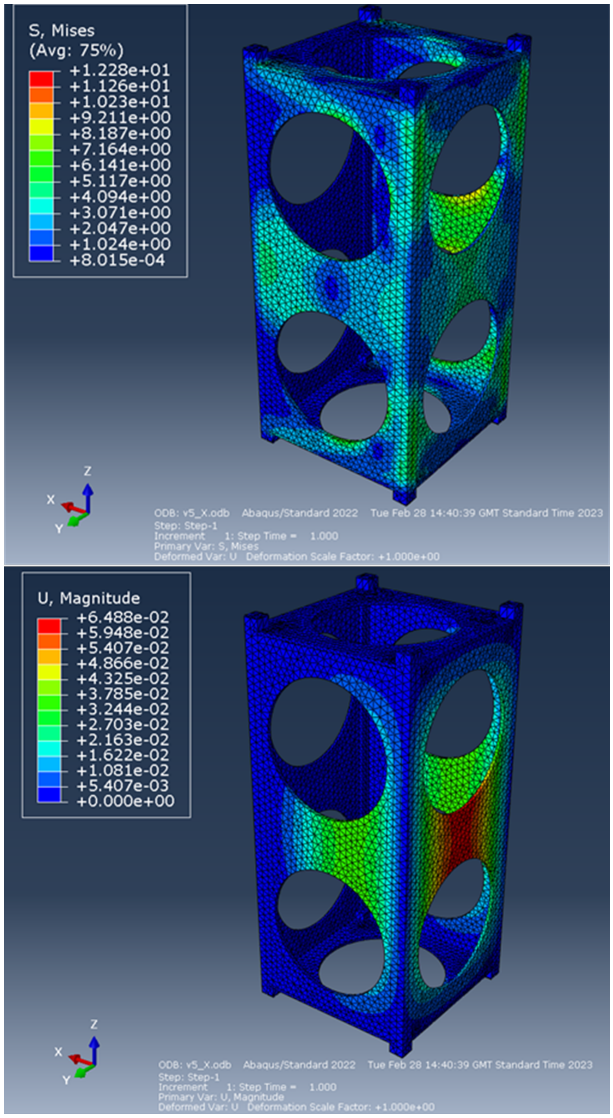


Figure 9.21: Von Mises stress contour plot (top) and displacement plot (bottom) of 1st iteration with load applied in the X direction

9.22 shows a maximum stress and displacement of 9.76 MPa and 0.060mm in the Y direction, respectively. Similar stress distribution characteristics can be seen for both the X and Y direction, the contour plots shows that the stresses are evenly distributed across the panels. Although minimal stress concentration around the circumference of the circular cut out can be observed for both the X and Y direction, the magnitude is very low compared to the design limit. For both directions, the displacement plots shows that the centre of the panel deflects the most. Again, since the magnitude of deflection is very low, it is not an area of concern

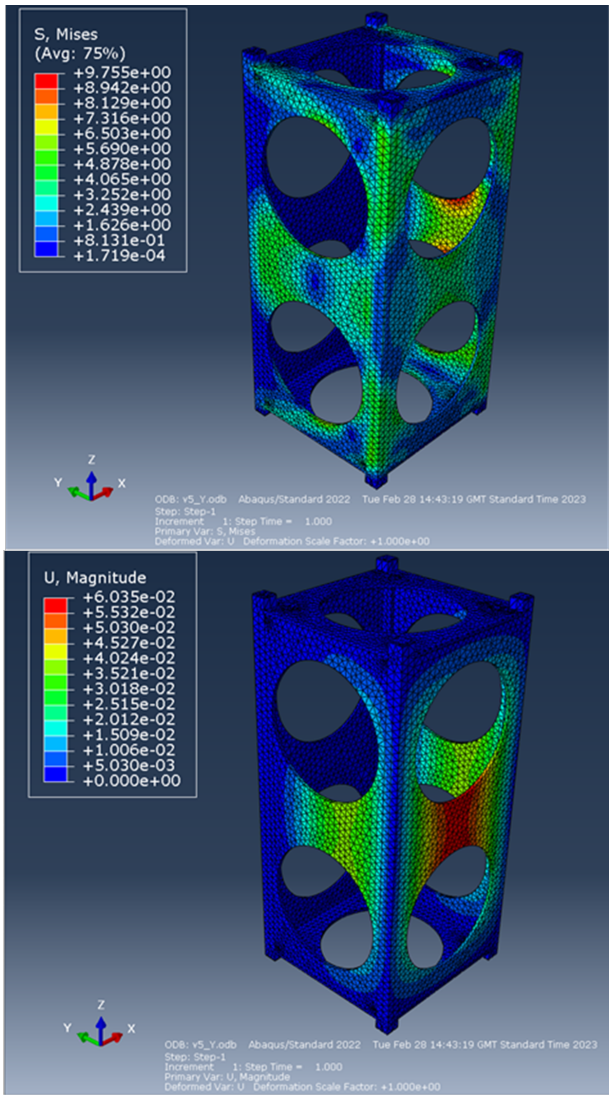


Figure 9.22: Von Mises stress contour plot (top) and displacement plot (bottom) of 1st iteration with load applied in the Y direction

In the Z direction, the maximum stress and displacement were also found to be below the limit. Studying the displacement plot, it shows that maximum displacement occurs at the rail in the -Z direction as it is where the load is applied.

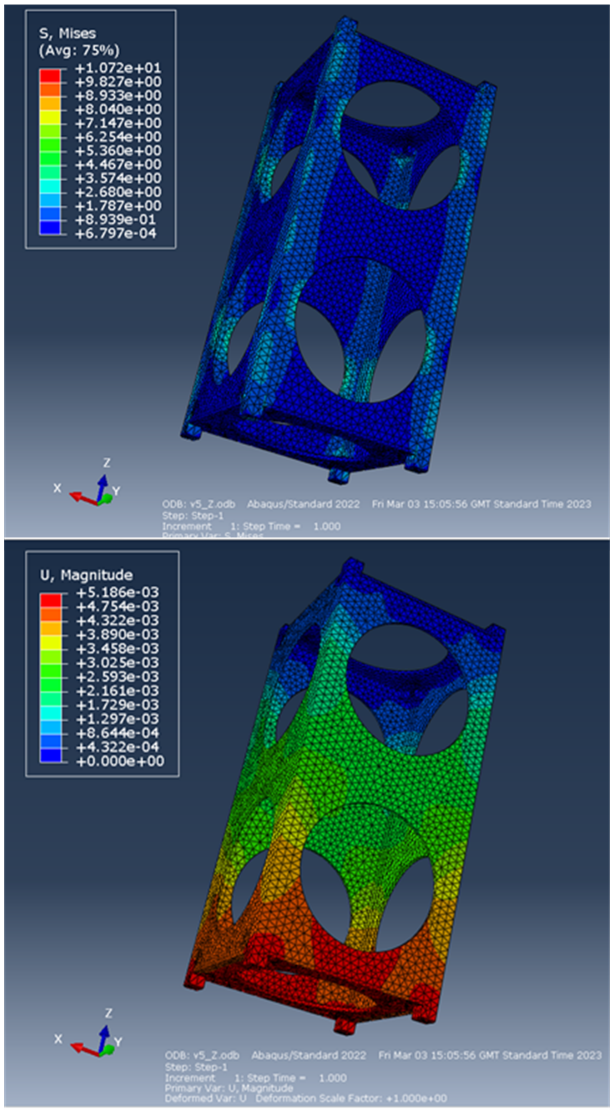


Figure 9.23: Von Mises stress contour plot (top) and displacement plot (bottom) of 1st iteration with load applied in the Z direction

9.6 2^{nd} iteration

Since the results from the previous iteration shows that there is a lot of margins for lightweighting, all panel thickness was reduced from 2mm to 1.5mm. This led to a mass reduction from 364g to 258g, which is a 29.1% reduction. Since the geometry of the chassis remained identical, the stress distribution is also very similar to the previous iteration. 9.25, 9.25, and 9.26 shows the FEA results for this iteration in the X, Y, and Z direction respectively.

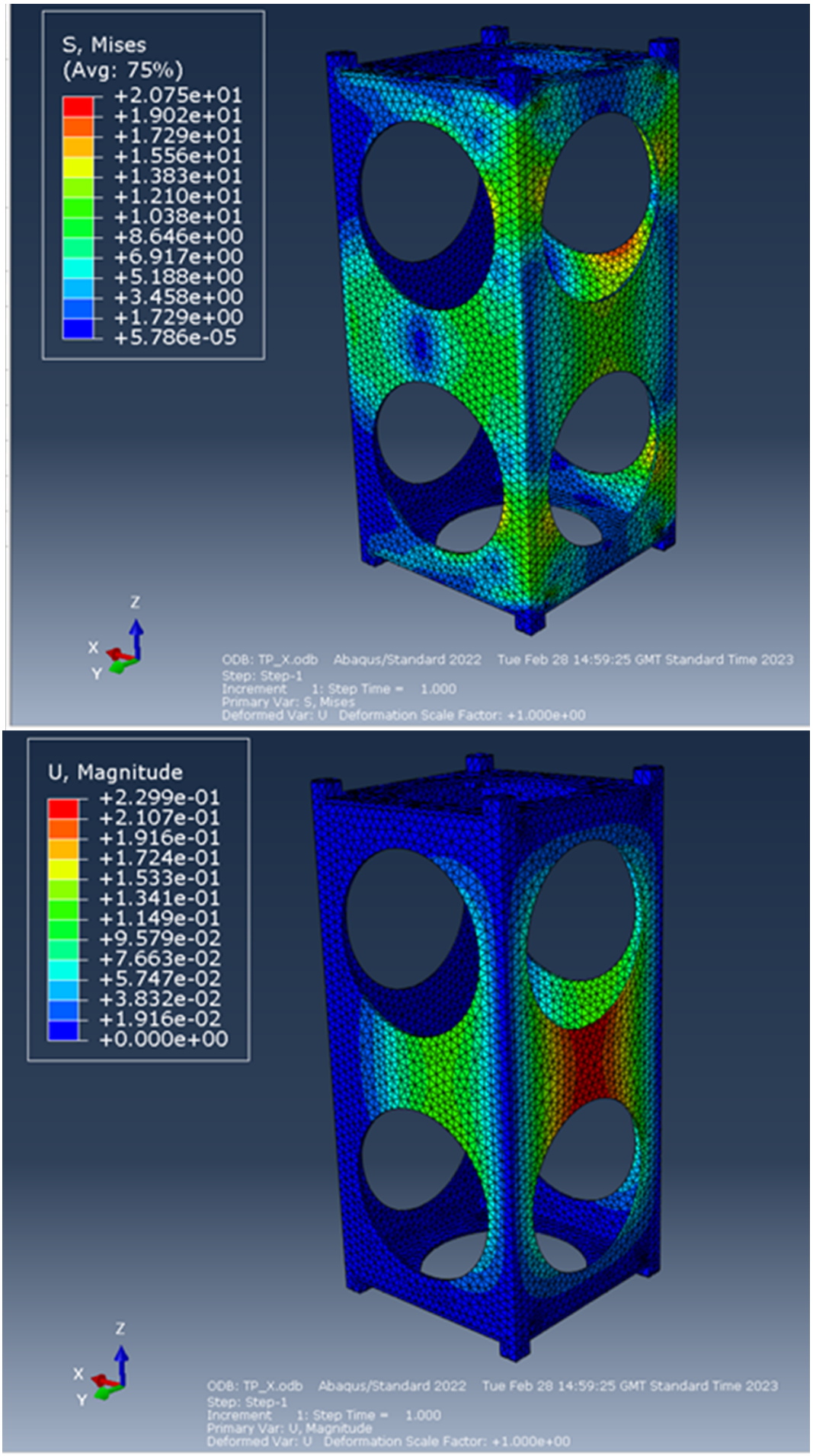
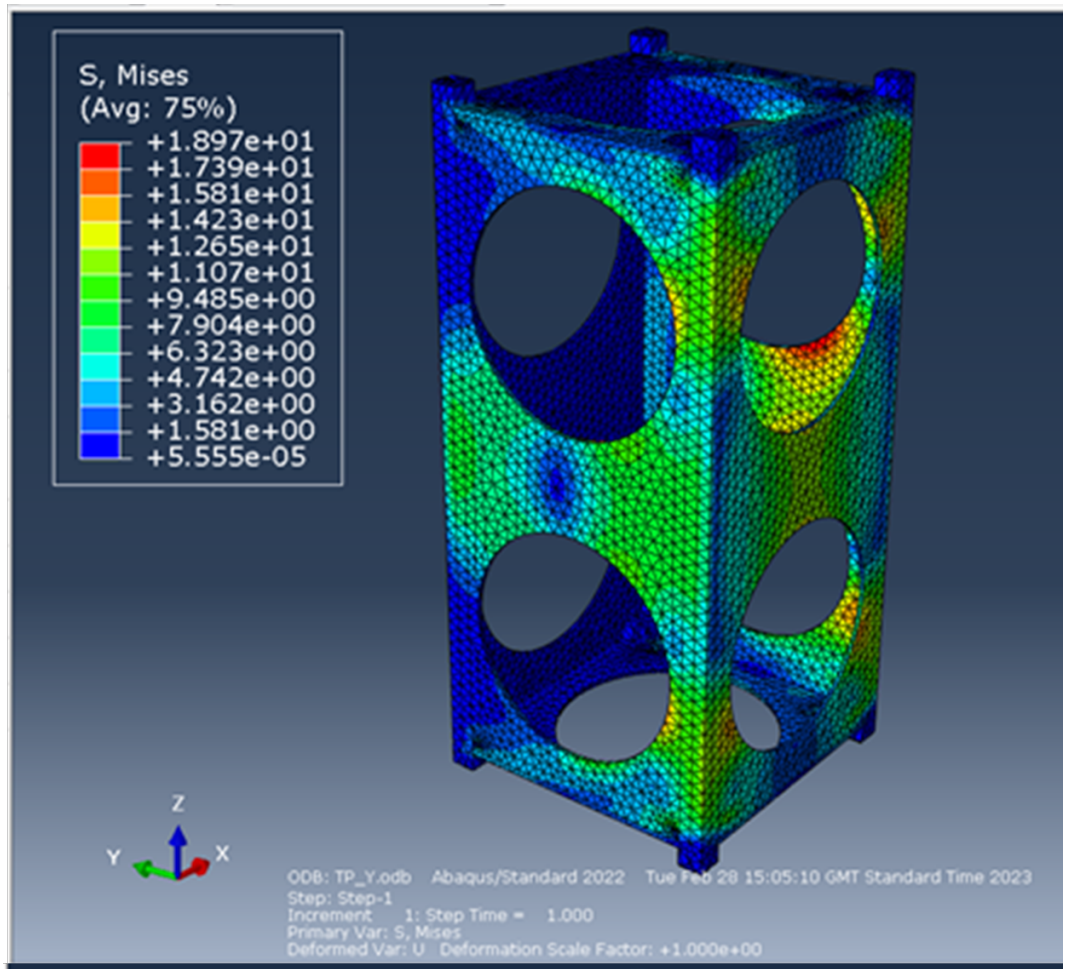


Figure 9.24: Von Mises stress contour plot (top) and displacement plot (bottom) of 2nd iteration with load applied in the Y direction

For the X and Y direction, the stress distribution remained largely the same with a slight increase in magnitudes only. However, whilst the deflection behaviour remained similar to the precious previous iteration, it can be seen that there is a significant increase in the displacement magnitude, a deflection of 0.23mm was found for both directions. The magnitude of deflection has increases roughly by an order of magnitude, from 10-2mm to 10-1mm. This is due to the thinner panels which leads to a greater degree of deflection. Despite the significant increase in displacement magnitude in both directions, the level of deflection is still acceptable for the tolerances of the CubeSat.



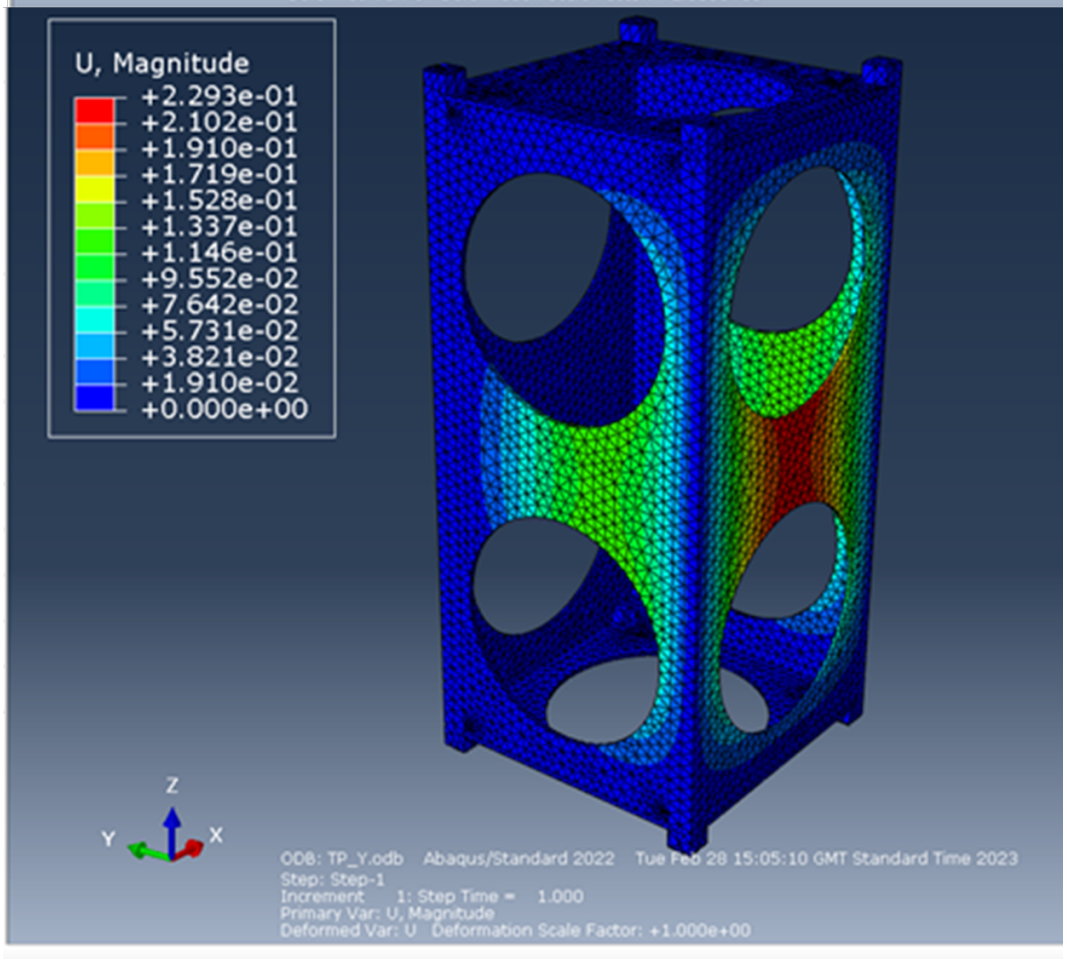
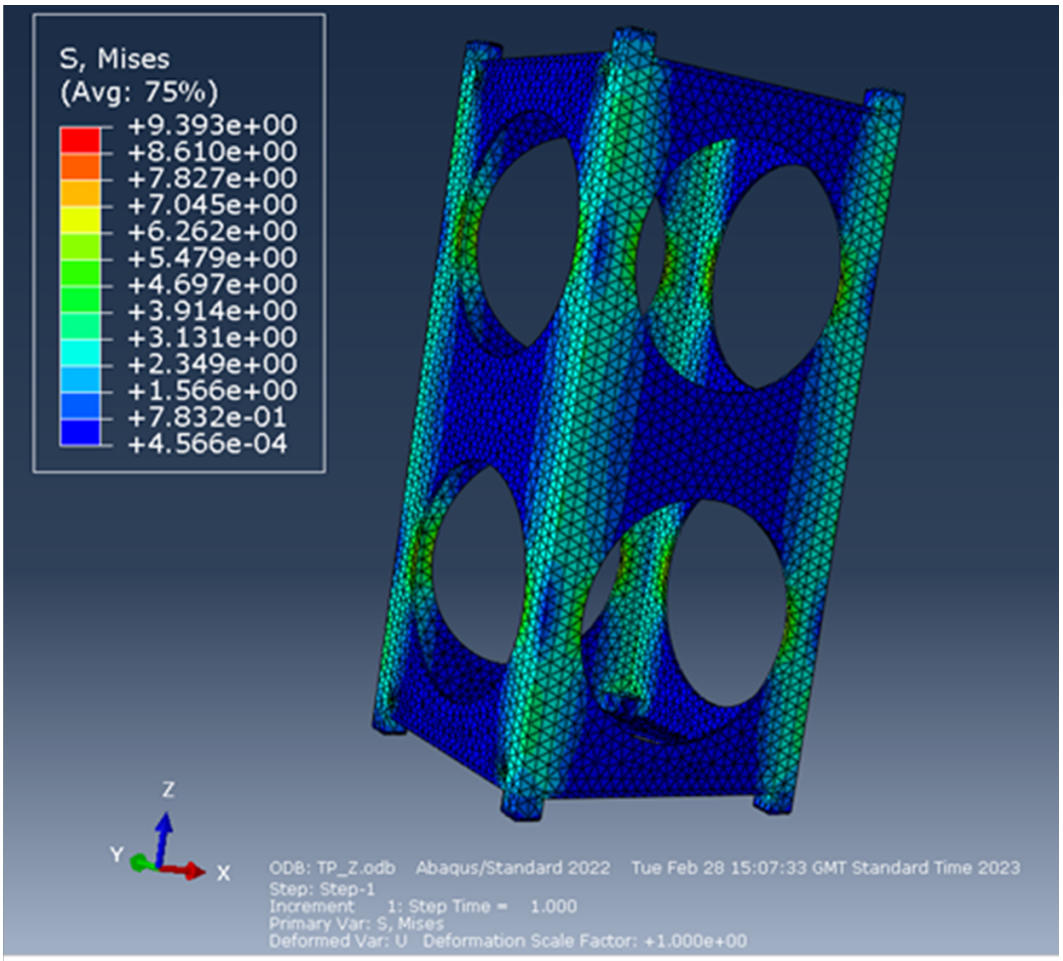


Figure 9.25: Von Mises stress contour plot (top) and displacement plot (bottom) of 2nd iteration with load applied in the Y direction

For the Z direction, 9.26 shows that the magnitude of stress and deflection remained in the same order of magnitudes in comparison to the previous iteration. This implies that reducing the thickness of the panel has minimal impact from stress loading in the Z direction. However, since reducing panel thickness significantly increase deflection in the X and Y direction, the thickness should not be further reduced as it may cause excess deflection and buckling.



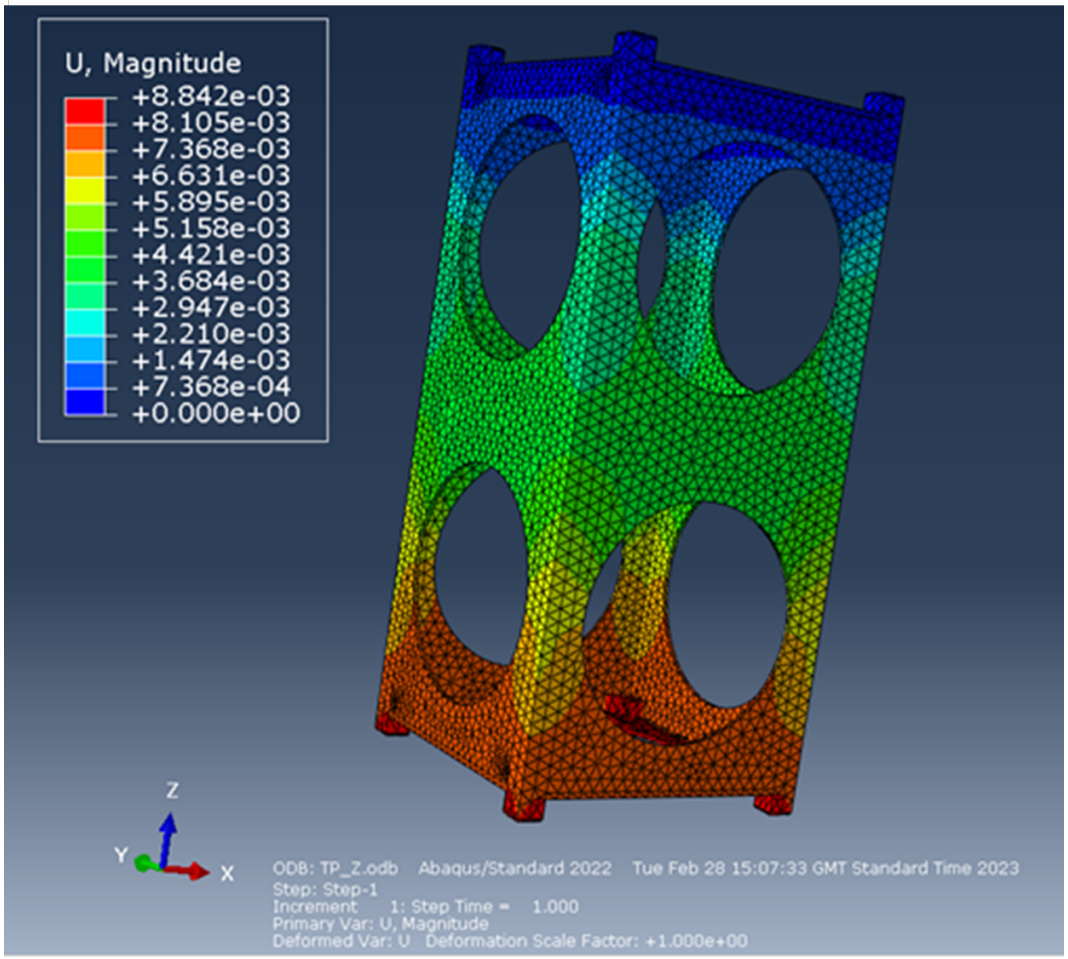


Figure 9.26: Von Mises stress contour plot (top) and displacement plot (bottom) of 2nd iteration with load applied in the Z direction

9.6.1 3^{rd} Iteration

To further reduce the mass of the chassis, the cut outs on the side panels were enlarged into rectangular shapes to remove more materials. 5mm fillets were applied to all corners of the rectangular cut outs to mitigate the effects of stress concentration in the corner. Moreover, design driven changes were also incorporated into this stage of the iterative design optimisation process. Firstly, according to the power budget estimations outlined in Section, the satellite would require 5.982W of power (during eclipse), meaning all the panels of the 2U CubeSat would need to be covered with solar panels to satisfy the power consumption requirements of the spacecraft. Therefore, to accommodate for the power requirements, screw holes have been added onto each of the panels to allow for solar panel attachment. However, since commercial off the shelf (COTS) solar panels are being used, it was found that the attachment points would interfere with the panel connection points in the existing design. Therefore, changes were made to align the solar panel attachment point with the panel connection point so that the same screw hole can be shared between the two. 9.27 below shows how the solar panel and the access panel share the same attachment point. With all the changes made to the chassis design, the mass of the chassis has reduced from 258g to 216g. Again, FEA simulations were carried out after the changes made, the results for the X, Y, and Z direction are shown in 9.28, 9.29, and 9.30 respectively.

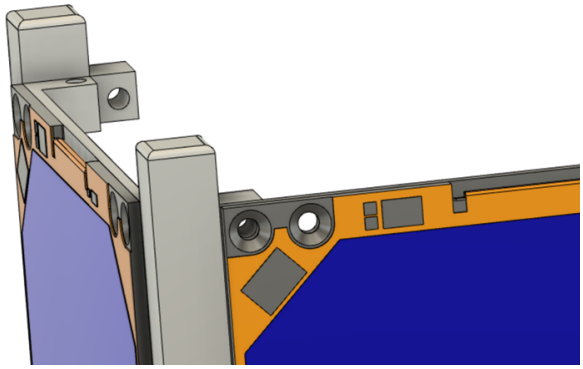


Figure 9.27: Solar Panel Attachment Design

After the changes made, it can be seen that there is a significant increase in the maximum Von Mises stress from around 20MPa to around 60MPa for the X and Y direction, as shown in Figure 17 and 18. The deflection magnitude has also increased slightly from around 0.2mm to 0.3mm. Although both stress and displacement has increased, they are still within the design limit.

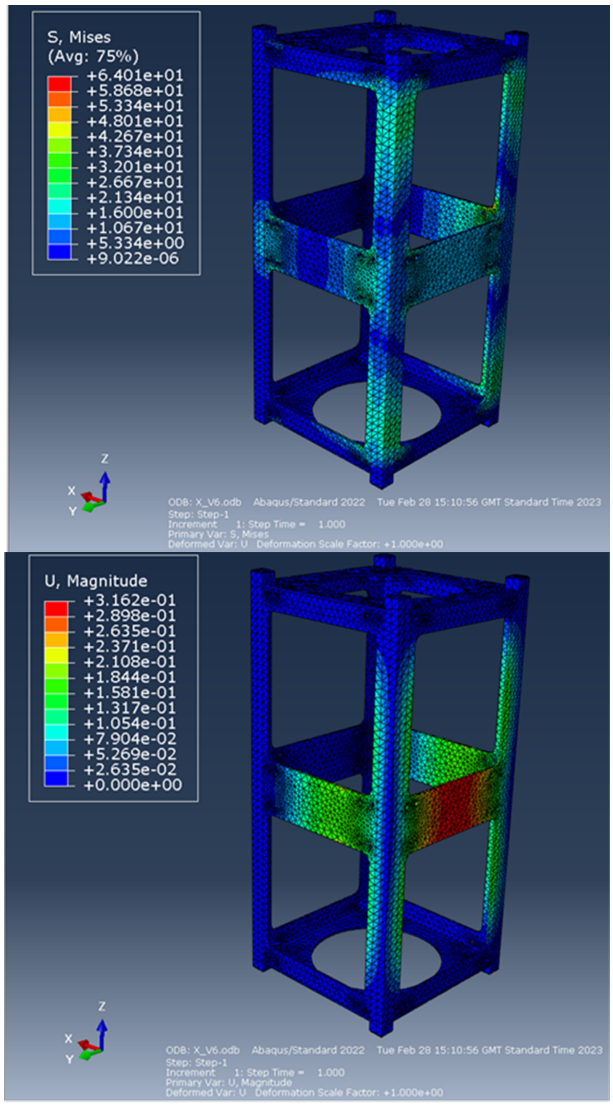


Figure 9.28: Von Mises stress contour plot (top) and displacement plot (bottom) of 3rd iteration with load applied in the X direction

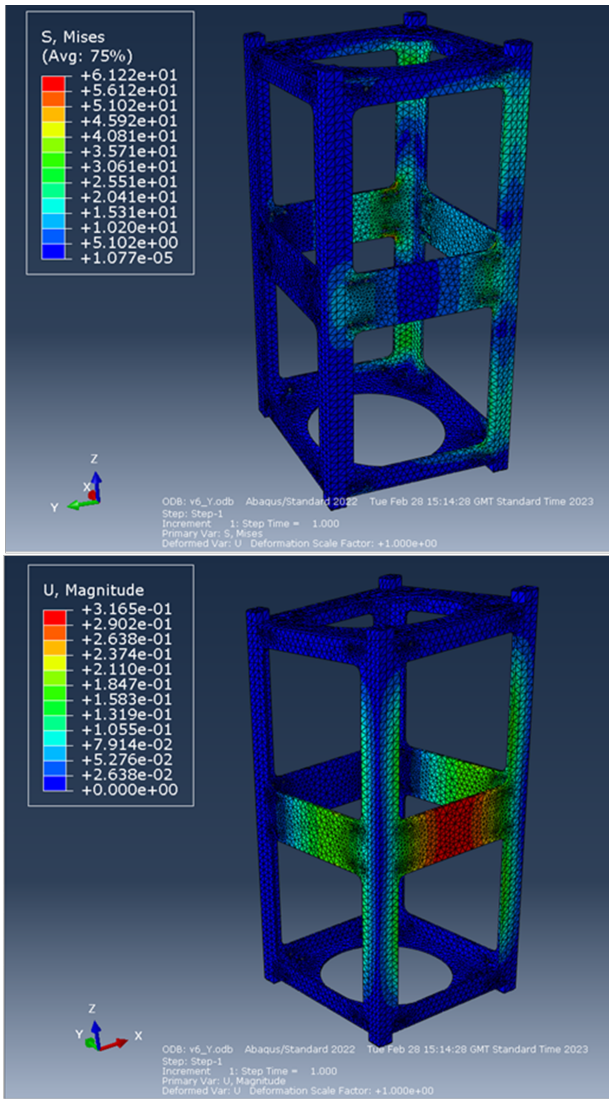


Figure 9.29: Von Mises stress contour plot (top) and displacement plot (bottom) of 3rd iteration with load applied in the Y direction

For the Z direction, 9.30 shows that the magnitude of stress and deflection has only slightly increased to 8.8MPa and 0.012mm. This demonstrates that the changes made has minimal impact on the loading conditions in the Z axis as most of the loads are transmitted along the rails, which remained unchanged throughout the optimisation process as they are constrained by the CubeSat specification. This is supported by the stress contour plot in 9.30, which shows that the majority of the stress are evenly distributed on the four rails along the Z axis.

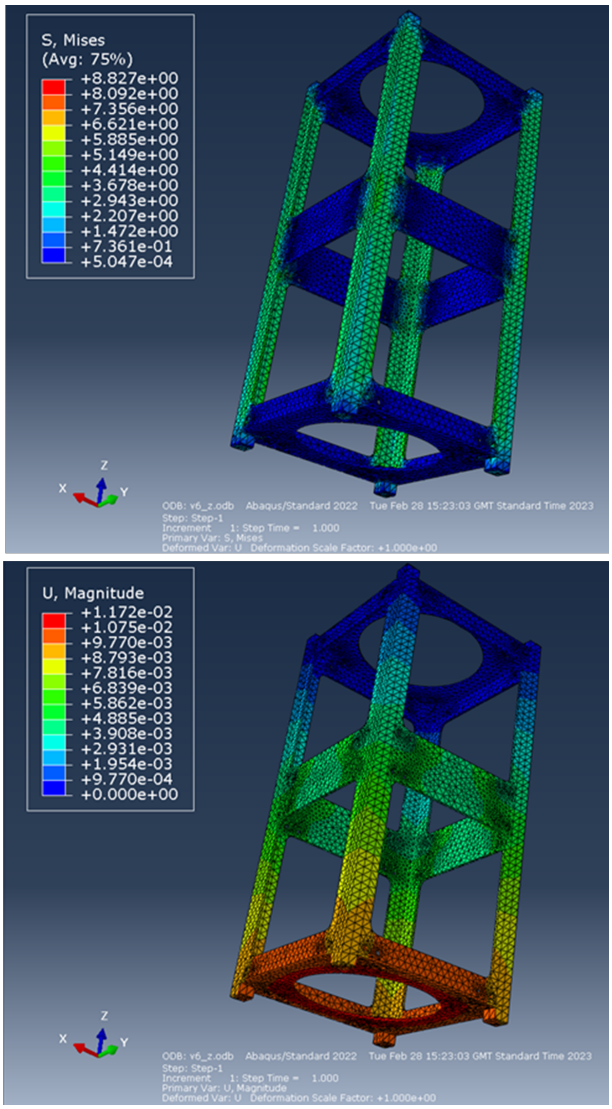
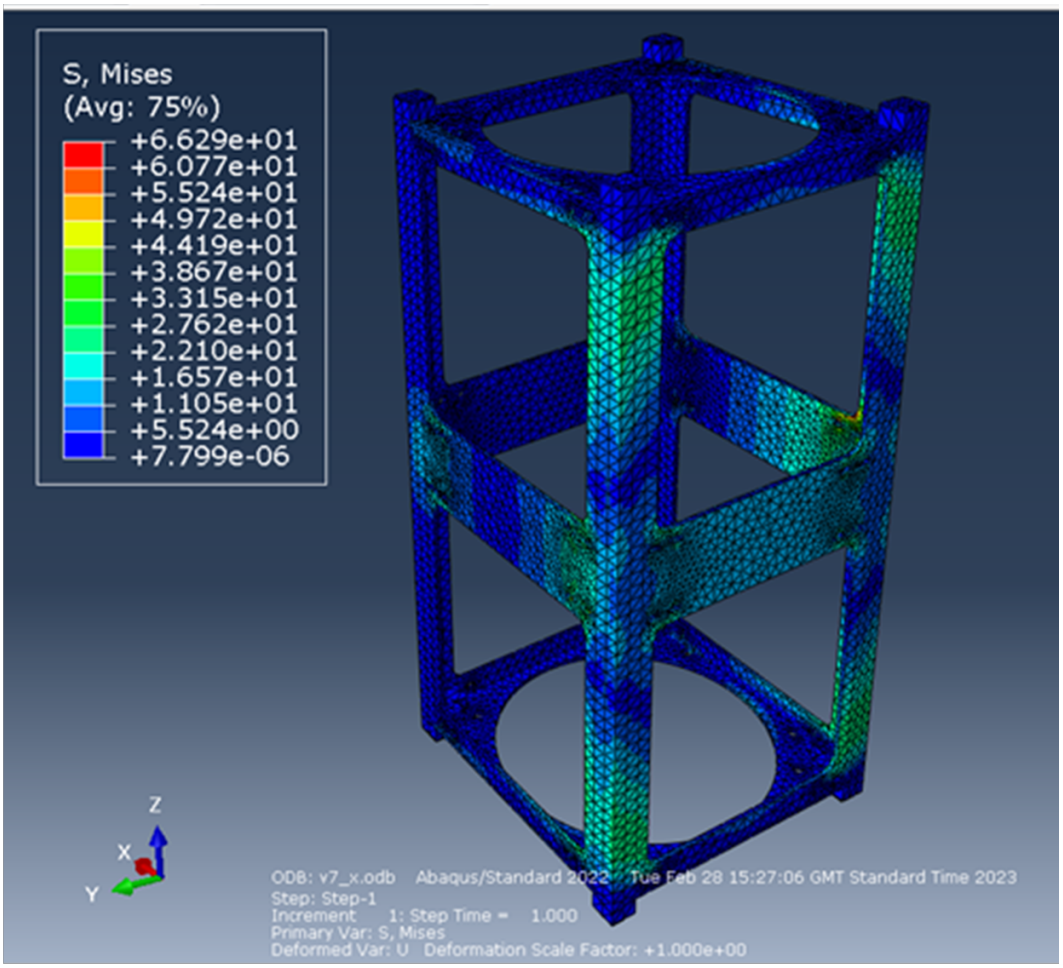


Figure 9.30: Von Mises stress contour plot (top) and displacement plot (bottom) of 3rd iteration with load applied in the Z direction

9.7 Final Design

For the final design, some changes were made to further optimise the design and reduce the mass of the chassis. The circular cut out on the top and bottom panels have been enlarged from 75mm in diameter to 90mm in diameter. The rectangular cut outs have also been made bigger whilst maintaining the filleted corners to mitigate stress concentration. As a result, the final mass of the chassis was found to be 187g, which is a significant reduction in comparison to the mass of 603g prior to optimisation process. 9.31, 9.32, and 9.33 shows the FEA results of the final chassis design in the X, Y, and Z direction.



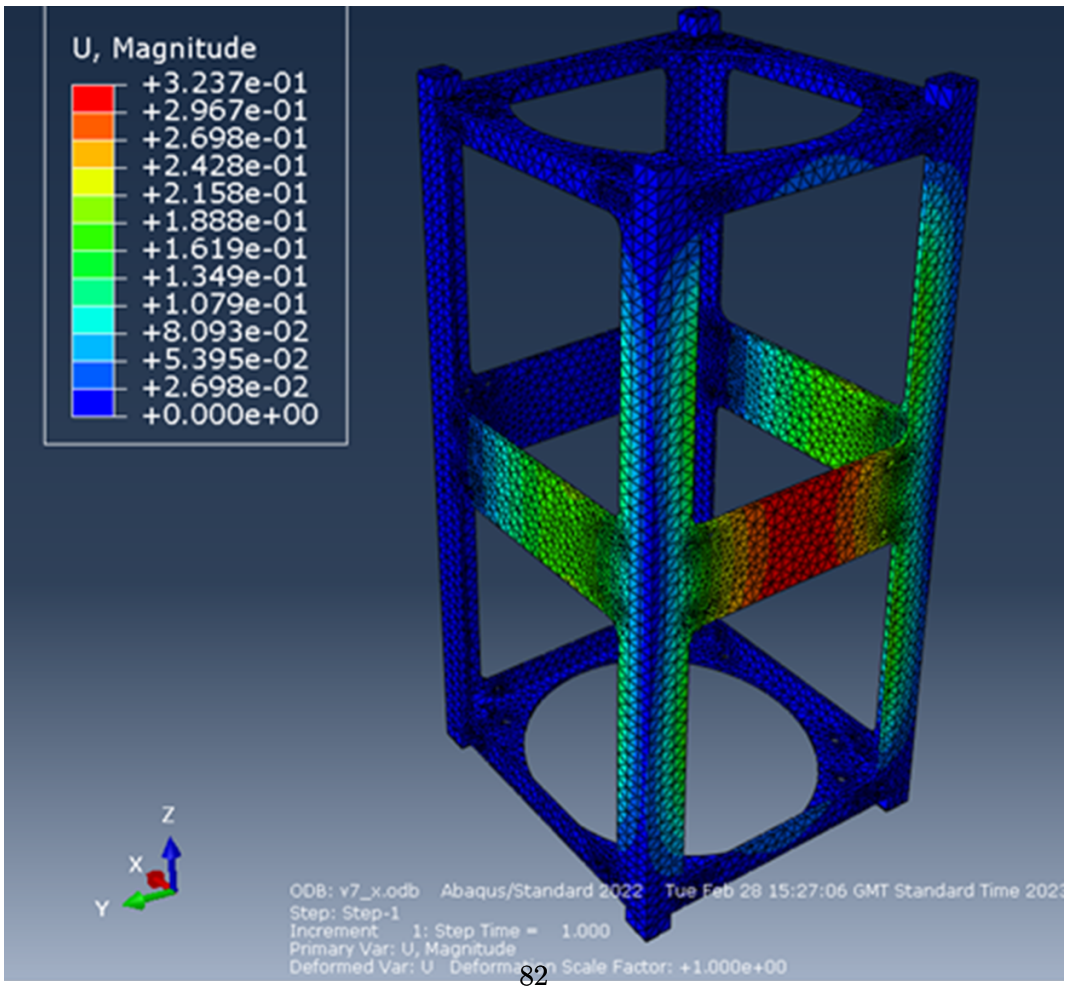
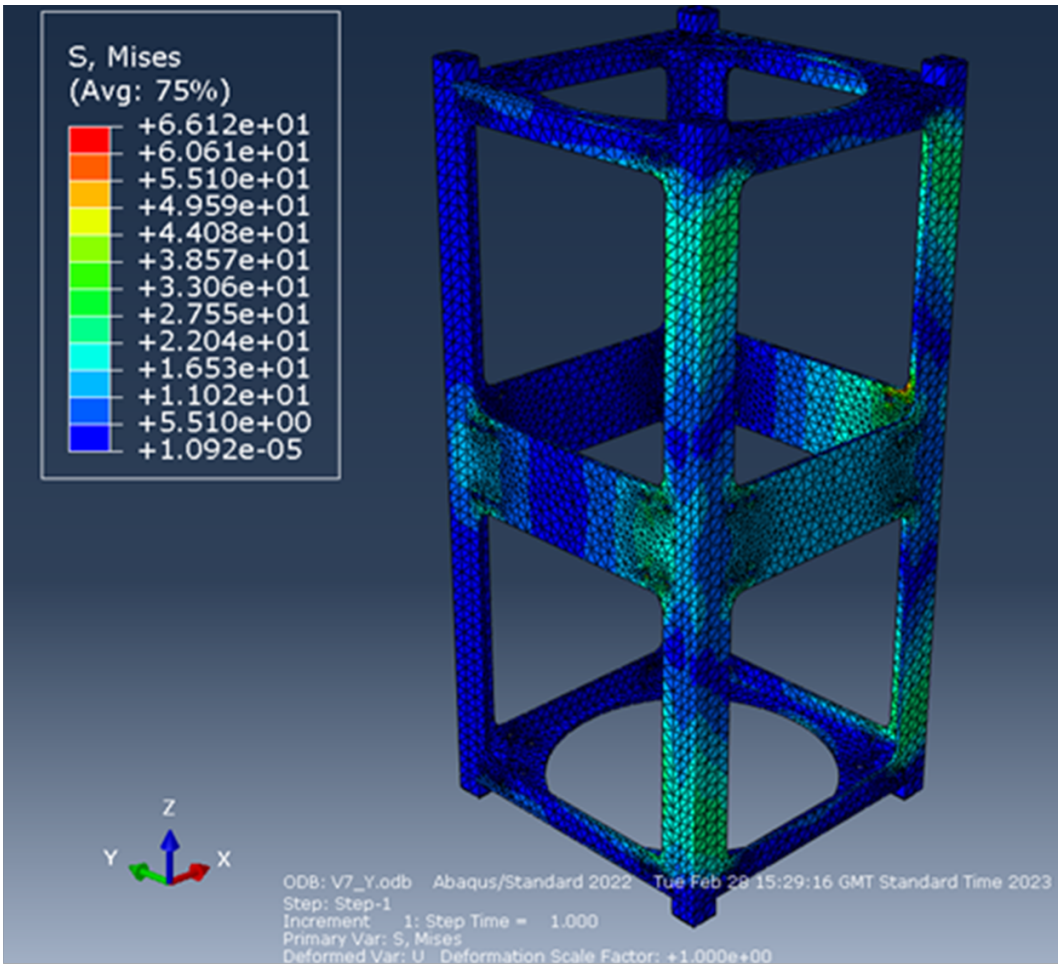


Figure 9.31: Von Mises stress contour plot (top) and displacement plot (bottom) of final design with load applied in the X direction

Similar results can be drawn when studying the FEA results in the X and Y direction. The contour plots for both shows that stress are evenly distributed across the face of the panels, which the corners of the fillets being point with the highest stress. The maximum stress was found to be around 66MPa for both the X and Y direction, which is significantly below 276MPa, the yield strength of aluminium 6061. Maximum deflection has also slightly increased to around 0.32mm for both direction, which is acceptable considering the spacing available between the internal components of the CubeSat and the chassis.



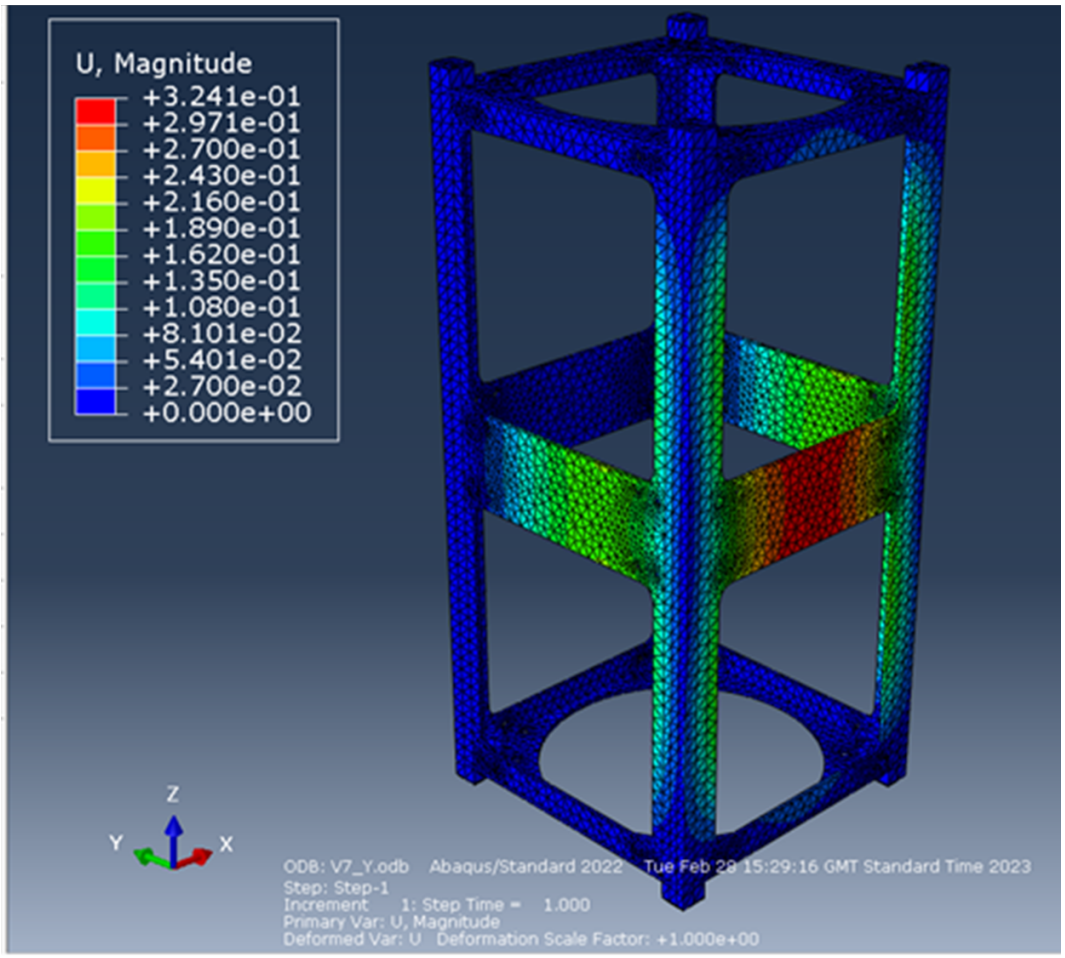


Figure 9.32: Von Mises stress contour plot (top) and displacement plot (bottom) of final design with load applied in the Y direction

For the Z direction, the stress distribution characteristics remain largely similar to the previous iteration. The maximum stress and displacement was found to be 6.8MPa and 0.012mm, which are both comfortably below the design limit. These FEA results suggests that the chassis can withstand an acceleration of 20g in all directions, therefore complying with the FYS design specification.

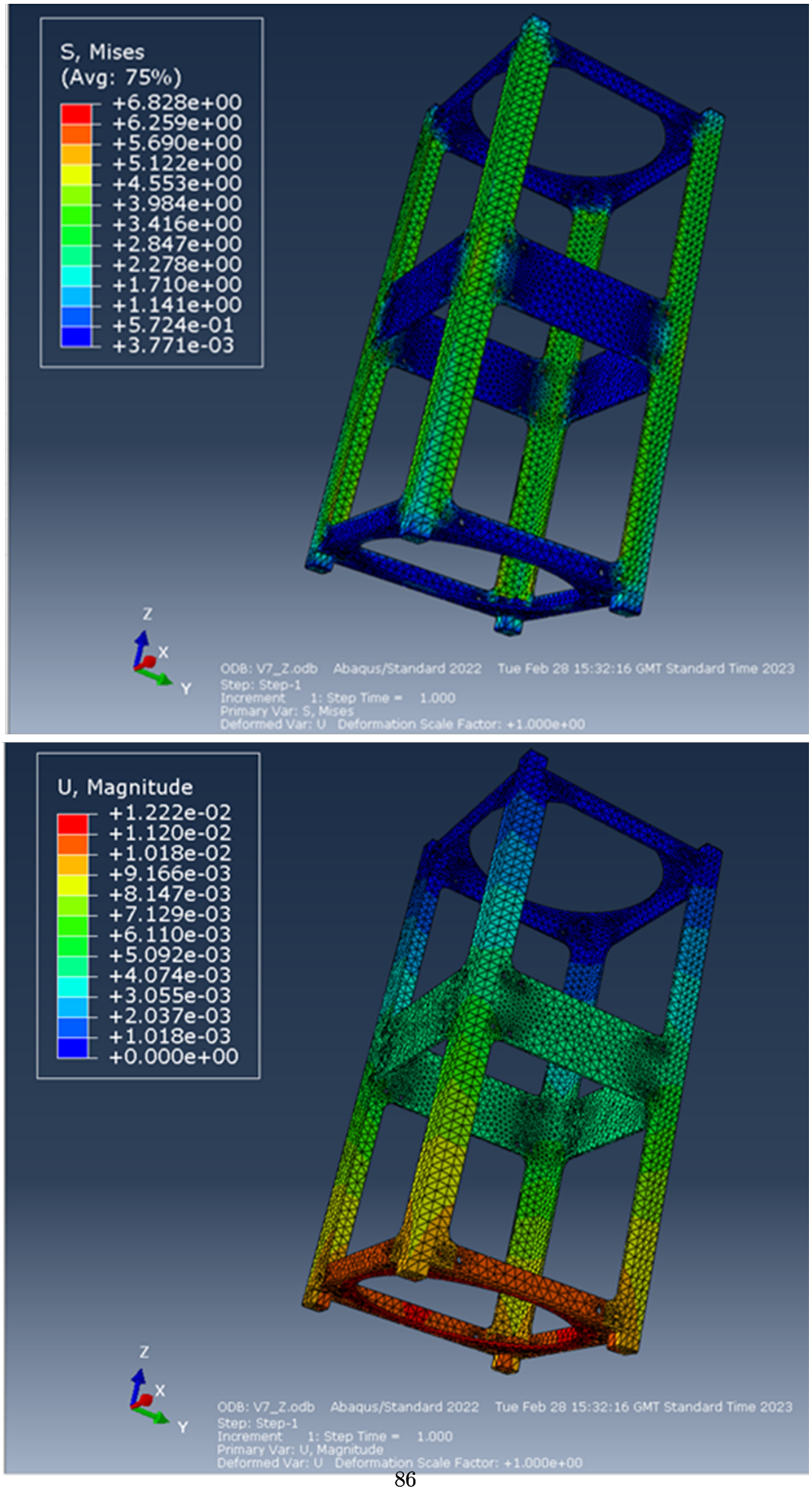


Figure 9.33: Von Mises stress contour plot (top) and displacement plot (bottom) of final design with load applied in the Z direction

Figure 9.34: Results of Natural Frequency Testing of the Final chassis structuren

E F F E C T I V E M A S S

MODE NO	X-COMPONENT	Y-COMPONENT	Z-COMPONENT	X-ROTATION	Y-ROTATION	Z-ROTATION
1	1.22685E-13	1.04950E-13	2.73424E-14	3.28720E-10	2.80346E-11	1.40530E-08
2	2.05234E-05	1.90766E-05	7.38356E-15	8.68397E-04	9.33054E-04	2.88621E-02
3	1.91537E-05	2.05685E-05	5.23427E-15	9.37874E-04	8.71665E-04	0.17114
4	4.29254E-14	1.04555E-12	1.31226E-14	3.07916E-09	7.36002E-11	9.57342E-08
5	8.97829E-14	1.52015E-13	1.35369E-16	7.73154E-12	4.25803E-12	4.52520E-08
6	3.45256E-10	5.10080E-11	1.18236E-15	1.21735E-08	8.93461E-10	0.27053
7	5.26695E-05	7.04678E-06	2.28868E-15	3.25170E-04	2.38242E-03	0.30055
8	7.08479E-06	5.26727E-05	3.66435E-13	2.40056E-03	3.17469E-04	1.09549E-03
9	7.21273E-15	3.99341E-14	2.27883E-15	3.95129E-10	6.63635E-10	8.56837E-09
10	2.47904E-12	1.07745E-13	8.00183E-16	7.56545E-03	8.14402E-03	5.26523E-08
TOTAL	9.94317E-05	9.93646E-05	6.51019E-13	1.20975E-02	1.26486E-02	0.77218

E I G E N V A L U E O U T P U T

MODE NO	EIGENVALUE	FREQUENCY (RAD/TIME)	FREQUENCY (CYCLES/TIME)	GENERALIZED MASS	COMPOSITE MODAL DAMPING
1	7.77795E+06	2788.9	443.87	1.0000	0.0000
2	1.44512E+07	3801.5	605.02	1.0000	0.0000
3	1.45269E+07	3811.4	606.61	1.0000	0.0000
4	2.80048E+07	5292.0	842.24	1.0000	0.0000
5	3.55353E+07	5961.2	948.75	1.0000	0.0000
6	5.48614E+07	7406.8	1178.8	1.0000	0.0000
7	6.35784E+07	7973.6	1269.0	1.0000	0.0000
8	6.36327E+07	7977.0	1269.6	1.0000	0.0000
9	1.05348E+08	10264.	1633.6	1.0000	0.0000
10	1.30657E+08	11431.	1819.2	1.0000	0.0000

9.8 Natural Freuency Testing and Analysis

In addition to static stress testing conducted in the section above, dynamic vibrational testing was also carried out to test determine the natural frequency of the chassis structure. This is to ensure compliance with the FYS specification – **"4.10.1. The first natural frequency of the CubeSat, when computed by analysis, shall be no less than 130 Hz, on the condition that the four rail ends on +/-Z are rigidly fixed."**. Determining the natural frequency is important as it helps to prevent dynamic coupling between the launch vehicle and the CubeSat, reducing the probability of large amplitude vibrations that can lead to structure failure[28]. As required by the FYS specification, boundary conditions were applied to the end of the rails along the Z axis. Table 3 below shows the results of the natural frequency test conducted on the final chassis design using Abaqus. The first frequency mode corresponds to the lowest natural frequency of the chassis, which was found to be 44.87Hz. This is comfortably higher than the minimum requirement of 130Hz, therefore showing that the chassis design is compliant with the FYS design specification. The mode with the effective mass closest to the total effectiveness mass is the most relevant mode for each axis, these are highlighted below in Table 3. For the X direction, the effective mass of the 7th mode is the closest mode to the total effective mass, therefore this mode corresponds to the mode of the natural frequency, also means that the maximum displacement of that axis occurs in this node. For the Y and Z axis, the most relevant mode was found to be the 8th mode. Although the magnitude of the natural frequency was found to be above the specified limit of 130Hz, the results also highlighted some potential issues with the chassis design. 9.35 shows the displacement plot of the lowest frequency mode, which indicates the magnitude of deflection being 209mm. 9.36 and 9.37 shows the displacement plot for the 7th and 8th natural frequency mode, which are the most relevant modes as discussed above. The magnitudes of these modes were found to be around 140mm, which is in a similar order of magnitude as the 1st mode. The magnitude of the deflection highlights potential issue with the chassis structure. Firstly, excess deflection can lead to failure of the chassis structure. Also, large degree of deflections would mean that the chassis would deform excessively and exceed dimensional constraints outlined by the CubeSat, hence interfering with the

launch vehicle. Given the time constraints of this project, it is not feasible for further simulations and design changes to be carried out. Therefore, this should be recorded as an open point of the current design. It is recommended that future teams should include internal components and secondary support structures when conducting another natural frequency simulation using Abaqus. This would mean the model used will be a more accurate representation of the CubeSat, which can alter and reduce the magnitude of deflection. It is also recommended that further works can be carried out in experimenting with different chassis designs that would result in deflections of smaller magnitudes.

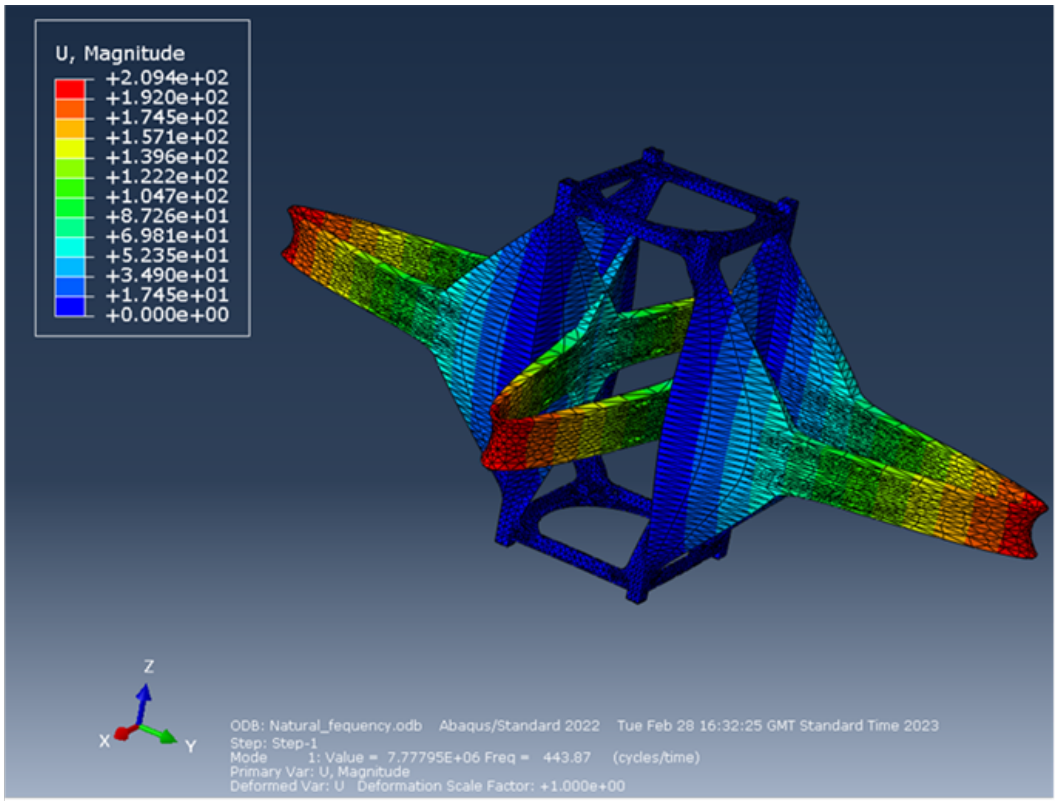


Figure 9.35: Displacement plot from natural frequency analysis of the final design showing the mode of lowest natural frequency (mode 1)

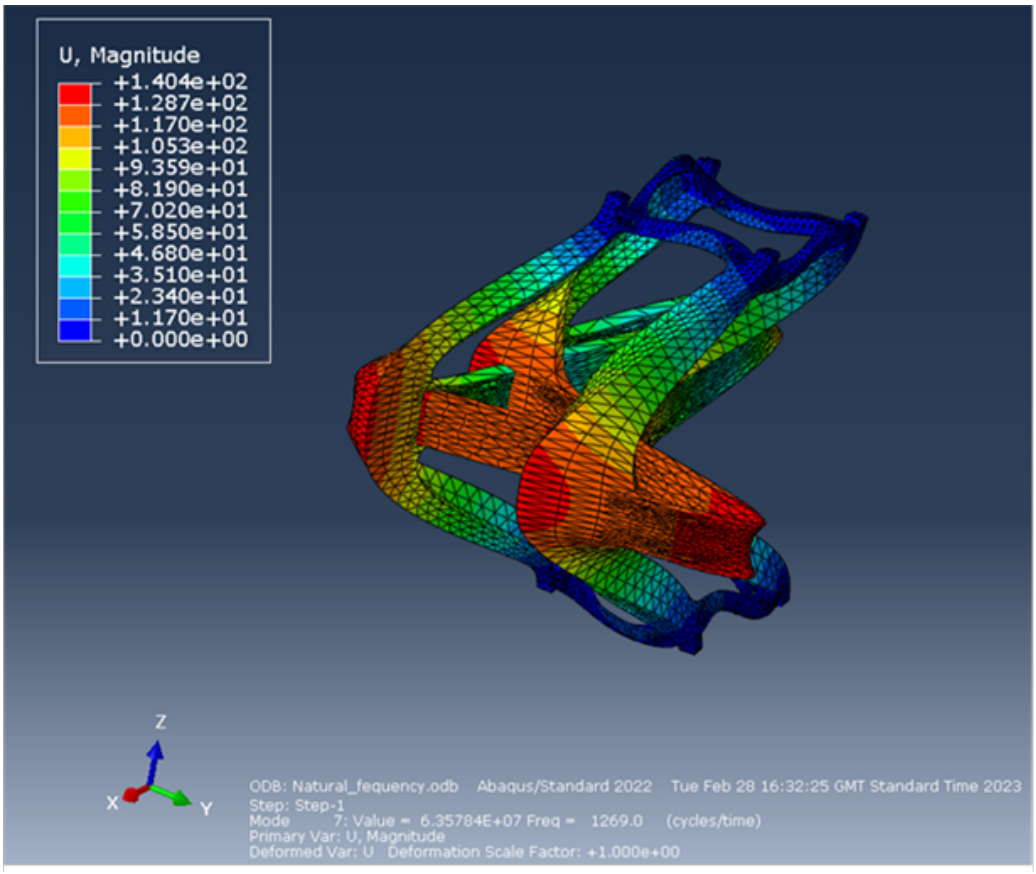


Figure 9.36: Displacement plot from natural frequency analysis of the final design showing the most relevant mode in the X axis (mode 7)

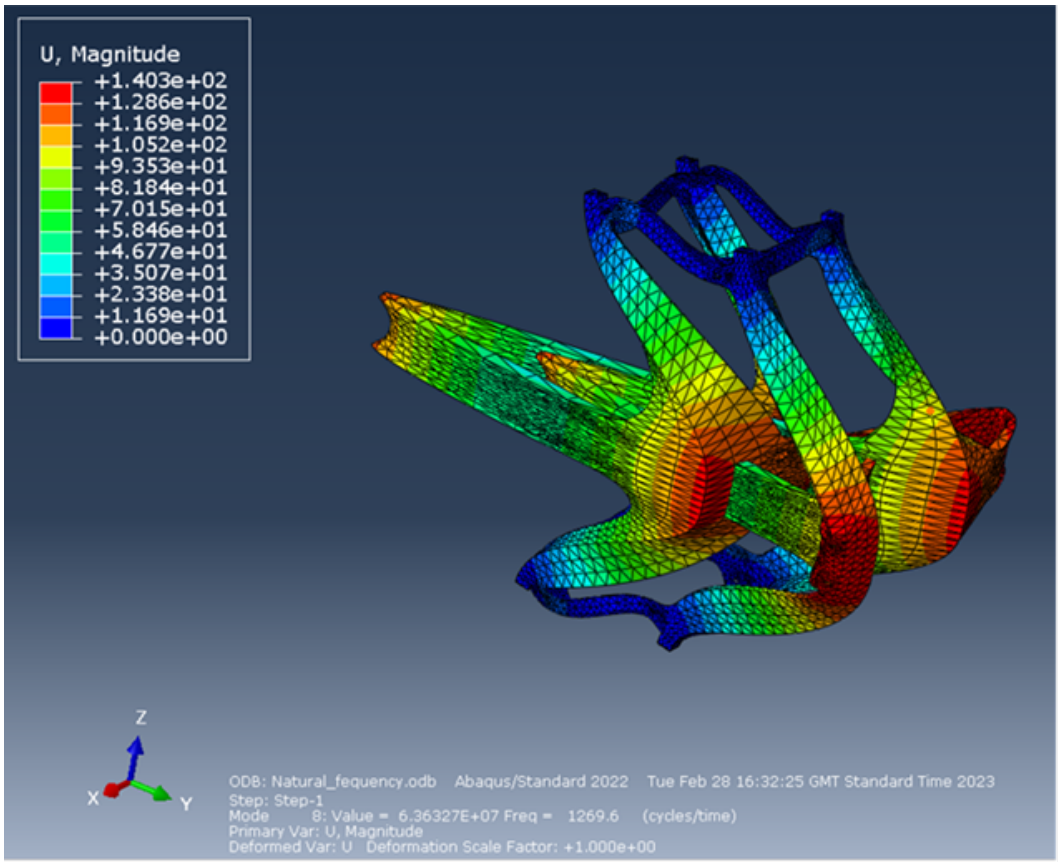


Figure 9.37: Displacement plot from natural frequency analysis of the final design showing the most relevant mode in the Y and Z axis (mode 8)

9.9 Thermal

The environment in space can cause the temperature of internal equipment to fluctuate from as high as 100°C, when in line with the Sun and to as low as -130°C when behind the Earth in an eclipse, during one orbit [26]. To regulate this, thermal analysis is conducted to control these temperatures. Due to the lack of time to learn ESATAN-TMS, an in-depth thermal analysis could not be conducted. Instead, the CubeSat was modelled to be 1 node and a preliminary mathematical thermal analysis was conducted. This method was taken from the 'Space Mission Engineering: The New SMAD' book [31]. To achieve this, the CubeSat was modelled to be a cylinder of height 0.2 m and diameter 0.1 m. For this analysis the two worst-case scenarios are considered: worst-case hot when the CubeSat is in line with the Sun and is dissipating maximum power, and, worst-case cold, where the CubeSat is in an eclipse and minimum power is being dissipated.

Each component within the satellite can operate between a set range of temperatures. Before the thermal analysis can be conducted the thermal range for each component needs to be identified. The most important thermal constraints tend to be the battery and the payload. The heating for the biological organisms will be controlled by the payload, so the only concern is for the electronics of the payload. From the FDSPP datasheet, this was found to be between -55°C and 125°C. The temperature of the battery is operational from -40°C to 80°C, but experiences optimum power at a range of 10 °C to 25 °C [21]. After determining what thermal range is suitable for the satellite the environment around the CubeSat has to be determined. The orbit greatly affects the thermal impacts on the satellite. The orbit altitude being assessed is 600 km and the orbit inclination is 82 degrees. When the CubeSat is in the orbit, the main factors affecting the thermal environment are the direct sunlight, the Earth's albedo and IR energy emitted from Earth's atmosphere [26]. Direct sunlight is the major source of heating for the satellite, due to Earth's elliptical orbit the intensity of solar energy reaching Earth varies. However, for a distance of 1 AU, the mean distance from the Earth and the Sun, the solar constant is equal to 1367 W/m2 [26]. For this simulation, the worst-case scenario is considered, and so an orbit inclination of 82 degrees is considered. Therefore, for the calculations, a lower limit of 1317 W/m2 is considered and a higher limit of 1419 W/m2 is used. The amount of time spent in an eclipse is also taken into account, with the orbit time being around 5700 s and the eclipse time being around 2150 s, the CubeSat spends around 37.7% of its time in eclipse.

Solar radiation is also reflected from Earth back into space, this is known as the albedo. The amount of solar radiation reflected depends on where over the Earth the satellite is, as the albedo is higher over land than water, and is also affected by cloud coverage [31]. For this analysis, an albedo percentage of 34% was chosen. The orbit inclination for the CubeSat is 82 degrees, so giving the minimum and maximum amount of emitted radiation from the albedo to be 218 W/m2 and 244 W/m2 respectively [31]. Both these values were multiplied by 0.34, the albedo percentage, for the value of $Q_{IncidentAlbedo}$.

Sunlight that has not been reflected is absorbed by Earth and then re-emitted as infrared energy, this can vary greatly, with a hotter region emitting more IR than a colder region. To solve this, a maximum and minimum value for the Earth's IR emission is used. For this analysis, the maximum IR emission is 261 W/m2 and the minimum IR emission is 217 W/m2.

It has been assumed that 75% of the surface has been covered in solar panels, which has an absorptivity of 0.85, and that 25% is left for instrument apertures, which is considered to have an absorptivity of 0.9 [31]. The solar panels are considered to not radiate any energy, and so the emissivity of the CubeSat is believed to be 0.9 [31].

The absorbed environmental heat can be calculated for both cold and hot cases.

Cold:

Firstly the solar energy absorbed is calculated. This is done by multiplying the solar constant, the surface area of the CubeSat, the duty cycle and the average absorptivity of the satellite.

$$Q_{Solar} = SA_P[\%SunTime]\alpha_{avg} \tag{9.7}$$

$$Q_{Solar} = 1317W/m^2 \times 0.1m^2 \times 0.38 \times ((0.85 \times 0.75) + (0.9 \times 0.25)) = 43.2W \tag{9.8}$$

Then the energy absorbed from the albedo is measured.

$$Q_{Albedo} = Q_{IncidentAlbedo}A_I R\alpha_{avg} \tag{9.9}$$

$$Q_{Albedo} = 82.2W/m^2 \times 0.0025\pi \times ((0.85 \times 0.75) + (0.9 \times 0.25)) = 0.56W \tag{9.10}$$

Lastly, the minimum amount of Earth's IR radiation absorbed is calculated.

$$Q_{IR} = Q_{IncidentIR}A_{IR}\epsilon_{avg} \tag{9.11}$$

$$Q_{IR} = 217W/m^2 \times 0.0025\pi \times 0.9 = 1.53W \tag{9.12}$$

Hot:

For the hot cases, the same equations are used. However, the values for the solar constant, incident albedo energy and infrared energy are different to account for the worst-case hot scenario.

$$Q_{Solar} = SA_P[\%SunTime]\alpha_{avg} \tag{9.13}$$

$$Q_{Solar} = 1419W/m^2 \times 0.1m^2 \times 0.38 \times ((0.85 \times 0.75) + (0.9 \times 0.25)) = 46.51W \tag{9.14}$$

$$Q_{Albedo} = Q_{IncidentAlbedo}A_I R\alpha_{avg} \tag{9.15}$$

$$Q_{Albedo} = 91.99W/m^2 \times 0.0025\pi \times ((0.85 \times 0.75) + (0.9 \times 0.25)) = 0.62W \tag{9.16}$$

$$Q_{IR} = Q_{IncidentIR}A_{IR}\epsilon_{avg} \tag{9.17}$$

$$Q_{IR} = 261W/m^2 \times 0.0025\pi \times 0.9 = 1.845W \tag{9.18}$$

After the environmental energy has been calculated for both scenarios, the minimum internal power dissipated, with no battery input, is added to the cold case. The maximum power dissipated, with maximum battery input, is also added to the hot case to generate both worst-case scenarios.

$$TotalQ_{in}cold = 0.55W + 45.29W = 45.84W \tag{9.19}$$

$$TotalQ_{in}hot = 9.31W + 48.94W = 58.25W \tag{9.20}$$

The energy being released from the CubeSat, Qout, is calculated:

$$Q_{Out} = \pi DL\epsilon\sigma T^4 \tag{9.21}$$

$$Q_{Out} = pi \times 0.1 \times 0.2 \times 0.9 \times \sigma \times T^4 = 3.206 \times 10^{-9}T^4 \tag{9.22}$$

The energy balance equation is conducted to calculate the temperature:

$$Q_{env} + Q_{in} = Q_{out} \tag{9.23}$$

$$T_{cold} = 345.8K = 72.8^oC$$
$$T_{hot} = 367.1K = 94.1^oC$$

From the temperatures achieved, it can be noted that they are higher than the ideal thermal requirements of the CubeSat. Therefore, heat has to be radiated from the satellite. However, due to the solar panels being body mounted, it is not possible to place a radiator on the external surface, unless they are deployed once the spacecraft is in orbit or solar panels are removed to accommodate radiators. Radiators are surfaces placed on the outer surface of a CubeSat which has a high emissivity and low absorptivity, this allows for heat to be released.

The analysis conducted considered the surface of the CubeSat to have the absorptivity of the solar panels. However, this energy is not transferred into the CubeSat as heat but rather energy so the temperature may not be as high as it is believed to be. Even if the solar panels generate heat along the back surface, an insulating material can be placed behind the solar panel preventing the heat of the solar panel from affecting the inside components. If the analysis conducted has led to higher temperatures than they truly are in reality then different options can be used to regulate the temperatures within the CubeSat. Multi-Layer Insulation can be used around the CubeSat or around components to maintain temperatures within. Another option could be to use heaters. However, this is an active solution rather than a passive one, so would require a great intake of power to run and so are not the preferred option.

The analysis conducted is only preliminary for the first iteration of the design. More in-depth analysis is required using computational software such as ESATAN-TMS, which can give a more accurate picture of the temperature of the CubeSat as rather than the entire satellite being modelled as one node each face can be analysed as some will be facing the Sun whilst others are facing the Earth. Moreover, the understanding of how each component is interacting thermally can be analysed. As this is the first year of this project this could not be done due to a limited amount of time to learn and implement the software and the components being used not being decided till near the end of the project deadline.

9.10 Configuration Model [V]

With all subsystems designed, a configuration model of the satellite was produced by assembling all parts of the chassis, subsystem components, and the FDSPP payload using Autodesk Fusion 360. The preliminary configuration of the satellite is shown in the exploded diagram in 9.38 below.

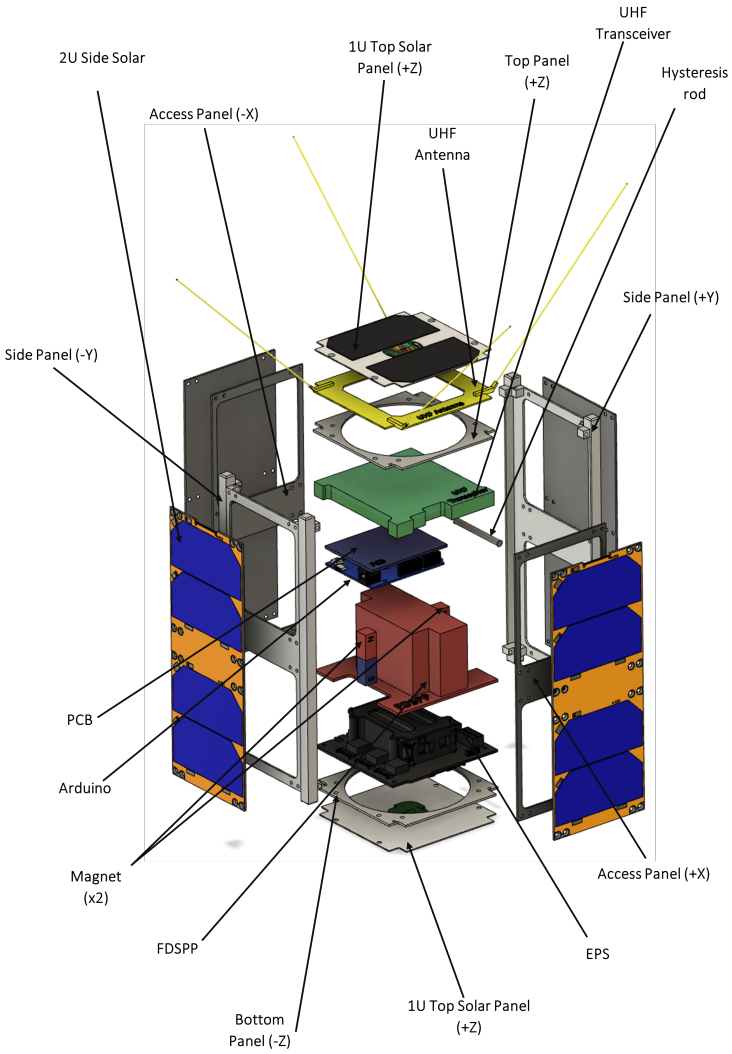


Figure 9.38: Exploded diagram of the CubeSat Assembly

9.39 shows how all the internal components fit into the CubeSat chassis. Solar panels and the front access panel are hidden to allow for better visualisation of the internal space. As seen in the Figure, not all spaces inside the satellite have been utilised. This indicates that there is room to further optimise the satellite design, such as considering the possibility of adapting a 1.5U configuration instead of a 2U. 9.40 illustrates the fully assembled satellite with the UHF antenna in the deployed position.

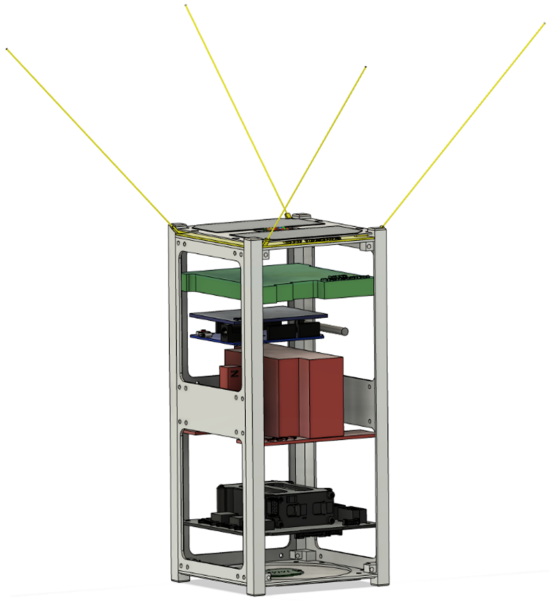


Figure 9.39: Fully Assembled CubeSat showing internal components

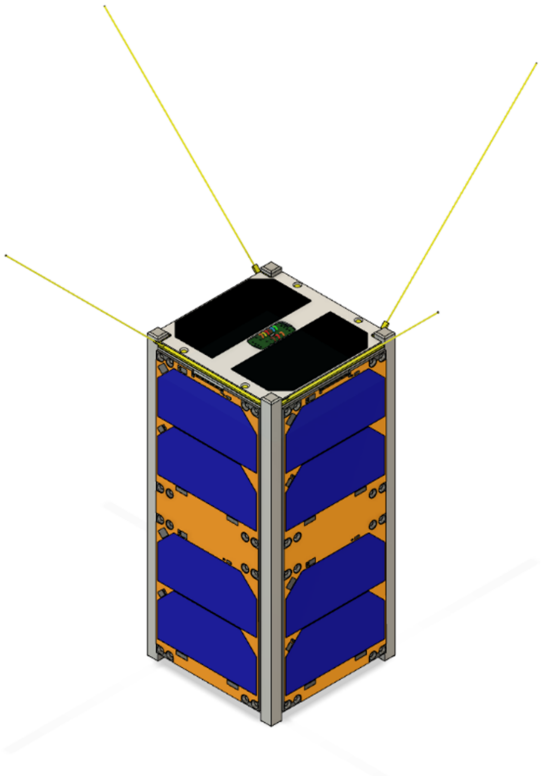


Figure 9.40: Fully Assembled CubeSat

10 Failure Mode Effects Analysis (FMEA)

Having designed the first iteration of a satellite a critical analysis of failure must be done, in order to better set requirements for the next iteration. Additionally, this can often bring to light any overlooked issues with subsystems, allowing them to be fixed before a design is finalised.

A Failure Mode Effects Analysis (FMEA) was done on the system, with the results summarised in "Design Portfolio Submission - 9.Subsystems - 9.6 Systems Engineering - FMEA - FMEA.xlsx". A subsection of the results shown there will be discussed here.

10.1 Failure Mode ID:8

The temperature in low earth orbit fluctuates wildly, if there were to be an unexpected temperature distribution the thermal subsystem heater may cause a too high temperature to be present in the satellite. This temperature would cause the biological experiment to die, thus preventing the payload from taking useful images. Considering the moderate uncertainty in the results of the Thermal Subsystem section, 9.9 there is a larger probability of this occurring. In the event that this did occur, it would be critically severe to the mission, as the payload would not be able to continue normal operations.

Using this a moderate risk was assigned to this failure mode. A suggestion of more precise orbital simulations and ESATAN modelling is suggested to reduce the probability of failure, thus reducing the associated risk.

10.2 Failure Mode ID:29

The exact orbit of the satellite remains unknown until communication can be established, and attitude data is gained. In this time before connection, an unexpected orbit could occur, due to deviations in rocket path, ejection velocity from the deployer or unaccounted-for effects of orbit decay. If the orbit were to change, there may be no way for a set ground station to contact a satellite. In this case, no telemetry or telecommands would be able to be received or sent to the satellite, preventing any knowledge gain from the mission.

These present a high risk to the satellite; thus, they must be accounted for. A suggestion of submitting a request to the North American Aerospace Defence Command (NORAD), once the launch date is set, for general satellite tracking would allow the orbit of the satellite to be tracked. Using this tracking, a different ground station may be able to be used to send and receive data, thus allowing the mission to continue.

10.3 Failure Mode ID:42

At the end of life of the satellite, a 'Kill Worms' command will be sent, from the ground station, this will terminate the biological experiment. If the command were to fail, due to an intrinsic failure, the payload would not be able to gain the images about the event, though no other systems will be affected. Additionally, there would not be a breach of space debris, as the biological experiment will burn up on re-entry.

Considering this a low risk is posed to the system. Nevertheless, to further ensure that this does not occur adequate random vibration testing should be performed on the payload, to confirm its robustness and present evidence as to why this is very unlikely to happen.

10.4 Conclusion

These are a few of the failure modes analysed, though there are many more within the associated excel file. The list of failure modes is not exhaustive, as not every facet of the design is known at this point, though illustrates the failures that should be considered in the design.

For future teams continuing the design of this mission, it is suggested that the list be expanded and repeatedly updated in order to keep up to date with the current design.

11 Conclusion

Overall the design of this CubeSat was an extremely good first step for the first year of a multiyear project covering; requirements from ESA to industrial partners; communication systems; power systems; mechanical design and OBC connections. A configuration model was produced to show the design of a 2U CubeSat with a functioning payload that should be able to communicate with a ground station.

The next stages of the project should have a good foundation to be building and purchasing physical parts to produce an engineering model for physical verification. They may specialize in developing more detailed requirements documents, as there is still considerable additional headroom in terms of power, link, and space budgets. Clear steps for moving the project forward have been set out for each subsystem as well as future failure modes that need to be considered as time continues.

As a conclusion, in this design portfolio report, a functionalized 2U CubeSat with a specific payload was designed for operation in low Earth orbit.

References

[1] J. Yiu, *The Definitive Guide to the ARM Cortex-M3, Second Edition*. Newnes, Oct. 2010.

[2] iCircuit, "Arduino boards pin mapping." `https://icircuit.net/arduino-boards-pin-mapping/141`, 2016. Accessed: March 7, 2023.

[3] Various, "Rca cdp1802 cosmac processor pinout.png." `https://en.wikipedia.org/wiki/File:RCA_CDP1802_COSMAC_processor_pinout.png`, 2013. Accessed: March 7, 2023.

[4] J. Wolff, "Reading and writing data to external eeprom using arduino." `https://www.instructables.com/Reading-and-Writing-Data-to-External-EEPROM-Using-/`, 2013. Accessed: March 7, 2023.

[5] S. Loff, "Cubesats overview," 2018.

[6] C. Poly, "Cubesat design specification." `https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779/CDS+REV14_1+2022-02-09.pdf`, 2022. Accessed: March 7, 2023.

[7] E. Howell, "Cubesats: Tiny payloads, huge benefits for space research," 2021.

[8] C. Poly, "Nanoracks cubesat deployer interface definition document." `https://nanoracks.com/wp-content/uploads/Nanoracks-CubeSat-Deployer-NRCSD-IDD.pdf`, 2022. Accessed: March 7, 2023.

[9] M. Swartwout, "The First One Hundred CubeSats: A Statistical Look," *Journal of Small Satellites*, vol. 2, pp. 213–233, Dec. 2013.

[10] J. Wolff, "Reading and writing data to external eeprom using arduino," 2013. Accessed: March 7, 2023.

[11] R. Pi, "Raspberry pi camera module 3." `https://thepihut.com/products/raspberry-pi-camera-module-3`, 2023. Accessed: March 7, 2023.

[12] Y. Shiyou, A. Ali, M. Tahir, S. Fahad, S. Rao, *et al.*, "Cubesats detumbling using only embedded asymmetric magnetorquers," *Advances in Space Research*, vol. 71, no. 5, pp. 2140–2154, 2023.

[13] T. T. Ramodimo, K. Mmopelwa, B. Basutli, and O. Matsebe, "Magnetic control for detumbling of satellites: A tutorial," in *2022 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–6, IEEE, 2022.

[14] A. M. Hassan and A. A. El-Badawy, "Novel omnimagnet actuation method for a cubesat nano-satellite," *Aerospace Science and Technology*, vol. 117, p. 106913, 2021.

[15] N. N. Shusharina, D. A. Borchevkin, V. V. Sapunov, V. A. Petrov, and M. V. Patrushev, "A wireless portable platform for physiological potentials registration and processing: A unified approach," *Journal of Pharmaceutical Sciences and Research*, vol. 9, no. 7, p. 1178, 2017.

[16] J. Lohfink, *RCA COSMAC microprocessor family: CMOS 1802/1805*. Newnes, 2000.

[17] D. Phillips and K. Parr, "The i2c bus: From theory to practice," *John Wiley & Sons*, vol. 1, no. 2, pp. 38–44, 2000.

[18] R. Chin, *Arduino and Raspberry Pi Sensor Projects for the Evil Genius*. McGraw-Hill Education, 2018.

[19] M. Yaqoob, A. Lashab, J. Vasquez, J. Guerrero, M. Orchard, and A. Bintoudi, "A comprehensive review on small satellite microgrids," *IEEE Transactions on Power Electronics*, vol. 37, p. 5, 2022.

[20] A. Edpuganti, V. Khadkikar, H. Zeineldin, M. Moursi, and M. Al Hosani, "Comparison of peak power tracking based electric power system architectures for cubesats," *IEEE Transactions on Industry Applications*, vol. PP, pp. 8–10, 2021.

[21] I. Group, "Iceps2 datasheet,"

[22] A. C. Space, "Pulsar-vutrx." `https://www.aac-clyde.space/what-we-do/space-products-components/communications/pulsar-vutrx`, 2023. Accessed: March 7, 2023.

[23] EnduroSat, "Uhf transceiver ii." `https://www.endurosat.com/cubesat-store/cubesat-communication-modules/uhf-transceiver-ii/`, 2023. Accessed: March 7, 2023.

[24] EnduroSat, "S-band transmitter." `https://www.endurosat.com/cubesat-store/cubesat-communication-modules/s-band-transmitter/`, 2023. Accessed: March 7, 2023.

[25] M. Macdonald and V. Badescu, *The international handbook of space technology*. Springer, 2014.

[26] W. J. Larson, J. R. Wertz, *et al.*, *Space mission analysis and design*, vol. 3. Springer, 1992.

[27] M. Macdonald and V. Badescu, *The international handbook of space technology*. Springer, 2014.

[28] M. Macdonald and V. Badescu, *The international handbook of space technology*. Springer, 2014.

[29] J. J. Wijker, *Spacecraft structures*. Springer Science & Business Media, 2008.

[30] C. Järmyr Eriksson, "Finite element analysis of stresses in the mist cubesat due to dynamic loads during launch," 2021.

[31] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space mission engineering: the new SMAD*. Microcosm Press, 2018.

Appendices

A Testing code 1

```
// nrf24_client

#include <SPI.h>
#include <RH_NRF24.h>

RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin

long double starttime = millis();
void setup()
{
  // standard comms speed
  Serial.begin(9600);
  //check the nrf24 is connected correctly
  while(!Serial);
    delay(1000); //waiting for board
  if (!nrf24.init())
    Serial.println("init failed");
  // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
  if (!nrf24.setChannel(1))
    Serial.println("setChannel failed");
  if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::
     TransmitPower0dBm))
    Serial.println("setRF failed");
}


void loop()
{
  Serial.println("Sending to ground station");
  uint8_t data[] = "Are you there?";
  //send message of nrf24, needs to be uint_8 format and within
     maximum message
  nrf24.send(data, sizeof(data));
  //fills buffer of nrf24 and clocks when sent so function
     ensures message was sent before proceeding
  nrf24.waitPacketSent();
  // Now wait for a reply
  uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

  if (nrf24.waitAvailableTimeout(900))
  {
    // Should be a reply message for us now
    //if message us within maximum buffer size and has been
       received then reply else is a failure or times out
    if (nrf24.recv(buf, &len))
    {
```

```
      Serial.print("Reply: ");
      Serial.println((char*)buf);
    }
    else
    {
      Serial.println("No message receive");
    }

  }
  else
  {
    Serial.println("We appear to have lost the satellite");
  }
  delay(1000);
}
```

B Testing code 2

```
#include <SPI.h>
#include <RH_NRF24.h>

long double starttime = millis();
RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin

void setup()
{
  //standard communication speed for board
  Serial.begin(9600);
  while (!Serial);
     delay(1000); // wait for serial port to connect. Needed for
        Leonardo only
  if (!nrf24.init())
    Serial.println("init failed");
  // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
  if (!nrf24.setChannel(1))
    Serial.println("setChannel failed");
  if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::
     TransmitPower0dBm))
    Serial.println("setRF failed");
}

void loop()
{
  if (nrf24.available())
  {
    // Should be a message for us now
    uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (nrf24.recv(buf, &len)&& starttime<5000)
    {
//      NRF24::printBuffer("request: ", buf, len);
      Serial.print("got request: ");
      Serial.println((char*)buf);
      // Send a reply
      uint8_t data[] = "Too early!";
      nrf24.send(data, sizeof(data));
      nrf24.waitPacketSent();
      Serial.println("Sent a reply");
    }
    else if((nrf24.recv(buf, &len)&& starttime=>5000)){
    {
//      NRF24::printBuffer("request: ", buf, len);
      Serial.print("got request: ");
       Serial.println((char*)buf);

       // Send a reply
       uint8_t data[] = "Howdy";
```

```
        nrf24.send(data, sizeof(data));
         nrf24.waitPacketSent();
        Serial.println("Sent a reply");
    }
    else
    {
      Serial.println("receive failed");
    }
  }
}
}
```

C Testing code 3

```
#include <SPI.h>
#include <RH_NRF24.h>
#include <Wire.h>
#define eeprom 0x50 //defines the base address of the EEPROM this
    will
                    //change depending on how the arduino is
                       wired!!!!
long double starttime = millis();
RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin


void setup()
{
  Wire.begin(); //creates a Wire object
  Serial.begin(9600);//standard communication speed for board
  while (!Serial);
     delay(1000); // wait for serial port to connect. Needed for
        Leonardo only
  if (!nrf24.init())
    Serial.println("init failed");
  // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
  if (!nrf24.setChannel(1))
    Serial.println("setChannel failed");
  if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::
     TransmitPower0dBm))
    Serial.println("setRF failed");
}


//function to write data to EEPROM
void writeTOstorage(int adress, unsigned int promadress, byte
   data){
  Wire.BeginTransmission(adress);
  Wire.write((int) (promadress>>8)); //writes first 8 bits of
     data as most important data
  Wire.write((int)(promadress & 0xFF)); //writes last bits of
     data to register
  Wire.write(data);
  Wire.endTransmission();
}


//function to read from data sotred in EEPROMS which will return
   a byte of data
byte readFROMstorage(int adress, unsinged int promadress){
  byte datafill = 0xFF;
  Wire.beginTransmission(adress);
  Wire.write((int) promadress >> 8 ));  //write msb
```

```
  Wire.write((int) (promadress & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}


void loop()
{
  if (nrf24.available())
  {
    // Should be a message for us now
    uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (nrf24.recv(buf, &len)&& starttime<5000)
    {
//      NRF24::printBuffer("request: ", buf, len);
      Serial.print("got request: ");
      Serial.println((char*)buf);

      // Send a reply
      uint8_t data[] = "Too early!";
      nrf24.send(data, sizeof(data));
      nrf24.waitPacketSent();
      Serial.println("Sent a reply");
    }
    else if((nrf24.recv(buf, &len)&& starttime=>5000))
    {
//      NRF24::printBuffer("request: ", buf, len);
      Serial.print("got request: ");

      if((char*)buf[0])=="1"){
        string data = string((char*)buf[1]);
        int data = toint(data);
        for(adress = 0;adress = data-1;adress++){
          uint8_t data[] = String(readFromstorage(eeprom,adress))
              ;
          nrf24.send(data, sizeof(data));
          nrf24.waitPacketSent();
        }

      }
      else if((char*)buf[0])=="0"){
        for(adress = 0;adress = (sizeof((char*)buf)-1);adress++){
          writeTOstorage(eeprom,adress,(char*)buf[adress+1]);
          Serial.print("writing data");
        }
```

```
      }
      else{
        Serial.print("invalid instruction")
      }
    }
    else
    {
      Serial.println("recv failed");
    }
  }
}
```

D Testing code 4

```
#include <SPI.h>
#include <RH_NRF24.h>
#include <Wire.h>
#define eeprom 0x50 //defines the base address of the EEPROM this
    will
                    //change depending on how the arduino is
                       wired!!!!
long double starttime = millis();
RH_NRF24 nrf24(8, 10);// For Leonardo, need explicit SS pin


void setup()
{

  Wire.begin(); //creates a Wire object
  Serial.begin(9600);//standard communication speed for board
  while (!Serial);
     delay(1000); // wait for serial port to connect. Needed for
        Leonardo only
  if (!nrf24.init())
    Serial.println("init failed");
  // Defaults after init are 2.402 GHz (channel 2), 2Mbps, 0dBm
  if (!nrf24.setChannel(1))
    Serial.println("setChannel failed");
  if (!nrf24.setRF(RH_NRF24::DataRate2Mbps, RH_NRF24::
     TransmitPower0dBm))
    Serial.println("setRF failed");
}
//function to write data to EEPROM
void writeTOstorage(int adress, unsigned int promadress, byte
   data){
  Wire.BeginTransmission(adress);
  Wire.write((int) (promadress>>8)); //writes first 8 bits of
     data as most important data
  Wire.write((int)(promadress & 0xFF)); //writes last bits of
     data to register
  Wire.write(data);
  Wire.endTransmission();
}
//function to read from data sotred in EEPROMS which will return
   a byte of data
byte readFROMstorage(int adress, unsinged int promadress){
  byte datafill = 0xFF;
  Wire.beginTransmission(adress);
  Wire.write((int) promadress >> 8 ));  //write msb
```

```
  Wire.write((int) (promadress & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
void loop()
{
  Serial.println("Sending to nrf24_server");
   uint8_t data[] = "131415";
   nrf24.send(data, sizeof(data));
   nrf24.waitPacketSent();
   uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];
   uint8_t len = sizeof(buf);
   delay(100);
   uint8_t data[] = "04";
   nrf24.send(data, sizeof(data));
   nrf24.waitPacketSent();
   int counter = 0;
   int ender = 0;
   if(nrf25.available()){
  while(counter < 5){
  if (nrf24.recv(buf, &len))
    {
      Serial.println("Reply: ");
      Serial.println((char*)buf);
      counter += 1;
    }
    else
    {
      Serial.println("No message receive");
      ender +=1
    }
    if(ender = 100){
      Serial.println("This has failed satellite lost");
      break;
    }
  }
   }
  else
  {
    Serial.println("We appear to have lost the ground station");
  }
  delay(1000);
}
```

E Satellite code

```
/*
*/
#include \<SPI.h\>
#include \<Wire.h\>
#define eeprom 0x50 // tbd  definitions such as this will need to
    be placed for all units such as imu, power sensor, stroage,
   communication unit and fdspp
#define imu          //tbd
#define power          //tbd
#define fdspp          //tbd
#define comms           //tbd

double datapacket
long double transmittime;
long double orbittime;
int sleeptime
void setup() {
  Serial.begin(9600);
  Wire.begin();
  while (!Serial){}; // wait for serial port to connect. Needed
     for Leonardo only
    SignalTransmit();
    rangefind();
    sleeptime = (orbittime-transmittime - 150000/1000)
}

void loop() {
  long double starttime = millis();
  normaltransmit();
  while(millis()-starttime<transmittime){
    if (Transmittingdata() == 1){
      Transmitdata();
    }
   else {
     while(millis()-starttime<transmittime){
     if(readFromcomms()!='\0'){
       instruction(readFromcomms());
       break
     }
   }
  }
  }
  for(int i = 0; i<= sleeptime; i++){
    LowPower.idle(SLEEP_1S, ADC_OFF, TIMER2_OFF, TIMER1_OFF,
       TIMER0_OFF,SPI_OFF, USART0_OFF, TWI_OFF);
  }
  while(millis()-starttime<orbittime){

  }
```

```
}

void Transmitdata(){
  for (j=0,j=127,j++){
      byte datapacket = readFromstorage(j)
  for (i=0,i=len(datapacket),i++){

    if (datapacket[i] != '\0'){
        writeTocomms(datapacket[i]);
        datapacket[i] = '\0';

    }
  }
  }
}
void instruction(string message){
  //there is no point over defining things here as too many
     unknowns
  //this will be a list of if functions to say what will do what
     when depending
  //on incoming signal
  //thoughts could be to check first byte of data as a 0-9 to
     state which instruction path
  //similar to testing rig but as of yet undefined.
  //good place to start for next year!

  if (message == ''){
    //whatever is wanted by
  }
  else if {message ==''}
  ...

}
bool Transmittingdata(){
  if (datapacket !=0){
    return 0;
  }
    else{
    return 1;
  }
}
void writeTocomms(string data){
  Wire.beginTransmission(comms);//whatever location of comms is
  Wire.write((int) comms >> 8 ));  //write msb
  Wire.write((int) (comms & 0xFF)); //writes lsb
  Wire.write(data);
  Wire.endTransmission;
}
void writeTofdspp(string data){
  Wire.beginTransmission(fdspp);//whatever location of fdspp is
  Wire.write((int) fdspp >> 8 ));  //write msb
```

```
  Wire.write((int) (fdspp & 0xFF)); //writes lsb
  Wire.write(data);
  Wire.endTransmission;
}
void writeTostorage(string data){
  Wire.beginTransmission(eeprom);//whatever location of storage
     is
  Wire.write((int) storage >> 8 ));  //write msb
  Wire.write((int) (storage & 0xFF)); //writes lsb
  Wire.write(data);
  Wire.endTransmission;
}
byte readFromstorage(int adress){
  byte datafill = 0xFF;
  Wire.beginTransmission(eeprom);
  Wire.write((int) storage >> 8 ));  //write msb
  Wire.write((int) (storage & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
byte readFromfdspp(int adress){
  Wire.beginTransmission(fdspp);//whatever location of fdspp is
  byte datafill = 0xFF;
  Wire.write((int) fdspp >> 8 ));  //write msb
  Wire.write((int) (fdspp & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
byte readFromimu(int adress){
  Wire.beginTransmission(imu);//whatever location of imu is
  byte datafill = 0xFF;
  Wire.write((int) imu >> 8 ));  //write msb
  Wire.write((int) (imu & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
byte readFrompower(int adress){
  Wire.beginTransmission(power);//whatever location of power is
  byte datafill = 0xFF;
```

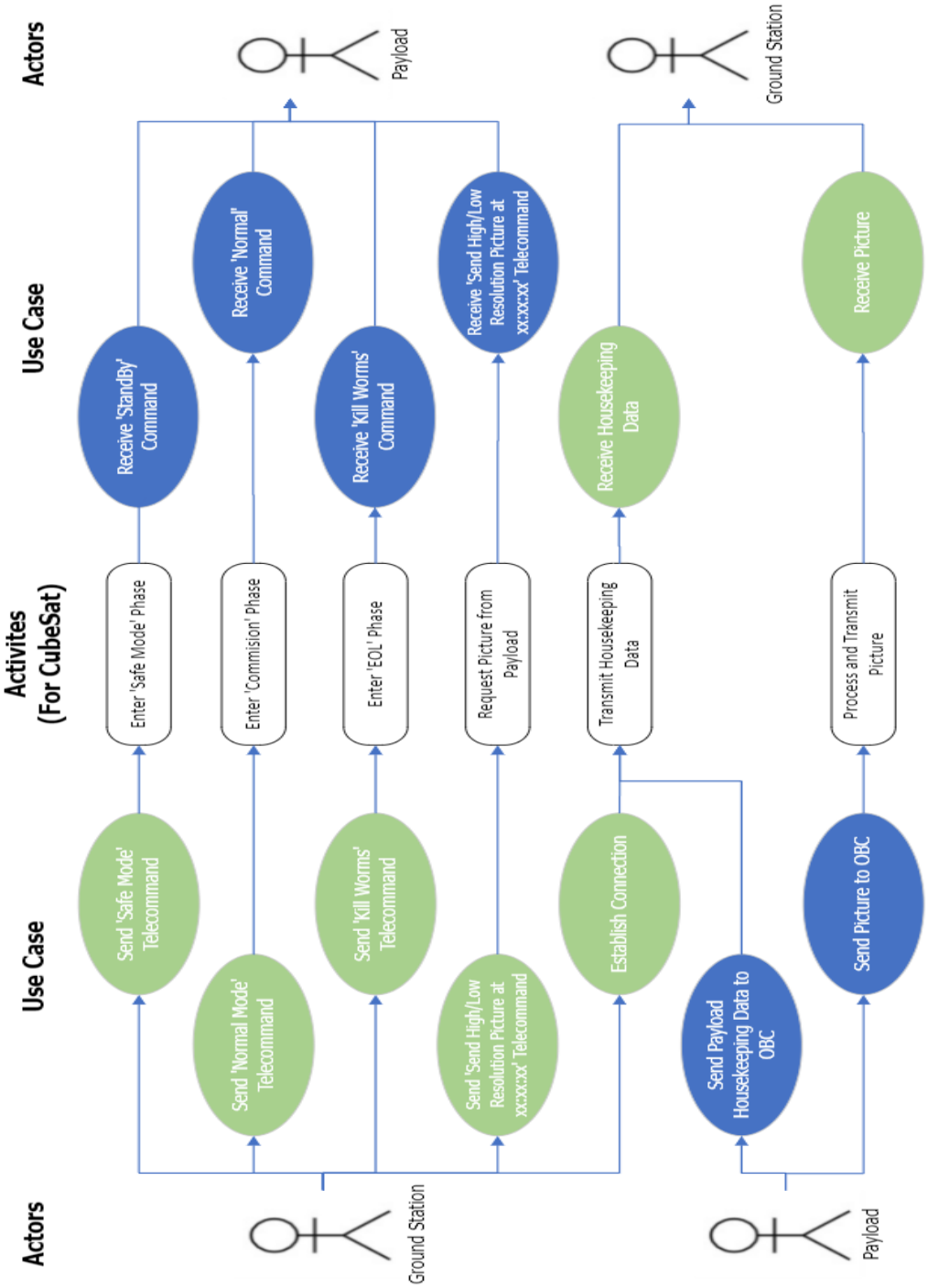
```
  Wire.write((int) power >> 8 ));  //write msb
  Wire.write((int) (power & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
byte readFromcomms(int adress){
  Wire.beginTransmission(comms);//whatever location of comms is
  byte datafill = 0xFF;
  Wire.write((int) comms >> 8 ));  //write msb
  Wire.write((int) (comms & 0xFF)); //writes lsb
  Wire.endTransmission();
  Wire.requestFrom(adress,1);
  if (Wire.available()){
    datafill = Wire.read();
    return datafill;
  }
}
void normaltransmit(){
    Wire.writeTocomms("Hello World!");
}
void SignalTransmit(){
  for (i = 1:37){
    normaltransmit();
    delay(150000);
  }
}

  void rangefind(){
  int i = 1;
  while {

    if (readFromcomms()!='\0'){
      long double Time = millis();
      break
  } else{
      for(int i = 0; i<= 20; i++){
        LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF,
           TIMER0_OFF,SPI_OFF, USART0_OFF, TWI_OFF);
        }
  }
  }
  while{
    if (readFromcomms()!='\0'){

  } else{
        long double stopTime = millis();
        if(i==20){
```

```
        break}
        else{
          i +=1;
        }
  }
  }
  for(int i = 0; i<= 300; i++){
        LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF,
           TIMER0_OFF,SPI_OFF, USART0_OFF, TWI_OFF);
        }
  while {

    if (readFromcomms()!='\0'){
      long double restartTime = millis();
      break
  } else{
      for(int i = 0; i<= 20; i++){
        LowPower.idle(SLEEP_8S, ADC_OFF, TIMER2_OFF, TIMER1_OFF,
           TIMER0_OFF,SPI_OFF, USART0_OFF, TWI_OFF);
        }
  }
  }
  while{
    if (readFromcomms()!='\0'){

  } else{
      for(int i = 0; i<= 20; i++){
        long double restopTime = millis();
        break
        }
  }
  }
  orbittime = restopTime-stopTime;
  transmittime = restopTime-startTime;
}
```

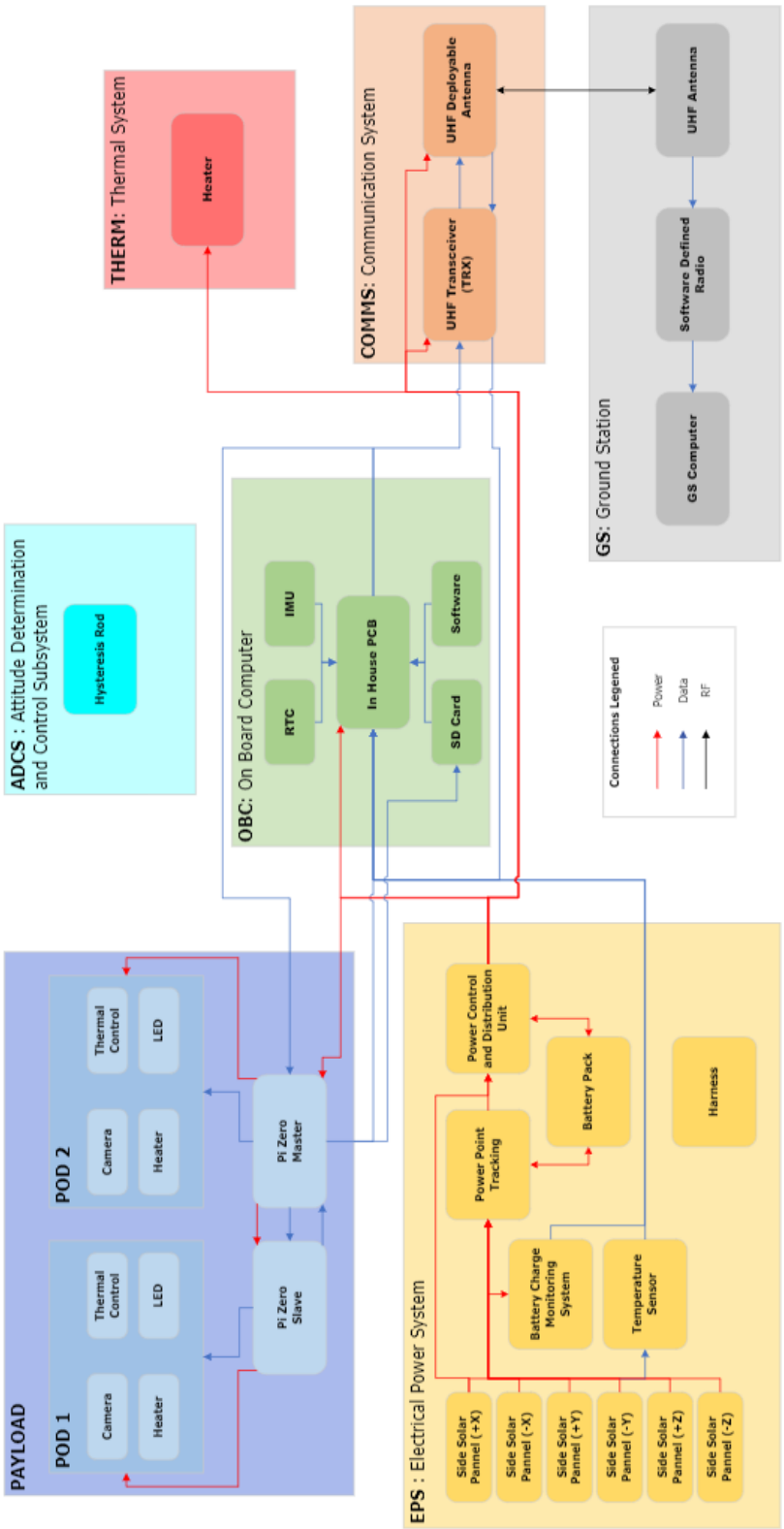
F CONOPS


Modes
Launch Phase
Powered Off
Early Operations Phase
Engage Solar Panels
Deploy Antenna
Engage Processor
Engage Heating Elements
Passive Hysteresis Detumbling
Establish Ground Segment Connection
Commisioning Phase
Comission Payload
Send Housekeeping Data
Processing Phase
Receive Data From Payload
Package Data From Payload
Transmit Phase
Send Packaged Photos
Send Housekeeping Data
Receive Phase
Receive Telecommands
EOL Phase
Deplete Battery
Safe Mode
Power Off Payload
Send Housekeeping Data
Receive Telecommands

G User Diagram



H System Level Architecture



I Matlab code for Trajector Analysis

```
startTime = datetime(2023,3,1,12,0,0);      %Sets start time for
   simulation
stopTime = datetime(2023,6,1,12,0,0);       %Sets stop time for
   simulation
sampleTime = 60; % seconds
sc = satelliteScenario(startTime,stopTime,sampleTime);  %Creates
   a scenario object to be edited

%Creates satellite parameters
semimajoraxis = 6978.14e3;
eccentricity = 1.8731e-16;
inclination = 82;
RAAN = 310;
argofperiapsis = 0;
trueanomaly = 0;
sat = satellite(sc,semimajoraxis,eccentricity,inclination, ...
    RAAN,argofperiapsis,trueanomaly,"OrbitPropagator","two-body-
       keplerian", ...
    "Name","WUSAT");

%Creates antenna parameters
names = sat.Name + " TX";   %Sets name for satellite transmitter
TX_Antenna = conicalSensor(sat, ...      %Assigns transmitter to
   satellite
    "Name",names,"MaxViewAngle",157.25); %Assigns viewing angle


%Creates ground station parameters
name = "Uni Of Warwick";
minElevationAngle = 0; % Sets min angle for ground station acess
geoSite = groundStation(sc, ...
    "Name",name, ...
    "MinElevationAngle",minElevationAngle, ...
    "Latitude",52.38262, ...
    "Longitude",-1.56138, ...
    "Altitude",930);

ac = access(TX_Antenna,geoSite);    %Creates access monitoring
   object

%Runs simulation of satellite orbit
v = satelliteScenarioViewer(sc,"ShowDetails",true,"Dimension","2D
   ");

%Shows labels of ground station and satellite
sat.ShowLabel = true;
geoSite.ShowLabel = true;
show(sat);
```

```
fov = fieldOfView(TX_Antenna); %Displays viewing angle of
   satellite
ac.LineColor = 'red';

acInt = accessIntervals(ac);    %Stores access durations
save acInt.mat acInt            %Saves access durations
%%

MeanDur = mean(acInt.Duration)*0.8; %Calculates mean duration,
   with margin
MinDur = min(acInt.Duration);       %Caluclates min duration

TotDur = sum(acInt.Duration)*0.8;   %Calculate total duration
dataRate = 19.6e3;                  %bps
lowRes = 256*256;
hihRes = 12e6;

TotLow = TotDur*dataRate/lowRes;
TotHih = TotDur*dataRate/hihRes;

MeaLow = MeanDur*dataRate/lowRes;
MeaHih = MeanDur*dataRate/hihRes;

%%
figure; hold on;
histogram(acInt.Duration,30)
ylabel("Frequency")
xlabel("Time (s)")
title("Distribution of Access Times")

%%

%Finds longest time with no access
[s,time] = accessStatus(acInt); %Finds access status logical
   array
zeros_idx = find(s==1);         %Finds indexes of connected
   points
diffs= diff(zeros_idx);         %Finds difference between each
   point
[~,diff_idx]=max(diffs(diffs>=0));  %Finds maximum difference
s_diff_idx = [zeros_idx(diff_idx),zeros_idx(diff_idx+1)];   %
   Finds index in s array
NoConnDur = time(s_diff_idx(2))-time(s_diff_idx(1));        %
   Calculates time difference
```

J Communication Tradeoff Parameters

Type	Brand	Name	Min Freq	Max Freq	Output Power	Input Power	Efficiency	Data Rate	Mass (g)	Dimensions (mm)	Dimensions (mm)	Dimensions (mm)
Transceiver	AAC Clyde Space	TRX-U	390	450	5	30.9	70	19200	140	83	57	16
	Endurosat	UHF-II	430	440	1	15.2	70	19600	90.2	89	95	11
	Nano Avionics	SatCOM UHF	395	440	3	20	70	9600	7.5	56.1	32.9	6.56

Table J.1: Transceiver Tradeoff Parameters

Type	Brand	Efficiency	Min Freq	Max Freq	Peak Gain
Ground Station	CS3	60	400	403	23.8
	Alen Space	60	435	438	1
	Dhruva Space	60	395	470	16.3
	ISIS Space	60	435	438	15.5

Table J.2: Ground Station Tradeoff Parameters

Type	Brand	Name	Min Freq	Max Freq	Max Output Power	Input Power	Efficiency	Gain (dB)	HalfPower Bandwidth (deg)	Link Margin (dB) (With	Mass (g)	Dimensions (mm)	Dimensions (mm)	Dimensions (mm)
Antenna	ISISpace	CubeSat Antenna	420	450	2	0.04	70	2.15	143.75	18.04	100	98	98	7
	Endurosat	Antenna 2U	435	438	2	0	70	2.15	143.75	18.04	210	198	98	12.1
	Nano Avionics	UHF 1U Antenna	400	500	10	0	70	1.37	157.25	17.26	33	99.6	99.6	7.7
	Nano Avionics	UHF 2U Antenna	400	500	10	0	70	2.86	132.46	18.75	50	222	99.6	7.7

Table J.3: Antenna Tradeoff Parameters

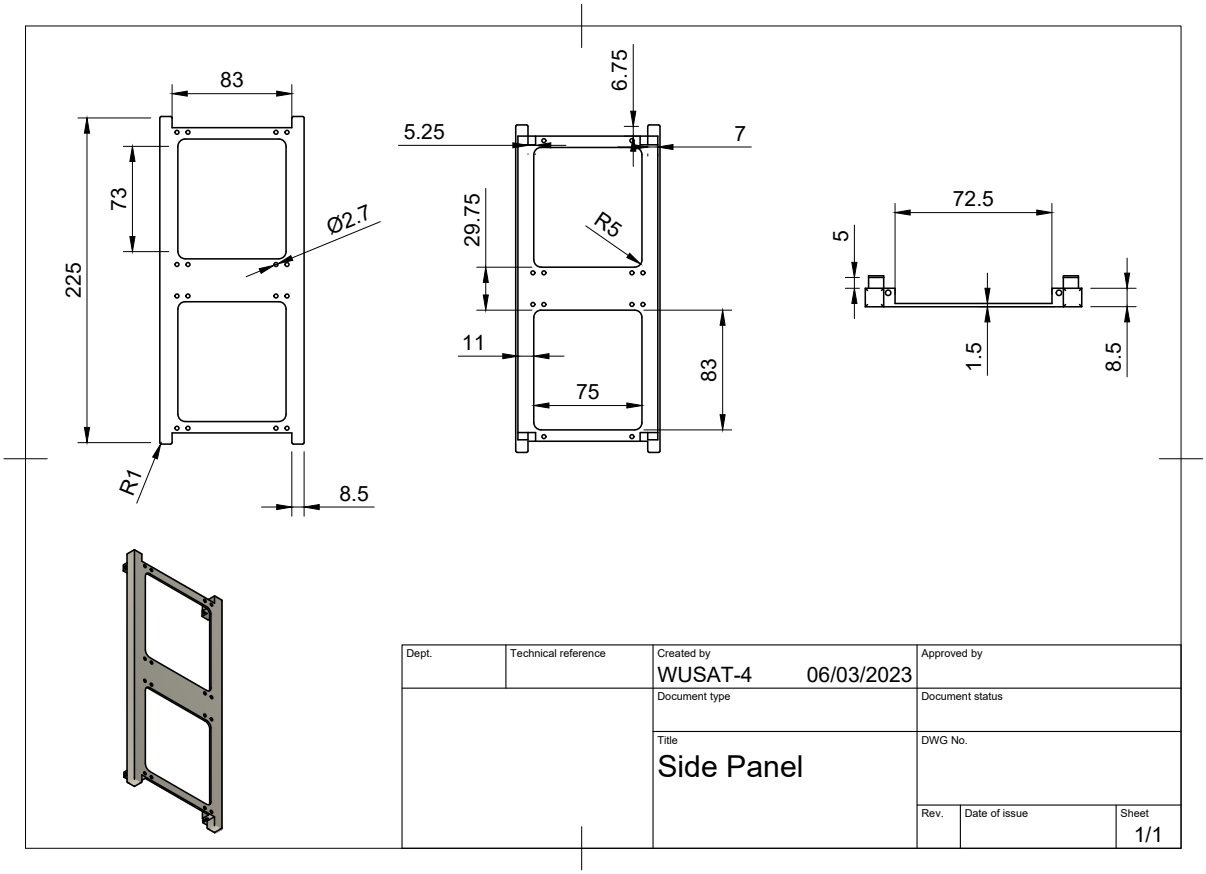


Figure 11.1: Side Panel Drawing

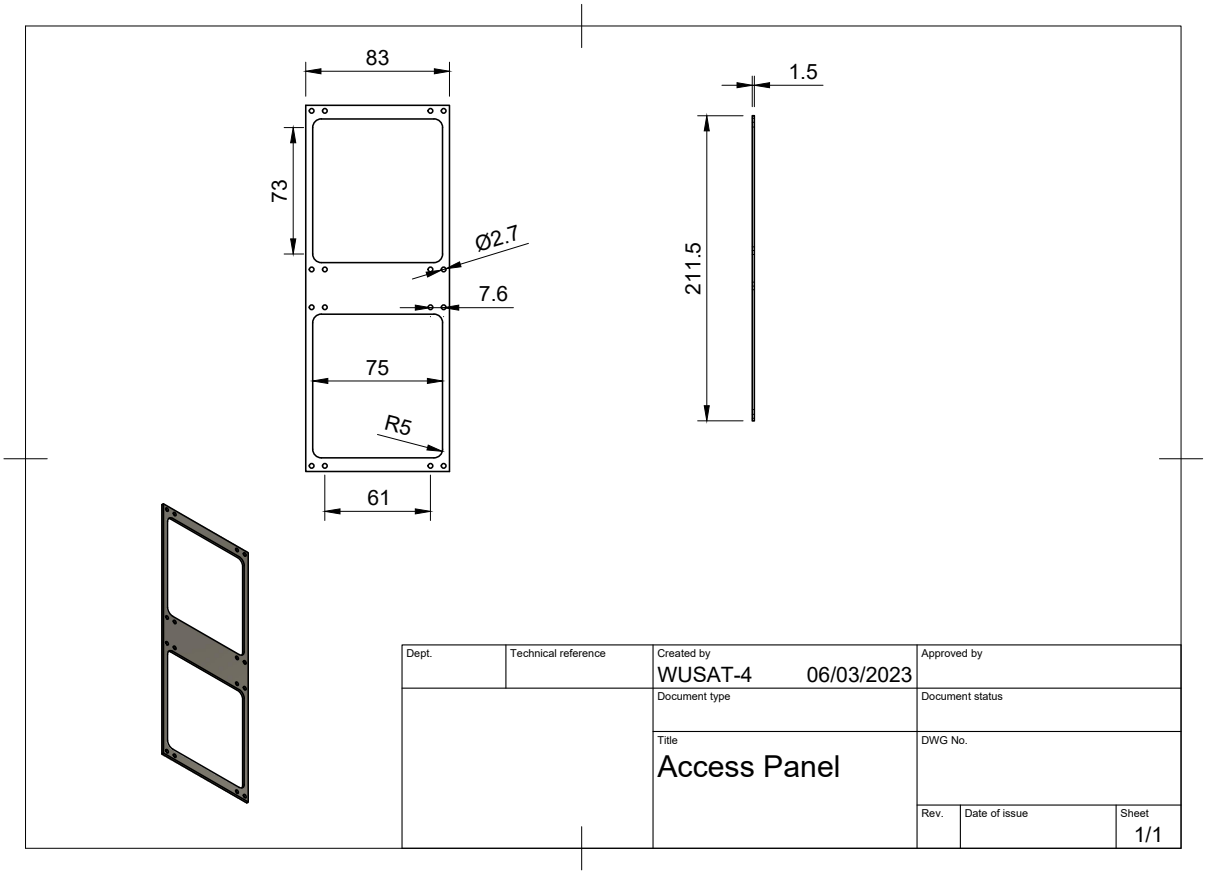


Figure 11.2: Access Panel Drawing

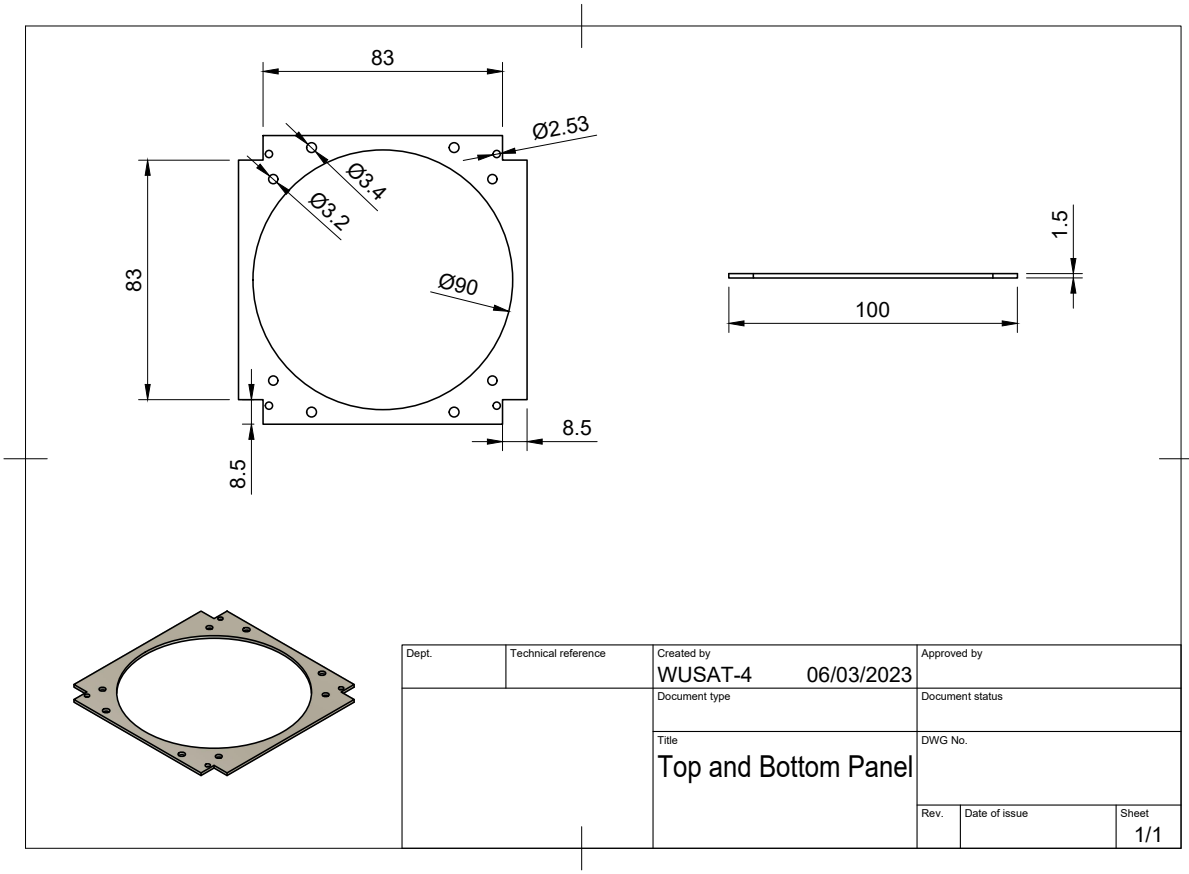


Figure 11.3: Top Panel Drawing