CRANFIELD UNIVERSITY

GUILLEM KHAÏRY

ROTOR DESIGN FOR SATELLITE TRACKING

School of Aerospace, Transport and Manufacturing

Individual Research Project report

MSc Astronautics and Space Engineering

Academic Year: 2020 - 2021

Supervisor: Dr Stephen Hobbs

Associate Supervisor: Mr Sébastien Hesse

September 2021

CRANFIELD UNIVERSITY

School of Aerospace Transport and Manufacturing
Individual Research Project Report

MSc Astronautics and Space Engineering

Academic Year 2020 - 2021

GUILLEM KHAÏRY

ROTOR DESIGN FOR SATELLITE TRACKING

Supervisor: Dr Stephen Hobbs
Associate Supervisor: Mr Sébastien Hesse
September 2021

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc Astronautics and Space Engineering

ABSTRACT

This report is a part of a student project created by the Spatial Centre of Montpellier University (CSUM) and is realized in the framework of the completion of the Cranfield University's MSc Astronautics and Space Engineering Course.

The aim of this report is to present the different tasks done in the completion of designing a new satellite tracking rotor for communication in UHF and S-band, to replace the current one on the CSUM. These tasks are grouped between: System Engineering, Mechanical Engineering, Power and Embedded Systems and Software.

The System Engineering focuses on establishing key requirement for the project and its Work Packages. The tasks in Mechanical Engineering includes the study of the efforts applied on the imagined concept's structure of the system, the compliance check and tests of motor proposed by the CSUM teams. Regarding the Power and Embedded Systems, the configuration for the electronics that will power and control the rotor is established and tested. Finally, the Software section presents the set of functions coded and used to produce the data necessary for motor simulation and satellite tracking.

The overall project results are promising and give a strong basis to work on for several domains of the project and end most of them for finally finishing the mission started, with an additional 6 month time.

Keywords:

Software, embedded systems, mechanical analysis, motor testing

ACKNOWLEDGEMENTS

As this project gathers various engineering fields from mechanical design to communication hardware and software, the author would like to thank the CSUM and Van Allen Foundation and its sponsors for giving the opportunity of such an insightful project.

Acknowledgements must also be given to the CSUM Engineers Christophe Prud'homme, Romain Briand and Amaury Knockaert for sharing their experience and help me orientate on the project. Special thanks to my supervisor Sébastien Hesse, who despite his growing responsibilities, always found the time for regular meeting and updating the project's inputs.

Finally, thanks to my fellow students present at the CSUM: Kenza Courtin, Axel Guiraud, Mathieu Roumieux, Osman Ertas, Sebastian Otto, to make me discover new methodologies and approaches of the various aspects of engineering.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

1U	1-Unit (CubeSat)
3U	3-Unit (CubeSat)
AC	Alternate Current
CAD	Computer Aided Design
CNES	(French) National Centre of Spatial Studies
CSUM	(French) Spatial University Centre of Montpellier
DC	Direct Current
FR	Functional Requirements
IDE	Integrated Development Environment
IRP	Individual Research Project
OR	Operational Requirements
MO	Mission Objective
RF	Radio Frequency
RPi	Raspberry Pi
TLE	Two-Line Elements (or ‘Two-Line Element Method’)
UHF	Ultra-High Frequency
WP	Work Package

I. Introduction

In 2013 a weather monitoring project of the Mediterranean coast begun between Météo France, the CNES (French Space Agency) and the CSUM. This project consists in a 3U CubeSat designed and assembled by the CSUM under the name “ROBUSTA 3A – Méditerranée”. Once launched, at the end of 2021, the aim will be to make a cooperative weather monitoring of storms and high rain in the region of Montpellier, and more generally in Mediterranean Sea; placing transmission and reception segments on boats, designed by the CSUM.

I.1 Context

I.1.1 The CSUM

The Spatial Centre of the University of Montpellier is the leader in the development of student nanosatellites in France. Using its own technology, the CSUM develops, tests, and produces 1U and 3U satellites. The projects carried by the centre are both of regional and European scope.

The centre possesses a ground segment, including a transceiver radio station and 3 sets of antennas. The CSUM also possesses AIT facilities including workshops and an ISO-8 clean room and a concurrent engineering centre as well as a proximity with all space actors of the region.

I.1.2 Current system configuration

One of the CSUM’s current set of antennas uses a 1990’s rotor, the YAESU G-5500. This rotor provides a tracking in both azimuth and elevation axis, with respectively 180° reached in 65 seconds in elevation and 360° in 60 seconds in azimuth. The axis that carries the antenna is

held by going all the way through the rotor. This system, while being robust and reliable, still holds critical flaws for the transmission quality.

SPECIFICATIONS

Voltage requirement:	110-120 or 200-240 VAC
Motor voltage:	22 VDC
Rotation time (approx.):	Elevation (180°): 65 sec. ± 20% Azimuth (360°): 60 sec. ± 20%
Maximum continuous operation:	3 minutes
Rotation torque:	Elevation: 12 kg-m (88 ft-lbs) Azimuth: 6 kg-m (44 ft-lbs)
Braking torque:	Elevation: 40 kg-m (289 ft-lbs) Azimuth: 40 kg-m (289 ft-lbs)
Vertical load:	200 kg (440 lbs)
Pointing accuracy:	±4 percent
Wind surface area:	1 m^2
Mast diameter:	38-63 mm (1-1/2 to 2-1/2 inches)
Boom diameter:	32-43mm (1-1/4 to 1-5/8 inches)
Weight (approx.):	Rotators: 8 kg (17.7 lbs) Controller: 3 kg (6.6 lbs)

Figure I-1 : Current rotor specifications

The first problem lies in the drawback of the very nature of the motor. Being a step motor, every movement of the axis generates a tumbling on the antennas. Such phenomenon produces loss of accuracy when pointing at the satellite and thus a less accurate transmission. Moreover, the long-term drawback of the tumbling is the constant de-alignment of the antennas on the mast due to deformation. Aluminium profile composing the main structure of the antennas, it can deform due to efforts induced by the wind, gravity, and the rotor's impulsion; this jeopardizes even more the quality of transmission. This effect is even more amplified as a trackable orbit occurs in average every 1,5 hours and the position is actualised every 33 to 34 seconds.

Figure I-2 : Tumbling-induced un-parallelism of the antennas

The second “undesirable” parameter lies in the current rotor’s structure. As mentioned above, both antennas are linked to a single mast, that goes through the rotor. This implies that both antennas are to move together every time, and that in case of one being blocked, the other would be as well. This could be avoided using two separated transmission system for each antenna.

Production of this rotor stopped, leaving the solution of replacement or repair in case of failure hard to achieve with reduced time/cost objectives. To solve this situation, the CSUM, which currently keeps a replacement model, aims to create its own tracking system, to control the technology, accessibility, and cost of its installation. The system also needs a change of its antennas to adapt for the next satellite to come into orbit while keeping tracks and communication link for the current orbiting satellites. The first draft of the system of the produced system for the CSUM will also serve as prototype for the boat tests by adding a gyroscopic platform.

The software that produces the data used for the tracking is Gpredict, an opensource software that uses Two Line Element to predict satellites orbits. These data are then interpreted by the rotor program to follow the satellite’s trajectory.

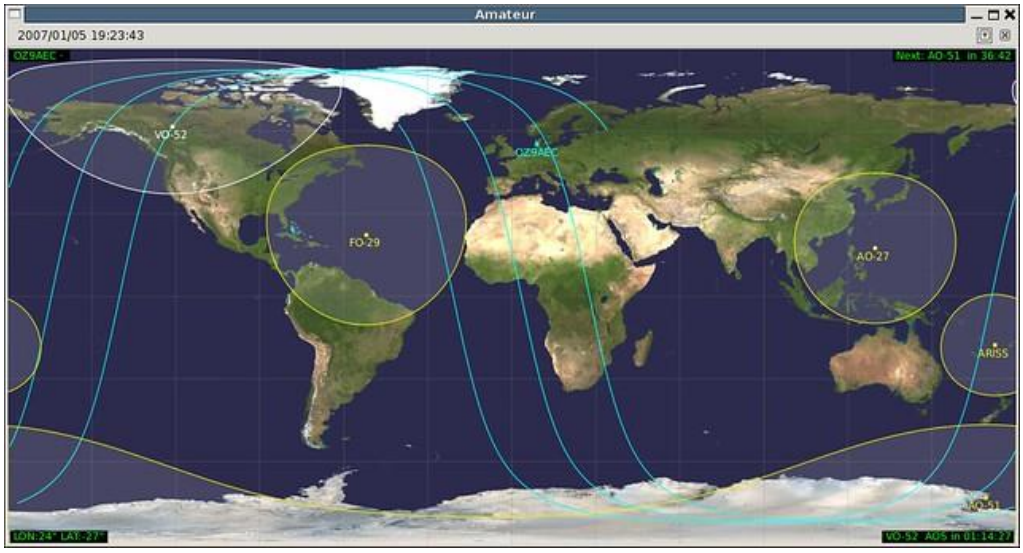


Figure I-3 : Display of the satellite trajectory using Gpredict

I.1.3 ROBUSTA-3A and ROBUSTA-1B

Since ROBUSTA-3A is set to reach orbit in late 2021 or beginning 2022, we cannot use it as an experimental means of testing the future system. The orbit of ROBUSTA-3A and ROBUSTA-1B are similar. Thus, the exceptionally long mission lifetime of ROBUSTA-1B allows a consequent number of orbits that can be analysed. To parameter the yet-to-be first prototype of the rotor in both mechanical and software configuration, this project uses the ROBUSTA-1B passing orbits data.

Figure I-4 : ROBUSTA-1B before launching an its current associated TLE-produced file

I.2 Project objectives

I.2.1 Mission Statement

The mission given by the CSUM is to find a cost-effective alternative to the current tracking system. The mission is guided by several requirements given by the supervising engineer and relayed through interns. It consists in the development and design of a prototype for the new system. The author of this report is the second student to pursue this mission.

I.2.2 Mission Objective (MO)

The objectives of the mission are plural. First, the final system shall be mechanically functional, i.e., being able to follow each ROBUSTA-1B's pass in both elevation and azimuth to guarantee the maximal exposition time for communication (MO-01).

The system shall carry 3-meters long antennas (MO-02), with the most reduced tumbling possible. The design of the system shall not impact the good transmission of the data flow, i.e., interferences, etc.

In addition, the system shall be remotely controlled from the ground segment using a microcontroller or microcomputer (MO-03). The GS operator can then launch the satellite tracking from the GS (M0-04) ... If possible, the whole system should be fully automatic (M0-04*).

Additionally, the system shall use a 12-24 Volts DC input from a 220/240 AC source and be placed on the roof of the CSUM (MO-05). Finally, an adaptation for boats using a gyroscopic platform is considered for participative weather monitoring.

I.3 Project Management

I.3.1 Initial state of the project

The author is the second person to take on this mission. Before the beginning of the project, the following tasks have been achieved by the previous person in charge, Ms COURTIN:

- Design and partial production of a 3D-printed model of the rotor possible configuration using FUSION 360 and FlashPrint
- Estimation of the average passing time of the orbiting satellite ROBUSTA-1B
- Selection of a driver (L293D from T.I.) and possible microcontroller for the system (STM32 NUCLEO)
- Study of the CAN bus for the transmission part of the system

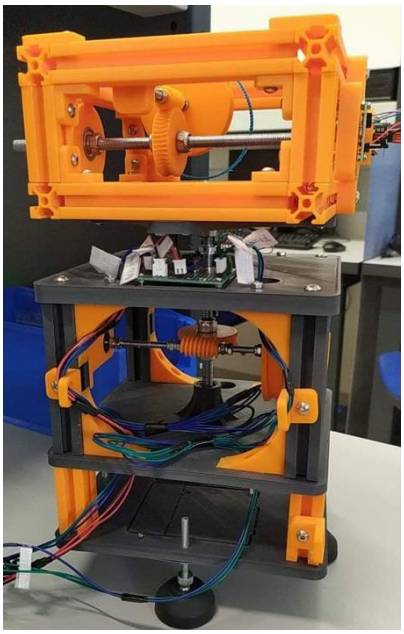


Figure I-5 : 3D-printed model, produced before and completed during the project

The three first advancements will be used in the scope of this project at various level.

First, the 3D-printed model produced, once added support for motors, was fully used as a test model for the project's various WP.

Regarding the two other data produced, they were used as comparative results for the project's development.

I.3.2 Project Breakdown

This project will be decomposed into four Work Packages (WP). The first one, System Engineering (WP01), in section **II**, consists in a translation of the mission objectives into various sets of requirements, budget analysis and project deliverables.

The second WP, Mechanical (WP02), in section **III**, regroups top-level study of the future system. This WP also includes suggested design of the mechanical transmission system for the rotor (e.g.: cogs, worm gears, etc).

The third WP, the Electrical WP (WP03), in section **IV**, emphasizes on the embedded systems and power distribution. This includes the choice of microcontroller and its additional parts (e.g.: drivers, etc), electronic and power configuration of the system.

Finally, Software and Automation (WP04), in section **V**, is focusing on the program that will be used by the new system to follow the orbiting satellite. This WP also covers the user side of the code and the transition between software and hardware (digital to analogic).

I.3.3 Project Organisation

While this project being solely handled by author of this report, it must still be constantly updated and explained to the CSUM's engineers. To keep track of the project's advancement, a diary of the daily operation is kept on an accessible drive. Weekly meetings are organized to brief the supervisor on the project updates and discuss the next steps and/or new requirements. In addition, Trello is used to fix the monthly objectives and their updates (which depends on the changes of constrains/requirements).

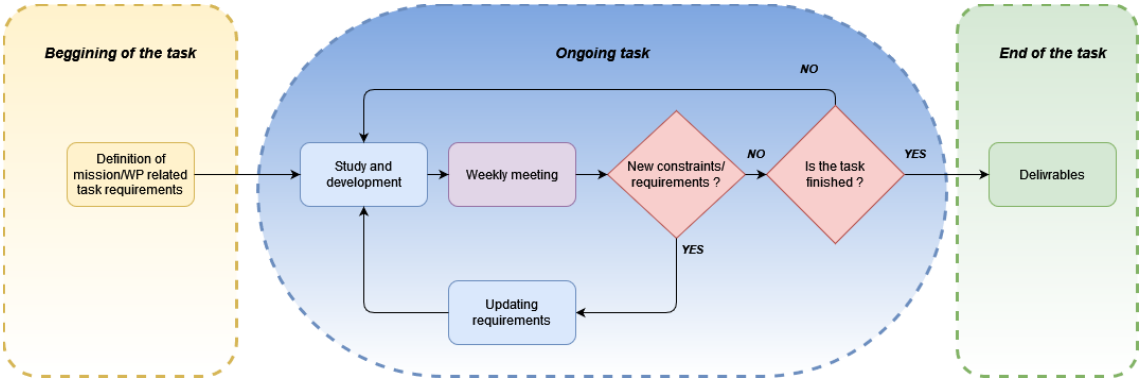


Figure I-6 : Task organisation scheme

Since the mission is set to last more than one intern's project, it is also critical to allow the maximum information transmission of the mission advancement once the project is done for the next intern. Therefore, tests, computing, etc, are contained in one single Excel document containing spreadsheets ordered chronologically. Finally, several deliverables have been made to fasten the handling of this project by the CSUM's teams: specifications for the system, update diary for programs, presentations, etc.

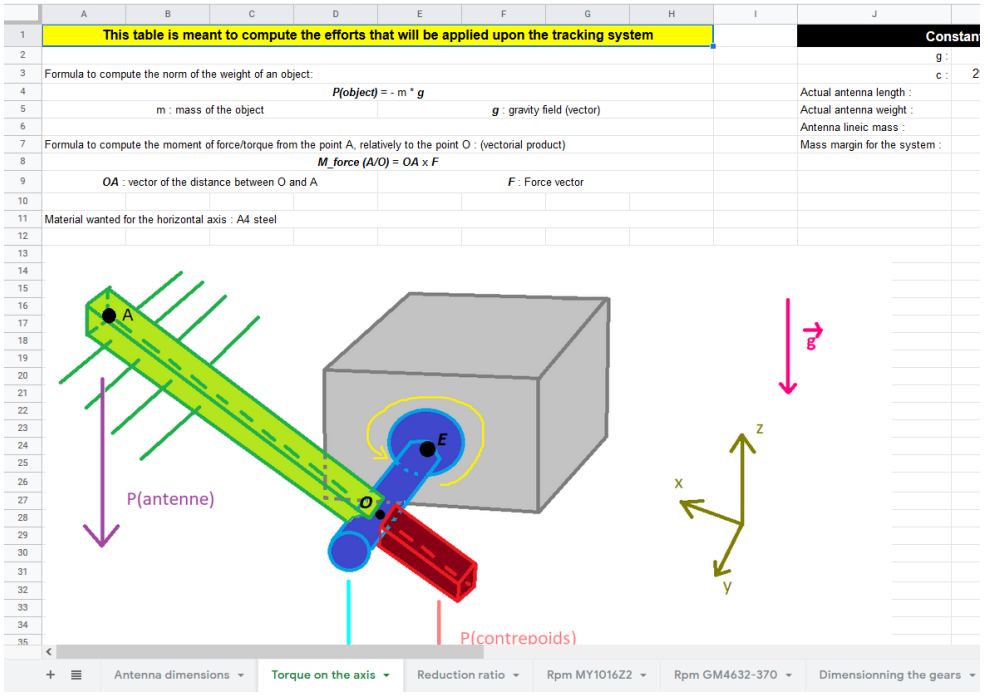


Figure I-7 : Example of deliverable - Computation spreadsheet

II. Mission requirements (WP01)

The objective of this section is to translate the objectives of the mission in clear and solvable requirements. The obtained requirements will provide necessary guidelines to begin each WP's study and are susceptible to evolve during the project.

II.1 Requirements management

II.1.1 Objectives to requirements

Starting from the need emitted by the CSUM at the beginning of the project, the methodology to establish the top-level requirements is the following:

1- List the key Mission Objectives (MO)
2- Translate them into Functional Requirements (FR)
3- Decompose the FR into several Operational Requirements (OR)
4- Regroup the found OR into their WP

The obtained OR will be used to shape the tasks necessary for the WP's development.

II.1.2 Derived requirements

During the advancement of the project, some requirements might change or evolve due to the progress made. These changes will be rated as additional requirement, keeping the same number as the original one but having a " * " on the side to mark this update.

The interest of such a procedure is to mark the progress made but also to keep a track on the original requirement in case of a need to step back on the project. As an example, in section V.3.2, this situation has occurred as the core program, in its test version, was too complex and needed to have feature removed for its user version.

II.2 Top-level requirements

II.2.1 Constraints

Prior the requirement's definition, this mission is subject to multiple and nature-varying constraints. This project aiming to be opensource and accessible to the most, the budget is obviously the first constraint and a key factor to emphasize on the project. Therefore, it should be kept between 1000 and 1500€[1].

On the operating side of the system, the length of the carried antennas must be kept the same as the current ones: around 3 metres. Moreover, this time, the antenna cables must be carried inside the structure to avoid them getting caught by the wind, birds, or getting blocked around the antenna itself.

Considering the amplitude of the movement done by the rotor during passing time, (more than 150° for some azimuth orbit) this implies that a minimal cable length of approximately 2 complete rotations. This way, all orbits at range would be possible to track. In addition, the single-mast configuration of the current model shall be replaced by two independent masts to prevent a jammed antenna to block the other of damage the mechanical system of the rotor.[2]

Finally, the internal structure's components (i.e., gears, motors, and embedded electronics) should be relatively accessible to allow onboard maintenance by the personnel without a qualified staff member. Onboard maintenance includes changing gear parts of the transmission system and change the connectors and a part of the electronics when possible. A solution for this problem would be to use the required modularity of the

[1] This constraint remains a guideline for the production and does not apply necessarily for the prototype production trials.
[2] This constraint implies the use of 2 separated motors or a differential

structural box to extract the electronic block for testing and maintenance. Repairs on the motors and the complex electronics are not considered as onboard maintenance.

II.2.2 Functional requirements

Once the MO(s) are stated and the constraints emitted, it is possible to establish the functional requirements for the system. Using the process detailed in section II, the following functional requirements have been obtained from the MO(s) and the constraints.

FUNCTIONAL REQUIREMENTS	
FR-01	The system should cost between 1000 and 1500€
FR-02	The system shall follow each passing of Robusta 1B (one passing for every 1.5 hour in average)
FR-03	The system shall carry two antennas, respectively S-band and UHF
FR-04	The system shall be fixed on the CSUM's roof
FR-05	The system shall be controlled remotely via the GS located inside the CSUM
FR-06	The system shall withstand constant exposure to the Southern France climate: drought, strong wind, and punctual heavy rain

Table II-1 : Functional requirements for the future system

II.2.3 Operational requirements

The functional requirements being determined, the operational requirements naturally follow. These will be split into each WP to add more readability of the project's structure.

Mechanical (WP02)	
OR-01-M	The system rotation shall be able to rotate of 720° (-360° to 360°)
OR-02-M	The system shall carry antennas that have a wave ratio of 1:1
OR-02-M*	The rotor shall carry two 3-meter-long, weighting the same mass and held at 1 meter of the rotor's structure
OR-03-M	The rotor shall use two different axes, one for each antenna (i.e., no transversal axis linking both antenna)
OR-04-M	The rotor shall match the satellite's orbit speed in both azimuth and elevation: respectively 4,480 and 14,63 deg/min for a total trajectory duration of 9,15 seconds
OR-05-M	The structure and the motor shall not interfere in the antenna radiation scheme (i.e., transmission and emission)
OR-06-M	The structure of the system shall give access to the embedded system, the mechanical transmission system, and the motors without any dismounting (e.g., using a hatch or a cover/lid)
OR-06-M*	The opening of the structure shall not prevent the rotor from working (in case of maintenance tests)

Table II-2 : Operational Requirements for WP02

Power and Embedded Systems (WP03)	
OR-07-PES	The system shall use 220/240 V AC input
OR-08-PES	The system shall run with a 12/24 V DC input
OR-09-PES	The system shall use have a controller for the motor
OR-10-PES	The system antennas shall act respectively as transmitter and receiver of UHF and S-band
OR-10-PES*	The system antennas shall both act as transmitter and receiver of respectively UHF and S-band
OR-11-PES	The embedded system and power admission shall not interfere in the antenna radiation scheme (i.e., transmission and emission)
OR-12-PES	The link between the GS and the embedded system shall not interfere in the antenna's emission and reception of the signal

Table II-3 : Operational Requirements for WP03

Software and Automation (WP04)	
OR-13-SA	The system shall provide a feedback precision of 0.5°
OR-14-SA	The program shall be run from the GS by an operator
OR-14-SA*	The program should run automatically after the first activation
OR-15-SA	The program shall be compatible with Raspberry Pi 3 OS
OR-16-SA	The program shall read into the files 'ROBUSTA-1B-passes' and extract the data of the wanted orbit

Table II-4 : Operational requirements for WP04

III. Structure and mechanical transmission system (WP02)

Mechanical is the WP that regroups the study of the effort on the structure of the rotor and the design of its transmission system. Due to the various need of the project in multiple WP, the mechanical study will be limited to a theoretical study for the structure and mechanical transmission system. The mechanical aspect is obviously crucial, but several reasons make more complex and foremostly longer to treat than the other WP due to ordering, machining time, etc. For this specific reason the time given (five months) for a general IRP that covers the study and development of this system is not enough.

Nevertheless, the elements brought by the following study and tests are compelling to proceed in the other WP. The report then presents the mechanical study of the system and suggests a possible design for the to-be-developed prototype while focusing on the already existing, 3D-printed, model.

III.1 General study of the system

III.1.1 Determining antenna minimal length

Following the requirements established in 0, the antenna length shall at least match the size of the biggest emitted wavelength. Therefore, the antenna is (at minimum) a 1:1-scaled antenna. The frequency bands used by the antenna which the tracking system carries are UHF and S-band; respectively 430 – 440 MHz and 2 - 4 GHz. UHF being the lowest between those two bands, it provides the minimal length for a 1:1 antenna.

A simplified, first approach aims to find the needed length is given by the characterization of the wavelength in a medium:

$$\lambda = \frac{c}{f} \tag{3.01}$$

With:

$$c = celerity\ of\ the\ wave$$
$$\lambda = wavelength\ of\ the\ wave$$
$$f = frequency\ of\ the\ wave\ (cst\ between\ media)$$

This notation fits the void medium but adapting this formula into standard atmosphere only requires adding the air coefficient:

$$\lambda_{air} = \frac{c_{air}}{f_{air}} = \frac{c}{n_{air} * f_{air}} \tag{3.02}$$

The values of n_{air} varying on the temperature, it has been assumed the average temperature of the region of Montpellier would be 20° through the year (which is close to the weather reality of the area).

$$n_{air}\ at\ 20° = 1{,}000272$$

$$c_{air} = 299710936{,}6\ m.s^{-1}$$

Using the lowest frequency included in the lowest band (for maximal antenna size), we obtain for the system:

$$\lambda_{air} \approx 70\ cm = 0{,}70\ m$$

The antenna being a 1:1 scale at minimum, λ_{air} also corresponds to the minimal antenna length. Therefore:

Minimal length antenna: $L_{min} = 70\ cm$

As stated in 0, an update on the antennas' size has been emitted: the antennas carried by the system shall measure 3 meters (***Table II-2)***. The minimal size determined by the previous computing confirms that,

measuring 3 meters, the minimal size requirement for a 1:1 antenna is respected as well as the updated requirement.

III.1.2 Simplified constraints on the system

Once the antenna size is determined, it is possible to produce a simplified representation of the upper system (i.e., the elevation part) to analyse the constraints withstood during the procedure.

First, several assumptions must be made to simplify the study:

1. The weather effect will be neglected (e.g.: wind, rain, etc)

2. The erosion and fatigue effects on the system will be neglected

3. The internal efforts will be neglected (friction/rubbing, heat losses)

4. The antenna is assumed to be homogenous, not deformable aluminium alloy[3]

These approximations settled; we need to also consider **OR-03-M** made in ***Table II-2***.The consequence of this requirement is the addition of a torque on the elevation axis. Thus, the system can be considered ***symmetric***, i.e., only studying the effort on one side will provide the information for the other side as well.

The important information to determine are then both weight (***W***) and induced torque (***T***) of the set {Mast + Antenna}. Using the following formulas:

$$\boldsymbol{W} = \vec{W} = m * \vec{g} \tag{3.03}$$

$$\boldsymbol{T} = \overrightarrow{T_{OA}} = \overrightarrow{OA} \times \vec{W} = m * \overrightarrow{OA} \times \vec{g} \tag{3.04}$$

[3] The aluminium is a classic material in antenna manufacturing because of its radiative properties

With:

$$m : mass\ of\ the\ analysed\ system$$
$$\vec{g} : gravity\ constant\ (9.81\ m.s^{-2})$$
$$\overrightarrow{OA} : distance\ between\ system's\ location\ A\ and\ application\ point\ of\ the\ torque\ (O)$$

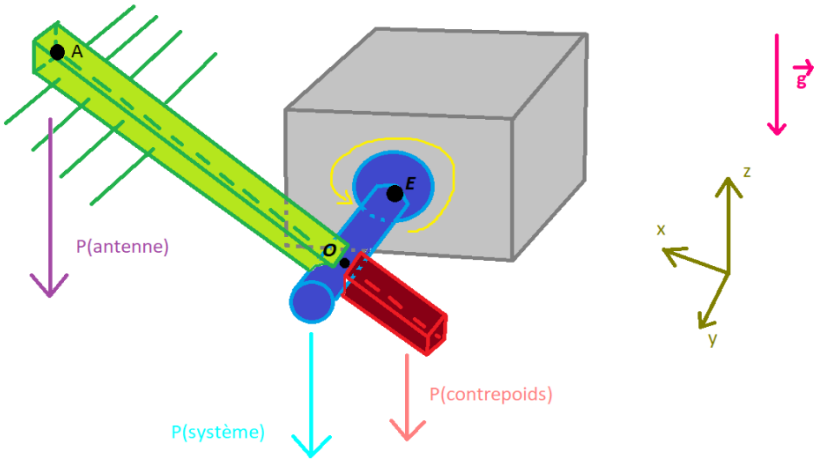


Figure III-1 : Efforts applied on the mast [4]

The maximum value of both ***W*** and ***T*** is obtained when ***g*** is orthogonal to the ground plane (and therefore ***OA***). This configuration will be the reference when determining the torque ***T***. According to ***4.***, the antenna is in homogenous aluminium alloy and measures 3 meters. The antenna is assumed to be a square beam with a section of 5*5 centimetre-square of 0.5 cm thickness. Therefore, the volumic mass of the aluminium being 2.7g/cm^3 we obtain a mass of 3,8475 kg.

Using (3.03) and (3.04), the effort applied by the antenna on the mast are:

$$|\boldsymbol{W_{antenna}}| = 3{,}8475 * 9{,}81 \approx 38\ Newton$$

[4] Translation: “antenne” = “antenna”, “système“= “system”, “contrepoids” = ”counterweight”

$$|\boldsymbol{T}_{\boldsymbol{antenna}}| = 3{,}8475 * 3 * 9{,}81 = 113{,}232 \leq 114\ Nm$$

Once the Torque induced by the system's weight on O has been determined, the value of the counterweight on the other side of the mast can be chosen using (3.04) again**Erreur ! Source du renvoi introuvable.**. The counterweight will depend on the arm size chosen to act against the antenna part of the mast. Since the mass m and the length |OA| acts as proportional inverses in (3.04), reducing the mass of the counterweight.

Example of counterweight configuration		
Torque to match	Length (m)	Mass (kg)
114	3	3,8475
	1	11,5425
	0,25	46,17

Table III-1 : Possible configuration for the counterweight

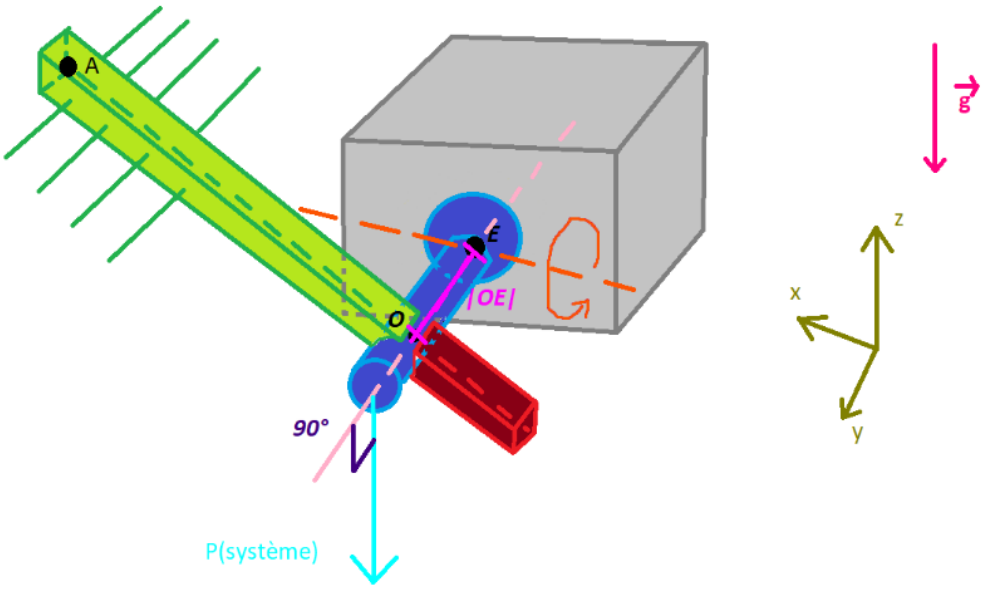


Figure III-2 : Efforts on the mast over the main structure

Each sized laying on one independent axis (in accordance with the requirement **OR-03-M**), it is critical to also give a magnitude for the {Mast + Antenna + Counterweight} induced torque on the system's structure at point E (as shown in the ***Figure III-2***).

According to the Varignon Theorem for mechanics:

$$\vec{T}_E^O = \sum \vec{T}_0^{O_i} + \sum \vec{R} \times \overrightarrow{O_i E} \tag{3.05}$$

Using this formula, we obtain the final value of the preliminary study of the efforts for one side of the system which is the addition on both torque on the mast:

$$\vec{T}_E^O = \vec{T}(antenna)_E^O + \vec{T}(counterweight)_E^O = 2 * 114\ Nm = 228\ Nm$$

The efforts applied on E are twice the effort applied on each side of the mast. With no possibility of balancing on the other side due to **OR-03-M**, the structure will have to handle the torque using a different way, un-deformable materials, or a counterweight system inside the system's structure.

These results only cover the most upper part of the system (the elevation block) as its movement makes it critical for the structure's design (while the azimuth block is supposed to be fixed) . Further study will have to be conduct for the azimuth part of the rotor but will cover other challenges such as the cable length to be passed through the structure for the antenna (according to **OR-01-M**).

III.2 Mechanical design

The mechanical transmission system is the segment that handles the physical positioning of the antenna. Since the gears need to fit the entry motor and rotational speed at the input and output, while handling the

efforts undergone by the system, its structure is highly dependent on the motor that are inputted on the design process.

As mentioned in II.2.1 the maximum budget shall not exceed 1500€, two motors were proposed by the CSUM team to be first tested for, respectively, the 3D (presentation) model and the first prototype. Their costs are relatively low (less than 45€ / 40£ each), and thus fit in the budget requirement. The first motor, the “GM4632-370”, is a small DC motor used for electronics and small motorized object. While the second, “MY1016Z2” is originally an electric bike motor.

Figure III-3 : (Left) GM4632-370 and (Right) MY1016Z2 [5]

The results obtained after studying and testing these motors shape the structure of the transmission system.

III.2.1 Determining the reduction ratio (theoretical study)

The first step in testing the compatibility between those motor and the future system is to determine the key parameters for mechanical transmission. The main value to obtain is the reduction ratio, which will shape the structure of the internal transmission system.

[5] Datasheets in reference

Assuming a quasi-perfect system (i.e., assuming power conservation) while considering the mechanical losses in the transmission system, the reduction ratio is obtained by using the following relations:

$$P_T = \omega_T * T_T = \omega_R * T_R = P_R \quad (3.06)$$

$$r = \eta_{system} * \frac{\omega_{input}}{\omega_{output}} \quad (3.07)$$

$$r = \eta_{system} * \frac{T_{input}}{T_{output}} \quad (3.08)$$

With:

$$P : Transmitted\ or\ received\ power$$
$$\omega : Input\ or\ output\ rotational\ speed$$
$$T : Transmitted\ or\ received\ torque$$

$$\eta_{system} : transmission\ system\ efficiency\ (\sim 95\ to\ 98\%)$$
$$r : reduction\ ratio$$

We can easily obtain formula (3.08) using (3.06) and (3.07). Nevertheless, both (3.07) and (3.08) bring complementary data regarding the characteristics required for the system.

The equation (3.07) determines the reduction ratio to respect, r, to match the trajectory's rotational speed, ω_{output}, knowing the rotational speed of the motor at the entry of the transmission system, ω_{input}.

Complementarily, (3.08) gives a measurement of the torque that the motor at the system entry needs to produce for the transmission to work, T_{input}. This value is determined by first computing a measurement of the torque applied onto the system, T_{output} (obtained in III.1.2) and using the reduction ratio obtained using (3.07).

(3.08) can also be used the reverse way. Depending on its quality, the manufacturer's datasheet can provide values of the motor torque. Then it is possible to compare the values required theoretically for the good working of the system with the one provided by the manufacturer.

The speed to match is the average speed in elevation and azimuth; respectively 4,840 and 14,63 deg/min (**OR-04-M**). These rotational speeds are considerably low compared to the one provided by the proposed motors.

Both azimuth and elevation speed must be verified. Nevertheless, elevation speed being the lowest one, it is appropriate to use it as a reference for the study and tests since it corresponds to the slowest possible configuration and thus, the highest required reduction ratio.

The two following sections correspond to respectively the 3D-printed model's proposed motor: GM4632-370 and the prototype's proposed motor: MY1016Z2. Each study uses a different method due to the project requirements and inputs.

III.2.1.1 GM4632-370

The GM4632-370 is a DC motor with the following characteristics:

Input Voltage (V)	24
Internal reduction ratio[6]	1:150
Rotation per minute, RPM (free of load)	80
Nominal RPM	55
Minimal current to be applied (mA)	≤ 80

[6] The internal reduction ratio is already included when the rotational speed of the motor is given. Mechanically, almost all DC motors of this size have internal reduction gear to avoid direct contact between the motor and the transmission system.

Nominal current (A)	0.7
Stall current (A)	≤ 2.0
Nominal torque (kg.cm)[7]	15
Maximal torque (kg.cm)	≥ 48

Table III-2 : GM4632-370 specifications

The 3D-printed model was design with a reduction ratio r = 220. Therefore, knowing the average rotational speed of the orbiting satellite, it is possible to determine the average required speed that the motor must match during the passing by rearranging (3.07). For the 3D-printed model, we will ignore the friction (η_{system} = 1) since the material composing the structure are light weighted.

$$r * \omega_{output} * \eta_{system} = \omega_{input}$$

$$\omega_{input} = 220 * 0{,}001408 * 1 = 0.30976 \approx 0{,}31\, rad/s$$

In comparison the nominal speed of the motor is 55 rpm or 5,76 rad/s. It is then obvious that it is either necessary to increase the reduction ratio (i.e., adding cogs to the gears) or reduce the rotational speed of the motor.

The first alternative requires additional configuration and making time, while the second approach necessitates only to reduce the voltage input. This second solution is feasible for the 3D-printed model using a power source to change the voltage (which is directly related to the rotational speed). However, this approach cannot be used for the prototype configuration.

Nevertheless, the datasheet used does not provide enough information regarding the voltage values and their associated speed. Thus, additional

[7] 1 kg.cm ≈ 0.098 Nm

testing in laboratory is necessary to find if the second solution is feasible or if the first solution is mandatory, which will be shown in section III.2.2.1.

III.2.1.2 MY1016Z2

Using the similar method than in the previous section, the datasheet provides the following specifications for the MY10116Z2:

SPECS	NO LOAD		RATED LOAD				
	SPEED RPM	CURRENT A	TORQUE N.M.	SPEED RPM	CURRENT A	P-OUT W	EFFICIENCY η
250W 24V	434±5%	≤1.8	6.65±5%	357±5%	≤13.7	250	≥76%

Table III-3 : MY1016Z2 specifications

This time, the unknown value being the reduction ratio r, the original form of equation (3.07) is used. Using the rated load rotational speed value, the result is:

$$r = \eta_{system} * \frac{\omega_{input}}{\omega_{output}}$$

$$r = 0.98 * \frac{\left(\frac{357 * 360 * 2 * \pi}{360 * 60}\right)^{8}}{0.001408} = 26021$$

This reduction ratio value is relatively high compared to the 3D-printed model. Since the prototype has not been design yet, it is possible to theoretically construct a system with the wanted reduction ratio. The figure below shows an example of a possible configuration.

[8] 357 rpm = 357 * 360 deg/min = 357 * 360 * (2*π/360) / 60 rad/s

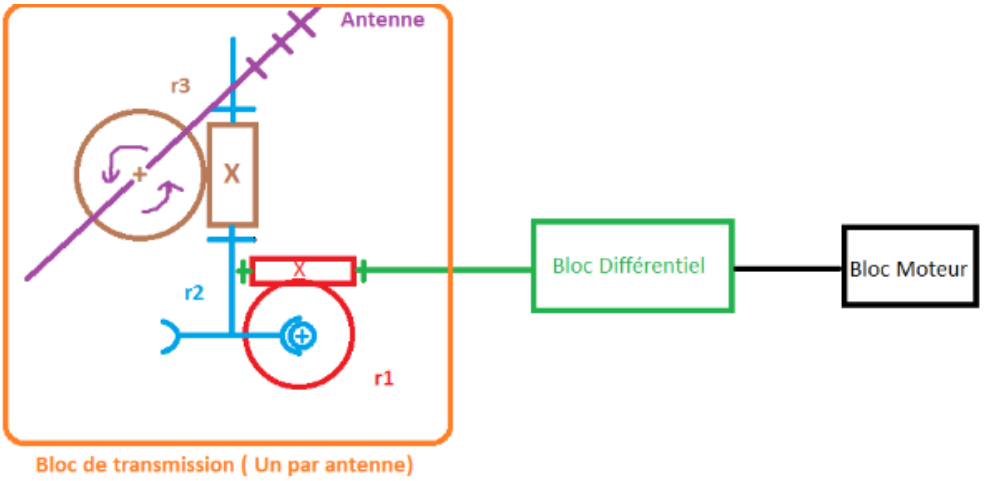


Figure III-4 : Example of possible configuration[9] (one block for each antenna)

The above configuration represents schematically the organisation of the different blocks of the mechanical transmission system. The Motor Block transmits the impulse to the block that allows to the Differential Block. The function of the differential block consists in preventing a jamming of the system in case of antenna blockage. The differential is a well-known and documented system used in automotive transmission that allows differential rotation of the wheels in turns but keeping them at the same pace in straight line. A system similar is adaptable to our antennas configuration.

Finally, the Transmission Block is composed of the gear that will reduce the motor rotational speed (i.e., the input speed). The configuration in *Figure III-4* illustrates a Transmission Block composed of 3 worm gears placed in series. An example of acceptable reduction factor for the worm gear configuration: a respective reduction factor of 40, 25 and 26 for a final r=26000. Another suitable configuration is 42 * 20 * 31 = 26040.[10]

---

[9] "Bloc de transmission "= " Transmission Block", " Bloc différentiel " = "Differential bloc ", " Bloc moteur " = " Motor Block"

[10] Considering the machining imperfections, friction, radiative pattern of the antennas for satellite transmission and the high value of reduction; reaching an approximate value around the dozen of the exact ratio of 26021 is acceptable.

Analytically to the GM4632-370, it is also possible to reduce the voltage and trying to decrease the rotational speed and find the relation between them.

III.2.1.3 Remark on the equation (3.08)

The interest of (3.08) is to provide an indicative value of the torque necessary to move the transmission system. In theory, the reduction factor obtained in (3.08) should be equal to the one obtained in (3.07). In practice, both results should be close: (3.08) verifies that the torque applied by the system is not under-sized, which would lead to an absence of movement of the system; or over-sized, which would cause stress that could break the gear of the system. Following the result obtained in section III.2.1.2, (3.07) gives:

$$r = 0.98 * \frac{\left(\frac{357 * 360 * 2 * \pi}{360 * 60}\right)^{11}}{0.001408} = 26021$$

While using the same datasheet provided in table III.2.1.2, (3.08) gives:

$$r = 0{,}98 * \frac{228}{6{,}65} \approx 34\ Nm$$

The value given by (3.08), i.e., the necessary reduction ratio for the motor to be able to move the gear, is consistently lower than the one given by (3.07). The motor is theoretically able to run the system with its specifications, but the difference in reduction ratio is such that the mechanical transmission system will undergo additional stress. The material that composes the gear is a key parameter here to handle such torques. Using a gear composed of steel, it is possible to have such a system. Nevertheless, it is an indicator that the motor might not be the most appropriate for this purpose.

[11] 357 rpm = 357 * 360 deg/min = 357 * 360 * (2*π/360) / 60 rad/s

III.2.2 Motors tests

Since the datasheet is not enough to compute the relation between the voltage and the angular speed of the proposed motors, testing them in operating condition is necessary to obtain a critical data for the rest of the study, especially for section V.3.

The data researched in these tests are:

1- Low and up threshold voltages: respectively the voltage when the motor starts to rotate without prior inertia and the maximum voltage appliable before damaging the motor (in both motor, capped at 24 Volts)
2- Computing the relation between the voltage and the angular speed of each motor

The relation between the voltage and angular speed being linear and depending on the motor, the test on the motor brings empirical, practical data to be used directly on the simulation.

The experimental protocol is basic and does not require much material besides a power source for the motor, putting a visual indicator on the rotor and using a camera; it is possible to visually compute the angular speed of the motor at different voltages, starting from the low threshold voltage and going by step of 1 Volt until reaching 24 Volts.

III.2.2.1 GM4632-370

Considering the weak mass moved in the 3D-printed model (less than 100g due to printed parts) this test was first conducted with an empty load for simplicity. Nevertheless, the motor has been tested afterwards, powering the whole elevation structure without any significant change on the values.

The threshold voltage for the GM4632-370 is 1.5 Volts, meaning that it will not be possible to apply a voltage lower than that to ensure the rotation.

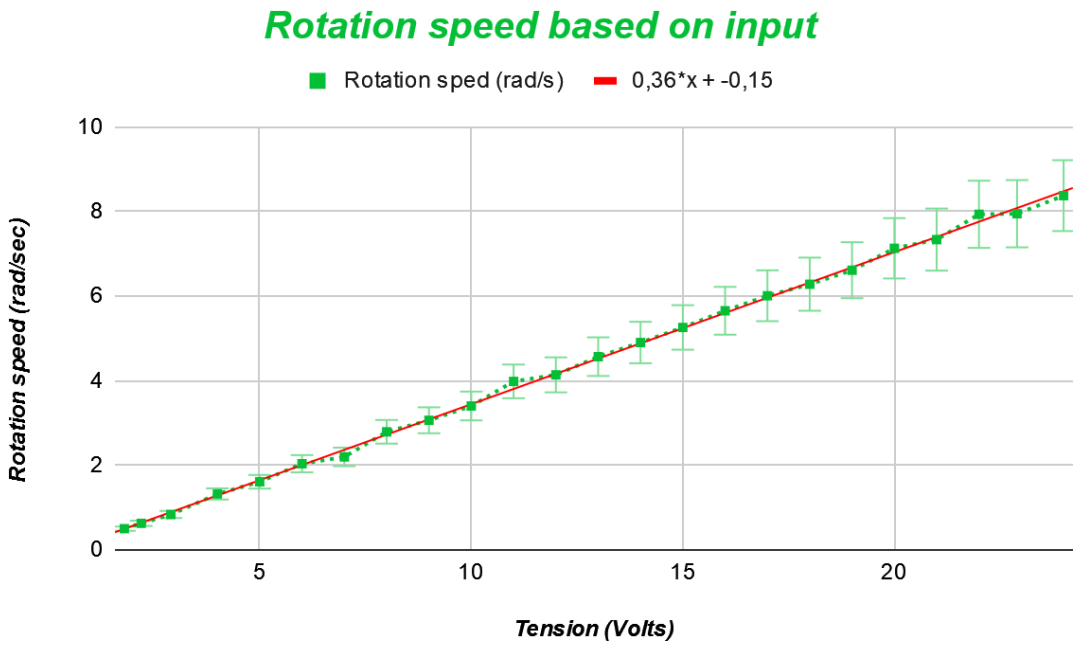


Figure III-5 : Angular speed based on voltage (GM4632-370)

The data obtained during the test produced the following equation and graph. Experimentally, the linear curve is fitting all the point of the graph, which is in accordance with the physical model of DC motors. Moreover, each angular speed value is compared to the wanted speeds in elevation and azimuth to determine the ideal reduction ratio to apply for fitting the satellite's passing.[12]

Voltage (V)	rotations	seconds	Average rpm	Average radian/s	Reduction factor necessary to reach 0,001408 rad/s	Reduction factor necessary to reach 0,004255419894 rad/s
1,8	1	12,5	4,8	0,5026548246	356,9991652	118,1210872
2,2	2	20	6	0,6283185307	446,2489565	147,6513591

Table III-4 : example of the obtained values table

However, the most useful data is the one illustrated in ***Figure III-6***. The reason for that is the need of a reverse methodology to simulate the motor for the tracking program. We use ***Figure III-5*** to obtain the reverse function

[12] The complete test data will be available on the appendix

of the one found by experiment: the voltage values based on the inputted angular speed and this function will go in the tracking program that will regulate the voltage for the motor. The obtained equation showed in ***Figure III-6*** can be seen in the program code provided in the **Erreur ! Source du renvoi introuvable.**.

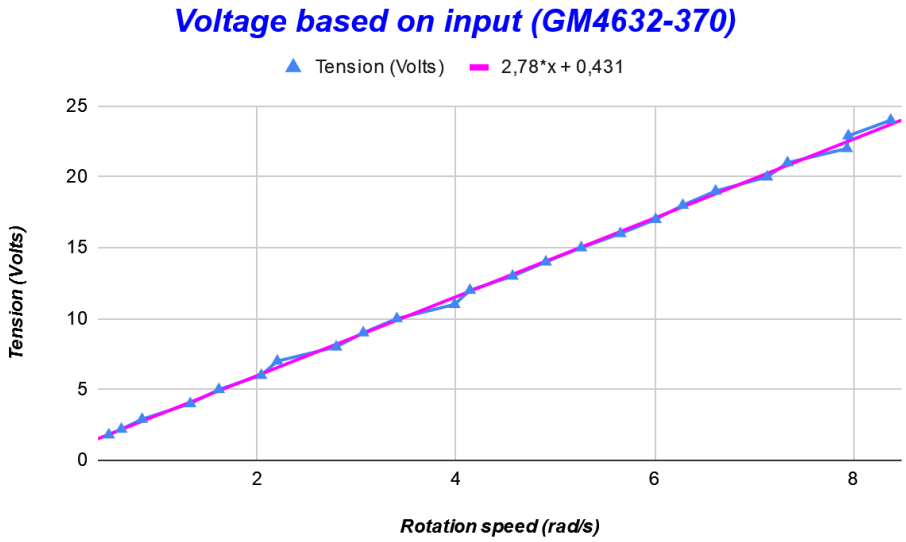


Figure III-6 : Voltage based on angular speed (GM4632-370)

III.2.2.2 MY1016Z2

The procedure for the MY1016Z2 is conducted the same way as the previous one. However, during the test, the motor was deemed unfitted by the RF engineer for its vibration and the generated noise that would have become interference close to the embedded system and antenna. The MY1016Z2 is then deemed inappropriate for the task and removed from the project. The results of the tests are yet available on the **Erreur ! Source du renvoi introuvable.**.

III.3 Conclusion of the mechanical study

The mechanical study provided useful insights and data for the prototype structure and the next WP(s).

First, the determination of the motor's couple constant and equation were necessary as they are a key element of the pseudo-tracking program explained in Software (see at section V.3).

Moreover, the capacity of the proposed motors has been assessed: the GM4632-370 is fitted to be used on the 3D-printed model while the MY1016Z2 has proven unable, too complex to fill out the requirements of the project. The choice of a low speed and more stable motor, in both vibration and torque provided is highly suggested. This is due to the heavy efforts on the mast and the slow rotation to provide during tracking.

Nevertheless, it is essential to remind that considering the current rotation ratio of the 3D-printed system, the voltage range for the good-working of the system might not be satisfied[13]. As a matter of fact, the section ***IV*** shows that the ideal gear holds a reduction ratio of 880. This does not prevent study and testing of WP03 and WP04 using the 3D-printed model while it remains a problem to solve. The most direct, mechanical approach would be to add additional gear and match the ideal reduction ratio: r=880. The gear modification consists in a belt-pulley and is relatively easy to produce at low cost using CAD and 3D-printing. However, we will use an electronic solution for this consideration (see section ***IV***). At first glance, it might only save time by sparring additional building, but it also brings a numerical way to avoid small variations of reduction factor in the gear being too compromising for the tracking system.

For the rest of the report, we will focus solely on the 3D-printed model and the GM4632-370 since the MY1016Z2 has been discarded. The first reason is time-related: other aspects of the project can be explored, and the selection of a new motor depends on many parameters (budget, rotational speed and associated torque, voltage and power, structure of the rotor, ordering for testing, delivery delays, etc). The time of this project being only 5 months, the author chose to focus on other aspects that could

[13] 1.5 to 24 Volts (without prior inertia), see section III.2.2.1III.2.2.1

still be developed. Moreover, this will not impact the other WPs for the following reasons:

1- The embedded system interface will not change between the 3D-printed model and the prototype as proof-of-concept achievement.
2- The program described in section V.3.3 is meant to adapt to the inputted system data (i.e., motor's characteristics, reduction ratio, etc). Thus, the core of the program is independent of the motor proposed and the associated model (3D-printed or prototype). It fits both mechanical configurations.

Finally, it is important to mention that if the next motor provided have a well-furnished datasheet (i.e., that provides enough Voltage/Rotational speed comparative values to describe its trend curve), it is possible to use the program shown in section ***V*** to directly simulate the motor properties. This saves time to check on the motor capability of fulfilling the speed requirements.

III.4 Remark: budget consideration on the mechanical WP

Since the motor proposed for the prototype model has been refused and no other motor have been proposed as explained in III.3, the budget associated for the prototype model cannot be measured. Nevertheless, an estimation of one motor's price would be between several hundreds of euros to several thousand depending on the motor quality and characteristics. In addition, the mechanical transmission system, once determined for the new motor, will add additional cost that could go beyond the required budget.

IV. Power and Embedded Systems (WP03)

The embedded system is the segment that will handle the hardware responsible for the digital operations inside the rotor's structure. It is the second most expensive WP after WP02. Considering the relative simplicity of the 3D-printed model, the electronics embedded inside the rotor will be composed of 2 motors, a motor driver, sensors for the motors, various connectors, an ethernet link and a controller. The motor being already selected and equipped with a rotation sensor, the two elements to choose are the controller and the driver.[14]

IV.1 Phase Width Modulation (PWM)

As mentioned in section III.3, the WP04 brings an alternative regarding the need of regulating the rotational speed to match the satellite's orbit. The input voltage being constant DC of 24 Volts, 2 choices are then proposed:

1- Adding a belt-pulley with a reduction factor of 4 modifies the gear and changes the reduction ratio from 220 to 880.
2- Using a PWM signal to "simulate" a lower voltage impulse that matches the rotational speed necessary to follow the satellite's orbit.

A Pulse Width Modulation (PWM) signal is a numerical process that simulates analogical devices to synthesize pseudo analogical signals, e.g.: transform an input signal of 24 Volts of amplitude into a 6 Volts amplitude signal in exit. As shown in the figure below, the step function is chopped by the PWM signal to construct an approximated analogue,

[14] The prototype system will be more likely to have additional sensors for weather and PID control over its motor. Nevertheless, the 3D-model is enough to assess the efficiency of the tracking system on a digital/analogical point of view.

sinusoidal one. The PWM signal varies between the '0' and the '1' state to change the increase and decrease of the output and create with, enough frequency, an approximated sinusoid of a defined, averaged amplitude.

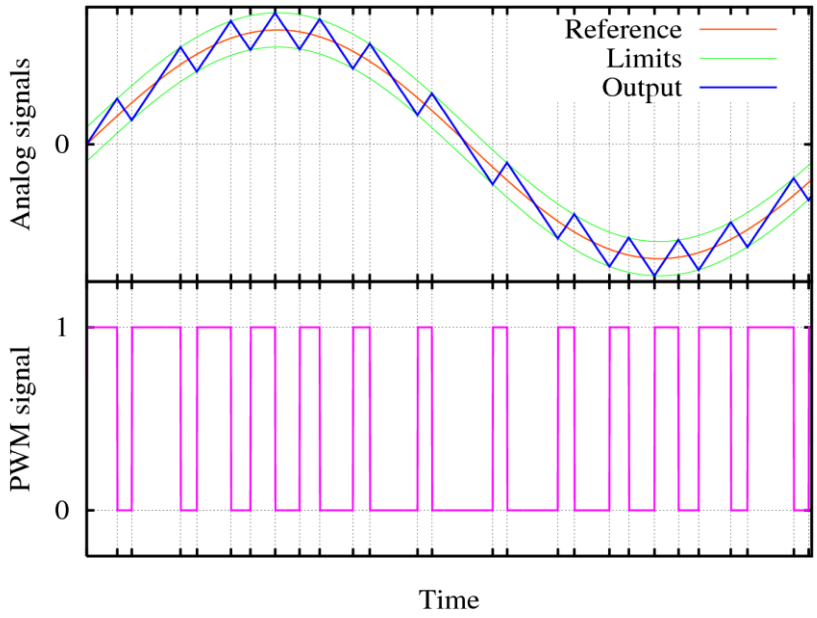


Figure IV-1 : Example of a PWM signal

The average amplitude of this "reconstituted" sinusoidal signal is driven by the duty cycle α:

$$\propto = \frac{\tau}{T} \tag{4.01}$$

With:

$$\propto : duty\ cycle\ of\ the\ signal\ (i.e., the\ \%\ of\ the\ input\ amplitude\ used\ in\ the\ signal)$$
$$\tau : total\ duration\ of\ state\ '1'\ during\ T$$
$$T : duration\ of\ the\ signal$$

We use the previous example of a 24 Volts DC input that needs to be transformed into a 6 Volts signal, which is a quarter of the amplitude of the input signal. In that case, α = 0.25 (or 25 %), the PWM signal will be at state 1 during 25% of the time T, with a step frequency depending on the wanted shape of the output signal. For a sinusoidal signal like showed in Figure IV-1, the chopping will have the same frequency. **Therefore, it Is**

possible to simulate voltage lower than the lower threshold voltage authorized with a constant 24 volts power source, which was the main issue showed by the test conducted in section III.2.2.1**.**

The second solution is simpler and more modular than having to change the reduction ratio of the system since the PWM factor is editable directly on the controller interface using its coding language. Therefore, the second solution is the one selected to solve the situation described in section III.3 and a PWM signal generator is required for the controller to be chosen.

IV.2 Selection of the controller

IV.2.1 Mission and project requirements for controller

As mentioned in section II.2.3.The required features for the controllers are:

3- Ability to generate PWM signal. Derived from **OR-09-PES** and the previous sections
4- Direct, physical connection to the ground segment not to cause interferences or damage the controller due to high amplitude signal coming from antenna (Caused by **OR-12-PES**).
5- Programming language accessible for the most CSUM member.
6- High modularity of the embedded system configuration

Two controllers are already available at the CSUM and thus are the two proposed choices for this project.

IV.2.2 First proposition: STM32

The STM32 is a type of 32-bit microcontroller and is the controller proposed at the beginning of this study. Including various sensors for data handling, sound/video capture/restitution, electromechanics and robotics, it uses the C and C++ as programming language. The NUCLEO variant is compatible with Arduino IDE. The programming of such a controller is

done using a computer linked to the board. It can generate a PWM from the pins showed on ***Figure IV-2*** *: STM32 Nucleo* As other microcontroller of its type, it can receive additional module for radio transmission, wi-fi, ethernet connection, etc.

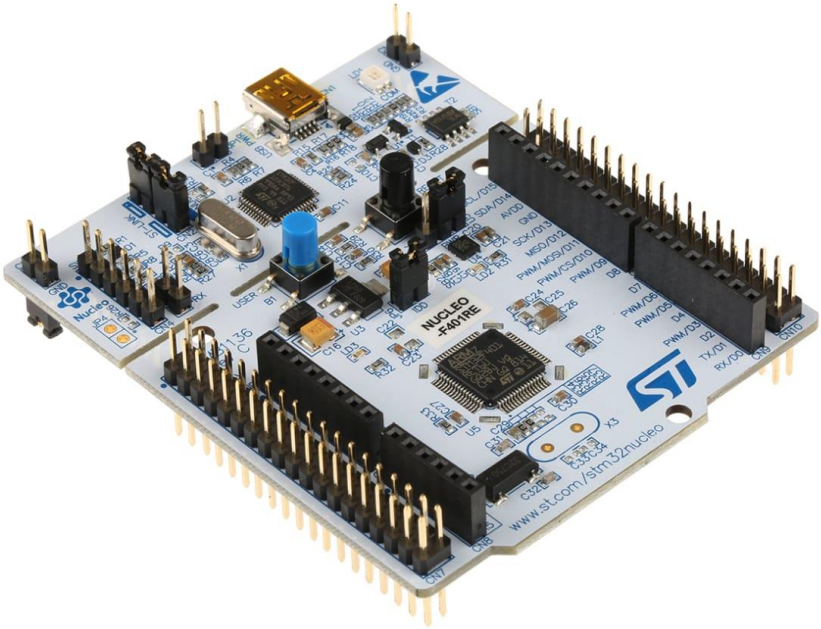


Figure IV-2 : STM32 Nucleo

This microcontroller possesses the required features to be used in the future system. Adding the required ethernet module, the price of such a controller gravitates between 30 - 40€.

IV.2.3 Second Proposition: Raspberry Pi 3B+

However, a second controller is proposed: The Raspberry Pi (or RPi). This controller is programmable without using another computer (since it is a microcomputer itself). In addition, following the requirement **OR-12-PES**, the controller must be connected permanently and physically to the GS. The RPi possesses already an ethernet port and various USB ports. Thus, it does not need additional port, it is more expensive than its counterpart: the pandemic has seen an increase of the RPi price, and it can be found now around 50 - 60€ without additional components.

Figure IV-3 : Raspberry Pi 3B+

However, the real advantages of the RPi are its onboard programming languages: Python and C++ and its accessibility. These two languages are different in construction, which allows more possibilities of programming for the CSUM (whose programming skills and languages knowledge varies among its team members). Using it has a full computer, it is also possible to connect the RPi directly to internet and import python (or C++) code and use it in its IDE.

IV.2.4 Final selection

Despite being more expansive, the RPi is selected and will replace the previous choice: the STM32 NUCLEO, selected at the original state of the project. The already present ethernet port, the possibility of programming directly on the computer and the use of the python language, which is well documented and possess various libraries. It is also more mastered than C++ in the CSUM teams, making the RPi the best choice for controlling the motors.

IV.3 Selection of the driver

Among the driver available at the CSUM, the one proposed by Ms COURTIN, the L293D is simple in utilisation. The mapping scheme (see ***Figure*** *IV-5 : Wanted motor configuration using the L293D* below ***Figure IV-4***) allows to control one motor on both direction for each side of the chip.

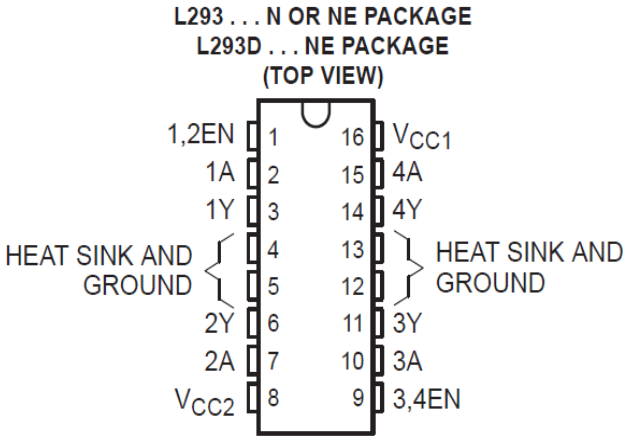


Figure IV-4 : Mapping of the L293D controller

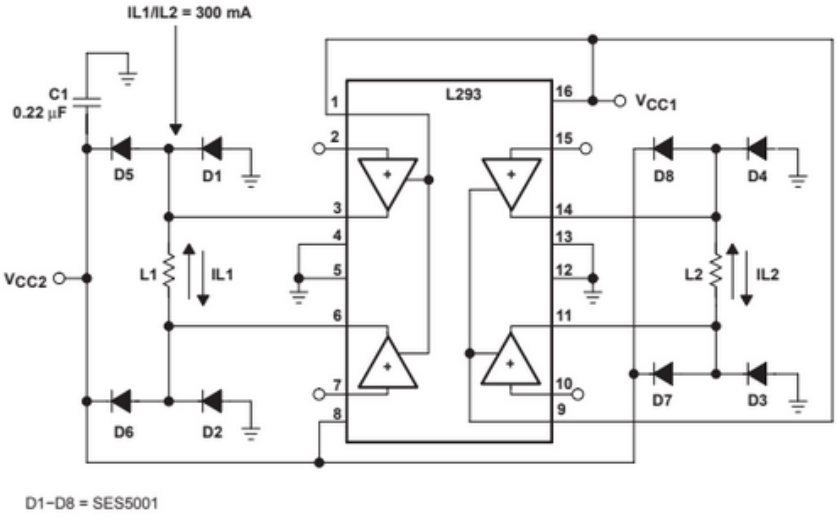


Figure IV-5 : Wanted motor configuration using the L293D

In addition, the connection for motor control requires few components as shown in the ***Figure IV-5***. With 8 identical diodes, and few resistances, the mapping is theoretically ready to work. However, after several attempts with different connectors and model of L293D, only one side of the configuration appeared to function properly. Nevertheless, the author will

not state the inefficiency of a configuration that might be due do a damaged shipment, or other electronic malfunction, (since the theoretical configuration has been validated since 2005).

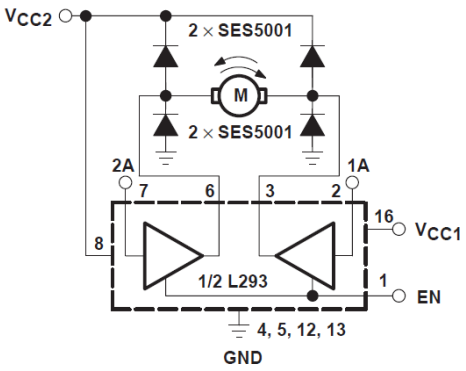


Figure 5. Bidirectional DC Motor Control

EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Figure IV-6 : One-sided configuration assembly

Theoretically, two L293D can still be used in 'one-sided' configuration to be directed by the controller, but this would use additional space and connectors. This is especially to the absence of H bridge[15] (symbolized on ***Figure IV-5*** by the diode assembly) inside the driver and thus needs to be manually added on the assembly.

As shown in ***Figure IV-7*** Another configuration was then proposed using the following components: H-bridge block (L6506) and two smaller drivers (L6201PS). This configuration spares more space than the previous one. Unfortunately, the needed delivery of components for this configuration were delayed due to the still ongoing pandemic.

[15] A H-bridge is an electronic assembly that allows the motor to go in both direction by controlling its voltage polarity.

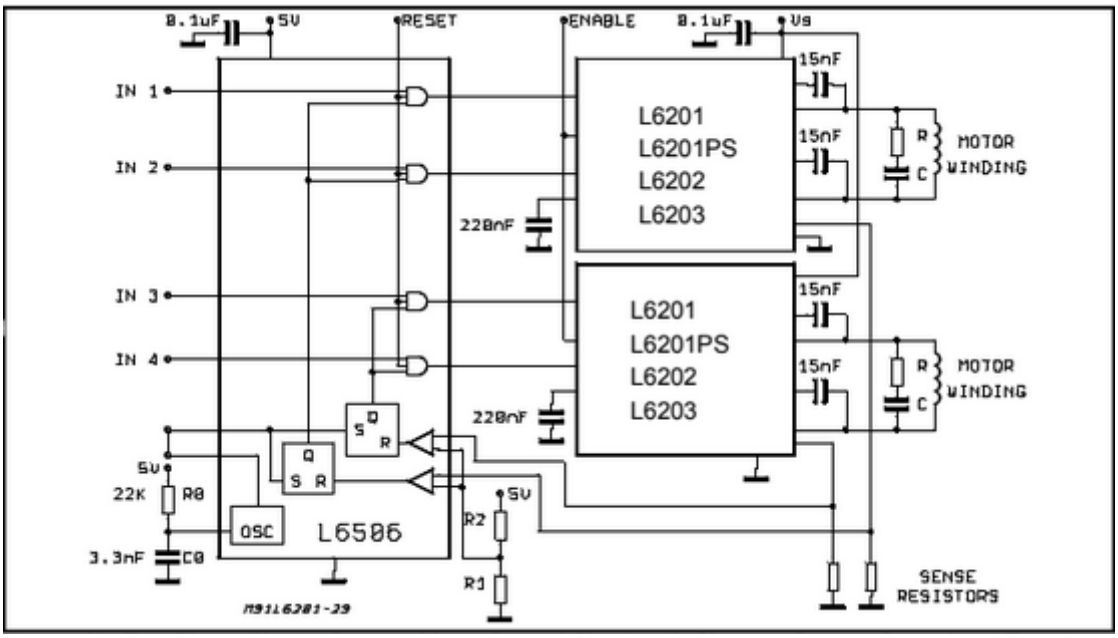


Figure IV-7 : Another proposed configuration for the motor control system

The final driver selected by default for the end of the project is then the L293D. In both cases, this configuration implies to solder properly numerous elements, which emphasizes the same issue in both configurations. These are relatively achievable in a close lab while being harder on board. Therefore, the electronics must be the simplest possible as stated in section IV.2.1. And in the case were the L293D is used as its one-sided mapping, the L6506 – L6201PS is the chosen configuration.

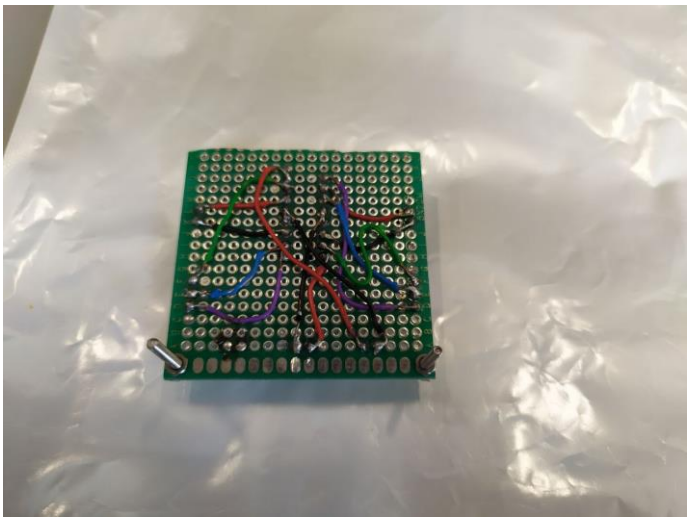


Figure IV-8 : Soldering the connectors for the L293D components

IV.4 Final configuration

For the rest of the rest of the project, the focus will be on a single-sided motor configuration because of the circumstances mentioned at the end of the last section. To test the PWM generation for one motor, we will use the pin number 12, 22 and 24 named respectively GPIO18, GPIO25 and GPIO8.

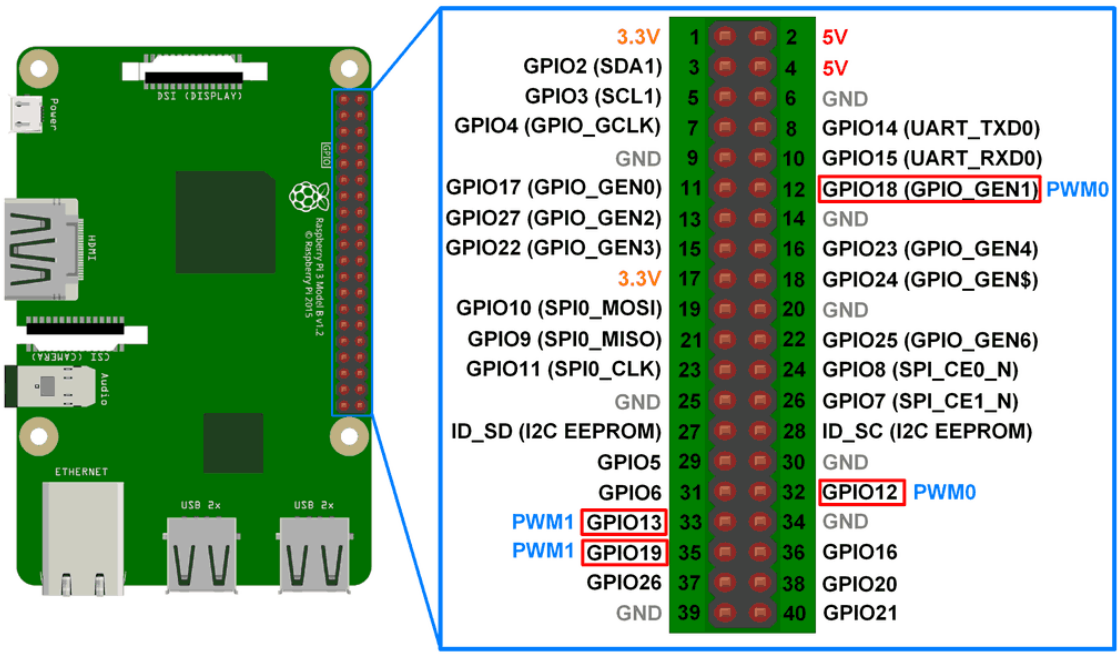


Figure IV-9 : Mapping of PWM ports of a RPi

Using ***Figure IV-6*** and ***Figure IV-9*** : GPIO18 is the pin that emits the PWM signal to the enable pin (Pin 1 on the driver), while the other two are connected to 1A and 2A (Pin 2 and 7 on the driver). The PWM signal will enable or not the system power depending on its state (high or low). The state of 1A and 2A will be driven by GPIO25 and GPIO8; their state defines the rotation direction of the motor (the table is given in ***Figure IV-6***).

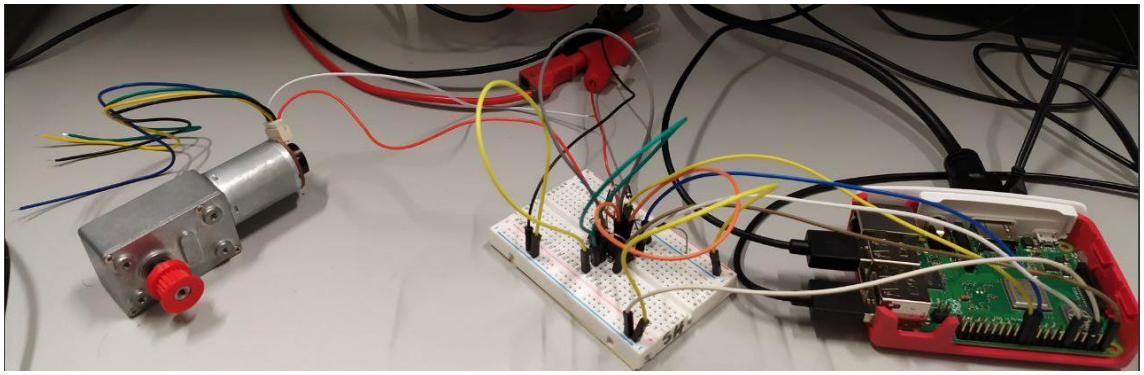


Figure IV-10 : Final configuration for program testing

Now that the final configuration for program test is established, we can command the motor with the RPi IDE and control its rotational speed using automatically or remotely the PWM signal generated by the RPi.

Remark: it is necessary to point out that in this configuration, the RPi is used (with its 5V input pin) to power the driver. This might not be necessary but remembered to avoid power spike that could destroy the controller or insufficient power injected to the driver during the prototype tests in case of wrong wiring.

V. Software (WP04)

The software WP regroups the software-related tasks of the project on both rotor system and GS. The development of this WP consists in finding the critical features of the program based on the established in Software requirements (see section 0), constraints and user's need.

The study conducted on the other WP defined the needs of the system in matter of software. The WP aims are then:

1- Compute the average time of an orbit (using the file presented in ***Figure I-4***). This task is already done in the other WP to compute the average rotational speed on the elevation and azimuth.
2- Simulates the wanted system to compute the motor voltage input using its voltage trend curve based on its rotational speed.
3- Extract from the file produced by Gpredict the data and translate them into useful inputs for the rotor's embedded system to track the satellite.

V.1 Gpredict-produced file structure and methodology

Gpredict is executed on the GS computer and produces the file containing the coordinates of the next 15 upcoming orbit of ROBUSTA-1B. The file containing the tracking coordinates produced is in text format (".txt") and is stored in a dedicated folder. The text file is named "ROBUSTA-1B-passes.txt" and is updated at each new orbit data produced by Gpredict.

ROBUSTA-1B-passes.txt - Bloc-notes

Fichier Edition Format Affichage Aide

Upcoming passes for ROBUSTA-1B
Observer: seb, Monpellier-Mediterrannee, France
LAT:43,58 LON:3,97

AOS	TCA	LOS	Duration	Max El	AOS Az	LOS Az
2020/04/06 19:50:04	2020/04/06 19:55:34	2020/04/06 20:01:04	00:10:59	31,52	144,29	357,87
2020/04/06 21:24:11	2020/04/06 21:29:19	2020/04/06 21:34:26	00:10:15	17,29	203,66	331,11
2020/04/07 08:50:30	2020/04/07 08:56:06	2020/04/07 09:01:43	00:11:12	35,42	20,20	174,41
2020/04/07 10:24:30	2020/04/07 10:29:30	2020/04/07 10:34:30	00:10:00	16,37	354,55	233,05
2020/04/07 19:29:23	2020/04/07 19:34:31	2020/04/07 19:39:38	00:10:15	18,76	130,77	3,74
2020/04/07 21:02:27	2020/04/07 21:07:59	2020/04/07 21:13:30	00:11:03	30,05	189,41	338,03
2020/04/08 08:29:31	2020/04/08 08:34:47	2020/04/08 08:40:04	00:10:32	20,17	26,85	160,45
2020/04/08 10:03:02	2020/04/08 10:08:27	2020/04/08 10:13:53	00:10:50	27,19	0,54	219,37
2020/04/08 19:08:56	2020/04/08 19:13:31	2020/04/08 19:18:05	00:09:08	11,24	116,42	10,25
2020/04/08 20:40:59	2020/04/08 20:46:42	2020/04/08 20:52:26	00:11:26	56,03	175,97	344,16
2020/04/08 22:18:30	2020/04/08 22:21:07	2020/04/08 22:23:44	00:05:13	2,35	248,94	304,14
2020/04/09 08:08:43	2020/04/09 08:13:25	2020/04/09 08:18:06	00:09:22	11,50	34,72	145,20
2020/04/09 09:41:41	2020/04/09 09:47:21	2020/04/09 09:53:01	00:11:20	48,41	6,20	206,20
2020/04/09 11:17:09	2020/04/09 11:20:03	2020/04/09 11:22:57	00:05:48	3,21	334,44	272,89
2020/04/09 18:48:50	2020/04/09 18:52:35	2020/04/09 18:56:19	00:07:29	6,06	100,38	18,14

Orbit 15492

Time	Az	El	Range	Footp
2020/04/06 19:50:04	144,29	0,00	2571	4892
2020/04/06 19:50:37	142,54	2,14	2346	4894
2020/04/06 19:51:10	140,40	4,48	2125	4897
2020/04/06 19:51:43	137,72	7,09	1907	4899
2020/04/06 19:52:16	134,32	10,04	1696	4902
2020/04/06 19:52:49	129,86	13,43	1494	4905
2020/04/06 19:53:22	123,85	17,33	1306	4908
2020/04/06 19:53:55	115,55	21,73	1139	4910
2020/04/06 19:54:28	103,98	26,32	1004	4913
2020/04/06 19:55:01	88,40	30,11	915	4916
2020/04/06 19:55:34	69,74	31,52	887	4919
2020/04/06 19:56:07	51,35	29,76	925	4922
2020/04/06 19:56:40	36,34	25,82	1022	4925
2020/04/06 19:57:13	25,29	21,27	1163	4928
2020/04/06 19:57:46	17,38	16,96	1333	4931
2020/04/06 19:58:19	11,65	13,16	1524	4934
2020/04/06 19:58:52	7,38	9,85	1727	4937
2020/04/06 19:59:25	4,13	6,96	1940	4939
2020/04/06 19:59:58	1,57	4,40	2158	4942
2020/04/06 20:00:31	359,53	2,10	2380	4944

Figure V-1 : Example of data produced by Gpredict for ROBUSTA-1B orbit

The file is composed of a headline and a chronological list of the upcoming orbit; then each orbit is detailed, the estimated position is actualized every 33-34 second. The coordinates are listed from the first appearance on sight of the antenna and until the last period before going below the antenna's sight.

Due to the average duration of an orbit, Gpredict average lists contains between 20 and 21 set of coordinates. One set containing the date of the orbit in format YYYY/MM/DD, HH:MM:SS, azimuth axis, elevation axis, range from the GS in km and the footprint it offers (i.e., the coverage area for communication).

Among those inputs, the data needed for the direct tracking[16] program are the date in HH:MM:SS, azimuth axis and elevation axis. The method that shapes the program consists in computing the rotational speed in azimuth and elevation using the n and n+1 coordinates and their time difference. After that, using the chosen reduction ratio to compute the selected motor's speed, we obtain the voltage value thanks to the selected motor's trend curve obtained in III.2.2.1.

V.2 Software requirements

Using the requirements established in 0, the critical points to emphasize on the program are:

(a) The program shall access and keep unmodified "ROBUSTA-1B-passes.txt".
(b) Simplicity of the program: the function used shall be simple and modular enough to be modified by another team member following clear instructions.
(c) Any input error from the user shall not damage the mechanical system or the embedded electronics
(d) The program should be coded in an accessible language for the CSUM member (and more generally for the overall user)
(e) The program shall be able to run on a RPi 3B+[17]
(f) The program shall be first run manually for the test phase

[16] This does not consider the pre and post positioning phase and thus considers that the rotor is already prepared to follow the satellite (which of course incorrect).
[17] Thanks to the direct connexion between the RPi and the GS, it is possible to run the main program on the GS computer. However, the RPi will still have to host the function that will control the rotor based on the data received from the main program. Moreover, this situation will be different outside the CSUM, on a boat, where the RPi, is likely to handle the operation by itself by downloading the "ROBUSTA-1B-passes.txt" file regularly.

(g) Following what was discussed in section III.3: it shall be possible to assess if a motor is suitable for a system, given the reduction ratio and orbit data.[18]

V.3 Operating mode of the program

Before beginning this section, the author wants to specify a term that will be used in an inaccurate way. The program that will be presented as “Tracking program” is a “pseudo-tracking program” in the way that it does not assess the position of the orbiting at any moment and in anyway. Neither does it compares nor corrects whatsoever data produced by Gpredict.

This can be done in the following program but complying to the simplicity requirement (b) and the time given for this project, this option has not been considered for the time being. For the rest of the section, the mention “Tracking program” will refer in a shorter way the “Pseudo-tacking program” used as a core program of this section.

The program presented in this section is called by its main function: “Voltage_Regulation_for_GS_Operator”. The whole program is available in the appendix.

V.3.1 First version (MATLAB)

Prior to the embedded version, while selecting the controller, a test version of the program was coded using MATLAB. The use of this language seemed appropriate due to the possibility to use a C++ compiler and its proximity with the Python language in syntax and library, both classical languages for controller.

Prior to the test, the input that are entered will be listed in the below table:

[18] It is also a way to save money by potentially preventing physical testing if the data provided on the voltage and rotational speed are enough

Reduction ratio	File name	Motor type	Curve type
R = 880	Test.txt	1	1

Table V-1 : Input parameters for the MATLAB test program

The signification of the input on Motor type and Curve type will be explained in the below description. The basic principle of the first version of the program consists in the following steps (illustrated with user-software interaction figures):

1- Enter the wanted reduction ratio:

```
Fitting_current_to_motor

prompt1 =

    'Enter the reduction ratio : '

Enter the reduction ratio : 880

p1 =

   880
```

Figure V-2 : Input for MATLAB program (1)

2- Enter the wanted file to be extracted (for the test version, only the file “Test.txt” is fitted):

```
prompt =

    'Name your datafile with its extension (eg, blablabla.txt): '


prompt2 =

    ' File for the predicted coordinates of the satellite :'

 File for the predicted coordinates of the satellite :Test.txt
```

Figure V-3 : Input for the MATLAB program (2)

3- Read “Test.txt”, the extract of “ROBUSTA-1B-passes.txt” is shaped as shown in the below figure. The program will extract each line until a blank line ends the section:

*Test.txt - Bloc-notes

Fichier Edition Format Affichage Aide

Orbit 15492

Time	Az	El	Range	Footp
2020/04/06 19:50:04	144.29	00.00	2571	4892
2020/04/06 19:50:37	142.54	02.14	2346	4894
2020/04/06 19:51:10	140.40	04.48	2125	4897
2020/04/06 19:51:43	137.72	07.09	1907	4899
2020/04/06 19:52:16	134.32	10.04	1696	4902
2020/04/06 19:52:49	129.86	13.43	1494	4905
2020/04/06 19:53:22	123.85	17.33	1306	4908
2020/04/06 19:53:55	115.55	21.73	1139	4910
2020/04/06 19:54:28	103.98	26.32	1004	4913
2020/04/06 19:55:01	088.40	30.11	915	4916
2020/04/06 19:55:34	069.74	31.52	887	4919
2020/04/06 19:56:07	051.35	29.76	925	4922
2020/04/06 19:56:40	036.34	25.82	1022	4925
2020/04/06 19:57:13	025.29	21.27	1163	4928
2020/04/06 19:57:46	017.38	16.96	1333	4931
2020/04/06 19:58:19	011.65	13.16	1524	4934
2020/04/06 19:58:52	007.38	09.85	1727	4937
2020/04/06 19:59:25	004.13	06.96	1940	4939
2020/04/06 19:59:58	001.57	04.40	2158	4942
2020/04/06 20:00:31	359.53	02.10	2380	4944

Figure V-4 : Test.txt

4- Extract the azimuth, elevation, and HH:MM:SS time data

5- With an iterative loop, compute the wanted angular evolution and rotational speed (the derivation is made using the n+1 and n angular values divided by the time interval) at each interval for both azimuth and elevation that the rotor needs to follow

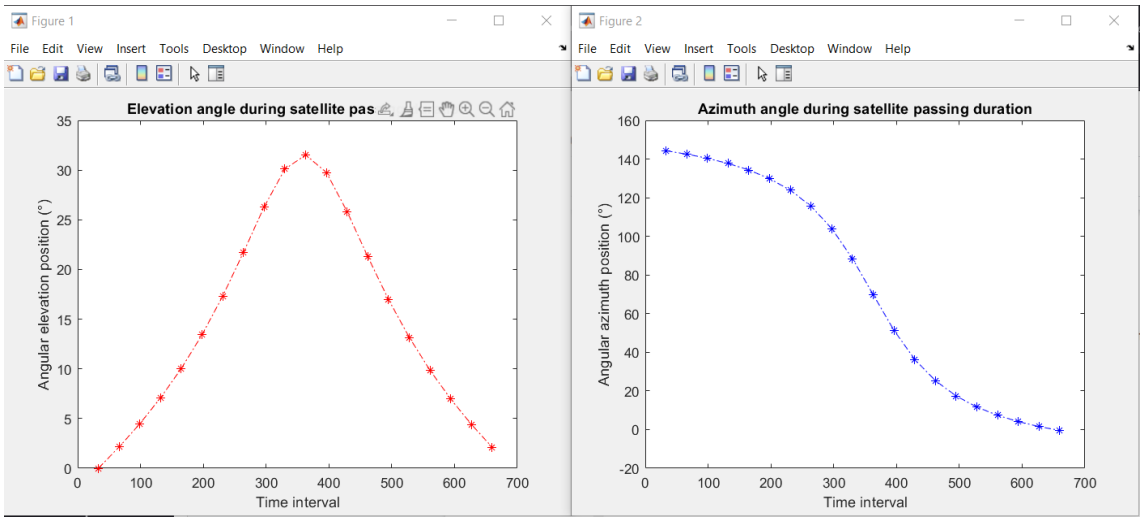


Figure V-5 : Wanted angular position evolution during the orbit

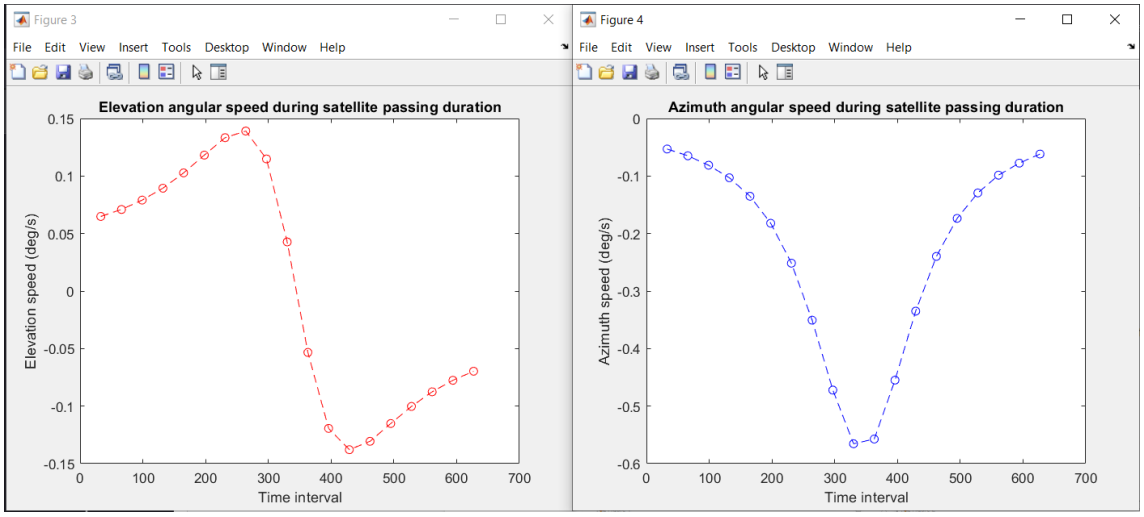


Figure V-6 : Wanted rotational speed during the tracking

6- Enter the wanted motor between the 2 proposed and tested; GM4632-370 (1) and MY10116Z2 (2). Enter the wanted approximated curve, linear (1) or second order polynomial (2)[19] :

```
choice1 =

    'Which motor do you wish to use (i.e., which model ?):\n - for GM4632-370, enter 1\n - for MY1016Z2, enter 2\n'


choice2 =

    'Which trend curve do you want to use to approximate your values ? Choose between linear (type 1) or 2nd order polynomial (type 2)\n'

Which motor do you wish to use (i.e., which model ?):
 - for GM4632-370, enter 1
 - for MY1016Z2, enter 2
1

motor_type =

     1

Which trend curve do you want to use to approximate your values ? Choose between linear (type 1) or 2nd order polynomial (type 2)
fx 1
```

Figure V-7 : Input for the MATLAB program (3)

7- Use the reduction factor to compute the linked motor rotational speed

[19] The second order polynomial approximation, as mentioned before, is less accurate physically than the linear approximation (that corresponds to the physical reality). Nevertheless, the author chose to keep it in the program to compare the results of two different approximation. This explains why all the tests are done using the linear approximation.

w_axis =		w_motor =	
0.0011	-0.0009	0.9960	-0.8145
0.0012	-0.0011	1.0891	-0.9960
0.0014	-0.0014	1.2147	-1.2473
0.0016	-0.0018	1.3730	-1.5824
0.0018	-0.0024	1.5778	-2.0758
0.0021	-0.0032	1.8151	-2.7972
0.0023	-0.0044	2.0479	-3.8630
0.0024	-0.0061	2.1363	-5.3849
0.0020	-0.0082	1.7639	-7.2513
0.0007	-0.0099	0.6562	-8.6848
-0.0009	-0.0097	-0.8191	-8.5591
-0.0021	-0.0079	-1.8338	-6.9860
-0.0024	-0.0058	-2.1177	-5.1429
-0.0023	-0.0042	-2.0060	-3.6815
-0.0020	-0.0030	-1.7686	-2.6669
-0.0018	-0.0023	-1.5405	-1.9873
-0.0015	-0.0017	-1.3451	-1.5126
-0.0014	-0.0014	-1.1915	-1.1915
-0.0012	-0.0011	-1.0705	-0.9495

Figure V-8 : Converting the wanted rotational speed to motor speed

8- Use the equation linking the voltage to the motor speed and return the voltage values during the duration of the orbit

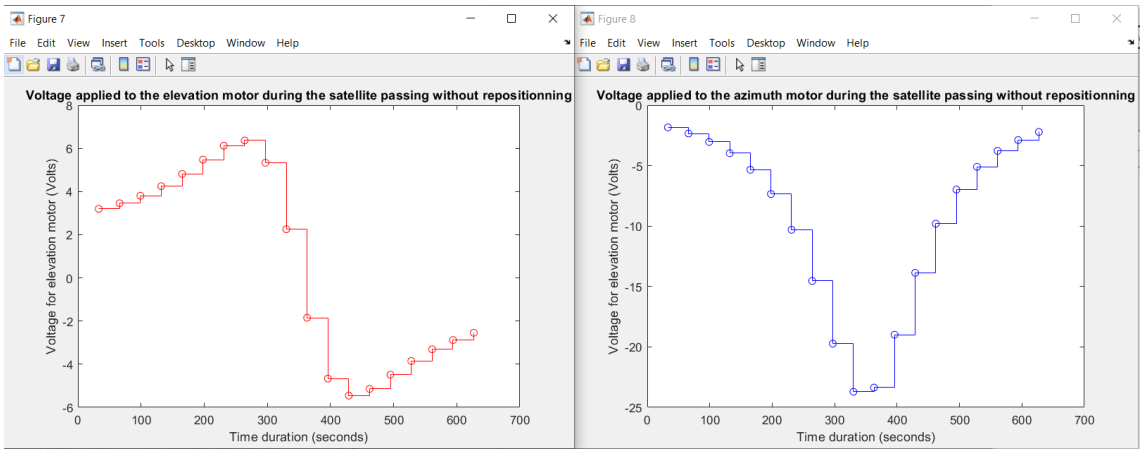


Figure V-9 : Final voltage applied to the motors

The result displayed to the user of this simulation is a step function (since we use average voltage on a definite interval of time) of the voltage needed to match the satellite's trajectory. These curves are a graphic way that allows the user to control the voltage values (and prevent any power spike). Nevertheless, the data transmitted to the rotor control system is a matrix that carries 3 vectors: elevation voltage, azimuth voltage, time duration.

V.3.2 Adaptability and modularity of the program

The test program has been thought to be modular as required in (b), to fulfil this requirement, the code has been separated in several simple functions. This approach necessitates more rigorous while coding the parts that call one another. Nevertheless, once done, it provides a highly modular potential.

As an example: the function linking the rotational speed to the voltage, called “Applied_Voltage”, only requires few lines to be modified if the user wishes to change the motor used (name of the motor and the equations, see the figure below). The function can be added other motor type or duplicated for each motor at the user’s convenience.

```
if motor_type==1
    if curve==2
        U_2nd_order_polynomial = 0.565 + 2.69 * w_motor + 0.0103 * ((w_motor).^2)
        V = U_2nd_order_polynomial
    else
        U_linear =  2.78 * w_motor + 0.431
        V = U_linear
    end
```

Figure V-10 : Lines to modify to change the motor data in Applied_Voltage (MATLAB)

The will of optimisation for simplification conducted the author to make various version of the same program. The versions of the program were ordered as followed.

Version 1 V1 (MATLAB)	The one tested in V.3.1. The user enters the reduction ratio, and the test file with a specific format. After the first computation, the user choses between two motor and curve type to obtain the results. A default mode exists and is trigger when the motor or curve type selected does not match the wanted results.
Version 2 V2 (Python)	The user enters the reduction ratio and the file name that contains all the orbits with the number of the orbit researched. The program will find the orbit and compute the relevant data. Then the user choses the same way as V1 for the end.
Version 3 V3 (Python)	V3 is like V2. Nevertheless, the program will look for the file name (which is in the same folder as the program) given and return an error message, and another try if the file name or the orbit number inside the orbit file is not found. It is then possible to change the file name or the orbit number (or both if needed).

Table V-2 : Test oriented version of the tracking program

These 3 first versions still allow all the operations to be monitored and triggered by the user, while the following versions are more oriented on the definitive rotor configuration.

Version 4 V4 (Python)	V4 is more compact. While all the previous version aimed to test the different possible features of the program, V4 aims to removes the most code to only fulfil the tracking task for a definite system. In V4, the reduction ratio and motor are chosen, the curve to match the voltage to the angular speed is defined and the file selected is always the same and updated regularly (and so is the next orbit to track). The user must still trigger the program.
Version 5 V5 (Python)	Same as V4 but the program is triggered on time using the clock of the RPi and thus is fully automatic. (Not achieved during this project)

Table V-3 : User-oriented versions of the tracking program

Considering the state of advancement of the project, the V3 and V4 will be respectively used for the prototype model and the 3D-printed model. V3 is more useful to the prototype since the motor and associated reduction ratio have not been selected, the test phase is then still ongoing. While the 3D-printed model has its defined reduction ratio and motor and uses the CSUM GS Gpredict folder and thus does not need more tests than the optimisation of the program (and its complementary functions) at this state.

V.3.3 Detailed (complete) program

The detailed version of the most complete version of the test program (V3) is described in the following diagrams:

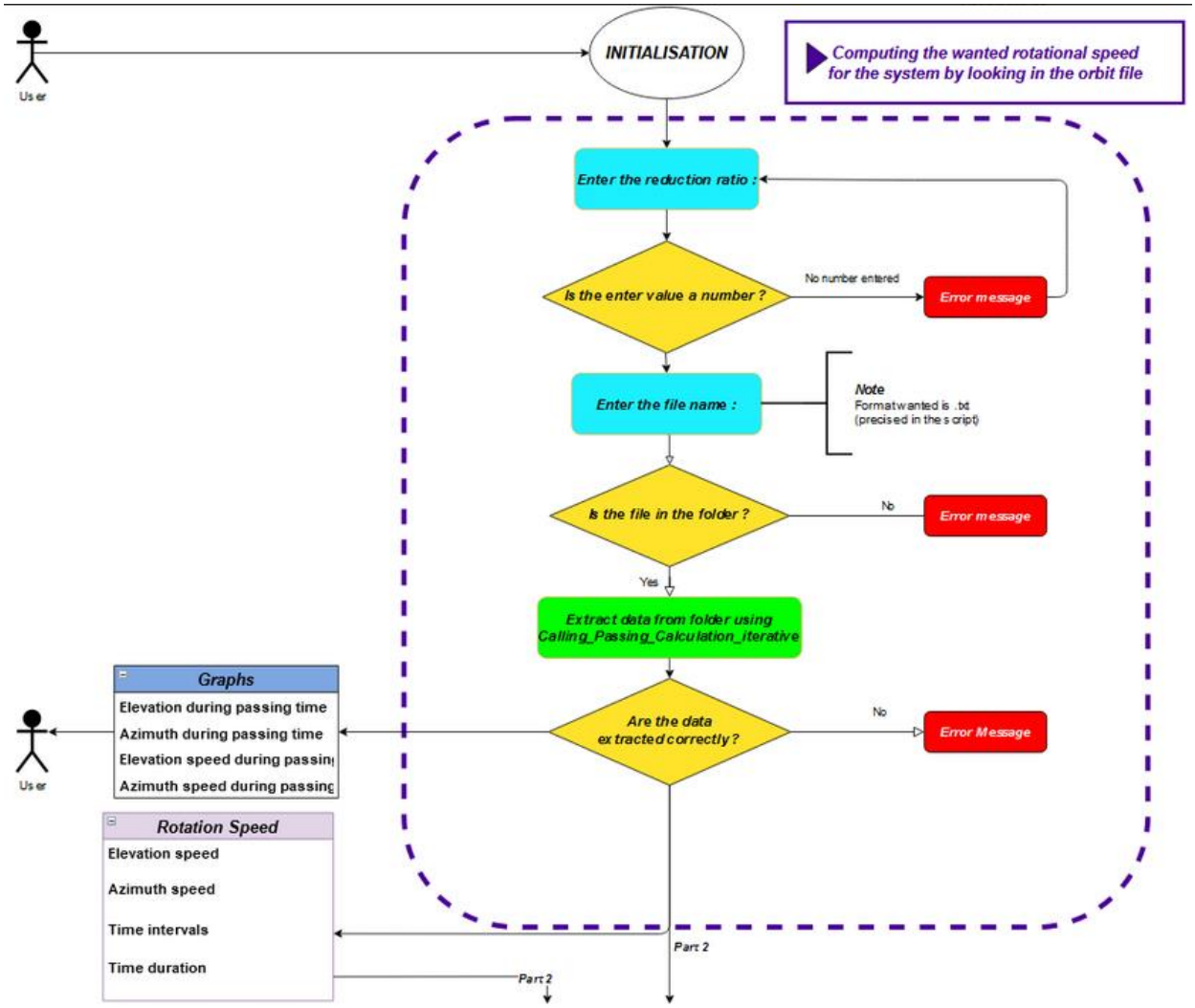


Figure V-11 : User/program interaction during the first phase of the tracking program

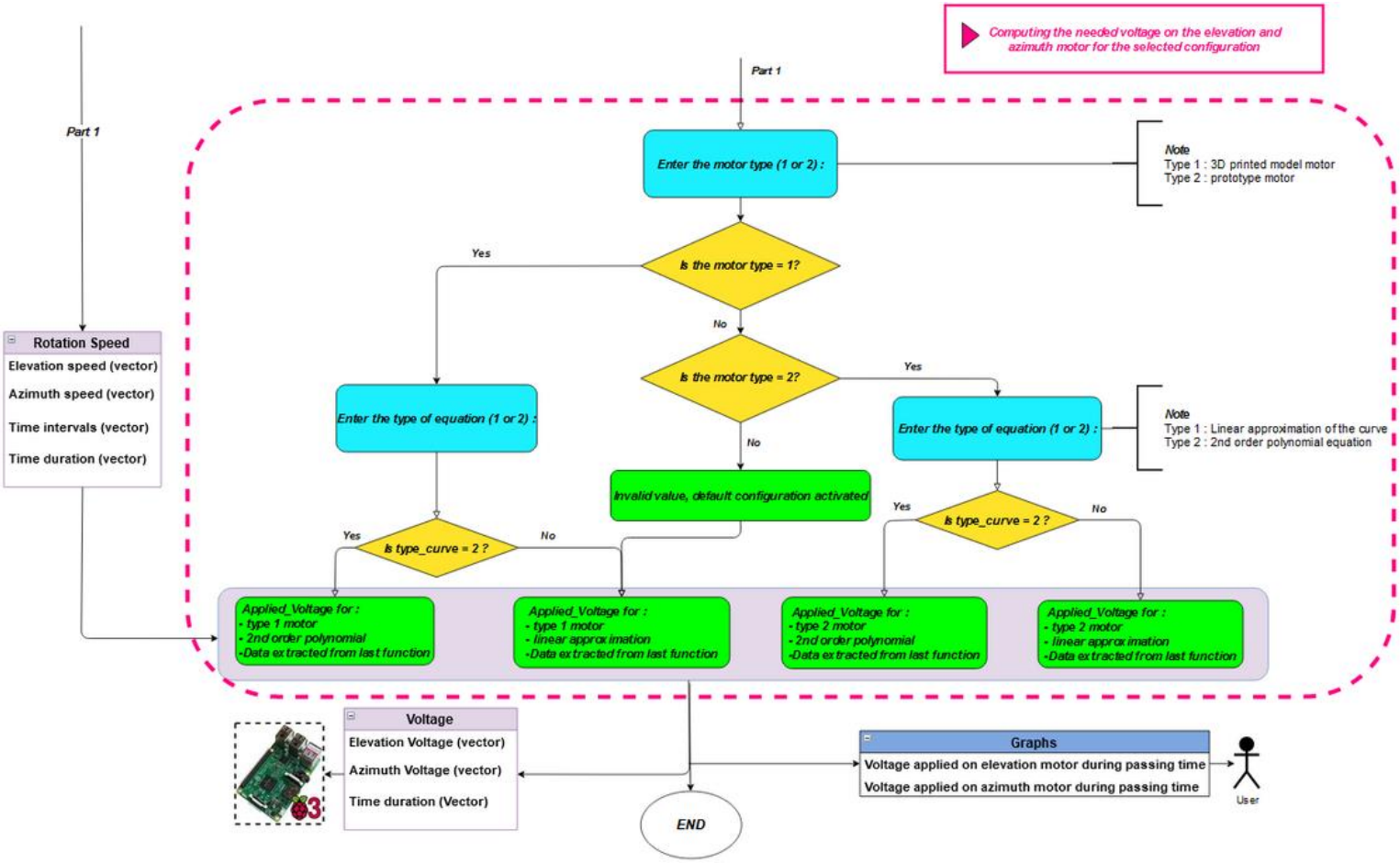


Figure V-12 : User/program interaction during the second phase of the tracking program

The operating process is the same as V1, but the modularity has increased (file selection, orbit, fail-safe mode, etc.). A test of the V3 is available in the VI.3B.3 section.

V.4 Embedded system transition/interface

Regarding the interface responsible to carry the results from the pseudo tracking program, the function used is in the RPi files. In the real case situation, the data used by this function shall come from the pseudo-tracking program. For a test situation, a set of data matching the format produced by the pseudo-tracking program is used. However, this function was not finished. The current function affects the pin for PWM generation and can send impulse to control the voltage of the motors (and therefore their direction), what needs to be added is an iterative loop that regulates the voltage based on the input data received from “Voltage_Regulation_for_GS_Operator” to assess that the interface is achieved.

V.5 Remark on the fail-safe mode

Due to the use of a PWM signal to regulate the voltage amplitude, no further considerations were brought to the subject. Nevertheless, for the development of the prototype, having a physical solution to prevent the power spikes such as fuses, etc to protect the embedded system and the motors will be necessary.

VI. Conclusion

VI.1 State of the mission at the end of the project

This project arriving to its end, it is necessary to evaluates the tasks filled and the deliverables produced for the next person that will continue the rotor design. From the situation described in I.3.1, the improvements made are the following:

1- The mechanical study of the proposed motors is achieved, and the 3D-printed model has now a full working structure. The GM4632-370 motor has been tested thanks to the mechanical and electrical study. In addition, it has been mounted to the whole structure with additional 3D-printing of support and tested with mock antennas (also printed) on the axis.

Figure VI-1 : 3D-printed model ready for testing

2- The RPi was plugged and tested with a configuration that allows at least the control of one motor. The control of this motor can be then automated or remotely from the GS.

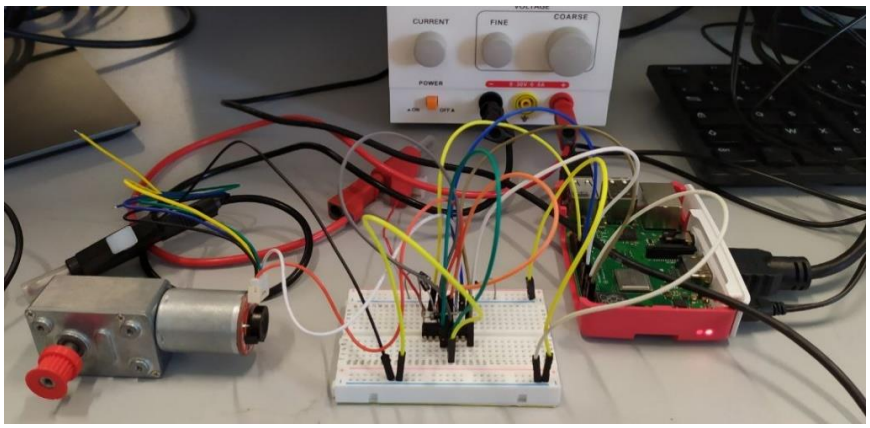


Figure VI-2 : RPi controlling the motor rotational speed

3- The pseudo tracking program that regulates the motor's tension is complete for the tracking phase. The program can simulate the motor's actions on the system using its voltage-speed characterisation to spare testing time when the data are provided. It is highly modulable to easily change the file used for data extraction, the reduction ratio of the system and the type of motor used. It also disposes a version with all parameters pre-defined for more simple applications.

Finally, a complete diary, pictures, and videos of tests, budgets, computing spreadsheets, etc, were provided with detailed explanations in both French (CSUM's official language) and English for the next person carrying the mission. The complete project will also be available on GitHub[20].

VI.2 Next improvements

As mentioned previously, the mission duration exceeds this project's duration. Therefore, further operations are necessary to pursue the project and end the mission. Improvements on the tasks done are also necessary. Following the same structure as VI.1, the next operations for the improvements of the developed parts are:

1- Once the first selected driver is received, the initial selected electronic configuration using the Power SO20 and the L6506 can be set on the 3D-printed model. Connecting the new electronic configuration, the motors and the Raspberry Pi will be fast as their position on the model has already been determined and design.
2- Regarding the interface between the RPi and the GS computer, the information produced by the Voltage_Regulation_for_GS_Operator.py (i.e., the voltage values across time in both elevation and azimuth) is not communicated to the motor control program. Therefore, the RPi needs another additional function that will read the results of

[20] See the link in the appendix

Voltage_Regulation_for_GS_Operator.py and produce the PWM that regulates the voltage assigned to each motor in real time. This can also be done by modifying the already existing function: Test_moteur_RPi.py by adding an iterative loop using the time scale given by Voltage_Regulation_for_GS_Operator.py and using each of its component in the 'sleep' function.

3- For the very pseudo tracking program, it can be adapted to fit any motor modifying the Applied_voltage.py function that gives the equation of the motor. The whole program can also be reduced to have predesigned parameters and spare manual input time once the system has proven functional. Nevertheless, the most important feature to add is the pre and post positioning prior and after the tracking to ensure that the tracking begins at the right coordinates. Another way to solve this problem is to add direct tracking or re-do Gpredict computation; this solution is by far more complicated considering the current program's structure. Finally, a new function must be coded to prevent voltage peak, it existed at the first draft of the program but was

Finally, the mechanical study shall be resumed by selecting a new motor for test phase and resume the structural design of the transmission system.

VI.3 Overall view of the project

This project brought significant contribution to the mission ordered y the CSUM. Starting from a list of needs and constraints, this translated those on requirements for the various WP that composed the project. Several advancements have been made and while improvements are still necessary, the next team to carry the project will dispose of strong material to pursue the development of the mission of designing a new rotor, CUM-made, to help ROBUSTA-3A in its mission of Mediterranean weather monitoring.

REFERENCES

Section II :

[GitHub - Courtin, K. (2020)] Website, GitHub page, last consulted in April 2021:

https://github.com/KenZone51/Rotor_for_satellite_tracking

[Kingston, J. (2020)] *Space System Engineering* course, provided at Cranfield University in October 2020.

Section III :

[Chevalier, A. (2004)] *Guide du dessinateur industriel,* **Hachette Technique**, p.297-305.

[Defauchy, D (2020)] *Mécanismes : Vitesse et accélération -Lois entrée/sortie,* provided as open class material. Available at : http://www.cpge-sii.com/

[iutenligne.net] Website, public university class on Inertial Transfer, last consulted in May 2021:

http://public.iutenligne.net/mecanique/mecanique-du-solide/charbonnieras/mecanique/1476_transfert_dinertie.html

Section IV:

[Adafruit] Website, last accessed in August 2021:

https://learn.adafruit.com/adafruit-raspberry-pi-lesson-9-controlling-a-dc-motor

[Alldatasheet (www)] Website, last accessed in July 2021:

https://www.alldatasheet.fr/datasheet-pdf/pdf/26889/TI/L293DNE.html

[Electronicwings (www)] Website, last accessed in August 2021:

https://www.electronicwings.com/raspberry-pi/raspberry-pi-pwm-generation-using-python-and-c

Section V:

[Aranacorp (www)] Website, last accessed in August 2021:

https://www.aranacorp.com/en/control-a-dc-motor-with-raspberry-pi/

[Pantechsolutions (www)] Website, last accessed in August 2021:

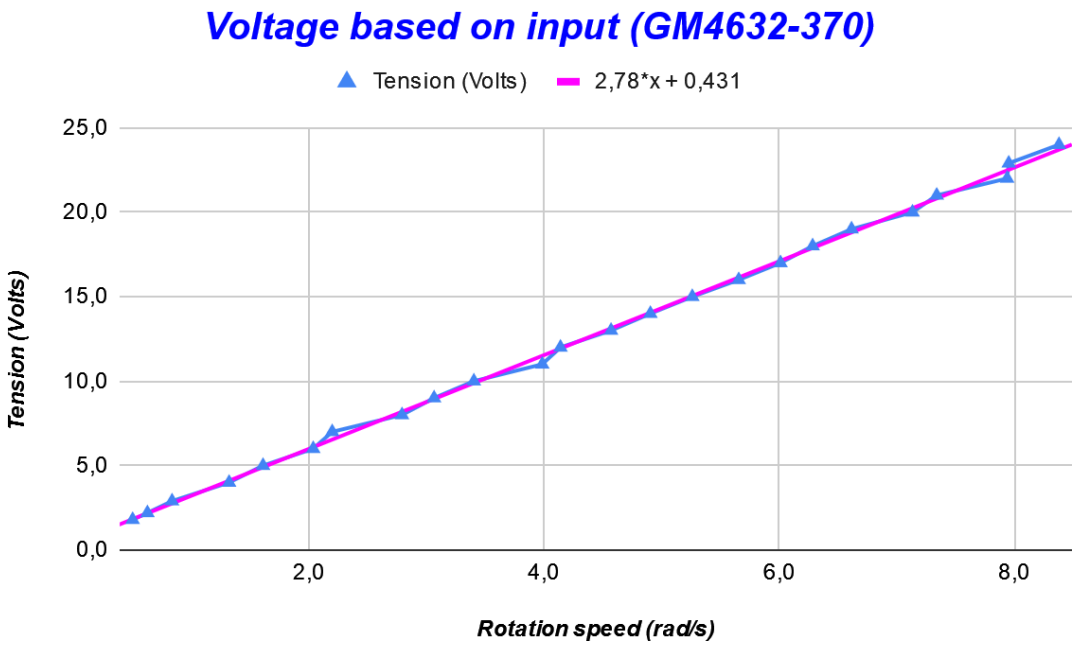
https://www.pantechsolutions.net/pwm-based-dc-motor-speed-control-using-raspberry-pi

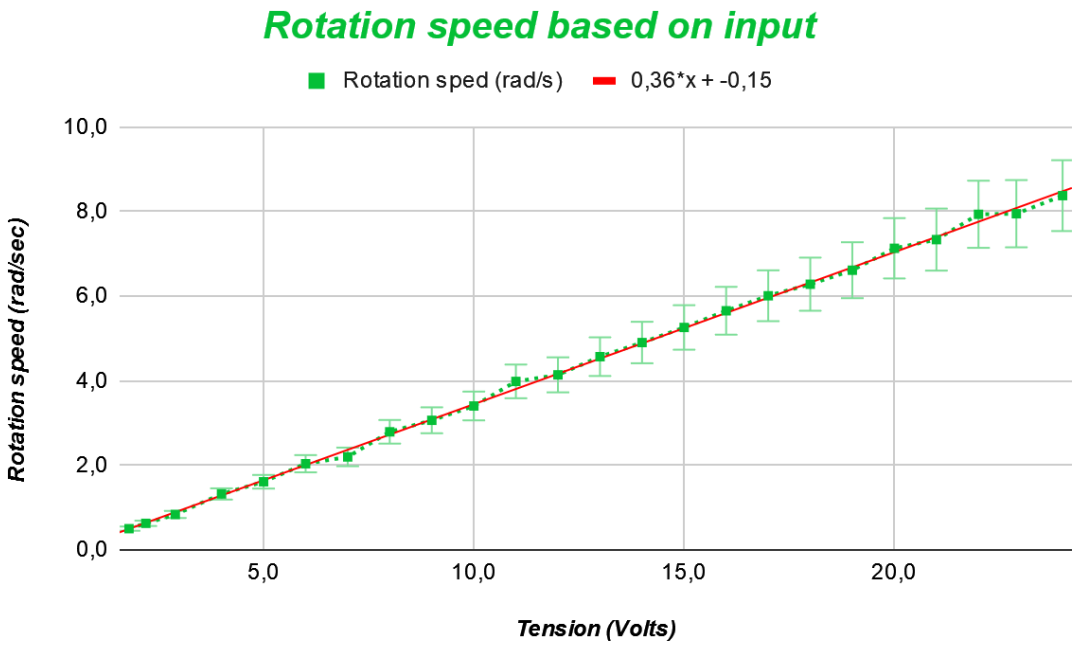
APPENDICES

In this appendix will be found additional material used for the project: the results of the motor tests and the explained code for the pseudo-tracking program and its connected functions.

Appendix A Motor tests results

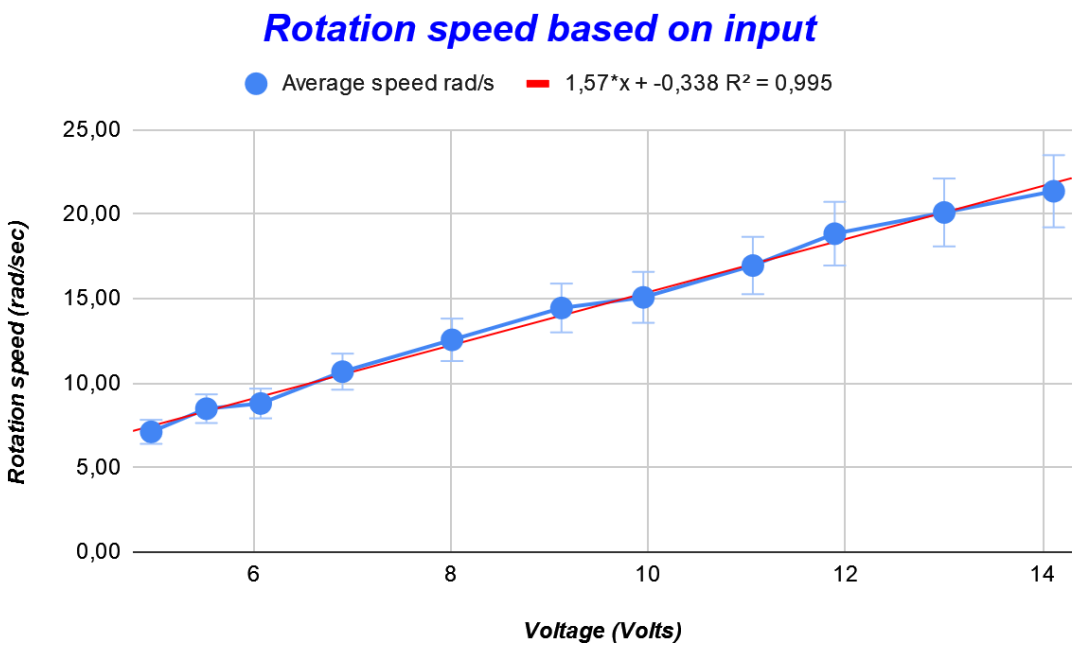
A.1 GM4632-370

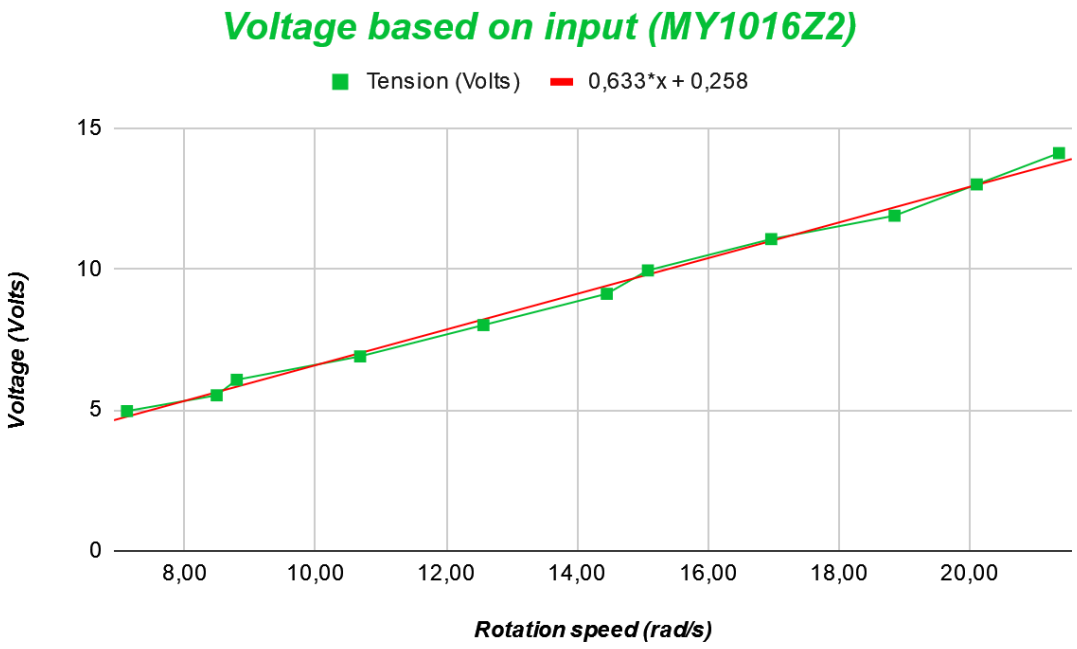




Power		*Rotation speed*					
Current (A)	Voltage (V)	rotations	seconds	Average rpm	Average radian/s	Reduction factor necessary to reach 0,001408 rad/s	Reduction factor necessary to reach 0,004255419894 rad/s
0,02	1,8	1,0	12,5	4,8	0,5	357	118
0,02	2,2	2,0	20,0	6,0	0,6	446	148
0,02	2,9	2,0	15,0	8,0	0,8	595	197
0,02	4,0	4,0	19,0	12,6	1,3	939	311
0,03	5,0	5,0	19,5	15,4	1,6	1144	379
0,03	6,0	6,0	18,5	19,5	2,0	1447	479
0,03	7,0	7,0	20,0	21,0	2,2	1562	517
0,04	8,0	10,0	22,5	26,7	2,8	1983	656
0,04	9,0	10,0	20,5	29,3	3,1	2177	720
0,04	10,0	13,0	24,0	32,5	3,4	2417	800
0,04	11,0	13,0	20,5	38,0	4,0	2830	936
0,04	12,0	14,0	21,3	39,5	4,1	2940	973
0,04	13,0	16,0	22,0	43,6	4,6	3245	1074
0,04	14,0	16,0	20,5	46,8	4,9	3483	1152
0,04	15,0	18,0	21,5	50,2	5,3	3736	1236
0,04	16,0	18,0	20,0	54,0	5,7	4016	1329
0,04	17,0	22,0	23,0	57,4	6,0	4268	1412
0,05	18,0	20,0	20,0	60,0	6,3	4462	1477
0,05	19,0	20,0	19,0	63,2	6,6	4697	1554
0,05	20,0	21,0	18,5	68,1	7,1	5066	1676
0,05	21,0	40,0	34,3	70,1	7,3	5212	1724
0,05	22,0	24,0	19,0	75,8	7,9	5637	1865
0,05	22,9	31,0	24,5	75,9	8,0	5646	1868
0,05	24,0	40,0	30,0	80,0	8,4	5950	1969

A.2 MY1016Z2





Similarly, to the test done on the GM4632-370, the above curve was used in the pseudo-tracking program to simulate this motor (see section B.2 of the appendix).

Power		*Rotation speed*					
Current (A)	Voltage (V)	rotations	seconds	rpm	Average rpm	Average radian/s	Reduction factor necessary to reach 0,001408 rad/s
0,67	4,07	4	4,2	57		5,98	4250
0,75	4,96	26	23	68	68	7,12	5055
		37	33	67			
		47	41	69			
		51	45	68			
0,77	5,52	14	10	84	81	8,49	6029
		33	25	79,2			
		40	30	80			
0,79	6,07	14	10	84	84	8,80	6247
		28	20	84			
		42	30	84			
0,82	6,9	17	10	102	102	10,68	7586
		34	20	102			
		51	30	102			
0,86	8,01	20	10	120	120	12,57	8925
		40	20	120			
		60	30	120			
0,89	9,12	23	10	138	138	14,45	10264
		46	20	138			
		69	30	138			
0,92	9,95	24	10	144	144	15,08	10710
		48	20	144			
		72	30	144			
0,96	11,06	27	10	162	162	16,96	12049
		54	20	162			
		81	30	162			
0,98	11,89	30	10	180	180	18,85	13387
		60	20	180			
		90	30	180			
0,99	13	32	10	192	192	20,11	14280
		64	20	192			
		96	30	192			
1,02	14,11	34	10	204	204	21,36	15172
		68	20	204			
		102	30	204			

Appendix B Detailed code

B.1 Computing the average orbit time and angular speeds

This function was used as a comparative mean for the given average orbit time provided at the beginning of the project. It turned out that the result obtained by sounding all the orbits (instead of 10 orbits) of a file gave a orbit time faster by more than a minute (which also impacts the rotational speed).

The function opens the “ROBUSTA-1B-passes.txt” file and extract in the orbit list (see ***Figure V-1***) the beginning and ending time of each orbit to compute their respective orbit time and averages it.

Passing_Calculation.m

```
% CThis program aims to compute the mean time where the satellite Robusta
% 1-B is in sight of the Groud Segment. And to extract the
% This satellite gives a sample to test as it is one of the satellites that
% will transmit to the antennas carried by the system



%this is the file that will be read by the function, between two different
%files, the format must remain the same

p = importdata('Passing_data_extracted.txt');
% this file has been used for
% the test at one iteration

passingtime_array = cell2mat(p);

p1 = passingtime_array(1,44:51);

%% Computing the average passing time for Robusta 1-B


% % initialize variables
NUM_SECONDS_PER_DAY = 86400.0;
% % timeStrings = {'14:54:25';'14:54:25';'14:54:25';'14:54:26';'14:54:26'};
% % convert times to fractional days using datenum
% timeFractionalDays = datenum(p1);
% % leave only the part with the most recent day fraction
% timeDayFraction = mod(timeFractionalDays,1);
% % multiply by number of seconds in a day
% timeInSeconds = timeDayFraction .* NUM_SECONDS_PER_DAY
```

Passing_Calculation.m

```
31
32 -    stop = size(p,1);
33 -    time_of_passing = [];
34
35 -   for i = 1:stop
36
37 -    timeFractionalDays = datenum(passingtime_array(i,44:51));
38      % leave only the part with the most recent day fraction
39 -    timeDayFraction = mod(timeFractionalDays,1);
40      % multiply by number of seconds in a day
41 -    timeInSeconds = timeDayFraction .* NUM_SECONDS_PER_DAY ;
42 -    time_of_passing(i+1) = timeInSeconds;
43
44 -   end
45
46
47 -    o = time_of_passing(:);
48
49 -    o_median = median(o); %this function aims to know what is the median value
50      % of the passing time to ensure there is no value that modify too much the average
51
52 -    standard_deviation_passing_time_seconds = std(o);
53 -    average_passing_time_seconds = mean(time_of_passing);
54
55      % I tried using 'quorem' function but it didn't work : problem of variable type
56
57      % For the average passing time expressed in minutes
58
59 -    apt_seconds_remaining = rem(average_passing_time_seconds, 60) ;
60 -    apt_minutes_remaining = (average_passing_time_seconds - apt_seconds_remaining)/60 ;
```

Passing_Calculation.m

```
61
62 -    apt_s = num2str(apt_seconds_remaining) ;
63 -    apt_M = num2str(apt_minutes_remaining) ;
64
65 -    average_passing_time = ['The average passing time of the satellite is : ',apt_M, ' minutes and ',apt_s,' seconds' ]
66
67      % For the standard deviation expressed in minutes
68
69 -    std_seconds_remaining = rem(standard_deviation_passing_time_seconds, 60) ;
70 -    std_minutes_remaining = (standard_deviation_passing_time_seconds - std_seconds_remaining)/60 ;
71
72 -    std_s = num2str(std_seconds_remaining) ;
73 -    std_M = num2str(std_minutes_remaining) ;
74
75 -    standard_deviation = ['The average standard deviation of the satellite passing time is : ',std_M, ' minutes and ',std_s,' seconds' ]
76
77
78
79      %% Computing the average angular speed for both azimuth and elevation
80
81 -    d = importdata('Rotation_data_Robusta1B_extracted.txt')
82
83
84      % For elevation angle
85 -    elevation_angle = d(:,1);
86 -    mean_elevation_angle = mean(2 * elevation_angle);
87 -    mean_elevation_angle_radians = deg2rad(mean_elevation_angle);
88
89 -    mean_elevation_rpm = (mean_elevation_angle / 360) / (average_passing_time_seconds/60); %speed in rpm
90 -    mean_elevation_dpm = (mean_elevation_angle) / (average_passing_time_seconds/60); %speed in deg/min
```

Passing_Calculation.m

```
83
84      % For elevation angle
85 -    elevation_angle = d(:,1);
86 -    mean_elevation_angle = mean(2 * elevation_angle);
87 -    mean_elevation_angle_radians = deg2rad(mean_elevation_angle);
88
89 -    mean_elevation_rpm = (mean_elevation_angle / 360) / (average_passing_time_seconds/60); %speed in rpm
90 -    mean_elevation_dpm = (mean_elevation_angle) / (average_passing_time_seconds/60); %speed in deg/min
91 -    mean_elevation_rotationnal_speed = mean_elevation_angle_radians / average_passing_time_seconds; %speed in radians per second
92
93
94      % For azimuth angle
95 -    azimuth_angle_bot = d(:,2);
96 -    azimuth_angle_eot = d(:,3);
97
98 -    mean_azimuth_angle = mean(abs((azimuth_angle_eot - azimuth_angle_bot)));
99 -    mean_azimuth_angle_radians = deg2rad(mean_azimuth_angle);
100
101
102 -    mean_azimuth_rpm = (mean_azimuth_angle / 360) / (average_passing_time_seconds/60); %speed in rpm
103 -    mean_azimuth_dpm = (mean_azimuth_angle) / (average_passing_time_seconds/60); %speed in deg/min
104 -    mean_azimuth_rotationnal_speed = mean_azimuth_angle_radians / average_passing_time_seconds; %speed in radians per second
105
106
107
108 -    fprintf('The average elevation rotation during the tracking is of the satellite is : %f  rpm or %f in deg/min or %f in rad/sec \n' ,...
109          mean_elevation_rpm , mean_elevation_dpm, mean_elevation_rotationnal_speed)
110 -    fprintf('The average elevation rotation during the tracking is of the satellite is : %f rpm or %f in deg/min or %f in rad/sec \n',...
111          mean_azimuth_rpm, mean_azimuth_dpm, mean_azimuth_rotationnal_speed)
```

```
>> Passing_Calculation

average_passing_time =

    'The average passing time of the satellite is : 9 minutes and 14.6906 seconds'


standard_deviation =

    'The average standard deviation of the satellite passing time is : 2 minutes and 42.1286 seconds'
```

```
The average elevation rotation during the tracking is of the satellite is : 0.013444  rpm or 4.839743 in deg/min or 0.001408 in rad/sec
The average elevation rotation during the tracking is of the satellite is : 0.040636 rpm or 14.629056 in deg/min or 0.004255 in rad/sec
```

B.2 First version (MATLAB V1)

Calling function – Fitting_current_to_motor.m:

```
%% Fitting_current_to_motor

% In this fonction, we will monitor the power input given to the motor
% using the given rotation speed and reduction ratio


%Test file : Time_sample_test_for_Passing_Calculation_iterative.txt

prompt1 = 'Enter the reduction ratio : '
p1 = input(prompt1)
prompt = 'Name your datafile with its extension (eg, blablabla.txt): '

prompt2 = ' File for the predicted coordinates of the satellite :'
x2 = input(prompt2,'s')
s2 = importdata(x2)


%% First, we compute the predicted rotational speed of both elevation and azimuth angle

[w_e,w_a, elevation_angle,azimuth_angle,Time_intervals,Time_duration ,...
    Preposition_elevation,Preposition_azimuth, Postposition_elevation,Postposition_azimuth]...
    = Calling_Passing_Calculation_iterative(x2)




%% Now that we have the rotation speed for both elevation and azimuth,
% we use the datasheet containing the motor's regime to find the best to
% use with the current reduction ratio

choice1 = 'Which motor do you wish to use (i.e., which model ?):\n - for GM4632-370, enter 1\n - for MY1016Z2, enter 2\n'
choice2 = 'Which trend curve do you want to use to approximate your values ? Choose between linear (type 1) or 2nd order polynomial (type 2)\n'
motor_type = input(choice1)
curve_type = input(choice2)


if motor_type==1
    if curve_type==2

%     Launch the function that matches the rotation speed to the reduction
%     ratio and the motor's regime

        [Voltage_elevation Voltage_azimuth Time_duration] = Applied_Voltage(p1,1,w_e,w_a,2,Time_duration)

    else
        [Voltage_elevation Voltage_azimuth Time_duration] = Applied_Voltage(p1,1,w_e,w_a,1,Time_duration)

    end

elseif motor_type == 2

%     Launch the function that matches the rotation speed to the reduction
%     ratio and the motor's regime
    if curve_type == 2

%     Launch the function that matches the rotation speed to the reduction
%     ratio and the motor's regime

        [Voltage_elevation Voltage_azimuth Time_duration] = Applied_Voltage(p1,2,w_e,w_a,2,Time_duration)

    else
        [Voltage_elevation Voltage_azimuth Time_duration] = Applied_Voltage(p1,2,w_e,w_a,1,Time_duration)

    end

else

    prompt = 'Default configuration selected (type 1)'
    [Voltage_elevation Voltage_azimuth Time_duration] = Applied_Voltage(p1,1,w_e,w_a,1,Time_duration)

end


% note : need to define the format of the motor's data (I can chose this
% one it is not imposed)
```

Function 1 – Calling_Passing_Calculation_iterative.m:

```
% CThis program aims to compute the mean time where the satellite Robusta
% 1-B is in sight of the Groud Segment. And to extract the
% This satellite gives a sample to test as it is one of the satellites that
% will transmit to the antennas carried by the system

function [e1,e2,a1,new_a2,rotational_elevation_speed_rps,rotational_azimuth_speed_rps,dt] = Passing_Calculation_iterative(Passing,j)

%this is the file that will be read by the function, between two different
%files, the format must remain the same

p = importdata(Passing)
% this file has been used for the test at one iteration

% note : this is not optimized, at every iteration the function will load
% again the txt file, I might change it later to only compute the
% derivative and extract the datas beforehand

passingtime_array = cell2mat(p(5:size(p)))

p1 = passingtime_array(j,13:21)
p2 = passingtime_array(1+j,13:21)

%% Computing the average passing time for Robusta 1-B

% To adapt to the new situation we will keep the same format for the
% passing file. Nevertheless, the passing duration will be replaced by data extracted for ROBUSTA-1A-passes.txt file
% The difference between this code and Passsing_Calculation is that it will
% compute the rotationnal speed using derivation between the two closest
% time element

% % initialize variables
NUM_SECONDS_PER_DAY = 86400.0;
% % timeStrings = {'14:54:25';'14:54:25';'14:54:25';'14:54:26';'14:54:26'};
% % convert times to fractional days using datenum
% timeFractionalDays = datenum(p1);
% % leave only the part with the most recent day fraction
% timeDayFraction = mod(timeFractionalDays,1);
% % multiply by number of seconds in a day
% timeInSeconds = timeDayFraction .* NUM_SECONDS_PER_DAY

timeFractionalDays1 = datenum(p1);
% leave only the part with the most recent day fraction
timeDayFraction = mod(timeFractionalDays1,1);
% multiply by number of seconds in a day
TimeInSeconds1 = timeDayFraction .* NUM_SECONDS_PER_DAY ;

timeFractionalDays2 = datenum(p2);
% leave only the part with the most recent day fraction
timeDayFraction2 = mod(timeFractionalDays2,1);
% multiply by number of seconds in a day
TimeInSeconds2 = timeDayFraction2 .* NUM_SECONDS_PER_DAY ;


dt = TimeInSeconds2 - TimeInSeconds1

%% Computing the average angular speed for both azimuth and elevation


% For elevation angle

E1 = passingtime_array(j,30:35);
E2 = passingtime_array(1+j,30:35);% note : don't forget to put a size()-1 not to go over the range

% note : some values have no character, i.e., 2.05° instead of 02.05°.
% If there is a problem I'd rather look at a way to change the empty
% character with a '0'

e1 = str2double(E1)
e2 = str2double(E2)

de = e2 - e1
de_rad = deg2rad(de)

rotational_elevation_speed_dps = de/dt; %speed in deg/s;
rotational_elevation_speed_rps = de_rad/dt; %speed in radians per second


% For azimuth angle
A1 = passingtime_array(j,22:27)
A2 = passingtime_array(1+j,22:27)



a1 = str2double(A1)
```

```
a2 = str2double(A2)

if j==1
    initial_da = a2-a1
else
    A0 = passingtime_array(j-1,22:27)
    a0 = str2double(A0)
    initial_da = (a1 - a0)/dt
end
controlled_a2 = Prototype_rotation_regulation(a1, a2, initial_da, dt)


new_a2 = controlled_a2

da = new_a2 - a1


rotational_azimuth_speed_dps = da/dt; %speed in degree per seconds;
rotational_azimuth_speed_rps = deg2rad(da)/dt; %speed in radians per second



end
```

Function 2 – Passing_Calculation_iterative.m:

```
function [e1,e2,a1,new_a2,rotational_elevation_speed_rps,rotational_azimuth_speed_rps,dt] = Passing_Calculation_iterative(Passing,j)

%this is the file that will be read by the function, between two different
%files, the format must remain the same

p = importdata(Passing)
% this file has been used for the test at one iteration

% note : this is not optimized, at every iteration the function will load
% again the txt file, I might change it later to only compute the
% derivative and extract the datas beforehand

passingtime_array = cell2mat(p(5:size(p)))

p1 = passingtime_array(j,13:21)
p2 = passingtime_array(1+j,13:21)

%% Computing the average passing time for Robusta 1-B

% To adapt to the new situation we will keep the same format for the
% passing file. Nevertheless, the passing duration will be replaced by data extracted for ROBUSTA-1A-passes.txt file
% The difference between this code and Passsing_Calculation is that it will
% compute the rotationnal speed using derivation between the two closest
% time element

% % initialize variables
NUM_SECONDS_PER_DAY = 86400.0;

timeFractionalDays1 = datenum(p1);
% leave only the part with the most recent day fraction
timeDayFraction = mod(timeFractionalDays1,1);
```

```
timeDayFraction = mod(timeFractionalDays1,1);
% multiply by number of seconds in a day
TimeInSeconds1 = timeDayFraction .* NUM_SECONDS_PER_DAY ;

timeFractionalDays2 = datenum(p2);
% leave only the part with the most recent day fraction
timeDayFraction2 = mod(timeFractionalDays2,1);
% multiply by number of seconds in a day
TimeInSeconds2 = timeDayFraction2 .* NUM_SECONDS_PER_DAY ;

dt = TimeInSeconds2 - TimeInSeconds1

%% Computing the average angular speed for both azimuth and elevation


% For elevation angle

E1 = passingtime_array(j,30:35);
E2 = passingtime_array(1+j,30:35);% note : don't forget to put a size()-1 not to go over the range

% note : some values have no character, i.e., 2.05° instead of 02.05°.
% If there is a problem I'd rather look at a way to change the empty
% character with a '0'

e1 = str2double(E1)
e2 = str2double(E2)

de = e2 - e1
de_rad = deg2rad(de)
```

```
de = e2 - e1
de_rad = deg2rad(de)

rotational_elevation_speed_dps = de/dt; %speed in deg/s;
rotational_elevation_speed_rps = de_rad/dt; %speed in radians per second

% For azimuth angle
A1 = passingtime_array(j,22:27)
A2 = passingtime_array(1+j,22:27)

a1 = str2double(A1)
a2 = str2double(A2)

if j==1
    initial_da = a2-a1
else
    A0 = passingtime_array(j-1,22:27)
    a0 = str2double(A0)
    initial_da = (a1 - a0)/dt
end
controlled_a2 = Prototype_rotation_regulation(a1, a2, initial_da, dt)

new_a2 = controlled_a2

da = new_a2 - a1

rotational_azimuth_speed_dps = da/dt; %speed in degree per seconds;
rotational_azimuth_speed_rps = deg2rad(da)/dt; %speed in radians per second

end
```

Function 3 – Prototype_rotation_regulation.m:

This function aims to correct the discontinuities while computing the angular speed due to the completion of one rotation. The process is simple, adding or removing 360° once a rotation is completed in a direction or its opposite. The new values are then return to Function 2 to compute the angular speed. These discontinuities only occur in azimuth as the elevation values are strictly comprised in the interval { 0°-90°; 90°-0° }.

```
function [new_angle2] = Prototype_rotation_regulation(angle1, angle2, previous_derivative, dt)

derivative = (angle2-angle1)/dt


if (previous_derivative > 0 && derivative < 0) || (previous_derivative < 0 && derivative > 0)

    if (previous_derivative > 0 && derivative < 0)
%     If angle0 = 290° , angle1 = 355° and angle2 = 3° we need to change
%     the values of angle2 to keep the same derivative sign and keep the
%     angle function monotonous and consider the rotation that has been made.
%     So angle2 = angle2 + 360.
        new_angle2 = angle2 + 360

    elseif (previous_derivative < 0 && derivative > 0)
%     If angle0 = 5° , angle1 = 0.5° and angle2 = 357° we need to change
%     the values of angle2 to keep the same derivative sign and keep the
%     angle function monotonous
        new_angle2 = angle2 - 360
    end
else

    new_angle2 = angle2
end

end
```

Function 4 – Applied_Voltage:

This function computes the voltage to be applied on each motor given the wanted rotational speed and the reduction ratio of the system and return the Voltage and the associated period inside 3 distinct vectors.

```
function [U_elevation,U_azimuth,time_duration] = Applied_Voltage(reduction_ratio,motor_type,elevation_speed,azimuth_speed,curve,time_duration)

a = elevation_speed'
b = azimuth_speed'
w_axis = [a,b]
w_motor = reduction_ratio * w_axis

if motor_type==1

    if curve==2

        U_2nd_order_polynomial = 0.565 + 2.69 * w_motor + 0.0103 * ((w_motor).^2)

        V = U_2nd_order_polynomial

    else

        U_linear =  2.78 * w_motor + 0.431

        V = U_linear

    end

elseif motor_type==2

    if curve==2

         U_2nd_order_polynomial = 1.42 + 0.0448 * w_motor + 6.52*(10^(-3)) * ((w_motor).^2)

         V = U_2nd_order_polynomial

     else

         U_linear =  0.633 * w_motor + 0.258

         V = U_linear

     end
else

     prompt = "Enter a valid value next time!\n Default value set to GM motor with a linear curve approximation\n"

```

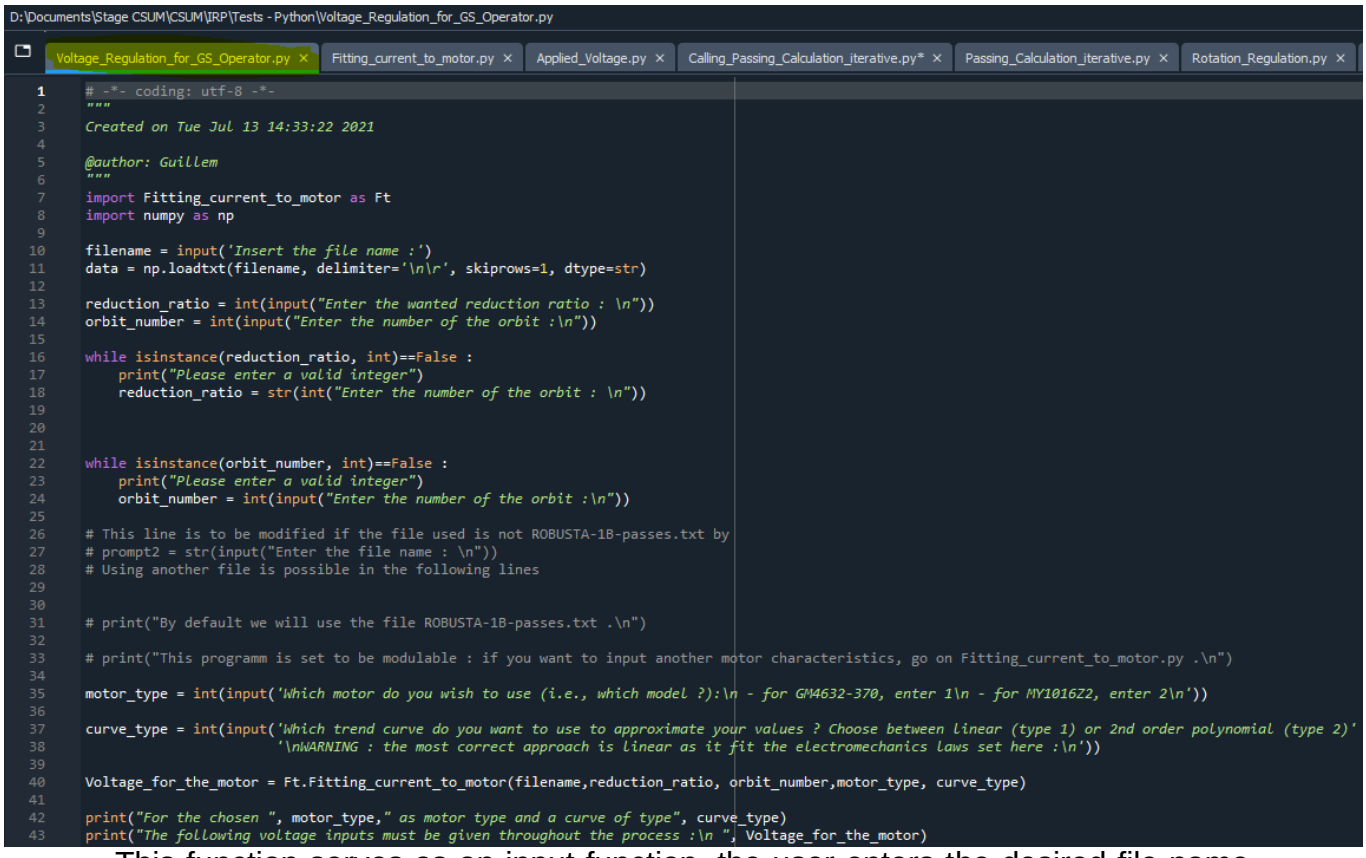
```
        U_linear =  2.78 * w_motor + 0.431

        V = U_linear


    end

    % Voltage = [V time_duration(2:end-1)]

    % (1:(size(time_duration)-1))

    U_elevation = V(:,1)

    U_azimuth = V(:,2)

    figure(5)
    plot(time_duration(1:end-1), U_elevation,'r-o')
    xlabel('Time duration (seconds)')
    ylabel('Voltage for elevation motor (Volts)')
    title('Voltage applied to the elevation motor during the satellite passing')

    figure(6)
    plot(time_duration(1:end-1), U_azimuth,'b-o')
    xlabel('Time duration (seconds)')
    ylabel('Voltage for elevation motor (Volts)')
    title('Voltage applied to the azimuth motor during the satellite passing')

    figure(7)
    stairs(time_duration(1:end-1), U_elevation,'r-o')
    xlabel('Time duration (seconds)')
    ylabel('Voltage for elevation motor (Volts)')
    title('Voltage applied to the elevation motor during the satellite passing without repositionning')

    figure(8)
    stairs(time_duration(1:end-1), U_azimuth,'b-o')
    xlabel('Time duration (seconds)')
    ylabel('Voltage for elevation motor (Volts)')
    title('Voltage applied to the azimuth motor during the satellite passing without repositionning')

    end
```

B.3 Complete test version (Python V3)

Despite a relative similarity in the coding syntax, the function used might differ from MATLAB. The ways used to achieve the same task then differs, especially regarding the file reading and data extraction. As an example, the way of returning vectors from a function in Python is not as modular as extracting a vector in MATLAB. The way of treating data from file is also different, which explains the increased amount of small function in the Python program.

Therefore, some functions have been changed between MATLAB and Python (e.g., Passing_Calculation_iterative) and other have been added. Such as Voltage_Regulation_for_GS_Operator.py that in Python replaces Fitting_current_to_motor.mat as calling function; Fitting_current_to_motor.py being a function that embodies the results of the program and is called by Voltage_Regulation_for_GS_Operator.py.

Calling function - Voltage_regulation_for_GS_operator.py:



This function serves as an input function, the user enters the desired file name (which is the same format as the one presented in ***Figure V-1***), the wanted

reduction ratio, orbit number to be found using the file name entered, motor type (“1” for the 3D-printed model, “2” for the prototype model) and finally the curve type.

```
Insert the file name :ROBUSTA-1B-passes.txt

Enter the wanted reduction ratio :
880

Enter the number of the orbit :
15507

Which motor do you wish to use (i.e., which model ?):
 - for GM4632-370, enter 1
 - for MY1016Z2, enter 2
1

Which trend curve do you want to use to approximate your values ? Choose
between linear (type 1) or 2nd order polynomial (type 2)
WARNING : the most correct approach is linear as it fit the
electromechanics laws set here :
1
```

Once every parameter is entered, the function calls Fitting_current_to_motor.py, that will return the voltage data on the rotational axis.

Function 1 - Fitting_current_to_motor.py:

Test_moteur_RPi.py × | Applied_Voltage.py × | Voltage_Regulation_for_GS_Operator.py × | Fitting_current_to_motor.py ×

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 12 15:36:25 2021

@author: Guillem
"""

""" (PYTHON VERSION)
In this first part, we ask the user to enter the parameters that will
make the program extract the right orbit parameters
"""
import sys

import Calling_Passing_Calculation_iterative as CPC

import Applied_Voltage as AV

from matplotlib import pyplot as plt


def Fitting_current_to_motor(filename,reduction_ratio, orbit_number,motor_type, curve_type) :


    while isinstance(orbit_number, int)==False :
        print("Please enter a valid integer")
        orbit_number = int(input("Enter the number of the orbit :\n"))

    orbit_number_str = str(orbit_number)

    with open(filename) as f:
        if orbit_number_str in f.read():
            print("The orbit researched is located in this file")
            location = True
        else:
            print("The orbit research is NOT located is this file")
            location = False


    if location == False :

        print("The orbit is not located in the file, the program will stop now.\nWould you try another file or another orbit ? ")
        print("If 'yes', please enter 'file' or 'orbit' to continue")
        print("Otherwise, enter anything else to stop the program")

        new_command = str(input("Type your answer here :"))

        if new_command == "file":

            print("Enter the new filename. WARNING : the file must have the same format than the one produced by Gpredict")

            new_filename = str(input("Please enter the file with its extension (e.g., 'blabla.txt')"))
            new_orbit_number = int(input("Enter the orbit number you are looking for :\n"))

```

```
            while isinstance(new_orbit_number, int)==False :
                print("Please enter a valid integer")
                new_orbit_number = str(int("Enter the number of the orbit :\n"))

            new_orbit_number_str = str(new_orbit_number)

            with open(new_filename) as f:
                if new_orbit_number_str in f.read():
                    print("The orbit researched is located in this file")

                else:
                    print("The orbit research is NOT located is this file")
                    sys.exit("Wrong orbit, exiting the program." )

            orbit_number = new_orbit_number
            orbit_number_str = new_orbit_number_str
            filename = new_filename

        elif new_command == "orbit":

            new_orbit_number = int(input("Enter the new wanted orbit :"))
            new_orbit_number_str = str(new_orbit_number)

            with open(filename) as f:
                if new_orbit_number_str in f.read():
                    print("The orbit researched is located in this file")

                else:
                    print("The orbit research is NOT located is this file")

                    sys.exit("The program will stop now.")

            orbit_number = new_orbit_number
            orbit_number_str = new_orbit_number_str

        else :

            sys.exit("The program will stop now.")

    print("The orbit has been located, the data will now be extracted")

    """ (PYTHON VERSION)
    Now, we compute the predicted rotational speed of both elevation and azimuth angle

    """

    extracted_data = CPC.Calling_Passing_Calculation_iterative(filename, orbit_number_str)
    w_e = extracted_data[0]
    w_a = extracted_data[1]
    Passing_duration = extracted_data[5]
```

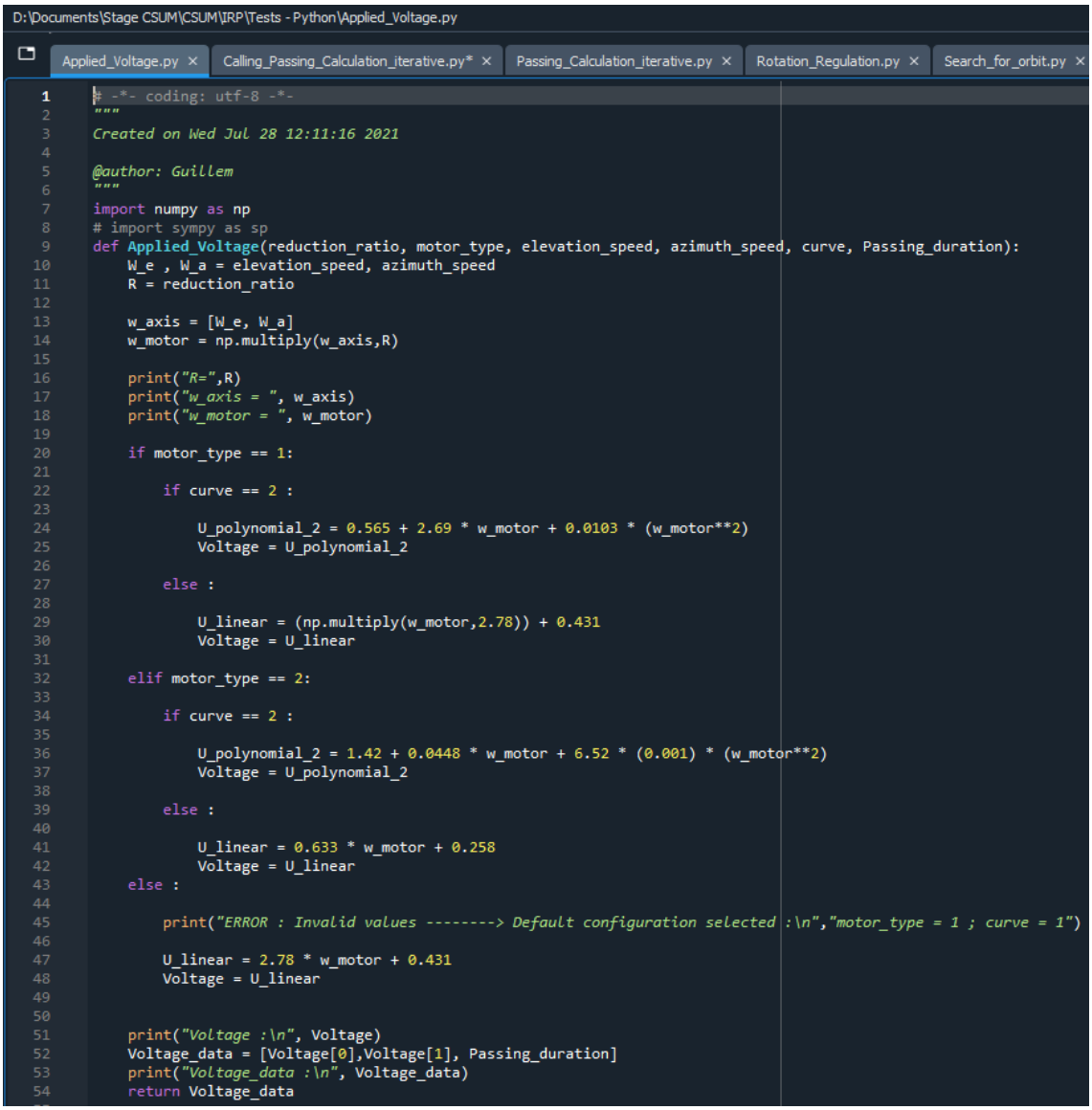
```
    """ (PYTHON VERSION)
    %% Now that we have the rotation speed for both elevation and azimuth,
    % we use the datasheet containing the motor's regime to find the best to
    % use with the current reduction ratio

    """
    if motor_type==1 :
        if curve_type==2 :

    # Launch the function that matches the rotation speed to the reduction
    # ratio and the motor's regime

            Voltage = AV.Applied_Voltage(reduction_ratio,1,w_e,w_a,2,Passing_duration)
            Voltage_elevation = Voltage[0]
            Voltage_azimuth = Voltage[1]
            Passing_duration = Voltage[2]

        else :

            Voltage = AV.Applied_Voltage(reduction_ratio,1,w_e,w_a,1,Passing_duration)
            Voltage_elevation = Voltage[0]
            Voltage_azimuth = Voltage[1]
            Passing_duration = Voltage[2]


    elif motor_type == 2 :

    # Launch the function that matches the rotation speed to the reduction
    # ratio and the motor's regime

        if curve_type == 2 :

    # Launch the function that matches the rotation speed to the reduction
    # ratio and the motor's regime


            Voltage = AV.Applied_Voltage(reduction_ratio,2,w_e,w_a,2,Passing_duration)
            Voltage_elevation = Voltage[0]
            Voltage_azimuth = Voltage[1]
            Passing_duration = Voltage[2]

        else :
            Voltage = AV.Applied_Voltage(reduction_ratio,2,w_e,w_a,1,Passing_duration)
            Voltage_elevation = Voltage[0]
            Voltage_azimuth = Voltage[1]
            Passing_duration = Voltage[2]


    else :

        print("Default configuration selected (type 1 motor with linear curve)")
        Voltage = AV.Applied_Voltage(reduction_ratio,1,w_e,w_a,1,Passing_duration)
        Voltage_elevation = Voltage[0]
        Voltage_azimuth = Voltage[1]
        Passing_duration = Voltage[2]
```

```
163     print("Elevation Voltage:\n", Voltage_elevation)
164     print("Azimuth Voltage : ",Voltage_azimuth)
165
166     plt.figure(5)
167     plt.plot(Passing_duration, Voltage_elevation ,'r-.*')
168     plt.xlabel('Time interval')
169     plt.ylabel('Elevation voltage (Volts)')
170     plt.title('Elevation voltage regulation during satellite passing')
171
172     plt.figure(6)
173     plt.plot(Passing_duration, Voltage_azimuth  ,'b-.*')
174     plt.xlabel('Time interval')
175     plt.ylabel('Azimuth voltage (Volts)')
176     plt.title('Azimuth voltage regulation during satellite passing')
177
178     plt.figure(7)
179     plt.step(Passing_duration, Voltage_elevation ,'r-.*')
180     plt.xlabel('Time interval')
181     plt.ylabel('Elevation voltage (Volts)')
182     plt.title('Elevation voltage regulation during satellite passing (step function)')
183
184     plt.figure(8)
185     plt.step(Passing_duration, Voltage_azimuth  ,'b-.*')
186     plt.xlabel('Time interval')
187     plt.ylabel('Azimuth voltage (Volts)')
188     plt.title('Azimuth voltage regulation during satellite passing (step function)')
189
190
191     return (Voltage_elevation, Voltage_azimuth, Passing_duration)
```

The above function works the same way as Fitting_current_to_motor.mat. However, it adds additional feature. First it allows the user to enter again the fill name or obit in case wrong input. Moreover, it has a default configuration for a bad choice of motor of curve.

Function 2 - Applied_Voltage.py:

No change compared to the MATLAB function in matter of purpose and values returned.




```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 28 12:11:16 2021

@author: Guillem
"""
import numpy as np
# import sympy as sp
def Applied_Voltage(reduction_ratio, motor_type, elevation_speed, azimuth_speed, curve, Passing_duration):
    W_e , W_a = elevation_speed, azimuth_speed
    R = reduction_ratio

    w_axis = [W_e, W_a]
    w_motor = np.multiply(w_axis,R)

    print("R=",R)
    print("w_axis = ", w_axis)
    print("w_motor = ", w_motor)

    if motor_type == 1:

        if curve == 2 :

            U_polynomial_2 = 0.565 + 2.69 * w_motor + 0.0103 * (w_motor**2)
            Voltage = U_polynomial_2

        else :

            U_linear = (np.multiply(w_motor,2.78)) + 0.431
            Voltage = U_linear

    elif motor_type == 2:

        if curve == 2 :

            U_polynomial_2 = 1.42 + 0.0448 * w_motor + 6.52 * (0.001) * (w_motor**2)
            Voltage = U_polynomial_2

        else :

            U_linear = 0.633 * w_motor + 0.258
            Voltage = U_linear
    else :

        print("ERROR : Invalid values --------> Default configuration selected :\n","motor_type = 1 ; curve = 1")

        U_linear = 2.78 * w_motor + 0.431
        Voltage = U_linear


    print("Voltage :\n", Voltage)
    Voltage_data = [Voltage[0],Voltage[1], Passing_duration]
    print("Voltage_data :\n", Voltage_data)
    return Voltage_data
```

Function 3 - Calling_passing_calculation_iterative.py

No change compared to the MATLAB function in matter of purpose and values returned. However, the function called to read and extract the data are different from the one from MATLAB.


```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  7 10:16:13 2021

@author: Guillem
"""
# import matlab.engine as eng

import numpy as np
from matplotlib import pyplot as plt
import Find_end_section_Tests as Fest
import Search_for_orbit as Sfo
import Passing_Calculation_iterative as PCi
import Txt_file_size as tfs

def Calling_Passing_Calculation_iterative(Passing, orbit_number_str):

    """
    Following the confirmation of the orbit location in the file in
    Fitting_current_to_motor.py, we will now look for the whole orbit data section

    i.e. the data contained until the next CRLF (all the orbit are separated
    from the others by a blank line)

    """

    fin = open(Passing, "r")
    # dot_to_comas(Passing)
    complete_file = fin.readlines()

    Passing_size = tfs.Txt_file_size(Passing)

    """
    First we look out for the position of the wanted orbit inside the orbit file
    """

    targeted_line = Sfo.Search_for_orbit(Passing,orbit_number_str)
    # Finding the orbit number inside the file also means finding the
    # beginning of its associated section


    """
    Then we look out for the end of the section : the blank line between
    our orbit and the next section
    To do that we look at the file from the line where we found the orbit
    until the next blank spot
    """

    end_of_section = Fest.Find_end_section_Tests(Passing,targeted_line,Passing_size)

    """
    Once it is done we can read the section lines corresponding to our orbit
    """
    # First we store the data of the section into a specific array

    section_data_str = []
```

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Calling_Passing_Calculation_iterative.py

Calling_Passing_Calculation_iterative.py × | Passing_Calculation_iterative.py × | Rotation_Regulation.py × | Search_for_orbit.py × | Find_end_section_

```
58
59      """
60      Here I cannot find why when I look for the end section (white line) my
61      function, Find_end_section, goes beyond the end from 7 lines...
62      TO BE IMPROVED !!!
63
64      """
65
66      for k in range((targeted_line+3) , (end_of_section)):
67
68          # +4 (therefore + 3 in Python) here is used to remove the headlines of
69          # the orbit section once it is found to leave only the data to be processed
70
71          add = complete_file[k]
72          section_data_str.append(add)
73
74      print("The data extracted are :", section_data_str)
75
76      """
77      We now have the data corresponding to the wanted orbit under the form of a
78      vector of string
79      What we will do now is to extract from these strings the values of
80      interest to compute the rotationnal speed needed for the rotor to match
81      the satellite's trajectory
82      """
83
84      # Now we use a loop to read each line of the gathered file
85
86      # As opposed as the MATLAB code, this one will use target the line before
87      # processing Passing_Calculation_iterative
88
89      Elevation_angle = []
90      Azimuth_angle = []
91      time_intervals = []
92      W_e = []
93      W_a = []
94
95      Initial_conditions = PCi.Passing_Calculation_iterative(section_data_str,0,orbit_number_str)
96
97      Elevation_angle.append(Initial_conditions[0])
98      Azimuth_angle.append(Initial_conditions[2])
99      time_intervals.append(Initial_conditions[6])
100     W_e.append(Initial_conditions[4])
101     W_a.append(Initial_conditions[5])
102
103     # Once the initial condition is done we trigger the loop
104     # The initial condition is necessary
105
106     for j in range(1,(len(section_data_str)-1)):
107
108         # Don't forget that the first 4 lines are unused for the calculation
109         # of the rotationnal speed (contains the headlines)
110
111         Computing_rotationnal_speed = PCi.Passing_Calculation_iterative(section_data_str,j,orbit_number_str)
112
```

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Calling_Passing_Calculation_iterative.py

Calling_Passing_Calculation_iterative.py | Passing_Calculation_iterative.py | Rotation_Regulation.py

```
112
113             Elevation_angle.append(Computing_rotationnal_speed[1])
114             Azimuth_angle.append(Computing_rotationnal_speed[3])
115             W_e.append(Computing_rotationnal_speed[4])
116             W_a.append(Computing_rotationnal_speed[5])
117             time_intervals.append(Computing_rotationnal_speed[6])
118
119         # W_elevation = np.deg2rad(W_e)
120         # W_azimuth = np.deg2rad(W_a)
121         # This must stay commented as the new return from Passing_calculation is
122         # the rad/s speed and not the deg/speed
123
124         time_duration = [0]
125
126         nb_points = len(time_intervals)
127
128         print("Elevation angle :\n",Elevation_angle)
129         print("Azimuth angle :\n",Azimuth_angle)
130         print("Time intervals :\n",time_intervals)
131         print("Elevation rotational speed (rad/s):\n", W_e)
132         print("Azimuth rotational speed (rad/s) :\n",W_a)
133
134         for i in range(1,nb_points):
135
136             td = time_duration[i-1] + time_intervals[i]
137             time_duration.append(td)
138
139         print("Time duration :", time_duration)
140
141         np.size(Elevation_angle)
142         np.size(Azimuth_angle)
143         np.size(W_e)
144         np.size(W_a)
145         np.size(time_duration)
146         np.size(time_intervals)
147
148         plt.figure(1)
149         plt.plot(time_duration, Elevation_angle  ,'r-.*')
150         plt.xlabel('Time interval')
151         plt.ylabel('Angular elevation position (°)')
152         plt.title('Elevation angle during satellite passing duration')
153
154         plt.figure(2)
155         plt.plot(time_duration, Azimuth_angle  ,'b-.*')
156         plt.xlabel('Time interval')
157         plt.ylabel('Angular azimuth position (°)')
158         plt.title('Azimuth angle during satellite passing duration')
159
160         plt.figure(3)
161         plt.plot(time_duration[0:len(time_duration)],W_e ,'r--o')
162         plt.xlabel('Time interval')
163         plt.ylabel('Elevation speed (deg/s)')
164         plt.title('Elevation angular speed during satellite passing duration')
```

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Calling_Passing_Calculation_iterative.py

Calling_Passing_Calculation_iterative.py × | Passing_Calculation_iterative.py × | Rotation_Regulation.py × | Search_for_orbit.py × | Find_end_

```
150         plt.xlabel('Time interval')
151         plt.ylabel('Angular elevation position (°)')
152         plt.title('Elevation angle during satellite passing duration')
153 
154         plt.figure(2)
155         plt.plot(time_duration, Azimuth_angle  ,'b-.*')
156         plt.xlabel('Time interval')
157         plt.ylabel('Angular azimuth position (°)')
158         plt.title('Azimuth angle during satellite passing duration')
159 
160         plt.figure(3)
161         plt.plot(time_duration[0:len(time_duration)],W_e ,'r--o')
162         plt.xlabel('Time interval')
163         plt.ylabel('Elevation speed (deg/s)')
164         plt.title('Elevation angular speed during satellite passing duration')
165 
166         plt.figure(4)
167         plt.plot(time_duration[0:len(time_duration)],W_a ,'b--o')
168         plt.xlabel('Time interval')
169         plt.ylabel('Azimuth speed (deg/s)')
170         plt.title('Azimuth angular speed during satellite passing duration')
171 
172         Rotation_speeds_of_rotor = [W_e,W_a,Elevation_angle, Azimuth_angle, time_intervals, time_duration]
173         return Rotation_speeds_of_rotor
174 
175         """
176         This part is to be completed after the first trials
177         In this part we will try to do the pre and post positionning phases
178 
179 
180         # el0 = Initial_conditions(0)
181         # preposition_elevation = [el0]
182         # preposition_elevation = preposition_elevation.append(el0/60)
183 
184 
185         # az0 = Initial_conditions(2)
186         # preposition_azimuth = [az0]
187         # preposition_azimuth = preposition_elevation.append(az0/60)
188 
189 
190 
191         # repositionning_elevation1 = (0 - Elevation_angle(end))
192         # repositionning_elevation2 = (360 - Elevation_angle(end))
193         # repositionning_elevation = min(repositionning_elevation1, repositionning_elevation2)
194 
195         # rotation_end_elevation = repositionning_elevation/dt
196 
197         # repositionning_azimuth1 = (0 - Azimuth_angle(end))
198         # repositionning_azimuth2 = (360 - Azimuth_angle(end))
199         # repositionning_azimuth = min(repositionning_azimuth1, repositionning_azimuth2)
200 
201         # rotation_end_azimuth = repositionning_azimuth/dt
202 
203         """
```

Function 4 - Passing_Calculation_iterative:

This function is the same as the MATLAB one

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Passing_Calculation_iterative.py

Passing_Calculation_iterative.py | Rotation_Regulation.py | Search_for_orbit.py | Find_end_section_Tests.py | Calli

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul  5 09:59:26 2021

@author: Guillem
"""
"""
CThis program aims to compute the mean time where the satellite Robusta
1-B is in sight of the Ground Segment. And to extract the
This satellite gives a sample to test as it is one of the satellites that
will transmit to the antennas carried by the system
"""

# import os
import numpy as np

# import dot_to_comas not needed if we use a certain function that replaces
# the character directly inside the data extracted

import get_sec_str as gss
import Rotation_Regulation as RR

# Passing = 'Test.txt'

def Passing_Calculation_iterative(Section_data_str,j,orbit_number_str):

    # function [e1,e2,a1,new_a2,rotational_elevation_speed_rps,
    #       rotational_azimuth_speed_rps,dt] = Passing_Calculation_iterative(Passing,j)

    # WARNING : the python version will use the string created in the upstream function
    #           instead of opening the file everytime.
    # THIS SAVES COMPUTING TIME

    """
    First we extract the various data from the targeted string :
       1) Time of passing hh:mm:ss
       2) elevation angle
       3) azimuth angle
       4) time intervals

    What we want to compute with these data are :
        1) Elevation variation
        2) Azimuth variation
        3) Elevation Speed
        4) Azimuth Speed
        5) Duration of passing
    """

    #  what we do first is to call the two line that we
    #  want to compute the wanted values
    #  If Possible I use Numpy to extract the character on specific position as substring

    print("j = ",j)
    p1 = Section_data_str[j]
    p2 = Section_data_str[j+1]
```

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Passing_Calculation_iterative.py

Passing_Calculation_iterative.py × | Rotation_Regulation.py × | Search_for_orbit.py × | Find_end_section_Tests

```
56
57          print(p1)
58          print(p2)
59
60          # extracts the datetime from p1 and p2 ...
61          date_time1 = p1[12:21]
62          date_time2 = p2[12:21]
63
64          # print(date_time1)
65          # print(date_time2)
66
67          # ... and converts it from string into seconds
68          time1 = gss.get_sec_str(date_time1)
69          time2 = gss.get_sec_str(date_time2)
70
71          print(time1)
72          print(time2)
73
74          dt = time2 - time1
75
76          # print("Time difference between t1 and t2 :",dt, "seconds")
77
78          # extracts the elevation angles from p1 and p2 ...
79          e1_str = p1[29:35]
80          e2_str = p2[29:35]
81
82          # print(e1_str)
83          # print(e2_str)
84
85          # ... and converts it from string into seconds
86          e1 = float(e1_str.replace(',','.'))
87          e2 = float(e2_str.replace(',','.'))
88
89          print(e1)
90          print(e2)
91
92          de = e2 - e1
93          print("Elevation angle difference :", de,"(degrees)")
94          de_rad = np.deg2rad(de)
95
96          rotational_elevation_speed_dps = de/dt #speed in deg/s
97          rotational_elevation_speed_rps = de_rad/dt #speed in radians per second
98
99          # extracts the azimuth angles from p1 and p2 ...
100         a1_str = p1[21:27]
101         a2_str = p2[21:27]
102
103         # ... and converts it from string into seconds
104
105         a1 = float(a1_str.replace(',','.'))
106         a2 = float(a2_str.replace(',','.'))
107
108
109         da = a2 - a1
110         print("Azimuth angle difference :", da,"(degrees)")
```

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Passing_Calculation_iterative.py

Passing_Calculation_iterative.py × | Rotation_Regulation.py × | Search_for_orbit.py × | Find_end_section_Tests.py × | Calling_Passing_Calculat

```
112         # Now we can start computing the rotationnal speeds
113
114         # The first step consists in removing some of the irregularities created
115         # by the file produced by Gpredict. These irregularities occur at the
116         # azimuth angle where making several rotation breaks the monotony of the
117         #  derivative
118
119
120
121         if j==0:
122             initial_da = (a2-a1)/dt
123         else:
124             A0 = Section_data_str[j-1] # j'aurai mis le tout dans une parenthèse avec une virgule, à revoir
125             A0_sub_str = A0[21:27]
126             a0 = float(A0_sub_str.replace(',','.'))
127             # print("a0 = ",a0)
128             # print("a1 = ", a1)
129             initial_da = (a1 - a0)/dt
130
131         # print("initial_da = ", initial_da)
132         controlled_a2 = RR.Rotation_Regulation(a1,a2,initial_da,dt)
133
134
135         new_a2 = controlled_a2
136
137         da = new_a2 - a1
138
139         # print("new_a2 = ",new_a2)
140         # print("da = ",da)
141
142         rotational_azimuth_speed_dps = da/dt #speed in degree per seconds;
143         rotational_azimuth_speed_rps = np.deg2rad(da)/dt #speed in radians per second
144
145         # print("rotational_azimuth_speed_dps = ",rotational_azimuth_speed_dps)
146         # print("rotational_azimuth_speed_rps = ", rotational_azimuth_speed_rps)
147
148         print('rotational speed on elevation :\n',rotational_elevation_speed_dps)
149         print('rotational speed on azimuth :\n',rotational_azimuth_speed_dps)
150
151         #We display the speed in deg/s but the useful data for the computing is
152         #the value. Nevertheless, if you choose to change to the return, you just
153         #have to change some lines in the call function
154
155         Return = [e1,e2,a1,new_a2,rotational_elevation_speed_rps,rotational_azimuth_speed_rps,dt]
156
157         print("Return = ",Return)
158
159         return (Return)
```

Function 5 - Rotation_regulation.py:

This function is analogue to the Prototype_rotation_regulation.m from the MATLAB program.

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Rotation_Regulation.py

Rotation_Regulation.py | Calling_Passing_Calculation_iterative.py | Passing_Calculation_iterative.py | Search_for_orbit.py | Find_

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 27 11:09:13 2021

@author: Guillem
"""

def Rotation_Regulation(angle1, angle2, previous_derivative, dt):

    derivative = (angle2-angle1)/dt

    if (previous_derivative > 0 and derivative < 0) or (previous_derivative < 0 and derivative > 0):

        if (previous_derivative > 0 and derivative < 0):

            new_angle2 = angle2 + 360

        elif (previous_derivative < 0 and derivative > 0) :
            new_angle2 = angle2 - 360
    else:
        new_angle2 = angle2

    return new_angle2
```

Function 6 - Search_for_orbit.py:

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Search_for_orbit.py

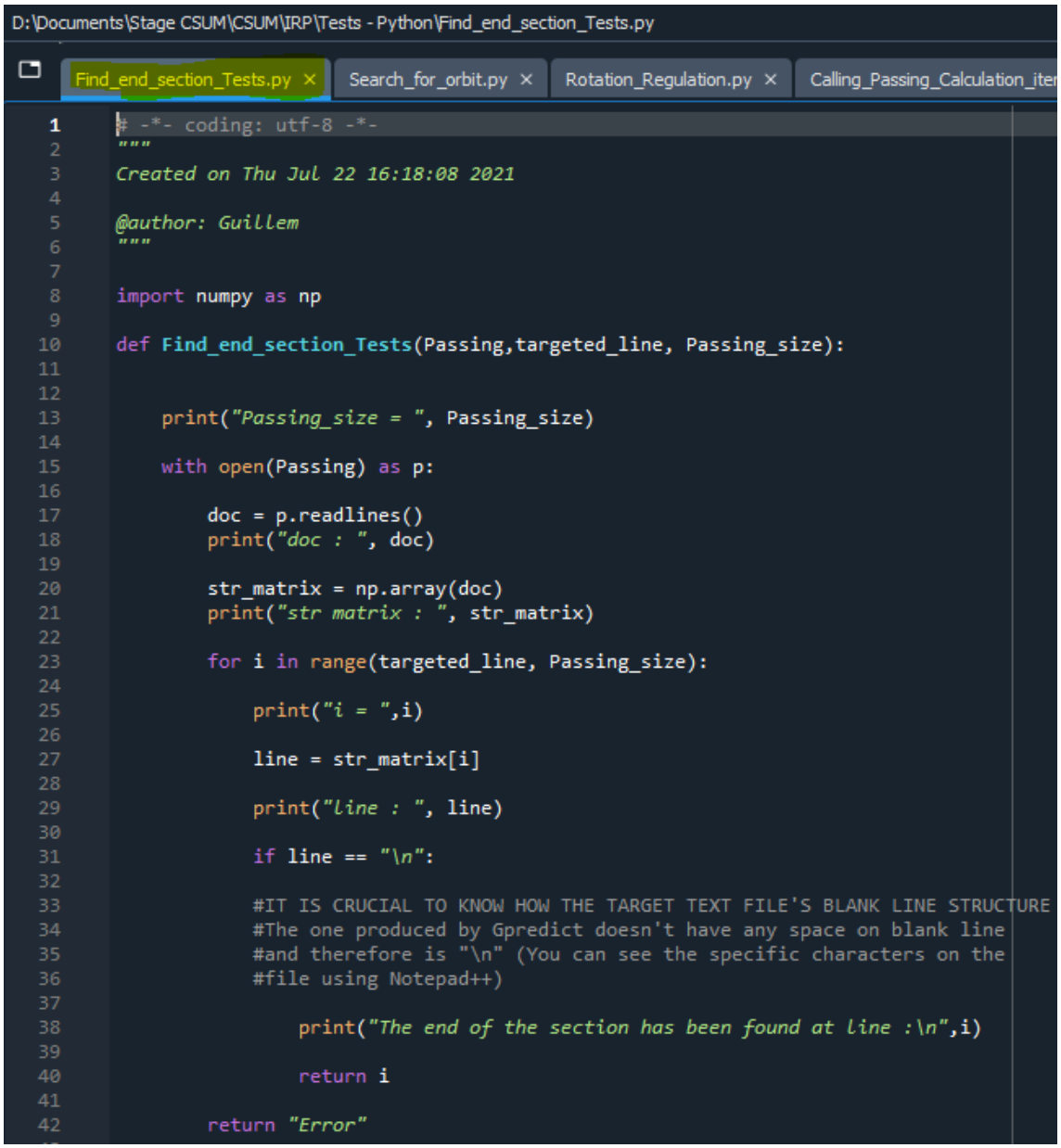
Search_for_orbit.py | Rotation_Regulation.py | Calling_Passing_Calculation_

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 22 15:22:51 2021

@author: Guillem
"""


def Search_for_orbit(Passing,orbit_number_str):
# opening a text file
    string1 = orbit_number_str
    file1 = open(Passing, "r")

    # setting flag and index to 0
    flag = 0
    index = 0

    # Loop through the file line by line
    for line in file1:
        index = index + 1

        # checking string is present in line or not
        if string1 in line:

          flag = 1
          break

    # checking condition for string found or not
    if flag == 0:
       print('String', string1 , 'Not Found')
    else:
       print('String', string1, 'Found In Line', index)

    # closing text file
    file1.close()
    return index
```

The above function seeks the orbit inside ROBUSTA-1B-passes.txt.

Function 7 - Find_end_section.py:

This function looks for the end of the section using the blank lines as stopping line.



Function 8 - Search_for_orbit:

This function opens the orbit file (Passing) and look out for the orbit number (orbit_number_str). Once/if it is found, it returns the number of the line where the orbit number has been found (index).

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Search_for_orbit.py

Search_for_orbit.py × Rotation_Regulation.py × Calling_Passing_Calculation_

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 22 15:22:51 2021

@author: Guillem
"""


def Search_for_orbit(Passing,orbit_number_str):
# opening a text file
    string1 = orbit_number_str
    file1 = open(Passing, "r")

    # setting flag and index to 0
    flag = 0
    index = 0

    # Loop through the file line by line
    for line in file1:
        index = index + 1

        # checking string is present in line or not
        if string1 in line:

          flag = 1
          break

    # checking condition for string found or not
    if flag == 0:
       print('String', string1 , 'Not Found')
    else:
       print('String', string1, 'Found In Line', index)

    # closing text file
    file1.close()
    return index
```

Function 9 -Txt_file_size.py:

Compute the given file (fname) size by enumerating amount of written line.

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\Txt_file_size.py

Txt_file_size.py × | Rotation_Regulation.py × | Passing_Calculation_iterative.py ×

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 25 11:29:57 2021

@author: Guillem
"""

def Txt_file_size(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    print("The number of line of this file is : ",i+1)
    return i + 1

```

Function 10 - get_sec_str.py:

The time data being in format HH:MM:SS, it is necessary to convert the extracted string from

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\get_sec_str.py

get_sec_str.py ×

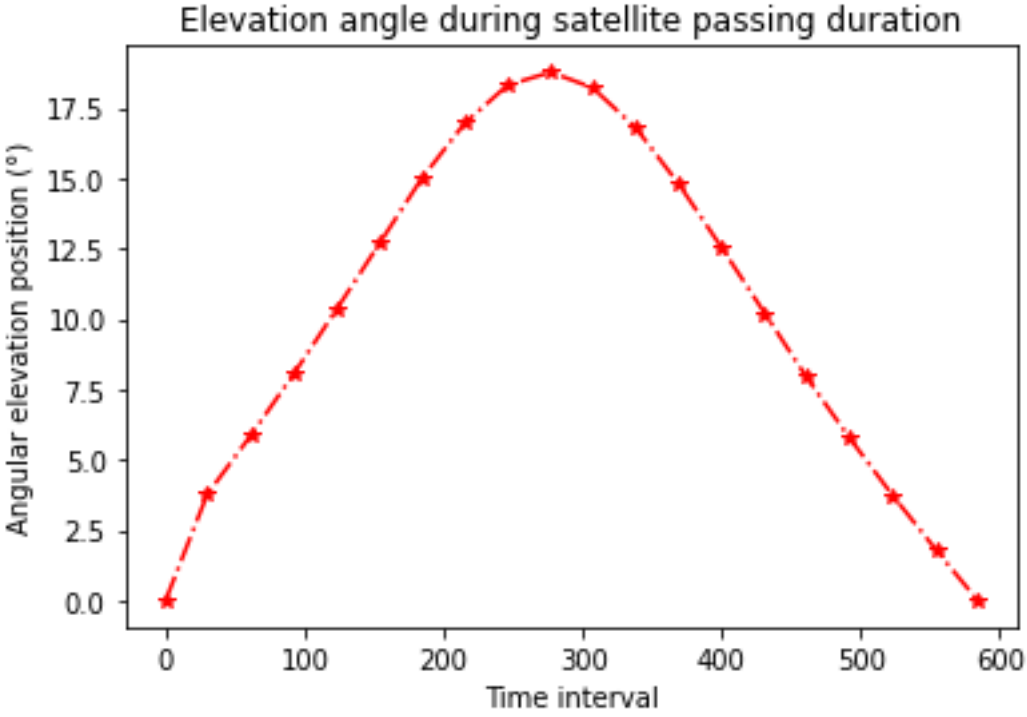
```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 22 15:04:17 2021

@author: Guillem
"""

def get_sec_str(time_str):
        """Get Seconds from time."""
        h, m, s = time_str.split(':')
        return int(h) * 3600 + int(m) * 60 + int(s)


# print(get_sec('1:23:45'))
# print(get_sec('0:04:15'))
# print(get_sec('0:00:25'))
```

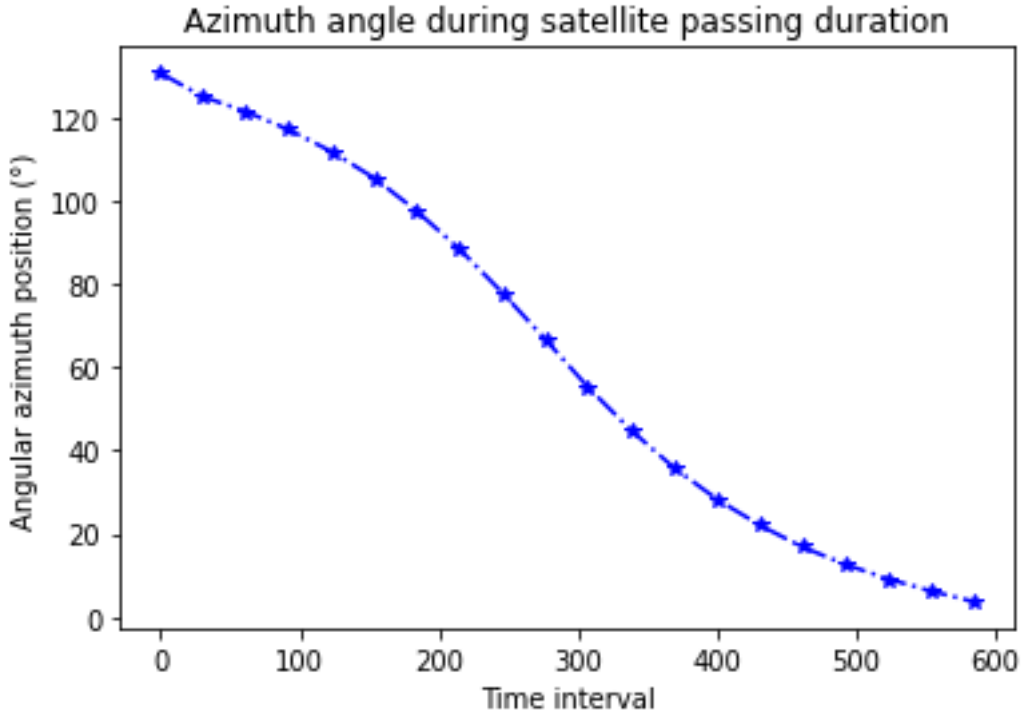
Test example for the V3:

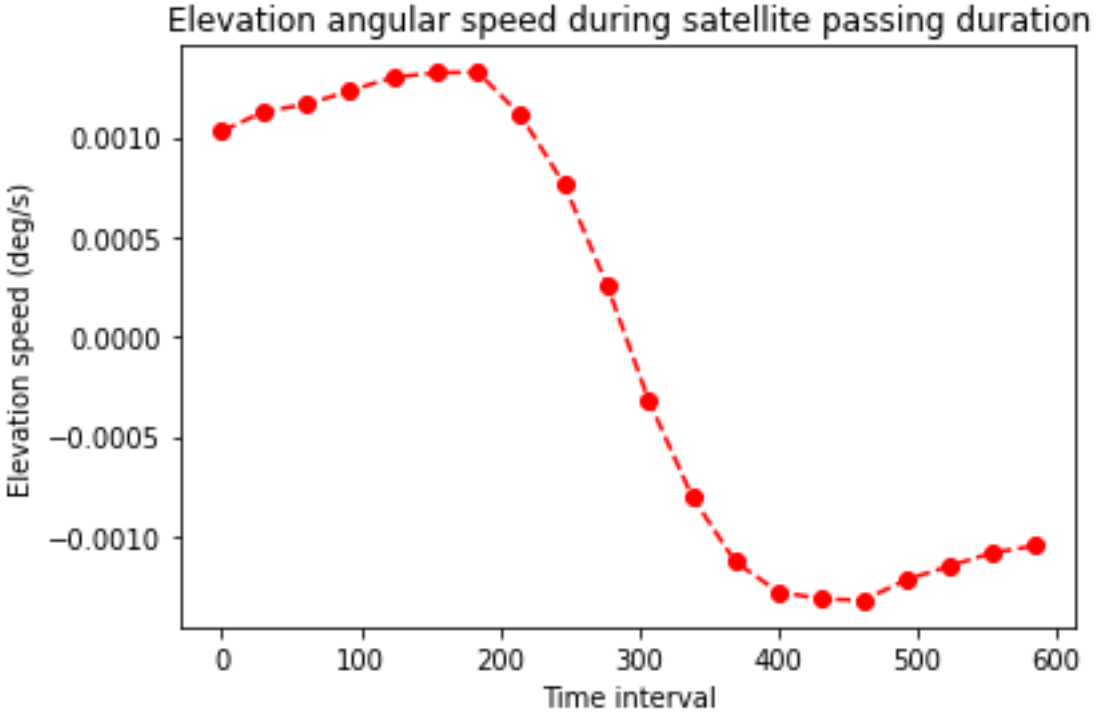
```
Insert the file name :ROBUSTA-1B-passes.txt

Enter the wanted reduction ratio :
880

Enter the number of the orbit :
15507

Which motor do you wish to use (i.e., which model ?):
 - for GM4632-370, enter 1
 - for MY1016Z2, enter 2
1

Which trend curve do you want to use to approximate your values ? Choose
between linear (type 1) or 2nd order polynomial (type 2)
WARNING : the most correct approach is linear as it fit the
electromechanics laws set here :
1
```

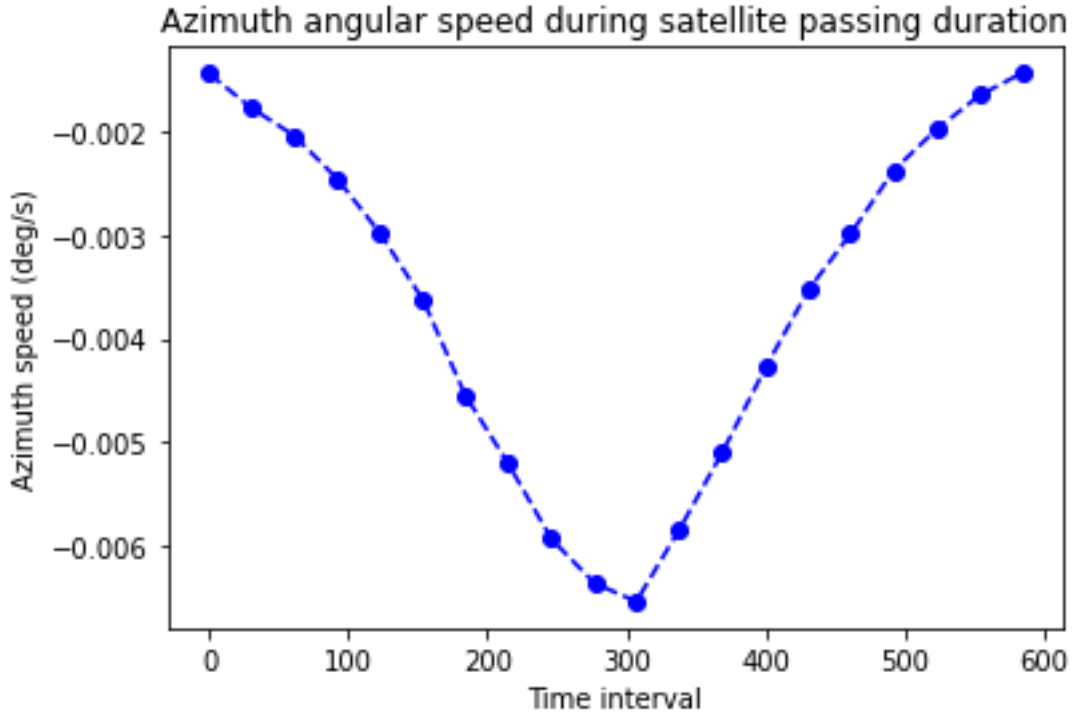
Orbit 15507

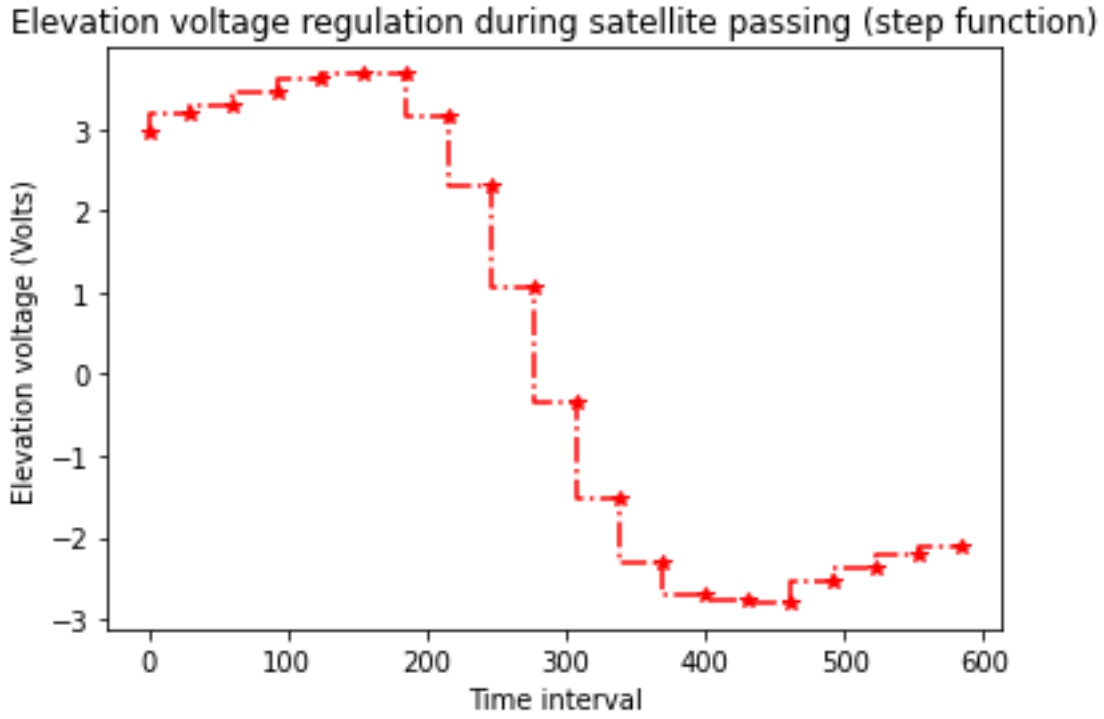
Time	Az	El	Range	Footp
2020/04/07 19:29:23	130,77	0,00	2575	4897
2020/04/07 19:29:54	128,21	1,83	2381	4899
2020/04/07 19:30:24	125,17	3,77	2193	4901
2020/04/07 19:30:55	121,55	5,84	2012	4904
2020/04/07 19:31:26	117,19	8,03	1840	4906
2020/04/07 19:31:57	111,89	10,34	1681	4909
2020/04/07 19:32:28	105,44	12,69	1539	4912
2020/04/07 19:32:58	97,63	14,97	1420	4914
2020/04/07 19:33:29	88,38	16,94	1328	4917
2020/04/07 19:34:00	77,84	18,30	1271	4920
2020/04/07 19:34:31	66,54	18,76	1254	4923
2020/04/07 19:35:01	55,30	18,21	1277	4925
2020/04/07 19:35:32	44,92	16,79	1340	4928
2020/04/07 19:36:03	35,87	14,79	1436	4931
2020/04/07 19:36:34	28,27	12,52	1559	4933
2020/04/07 19:37:05	22,01	10,19	1704	4936
2020/04/07 19:37:35	16,87	7,92	1865	4938
2020/04/07 19:38:06	12,65	5,76	2039	4941
2020/04/07 19:38:37	9,14	3,72	2221	4943
2020/04/07 19:39:08	6,21	1,80	2411	4945
2020/04/07 19:39:38	3,74	-0,00	2605	4947

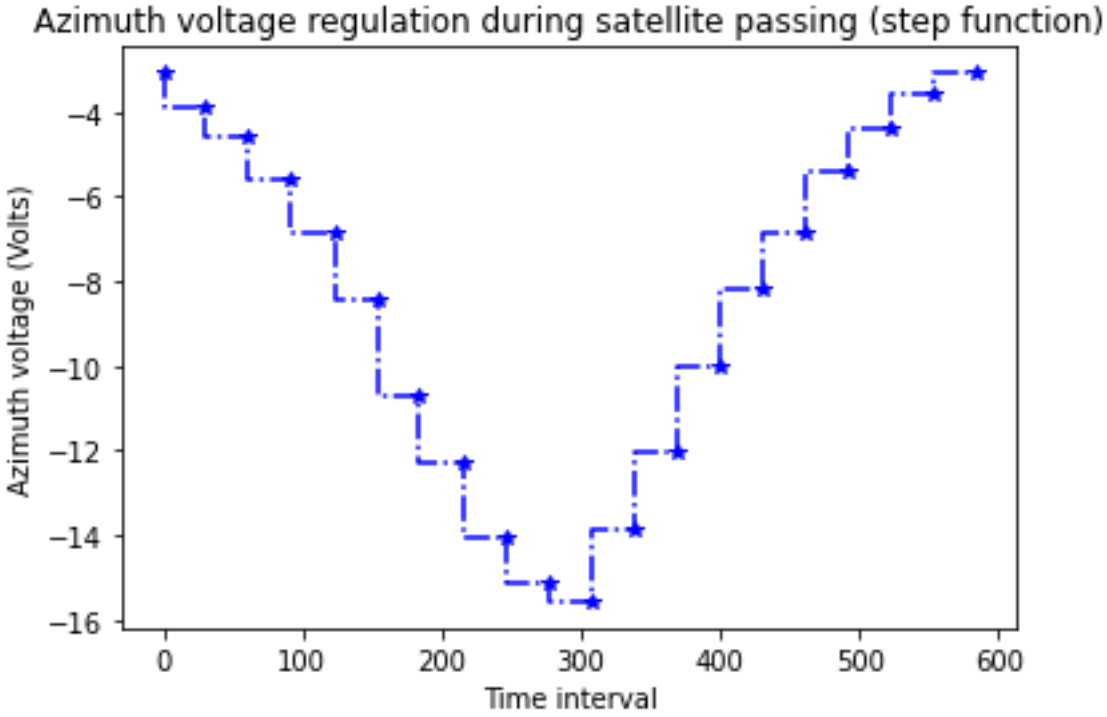

Elevation angle during satellite passing duration
Angular elevation position (°)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
100
200
300
400
500
600
Time interval


Azimuth angle during satellite passing duration
Angular azimuth position (°)
120
100
80
60
40
20
0
0
100
200
300
400
500
600
Time interval


Elevation angular speed during satellite passing duration
Elevation speed (deg/s)
0.0010
0.0005
0.0000
−0.0005
−0.0010
0
100
200
300
400
500
600
Time interval


Azimuth angular speed during satellite passing duration
Azimuth speed (deg/s)
−0.002
−0.003
−0.004
−0.005
−0.006
0
100
200
300
400
500
600
Time interval





This shows again that without the possibility of voltage regulation with a PWM signal, the ideal reduction ratio for the GM4632-370 is 880 since the values are voltage values are inside the authorized interval.

B.4 PWM testing on RPi for motor control

This function is the one that is to be modified to control the motors' voltage computed by the pseudo-tracking program. In its current state, it can only manually power the motor for a definite time.

D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\RaspberryPi\Test_moteur_RPi.py

Test_moteur_RPi.py*

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 26 13:46:55 2021

@author: Guillem
"""

import RPi.GPIO as GPIO

from time import sleep

"""
Due to a lack of component, we did not have the right controller at the time
to do the 2 motors test. Nevertheless we can do it with a single motor
configuration using a L293D from TI.

Note : on the datasheet it appears possible to make a 2 motor configuration using
the L923D but after numerous attempted we decided to stick to a single motor
configuration for now
"""

#Might be changed to GPIO.BCM configuration if the RPi model changes
#(because the mapping of the pin could change as well whereas the pin ID won't)
GPIO.setmode(GPIO.BOARD)

#One of the PWM enabling channel is located on pin 12
Motor_1_Enable = 12

#On pin 22 and 24 there are channel able to command the motor direction
#(forward and backward)
Motor_1_A = 22
Motor_1_B = 24

GPIO.setup(Motor_1_Enable, GPIO.OUT) #This is the enable pin for the controller IC

#These are the input pins for the controller IC
GPIO.setup(Motor_1_A,GPIO.OUT)
GPIO.setup(Motor_1_B,GPIO.OUT)

#defining the directions given by the pin linked to the motor
forward = GPIO.PWM(Motor_1_A,100)
backward = GPIO.PWM(Motor_1_B,100)

if __name__ == '__main__':

    forward.start(0)
    backward.start(0)

    """
    The value that is given as duty cycle (ChangeDutyCycle) represents the percentage of the
    voltage that will be inputed in the motor. The RPi sends a PWM that
    matches this duty cycle. Since the Voltage given by the source is 24 cst,
    using a PWM allows for lower tension : 50% for 12 Volts etc.
```

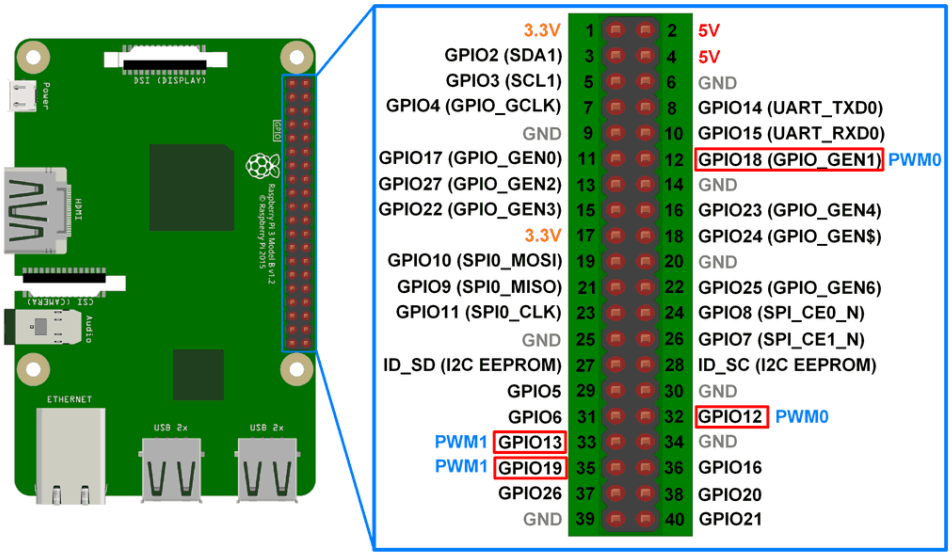
D:\Documents\Stage CSUM\CSUM\IRP\Tests - Python\RaspberryPi\Test_moteur_RPi.py

Test_moteur_RPi.py*

```
28
29      #On pin 22 and 24 there are channel able to command the motor direction
30      #(forward and backward)
31      Motor_1_A = 22
32      Motor_1_B = 24
33
34      GPIO.setup(Motor_1_Enable, GPIO.OUT) #This is the enable pin for the controller IC
35
36      #These are the input pins for the controller IC
37      GPIO.setup(Motor_1_A,GPIO.OUT)
38      GPIO.setup(Motor_1_B,GPIO.OUT)
39
40      #defining the directions given by the pin linked to the motor
41      forward = GPIO.PWM(Motor_1_A,100)
42      backward = GPIO.PWM(Motor_1_B,100)
43
44      if __name__ == '__main__':
45
46          forward.start(0)
47          backward.start(0)
48
49          """
50          The value that is given as duty cycle (ChangeDutyCycle) represents the percentage of the
51          voltage that will be inputed in the motor. The RPi sends a PWM that
52          matches this duty cycle. Since the Voltage given by the source is 24 cst,
53          using a PWM allows for lower tension : 50% for 12 Volts etc.
54
55          The input source IS NOT given here, only the PWM to regulate it so you
56          have to know in advance the voltage that you will use.
57          """
58
59          #Here we'll go forward at 100% of the voltage, 0% of the voltage in the
60          #backward direction and for 20 seconds
61          print("Go forward")
62          GPIO.ouput(Motor_1_Enable,GPIO.HIGH)
63          forward.ChangeDutyCycle(100)
64          backward.ChangeDutyCycle(0)
65          sleep(20)
66
67          #Here we'll go backward at 50% of the voltage, 0% of the voltage in the
68          #forward direction and for 10 seconds
69          print("Go backward")
70          GPIO.ouput(Motor_1_Enable,GPIO.HIGH)
71          forward.ChangeDutyCycle(0)
72          backward.ChangeDutyCycle(50)
73          sleep(10)
74
75          #Stop motor
76          print("Stop motor")
77          GPIO.ouput(Motor_1_Enable,GPIO.LOW)
78          forward.stop()
79          backward.stop()
80
81          GPIO.cleanup()
```

The operating mode of this function is the following:

1- On line 24, setmode(GPIO.BOARD) maps the RPi pins for the function using their physical position, i.e., their Pin Position Number. (See the figure below)



2- Then the Pin of use are selected, one to transmit the Enable signal, Pin n°12 (generating a PWM signal), linked to the Enable port of the drivers (see figure below). Two other Pin (here 22 and 24 on the RPi) able to generate a voltage big enough to activate the driver's 1A and 2A port are also connected to the driver. The directions are defined with these two pins on line 41 and 42: "forward" corresponding to an activation of 1A (= Pin n°22) and "backward" to 2A (= Pin n°24).

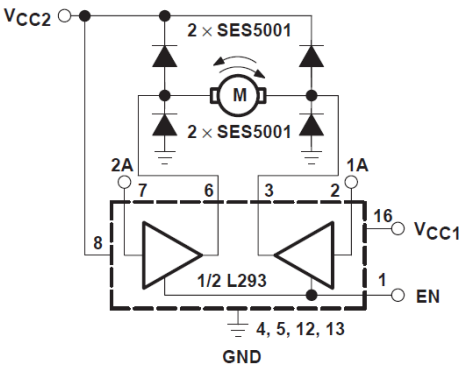


Figure 5. Bidirectional DC Motor Control

EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

3- The connected motor is then commanded using 3 other functions:

a. GPIO.output(Motor_1_Enable, GPIO.HIGH): On line 62, 70, 77, this first function defines the motor state of activation of the Enable Pin, "HIGH" for the motor to be activated, "LOW" to stop. (The below figure

shows the motor function table commanding the pins state from the controller)

b. ChangeDutyCycle(X): This function regulates the Voltage inputted to the motor by the power source. The working procedure is the following, if we consider the power input transmitted to the motor as constant, the X value inside ChangeDutyCycle(X) is a percentage of this voltage. Using again the example of ***Phase Width Modulation (PWM)*** : if the power input is 24 Volts and the motor only requires 6 Volts, X = 25. The consequence of that is that the pin responsible for activating the motor will be a state “1” only 25% of the time and 75% of the time at the “0” state.
c. sleep(T), this function locks the ongoing operation for the time value given by T

Having these function working, it is possible to create an iterative loop using the following process: The voltage and time duration/interval associated for each axis being computed and regrouped in one matrix thanks to the pseudo-tracking program, it is possible for each motor (elevation and azimuth) and for each voltage value, to determine the duty cycle using proportionality rule and input it in sleep(T).

Important remark:

Regarding the mapping of the pin, it is possible to use another function called “GPIO.BCM “. This function maps the pin by their identification name (see the previous mapping figure. As an example: “Pin n°22” becomes “GPIO25”. This method is more robust if the new version of RPi had to change their pin mapping, because “GPIO.BCM” maps them using their function. This means that even with a change of mapping, Motor_1_Enable would still be a PWM generator using “GPIO.BCM” while this would not be sure using “GPIO.BOARD”. However, during the test, the author did not manage to the wanted control with this method. It is still an element of interest considering the proposed advantages.