AERO497 | Senior Design Project II

KU CubeSat (2U) – Stage IV:

Attitude Determination and Control System (ADCS) using Reaction Wheel for a Nominal Control Mode

Final Report

Project Members:

Asmaa N. ALyammahi	100044611	Aerospace Engineering
Maha O. ALmazrouei	100044606	Aerospace Engineering
Reem A. ALsulaimani	100045002	Aerospace Engineering
Khulood A. ALmarzooqi	100044400	Aerospace Engineering
Bashayer M. ALmehrzi	100042715	Aerospace Engineering

Project Advisors:

Dr. Elena Fantino	Aerospace Engineering Department
Dr. Ashraf AL Khateeb	Aerospace Engineering Department
Dr. Sean Shan Min Swei	Aerospace Engineering Department

Due Date: 6th of May 2021

ABSTRACT

The Attitude Determination and Control System (ADCS) is a very important subsystem for stabilizing a CubeSat when in orbit and to ensure that it points in the required direction for a successful mission. This final report provides the details behind our senior design project (SDP), which is focused on designing an ADCS subsystem using reaction wheels for KU CubeSat (2U) main mission. The team members have carefully researched the design methodology of an ADCS subsystem and developed the main research, orbit and mission analysis and the main requirements to carry on the designing process throughout the course of this project. In addition, multiple requirements and theoretical assumptions have been made to make the design feasible and within the students reach. Moreover, the satellite's environment of operation was carefully studied and considered based on the previous phases that worked on developing the CubeSat mission and designing it up to this point. An orbital representation and parameters have been assigned and taken into account alongside a numerical estimation of the number of disturbances torques that KU-Sat might face during its mission on the specified orbit environment. Lastly, after selecting and testing the suitable components needed for the success of the mission as well as providing the mathematical models that represents the overall system kinematics and dynamics, our team spent the last few months on building the simulation blocks for the full ADCS project under the careful supervision of YahSat lab using MATLAB/Simulink. This report will discuss the detailed steps and improvements that the students have done from start to finish, starting from the beginning research process until delivering the final project. A fully functioning simulation model of an ADCS subsystem is presented alongside a Reaction Wheel System (RWS) protype and discussed thoroughly in this report.

TABLE OF CONTENT

TABLE OF TABLES

TABLE OF FIGURES

CONSTANTS AND NOTATIONS

NOTATIONS

(ADCS)	Attitude Determination and Control System
(ADS)	Attitude Determination System
(ACS)	Attitude Control System
(FOV)	Field of View
(LEO)	Low Earth Orbit
(PA)	Product Assurance
(QA)	Quality Assurance
(ISS)	International Space Station
(ECI)	Earth Centered Inertial
(BRF)	Body Reference Frame
(DCM)	Direction Cosine Matrix
(CoM)	Center of Mass
(CoP)	Center of Pressure
(COTS)	Commercial Off-The-Shelf
(IMU)	Inertial Measurement Unit (IMU)
(BLDC)	Brushless Direct Current
(RWS)	Reaction Wheel System
(ISS)	International Space Station
(BRF)	Body Reference Frame
(COG)	Center of gravity
(PD)	Proportional derivative
(ESC)	Electronic Speed Controller
(PWM)	Pulse Width Modulation
(QA)	Quality Assurance
(PA)	Product Assurance

CONSTANTS

Density ($\rho_{@alt=420km}$) $= 3.745 \times 10^{-12} \frac{\text{kg}}{\text{m}^3}$

Earth's Gravitational Parameter (μ) $= 3.986 \times 10^{14} \frac{m^2}{s^2}$

Earth's Magnetic Moment (M) $= 7.96 \times 10^{15} \text{tesla} \cdot \text{m}^3$

Solar Flux (F_s) $= 1358 \frac{W}{m^2}$

Speed of Light (c) $= 3 \times 10^8 \frac{m}{s}$

ACKNOWLEDGEMENTS

We would like to first thank Ms Aysha Alharam, Mr Yaqoob Al Qassab, Mr Abdulla Al Ansari, Mr. Vu Thu, and Mr Basel from YahSat Lab for their continuous guidance, patience, help and support and whom without we couldn't have reached this level of sophisticated outcome and understanding level.

Secondly, we would like to express our deepest gratitude for Mr. Ahmed Al Hantoobi for the endless support and guidance in the printing and assembly of the mechanical systems, and all the time and effort invested into this project. Also, our thanks go to Ms. Najla Al Harbi for the time and effort sat into supporting us in the reach of the final destination in this project.

Most importantly, we would like to thank one another for the continuous support, love and encouragement that sat us ahead into reaching the final stage of our project and for all the hard work established and proved since day one. Lastly, we would like to thank our supervisors Dr. Elena Fantino, Dr. Ashraf AL Khateeb and Dr. Sean Shan Min Swei for the continuous support and guidance.

Chapter 1. INTRODUCTION

1.1. CubeSat and Space Missions

Small satellites, generally with a 500 kg mass or less, have been around for over half a century, but the interest in this field of space implementation is growing rapidly. “Numerous academic, government, and commercial entities, and even pre-college age hobbyists, are undertaking small satellite projects around the world” [1].

Table (1): Classification of small satellites, highlighted in green is our Satellite of interest

Class	Wet Mass (kg)
Minisatellite	100 - 500
Microsatellite	10 - 100
Nanosatellite	1 - 10
Picosatellite	0.1 – 1

The word” CubeSat” refers to a satellite with a cubic shaped structure with a fundamental size of 10cm×10cm×10cm and having a mass of 1 kg or less only. The standard size of a cubic satellite is usually referred to as “1U”, with U meaning a” unit”. “CubeSats are often configured in 1.5U, 2U, and 3U sizes as well, with a corresponding length of 15, 20, and 30 cm respectively while maintaining a 10 cm × 10 cm face” [1].

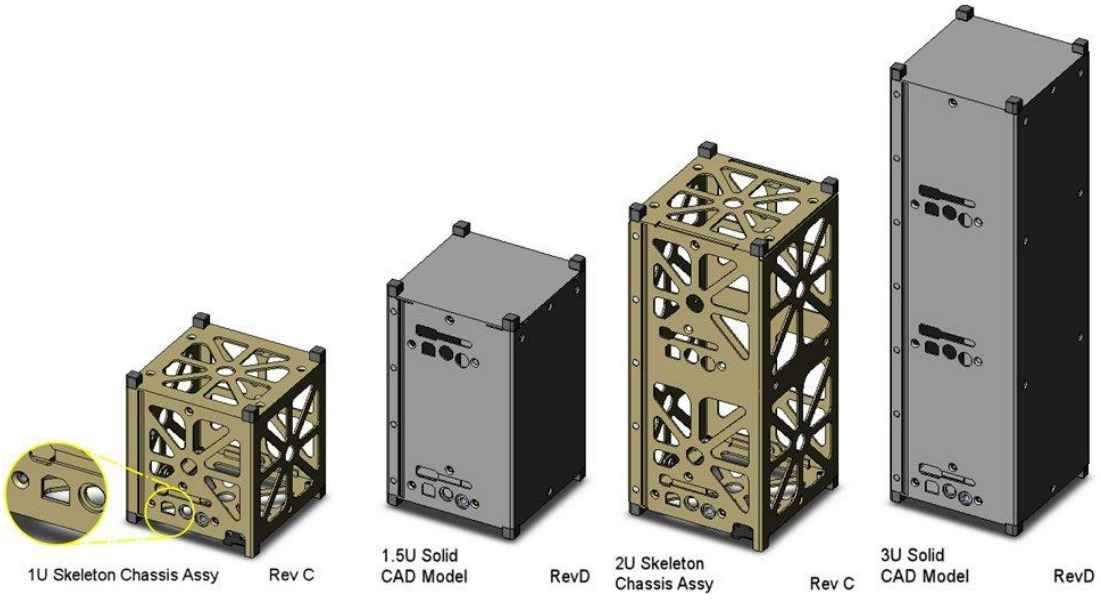


Figure (1): A CAD model of CubeSats with different units' configuration [2]

The CubeSat concept was first proposed in 1999 by Jordi Puig-Suari (California Polytechnic State University) and Bob Twiggs (Stanford University). Within 15 years, this conventional concept has been implemented by universities, commercial and civil sectors, and military users. Thanks to advancing technology and rising CubeSat fame, the complexity and competences of the so far launched CubeSats have improved, which has allowed many several commercial off-the-shelf (COTS) subsystems to be available and easily prototyped [3].

One of the many reasons behind the success of the CubeSat concept's is their fast development time, which enables undergraduate students like ourselves and postgraduate students to be involved in spacecraft projects from their inception through design and finally to launch and operations (such as Dhabi Sat which was launched early this year). Such feasibility is not possible with traditional spacecraft projects giving us the advantage of learning and diving deep into this useful application [3].

Recent advances in size reduction have enabled CubeSats to be equipped with the so-called reaction wheels, providing greater response speeds and pointing accuracy. "With current commercial state-of the-art systems, sub-degree pointing accuracy and slew rates >10 deg/sec (3U CubeSat) can be achieved." Such state-of-the-art COTS Attitude Control Systems (ACS) take less than 1U and include three reaction wheels (one on each axis) and three magnetic torquers, as well as the conventional associated attitude determination equipment (e.g., start-tracker, sun sensors, angular velocity sensors, and magnetometer) and control computers. Such configuration will be discussed thoroughly in the next section [3].

1.2 Attitude Determination and Control System (ADCS)

The word "Attitude" refers to "the three-dimensional orientation of a vehicle with respect to a specified reference frame". Generally, Attitude Systems includes a set of sensors, actuators, avionics, algorithms, software, and ground support equipment used to determine and control the attitude or in other words the orientation of a vehicle. Attitude systems are commonly referred to by a variety of names, such as

- Attitude Determination and Control System (ADCS)
- Attitude Ground System (AGS)
- Attitude and Orbit Control System (AOCS)
- Guidance, Navigation and Control (GNC)

or whatever other term suitable to describe the engineers' focus in achieving the desired attitude for a particular mission. When we use an acronym in this report, we will use ADCS, so that it fits the addressing of our space mission properly [4]. For a better understanding of the overall system, we will casually refer to it as a combination of two subsystems, an Attitude Determination System (ADS) and an Attitude Control System (ACS).

Attitude Determination is "the process of combining available sensor inputs with knowledge of the spacecraft dynamics to provide an accurate and unique estimation of the attitude state as a function of time." The results of running an attitude determination, the attitude estimate or solution, is a knowledge of the vehicle's current and desired orientation in space and this can be obtained by using a set of sensors to relate the information regarding the surrounded external references, such as the stars, the Sun, the Earth, or other celestial bodies, to the orientation of the spacecraft. Commonly, any single attitude sensor will have a certain noise level or other disadvantages that prevents it from providing a fully acceptable attitude data at all times. Therefore, more than one sensor is needed to meet all mission requirements for a given mission objectives [4].

Attitude control is "the combination of the prediction of and reaction to a vehicle's rotational dynamics." A spacecraft can be passively controlled because it exists in an environment of small and habitually highly predictable disturbances. That is, a spacecraft Attitude Control System (ACS) can be designed such that it uses the environmental disturbances to cause the spacecraft attitude to change to the orientation needed to meet the mission goals. Also, a spacecraft may include a set of actuators that can be applied to actively control the spacecraft attitude or in other word orientation [4].

1.3 KU (2U) CubeSat Phase IV

In 2016, Khalifa University (KU) started the **"KU CubeSat (2U) project"** and it was divided into five major stages. (Phase I) involved studying the mission and designing the main structure of the CubeSat. Then, in (phase II), the student used thermal and vibration tests to examine the main structure of the CubeSat, while in (Phase III), the students built the full electrical system for the communication and the overall power subsystems of the CubeSat. Currently, as a part of our Senior Design Project, we will carry on the work of (phase IV) for the KU CubeSat Project, where we will be designing and building the Attitude Determination and Control Subsystem (ADCS) using reaction wheels that fits the requirements of the previously mentioned CubeSat mission and prototype.

In (phase IV) the students were expected to build on the existing KU CubeSat model by designing a well fitted, compatible (ADCS) subsystem that meets the design requirements and constrains from the previously mentioned stages of the main project. Our respected clients (project advisors), has required a fully functioning (ADCS) simulation model and a reaction wheels prototype in which we gladly worked on to deliver via this paper now that we reached the end of our senior design project.

It is safe to say that we started the project with the following objectives in mind. However, we can proudly state that looking back at them now, a tremendous level success was reached.

- To build on the KU CubeSat (2U) main project with a compatible subsystem
- To design a fully functioning ADCS system prototype using Reaction Wheels
- To program and simulate the ADCS prototype to uphold an adequate performance
- To fulfill all the design requirements presented by the clients with high efficiency
- To apply our deep engineering knowledge and skills in real life projects
- To learn and have a hand on experience with Space and orbit mechanics

1.3.1 Goals & Aim

The goal behind stage "IV" of KU-Sat Senior Design Project is to carry on the previous 3-years of work dedicated to designing a full functioning 2 units nanosatellite. During the academic year (Fall 2020-Spring2021) our group worked thoroughly on designing the appropriate Attitude Determination and Control Subsystem (ADCS) to work on achieving the main "Earth Observation" purposes.

Our main aim was to develop a full ADCS simulation model using MATLAB Simulink that can estimate the attitude of a 2U CubeSat to determine its current position when in orbit. Based on the reading established by the simulated model, a full simulation of actuating control model will be developed to show the desired earth pointing control mode.

Alongside simulations, since combining a full ADCS system hardware is very challenging when considering the limited resources and time; we will test each hardware component separately to show their input/output data and discuss the ways of combining them by the end of this project. Also, we developed a Reaction Wheel System (RWS) prototype to show the capability of using Commercial Off the Shelf (COTS) components in reaching space applications quality. We summarize the overall scope and deliverables achieved in the following table,

Table (2): Summary of the deliverable's goals and objectives for the project

Aim	Status	Final Deliverables
Acquisition of theoretical background and state of the art on ADCS	Accomplished	Literature review
Understand the ADCS working principles, use and implementation	Accomplished	Literature review
Development of a conceptual design	Accomplished	System Plant
Attitude Determination individual component test and calibration (Sun sensor, magnetometer and gyroscope)	Accomplished	Process / output Documentation
Development and fabrication of (1-3 units) reaction wheel prototyped system for Attitude Control	Accomplished	Process / output Documentation
A full ADCS simulated model using SIMULINK/MATLAB	Accomplished	SIMULINK model
Full documentation and dissemination	Accomplished	Final Report

1.3.2 Technical Overview

1.3.2.1 Attitude Determination and Control System (ADCS)

Maintaining the desired orientation in space, with a specified level of accuracy is a mission requirement for every spacecraft. Attitude Determination and Control system is responsible for satisfying such requirements. The orientation of a satellite in space is called “attitude”. So, simply the spacecraft’s Attitude Determination and Control system (ADCS) refers to the entire range of methods used to

- Orient the satellite in the desired orientation
- Sense the orientation of the satellite relative to reference (e.g. inertial) points
- Stabilize the satellite to the desired pointing direction
- Control the satellite orientation despite external disturbance torques acting on spacecraft

To ease the understanding/working process, it is safe and more efficient to deal with the system like we specified before as two separate subsystems, an Attitude Determination System (ADS) and an Attitude Control System (ACS) [2].

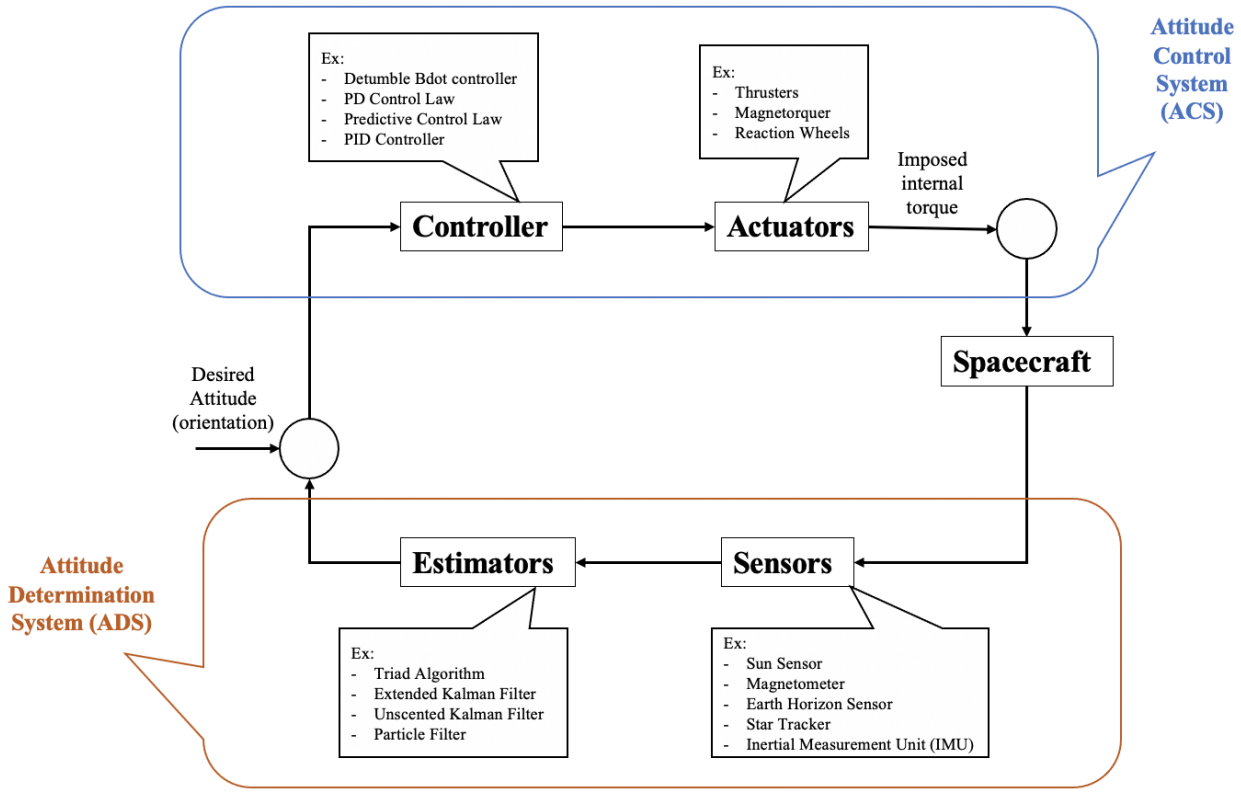


Figure (2): General Dynamic model of an Attitude Determination and Control system

1.3.2.2 Attitude Determination System (ADS)

In general, satellite attitude determination can be seen as a two-stage process, where the first stage is to get position data from a reference sensor using a reference space element such as the sun, the stars, the earth horizon and the magnetic field to get a vector position data alongside an inertial sensor to provide angular rate readings [5].

In the second stage, methods called "estimators" use the spacecraft's mathematical model to obtain an estimation of the spacecraft's attitude position and rotational rate. This estimate however is then compared to the one obtained from the sensors reading to provide a more accurate attitude representation. These estimate techniques tend to be critical and very time consuming, depending on that we were advised to only use the "Triad Algorithm Estimation Method". Other common estimate methods used in a general ADCS system are [5],

- Extended Kalman Filter
- Unscented Kalman Filter
- Particle Filter
- Triad

The attitude (orientation) of the satellite must be measured and/or estimated in order to accurately control its position in the desired direction. A sufficient number of sensors must be implemented to get the necessary accuracy on the estimated values to achieve the full 3-axis control of the S/C [3].

Two classes of sensors are often mixed in a simple ADCS arrangement [5],

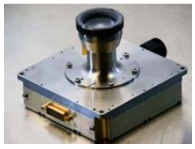
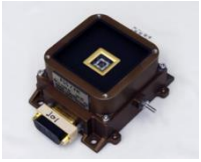
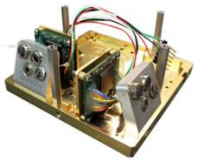
1. **Reference sensors:** Use references such as the Earth, the Sun or the stars and provide vector observations. Often measurements from more than one reference sensor and measurements over time are fused, to overcome the problem of unobservability where some references might not appear in the satellites field of view. Combining more than two sensors solves the problem of unobservability and improves the attitude determination [5].

2. **Inertial sensors:** Can provide angular rate relative to an inertial frame. They can also step in when the urgent matter of unobservability appear, where the ex. gyroscope can help propagate the attitude estimate. However, inertial sensors are limited by noise and bias errors and will not work alone, due to unbounded errors in the attitude estimate over time [5].

The main sensors used for attitude determination purposes are star trackers, horizon sensors, sun sensors, GPS antenna arrays, magnetometers and angular rate gyroscopes.

A small summary explaining each is presented in the following table.

Table (3): Examples of commonly used sensors in a typical satellite ADCS system [6]

Device	Working Principle	Disadvantages	Accuracy
Reference Sensors			
Star Tracker 	Takes images of a part of the sky then compares it to a map stored in its memory to finally determine the orientation relatively to the stars	In need of a reference map and heavy data processing. Also, it is very heavy for the application of CubeSats	1 arcsec – 1 arcmin (0.003-0.001 deg)
Sun Sensor 	provide the satellite with the azimuth and elevation of the sun vector, i.e. the direction to the Sun, giving two axis of attitude knowledge.	They need direct sunlight (so they need to be on the sides of the CubeSat) with a very low accuracy	(0.005 – 3 deg) 0.01 deg nominal
Horizon Sensor 	provide the satellite with attitude knowledge relative to the Earth. It measures infrared radiation coming from the direction of Earth's horizon so that the angle between the horizon and the sensor can be determined.	It has a coarse accuracy that is affected by the Sun and Moon. Not as widely used as sun sensors.	<0.1 – 0.25 deg
Magnetometer 	Measure the Earth's magnetic field vector local to the satellite. They might also be sensitive enough to pick up eddy currents from the satellite's other systems, so care must be taken to distinguish these two.	The magnetic field is not completely known. It is affected by on-board electronics and only applicable in LEO.	0.5 - 3 deg
Inertial Sensor			
Gyroscope 	Do not detect the absolute attitude of the satellite, but the changes in its rotation and acceleration. As the satellite rotates a gyroscope stays in place due to gyroscopic rigidity and the deflection can be measured.	Tend to drift over time, they should be calibrated with an absolute method at specified intervals	0.001/h (drift)

1.3.2.3 Attitude Control System (ACS)

After getting the information about the spacecraft attitude in space from the ADS, we must now focus on finding ways to control it or in other word manoeuvre its orientation into the desired position. In many ways, this system is very crucial, it controls the stability of the aircraft and it is the main contributor to mission's success [5].

Spacecrafts in general use the ACS system for two main functionalities, first, to point the spaceship into the desired location depending on the mission's need which is what we call "slew manoeuvre". Secondly, since the environment in space has a very large influence on the S/C body, sometimes natural phenomena such as earth's magnetic field, aerodynamic forces, solar radiation pressure and earth's gravity field impose an undesired torque into the satellite body. We call these undesired torque "Disturbances Torque", The ACS system is to force an internal counter torque provided by the reaction wheel system to bring back the S/C into the original trim position once such disturbance occurs [5]. A further detailed version of such disturbances is provided in the mathematical model section of this report.

Controlling the satellite's attitude, in other words imposing an internal torque is called attitude control. However, the methods used to rotate the spacecraft by imposing a torque motion can be divided into two categories [5],

- **Passive Attitude Control [5]:**
  - Take advantages of basic physical principles and/or naturally occurring forces by designing the spacecraft so as to enhance the effect of one force, while reducing the effect of others which results in a change in attitude
  - Continuously orient the satellite but it cannot change its attitude or position based on external feedback
  - Uses Passive Actuators such as
    1. Gravity-gradient stabilization
    2. Spin stabilization
    3. Permanent Magnets

- **Active Attitude Control [5]:**
  - Directly senses the spacecraft attitude and supply a torque command to alter it as required
  - Feeds commands to the satellite in order to activate some actuators which re- position and reorient the satellite
  - Uses active actuators such as:
    1. Thrusters
    2. Magnetic torquers
    3. Momentum-control devices

Table (4): Summary of typical passive and active actuators used in an ADCS system [5]

Type	Actuator	Working Methods	Typical Accuracy
Passive	Gravity-gradient stabilization	Utilize that the Earth's gravitational field is non-uniform. The satellite is stabilized around two axes by adding a gravity boom to the satellite or by making it long enough to take advantage of the gravity-gradient.	± 5 deg (2 axes) (simple)
	Spin stabilization	Intentionally spinning the satellite rapidly around the third axis utilizing the gyroscopic effect created thereby to stabilize the spaceship in two axes.	±0.1 deg to ±1 deg (2 axes) (rotation)
	Permanent Magnets	works the same way as the magnetorquers, however, the permanent magnet does not require any power to control the satellite and can help ensure two axis stabilization the entire lifetime of the satellite, if the satellite has been detumbled.	± 5 deg (2 axes)
Active	Thrusters	Utilize Newtons third law of motion by expulsion of propellant at high velocity in one direction, thereby causing the satellite to be pushed in the opposite direction with an equal force.	±0.1 deg to ±1 deg (Propellent limited) (Large impulse)
	Magnetic torquers	a commonly used method to control small satellites in Low Earth Orbit (LEO). The magnetorquer creates a magnetic field, which will interact with the Earth's magnetic field, causing the satellite to rotate, until the two fields are aligned.	±1 deg (high current)
	Momentum-control devices	Utilize Newtons third law of motion by rotating a mass, creating a torque, which affects the satellite in the opposite direction of the rotation	±0.001 deg to ±0.1 deg (High mass and power) (Momentum damping)

1.3.3 Overview of the report

This report will describe the entire SDP project in seven chapters. The first chapter is an introduction to the project. It comprises of a simple introduction of CubeSat and space mission, simple information about ADCS, goals and objectives of our project and an overview of the technical area. The second chapter is about system overview where it will discuss the requirement and specification of our ADCS design. The third chapter is about the theoretical foundation: engineering model and what standards and regulation we should follow in addition to the details of the theoretical model. The fourth chapter will introduce the concept generation and design evaluation in details. The fifth chapter will illustrate the implementation and fabrication of the selected design from chapter four. The sixth chapter will illustrate the full design simulation model of the system which the core results of our work. The last chapter concludes all the work that we did and recommendation for future work.

Chapter 2. SYSTEM OVERVIEW

2.1 Requirements

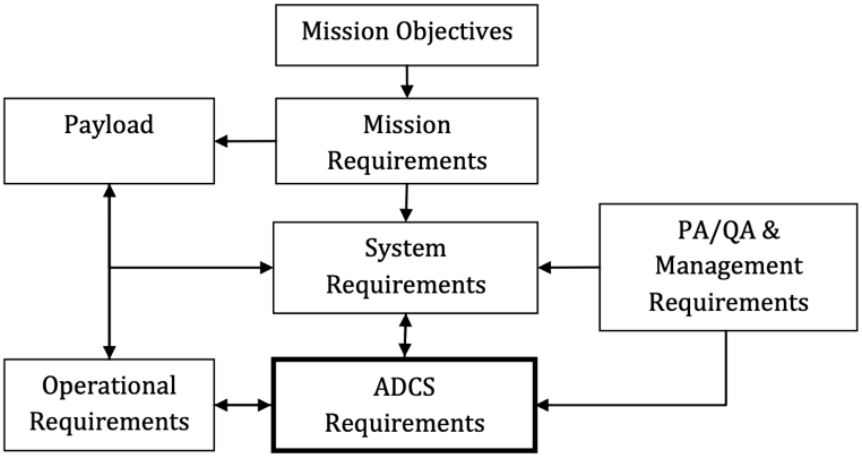


Figure (3): ADCS requirement derivation [7]

We established the requirements of our project based on the flow presented in the chart above.

In this section of the report, we present the following requirement tables based on the work done in the past three years and our rising established requirements as we dug deep into what is an ADCS system is within the domain of this year.

Table (5): KU-Sat Mission Objectives [8]

Mission Objective	Statement from previous phases
1	To collaborate with a governmental real estate project that will commence in late 2019 or beginning of 2020
2	To capture the development of the land over a period of time and detail the progress of the land
3	To share the captured images with investors via an online platform, or through a governmental entity
4	To share the captured images with research institutes to study the effect of economic rise or decline on the development of land

Table (6): KU-Sat Top Level Requirements [8]

Functional Requirements	**Statement from previous phases**
1	The launch Vehicle shall deliver the CubeSat to the ISS
2	The Captured images shall have adequate resolution
3	The land progress shall be tracked within a long-time gap
4	The transceiver shall be compatible to transfer the data byte of the captured image
5	The Power system shall be capable to supply power to all the subsystem
Operational Requirements	**Statement from previous phases**
1	The LV shall deliver the CubeSat to an orbit of 400 km altitude
2	The captured images shall have resolution of 1280x1024 pixels
3	The land progress shall be tracked monthly
4	The transceiver shall have a data rate of 100 - 10,000 bits/ sec
5	The power system shall be capable to a supply power of 15 W

Table (7): KU-Sat ADCS Upper-Level Requirements [7]

Requirement	**Specifications**
1	The ADCS shall be able to withstand the operation environment
2	The ADCS shall meet the operational requirements for the mission
3	The ADCS shall be compatible with the system interfaces
4	The ADCS shall be compatible with the project's schedule and model philosophy
5	The ADCS shall be compatible with the project's Product Assurance (PA)/Quality Assurance (QA) philosophy

The upper-level requirements are established at the beginning of the project and were chosen from the mission objectives, requirements and progress made by previous groups (stage 1 – 3).

Nonetheless, the lower-level requirements are derived from the upper-level ones. It is safe to say that the lower-level requirement altered depending on the feasibility of the current health situation, but they are established to aid what we aspire to deliver now that we reached the end of this project. A summary of the lower-level requirements table is presented in the next page.

Table (8): KU-Sat ADCS System Requirements [9,8,10]

System Requirements	Specifications
Interfaces (mechanical, electrical, thermal, communication etc.)	Develop a separate testing prototype
Dimensions	100mm × 100mm × 9.3740mm
Power Input	0.2 (watts) (Idle) 1.32 (watts) (Active)
Power Consumption	1.32 (watts) (Actual Power) 1.612 (mWh) (Energy Consumption)
Mass	1.758 (kg)
Durability (radiation, thermal etc.)	Operational Temperature (-40 to +70 °C)

Table (9): KU-Sat Operational Requirements

Operational Requirements	Specifications
Pointing Requirements	
Pointing accuracy and knowledge	Depending on best results established
Slew rates	Depending on best results established
Operational Modes	
Camera Pointing Mode	Necessary, must be accomplished
Slew Maneuver Mode	Necessary, must be accomplished
Reaction Wheel Desaturation Mode	Not possible (Lack of testing facilities)
Detumble Mode	Was not considered due to shortage of time

Table (10): KU-Sat ADCS Management Requirements

Management Requirements	Specifications
Model / Prototype	- A fully functioning ADCS model using SIMULINK - An individual test/calibration for each selected hardware - A Reaction Wheel System (RWS) prototype using (COTS) Components
Schedule	- Call for SDP's – Beginning of the Semester - Publication of SDP list – (3 Sep 2020) - Submission of Project Selection Form – (08 Sep 2020) - Publication of project allocations – (10 Sep 2020) - Submit /Project Proposal – (1 Oct 2020) - Progress Report – (12 Nov 2020) - Mid-term Report – (04 Feb 2021)

	- Mid-term Presentation – (07 Feb 2021) - Final Report Submission – (06 May 2021) - Final Project Presentation – (Not announced yet)
Documentation requirements	- Project Proposal (Submitted) - Project Progress Report (Submitted) - Project Mid-term Report (Submitted) - Final Project Report (Current)

Table (11): KU-Sat ADCS Final Verification Requirements

Verification Requirements	Methods	Specifications
ADS Simulation Model	SIMULINK	Provides an accurate attitude information
ACS Simulation Model	SIMULINK	Provides the required internal torque
Combined ADCS Simulation model	SIMULINK	Satisfies all the mission requirements mentioned above
Reaction Wheel System	Prototype	(1-3 units) Prototype that provides the necessary torque value
Magnetorquer rod	Hardware	Simple test and configuration
Gyroscope	Hardware	Full testing, reading extraction and calibration
Sun Sensor	Hardware	Full testing, reading extraction and calibration

2.2 System Specification

As per the requirements established for KU-Sat from phase one, the orbit where KU-Sat is intended to be deployed in is the Low Earth Orbit (LEO) from the International Space Station (ISS) at an altitude of 420km for the purpose of observing the changes that occur to the lands in the United Arab Emirates (UAE) for a mission period of two years.

At an earlier stage, a camera model was chosen and mounted on top of the CubeSat, the aim of our project, the ADCS system however, is to point this Camera to the desired UAE lands once it finds a clear field of view (FOV) of the intended site. To do so, we established the so called (Pointing Mode) and this will be the core work of our system's design and control simulation toward delivering the final product.

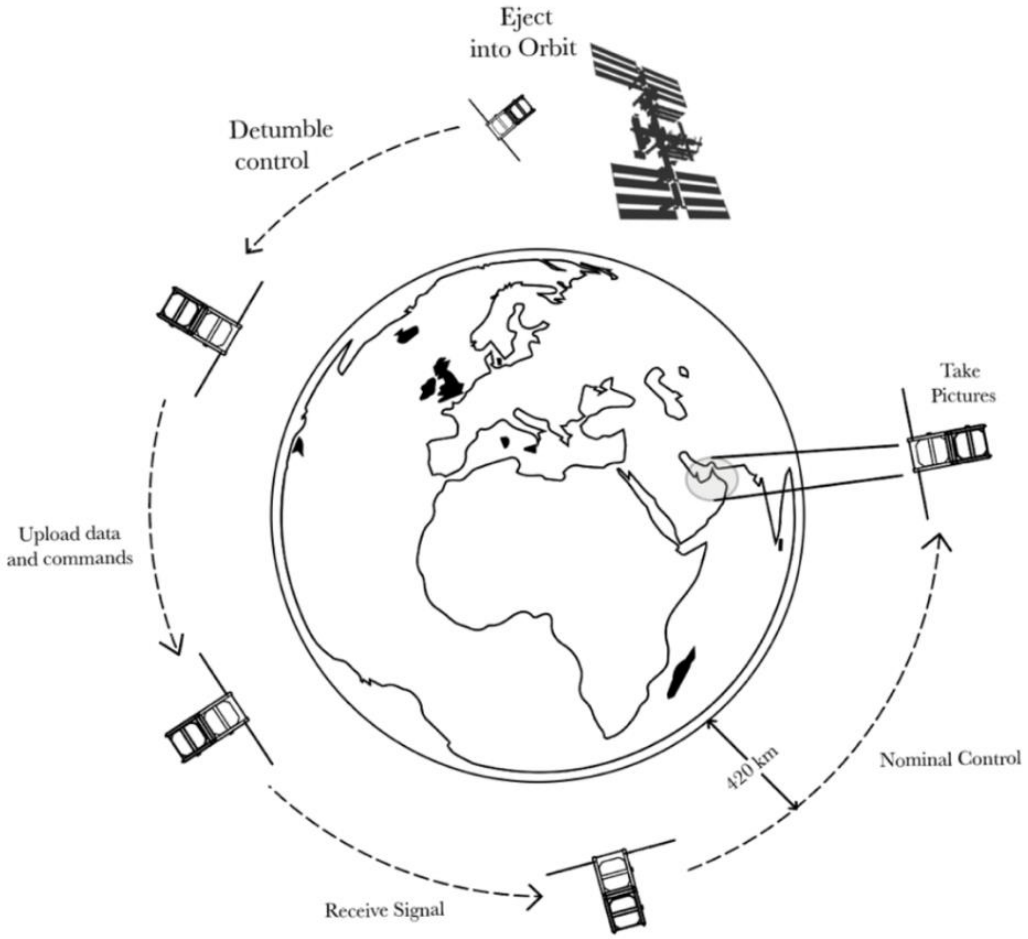


Figure (4): KU-Sat estimated mission in LEO Orbit

2.2.1 ADCS Subsystem Plant

Now that we have reached the end of our SDP journey, we can gladly present a simulation model of the established ADCS subsystem via MATLAB, SIMULINK and other suggested software by YahSat lab. Safe to say, the testing of each separate ADS and ACS component individually and the designed RWS prototype is discussed in the coming sections since they are not the main deliverables of this project.

The reason behind this scope of work and deliverable being is that any ADCS system full hardware prototype is quite advance and the overall assembly of all components from both parties needs a lot of calibration and advanced testing facilities in which we lack in our available resources and labs. Nonetheless, the simulation model was our main focus and is thoroughly summarized under this paragraph in the hope of giving an overview of the designed system.

The main simulated plant consists of two main blocks, namely, the Estimation block and the Control Block. Each of the presented block has its own purpose and design choices.

- From the sensor readings and the defined initial models (IGRF and Sun Model) a set of information is fed into the Estimation plant
- The Estimation Plant is a set of algorithms responsible for determining the current and desired position from the inserted mathematical model data and the fed information. The outcome of this plant is then fed to the Controller plant
- The Controller Plant is then responsible of acting upon the attitude information by giving the necessary commands to the main actuators, namely, the reaction wheel model to change the orientation of the CubeSat from the current position to the desired one.

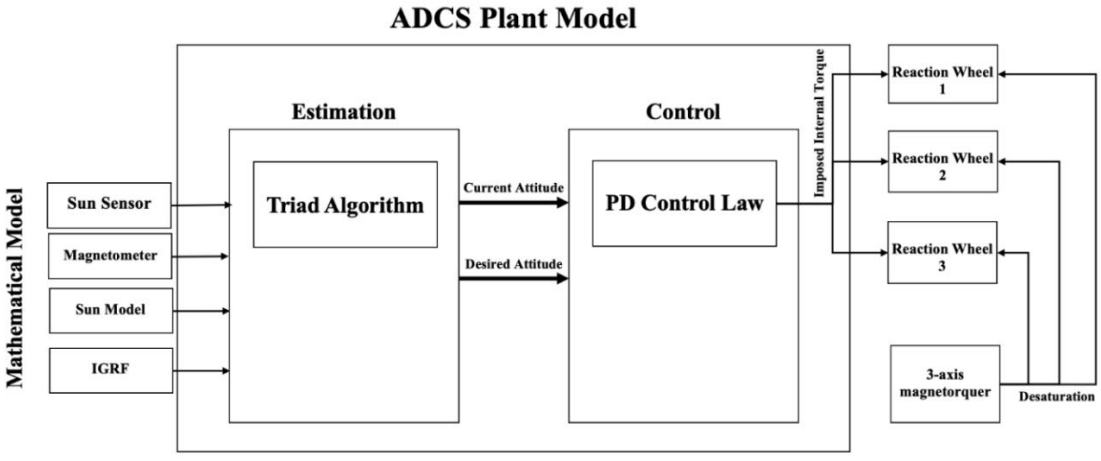


Figure (5): KU-Sat's designed full ADCS plant

The Estimation Plant itself is not a simple straight forward concept of application. It consists of many small details and requirements that are above and beyond the level of an undergraduate engineer. Nonetheless, we tried our best to establish a simple yet a complete model that does the job of attitude estimation while fairly giving all design factors justice and quality focus.

- The Mathematical model consists of a set of sensors needed for data extraction from the Sun Sensor, Magnetometer and Gyroscope reading. This choice was limited to the lowest number of sensors needed that can do the estimation job properly. The other two elements of the mathematical model are a default input for the estimation algorithm to understand the environmental surrounding of the CubeSat but we only considered one since we were limited by assumption and recourses.
- The type of algorithm used to determine the current/desired attitude information was limited to Triad Algorithm since it is the easiest and fastest one within reach of our knowledge domain. Other estimation methods tend to be pretty complex so this decision was verified as attainable by our respected colleagues in YahSat Lab.
- The output of this plant is two data set (two vectors) containing the quaternion knowledge of orientation for both the current position and the desired one. This information is fed into the Control plant to be acted upon by the actuators in position.

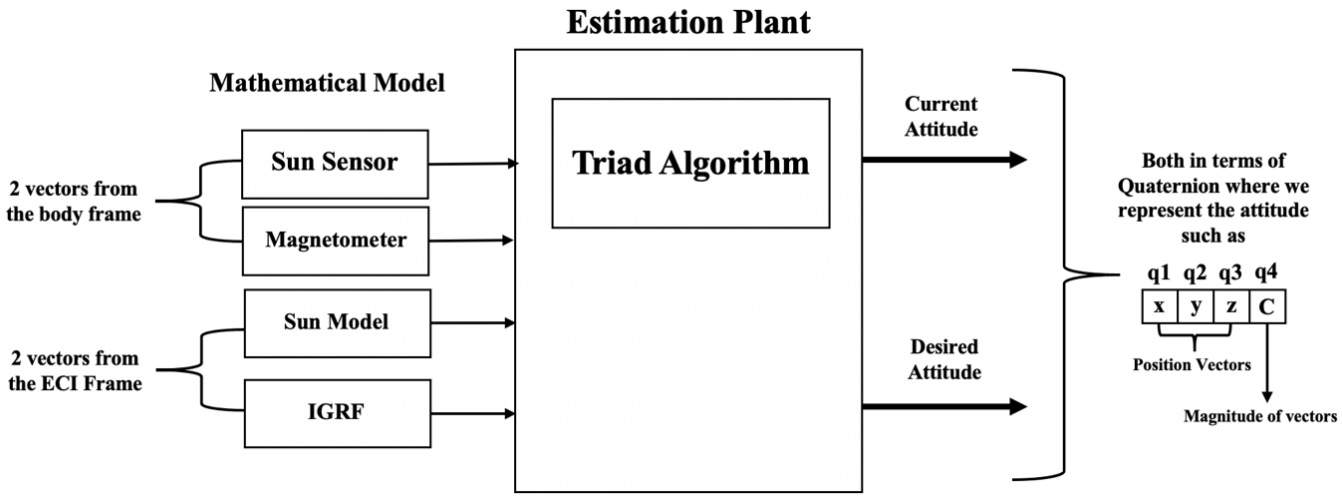


Figure (6): KU-Sat's designed ADS plant

Lastly, the Attitude Control Plant, this plant main aim is to read and encrypt the knowledge received from the Estimation plant into a set of commands to the main actuators, the reaction wheel system. As specified before, there are many ways to control a CubeSat, in this project we decided to have the main controller be a PD Controller since all members are familiar with it and safe to say it is

widely used for earth pointing modes in general. Also, the selection of the main actuator was a design limitation since our project emphasized on the use of an active controlling method with a set of reaction wheels.

- The encryption of the attitude knowledge is done via the application of the known Dynamic and Kinematic equation of motion representing the satellite behaviour.
- After the manipulation of such data, a voltage command is then sent to each reaction wheel controlling each axis to move the CubeSat in a way that deform it from the current position to the desired position when in orbit to perform the necessary manoeuvre that points the camera toward the desired UAE lands.
- A 3-axis magnetorquer is added for the sake of completeness in the purpose of desaturating the reaction wheel since it tends to have many issues in this domain as stated in many open-source researches.

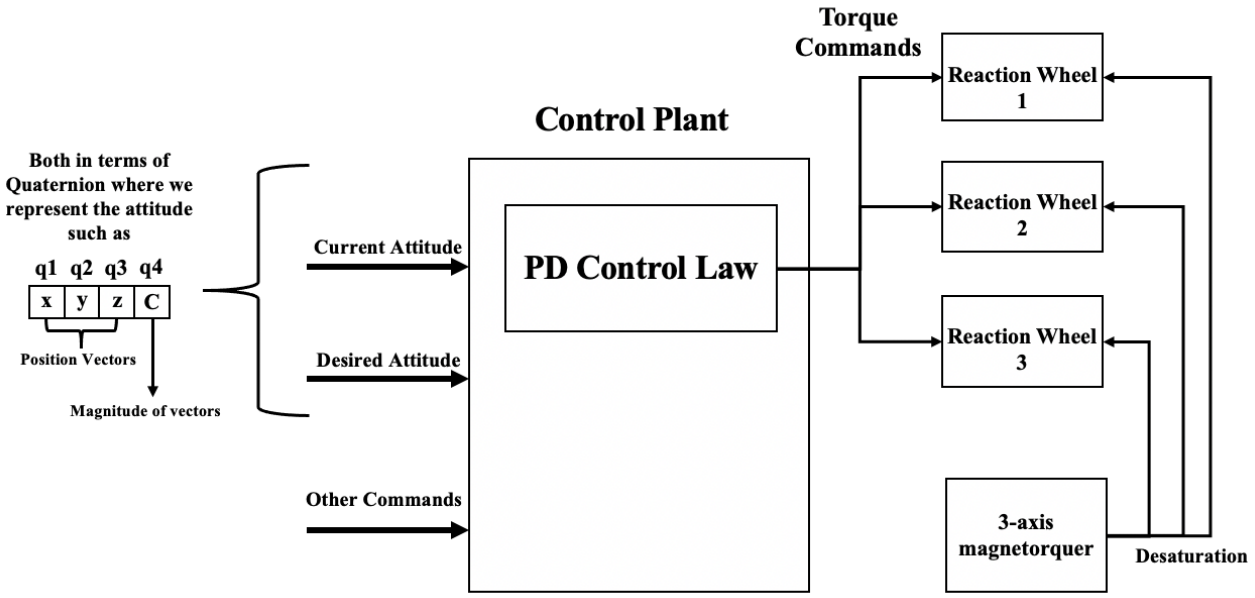


Figure (7): KU-Sat's designed ACS plant

Chapter 3. THEORETICAL FOUNDATION

3.1 Introduction

When it comes to space missions and projects, a field of depth is needed prior to choosing an engineering model or a specific prototype design. Spacecrafts in general must perform their task in a foreign atmosphere surrounded by the unpredictable space conditions. In this section, we will provide all the introductory theoretical parts that led or limited the proper choice of the compatible ADCS components and system allocations that works cohesively with the success of KU-Sat main mission and our course deliverable requirements.

3.2 Standards and Regulations

3.2.1 Design Standard

An overview of the different tasks that must be performed in the design of an ADCS system is shown in Figure 8. This flow was used and discussed in the coming sections for a better understanding of how we came up with our design and delivered the final product.

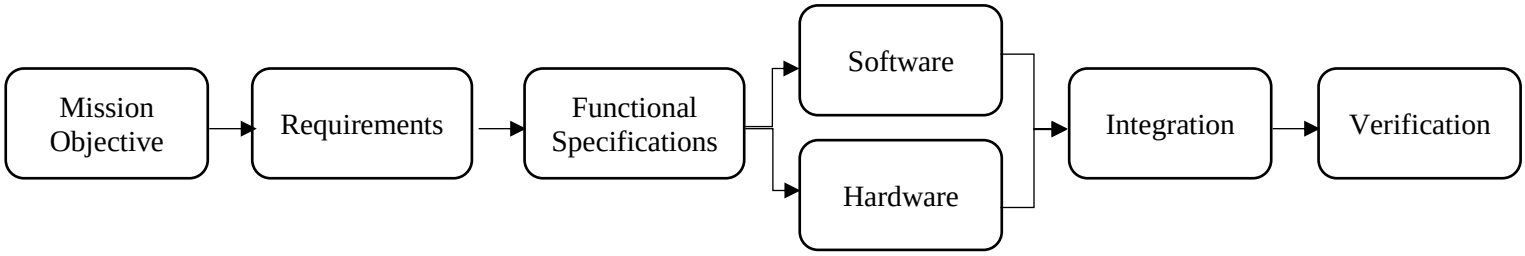


Figure (8): General ADCS Design Path [11]

Defining the mission objective will help with understanding how it affects the ADCS regarding limitations and requirements for the system. Due to that, several different modes of operation can be defined, these modes decide on how the ADCS operates in different situations. Understanding the requirements of the mission will provide guidance on the rest of the design process. As well as, it will help maintaining and providing a system that is compatible with the rest of the satellite. The requirements will also define the functions that the ADCS should perform. These functions are the base for the design definition of the system as they affect the ADCS actions in any given situation. The design definition features the hardware and software design of the ADCS system and is the result of the preceding work. The system is then tested and integrated together with the other subsystems that constitute the whole satellite, and the requirements are then validated as the system is proved to perform accordingly with the defined requirements of the chosen mission for the satellite.

Table (12): Summary of the steps in attitude system design [12]

No.	Step	Inputs	Outputs
1 (a)	Definition of control modes	Mission requirements, mission profile, type of insertion for launch vehicle	List of different control modes during mission. Requirements and constraints.
1 (b)	derivation of system-level requirements by control mode		
2	Quantifying the disturbances environments	Spacecraft geometry, orbit, solar/magnetic models, mission profile	Values for torques from external and internal sources
3	Selection of the type of spacecraft control by attitude control mode	Payload, thermal & power needs, orbit, pointing direction, disturbance environment, accuracy requirements	Method for stabilization & control: three-axis, spinning, gravity gradient, etc.
4	Selection and sizing ADCS hardware	Spacecraft geometry and mass properties, required accuracy, orbit geometry, mission lifetime, space environment, pointing direction, slew rates.	Sensor suite: Earth, Sun, inertial, or other sensing devices. Control actuators: reaction wheels, thrusters, magnetic torquers, etc. Data processing avionics, if any, or processing requirements for other subsystems or ground computer.
5	Definition of the determination and control algorithms	Performance considerations (stabilization methods, attitude knowledge & control accuracy, slew rates) balanced against system-level limitations (power and thermal needs, lifetime, jitter sensitivity, spacecraft processor capability)	Algorithms and parameters for each determination and control mode, and logic for changing from one mode to another.
6	Iteration and documentation	All of above	Refined mission and subsystem requirements. More detailed ADCS design. Subsystem and component specifications.

3.2.1 Applicability in the Current Design

3.2.1.1 Mission Objective

The CubeSat main mission objective was previously established to be the focus of an “Earth Observation” satellite mission since phase 1 in which a Camera was purchased and mounted on the CubeSat body to capture and provide high-quality images of the UAE lands for multiple economic and development purposes [13]. The purpose of our project as discussed before is to build and design the ADCS system that will orient the satellite to point this camera toward the desired land for taking a high-quality image from space.

3.2.1.2 Operational Modes

From the mission objective, it is possible to define the appropriate modes of operation that our ADCS system must operate upon. For KU-Sat ADCS model, where the satellite performs in low Earth Orbit (LEO) employing reaction wheels for attitude control there are four different modes of interest that we initially took into account:

- **Camera Pointing mode**: During the camera pointing mode, the CubeSat must be able to point at a target with high attitude accuracy. A target can be selected by longitude and latitude information so that a picture can be taken at a predefined time.

- **Slew Manoeuvre mode**: Refers to the CubeSat's orientational movement phase or the duration of movement in reference to a plane or fixed position such as Earth, the Sun, another celestial body or other point in space.

- **Reaction Wheel Desaturation mode**: Having reaction wheels in an ADCS system for a CubeSat subjects the overall attitude motion to a period of saturation. This saturation can be absorbed or in other word the overall reaction wheel can be desaturated by using a magnetorquer rod to carry on the mission objectives mission in space.

- **Detumble mode**: As the satellite is deployed into orbit, it will have an unstable angular velocity (undesirable rotating motion) that must be reduced towards zero for an overall stable mission command.

3.2.1.3 Requirements

With the list of requirements defining how the ADCS module shall perform and which limitations it has going forward, it is possible to define the functional specification of the ADCS. The functional specification specifies all the functions that the ADCS module must do and how it is done. It is safe however to note that the requirements have been already discussed in the earlier section. For the sake of continuing the understanding of the design flow, we decided to keep this subsection a minimal to refer as a reminder to the previously established requirements.

3.2.1.4 Design Definition

After defining all the functions that the ADCS shall perform, the actual design was then created. The design definition can be divided into a hardware part and a software part. The coming two subsection explains in details how our design work and was put all together.

3.2.1.4.1 Hardware

The hardware of the ADCS is defined to contain all the components outside the microcontroller that does the job of reading the position related data and act upon the giving command to carry on the mission modes. All of our system architecture is shown in Figure 9.

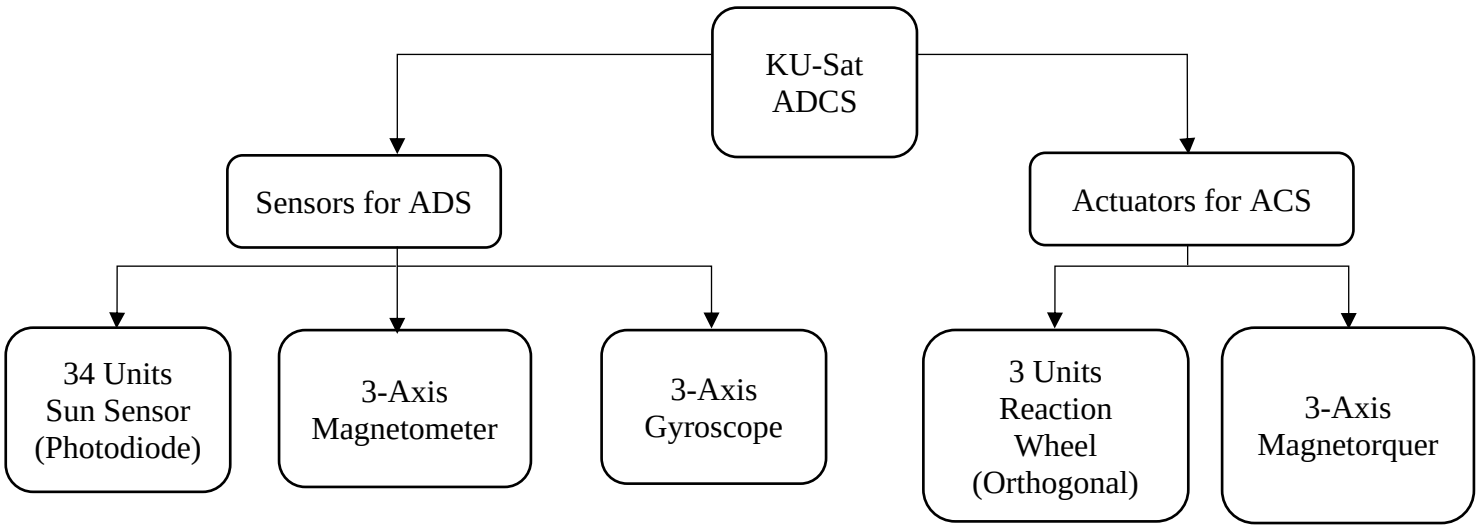


Figure (9): Hardware selection diagram

As we mentioned before, there will not be a full integrated ADCS System hardware. This choice was made due to limitation regarding testing facilities and component budgeting. However, an individual test of each element was done to better understand how they behave and perform in terms of input/output relations. A more in-depth information of how our system components behave and benefit the overall system is discussed in the following tables,

Table (13): Architecture of ADS for KU-Sat

Attitude Determination System (ADS) Hardware Components	
Component	**Work Methodology**
Sun Sensor	- A device that gives the position of the CubeSat relative to the sun - takes in two inputs to output a current value that can then be integrated to a sun vector in body coordinates - Using this sun vector, the ADCS will need one more independent vector to figure out the exact orientation of the CubeSat with respect to the Earth.
Magnetometer	- A device that gives the position of the CubeSat depending on the magnetic field intensity. - Takes in the direction of the Earth's magnetic field and quaternions as its two inputs. It then outputs a magnetic vector in body coordinates. - Using a helper block, the International Geomagnetic Reference Field (IGRF), the magnetometer simulation is able to output a magnetic vector to provide the second independent vector contributing to the attitude knowledge. - In conjunction with the independent sun vector, the magnetometer is able to help provide an appropriate orientation representation of the ADCS with respect to Earth.
Gyroscope	- A device used for measuring the angular velocity of the CubeSat.

Table (14): Architecture of ACS for KU-Sat

Attitude Control System (ACS) Hardware Components	
Component	**Work Methodology**
Reaction Wheel System (3 orthogonal units)	- A brushless dc motor attached to a flywheel used by a spacecraft. It is primarily used for attitude control of the CubeSats. - The control of the reaction wheel is such that it helps it running to keep a satellite in a proper attitude when necessary or change its attitude from one orientation to the other. - In order to orient the vehicle or spacecraft in a particular direction, one has to spin the wheel in its opposite direction if it is the case of an orthogonal configuration.
3-Axis Magnetorquer	- A device used primarily for momentum un-loading and desaturation of the reaction wheel. - It can also be used to control the orientation of the CubeSats. The control law will determine how much torque is needed to go from the current attitude to the desired attitude.

3.2.1.4.2 Software

The purpose of the software namely the simulation model is to integrate the inputs coming from the available ADS hardware and to transform them into a set of commands sent directly to the ACS hardware for a successful mission that is able to meet the functional requirements.

Table (15): Algorithm general breakdown for KU-Sat ADS

ADCS Algorithm	
Attitude Determination	
Mathematical Model (Inputs)	**Description**
Sensors reading	All the data obtained from the selected sensors
International Geomagnetic Reference Field (IGRF)	The International Geomagnetic Reference Field (IGRF) is a collection of mathematical models that describe the large-scale internal component of the Earth's magnetic field from 1900 until today. It is mainly used by commercial organizations and individuals as a source of orientation information." [14]
Sun Model	Not considered in the analysis
Estimation Algorithm	**Description**

TRIAD Algorithm	TRIAD algorithm is among the earliest and yet most basic solutions to the problem of determining the attitude of a CubeSat. It involves developing two orthonormal bases from two sets of vector measurements coming from the sensor's readings. Given the knowledge of two vectors in the reference and body coordinates of a satellite, the TRIAD (TRIaxis Attitude Determination) algorithm gets the direction cosine matrix relating both frames. The two vectors are typically the unit vector to the sun (coming from the sun sensor) and the Earth's magnetic field vector (coming from the magnetometer readings). This algorithm is not an optimal solution, but it provides a reliable estimation of the satellite's attitude while being quite cheap regarding computation needs. [15]
Attitude Control	
Control Law	**Description**
PD Control Law	Different satellite missions use different control laws based on the type of modes they are targeting to accomplish the overall mission. For KU-Sat, we decided to stick to the well-known PD Control Law since it is the most familiar method to all group members. PD-controller is combination of feedforward and feedback control, because it operates on both the current process conditions and predicted process conditions. It contains the proportional control's damping of the fluctuation and the derivative control's prediction of process error. [16]

3.2.1.5 Integration

After both the hardware prototype and the software are finished it must be integrated all together. Throughout the design process it is recommended that there is a continuous dialogue between the people responsible for the hardware and software as the systems are developed. However, for KU-Sat project, the team will focus on the software and the simulation rather than the whole integration of the hardware and software as discussed and approved by the supervisors in reasoning before.

3.2.1.6 Verification

After the software systems have been integrated together and is operational, the requirements must be verified and validated. For KU-Sat, the requirements will be verified after developing a full ADCS Simulink model by members in YahSat lab that worked on Dhabi-Sat ADCS System that was launched earlier this year.

3.3 Details of The Theoretical Model

3.3.1 Mathematical Model

3.3.1.1 Satellite Kinematic Equation of Motion

Before introducing the kinematic equation of motion that represents the satellite, it is safe to introduce the following denotations,

- ($\hat{\boldsymbol{e}}$) represents the rotational axis of a body frame relative to a reference frame by a unit length vector
- ($\boldsymbol{\alpha}$) represents the rotational angle around the rotational axis
- ($\boldsymbol{q_0 = cos(\frac{\alpha}{2})}$) represents the scalar component of the quaternion
- ($\boldsymbol{q = [q_1 \quad q_2 \quad q_3] = \hat{e}\, sin(\frac{\alpha}{2})}$) represents the vector component of the quaternion

then, the quaternion that represents the rotation matrix of the body frame relative to the reference frame is given by [17],

$$q = [q_0, q^T]^T \quad \cdots (1)$$

$$q = \left[cos(\tfrac{\alpha}{2}),\ \hat{e}\ sin(\tfrac{\alpha}{2})\right]^T \quad \cdots (2)$$

Let ω be the spacecraft *body rate* with respect to reference frame represented in the body frame we can say that [17],

$$\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -\omega_1 & -\omega_2 & -\omega_3\\ \omega_1 & 0 & \omega_3 & -\omega_2\\ \omega_2 & -\omega_3 & 0 & \omega_1\\ \omega_3 & \omega_2 & -\omega_1 & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix} \quad \cdots (3)$$

$$= \frac{1}{2}\begin{bmatrix}q_0 & -q_1 & -q_2 & -q_3\\ q_1 & q_0 & -q_3 & q_2\\ q_2 & q_3 & q_0 & -q_1\\ q_3 & -q_2 & q_1 & q_0\end{bmatrix}\begin{bmatrix}0\\ \omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \quad \cdots (4)$$

Therefore, the nonlinear spacecraft kinematics equations of motion can be represented by the quaternion as follows [17],

$$\begin{cases}\dot{\boldsymbol{q}} = -\frac{1}{2}\omega \times \boldsymbol{q} + \frac{1}{2}q_0\omega\\ \dot{q}_0 = -\frac{1}{2}\omega^T\boldsymbol{q}\end{cases} \quad \cdots (5)$$

Further simplification can be done using the fact that

$$q_0 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2} \quad \cdots (6)$$

we will have

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\sqrt{1-q_1^2-q_2^2-q_3^2} & -q_3 & q_2\\ q_3 & \sqrt{1-q_1^2-q_2^2-q_3^2} & -q_1\\ -q_2 & q_1 & \sqrt{1-q_1^2-q_2^2-q_3^2}\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} \cdot\ \cdot\ (7)$$

$$= \frac{1}{2}\ \boldsymbol{Q}(q_1, q_2, q_3)\omega \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ (8)$$

$$= \ \boldsymbol{g}(q_1, q_2, q_3, q\omega)$$

It is easy to verify that,

$$\det \boldsymbol{Q}(q_1, q_2, q_3) = \det\begin{pmatrix}\sqrt{1-q_1^2-q_2^2-q_3^2} & -q_3 & q_2\\ q_3 & \sqrt{1-q_1^2-q_2^2-q_3^2} & -q_1\\ -q_2 & q_1 & \sqrt{1-q_1^2-q_2^2-q_3^2}\end{pmatrix} \cdot (9)$$

$$\det \boldsymbol{Q}(q_1, q_2, q_3) = \frac{1}{\sqrt{1-q_1^2-q_2^2-q_3^2}} \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ (10)$$

Hence, $\boldsymbol{Q}(q_1, q_2, q_3)$ is always a full rank matrix except for when $\boldsymbol{\alpha} = \pm\boldsymbol{\pi}$.
This means that unless $\boldsymbol{\alpha} = \pm\boldsymbol{\pi}$, the kinematics equation of motion using reduced quaternion representation can be simplified from **(8)** to **(9)**.

The main advantages of using reduced quaternion instead of the original one is the following:

- The system dimension is reduced from 7 to 6, yielding a simpler model
- The linearized system is controllable
- The stability analysis can be directly conducted based on the linearized system
- All closed loop eigenvalues can be assigned to any position by appropriate feedback control law because the linearized system is controllable.

3.3.1.2 General Satellite Dynamic Equation of Motion

Let ***J*** be the matrix of a spacecraft body mass moment of inertia defined by [17]

$$\mathbf{J} = \begin{bmatrix} \mathrm{J}_{11} & \mathrm{J}_{12} & \mathrm{J}_{13} \\ \mathrm{J}_{21} & \mathrm{J}_{22} & \mathrm{J}_{23} \\ \mathrm{J}_{31} & \mathrm{J}_{32} & \mathrm{J}_{33} \end{bmatrix} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (11)$$

$\omega_I = [\omega_{I_1} \quad \omega_{I_2} \quad \omega_{I_3}]^T$ be the angular velocity vector of the spacecraft body with respect to the inertial frame, represented in the spacecraft body frame, $\boldsymbol{h}_I$ be the angular momentum vector of the spacecraft about its center of mass represented in the inertial frame, $h = \mathbf{J}\omega_I$ be the same vector of $\boldsymbol{h}_I$ but represented in the body frame, **m** be the external torque acting on the body about its center of mass. Then, we have [17]

$$\boldsymbol{m} = \left(\frac{d\boldsymbol{h}_I}{dt}\right)\Big|_b \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (12)$$

$$\boldsymbol{m} = \left(\frac{d\boldsymbol{h}_I}{dt}\right)\Big|_b = \left(\frac{d\boldsymbol{h}_I}{dt}\right) + \omega_I \times \boldsymbol{h} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (13)$$

$$\left(\frac{d\boldsymbol{h}}{dt}\right) = \mathbf{J}\dot{\omega}_I = -\omega_I \times \mathbf{J}\boldsymbol{w}_I + \boldsymbol{m} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (14)$$

The external torques m are normally composed of (a) disturbance torques td due to gravitational, aerodynamic, solar radiation, and other environmental torques in body frame, and is expressed by [17],

$$\mathbf{t}_d = \left[t_{d_1}, t_{d_2}, t_{d_3}\right]^T \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (15)$$

and (b) the control torque u expressed by [17]

$$\mathbf{u} = [u_1, u_2, u_3]^T \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (16)$$

Therefore, the satellites' dynamic representation is [18]

$$\mathbf{J}\dot{\omega}_I = -\omega_I \times (\mathbf{J}\omega_I) + \mathbf{t}_d + \mathbf{u} = -\mathbf{S}(\omega_I)(\mathbf{J}\omega_I) + \mathbf{t}_d + \mathbf{u} \cdot \cdot \cdot \cdot \cdot (17)$$

However, this equation is not preferred when dealing with a Reaction Wheel System (RWS), this information was obtained from practical knowledge by members in YahSat Lab.

The equation used to model the Satellite's Dynamics is the following,

3.3.1.3 Dynamics of satellite with reaction wheels

From Euler's moment equation we know that,

$$\underline{\dot{h}}_{TOT} + \underline{\omega}_B^{\times}\underline{h}_{TOT} = \underline{T}_{ex}^{B} \quad \cdots\cdots\cdots\cdots\cdots\cdots (18)$$

Where,

- $\underline{h}_{TOT} = \sum_J^N I_J\omega_J$ is the summation of all (N) parts within the satellite.
- ω_b is the angular velocity vector of the satellite's body with respect to the inertial frame.
- $\underline{T}_{ex}^{B}$ is the total external torque.

It is safe to say however that,

$$\underline{h}_{TOT} = \underline{h}_B + \underline{h}_{rw} \quad \cdots\cdots\cdots\cdots\cdots\cdots (19)$$

And that,

$$\underline{h}_{rw} = \underline{J}_{rw}\underline{\omega}_{rw} \quad \cdots\cdots\cdots\cdots\cdots\cdots (20)$$

$$\underline{J}_{rw} = \begin{bmatrix} J_{rw,x} & 0 & 0 \\ 0 & J_{rw,y} & 0 \\ 0 & 0 & J_{rw,z} \end{bmatrix} \quad \cdots\cdots\cdots\cdots\cdots\cdots (21)$$

And that,

$$\underline{h}_B = \underline{J}_B\underline{\omega}_B \quad \cdots\cdots\cdots\cdots\cdots\cdots (22)$$

This equation however, is found to be more compatible representing a satellite with a reaction wheel system on board. Accordingly, the members decided to proceed with this equation to represent the dynamics of the current system in the SIMULINK model.

3.3.1.4 PD Control Law

PD controllers are used to send commands that helps orient the CubeSat to the required position for Earth pointing mode or any other different type of pointing modes. The equation of the PD controller is as follows

$$T_c = -K_p q_{ob} - K_d \omega_{ob}^b \cdots \cdots \cdots \cdots (23)$$

Where,

- K_p is the proportional gain
- K_d is the derivative gain
- q_{ob} is the quaternion error between the desired and determined quaternion
- ω_{ob}^b is the angular velocity in the body frame with respect to the orbit frame

Both gains can be found using multiple approaches but the most common one is trial and error in which we will be implementing after the construction of the model.

The following figure show the closed control loop used to simulate the PD controller and implement pointing:

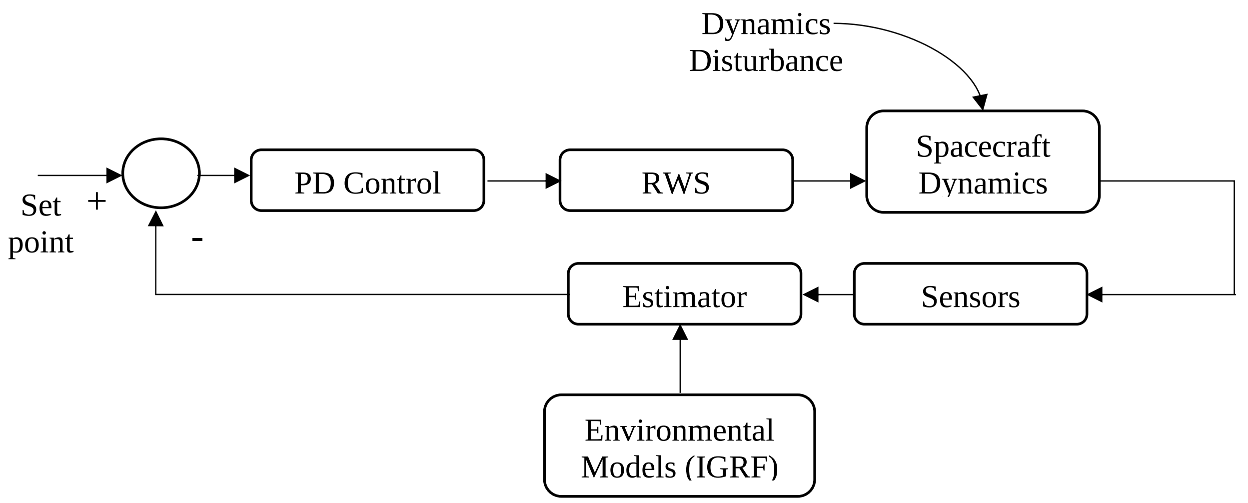


Figure (10): Closed Control loop for changing attitude

3.3.1.5 Estimation of The Environmental Disturbances

Quantifying the possible external disturbance torques that might occur during mission lifetime due to natural phenomena is important when designing an Attitude Control subsystem for a CubeSat mission.

Since this analysis have been done from the first few months of this project and was highlighted in many major submitted reports, we decided to keep this section a minimal with the overall results and summary of calculations. A detailed calculation process is still presented in the Appendix for the reader's reference, it is safe to say that many numerical values were taken from the 2U previous CubeSat mission "Dhabi Sat" data and other bibliography. In general, these torques will act in different directions (represent vector quantities). Assuming a conservative approach (all torques are constant and acting in the same direction) then sum the magnitudes as follows,

Table (16): disturbance torques in their worse-case condition magnitudes are listed.

Disturbance Type	**Definition**	**Extreme Case Assumptions**	**Magnitude (N.m)**
Aerodynamic	Satellite is affected by drag forces caused by the atmospheric density this type of drag creates a disturbance torque generated by the location difference between the aerodynamic center of pressure and the center of mass	• Coefficient of drag is assumed C_D=2. • C_g is assumed to be at geometrical center.	4.396×10^{-9} $(N \cdot m)$
Magnetic	The Earth magnetic field creates a disturbance torque on the satellite that results in magnetic torque	• Spacecraft residual magnetic m=0.01 A m^2	5.0855×10^{-7} $(N.m)$
Gravity Gradient	satellite will encounter a more significant attraction in the lower side of the satellite than the upper side. This difference of attraction results in a torque that rotate satellite to align its minimum inertia matrix with the local vertical	• $\sin(2\theta) = 1$	1.270×10^{-8} $(N.m)$
Solar Pressure	The solar radiation is based on the momentum that have a high perturbation effect on the satellite based on the altitude. The higher the altitude, the higher the solar radiation effect	• incidence is normal to the surface, i = 0 • C_g is assumed to be at geometrical center. • reflection facto $\rho = 0.6$	1.45×10^{-9} $(N.m)$
Total Disturbance= 5.27046×10^{-7} $(N.m)$			

3.3.1.6 Flywheel Inertia

A flywheel sizing is a necessity and a must when it comes to designing a RWS, after all, the mechanics behind the flywheel is what will impose the torque into the CubeSat body. We know from before that a single reaction wheel is nothing more or less than Flywheel attached to the shaft of a brushless dc motor. Therefore, the specific dimension of such mechanism is crucial and need to be dealt with carefully.

To design a compatible flywheel, we computed the following numerical analysis,
Specifications of the flywheel

- Material: PLA (only material available for accessible 3d printing in Uni)
- Percentage: 95%
- Density of PLA = 1.24 $\frac{g}{cm^3}$ = 1240 $\frac{Kg}{m^3}$

we know from before that the General Satellite Dynamic Equation of Motion is,

$$Torque = \frac{dh}{dt} = J\dot{\omega}_I = -\hat{\omega}_I \times \left(\hat{J}\hat{\omega}_I\right) + T_d + \hat{u} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (24)$$

And that KU-Sat has the following design dimensions,

Table (17): KU-Sat hardware Model Specifications

KU-Sat Specifications	
Mass	2.66 kg
length	2 cm
width	1 cm
CoG location	Origin (0,0)
CP location	(0,1)
Moment of inertia	$I_{xx} = I_{yy}$ $= 11083\ kg.mm^2\ I_{zz}$ $= 4433\ kg.mm^2$
Number of units	2U

The moment of inertia however was computed by a simple approach. To ease the moment of inertia calculation process, we assumed the structure to be an empty 2-unit box with the same dimensions as KU-Sat (thickness, height… etc) . The weight is set constant and equal to 2.66 *Kg.*

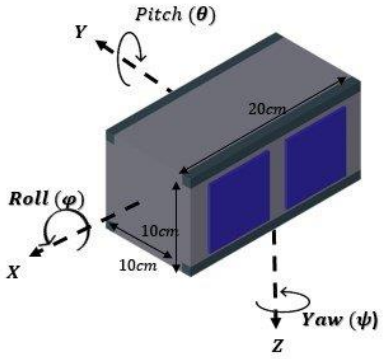


Figure (11): KU-Sat hardware Model Specifications

Calculations of KU-Sat's moment of inertia:

$$J_{xx} = J_{yy} = \frac{1}{12} m(h^2 + d^2) \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (25)$$

$$= \frac{1}{12}(2.66)(0.2^2 + 0.1^2) = 0.011083\ (kg.m^2)$$

$$J_{zz} = \frac{1}{12} m(w^2 + d^2) \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (26)$$

$$= \frac{1}{12}(2.66)(0.1^2 + 0.1^2) = 0.004433\ (kg.m^2)$$

And the moment of inertia matrix can be presented as,

$$J = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (27)$$

$$J = \begin{bmatrix} 0.011083 & 0 & 0 \\ 0 & 0.01183 & 0 \\ 0 & 0 & 0.004433 \end{bmatrix}$$

Control Torque Calculations (u)

Assumptions made:

- For a trim condition (non-rotating) the torque on the CubeSat body was sat to be = 0
- The time to finish a full maneuver $t = 400\ s$. This was taken from a previous paper of a similar CubeSat as KU-Sat and was considered as a reasonable period of operation [18]

- $\widehat{\omega}_I = \begin{bmatrix} 0.0872665 \\ 0.0872665 \\ 0.0872665 \end{bmatrix}$ which was taken from YahSat's Dhabi-Sat data.

Therefore,

$$0 = \begin{bmatrix} 0.0872665 \\ 0.0872665 \\ 0.0872665 \end{bmatrix} \times \begin{bmatrix} 0.01183 & 0 & 0 \\ 0 & 0.01183 & 0 \\ 0 & 0 & 0.004433 \end{bmatrix} \begin{bmatrix} 0.0872665 \\ 0.0872665 \\ 0.0872665 \end{bmatrix} + 5.27046 \times 10^{-7} + u$$

$$0 = \begin{bmatrix} 0.0872665 \\ 0.0872665 \\ 0.0872665 \end{bmatrix} \times \begin{bmatrix} 0.001032363 \\ 0.010323626 \\ 0.000386852 \end{bmatrix} + 5.27046 \times 10^{-7} + u$$

$$u = \begin{bmatrix} 0.0008671 \\ -0.00005633 \\ -0.0008108 \end{bmatrix} - 5.27046 \times 10^{-7} = \begin{bmatrix} 0.0008666 \\ -0.00005686 \\ -0.0008113 \end{bmatrix}$$

Finding the Required Dynamic Range (H_{req})

$$H_{req} = u \times t = \begin{bmatrix} 0.0008666 \\ -0.00005686 \\ -0.0008113 \end{bmatrix} \times 400s = 3.23984 \times 10^{-3}\ Nms$$

Calculating the Flywheel Inertia (I_{zz})

The following equation was used in calculating the flywheel inertia in Drive-Sat CubeSat and was also found in multiple academic references [17].

$$I_{zz} \geq \frac{30 \times H_{req}}{\pi\, n_{max}} - I_{motor} \quad \cdots\cdots\cdots\cdots\cdots\cdots (28)$$

$$I_{zz} \geq \frac{30 \times 3.23984 \times 10^{-3}}{4000\,\pi} - 8.1 \times 10^{-7}$$

$$I_{zz} = 7 \times 10^{-6}\ Kgm^2$$

This methodology might have not been 100% accurate but it produced a satisfactory level of results as you shall see later in the coming sections. However, it was a great start to see how the flywheel behave when attached to a brushless dc motor and it allowed us to explore other possibilities of enhancement and approaches. But as far as this project goes, this is the method that was implemented to size the flywheel prototype.

3.3.1.7 Coordination System

In this subsection the coordinate system selected to represent the attitude information of KU-Sat is presented. Such coordinate is used to define vectors in R3 so that we describe the CubeSat's position and dynamics. At least one reference frame alongside a body frame is needed for a full attitude knowledge. The following two frames are used to represent the position of KU-Sat when in orbit

1. **Earth Centered Inertial Reference Frame (ECI)**

This earth centered inertial reference frame $\mathcal{F}_i = \{ \hat{\imath}_x , \hat{\imath}_y , \hat{\imath}_z \}$ has its origin in the center of the Earth and is an inertial, non-rotational frame in which Newton's laws of motion apply. $\hat{\imath}_z$ points at 90° angle relative to the Earth's equatorial plane where it coincides with the Earths rotational axis and continues through the celestial North Pole. $\hat{\imath}_x$ points in the vernal equinox vector direction, which is the vector pointing from the center of the Sun to the center of the Earth at the vernal equinox. The vernal equinox is a time of the year when the Earths orbital plane as it rotates around the Sun coincides with the equatorial plane, i.e. the center of the Sun lies in the same plane as the Earths equator. Finally, $\hat{\imath}_y$ completes the three axis orthonormal frame according to the right-hand rule [19].

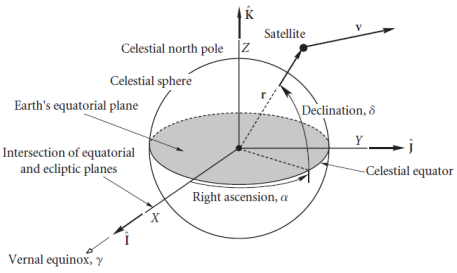


Figure (12): ECI

2. **Body Reference Frame (BRF)**

The origin of the body reference frame $\mathcal{F}_b = \{ \hat{b}_x , \hat{b}_y , \hat{b}_z \}$ is a moving coordinate system which also has its origin at the CubeSat's center of mass. However, unlike the orbit frame it is fixed to the CubeSat, and so it rotates and moves with the CubeSat. For simplification, it's preferable to let the unit vectors coincide with the CubeSat's axes of moments of inertia. $\hat{b}_x$ is referred to as the roll axis, $\hat{b}_y$ is the pitch axis, and $\hat{b}_z$ is the yaw axis [19].

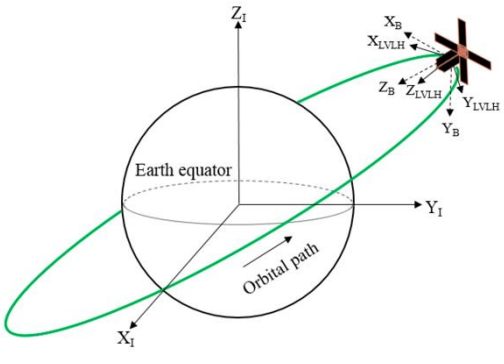


Figure (13): BRF

It is safe to say however that the attitude must be represent by an mathematical attitude parametrization method. In our case, we chose Quaternion method. The mathematical details about this specific method can be found in the Appendix

Orbit Analysis

When dealing with space mission, a person must take into the account the environment of operation in which our system will perform at. To properly address this environment, the orbit, we must first present the orbital element that describes it,

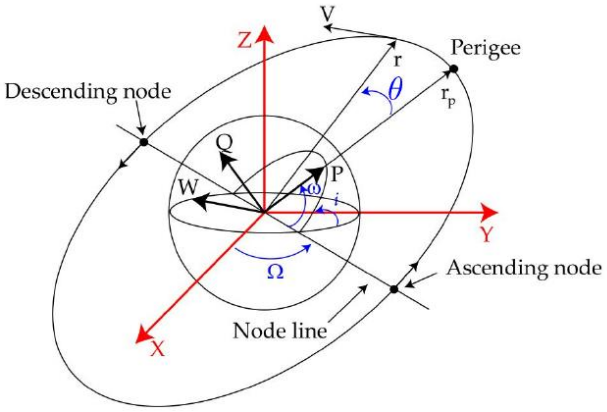
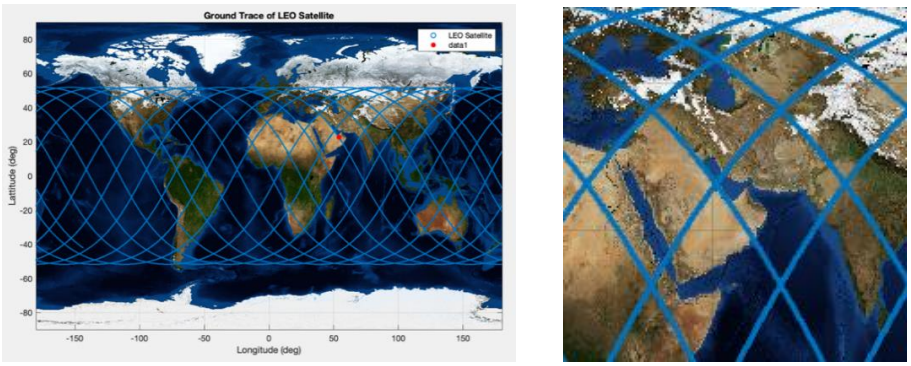


Figure (14): Classical orbital element (Two-line Elements)

Table (18): KU-Sat orbital element

Orbit Parameter	**Symbol**	**Magnitude**	**Unit**
Inclination	i	51.6	Degree
Argument of periapsis	ω	80.5	Degree
Orbital Period	T	92.68	Minutes
Orbit Epoch	14th of May 2019 13:09:29 UTC		
Longitude of ascending node	Ω	29.7	Degree
Mean anomaly at epoch	M	38.2	Degree



Figure(15): Fill earth ground trace of KU-Sat (Left), zoomed in trace of KU-Sat near UAE

The ground trace is propagated via an open-source MATLAB code found in the Appendix. We can clearly see that if the satellite is continuously pointed downward toward the earth, a small slew manoeuvre is necessary to best capture the UAE's land since the ground trace doesn't pass over it directly.

3.3.2 Discrete or Other Approximations

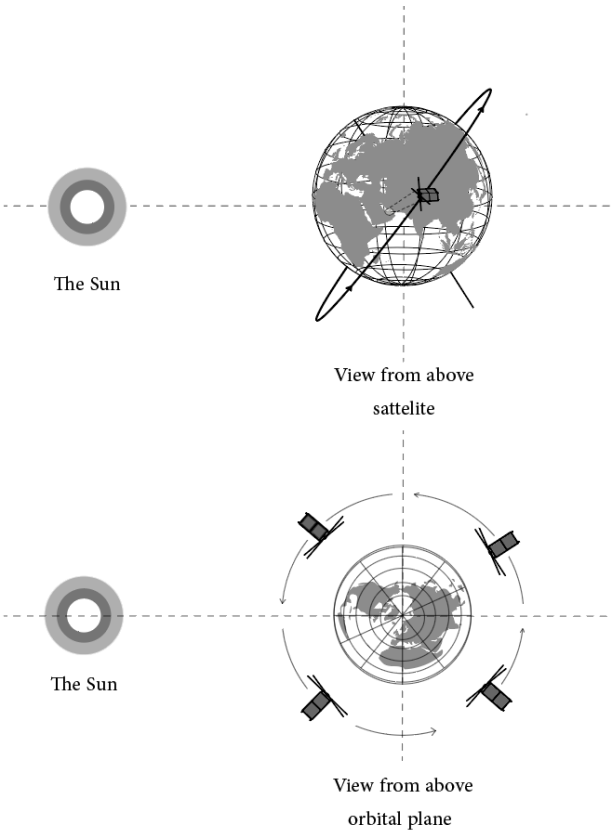


Figure (16): KU-Sat Orbit Representation and Assumption diagram

Table (20): Orbital influence on the ADCS system design

No.	Requirement and assumption established based on the influence of the mission
1	All dominant disturbances in the altitude of deployment must be taking into account when sizing the actuator of the ADCS system
2	For the sake of the analysis, we will assume that the satellite has a zero-deviation angle from the vertical no matter the position unless a controlled slew manoeuvre is applied
3	Satellite orientation and slew manoeuvre will be taken into consideration and estimated based on the ground trace of the orbit
4	Orbit propagation will be neglected, the satellite is assumed to be in the same orbit disregarding the changes in the orbital period

Chapter 4. CONCEPT GENERATION and DESIGN EVALUATION

4.1 Introduction

This Chapter highlights the process of selecting and evaluating the best ADCS system design that is compatible with the mission objective behind KU-Sat project. We have considered different choices of sensors and actuators and evaluated each and every one of them. At the end, the methods that best met the design requirements was approached. The process shows the comparison and evaluation of the designs and the reasoning behind choosing the appropriate and final ADCS design. At the end of this chapter, a discussion of future work will be suggested. The chosen design was selected according to the mission requirements, performance, availability, and the provided budget. Moreover, it was carefully researched and designed, and most importantly was approved by the project supervisors.

4.2 Different Design options

Four designs were proposed for KU-Sat ADCS system. They differ in the quantity and types of actuators, sensors, micro-controllers, and control modes. The tables below show each design's specifications and architecture, as well as a comparison between the designs.

Table (21): Alternative ADCS design selections

Element considered	Design 1	Design 2	Design 3	Design 4
Actuators				
Reaction Wheels	✓	✓		
Self-developed reaction wheels using brushless DC motors			✓	✓
Magnetorquers		✓		✓
Sensors				
Sun sensor	✓	✓	✓	✓
Magnetometer		✓	✓	✓
Gyroscope	✓	✓		✓
Micro-controllers				
Arduino Nano		✓		✓
ATxmega256A3U	✓			
Arduino Uni			✓	

Control Modes				
Camera Pointing Mode	✓		✓	✓
De-tumble Mode				
Reaction wheels De-saturation Mode		✓		✓
Slew Maneuver Mode	✓	✓		✓

Table (22): Comparison between the different designs

Comparison Factors	Design 1	Design 2	Design 3	Design 4
Cost	Exceeds the budget	Exceeds the budget	Within the budget	Within the budget
Feasibility	Feasible	Feasible	Feasible	Feasible
Mission requirements	Meets the mission requirements	Meets the mission requirements	Meets the mission requirements	Meets the mission requirements
Customer needs	Meets the customer needs	Meets the customer needs	Does not meet the customer needs	Does not meet the customer needs
Performance	Exposed to saturation	Good	Exposed to saturation	Good

Table (23): ADCS designs analyzation

Attitude Determination and Control System Design Options		
Design No.	**Reasons of Selection**	**Reasons of Elimination**
1	- Meets the mission objective and requirements - The appropriate types of control modes that will help deliver the mission	- Reaction wheels are very expensive for an educational ADCS prototype design and we are limited to a limited budget - Not enough sensors to determine the attitude of the CubeSat - Exposed to saturation where the system comes to a point where it has stored the max momentum it can

2	- Meets the mission objective and requirements - Enough number of sensors to determine the attitude of the CubeSat	- Reaction wheels are very expensive for an educational ADCS prototype design and we are constrained with a limited budget - Does not have camera pointing mode while the main mission involves pointing a camera toward Earth
3	- Alternative method to replace the reaction wheel by using Brushless DC motor and therefore the design will be within the budget	- Exposed to saturation where the system comes to a point where it has stored the max momentum it can - Not enough sensors to determine the attitude of the CubeSat - Not enough control modes to control the CubeSat
4	- Alternative method to replace the reaction wheel by using Brushless DC motor and therefore the design will be within the budget - Enough sensors to determine the attitude of the CubeSat - The appropriate types of control modes	

Therefore, As shown and proven in the tables above, design four was our best design that will help deliver and achieve the mission of KU-Sat successfully and meet its objectives.

4.2.1 The Final Design Architecture

Sensors and actuators are essential aspects of attitude determination and control. Choosing the best Models of these components is important considering its effect on the simulation environment for realistic evaluation of performance. Suitable models of sensors and actuators are therefore identified in this Section respectively.

4.2.1.1 Attitude Determination hardware Selection (Sensors)

4.2.1.1.1 Sensors Evaluation

General feedback control has no meaning if sensing is not employed. The attitude of the satellite must be measured and estimated in order to control it. A sufficient number of sensors have been implemented to get the necessary accuracy of the estimated attitude. This section first discusses and evaluates different sensor possibilities and the reasons behind acquiring the chosen sensors for KU-Sat.

Table (24): Advantages and disadvantages of different reference sensor types, if used in a CubeSat [20, 21, 22].

Sensor type	Advantages	Disadvantages
Sun tracker	- Very high accuracy. - Lost in space function.	- Dependent on star identification. - Expensive. - Heavy and large. - High power consumption. - Sensitive to spin rates. - No output when pointing towards the Sun
Sun sensor	- Simple sensor. - Low power consumption - Analog sun sensors are cheap.	- Coarse accuracy. - Disturbed by Earth Albedo. - No output during eclipse. - Digital 2-axis sun sensors are expensive
GPS antenna array	- Resistant against spin rates - Also provides position, velocity and time.	- Expensive. - Large antenna array setup. - High power consumption.
magnetometer	- Simple sensor - Low power consumption. - Very cheap. - Very small - Always available	- Coarse accuracy. - Magnetic field not completely known. - Affected by on-board electronics. - Only applicable in LEO

To determine the attitude of the satellite, at least two linearly independent sensors are needed. For our project we have chosen three sensors which are the sun sensor, the magnetometer and the gyroscope. In order for the sun sensor to work, it needs to have a direct field of view with the sunlight all the time. However, in the eclipse case the sun sensor cannot work due to the absence of the sunlight. Therefore,

the two other sensors; the gyroscope and the magnetometer, can step in and be reliable to calculate and estimate the attitude.

Star trackers and GPS antenna arrays are both very expensive. The accuracy requirement for the mission does furthermore not require the use of e.g. a high precision star tracker. In the case of eclipses, adding a star tracker would definitely be more accurate. However, we did not consider using it for two reasons. First, depending on the manufacturer, the star tracker needs to view at least two or three stars to work. Which means that in the orbit (LEO), there are so many areas where the star tracker will not be able to work. The second reason is that the stars tend to disappear and we do not have the ability to predict the stars' location. Even if the star tracker was chosen to be the main sensor, we would still need to add both the magnetometer and the sun sensor. Hence, the three chosen sensors are more than enough to deliver the job and complete the mission successfully.

4.2.1.1.2 Sensors Model Selection

a. Sun Sensor (Photodiode)

The Sun sensors are in charge of measuring the Sun vector position in the body frame. In our design analysis the type of sun sensor selected for KU-Sat mission is **SLCD-61N8 photodiode.** SLCD-61N8 photodiode has many features, they are at low cost, high reliability, and linear short circuit current. In addition, it estimates an accurate determination of sun-angle. When the sun is not in the field of view, magnetometers can work as a compliment to the sun sensor.

Table (25): Properties of the SLCD-61N8 photodiode [23]

Type	Light Sensing Photodiodes
Half angle	60 [deg]
Short circuit current	170 [μA] (typ)
Open circuit voltage	0.4 [V] (typ)
Reverse dark current	1.7 [μA] (max)
Interface	Analog
Size	3.4x1.3 [mm2] surface mount

The chosen model has been selected based on a similar CubeSat mission, the same as in the AAUSAT-II CubeSat. It is a Commercial Off-The-Shelf (COTS) hardware component and not a space rated component. Such components are usually used in ground systems to estimate the Sun position in

reference to the sensors position. The sensor has a silicon base with the two metallic contacts to obtain the current generated. It has to be adapted and amplified, by using a transresistance amplifier circuit ($\Delta V/\Delta I$), for a voltage Analog-to-Digital converter. This circuit has to be calibrated to obtain the correct light power as seen by the sensors.

b. 3-Axis Gyroscope sensor

Gyro sensors work by measuring vibrations caused by angular rotations. They are small, lightweight, inexpensive and have a low power requirement. They are used to measure the CubeSat's angular rate that must be integrated to determine the orientation and axis of rotation of the whole satellite body.

Table (26): Properties of the gyroscope sensors [24, 25, 26]

Model/Requirements	**IDG-1215 & ISZ-1215**	**MPU 6050**	**MPU 9250**
Type	Angular rate gyroscope	Angular rate gyroscope	Angular rate gyroscope
Range	±67 [deg/s]	±250[deg/s]	±250 [deg/s]
sensitivity	15 [mV/deg/s]	131 (LSB/deg/s)	131(LSB/deg/s)
Non linearity	< 1% of full scale	0.2 %	± 0.5%
Cross axis sensitivity	± 1%	± 2%	± 2%
Interface	Analog	I^2C	I^2C
size	4x5x1.2 [mm^3]	4x4x0.9[mm^3]	3x3x1[mm^3]
Shock tolerance	10.000 [g]	10.000 [g]	10.000 [g]
Operating temperature	-40 to 105 [°C]	-40 to 85 C[°C]	-40 to 85 C [°C]
Cost ($)	4.95	9.80	11.43

The chosen model of gyro sensor was IDG-1215, MPU 9250 and MPU 6050. It was recommended by YahSat lab that we work with 9 axis MPU 9250 for a better data accuracy. MPU 9250 is designed for low cost and high performances sensor. It has more features than MPU 6050 that measure velocity, orientation, acceleration, displacement and earth magnetic field. It consists of three-axis accelerometer, three-axis gyroscope, and three axis magnetometers. They are all combined with each other to excel in precise different attitude determination goals. MPU 9250 is compatible with Arduino that made the testing method easier and output the angular rates in x, y and z direction.

c. 3-Axis Magnetometer sensor

Table (27): Properties of the Magnetometer sensors [27,26]

Model/ requirement	**HMC6343**	**3-Axis Magnetometer (MPU-9250)**
Type	Triaxial digital magnetometer	silicon monolithic Hall-effect magnetic sensor
Range	±2 [G]	±4800 [μT]
Interface	I^2C	I^2C
Sampling frequency	1, 5 or 10 [Hz]	8 [Hz]
size	9x9x1.9 [mm^3] LCC surface mount	3x3x1[mm^3]
Operating temperature	-40 to 80 [◦C]	-40 to 85 [◦C]
Cost ($)	154.95	11.43

Magnetometers Measure the Earth's magnetic field vector local to the satellite. This magnetometer was chosen because of its reliability an accuracy in reading data. Also it comes in a one ship alongside with the a gyroscope and an accelerometer.

4.2.1.2 Attitude Control hardware Model Selection (Actuators)

4.2.1.2.2 Reaction Wheel System (RWS)

a. Selection of Brushless DC (BLDC) Motor

Selection of the Brushless DC Motor was dependent on many factors, some of which are,

- High speed motor
- Low torque
- High inertia rotor
- Low Working volt
- Less electric power losses
- Less thermal management
- Endure launch loads
- Reliability
- Durability of about 2 years
- No magnetic leakage
- Light weight
- Compact

For that purpose, we searched the availability of such superior Faulhaber BLDC Motors that are compatible with the purpose of building the RWS. For the sake of simplicity, we presented the consistent candidate for our project in the following table,

Table (28): Best Candidate for building a RWS using BLDC Motor

Model/Requirements	**1202 004 BH**	**1608 004 BH**	**2209T005S**	**2610T012B SC**
Torque [mNm]	0.16	0.205	0.094	0.025
Temperature Range [ºC]	-30 TO +85	-30 to +85	-30 to +85	-25 to +80
Maximum Velocity [rpm]	40 000	12 000	10 000	7 000
Nominal Voltage [V]	4	3	5	12
Power [W]	0.652	0.116	0.06	1.91
Sensor	Hall Sensor	Hall Sensor	Hall Sensor	Hall Sensor
Cost ($)	82.77	121.45	177	143.99
Availability	Not available	Not available	Not available	Available

Due to various limitation sat by the COVID-19 situation, our BLDC Motor selection was SC Flat Brushless DC Motor because it was the only option available for delivery to the UAE in our limited timeline. However, the other motor may carry a better qualification, the selected one is still of a superior quality and can contribute valuably to the overall outcome of the RWS design. The Datasheet for the selected motor is given in the Appendix**.**

a. Selection of RWS Configuration

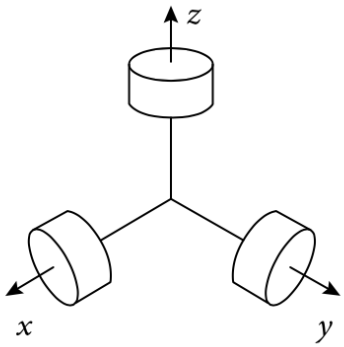


Figure (17): Orthogonal Configuration of Reaction Wheel

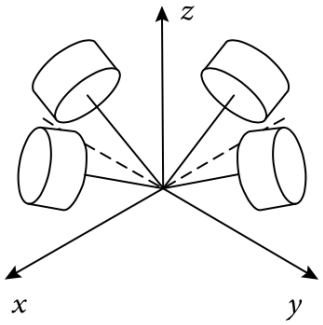


Figure (18): Pyramid Configuration of Reaction Wheel

Nonetheless, this configuration is not as redundant as the pyramid configuration. Still, it meets the requirement and provides us with more time to ensure the system's effectiveness since it is easier to implement and control.

b. 3-Axis Magnetorquer rod

Since the only available 3-axis magnetorquer rods in the market are for space use only, hence, expensive and exceeds the provided budget. We respectively asked YahSat lab to lend us a spare magnetorquer that they have, so that we to try test its data output and learn how to control it. For the sake of the project which is to design an ADCS System for a pointing Control mission that uses RWS System as the main actuator, we decided to not include the 3-axis magnetorquer rod in the produced hardware prototype of the RWS for the desaturation purposes.

Table (29): Properties of the Magnetorquers

Model/ Requirement	NCTR-M002	iMTQ Magnetorquer Board
Operating range	-35°C to +75°C	-40°C to +70°C

Power	200mW	4 No actuation: 175 mW 4 Full actuation (3-Axis): <1.2W
Dimensions	70 mm x 15 mm x 13 mm	95.9 x 90.1 x 17 mm3
Supply voltage	5V	5V
Cost	$ 1200	Provided by the Yahsat Lab

4.3 Key Design Constraints

The key design constraints of the produced ADCS system were taken into consideration with every step from the very beginning. We tried our best to not exceed these limitations that were sat by the respective clients. Since the scope of the project is focused on simulation and hardware testing, mainly the constraints were budget, resource, and time limits.

4.3.1 Budget limitation

One of the most important constraints of this project is the budget, it must not be exceeded. Therefore, an evaluation has been made for choosing the best design that would deliver the purpose of the project and at the same time be within the budget. For our case, and sense the space rated components are expensive, the clients have agreed on using COTS hardware. For example, reaction wheel system was designed by the team using simple components rather than buying the actual reaction wheel system. All the hardware used for the ADCS system are cost of the shelf components.

4.3.2 Resource and time limitations

Some of the hardware need special testing chambers or setup due to their sensitivity which we did not have. Lack of resources have led to the failure of testing the magnetorquer. In addition, we were constrained by time as we think of our project as a huge project that needed more time to execute. However, we succeeded to finish in time and deliver an ADCS system simulation model.

4.4 Key Design Assessment

After evaluating these different aspects of our project, we were able to improve the efficacy of our design step by step until we got to the desired and productive outcomes. As was discussed in this section, several factors had been considered when selecting the ADCS system design, the factors have

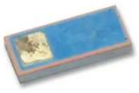
varied between cost, performance, and the availability. Selecting suitable components was necessary in order to meet the project requirements. The project is to develop a full ADCS simulation model that can estimate the attitude of a 2U CubeSat to determine its current and desired position when in orbit. The design fits the requirements and is able to deliver the mission objective. The designed ADCS system can also be improved in the future to perform more control modes and star trucker sensor for more accurate readings data.

Chapter 5. IMPLEMENTATION AND FABRICATION

As per the scope of this project and as specified before, apart from the Simulink model we did an individual testing for each component in the ADCS selected architectural design. Also, this section will include the full implementation and fabrication of sizing and building the orthogonal reaction wheel model. The results of all this implementation are discussed thoroughly in steps per component testing.

5.1 Working Principles

Table (30): ADCS Hardware and their working principle

Hardware	Model	Description
3-Axis Gyroscope	3-Axis Gyroscope (GY-9250) 	Three axis gyroscope that should be placed align to the reaction wheels, that would determine the angular velocity of the CubeSat. Gyroscopes are subject to a noise from space environment that needs to be filtered to give the appropriate results.
3-Axis Magnetometer	3-Axis Magnetometer (MPU-9250) 	Three axis magnetometers aboard the CubeSat would detect the value of Earth's magnetic field strength, resulting in a vector normal to the Earth's magnetic field at the CubeSat's current orbital position. The magnetometer is limited to the earth magnetic field so it fits with our mission in LEO.
Sun sensor (Photodiodes)	Solderable Planar Photodiode (SLCD-61N8) 	Multiple sun sensors read the current from multiple position on the CubeSat's body, knowing the sensor position that reads the highest current value results in a sun vector. This vector alone would not be sufficient in determining the exact location of the CubeSat. It is important to note that the sun sensor would be inadequate during solar eclipse. So having other sensors will help such as the gyroscope and the magnetometer.
3-Axis Magnetorquer Rod	iMTQ Magnetorquer Board 	three magnetorquer axis is placed so that its perpendicular to each axis around the CubeSat's X-, Y- and Z-axes. When the magnetorquers are turned on, they will create a magnetic field that interacts with the Earth's magnetic field, that will allow the CubeSat to desaturate.
Reaction Wheel Using (COTS) BLDC Motor	Faulhaber 2610T012B SC Flat Brushless DC Motor 	Three reaction wheels are mounted orthogonally. The reaction wheel is designed using BLDC Motor and 3D printed flywheel. The reaction wheel works as an actuator which relies on the conservation of angular momentum. So that if the reaction wheel is rotating CW the CubeSat will rotate CCW.

5.2 Component Testing and Optimization

5.2.1 Attitude Determination Individual Component Testing

5.2.1.1 Gyroscope Testing

we designed a simple setup to test the Gyroscope's reading and how reliable can they be in comparison with the actual RPM induced by a body calculated directly by a device called Tachometer. The testing circuit included the following to power out the gyroscope for data extraction,

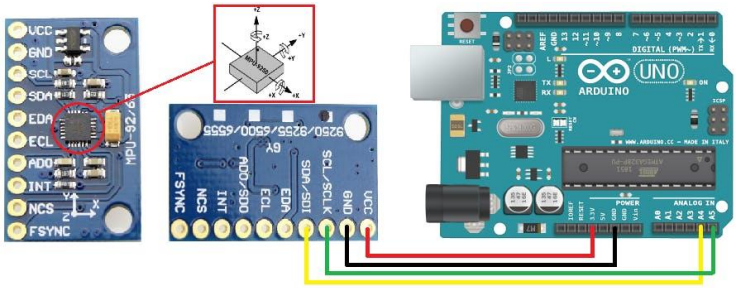


Figure (19): Gyroscope testing setup

The test included the use of a rotating desk, the gyroscope was mounted on top and all the connection above have been made. After that, we ran the rotating desk at a certain speed and started extracting the (x-axis) angular velocity readings while simultaneously pointing the tachometer on a reflective surface attached to one of the rotating desk sides for a real-time measurement. Two set of angular speed have been tested namely, 3.8 RPM and 5.6 RPM. Both were repeated 3 times for an accurate testing result. The full reading data from the gyroscope is presented in the Appendix

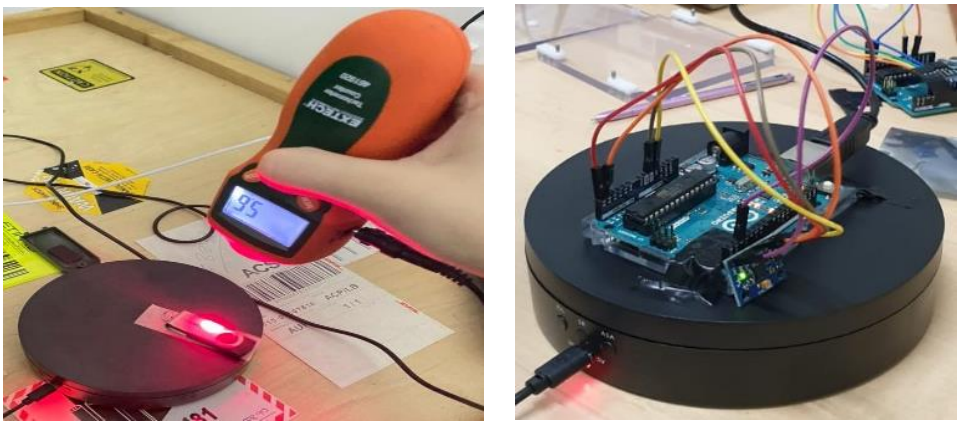


Figure (20): Rotational speed measurement using tachometer (Left), MPU 9250 connection setup (Right)

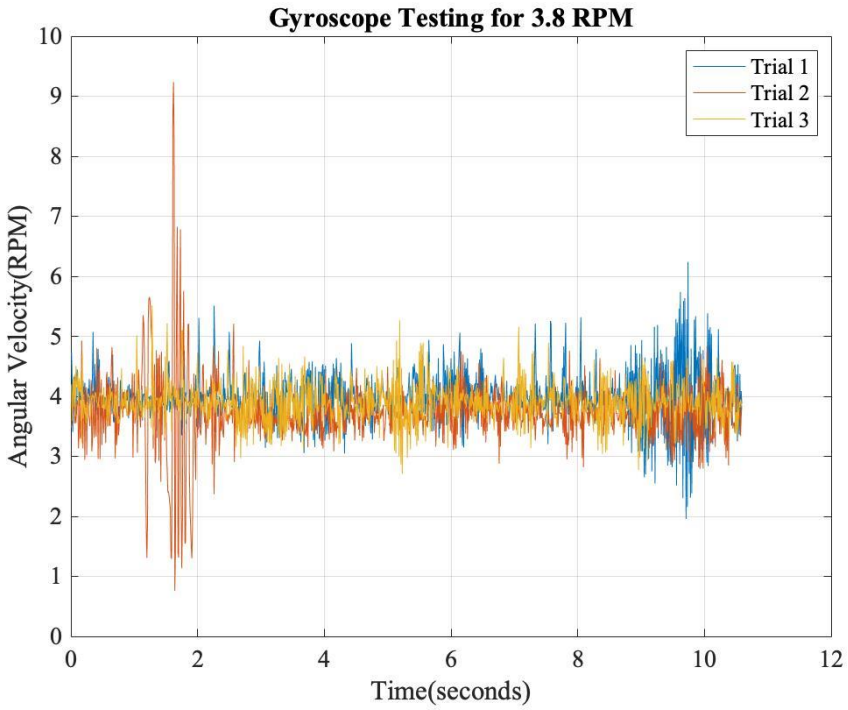


Figure (21): Experimental data of the Gyro (testing for 3.8 RPM readings)

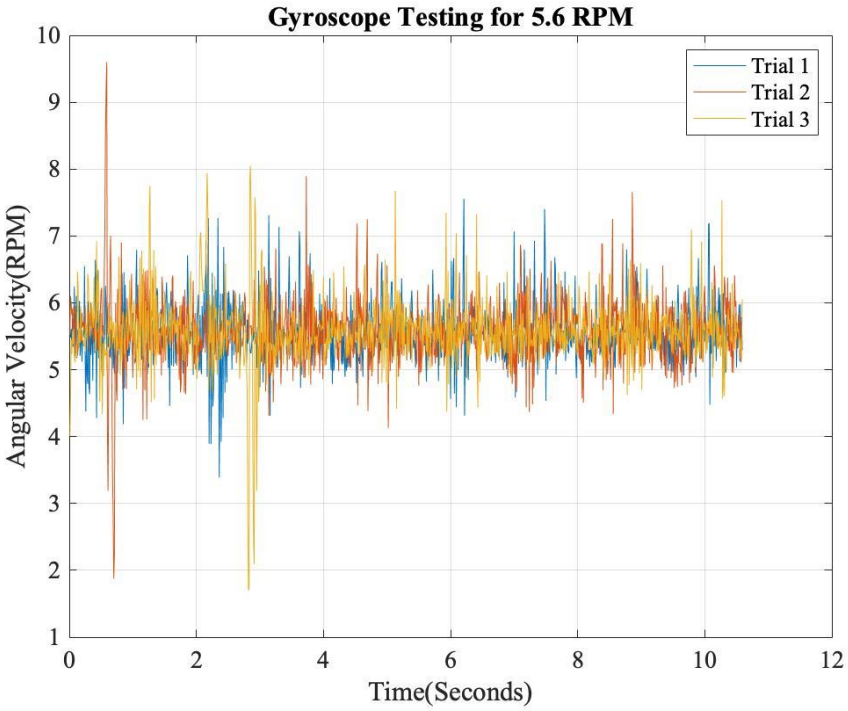


Figure (22): Experimental data of the Gyro (testing for 5.6 RPM readings)

Clearly from the two plots, reading appears to be within an acceptable range but with a lot of noises. An overall SIMULINK model was generated to filter out these data.

5.2.1.2 Gyroscope Filtering

The MEMS gyro Simulink model is to represent the real physical hardware with our system design. The input data that developed the gyro model are obtained from MPU 9250 datasheet and gyro experimental output data.

Table (31): simulation assigned values [28]

Data obtained from MPU 9250 data sheet	
Gain value (Sensitivity Scale Factor)	0.000763
Sample time for the gyro (Total RMS Noise)	1/92 s
Rate noise density	0.0067 degree/ second
Cut off frequency	25 Hz
Assumed Data	
Natural frequency of the gyro dynamics	2000HZ
Damping Ratio	Sqrt(2)/2

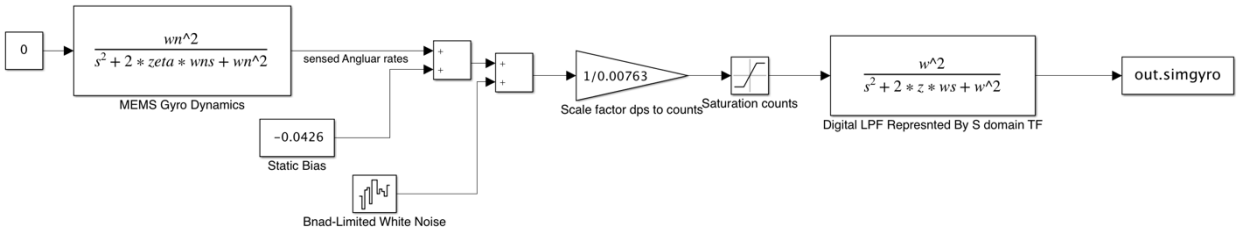


Figure (23): Model represents the real physical hardware of MPU 9250

The Simulink model can be used to test how linear control system design in a nonlinear simulated environment, which is handy in case if we want to control parameters without loading new software in our device and tested physically over and over again. Where it can linearize a non- linear model over and over again in order to develop control law to check for stability as well.

- Our model will include an input and an output. The real input into mems gyro is the angular rates and the output is a count or some value that can be represented by 16-bit integer

- The Absolute simplest conversion between angular rates and counts are just simple gain who is value is a conversion factor between the two where the gain value obtained from the data sheet which called sensitivity (gain value = 1/ sensitivity) to converted from degree per second to counts.

- The dynamic aspect of MEMS gyro it has a small vibrating mass, when the gyro is rotated the mass experience a small Coriolis force which displaces the vibrating mass from its original path. The gyro uses capacitance to sense this displacement in output proportional to number of counts. If the gyro is rotated back and forth faster than the natural frequency, then the output will experience a drop in gain and some phase lag but for our model the natural frequency is generally high. Generic Second order transfer function in line between the input and the scale factor to account for this dynamic.

- The gyro and all sensor in general are subjected to a static bias and sensor noise. The static bias is the average output when the gyro is not rotated, so we can just add a constant value to our input.

- The noise will be modeled as band limited wide noise, which means the noise will have equal power at all frequencies and this pretty close of how real noise exists into mems gyro. Band width block will be sample time of the gyro 1/92. We can get the number for the amount of the noise in our system directly from the data sheet called angle random walk or rate noise density 0.0067 HZ. To go from a fast orient transform version of rate density and noise power, the noise power must be squared.

- The Saturation block must be added before the output because the gyro can output at infinite rate, there only 16 bit of data no matter what is input data. The used gyro has an option for built in low pass filter, we can select the cut off frequencies by setting a register for the gyro.

- MPU 9250 has the option for built in a low pass filter by selecting a cut off frequency. Since it not clears how exactly the low pass filter is set up within the gyro, so it will be approximated as second order filter with a cut off frequency of 25 HZ.

- The digital filter which is Z domain but it will be modeled in S domain since the rest model in S domain and also because the difference between the two are really small in frequencies where it will be operating our gyro. Therefore, the low pass filter will be placed after the saturation block which mean it can potentially get values larger than 16 bits. The saturation block should be placed at the end and the output is a data is doubled in this model. Real gyro will output

only signed integer, so to make the model complete we must have a block at the end that convert the value into a signed integer.

From the previously discussed experimental data that contained noises namely figure 21 and figure 22. To remove these noises, we added a bias and noise block to our model. The bias value is obtained by getting the mean value for the signal.

The bias modelled as degree per second, it must be converted before we added to the model. A signal with a bias removed to check the frequency content with noise We will run a fast transform for this variable real gyro no bias using built in function called myfft function to generate the frequency content.

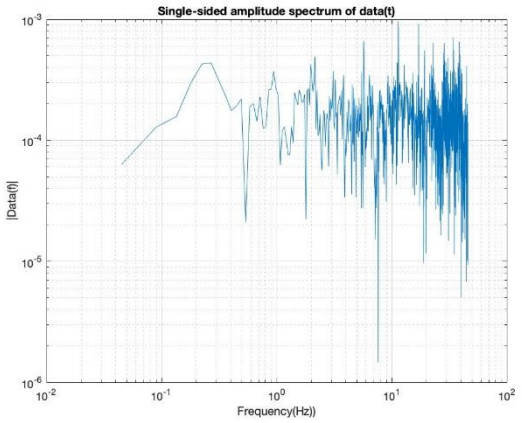


Figure (24): Frequency content of Real Gyro no bias

A low pass filter is added with a cut off frequency of 25 Hz, the rate noise density will be found using Allan variance function [29] that is equal to 0.0067 degree/ second.

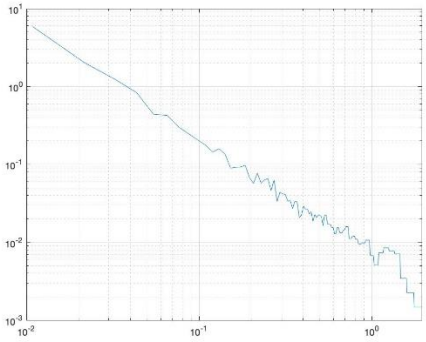


Figure (25) : Allan variance for non-bias data

Finally, we have our simulated gyro data where we can compare real gyro data to see how well our model does. The final plot will compare both real and gyro model and we will check how to

linearize the model so you can use it to design and analyse closed control system. A Set up closed loop control system with a reference controller and a plant and we feed back the output using the sensor.

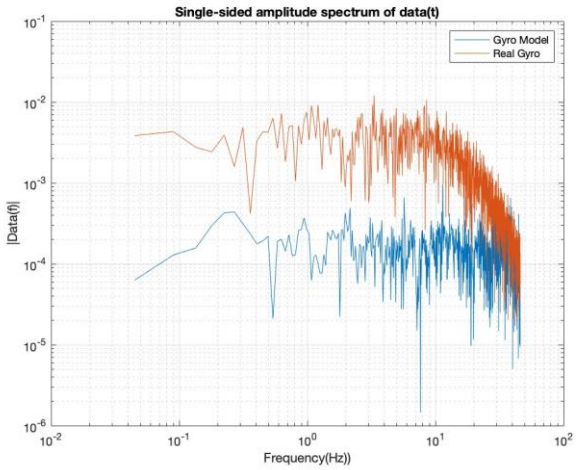


Figure (26) : Real gyro data versus experimental Gyro model

Comments:

As shown in figure 39, the real gyro data obtained using simulation data has higher values the experimental gyro model this might happen due to experimental error data.

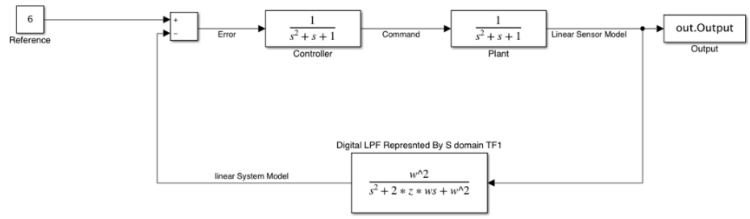


Figure (27): Closed loop control system for linear sensor model using low pass filter

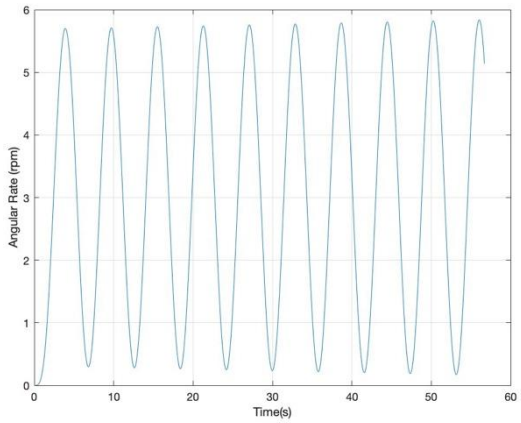


Figure (28): linearized output data using low pass filter

5.2.1.3 Magnetometer Testing

The testing of the Magnetometer reading was done simultaneously with the gyroscope as the same chip contains a 3-Axis Gyroscope, a 3-Axis Magnetometer and a 3-Axis Accelerometer. However, since we are not changing altitude, the reading of the magnetometer remained constant throughout the entire experiment.

5.2.1.4 Sun Sensor Testing

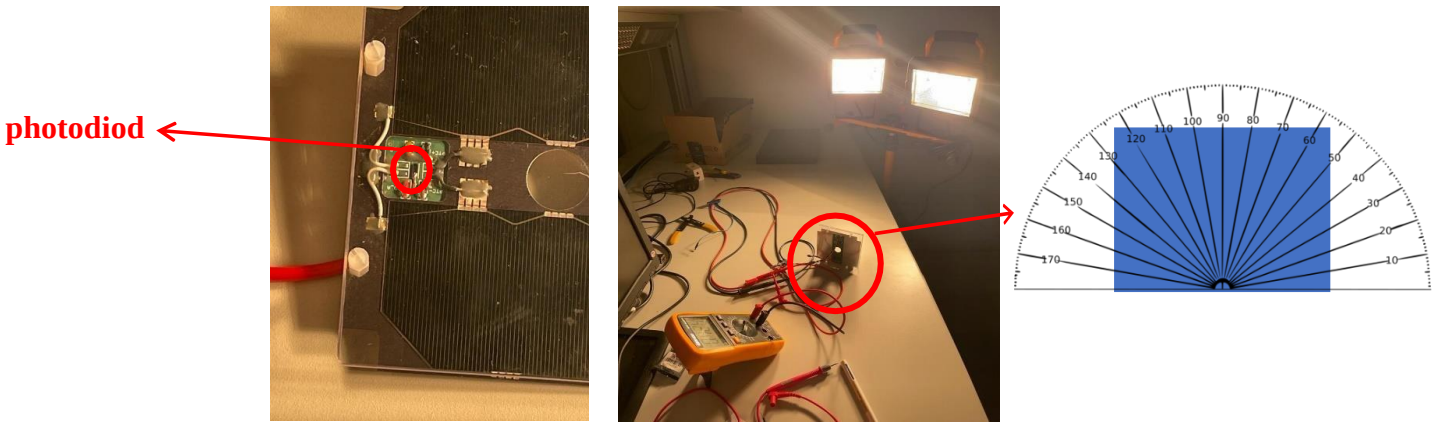


Figure (29): Diagram showcasing the setup of testing the single photodiode

The output readings of Voltage/Current in a sun sensor (photodiodes) vary with the incidence angle. This known relation can be used to estimate a sun vector (direction vector pointed toward the source of light) just like it operates in space where we estimate the direction vector pointing toward the source of light (sun). Accordingly, this vector helps the process of determining the attitude position of the CubeSat relative to the sun by using TRIAD algorithm. We conducted a testing experiment to find the voltage reading of a single photodiode while varying the incident angle or in other words the position of the source of light. A single photodiode was connected to a Multimeter and a light source to imitate the light coming from the sun. The experiment was conducted as follows,

In the experiment, the variation of the incident angle relative to the photodiode reading was tested. As we mentioned before, as we orient the photodiode away from the source of light the reading of the voltage decreases. The output voltage readings resulting from varying the angle from 0° to 100° for our selected piece are shown in the figure below,

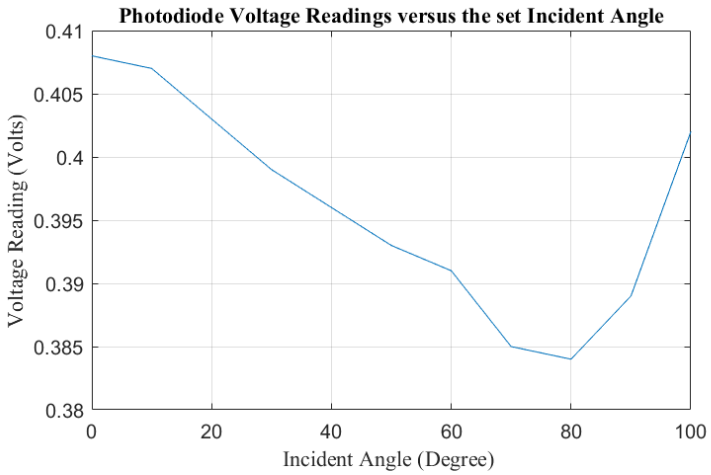


Figure (30): Results of testing a single photodiode, voltages versus angle of incidence graph

From the graph we can see clearly that when a photodiode is on the face of the direct light (incident angle = 0 degree), the voltage is max. The results also show that the voltage decreases to a minimum at angle 80° where the light is barely visible. After 80° the angles are repeated such that the photodiode starts to come closer again to the light source and the voltage starts to increase again which can be directly detected by the sudden increase in voltage readings.

Since the selection of attitude determination components in our ADCS design was limited to a sun sensor, magnetometer and a gyroscope, the number of sun sensor needed to be mounted had to be carefully studied to ensure that the sun is seen by at least a certain number of photodiodes at the same time anywhere in space. The proof of how many total numbers of photodiodes needed was done via a simple algorithm discussed in the following subsection.

5.2.1.4.1 Estimation of how many sun sensors needed

To estimate the number of sun sensor needed for our mission we designed a simple algorithm with the following steps To estimate the number of sun sensor needed for our mission we designed a simple algorithm with the following steps, starting by finding the position of the satellite through the orbit at different time, followed by finding the location of the sun. Through the orbit we have to know if the sun is in the field of view of the sun sensors in order to know the attitude of the CubeSat. The flow chart below descried the implemented MATLAB code that does this work:

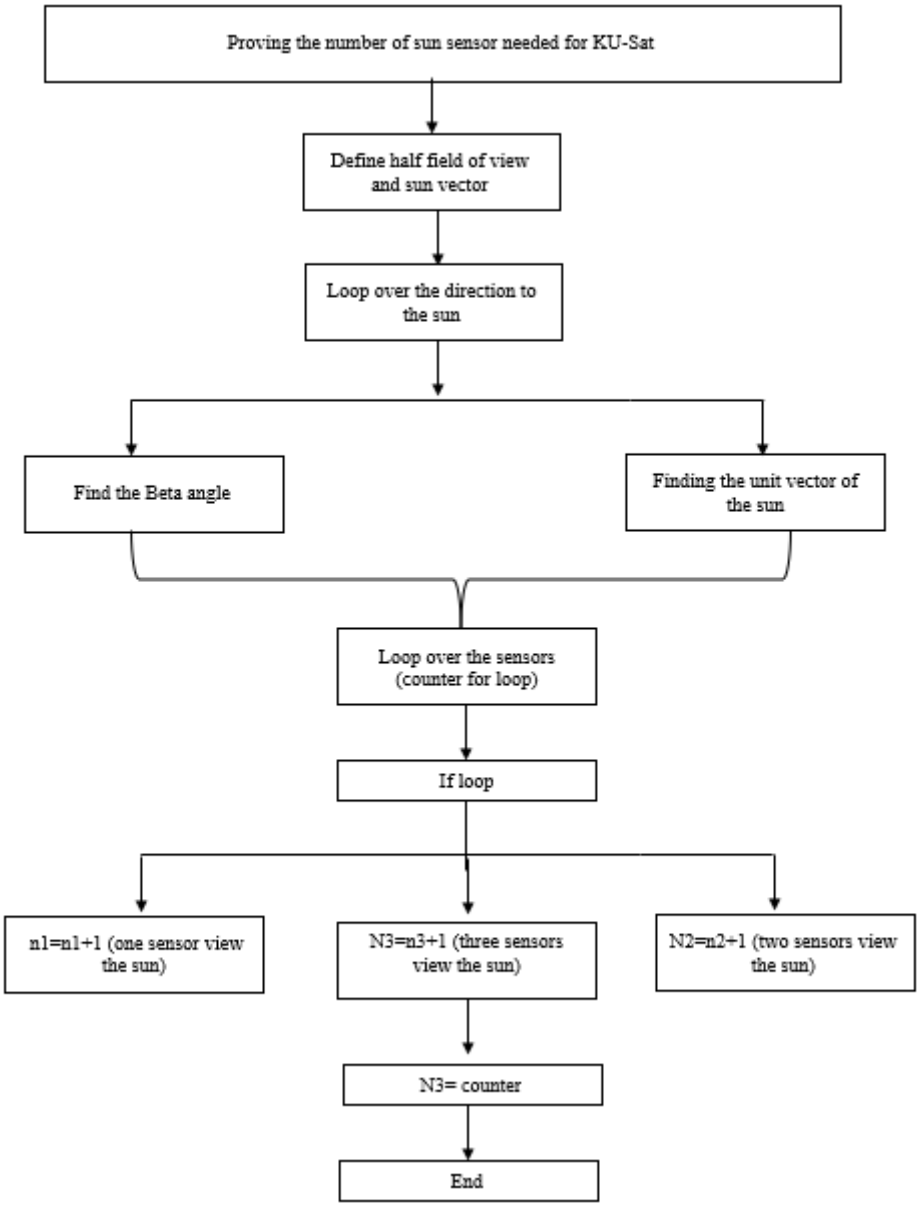


Figure (32): Estimation of needed number of sun sensor algorithm

1. From the data sheet of the selected photodiode (SLCD_61N8), we define the semiaperture of field of view.

2. We define the vectors parallel to the axes of the sensors (unit vectors), example:

u1 = [1.0 0.0 0.0];
u2 = [-1.0 0.0 0.0];
u3 = [0.0 1.0 0.0];

3. We did a loop over the direction of these vector to the sun (spherical coordinates: lambda is longitude, phi is latitude). Until we covered an entire sphere

4. We defined beta = zeros(34,1); which is a vector to save the angle between each *u* vectors (presented in point 2) and *sun vector*s (presented in point 5a)
5. We established Two "for loop": the first one to loop over lambda (for lambda = 0.0: dlambda:360.0) and second one to loop over phi (for phi = -90.0: dphi:90.0), and a counter of sun positions considered to the sun =k.

a. Find the unit vector to the sun

```
xsun = cosd(lambda)*cosd(phi);
ysun = sind(lambda)*cosd(phi);
        zsun = sind(phi);
rsun = [xsun ysun zsun];
```

b. find beta angles

```
beta(1) = acosd(dot(u1,rsun));
beta(2) = acosd(dot(u2,rsun));
beta(3) = acosd(dot(u3,rsun));
```

c. loop over the sensors and check if sun is in view

```
for i = 1:34
if(beta(i) < half_FOV)
                sun_in_view = sun_in_view + 1; this will give the number of
sensors can see the sun
if(sun_in_view >= 3);
n3 = n3 + 1; update the counter for sun positions that are in view of at least
```

d. Stop when n3 = k.

From that we generated the following output,

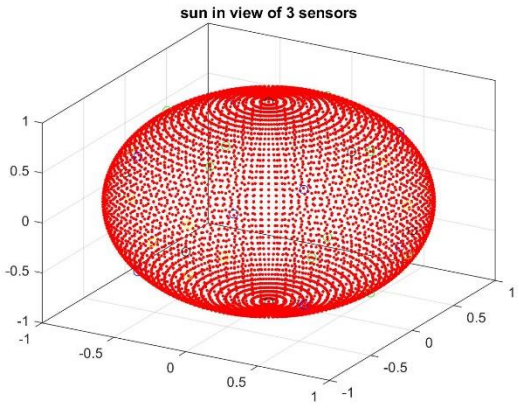


Figure (33): Sun in view of 3 sensors

5.2.2 Attitude Control Individual Component Testing

5.2.2.1 (COTS) BLDC Motor

As we have emphasized on earlier, the Reaction Wheel System was built from scratch as a proof of concept since it consists of nothing but a flywheel attached to a Brushless DC (BLDC) Motor that works on the principles of newton's second law. However, the RWS prototype will be discussed thoroughly in the coming subsection, in this part we will focus on testing the motor that derives the overall RWS to orient the satellite in the desired direction.

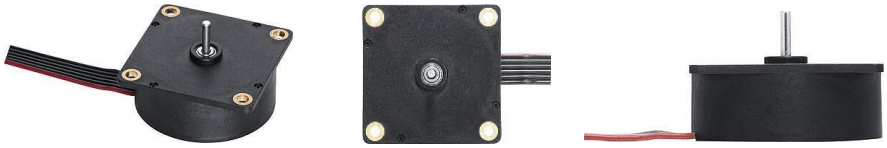


Figure (34): Faulhaber 2610T012B SC Flat Brushless DC Motor[1B]

The first testing set up was constructed to measure the actual RPM Provided by the motor for a given PWM signal. PWM Signal refers to a command sent to the motor by the Arduino to send a portion of the total voltage provided by the battery to the motor, namely, the signal varies from 0 to 255 refers to the range of command sending voltage to the motor from a range of 0 to 11.1 v. The following circuit was used to derive the motor, the full Arduino code used is provided in Appendix

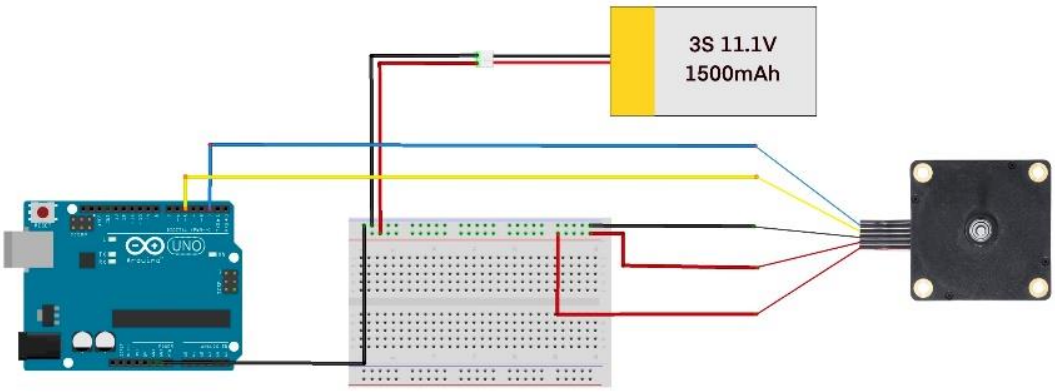


Figure (35): Motor Connections and Circuit Diagram

The testing setup consisted of a reflective flat surface, Tachometer and a LiPo Battery. The reflective flat surface was used to reflect the coming laser beam from the tachometer to measure the real-life RPM.

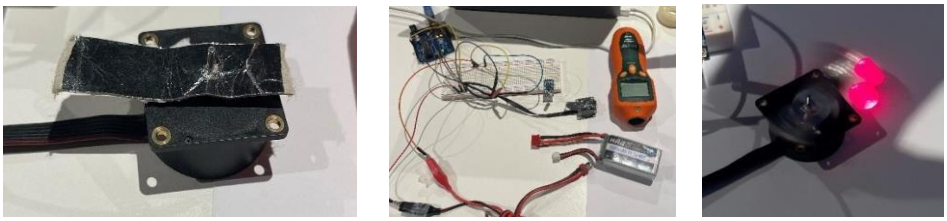


Figure (36): Testing Setup of BLDC Motor

From the manual documentation of data, we generated the following graph to show the input (PWM) output (RPM) to show the relationship,

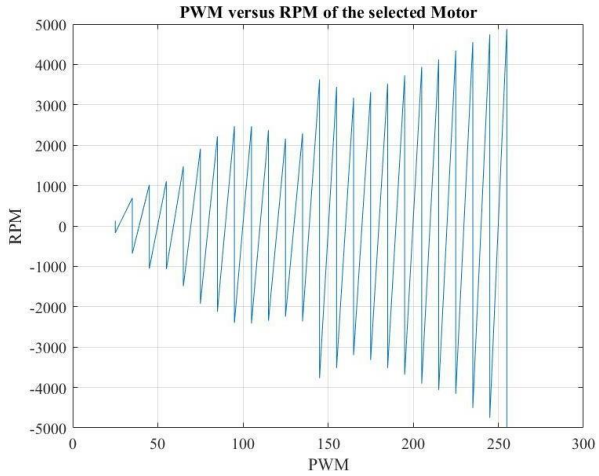


Figure (37): Experimental BLDC motor data

The positive obtained data refers to the RPM value in the clockwise direction and accordingly the negative data refers to the RPM value in the counter clockwise. These values were obtained using 11.1 volts for a safer approach.

Also, from the same set of data we converted the RPM values to (rad/sec) and the PWM signal to the amount of voltage sent to test the relationship between the angular velocity and the commanded work. This established linear relationship is used in the SIMULINK model late.

See in the coming figure how the angular velocity given by the motor responds to a certain commanded torque,

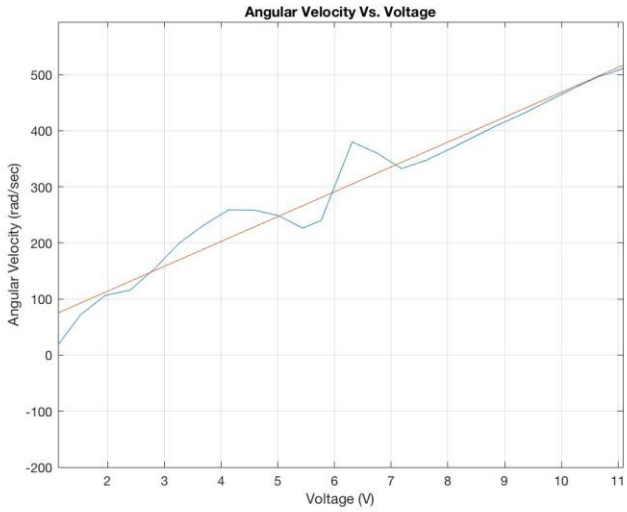


Figure (38): Commanded torque and the resulted angular velocity relation

The second testing set up however was done to ensure that the 3 motors can work simultaneously with one another using the same micro controller and the same battery to lower the weight and increase the overall efficiency of the code. It was decided previously by members in charge that the full system must perform under the same code for the ease of possible future enhancements.

In this test, an Arduino Nano was used rather than an Arduino Uno to best fit inside the CubeSat Structure that we have built (discussed in the coming sections). The following circuit was generated to introduce the reader of how the 3 motors system was connected,

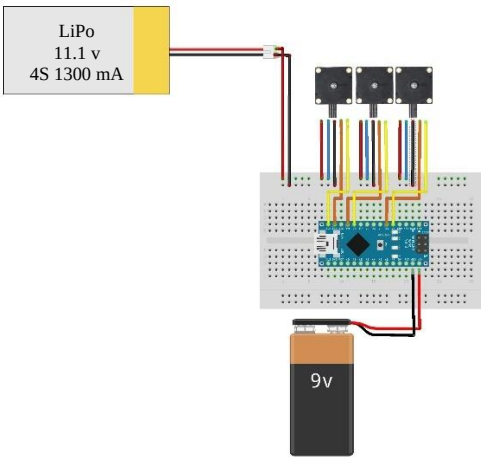


Figure (39): Motor Connections and Circuit Diagram

The following test setup was made to ensure that the motor can perform inside the CubeSat 3-D printed platform (highlighted by red circles in the next figure). To start the fabrication of the overall

setup we needed to make sure that the wiring is well positioned inside the setup to ensure safety of members and environment.

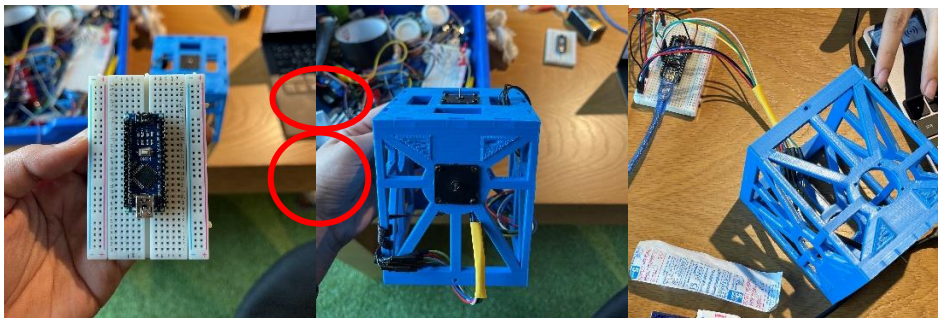


Figure (40): Testing Setup of BLDC Motor inside the CubeSat frame

In conclusion, the circuit worked perfectly and was implemented in the CubeSat body for an initial testing set up. The full RWS fabrication and analysis is provided in a coming section since this one is mainly concerned with the testing of input/output and the optimization of each individual component in the ADCS system.

5.2.2.1.1 Control Analysis for the BLDC Motor Plant

The transfer function from the input voltage (*V*), to the angular velocity (*rad/sec*) of any brushless dc motor can be expressed using the following equation,

$$\frac{\omega(s)}{V(s)} = \frac{K_m}{(Ls+R)\left(Js+K_f\right)+K_mK_b} \quad \cdots\cdots\cdots\cdots (29)$$

For our chosen model of brushless dc motor, the following data were obtained from the datasheet provided. Detailed calculations and conversions are provided in the appendix.

Table (32): Data of the brushless dc motor

V	Voltage	12
R	Terminal Resistance (Ohm)	28.2
L	Terminal Inductance (H)	0.00194
K_m	Torque Constant (Nm/A)	0.0181
K_e	Back emf Constant (Vs/rad)	0.01809592
K_f	Viscous Friction (Nms/rad)	0.0000125204
J	Rotor Inertia (Kgm^2)	0.00000081
ω	Angular Velocity (rad/s)	613.40

Using these values, we obtained the transfer function that was used to study the brushless dc motor theoretically. This transfer function can be used as a reference later to compare with the experimental results if obtained.

$$\frac{\omega(s)}{V(s)} = \frac{11459322.57}{s^2 + 14476.917\,s + 430902.82}$$

The transfer function characteristics from control theory point of view:

- Second order system
- Two negative real poles
- Stable system due to the negative poles
- System has no zeros

The standard second order transfer function form

$$T(s) = \frac{{\omega_n}^2}{s^2 + 2\xi\omega_n + {\omega_n}^2} \cdots\cdots\cdots\cdots (30)$$

Table (33): Definition of the symbols of equation 30

Symbol	definition
w_n	The natural frequency is the oscillation frequency if there is no damping and is an indication of the relative speed of response of the system
ξ	The damping ratio is a parameter, usually denoted by ζ (zeta), that characterizes the frequency response of a second-order ordinary differential equation

From the transfer function we can simply find the natural frequency and the damping ratio as follows,

$$\omega_n = \sqrt{430902.82} = 656.431\, rad/sec$$

$$2\xi w_n = 14476.917$$

$$2\xi(656.431) = 14476.917$$

$$\xi = 11.03$$

The damping ratio is greater than one ($\xi > 1$) which means that the system is overdamped. An over damped system takes longer to return to equilibrium. Also, the positive damping ratio indicates that the system is stable.

The following plots were generated in MATLAB:

1. **Step response**

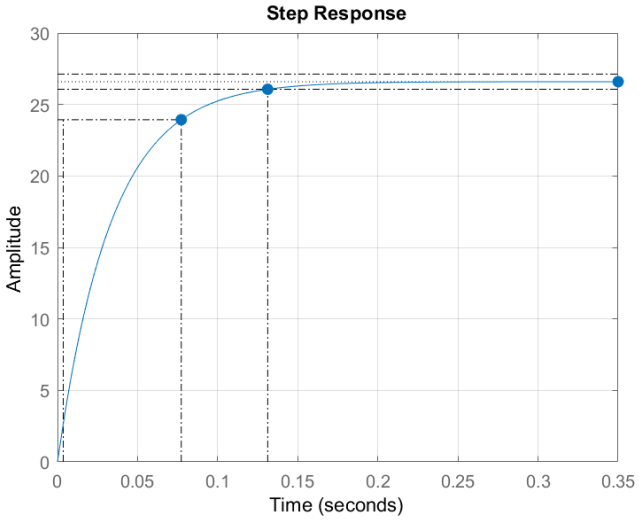


Figure (41): Step Response of the BLDC motor theoretical plant

The system characteristics:

- Overshoot = 0%
- Peak response = steady state value = 26.6
- Settling time = 0.131 *seconds*
- Rise time = 0.0737 *seconds*

2. **Bode Plot**

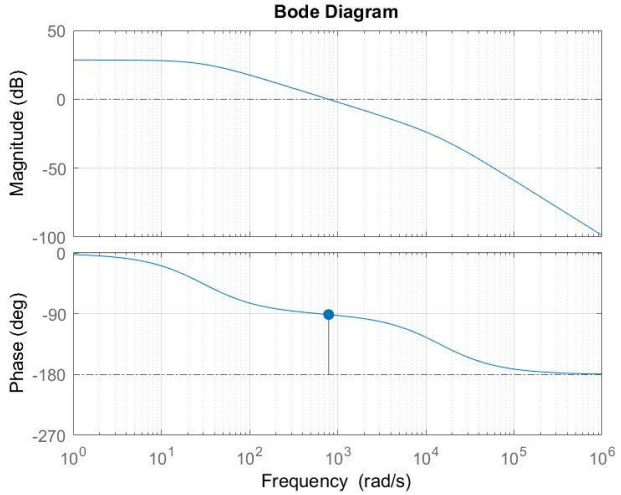


Figure (42): Bode Plot of the BLDC motor theoretical plant

The system characteristics:

- Gain Margin = Infinity

An infinite gain margin means the system will never become unstable anyhow in future, no matter how much gain we keep on increasing. Or the margin to reach the verge of instability is infinity. This concept not attainable in reality but as we said, since this is a theoretical model, a perfect system is somehow a reasonable justification but will never be attained in reality

- Phase Margin = 89 *degrees* at frequency = 791 *rad/sec*

System is stable as indicated from the plot

- **Ziegler Nicolas PD Controller Tuning**

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{11459322.57}{s^2 + 14476.917\,s + 430902.82}$$

PD controller:

$$G_c(s) = K_p + K_D s \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \quad (31)$$

Closed-loop transfer function

$$T(s) = \frac{G(s) \cdot G_c(s)}{1 + G(s) \cdot G_c(s)} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \quad (32)$$

$$T(s) = \frac{\frac{11459322.57}{s^2 + 14476.917\,s + 430902.82}(K_p + K_D s)}{1 + \frac{11459322.57}{s^2 + 14476.917\,s + 430902.82}(K_p + K_D s)}$$

$$T(s) = \frac{\frac{11459322.57}{s^2 + 14476.917\,s + 430902.82}(K_p + K_D s)}{\frac{s^2 + 14476.917\,s + 430902.82 + 11459322.57(K_p + K_D s)}{s^2 + 14476.917\,s + 430902.82}}$$

$$T(s) = \frac{11459322.57(K_p + K_D s)}{s^2 + 14476.917\,s + 430902.82 + 11459322.57(K_p + K_D s)}$$

$$T(s) = \frac{11459322.57(K_p + K_D s)}{s^2 + (14476.917 + 11459322.57K_D)\,s + (430902.82 + 11459322.57K_p)}$$

As previously found

$$\omega_n = 656.431\,rad/sec$$

$$\xi = 11.03$$

According to the standard second order transfer function form

$$\omega_n{}^2 = 430902.82 + 11459322.57K_p$$

$$430902.82 - 430902.82 = 11459322.57K_p$$

$$2\xi w_n = 14476.917 + 11459322.57K_D$$

$$14476.917 = 14476.917 + 11459322.57K_D$$

This results in zero gain values; this can be justified since the plant we found imbodied a perfect system since It has a gain margin = infinity. Other than that, the real-life transfer function can be obtained with different methods such as system identification toolbox but unfortunately, we didn't have the time to test this possibility given the limited time period.

5.2.2.2 Magnetorquer Rod Testing

To test the magnetorquer we decided with the help of YahSat lab to do the only available test since there is no enough equipment's to properly test the magnetorquer capabilities and performance. The test was performed to understand the mechanism of the magnetorquer and the physics behind it.

The testing was conducted to measure the magnetic field intensity when each axis of the magnetorquer is powered on individually. This was verified by placing the previously discussed chip that has a built-in 3-Axis Gyroscope, 3-Axis Magnetometer and a 3-Axis Accelerometer to use the magnetometer capabilities to measure the magnetic field. The magnetometer should read different results when we switch the magnetorquer on and when we switch it off since it affects the magnetic field value.

Unfortunately, when we changed the magnetometer position slightly for couple of trials the results showed a large difference. This inconsistency of such output might have happened because of the decupling of the earth magnetic field and the magnetorquer dipole. In this trial the magnetometer was fixed in place as shown in figure 43 in the hope of determining a reliable reading values to build the study upon. However, when we repeated the process for a couple of trials to ensure accuracy, the data differed magnificently which proved to us that the testing has failed. Eventually, after many discussions with YahSat lab members, it was concluded that without the proper testing facilities we will not be able to properly test this component without a huge set of errors and inaccuracies.

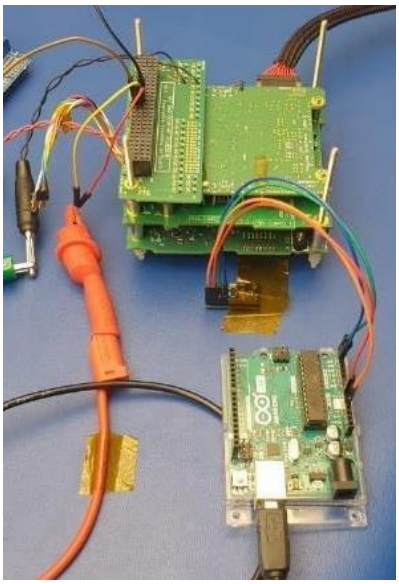


Figure (43): Magnetorquer rod testing setup

The plots shown below reflect the results of one of the trials that we did by switching each axis to the maximum intensity available and turning it off.

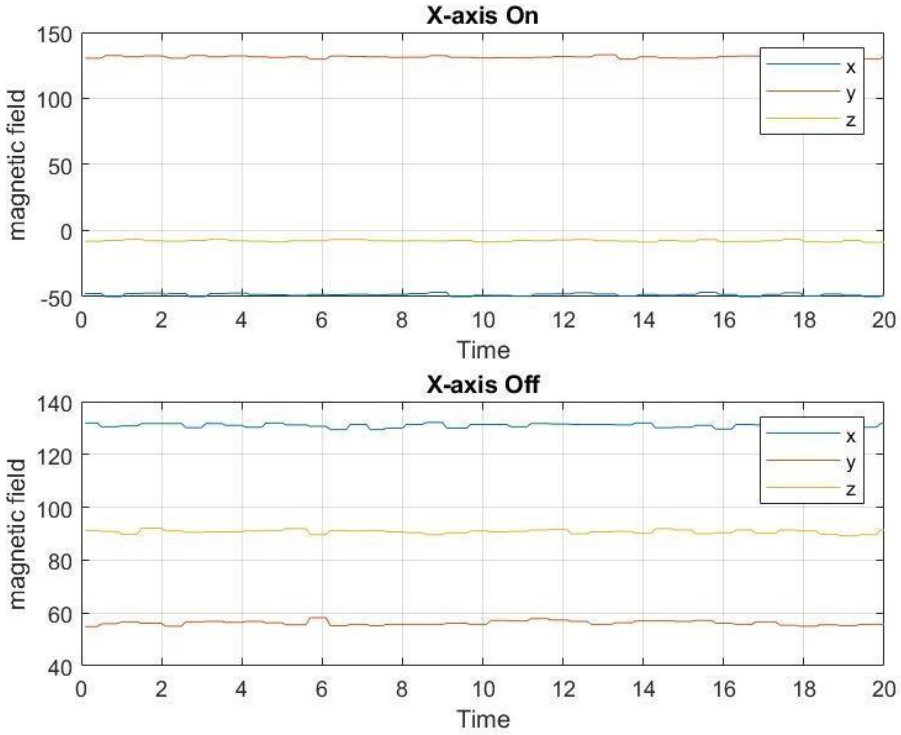


Figure (44): The magnetorquer X-axis turned on / off

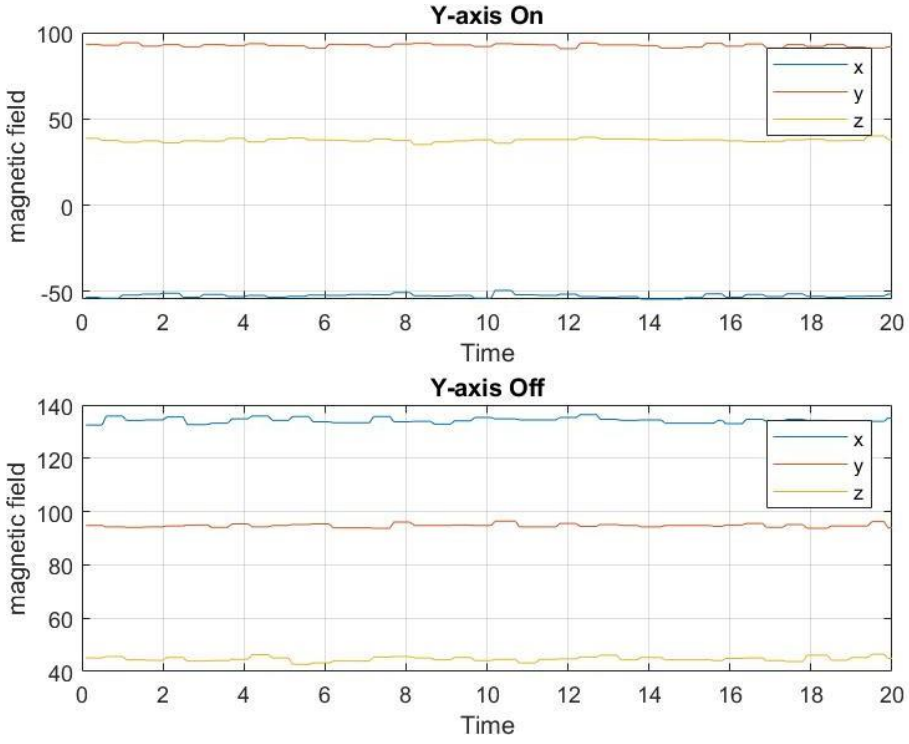


Figure (45): The magnetorquer Y-axis turned on / off

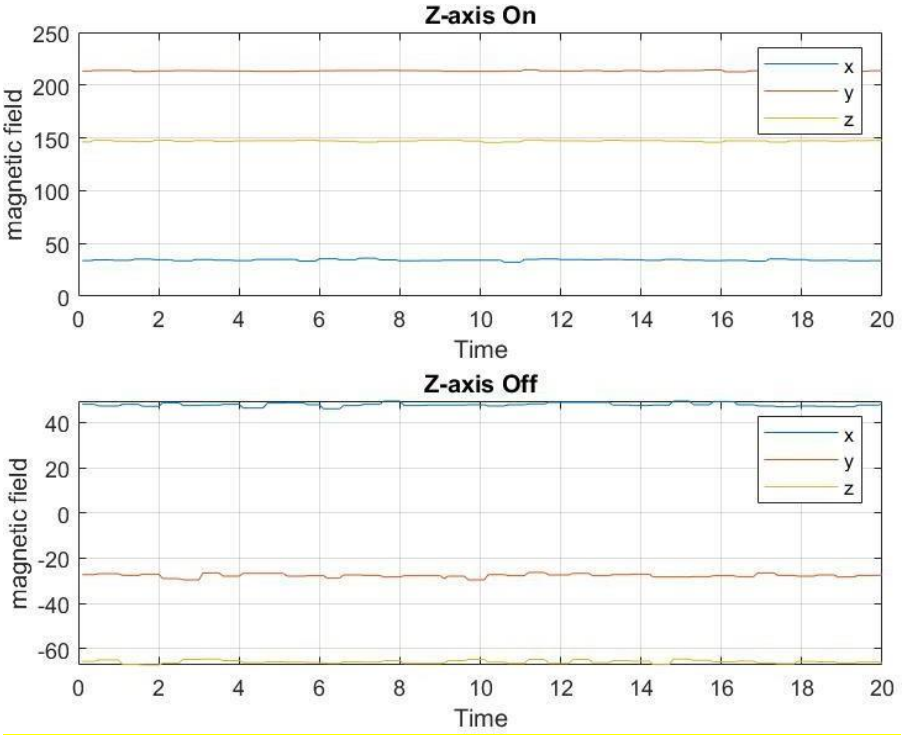


Figure (46): The magnetorquer Z-axis turned on / off

The first graph of each in figure (44,45,46) show the magnetic field when X,Y,Z-axis torque are ON the results shows a very large magnetic field even though the earth magnetic field is low and constant. Whereas the second graph in each figure (44,45,46) shows the magnetic field when all torques are off which should show all three have the same value. Which is not the same in our trials.

5.3 Fabrication of Reaction Wheel System (RWS)

Reaction wheels are simply a combination of a brushless dc motor and a flywheel, the functionality relies on the conservation of angular momentum concept; if the wheel accelerates (rotate) in one direction, the CubeSat will accelerate in the opposite direction causing a rotational motion of the entire body around the axis in which the motor is orthogonal to. [30] Reaction wheels are capable of delivering torques that are both larger and more accurate than the magnetorquers so using them for the Pointing Control Mode is more sufficient. Their limitation is that they have a maximum rotation speed and can therefore become saturated. In this case a simple Magnetorquer design can be used to desaturate the wheel if necessary. [30]

In the preliminary design phase of KU-Sat it was decided that a Reaction Wheel System (RWS) would be used as the main actuator for fast response attitude control. Although the magnetorquers are used to generate small torques over an extended period of time, reaction wheels are used to generate relatively high torques over short periods of time which is what we need to achieve the camera pointing mode. Together with a comprehensive set of sensors, the magnetorquers and reaction wheels provide KU-Sat with the required full three-axis active control. [30] It was decided early in the previous semester that the reaction wheels would be designed in-house using Commercial Off-The-Shelf (COTS) Brushless Direct Current (DC) motors. Only DC motors are considered, because they are widely available and more compatible with Requirements. Fully developed reaction wheels are available on the market, but this was not deemed compatible with the educational objective of KU-Sat due to their very high price and high level of design. [30]

Since the motor have been discussed thoroughly in many earlier sections. We will focus in this section on the implementation of the flywheel and the testing frame. Also, the overall results and finding of the overall prototype performance is going to be discussed in details in this section.

5.3.1 Flywheel Sizing

A flywheel is a spinning wheel, or disc, or rotor, rotating around its symmetry axis. Energy is stored as kinetic energy, more specifically rotational energy, of the rotor.

A flywheel sizing is a necessity and a must when it comes to designing a RWS, after all, the mechanics behind the flywheel is what will impose the torque into the CubeSat body. We know from

before that a single reaction wheel is nothing more or less than Flywheel attached to the shaft of a brushless dc motor. Therefore, the specific dimension of such element is crucial and need to be dealt with carefully. The mechanical structure of the flywheel was also printed using PLA, however, the design parameters are discussed extensively after the visual presentation of the current flywheel design,

Table (34): Design parameters of the 3d printed flywheel

Size	*Value*
Mass	*21 g*
Inner Diameter	*3.6 cm*
Outer Diameter	*4.6 cm*
Inner Height	*1 cm*
Outer Height	*1.5 cm*
Ring Thickness	*1 cm*

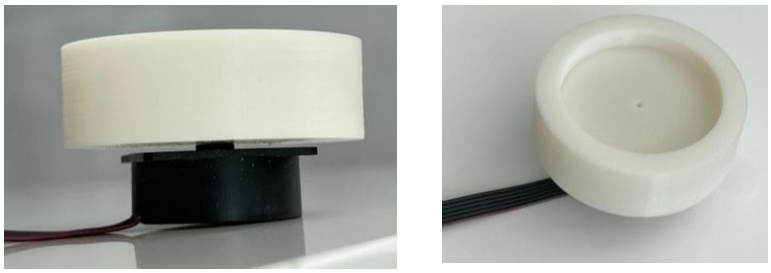


Figure (47): The 3D printed flywheel attached to a BLDC motor

From the flywheel inertia analysis in section 3.3 – f, we found a reasonable value for an initial flywheel that can perform the mission with a reasonable result. Namely,

$$I_{zz} = 7 \times 10^{-6}\ Kgm^2$$

Now that we have an initial value for the total mass moment of inertia of the wheel, we generated an initial testing design parameter using the following procedure,

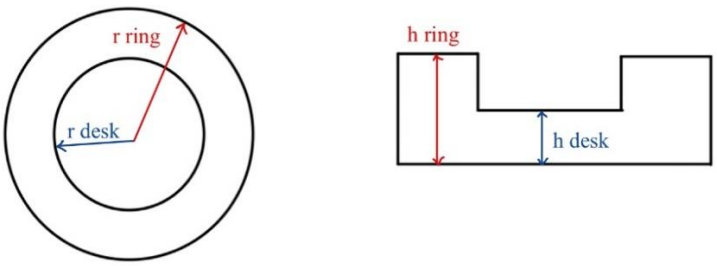


Figure (48): Flywheel design sketch

$$I_{zz} = I_{disk} + I_{ring} \quad \cdots\cdots\cdots\cdots\cdots\cdots (33)$$

$$I_{disk} = \frac{1}{2} m_{disk} r^2_{disk} = \frac{1}{2} [\rho \pi r^2_{disk} h_{disk}] \; r^2_{disk} \quad \cdots\cdots\cdots\cdots (34)$$

$$I_{disk} = \frac{1}{2} \rho \pi r^4_{disk} h_{disk} = \frac{1}{2} (1178) \pi r^4_{disk} (0.01)$$

$$I_{ring} = \frac{1}{2} m_{ring}\left(r^2_{ring} + r^2_{disk}\right) \quad \cdots\cdots\cdots\cdots\cdots (35)$$

$$I_{ring} = \frac{1}{2} \left[\rho \pi\left(r^2_{ring} - r^2_{disk}\right) h_{ring}\right]\left(r^2_{ring} + r^2_{disk}\right) \quad \cdots\cdots\cdots\cdots (36)$$

$$I_{ring} = \frac{1}{2} [(1178) \; \pi(0.69 r^2_{disk}) (0.015)](2.69 r^2_{disk})$$

$$I_{zz} = \frac{1}{2} \rho \pi r^4_{disk} h_{disk} + \frac{1}{2} \left[\rho \pi\left(r^2_{ring} - r^2_{disk}\right) h_{ring}\right]\left(r^2_{ring} + r^2_{disk}\right)$$

$$7 \times 10^{-6} = \frac{1}{2} (1178) \pi r^4_{disk} (0.01) + \frac{1}{2} [(1178) \pi(0.69 r^2_{disk}) (0.015)](2.69 r^2_{disk})$$

$$7 \times 10^{-6} = 18.504 \, r^4_{disk} + 51.52 \; r^4_{disk} = 70.022 \; r^4_{disk}$$

Solving the system, we will get,

$$r_{disk} = 0.0177814 \, m$$
$$r_{ring} = 0.0231138 \, m$$

Both values were used to design the CAD module of the initial flywheel testing and 3 units were sent for printing.

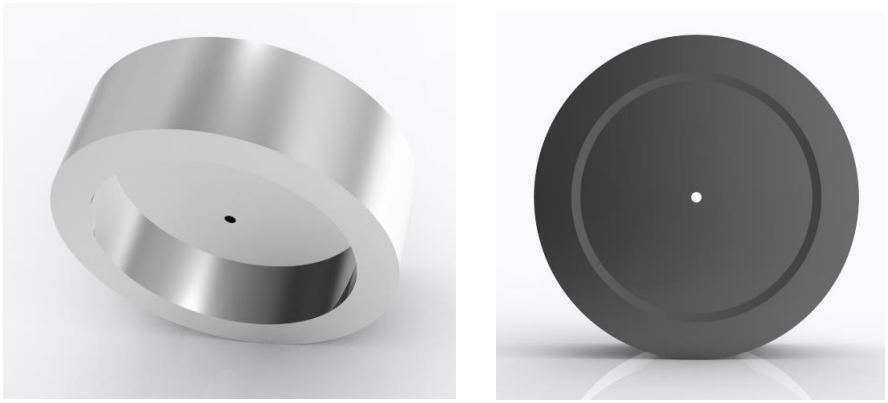


Figure (49): Default view of the reaction wheel (left), front view of the reaction wheel generated via Creo Rendering

5.3.2 RWS Platform

To best fit in the requirements of the project, we decided to print the RWS platform to be in the shape of 2U CubeSat. Basically, the platform consists of two units detachable CubeSat frame with empty holes on the wall for the reaction wheel assembly. All the sizes taken into account were optimized to best meet the design produced by the earlier stages of this project.

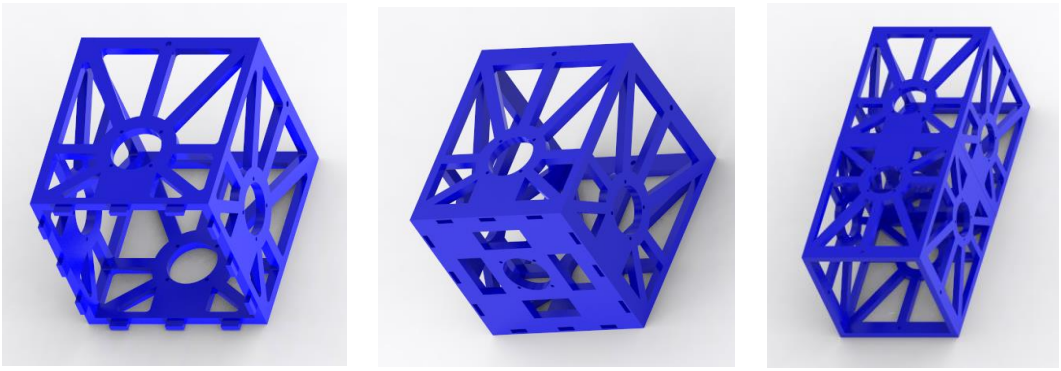


Figure (50): RWS Platform as a 2U CubeSat nonensemble and assembled via Creo Rendering

Due to lack of resources and time constraints our design was limited to 3-D Printed PLA with the following specifications,

Table (35): Design parameters of the 3d printed CubeSat Frame

Size	*Value*
Mass	*248 g*
Height	*22.8 cm*
Width	*9.6 cm*
Thickness	*0.5 cm*

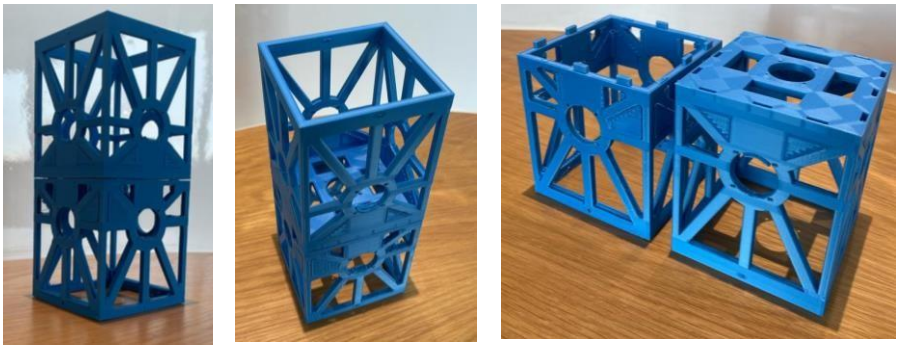


Figure (51): 3D printed 2U CubeSat using PLA

5.3.3 Testing Frame

To properly test the RWS, we established multiple testing setups. At first, we suspended the CubeSat using a wire to test the reaction wheel aligned with z-axis. Secondly, we used a ball bearing and a carbon fiber rod to suspend the CubeSat again for the testing of the reaction wheel aligned with

z-axis. Lastly, two set of testing frames were developed in the hope of giving a full suspension of 2 degree of freedom to test the performance of all reaction wheel assembled.

5.3.3.1 Testing Setup – 1

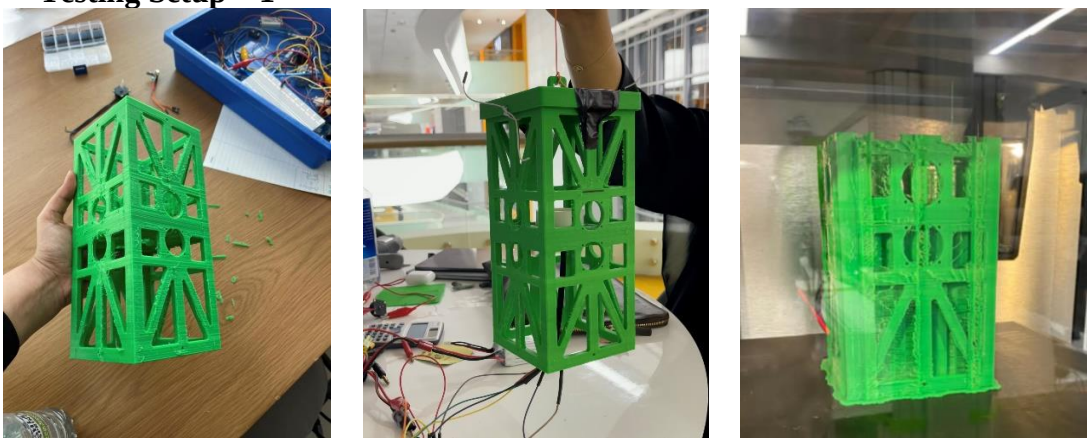


Figure (52): First testing setup for the RWS Combination

In this testing set up a previously printed 2U CubeSat structure was used. However, this 3-D printed frame was ditched due to modification in frame and flywheel placement adjustments. Nonetheless, by suspending it by a wire, and by using the previously discussed flywheel mechanism this structure rotated around z-axis by almost 280 degree. A proof of that is shown in a video clip, the video can be accessed by a link in the Appendix.

5.3.3.2 Testing Setup – 2

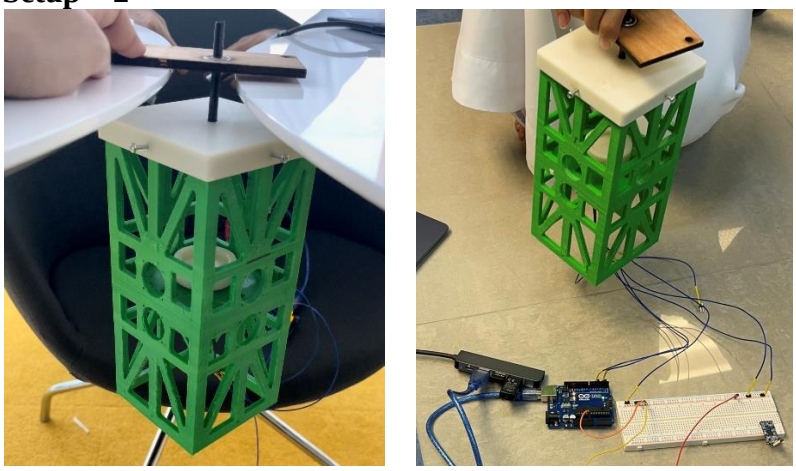


Figure (53): Second testing setup for the RWS Combination

The second test consisted of the same element but this time to avoid the tension held in the wire we used a carbon fiber rod and a ball bearing. This testing set up didn't perform as well as the previous one. The frame didn't complete a full 90-degree rotation due to high friction between the

carbon fiber rod, structure frame and ball bearing. A proof of that is shown in a video clip, the video can be accessed by a link in Appendix.

5.3.3.3 Testing Frame Setup – 1

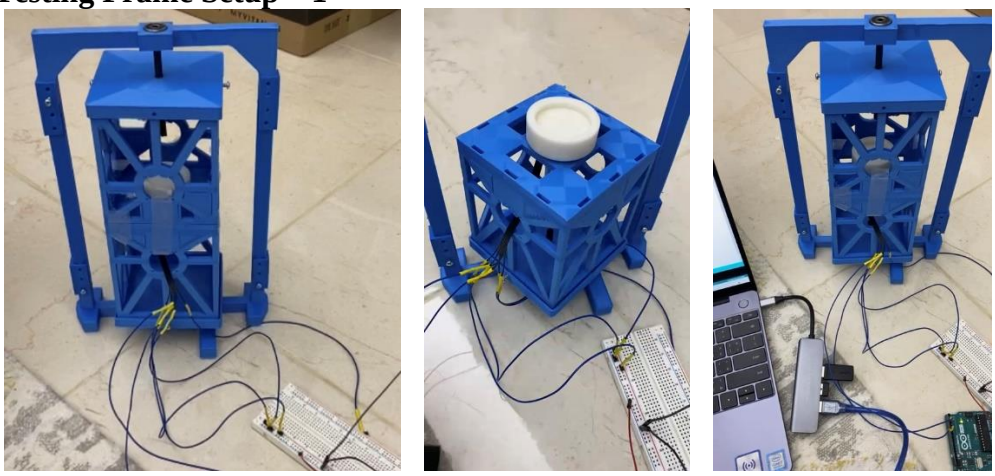


Figure (54): Third testing setup for the RWS Combination

In this setup, a full 1-degree of freedom testing axis was developed to suspend the CubeSat frame and to allow for a smooth rotation. The performance of this setup overlooked that of the previous ones. The rotation was controlled and smooth around the z-axis, this resulted from the overall suspension stability of the frame. Unfortunately, due to limited wiring assembly we couldn't test the full rotation without fixing a better, longer more stable wire and electrical connection within the testing frame. A proof of that is shown in a video clip, the video can be accessed by a link in Appendix.

5.3.3.4 Testing Frame Setup – 2 (Final Design)

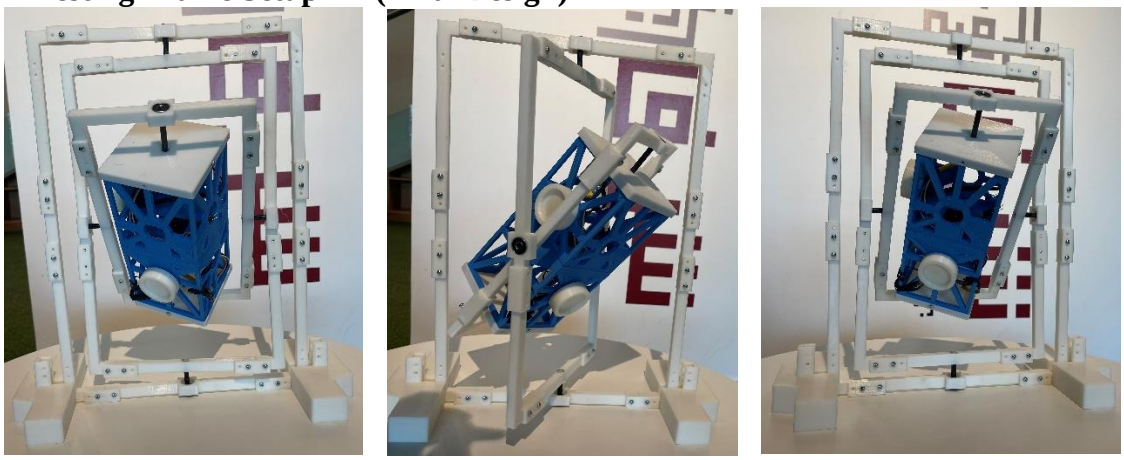


Figure (55): fourth testing setup for the RWS Combination

In this final setup, since we didn't have enough time to test each individual axis alone, we decided to design a frame that can test 2-3 axis at the same time to test more than one axis at a time without

changing the overall arrangement of the setup. This setup is not the best one possible but it is the best as far as our progress have reached. The setup compiles of 3 interfacing rectangles, each one is responsible of rotating around one axis. The CubeSat structure and the printed flywheels were assembled all together for a final mechanism testing. It is safe to say however that in this approach we connected all the wires across the CubeSat body for a fair chance of rotation. Unfortunately, due to time constrains we were not able to test the full testing bench since it contained many wiring issues and needed an extra safety precautions with a supervisor. We decided to leave this testing for future work.

5.3.4 Assembly

In this section, we will only discuss the assembly of the final design. This is due to the fact that this is the final submitted design and contained all the correction of the other one presented.

We started by designing the frame in CREO to obtain the best attainable design within the capability of the available 3-D printer. A view of the frame rendered via CREO and it looked like following,

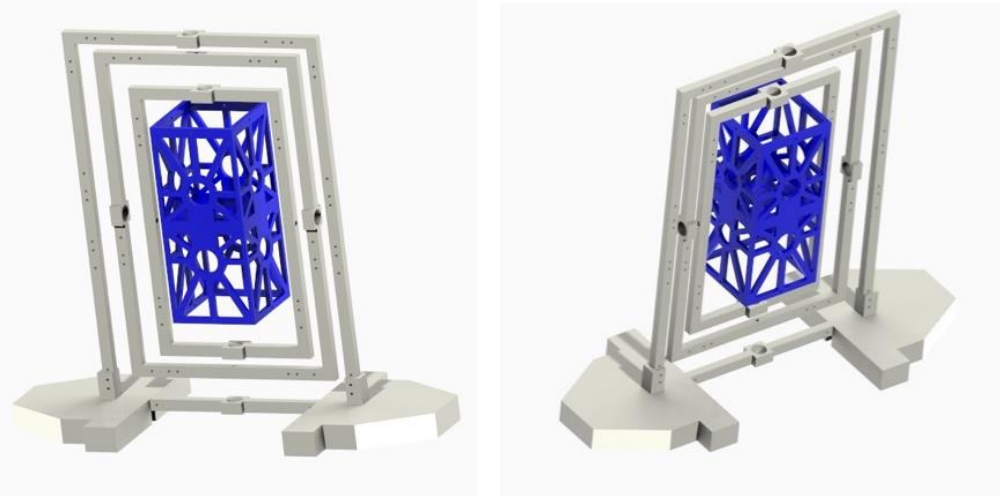


Figure (56): RWS System and Testing Setup assembly in CREO

Following that, the assembly of each individual component started. The following pictures summarize the process as follows,

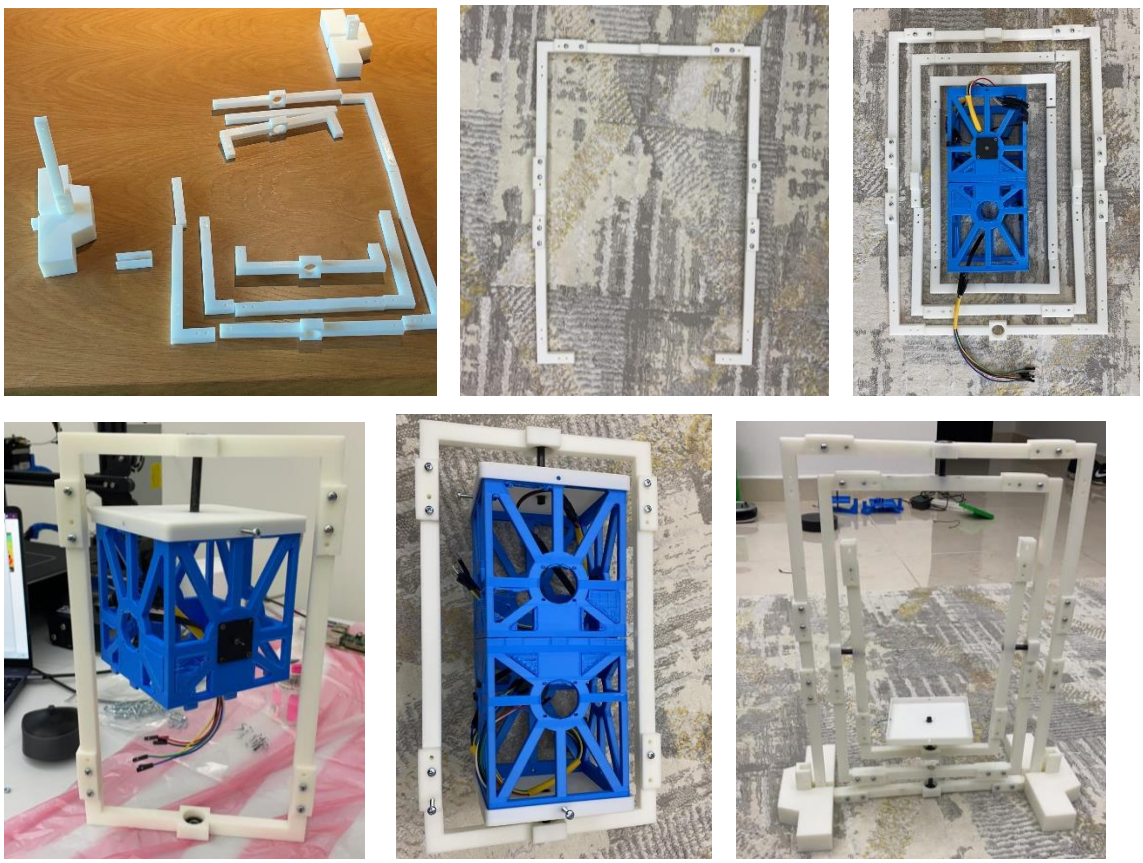


Figure (57): RWS System assembly of mechanical components

Following that, we started to assemble the wires within the frame and finish the final design assembly as follows,

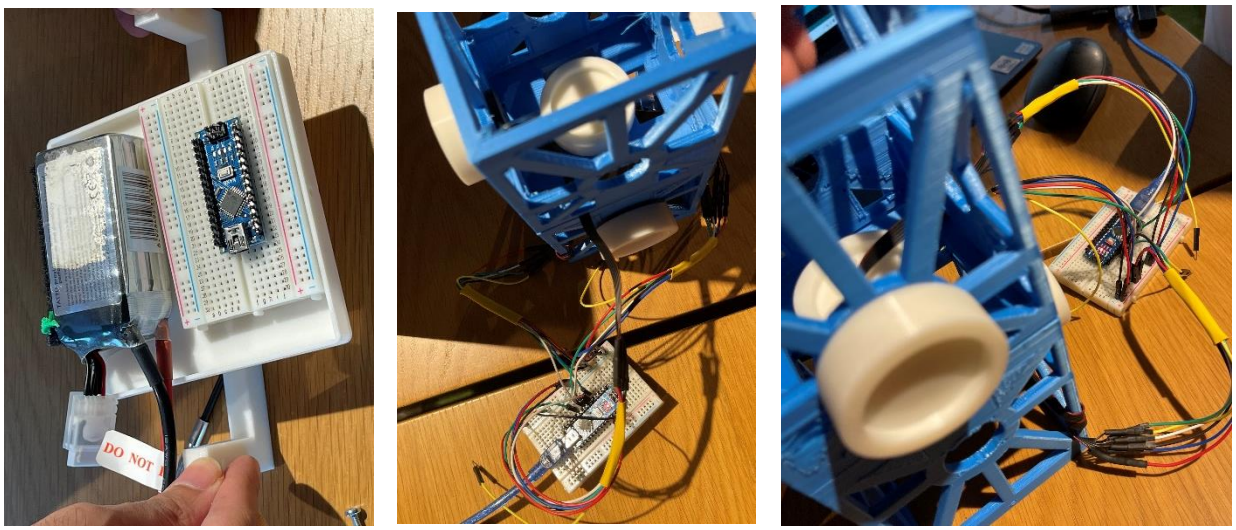


Figure (58): RWS System assembly of electrical components

5.3.5 Results, Evaluation and Reflection

Since the established flywheel capability demonstrated its efficiency in test setup 1, all that happened afterward was an attempt to make the overall result better demonstrated with a smoother outcome and a more compact final design. Unfortunately, as stated, the risks behind using a LiPo batter and a not very secure wiring arrangement was a big challenge we faced by the last week of this project and decided to not proceed with the testing to ensure safety of all members.

However, this can be discussed in future plan and the capability of this specific testing bench can be discussed thoroughly in the coming future. Also, since the last testing frame supports the movement of two axis only, a future plan developing a testing bench that provides 3 degrees of freedom is in order.

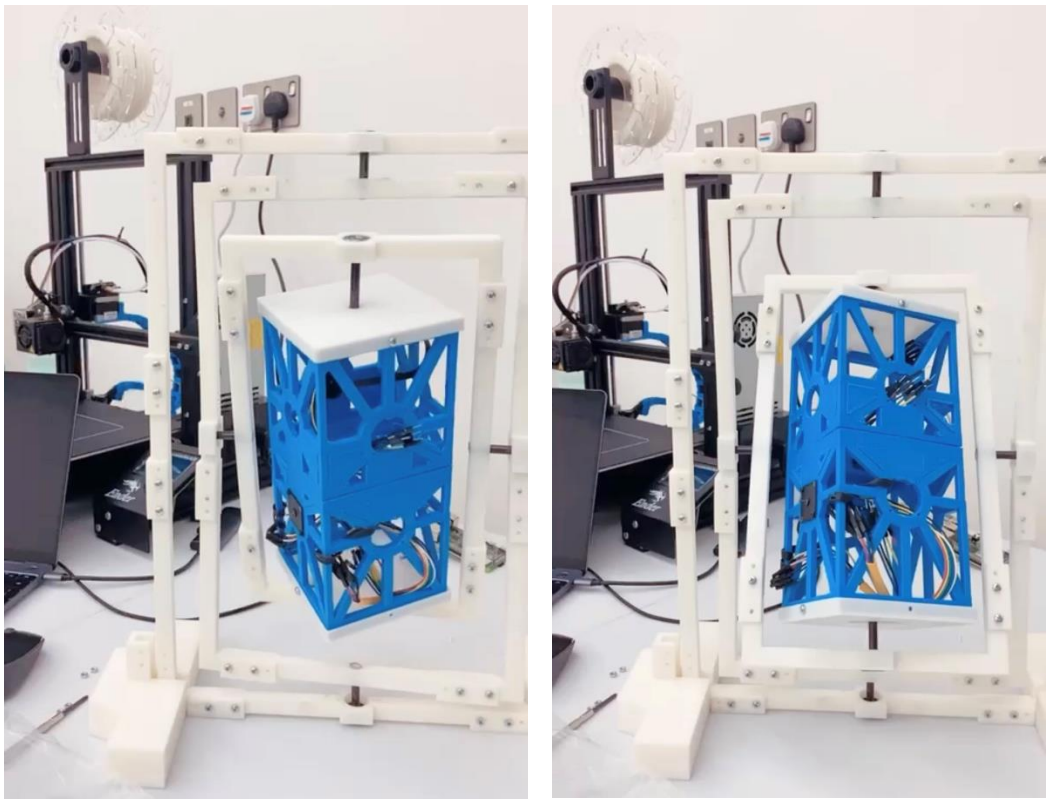


Figure (58): RWS System full assembly

Chapter 6. Full ADCS SIMULINK MODEL

6.1 Engineering Design Analysis and Detailed Documentation

SIMULINK/MATLAB were used to create the full model of the designed ADCS. Using the Aerospace and Defense toolbox, and by modelling the selected sensors and actuators, accurate simulation results were obtained to demonstrate the performance of the ADCS. It is safe to say however that this model was approved valid and entirely correct by YahSat Lab.

6.1.1 Attitude Determination Simulink Model

Figure 60 shows the overall attitude determination system used to determine the attitude throughout the orbit of the CubeSat.

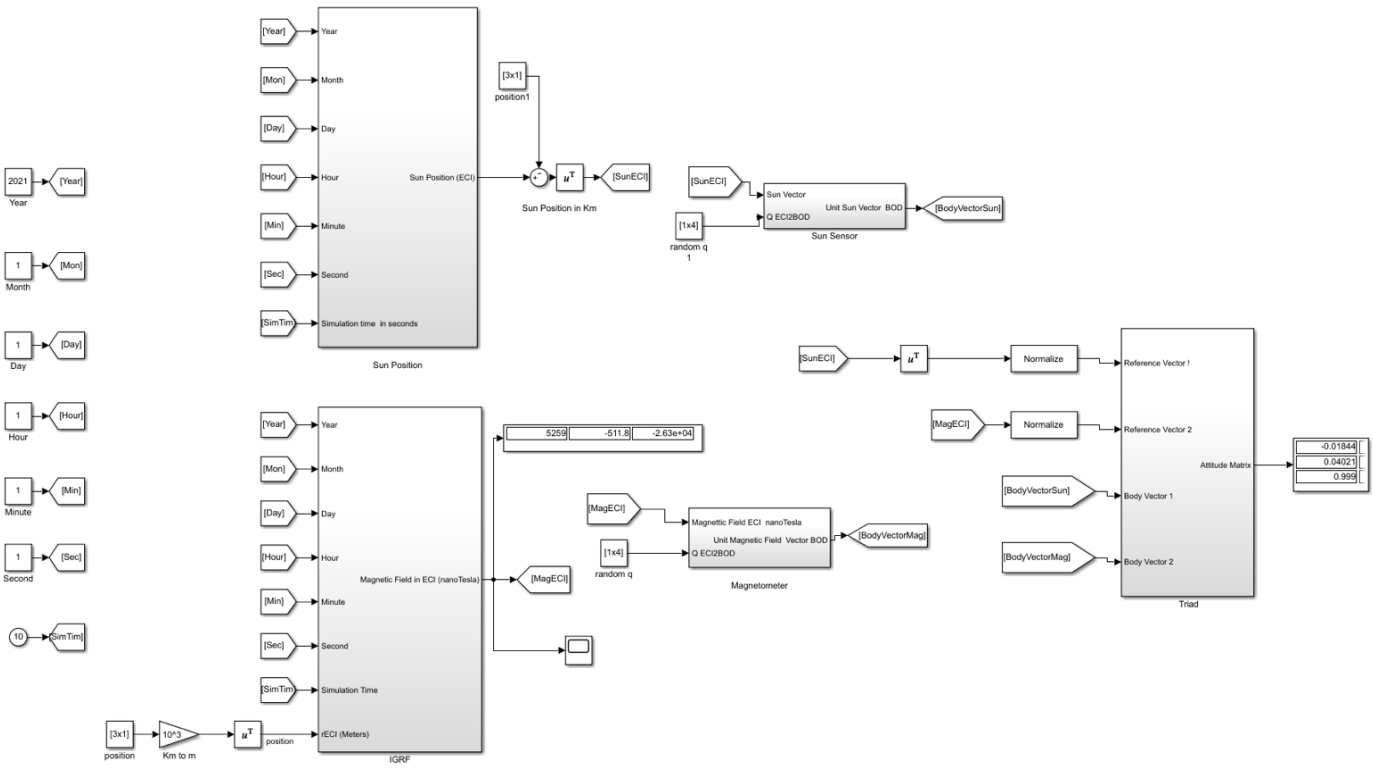


Figure (60): KU-Sat attitude determination system plant

The ADS system plant includes five subsystems connected together; each sub-system will be explained in detail in the next few pages.

6.1.2 Attitude Control Simulink Model

The following plant represent the whole attitude control system. This attitude control system is responsible for the reduction of the deviation between the measured attitude and a desired guidance attitude.

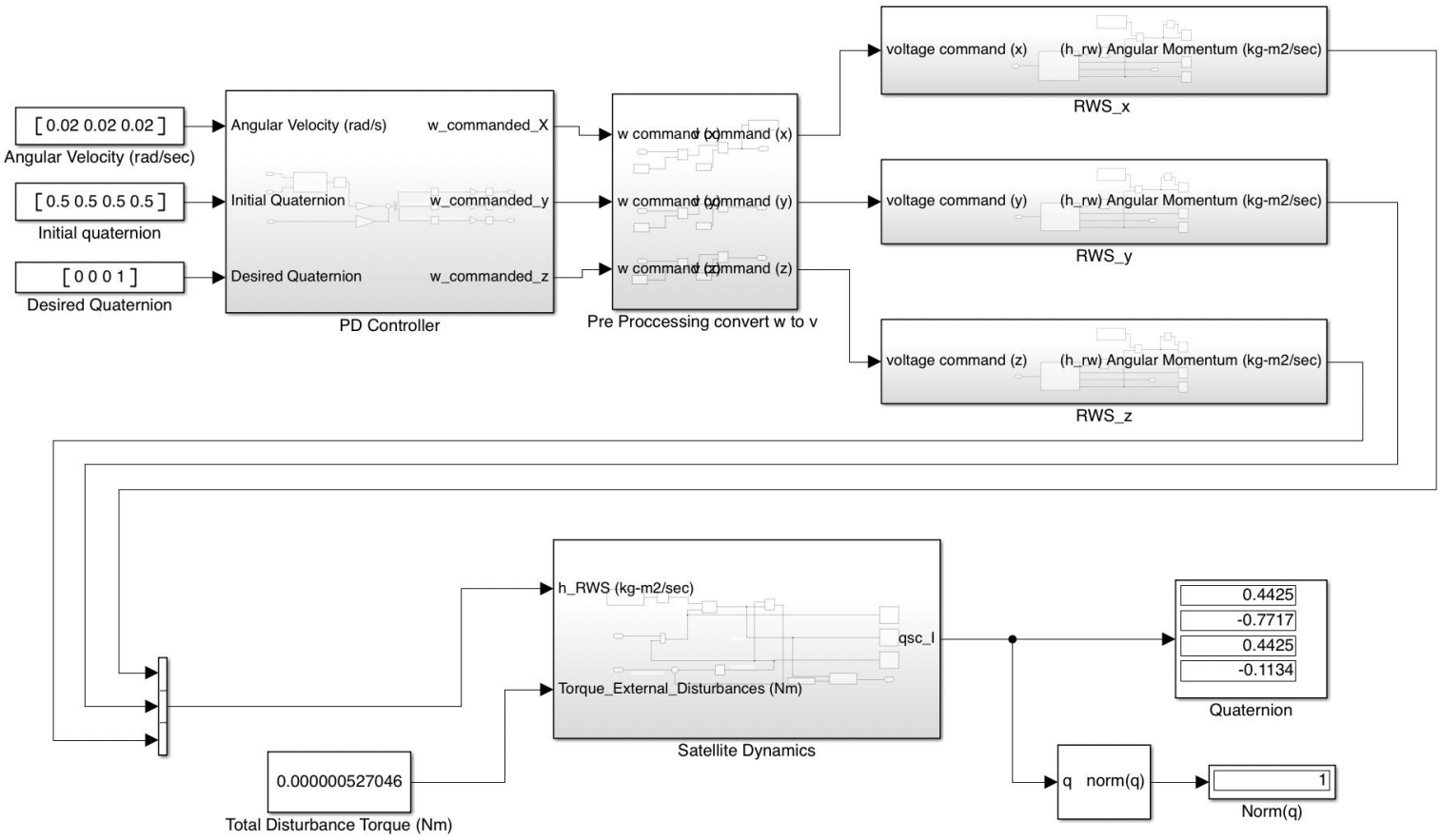


Figure (60): KU-Sat attitude control system plant

The ACS system plant includes four subsystems connected together; each sub-system will be explained in detail in the next few pages. As shown, the inputs of the system are the current angular velocity vector, the initial/current quaternion, and the desired quaternion. While the in our case the output of this system is random set of data used for testing only. The overall system Simulink model works perfectly as it was approved by YahSat lab. In addition, we were introduced to a concept that states that if the final quaternion norm is equal to one, then our system and data used are all correct and works successfully.

6.2 Detailed Design Documentation (ADS)

6.2.1 Sun Position Simulink Block

Figures 61 and 62 show the sun position block that calculates the position of the sun in ECI coordinates. It takes a date in calendar format and the simulation time as inputs and uses functions julian and fracjday to convert the input date from calendar format to Julian date format [31]. This is then used, along with the desired position of the satellite, to calculate the sun's position in ECI coordinates.

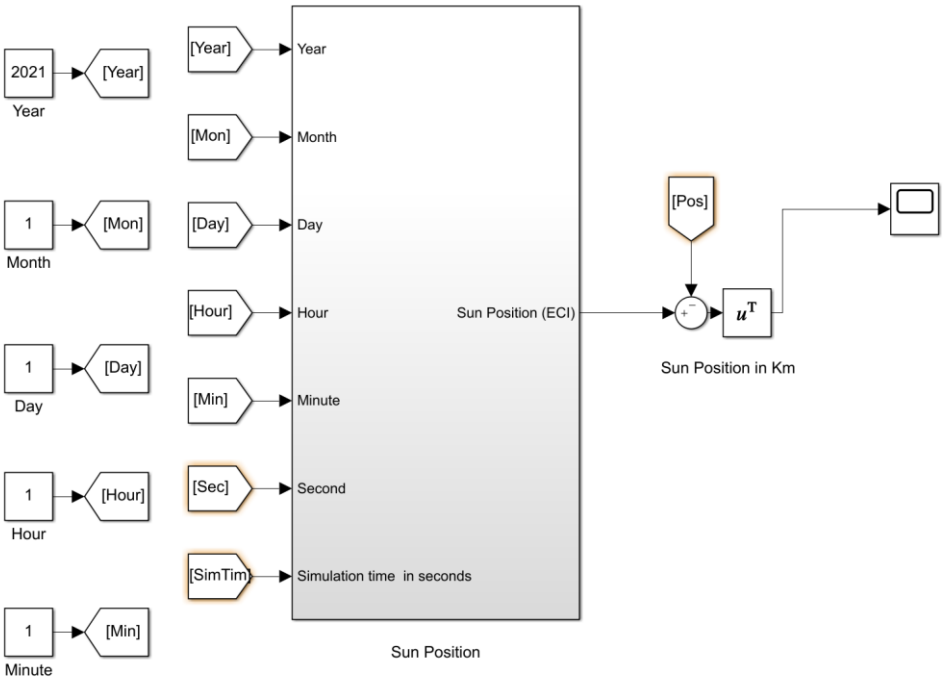


Figure (61): Level 1 of the Sun Position Block

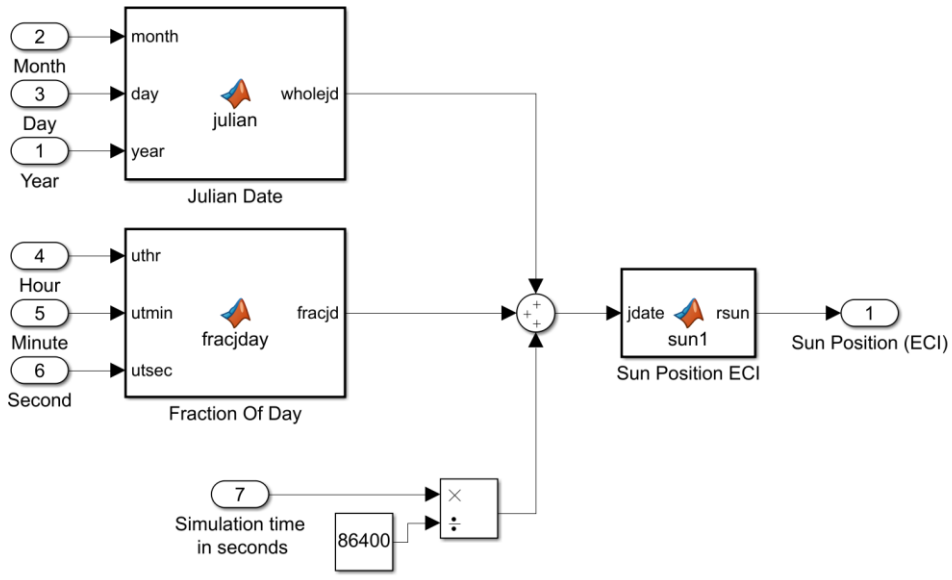


Figure (62): Level 2 of the Sun Position Block

When running the Simulink blocks, the following plot is obtained showing the output.

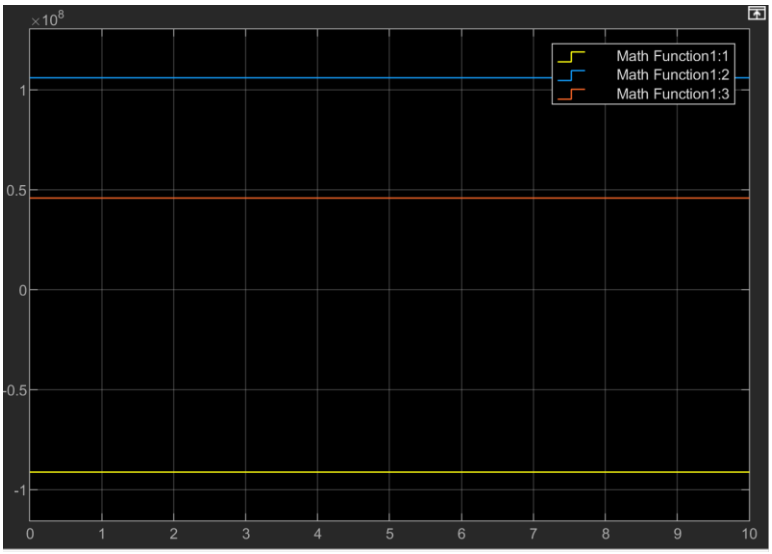


Figure (63): results of the Sun Position Block

To check wether the sun position Simulink model works correctly or not, we must check that the results are constant. As we can clearly see form the above graph, the results (Yellow x direction, blue y direction and red z direction) are all constant which means that the model is working perfectly.

We used another method that is more accurate to make sure that the sun position Simulink model is working. We used GMAT to get the position of the sun

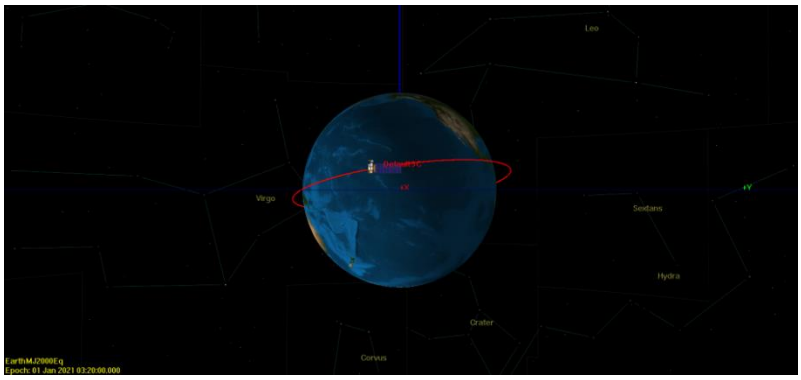


Figure (64): satellite orbit

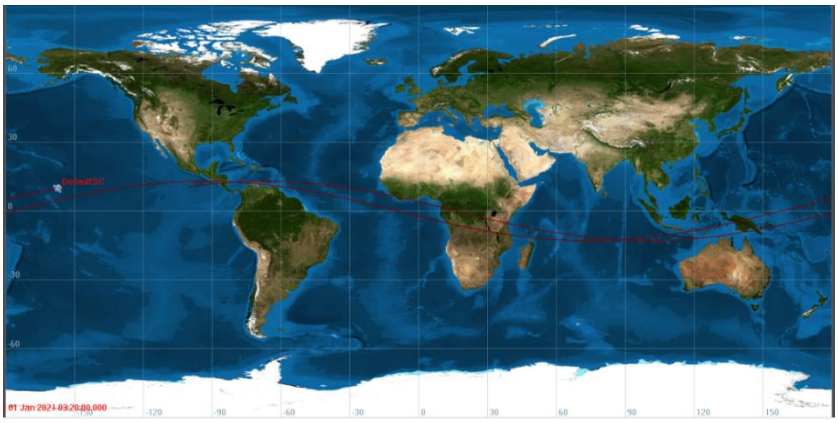


Figure (65): satellite trace track

The following results were obtained, and clearly from the results, the X, Y and Z positions shows to be constant all the time.

DefaultSC.EarthMJ2000Eq.X	Sun.EarthICRF.X	Sun.EarthICRF.Y	Sun.EarthICRF.Z	DefaultSC.UTCG
7100	26797447.75467018	-132700760.9778062	-57525179.5873885	01 Jan 2021 00
7086.464342198976	26799233.68286329	-132700456.4246315	-57525047.51699517	01 Jan 2021 00
7011.129674319789	26802033.28788305	-132699978.968536	-57524840.46675649	01 Jan 2021 00
6860.014454132992	26805005.67982491	-132699471.9894968	-57524620.61383648	01 Jan 2021 00
6635.150478727528	26808007.32349062	-132698959.963558	-57524398.57234528	01 Jan 2021 00
6339.115894348704	26811021.67996323	-132698445.7107454	-57524175.56519569	01 Jan 2021 00
5974.915725657839	26814046.33682377	-132697929.6419511	-57523951.77057254	01 Jan 2021 00
5546.242056143192	26817080.42994604	-132697411.9040364	-57523727.25216293	01 Jan 2021 00
5057.52304831911	26820123.12292799	-132696892.6391965	-57523502.07163055	01 Jan 2021 00
4513.880369270696	26823173.54791689	-132696371.995059	-57523276.29299469	01 Jan 2021 00
3921.095948539844	26826230.69679504	-132695850.1432625	-57523049.99068713	01 Jan 2021 00
3285.533687244151	26829293.52952933	-132695327.2609731	-57522823.24153657	01 Jan 2021 00
2614.086426366425	26832360.89513641	-132694803.5443822	-57522596.1306221	01 Jan 2021 00
1914.093442857604	26835431.57021082	-132694279.2021395	-57522368.7484252	01 Jan 2021 00
1193.231463572255	26838504.36865652	-132693754.4366185	-57522141.18270575	01 Jan 2021 00
459.4553928492403	26841578.00966264	-132693229.4664646	-57521913.52827986	01 Jan 2021 00

Figure (66): GMAT report file

6.2.2 Sun Sensors

The sun sensor model takes in two inputs to output a sun vector in body coordinates. Using this sun vector, the ADCS will need one more independent vector to figure out the exact orientation of the CubeSat with respect to the Earth.

The sun sensor is modelled in Simulink to accept two inputs (Figure 67). The position of the sun is in ECI coordinates, calculated using the sun position model and a quaternion that will rotate from Earth-Cantered Inertial (ECI) coordinates to body coordinates. It outputs a sun vector in body coordinates, or the position of the sun as seen by the CubeSat.

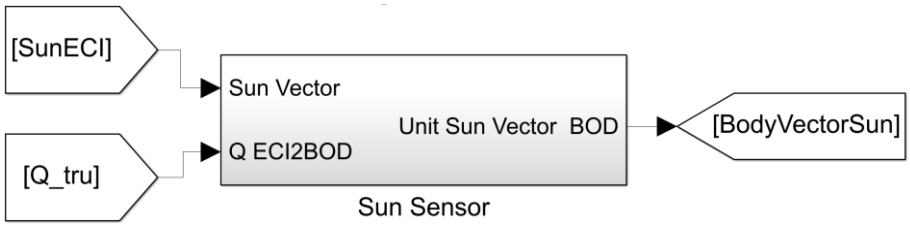


Figure (67): Sun Sensor Model in Simulink

In the sun sensor block (Figure 68) a random quaternion is calculated using the errorquatgen function in MATLAB (more details are provided as comments on the MATLAB script in the appendix). This is applied to the sun vector to add noise. After that, the vector is rotated into body coordinates, by applying the input quaternion, and normalized to give the output sun vector.

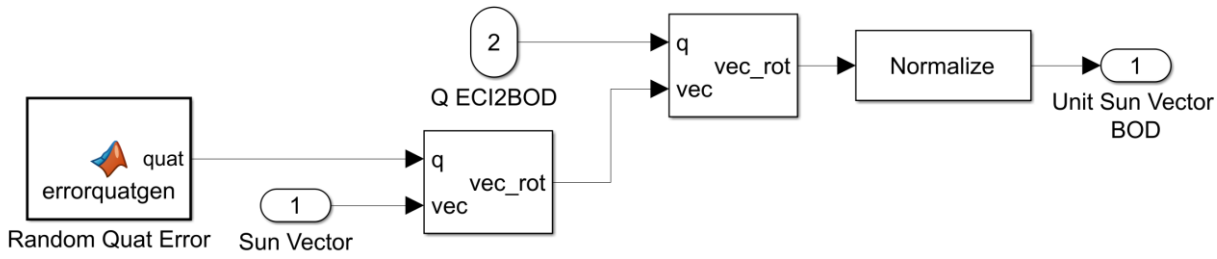


Figure (68): Level Two Sun Sensor Model in Simulink

6.2.3 Magnetometer

The magnetometer takes in the direction of the Earth's magnetic field in ECI coordinates and quaternions that will rotate from ECI coordinates to body coordinates using quaternions rotation toolbox, as its two inputs.

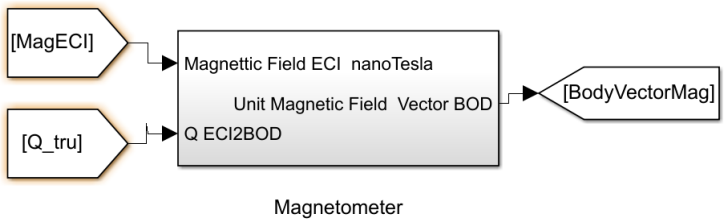


Figure (69): level one of Magnetometer Simulink model

It then outputs the direction of the Earth's magnetic field sensed by the CubeSat which is the magnetic vector in body coordinates. Using a helper block, the International Geomagnetic Reference Field, the magnetometer simulation is able to output a magnetic vector to provide the second independent vector. Then the magnetic vector is rotated to body coordinates and normalized. A noise block is added to the sensor using the function Noise Generator to accurately simulate the space environment of the CubeSat. The magnetic field and the quaternions input are constants for experimenting purposes. In conjunction with the independent sun vector, the magnetometer is able to help provide the exact orientation of the ADCS with respect to Earth.

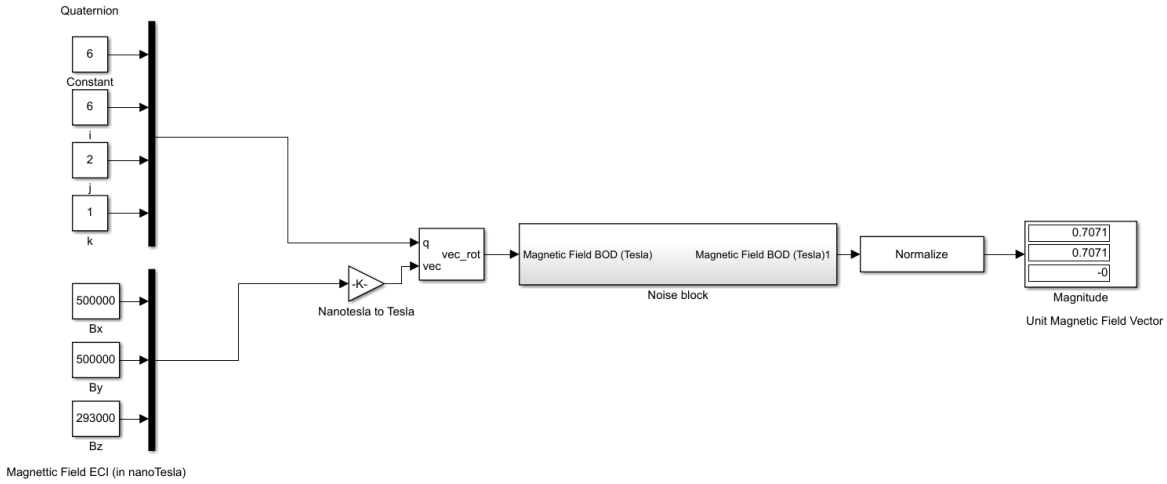


Figure (70): level two Magnetometer Simulink model

6.2.4 Orbit propagator

Propagator is a model whose objective is to determine the position of satellite at any instance of time, with given acceleration and initial velocity. There are some factors which affect the propagation more than others. For example, a satellite which is orbiting around earth will be greatly affected by earth's oblateness rather than gravitational field of sun and moon (as they are very far away). As the number of the factor that taken into account in the model increase, the complexity of the model increases. For less complexity we consider the orbit to be circular as the radius will constant.

6.2.5 International Geomagnetic Reference Field (IGRF)

Figures 71 and 72 show the IGRF model: it takes in the desired position in meters (the output from the orbit propagator), along with a calendar format date and the simulation time. It outputs the magnetic field in ECI coordinates. The sub-block contains the functions Magfield_spherical, locst, rECI2ECEF and msph2inert. The Magfield_spherical function converts the magnetic field into local spherical coordinates [32]. The locst function takes in the date and the longitude and computes the local sidereal time [33]. The rECI2ECEF function converts from ECI to earth-cantered, earth-fixed (ECEF) coordinates [34]. Finally, the msph2inert converts local spherical coordinates into ECI coordinates.

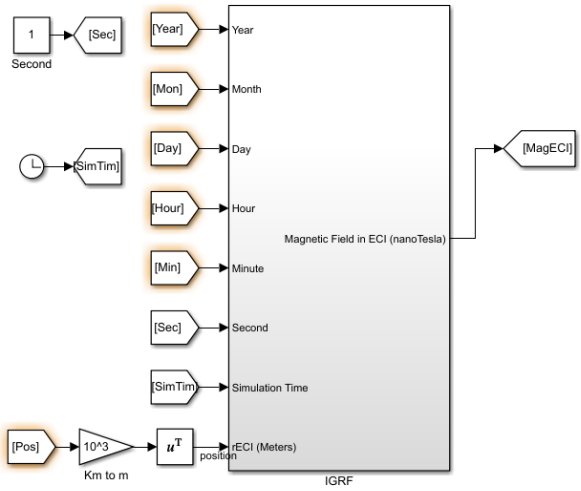


Figure (71): Level 1 of the IGRF Block

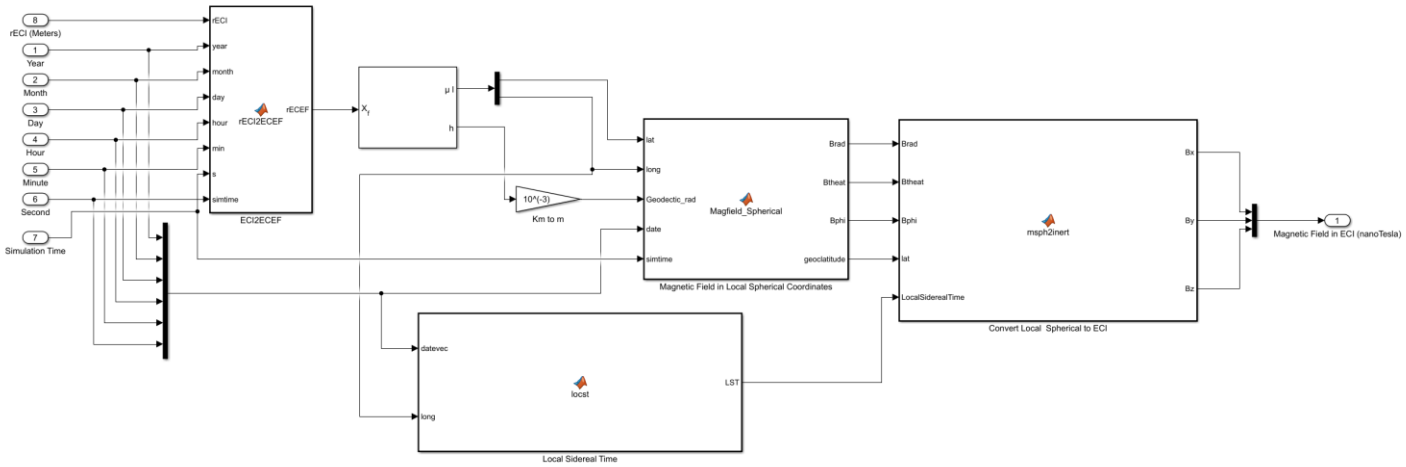


Figure (72): Level 2 of the IGRF Block

We continued by checking if the model works correctly or not. First, we found a realistic position vector through looking and pointing to the UAE [35].

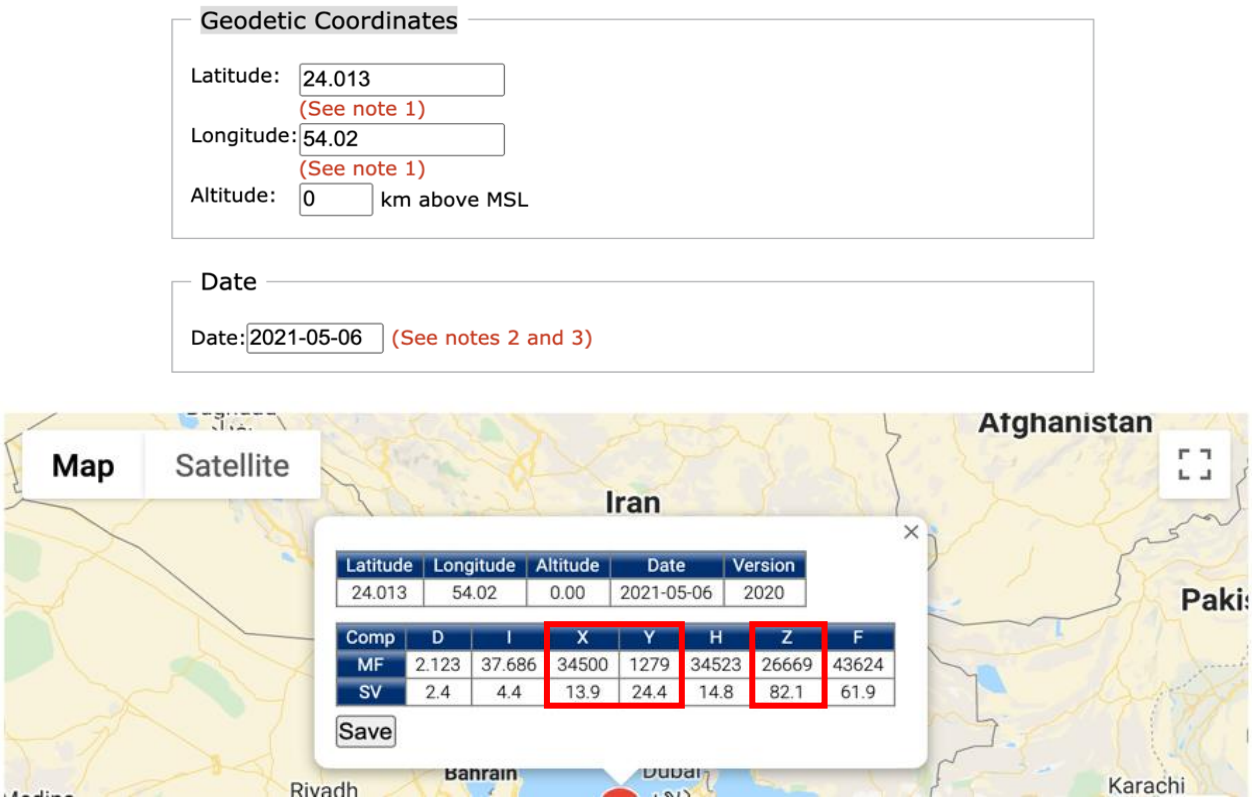


Figure (73): The magnetic field intensity

X is north

Y is east intensity

Z is vertical intensity

These values to be compared with our Magnetic field output from the IGFR, but still, it won't be same (it is way of testing only)

The Geodetic coordinates will be converted to geocentric coordinates [36]. Then the geocentric coordinates are converted to K_m, and this later on will be the input position for the IGRF. To check if the Simulink model of the IGRF works correctly, we substitute the position matrix to see if we can get the magnetic field vector. As results of the above simulated model, we found the following results, but still, the output is not realistic to match the real data:

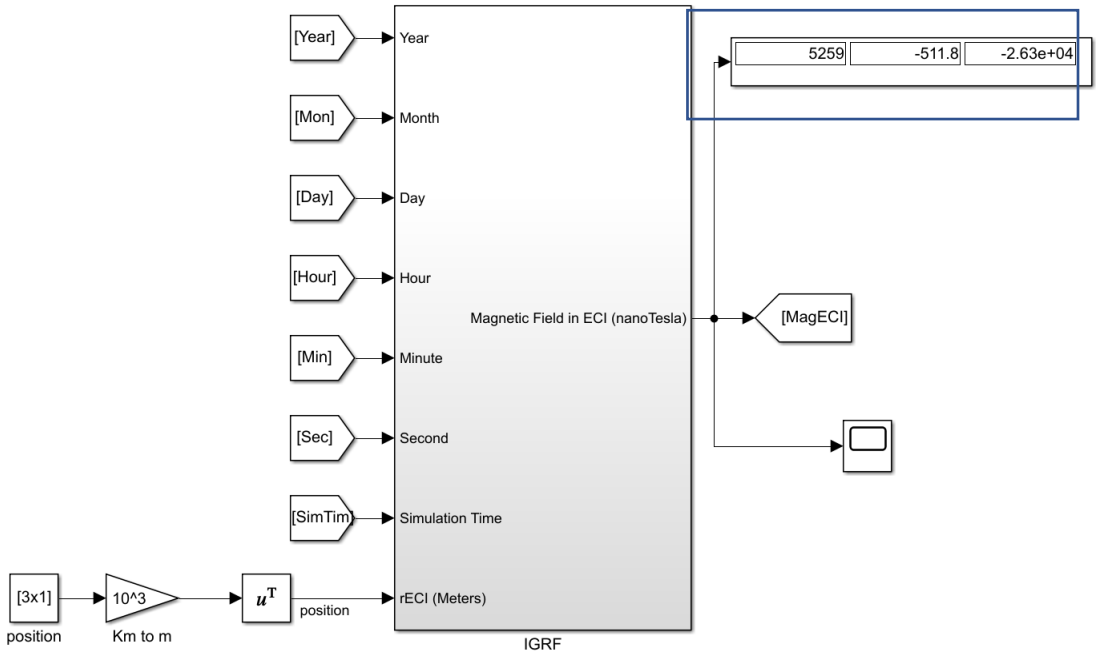


Figure (74): results of the IGRF Block

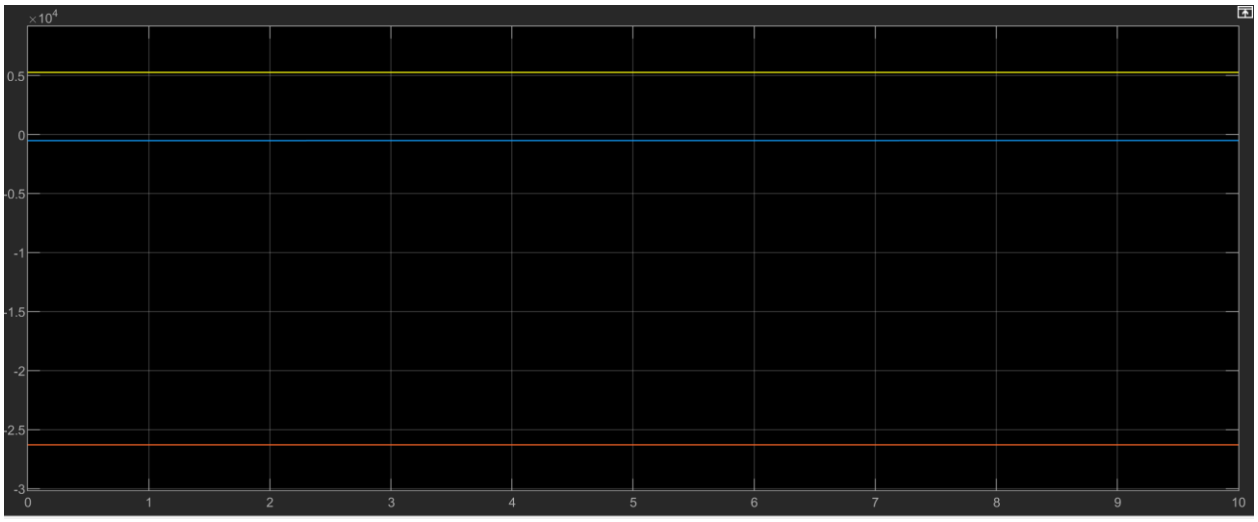


Figure (75): results of the IGRF Block

The magnetic field is constant; therefore, the Simulink model works perfectly.

6.2.6 Tri-Axel Attitude Determination (TRIAD)

The TRIAD gives an estimate of the coordinates of the sun sensor and the magnetometer outputs in vector form. In the first level of the TRIAD it takes in two reference vectors, two body vectors, and outputs the attitude matrix (Figure 76). In the second level of the TRIAD (Figure77), the TRIAD functions generate an attitude matrix using the two reference vectors and two measured vectors.

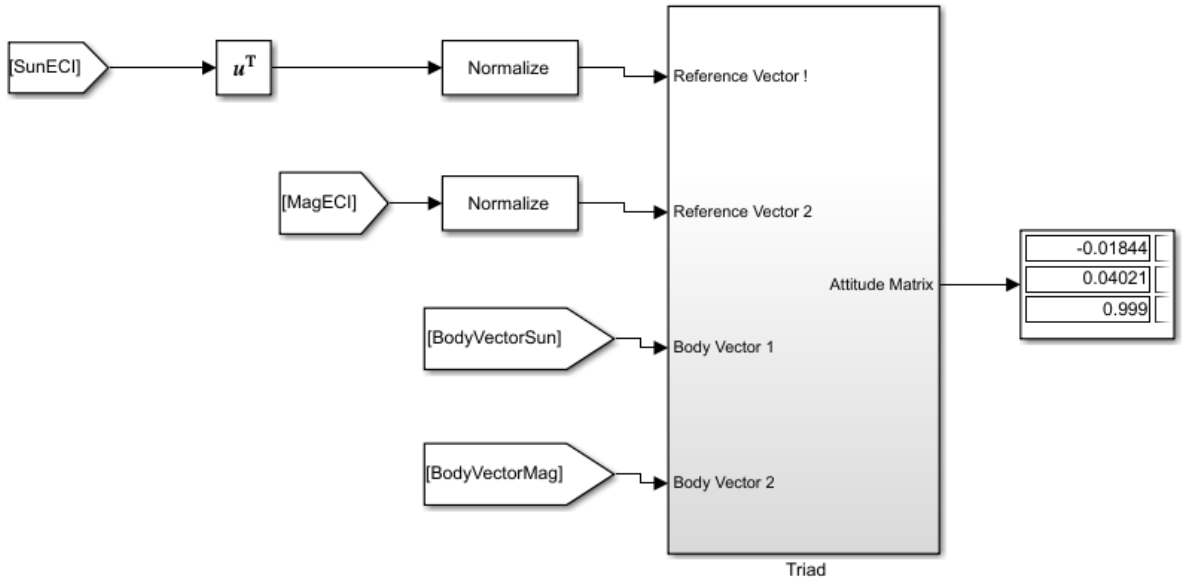


Figure (76): Level 1 of the TRIAD Model in Simulink

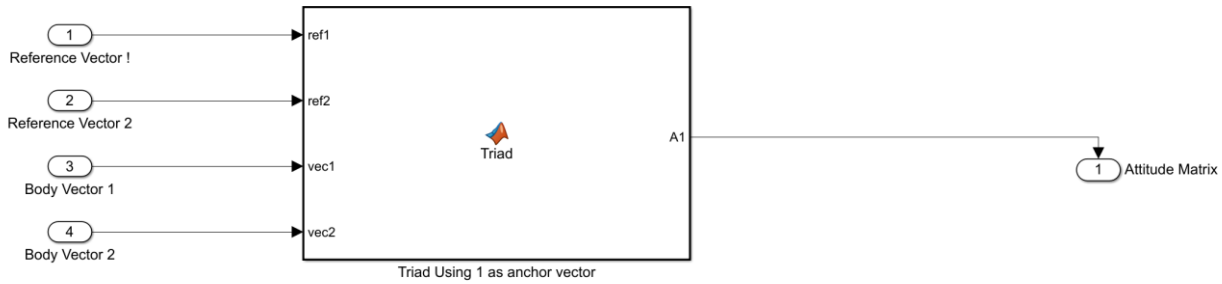


Figure (77): Level 2 of the TRIAD Model in Simulink

6.3 Detailed Design Documentation (ACS)

6.3.1 PD Controller Block

The PD controller subsystem used in KU-Sat is shown in the following figure,

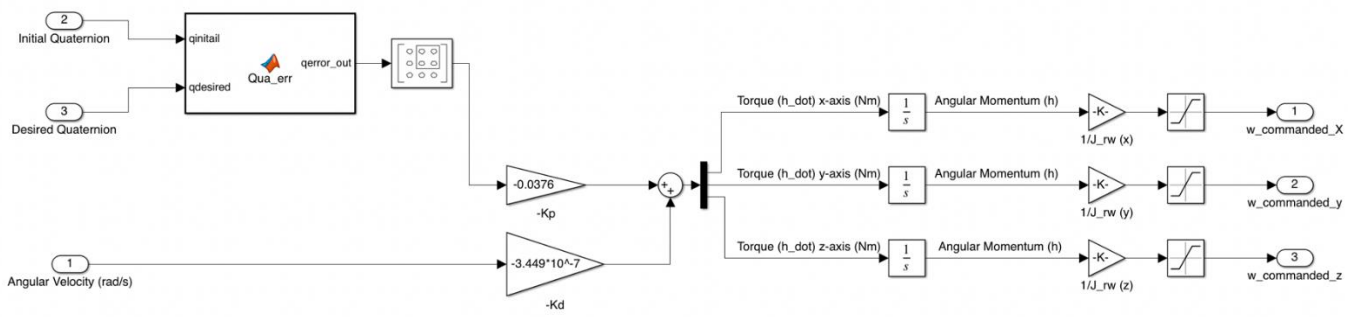


Figure (78): PD controller plant

The PD controller plant have three inputs. First, it uses a function to calculate the quaternion error between the initial quaternion and the desired quaternion, this function can be found in the appendix. Then, it adds the angular velocity to the previously obtained quaternion error to output a torque in each axis (x, y, and z). It then obtains the angular velocity in each axis by integrating the torque. The gains are basically for tuning the controller, adjusting the parameters to the optimum values for the desired control response.

6.3.2 Pre-Processing Block ($\omega \rightarrow V$ Converter)

As the reaction wheel systems requires a voltage command input, this block was created for the purpose of converting the output of the PD controller to match the required input of the reaction wheel system block. Using the equation that was found in previous sections that describes the linear relationship between the angular velocity and the voltage, the following converter was developed.

The linear relationship equation is shown, and its plot was discussed earlier

$$\omega = 44.32V + 25.364$$

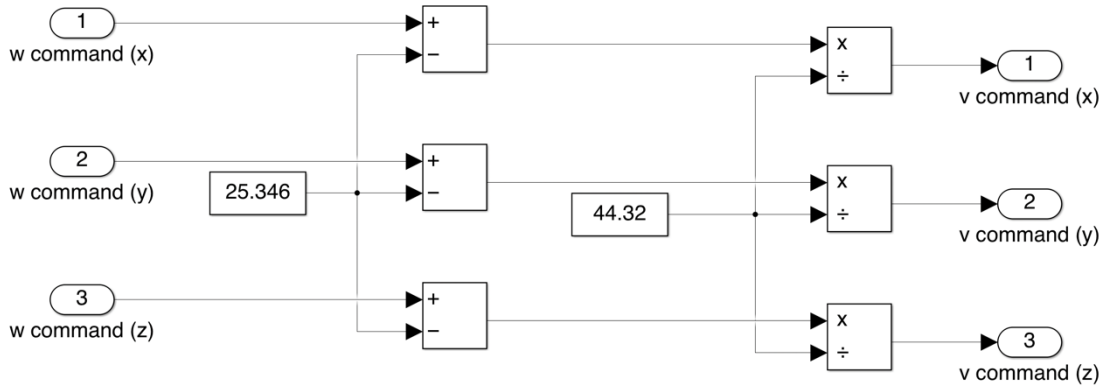


Figure (79): Pre-Processing plant

6.3.3 Reaction Wheel Systems Blocks

There are three reaction wheel system blocks as each reaction wheel will work in one axis. The blocks contain the designed reaction wheel systems using brushless dc motors.

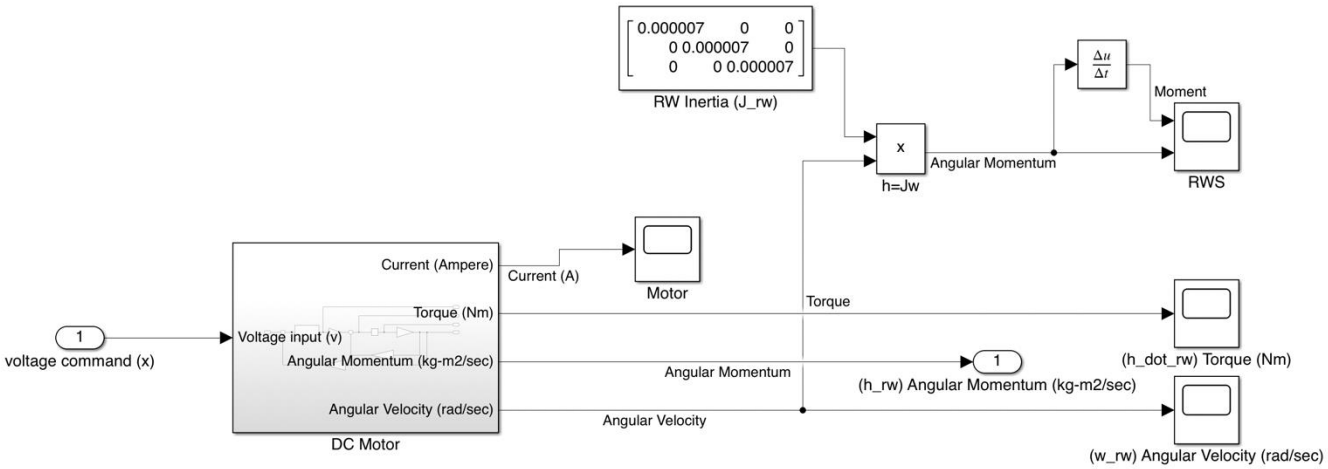


Figure (80): Reaction wheel system block

The input to the reaction wheel system is a voltage command that goes directly to the dc motor block which is shown below.

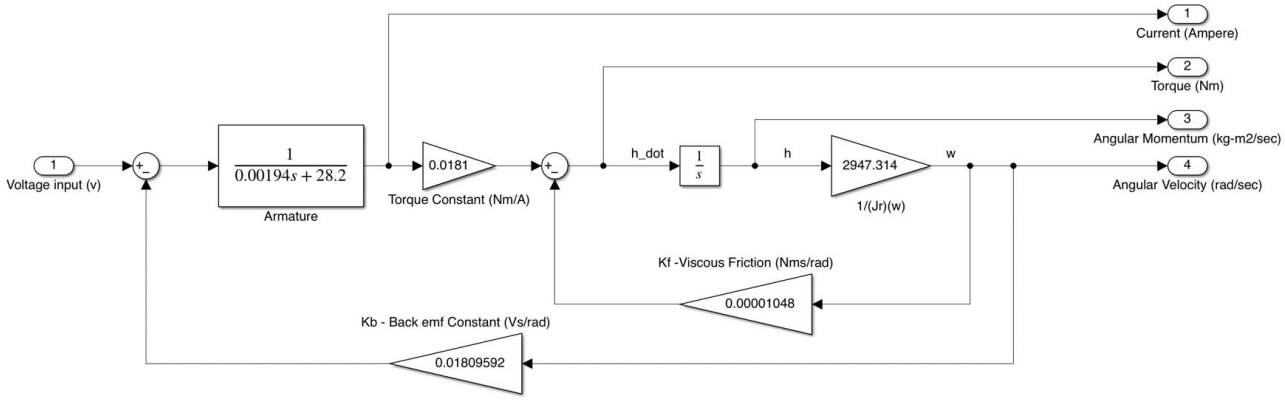


Figure (81): Level 2 Brushless dc motor block

From existing Bibliography, we found out that for any standard Brushless DC Motor, we can represent the SIMULINK Plant such as shown. from the output of the dc motor block, we can find the behaviour of the current, torque, and angular velocity.

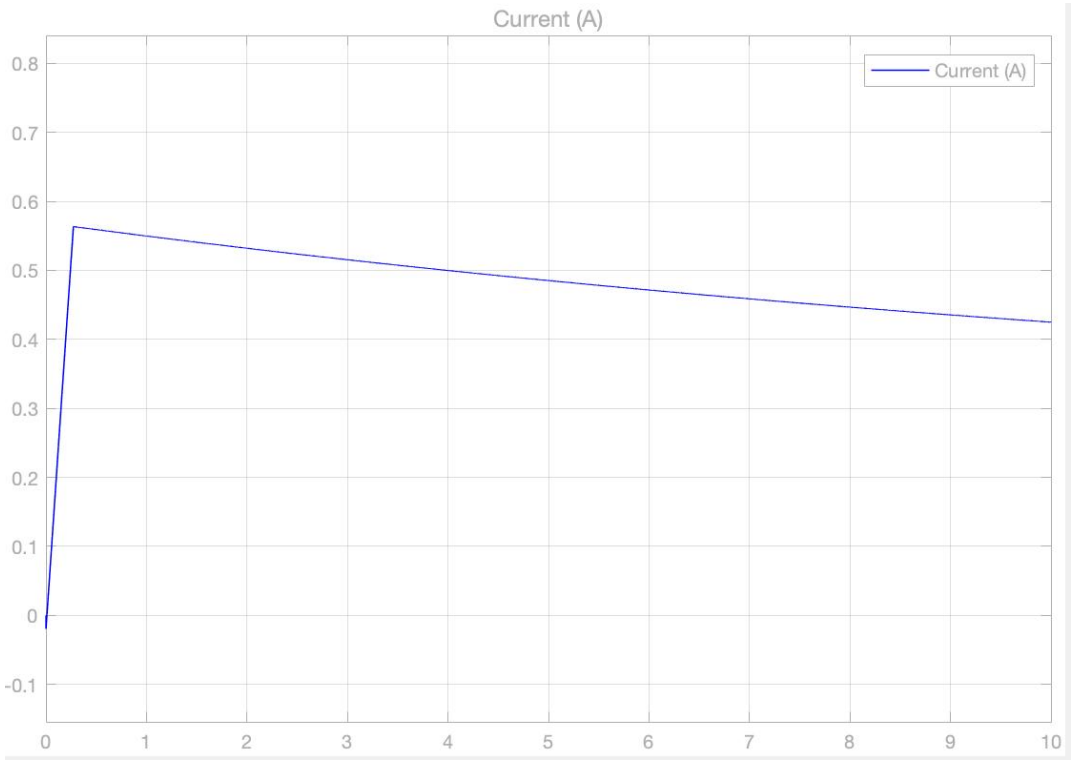


Figure (82): Current output of the reaction wheel system

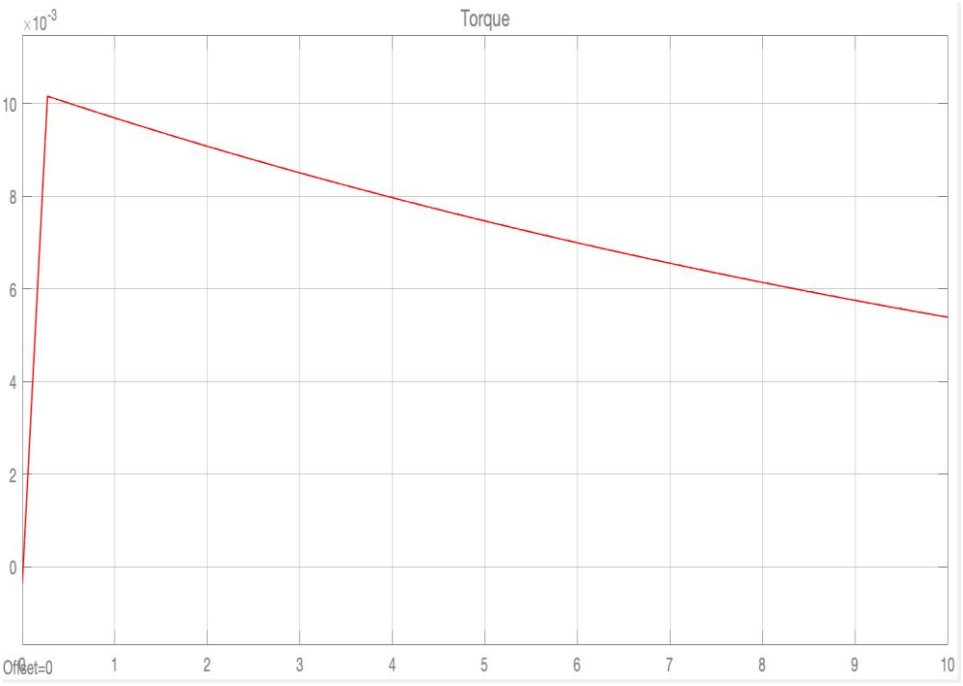


Figure (83): Torque output of the reaction wheel system

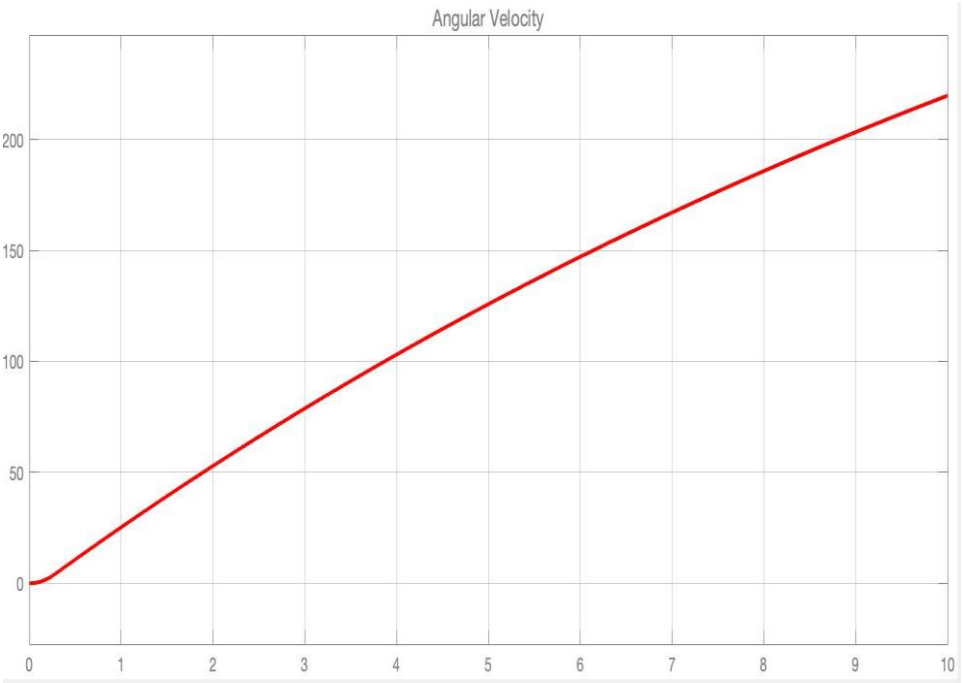


Figure (84): Angular velocity output of the reaction wheel system

Going back to the whole reaction wheel system, we found the moment and angular momentum of the reaction wheel system using the following equation

$$H = J\omega \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot (37)$$

The reaction wheel inertia was calculated in the mathematical model. After incorporating the equation in Simulink, the obtained moment and angular momentum of the reaction wheel are as follows,

$$\text{Angular Momentum} = \begin{bmatrix} 0.001538 & 0 & 0 \\ 0 & 0.001538 & 0 \\ 0 & 0 & 0.001538 \end{bmatrix}$$

$$\text{Moment} = \begin{bmatrix} 0.0001111 & 0 & 0 \\ 0 & 0.0001111 & 0 \\ 0 & 0 & 0.0001111 \end{bmatrix}$$

6.3.4 Satellite Dynamics

Using the satellite dynamic equation of motion integrated in the mathematical model section, the following block was generated,

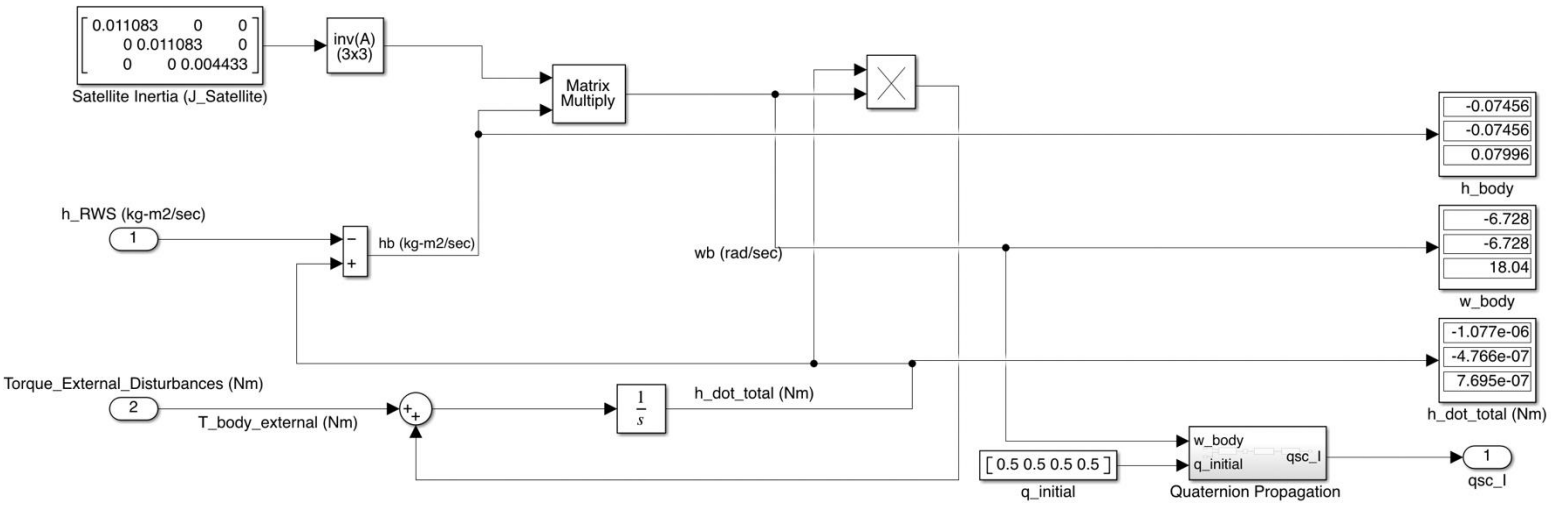


Figure (85): Satellite dynamic block

There are three inputs for the satellite dynamics block which are the satellite inertia, which was found and calculated in the report, the angular momentum coming from the reaction wheel block, and finally, the external torques that were calculated in the early stages of the project and can be found in the mathematical model. After implementing the dynamic equation in Simulink, we had to add the quaternion propagation block to find a random set of data for the sake of testing.

The quaternion propagation block consists of a set of functions that are provided in details in the appendix A. They are used to, first, quaternion integration. Second, quaternion normalization, and finally, changing the quaternion signs. This is how the blook looks like,

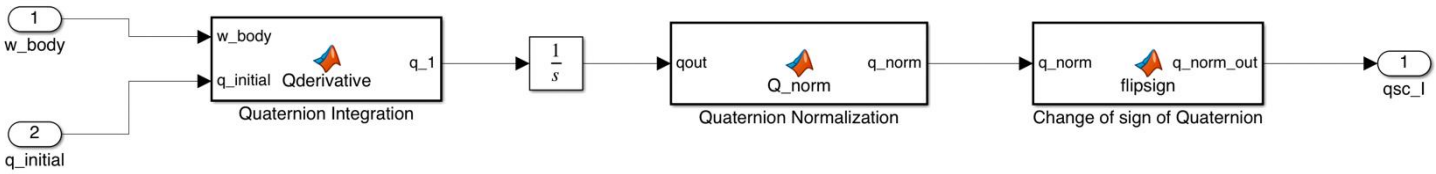


Figure (86): Level 2 Quaternion propagation block

6.3.5 Magnetorquer Rod

This model was generated for a complete design approach but unfortunately, the application of desaturation the reaction wheel system was very intense and we didn't have the proper time or material to proceed and study it thoroughly. But a simple SIMULINK block that represent the magnetorquer rod dynamics was established as follows,

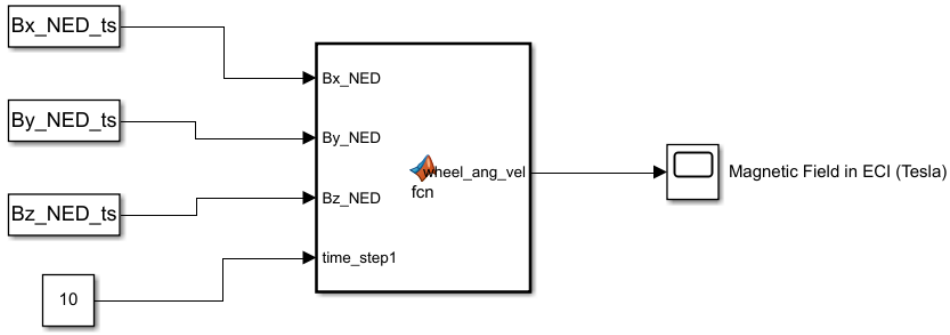


Figure (87): desaturation Simulink model

The Simulink model shows four inputs and a one output function which is available in the Appendix. The input for this function block is the earth magnetic field in NED (North, East, Down) frame which is coming from the IGFR. The function first converts the earth magnetic field from NED to orbital frame using `ned_to_orb` function which will be available in Appendix A. then the earth magnetic field is converted from nanotesla to tesla.

To find the magnetorquer magnetic moment needed to desaturate the reaction wheel this function has been used as mentioned above

$$M = k(B \times \Delta h)$$

Where the wheel angular momentum

$$\Delta h = I\omega$$

In this case the wheel moment of inertia was calculated to be as following

$$I = \begin{bmatrix} 7 \times 10^{-6} & 0 & 0 \\ 0 & 7 \times 10^{-6} & 0 \\ 0 & 0 & 7 \times 10^{-6} \end{bmatrix} kg/m^2$$

We have set the wheel angular velocity to the maximum that our BLDC motor can run which is

$$\omega = [5000 \quad -5000 \quad 5000]\, rpm$$

After computing the code to desaturate the rection wheel we got the following results from Simulink

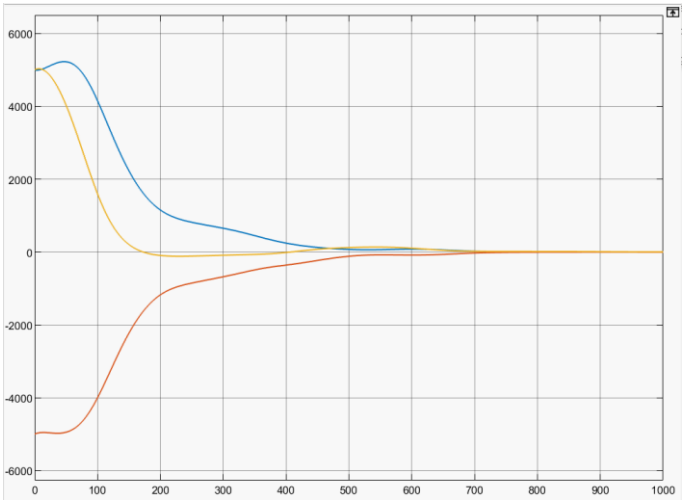


Figure (88): reaction wheel desaturation

The graph shows that the reaction wheel needs around 800s (13.33 min) to desaturate from the maximum angular velocity to reach zero angular velocity.

Chapter 7. CONCLUSION

7.1 Summary of work done

In the beginning, meetings were conducted between the group members and the advisors to discuss the project topics and define the project's difficulties and strengths to decide what scope suits the member’s skills and level of expertise. Afterward, the research and learning stage started, and weekly meetings were conducted with the respected advisors to discuss the ideas and the upcoming steps for developing the main conceptual design. After the selection and approval of the required components from both the advisors and YahSat lab, students have divided the design into two main parts (attitude estimation and attitude control) and each category was also divided to smaller component (Attitude estimation components namely gyroscope, magnetometer and sun sensor. attitude control components namely Reaction Wheel Using BLDC Motor and the Magnetorquer Rod) were each student work focused on one component thoroughly due to the COVID-19 situation. While the situation got better and University facilities were back to hybrid systems; on campus meeting were conducted to combine all parts together and finalizing the final system hardware and software. Testing, fabrication, evaluation, prototype 3-D printing and full ADCS subsystem Simulink model; were done in order to meet the project requirements. At the end of this period, we can gladly conclude that the final design met all requirements and successed in accomplishing the desired subsystem.

7.2 Conclusions

This report summarizes the work behind stage 4 of the continuous work behind building a fully functioning 2U CubeSat for an earth observation mission in LEO. The main project as specified earlier was split into 5 stages and it covers the period from Fall 2016 up until Spring 2022. This phase of the project however was implemented to focus on design the appropriate Attitude Determination and Control Subsystem (ADCS) that can carry on the main mission objective of observing the earth in a two years' timeline.

This stage of the project demanded the current team members to produce a full simulation model implementing a suitable design for KU (2U) CubeSat ADCS using a Reaction Wheels System (RWS) alongside producing a simple prototype that implements and demonstration the dynamics of how the Reaction Wheel System (RWS) is used to orient the CubeSat in space. Where we understood

the fundamental background on attitude dynamics, attitude actuators and attitude control, reached a conceptual understanding of an ADCS subsystem and its use. Then, laid out the requirements behind the ADCS of a CubeSat model developed over the previous stages and designed and built a Simulink model according the requirements as shown in the report.

7.3 Critical appraisal of work done

In the beginning, the students have had several choices to make related to the design of the ADCS subsystem. There are many design and types of each component for the system and each component also differ regarding the controlling and estimation process. The team decided to go with the practical, cheapest, and most effective design to save time and money while keeping in mind that it must fit the requirement and needs of this main project. However, the resources related to some tasks were minimal and not sufficient. The team had chosen to deliver a full Simulink model of the ADCS subsystem instead of building a full-on prototype; because, it is for an educational use only, and the limitation of testing the functionalities since we don't have the facilities for it and the high price for space rated components that makes up the overall full prototype. Moreover, the Simulink model seemed to be the best choice for buildings the subsystem due to the ease of application of its tools and the abundance of online resources, also, it is relevant because in reality this is what engineer develop while receiving the full hardware as a compact one unit. Using MATLAB and Simulink, the group have implemented and simulated the ADCS subsystem and test the overall functionality as if it is a real prototype with the help of the respected students in YahSat lab. For the main task, which is building the system using reaction wheels, students have 3-D printed the 2U CubeSat and successfully controlled its positioning angle using the sized reaction wheels from commercial off the shelf components. Students chose this idea to avoid complexity and it resulted in successfully working ADCS subsystem.

7.4 Recommendations for further work

Some recommendations could improve and enhance the ADCS subsystem ability and functionality for any future application and use. Improvements could be regarding accuracy, for example having a star tracker will increase the accuracy of the attitude estimation of the CubeSat but will also increase the complexity of the attitude estimation model. Also, we could use more complex

algorithm which will for sure have a better more accurate result, for example, changing the simple TRIAD algorithm to Kalman Filter or Extended Kalman Filter. Additionally, the reaction wheel system can be implemented using the pyramid approach rather than the orthogonal for a more redundant system. By working on these suggestions and improvements, the subsystem will be more practical and might satisfying the customer's demands at the best professional level possible.

References

[1] Almahri, A., Alhmoudi, A., Alhindaassi, A., Alkaabi, A., Alyaarbi, S., KU CubeSat (2U)-Stage III Final Project Report, Bachelor thesis, Aerospace Engineering Department, KUSTAR, Abu Dhabi, 2018.

[2] Kerschen, G. and Vila Fernández, M. (2010). Mission analysis of QB50, a nanosatellite intended to study the lower thermosphere. [online] . Available at: https://upcommons.upc.edu/bitstream/handle/2099.1/14693/10-11_Marc_Vila_…

[3] Polat, H., Virgili-Llop, J. and Romano, M. (2016). Statistical Analysis and Classification of Launched CubeSat Missions with Emphasis on the Attitude Control Method. Survey, [online] 5(3), pp.513–530. Available at: https://core.ac.uk/download/pdf/81222665.pdf

[4] Eterno, j., 2021. [online] Available at:<https://ntrs.nasa.gov/api/citations/20110007876/downloads/20110007876.pdf

[5] Larsen, J. and Wisniewski, R., 2010. Attitude Determination And Control System For AAUSAT3. [online] Inpe.br. Available at: http://www.inpe.br/crn/conasat/arquivos/projetos/AAUSAT-III/AAUSAT-3-ADCS-AttitudeDeterminationandControlSystem.pdf

[6] ADCS: Attitude Determination And Control System - ECE3SAT (2021). Available at: http://www.ece3sat.com/cubesatmodules/adcs/

[7] Tikka, T., 2012. Attitude Determination And Control System Implementation For Nanosatellites. [online] Aaltodoc.aalto.fi. Available at: https://aaltodoc.aalto.fi/bitstream/handle/123456789/5196/master_tikka_tuomas_2012.pdf?sequence=1&isAllowed=y

[8] Alali, O., Almazem, M., Abdullah, I., Almazmi, H., Adheem, A., 2U-CubeSat- Phase I Final Project Report, Bachelor thesis, Aerospace Engineering Department, KUSTAR, Abu Dhabi, 2017.

[9] Almahri, A., Alhmoudi, A., Alhindaassi, A., Alkaabi, A., Alyaarbi, S., KU CubeSat (2U)- Stage III Final Project Report, Bachelor thesis, Aerospace Engineering Department, KUSTAR, Abu Dhabi, 2018.

[10] Alshehhi, N., Saeed, H., Alahmed, N., KU CubeSat Phase II Final Project Report, Bachelor thesis, Aerospace Engineering Department, KUSTAR, Abu Dhabi, 2018.

[11]]"A design guide for attitude determination and control systems for picosatellites", 2009. [Online]. Available: https://www.researchgate.net/publication/251889728_A_design_guide_for_attitude_determination_and_control_systems_for_picosatellites.

[12] S. Starin and J. Eterno19.1 Attitude Determination and Control Systems, 19.1 Attitude Determination and Control Systems. Washington, D.C.: NASA Goddard Space Flight Center, 2011.

[13] Alali, O., Almazem, M., Abdullah, I., Almazmi, H., Adheem, A., 2U-CubeSat- Phase I Final Project Report, Bachelor thesis, Aerospace Engineering Department, KUSTAR, Abu Dhabi, 2017.

[14] Compston, D., 2021. *International Geomagnetic Reference Field (IGRF) Model*. [online] Mathworks.com. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/34388-international-geomagnetic-reference-field-igrf-model

[15] Dept.aoe.vt.edu. 2021. [online] Available at: <http://www.dept.aoe.vt.edu/~cdhall/courses/aoe4140/attde.pdf> [Accessed 6 May 2021].

[16] Downloads.hindawi. 2021. Design of Attitude Control Systems for CubeSat-Class Nanosatellite. [online] Available at: <https://downloads.hindawi.com/journals/jcse/2013/657182.pdf.

[17] Yang, Y. 20190206, Spacecraft Modeling, Attitude Determination, and Control, CRC Press. Available from: vbk://9780429822131

[18] Edlerman, E., Agalarian, H., Balabanov, V., Roychowdhury, D. and Gurfil, P., 2021. Development ofthe Orbit and Attitude Control Hardware of the DriveSat CubeSat. [online] Research gate. Available at: https://www.researchgate.net/publication/328901800_Development_of_the_Orbit_and_Attitude_Control_Hardware_of_the_DriveSat_CubeSat

[19] Periscal, A., 2020. Study of feasibility of Attitude Control System for a 3U cubesat based on gravity-boom. [online] Upcommons.upc.edu. Available at: https://upcommons.upc.edu/bitstream/handle/2117/329895/REPORT%20sense%20dades%20confidencials.pdf

[20] J. R. Wertz, Spacecraft Attitude Determination and Control. Kluwer Academic Publishers, 1st ed., 1994.

[21] T. Bak, Spacecraft Attitude Determination - A Magnetometer Approach. PhD thesis, Department of Control Engineering - Aalborg University, 1999. 2nd ed., pp. 9-69.

[22]C. Arbinger and W. Enderle, “Spacecraft attitude determination using a combination of gps attitude sensor and star sensor measurements,” ION GPS 2000, pp. 2634–2642, September 2000.
[23] SILONEX Inc., “Slcd-61n8 - solderable planar photodiode.” http://www1. silonex.com/datasheets/specs/images/pdf/104118.pdf.
[24] InvenSense Inc., “Idg-1215 - dual-axis gyroscope.” http://invensense.com/ mems/gyro/documents/PS-IDG-1215B-00-02.pdf.

[25] InvenSense Inc.,“Isz-1215 - “single-axis z-gyro.” http://invensense.com/mems/ gyro/documents/PS-ISZ-1215B-00-03.pdf.

[26]Invensense.tdk.com. 2021. [online] Available at: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[27] Invensense.tdk.com. 2021. *MPU-9250 Product Specification*. [online] Available at:<https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[28] Invensense.tdk.com. 2021. *MPU-9250 Product Specification*. [online] Available at:<https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[29] Hopcroft, M., 2021. *allan*. [online] Mathworks.com. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/13246-allan
[30] Hoevenaars, A. Hoevenaars, A. (2012) Design, Integration and Verification of the Delfi-n3Xt Reaction Wheel System, Repository.tudelft.nl. Available at: https://repository.tudelft.nl/islandora/object/uuid%3A282ce049-a4e9-4e3d-84c6-8b16f9e5414d
[31] D. Eagle, "Cowell's Method for Earth Satellites", Mathworks.com, 2013. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/39703-cowell-s-method-for-earthsatellites?focused=3772928&tab=function.

[32] D. Compston, “International Geomagnetic Reference Field (IGRF) Model”, Mathworks.com, 2016. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/34388-international-geomagneticreference-field-igrf-model.
[33] "Local Sidereal Time", Small Satellites. [Online]. Available: https://smallsats.org/2013/04/14/local-sidereal-time/.

[34] D. Koblick, "Convert ECI to ECEF Coordinates", Mathworks.com, 2012. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/28233-convert-eci-to-ecefcoordinates.
[35] mailto:www-bgs@bgs.ac.uk, B., 2021. *World Magnetic Model Calculator*. [online] Geomag.bgs.ac.uk. Available at: <http://www.geomag.bgs.ac.uk/data_service/models_compass/wmm_calc.html

[36] Tool-online.com. 2021. *Free online coordinates converter*. [online] Available at: <https://tool-online.com/en/coordinate-converter.php

8. APPENDIX

ADCS Simulink model link:

https://www.mediafire.com/folder/b5e30hv7wkkgl/ADCS+simulink+model

MATLAB codes:

A. Sun Sensor Proving

```
half_FOV = 61.0; % semiaperture of field of view (degree)

% unit vectors parallel to the axes of the sensors

c3 = 1.0/sqrt(3.0);

c4 = 1.0/sqrt(2.0);

u1 = [ 1.0 0.0 0.0];

u2 = [-1.0 0.0 0.0];

u3 = [0.0  1.0 0.0];

u4 = [0.0 -1.0 0.0];

u5 = [0.0 0.0  1.0];

u6 = [0.0 0.0 -1.0];

u7 =  [ c3  c3  c3];  % opposite to u12

u8 =  [ c3  c3 -c3];  % opposite to u14

u9 =  [-c3  c3 -c3];  % opposite to u13

u10 = [-c3  c3  c3];  % opposite to u11

u11 = [ c3 -c3 -c3];

u12 = [-c3 -c3 -c3];

u13 = [ c3 -c3  c3];

u14 = [-c3 -c3  c3];
```

```
u15 = [c4  c4  0];

u16 = [-c4  c4  0];

u17 = [c4  -c4  0];

u18 = [c4  0  c4];

u19 = [-c4  0  c4];

u20 = [c4  0  -c4];

u21 = [0  c4  c4];

u22 = [0  -c4  c4];

u23 = [0  c4  -c4];

u24 = [1/4 ((sqrt(15))/4) 0];

u25 = [-1/4 ((sqrt(15))/4) 0];

u25 = [1/4 -((sqrt(15))/4) 0];

u26 = [0 ((sqrt(15))/4) 1/4];

u27 = [0 ((sqrt(15))/4) -1/4];

u28 = [0 -((sqrt(15))/4) 1/4];

u29 = [((sqrt(15))/4) 0 1/4];

u30 = [-((sqrt(15))/4) 0 1/4];

u31 = [((sqrt(15))/4) 0 -1/4];

u32 = [-1/4 -((sqrt(15))/4) 0];

u33 = [0 -((sqrt(15))/4) -1/4];

u34 = [-((sqrt(15))/4) 0 -1/4];



%loop over direction to the sun (spherical cordinates: lambda is longitude,

%phi is latitude).  We cover an entire sphere
```

```
dlambda = 3.0;  % step in lambda (degree)

dphi = 3.0;     % step in phi (degree)

beta = zeros(34,1);    % vector to memorize angle between each u vector and sun vector

figure(); hold on; grid on; box on;

k = 0;          % counter of number of sun positions

n3 = 0;         % counter of number of sun positions that are visible by at least 3
sensors

n2 = 0;         % counter of number of sun positions that are visible by at least 2
sensors

for lambda = 0.0:dlambda:360.0   % loop over lambda

    for phi = -90.0:dphi:90.0    % loop over phi

        k = k + 1              % counter of sun positions considered

        sun_in_view = 0;

        % unit vector to the sun

        xsun = cosd(lambda)*cosd(phi);

        ysun = sind(lambda)*cosd(phi);

        zsun = sind(phi);

        rsun = [xsun ysun zsun];

        % beta angles

        beta(1) = acosd(dot(u1,rsun));

        beta(2) = acosd(dot(u2,rsun));

        beta(3) = acosd(dot(u3,rsun));

        beta(4) = acosd(dot(u4,rsun));

        beta(5) = acosd(dot(u5,rsun));

        beta(6) = acosd(dot(u6,rsun));
```

```
        beta(7) = acosd(dot(u7,rsun));

        beta(8) = acosd(dot(u8,rsun));

        beta(9) = acosd(dot(u9,rsun));

        beta(10) = acosd(dot(u10,rsun));

        beta(11) = acosd(dot(u11,rsun));

        beta(12) = acosd(dot(u12,rsun));

        beta(13) = acosd(dot(u13,rsun));

        beta(14) = acosd(dot(u14,rsun));

        beta(15) = acosd(dot(u15,rsun));

        beta(16) = acosd(dot(u16,rsun));

        beta(17) = acosd(dot(u17,rsun));

        beta(18) = acosd(dot(u18,rsun));

        beta(19) = acosd(dot(u19,rsun));

        beta(20) = acosd(dot(u20,rsun));

        beta(21) = acosd(dot(u21,rsun));

        beta(22) = acosd(dot(u22,rsun));

        beta(23) = acosd(dot(u23,rsun));

        beta(24) = acosd(dot(u24,rsun));

        beta(25) = acosd(dot(u25,rsun));

        beta(26) = acosd(dot(u26,rsun));

        beta(27) = acosd(dot(u27,rsun));

        beta(28) = acosd(dot(u28,rsun));

        beta(29) = acosd(dot(u29,rsun));

        beta(30) = acosd(dot(u30,rsun));
```

```
        beta(31) = acosd(dot(u31,rsun));

        beta(32) = acosd(dot(u32,rsun));

        beta(33) = acosd(dot(u33,rsun));

        beta(34) = acosd(dot(u34,rsun));

        % loop over the sensors and check if sun is in view

        for i = 1:34

            if(beta(i) < half_FOV)

                sun_in_view = sun_in_view + 1;  % this will give the number of sensors
that can see the sun

            end

        end

        % if(sun_in_view >= 1)  % mark a dot if the sun is in view of at least one sensor

        %  plot3(xsun,ysun,zsun,'r.');

        %end

        if(sun_in_view >= 2)  % mark a star if the sun is in view of at least two sensors

            %plot3(xsun,ysun,zsun,'m.');

            n2 = n2 + 1

        end

        if(sun_in_view >= 3)  % mark a star if the sun is in view of at least three
sensors

           plot3(xsun,ysun,zsun,'r.');

           n3 = n3 + 1      % update the counter for sun positions that are in view of at
least

                            % three sensors simultaneously

        end

        if(sun_in_view == 2)
```

```
            plot3(xsun,ysun,zsun,'c*');

        end

    end

end

% plot the tips of the unit vectors along the axes of the sensors (black for the old

% sensors, blue for the newly added sensors)

plot3(u1(1),u1(2),u1(3),'ok');

plot3(u2(1),u2(2),u2(3),'ok');

plot3(u3(1),u3(2),u3(3),'ok');

plot3(u4(1),u4(2),u4(3),'ok');

plot3(u5(1),u5(2),u5(3),'ok');

plot3(u6(1),u6(2),u6(3),'ok');

plot3(u7(1),u7(2),u7(3),'ob');

plot3(u8(1),u8(2),u8(3),'ob');

plot3(u9(1),u9(2),u9(3),'ob');

plot3(u10(1),u10(2),u10(3),'ob');

plot3(u11(1),u11(2),u11(3),'ob');

plot3(u12(1),u12(2),u12(3),'ob');

plot3(u13(1),u13(2),u13(3),'ob');

plot3(u14(1),u14(2),u14(3),'ob');

plot3(u15(1),u15(2),u15(3),'og');

plot3(u16(1),u16(2),u16(3),'og');

plot3(u17(1),u17(2),u17(3),'og');

plot3(u18(1),u18(2),u18(3),'og');
```

```
plot3(u19(1),u19(2),u19(3),'og');

plot3(u20(1),u20(2),u20(3),'og');

plot3(u21(1),u21(2),u21(3),'og');

plot3(u22(1),u22(2),u22(3),'og');

plot3(u23(1),u23(2),u23(3),'og');

plot3(u24(1),u24(2),u24(3),'oy');

plot3(u25(1),u25(2),u25(3),'oy');

plot3(u26(1),u26(2),u26(3),'oy');

plot3(u27(1),u27(2),u27(3),'oy');

plot3(u28(1),u28(2),u28(3),'oy');

plot3(u29(1),u29(2),u29(3),'oy');

plot3(u30(1),u30(2),u30(3),'oy');

plot3(u31(1),u31(2),u31(3),'oy');

plot3(u32(1),u32(2),u32(3),'oy');

plot3(u33(1),u33(2),u33(3),'oy');

plot3(u34(1),u34(2),u34(3),'oy');

view(30.0,30.0);

hold off;

%print('-djpeg','Sun_in_view_of_1_sensor.jpg');

%print('-djpeg','Sun_in_view_of_2_sensors.jpg');

print('-djpeg','Sun_in_view_of_2_or_3_sensors.jpg');
```

B. International Geomagnetic Reference Field (IGRF)

```
https://www.mediafire.com/file/unxiy8xguxk7psn/IGRF_ku_2u_cubesat.zip/file.
```

C. Allan Variance

```
function [tau, AVAR] = allan(time, data)
if nargin < 2,
    disp('Usage: allan(time, data)');
else

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    numtaus = floor(length(time)/9);
    for b = 1:numtaus
    tau(b) = time(b+1) - time(1);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%
            for h = 1:numtaus
            binvar = 0;
            avebin = 0;
            totalbins = floor(time(length(time))/ tau(h));
            lengthbins = floor(length(time) / totalbins);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            for j = 1:totalbins
                avebin(j) = mean(data((lengthbins * (j-1)) + 1:lengthbins * (j)));
            end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            for j = 1:(totalbins-1)
                binvar(j) = (avebin(j+1) - avebin(j))^2;
            end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            AVAR(h) = sqrt(sum(binvar)/2*(totalbins-1));
            end

       % plot the allan variance on a log log scale
       loglog(tau,AVAR)
       grid on;

end
end
```

D. myfft function

```
function [f, amp] = myfft(data,tsamp)
Fs = 1/tsamp;


T = tsamp;


L = length(data);


x = data;


NFFT = 2^nextpow2(L);


Y=fft(x,NFFT)/L;


f=Fs/2*linspace(0,1,NFFT/2+1);


amp= 2*abs(Y(1:NFFT/2+1));


loglog(f,amp)
```

```
title('Single-sided amplitude spectrum of data(t)')
xlabel('Frequency(Hz))')
ylabel('|Data(f)|')
grid on;
```

E. Gyroscope Model

```
time = 0:1/92:1613/92;


plot(time,RealGyroData)


xlabel('Time(s)');
ylabel('Angular Rate (rpm)')
grid on;
mean(RealGyroData)
mean(RealGyroData)*0.00763
realGyroNoBias = RealGyroData - mean(RealGyroData);

[tau, AVAR] = allan(time, realGyroNoBias*0.00763);
tau(96)
AVAR(92)

% from TF
wn  = 2000* pi *2;
w = 2*pi*25;
zeta = sqrt(2)/2;
z=zeta;
simgyro = simgyro(1:end-1);
```

F. Real Gyroscope Simulation

```
plot(time,[RealGyroData,simgyro1])


simgyrobias = simgyro1 - mean(simgyro1);


[f,amp]= myfft(realGyroNoBias * 0.00763, 1/92);


hold on;
[f,amp]= myfft(simgyrobias*0.00763, 1/92);
legend('Gyro Model','Real Gyro')
grid on;
```

ADS system Simulink functions:

g. Julian function

```
function wholejd = julian (month, day, year)


wholejd=0;


y = year;


m = month;
```

```
b = 0;

c = 0;

if (m <= 2)

   y = y - 1;

   m = m + 12;

end

if (y < 0)

   c = -.75;

end

% check for valid calendar date

if (year < 1582)

   % null

elseif (year > 1582)

   a = fix(y / 100);

   b = 2 - a + floor(a / 4);

elseif (month < 10)

   % null

elseif (month > 10)

   a = fix(y / 100);

   b = 2 - a + floor(a / 4);

elseif (day <= 4)

   % null

elseif (day > 14)

   a = fix(y / 100);
```

```
   b = 2 - a + floor(a / 4);

   fprintf('\n\n  this is an invalid calendar date!!\n');

   return;

end

jd = fix(365.25 * y + c) + fix(30.6001 * (m + 1));

wholejd = jd + day + b + 1720994.5;
```

H. fracjday function

```
function fracjd=fracjday(uthr, utmin, utsec)

fracjd=uthr / 24 + utmin / 1440 + utsec / 86400;
```

I. sun1 function

```
function rsun  = sun1 (jdate)

atr = pi / 648000;

rsun = zeros(3, 1);

% time arguments

djd = jdate - 2451545;

t = (djd / 36525) + 1;

% fundamental arguments (radians)

gs =2.0*pi*((0.993126+0.0027377785*djd)-fix(0.993126+0.0027377785*djd));

lm = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));

ls = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));

g2 = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));

g4 = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));

g5 = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));
```

```
rm = 2.0*pi*((0.606434+0.03660110129*djd)-fix(0.606434+0.03660110129*djd));

% geocentric, ecliptic longitude of the sun (radians)

plon = 6910*sin(gs)+72*sin(2*gs)-17*t*sin(gs);

plon = plon-7*cos(gs-g5)+6*sin(lm-ls)+5*sin(4*gs-8*g4+3*g5);

plon = plon-5*cos(2*(gs-g2))-4*(sin(gs-g2)-cos(4*gs-8*g4+3*g5));

plon = plon+3*(sin(2*(gs-g2))-sin(g5)-sin(2*(gs-g5)));

plon = ls+atr*(plon-17*sin(rm));

% geocentric distance of the sun (kilometers)

rsm = 149597870.691*(1.00014-0.01675*cos(gs)-0.00014*cos(2*gs));

% obliquity of the ecliptic (radians)

obliq = atr*(84428-47*t+9*cos(rm));

% geocentric, equatorial right ascension and declination (radians)

a = sin(plon)*cos(obliq);

b = cos(plon);

epsilon = 0.0000000001;

pidiv2 = 0.5 * pi;

dontenterflag=0;

c=0;

if (abs(a) < epsilon)

    rascy = (1 - sign(b)) * pidiv2;

    dontenterflag=1;

else

    c = (2 - sign(a)) * pidiv2;

end
```

```
if ((abs(b) < epsilon)&&dontenterflag~=1)

    rascy = c;

else

    rascy = c + sign(a) * sign(b) * (abs(atan(a / b)) - pidiv2);

end

rasc = rascy;

decl = asin(sin(obliq) * sin(plon));

% geocentric position vector of the sun (kilometers)

rsun(1) = rsm * cos(rasc) * cos(decl);

rsun(2) = rsm * sin(rasc) * cos(decl);

rsun(3) = rsm * sin(decl);
```

J. errorquatgen() MATLAB Code

```
function quat = errorquatgen()

% genertaged quaternion for rotation of given angle (rad) around random axis

% computing first term of quaternioin

angle = normrnd(0,0.0029);

q0 = cos(angle/2);

% random axis

a = rand;

b = rand;

c = rand;

axis = [a,b,c];

% normalise the random axis
```

```
n = norm(axis);

a = a/n;

b = b/n;

c = c/n;

% compting quaternion

q1 = a*sin(angle/2);

q2 = b*sin(angle/2);

q3 = c*sin(angle/2);

quat = [q0 q1 q2 q3];

end
```

K. Triad(ref1,ref2,vec1,vec2) MATLAB Code (A1)

```
function A1=Triad(ref1,ref2,vec1,vec2)

%This function generates an attitude matrix two reference and two measured vectors.

%Calculate body tirad

triad1body=vec1./norm(vec1);

triad2body=cross(triad1body,vec2)./norm(cross(triad1body,vec2));

triad3body=cross(triad1body,triad2body)./norm(cross(triad1body,triad2body));

%Calculate reference tirad

triad1rot=ref1./norm(ref1);

triad2rot=cross(triad1rot,ref2)./norm(cross(triad1rot,ref2));

triad3rot=cross(triad1rot,triad2rot)./norm(cross(triad1rot,triad2rot));

%Calculate attituce matrix

A1= triad1body*(triad1rot.') + triad2body*(triad2rot.') + triad3body*(triad3rot.');
```

L. The rECI2ECEF function (IGRF)

```
function rECEF=rECI2ECEF(rECI, year, month, day, hour, min, s, simtime )

d2000=367*year-
floor((7*(year+(floor(month+9)/12)))/4)+floor(275*month/9)+((hour+(min/60)+((s+simtime)/36
00))/24)+day-730531.5;

theta=280.46061837+360.98564736628*d2000;

trans_mat=[cosd(theta) sind(theta) 0; -sind(theta) cosd(theta) 0; 0 0 1];

rECEF= rECI*trans_mat;

end
```

M. The Magfield_spherical function

```
function [Brad,Btheat,Bphi,geoclatitude] = Magfield_Spherical(lat,long,Geodectic_rad,date,
simtime)

%Magnetic field in Spherical Coordinates

if (lat>-0.00000001 && lat<0.00000001)

    lat=0.00000001;

elseif(lat<180.00000001 && lat>179.99999999)

    lat=179.99999999;

end

costheta = cos((90 - lat)*pi/180);

sintheta = sin((90 - lat)*pi/180);

 ra = 6378.137; f = 1/298.257223563; b = ra*(1 - f);

    rho = hypot(ra*sintheta, b*costheta);

    r = sqrt( Geodectic_rad.^2 + 2*Geodectic_rad.*rho + ...

        (ra^4*sintheta.^2 + b^4*costheta.^2) ./ rho.^2 );

    cd = (Geodectic_rad + rho) ./ r;
```

```
    sd = (ra^2 - b^2) ./ rho .* costheta.*sintheta./r;

    oldcos = costheta;

    costheta = costheta.*cd - sintheta.*sd;

    sintheta = sintheta.*cd + oldcos.*sd;

    geoclatitude=90-acos(costheta)*180/pi;

 phi = long*pi/180;

setdate=[2015 1 1 0 0 0];

givenjd=findjd(date)+simtime/86400;

setjd=findjd(setdate);

days=givenjd-setjd;

a=6371.2;

gm=[0;1;0;1;2;0;1;2;3;0;1;2;3;4;0;1;2;3;4;5;0;1;2;3;4;5;6;0;1;2;3;4;5;6;7;0;1;2;3;4;5;6;7;
8;0;1;2;3;4;5;6;7;8;9;0;1;2;3;4;5;6;7;8;9;10;0;1;2;3;4;5;6;7;8;9;10;11;0;1;2;3;4;5;6;7;8;9
;10;11;12;0;1;2;3;4;5;6;7;8;9;10;11;12;13];

gn=[1;1;2;2;2;3;3;3;3;4;4;4;4;4;5;5;5;5;5;5;6;6;6;6;6;6;6;7;7;7;7;7;7;7;8;8;8;8;8;8;8;8;
8;9;9;9;9;9;9;9;9;9;9;10;10;10;10;10;10;10;10;10;10;10;11;11;11;11;11;11;11;11;11;11;11
;12;12;12;12;12;12;12;12;12;12;12;12;12;13;13;13;13;13;13;13;13;13;13;13;13];

 gsvi=[10.3000000000000;18.1000000000000;-13.0500000000000;-
5.71580000000000;1.81870000000000;8.50000000000000;-16.8400000000000;-1.35550000000000;-
7.98480000000000;-3.06250000000000;1.10680000000000;-35.6090000000000;8.57580000000000;-
3.17990000000000;-1.57500000000000;5.08330000000000;-9.99080000000000;-
0.470620000000000;3.10590000000000;2.73610000000000;-4.33120000000000;-1.89030000000000;-
10.4610000000000;20.9220000000000;-
6.54820000000000;0.698040000000000;1.07470000000000;8.04370000000000;-7.09390000000000;-
14.4800000000000;26.6220000000000;1.23490000000000;-3.70470000000000;-
1.93750000000000;0.129450000000000;10.0550000000000;0;-33.6490000000000;20.7100000000000;-
5.34720000000000;5.93220000000000;0.686520000000000;-
1.00270000000000;0.188010000000000;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];

 gvali=[-29442;-1501;-
3667.60000000000;5218.50000000000;1452.10000000000;3376.80000000000;-
7202.40000000000;2373.40000000000;460.110000000000;3970.80000000000;4503;471.140000000000;
-700.490000000000;52.0620000000000;-1831.70000000000;3661;1478.60000000000;-
663.110000000000;-
```

```
349.420000000000;2.87640000000000;1010.60000000000;1279.70000000000;1086.40000000000;-
1294.20000000000;-157.700000000000;30.7140000000000;-47.6230000000000;2187.90000000000;-
2699.20000000000;-196.930000000000;1060.80000000000;185.230000000000;58.0400000000000;-
6.78110000000000;4.40140000000000;1216.60000000000;589.880000000000;-947.790000000000;-
132.540000000000;-550.770000000000;198.730000000000;80.3230000000000;-39.8590000000000;-
1.25340000000000;512.790000000000;1121.20000000000;336.820000000000;-
273.840000000000;39.4630000000000;-448.090000000000;-1.73980000000000;65.5420000000000;-
23.5140000000000;-6.39500000000000;-342.810000000000;-
1532.70000000000;21.0690000000000;82.6400000000000;-58.4350000000000;133.050000000000;-
28.9240000000000;42.0910000000000;19.6380000000000;-4.77860000000000;-
2.13710000000000;1067.80000000000;-699.580000000000;-940.810000000000;655.940000000000;-
191.610000000000;95.0540000000000;-65.8820000000000;9.92090000000000;38.6920000000000;-
1.76300000000000;1.08810000000000;2.02990000000000;-1254.40000000000;-
179.410000000000;318.050000000000;779.060000000000;-
389.530000000000;300.620000000000;20.8300000000000;58.5270000000000;-17.5580000000000;-
10.2170000000000;1.88650000000000;-2.50330000000000;0;0;-
1557.20000000000;619.010000000000;641.570000000000;-492.060000000000;695.880000000000;-
90.3090000000000;213.710000000000;-
14.2790000000000;20.4220000000000;2.83880000000000;5.01840000000000;-1.13550000000000;-
0.167020000000000];

hm=[0;1;0;1;2;0;1;2;3;0;1;2;3;4;0;1;2;3;4;5;0;1;2;3;4;5;6;0;1;2;3;4;5;6;7;0;1;2;3;4;5;6;7;
8;0;1;2;3;4;5;6;7;8;9;0;1;2;3;4;5;6;7;8;9;10;0;1;2;3;4;5;6;7;8;9;10;11;0;1;2;3;4;5;6;7;8;9
;10;11;12;0;1;2;3;4;5;6;7;8;9;10;11;12;13];


hn=[1;1;2;2;2;3;3;3;3;4;4;4;4;4;5;5;5;5;5;5;6;6;6;6;6;6;6;7;7;7;7;7;7;7;7;8;8;8;8;8;8;8;8;
8;9;9;9;9;9;9;9;9;9;9;10;10;10;10;10;10;10;10;10;10;10;11;11;11;11;11;11;11;11;11;11;11;11
;12;12;12;12;12;12;12;12;12;12;12;12;12;13;13;13;13;13;13;13;13;13;13;13;13;13;13];

 hsvi=[0;-26.6000000000000;0;-47.4580000000000;-12.2110000000000;0;25.1070000000000;-
0.774600000000000;1.42300000000000;0;-7.19420000000000;20.7400000000000;6.06580000000000;-
3.84550000000000;0;6.09990000000000;13.0650000000000;-
5.64750000000000;7.54300000000000;0;0;0;-31.3830000000000;-
6.97400000000000;1.09140000000000;2.09410000000000;0.671690000000000;0;28.3760000000000;11
.5840000000000;-4.09570000000000;-3.70470000000000;-3.70470000000000;0.242180000000000;-
0.129450000000000;0;-20.1090000000000;16.8250000000000;4.14200000000000;13.3680000000000;-
2.96610000000000;-
2.05960000000000;0.752050000000000;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];

 hvali=[0;4797.10000000000;0;-4928.70000000000;-555.900000000000;0;-
353.030000000000;474.250000000000;-425.640000000000;0;1567.80000000000;-
738.410000000000;378.380000000000;-243.670000000000;0;480.880000000000;1514;-
561.450000000000;35.4960000000000;70.2960000000000;0;-
393.180000000000;496.150000000000;586.810000000000;-
363.970000000000;16.9860000000000;42.0480000000000;0;-1918.90000000000;-
564.740000000000;116.730000000000;301.310000000000;20.9930000000000;-66.3580000000000;-
```

```
1.42400000000000;0;677.020000000000;-1026.30000000000;550.880000000000;-
390.350000000000;240.260000000000;39.1320000000000;-22.8120000000000;1.31610000000000;0;-
2751.90000000000;1173.40000000000;979.200000000000;-383.360000000000;-
232.470000000000;135.700000000000;7.53350000000000;-
10.3360000000000;5.11600000000000;0;778.520000000000;-
84.2770000000000;760.290000000000;514.230000000000;-583.930000000000;-24.7920000000000;-
84.1810000000000;-22.9110000000000;-3.18570000000000;-5.16460000000000;0;-
46.6390000000000;818.100000000000;-229.580000000000;-263.470000000000;126.740000000000;-
18.8240000000000;-109.130000000000;-31.8640000000000;-22.0370000000000;-5.44070000000000;-
1.39200000000000;0;-986.730000000000;318.050000000000;1233.50000000000;-
1071.20000000000;100.210000000000;145.810000000000;-
11.7050000000000;17.5580000000000;5.10870000000000;-8.48920000000000;-
0.278150000000000;0.397440000000000;0;-1557.20000000000;619.010000000000;2053;-
492.060000000000;-835.050000000000;-45.1550000000000;106.860000000000;-
14.2790000000000;27.2290000000000;14.1940000000000;-3.01100000000000;-1.13550000000000;-
0.445390000000000];

N=max(gn);

g=zeros(N,N+1);

Geodectic_rad=zeros(N,N+1);

for x=1:length(gn)

    g(gn(x),gm(x)+1) = gvali(x) + gsvi(x)*days/365;

    Geodectic_rad(hn(x),hm(x)+1) = hvali(x) + hsvi(x)*days/365;

end

Brad=0; Btheat=0; Bphi=0; P20=0; dP20=0;

P11=1; P10=P11;

dP11=0; dP10=dP11;

for m=0:N

    for n=1:N

        if m<=n

            % Calculate Legendre polynomials and derivatives recursively

            if n==m

                P2 = sintheta*P11;
```

```
            dP2 = sintheta*dP11 + costheta*P11;

            P11=P2; P10=P11; P20=0;

            dP11=dP2; dP10=dP11; dP20=0;

        elseif n==1

              P2 = costheta*P10;

              dP2 = costheta*dP10 - sintheta*P10;

              P20=P10; P10=P2;

              dP20=dP10; dP10=dP2;

        else

            K = ((n-1)^2-m^2)/((2*n-1)*(2*n-3));

            P2 = costheta*P10 - K*P20;

            dP2 = costheta*dP10 - sintheta*P10 - K*dP20;

            P20=P10; P10=P2;

            dP20=dP10; dP10=dP2;

        end

        % Calculate Br, Bt, and Bp

        Brad = Brad + (a/r)^(n+2)*(n+1)*...

            ((g(n,m+1)*cos(m*phi) + Geodectic_rad(n,m+1)*sin(m*phi))*P2);

        Btheat = Btheat + (a/r)^(n+2)*...

            ((g(n,m+1)*cos(m*phi) + Geodectic_rad(n,m+1)*sin(m*phi))*dP2);

        Bphi = Bphi + (a/r)^(n+2)*...

            (m*(-g(n,m+1)*sin(m*phi) + Geodectic_rad(n,m+1)*cos(m*phi))* P2);

    end

end
```

```
end

Btheat =-Btheat;

Bphi= -(Bphi/(sintheta));

end

function jd=findjd(datevec)

yr=datevec(1); mon=datevec(2); day=datevec(3); hr=datevec(4); min=datevec(5);
sec=datevec(6);

        jd = 367.0 * yr  ...

             - floor( (7 * (yr + floor( (mon + 9) / 12.0) ) ) * 0.25 )    ...

             + floor( 275 * mon / 9.0 ) ...

             + day + 1721013.5  ...

             + ( (sec/60.0 + min ) / 60.0 + hr ) / 24.0;

end

    function c=hypota(a,b)

        c = sqrt(abs(a).^2 + abs(b).^2)

    end
```

N. The locst function

```
function LST = locst(datevec,long)

yr=datevec(1); mon=datevec(2); day=datevec(3); hr=datevec(4); min=datevec(5);
sec=datevec(6);

UT=( (sec/60.0 + min ) / 60.0 + hr );

J0 = 367.0 * yr  ...

- floor( (7 * (yr + floor( (mon + 9) / 12.0) ) ) * 0.25 )    ...

+ floor( 275 * mon / 9.0 ) ...

+ day + 1721013.5;
```

```
JD=J0+(UT/24);

JC = (J0 - 2451545.0)/36525;

GST0 = 100.4606184 + 36000.77004*JC + 0.000387933*JC^2 - 2.583e-8*JC^3; %[deg]

GST0 = mod(GST0, 360);  % GST0 range [0..360]

GST = GST0 + 360.98564724*UT/24;

GST = mod(GST, 360);  % GST0 range [0..360]

LST = GST + long;

LST = mod(LST, 360);  % LST range [0..360]
```

O. The msph2inert function

```
function [Bx,By,Bz] = msph2inert(Brad,Btheat,Bphi,lat, LocalSiderealTime)
```

```
lat=lat*pi/180; LocalSiderealTime=LocalSiderealTime*pi/180;

% Coordinate transformation

Bx = (Brad*cos(lat)+Btheat*sin(lat))*cos(LocalSiderealTime) - Bphi*sin(LocalSiderealTime);

By = (Brad*cos(lat)+Btheat*sin(lat))*sin(LocalSiderealTime) + Bphi*cos(LocalSiderealTime);

Bz = (Brad*sin(lat)+Btheat*cos(lat));
```

ACS Simulink model functions:

P. PD-Controller Block – Calculating the quaternion error

```
function qerror_out= Qua_err(qinitail, qdesired)
 qdesired = qdesired(:);
 qest = qinitail(2:4);
 qlast = qinitail(1);
 M = [0,-qest(3),qest(2);
     qest(3),0,-qest(1);
     -qest(2),qest(1),0];
```

```
 Qerr = [qlast*eye(3)-M,-qest;qest',qlast];
 qerror = Qerr*qdesired;

 if qerror(1) < 0.0
    qerror_out = -qerror;
 else
     qerror_out= qerror;
 end
 end
```

Q. Satellite Dynamic Block – Quaternion Sub-Block – Quaternion Integration

```
function q_1 = Qderivative(w_body,q_initial)
    q_last = q_initial(1);
    q = q_initial(2:4);
    M = [0,-q(3),q(2);q(3),0,-q(1);-q(2),q(1),0];
    q_1 = 0.5*[q_last*eye(3)+M; -q']*w_body;
end
```

R. Satellite Dynamic Block – Quaternion Sub-Block – Quaternion Normalization

```
function q_norm= Q_norm(qout)
 q_norm = zeros(4,1);
 qtemp = qout(1)*qout(1) + qout(2)*qout(2) + qout(3)*qout(3) + qout(4)*qout(4);
 if qtemp <= 1.0e-10
     for i=1:3
         q_norm(i) = 0;
     end
     q_norm(1) = 1.0;
 else
     qtemp = sqrt(qtemp);
     for i=1:4
         q_norm(i) = qout(i)/qtemp;
     end

 end
```

S. Satellite Dynamic Block – Quaternion Sub-Block – Change Sign of Quaternion

```
function q_norm_out = flipsign(q_norm)

 if q_norm(1) < 0.0
     q_norm_out = -q_norm;
 else
     q_norm_out= q_norm;
 end
```

Arduino Codes:

T .Motor Code

```
// Connect 1st wire to 4-18 V DC Supply
//2nd wire to 7-18V DC Supply
//3rd wire to ground of both supplies and arduino gnd

const int Unsoll = 5; //Connect 4th Wire(Unsoll) of motor to pin2
const int Dir = 3;    //Connect 5th wire(DIR) of motor to pin3
int x=100;
int z=0;

void setup() {

  pinMode(Unsoll,OUTPUT);
  pinMode(Dir,OUTPUT);
digitalWrite(Dir,HIGH);
}

void loop() {
  if (z==0) {
  analogWrite(Unsoll,x);
  delay(50);
  x=x+2;
  if (x>=255){
    z=1;
  }
  }
  if (z==1) {
  x=100;
  digitalWrite(Dir,LOW);
  analogWrite(Unsoll,255);
  delay(3000);
  z=0;
  }

}
```

Calculations:

Estimation of the Environmental Disturbances

Earthorbiting spacecraft experience incessant physical disturbances caused by multiple sources within the orbital environment, each imparting a torque on the satellite. These torques are produced by gravity gradient effects, magnetic disturbances, solar radiation pressure, and atmospheric drag. The accumulation of these disturbance forces, either cyclical or secular in nature, act to reorient the vehicle and are therefore of direct concern to the ADCS and must be estimated and taken into consideration prior to the system's analysis.

Type of Disturbance	Source	Dependence on distance from earth	Region of space where dominant
Aerodynamic Torque	Drag Force	$e^{-\propto r}$	Altitudes Below $\sim 500\ km$
Magnetic Torque	Earth Magnetic Field	$\frac{1}{r^3}$	$-\sim 500\ km$ to $\sim 35000\ km$ (out to about synchronous altitude)
Gravity Gradient Torque	Earth's Gravity Gradient		
Solar Radiation Torque	The Sun's Solar Radiation (Photons)	Independent	Interplanetary space above synchronous altitude

Quantifying the possible external disturbance torques that might occur during mission lifetime is important when designing an Attitude Control subsystem for a CubeSat mission, to be specific, this will help us size the proper Reaction Wheel needed to carry on the mission successfully by providing the necessary amount of counter torque to bring back the stability of the CubeSat after any possible occurrence of a disturbances.

A detailed calculation process is presented in the next few pages, it is safe to say that many numerical values were estimated based on 2U CubeSat mission "Dhabi Sat".

- **Aerodynamic Torque**

In Low Earth Orbit (LEO), the Earth atmospheric density effect is still dominant on the satellite body causing an undesirable drag force on the CubeSat surface. This drag force creates a disturbance torque generated by difference in distance between the aerodynamic center of pressure and the center of mass of the S/C rigid body.

The discussed aerodynamic force can be represented by the following equation: [1d]

$$F_{aero} = Drag = -\frac{1}{2}\rho_{@alt=420km} C_D A_{sat} |v|^2$$

Where, F_{aero} is the aerodynamic force acting on the satellite surface in (Newtons), ρ is the atmospheric density at the desired orbit and can be estimated from the appendix presented in reference [2d] and is equal to 3.745×10^{-12} $(\frac{\text{kg}}{\text{m}^3})$, C_D is the coefficient of drag and a good estimation for a general 2-U CubeSat is any value between $(2 - 2.5)$ based on many references such as [1d] , and, A is the cross sectional area of the CubeSat, since KU-Sat is a 2-U nano satellite, we will consider the larger face to obtain the largest possible disturbance occurrence. Following that, v is the translational velocity vector of the satellite. Lastly, $\hat{v}_{sat}$ is a unit vector in the direction of the latter.

As mentioned before, this force causes an undesired disturbance to the CubeSat body. This disturbance torque (T_{aero}) can be estimated by using the following equation, [1d]

$$T_{aero} = r_{cp} \times F_{aero}$$

Where r_{cp} is nothing more than the vector (distance) between the Center of Mass (CoM) of the satellite and its Center of Pressure (CoP).

Therefore, we can carry on the calculations to estimate the final value of the aerodynamic disturbance torque. A summary of the calculations is presented below.
Using the following quantities to,

- $c_p = 0.01$ (m) (assumed to be away from the geometrical center, to the right, but very close because CubeSat is generally symmetric)
- $c_m = 0$ (m) (assumed to exactly at the geometrical center)
- $c_D = 2$ (unitless) (believed to be a good estimation) [1d]
- $R = r_{earth} + h_{orbit} = 6370 + 420 = 6790$ (km)
- $\mu = 3.986 \times 10^{14}$ $(\frac{m^2}{s^2})$ = const (earth's gravitational parameter) [3d]
- $v = \sqrt{\frac{\mu}{R}} = \sqrt{\frac{3.986 \times 10^{14}}{6790 \times 10^3}} = 7661$ $\left(\frac{\text{m}}{\text{s}}\right)$
- $A_{sat} = (0.1\text{m} \times 0.2\text{m}) = 0.02$ (m^2) (for the largest surface in the 2-U CubeSat)

The final value can be found such as,

$$F_{aero} = Drag = -\frac{1}{2}\rho_{@alt=420km} C_D A_{sat} |v|^2$$

$$F_{aero} = Drag = -\frac{1}{2}(3.745 \times 10^{-12})(0.2)(0.02)|7661|^2(N)$$

$$F_{aero} = 4.396 \times 10^{-7}\ \hat{v}_{sat}\ (N)$$

$$T_{aero} = r_{cp} \times F_{aero}$$

$$T_{aero} = (0.01 - 0) \times (4.396 \times 10^{-7})$$

$$T_{aero} = 4.396 \times 10^{-9}\ (N \cdot m)$$

- **Magnetic Torque**

This source of disturbance comes from Earth's magnetic field, since our mission is based in the (LEO). The influence of the Earth's phenomena is significant since the orbit of interest is closer to the earth. The magnetic field consists of charged particles moving in space, because of the interaction between the field's charged particle and the magnetic component on board "the surface of a spacecraft can develop a charge of its own giving it a distinct dipole—north/south ends, like a compass. Just as a compass needle rotates to align with Earth's magnetic field, the dipole charged spacecraft will similarly try to rotate as it passes through the magnetic field". [4d]

To approximate the possible magnetic disturbance that might occur during mission lifetime, we used the following formula [4d]

$$T_m = m \times \bar{B}$$

Where T_m is the magnetic torque that disturbs the S/C measured in $(N \cdot m)$, m is the S/C residual spacecraft magnetic dipole measured in $(amps \cdot m^2)$, B is the strength Earth's magnetic moment measured in (tesla) which varies with altitude and latitude. [4d]

- At the poles, $B = \frac{2M}{R^3}$ (tesla = kg/amp-s2)
- At the equator, $B = \frac{M}{R^3}$ (tesla = kg/amp-s2)

Where M is the earth's magnetic moment (constant value $= 7.96 \times 10^{15}$tesla . m3) [4d], and R is the mission's orbit radius provided previously in (km).

Due to current loops and residual magnetisation in the satellite the residual dipole (m) can only be estimated experimentally with the satellite of study. Nevertheless, as the object of study is a 2 Unit CubeSat these due to its small weight, the value of residual dipole was obtained from multiple bibliography where they investigated this parameter for a 2U nanosatellite and it is equal to 0.01 Am^2 [5d] Therfore,

$$T_m = 0.01 \times \frac{2(7.96 \times 10^{15})}{(6790 \times 10^3)^3} = 5.0855 \times 10^{-7}\ (N.m)$$

- **Gravity Gradient Torque**

The gravitational field of planets, in our case Earth's gravitational field, has an inversely proportional relationship with the distance radius from the planet's centre. An undesired torque results from the difference in gravitational force exerted on different parts of a spacecraft due to difference in the distribution of weight. This difference in attraction results in a rotational motion that aligns the satellite minimum inertia matrix with the local vertical. [1d] The disturbance torque caused by earth's gravity gradient can be expressed as: [1d]

$$T_g = \frac{3\mu}{2R^3}\left|I_z - I_y\right| \sin(2\theta)$$

Where T_g is the gravity gradient torque, μ is the earth's gravitational parameter (constant= $3.986 \times 10^5\ \frac{km^2}{s^2}$) [3d], R is the orbit radius mentioned previously in (km), I_z, I_y are spacecraft's mass moments of inertia in (kgm^2), and finally, θ is maximum deviation away from vertical (the S/C body z axis and the local vertical). [1d]

The maximum gravity gradient occurs when CubeSat body is positioned horizontally when in orbit, in other word when sin(2θ) = 1 where θ =90 degrees due to the gravity force pulling unevenly on the CubeSat body. [1d] We can estimate the maximum possible gravity gradient torque by evaluating, [1d]

$$I_x = I_y = \frac{1}{12}m(h^2 + d^2)$$

$$= \frac{1}{12}(2.66)(0.2^2 + 0.1^2) = 0.011083\ (kg.m^2)$$

$$I_z = \frac{1}{12} m(w^2 + d^2)$$

$$= \frac{1}{12}(2.66)(0.1^2 + 0.1^2) = 0.004433\ (kg.m^2)$$

$$T_g = \frac{3\mu}{2R^3}\left|I_z - I_y\right| \sin(2\theta)$$

$$T_g = \frac{3(3.986 \times 10^5)}{2(6790)^3}|0.004433\ - 0.011083|(1) = 1.270 \times 10^{-8}\ (N.m)$$

- **Solar radiation Pressure Torque**

This type of disturbance torque comes from the sun. The solar radiation propagated from the sun exerts an ever-so-slight force called solar-radiation pressure on exposed surfaces; this solar radiation is based on a very small bundle of energy called photons. Photons are massless but do have a momentum, thus, as they strike the exposed surfaces of the Satellite, they transfer this momentum to the surface causing an undesired torque into the CubeSat body. [1d]

The formula below is used to estimate the force exerted on the surface by the solar radiation photons, [1d]

$$F = \left(\frac{F_s}{c} A_s(1 + r)\, cos(i)\right)$$

Where F is the solar radiation force applied on the exposed surface in (Newtons), F_s is the solar flux (constant value$= 1358 \frac{W}{m^2}$) [1d], c is the speed of light (constant value $= 3 \times 10^8 \frac{m}{s}$) [1d], A_s is the S/C cross sectional area, r is the reflection factor ((where r = 1 for a perfect reflector and 0 for a perfect absorber), i=sun incidence angle in (degree)

We assume that [1d] this force acts at the center of pressure. The moment arm is the distance from the center of pressure to the spacecraft's center of mass, that the larger side of the CubeSat is exposed to the sun. A first order approximation reflection factor can be assumed constant and equal to 0.6.

We can simply estimate the possible disturbance torque resulting from the solar pressure by simply evaluating such as,

$$T_{srp} = \left(\frac{F_s}{c} A_s (1 + r)\, cos(i)\right)(c_p - c_g)$$

$$T_{srp} = \left(\frac{1358}{3 \times 10^8}(0.02)(1 + 0.6)\, cos(0)\right)(0.01 - 0) = 1.45 \times 10^{-9}\,(N.m)$$

In general, these torques will act in different directions. Assuming a conservative approach (all torques are constant and acting in the same direction) then sum the magnitudes as follows,

Summary of Environmental Disturbances

Type of Disturbance Torque	**Torque Value (Nm)**
Aerodynamic Torque	4.396×10^{-9} (N · m)
Magnetic Torque	5.0855×10^{-7} (N. m)
Gravity Gradient Torque	1.270×10^{-8} (N. m)
Solar Pressure Torque	1.45×10^{-9} (N. m)
Total Torque	$\mathbf{5.27046 \times 10^{-7}}$ **(N. m)**

1- Attitude Parametrization

Three parameters are required to define a rotational orientation (attitude) of a rigid body in a three-dimensional Euclidean space. There exist various parametrization methods for the mathematical representation of a rigid body's attitude transformation or rotation. [quat_paper]

The attitude – or orientation – of the spacecraft needs to be described (parametrized) in an explicit way. There are several ways of doing this. Some of the parametrization methods are summarized in the following table, [Blue_slides]

Summary of possible attitude parametrization methods. [blue slides]

Parameterization	**Advantages**	**Disadvantages**	**Common Applications**
Direction Cosine Matrix (DCM)	• No Singularities • No trigonometric functions • Convenient product rule for successive rotations	• Six redundant parameters	In analysis, to transform vectors from one reference frame to another

Euler Angles	• No redundant parameters • Physical interpretation is clear in some cases	• Trigonometric functions • Singularity possible • No convenient product rule for successive rotations	Analytical studies and as input/output for human manipulation
Quaternion	• No singularities • No trigonometric functions • Convenient product rule for successive rotations	• One redundant parameter • No obvious physical interpretation	• Onboard the spacecraft, in the control laws.

To define the three-dimensional orientation of KU-Sat in Euclidian space a method or two must be used, in this mission the parameterization method used to achieve the required attitude estimation will be Quaternion.

The reason behind this choice is that Euler angles is easy to develop and to visualize, but computationally intense. Also a singularity problem occurs when describing attitude kinematics in terms of Euler angles and therefore it is not an effective method for spacecraft attitude dynamics and neither is the DCM method. [quat_paper]

Quaternion

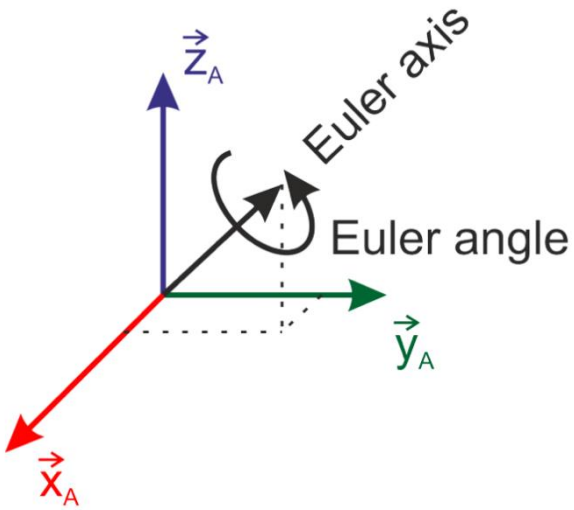
The widely used quaternion representation is based on Euler's rotational theorem which states that the relative orientation of two coordinate systems can be described by only one rotation about a fxed axis. A Quaternion is a 4 × 1 matrix which elements consists of a scalar part **s** and a vector part **v**. Note the scalar part is the first element of the matrix. [quat_paper]

$$q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

According to Euler's rotational theorem a quaternion is defined by a rotational axis and a rotation angle. A quaterionion respresenting a coordinate transformation from system A to system B, $q_{B\to A}$, is defined by Equation, [quat_paper]

$$q = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \|\vec{e}\| \cdot sin\frac{\theta}{2} \end{bmatrix}$$

where $\|\vec{e}\|$ is the normalized rotational axis and θ is not the rotational angle but the transformation angle. [quat_paper]



concept of Euler's rotational theorem of a quaternion [quat_paper]

- **Quaternion Mathematics**

Quaternions are a number system including a wide and complex mathematical theory. In this article, only computational rules required for the purpose of attitude representation are introduced briefly. [quat_paper]

1. Norm

The norm of a quaternion is given by Equation,

$$|q| = \sqrt{q_s^2 + q_x^2 + q_y^2 + q_z^2}$$

A Quaternion with the norm $|q| = 1$ is called unit quaternion. All quaternions for attitude representation are unit quaternions.

2. Normalization

To normalize a quaternion, i.e. transform into a unit quaternion, it is divided by its norm:

$$\|q\| = \frac{q}{|q|}$$

3. Conjugate

The conjugate quaternion has an inverted vector part:

$$q^* = \begin{bmatrix} q_s \\ -q_x \\ -q_y \\ -q_z \end{bmatrix}$$

4. Inverse

To obtain the inverse of a quaternion, its conjugate is normalized.

$$q^{-1} = \frac{q^*}{|q|}$$

For all unit quaternions, the inverse is equal to the conjugate, as they have the norm one.

5. Multiplications

The product q of two quaternions q1 and q2 is defined by:

$$q_1 \times q_1 = q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s_1 \,.\, s_1 - \vec{v}_1 \,.\, \vec{v}_2 \\ s_1 \,.\, \vec{v}_2 + s_1 \,.\, \vec{v}_1 + \vec{v}_1 \times \vec{v}_2 \end{bmatrix}$$

- **Conversion of attitude representations**

o DCM to Quaternion

The single elements of a quaternion can be calculated from the main diagonal of the corresponding direction cosine matrix DCM. In case an element is found which is not zero, all other elements can be calculated from this element and the sub-diagonals. As elements close to zero can cause inexact solutions, one can first calculate all four elements from the main diagonal and then take the greatest value as basis for further calculation. As all equations have a square root on the right side, the maximum value is also the maximum absolute value. [quat_paper]

$$\mathrm{q_s} = \sqrt{\frac{1}{4} \,.\, (\,1 + \mathrm{T_{11}} + \mathrm{T_{22}} + \mathrm{T_{33}}\,)}$$

$$\mathrm{q_x} = \sqrt{\frac{1}{4} \,.\, (\,1 + \mathrm{T_{11}} - \mathrm{T_{22}} - \mathrm{T_{33}}\,)}$$

$$\mathrm{q_y} = \sqrt{\frac{1}{4} \,.\, (\,1 - \mathrm{T_{11}} + \mathrm{T_{22}} - \mathrm{T_{33}}\,)}$$

$$\mathrm{q_z} = \sqrt{\frac{1}{4} \,.\, (\,1 - \mathrm{T_{11}} - \mathrm{T_{22}} + \mathrm{T_{33}}\,)}$$

The maximum value from the above equation is chosen directly and all others are re-calculated as shown in below Tabel. The first column lists all values found using equations, while the columns two to five list the re-caluclated values for all other elements. [quat_paper]

Building quaternion elements from sub-diagonal [quat_paper]

Maximum	$\mathbf{q_s}$	$\mathbf{q_x}$	$\mathbf{q_y}$	$\mathbf{q_z}$
$\mathbf{q_s}$	q_s	$\frac{T_{32} - T_{23}}{4 \cdot q_s}$	$\frac{T_{13} - T_{31}}{4 \cdot q_s}$	$\frac{T_{21} - T_{12}}{4 \cdot q_s}$
$\mathbf{q_x}$	$\frac{T_{32} - T_{23}}{4 \cdot q_s}$	q_x	$\frac{T_{21} - T_{12}}{4 \cdot q_s}$	$\frac{T_{13} - T_{31}}{4 \cdot q_s}$
$\mathbf{q_y}$	$\frac{T_{13} - T_{31}}{4 \cdot q_s}$	$\frac{T_{21} - T_{12}}{4 \cdot q_s}$	q_y	$\frac{T_{32} - T_{23}}{4 \cdot q_s}$
$\mathbf{q_z}$	$\frac{T_{21} - T_{12}}{4 \cdot q_s}$	$\frac{T_{13} - T_{31}}{4 \cdot q_s}$	$\frac{T_{32} - T_{23}}{4 \cdot q_s}$	q_z

- ## Quaternion to DCM

The conversion from a transformation quaternion q to a direction cosine matrix T is given in the following equation. [quat_paper]

$$T = \begin{bmatrix} q_s^2 + q_x^2 - q_y^2 - q_z^2 & 2 \cdot (q_x \cdot q_y - q_z \cdot q_s) & 2 \cdot (q_x \cdot q_z - q_y \cdot q_s) \\ 2 \cdot (q_x \cdot q_y - q_z \cdot q_s) & q_s^2 - q_x^2 + q_y^2 - q_z^2 & 2 \cdot (q_y \cdot q_z - q_x \cdot q_s) \\ 2 \cdot (q_x \cdot q_z - q_y \cdot q_s) & 2 \cdot (q_y \cdot q_z - q_x \cdot q_s) & q_s^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

- ## Euler to Quaternion

To obtain a quaternion from Euler angles, one can take an intermediate step and convert into a DCM first. The elements of the transformation matrix are then used to compute the quaternion elements as shown before. [quat_paper]

$$T = \begin{bmatrix} -sin\theta_x sin\theta_y sin\theta_z + cos\theta_x cos\theta_z & sin\theta_x sin\theta_y cos\theta_z + cos\theta_x sin\theta_z & -sin\theta_x cos\theta_y \\ -cos\theta_y sin\theta_z & cos\theta_y cos\theta_z & sin\theta_y \\ cos\theta_x sin\theta_y sin\theta_z\_sin\theta_x cos\theta_z & -cos\theta_x sin\theta_y cos\theta_z + sin\theta_x sin\theta_z & cos\theta_x cos\theta_y \end{bmatrix}$$

- ## Quaternion to Euler

Converting a quaternion to Euler angles, we use the inverse procedure of (Euler to Quaternion). So, we convert into a DCM first (shown in EqX) and then calculate the Euler angles using the following equations. [quat_paper]

$$\theta_z = tan^{-1}\left(-\frac{T_{21}}{T_{22}}\right)$$

$$\theta_x = tan^{-1}\left(-\frac{T_{13}}{T_{33}}\right)$$

$$\theta_z = tan^{-1}(-T_{23})$$

These transformations can be used to represent the attitude information in other formatting such as Euler and DCM, but, since we are basing the entire analysis on quaternion method. No other method will be shown further in this work.

2- Brushless DC Motor Theoretical Transfer Function

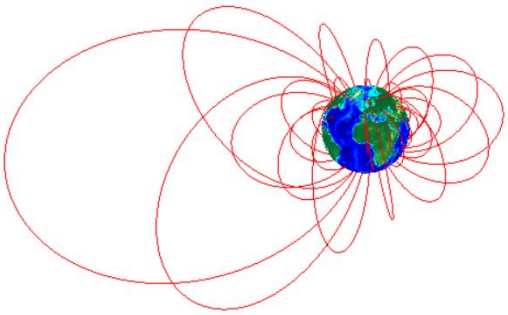
$$\frac{\omega(s)}{V(s)} = \frac{K_m}{(Ls+R)(Js+K_f)+K_mK_b}$$

$$\frac{\omega(s)}{V(s)} = \frac{0.0181}{(0.00194\,s+28.2)(0.81\,s+0.00001157676)+(0.0181)(0.01809592)}$$

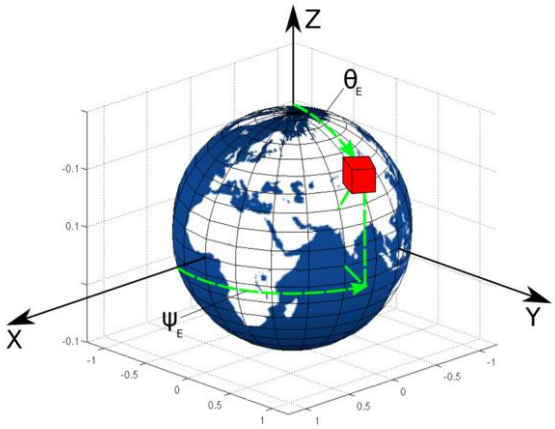
$$\frac{\omega(s)}{V(s)} = \frac{0.0181}{0.0015714s^2+22.842s+0.000654}$$

$$\frac{\omega(s)}{V(s)} = \frac{11459322.57}{s^2+14476.917\,s+430902.82}$$

International Geomagnetic Reference Field (IGRF) Model



The International Geomagnetic Reference Field (IGRF) is an internationally agreed upon mathematical model of the Earth's magnetic field. For out project, a MATLAB code from math works was used. The result of utilizing this model is the ability to provide any position coordinate of the satellite to the module and have the model return the magnetic field strength in East, North, and Vertical Coordinates. Simply, the inputs are time, latitude, longitude, altitude and coord, assuming an inertial frame with the z-axis pointing through the North Pole and the x axis pointing through the equator at the prime meridian as seen in Figure. This is known as the Earth-Centered Inertial (ECI) coordinate system.



Inputs:

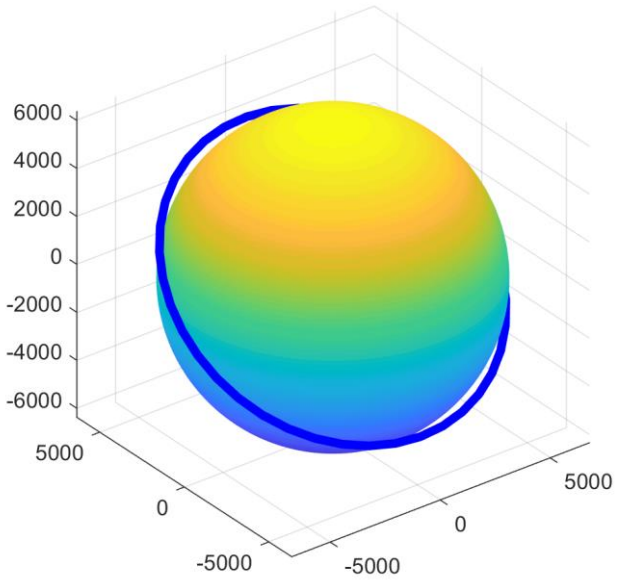
1- TIME: Time to get the magnetic field values either in MATLAB serial date number format or a string that can be converted into MATLAB serial date number format using DATENUM with no format specified.
2- LATITUDE: Geocentric or geodetic latitude in degrees.
3- LONGITUDE: Geocentric or geodetic longitude in degrees.
4- ALTITUDE: For geodetic coordinates, the height in km above the Earth's surface. For geocentric coordinates, the radius in km from the center of the Earth.
5- COORD: String specifying the coordinate system to use. Either 'geocentric' or 'geodetic' (optional, default is geodetic).

Outputs:

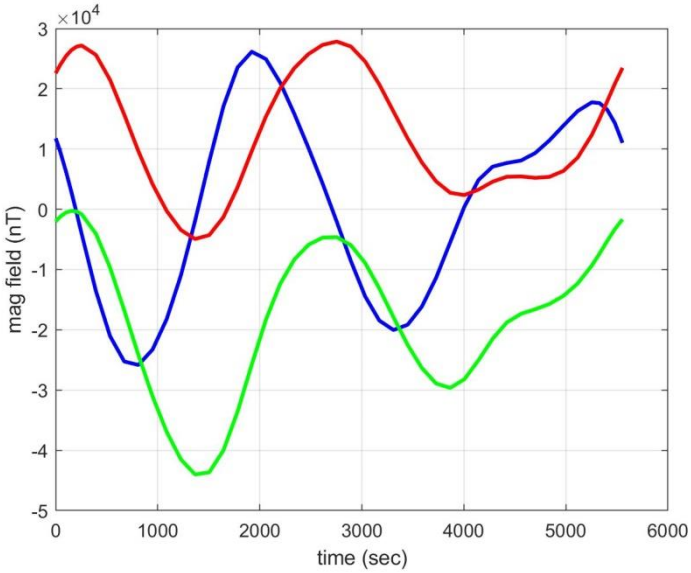
1- BX: Northward component of the magnetic field in nanoteslas (nT).
2- BY: Eastward component of the magnetic field in nT.
3- BZ: Downward component of the magnetic field in nT.

B: [BX(:), BY(:), BZ(:)].

The below Figure presented next page shows an example 51.6 degree inclination orbit, 420 km above the Earth's surface. The orbit begins with the satellite above the equator and the prime meridian and assumes the Earth does not rotate.



The below Figure shows the magnetic field during the orbit in the inertial frame. PCI stands for Planet Cantered Inertial which in this case is the same as the ECI frame since the planet is Earth.



Also, there is one more thing to do, as per the requirement of the KU-2U CubeSat mission, which is converting the data from ECI frame to orbital frame.

Datasheet:

1. Brushless DC Motor Data Sheet

		2610 T		006 B	012 B	SC
1	Nominal voltage	U_N		6	12	Volt
2	Terminal resistance, phase-phase	R		7,0	28,2	Ω
3	Output power [1]	$P_{2\ max.}$		1,92	1,91	W
4	Efficiency	$\eta_{max.}$		78	78	%
5	No-load speed	n_o		6 200	6 200	rpm
6	No-load current	I_o		0,012	0,006	A
7	Stall torque	M_H		7,73	7,68	mNm
8	Friction torque, static	C_o		0,025	0,025	mNm
9	Friction torque, dynamic	C_v		$1,35 \cdot 10^{-5}$	$1,35 \cdot 10^{-5}$	mNm/rpm
10	Speed constant	k_n		1 055	528	rpm/V
11	Back-EMF constant	k_E		0,948	1,895	mV/rpm
12	Torque constant	k_M		9,05	18,1	mNm/A
13	Current constant	k_I		0,111	0,055	A/mNm
14	Slope of n-M curve	$\Delta n/\Delta M$		816	822	rpm/mNm
15	Terminal inductance, phase-phase	L		480	1 940	µH
16	Mechanical time constant	τ_m		69	70	ms
17	Rotor inertia	J		8,1	8,1	gcm^2
18	Angular acceleration	$\alpha_{max.}$		9,5	9,5	$\cdot 10^3 rad/s^2$
19	Thermal resistance	$R_{th\,1}$ / $R_{th\,2}$	33 / 27			K/W
20	Thermal time constant	τ_{w1} / τ_{w2}	20 / 230			s
21	Operating temperature range		-25 ... + 80			°C
22	Shaft bearings		ball bearing, preloaded			
23	Shaft load max.:					
	– radial at 3 000/7 000 rpm (3 mm from mounting flange)		4,0 / 3,5			N
	– axial at 3 000/7 000 rpm (push-on only)		3,5 / 3,4			N
	– axial at standstill (push-on only)		17,5			N
24	Shaft play:					
	– radial	≤	0,015			mm
	– axial	=	0			mm
25	Housing material		plastic			
26	Weight		20,1			g
27	Direction of rotation		electronically reversible			
28	Number of pole pairs		2			
Recommended values - mathematically independent of each other						
29	Speed up to	$n_{e\ max.}$		7 000	7 000	rpm
30	Torque up to [1] [2]	$M_{e\ max.}$		3,14 / 3,72	3,13 / 3,70	mNm
31	Current up to [1] [2]	$I_{e\ max.}$		0,40 / 0,47	0,20 / 0,24	A

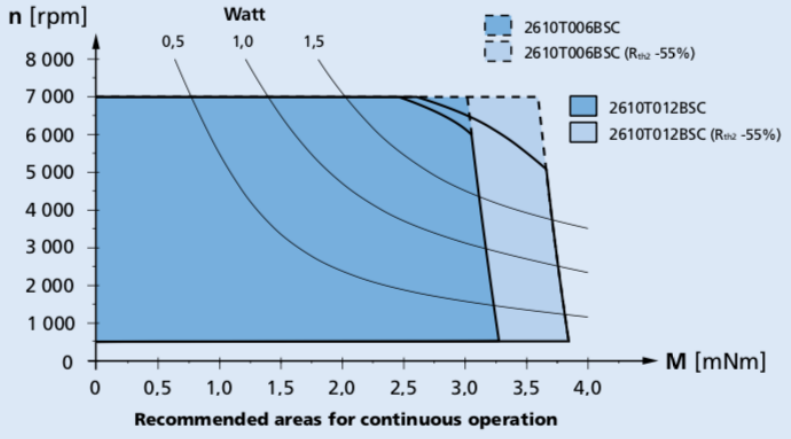
[1] at 5 000 rpm
[2] thermal resistance $R_{th\,2}$ not reduced / thermal resistance $R_{th\,2}$ by 55% reduced

Note:

The diagram indicates the recommended speed in relation to the available torque at the output shaft for a given ambient temperature of 22°C.

The diagram shows the motor in a completely insulated as well as thermally coupled condition ($R_{th\,2}$ 55% reduced).

The area of the curve is defined by the maximum allowable supply voltage of the integrated speed controller as well as the control performance characteristics.



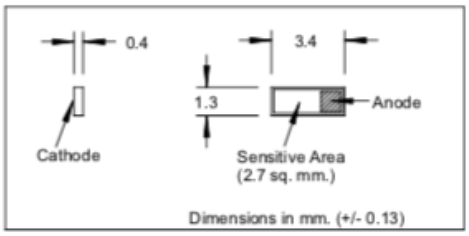
2. Photodiode Data Sheet

SLCD-61N8

Solderable Planar Photodiode

Features

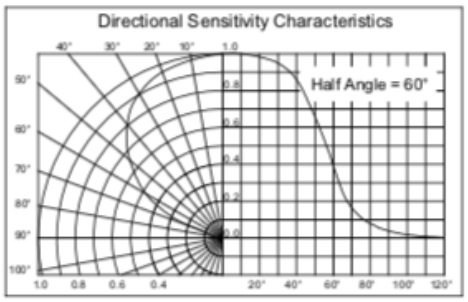
- Visible to IR spectral irradiance range
- High reliability
- Oxide passivation
- Linear short circuit current
- Low capacitance, high speed
- Available in arrays where # indicates number of elements (maximum of 8 elements)



Also available with leads as part number SLSD-71N8

Description

The Silonex series of silicon solderable planar photodiodes feature low cost, high reliability, and linear short circuit current over a wide range of illumination. These devices are widely used for light sensing and power generation because of their stability and high efficiency. They are particularly suited to power conversion applications due to their low internal impedance and relatively high shunt impedances, and stability. These devices also provide a reliable, inexpensive detector for applications such as light beam sensing and instrumentation. The electrical characteristics below are per element. In the multielement arrays the cathodes are common to a all elements.



Absolute Maximum Ratings

Storage Temperature	-40°C to +125°C
Operating Temperature	-40°C to +125°C

Electrical Characteristics (T_A=25°C unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
I_{SC}	Short Circuit Current	100	170		µA	V_R=0V, Ee=25mW/cm^2 (1)
V_{OC}	Open Circuit Voltage		0.40		V	Ee=25mw/cm^2 (1)
I_D	Reverse Dark Current			1.7	µA	V_R=5V, Ee=0
C_J	Junction Capacitance		100		pF	V_R=0V, Ee=0, f=1MHz
S_λ	Spectral Sensitivity		0.55		A/W	λ=940nm
V_{BR}	Reverse Breakdown Voltage	20			V	I_R=100µA
λ_P	Maximum Sensitivity Wavelength		930		nm	
λ_R	Sensitivity Spectral Range	400		1100	nm	
$\theta_{1/2}$	Acceptance Half Angle		60		deg	

Specifications subject to change without notice

104118 REV 0

Notes: (1) Ee = light source @ 2854 °K

Gantt Chart

KUSat (2U)			
The ADCS concept			
Name	**Person**	**Status**	**Date**
Research and investigate the previous phases	All	Done	2020-09-10
Building and designing the ADCS	All	Done	2020-09-10
identifying he requirements and understanding the mission	All	Done	2020-09-25
Defining the orbit parameters and assigning assumptions	All	Done	2020-10-09
Investigating the disturbances and finalizing torque calculations	All	Done	2020-10-10
ADCS alternative Design proopsal			
Name	**Person**	**Status**	**Date**
selecting a micro controller	All	Done	2020-10-15
selecting control mode, sensor and actuators	All	Done	2020-10-17
purchasing and finalizing hardware components selection	All	Done	2020-12-21
Numerical Modeling			
Name	**Person**	**Status**	**Date**
Define inertial and axes frame of work	All	Done	2021-01-01
Numerical Modeling of KUSat in orbit	All	Done	2021-01-06
Estimation of the Environmental Disturbances	Basahyer	Done	2021-01-05
Numerical Modeling of KUSat Dynamics	Asma,Reem	Done	2021-01-08
Numerical Modeling of KUSat kinematics	Asma,Reem	Done	2021-02-09
Components selection and Testing			
Name	**Person**	**Status**	**Date**
Understanding the working principle for each hardware components	All	Done	2020-12-24
Hardware components search, selection, and order.	All	Done	2020-12-26
implementation of Gyroscope sensor connection and testing	Khulood	Done	2021-02-01
implementation of brushless DC motor connection and testing	Asma,Reem	Done	2021-01-20
Finalizing sun sensor model proof (simulations)	Maha	Done	2021-04-24
fly wheel design	Asma,Reem	Done	2021-04-30
Defining the algorithms			
Name	**Person**	**Status**	**Date**
Selection of a Control and Determination Laws	All	Done	2021-01-11
start testing PD controller	Asma,Reem	Done	2021-03-18
Understanding the working principle of Triad algorithms	Maha ,Khulood	Done	2021-04-15
Simulations			
Name	**Person**	**Status**	**Date**
Triad algorithms simulation implementation	Maha ,Khulood	Done	2021-05-04
PD controller simulations	Asma,Reem	Done	2021-04-23
Gyro simulation model	Khulood	Done	2021-04-08
magnetometer Simulation	Basahyer	Done	2021-04-29
Magnetorquer simulation	Basahyer	Done	2021-05-01
IGRF coding	Maha ,Khulood, Bashyer	Done	2021-04-15
Sun model Coding	Maha	Done	2021-04-17
Sun sensor Testing	Maha	Done	2021-04-08
Full Attitude determination Simulation	Maha ,Khulood, Bashyer	Done	2021-05-03
Full control Simulation	Asma,Reem	Done	2021-05-03

جامعة خليفة
Khalifa University

Project Clearance form I

Project Title: KU CubeSat (2U) – Stage IV: Attitude Determination and Control System (ADCS) using Reaction Wheel for 3-Axis Control

Note: Please fill in part A if your project results as software or simulation or both. Fill in part B if project results are hardware (including models and working prototype). If your project results into both hardware and software then please fill in both part A and part B.

Part A: Software/Simulation Projects

Applicable ☒ Not Applicable ☐

We (students) hereby declare that we have submitted all software/simulation and relevant user manuals resulted from our final year project to our supervisor.

Part B: Projects with hardware (including models and working prototype)

Applicable ☐ Not Applicable ☒

We (students) declare that we have returned all hardware and relevant user manuals used in our final year project to:

Supervisor ☐ Lab engineer ☐ SDP Coordinator ☐ Facilities manager ☐

Student's name and signature: Maha O. ALmazrouei Date: 6-5-2020

Student's name and signature: Asmaa N. ALyammahi Date: 6-5-2020

Student's name and signature: Reem A. ALsulaimani Date: 6-5-2020

Student's name and signature: Khulood A. ALmarzooqi Date: 6-5-2020

Student's name and signature: Bashayer M. ALmehrzi Date: 6-5-2020

SDP Coordinator's signature upon receipt of this form: _____
Date: 6-5-2020

Project Clearance form II

Project Title: KU CubeSat (2U) – Stage IV: Attitude Determination and Control System (ADCS) using Reaction Wheel for 3-Axis Control

Main supervisor name: Dr. Elena Fantino

Department: Aerospace Engineering Department

SDP Coordinator in the Department: Sanjeev Rao

Part A: Software/Simulation Projects

Applicable ☐ Not Applicable ☐

I (supervisor) confirm that students have submitted all software/simulation and relevant user manuals resulted from their project to me and I recommend that the above will be:

Kept for demo and/or future exhibitions ☐

Kept for other purposes ☐ please indicate the purpose: _____

Part B: Projects with hardware (including models and working prototype)

Applicable ☐ Not Applicable ☐

I (supervisor) confirm that students have returned all hardware and relevant user manuals to the person mentioned below (Part C) and I recommend that the hardware should be:

Dismantled ☐ Kept for demo and/or future exhibitions ☐

Kept for other purposes ☐ please indicate the purpose: _____

Supervisor: _____ Signature: _____ Date: _____

Part C: I acknowledged the receipt of all hardware used in this project.

Supervisor ☐ Lab engineer ☐ SDP Coordinator ☐ Facilities Manager ☐

Name: _____ Signature: _____ Date: _____