Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Tesi Magistrale

PACMAN: Position and Attitude Control with MAgnetic Navigation for CubeSats. An experiment under the ESA - Fly Your Thesis! programme

Candidato:
Matteo Vitturi
Matricola 1128686

Relatore:
Angelo Cenedese

Anno Accademico 2017–2018

Abstract - EN

Since the beginning of the space era, proximity manoeuvres have been performed with the goal of rendezvous and docking between spacecrafts. As of today, for both large and small satellites, proximity manoeuvres are performed using mainly propulsive units for both position and attitude control.
Therefore, in the last few years there has been an increasing interest in the development of different technologies for proximity navigation and rendezvous manoeuvres, addressing the main issues of fuel consumption and the strong impact of close range navigation subsystems on small scale satellites.
In this framework, PACMAN (Position and Attitude Control with MAgnetic Navigation) is a technology demonstrator experiment designed to exploit magnetic interactions between spacecrafts for relative position and attitude control during proximity manoeuvres.
PACMAN experiment has been proposed to ESA and selected to fly during the 68th ESA Parabolic Flight Campaign, in the framework of ESA Education *Fly Your Thesis!* 2017 programme.
In this work we present an overview of the overall design of the experiment; starting with a description of all the systems of which it is composed, we then present the dynamical simulations that led to the final design and finally, we present a preliminary analysis of the data obtained during the flight campaign that has taken place in Bordeaux, on December 2017.
A first analysis has shown promising results but also highlighted the limits of the designed systems.

Abstract - IT

Sin dall'inizio dell'era spaziale, vengono eseguite manovre di prossimità tra satelliti per l'avvicinamento e l'attracco degli stessi. Ad oggi, per satelliti di piccole e grandi dimensioni, tutte le manovre di controllo di posizione e assetto vengono eseguite utilizzando principalmente unità propulsive.
Pertanto, negli ultimi anni si è verificato un crescente interesse nello svilupppo di tecnologie alternative per la navigazione di prossimità, affrontando i principali problemi del consumo di carburante e il forte impatto dei sottosistemi di navigazione a corto raggio su satelliti di piccola scala.
In questo contesto, PACMAN (Position and Attitude Control with MAgnetic Navigation) è un esperimento il cui obiettivo è quello di dimostrare una tecnologia che sfrutti le interazioni magnetiche tra veicoli in assenza di gravità, per il controllo di posizione ed assetto durante le manovre di prossimità.
PACMAN è stato proposto all' Agenzia Spaziale Europea (ESA) ed è stato selezionato per essere testato durante la 68° campagna di voli parabolici, nell'ambito del programma di ESA Education: Fly Your Thesis!.
In questa tesi viene presentato il design dell' intero esperimento; a partire da una descrizione di ogni sistema di cui è composto, vengono presentate le simulazioni dinamiche del sistema che hanno portato al design definitivo dell'esperimento e infine, viene presentata un'analisi preliminare sui dati ottenuti durante la campagna di volo tenutasi a Bordeaux, nel Dicembre 2017.
Una prima analisi ha mostrato risultati promettenti, evidenziando però anche i limiti del sistema sviluppato.

Una dedica speciale ai miei genitori,
che mi hanno sempre sostenuto e incoraggiato ad inseguire i miei sogni,
a mio fratello, che anche se distante per me è sempre presente,
alla mia ragazza, Jessica, che mi sostiene ogni giorno
e rende la mia vita migliore, senza la quale non sarei la persona che sono oggi,
al mio relatore, Angelo Cenedese, che mi ha introdotto a questa fantastica
esperienza e dato supporto lungo tutto il suo percorso,
ed infine al team PACMAN, grazie ragazzi, sono stati mesi duri,
ma siamo riusciti a fare l'inimmaginabile!

Contents

List of Figures

1 Introduction

1.1 Fly Your Thesis!

The *Fly your Thesis!* programme is one of the Hands-on space projects of ESA (European Space Agency) Education programme, together with the *Drop your thesis!*, *Spin your thesis!* (Human or non human edition) and *REXUS/BEXUS rocket & balloon* projects, which grant the opportunity to test experiments on different gravity conditions, from μg levels up to ≈ 20 times the normal gravity.
The ESA Fly your thesis! gives the unique opportunity to fly technological research or scientific experiments related to physics, chemistry, biology and much more in microgravity conditions, situation that cannot be easily achieved experimenting on Earth's ground.
Furthermore, this is the only micro-gravity platform that allows scientists to interact with their own experiment while it is experiencing weightlessness, rather than doing it by remote on a robotic capsule or in the ISS (International Space Station) where only astronauts can interact with it.

Figure 1.1: The Zero-G aircraft during a parabolic flight. Photo: ESA.

The FYT campaign consists of three flights taking place on a two-engine modified Airbus A310 Zero-G (Figure 1.1), which performs 31 parabolas on each flight, providing about 20 seconds of microgravity each.
The programme launches a call for proposal once a year, during which master and PhD candidates, together with an endorsing professor, can send their proposal for experiments tailor-made for parabolic flights.
ESA's Education office makes a first selection of the teams that are then invited to present their proposal to a Review Board. Based on this presentation and the required documents, up to four teams can be selected and given the opportunity to participate in a 2 weeks parabolic flight campaign that takes place in Bordeaux, France.
From the last phase on, the selected teams have about 9 months to develop and build their experiments [5].

1.2 What is a parabolic flight

A parabolic flight is an aircraft flight that performs a series of manoeuvres, called parabolas, each providing $\approx$ 20 seconds of microgravity or weightlessness, during which scientists are able to test their experiments in low gravity conditions and obtain data that would otherwise not be possible on Earth.
As previously mentioned, during a campaign typically three flights are carried out, each performing 31 parabolas for a total of about 30 minutes of weightlessness.
Furthermore, before and immediately after each parabola there are two periods of increased gravity ($\approx 1.5 - 1.8$ g) during which, if necessary, the experiments can also be tested (see Figure 1.2).
The reasons why parabolic flights are such an important tool for scientists are many, starting from the level of low gravity that can be achieved (with an accuracy in the order of 10^{-2}g), to the low-cost research opportunity, the possibility of executing a series of experimental runs and the possibility to modify the experimental setup by the research team between flights.
But most of all, parabolic flight airplanes are the only sub-orbital carriers that provide the possibility of direct intervention by the research team during flight, and the only which provides users with the opportunity to do research on human subjects under conditions of micro gravity, complementing studies conducted in space.

In order to achieve such a particular manoeuvre, the aircraft is driven by three pilots; in particular, in the cockpit two pilots and one flight engineer collaborate to fly the parabola, the two pilots control two different axis, the pitch and the roll, while the flight engineer handles the power thrust.
The principle behind weightlessness is quite simple: the aircraft and everything inside of it accelerates towards the Earth at the same rate of a free falling object in a

vacuum, $\approx 9.8\frac{m/s}{s}$.
From a stabilized level-flight attitude, one pilot gradually pulls up the nose of the aircraft, and it starts climbing at a steadily increasing angle. In this phase, which lasts for about 20 seconds, inside the air plane everything is pressed to the floor with a force between 1.5 and 1.8 times it's normal weight, i.e. $1.5 - 1.8$g.
Once the aircraft is climbing at about 50 degrees, the pilots reduce the thrust to the minimum, very skilfully adjusting the engine thrusts to compensate the air-drag.
At this point, the aircraft follows a free-fall ballistic trajectory, which lasts for about 20 seconds and during which everything inside the air plane is experiencing weightlessness.

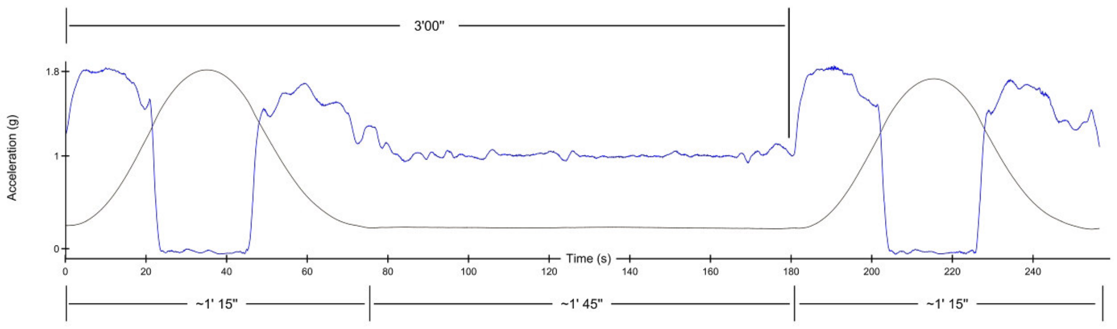


Figure 1.2: Gravity levels during a parabola manoeuvre. [11]

At the end of this phase, the aircraft must pull out of the parabolic arc, subjecting again everything inside of it to hyper gravity conditions, as the air plane accelerates upwards.
Finally, after about 20 seconds, the aircraft again flies a steady horizontal path at 1g and scientists have then about 1 minute and 40 seconds minimum before the next parabola to set up their experiments again.
In Figure 1.2 it is reported the acceleration profile of the aircraft during a parabolic flight, while in Figure 1.3 it is illustrated the different phases of a parabola explained above.

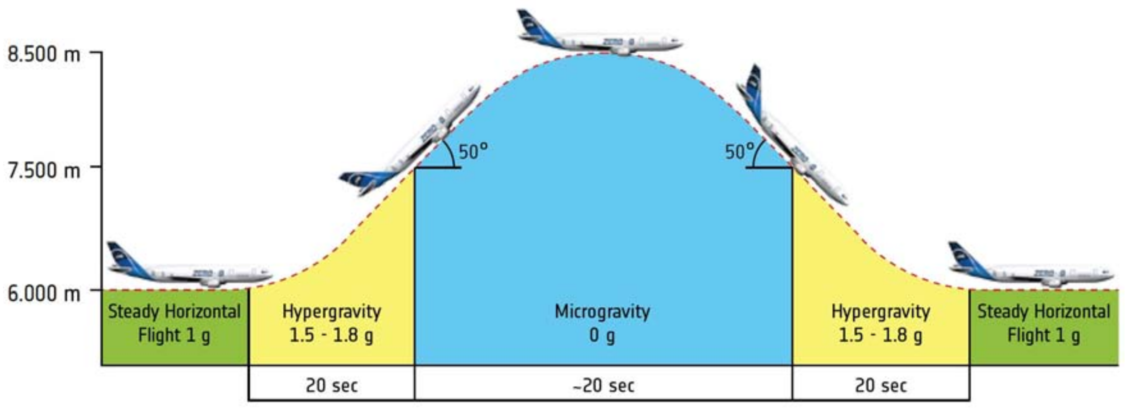


Figure 1.3: Parabolic flight manoeuvre profile. Image adapted from NoveSpace.

1.2.1 A brief history of parabolic flights

Aircrafts flying parabolic trajectories provided important informations in space exploration and research for the past years. For example, they have been carried out to improve aerospace medicine knowledge, to train astronauts and more in general to offer a low gravity platform to the scientific community. This simple idea took a few years to perfect, but is now routinely used to produce repeated periods of microgravity, which are mainly used in close link to space exploration [11].

The first noteworthy flight that achieved microgravity condition was in 1938, when an aeronautical medical doctor, Dr Heinz von Diringshofen, studied the physiological effects of hyper and micro gravity during flight manoeuvres. At that time, weightlessness phases were achieved during nose-down manoeuvres.

Two years later, in 1940, an aerospace medicine researcher, Dr Hubertus Strughold, studied the influence of load factors on pilots' disorientation, but again this study was performed under similar conditions to the previous mentioned.

In 1950, two scientists, Dr Fritz Haber and Dr Heinz Haber published a theory to mimic microgravity conditions on board an aircraft flying a parabola trajectory and the following year their theory was verified by several test flights.

In 1955 the parabolic flight technique was refined and in 1958, when US program Mercury was launched, parabolic flight were flown on board a Convair C-131 Samaritan.

In 1973 NASA proceeded with the program on board a Boeing KC-135 and in 1984 the ESA offered the European scientific community to participate in NASA's parabolic flight program.

Then, in 1989, under the leadership of French astronaut J.F. Clervoy, assisted by French astronaut J.P Haigneré, the ESA and CNES (Centre National d'Ètudes Spatiales, the French space agency) launched a parabolic flight program on board a Caravelle, accessible to European scientists. Novespace, a subsidiary of CNES, was tasked with organizing the flights.

After a few years, in 1996, Novespace switched from the Caravelle to an Airbus A300, named A300 ZERO-G, which offered the largest experimental area for parabolic flights in the world.

In 2004 the US company ZERO-G Corporation opens parabolic flights to the public on board a Boeing 727.

Finally, in 2014 the Airbus A300 ZERO-G flown its last parabolic flight, and in 2015 the Airbus A310 ZERO-G replaced it. The A310 ZERO-G is currently used to carry out parabolic flights and it is the aircraft on which the PACMAN experiment has been tested.

1.3 PACMAN: Introduction to the experiment

Since the beginning of the space era, proximity manoeuvres performed with the goal of rendezvous and docking between spacecrafts have been a hard task to accomplish. The ability to approach and dock between space vehicles is of fundamental importance for several activities such as material and crew transfer, inspections, modular structure assembling and much more.
Probably the most famous example comes from the assembly of the International Space Station (ISS), which is a modular station placed in orbit thanks to the rendezvous and assembly of several different modules delivered by subsequent launches and manoeuvred to mate.
As of today, for both large and small satellites, proximity manoeuvres for both position and attitude control are performed using mainly propulsive actuators. Furthermore, no competitive or commercial solution is available to perform autonomous rendezvous and docking between small satellites.
Therefore, in the last few years there has been an increasing interest in the development of different technologies for proximity navigation and rendezvous manoeuvres, addressing the main issues of fuel consumption and the strong impact of close range navigation subsystems on small scale satellites. In particular, one promising solution is represented by relative magnetic navigation, where one vehicle, the chaser, can control its position and attitude relative to a target vehicle, thanks to magnetic

interactions between them, conserving the system momentum while allowing to save fuel.

In this framework, PACMAN (Position and Attitude Control with MAgnetic Navigation) Experiment is a technology demonstrator designed to exploit the advantages of magnetic interactions for relative position and attitude control during proximity manoeuvres.
PACMAN Experiment has been designed and developed by a team composed of 6 Italian students from the University of Padova, each with different skills required to the realization of the experiment: 2 PhD Students in Science, Technologies and Measurements for Space, 1 Master Student in Automation Engineering, 1 Master Student in Electrical Engineering, 1 Master Student in Aerospace Engineering and 1 Master Student in Electronic Engineering.
The experiment, supported by the University of Padova, has been proposed to ESA and selected to fly during the 68$^{\text{th}}$ ESA Parabolic Flight Campaign, in the framework of ESA Education *Fly Your Thesis!* 2017 programme.

In summary, the main goal of the project is to develop and validate in low-gravity conditions an integrated system for proximity navigation and soft-docking based on magnetic interactions, suitable for small-scale spacecrafts or nano-satellites. This has been accomplished by launching a miniature spacecraft mock-up towards a free-floating target that generates a static magnetic field in low gravity conditions. A set of actively-controlled magnetic coils on-board the spacecraft mock-up, i.e. the chaser, assisted by dedicated localization sensors, have been used as actuators to control its attitude relative to the target.
As this solution is only based on magnetic interactions, there is no need of thrusters, thus significantly reducing mass and the overall system complexity.

1.4 Rendezvous and docking phases

In order to better understand how rendezvous and docking manoeuvres are accomplished, it is here presented a summary of the phases of a rendezvous mission.
In the aerospace framework, after the launch, a rendezvous mission is usually divided into five major phases, as reported in Figure 1.4: phasing, far range rendezvous, close range rendezvous, mating and finally the separation and departure phases [6].

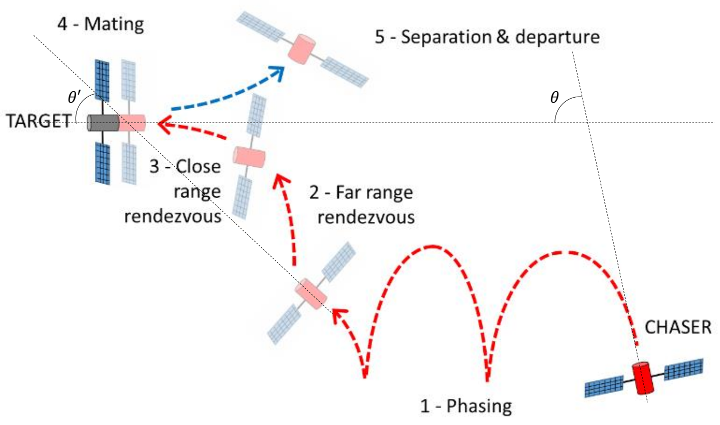


Figure 1.4: Standard rendezvous and docking manoeuvre.

With reference to Figure 1.4, on the first phase the misalignment angle θ between the chaser and the target spacecraft is reduced to θ' and as a rule, the manoeuvres are controlled from the ground. The phasing ends with the acquisition of an 'initial aim point' or with the achievement of a set of margins for position and velocity at a certain range.
Then, the far range rendezvous phase starts, whose major objective is the achievement of position, velocity and angular rate conditions, necessary for the initialization of the next phase.
When the distance between the two spacecrafts is of few kilometres, the close range rendezvous phase can start. This phase is usually divided into two subphases: a preparatory phase called 'closing', whose main objective is to prepare the chaser to start the final approach, and the final approach phase, which leads to the mating conditions.
Finally, during the mating, dedicated mechanisms can be used to realise soft joining before activating the hard-docking latches.

From the navigation point of view, the close range rendezvous phase is the most expensive and demanding, due to the strict safety constraints that limit the possible approach strategies and the manoeuvring velocity. For this reason, the autonomous rendezvous and docking was hardly addressed in space, with few important exceptions like European ATVs and Russian Progress spacecraft.
In this framework, the self alignment capability of two small satellites, guaranteed by controlled magnetic fields interaction, can greatly improve the performances of close range rendezvous phase.

1.5 Contributions and related works

Most of the studies conducted on this topic are based on numerical simulations, without an experimental setup or tests on relevant environment which verify and validate the models.
Electromagnets' self docking capabilities have not been studied deeply yet, but analogous studies on the effects of electromagnetic interactions for relative motion of spacecraft have been investigated. For example, Electromagnetic Formation Flight (EMFF) clearly expresses the concept of using electromagnets, coupled with reaction wheels, to grant all the required forces and torques needed to maintain a satellite's relative position and attitude in a formation of satellites [20].
The most relevant contribution in this field is represented by the RINGS (Resonant Inductive Near-field Generation System) project, which was tested on the International Space Station in 2013 [16]. This experiment studied an innovative electromagnetic system used to perform formation flight between SPHERES, which are nano-satellite mock-ups provided with autonomous navigation and control capabilities. As actuators, to create the required forces and torques between close vehicles, RINGS used large coils, whose dimensions and mass have a strong impact on the vehicles themselves. Even if the project represents a relevant study, RINGS was not used for electro-magnetic docking.
Moreover, authors of "Dynamics modeling of electromagnetic formation flight" [7] analysed attraction and repulsion forces for EMFF. They developed and verified a Simulink dynamics model at the Space Systems Laboratory (MIT) with the RINGS project. The simulations have been accurate in modelling the forces and torques induced by resonant coils as a result of their relative position and orientation, thereby allowing the development and test of formation flying control algorithms in the future.
The most recent electromagnetic docking system technology was probably designed and tested at the Surrey Space Centre [21], in the framework of AAReST program for in-orbit assembly of a space telescope, validating the procedure both with simulations and laboratory tests on ground on a low friction table, but not in a relevant environment yet. The objective of the project was to perform an autonomous in-orbit assembly and evaluate its potential in terms of effectiveness and cost savings. The AAReST mission involved two 'nanosatellites' based on 3U CubeSat-type structure, each carrying an electrically actuated adaptive mirror, and each capable of autonomous docking with a central 15U microsatellite, which houses two fixed mirrors and a 'camera package' mounted on a deployable pole.
Another interesting work on this topic is the On-Orbit Autonomous Assembly from Nanosatellites (OAAN) project, a collaboration between NASA and Cornell Univer-

sity [12], which is investigating autonomous control algorithms for rendezvous and docking manoeuvres using reconfigurable magnetic arrays to create a robust and low power docking mechanism.
Similarly, a collaboration between NASA and Tyvak aims to perform several in-orbit tests of rendezvous, proximity operations and docking by means of low-cost off the shelf components with 'CubeSat Proximity Operations Demonstration' (CPOD, two identical 3U modules).
Finally, another relevant example is represented by FELDs (Flexible Electromagnetic Leash Docking system) experiment, from the University of Padova, which has been tested in the framework of ESA *Drop your thesis!* programme. The project examined the self alignment capabilities of a tethered ferromagnetic probe launched towards a target electromagnet [13], demonstrating the effects of the magnetic interactions on the probe, which was passively guided to mate with the target.

In this context, PACMAN experiment improves the FELDs technology using an autonomous nanosatellite based on 1U CubeSat-type structure and provided with a set of actively controlled electromagnets, used as actuators to accomplish and validate a soft docking manoeuvre.
PACMAN goals are to increase the comprehension of the electromagnetic proximity interactions problem and to validate an active navigation and control system for close range rendezvous manoeuvres. Furthermore, PACMAN layout may be used as a baseline for future space applications on small satellites thanks to its scalability and represents a key interface to enable the assembly of modular satellites in space.
The results of this project have led to the preparation of two contributions to conferences related to space activities, in particular related to CubeSats and nano satellites. At the time of the writing of this thesis, it has been proposed for the 4S Symposium an abstract with the title *"PACMAN experiment: a CubeSat-size integrated system for proximity navigation and soft-docking."* and it has been submitted and accepted for the 2nd Symposium on Space Educational Activities an abstract with the title *"PACMAN experiment: a Parabolic Flight Campaign student experience."*.

2 PACMAN Experiment

The idea of PACMAN arises from the PhD thesis of the team leader of the project, Matteo Duzzi, involved in research concerning Spacecraft RendezVous and Docking (RVD) using electro-magnetic interactions.
In this context, as already mentioned, PACMAN Experiment aims to validate a technique for proximity navigation exploiting magnetic interactions in a low gravity environment. A schematic of the experiment is proposed in Figure 2.1.
As it can be seen, PACMAN Experiment is mainly composed by four major systems, each of them split in different subsystems, which will be analysed in detail in this work. The four main systems are represented by:

- A spacecraft mock-up, referred to as the CUBE;
- A Free Floating Target, referred to as FFT;
- The test chamber, hereinafter referred to as the CHAMBER;
- The Support Electronics.

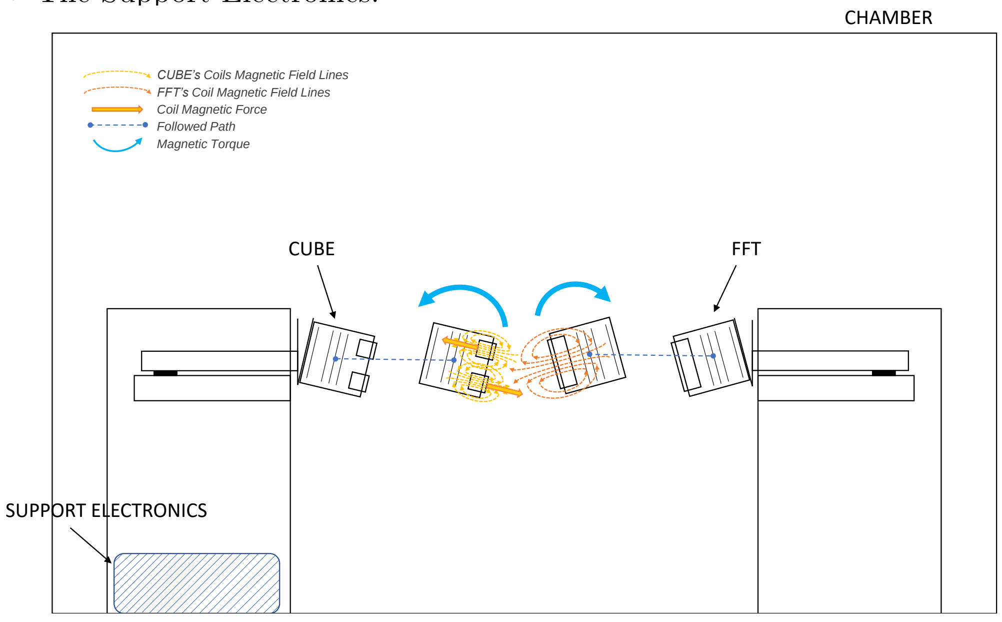


Figure 2.1: Experiment schematic.

2.1 CUBE

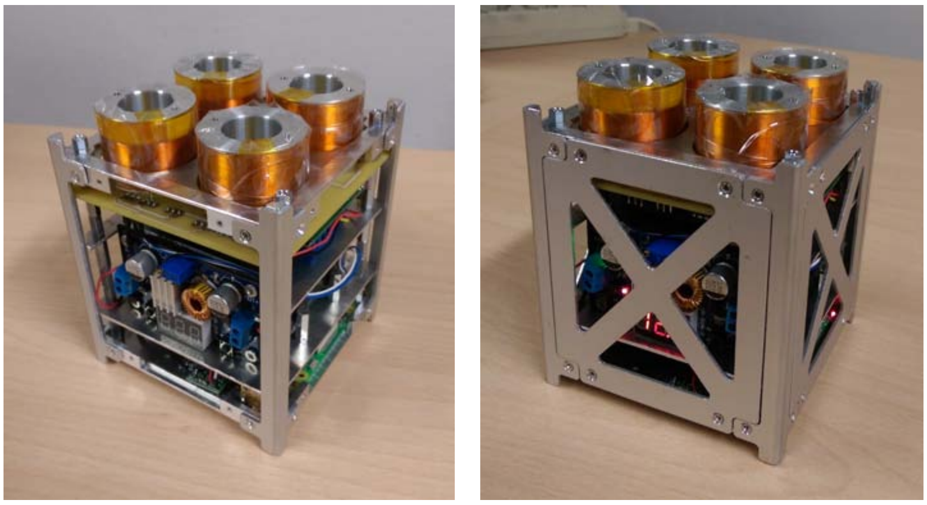

(a) CUBE without the side plates (b) Fully assembled CUBE

Figure 2.2: Photos of the CUBE.

Since the CUBE (shown in Figure 2.2) is the autonomous spacecraft mock-up actively regulated to control its relative pose with respect to the Free Floating Target, it is the most important system of the whole experiment.
In the CUBE system, five subsystems can be recognised:

- the structure;
- the navigation subsystem;
- the data handling and communication subsystem;
- the power subsystem;
- the software controller.

2.1.1 CUBE: Structure

The design of the structure follows the CubeSat standards, which are a type of miniaturized satellites made up of $10 \times 10 \times 11.35$ cm^3 units, designed to provide $10 \times 10 \times 10$ cm^3 of useful volume (1U CubeSat).
As already mentioned, the CUBE is composed of a 1U CubeSat structure, the smallest standard size, that can be seen in Figure 2.3.

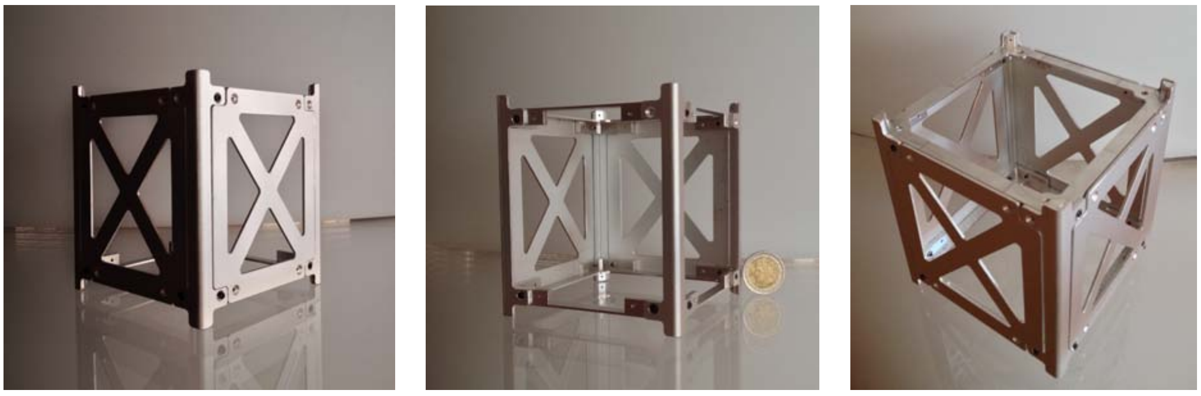

(a) Partially assembled CUBE structure. (b) Comparison between a 1U CubeSat structure and a 2 Euro coin. (c) Fully assembled empty CUBE structure.

Figure 2.3: Photos of the complete CUBE structure in Aluminium.

The boards placed inside the structure contain all the electronic components in the correct position inside the CUBE.
As shown in Figure 2.4, they are represented by:

– the *Camera layer*, which keep the Camera vision system of the CUBE pointing in the right direction (Figure 2.7);
– the *Driver Circuit Board*, which keep the coils, the Arduino microcontroller, the coil drivers and the support electronics needed to communicate with the Raspberry Pi board fixed;
– the *IMU layer*, which keeps the IMU board fixed;
– the *Battery layer*, which keeps the batteries and the buck converter used to power the coils fixed;
– the *Data Handling layer*, which keeps the Raspberry Pi board, the buck converter needed to power it and the anti-reverse voltage circuit board fixed.

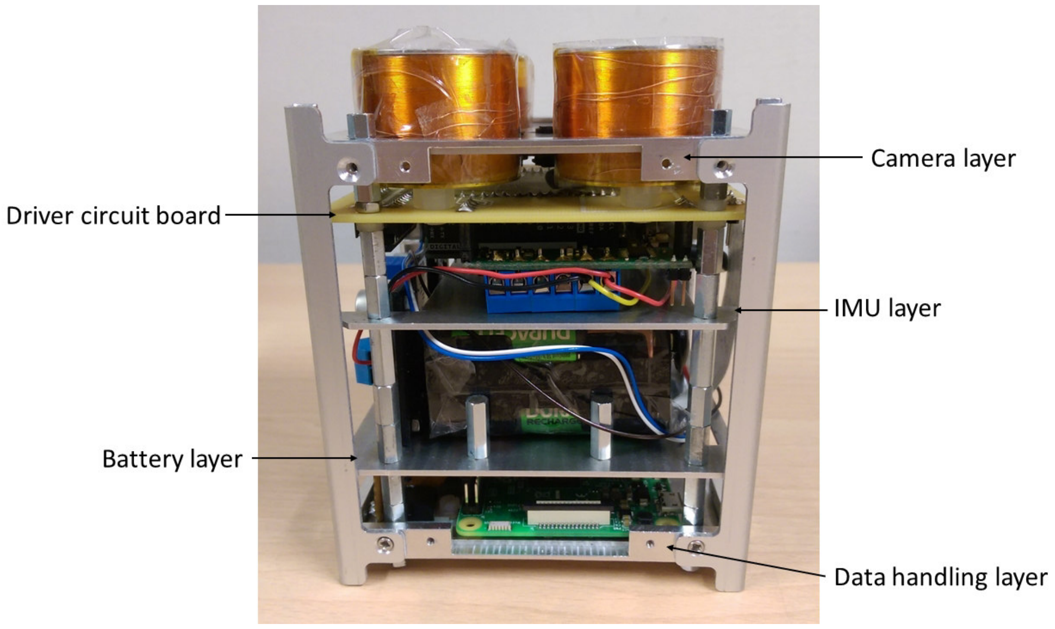


Figure 2.4: CUBE layers.

Finally, at the bottom of the structure it is mounted a Hold & Release System Interface, composed of an iron plate properly modelled to hold the CUBE in position during the hyper gravity phase and easily release it during the micro gravity phase. An electromagnet mounted on the Hold & Launch Subsystem, part of the CHAMBER System, match precisely a small housing in the interface plate and two small plastic teeth ensures the correct alignment of the CUBE.

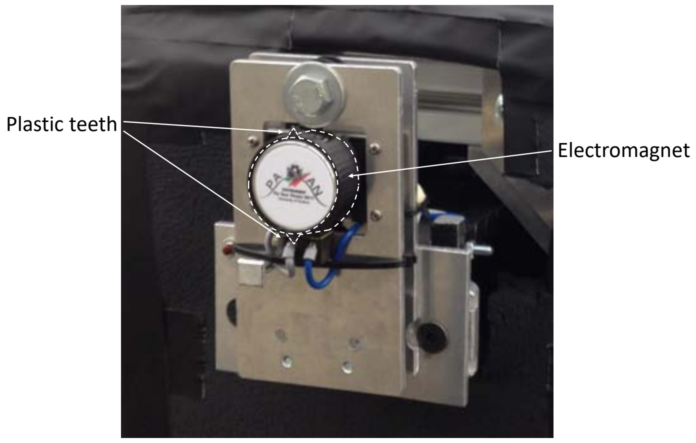


Figure 2.5: Photo of the Hold and Launch subsystem; it is highlighted the plastic teeth that ensures the correct alignment of the CUBE and the electromagnet.

The magnetic field of the electromagnet generates a force on the plate that holds the CUBE in position during pre-test operations. Then, during the low-gravity phases, the electromagnet can be switched off to release the CUBE. Moreover, as it can be noticed from Figure 2.5, in order to ease the release of the Cubesats during the micro gravity phases, a sticker has been attached to the electromagnet to create a little gap between the latter and the iron plate, which help to reduce the problem of the residual magnetization.

2.1.2 CUBE: Navigation Subsystem

The Navigation subsystem of the CUBE includes all the components needed to estimate and control its relative attitude with respect to the FFT.
These are: the Magnetic Coils, as actuators for the rendezvous operation, and the navigation sensors, which are a Camera for visual navigation and an IMU Board for inertial navigation.

Magnetic coils

The Magnetic coils are the actuators of the attitude control system.
To correct the CUBE's attitude with respect to the FFT, the Magnetic coils each generate a small electromagnetic field, whose intensity and direction are controlled by the CUBE's on-board software controller. The electromagnetic fields interact during the free floating phases with the static electromagnetic field generated by the FFT, thus creating a magnetic torque on the CUBE that makes it to rotate and correct its position and attitude.
It is very difficult to extract an analytical expression for the forces and torques generated between two electromagnets of different sizes and not perfectly parallel.
Considering two circuits A and B (shown in Figure 2.6) and their dipole model, authors from [10] proposed the equation of the force between their dipoles as the following:

$$\begin{aligned}
\boldsymbol{F}_{AB} &= \frac{3\mu_0 m_A m_B}{4\pi r^4}\left(\hat{\boldsymbol{r}}\left(\hat{\boldsymbol{m}}_A\cdot\hat{\boldsymbol{m}}_B\right)+\hat{\boldsymbol{m}}_A\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{m}}_B\right)+\hat{\boldsymbol{m}}_B\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{m}}_A\right)\right. && (2.1)\\
&\quad \left. -5\hat{\boldsymbol{r}}\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{m}}_A\right)\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{m}}_B\right)\right)\\
&= \frac{3\mu_0}{4\pi r^4}\left(\left(\hat{\boldsymbol{r}}\times\boldsymbol{m}_A\right)\times\boldsymbol{m}_B+\left(\hat{\boldsymbol{r}}\times\boldsymbol{m}_B\right)\times\boldsymbol{m}_A-2\hat{\boldsymbol{r}}\left(\boldsymbol{m}_A\cdot\boldsymbol{m}_B\right)\right. && (2.2)\\
&\quad \left. +5\hat{\boldsymbol{r}}\left(\hat{\boldsymbol{r}}\times\boldsymbol{m}_A\right)\cdot\left(\hat{\boldsymbol{r}}\times\boldsymbol{m}_B\right)\right)
\end{aligned}$$

where a bold character represents a general vector, the circumflex ˆ represents a unit vector, a plain character represents the magnitude of the vector ($\boldsymbol{r} = r\hat{\boldsymbol{r}}$), μ_0 is the magnetic constant, the permeability of free space, equal to $4\pi \times 10^{-7} NA^{-2}$, $\boldsymbol{r}$ is the vector from the centre of circuit A to the centre of circuit B and $\boldsymbol{m}_A$ and $\boldsymbol{m}_B$ are the magnetic dipole moments of the two circuits, defined as:

$$\boldsymbol{m}_A = \frac{I_A}{2} \oint \boldsymbol{\rho}_A \times \boldsymbol{d}_A, \qquad \boldsymbol{m}_B = \frac{I_B}{2} \oint \boldsymbol{\rho}_B \times \boldsymbol{d}_B \tag{2.3}$$

where I_A and I_B are the currents flowing on the relative circuit and, with reference to Figure 2.6, $\boldsymbol{\rho}_A$ is the vector from the centre of the magnetic dipole A to circuit element $\boldsymbol{d}_A$ and $\boldsymbol{\rho}_B$ is the vector from the centre of the magnetic dipole B to circuit element $\boldsymbol{d}_B$.

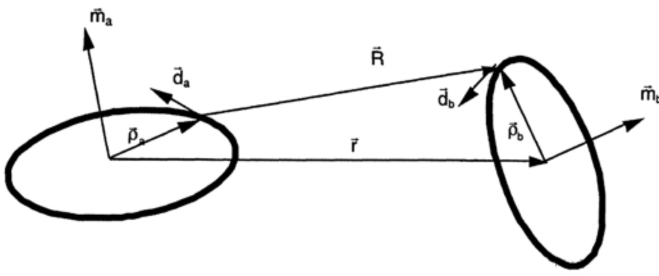


Figure 2.6: Geometry of the two magnetic dipoles. [10]

Furthermore, in [10] authors calculated the resulting torque generated by the dipole A on the dipole B as:

$$\begin{aligned}\boldsymbol{\tau}_{AB} &= \boldsymbol{m}_B \times \boldsymbol{B}_{AB} \\ &= \frac{\mu_0}{4\pi r^5}\left[3\boldsymbol{m}_B \times (\boldsymbol{m}_A \cdot \boldsymbol{r})\,\boldsymbol{r} - r^2\,(\boldsymbol{m}_B \times \boldsymbol{m}_A)\right] \\ &= \frac{\mu_0 m_A m_B}{4\pi r^3}\left[3\,(\hat{\boldsymbol{m}}_A \cdot \hat{\boldsymbol{r}})\,(\hat{\boldsymbol{m}}_B \times \hat{\boldsymbol{m}}_A) + (\hat{\boldsymbol{m}}_A \times \hat{\boldsymbol{m}}_B)\right]\end{aligned} \tag{2.4}$$

where $\boldsymbol{B}_{AB}$ is the magnetic field generated by the dipole A at the location of dipole B.
From this analysis, it is interesting to notice that both the forces and the torques between two electromagnets are inversely proportional to the magnitude of the vector $\boldsymbol{r}$, i.e. the distance between the centre of the two electromagnets. In particular, the equation of the force is inversely proportional to r^4 and the equation of the torque is inversely proportional to r^3, which is very representative of how small the magnetic interactions can be even at relatively small distances.

In order to save space for the electronic boards and to maximize the effectiveness of the control in terms of torque, the 4 Magnetic coils have been located at the four corners of the top layer of the CUBE, as illustrated in Figure 2.7.

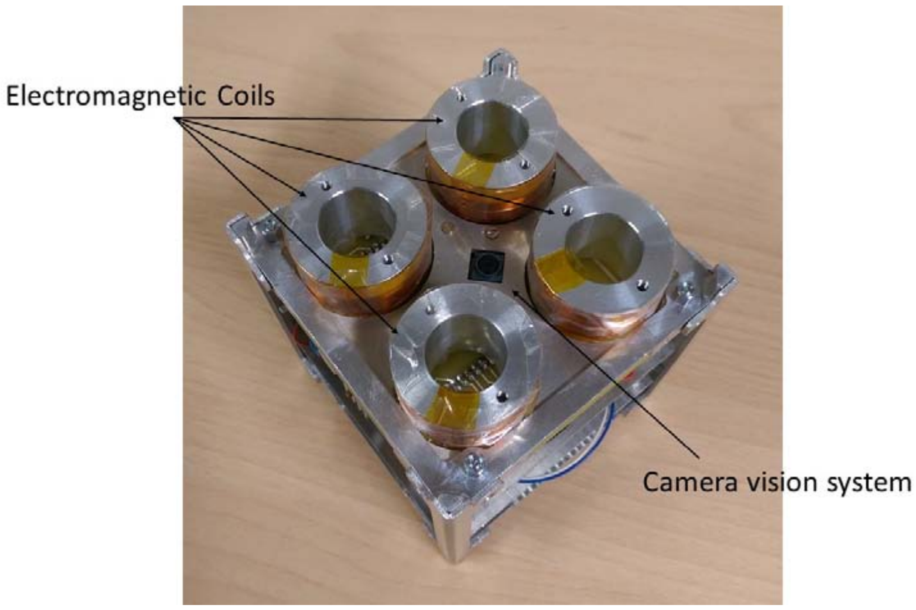


Figure 2.7: CUBE's magnetic coils and vision system.

The solenoids are composed of an insulated copper wire, wrapped on a structural rod without ferromagnetic coil, in order to reduce the total mass of the CUBE. Each coil has an internal diameter of 36 mm and it is composed of 317 windings of 34SWG wire, with a cross section of 0.0429 mm^2.

Camera

The Camera used for visual navigation is the Raspberry Pi NoIR Camera v2 [18]. It is used to estimate the relative position of the CUBE with respect to the FFT from the recorded images.
As it can be seen in Figure 2.7, it is located at the centre of the top layer of the CUBE, with its optical axis oriented towards the motion direction of the CUBE.
Since the camera is used to extract metric informations from the recorded images, it needs to be calibrated. The calibration has been done by means of the technique exposed in [22] by Zhengyou Zhang. He proposed a flexible new calibration technique which only requires the camera to observe a planar pattern shown at a few different orientations. Moreover, the pattern does not need to be an expensive calibration apparatus, but can be printed on a laser printer and attached to a planar surface, solution that gives the opportunity to calibrate the camera in any environment.
An example of a possible pattern is shown in Figure 2.8.

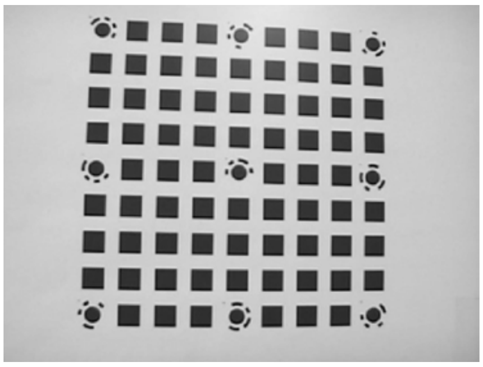


Figure 2.8: Example of pattern used for calibrate the camera. Photo from [22].

The technique is based on an analytical solution with $2n$ equations and 6 unknown parameters, where n is the number of the taken images of the pattern (typically 4-5 images taken at different orientations are enough to obtain a good calibration). The solution is then followed by a nonlinear optimization technique based on the maximum-likelihood criterion, needed to refine the solution.
Finally, once the camera is calibrated, we can proceed with the pose and attitude estimation.
Since the image processing time is related to the number of features in an image plane, some solutions has been taken into account to boost the performances of the vision subsystem.
First of all, the FFT has been equipped with a defined pattern of IR LEDs, whose coordinated are known a priori. Furthermore, to reduce the lens flare effect, they have been covered with paper tape which helps to diffuse the light, as shown in Figure 2.9.

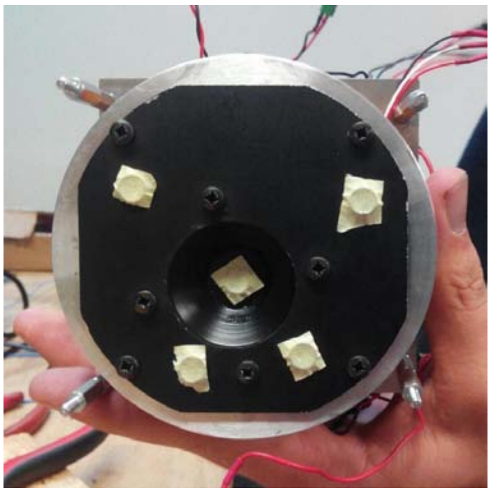


Figure 2.9: Free Floating Target IR led pattern. The IR LEDs are covered with paper tape in order to diffuse the light and reduce the lens flare effect.

Secondly, in order to capture only the brightest features present in the scene, the exposure time of the camera sensor has been reduced to the minimum, being anyway long enough to see the required features.
Finally, through a thresholding operation the captured image is converted to a binary image where the only visible features are the IR LEDs on the FFT.

To estimate the pose and attitude of the CUBE, first define ${}^{T}\boldsymbol{X}_i$ $(i = 1, \ldots, 5)$ as the 3D feature points located on the observed surface (initially know with respect to the target reference frame T) and ${}^{C}\boldsymbol{x}_i$ their relative projections in the camera frame C, as shown in Figure 2.10.
Then, the first step of the measurement procedure is to identify the projections ${}^{C}\boldsymbol{x}_i$ of each 3D feature point ${}^{T}\boldsymbol{X}_i$ $(i = 1, \ldots, 5)$.

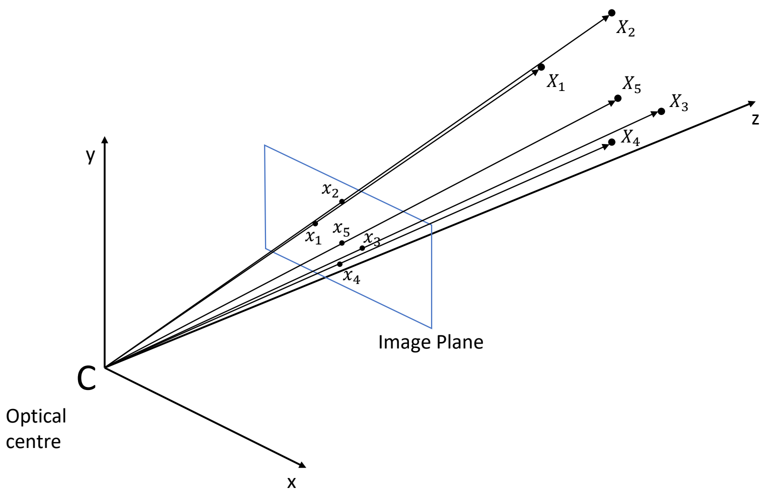


Figure 2.10: Framework of the CUBE's camera.

Once the 2D and 3D points are correctly associated, since the 3D points coordinates of the IR leds are initially known with respect to the target reference frame T, the relative position and orientation between the camera (and so the CUBE) and the target can be retrieved by solving the Perspective-Three-Point (P3P) problem.
The problem has been solved by means of the solver proposed in [9] by Kneip et al., which provide a closed-form solution that computes the aligning transformation directly in a single stage.
The P3P solver, using as input three 3D points ${}^{T}\boldsymbol{X}_i$ and their corresponding 2D projection ${}^{C}\boldsymbol{x}_i$ $(i = 1, 2, 3)$ in the image plane, provides as output up to four rotation matrices ${}^{C}_{T}\boldsymbol{R}_j$ and position vectors ${}^{C}\boldsymbol{t}_{T,j}$ $(j = 1, 2, 3, 4)$. The two represent the trans-

form coordinates from the target reference frame T to the camera reference frame C and the position of the target frame expressed in the camera frame, respectively.
Once the rotation matrices ${}^{C}_{T}\boldsymbol{R}_j$ and translation vectors ${}^{C}\boldsymbol{t}_{T,j}$ $(j = 1, 2, 3, 4)$ are calculated, the fourth and fifth points are then backprojected to the image plane as follows:

$$ {}^{C}\boldsymbol{X}_{i,j} = \left[{}^{C}X_{X_{i,j}}, {}^{C}X_{Y_{i,j}}, {}^{C}X_{Z_{i,j}}\right]^T = {}^{C}_{T}\boldsymbol{R}_j{}^T \boldsymbol{X}_i + {}^{C}\boldsymbol{t}_{T,j} \tag{2.5} $$

$$ {}^{C}\bar{\boldsymbol{x}}_{i,j} = \left[{}^{C}\bar{x}_{X_{i,j}}, {}^{C}\bar{x}_{Y_{i,j}}, {}^{C}\bar{x}_{Z_{i,j}}\right]^T = \left[\frac{{}^{C}X_{X_{i,j}}}{{}^{C}X_{Z_{i,j}}}, \frac{{}^{C}X_{Y_{i,j}}}{{}^{C}X_{Z_{i,j}}}, 1\right]^T \tag{2.6} $$

$$ \text{with} \qquad i = 4, 5 \tag{2.7} $$

Finally, a reprojection error ϵ_j can be calculated as the average of the Euclidean distances $d_{i,j}$ between the two measured points ${}^{C}\boldsymbol{x}_i$ and their projections ${}^{C}\bar{\boldsymbol{x}}_{i,j}$ $(i = 4, 5,\ j = 1, 2, 3, 4)$:

$$ d_{i,j} = \sqrt{\left({}^{C}\bar{x}_{X_{i,j}} - {}^{C}x_{X_i}\right)^2 + \left({}^{C}\bar{x}_{Y_{i,j}} - {}^{C}x_{Y_i}\right)^2 + \left({}^{C}\bar{x}_{Z_{i,j}} - {}^{C}x_{Z_i}\right)^2} \tag{2.8} $$

$$ \epsilon_j = \frac{d_{4,j} + d_{5,j}}{2} \tag{2.9} $$

The algorithm calculates the reprojection error ϵ_j for each combination of three points over the five available. Then, the correct rotation matrix ${}^{C}_{T}\boldsymbol{R}$ and position vector ${}^{C}\boldsymbol{t}_T$ are selected such that minimize the reprojection error.

IMU Board

Finally, as last component of the Navigation subsystem, the IMU Board is the "PhidgetSpatial Precision 3/3/3 High Resolution" [14], responsible for the inertial navigation.
As it can be seen in Figure 2.11, it is located approximately in the centre of the CUBE structure, right above the battery pack.

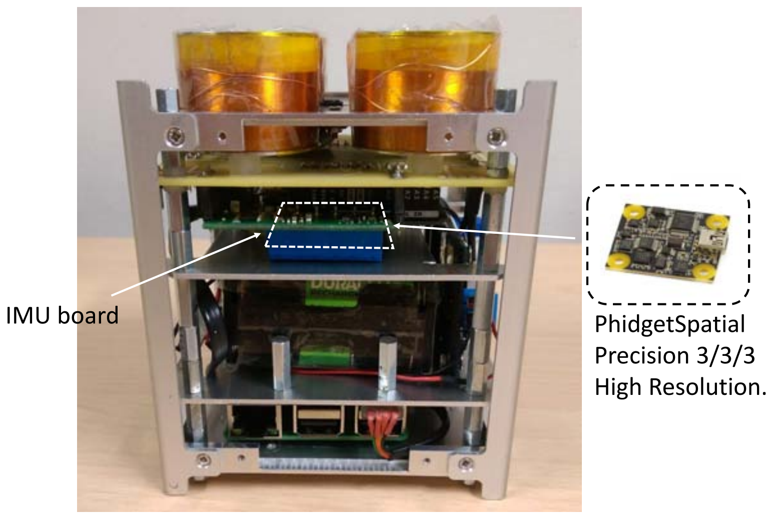


Figure 2.11: IMU Board inside the CUBE.

The IMU board is equipped with:

- an accelerometer with a range of ± 2g and a resolution of 76.3μg, used during the micro-gravity phases to store the accelerations data and detect in post process the actual release instants of the CUBE and any source of disturbance;
- a gyroscope capable to measure a maximum angular velocity of $\pm 400°/s$ on its x and y-axis and $\pm 300°/s$ on its z-axis, which has been used to reconstruct the CUBE motion when the measures coming from the camera were not available, i.e. when the pattern was out of the field of view of the visual navigation system.
- a magnetometer which has not been used, due to the close electromagnetic fields which saturate the sensor.

Moreover, the board has backup devices with reduced performances used in case of fault of the principal sensors.
A datasheet including the complete specifications of the board is reported for completeness in Appendix A.

2.1.3 CUBE: Data Handling & Communication Subsystem

The CUBE Data Handling and Communication subsystem consists of all the components that receive and process data, from which they obtain important information used to control the attitude of the CUBE.
These include:

- The controller boards, which consist of a microcontroller board, the Arduino uno Wifi [1], and a Raspberry Pi 3 model B [17], which communicate together to collect data and implement the control actions;
- The custom Arduino Shield.

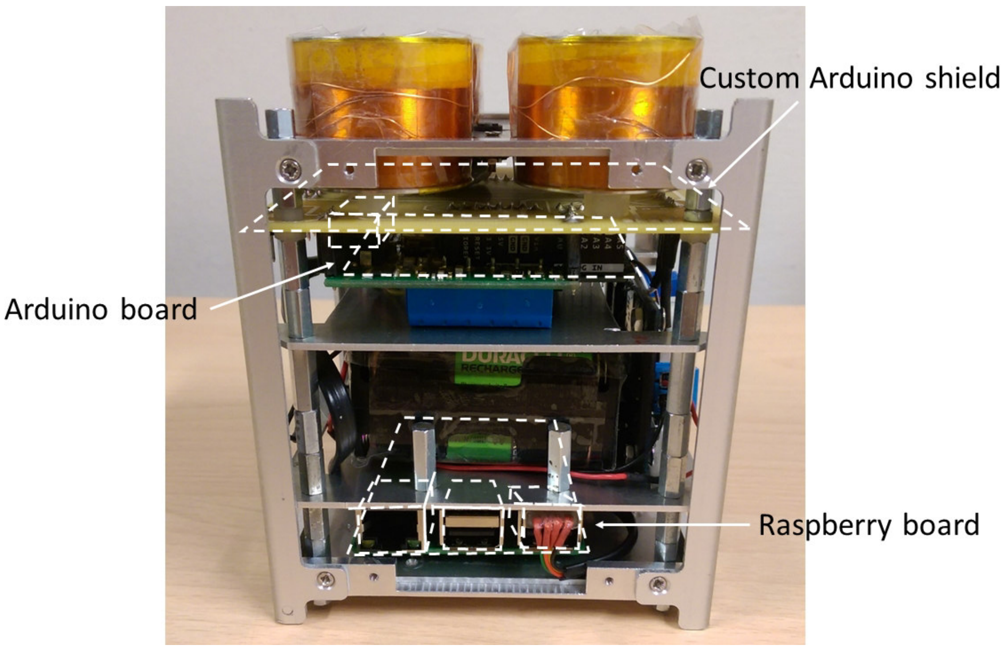


Figure 2.12: CUBE's controller boards and the custom shield.

Controller boards

The controller boards are used for sensor reading, data handling and control logic. The selection of the controller board has been crucial for the experiment: in PAC-MAN we need a small board, capable of fast computing and able to generate high frequency PWM signals to control the coils through the dedicated drivers. After a preliminary selection based on a trade-off between the hardware dimensions, its performances and the ease of use, we finally decided to select two different boards, each able to perform different tasks: a Raspberry Pi 3 model B and an Arduino Uno Wifi edition.
The choice of two different boards has been dictated by the need of having a hardware capable of acquiring images and process them in real-time and at the same time implement the control actions without interruptions due to Operative System's processes. Furthermore, the microcontroller board is equipped with an Analog-to-Digital Converter unit (ADC) to read sensor data from the temperature and proximity sensors,

needed for the proper functioning of the experiment.

In order to estimate the CUBE's current pose with respect to the FFT, the Raspberry Pi board acquire images from the Raspberry Pi NoIR Camera and process them in real-time as explained in the previous Section (2.1.2), retrieving the actual relative position and attitude of the CUBE. Moreover, a parallel thread on the main program acquires the data of the inertial navigation from the IMU board, which sensors are read with a frequency of ≈ 80 Hz.
Each time the IMU sensors are read or an image is processed, the obtained data is stored and processed instantly, using it as an input of a Kalman filter.
Then, the filtered data, output of the Kalman filter, is sent to the Arduino board through serial communication (Figure 2.13).

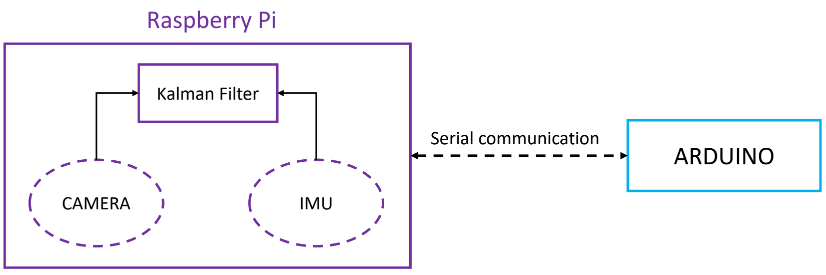


Figure 2.13: CUBE's data handling schematic. The dashed boxes represents threads in the software.

The Arduino Wifi, a circuit board based on the ATMEGA328 microcontroller, is responsible for the computation and implementation of the control actions.
Each time the Arduino board receives a new measurement of the CUBE's attitude coming from the Raspberry Pi board, it uses such information as an input for the controller and then, based on its output, implements the required control actions.
Another task performed by the Arduino board, is that of the analysis of the output of a proximity sensor at the beginning of each parabola, in order to detect when the CUBE has been released from the launch system and start the control actions.
Finally, Arduino constantly keeps under control the internal temperature of the CUBE with two small temperature sensors, placed on a coil and inside the battery pack, in order to ensure that unexpected overheating does not damage the CUBE.

Custom Arduino shield

The custom Arduino shield, namely the Driver Circuit Board, allows the connection between the Arduino board and the rest of the electronic components, such as the coil drivers, the temperature sensors and the proximity sensor. Moreover, since the two controller boards use different logic levels (in particular, Arduino works with 5V signals while Raspberry Pi with 3.3V signals), allows the serial communication between the boards converting the logic level of the serial transmission of Arduino from 5V to 3.3V through an Operational Amplifier wired as a voltage follower, as shown in the schematic in Figure 2.14.
The shield is a Printed Circuit Board (PCB), specifically designed to fit the PC104 standard mounting holes and keep the actuators, namely the coils, fixed on the CUBE. The schematic of the circuit is reported in Figure 2.14 while the designed PCB board is shown in Figure 2.15, where the PC104 as well as the coils mounting holes can also be noticed.

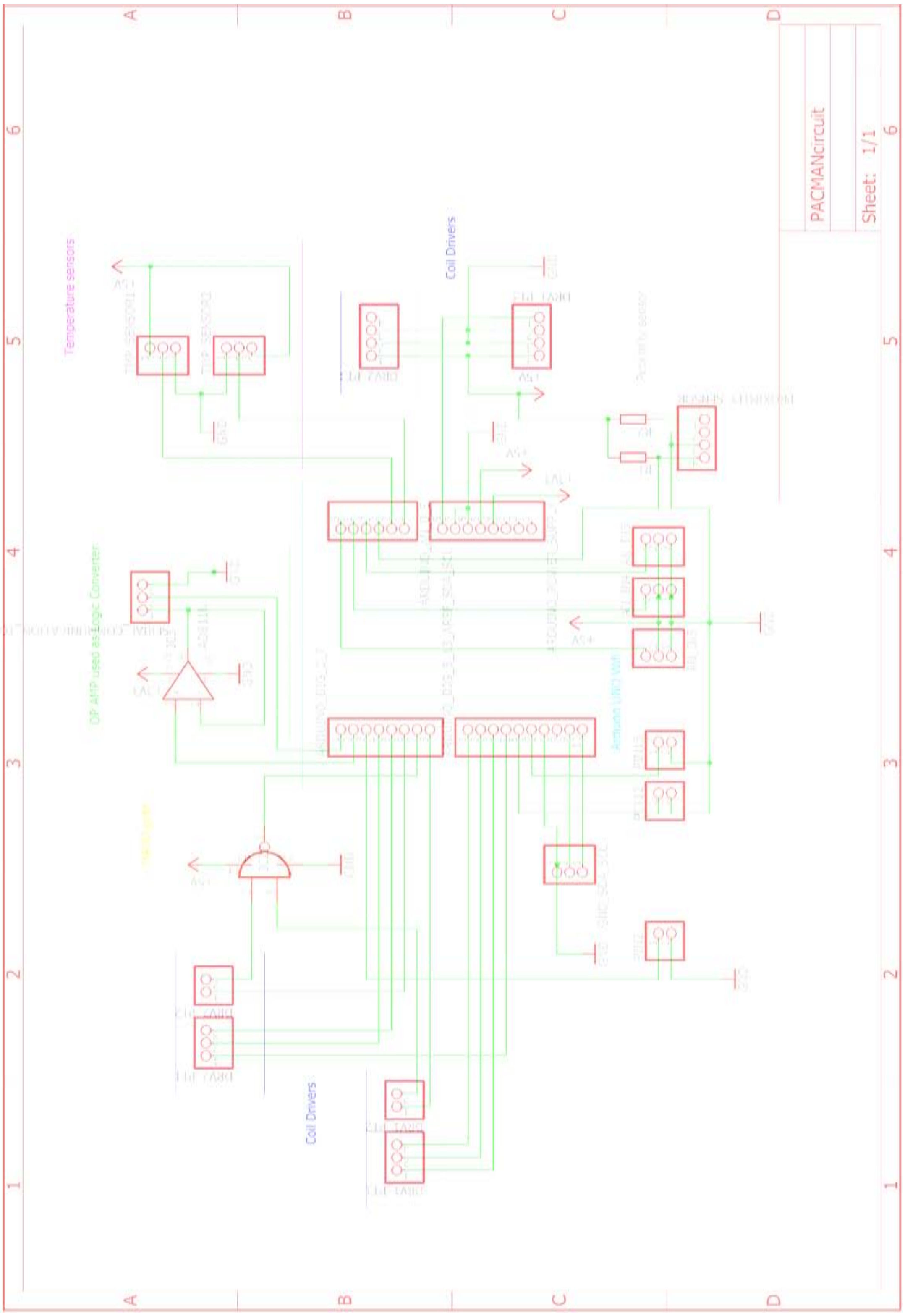
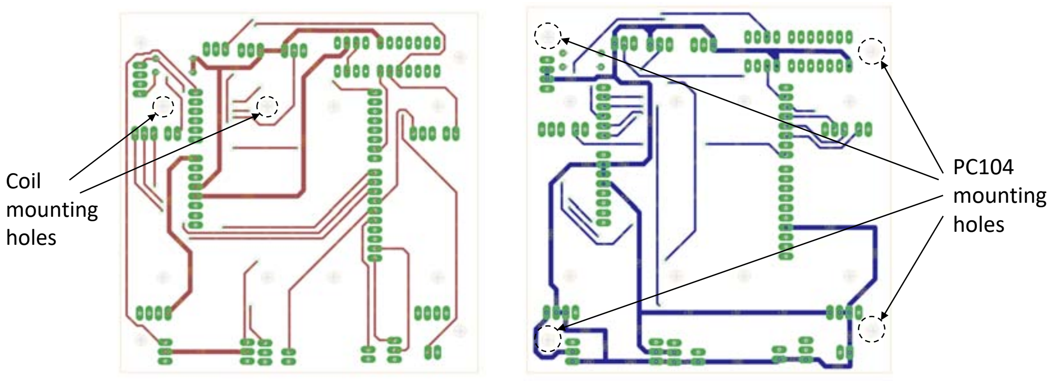


Figure 2.14: Custom Arduino shield schematic. The highlighted parts represent the main components of the board, while the rest is used for debugging purposes.



(a) Top layer. (b) Bottom layer.

Figure 2.15: Custom Arduino shield board.

In order to reduce the signals' noise, the connections from the drivers to the coils, which carry high power ($\approx$ 2W each coil), have been kept out of the PCB, connecting the coils directly to the drivers.
Furthermore, the coil drivers use a dedicated digital output for diagnostic, which offers feedback about the state of the drivers. These pins are open-drain outputs that are driven low by the internal chip to indicate faults, where a low signal indicates an over-current, over-voltage, or over-temperature condition. For this reason, as reported in the schematic, the shield is equipped with a NAND gate, which properly merge the diagnostic pins of the two drivers to indicate faults.

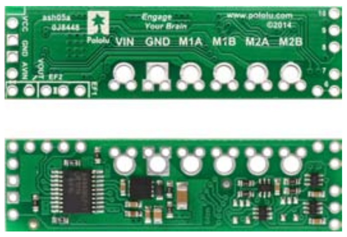
2.1.4 CUBE: Power Subsystem

The Power Subsystem of the CUBE is composed of all the components that supply power to other subsystems, such as the coil drivers, the buck converters and the batteries.

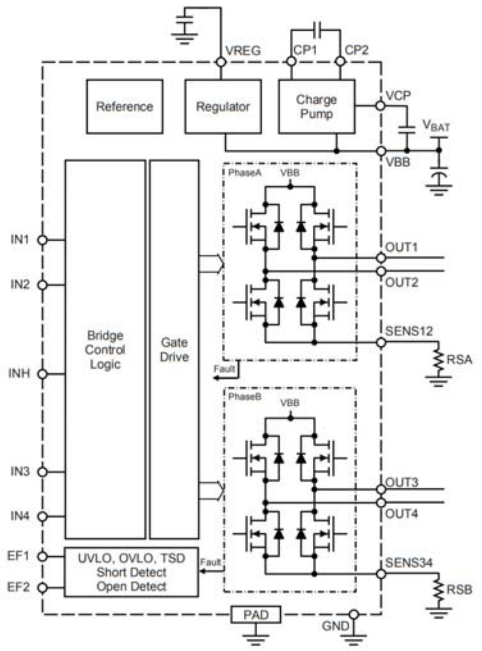
Coil Drivers

The coil drivers are used to supply the proper current to the magnetic coils. They are an essential component for the CUBE, as they allow the microcontroller to implement the control actions by supplying the proper current to each individual coil independently from the others.
After a preliminary selection and laboratory tests, the Pololu A4990 Dual Motor Driver [15] (shown in Figure 2.16a) have been selected as drivers for the electromagnetic actuators.

The board is equipped the Integrated Circuit A4990, a dual H-Bridge driver for small brushed DC motors which operates from 6V to 32V, that fit perfectly our application. Each H-Bridge is capable of supply a maximum continuous current of ≈ 0.65A to the coil and allow the controller to invert the current on the coils with a digital signal. Furthermore, the driver board feature a protective control circuit based on an on-board sense resistor which limits the maximum peak current to 0.9A.
A functional block diagram of the driver is reported in Figure 2.16b.



(a) Pololu A4990 Dual Motor Driver, top and bottom sides.



(b) Functional scheme of IC A4990.

Figure 2.16: Pololu A4990 Dual Motor Driver.

The Pololu Drivers can drive two coils at once, thus two drivers have been used to properly operate the coils. Moreover, the drivers also supply power to Arduino through its Vin pin.
On the other hand, the Arduino board uses two pins to drive each coil through the drivers: the first carry the PWM signal, which modulates the current supplied to the single coil, while the other is a digital output that gives the information about the direction of the current flowing into the coil. A high frequency PWM signal limit the current ripple during the power supply and ensures the correct response of the electromagnetic actuators.

Batteries

The batteries provide power to the electronic boards and to the electromagnetic actuators. The battery pack is composed of 10 Duracell NiMH DX1500H connected in series for a nominal total voltage of 12V.
The Duracell NiMH batteries have been chosen after a risks assessment related to Lithium batteries, which could provide more capacity for the same size. Because of the specific safety constraints required to fly in the parabolic flight, the Lithium batteries are strictly forbidden for free floating objects like the CUBE, thus the NiMH batteries represented the best solution in terms of stored capacity, power drain and safety norms.
The battery pack occupies large part of the room available inside the CUBE and it is the component that contributes most to its weight ($\approx$ 320g up to a total of $\approx$ 1.3kg), for this reason, in order to help balance the cube, it has been placed at the centre of the structure, as shown in Figure 2.4.

Power converters

Finally, buck converters are used to convert the battery pack voltage to the voltage needed by the electronic components. For the Raspberry Pi board power supply it has been used a small buck converter, which drops down the voltage from the power source to 5V, needed from the board to work properly.
For the Coils instead, in order to generate the required electromagnetic field, keeping in mind the coils characteristics such as their resistance and inductance, a bigger buck converter has been used to supply $\approx$ 7V. This also provides the power supply to Arduino through the drivers as already mentioned, and has a digital led screen that indicates the voltage of the battery pack, in order to prevent an over-discharge while using the cube.

Lastly, for completeness, in Figure 2.17 it is reported the complete wiring diagram of the CUBE, where there can be noticed two memories that are also connected to the Raspberry Pi board: a SD card that stores Operative System and data, and a USB memory, which is used as a backup memory to have redundancy of the collected data.

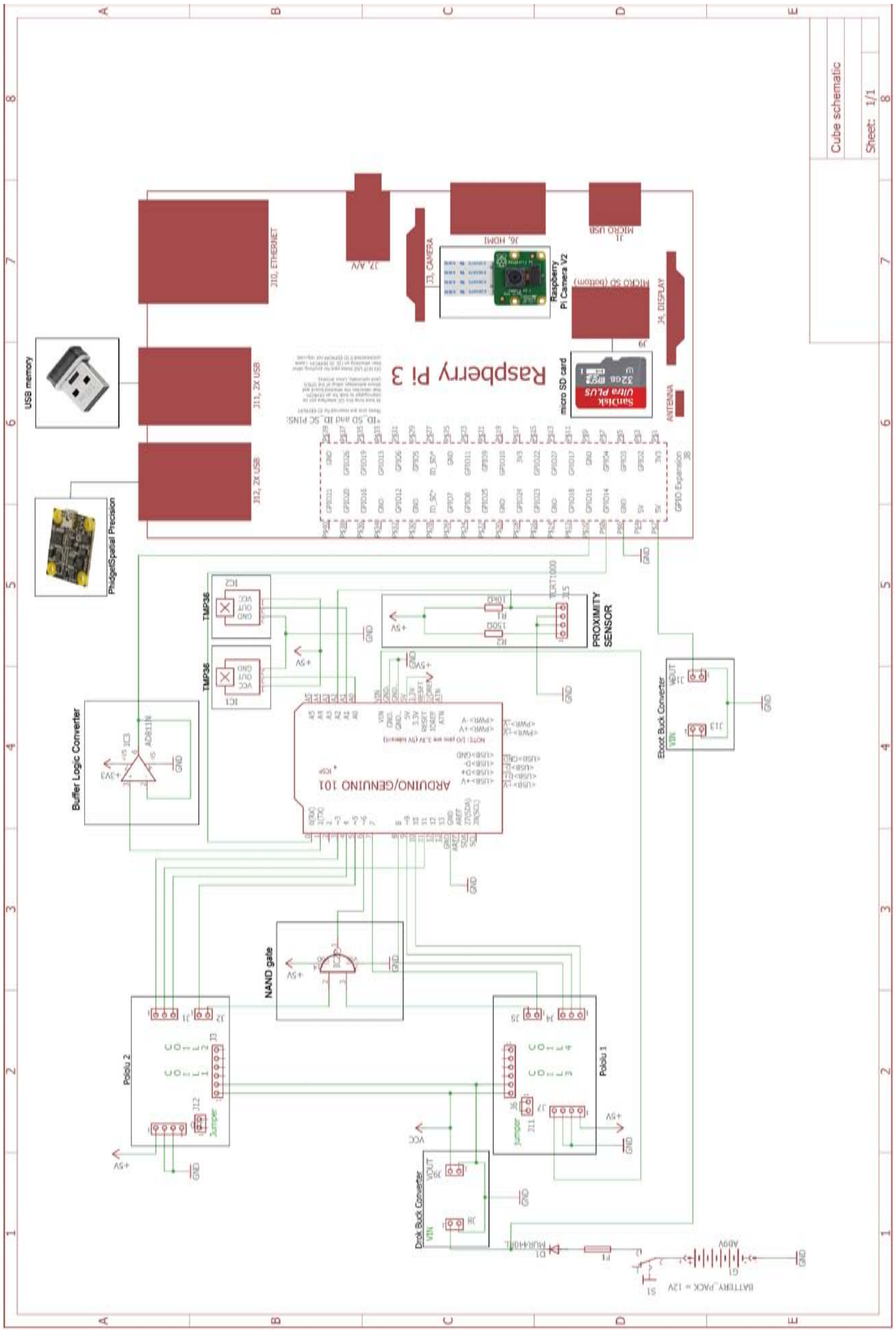


Figure 2.17: CUBE's complete wiring diagram.

2.1.5 CUBE: Software

Given the high number of components that consist of the CUBE, the Software design has been critical to perform a very responsive and reliable system.
All the software is based on the serial communication between the two controller boards, Arduino and Raspberry Pi. Moreover, in order to make debugging easier and do not burden the code, the software has been divided into several libraries that communicate with each other (see Figure 2.18).

Software structure and sequence of operations

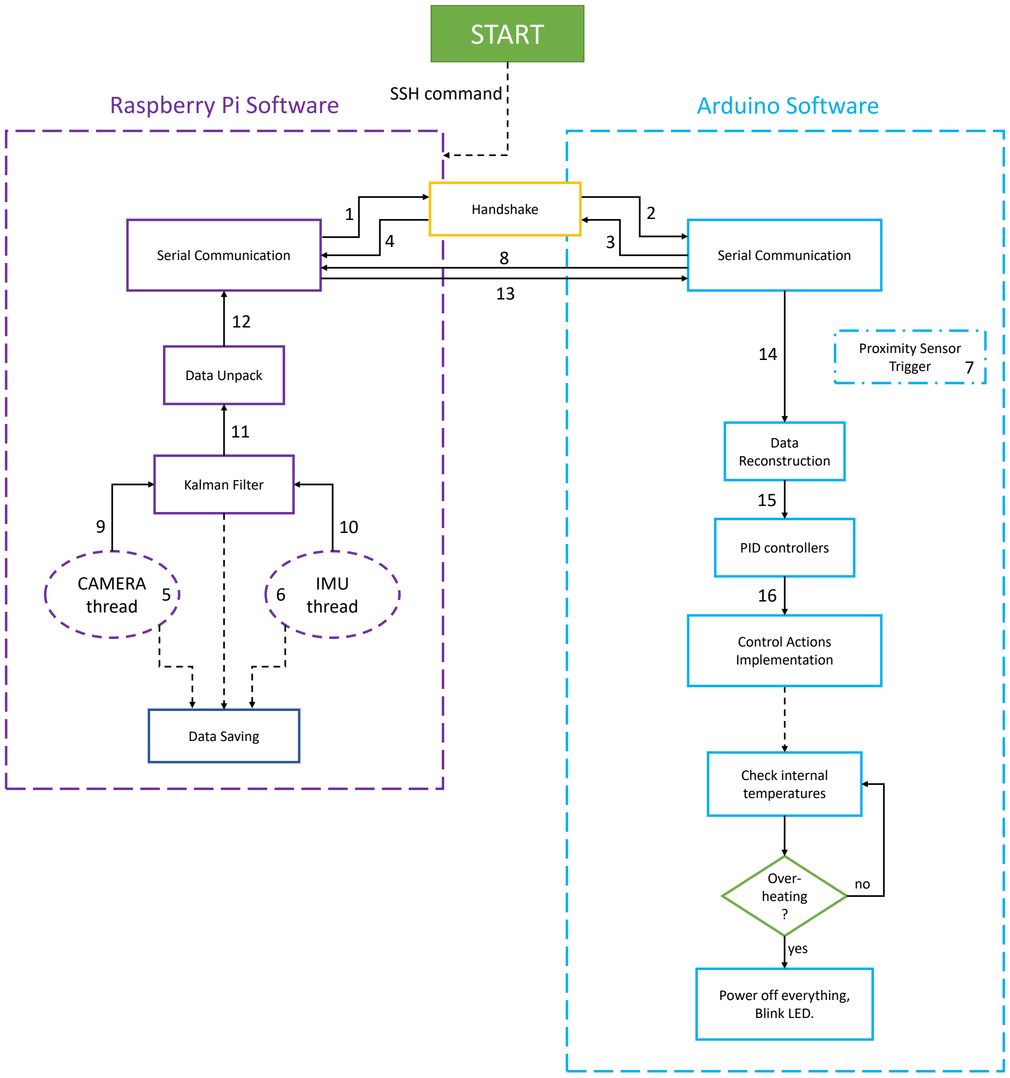


Figure 2.18: CUBE's software block diagram.

The Software on the CUBE is started $\approx 3-5$ seconds after the injection phase of the parabola, through a Secure SHell (SSH) command operated by the user through the main program running on the Laptop.
The SSH command activates the main program on the Raspberry Pi board, which starts the first phase of initialization of the serial communication with Arduino.
Given the fact that the two board's serial communications have been setted up with the same parameters, to initialize them, a similar two-way handshake protocol has been used, in order to synchronize the two boards. In particular, the Raspberry Pi board starts the communication by sending a SYNC byte, which Arduino knows a priori (1-2 in Figure 2.18). When Arduino receives the byte and correctly interprets it, responds with an other SYNC byte to Raspberry, to notify that the synchronization has been performed properly (3-4 in Figure 2.18).
When the synchronization has been completed, the main program in Raspberry Pi starts two threads for the collection of the IMU and camera data (5-6 in Figure 2.18), which initialize and start storing the navigation data.
At this moment the CUBE is still anchored to the Hold & Launch subsystem, thus it must only initialize the data collection threads but not send any data to the Kalman filter or to the Arduino board for the attitude control, as it would be useless.

When the CUBE is released from the Hold & Launch subsystem, the proximity sensor detect this event and sends a trigger signal to the Arduino board (7 in Figure 2.18), which then sends a trigger byte through serial communication to notify the detachment to the Raspberry Pi board (8 in Figure 2.18).
Then, when the trigger byte is received and correctly interpreted by the software on the Raspberry Pi board, the two threads previously initialized are enabled to send the collected data to the Kalman filter (9-10 in Figure 2.18).

The Kalman filter is initialized with the first data coming from the camera, therefore, as the Camera and IMU threads provide data at different frequencies (respectively $\approx$ 30Hz for the camera and $\approx$ 80Hz for the IMU), when the camera thread is processing an image, the Kalman filter analyses the data coming from the IMU to propagate the relative state of the CUBE through the inertial data.
Immediately upon the data is processed, it is sent to the Arduino board through serial communication.
Since the serial communication and the Kalman filter are common resources used by the two threads Camera and IMU, in order to manage the access to the same resource without compromise its performances, a mutex type variable has been exploited, giving the priority to lock the resources to the Camera thread, as it provides the most accurate measures.
Moreover, keeping in in mind that the serial communication is able to send only one byte at a time and the data obtained by the filter are composed of float variables, composed of 4 bytes each, it was necessary to decompose the float variables in pack-

ets of 4 bytes each, before sending them correctly to Arduino. To do this, a function process the memory location of the filtered data and decompose it into 4 bytes before sending it to the Arduino board (11-12-13 in Figure 2.18). Once Arduino receives the decomposed data, it reconstructs them with a reverse function and uses the reconstructed data as an input for the controller (14-15 in Figure 2.18).
Finally, the controller algorithm calculates the required current for each electromagnetic actuator and implements the control actions through a dedicated library specifically written for the drivers described in Section 2.1.4.
Through the drivers, the library controls the current flowing on each coil by modulating the duty cycle of a PWM signal with a frequency of 31.250 kHz, generated by Arduino. Thus, it has been possible to vary the intensity of the magnetic field generated by the CUBE and control its relative attitude with respect to the Free Floating Target.

Moreover, as already mentioned, before and during the implementation of the control actions, the temperature of the coils and the battery pack have been kept under control by two temperature sensors, which are constantly read by the Arduino software. If it detects an overheating of any of the systems, it turn off the coils, terminates any other action and start blinking a LED to notify the experimenters of the overheating.

At the end, after a fixed period of time, when the CUBE and the FFT has already met each others, the main program on the Raspberry Pi stops the threads and closes the serial communication with Arduino by sending a CLOSE byte.
When Arduino receives the CLOSE byte, turn off all the coils and reset itself, ready for the next parabola.

A diagram block of the software structure is reported in Figure 2.18.

Kalman Filter

The estimation of the relative attitude of the CUBE with respect to the FFT is a measure problem where the main issue is represented by the precision of the results. The Kalman Filtering, also known as Linear Quadratic Estimation (LQE), is an algorithm which can be applied to a wide range of applications.
It uses a series of noisy measurements to produce estimates of unknown variables through the estimation of a joint probability distribution over the variables at each iteration.

We recall that the two equations of Kalman Filter can be written in a discrete time domain as follows:

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_{k-1} \tag{2.10}$$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.11}$$

meaning that state values $\boldsymbol{x}_k$ may be evaluated by using a linear stochastic equation (state Equation, Eq. 2.10). In particular, $\boldsymbol{x}_k$ is a linear combination of its previous state value $\boldsymbol{x}_{k-1}$ plus an input signal $\boldsymbol{u}_k$ and a *process noise* $\boldsymbol{w}$. Moreover, the measurement Equation, Eq. 2.11, states that any measurement value $\boldsymbol{z}_k$ is a linear combination of the signal value $\boldsymbol{x}_k$ corrupted by the *measurement noise* $\boldsymbol{v}_k$.
The entities $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ are in general matrices and represents, respectively, the state-transition model, the control-input model and the observation model.
In our system, for simplicity we assume $\boldsymbol{u}_k = 0 \ \forall \ k$ and $\boldsymbol{z}_k = \boldsymbol{x}_k + \boldsymbol{v}_k$, which are the camera measurements coming from the P3P solver described earlier, meaning that $\boldsymbol{H} = \mathbb{I}$.
Besides, keeping in mind that we used quaternions to represent the state of the system and considering the system composed by the CUBE and the FFT as a closed system, we exploited the conservation of angular momentum, which states: *"In a closed system, no torque can be exerted on any matter without the exertion on some other matter of an equal and opposite torque."*. It follows that:

$$\dot{\boldsymbol{x}}_k = \frac{1}{2}\boldsymbol{\Omega}\,\boldsymbol{x}_k \tag{2.12}$$

where $\boldsymbol{\Omega}$ is the skew symmetric matrix extracted from the IMU's gyroscope measurements $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$:

$$\boldsymbol{\Omega} = \left[\begin{array}{c|ccc} 0 & & -\boldsymbol{\omega}^T & \\ \hline & 0 & \omega_z & -\omega_y \\ \boldsymbol{\omega} & -\omega_z & 0 & \omega_x \\ & \omega_y & -\omega_x & 0 \end{array}\right] = \left[\begin{array}{c|ccc} 0 & -\omega_x & -\omega_y & -\omega_z \\ \hline \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{array}\right] \tag{2.13}$$

Thus, the state of the system can be approximated through the Taylor expansion series:

$$\begin{aligned} \boldsymbol{x}_k \simeq & \quad \boldsymbol{x}_{k-1} + \Delta t\,\dot{\boldsymbol{x}}_{k-1} + o\left(\Delta t\right) \\ = & \quad \boldsymbol{x}_{k-1} + \Delta t\left[\tfrac{1}{2}\boldsymbol{\Omega}\boldsymbol{x}_{k-1}\right] + o\left(\Delta t\right) \\ = & \quad \underbrace{\left[\mathbb{I}_{4\times 4} + \frac{1}{2}\Delta t\boldsymbol{\Omega}\right]}_{\boldsymbol{A}}\boldsymbol{x}_{k-1} + o\left(\Delta t\right) \end{aligned} \tag{2.14}$$

and we finally found $\boldsymbol{A}$.

The Kalman filter algorithm works in a two-steps process iterated: a prediction step, also called Time Update, and a correction step, also called Measurement Update.
In the prediction step the Kalman filter uses the inertial navigation data to produce

estimates of the current state variables, while in the correction step updates the estimates with the camera measurement using a weighted average, giving more weight to estimates with higher certainty.
For the two steps, we have two distinct set of equations:

Time Update

Project the state ahead:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\,\hat{\boldsymbol{x}}_{k-1}^- \tag{2.15}$$

Project the error covariance ahead:

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q} \tag{2.16}$$

where $\hat{\boldsymbol{x}}_k^-$ is the prior estimate, obtained before the measurement update correction and $\boldsymbol{P}_k^-$ is the prior error covariance.
Furthermore, $\boldsymbol{Q}$ is the covariance of the process noise.

Measurement Update

Compute the Kalman gain:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \left(\boldsymbol{P}_k^- + \boldsymbol{R}\right)^{-1} \tag{2.17}$$

Update the estimate via $\boldsymbol{z}_k$:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \hat{\boldsymbol{x}}_k^-\right) \tag{2.18}$$

Update the error covariance:

$$\boldsymbol{P}_k = \left(\mathbb{I} - \boldsymbol{K}_k\right) \boldsymbol{P}_k^- \tag{2.19}$$

where $\hat{\boldsymbol{x}}_k$ is the estimate of the state of the system at time k, $\boldsymbol{P}_k$ is the error covariance, $\boldsymbol{R}$ is the covariance of the measurement noise and $\boldsymbol{K}_k$ is the Kalman gain, the most important Equation of the set of equations.

To start the process, the initial state has been set by a camera measurement and for simplicity, the error covariance matrix $\boldsymbol{P}_0$ has been initially set to $\mathbb{I}$.
Lastly, the covariance matrices of the process noise and the measurement noise have been calculated starting from the error noise of the sensors. For the camera, from laboratory tests we found a suitable covariance matrix of the measurement noise as:

$$\boldsymbol{R} = \begin{bmatrix} 0.9992 & 0 & 0 & 0 \\ 0 & 0.0019 & 0 & 0 \\ 0 & 0 & 0.0219 & 0 \\ 0 & 0 & 0 & 0.0348 \end{bmatrix} \tag{2.20}$$

while for the process noise, for simplicity, we used the following matrix:

$$\boldsymbol{Q} = \begin{bmatrix} 0.9999989696 & 0 & 0 & 0 \\ 0 & 0.0008283434 & 0 & 0 \\ 0 & 0 & 0.0008297180 & 0 \\ 0 & 0 & 0 & 0.0008283434 \end{bmatrix} \tag{2.21}$$

extracted from the value of the Gyroscope White Noise σ reported in the datasheet of the IMU board (Appendix A).

The whole Kalman filter library was initially written for the Arduino board, exploiting C++ pointers in order to allocate the minimum required memory for the variables, due to the hardware restriction of the ATMEGA328 microcontroller which has a total SRAM memory (Static Random-Access memory, used to store the variables) of just 2048 bytes.
Unfortunately, after some tests the 8bit-microcontroller resulted to be not fast enough to handle the filter computations and for this reason, the whole Kalman filter has been executed by the Raspberry Pi board.

Controller

The CUBE system has been designed to be a completely autonomous system, thus the attitude control is completely autonomous and does not need any human intervention.
The automatic control aims to modify the behaviour of the system through the elaboration of its inputs or a function of them.
With the term “control”, we define the action performed to bring and keep a physical parameter at a prefixed value. In our application, the inputs of the controller are represented by two over the three angle outputs of the Kalman filter, namely the Roll and Pitch angles in the CUBE's reference frame, shown in Figure 2.19.

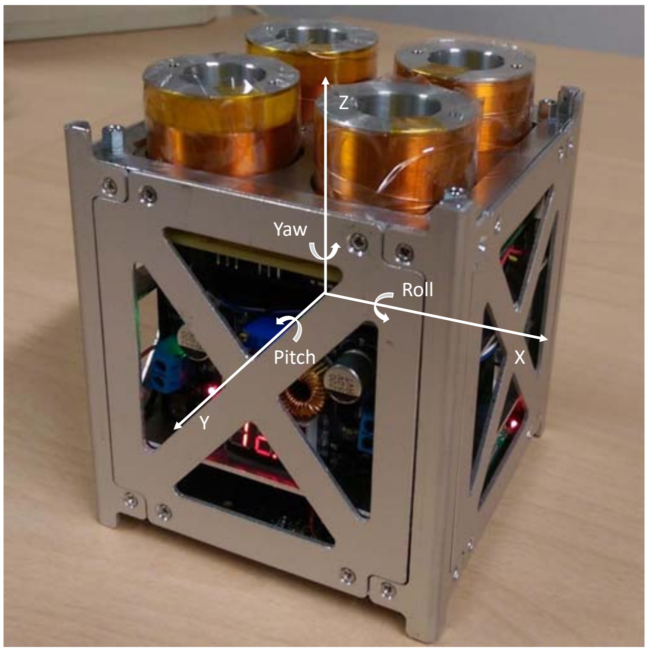


Figure 2.19: CUBE's Reference Frame.

We can model the controller as a generic block that requires an input stress to produce an appropriate output. In this regard, since we would control two degrees of freedom of the CUBE system, namely the Roll and Pitch angles, we designed a controller that consists of two identically, independent PID controllers working in parallel, each controlling one degree of freedom.
Denoting the CUBE's coils as in Figure 2.20, the output of each PID regulator is then used to set the current flowing on the coils in pairs; in particular, the PID controlling the Roll angle applies to the coils A and B a positive signal, while a negative signal with the same magnitude is applied to the coils C and D. The PID controller for the

Pitch angle instead applies a positive control to the coils B and D and a negative control to the coils A and C. Therefore, the individual control signals for each coil are linearly combined and the resulting signal is then applied to control the coils correctly.
Applying simultaneously the same control actions with positive and negative direction of the current flowing on the coils, provide the CUBE a faster and stronger overall control action, which allows to recover angles of misalignment much larger than those recoverable if only a direction of the current flow was possible.

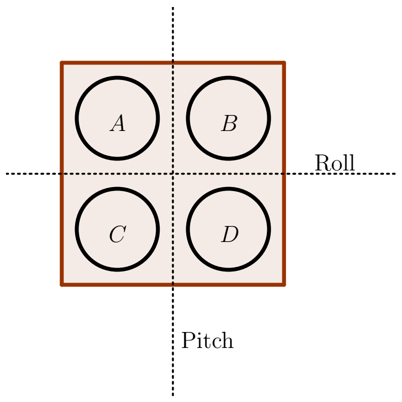


Figure 2.20: CUBE's coils schematic.

The PID regulators are control loop feedback mechanisms widely used in control applications.
They are, at least in their ideal form, SISO (Single Input-Single Output) dynamic systems, improper, as the input $u(t)$ directly influences the output $y(t)$, linear and continuous-time systems, because they deal with real systems.
The PID controller continuously calculates an error parameter $e(t)$ as the simple difference between the measured process, in our case the Roll or Pitch angle, and a Setpoint value, trying to correct the measured process through a correction based on three different control laws: a Proportional action, an Integral action and a Derivative action, which summed together give the controller its name.

In our experiment, we aim to realize a perfect soft docking of the two nano-satellites controlling their relative attitude in order to bring to zero the measured angles, namely setting to zero the Setpoint values of both the PID controllers for the Roll and Pitch angles.
An advantage of using PID regulators is given by the fact that, since they only relies on the measured process variable and not on the knowledge of the underlying process,

they are extremely simple to design and have always the same structure

$$u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)d\tau + K_D \cdot \frac{de(t)}{dt} \tag{2.22}$$

or, equivalently

$$u(t) = K_P \left(e(t) + \frac{1}{T_I} \int_0^t e(\tau)d\tau + T_D \frac{de(t)}{dt} \right) \tag{2.23}$$

where K_P, K_I and K_D are respectively, the proportional, derivative and integral gains while $T_I = \frac{K_P}{K_I}$ and $T_D = \frac{K_D}{K_P}$ are called integral and derivative time.

In the following, in order to better understand the controller subsystem, it is given a brief description of how the PID regulator works and how its single actions affects the control.

The **proportional** action is so called because its input $e(t)$ and output $u(t)$ are algebraically linked by a coefficient K_P.

$$u_P(t) = K_P \cdot e(t) \tag{2.24}$$

As a result, the regulator provides a control action that is proportional to the current error, decreasing the time response of the system with increasing of the proportional gain K_P. Indeed, a large proportional gain results in a large output of the controller even for small errors. Anyway, if the proportional gain is too high, the resulting controlled system can become unstable, i.e. its output diverges and is limited only by saturation of the controlled actuators. In contrast, if the proportional gain is too small, the control action may result to be too weak when responding to system disturbances.

The **integral** action instead, is proportional to both the magnitude and the duration of the error, since it is proportional to its integral over time. The integral term is given by the control law:

$$u_I(t) = K_I \cdot \int_0^t e(\tau)d\tau \tag{2.25}$$

so that its action is given by the sum of the instantaneous error over time multiplied by the integral gain K_I.

The integral action permits to bring to zero the error faster and eliminates the residual Steady State Error. In fact, through the integral action, even a very small positive error will lead to an increment of the control action to correct it, and a negative error

will lead to a decrement of the same in order to bring to zero the total error.
However, since the integral action is based on the accumulated errors from the past, the limits of a real scenario, such as the saturation of the controlled actuators, can cause high overshoots and a long settling time until the accumulated error is unwound.
This phenomena is known as “integral Wind-Up” and is caused by the saturation of the control actions, which lead to an accumulated error larger than the maximal regulation value.
However, this problem can be addressed by implementing a saturation detection mechanism, used within the controller to detect when the actuator output is saturated.
When a saturation occurs, the integrator input is compensated by a signal proportional to the amount of saturation occurring in the actuator.
In Figure 2.21 it is shown a possible implementation of the “anti Wind-Up” mechanism through Simulink blocks.

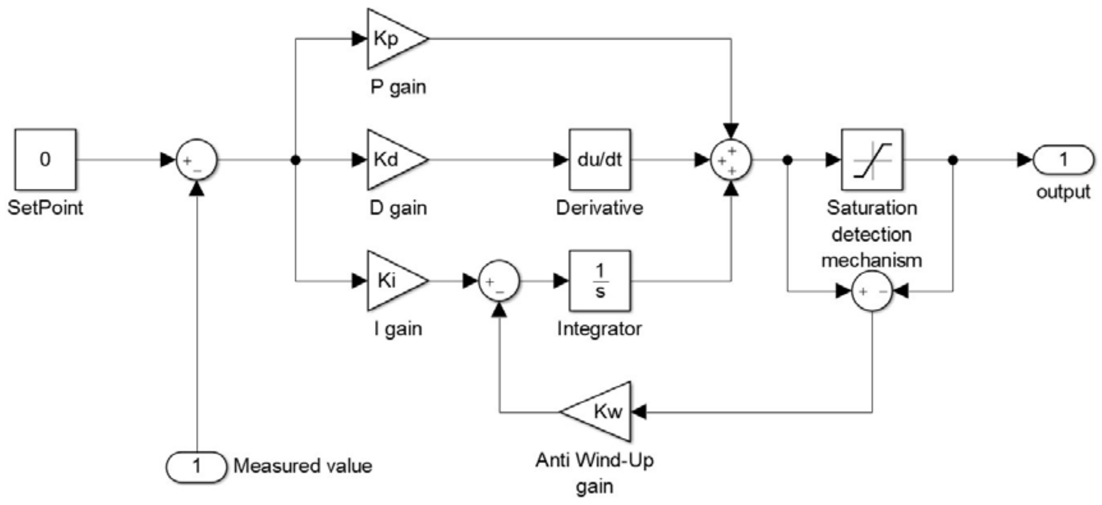


Figure 2.21: PID controller scheme with anti-windup mechanism.

Finally, the **derivative** action provide to the controller’s output the derivative of the process error, calculated by determining the slope of the error over time, which is then multiplied by the derivative gain K_D. The derivative term is so represented by the control law:

$$u_D(t) = K_D \cdot \frac{de(t)}{dt} \tag{2.26}$$

The derivative action is used to predict the system behaviour thus improving the stability and settling time of the controlled system. For this reason, the derivative

action is particularly important to control phenomena that occur in a very short time, although it is quite sensitive to noise in the error signals.
However, in real applications the derivative action together with the proportional action can lead to spikes in the output signal when a system is subjected to an instantaneous step change in the error, such as a large Setpoint change. In the case of the derivative term, since the error is calculated as $e(t) = \text{Setpoint}(t) - \text{Input}(t)$, any change of the Setpoint value lead to an instantaneous change of the error. The derivative of this change is ideally infinite, but in a real scenario it results in a very large value, thus causing the spikes in the output signal.
Indeed, since the error derivative is given by

$$\frac{de(t)}{dt} = \frac{d\,(\text{Setpoint}(t) - \text{Input}(t))}{dt} = \frac{d\,\text{Setpoint}(t)}{dt} - \frac{d\,\text{Input}(t)}{dt}, \tag{2.27}$$

A possible solution to this problem, is to consider the derivative of the process variable

$$\frac{de(t)}{dt} = -\frac{d\,\text{Input}(t)}{dt} \tag{2.28}$$

since when the Setpoint value is constant, its derivative is zero.

In our application, the change of the Setpoint is not concerned because it is always set to 0, thus this phenomena does not affect our experiment.

After the analysis of the single control actions that contribute to a PID controller, an other interesting aspect is that of the correct tuning of the PID parameters, namely the proportional, integral and derivative gains.
There are several methods for tuning a PID loop, the most effective usually involves the development of a process model, which can be very useful for systems with long response times. However, since our application has relatively fast response times, for simplicity only the manual tuning will be taken into account.

The manual tuning is based on a trial-and-error approach, keeping in mind the effects of increasing the single parameters independently.
We can resume these actions in the following table:

Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**Steady-state Error**
Proportional K_P	Decrease	Increase	Small Change	Decrease
Derivative K_D	Small Decrease	Decrease	Decrease	None
Integral K_I	Decrease	Increase	Increase	Eliminate

An effective method for manual tuning the PID parameter can be that of initially increasing the proportional gain, keeping to zero the other two terms. Once the controlled system start oscillating (as the onset of an unstable behaviour), the value of K_P can be set approximately half of the value for which the oscillations started. Then, the integral gain K_I can be increased, until any offset is corrected in sufficient time for the process. Finally, the derivative gain should be increased, until the controlled system is sufficiently quick to reach the Setpoint value.

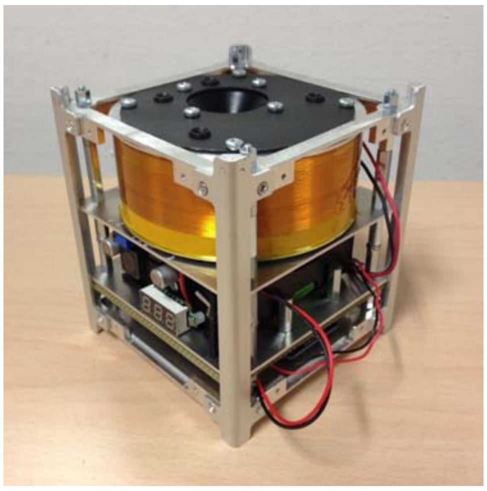
In order to find a good set of parameters for correctly tune the two PID controllers, this technique has been exploited in simulation, giving as a result the following parameters:

$$K_P = 8, \qquad K_I = 0, \qquad K_D = 3.6 \tag{2.29}$$

where the integral gain has been kept to zero because of the fast dynamics of the experiment, obtaining two PD controllers.
During the first flight, this set of parameters has been tested for both the PID controllers, but the resulting system was a bit slow, meaning that the time response of the system was too high (see Section 4.2).
Successively, in order to speed up the attitude recovery process, during the two subsequent flights the proportional gain has been increased up to the value of $K_P = 12$, keeping the other two terms unchanged.

The PID controller in Arduino software has been implemented by means of the "Arduino-PID-Library" developed by Brett Beauregard [2], which includes all the solutions to the PID problems outlined here, resulting in a very robust and reliable algorithm.

2.2 Free Floating Target (FFT)


(a) FFT without the side plates

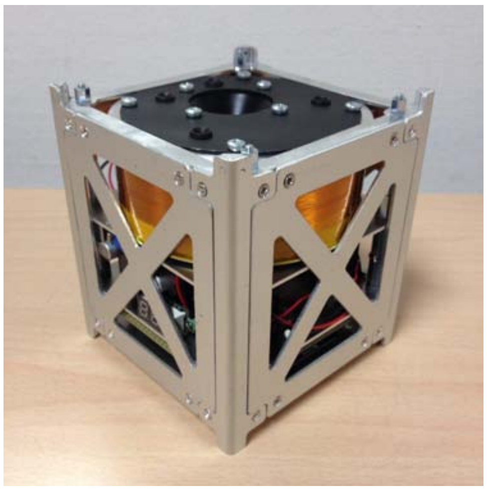

(b) Fully assembled FFT

Figure 2.22: Photos of the Free Floating Target.

The Free Floating Target, shown in Figure 2.22, is the second CubeSat of the experiment, launched simultaneously with the CUBE one towards the other, in order to reproduce a possible real scenario in microgravity where the CUBE approach the Target satellite orienting its asset by means of electromagnetic interactions with it.
The FFT system can be divided into five different subsystems:

- the structure;
- the Target electromagnet;
- the Data Handling and Navigation subsystem;
- the power subsystem;
- the Software.

2.2.1 FFT: Structure

The design of the structure for the FFT is exactly the same of the CUBE, described in Section 2.1.1. The main difference between the CUBE and the FFT structure, is represented by the intermediate layers which keep the electronic components fixed inside the CubeSats.
As it can be seen in Figure 2.23, they are represented by:

– the Led Pattern layer, where the IR LEDs that helps the reconstruction of the relative pose between the two spacecrafts are fixed;
– the Target layer, where the target electromagnetic coil is fixed;
– the Battery layer, which keeps the batteries and part of the power subsystem, including the switch that power on the FFT, fixed;
– the driver circuit board, where the microcontroller, part of the sensors and the rest of the power subsystem are fixed.

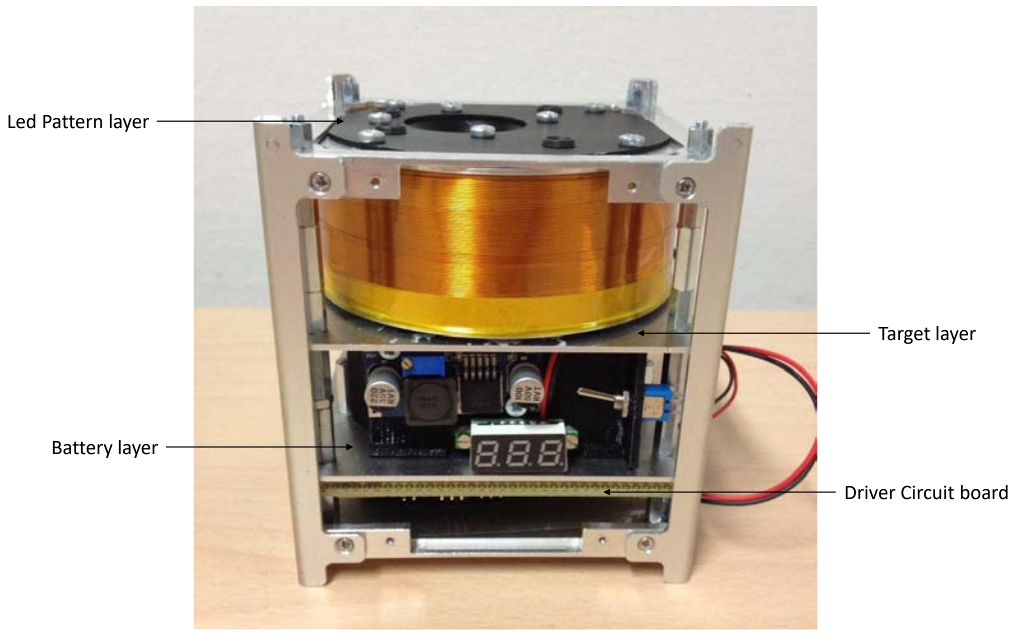
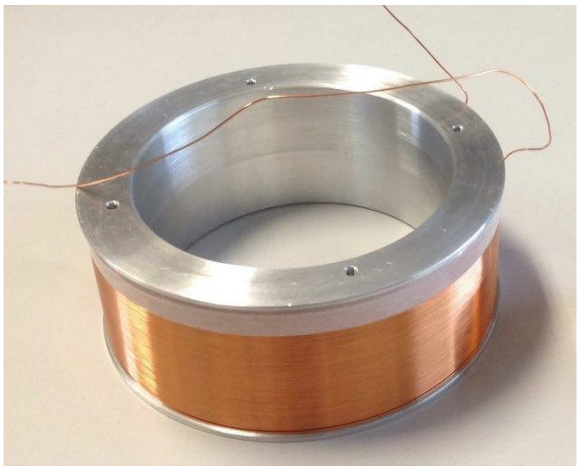


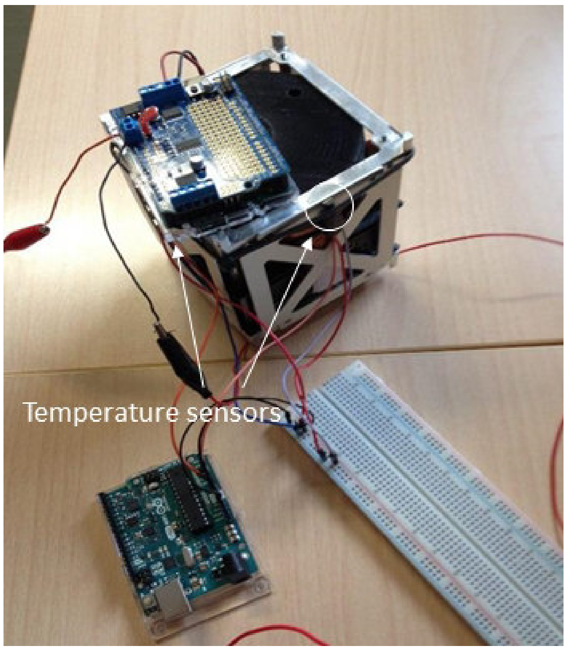
Figure 2.23: FFT layers.

Finally, as in the CUBE system, at the bottom of the structure it is mounted a Hold & Launch Interface, composed of an iron plate which holds the FFT in position during the hyper-gravity phases and release it during the micro-gravity phases through a small electromagnet mounted on the Hold & Launch subsystem.

2.2.2 FFT: Target Electromagnet

The Target Electromagnet, shown in Figure 2.24a, is the main component of the FFT system. It creates an electromagnetic field that is exploited by the CUBE to control its relative alignment.
The target electromagnet is composed of a copper wire wrapped on an aluminium structural rod, without ferromagnetic core to reduce the overall FFT's weight.
The wire used to form the solenoid has a cross section of 0.164mm^2 (26SWG) and it has been insulated by using a polymer coating.
The coil is composed of 375 turns and it has been powered through the power subsystem with a current of $\approx$ 1.2A, thus creating a magnetomotive force of $\approx$ 450At (Ampere-turn). Moreover, a layer of Kapton tape has been added to the coil in order to protect its windings.


(a) FFT coil.



(b) FFT mock-up with the temperature sensors positioned in two significant positions.

Figure 2.24: Free Floating Target coil and setup for temperature tests made in laboratory.

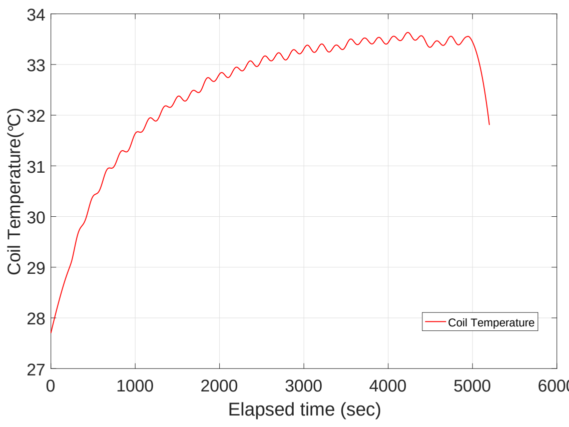
Because of the high power consumption of the electromagnet, a temperature test has been necessary to ensures that the FFT structure did not exceed the maximum allowable temperature of 49°C, thus fitting the safety requirements needed to fly.

The temperature sensors have been placed on the coil surface and on the external surface of the structure, as shown in Figure 2.24b, in order to evaluate the temperature behaviour of both the solenoid and the structure of the FFT.
To simulate flight conditions, recalling the time duration of each parabola in Table 2.1, during the test the electromagnet has been powered on for a period of 20s and switched off for 150s, repeating this cycle for 30 times.

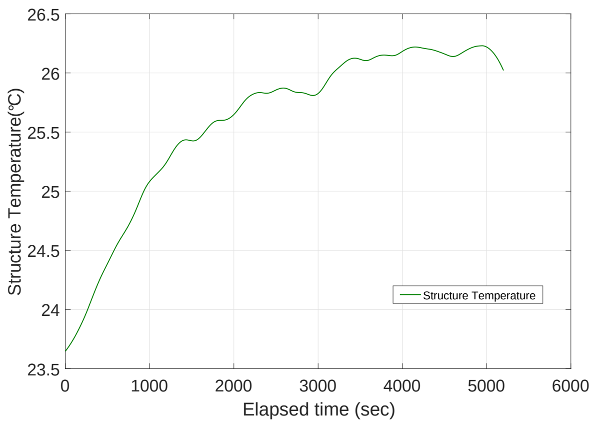
Duration [s]	**Phase**
≈ 20	Hyper Gravity
≈ 4	Transition
≈ 20	Low Gravity
≈ 4	Transition
≈ 20	Hyper Gravity
≈ 110	Normal Gravity

Table 2.1

The tests, performed in an environment at 23°C, shown that the temperature of the electromagnetic coil in the simulated flight conditions does not exceed 34°C, while the structure external surface does not exceed 27°C, which is compliant with the safety requirement needed to fly.
The results of this test are reported in Figure 2.25.



(a) Coil Temperature



(b) Struct Temperature

Figure 2.25: FFT Temperature trend during the test, considering a possible real mission profile.

2.2.3 FFT: Data Handling & Navigation Subsystem

The Data Handling and Navigation Subsystem of the FFT consists of all the components used to retrieve, manage and store useful data for the correct functioning of the same and post processing.
These can be identified as:

- the controller board, which consists of an Arduino Uno wifi;
- the custom Arduino shield, which also includes the driver used to power the electromagnetic target.

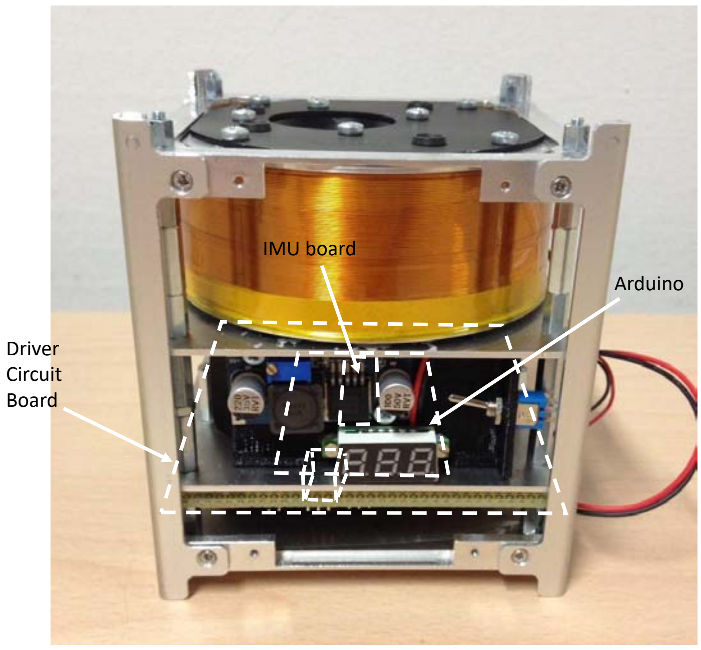


Figure 2.26: Free Floating Target Data Handling & Navigation Subsystem.

Controller board

The controller board is used for sensor reading, data handling and properly driving the current into the target electromagnet when the FFT is released in the microgravity phases.
It is composed of an Arduino Uno Wifi board, since no large data analysis is needed in real time.
Through a proximity sensor, as in the CUBE system, the board detects when the FFT is released from the Launch subsystem and then power on the electromagnetic coil.

Furthermore, when the FFT is released during the microgravity phases, it constantly reads and stores the data coming from an IMU board mounted on the custom Arduino shield, in order to obtain further informations about the entire system dynamics for post processing.
The IMU board of the FFT is the "Sparkfun 9 Degrees of Freedom - Sensor Stick" (Figure 2.27), which communicates to the Arduino board through a simple I2C interface.

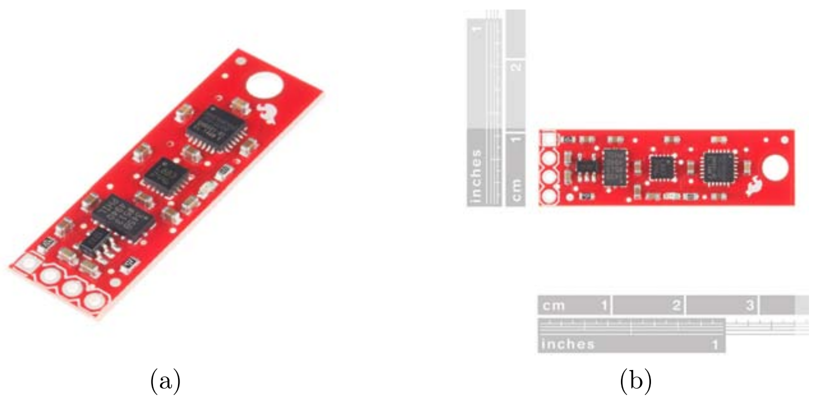


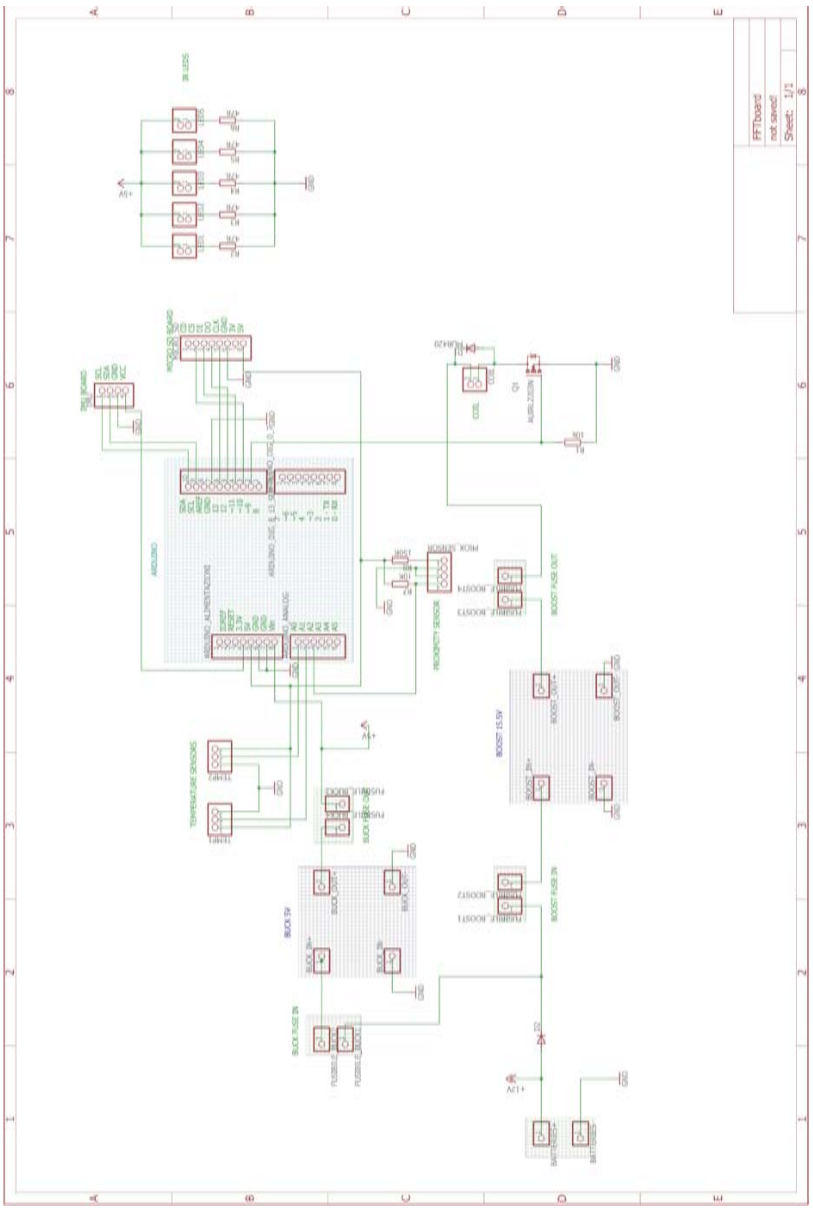
Figure 2.27: Free Floating Target IMU board, the "Sparkfun 9 Degrees of Freedom - Sensor Stick".

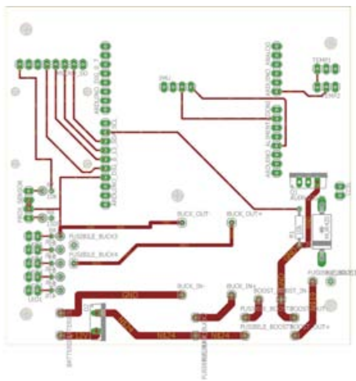
Thanks to its tiny dimensions it has been placed between the Arduino board and the custom shield, as highlighted in Figure 2.26, allowing to save space for other important components.
Finally, in order to keep under control the internal temperature of the Free Floating Target, a process in Arduino's software constantly checks the temperature of the electromagnet and the battery pack from two temperature sensors, as the two are the most critical components in terms of overheating.

Custom Arduino shield

As in the CUBE system, due to the high number of electrical components connected together to form the FFT, it has been necessary to design a custom Arduino shield, to keep them in place and connect them properly.
Again, the shield has been specifically designed to fit the PC104 standard mounting holes.
Through a buck converter mounted on the shield, the microcontroller board and the IR LED pattern are powered when the FFT is switched on. Moreover, the shield connects the IMU board and a micro-SD card board to Arduino, which allows to

collect and store the data of the inertial navigation.
As already mentioned, two temperature sensors are used to keep under control the temperature of the critical components inside of the CubeSat, and are connected to Arduino through the custom shield. Furthermore, a proximity sensor and its driver circuit have been connected to the shield, which power it and brings its output signal to Arduino. Finally, the custom shield for the Arduino board include the driver circuit that allows to power on the electromagnet when the microgravity phase and the release of the FFT are detected. Since in the FFT system it is not needed to modulate the current flowing into the coil, a simple MOSFET used as a switch has been utilized as the driver circuit, which carry the current from a boost converter, that increases the voltage of the battery pack up to 15.5V, to the electromagnetic coil.
The complete schematic and the PCB board designed are reported in Figures 2.28a and 2.28b-2.28c, respectively.


(a) Custom Arduino shield schematic.


(b) Custom Arduino shield board: Top layer.

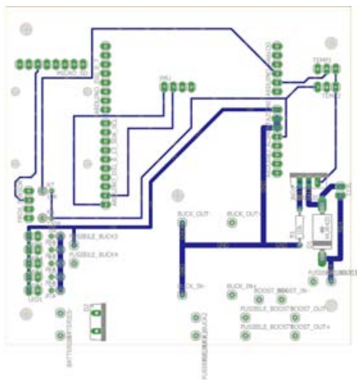

(c) Custom Arduino shield board: Bottom layer.

Figure 2.28: Custom Arduino shield schematic and prototyped board for the Free Floating Target.

2.2.4 FFT: Power Subsystem

The Power Subsystem of the Free Floating Target is composed of all the components which supply power to other subsystems.
The power subsystem includes the battery pack, the buck converter connected to the shield and a boost converter, used to power the coil.

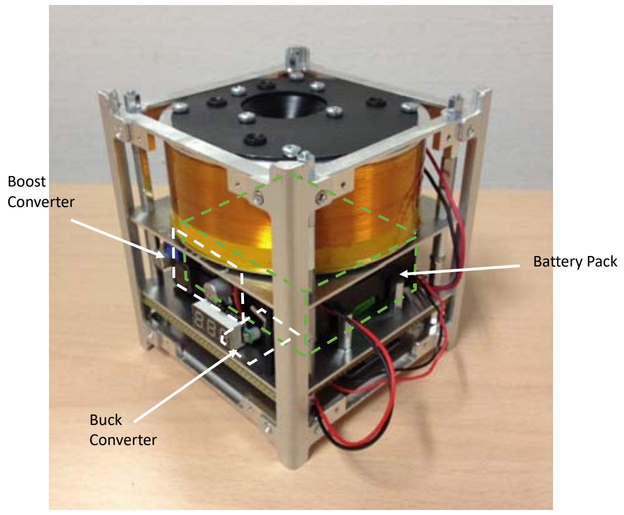


Figure 2.29: FFT Power Subsystem.

Batteries

The battery pack is the same used for the CUBE system, it is composed of 10 Duracell NiMH DX150H and provides power to all the electronic components inside the FFT. As it can be seen in Figure 2.29, it has been placed at the centre of the structure and a small volt-meter has been connected at its ends in order to keep the battery pack's voltage under control by the experimenters and to prevent over-discharge during the tests.

Power converters

A buck converted has been used to step down the voltage of the battery pack and power the IR LEDs, the Arduino board and all the components connected to it. Thanks to its small dimensions it has been placed between the Arduino board and the custom shield, allowing to save space for the other electronics.

Finally, as already mentioned, in order to provide to the electromagnetic coil the current of 1.2A, due to the coil internal specifications, a boost converter has been used to increase the voltage of the battery pack up to 15.5V.
As it can be seen in Figure 2.29, it has been placed on the battery layer, due to the possible heat produced by its components which could be dangerous for the rest of the electronics.
The boost converted has been used to only power the coil, since no other components needed the high voltage provided.

2.2.5 FFT: Software

Even if the Free Floating Target is not actively controlled but act as a passive target, the development of the software running on the microcontroller board has been critical to optimize its functioning and obtain a reliable system. The software of the FFT is very simple if compared to the CUBE, mostly because it does not need to elaborate any data in real time except for the safety and check values read from the temperature and proximity sensors.
A diagram block of the overall software structure is shown in Figure 2.30.

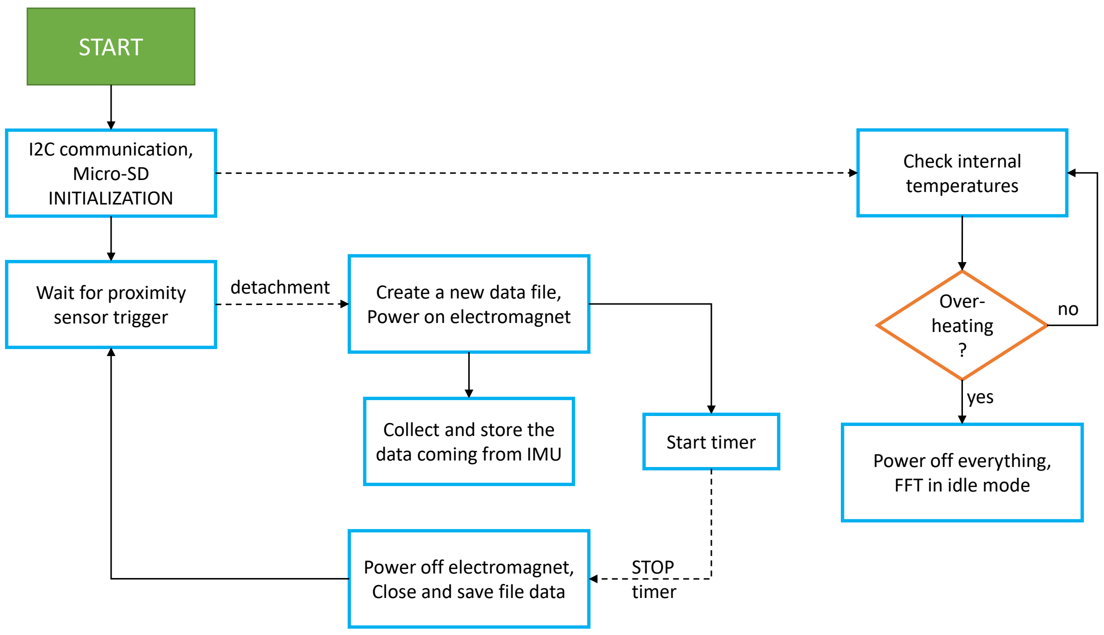

START
I2C communication, Micro-SD INITIALIZATION
Check internal temperatures
Wait for proximity sensor trigger
detachment
Create a new data file, Power on electromagnet
Over-heating ?
no
yes
Collect and store the data coming from IMU
Start timer
Power off everything, FFT in idle mode
Power off electromagnet, Close and save file data
STOP timer

Figure 2.30: FFT software schematic.

The Software on the FFT is constantly running in loop, starting when the system is switched on and placed on the Hold & Launch subsystem.
At the start up, the software initialize the I2C communication with the IMU board and the micro-SD card, used to store the collected data. Furthermore, it checks the temperature of the internal components and if it does not detect any unexpected over-heating, starts the loop control.
In the loop, the software is constantly reading the proximity sensor in order to detect

the CubeSat detachment from the Hold and Launch subsystem.
When the FFT is released, the proximity sensor instantly changes its output and the software in the microcontroller board detects it. Once the detachment has been detected, the FFT starts to float and the Arduino instantly power on the electromagnet through its specifically designed driver circuit, and at the same time, it creates a new file for the incoming data of the inertial navigation. While the electromagnetic coil is switched on, Arduino constantly collects the data coming from the IMU board and stores it in the newly created file, inside a micro-SD card.
In order to reduce the power consumption, at the detection of the FFT detachment a watchdog timer is started, which, after a fixed period of time, switch off the electromagnet and close the file containing the collected data, putting then the Arduino and the FFT in an idle phase, waiting for the next parabola.
Instead of power on the electromagnetic coil for the whole duration of the microgravity phases, in fact, keeping in mind the nominal values of the initial relative velocity imposed in simulation between the two CubeSats of $\approx 4\,cm/s$ and the maximum initial distance between them of $\approx 20\,cm$, we imposed as total time of power on of the coil, 7 seconds, considering ≈ 2 seconds of safety margin.
Thanks to this precaution, a single battery pack has been enough to power the FFT for more than 30 parabolas, allowing us to not change the battery pack during the flight.
Furthermore, as in the CUBE's software, the internal temperature of the FFT components is constantly kept under control, in order to avoid unexpected over-heatings. If the software detects an over heating on any of the components, it immediately switch off the electromagnet, terminates any other action and puts the FFT in idle mode until it is switched off.

2.3 CHAMBER

In order to perform all the experiments on the FFT and the CUBE system in a safe way and provide starting launch conditions, also a containing chamber need to be designed and built.
Indeed, the CHAMBER is a safe environment for the CUBE and the Free Floating Target to float, on the one hand it avoid the risk of hurting other people or damaging other experiments and the support electronics during the flights; at the other it supports an external camera vision system to monitor the experiment.

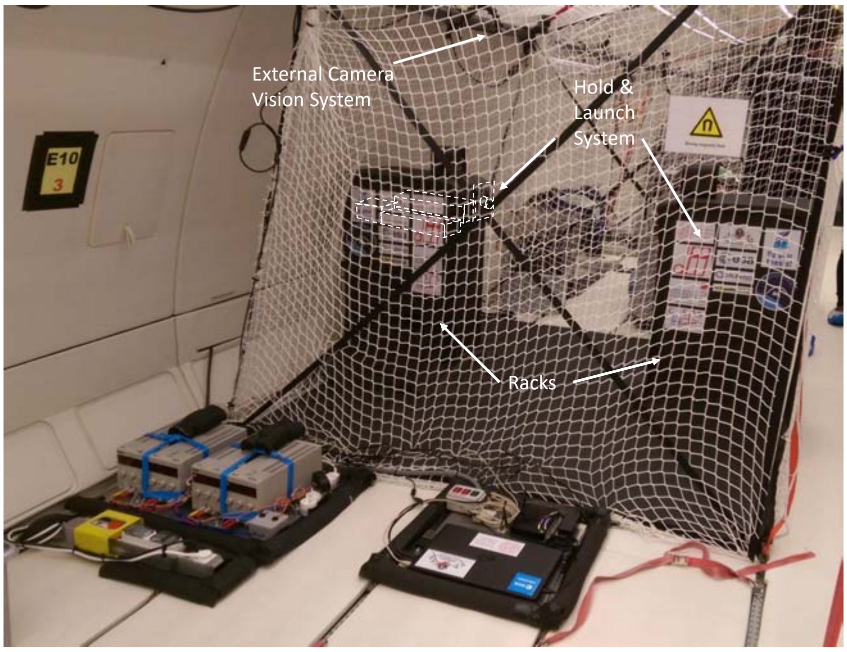


Figure 2.31: CHAMBER photo inside the aircraft. At the bottom can be also noticed the Support Electronics.

As it can be seen in Figure 2.31, a net has been used to limit the motion of both the CubeSats, where the total volume available for the free floating area was of $\approx 8\,\mathrm{m}^3$, i.e. approximately a cube of $2 \times 2 \times 2\ \mathrm{m}^3$.
The CHAMBER is mainly composed by three subsystems:

- the structures, namely the racks;
- the Hold & Launch subsystems;
- the external vision subsystem.

2.3.1 CHAMBER: Structures

The structures are used to keep in place the Hold & Launch subsystem and consists of two completely identical racks, facing each other.

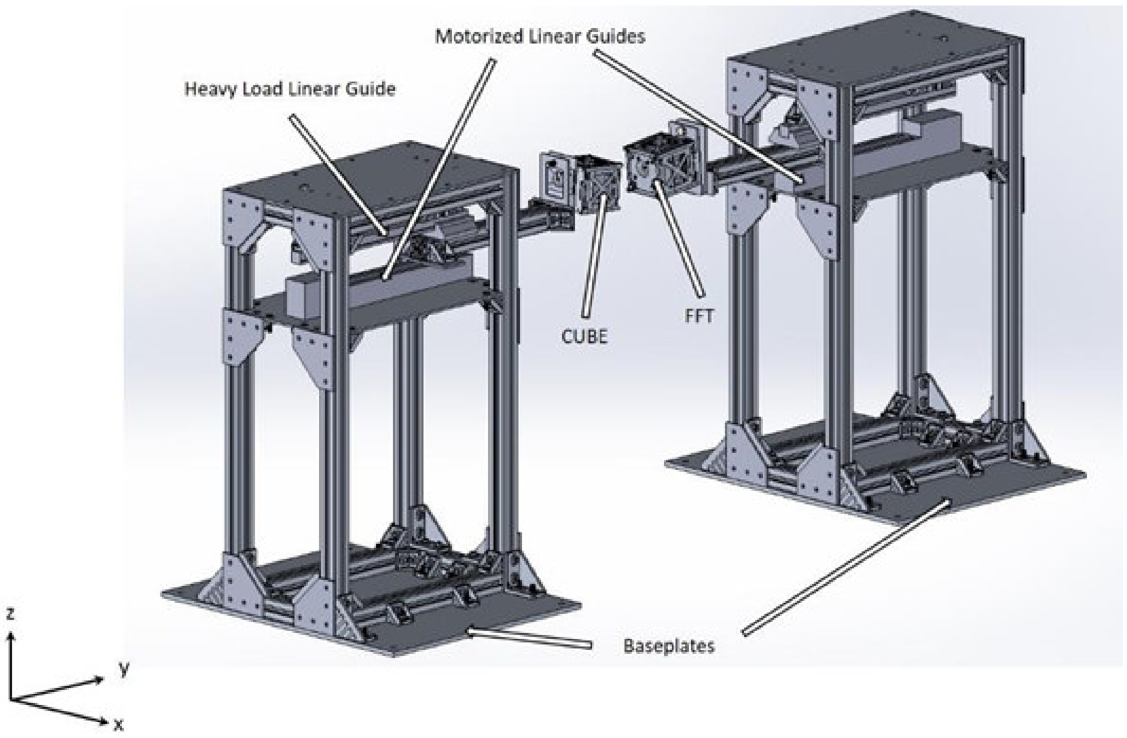


Figure 2.32: CHAMBER racks, including the Hold & Launch subsystem.

As shown in Figure 2.32, the two systems consists of a structure built with Bosch profiles and brackets to connect them together. Moreover, in order to secure the structure for safety reasons, gussets has been added at each external corner.
The systems are then fixed on two identical rectangular baseplates made of aluminium, which are used to bolt the structures to the aircraft rails.
The distance between the baseplates has been chosen such that at the maximum extension, two linear guides bring the distance between the two CubeSats from $\approx 15\,cm$ to $\approx 20\,cm$, values selected through simulations as minimum and maximum allowable distances to have a correct functioning of the experiment, taking into account the time scale of the system and the g-jittering random disturbances experienced during the microgravity phases.
Furthermore, an intermediary shelf is included into the structures, which fix the Hold & Launch subsystems aligned and facing each other.

2.3.2 CHAMBER: Hold & Launch Subsystems

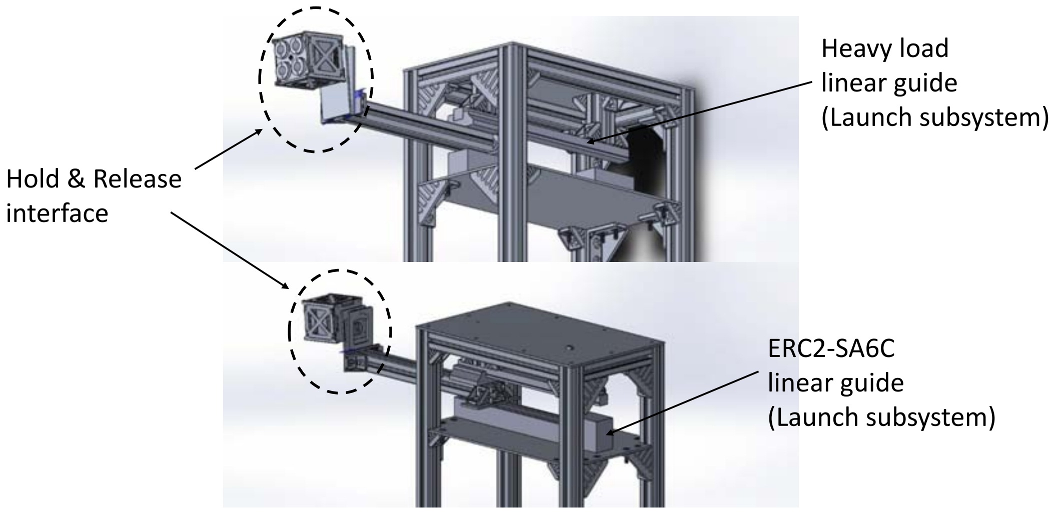


Figure 2.33: Closeup of the Hold & Launch subsystem.

The Hold & Launch subsystems are responsible for the holding of the two CubeSats during the normal and hyper gravity phases and for the launch and release during the low gravity phases, providing the CubeSats a proper initial velocity.
It consists of two parts: the Launch system and the Hold and Release interface.

Launch System

The Launch system is used to impose an initial linear velocity to the CubeSats and is based on a motorized linear guide.
After a preliminary selection, a trade off between performances, ease of use and reliability lead to the choice of the ERC2-SA6C linear guide produced by RoboCylinder as the motorized linear guide of the Launch system.

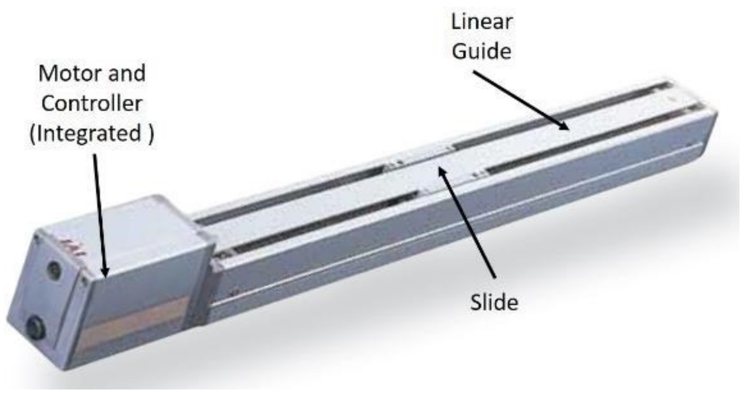


Figure 2.34: ERC2-SA6C Slider Type Electric Actuator with Built-in Controller.

As shown in Figure 2.32, the two motorized linear guides are bolted on the intermediary shelf and positioned at one extremity of the rack.
They are programmed through the dedicated programming software and controlled by the Support Electronics system through an Arduino Mega, which control the movements of the slide of each linear guide independently.
On top of the ERC2-SA6C, a Bosch profile has been fixed to the slide of each linear guide with the function to move the CUBE and the FFT outside the rack and to put them closer before the release during the low gravity phases.
At the free extremity of the profile, the Hold & Release interface has been mounted through two brackets, as shown in Figure 2.33.
Furthermore, in order to lower the moment and the load acting on the ERC2-SA6C, as highlighted in Figure 2.33, a heavy load linear guide, non motorized, has been fixed under the top plate of the racks. It is composed of a rail parallel to the stroke of the motorized linear guide and a slide, which sustain the load of the Cubesat and the Hold & Release interface.
Thanks to this particular system, it has been possible to easily change the release position, the initial linear velocity and to adapt the flight time of the two CubeSats in the pauses between each parabola during the flights, permitting the experimenters to test different initial conditions and to adjust the parameters of the experiment with respect to the different parabolas and microgravity phases.

Hold & Release interface

The Hold & Release interface is instead used to keep the CUBE and the FFT in position, providing them with different orientations.

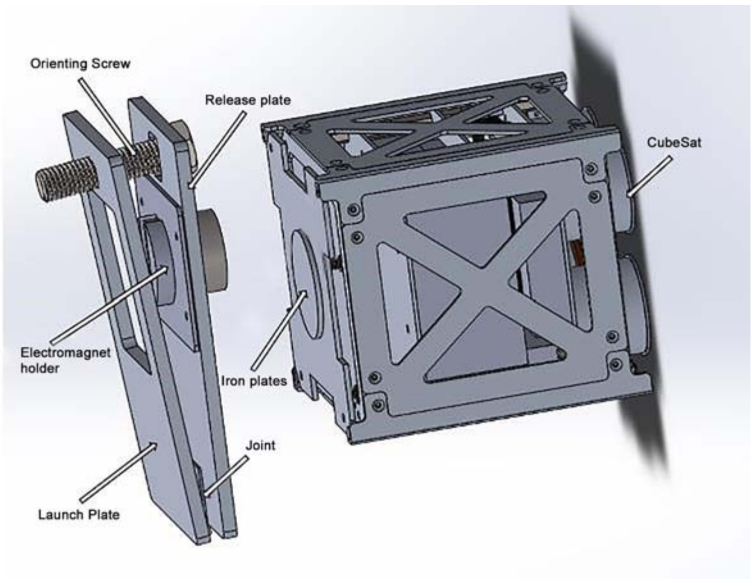


Figure 2.35: Hold & Release interface.

It consists of two aluminium plates: the launch plate, mounted on the Bosch profile of the Launch system and the Release plate, connected together with a joint and an orienting screw, as shown in Figure 2.35.
Moreover, a small holder electromagnet has been placed in the Release plate and controlled by the Support Electronic system, synchronized with the motorized linear guide in order to release the nano-satellites at the right moment during the micro gravity phases.
As already mentioned, an interface mounted at the bottom of both the CUBE and the FFT, matches precisely the holding electromagnet and the release plate, keeping in place the CubeSats.
Finally, the orienting screw allows to regulate the initial CubeSats orientation simply by screwing or unscrewing it, operation that has been performed during the pauses among the parabolas.
Since the Hold & Release interface holds the CubeSats during the hyper gravity phases, static load tests has been necessary to verify that the system was able to hold the loads in these conditions. As a result, the Hold & Release interface has been able to bear a maximum load of more than $8kg$, which is more than twice the weight of

both the CubeSats in the hyper-gravity phase, considering the actual weight of the CUBE ($\approx 1.3kg$) and of the FFT ($\approx 1.4kg$).

2.3.3 CHAMBER: Vision Subsystem

In order to obtain further information about the entire system dynamics, a stereo camera has been placed inside the CHAMBER. It has been used to acquire images of the CUBE and the Free Floating Target during the floating phases, which will be exploited in post-processing to retrieve the actual relative pose of the two spacecrafts. In order to help the recognition of the actual pose of the CubeSats, the external surface of both the CUBE and the FFT have been covered with chessboards (Figure 2.37).
Thanks to the use of chessboards with known geometry, a monocular system would have been sufficient to estimate the pose of the two floating CubeSats, however, using more cameras boost the performances of the algorithms thanks to the redundancy of the measurements. Moreover, it provides the capability to recover information also about the depth by means of triangulation.
During the design phase of the vision subsystem, several important aspects have been taken into account, such as the synchronization of the cameras, the possible distortion of the images due to lens problems or rolling shutter cameras and the field of view of the entire camera system, which needs to always see both the CubeSats.
The selected camera system is represented by the stereo camera DUO MC, shown in Figure 2.36, which is a very compact system (with dimensions of just $57 \times 30 \times 15mm$ and a weight of $\approx 0.015kg$), equipped with two cameras which feature global shutter and whose optical sensor are hardware synchronized.

Figure 2.36: Stereo camera DUO MC, used as external reference camera to acquire images of the CUBE and the Free Floating Target during the free floating phase.

Furthermore, the stereo camera features a wide field of view of 170 degrees.
For this reason, it has been possible to position the camera above the structures inside the CHAMBER, so as to always see the two nano-satellites without any obstacle.
Finally, the camera has a dedicated software which allows to easily set the camera

settings and record images.
In Figure 2.37 it is reported a frame of one of the recorded parabolas.

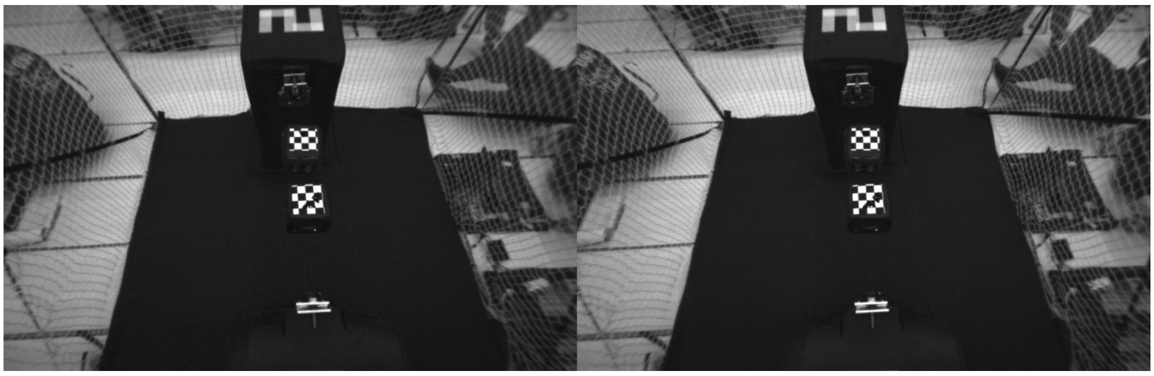


Figure 2.37: Frame of a parabola taken during the 2nd flight with the DUO MC.

2.4 Support Electronics

The Support Electronics system is of fundamental importance for the experiment. It is responsible for the correct functioning of the Hold and Launch subsystem, for the wireless start of the software on the CUBE and for starting the recording of images from the external vision system.

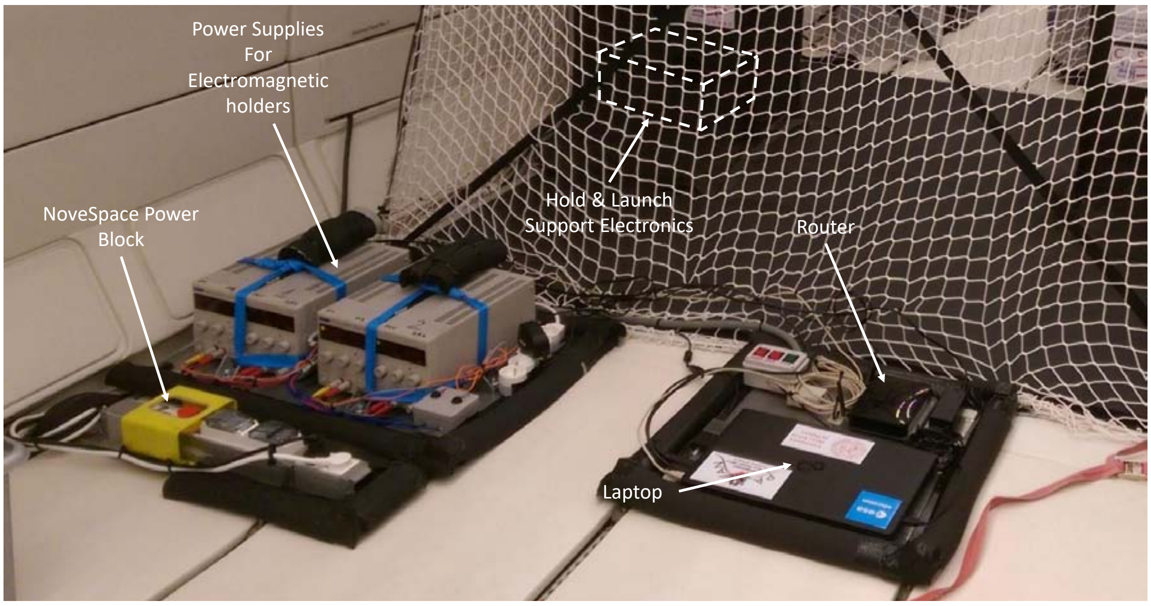

Power Supplies For Electromagnetic holders
NoveSpace Power Block
Hold & Launch Support Electronics
Router
Laptop

Figure 2.38: Support Electronics components inside the aircraft.

It consists of all the components that support other subsystems to work properly, in particular, as highlighted in Figure 2.38, it is composed by:

- the NoveSpace power block;
- the Power Supplies for the Hold & Launch System;
- the Hold & Launch support electronics;
- the Router;
- the Laptop with the dedicated software developed.

NoveSpace power block

The NoveSpace power block is an electrical distribution block provided by NoveSpace for each experiment. It provides power to all the experiment, except for the two floating CubeSats.
For safety reasons, it is equipped with an emergency button and two fuses, one for the phase and one for the neutral (bipolar phase / neutral protection).

Power Supplies

The Power Supplies are used to switch on and off the electromagnets which hold and release the CUBE and the FFT during the flights.
In order to avoid residual magnetization between the CubeSats and the electromagnetic holders, a current control inside the power supplies forces the electromagnets to switch off instantly and reduces the residual magnetization between the systems. However, even if the problem of the residual magnetization has been a very interesting problem to solve, it is not subject of this thesis and it has been reported here for completeness.
The outputs of the power supplies have been controlled by the control circuit inside the Hold & Launch support electronics, synchronizing them with the two motorized linear guides.

Hold & Launch support electronics

The Hold & Launch support electronics is used to control both the motorized linear guides and the electromagnets. It is mainly composed of a Printed Circuit Board specifically designed to work with the Hold and Launch subsystem.
It is based on an Arduino Mega, which thanks to its numerous outputs can drive the linear guides and the electromagnets independently.
The schematic of the PCB is reported in Figure 2.39.

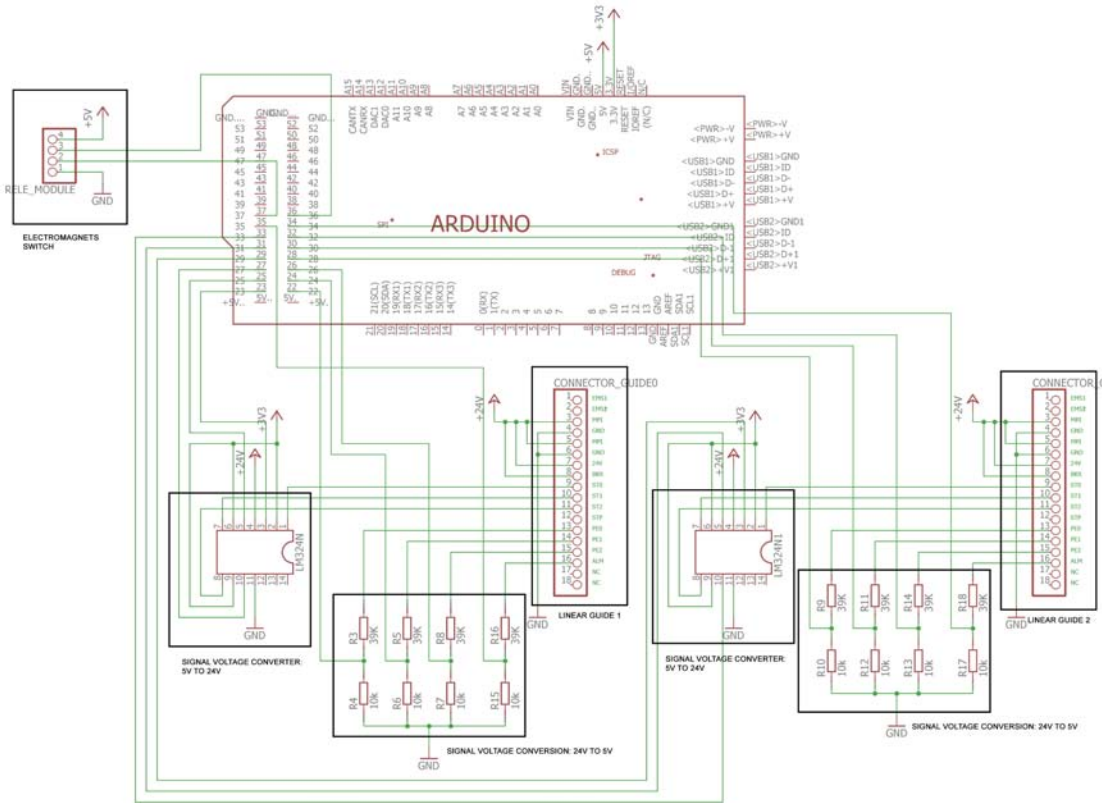


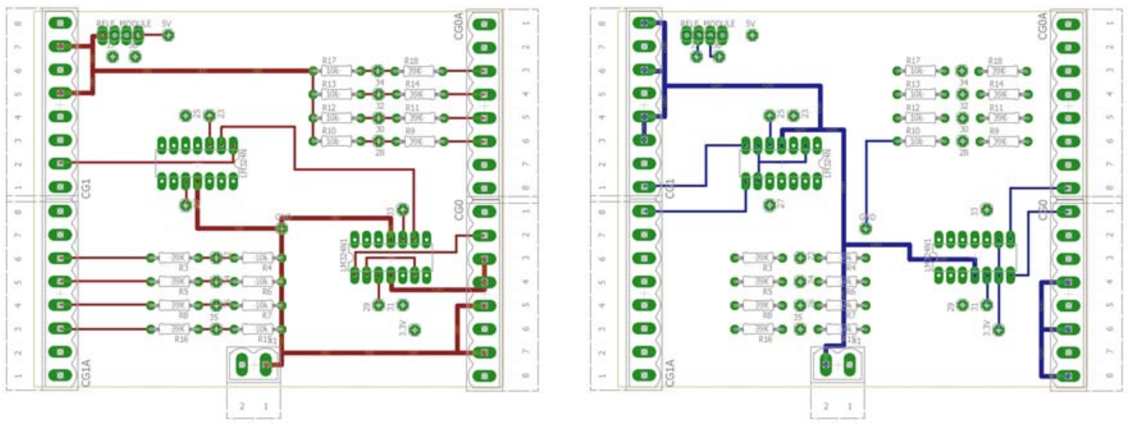
Figure 2.39: Schematic of the Printed Circuit board.

As it can be noticed from the schematic, the designed circuit features two different signal conversion blocks, because of the different voltage levels of Arduino Mega (5V) and the motorized linear guides (24V). The conversion block that steps up the voltage of the Arduino Mega's outputs is based on a LM324N integrated circuit, which consists of four independent, high-gain operational amplifiers, designed to operate over a wide range of voltages.
For each LM324N used on the board only 3 operational amplifiers have been used as comparators, which convert the signals used to control the position of the slide of each motorized linear guide.
On the other hand, the conversion block that drops down the voltage of the linear guides's signals to the 5V signals for Arduino, is based on a simple voltage divider, which, accordingly to the theory and considering for example the resistors R_3 and R_4 in Figure 2.39 provides as output the voltage:

$$V_{out} = V_{in}\frac{R_4}{R_3 + R_4} = 24 \cdot \frac{10k}{39k + 10k} \simeq 4.9\text{V} \tag{2.30}$$

Furthermore, a relay board controlled by Arduino Mega has been used to switch

on and off the electromagnets, supplying them the power from the power supplies previously described.
Finally, for completeness, the designed PCB board is reported in Figure 2.40.



(a) Top layer. (b) Bottom layer.

Figure 2.40: PCB board used to control the motorized linear guides and the electromagnets of the Hold & Launch subsystem.

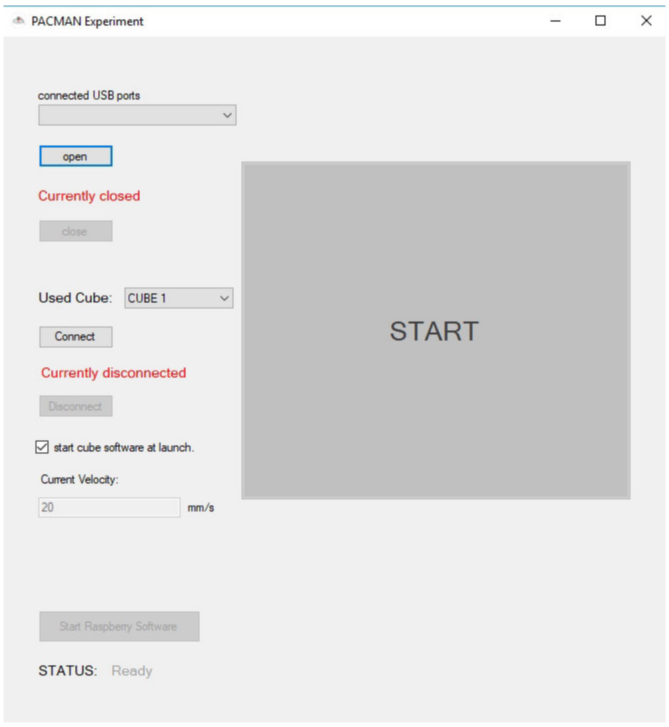
As it can be noticed from the above Figure, the PCB board has not been designed as a custom Arduino Mega shield, but the Arduino board and the PCB are connected by means of wires.

Router

A router has been used to create a Local Area Network (LAN) in order to communicate to the CUBE's controller boards. The router is the NETGEAR DGN2200v3. Thanks to the LAN, it has been possible to send the starting SSH command from the Laptop to the Raspberry board inside the CUBE.
Moreover, since the microcontroller boards (Arduino Uno) used for both the CUBE and the FFT have an integrated Wifi Module, they have been programmed and debugged through the network without the need of disassembly the CubeSat to extract the microcontroller boards.

Laptop and Software

The Laptop is responsible for several tasks, such as programming the motorized linear guides, control the Hold and Launch system, acquire images from the CHAMBER Vision Subsystem and start the software controller of the CUBE during the low gravity phases.
In order to accomplish all the required tasks during the flight, a dedicated software with a Graphic User Interface (GUI) has been developed.
The software has been developed in C# language and it is responsible for the movements of the linear guides, as well as for the start of the CUBE's software.
In Figure 2.41 it is reported a screenshot of the developed GUI used during the flights.



Currently closed

Used Cube: CUBE 1

Currently disconnected

☑ start cube software at launch.

Current Velocity: 20 mm/s

STATUS: Ready

Figure 2.41: Screenshot of the Laptop software developed for the experiment.

From the GUI, through the "open" button, the experimenters were able to establish a serial communication with the board, while through the "Connect" button, the software established a Secure SHell connection with the Raspberry board inside the CUBE system.
Once the connections have been established, through the "Current Velocity" Textbox the experimenters could set the selected initial velocity imposed by the Hold and Launch system to both the CUBE and the Free Floating Target.

When everything is connected and properly initialized, the “START” button enables and when pressed, the software send a start byte to the Arduino Mega board through serial communication, which recognize the byte and start sending a sequence of signals to properly move the motorized linear guides. Furthermore, when the button is pressed, as already mentioned, the software also send an SSH command to the Raspberry Pi board of the CUBE, which then starts the autonomous controller software.

Finally, for completeness, in Figure 2.42 it is reported a block diagram of the whole Support Electronics, which shows all the connections between the components of the system.

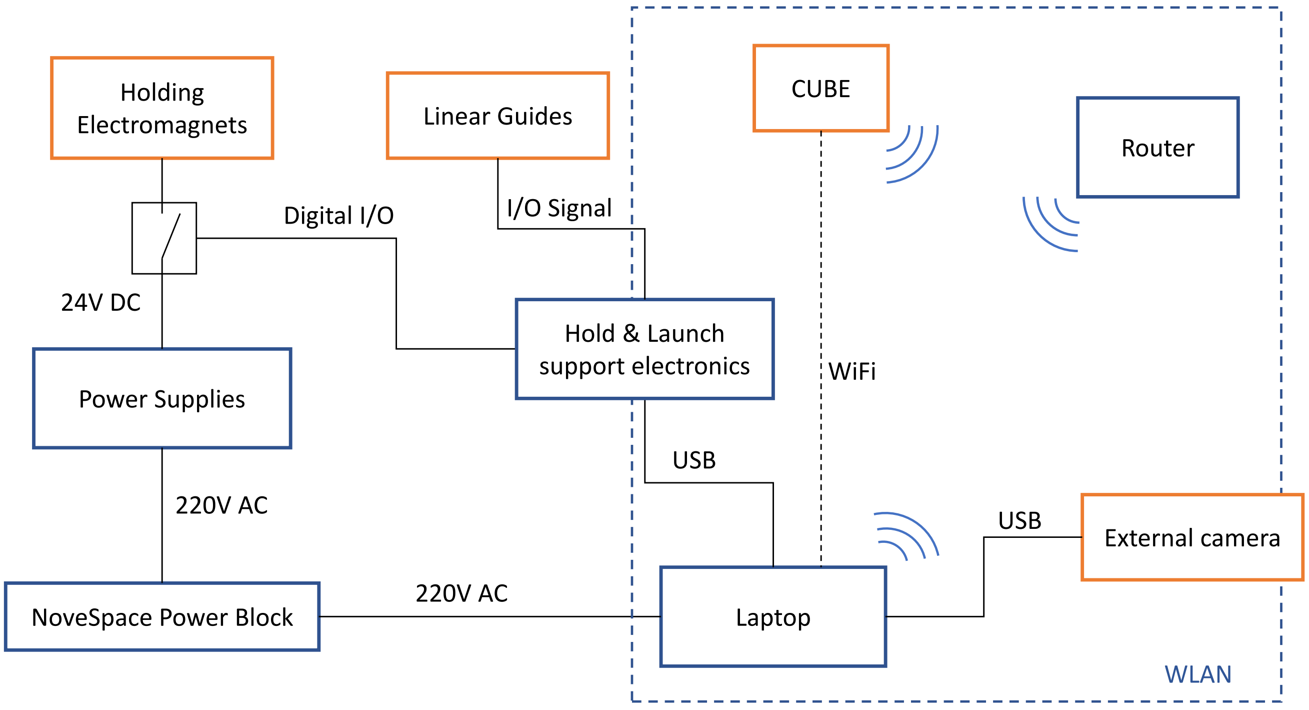


Figure 2.42: Block diagram of the Support Electronics. The blue boxes represents the components of the Support Electronics, while the orange boxes all the other components that interact with them.

3 Dynamical Simulations

3.1 Model

The whole PACMAN experiment is based on the results obtained from a dynamical model implemented in Matlab-Simulink.
The model is an accurate simulator which is almost entirely parametrized, so that there are no constants in the simulations and all the parameters can be changed to test every required condition.
In the dynamical simulator, for simplicity, both the CUBE and the FFT have been modelled as two masses of $1.5kg$, with inertia matrices extracted from the CADs.
The simulator uses as inputs several parameters, such as the imposed initial velocity and orientation of both the nano-satellites, their relative position, their mass characterization and the electrical and geometrical characterization of the coils of both the CUBE and the Free Floating Target. Thanks to these, it can provide as output the forces and torques applied to the spacecrafts, the mutual magnetic interactions between them and their attitude, position and velocities during the simulation, so as to determine the estimated system dynamics.
The camera vision system of the CUBE and its relative pose estimation algorithm has not been modelled because of the increase of the complexity that the system could have added; instead, in order to feed the control block, the actual relative orientations extracted from the model and corrupted with white noise (to simulate a more realistic scenario) have been used.
A simplified block diagram of the simulator is reported in Figure 3.1, where the following features have been modelled:

- the CUBE, in terms of its geometry, mass and inertia;
- the Free Floating Target, in terms of its geometry, mass and inertia;
- the magnetic interactions between the free floating objects due to the produced electromagnetic fields.
- the initial conditions of both the CUBE and the FFT in terms of relative distance, launch velocity and initial angular velocities;

– the dynamics of the system also considering the disturbances on-board the aircraft, extracted from the acceleration profiles provided by NoveSpace;
– the controller algorithm.

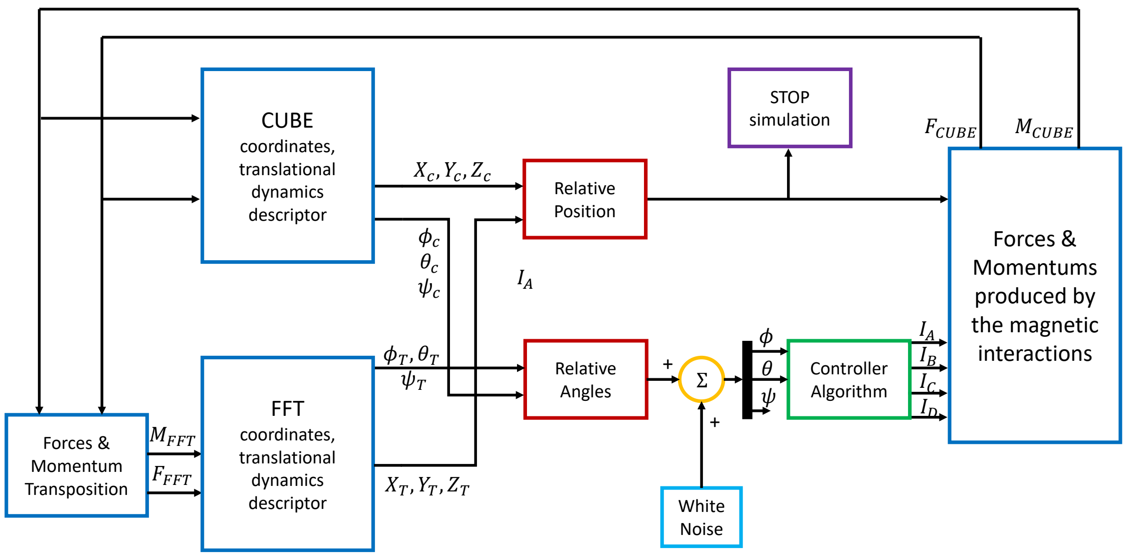


Figure 3.1: Simplified block diagram of PACMAN dynamic model.

As it can be notice from the block diagram, the instantaneous values of the orientation angles given as input to the controller block are corrupted with a random noise. We found that the implemented control algorithm was very robust to white noise, meaning that no significant error was induced by the noise in simulation and the obtained dynamical behaviour of the spacecrafts was approximately the same with or without noise disturbances.
The controller algorithm implemented in simulation was composed of two PID algorithms, both including an anti wind-up scheme to allow faster response of the feedback control loop.
Considering as relative orientation angles the Euler angles, oriented as in Section 2.1.5, Figure 2.20, the two PID regulators control a degree of freedom each, namely the Pitch (θ) and the Roll(ϕ) angles, because the Yaw(ψ) angle is not affected by the actuator's control actions and thus it cannot be controlled.
The PID algorithms calculate at each step the instantaneous current value to apply to the coils to generate a magnetic field strong enough to recover the misalignment angles that the controllers receive as an input.
Furthermore, like in the implemented algorithm, the PID regulators set the current

value of the coils in pairs, summing the single control signals acting on each coil. The Simulink scheme of the resulting controller block is reported in Figure 3.2.

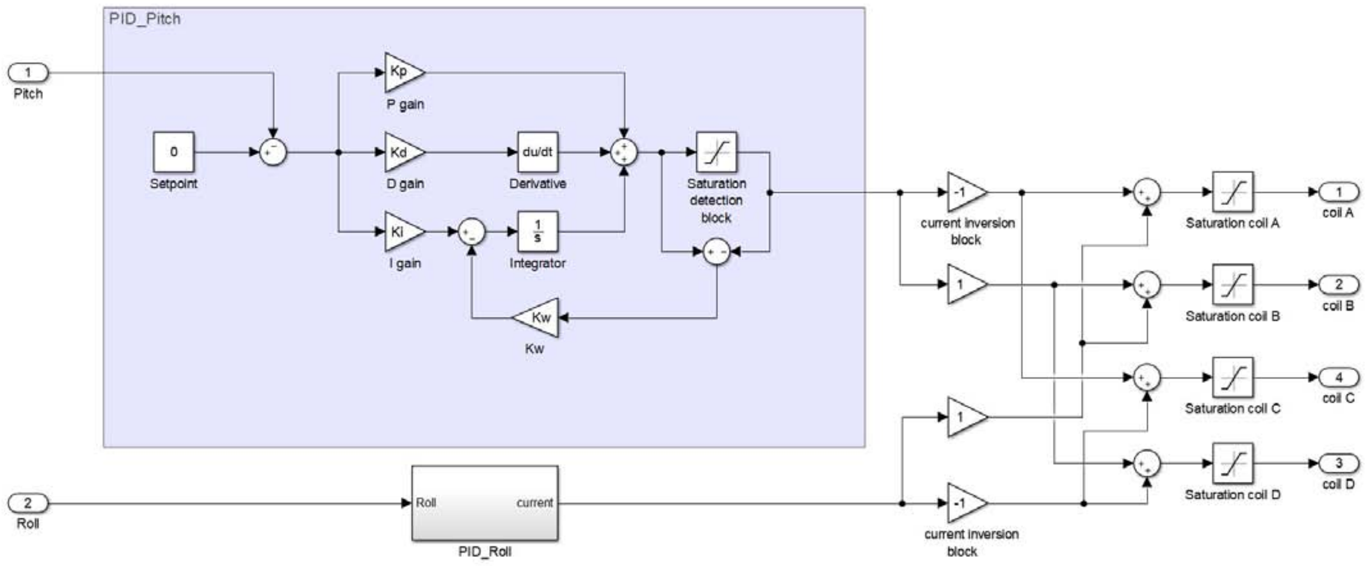


Figure 3.2: Simulink model of the controller algorithm.

3.2 Simulation Results

According to the procedure described in Section 2.1.5, several trial-and-error tests have been done in simulation in order to find a set of parameters which could assure good performances in terms of rising time and overshoot.
In particular, for the experiment purposes we searched for a set of parameters which could grant fast time response (low rising times) and almost null overshoot in order to avoid instabilities.
Since the timescale of the experiment is very short (with the nominal launch conditions, namely setting the relative distance to $20\,cm$ and the initial velocity of both the nano-satellites to $\approx 2cm/s$, we have a manoeuvre time of $\approx 5\,s$), if it occurs an overshoot, it will be at the end of the manoeuvre, which is the most critical phase. Indeed, recalling the Equations 2.1 and 2.4 in Section 2.1.2, when the CubeSats are close, the electromagnetic interactions produces bigger forces with respect to when the CubeSats start the manoeuvre; for this reason, an overshoot in the last phase of the manoeuvre could lead to unrecoverable misalignments in the docking phase or even divergences.

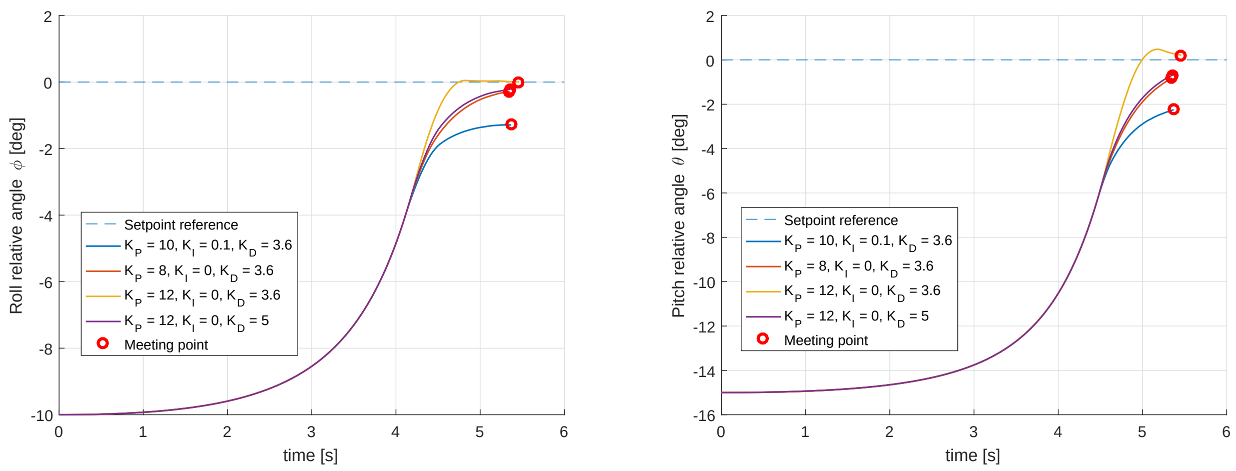
After several tests, we found as good sets of PID parameters the following:

$$\begin{array}{cccc} K_P = 10 & K_I = 0.1 & K_D = 3.6 & K_W = 0.05 \\ K_P = 8 & K_I = 0 & K_D = 3.6 & K_W = 0 \\ K_P = 12 & K_I = 0 & K_D = 3.6 & K_W = 0 \\ K_P = 12 & K_I = 0 & K_D = 5 & K_W = 0 \end{array}$$

where K_W is the gain of the anti wind-up signal applied to the integrative term of the PID controllers, as shown in Figure 3.2. Furthermore, since the model represents all the coils perfectly equal and the CubeSats perfectly aligned, we assume that there are no differences between the controlled degrees of freedom, so that the two PID regulators are provided with the same set of parameters.
The results of the simulations have been reported in Figure 3.3, where the model has been tested with the following initial conditions for all the tests:

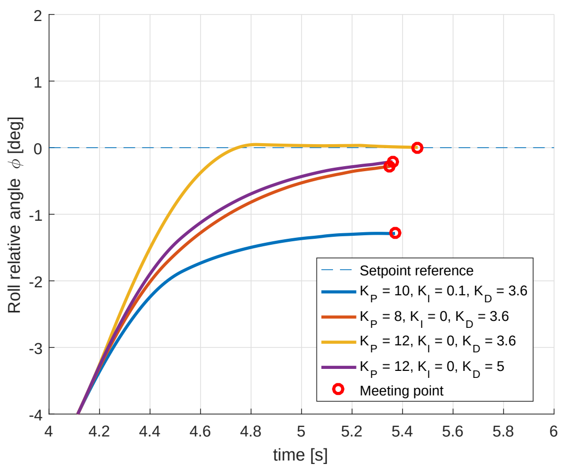
- Initial CUBE Roll relative angle w.r.t. the FFT: $-10\,deg$;
- Initial CUBE Pitch relative angle w.r.t. the FFT: $-15\,deg$;
- Initial CUBE angular velocity in its x axis: $0\,deg/s$;
- Initial CUBE angular velocity in its y axis: $0\,deg/s$.

In Figure 3.4 it is reported a closeup of the last phase of the manoeuvres.

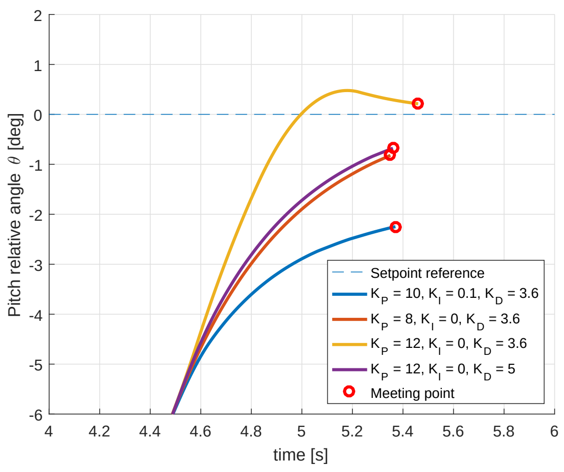


(a) Roll (b) Pitch

Figure 3.3: Orientation angles recovery with different sets of PID parameters. The simulations stop when the two CubeSats meet.



(a) Closeup of docking phase: Roll



(b) Closeup of docking phase: Pitch

Figure 3.4: Closeup of the docking manoeuvres reported in Figure 3.3.

From Figure 3.3 it can be noticed the performance of the designed controller. In almost all the tests, the controller has been able to recover the initial misalignment on both the controlled axes, with a final misalignment smaller than 1 degree, which is an acceptable result.
In particular, it can be seen how the integrative term slow down the system; in fact, the worst performances for both the controlled axes have been obtained with the set that includes an integrative action, reason for which the final controller has been implemented as a PD regulator, which is faster and resulted perfect for our system.
Besides, the third set ($K_P = 12$, $K_I = 0$, $K_D = 3.6$) performed the best performances for the Roll angle, whose initial misalignment was of -10 degrees, while in the Pitch angle, whose initial misalignment was of -15 degrees, it led to an overshoot. For this reason it is possible to state that this set of parameters characterize a controller which performs well for small angle of misalignment, but it is not well suited for large misalignment angles.

Finally, from the Figures 3.4a and 3.4b it is clear that the second and fourth sets of parameters led to two controllers that show almost the same performances for both the Pitch and the Roll angles. For this reason, they have been selected as the most reliable set of parameters and the second set has been chosen as first set of parameters to be tested in the CUBE during the first flight. Even if the fourth set was a little faster than the second one, it has been discarded because of the higher derivative gain, which could lead to instabilities in different conditions, in particular in presence of measurement noise.

An other interesting point of view of the dynamical behaviour, is that of the angular velocity induced by the electromagnetic interactions. It is important to notice here that the designed controller does not control directly the angular velocities of the spacecraft, but its relative orientation with respect to the FFT.
Since the system composed by the Hold & Launch interface and the CubeSats is not frictionless in the real environment, in the detachment and retraction phases of the Launch system, some angular velocities can be induced to the CubeSats. For this reason it has been important to test if the designed controller was able to recover an initial angular velocity imposed by it.
In order to test this, the controller has been tested in simulation with the following initial conditions and parameters:

– Initial CUBE Roll relative angle w.r.t. the FFT: $0\,deg$;
– Initial CUBE Pitch relative angle w.r.t. the FFT: $0\,deg$;
– Initial CUBE angular velocity in its x axis: $0\,deg/s$;
– Initial CUBE angular velocity in its y axis: $5\,deg/s$;
– PID parameters: $K_P = 8,\ K_I = 0,\ K_D = 3.6$.

In Figure 3.5 there are reported the obtained results, where only the measures on the interested axis (the y axis) are shown, because the others were identically zero.

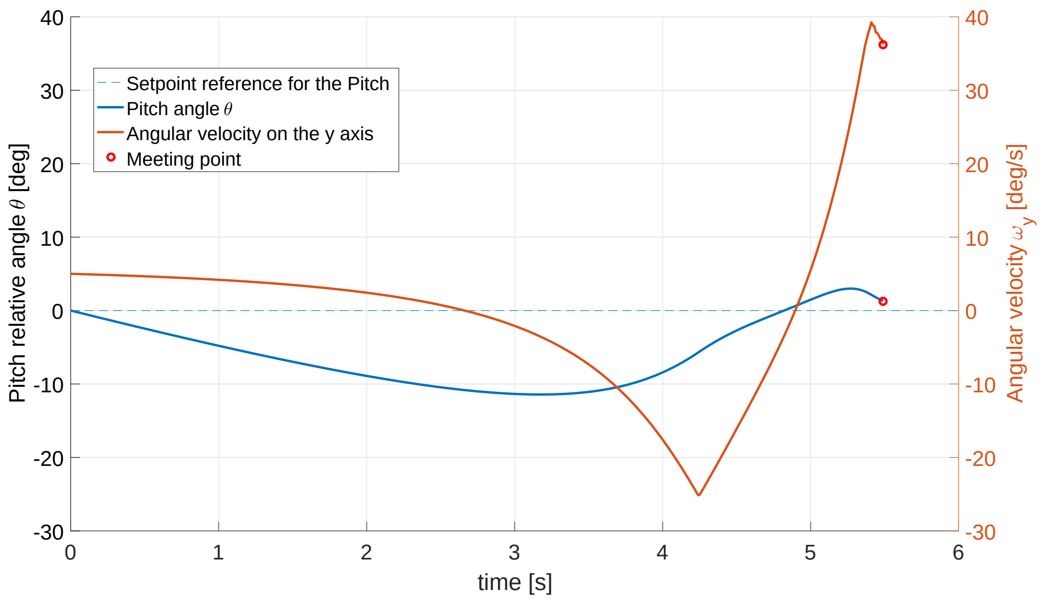


Figure 3.5: Angular velocity ω_y on the y axis and the relative Pitch angle, given the initial conditions reported above. Meeting point is at $t = 5.49$ s

As it can be observed from Figure 3.5, the designed controller is able to recover small initial velocities that can be induced by the Hold and Launch system.
In particular, the final misalignment of the CUBE with respect to the FFT was of about 1.5 degrees, which is a very good result given the imposed initial conditions. A little overshoot of maximum amplitude of ≈ 3 degrees can also be noticed, but it did not impaired the last phase of the docking manoeuvre.
The fast changes of the angular velocity obtained at about 4.3s and 5.4s are due to the change of the direction of the current flowing on the coils, which instantly change the angular velocity behaviour in the simulations.

Finally, it can be worth to notice from the model in Figure 3.2, that the resulting control signals for each coil pass through saturation blocks, which simulate the realistic scenario where the coil current is limited to a maximum value by the driver.
In order to understand how the current limitation influence the dynamical behaviour of the CubeSats, we performed some tests with and without the saturation limitation blocks, keeping unchanged the parameter values of both the PID regulators for the tests $(K_P = 8,\ K_I = 0,\ K_D = 3.6)$.

The tests have been performed with the following initial conditions:

- Initial CUBE Roll relative angle w.r.t. the FFT: $-10\, deg$;
- Initial CUBE Pitch relative angle w.r.t. the FFT: $0\, deg$;
- Initial CUBE angular velocity in its x axis: $0\, deg/s$;
- Initial CUBE angular velocity in its y axis: $0\, deg/s$;

and the obtained results are reported in Figure 3.6.

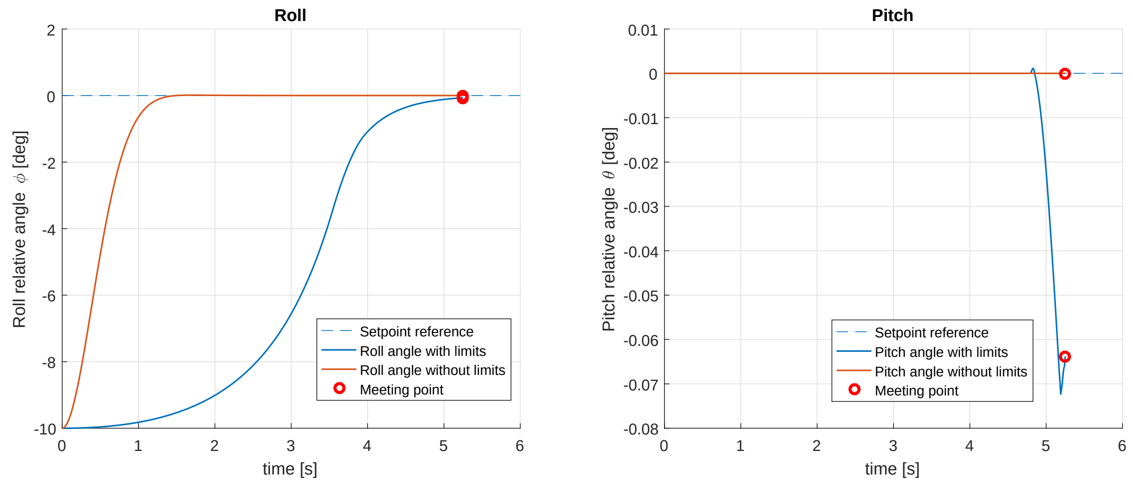
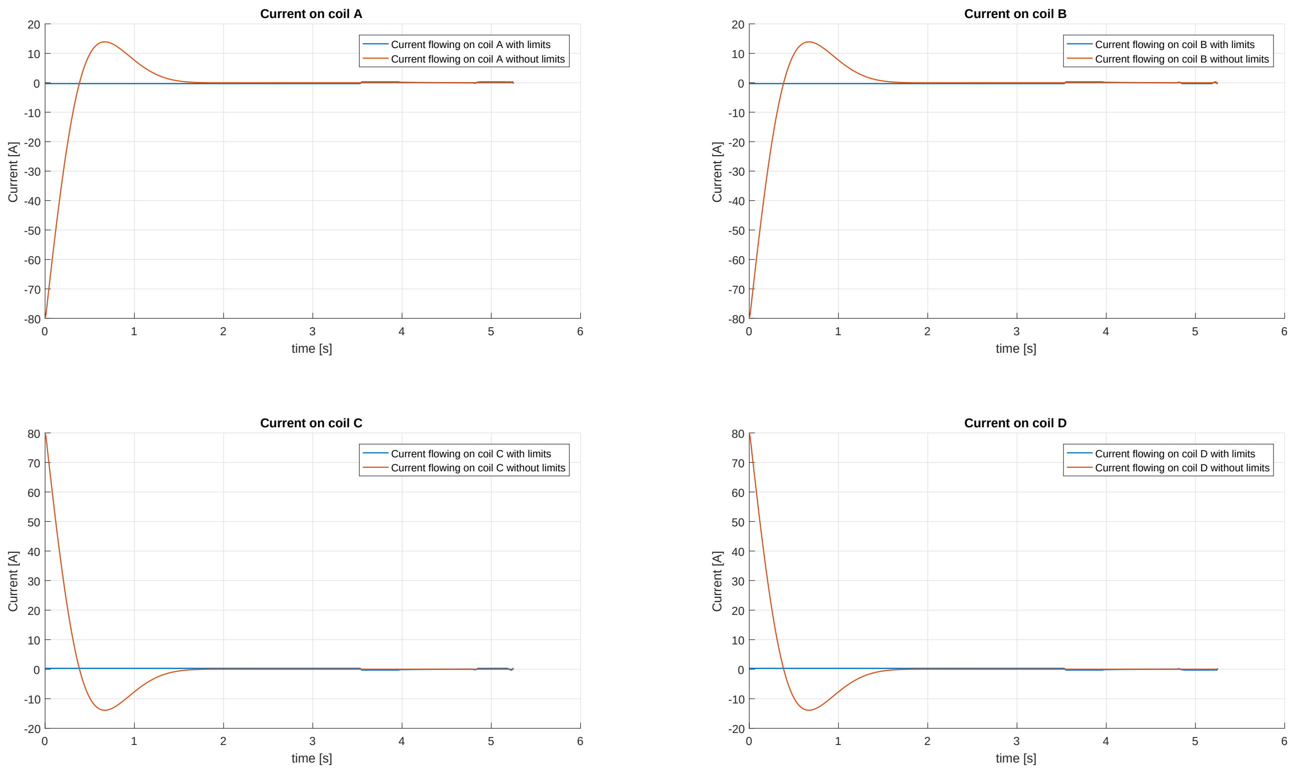
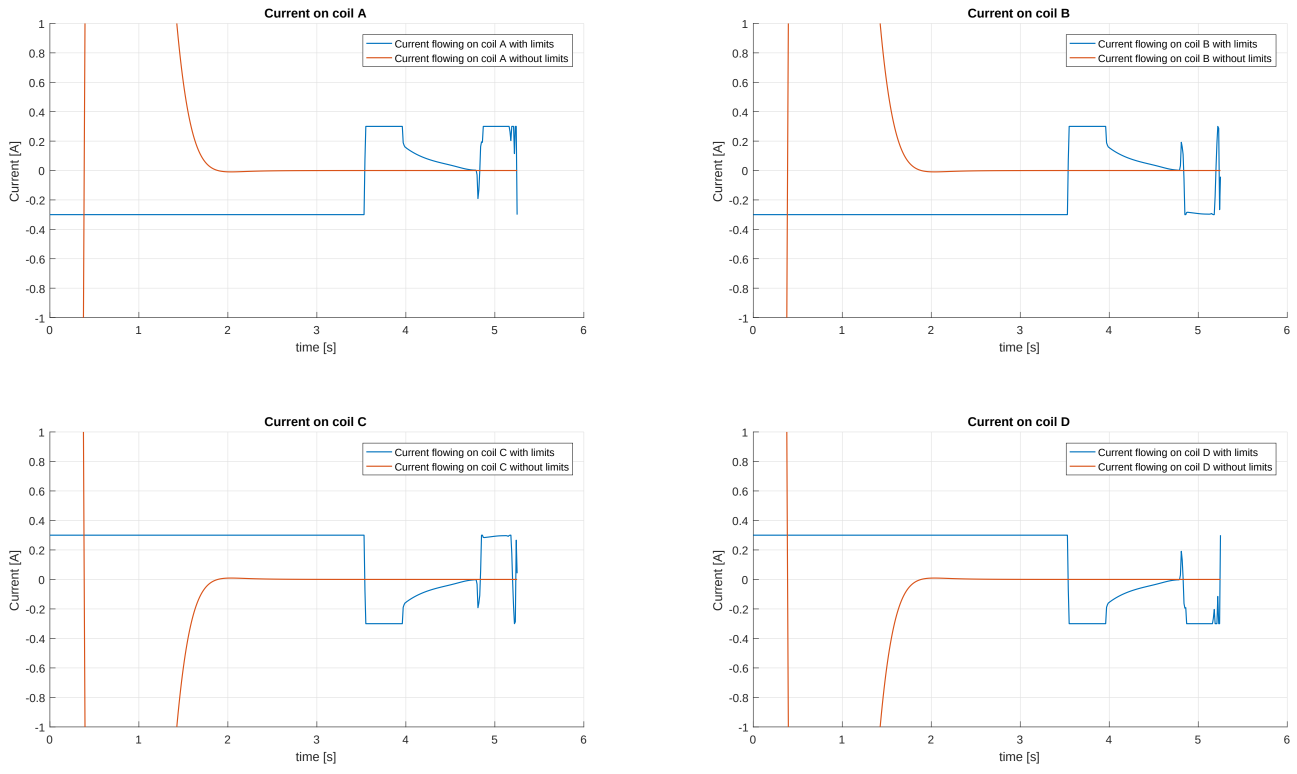


Figure 3.6: Relative orientation angles between the CUBE and the FFT with and without limits on the coil current in simulation.

With reference to Figure 3.6, it is obvious that the control actions ideally implemented without limiting the current flowing on the coils lead to a very fast and precise controller. In fact, in these conditions the controller recovers the misalignment angle in $\approx 1.5\,s$, while the controller using the realistic current limits recover it in $\approx 5\,s$. The Pitch angle instead remains 0 in both the configurations, except for the last half second where the system with limits had a small error ($\approx 0.07°$), which is probably due to uncertainties in the model when the two CubeSats are very close.
However, it is important to stress the fact that the condition of a controller without limits is not realistic or implementable and can be observed only in simulation for theory purposes.
In fact, the current applied to each coil without limitations is of about 80 A (Figure 3.7), which, considering that the coils are made up of $\approx$ 315 windings, provided a magnetomotive force of $\approx$ 25200 At, which is unrealistic for the developed system.
For completeness, in Figure 3.7 are reported the imposed currents flowing in the coils during the simulations with and without the limits given by the saturation of the drivers.



(a) Current flow on each coil with and without limits in simulation.



(b) Closeup of current flow on each coil with and without limits in simulation.

Figure 3.7: Current flow on each coil in simulation.

4 Experiment

4.1 Experiment Procedure

As already said, the experiment's objective is to validate in a relevant environment, namely in micro-gravity, an integrated system for proximity navigation and soft docking, based on magnetic interactions. This has been accomplished by launching two nano-satellites one toward the other: the Free Floating Target, which is a miniature spacecraft that generates a static magnetic field, and the CUBE, which is the autonomous spacecraft that can change its relative attitude with respect to the FFT exploiting the magnetic interactions generated between them.
The two nano-satellites have been launched one toward the other with fixed initial conditions, imposed by the experimenters during the flights.
Since each parabola provides about 20 seconds of microgravity, the experiment timescale had to be fitted in this time constraint.
In order to better understand the experiment dynamics, as an outline, we present here the experiment flowchart, which shows the different events that have taken place during the parabolic flights. In Figure 4.1 it is also presented an approximate time line of the experiment.
For completeness, we also report here the Initialization phase, which take place at the beginning of each flight.

- **Initialization Phase**
  Starting at the beginning of each flight, the overall experiment system needs to be initialized.
  In this phase the Laptop and all the Support Electronics are turned on, as well as both the CubeSats.
  Once that everything is powered, the motorized linear guides are initialized, by means of commanding their slide position to home (position 0) through the proprietary program used also to program them.
  Finally, the PACMAN GUI and the Camera software can be opened and initialized as well, and both the CubeSats can be placed on the Hold and Launch

subsystems.
This phase takes place only once during the flight.

- **Injection Phase**
  In this phase starts the microgravity; after $\approx 3 - 5$ seconds the experimenters start to record the images from the external Camera Vision system and start the experiment through the PACMAN GUI.

- **Launch Phase**
  During this phase, the motorized linear guides start moving, but both the nano-satellites are still held by the Hold & Launch subsystems.
  The SSH command sent by the PACMAN software initialize the CUBE system, which starts recording images from its camera system and initialize the serial communication. Then, it starts collecting and storing data from the navigation sensors, Camera and IMU, until the software detects the Release phase.

- **Release Phase**
  The linear guides stop, the electromagnet holder is turned off and the linear guides are retracted immediately, in order to free space for the movements of the two CubeSats.
  The CubeSats are then released with a fixed initial velocity and relative distance between each other, both established by the experimenters.
  Once released, the proximity sensors on both the CUBE and the FFT triggers and sends a signal to start the experiment. Then, the FFT turns on the electromagnetic target generating a static magnetic field, and the CUBE starts the control actions.

- **Free-Floating Phase**
  In this phase the two nano-satellites are free floating, approaching each other and controlling their relative attitude.

- **Soft-Docking Phase**
  The two nano-satellites meet and dock. The control actions are then terminated and the static electromagnetic field generated by the FFT is turned off.

- **Hyper-gravity Phase - end of the experiment**
  The two nano-satellites, which were floating, drop to the floor and the external Camera Vision system stops recording images.
  At the end of this phase the CubeSats are then re-positioned in the Hold & Launch subsystems, ready for the next parabola.

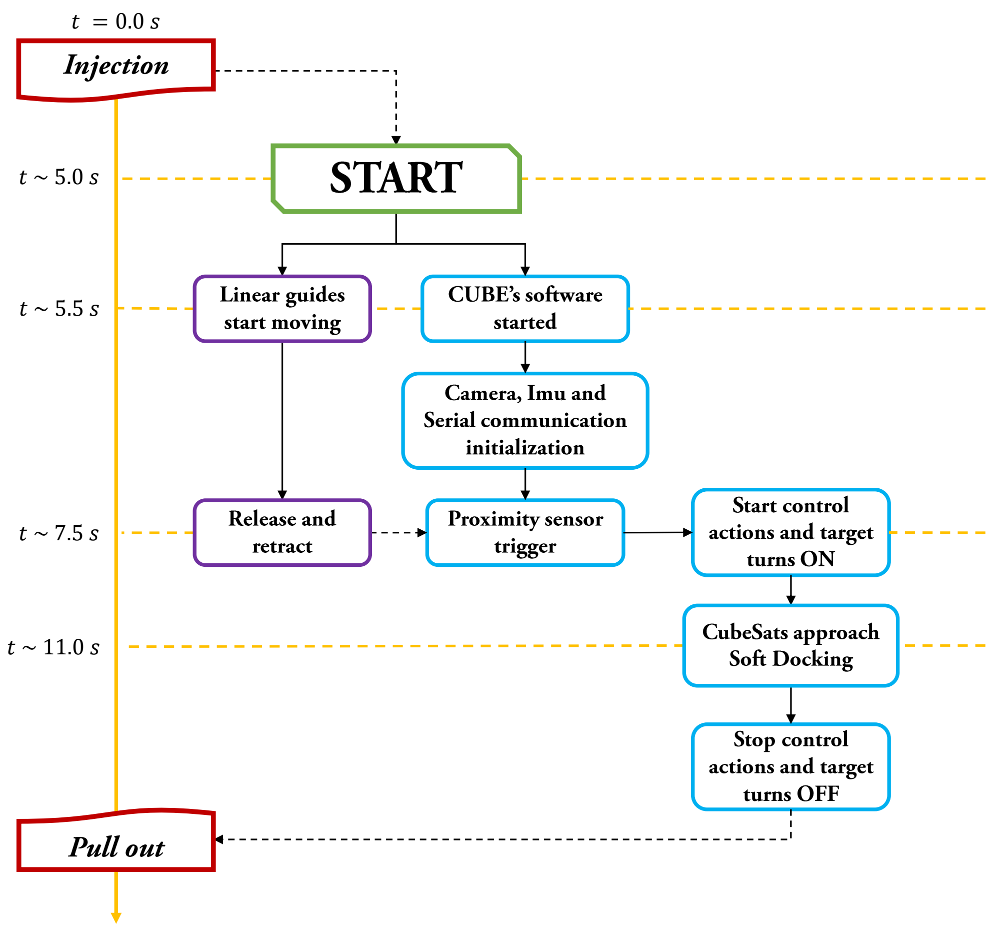

t = 0.0 s
Injection
t ~ 5.0 s
START
t ~ 5.5 s
Linear guides start moving
CUBE's software started
Camera, Imu and Serial communication initialization
t ~ 7.5 s
Release and retract
Proximity sensor trigger
Start control actions and target turns ON
t ~ 11.0 s
CubeSats approach Soft Docking
Stop control actions and target turns OFF
Pull out

Figure 4.1: PACMAN Experiment flowchart. In Red are reported the flight events, where injection represents the start of the micro-gravity and Pull Out the start of the hyper-gravity. The violet rectangles report the Hold & Launch system events, while the blue rectangles represent the CubeSats events. Finally, on the left is reported an approximate timeline of the experiment.

It is worth to underline here that the experiment is almost completely automated, the experimenters in fact only have to initialize the system and send the start signal during the micro gravity phase through the dedicated GUI.

4.2 Experimental Results

The flight campaign has been very productive, but not all the parabolas provided in the 3 successive flights produced the desired results.
After the last flight, we noticed a thinning of the adhesive placed between the electromagnet and the Hold & Release Interface of the CubeSats; probably for this reason, the experiment failed in the last parabolas, because the residual magnetization in the CubeSat iron plates did not allow the perfect detachment during the micro-gravity phases.
During the experimental tests, the most difficult task has been that of correctly adjust the release instant and the initial linear velocity provided to both the CubeSats through the Hold and Launch subsystems.
In fact, because of the residual magnetization on the ferromagnetic core of the electromagnets and the Hold & Release interfaces of the CubeSats, the imposed initial velocity from the motorized linear guides was not very precise and reliable.
Furthermore, the friction between the CubeSats and the Hold & Launch subsystems imposed at the release instant an unwanted initial angular velocity to the nanosatellites in many tests.
For these reasons, the experimental data are very difficult to analyse, thus only a preliminary analysis has been performed at the moment of the writing of this thesis.

During the first flight, the first two sets of parabolas have been used to find a good set of parameters to initialize the overall system in terms of initial linear velocity imposed, release relative distance between the CubeSats and release instant during the micro-gravity phases.
During the flights it was not possible to measure the actual imposed initial velocity, thus it has been set approximately by the experimenters, resulting in post-process probably too high for the developed system.
However, after setting the initial conditions, some promising results have been obtained.

From the collected data of the accelerometer it can be retrieved the actual instants when the CUBE has been released and has docked during the parabola, so as to calculate the total time of the manoeuvre and isolate the data of the angular velocities necessary for the reconstruction of the motion, as shown in Figure 4.2.

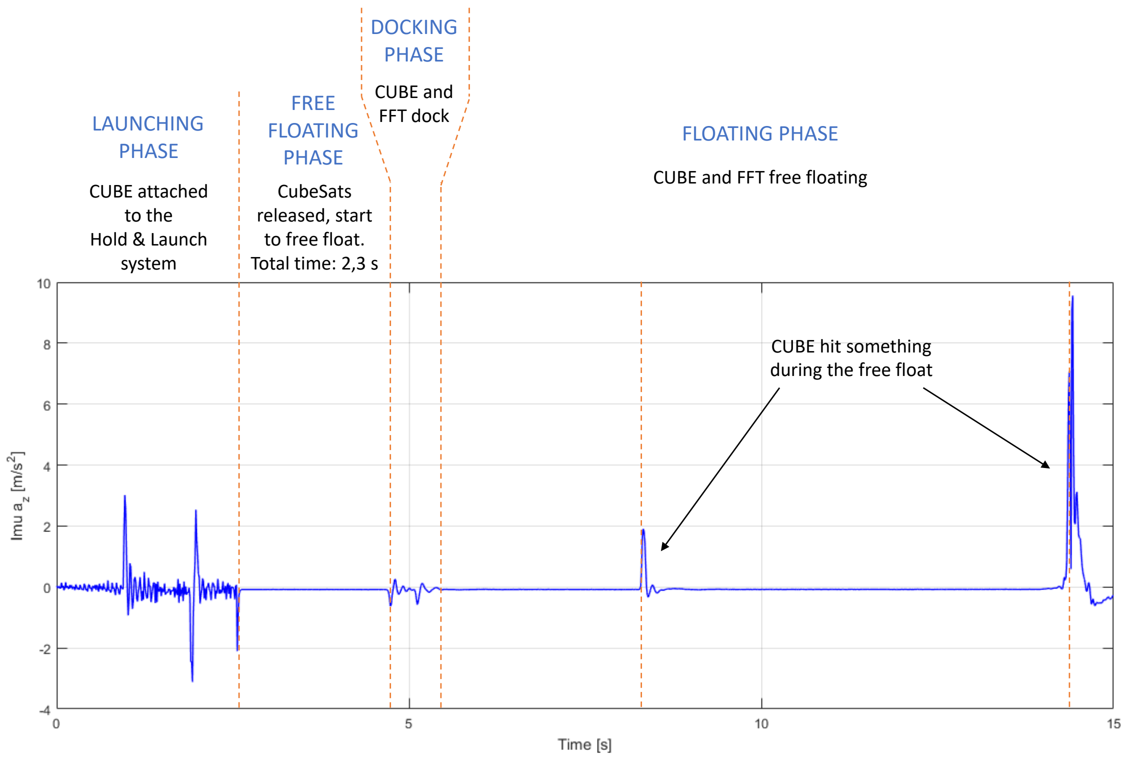


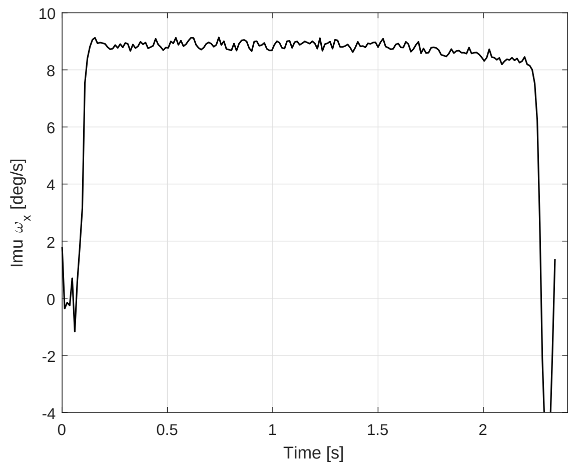
Figure 4.2: Acceleration profile of the CUBE on its z-axis collected by the on-board IMU during the 28th parabola of the 1st flight..

Furthermore, from the obtained data, given the release relative distance, it is also possible to estimate the actual initial velocity imposed by the Hold & Launch subsystems. For example, in this case, the two CubeSats were release at $\approx 18\,cm$ of distance between each other, thus, considering the total time of the free floating phase of $\approx 2.3\,s$, we can estimate the initial linear velocity v imposed to both the CUBE and the FFT as:

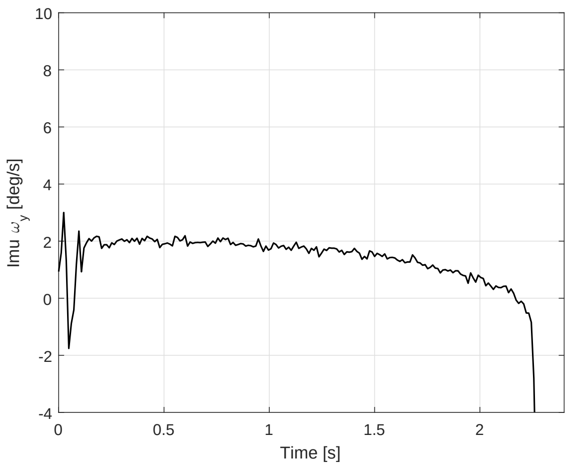
$$v = \frac{\text{relative distance}}{\text{total manoeuvre time}}/2 \simeq \frac{18\,cm}{2.3\,s}/2 \simeq 3.9\,cm/s \tag{4.1}$$

which is almost double of the simulated initial velocity imposed of $2\,cm/s$.
In Figure 4.3 there are reported the collected data of the angular velocities of this particular docking manoeuvre, coming from the IMU board inside of the CUBE.
In this test, the CUBE's controller was set with the parameters of

$$K_P = 8, \qquad K_I = 0, \qquad K_D = 3.6$$



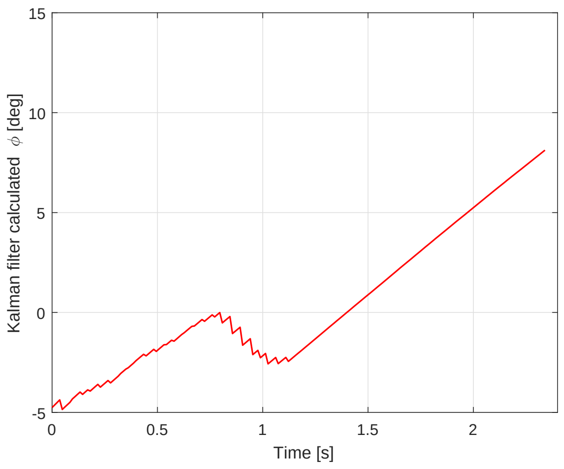
(a) Angular Velocity ω_x



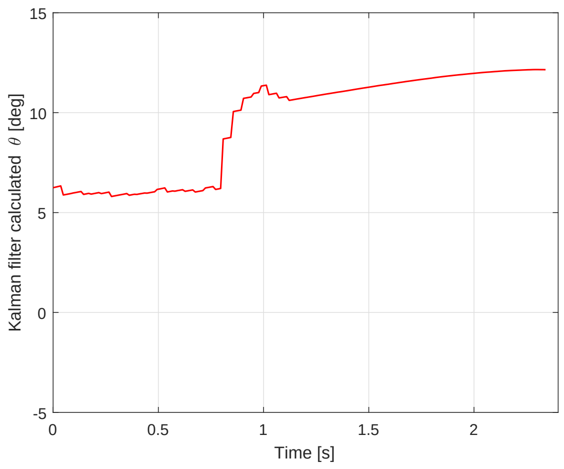
(b) Angular Velocity ω_y

Figure 4.3: Angular velocities collected by the IMU board inside the CUBE during the 28th parabola of the 1st flight.

As it can be noticed, the friction between the CUBE and the Hold & Launch subsystems imposed a relevant initial angular velocity on the x and y-axes of the CUBE, considering the CUBE's reference frame reported in Figure 2.19.
In the last phase of the manoeuvre, where the two CubeSats are close and the magnetic interactions are stronger, the initial angular velocities were a little attenuated by the controller, but not enough to assure a perfect docking.
In fact, as it can be seen in the filtered data of the relative orientation in Figure 4.4, the resulting control actions were not strong enough to stop the CUBE's initial angular velocities and both the Roll and Pitch misalignment angles was not recovered during this particular parabola.
Furthermore, from Figure 4.4 it can also be noticed that in the middle of the manoeuvre, when the CubeSats were close, the camera software poorly estimated the relative orientation, increasing the misalignment error of about 5 degrees for the Pitch angle and decreasing it of ≈ 2.5 degrees for the Roll angle.
This error is probably due to the exit from the camera field of view or the reflections of one or more LEDs of the pattern, which led the CUBE's software to calculate the relative orientation with a large error.



(a) Roll



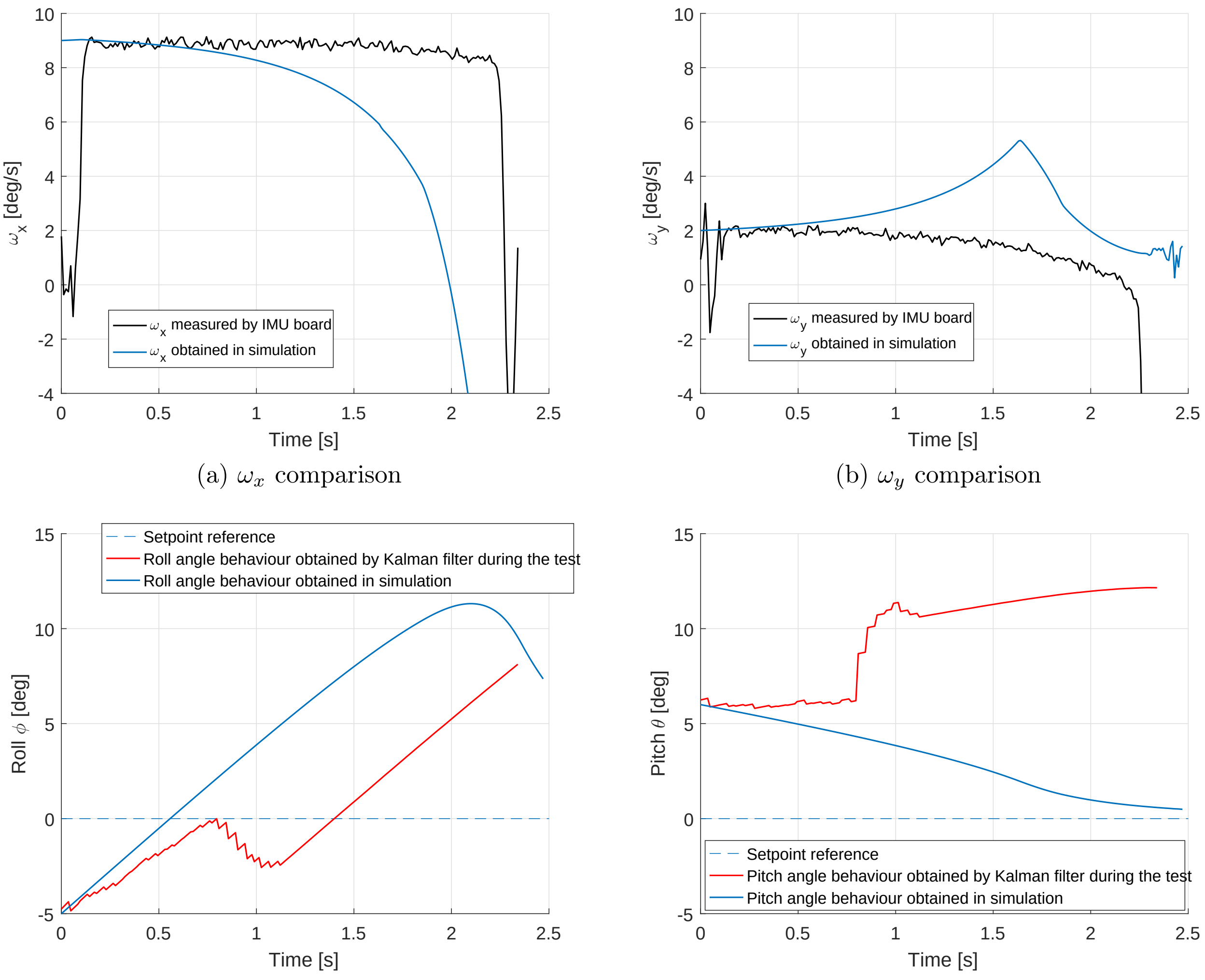
(b) Pitch

Figure 4.4: Filtered data of the relative orientation of the CUBE with respect to the FFT. Data obtained during the 28th parabola of the 1st flight.

It can be worth here to compare this data with the simulation data obtained by imposing almost the same initial conditions obtained during this particular test. Then, by imposing the following parameters:

– Initial CUBE Roll relative angle w.r.t. the FFT: $-5\,deg$;
– Initial CUBE Pitch relative angle w.r.t. the FFT: $6\,deg$;
– Initial CUBE angular velocity in its x axis: $9\,deg/s$;
– Initial CUBE angular velocity in its y axis: $2\,deg/s$;
– Initial linear velocity of both the CUBE and the FFT : $3.9\,cm/s$;
– Relative release distance: $18\,cm$;
– PID parameters of both the regulators: $K_P = 8,\ K_I = 0,\ K_D = 3.6$.

the obtained results are compared with the data collected during the test in Figure 4.5.



(a) ω_x comparison

(b) ω_y comparison

(c) Roll comparison

(d) Pitch comparison

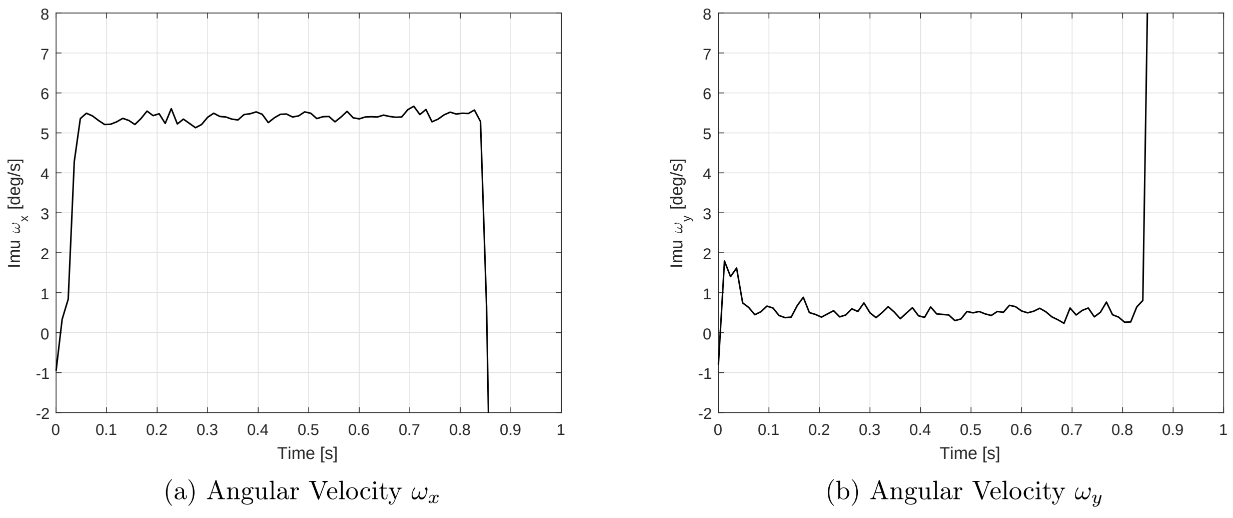
Figure 4.5: Comparison between the experimental and the simulation results obtained setting approximately the same initial conditions.

As it can be observed from Figure 4.5, the simulated behaviour of both the angular velocities and the Pitch and Roll angles is faster than the actual data obtained from the test, meaning that the real interactions between the two CubeSats were less effective in the real environment than in the simulations, thus to obtain the same performances probably faster and stronger control actions were needed.
However, if the camera measures did not get the error previously explained, the Roll and Pitch angles behaviour could have been similar to the simulated ones, but probably with slower performances.

Another interesting test comes from the second flight, where the controller parameters of the PID regulators of the CUBE were set to

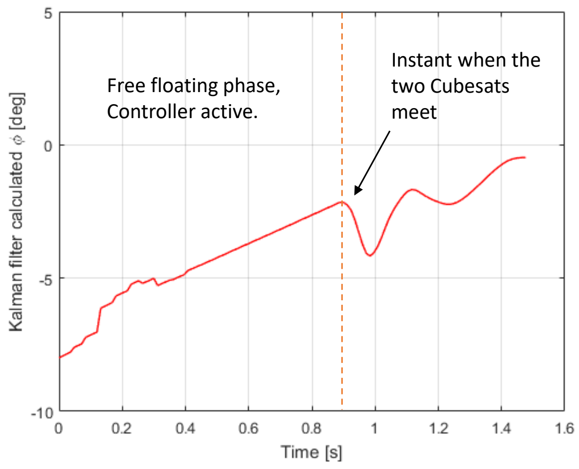
$$K_P = 12, \qquad K_I = 0, \qquad K_D = 3.6$$

where the proportional action gain was increased trying to improve the performances of the overall system.
The collected data of the angular velocities is reported in Figure 4.6, while the relative filtered data, merging the estimations coming from the camera and the IMU angular velocities, are reported in Figure 4.7.
The data relative to the free floating phase has been obtained as explained above, from the acceleration profiles of the CUBE.
In this test, the total free floating time before the docking phase was even smaller, of about $\approx 1\,s$ with an initial relative distance of $\approx 15\,cm$, which lead to an estimated imposed initial velocity of approximately $7.5\,cm/s$.
In these conditions, thanks also to the initial angular velocity imposed by the Hold & Launch system due to the friction, the two CubeSats docked with a good alignment, with a final error of ≈ 2 degrees for both the Roll and Pitch angles.
However, also in this test the control actions were not strong enough to impose a perfect alignment in such a small time.

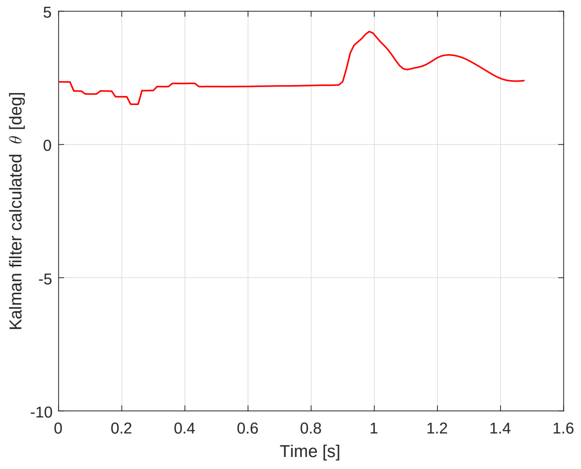


(a) Angular Velocity ω_x (b) Angular Velocity ω_y

Figure 4.6: Angular velocities collected by the IMU board inside the CUBE during the 2nd parabola of the 2nd flight.



(a) Roll



(b) Pitch

Figure 4.7: Filtered data of the relative orientation of the CUBE with respect to the FFT. Data obtained during the 2nd parabola of the 2nd flight.

In particular, with reference to Figure 4.7, in the last phase of the manoeuvre, it can be observed the docking phase between the two CubeSats (the Pitch and Roll angles start oscillating), when they meet and finally almost perfectly dock.

Finally, in order to show the variability of the undesired initial angular velocity imposed by the friction between the CUBE and the Hold & Launch subsystem, in Figure 4.8 it is reported a comparison of the angular velocities recorded by the IMU inside the CUBE between 6 parabolas accomplished during the first flight.

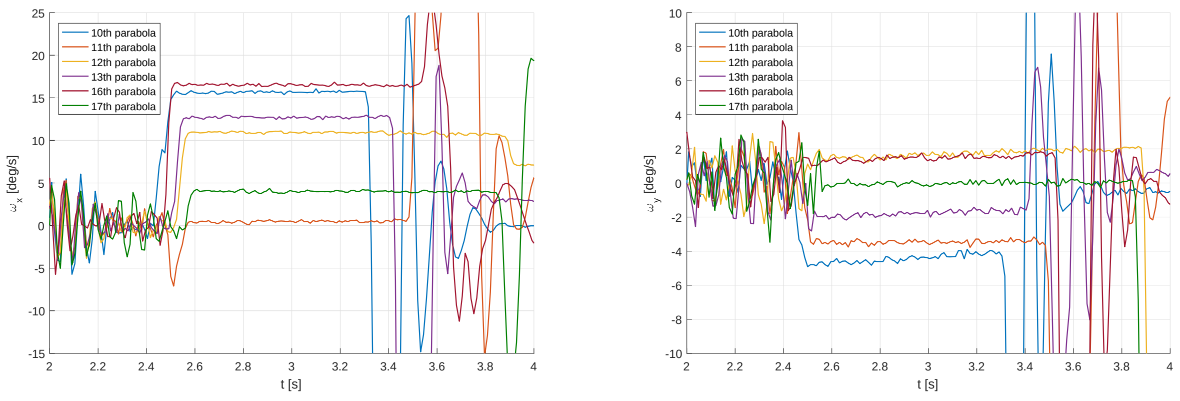


Figure 4.8: Comparison of the angular velocities recorded by the IMU inside the CUBE during 6 parabolas accomplished in the 1st flight. Tests performed with the same initial conditions. In all the selected parabolas at $t \approx 2.45$s the CUBE is released during the micro-gravity phases.

In the Figure 4.8, the time at which the two CubeSats meet is represented by the step change of the angular velocities, which stop their linear behaviour.
As it can be noticed, due to the experiment conditions and some un-modelled dynamics, the repeatability of the experiment with the developed launching system is not very high, on the contrary, on each parabola the imposed initial angular velocity has a different magnitude, even keeping unchanged the initial conditions.
For this reason, it is very difficult to compare and analyse the obtained data, because in each test the actual initial conditions are very different from others.
In order to reduce this problem, a solution can be that of remove the plastic teeth on the Hold & Launch Interface (see Figure 2.5) or move them along a horizontal axis, parallel to the plane's ground, to reduce the effect of the friction during the release phase. Another solution can be that of changing completely the launching system, using for example a spring based system, activated burning a cable that keep the spring compressed before the micro-gravity phases. However, such a solution was considered during the design phase of the experiment, but it has been discarded for feasibility and safety reasons imposed by NoveSpace to fly.

In general, in all the analysed tests the initial conditions were not as good as the imposed ones in the simulation environment; in particular, the estimated imposed initial linear velocities have been found to be probably too high for the developed system, leading to results that are difficult to interpret.
Moreover, in some tests the data files appeared to be corrupted and so some parabolas need to be analysed only with the data collected from the external camera vision system, which needs more time and post-process to be properly analysed.
However, from a preliminary analysis we can conclude that the soft docking and proximity navigation exploiting the magnetic interaction between two spacecrafts in micro gravity conditions is a promising technology, but it requires a longer time to that provided during the flights. For example, in a real scenario of soft docking between two spacecrafts in orbit, there is no limit to the manoeuvre time, thus the initial linear velocity can be very small, the magnetic interactions can be controlled more efficiently and the controller does not need to grant fast rising time with almost null overshoots, as in our test conditions.

5 Conclusions

In this work we examined in detail the overall PACMAN experiment and all its subsystems, putting more focus on the controller and the software system that controls the experiment.
We started from the description of every system that composes the experiment and we analysed in detail each component of which it is made of. In particular, we saw how the CUBE, the key system of all the experiment, analyses the data coming from its sensors in order to produce an appropriate output and adjust its relative attitude with respect to the Free Floating Target through the electromagnetic actuators.
We also shown all the other systems that work with and around the CUBE, such as the FFT, which generate the static magnetic field necessary to exploit the magnetic interactions by the CUBE, the CHAMBER where the two CubeSats are free to float and the Support Electronics, which is necessary to impose the required initial conditions.

We then reported the simulated model on which all the experiment is based, and we shown the results obtained from the simulation environment.
Besides, we presented the actual experiment procedure employed during the flight campaign in Bordeaux, on December 2017.
Finally, a preliminary analysis on the collected data has been reported in order to understand the actual possibilities and limits of the developed system.
As discussed in Section 4.2, the main issue of the developed system was represented by the friction between the Hold & Launch subsystems and the CubeSats, which imposed an initial angular velocity when the latter were released and the Launch system was retracting its arm.
As shown in Figure 4.8 the imposed angular velocity is of random magnitude and introduces a large error on the final relative orientation of the CUBE in most of the tests.
Furthermore, during the flights we could not measure the actual initial linear velocity imposed by the Launch system to both the CubeSats, so it has been set approximately by the experimenters, resulting in an initial linear velocity probably too large for the developed system.

A solution to both these problem, could be that of using a different Launch system, based for example on springs to launch the CubeSats one towards the other, but then it must be developed a different system able to hold the two nano-satellites during the hyper-gravity phases, which should be also compliant with the safety requirements needed to fly.
However, comparing the collected data to the data obtained in the simulation environment has shown that the real interactions between the two CubeSats were in general less effective than in the simulations.
For this reason, during the second and third flights the proportional gains of the controller have been increased in order to improve the time response of the overall system. As a result, the control actions resulted to be still not strong enough to properly overcome the unwanted imposed initial angular velocities.
Besides, in some cases the estimated relative orientation provided by the camera has been corrupted by an error due to the exit of the pattern from the field of view of the camera or by reflections due to the LEDs too close to the camera lens.
Nevertheless, interesting data have been obtained that can be further analysed together with the data coming from the external vision system, from which it can be obtained a better estimation of the dynamical behaviour of the experiment.

In the end, the experiment can be considered as a good starting point for the study of the magnetic interactions for proximity navigation in the next years.

In conclusion, the overall experiment developed under the ESA Fly your thesis! programme has been an amazing experience and working side by side with ESA and NoveSpace engineers has been a priceless occasion to increase our knowledge and experience. During the last year, we worked hard to properly design and develop every single subsystem and detail of the PACMAN experiment, even facing big and small problems emerged during the development phase.
Nevertheless, all the work has been turned into a large payload, which merged with the experience and competences gained, made this experience invaluable and unforgettable.

Bibliography

[1] Arduino: *"ARDUINO UNO WIFI"*. Available at: `https://store.arduino.cc/usa/arduino-uno-wifi`

[2] Brett Beauregard: *"PID library"*. Available at: `http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/`

[3] Ruggero Carli: *"Lecture notes"*, Academic year 2016/2017.

[4] Angelo Cenedese: *"Lecture notes"*, Academic year 2016/2017.

[5] ESA: *"Fly your thesis! programme"*. Available at: `http://www.esa.int/Education/Fly_Your_Thesis/Fly_Your_Thesis!_programme`

[6] Wigbert Fehse, *"Automated Rendezvous and Docking of Spacecraft"*

[7] Andrew R. Hilton, Gregory J. Eslinger and David W. Miller, *"Dynamics modeling of electromagnetic formation flight"*

[8] Faisal Karmali, Mark Shelhamer: *"The dynamics of parabolic flight: flight characteristics and passenger percepts"*

[9] Laurent Kneip, Davide Scaramuzza, Roland Siegwart: *"A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation"*

[10] Peter B. Landecker, Daniel D. Villani and Kar W. Yung: *"An Analytic Solution For The Torque Between Two Magnetic Dipoles"*

[11] Novespace, *"A310 ZERO-G USER GUIDE"*

[12] OAAN Fact Sheet: *"On-Orbit Autonomous Assembly from Nanosatellites"*. Available at: `https://www.nasa.gov/sites/default/files/atoms/files/oaan_fact_sheet-26oct2015.pdf`

[13] D. Petrillo et al.*"Flexible Electromagnetic Leash Docking system (FELDs) experiment from design to microgravity testing"*

[14] Phidgets: *"PhidgetSpatial Precision 3/3/3 High Resolution"*. Available at: `https://www.phidgets.com/?tier=3&catid=10&pcid=8&prodid=1038`

[15] Pololu Robotics & Electronics: *"Pololu A4990 Dual Motor Driver Shield for Arduino"*. Available at: `https://www.pololu.com/product/2512`

[16] Allison K. Porter et al., *"Demonstration of Electromagnetic Formation Flight and Wireless Power Transfer"*

[17] Raspberry: *"RASPBERRY PI 3 MODEL B"*. Available at: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`

[18] Raspberry: *"PI NOIR CAMERA V2"*. Available at: `https://www.raspberrypi.org/products/pi-noir-camera-v2/`

[19] Luca Schenato: *"Lecture notes"*, Academic year 2015/2016.

[20] Samuel A. Schweighart, *"Electromagnetic Formation Flight Dipole Solution Planning"*.

[21] Craig Underwood, Sergio Pellegrino, Vaios J. Lappas, Christopher P. Bridges, John Baker: *"Using CubeSat/micro-satellite technology to demonstrate the Autonomous Assembly of a Reconfigurable Space Telescope (AAReST)"*

[22] Zhengyou Zhang: *"A Flexible New Technique for Camera Calibration"*

[23] Zizung Yoon: *"Introduction into quaternions for spacecraft attitude representation"*, May 31, 2012.

A PhidgetSpatial Precision 3/3/3 High Resolution datasheet

Product Specifications	
Accelerometer	
Acceleration Measurement Max	± 2 g
Acceleration Measurement Resolution	76.3μg
Acceleration Bandwidth	497 Hz
Accelerometer White Noise σ	280μg
Accelerometer Minimum Drift σ	40.6μg
Accelerometer Optimal Averaging Period	398 s
Backup Accelerometer	
Acceleration Measurement Max	± 8 g
Acceleration Measurement Resolution	976.7μg
Accelerometer White Noise σ	2.8 mg
Gyroscope	
Gyroscope Speed Max (X-Axis, Y-Axis)	$\pm 400°/s$
Gyroscope Speed Max (Z-Axis)	$\pm 300°/s$
Gyroscope Resolution (X-Axis, Y-Axis)	$0.02°/s$
Gyroscope Resolution (Z-Axis)	$0.013°/s$
Gyroscope White Noise σ	$0.095°/s$
Gryoscope Minimum Drift σ	$0.0042°/s$
Gyroscope Optimal Averaging Period	7743 s

Backup Gyroscope	
Gyroscope Speed Max	$\pm 2000°/s$
Gyroscope Resolution	$0.07°/s$
Gyroscope White Noise σ	$0.59°/s$
Compass	
Magnetic Field Max	5.5 G
Compass Resolution	3 mG
Compass White Noise σ	1.1 mG
Compass Minimum Drift σ	78μG
Compass Optimal Averaging Period	1443 s
Board	
Controlled By	USB
API Object Name	Accelerometer, Gyroscope, Magnetometer
Current Consumption Max	55 mA
Sampling Speed Min	1 s/sample
Sampling Speed Max	4 ms/sample
Sampling Speed Min (Webservice)	1 s/sample
Sampling Speed Max (Webservice)	12 ms/sample
Analog to Digital Converter Resolution	16 bit
USB Voltage Min	4.4 V DC
USB Voltage Max	5.3 V DC
USB Speed	Full Speed
Operating Temperature Min	-40 °C
Operating Temperature Max	85 °C