Czech Technical University in Prague
Lulea University of Technology

Simulation of Attitude and Orbit Control for APEX CubeSat

Master thesis

BSc Niels de Graaf

Msc programme: Joint European Master in Space Science and Technology
Master in Cybernetics and Robotics
Supervisor: Doc. Ing. Daniel Stefl, Ph.D.
Examiner: Martin Hlinovsky, Ph.D.
Kristian Hengster Movric, Ph.D.
Anita Enmark, Ph.D.

Prague, July 2020

Thesis Supervisor:
Doc. Ing. Daniel Stefl, Ph.D.
Managing Director
Huld s.r.o
náměstí Winstona Churchilla
1800/2, Praha - Žižkov
Czech Republic

Declaration

I hereby declare I have written this Master thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, July 2020

.....
BSc Niels de Graaf

Abstract

CubeSats are becoming a game changer in the space industry. Appearing first for university mission, its popularity is increasing for commercial use and for deep space missions such as the on HERA mission that will orbit in 2026 around an asteroid as part of a planetary defence mission. Standardisation and industrial collaboration is key to a fast development, assuring the product quality and lower development expenditures.

In this study the focus is set elaborating a low cost demonstrator platform to be used for developing and testing onboard software on physical hardware: a Hardware-Software testing facility. The purpose of such a platform is to create an interactive and accessible environment for developing on board software. The application chosen to be elaborated on this platform is a module the subsystem of attitude and orbit control of the satellite orbiting around asteroid.

In order to create this platform the simulation of the asteroid environment of the CubeSat has been made using open source software libraries. During this task the performance of open source libraries has been compared to commercial alternatives. In the development of simulation different orbit perturbations have been studied by modelling the asteroid as a cube or spheroid and additionally the effect of a third perturbing body and radiation pressure.

As part of this project two microcontroller have been set up communicating using a communication bus and communication protocols used for space applications to simulate how the attitude and orbit control is commanded inside the CubeSat.

Keywords: Orbital Control Simulation, Asteroid, Open Source, CAN bus, Microcontrollers, Software Verification Facility.

Acknowledgements

I would like to thank Daniel Stefl for giving me the freedom and opportunity to set up my work in a very knowledgeable environment such as the company Huld in Prague. A kind thought to Marek Sedlacek who supported me during this project with great interaction, interest and advice. A warm thought to Juan Luis Cano for providing help during the development of the simulations.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

The focus of this master thesis is to create a module of a Hardware-Software testing Facility and Spacecraft Simulator. The subsystem of the spacecraft that is targeted for the simulation is the Attitude and Orbit Control(AOCS). This is functionality of the spacecraft to control the course of its navigation as well as determining its position and orientation [1]. The module that will covered by the Hardware-Software testing facility for this study is the communication bus, this includes communication protocols and physical links between the components of the spacecraft.

In this project the Software Testing Facility will be made for a CubeSat mission around an asteroid. The term CubeSat designates satellites strictly given a form of a cube of the edge 10 cm, which is known as one CubeSat unit, 10 cm being denoted as 1U and these satellites are usually made with the dimensions of 1U, 1.5U, 3U up to 12U [2]. The study covers the simulation of an asteroid environment, controlling the motion of the CubeSat and communication.

Making software ready to fly in space requires a lot of testing. This is the most critical part of the entire life cycle of the software development. There are many standards and processes that have to be applied in order to limit the risk of failure.

One of the applications of software testing is the Hardware-Software Environment testing in which the software is connected to hardware and has to respond to a simulated scenario and test cases. This is described in the guidelines from the ECSS-Q-80, published by the European Cooperation for Space Standardization. This cooperation releases the standards that the companies contracted by ESA (European Space Agency) need to comply with. Nevertheless there is room for tailor made solutions with respect to the project. Verifying and validating the project requirements include Hardware-Software Interaction tests in which companies have to ensure that the software behaves accordingly [3].

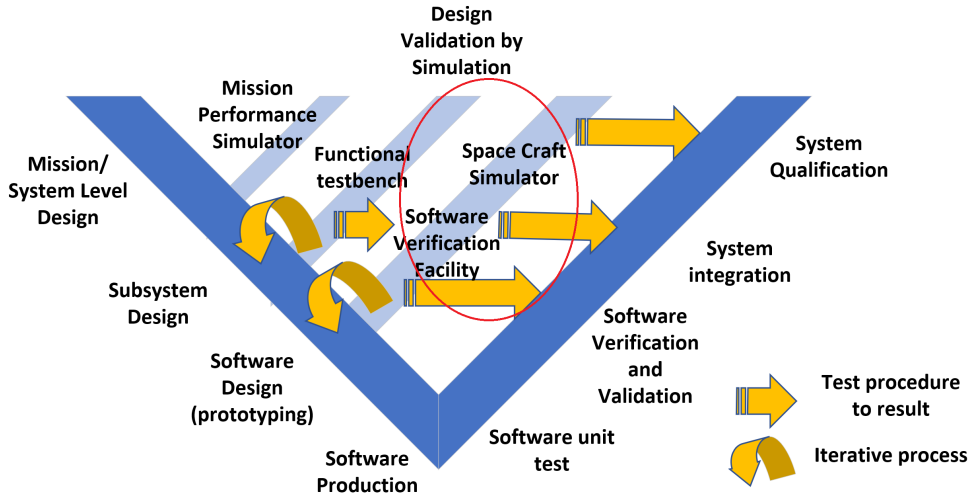

Figure 1.1: Simulation in the project life cycle [4]

Other important modules of testing are Simulations. The ECSS-E-TM-10-21A describes guidelines on system modelling and simulation for Space engineering. Simulation can be done for great amount of aspects of the project such as Spacecraft Dynamics, in which the motion of the spacecraft according to its environment is made. Simulating is an omnipresent process integrated in the project life cycle described as in the V-shape model as shown in Figure 1.1. The process of interest of this study is the Software Verification Facility and Space craft Simulator, encircled in red.

1.1 Project Background information

The master thesis is conducted at the Czech-Finnish company Huld, previously Space Systems Czech(2019). The company develops software for space applications, onboard computer (OBC) as well as ground segment services. Huld has the ambition to continue with missions around asteroids in the future and wants to develop a Software Verification Facilities using open source software. Therefore the focus of this study is on the elaboration of the Hardware-Software testing Facility and Simulation module. This will be used for easing software prototyping as well as verification and validation in order for the system to reach its maturity and meet its design criteria. The purpose of this master thesis is then primarily to create a demonstrator of such a facility that is interactive for the user to enhance creativity and ideas of development for space with off the shelf components.

Huld is currently working inside a European consortium contracted by ESA on the deep space project HERA. The mission is a follow up of the NASA(National Aeronautics and Space Administration) DART (Double Asteroid Redirection Test) mission. The purpose

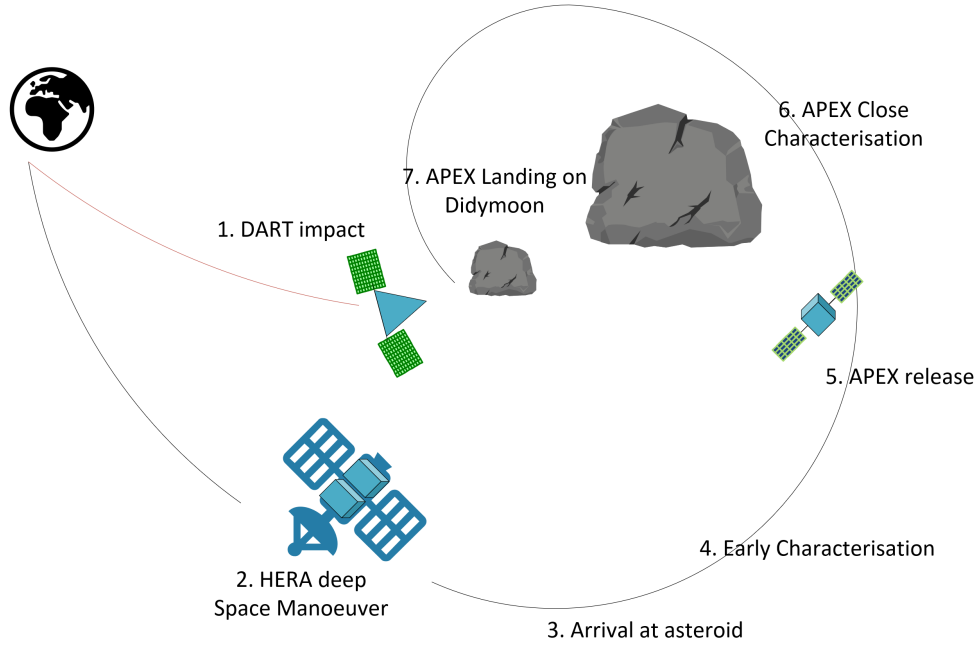

Figure 1.2: HERA and APEX mission overview [5]

of both missions is to evaluate the effectiveness of the kinetic impactor, which is the concept of deflecting bodies trajectories with momentum transferred by colliding a space craft with the asteroid. The application of such a concept is for planetary defence.[6]

The DART mission launch date is scheduled for summer 2021 and the collision for September 2022. In here the target is the binary system of the Didymos asteroid and its moon Didymoon. The NASA DART mission launches the spacecraft responsible of impacting the asteroid Didymoon and the SpaceCrafts of the HERA mission are responsible for characterizing the event as well as the bodies. The launch is set to 2024 for the binary system to be reached in 2026.[5]

The APEX CubeSat is one of the payloads of HERA. Huld is also responsible for the onboard software of this spacecraft part of the Hera mission but only takes action at the encounter with Dydimos system where the CubeSat will be deployed and released from the main spacecraft. The APEX mission starts at the CubeSat release phase which comes after the arrival at the asteroid and the early characterisation phase of the body. This spacecraft will be basis for the development of the Software testing Facility.

1.2 Current situation

In the current and previous missions that the company worked on, the software is developed by different companies from the same consortium. Therefore the company is then provided with simulators to test and implement higher level functionalities while the low level software is developed by the other company simultaneously. Huld want to broaden their capabilities by gaining the possibility to test software at lower levels. A demonstrator for Hardware-Software testing would be beneficial for creating a culture of knowledge inside the company and familiarise colleagues with hardware.

In addition to simulators used by companies as a "black box" approach, Simulators and Software used for space application development are often commercial platforms. More and more open source software for these application are being developed and could be a great potential for companies like Huld to cut down the price of their development cost as well as getting a bigger autonomy.

1.3 Thesis Goals

The goal of this thesis is to create a demonstrator of such a Software Testing Facility which will include spacecraft dynamics simulator and a Hardware Software validation interface. The main objectives are listed below:

1. To create a model of the dynamics of the CubeSat subjected to the disturbance for the asteroid environment to plot its trajectories using python libraries developed for orbital dynamics.
2. Attitude and Orbital control applications will be programmed on a microcontroller to calculate the new trajectories for the simulation.
3. An Interface will be made to send simulated data using the standard protocols developed for spacecrafts and using the communication bus used on the APEX spacecraft.

An overview of the project is in the given in the Figure 1.3 where it can be seen the how the simulation of the APEX spacecraft is connected with the Hardware-Software testing facility and interfacing with with possible front end application.

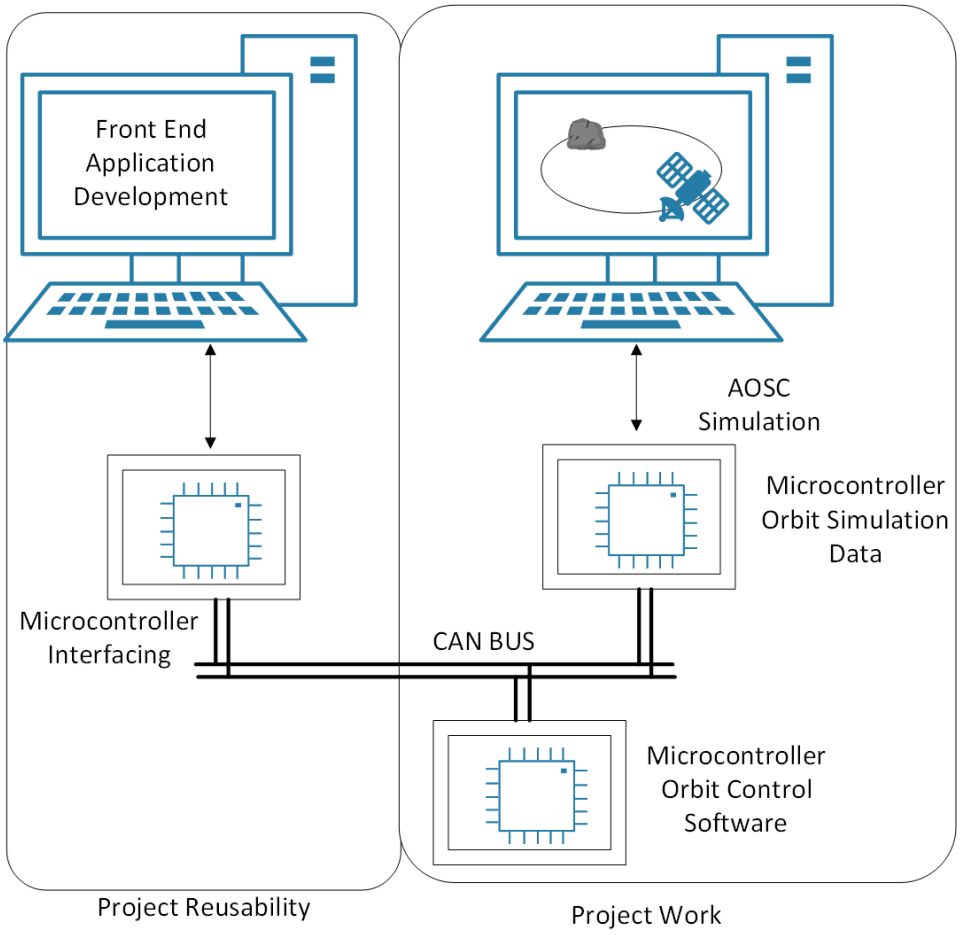

Figure 1.3: Project overview

1.4 Methodology

The way that this assignment is approached is by working from Low level to top level. The communication bus used on the APEX CubeSat is the CAN bus (Controller Area Network) which is a message based protocol used for microcontrollers and devices to transfer information and commands[7]. This will be built starting from the physical bus to the higher level protocols. CAN bus communication is widely used in the space and automotive industry and was developed in order to reduce signal interference and a reliable communication protocol for multiple end nodes.

The first step is to create the physical connection with hardware to allow this CAN bus communication between nodes. The APEX CubeSat makes use of a protocol developed for space that is built upon the CAN bus communication. The next step is to implement a higher level protocol on top of the CAN bus. Using this communication is convenient for reusability because it is a design choice of many satellite which is experiencing a considerable growth on the market.

When it comes to the simulation of the APEX dynamics here is the method that is put in use: the thesis will investigate the use of Python libraries developed for orbital dynamics. Python programming language usage has increased and become more and more popular in the last decade due to its convenience and interactivity. One of the first steps to frame the potential of these open source Python libraries is to create an equivalent application on this platform as well as on Matlab and STK to compare performances. Afterwards the next target is to build the environment of the asteroid and investigate the effect of disturbances on the orbit: 3rd body disturbances, the shape of the object and other possible disturbances.

This thesis will also be the opportunity to investigate different control routines for the embedded software and their performance since this is this study is performed at the department of Cybernetics and it is a major part of the scope of this discipline. The choice is then to develop this orbit control software using the tailored version of Python for microcontrollers: MicroPython which then brings a certain consistency in this project in using the same programming language.

1.5 Thesis synopsis

The 2nd chapter deals with the methods used for modelling the asteroid environment. This will include the different perturbations of the binary asteroid system that the APEX CubeSat will have to navigate in. The choices of Python libraries used for developing this simulation will be elaborated as well in this chapter. The 3rd chapter defines the characteristics of the APEX CubeSat regarding the simulation and the Hardware-Software development. The 4th chapter brings about the development of CAN bus from bottom to top level. The 5th chapter deals with AOCS of the simulated space craft in its environment and the routines on the embedded software from the test facility.

Chapter 2

Asteroid environment simulation

This chapter will deal with the modelling for the simulation of the environment of the satellite. For this particular purpose, the environment that is being modelled is the Didymos asteroid. The Didymos system with the main body having a 780 meter diameter and a 160 meter diameter moon, Didymoon. In a first part the tools used for orbital simulations will be shown and comparing its performances to a commercial solution. Afterwards the different aspects of the environment used for simulating the perturbations that the satellite can be subjected to during the course of its mission will be explained. The software developed for the simulation is available here:

`https://github.com/NielshuldC/Simulation-AOCS-APEX-CubeSat/tree/master/Asteroid`

2.1 Orbit Propagator

An orbit propagator is a term given to algorithms, software and computer applications used for calculating the coordinates of the position of a body within a certain reference frame that is orbiting around another one. This method enables to know the position and velocity variations over time of a satellite mission. The orbit propagator outputs are numerical results from solving the equations of motions that are representing the movement of bodies subjected to different forces acting on them. The major force to which the bodies are subjected and computed by the orbit propagators is gravity. [8]

These computations allow to make approximations of these motions since exact solutions can only performed when dealing with point-mass bodies. The latter are idealisations of solid bodies and rare are applications in the space industry where the bodies involved

are ideal, therefore estimations and reliable computations become important. These applications done by simplification called the General Perturbations methods or numerical integration or sometimes a combination of both of those methods.[9]

There is a wide range of orbit propagators coded in different programming languages developed by commercial and open source communities. For example different orbit propagation algorithms are provided by the company MathWorks in their product Matlab which is a software environment that has a long history of development since 1970 and is used in the aerospace industry[10]. Another company providing more graphical integrated environment is Systems Tool Kit usually refered as STK. The development of this commercial software tool started in 1989 and now used by a great number of aerospace organisations such as ESA, NASA, Airbus and Boeing [11].One of the biggest competitors nowadays of STK developed by AGI is FreeFlyer which is commercial platform that is been in use since 1997[12].

With the commercialising and democratising of space where there is an emergence of commercial and student missions thanks to the development CubeSats there is an need for educational purposes and development of open source orbit propagators. There are for instance libraries for the Java programming language called Orekit and JAT. The Java AstroDynamics Toolkit that are nowadays also used by Swedish Space Corporation and Thales [13]. Furthermore the General Mission Analysis Tool developed in NASA has also been open sourced and is considered as an alternative to STK [12].

For the demonstrator of this thesis project the coding language that is used is Python and Micropython and uses the libraries called Poliastro and Astropy which have been developed for orbit propagation and plotting. Poliastro started its development in 2013 by Juan Luis Cano and is an open source assortment of Python functions licensed by MIT. This collection of software is used to solve problems in Orbital Mechanics and Astrodynamics. Some of the applications that are already implemented in this set of algorithms for converting vectors to classical orbital elements, Trajectory plotting and most importantly for the need of the development of this demonstrator: Analytical and numerical Orbit propagators [14].

Poliastro makes use of another library already widely used in universities called Astropy which is also a group of software libraries written in the Python programming language. These software packages were primarily more focused on astronomy. The advantage of Astropy is that it can be easily combined with other Python packages [15].

The reason why Astropy and Poliastro will be used for the demonstrator is mainly for

the the Programming language they have been written in. The usage Python has been growing drastically in the industry for the last decade and the open source libraries and collaborative platform allow a fast development. This synergy is often described as being inscribed in what is called Industry 4.0 which an name given fourth industrial revolution in which interoperability and smart technologies are combined[16]. Moreover the development of Numba which is a high performance Python compiler. Numba allows to make the processing time shorter making the Python language more attractive to the industry and boost its popularity and has been used for this project [17]. Furthermore Poliastro has been chosen for its constant updates and active communities and intuitive structure. Poliastro is therefore the candidate for implementing the simulation in this low cost Verification Facilities demonstrator.

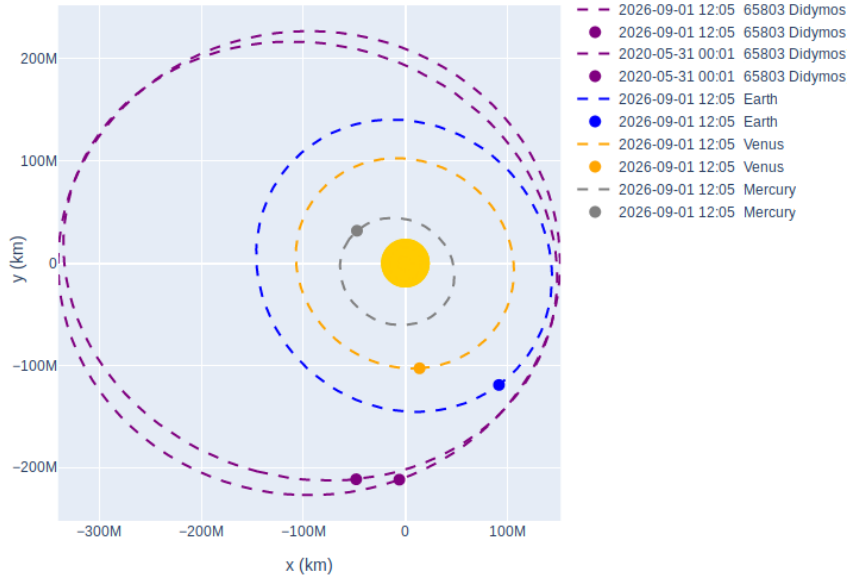

Figure 2.1: Ephemeris plotting arrival of HERA mission to Didymos

In the Figure 2.1 the capabilities of Poliastro are shown. This is the plot of the ephemeris of the Sun, Mercury, Venus, Earth and the body of interest when it comes to the HERA mission that the company Huld is working on: the binary system of Didymos that has the official name of "65803 Didymos". The ephemeris is a term that covers the database of celestial objects containing their calculated position at a certain interval over a period of time[18]. In purple there is the plotted position of Didymos from the ephemeris database from NASA at the time of when the thesis is conducted. Next the orbit and the position that Didymos will have in September 2026 when the HERA mission will reach this system has been calculated by using a numerical orbit propagator provided by Poliastro.

The positions of Didymos are then in the Heliocentric Coordinates in 2026-09-01 at the beginning of the mission and on the 2016-12-01 at the end of the mission are respectively

in kilometers:

$$[x_i, y_i, z_i] = [67969641.02827488, -2.236529715e+08, -7722639.23892914] \quad (2.1)$$

and:

$$[x_f, y_f, z_f] = [1.60778844e+08, -27515205.2200428, -9639133.51806481] \quad (2.2)$$

When Calculating the coordinates with Matlab and STK for the beginning of the mission the following result is [19]:

$$[x_i, y_i, z_i] = [67969641.02827486, -2.236529714e+08, -7722639.23892911] \quad (2.3)$$

Both application provide similar results with a difference of $10^2 m$. Developers of Poliastro also tested simulations and compared with numbers given by ESA and had close results [14]. Poliastro can legitimately be used for the simulation.

The solution is in both cases easily implementable with a comparable accuracy. Nevertheless Python has the ability to call other libraries that are developed for plotting the results which bring more freedom in the format than fully integrated software environment. The plots here are implemented using the Matplotlib libraries as well as the Plotly libraries which allow to display on the web browser.

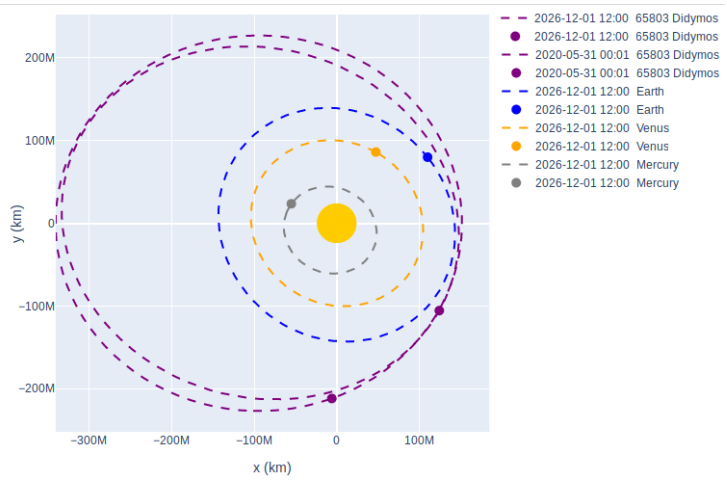

Figure 2.2: Ephemeris plotting propagation at end of the HERA mission

As discussed before Orbit propagators are a collection of numerical, analytical and hybrid algorithms. The one that will chosen for calculating the position of the APEX CubeSat in the Dydimos environment will be the Cowell method that Poliastro has implemented along with several others. The Cowell method has great advantages when comes for getting faster results and is an easier application in programming since it is a numerical

integrator [20]. The disadvantage of this method is when the perturbation forces become large in magnitude the error also drastically increases. Moreover it is necessary to carry many significant digits in the arithmetic because in the case of a large difference in the forces of the central body and the perturbing bodies. In the Didymos system there is not a great difference between bodies and forces such as Sun compared to Earth scales reducing the errors on that side. The chosen method has limitations that will have to be taken into account but will not impact to show the performance of the application of the demonstrator.

2.2 Initialising the Orbit

The orbit propagator has to calculate the coordinates of the satellite over time when subjected to perturbations since orbiting the Didymos system is more complex than only orbiting a perfect spherical body. The initial Orbit will be defined in this simplified system where Didymos will be spherical body with a diameter of 780 m as shown on the properties Table 2.1 available on the website of NASA.

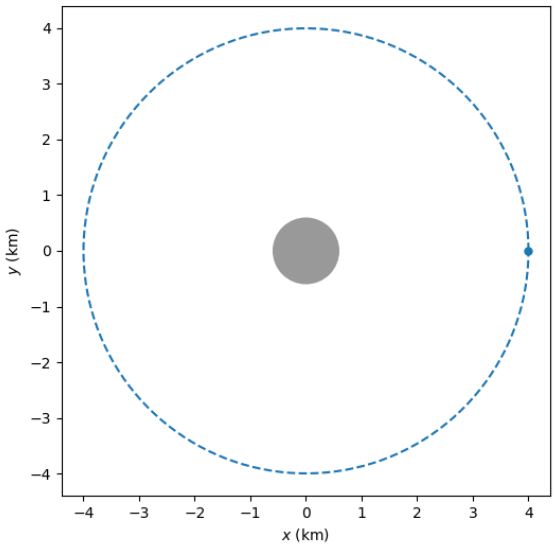

Figure 2.3: Circular orbit of APEX around Didymos without perturbation

The orbit will be inspired by the one that will done by APEX at the beginning of its mission. It is the defined as a circular orbit with at an altitude of 4km with an inclination of 30 degrees. The initial vectors of the Satellites are: $r_0 = [4, 0, 0]$ km and the velocity vectors are then $v_0 = [0, 7.270 \cdot 10^{-5}, 4.198 \cdot 10^{-5}]$ km/s. The following orbit is shown in Figure 2.3.

Discovery	April 11 1996
Known Satellites	1
Rotation period	2.26 h
Distance of Didymoon	1.18km
Orbital period of the Didymoon	11.92 h
Diameter of Didymos	780 m
Diameter of Didymoon	160 m
Systems mass	5.279e11 kg
Density	1.7(+/- 0.4) g/cm^3

Table 2.1: Properties of the Didymos system[5]

2.3 Irregular body

The first perturbation that comes to mind when thinking of orbiting around a system of asteroids is the fact that they are not regular spherical bodies. Asteroids unlike larger bodies such as terrestrial planets and even gas giants are not forced to take spherical shapes due to the crushing gravity of mass accumulating in addition to the strong centrifugal force. Asteroids come in all sorts of shapes and sizes with uneven mass distributions and often with nonuniform revolutions around their axis [21].

One way to model and represent such an irregular body that gives a reasonably generic perturbation force that resembles the one that satellites can be subjected to is to give different regular geometric shapes to the bodies. The field of the gravity of such a body, that has a inconsistent mass distribution, has the property of not being centered compared to spheroid shaped geometries. This result in satellites orbiting with a different behaviour than keplerian orbits [22]. The latter is named to describe orbits subjected to the laws developed by Johannes Kepler and neglecting the perturbations, which will not be the case around this environment due to the fact that perturbation of the gravitational field will be taken into account[21].

As a previous study done at National Institute for Space Research of Brazil in 2012 the cube has been chosen for representing the irregular gravitational field. The effect and magnitude of the perturbation force has been defined by previous works in the last century of Mac Millan who defined the gravitational potential by the the following equation[23]:

$$U = \frac{Gm}{r} - \frac{7a^4Gm}{30r^9}[x^4 + y^4 + z^4 - 3(x^2y^2 + x^2z^2 + y^2z^2)] \tag{2.4}$$

The variables x, y and z are the coordinates of the position of the satellite that is orbiting around the body which is considered the origin of the frame. G is the gravitational

constant and m is the mass of the body. In this case the calculated mass of the body is $4.5 \cdot 10^{11}$ kg. In the equation a is the length of the edge of the cube which has been calculated to be 0.68 km in order for the simulated cube Didymos to have the same volume. To find the magnitude of the force of this perturbation in each axis this gravitational potential equation has to be derived to find the F_x, F_y and F_z components described in the following equations[24]:

$$F_x = \frac{7a^4Gmx(x^4 - 5x^2(y^2 + z^2) + (y^4 - y^2z^2 + z^4))}{6(x^2 + y^2 + z^2)^{\frac{11}{2}}} \tag{2.5}$$

$$F_y = \frac{7a^4Gmy(3x^4 + y^4 - 5y^2z^2 + 3z^4 - x^2(5y^2 + 3z^2))}{6(x^2 + y^2 + z^2)^{\frac{11}{2}}} \tag{2.6}$$

$$F_z = \frac{7a^4Gmz(3x^4 + 3y^4 - 5y^2z^2 + z^4 - x^2(3y^2 + 5z^2))}{6(x^2 + y^2 + z^2)^{\frac{11}{2}}} \tag{2.7}$$

This perturbation has not been implemented in Poliastro before, therefore a function with these perturbations had to be created in order to be able to plot and propagate for the time of the mission. From the properties of the cube and the initial orbit of the satellite the order of magnitude of the perturbation force is $10^{-12} km/s^2$ which has a considerable impact on the orbit.

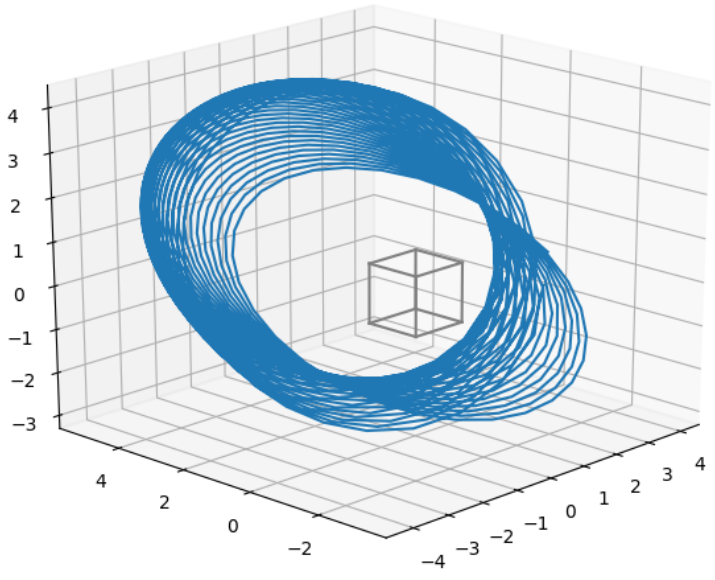

Figure 2.4: Cube perturbation plot

In Figure 2.4 the Orbit around the simulated cube with the perturbation over time is shown. The first noticeable effect that is important for the mission safety is that simulating this orbit around a cube creates a considerable change in altitude of the orbit going from

6 km of altitude down to 1.2 km which is very close to the distance to which Didymoon is orbiting. The right ascension of the ascending node also know as RAAN, which is the angle with respect to a reference axis at which the satellite passes over the defined equator of the body[25]. The RAAN over the mission time of 3 months oscillates from 6.3 degrees to 5.2 degrees following a cosine function. Moreover a lower magnitude oscillation over a couple of rad follow a sinus function is also noticeable which represents a shift every time that a new orbit period is made. The plots of the evolution of the Altitude and the RAAN of the perturbation are in Appendix D.

Other simulations of asteroids have been made by simulating a sphere like object with an exaggerated oblateness [26]. This term is signifies that the body is simulated with having a certain flattening effect making its shape become oval rather than round. This effect is already widely studied and taken into account when planning Low Earth orbit satellites. This effect also called J2 perturbation is defined by [27]:

$$J_2 = \frac{2\epsilon}{3} - \frac{R^3\omega^2}{3Gm} \tag{2.8}$$

In which ϵ is the ratio of the flatness of the body. R is the radius of the body and ω the rotation rate of the body. In the case of Dydimos the choice is made simulate with different oblateness ratios: 680m and 580m of diameter for the flattened part. As mentioned in Table 2.1 the main body does have a certain spin that also is taken into account in the equation.

J2 perturbation simulating an oblateness of the body is shown in Figure 2.5. It is important mentioning that there is no variation in the altitude with this simulation compared to the cube geometry. The RAAN changes faster when the flatness of the body is increased. Compared to the cube simulation the visual coverage of the body from the satellite point of view is higher, meaning that the satellite passes over a higher amount of portions of the surface of the body. The body spin will reduce the effect of the oblateness. If the body spins fast enough a not perfectly round body will have a lower impact on the orbit of the satellite.

2.4 3rd Body perturbation

Another important factor of the environment of Didymos is that it is a binary system. Didymos has a natural satellite Didymoon. As shown in table 2.1 Didymoon is orbiting around Didymos at a distance of 1.18km and has a mass of $1.1 \cdot 10^{11}$ kg. This is not an

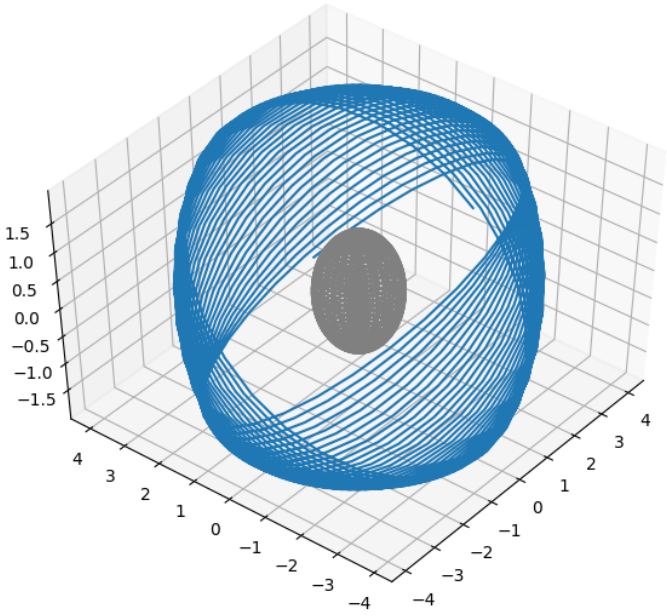

Figure 2.5: Oblateness perturbation plot

negligible perturbation when it comes to modelling such an environment. Therefore the following perturbation is defined as follows [28]:

$$a_{2/1} = \left(Gm_1 \frac{R_1 - R_2}{\parallel R_1 - R_2 \parallel^3} + Gm_3 \frac{R_3 - R_2}{\parallel R_3 - R_2 \parallel^3} \right) - \left(Gm_2 \frac{R_2 - R_1}{\parallel R_2 - R_1 \parallel^3} + Gm_3 \frac{R_3 - R_1}{\parallel R_3 - R_1 \parallel^3} \right) \tag{2.9}$$

There are two distinguishable parts to the equation which are two accelerations subtracted from each other. This is because it is the calculation of the relative acceleration of the bodies. R1 is the position main body, R2 the position of the satellite and R3 the position of the perturbation body, in this case Didymoon. The masses m1, m2 and m3 are respectively the ones of the mention bodies above.

From the simulations as shown in Figure 2.6 it can be noticed that compared to the two other perturbations modeled above the satellite does not keep orbiting around the body and is pushed out by the perturbation body after a time interval of four days. The initial velocitiwa of the satellite are to important for remaining in orbit with the perturbation of the 3rd body. The escape of the orbit can also demonstrate the limits of the Cowell method where the magnitude of the force errors add up drastically. Nevertheless for the demonstrator it is ideal for creating a controller to counter this effect in order for the

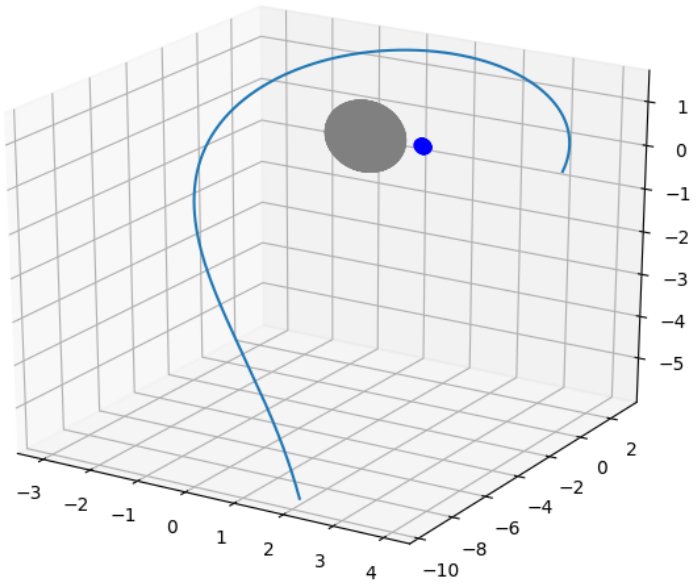

Figure 2.6: 3rd body perturbation plot

satellite to stay in orbit.

2.5 Radiation pressure

In the model another perturbation can be added which could affect the satellite. This perturbation is known as the solar radiation pressure which is a mechanical pressure applied on the body exposed to the force of the electromagnetic radiation of each wavelength that is a absorbed by the surface of the satellite. The radiation pressure is then dependant on the distance of the body to the star, the time of exposure and the surface of the satellite subject to the pressure. The radiation pressure is defined by the following [29]:

$$\overrightarrow{p} = -v\frac{S}{c}\frac{C_r A}{m}\frac{\overrightarrow{r}}{r} \tag{2.10}$$

C_r represents a dimensionless pressure coefficient between 1 and 2 depending on the geometry of the satellite, A the surface of the satellite and m its mass. S is the radiation power coming from the star and c the velocity of the light. The variable v is defined by algorithm implemented in Poliastro which is the shadow function.

The shadow function was used to determine depending on the radius of the body and the satellites position when the satellite is not subjected to radiation pressure because the satellite is not always subjected to this force. Finally r is the position vector of the satellite.

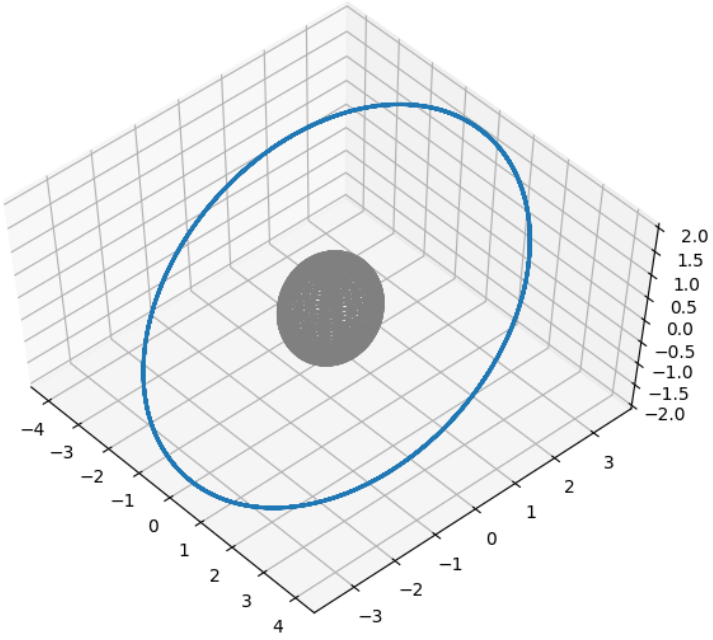

Figure 2.7: Radiation pressure perturbation plot

As shown in Figure 2.7 the radiation pressure does not have a clear effect on the orbit of the satellite within the boundaries of the simulation. The hypothesis is that the effect of solar pressure would have more effect on a longer mission and also if the satellite was orbiting around the bodies at higher altitude. Usually solar pressure has very little effect on low earth orbit satellites and has more consequences on the Medium Earth orbits and High Earth orbits in which the satellite will have a increase in the velocity resulting a change in position during the time of exposure [30].

In this chapter the different perturbations have been shown to model the effects having impact on different variables of the orbit such as the altitude, the RAAN and also the fact that some perturbations need to be controlled for the mission to run the way it has been planned. Nevertheless this part of the study to create the demonstrator of this Hardware-Software facility has been focused on the Didymos environment only. Therefore for the re-usability the collection of tests code made for plotting orbits will put online.

More specific to the company, the environment created with Python will be set on a virtual box installed on the server of the company in order to facilitate the use of this application and create more accessibility when developing projects. A code is written in order for people to easily put as input a combination of perturbations they want to see the effect from, the asteroid of interest and change its properties. Now having the software simulations of the Orbit the means to connect to the hardware have to be designed, which is the topic of next chapter.

Chapter 3

CubeSat model for simulation

In the previous chapter the creation of the simulation of the environment was discussed, which is one module of the Hardware-Software Facility. In this chapter, setting up the software and the hardware representing the onboard system of the satellite for the demonstrator will be elaborated. In a first part specifications of the APEX CubeSat will be reviewed since the demonstrator is motivated by the payload of the HERA mission. Furthermore the design of the Hardware Software facility will be elucidated to show how to connect the different components together, followed by a section explaining the choice of the hardware in use for the demonstrator and finally a section about the embedded software that will be used for this project.

3.1 APEX CubeSat Specifications

The APEX CubeSat will be released from the parent spacecraft after reaching the arrival at the Didymos asteroid system. It is part of the Planetary Defence objective of the mission in the sense that it will study the the effect of the DART impact by gathering data from the both asteroids. Its secondary objective is the study of the asteroids in the purpose to get a better understating of the formation of our Solar System.

Size (deployed)	100x365.9x226.3mm
Size (stored)	1155.16x365.9x2526.3mm
Mass	11.86kg
Propellant	120g (with margin)

Table 3.1: APEX CubeSat structure [31]

The data from table 3.2 containing the dimensions and the mass of the satellite have been used for determining the effect of radiation pressure on the mission of the satellite in the simulation in Poliastro. The specifications of the CubeSat are then used as input to the simulations equations when it comes to the mass and surface of the body. Another variable that is taken into account for the simulations is the mission time of the Satellite. The mission is scheduled for 3 nominal months which is the reason why 120 days is the time used from propagating the simulations.

In the paragraph above the information of APEX used in the software simulation has been mentioned. The focus should be brought to the elements that are modeled by the hardware part of the demonstrator. Therefore it is important to mention that the APEX satellite even by being a payload of another space craft is a stand alone system made up of several subsystems as any other satellite.

The APEX satellite that has payloads of its own which are a Hyperspectral Imager, a Fluxgate magnetometer and a Mass spectrometer. The first payload called ASPECT will be perform measurements from 500 nm to 2500 nm wavelenght. This bandwidth is covered by 3 channels respectively capturing the visible spectrum, the near infrared and the short wave infrared in order characterise the surface Didymoon and Didymos. The second instrument is used to study the magnetic field of the bodies and the last one is used to study the composition of the charged particles around the asteroid. In addition to these payloads to fulfill its scientific objectives the satellite also the other subsystems needed for functioning: the telemetry and telecommand, the Attitude and Orbit control and the power subsystem.

In order to control all these different subsystems the onboard data handling system (OBDH) is the one operating the commands and routines. The central part of this process is the onboard computer which is the Texas Instruments RM48L952. It will have the function to ensure the processing power for each subsystem, manage the telemetry and provide autonomy. A lot of data comes from the payload which are not directly processed by the onboard computer but by auxiliary processors. Therefore only the required data for decision making is communicated to the onboard computer [31].

The way that the communications is made between these different processors and subsystems is via a CAN Bus, which will be further explained in Chapter 4.

These two elements: the CAN bus and the on board computer are then chosen to be the target of what the hardware should represent. The way that they will interact with the simulation will be explained in the following section.

3.2 Hardware-Software Test Facility Design

The different parts that make up the Hardware-Software testing facility are shown in Figure 3.2. Unified Modelling Language to represent each dependency and functionality that makes up the system. The diagram is showing at a higher level of abstraction the required components of the system before having to choose or decide on specific hardware components, of software language and the type of communication bus. The main choices for the design of the hardware part of the testing facility is to provide a modular design that can be easily reused for other development projects. Another important factor is to make it user friendly since this project aims at demonstrating the possibilities of Hardware-Software testing using open source applications in order to raise the creativity of future developers.

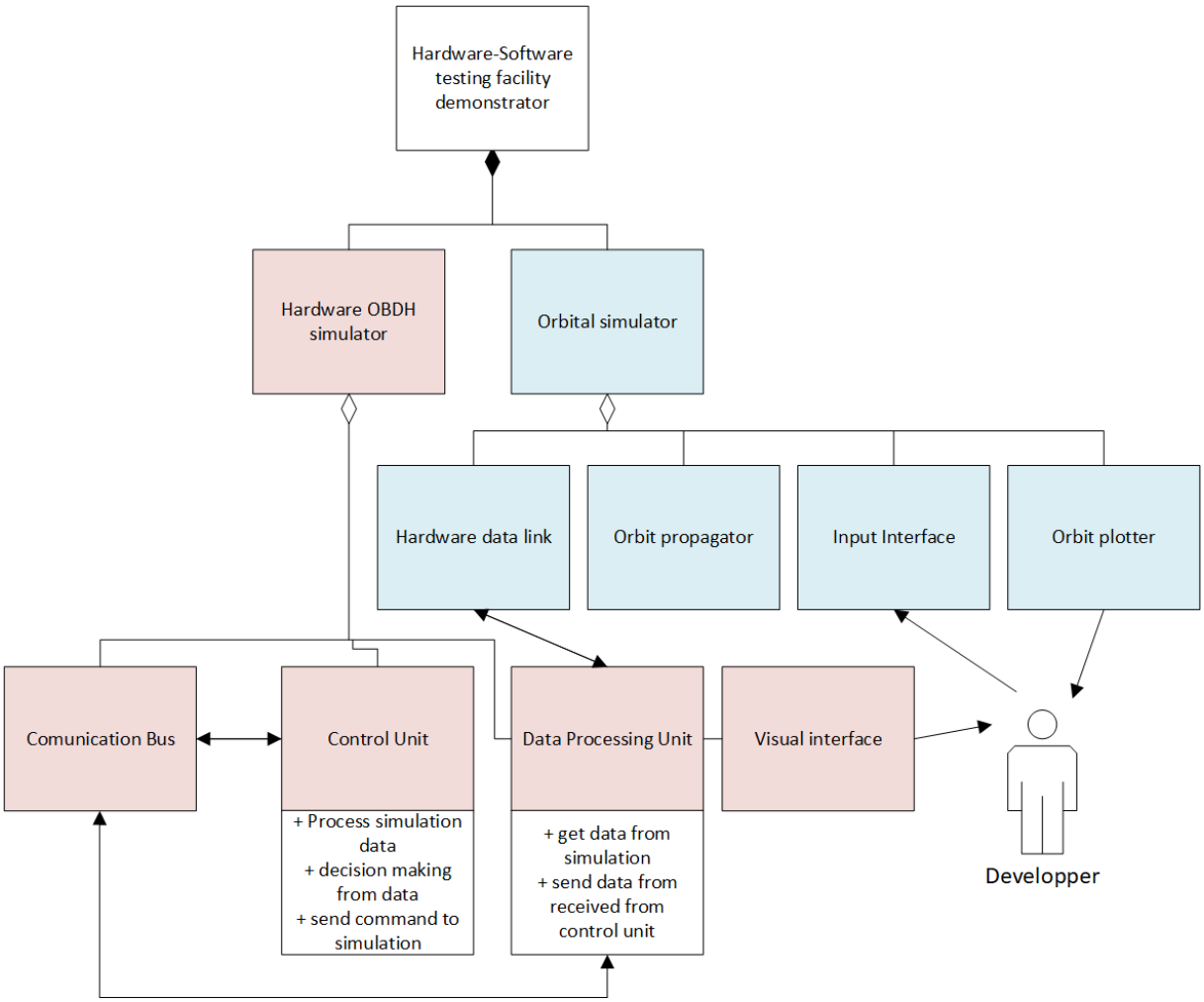

Figure 3.1: System model of the testing facility

It can be noticed in the first place the clear separation of the elements needed for the orbital simulator that are software applications running on a computer and the Hardware

part simulating the OBDH subsystem as well as the communication bus used in the satellite between the different subsystems. The Hardware-Software testing facility is a composition of those two elements the Hardware and the Simulator.

The Orbital simulator is an aggregation of several elements that are independent from each other. In the following order: the user interacts with the interface where he inputs the properties of the satellite orbit, the properties of the environment and the perturbations involved. The orbit propagator algorithms calculate the position over time of the satellite, the Hardware data link checks if any commands are sent from the hardware and sends the orbital data. For the testing facility to be interactive and visual the Orbital plotter is also a important part of the system.

The OBDH and communication bus from spacecraft simulator is first of all an aggregation between the data processing unit, the communication bus and the control unit. The data processing unit is the entity that is required to make the bridge between the simulation on the computer and the hardware simulating the OBDH on the satellite. Its functions are: acquire the data from the simulation needed to be processed, for instance in the frame of this project the data of the position and velocity vectors of the satellite from Poliastro and send it via the physical communication bus to be processed as well as receive data from the communication but to send it as input to the propagation of the orbit in the simulation. The communication bus has the function to make the link between the Control unit and data processing Unit and in that way to mimic exactly the way the data will be transferred within the satellite. Moreover the Control unit has the function to send the commands to the simulation through the other entities when the data processed leads to a decision on the orbit control.

The user has also a visual interface to increase the interactivity of the demonstrator on the hardware side in addition to the visual and interactive interfaces present on the side of the orbit simulation.

The advantage of such a design is that most of the elements are not compulsory in order of this Hardware-Software testing facility to be operational. The orbital simulations can work as a stand alone not needing the hardware part to be connected. Within the orbital simulator the orbit plotting and the interface with the user are parts to improve the exchanges with the user but can be disregarded. Same goes with the hardware part, one can disregard the communication bus to only focus using one processing unit for other applications.

3.3 Hardware of the Testing Facility

After creating the design of the Testing facility, the next step is to choose the adequate hardware that will fit the needs of this project. The importance of the choice of the hardware is not only for the relevance for the project but also should be thought for developing other applications.

Properties	APEX Microntroller	Demonstrator 1	Demonstrator 2
Micro computer	RM48L952	ATSAMC21J18A	STM32F446RET6
Frequency (MHz)	220	48	180
Flash (KB)	3072	256	512
Data flash (KB)	64	32	32
USB	2	1	2
CAN	3	2	2
I2C	1	6	1

Table 3.2: Microcontroller choice for demonstrator

Two microcontrollers were candidates for being used as hardware of the demonstrator. Two of them will be needed to create the the Hardware test facility. Both the ATSAMC21J18A and STM32F446RET6 became candidates because of the experience of working with these microcontrollers. It is important for them to have CAN and I2C for testing and developing projects since these are commonly used communication buses within Satellites and other industries.

The ATSAMC21J18A is programmable and supported in the C language which has software examples for CAN bus communication and other applications by the Microchip programming environment. It has space heritage since it is used in the Space industry for controlling a wide range of instruments [32]. It is for instance used by the French thruster developing company Thrustme to control their ion thrusters. It comes along the Xplained XPRO evaluation board for easing project development and testing.

The STM32F446RET6 has supported language C as well but also has the possibility to be programmed with MicroPython which makes a greater cohesion with the orbital simulator that is written in Python [33]. It does not have space heritage but it has nevertheless been used in other thesis projects in the aerospace industry such as in the university of Delft, The Netherlands. The choice has been made for this microcontroller development board NUCLEO-144 due to the programming language it can support, higher performance and availability in the company Huld.

3.4 Embedded Software of the testing Facility

The choice of the programming language for the testing facility is MicroPython. It is an effective implementation of the Python 3 programming language that includes a simplified portion of the Python basic library and is optimised to be used on microcontrollers and in restraint environments [34].

One of the major aims of this programming language is to ensure the compatibility with the regular Python scripts in order to facilitate the development of routines. As a consequence MicroPython allows the possibility to transfer the Python scripts made on the Desktop directly to the microcontroller or other embedded system used for projects.

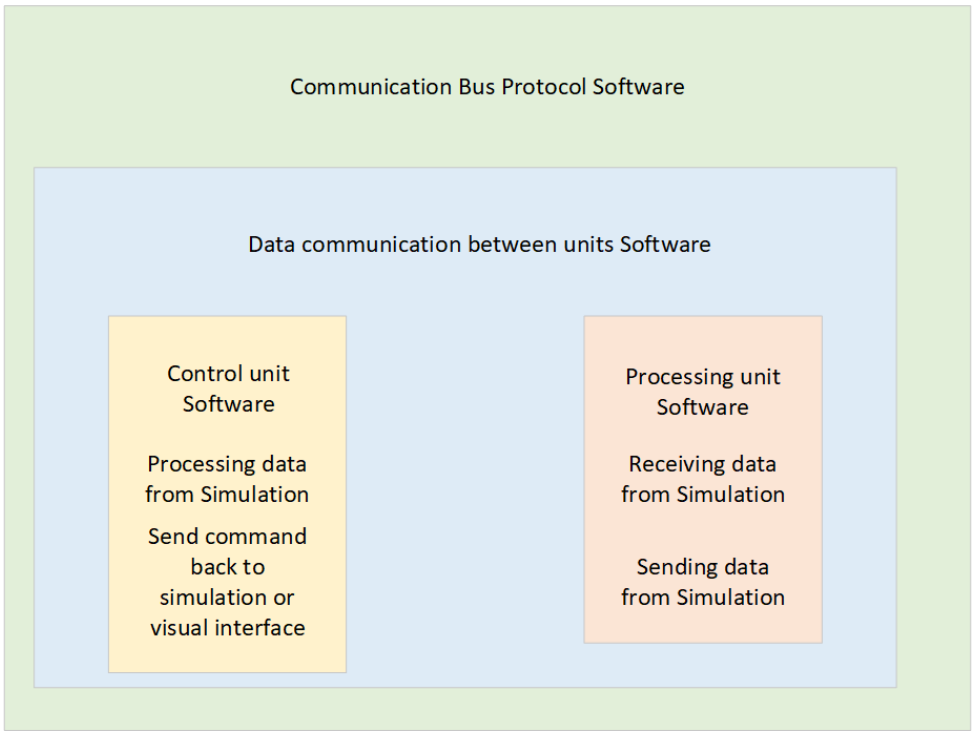

Figure 3.2: Embedded software of the testing facility

As shown in Figure 3.2 the embedded software that is made and used for the Hardware-Software Facility consists of different layers. The top layer which consists of the libraries defining the communication protocols used for the buses between micro controllers and then for both controllers the lower level communication routine.

Chapter 4

Implementation of the CAN BUS

This chapter explains the implementation of the CAN Bus within the project of the Hardware-Software testing facility. The purpose of this setup is to simulate the communication between subsystems on the APEX mission and other potential satellite missions the company Huld will be working on. This implementation demonstrates that software simulations can be tested directly with physical hardware at a reduced cost and in a simultaneous way, debugging the software failures and Hardware-Software related issues. Therefore the use of the CAN bus on the APEX mission will be elucidated in a first part. Later the design of the physical CAN Bus will be explained followed by the structure and protocol used for the implementation. The software of the CAN bus of this project is available on:

`https://github.com/NielshuldC/Simulation-AOCS-APEX-CubeSat/tree/master/CAN_bus`

4.1 CAN Bus in CubeSat

The CAN bus is a communication system originally developed for the automobile industry but rapidly adopted in the space industry for its design being reliable and robust and reducing on the amount of physical wiring needed to connect a greater amount of devices [7]. Reliability is the key to the usage of this vehicle bus, it is not designed for transferring data taking a lot of volume such as images and data science [35]. This communication system allows microcontrollers and other devices to send messages on the same bus without having a main computer acting like an hub to transfer the messages from device to

device.

The CAN bus is a protocol based on messages in which each node or device connected is able to send a receive messages. Nevertheless the nodes cannot communicate simultaneously, therefore the first part of the message or frame contains the identifier in front of the data to be transmitted that defines the priority of the message. The frames are then transmitted sequentially by order of higher priority to lowest priority leaving one device send on the CAN bus while the others receive. Each message is received by all the devices connected to the system as well as the device that is transmitting.

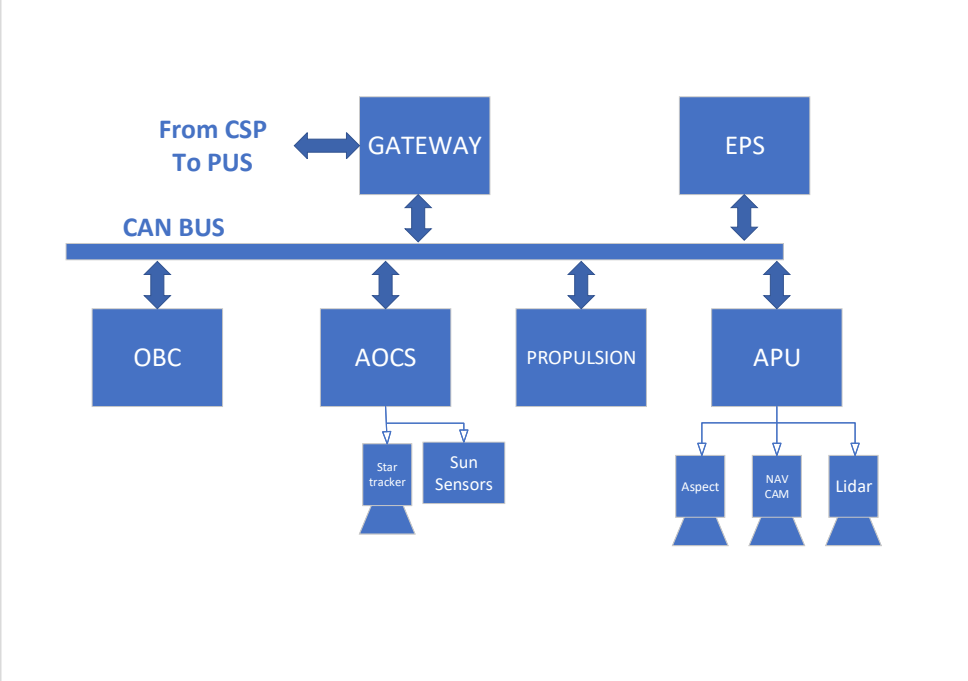

Figure 4.1: CAN BUS diagram APEX [31]

In the APEX mission the CAN bus is used for connecting the different subsystems as shown in Figure 4.1. The nodes that communicate are the onboard Computer, the AOCS that gets the orientation of the CubeSat, the Electrical Power System (EPS), the propulsion unit, the gateway and the Auxiliary Processing Unit (APU). The decisions of making of a maneuver and making use of the propulsion system in this design comes down to a message sent to the propulsion via the CAN bus which will be one of the applications of the demonstrator of this project.

In addition to the lower level CAN bus an higher layer protocol is used by the APEX satellite: the CubeSat Space Protocol (CSP) which is a protocol written for embedded system written in C. The role of the gateway is for the Satellite to communicate with the main spacecraft HERA: it receives commands in the form of space packets transmitted with the Packet Utilization Standard (PUS) and converts these headers with CSP so that

the commands can be transmitted via the CAN bus to the OBC.

4.2 Physical CAN Bus

As mentioned in the section above, the communication system has several protocol layers. In this section the hardware and lower level will be discussed. The first step when it comes to create a CAN bus is to create a signal which has a different logic the the regular pin output of a microcontroller.

The microcontrollers STM32F767 have a CAN controller but lack a CAN transceiver to convert the data to the physical layer therefore the MCP2551 transceiver was used for this purpose and connect as on Figure 4.2.

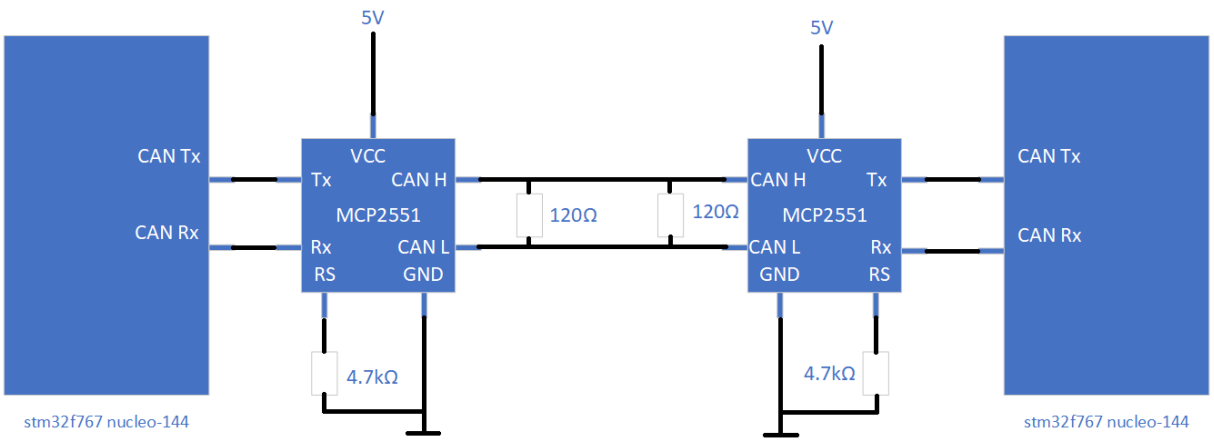

Figure 4.2: CAN BUS diagram testing facility [36]

The next step is to tune the timing of the CAN bus of each bit which was done by software in Micropython. Since it is an asynchronous communication protocol the moment of the sampling point has to be adjusted for each microcontroller to receive the messages correctly and was done following guidelines presented in Appendix B. The reason why the lower level was made first is for reusability of the knowledge and hardware to connect other microcontrollers than the STM32F767. In this project Arduino boards have also been connected to the CAN bus for testing.

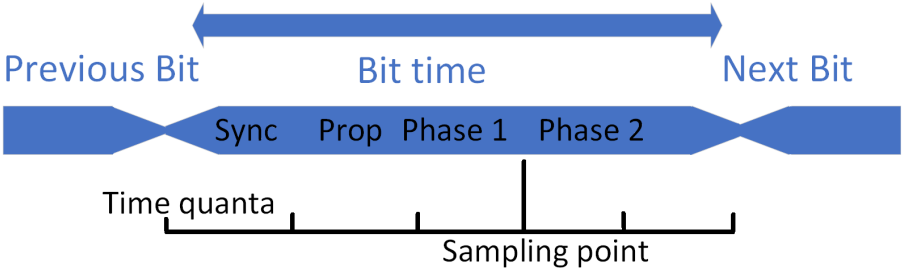

Figure 4.3: CAN BUS timing diagram [7]

4.3 CAN Bus structure

For the higher level structure of the CAN bus to comply with specifications from the ECSS-E-ST-50-15C standard the SpaceCAN protocol is implemented. In addition to be a space qualified protocol the choice of this usage has been made since the libraries have also been developed for Micropython matching the requirements of this demonstrator project.

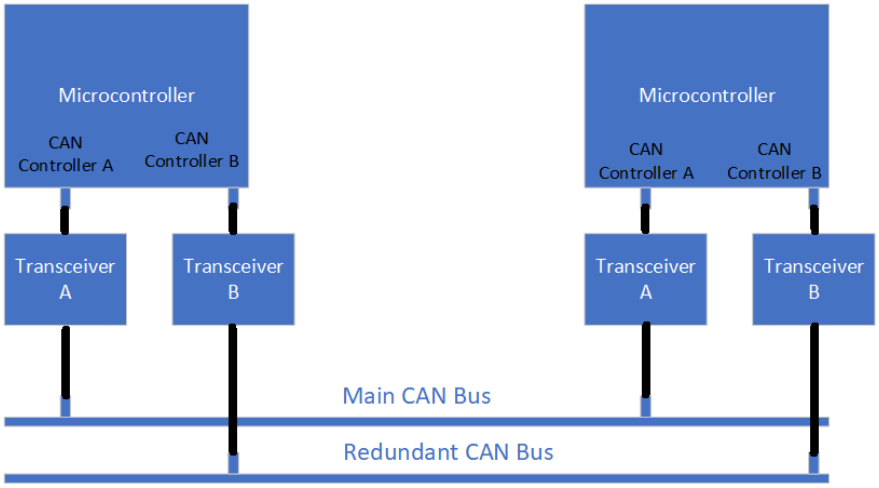

Figure 4.4: CAN BUS parallel access redundancy

From the ECSS guidelines the peculiarity of SpaceCAN is that it is developed for a system with a main and redundant CAN bus to ensure a reliable communication. For this project the parallel bus access design has been implemented as shown in Figure 4.4. This protocol follows guidelines also set on a lower level with the maximum bit rate of this implementation being 1 Mbps and an identification system for priority is 11-bits in which 4-bits encode the type of service and 7-bits for the identifier address [35].

Chapter 5

Attitude and Orbit Control of the CubeSat

This chapter deals with the Attitude and Orbit Control of the project. This is the subsystem that enables in every satellite the information to be processed for its navigation and position. First the AOCS that inspires the demonstrator will be elucidated followed by the first application of the demonstrator to acknowledge the position of the satellite. Furthermore the application of orbit transfer will be developed and finally the application to counter the perturbations of the asteroid environment. Software related to this chapter can be found on:

`https://github.com/NielshuldC/Simulation-AOCS-APEX-CubeSat/tree/master/AOCS`

5.1 AOCS in the APEX mission

The AOCS in order to acquire the position of the APEX CubeSat uses a sensors that are shown in Table 5.1. The Star Tracker, sun sensors and Inertia measurement unit provide the data of the orientation of the CubeSat, the Lidar and the Navigation Cameras estimate the distance of the Satellite to the body it is orbiting.

Along with the sensors, when the position is estimated, aligning or changing the orbit of the CubeSat has to be performed with the use of actuators. Reaction wheels permit to change the orientation of the satellite, by rotating create a change in the moment of inertia in order to either stabilise, which is also referred as detumbling or change its orientation for instruments to have the asteroid in its field of view.

Sensors	1 x Star Tracker 4 x Sun Sensors 2 x Navigation Cameras 1 x Lidar 1 x IMU
Actuator	3 x Reaction Wheels (6mNms each) 8 x Thruster Heads (1mN each)
Propellant	Butane

Table 5.1: APEX CubeSat ADCS and GNC [31]

The actuators that will be simulated in Poliastro and controlled by the microcontroller in the demonstrator of this project are the thrusters. The propulsion system is used for the CubeSat to stay in orbit by exerting force using the propellant for countering the forces that might endanger the orbit of the satellite as well as transferring it to a new orbit.

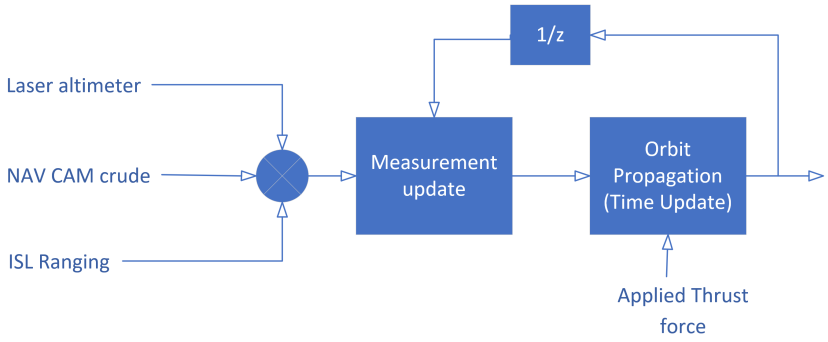

Figure 5.1: AOCS diagram APEX [31]

The way that the control of the orbit is made for the APEX is as follows in the diagram in Figure 5.1. The position of the Satellite is estimated using 3 different methods. The first instrument used for estimating the position is the LIDAR, Laser imaging detection and ranging, one of the payloads of the APEX mission which is used for scanning the surface of the asteroids but also used for measuring the distance between the body and the satellite.

Furthermore the data gaps are completed using a second payload of the mission the Navigation Camera. It is used for characterising the surface of asteroids but also used for determining the position since it has to work along with the LIDAR that can only gather data at close range of the body.

Finally the last method for estimating the position is by using the Inter satellite link which is the communication network in this case between the APEX CubeSat, the main space craft HERA and the other payload CubeSats. This data transmission method is also reused for determining the distance between the satellites by using the time division

signal of the ISL [37].

All this data is then used for estimating the position of the satellite relative to the asteroid and the other satellites. As shown in the diagram the control of the orbit, decision to apply thrust is evaluated after every update of the position therefore the control system is shown as a discrete time feedback system.

5.2 Orbit determination application

For the first application of the Hardware-Software testing facility an orbit determination communication has been developed. This application is made to simulate how satellite position data will be transferred from one processing unit to another one in the satellite. This application demonstrates how code developed in Python can be uploaded on the microcontroller to directly test the software and hardware through the CAN bus along with the orbit simulation.

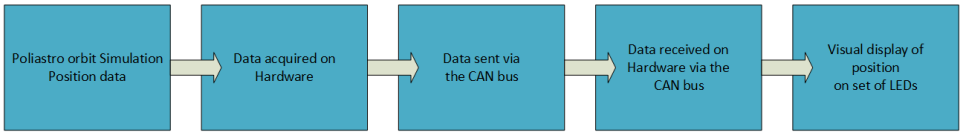

Figure 5.2: Orbital data transfer diagram

In this project, abstraction is made from the sensors used for determining the position of the satellite. The input of the data is be taken directly from the orbit propagator and assumed to be the one that the simulated hardware of the satellite is getting for knowing its position. Therefore, the position will be given in 3 dimensions with the origin being at the center of the main asteroid, Didymos.

In this application the data from the orbital simulation from Poliastro as seen in Figure 5.2 goes via serial communication to the microcontroller to be acquired on the hardware and to the other hardware unit via the CAN bus. To make this demonstrator more interactive and more user friendly, to enhance creating more developed applications, 4 LEDS are connected to the microcontroller as a visual interface.

Similar to an instrument tuner showing when right frequency is reached, gives an indication whether or not the nominal orbit is followed, the one that the satellite would follow if no perturbations were involved. With the set of LEDs the user can see if external factors make the orbit decrease or increase in altitude of the satellite with respect to the center of the body.

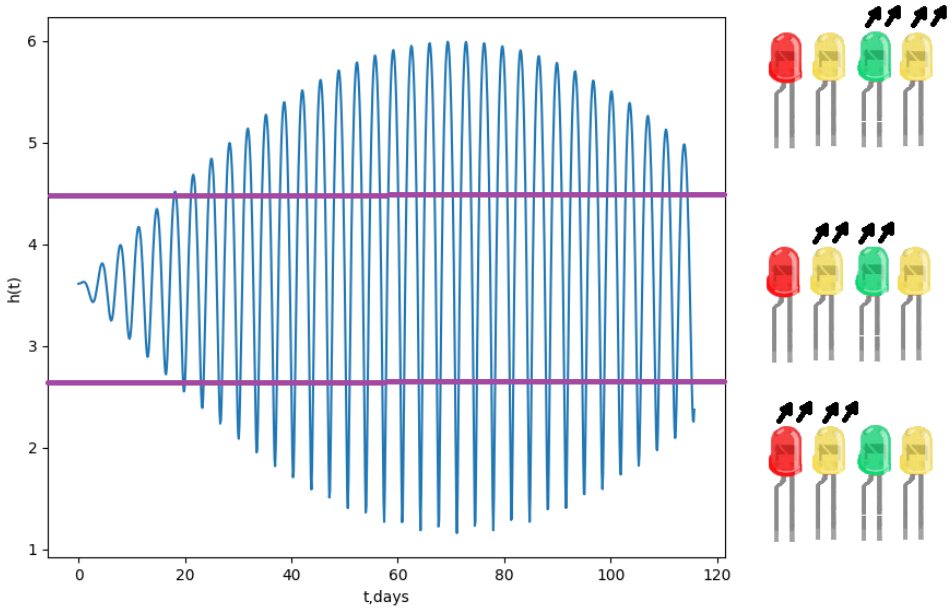

Figure 5.3: Altitude acquisition and visualisation

As shown in Figure 5.3 during a simulation of orbit with cube shaped object perturbation the LEDs in the center are lit up when the satellite is within the nominal altitude it is orbiting at. When the orbit altitude goes lower or higher than the nominal the LEDs for the sides are accordingly lit up.

5.3 Maneuvering of the satellite application

APEX will have to perform several maneuvers during its life cycle. From its release from the main space craft HERA, the CubeSat will have to perform several orbit transfers. These changes of orbit and mission length are shown in Table 5.2 to show typical mission phases that can be planned for such CubeSats. During the mission the different orbits that the APEX CubeSat will go through are circular orbits around Didymoon and Didymos and hyperbolic orbits to transfer from one to the other. Each orbit transfer can be performed autonomously by the Satellite and makes uses of the data acquisition of the orbit and internal clock of the satellite. Nevertheless the orbit transfer can also be done by command that the CubeSat can receive from the ISL from the main space craft.

Therefore, the demonstrator has one application developed for the orbit control for making such an orbit transfer. The data from the orbit position and the time of the simulation are transferred via the CAN bus to be processed on the microcontroller. A decision is then taken whether to change orbit or not by sending a command again over the CAN bus and then via the serial communication back to the simulation in Poliastro.

Phase	**Tasks**	**Trajectory and attitude**	**Duration**
Preparation	space craft Checkout	Same as the main space craft	
REOP	P/F commissioning	Hyperbolic/ Didymain pointing	12 days
OSSO	D1/D2 mapping ACA/MAG commissioning	4 km circular/ Didymoon pointing	31 days
ISSO	Crater side mapping D1/D2 science	L5/Didymain and Didymoon pointing	23 days
ISSO	No-crater side mapping D1/D2 science	L4/Didymain and Didymoon pointing	24 days
ISSO	Science D2 surface mapping	Flyby between L1 and L2/ Didymoon pointing	70 days
Landing	D2 proximity and contact	To moon surface	20 days

Table 5.2: APEX CubeSat mission stages [31]

To make the demonstrator more interactive, as in the previous application made for the hardware-software testing facility, user buttons present on the STM32F767 are used as well to send commands to make a change of orbit on the simulator. In addition to data processing of the simulation the orbit transfer can also be triggered by pressing a physical button.

The type of orbit transfer used for the demonstrator is the Hohmann transfer. It is typically used for transferring a satellite to higher or lower orbit for earth missions or interplanetary ones. The principle of this orbit transfer is that it is done in 2 steps. In the first step the satellite is in its original orbit and is subjected to an impulse from the thrusters in opposite direction of its trajectory if the satellite has to be transferred to a higher orbit and in the direction of the trajectory if the transfer is to a lower orbit. After this impulse the satellite goes from a circular orbit to a orbit with a high eccentricity.

To return to a circular orbit a second step has to be made in order to finish the entire process of the Hohmann transfer. After half a period of the new orbit of the satellite a second impulse is given. The second impulse is given in the same direction as the first impulse in order to remove the eccentricity to the orbit. After this second step the satellite

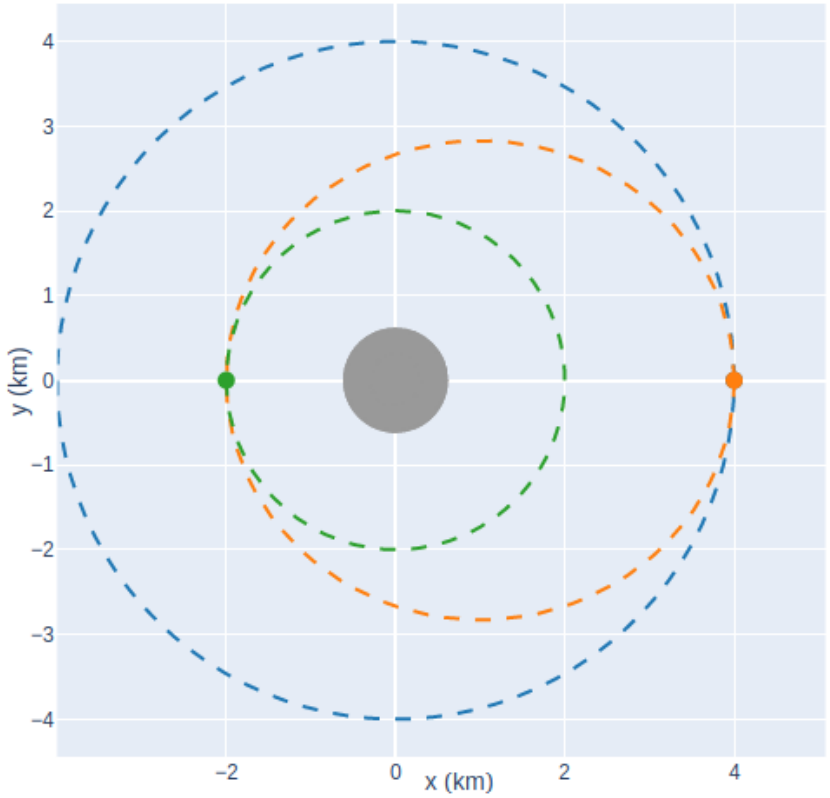

Figure 5.4: Hohmann transfer testing facility application

is in a higher or lower circular orbit than its original one [38].

The microcontroller sends then the command to start a Hohmann with the value needed for the impulse via the CAN bus and then via serial communication. Once the command is received in Poliastro, the simulating adds a perturbation to the simulation. This perturbation is already programmed in Poliastro and is programmed to simulate thrust from the propellant of the satellite. The function of this perturbation takes 2 inputs: the velocity change δv and the duration of the impulse. Therefore when the command is received on the simulation the new perturbation of the impulse of the thrusters is then propagated in the simulation making the change in the orbit.

In this application demonstrating the commanding an orbital transfer via the CAN bus as it would be done on the satellite at the end of the period of propagation from the simulator using Poliastro is shown in Figure 5.4. This transfer is commanded by the microcontroller autonomously or manually by the user.

5.4 Countering perturbation application

Along with acquiring the data of the position of the satellite and performing orbital maneuvers the satellite usually makes use of its thrusters to counter the disturbances that have an effect on the orbit. Disturbances as shown in Chapter 2 can have an effect on the altitude of the satellite, the RAAN and the length of the mission. This application of orbit control allows to extend the lifetime of the satellite by keeping it longer in an operational orbit.

For the demonstrator the application has been developed to counter the perturbation simulated with Poliastro and to get the satellite to return to its nominal orbit by inducing thrust perturbation from the propulsion system of the satellite. The orbital data is still sent via the CAN bus and then an algorithm on the microcontroller sends back to the simulation the values of the thrust perturbation to be applied.

A classical control system is implemented calculating the thrust needed in which the signal is the satellites position and it is compared to the its reference position which are the coordinates that the satellite would be in if it was not subjected to perturbations. The difference between the two signals is then characterised as the positional error of the

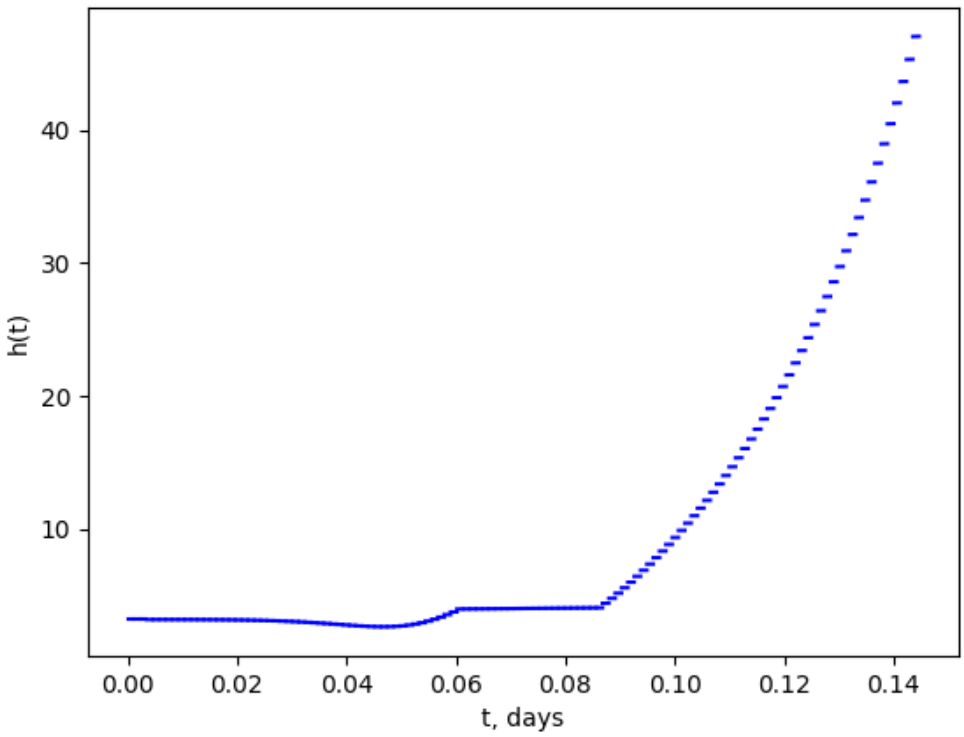

Figure 5.5: PD controller resulting orbit with 3rd body perturbation

satellite which leads to a decision taken to induce an impulse from the thrusters. The

force needed to be applied in during the impulse is calculated by taking the positional error and apply a PID controller implementation. The error signal is then multiplied by a factor, integrated and differentiated and summed up to generate the force to be applied.

Since the only perturbation that following the simulation threatened the orbit of the satellite was the 3rd body perturbation, it has been the one used for developing the controller. For the first algorithm a PD controller has been implemented since the values given by the simulations where the position of the satellite as well as its velocity giving directly the signal to be proportional part of the controller and the differential part. Nevertheless the controller with such inputs does not stabilise the system as shown in Figure 5.5.

For this application of the demonstrator a PI controller is finally implemented for controlling the system with the resulting orbit of 120 days shown in Figure 5.6. The system remains unstable with high overshoots but it keeps the satellite orbiting around the Didymos system without leaving or crashing into the body as in Figure 2.6.

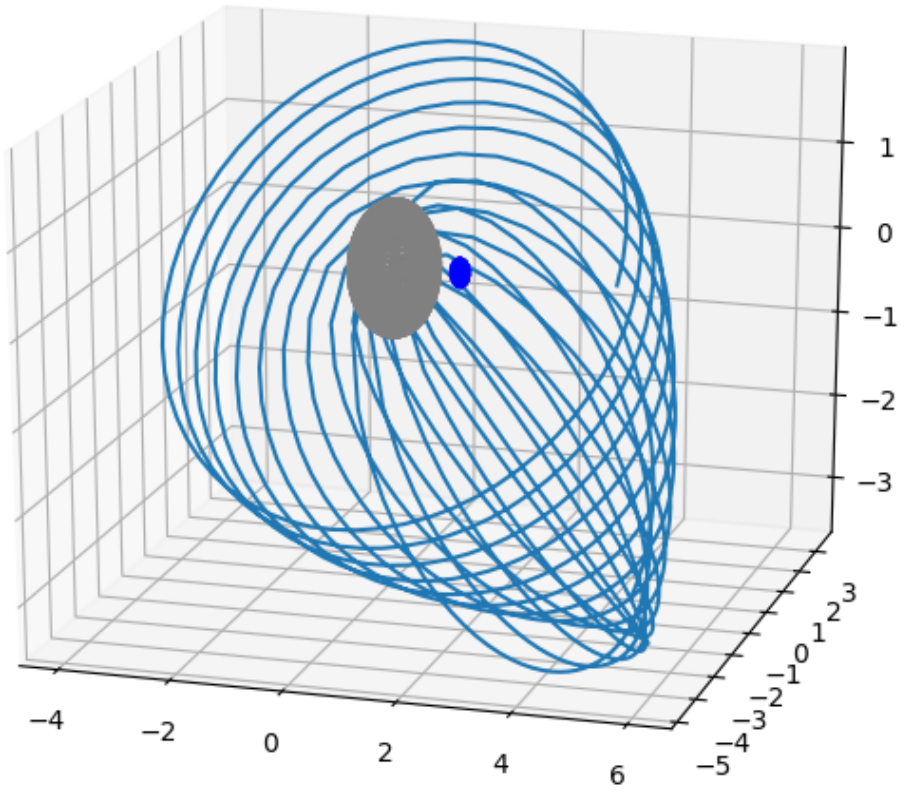

Figure 5.6: PI controller resulting orbit with 3rd body perturbation

Chapter 6

Conclusion

6.1 Summary of thesis

With as purpose to accelerate the development of satellite software applications and testing, during this work a Hardware-Software testing facility demonstrator has been developed. The application of the testing facility was a module of the simulation of the Attitude and Orbit control of a satellite. The design choices of the simulation and the testing facility were motivated by the the APEX CubeSat, a current project of the company Huld where the study was effectuated. The CubeSat will be sent to an asteroid system in 2024 as part as a planetary defense mission.

The thesis aims to explore the use of available open source applications to reduce cost of hardware and software simulators nowadays used by the company. Therefore the first step of the work was to developed a simulator for the environment that the APEX CubeSat will subjected to orbit around during its mission. After comparing the performance of the open source Python programming language libraries of Poliastro with commercial software it has been chosen to create the environment.

The creation of the asteroid environment for the simulation was an opportunity to study the effect of different ways to model the perturbations that an asteroid can have on the orbit of a satellite. Comparisons have been made between a cube shaped asteroid or a spheroid that was suggested by different studies. The asteroid being a none regular shape the best approximation is a combination of both those perturbations. In addition the effect of the asteroid having a natural satellite making the satellite influenced by different bodies has also been studied as well as the perturbation due to solar radiation.

The importance of the Hardware-Software testing facility is to demonstrate that software can be developed and directly subjected to the Hardware limitations for testing. The embedded software was chosen to be Micropython in order to have the same programming language as on the simulator. The idea was to connect the orbit simulation, which will be used as data coming from a possible instrument on board a satellite, to hardware and make them communicate and send commands the same way that it would be done on board. Two microcontrollers, the STM32F767, have then been chosen for this purpose and are connected as on the APEX CubeSat via physical CAN bus that has been build using MCP2551 transceivers.

Once the simulation and the microcontrollers were connected via the CAN bus several applications have been build to demonstrate the possibilities of such a testing facility. The SpaceCAN protocol was implemented for the communication on the bus in order to work with a robust system that is used for internal communication on existing missions in space and was also developed for Micropython. The goal was to make interactive applications related to Attitude and Orbit control of the satellite that the user can experiment with visual feedback from the hardware and from the simulation with orbit plotting.

The 3 applications for the Attitude and Orbit control on the hardware connected to the orbit simulation were developed.

1. The first one is the visual tracking of the error in altitude of the orbit of the satellites communicated via the CAN bus on a set of LEDs connected to the microcontroller.
2. The second application the sending of the command of orbit transfer to the simulation via the CAN bus by the microcontroller decision taking from the simulated data or by the action of the user.
3. The last application developed is the control of the orbit subjected to the 3rd body perturbation using a PI controller on the microcontroller.

6.2 Fulfilment of targets

At the beginning of the project three major objectives were set that are repeated here below:

1. Create a model of the dynamics of the CubeSat, its disturbance environment and to plot its trajectories using python libraries developed for orbital dynamics.

2. Program Attitude and Orbital control on a microcontroller to calculate the new trajectories for the simulation.

3. Create an Interface to send simulated data using the standard protocols developed for space crafts and using the communication bus used on the APEX spacecraft.

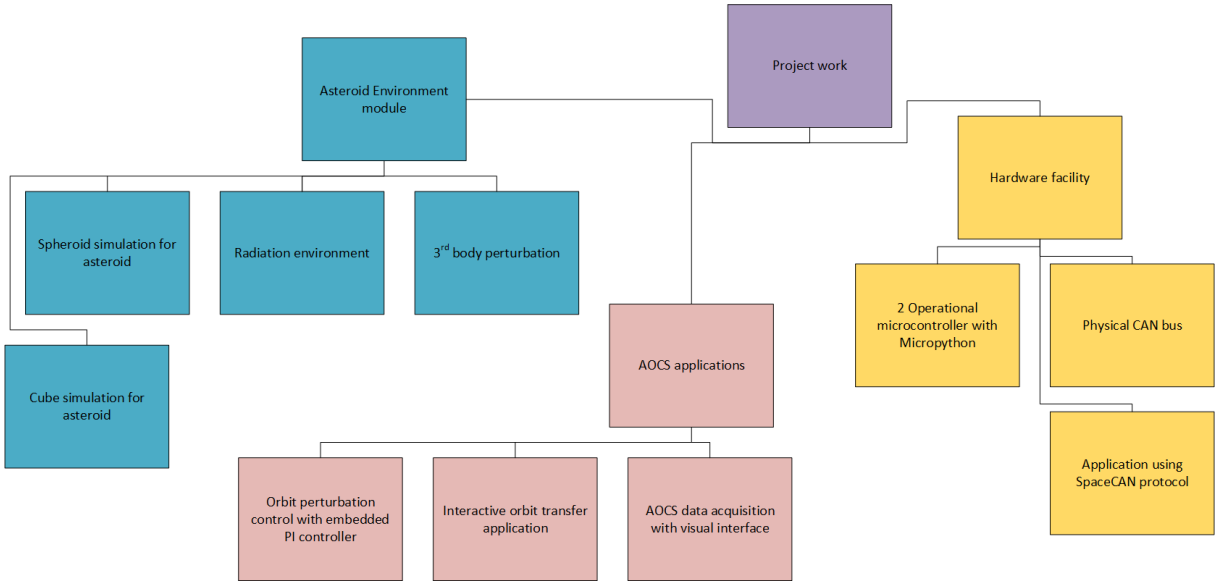

Figure 6.1: Project work modules

As shown in Figure 6.1 these are the different outcomes of the project. Three distinct outputs can be named, the work related to the exploring of the usage of open source software for the creation of Orbit simulations around an asteroid, followed by the elaboration of a hardware module to accommodate the software to be tested for different applications and finally 3 applications of AOCS developed using the Hardware-Software testing facility.

The first objective is reached by generating Python scrips using the Poliastro libraries for simulating the different perturbations that the satellite could be subjected to during its mission around the Didymos asteroid. It should be recognised that the simulations have only be been done using the Cowell method which is only one of the methods used for simulating orbits and their perturbations, it is a method easily implementable in software but presents limit with respect to the accuracy.

The second objective is reached by a module of the AOCS applications of the project programmed on the microcontroller connected to the orbit simulations. The embedded software has a script to generate orbit transfers on the simulation as well as a straightforward orbit PI controller. The orbit control has only been performed on the 3rd body perturbation being the one with most consequences on the simulation and the orbit transfer application does not involve other perturbations.

The third objective is reached by the work module related the creation of the CAN bus that has been made physically and at lower level in order to simulate the commands of the AOCS. Moreover SpaceCAN protocol has been used as well but not implemented for every application.

6.3 Further extensibility and recommendations

For the extensibility of the project two main activities could be undertaken. The first one that can be undertaken would be activities related to make the Hardware-Software testing facility more accessible by turning it into a plug and play type of product. The other activities that could be performed would be to extend the applications in order to explore the possibilities that could be developed.

To make the Hardware-Software testing facility more accessible the action could be taken to make an graphical user interface where the user could enter the asteroid he wants for the simulation, enter which properties and which perturbations involved. For the CAN bus a small printed circuit board (PCB) to make a ready for use platform to develop software.

When it comes to extending the applications in the simulations, more tests with combinations of perturbations could be made and most importantly with different integration methods than the Cowell method for propagating orbits to see the effects. To simulate exactly how the communication would be made on board the APEX mission the CubeSat Space Protocol (CPS) would have to be implemented as well as an application where a microcontroller translates Space Packets (PUS) into CSP. Other more robust control methods such as state space or even machine learning algorithms.

Recommendations relative to developing such a project are first of all related to the simulation. It is important to compare different integration methods for orbit propagators since the results can vary a lot. For open source libraries it is crucial to choose on that has an active community since this is the difference with commercial software where there is customer support. Furthermore in appendix B there are a lot of recommendations regarding how to set up a CAN bus as well as to make it operate on boards with Micropython on it.

Appendix A

CAN bus Setup

Here below is the wiring diagram showing the connection need to create the CAN bus between two STM32 Nucleo-144. When having Micropython the CAN controllers are not located on the pins indicated on the board but on the ones shown here in the diagram.

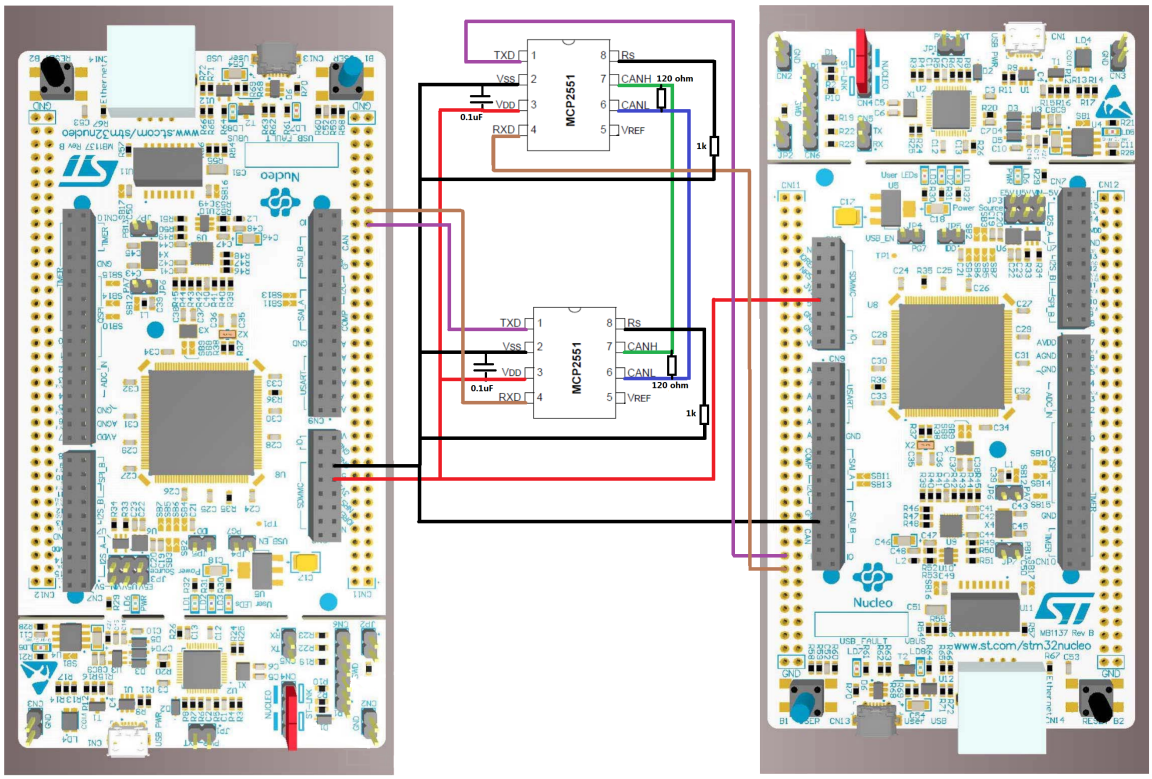

Figure A.1: CAN bus setup between 2 microcontrollers

Appendix B

Debugging CAN Bus

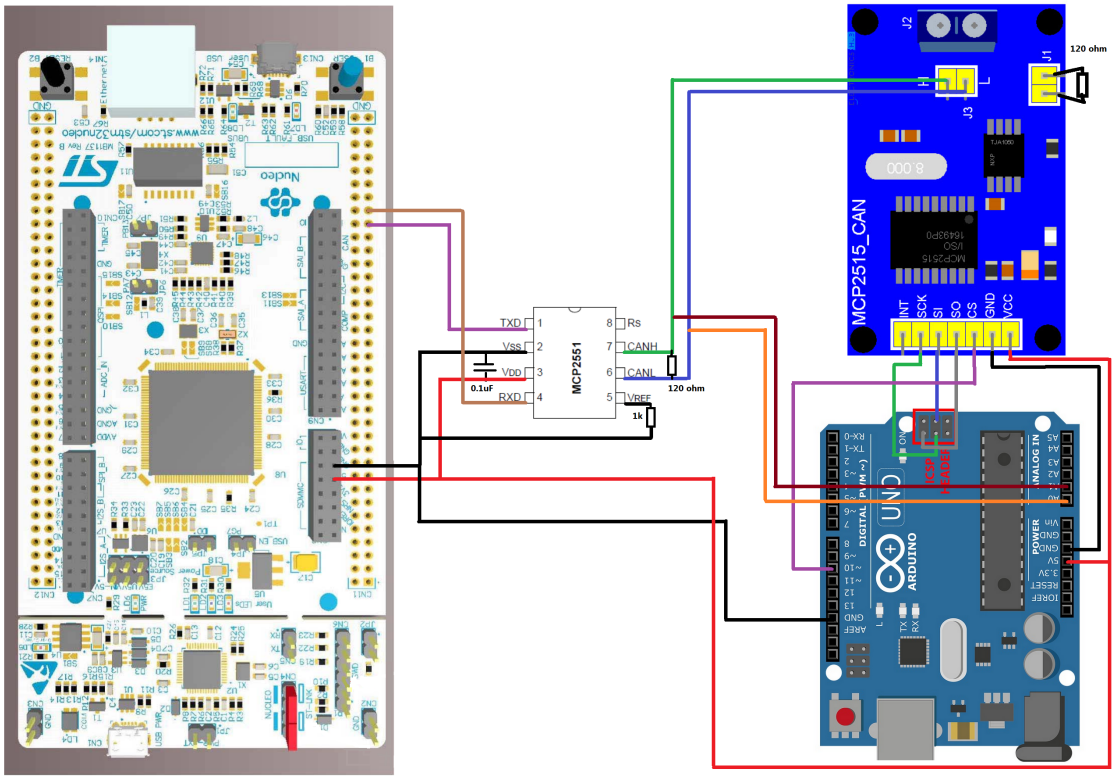

Figure B.1: CAN BUS debbugging with Arduino

To connect a board such as an Arduino board that does not have a CAN controller a MCP2515 CAN controller and transceiver can be used as shown in the diagram above. An Arduino can be useful while setting up the can bus since it can plot outputs from the analog pins to monitor the signals of the CAN bus.

In Micropython writing the function for setting up filters of CAN IDs is compulsory otherwise the node will not listen to the messages being sent.

Another important function in the Micropython for the CAN bus is the the CAN.init in which the bit timing is set up. All nodes should be having the same Nominal Bit Rate (NBR).

For programing in C here are some guidelines for the bit timing: `https://community.st.com/s/question/0D50X00009XkgOI/stm32f103vct6-can-bit-timing-settings`

When using the MCP2515 bit timing information:

`http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-pdf`

The bit timing is divided in time quantas In Micropython four variables can be controlled:

- The prescaler to divide the internal clock.
- SJW: 1-4 time quantas, adjusts the bit clock to maintain synchronization
- Bs1 the amount of time quanta defining the sampling point.
- Bs2 the amount of time quanta defining the transmitting point.

If the internal clock of the board is not known the function pyb.freq() can help to find it.

More information over the CAN bus in micropython can be found in:

`https://docs.micropython.org/en/latest/library/pyb.CAN.html`

Appendix C

Flashing micropython on microcontroller

Here are the steps to Flash Micropython on the Microcontroller of this project.

Step 1: `https://my.st.com/` get the: STSW-STM32080 software to be able to flash the board

Step 2: Connect from the board VCC and BOOT0 Like on Figure C.1

Step 3: Run DfuseDemo software

Step 4: Files to upload for flashing are available on: `https://micropython.org/download#other` upload the latest version of the file .dfu

Step 5: Click upgrade, verify. Micropython should be installed

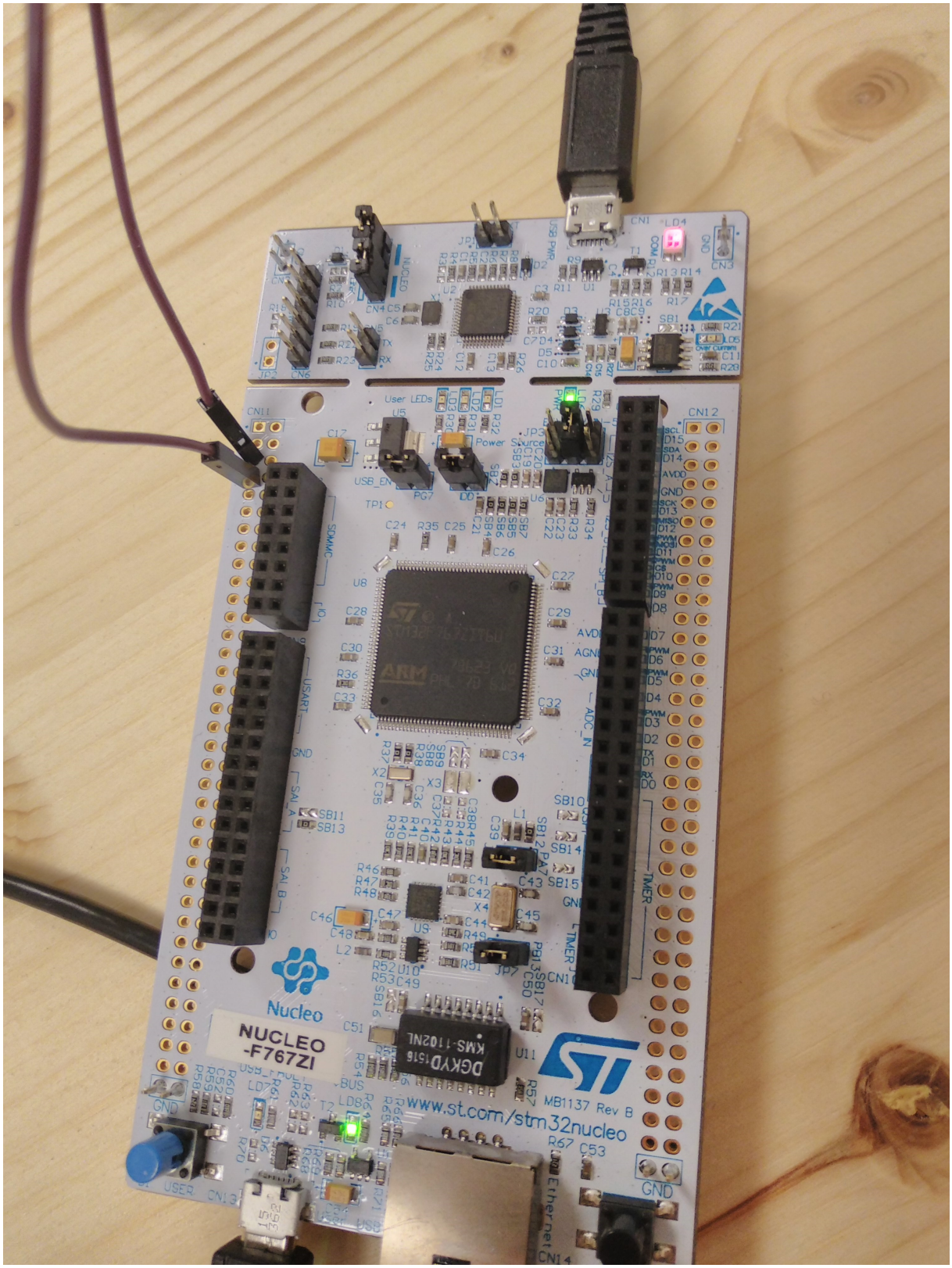

Figure C.1: Flashing board wiring

Appendix D

Orbit Perturbation Graphs

Here are the graphs used for the interpretations done in Chapter 2 on the effect of the different perturbations influencing the orbit of the satellite. In each case the altitude over time in kilometers is plotted as well as the RAAN in degrees for each perturbation: cube shaped, spheroid J2, 3rd body and radiation pressure.

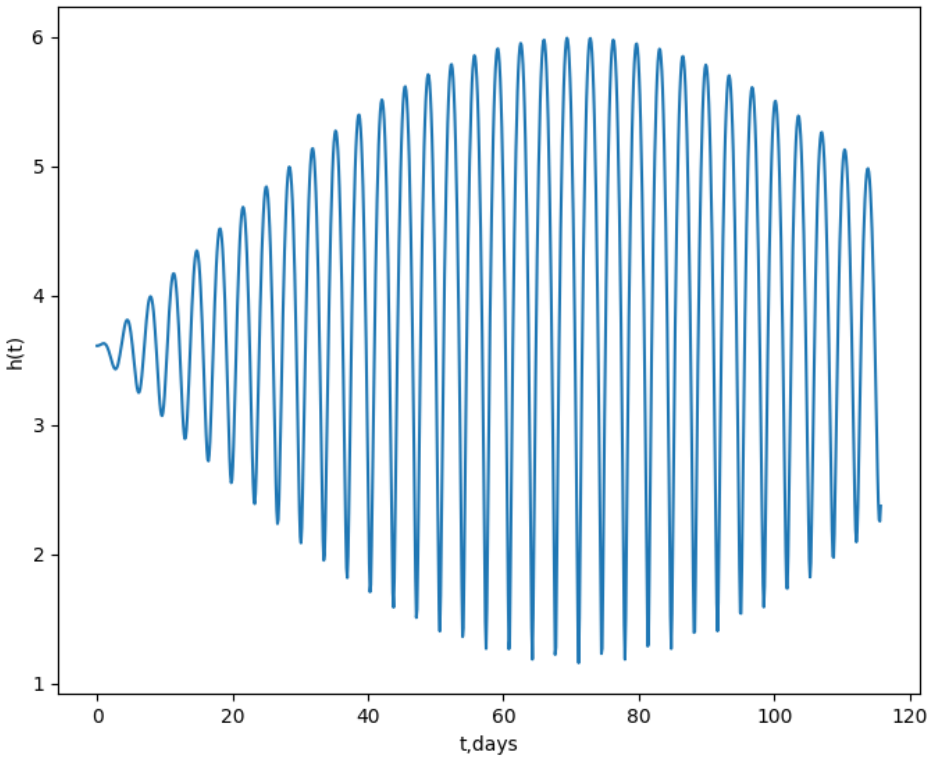

Figure D.1: Altitude cube perturbation

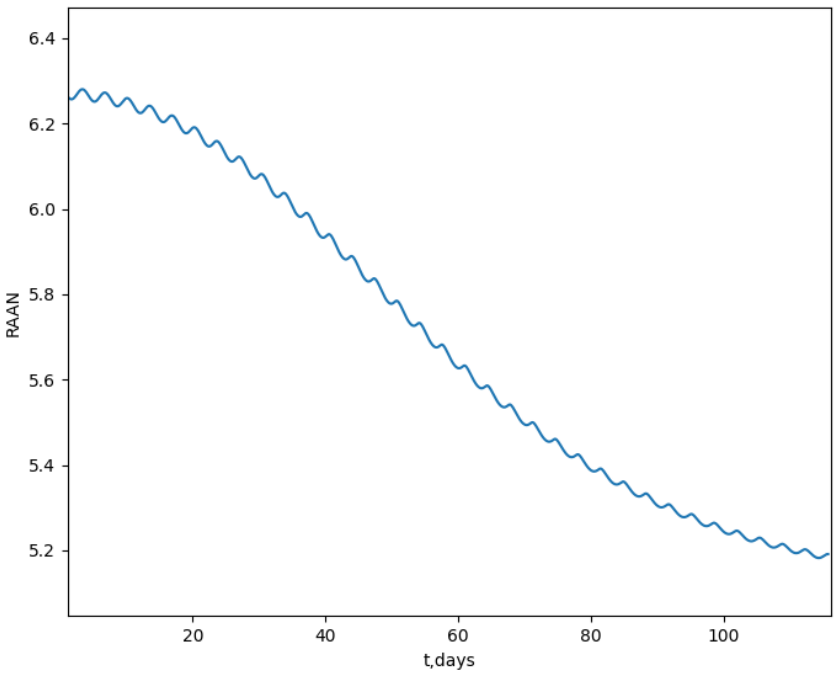

Figure D.2: RAAN cube perturbation

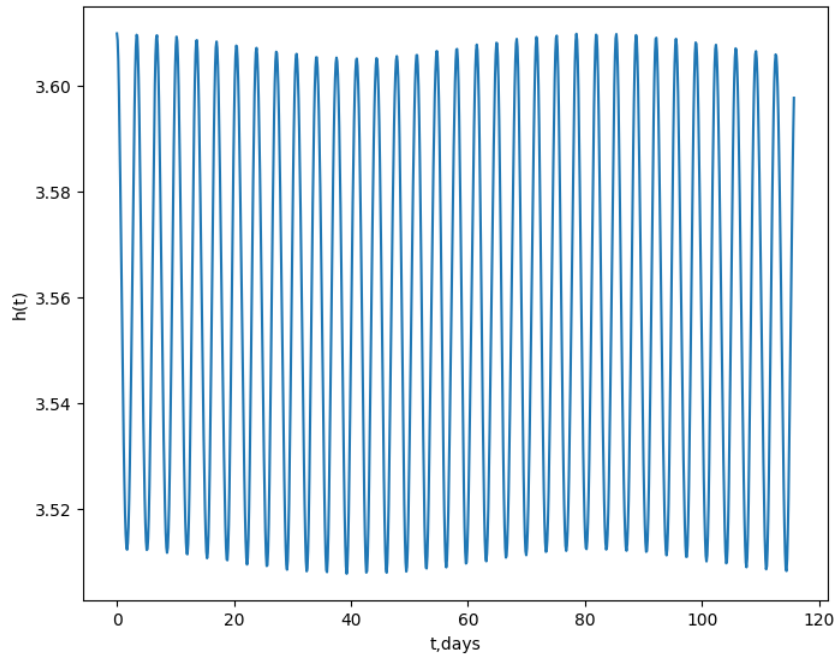

Figure D.3: Altitude spheroid perturbation

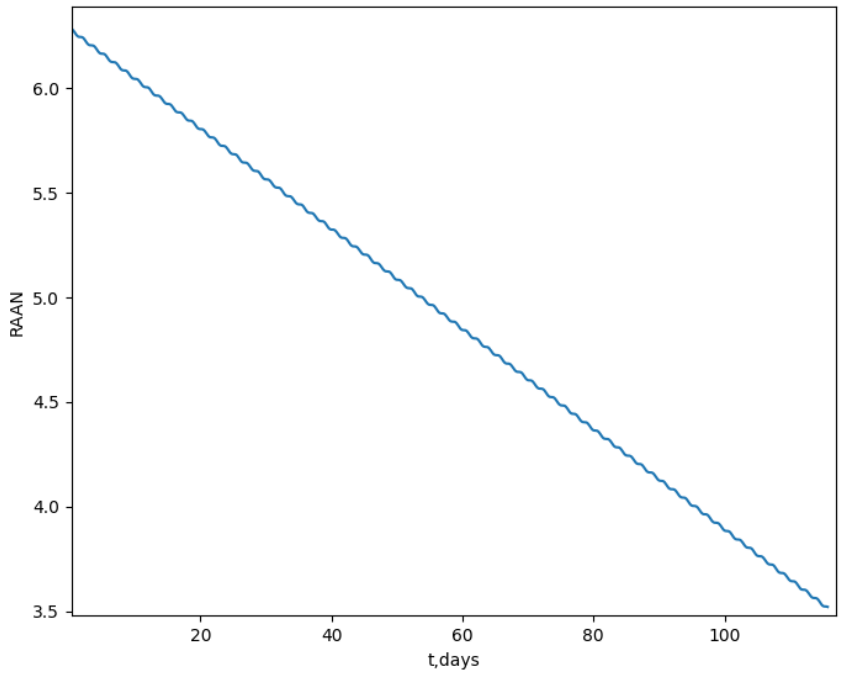

Figure D.4: RAAN spheroid perturbation

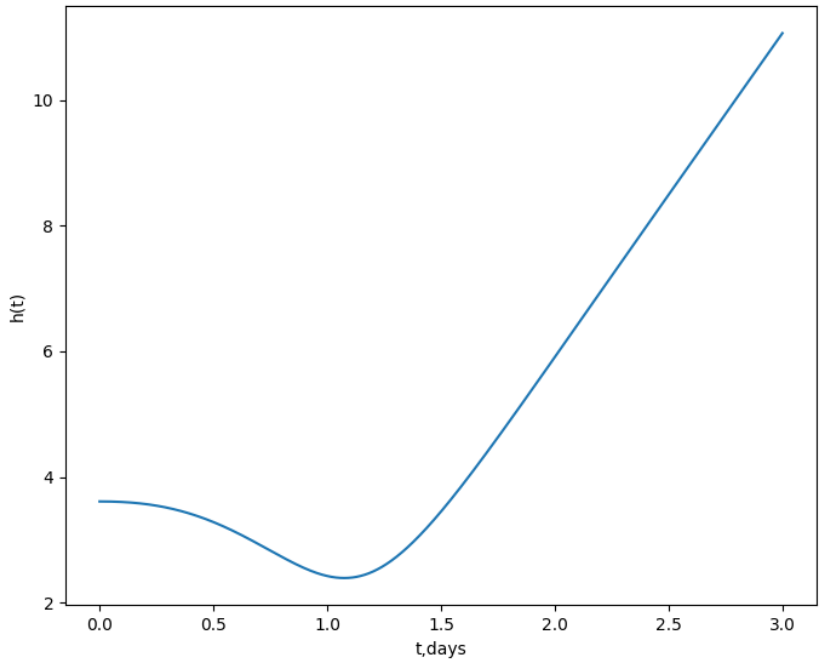

Figure D.5: Altitude 3rd body perturbation

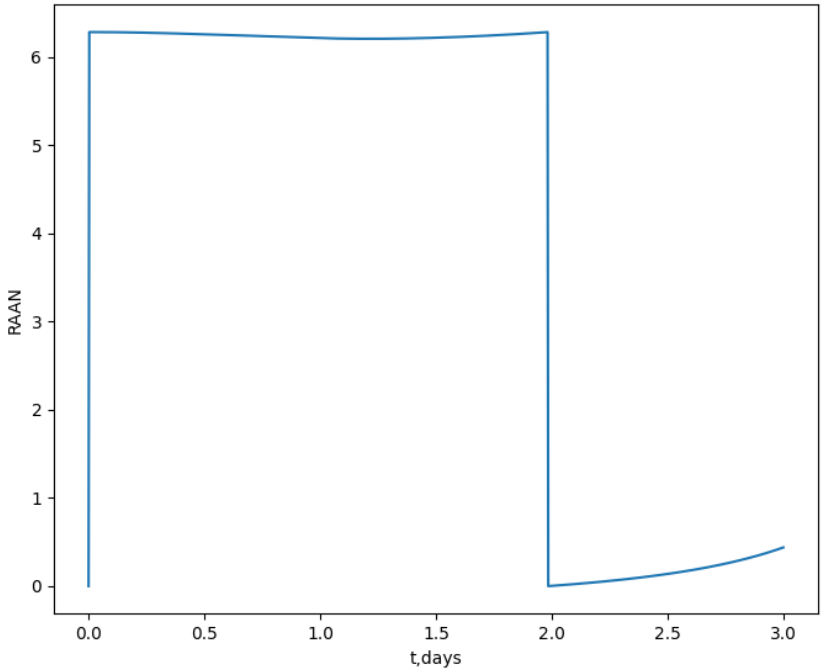

Figure D.6: RAAN 3rd body perturbation

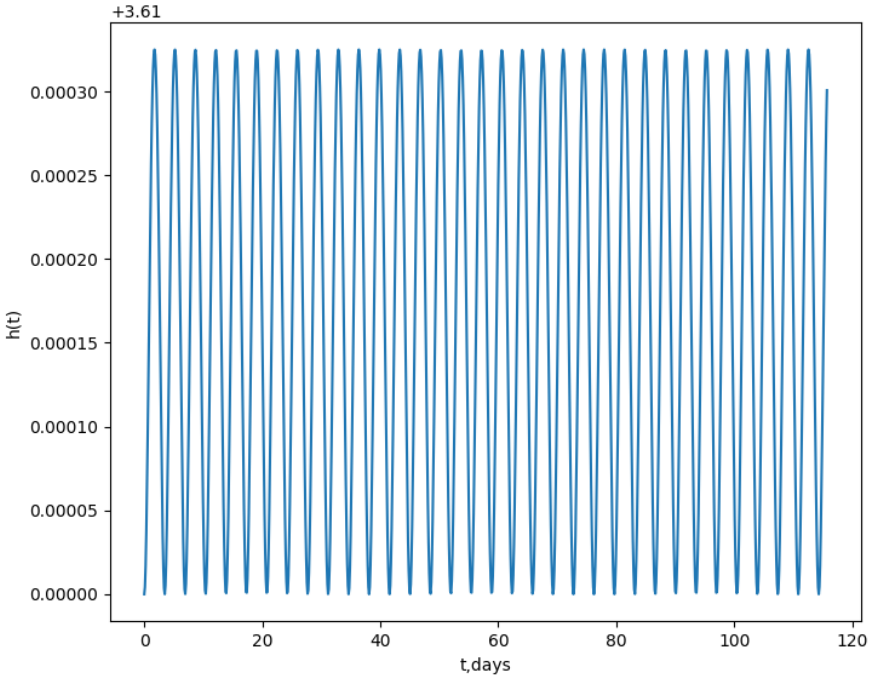

Figure D.7: Altitude radiation pressure

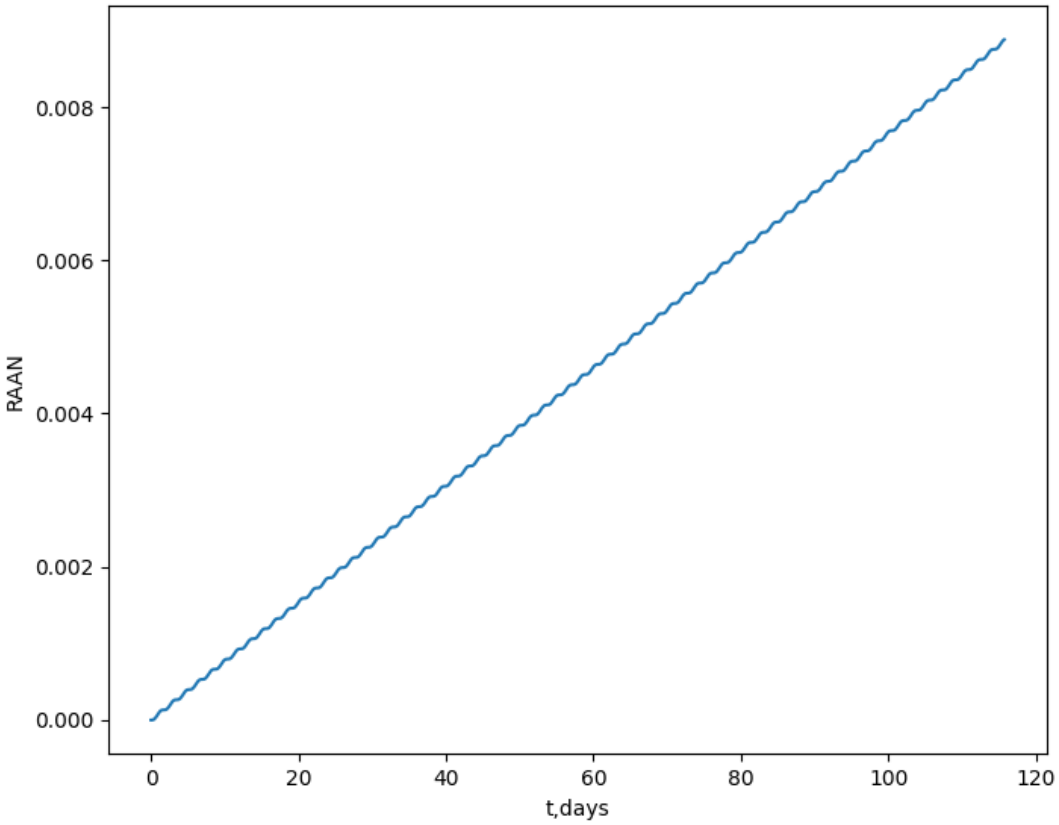

Figure D.8: RAAN radiation pressure

Bibliography

[1] J. R. Wertz, *Spacecraft Attitude Determination and Control*, Springer, The Netherlands, 1978.

[2] S. Lee A. Hutputanasin A. Toorian W. Lan R. Munakata J. Carnahan D. Pignatelli A. Mehrparvar, *CubeSat Design Specification*, Rev.13, California Polytechnic State University, 2015. [Online]. Available: `https://www.cubesat.org/resources/`.

[3] European Cooperation for Space Standardization, *Space Product Assurance Software Product Assurance*, Chap.6.2.2, ESTEC Noordwijk, The Netherlands, 2017.

[4] European Cooperation for space Standardization, *Space engineering System modelling and simulation*, Chap.4.2, ESTEC Noordwijk, The Netherlands, 2010.

[5] *HERA*, Safety and Security, European Space Agency. [Online]. Available: `https://www.esa.int/Safety_Security/Hera/Hera`.

[6] B. Dunbar T. Talbert, *Double Asteroid Redirection Test (DART) Mission*, National Aeronautics and Space Administration, 2020. [Online]. Available: `https://www.nasa.gov/planetarydefense/dart`.

[7] W. Voss, *A Comprehensible Guide to Controller Area Network*, Copperhill Technologies Corporation, Greenfield MA, 2008.

[8] A. E. Roy A. Hilger, *Orbital motion*, Adam Hilger Bristol and Philadelphia, 1988.

[9] P. R. Escobal, *Methods of Orbit Determination*, John Wiley and Sons Inc., 1965.

[10] C. Moler, *The Origins of MATLAB*, MathWorks., 2004. [Online]. Available: `https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab`.

[11] *Analytical Graphics (AGI) Inc. Systems tool kit (STK)*. [Online]. Available: `https://www.agi.com/products/stk/Accessed:2020-05-12`.

[12] J. Hernanz Gonzalez, T. Gateau, L. Senaneuch, T. Koudlansky, P. Labedan, *JSatOrb: ISAE-Supaero's open-source software tool for teaching classical orbital calculations*, ISAE-SUPAERO, Université de Toulouse, FRANCE, 2017. [Online]. Available: `https://indico.esa.int/event/224/papers/4080/files/211-paper.pdf`.

[13] *Orekit*, Space flight dynamics applications, Aug. 2019. [Online]. Available: `https://www.orekit.org/`.

[14] J. L. Cano, *Poliastro Astrodynamics in Python*, May 2020. [Online]. Available: `https://docs.poliastro.space/en/stable/`.

[15] *Astropy a Community Python Library for Astronomy*, Apr. 2020. [Online]. Available: `https://www.astropy.org/about`.

[16] J. Dasgupta, *Imparting Hands-on Industry 4.0 Education at Low Cost Using Open Source Tools and Python Eco-System*, Springer, Switzerland, Aug. 2019.

[17] *Numba Accelerate Python Functions*, Anaconda. [Online]. Available: `http://numba.pydata.org/`.

[18] E.V. Pitjeva, N.P. Pitjev, *Development of planetary ephemerides EPM and their applications*, Celestial Mechanics and Dynamical Astronomy volume 119, pages237–256, Springer, 2014.

[19] *Matlab STKOrbit.m*, Princeton Satellite Systems, Inc. [Online]. Available: `https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/65613/versions/1/previews/Aircraft%20Control%20Toolbox/Common/Graphics/STKOrbit.m/index.html/Accessed:2020-05-12`.

[20] J.M.A. Danby, *Fundamentals of Celestial Mechanics*, 2nd edition, Chap.11, Willmann-Bell, Inc., 2014.

[21] J.Lissauer, I. De Pater, *Fundamental Planetary Science: Physics, Chemistry and Habitability*, Chap.11, Cambridge: Cambridge University Press., 2013.

[22] F. C. F. Venditti, E. M. Rocco1, A. F. B. A. Prado, *Trajectory control around non-spherical bodies modelled by parallelepipeds*, Journal of Physics: Conference Series, Volume 465, XVI Brazilian Colloquium on Orbital Dynamics, São Paulo, Brazil, Nov. 2013.

[23] W. D. MacMillan, *The theory of the Potential*, Dover Publications New York, 1930.

[24] V. M. Gomes, F. C. F. Venditti, A. F. B. A. Prado, *Mapping Orbits regarding Perturbations due to the Gravitational Field of a Cube*, Hindawi Publishing Corporation, 2015.

[25] L. Rogers, *It's ONLY rocket science: An Introduction in Plain English, Astronomers' Universe*, Appendix A: Orbital Elements, Springer, New York, Jan. 2008.

[26] M. J. Nadoushana, M.Ghobadib, M.Shafaee, *Designing reliable detumbling mission for asteroid mining*, Faculty of Aerospace Engineering, K. N. Toosi University of Technology, Tehran, Iran, 2019.

[27] C. D. Brown, *Elements of spacecraft design*, American Institute of Aeronautics and Astronautics, Reston, VA, 2002.

[28] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 3rd Edition, Chap.12.10, Butterworth-Heinemann, 2013.

[29] ——, *Orbital Mechanics for Engineering Students*, 3rd Edition, Chap.12.9, Butterworth-Heinemann, 2013.

[30] A. S. Baron, *Study the Effect of Solar Radiation Pressure at Several Satellite Orbits*, Baghdad Science Journal, Nov. 2013.

[31] T. Shimizu, *System Design Report (SDR) for the Asteroid Prospection Explorer (APEX)*, Apr. 2017.

[32] *32-bit ARM Cortex-M0+ with 5V Support, CAN-FD, PTC and Advanced Analog*, Microchip Technology Inc., 2017. [Online]. Available: `https://www.infinity-electronic.hk/product/Micrel-Microchip-Technology_ATSAMC21J18A-MUT.aspx`.

[33] *STM32 Nucleo-144 boards*, STMicroelectronics, Apr. 2020. [Online]. Available: `https://www.st.com/en/evaluation-tools/nucleo-f767zi.html`.

[34] *MicroPython*. [Online]. Available: `https://micropython.org/Accessed:2020-05-12`.

[35] *SpaceCAN*, librecube.org, Sep. 2019. [Online]. Available: `https://wiki.librecube.org/index.php?title=SpaceCAN/`.

[36] Y. Khedkar, *STM32F032 CAN bus*, ST Community, May 2018. [Online]. Available: `https://community.st.com/s/question/0D50X00009XkWRUSA3/stm32f032-can-bus`.

[37] L. Chen, H. Lin, Z. Lu, J. Li and G. Ou, *High Orbital Spacecraft Fast Positioning Algorithm Assisted by Inter-Satellite Links*, 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2019.

[38] H. D. Curtis, *Orbital Mechanics for Engineering Students*, Elsevier. p. 264, 2005.

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **de Graaf Niels** Personal ID number: **489933**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Simulation of AOCS for APEX CubeSat

Master's thesis title in Czech:

Simulace AOCS pro APEX CubeSat

Guidelines:

The purpose of this assignment is to create an module of Software Verfication Facility and demonstrate the relevance of using opensource software and standardized communication using the APEX CubeSat that will orbit the asteroid Didymos as mission basis.
1. The task will be to create a model of the dynamics of the CubeSat, its disturbance environment to plot its trajectories using python libraries developed for orbital dynamics.
2. Attitude and Orbital control will be programmed on a microcontroller to calculate the new trajectories for the simulation.
3. An Interface will be made to send simulated data using the standard CubeSat Space Protocol.

Bibliography / sources:

[1] Blanke, M., & Larsen, M. B. - Satellite Dynamics and Control in a Quaternion Formulation (2nd edition) - Technical University of Denmark, Department of Electrical Engineering, 2010.
[2] Vendittil F. C. F. , Rocco E. M. , Prado A. F. B. A. - Trajectory control around non-spherical bodiesmodelled by parallelepipeds - National Institute for Space Research, INPE, Sao Jose dos Campos, SP-Brazil, 2013

Name and workplace of master's thesis supervisor:

Ing. Daniel Štefl, Ph.D:, Space Systems Czech s.r.o., nám. Winstona Churcila 1800/2, Praha 130 00

Name and workplace of second master's thesis supervisor or consultant:

Ing. Martin Hlinovský, Ph.D., Department of Control Engineering, FEE

Date of master's thesis assignment: **13.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until:
by the end of winter semester 2021/2022

Ing. Daniel Štefl, Ph.D:	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student's signature