Treball de fi de Màster

Màster Universitari en Enginyeria Industrial (MUEI)

Design, implementation and integration of an autonomy payload for a quadruped-legged robot.

Joan López Zamora

Director: Solà Ortega, Joan
Co-director: Santamaria-Navarro, Angel
Colaborador: Duarte, Hugo Miguel

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

July, 2023

Resum

Aquesta tesi final de màster versa sobre la descripció, disseny i implantació d'un conjunt de millores que fan possible l'autonomia d'un robot quadrúpede.

El conjunt de millores descrites es troben separades en dos grups per millorar la comprensió el lector. Aquests són: *Hardware* i *Software*.

Durant la primera part, es descriu al complet: el dimensionament, el disseny i la implementació dels elements mecànics i electrònics (bateries, ordinador a bord, càmera, i elements passius) que formen part de la millora en el *hardware*. Aquesta inclou el disseny i impressió en 3D, així com el procés d'acoblament al robot original.

Pel que fa a la millora en "Software"; es planteja, descriu, i implementa un observador d'estat amb la intenció futura d'implementar-lo al mateix robot. L'observador es descriu mitjançant l'ús d'un conegut algoritme de fusió de dades anomenat Filtre de Kalman (on, en aquest cas, se n'utilitza la versió estesa).

Per provar l'observador dissenyat, es realitza una simulació amb dades generades artificialment a partir d'un model cinemàtic. A continuació, tot i que la futura implantació es pretén que sigui al robot quadrúpede, per poder prova el filtre amb dades reals, es registren les dades del vol d'un dron fent ús de la comunicació ROS[1] 2 Galactic. De les mateixes dades, es torna a experimentar amb el filtre, però, aquest cop, fent ús del registre de dades del controlador del dron (model PX4).

El treball conclou amb quatre estudis addicionals sobre la tasca: econòmic, temporal, social i mediambiental; així com amb unes conclusions que inclouen propostes de futures tasques a portar a cap.

[1] Robot Operating System (ROS) [Ope23]

Abstract

This master thesis is based on the description, design and implementation of a set of improvements that make possible the autonomy of a quadruped robot.

The set of improvements described are separated into two large groups that are disassociated in this memory as Hardware and Software, in order to improve the reader's comprehension.

During the first part, the dimensioning, design and implementation of the mechanical and electronic elements (batteries, on-board computer, camera, and passive elements) that are part of the improvement in the *hardware* is described in full. This part, in turn, includes the design and 3D printing, as well as the assembly process to the original robot.

As for the improvement in software, a state observer is proposed, described and implemented with the future intention of being implemented in the robot itself. This is described through the use of a well-known data fusion algorithm called Kalman Filter (which, in this case, is used through its extended version).

To test the designed observer, three experiments are performed. The first one is based on a simulation with artificially generated data from a kinematic model. The second is based on real data collected from the flight of a drone. It also uses the ROS[2] 2 Galactic operating system to establish communication between data. Finally, the third uses data extracted from the drone's own controller (PX4 model).

The thesis ends with four additional studies: economic, temporal, social, and environmental, as well as conclusions that incorporate proposals for future lines of work.

[2]Robot Operating System (ROS) [Ope23]

Resumen

Esta tesis final de máster se basa en la descripción, diseño e implementación de un conjunto de mejoras que hacen posible la autonomía de un robot cuadrúpedo.

El conjunto de mejoras descritas se encuentran separadas en dos grandes grupos que se desasocian en esta memoria como *Hardware* y *Software*, a fin de mejorar la compresión del lector.

Durante la primera parte, se describe al completo: el dimensionamiento, el diseño y la implementación de los elementos mecánicos y electrónicos (baterías, ordenador a bordo, cámara, y elementos pasivos) que forman parte de la mejora en el *hardware*. Esta parte, a su vez, incluye el diseño e impresión en 3D, así como el proceso de ensamblaje al robot original.

En cuanto a la mejora en *software*; se plantea, describe e implementa, un observador de estado con la intención futura de ser implementado en el mismo robot. Este se describe mediante el uso de un conocido algoritmo de fusión de datos llamado Filtro de Kalman (que, en este caso, es usado mediante su versión extendida).

Para probar el observador diseñado, se realizan tres experimentos. El primero se basa en una simulación con datos generados artificialmente a partir de un modelo cinemático. El segundo, aunque la futura implementación está pensada sobre el robot cuadrúpedo, se realiza utilizando datos reales recogidos del vuelo de un dron, haciendo uso de la comunicación ROS[3] 2 Galactic. Por último, el tercero utiliza datos extraídos del propio controlador del dron (modelo PX4).

La tesis finaliza con cuatro estudios adicionales sobre la misma: económico, temporal, social y medioambiental; así como, con unas conclusiones que incorporan propuestas para futuras líneas de trabajo.

[3]Robot Operating System (ROS) [Ope23]

Contents

Acronyms

ABS Acrylonitrile Butadiene Styrene. 7, 15, 17–20, 51, 54

CSV Comma-Separated Values. 36, 41, 44, 58

ECTS European Credit Transfer and Accumulation System. 50, 52

EKF Extended Kalman Filter. 4, 25, 27, 41, 45–48

FDM Fused Deposition Modeling. 7, 15

FLU Front Left Up (frame convention). 39, 42, 45, 58

FRD Front Right Down (frame convention). 42

HIDRO High Dynamics Robots. 1, 3, 8, 10, 11, 13–15

HIPS High Impact Polystyrene. 17, 18, 51, 54

IMU Inertia Measurement Unit. 2, 4, 5, 7, 14, 15, 23, 28–31, 33, 34, 37–42, 45, 58, 67, 69, 75, 76

IRI Institut de Robòtica Industrial. 1

KF Kalman Filter. 24–26

LEKF Lie Extended Kalman Filter. 4, 23, 26–28, 33, 34, 38–40, 45, 47–49, 70

LIDAR Light Detection and Ranging. 14, 16, 58

LiPo Lythium Polymer. 4, 8, 9, 13, 58

ODRI Open Dynamic Robot Initiative. 1, 6, 7, 12, 17

OT Opti Track. 2, 4, 5, 14, 15, 31, 33, 34, 36, 38, 42–49, 58, 69, 71, 74–76

PCB Printed Circuit Board. 13

PD Power Distributor. 13

PDB Power Distribution Board. 13, 18

ROS Robot Operating System. 1, 33, 36, 38, 41, 45, 58

URDF Unified Robot Description Format. 22, 58

Glossary

LOG LOG files are computer-generated data files. 4, 36, 41, 42, 48

NUC NUC is the commercial name of a series of mini-pcs sold by Intel Mini PC Intel®. 4, 5, 14–16, 18, 19, 21, 22, 58, 66

PX4 PX4 is an open source flight control software for drones and other unmanned vehicles [aut23]. 4, 36, 41–43, 45–48, 58, 76

SOLO SOLO is the name of the quadruped robot subject of this thesis. 1–8, 10–15, 17, 18, 20, 21, 28, 36, 46, 51, 53, 58

List of Figures

List of Tables

1 Preface

1.1 Introduction and context

Studies on quadrupedal robots, also known as quadrupeds, date back to the mid-20th century. It forms part of a long tradition of animal-based studies, which provides an everlasting source of inspiration to physicists and engineers.

In the latest models available on the market, their dynamic capabilities show robustness to adapt to any situation. Given this fact and their tendency to incorporate manipulator's arms, these robots are beginning to gain a presence in industrial sectors such as the chemical industry (performing maintenance and safety tasks, among others).

One of the first notable precursors of the quadruped's investigation is the Stanford Quadruped [Kau+19], first developed by the Stanford Leg Lab in the 1980s. With this robot, more universities are doing their own investigations with famous names like the Massachusetts Institute of Technology (MIT) with its robot Cheetah [Tec19] or their spin-off Boston Dynamics with their Spot [MIT23]. European institutes are also developing a quadruped model with the spin-offs from the Swiss Federal Institute of Technology (ETH Zurich) with ANYbotics and their ANYmal [ANY23].

1.2 Open Dynamics Robot Initiative

Among the research on quadruped robots, the Open Dynamic Robot Initiative (ODRI) offers a cheap, easy-to-assemble solution for those centers interested in this field [ODR23d]. In the form of a quadruped robot, SOLO is a full 3D-printed robot composed of a set of electronic and mechanical components that have been changing as new versions have come up.

SOLO 12, the last version of the quadruped, is an upgraded version of its predecessor, SOLO 8. The great improvement this second version offers is an increase of 4 degrees of freedom (via 4 new engines).

Aware of the existence of this second version, a group of researchers of High Dynamics Robots (HIDRO) at the Institut de Robòtica Industrial (IRI)[IRI23], started to use it to develop and study high-dynamic movements (e.g., acrobatics, sprints, agile movements, etc.). The purchase of SOLO 12 can be done through the company PAL Robotics [PAL23].

(a) SOLO 8. (b) SOLO 12.

Figure 1: Both versions of quadruped robot SOLO.

1.3 Motivation

What motivates this thesis is the possibility of making the SOLO 12 robot more autonomous within the framework of high dynamics.

As it will be seen, SOLO 12's actual version is somehow limited in its ability to perform a wide range of agile maneuvers. That is because it requires a wired connection to a constant power supply as well as to an off-board computer with an Ethernet connection. The intention with the addition of the payload, then, is to be able to practice agile maneuvers without the constraint of wires or powerful onboard electronics.

On the other hand, to study the control required to perform most of the agile performances, it is a good practice to develop a state estimator that specifies the relevant variables of the robot at high frequency. That is why the definition of a state estimation system that fuses a high-frequency sensor, the Inertia Measurement Unit (IMU), and a high-precision sensor, the motion registration system Opti Track (OT), is considered a step in the autonomy upgrade.

The motivation of the project, in short, could be summarised in a list that establishes where the project starts from, and where it wants to go.

- From being mechanically constrained by wires such as power supplies to establishing only onboard wiring.
- From having a limited power of computation to establishing a powerful onboard computer.
- From requiring an ethernet connection from an offboard computer to establishing a connection through WiFi between computers.
- From a poor sensory system to being able to establish a connection with the environment.
- From only sharing odometry from the motors to begin the path to a whole-body estimator.

1.4 Open source

This project is possible, in a way, thanks to the open-source philosophy. For those not versed in this terminology, open source refers to a collaborative approach to software and hardware development where the source code of software and the design of hardware are made freely available and can be modified, studied, and distributed by anyone. This open nature encourages transparency, collaboration, and community-driven innovation.

Open source projects are typically governed by licenses, such as the GNU General Public License (GPL) or the MIT License (MIT), which define the terms and conditions for using, modifying, and distributing the software.

That being said, the whole project of SOLO is open to anybody through GitHub [ODR23c], a popular web-based platform for version control and collaboration that plays a significant role in the open-source ecosystem [GIT23].

1.5 Scope and extend of the work

This work covers both the dimension, design, and implementation of the autonomy payload. It also covers the definition, design, simulation, and implementation of an estate estimator. Both subjects are brought into reality through 3D printing and real experiments.

When it comes to future elements for upgraded versions, the mechanical design leaves space for the implementation of new sensors, communication devices, or improved electronic devices.

On the other side, the state estimator version is expected to be upgraded with features that include delay problems, faster compilation, etc. Commonly, during the testing of any filter that fuses data, many signal problems appear. To deal with some of them would require a far more extensive scope than this work does not consider. That is why the experimentation, the last chapter of the state estimator, focuses on trying to determine if the fusion filter works without going beyond whether the solving requires specific disciplines.

1.6 Objective

On one hand, as an initiative of the IRI, this master's thesis aims to develop an autonomous payload for SOLO 12. Through this update, the group at High Dynamics Robots (HIDRO) sets up the path to investigate complex control performances thanks to the required hardware. These required pieces of hardware must fulfill the next objectives:

- Wireless communication with the robot.
- A lightweight alternative to the actual power supply.
- Adding extra sensors.
- Design of 3D printed parts for the assembly that keep the new hardware parts fixed and save.

On the other hand, after the hardware implementation and following the objective of setting the conditions needed to perform high-dynamics techniques, it is desired to establish some "first steps" toward a complete state estimator. That is why the second main objective of this master's thesis is to define and test the estimator. The development of the estimator must fulfill the following requirements:

- Establish a theoretical approximation to the fusing of a high-rated low-accuracy sensor with a low-rated high-accuracy sensor
- Get a software version of such filter in order to test it through simulated data.
- Perform tests with the laboratories' drone to analyze its performance on a real mobile robot.

2 Hardware

2.1 Main hardware parts

When discussing the minimum hardware parts that most robots would have, there are several essential components to consider. These components enable robots to interact with their environment, perceive information, process data, and execute actions. Among any possible list, there can be a distinction between the hardware parts that allow physical tasks through actions (mechanical) and those that allow the same tasks through the software inside them (electronics). The mechanical parts would not be able to operate under precise control without the electronics, and the electronics would lack sense without the mechanical parts.

If the focus is on mobile robotics, such as quadrupeds, the distinction made before could be summarized in the next two tables. Both Table 1 and Table 2 give a general description of the elements listed in the left column. This description is expected to guide the following chapters in order to know what to change if an autonomous mobile robot is desired. In the next chapter, 2.2 *Initial hardware's state*, this same list will take into account SOLO 12 initial hardware specifications.

Item	Basic description
Body/ chassis	Allows assembly of any hardware part. Its design is thought to allow the functionality of the robot and, at the same time, protect the rest of the components (mainly electronics and wires).
Legs/ arms/ wheels	Mobile robots, by definition, must be able to have some degrees of freedom (DoF) by which they can move. In some cases, these can be propellers, wheels, or legs. When it comes to functionalities, besides movement, the same elements, or a combination of them, can be used to perform duties such as grasping, cutting, etc.[RB23]
Actuators	For sure, the DoF must be produced by some actuator that transforms a source of energy, such as electricity, into mechanical forces (either linear ones or rotating ones). When it comes to motors, the most commonly used actuators in robotics are BLDC (direct current brushless), which is one of the most common to find [sin23]. Nevertheless, it is worth knowing that other actuators can also be found on robots such as Disney [Gui23], which combines electricity with fluids like water or air.
Gears, driving belts	The torque, or rotational force, offered by motors is in need of gears and driving belts in order to achieve the desired power in a certain space. That's why most of the robots, inside their chassis, use a combination of gears and belts in order to reduce and/or increase velocities and forces.
Sensors	Either for agile control or mapping, sensors are the hardware elements that collect any type of data. Depending on the sensor and where it is placed, this information will vary depending on how it is collected and treated. For instance, some of the "most attached to the body" sensors are IMU. IMUs are able to track variations in linear acceleration, rotational velocity, and, in some cases, magnetic fields. This kind of sensor is able to do so through gyroscopes, accelerometers, and magnetometers. On the other hand, special sensors such as GPS, tracking cameras, etc. are placed outside the robot's body in order to track its movement.
Power Supply	Robots need a power source to operate their electronic components and actuators. The power supply can vary depending on the robot's size, mobility, and power requirements. It can include batteries, power cables, or even energy harvesting systems.

Table 1: Average mechanical hardware of a quadruped robot.

Item	Basic description
Micro-drivers	Combining a microprocessor core, some peripherals, and memory, micro-drivers are printed circuit boards (PCB) that provide a compact and cost-effective solution for embedded control applications that require real-time operation, low power consumption, and the integration of various functions on a single chip. Micro-drivers are commonly referred to as motor drivers.
Master board	In the case of a system composed of multiple micro-drivers, sensors, and a computer to communicate with, a master board is one board used to centralize the data provided by those elements. By making sensors and micro-drivers act as slaves, the master board becomes a computer (making the master board act like a broadcaster). Common communication protocols between the master board and its slaves are i2C and *SPI*. For communication with the computer, the most common ones are Ethernet or WiFi.
Computer	Either attached to the body or fixed in an external position, computers allow for more complex computations, advanced algorithms, and sophisticated data processing. The PC can handle computationally intensive tasks such as image processing, machine learning, or 3D mapping, which might be beyond the capabilities of a microcontroller alone.

Table 2: Average electronics hardware of a quadruped robot.

2.2 Initial hardware's state

As said before, this chapter wants to describe the initial state of the hardware's mechanics and electronics of SOLO 12. The description given in Tables 3, and 4 is based on the latest commercial version of SOLO 12. Like this, any information about the hardware components of the robot can be obtained on the GitHub site created by ODRI [ODR23e]. Compared to Tables 1 and 2, in order to be more specific, the Tables inside this subsection also include the number of items present in the quadruped.

Besides the tables below, for the following chapter, it is essential to know the dimensions of the quadruped. The initial dimensions of SOLO are shown in Figure A.1.1, the general dimensions are 45x30x6 cm (length, width, and height), and, on the other side, it comes in handy to know the height of SOLO with its legs totally extended and half stretched. For these two configurations, the heights are 34 and 24 cm, respectively.

Item	Basic description	Number of items
Body/ chassis	SOLO uses its chassis as a place to assemble the electronic parts. As said before, it also helps to protect sensible parts and order wires. It is manufactured through the 3D printing of parts made by Acrylonitrile Butadiene Styrene (ABS) through Fused Deposition Modeling (FDM).	1
Legs	In the case of SOLO, legs are the extremities that allow its movement and, thus the hardware part that allows any desired action. The material, and the way of manufacturing, is the same as the body.	4
Motors	The motors used in this robot are BLDC motors of 300 kV, 24 magnets, 12 pole pairs, and 18 slots. Their configuration is pf 3 phase WYE/ Star Configuration.	12
Gears and Driving belts	SOLO gets a total reduction of 9:1 through two driving belts per motor	24
Power supply	The power supply expected for SOLO is expected to be a fixed station that, by the power supply, works at 22.2 V (24 V at most) and direct current (DC).	-
Sensors[4]	The robot has an IMU, of the type Lord Microstrain 3DM-CX5-25 [LOR23]. The other sensor is an optical encoder of the AEDT-9819-Z00 type [ODR23a]	1 IMU, 12 optical encoders.

Table 3: Initial mechanical hardware of SOLO 12.

Item	Basic description	Number of items
Microdrivers	microdrives installed in SOLO 12 are designed for controlling two brushless motors and perform torque control at 10 kHz for each of them. The communication between them (the micro drivers) and the master board is via SPI at 1 kHz. These are crafted by the team at ODRI, inspired by the TI evaluation Boards [ODR23f]	6 (one each of two motors)
Master board	The master board manufactured by the ODRI team is programmed by the ESP-IDF toolchain. [Esp23]. It is able to connect via WiFi or Ethernet. Its General Purpose Input/Output (GPIO) has four free. Two of them connect to the IMU.	1
Computer	SOLO does not have an onboard pc, per se. But the master board contains an Ethernet connection to which an offboard computer can be connected.	-

Table 4: Initial electronic hardware of SOLO 12.

2.3 Requirements

As can be seen in both Tables 3 and 4, if the robot is to be autonomous, as a minimum, it must have a battery or a wireless power supply.

In the same way, and as described in the last section of Table 2, the desired level of dynamics requires complex algorithms that cannot be achieved with the present electronics. For this reason, a powerful onboard computer must be installed.

Finally, to reinforce the dynamics of exploration functions or, in turn, to increase the SOLO's 12 functionalities, a sensor capable of obtaining information from its environment must be installed.

Regarding the location of the elements for their subsequent assembly, the decisions taken are based on the optimization between the functionalities of the elements and the minimization of the new constraints they add (this point is discussed in subsection 2.4 *Mechanical design and assembly*).

That being said, in the next subsections, there is an explanation of the specifications and dimensioning of each feature required. The decisions are made through a strict analysis that combines the collection of data with the professional know-how of the HIDRO team.

Batteries

If one wants to install a portable battery equivalent to the current power supply system, it is necessary to know what needs, or requirements, are expected of it.

The number of batteries for robotics nowadays is pretty large, but when it comes to lightweight batteries, Lythium Polymer (LiPo) has become the most famous [Eng23][Dyn23]. This precise feature is, by far, the most valuable for the aim of this project. This, along with the fact that the lab already had experience with them, is the main reason for the choice.

Once the composition of the cells is decided, it is necessary to take into account the characteristics that describe any battery and, in particular, the LiPo kind. These can be arranged in the following list[5]:

- Voltage required

  For the voltage required, as it has been mentioned in Table 3, a 22.2 V is expected. So, in order to use a portable battery, it is either a matter of using one 22.2 V battery or a set of batteries that are equivalent to the voltage required.

  For a single 22.2 V battery, the most available ones on the market are either too big to share space with a commercial onboard PC [Gen23], or too heavy for a light-weight robot [dro23a]. That leaves no option but to use two or more batteries in order to reach the desired values.

[5]For this estimation, and some other tips on manufacturing mobile robots, the project *Articulated Robotics* is super helpful [New23].

It must be understood that two batteries connected in series achieve a total voltage equal to the sum of each other, leaving the capacity of the total equal to the smallest of the individual values. And, on the other hand, two batteries connected in parallel increase their capacity as the sum of their individual values, leaving the total voltage undefined as the individual voltage value can fluctuate during usage.

Cells inside a commercial battery work with the same logic explained before. That is why any already manufactured battery is expected to show the number of cells in series (S) and the number of cells in parallel (P). In the example below (Fig. 2), this 22,2 V battery is composed of 6 cells in series (6S) of 3.7 V each and 1 in parallel (1P), which set the capacity at 3000 mAh.

Figure 2: Example of a LiPo battery.

As the quadruped needs 22.2 V, the simplest way to combine a set of batteries is to connect one pair of 11.1 V batteries in series. By doing so, the voltage required is achieved, but the capacity would depend on the individual values of each battery. If, for instance, both of them share the same capacity, the total would be their individual value. It is, then, necessary to know the capacity required for the quadruped.

- Capacity

  The amount of current that a battery is able to feed in a given space of time is known as its capacity. As with the voltage, in order to choose a portable power supply, the required capacity must be known.

  Following the definition given above, in order to specify the capacity, the total consumption of the robot and the time it is required to work must be known.

  To know the consumption expected, the following experiment is proposed:

  1. Through a controllable power supply, set a constant voltage of 22.2 V.
  2. Feed the robot with voltage and, at the same time, use an open-loop control to make it stay steady.
  3. During a long enough period of time, for example, 10 s, introduce little perturbations to make the robot recover its position.
  4. Meanwhile, collect the current consumed by the robot with respect to the power source.

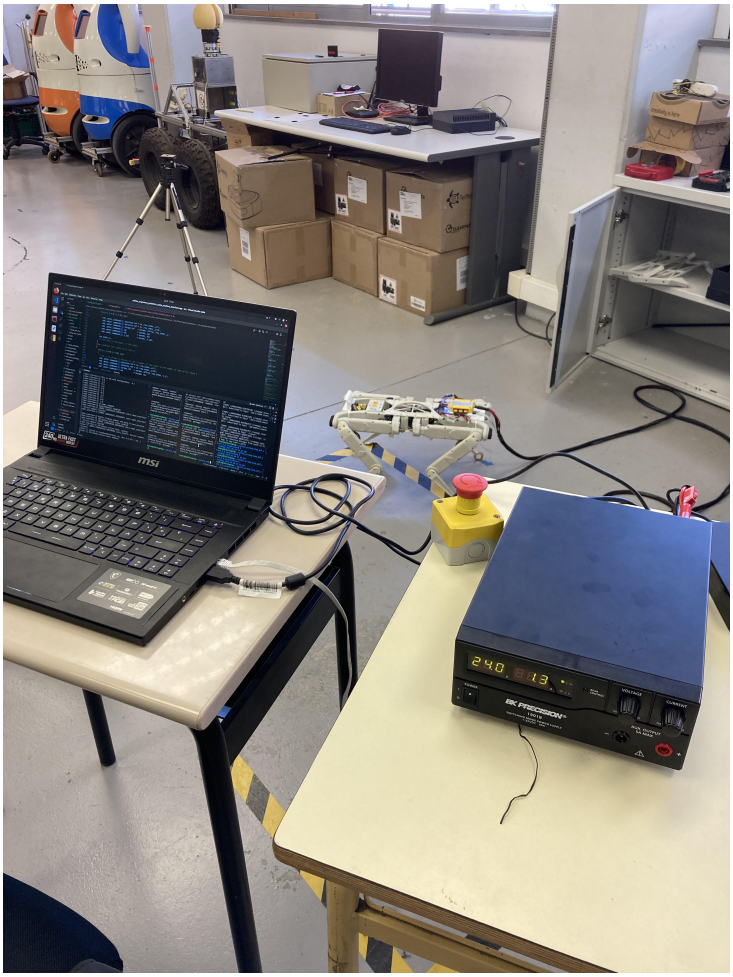


Figure 3: Current consumption experiment set-up.

It has already been said that SOLO 12 is powered by a power supply. As it can be seen in Fig.3, the user can either set the voltage to a desired value or, on the other hand, set the current. As the future batteries will be changed for charged ones once they reach a limit voltage, the voltage is set constant during the experiment.

To give a realistic consumption scenario, at the same time that the script to control the power supply is running, a script that keeps the robot steady is also running.[6]

[6]This control script is developed by the HIDRO team. This position is an open loop control that requires three values of Kff, Kd, and Kp, in this case, those are $Kff = 1$, $Kd = 0.1$, and $Kp = 3$. A simplified scheme of the control strategy can be seen in annex A.2 *Control scheme*.

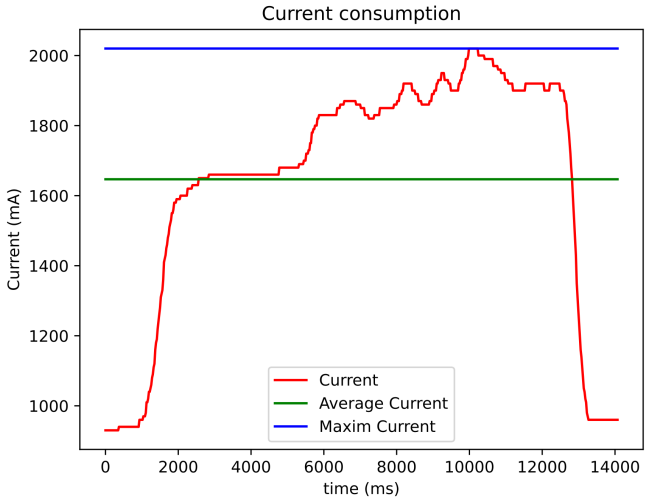


Figure 4: Current consumption during experiment.

The results of the experiment are shown in Fig. 4. The current, when the power supply is switched to *ON*, stays at 700 mA. Once the control script is running, this current goes up to 1660 mA. At 6 s, some perturbations are introduced into SOLO. This new demand makes the current oscillate around 1900 mA. Finally, the robot is disconnected, and the current goes back to 700 mA.

In order to sum up the information, the average current value and the maximum value reached are computed. The average consumption during the experiment is 1.65 A (1646.78 mA), and the peak value is 2.02 A (2020.00 mA).

To consider the average consumption as an accurate representation of the value expected is not proper because, with the new features, more consumption is going to be required. That is why, to calculate the average value, the consumption of the camera and the intensity demands of the on-board PC must be summed.

Firstly, for the camera power supply, a value of 0.7 A (700 mA) is expected (Fig. A.3.1 in the *Annex*).

Secondly, for the onboard PC, taking into account the experience of the HIDRO team with other projects, a value of, at most, 1.5 A[7] is expected.

Finally, on the one hand, the total amount of current expected when the system is working is around 3846.78 mA.

[7]The consumption of any onboard PC depends on the task asked to be performed. It is also explained in the chapter Onboard Computer, but for compiling tasks for model predictive control, this value is correct.

On the other hand, for the dimension of the capacity of the batteries, it is also needed to know the amount of time the robot is required to work.

In this case, it is necessary to strike a balance between the time required for the desired experiments and the specifications of the motors.

The first condition is that, as has been said, high-dynamic-range operations do not require a lot of time. In fact, even for an experiment where it is intended to practice exploring routines, the time will not exceed 10 minutes. That's why, as an average value, it is considered that 10 minutes is enough.

The second one is related to the motor's specifications. For the motors of SOLO, as advised by the team at ODRI, the heating is expected to be high for complex maneuvers, such as backflips, that are done repeatedly in a short time. Knowing that information, the 10-minute estimation is expected to even be over-dimensioned.

- Discharge rate (C - rate)

  The last characteristic to consider is the discharge rate, or C-rate.

  The discharging rate indicates the amount of current that any battery can feed to the system in relation to its capacity. This means that, for a battery with a 2 C-rate and a capacity of 1000 mAh, the maximal discharging current is 2000 mA.

  The C-rate specification for quadrupeds must be high in order to have a robust energy source when situations like hits, load changes, or stabilization efforts appear. So, in the particular case of SOLO, among the candidates, the higher the rate, the better the option.

After the list of specifications for the battery has been described, it is useful to sum up all of them in the next table:

Feature	Desired Value
Total Voltage	Between 22.2 and 24.0 V
Required time	Experiments no longer than 10 min
Required capacity	500 mAh[8]
Discharge rate	70 - 90 C

Table 5: Requirements for the battery as a power source.

The values of Table 5 are easy to find in a lot of commercial companies. Most of them sell batteries for drones or radio-controlled vehicles (RC cars, commonly).

From a search on the web, these five candidates are choose (Table 6):

Commercial name	Dimension (mm)	Weigh (g)	Price (€)
Batería LiPo Gens Ace 3s 11.1V 800mAh 45C	58x30x23	70.0	13.69
Batería LiPo TATTU 3s 11.1v 850mAh 75C (XT30)	59x30x24	85.0	14.5
Batería LiPo TATTU R-Line 3s 11.1V 750mAh 95C	58x30x19	62.0	15.5
Batería LiPo TATTU 3s 11.1v 850mAh 75C	59x30x24	85.0	13.99
Gens ACE 3s 11.1V 1200mAh 25/50C - Airsoft	126x19x16	87.0	18.9

Table 6: Battery candidate comparasion.

As can be seen in Table 6, among the five candidates, there is not much difference between the dimensions, the weight, or the price. Therefore, it is decided that, of the three characteristic values, weight is the most significant one. Taking that into account, the final and selected candidate is the *LiPo TATTU R-Line 3s 11.1V 750 mAh 95C*.

Power Distribution Board (PDB)

Any system with one main source of power supply and various hardware elements that require such power must incorporate a Power Distributor (PD).

With the more extensive use of Printed Circuit Board (PCB)s, it is common that PDs are implemented inside one, and, for that, they get called Power Distribution Board (PDB). PDBs for mobile robotics usually include an input port, a desired number of output ports, a DC/ DC convertor, and a battery monitor [Moh+17].

For the case of SOLO 12, all initial systems on it can work between 22.2 and 24.0 V, but, as it will be seen in the next subsection, 2.3 *Onboard computer*, it is needed to establish a 15 V channel.

The design of the PDB is taken care of by the team at HIDRO, and the DC/DC, which is assembled separately, is already configured and ready. Therefore, the requirements and selection of this extra feature are beyond the scope of this thesis.

Onboard computer

The choice of the onboard computer requires an in-depth knowledge of specific electronics and communications that is beyond the scope of this work. However, there are some basic requirements that should be mentioned, as they are the ones that have built the reasoning necessary to choose a candidate.

First, as with the power system, the mass of the computer must be considered a critical condition. It should be remembered, once again, that the purpose is to perform experiments that require agile dynamics. Therefore, among all possible candidates, the so-called mini PC computers fit optimally.

Mini PCs offer the same amount of features as any laptop or desktop computer, with the convenience of being smaller than their competitors. As with any other computer, when it comes to choosing one, it is necessary to know which processor is wanted, how much RAM is needed, if a special graphical board will be necessary, and which and how many ports are going to be used.

Among the requirements listed, the focus needs to be on the processor. The main reason is the need to execute complex control techniques such as model predictive control (MPC). Being able to satisfy the computational demands of an MPC algorithm requires a CPU with a sufficient number of cores and a high clock speed.

In addition, one important requirement is the number of ports available. That is due to the fact that any extra feature, like a sensor, will probably require a power supply from the onboard PC.

For these requirements, a known model of Intel, the NUC NUC7i7BNH, fits conveniently. This onboard PC is part of the equipment used by HIDRO, so their experience with it is also valuable. Table 7 sums up some of NUC's values.

Specifications	Intel® NUC NUC7i7BNH
Processor	2 cores with 4 subprocesses each
RAM (GB)	4
Storage (GB)	32
Graphical card	Intel Iris Graphics 650
Ports	1 ThunderBolt and 6 USB

Table 7: NUC basic feature description.

Sensors

The sensory system detailed in Table 3 is useful either for control techniques or for estimation purposes. Both the encoders and the IMU provides a description of the system using lightweight hardware.

At the same time, although not mentioned in Table 3, all versions of SOLO offer a chassis designed with the possibility of incorporating some special markers that record its position and orientation. This is done through motion-capture cameras that are usually displayed all over the area where the robot is expected to be. In this case, these are referred to as Opti Track (OT).

At the same time, in order to expand the robot's capacities beyond this work, a depth camera and a Light Detection and Ranging (LIDAR) are proposed to be attached to the robot.

For the camera, an *Intel RealSense Depth Camera D435i* is chosen. This model offers lightweight hardware while also offering depth vision for object recognition and robotic navigation.

The choice of the LIDAR, and its design, are beyond the scope of this thesis. However, its future addition is considered in subsection 2.4 *Mechanical design and assembly.*

Before finishing this section, it may be interesting to emphasize that, in a multiple-sensor robot like this, the fusion of data is crucial. If a robot is desired to be autonomous, knowledge of variables, such as its position or velocity, is required to be at the highest frequency and with the most precision possible. About this matter, the second part of this thesis, 3 *Software*, explains how the IMU and OT could be fused.

Summary of the new features

A summary of the components chosen for the new hardware autonomy load can be found in Table 8. It contains the commercial name of the product, a link to the web-page of the seller, and the number of items requested.

Item	Basic description	Number of items
Batteries	*Tattu R-Line* 750mAh 11.1V 95C 3S1P [dro23b]	2
Onboard PC	*Kit Intel® NUC NUC7i7BNH* [Int23a]	1
Camera	*Intel RealSense Depth Camera D435i* [Int23b]	1

Table 8: New features for the autonomy payload.

2.4 Mechanical design and assembly

Following the idea of a robot that is all made of 3D printed parts and, taking into account the fact that the laboratory already has one 3D printer, the design of all the assembly parts follows the same procedure.

In this case, the parts are crafted using ABS, a commonly used plastic for 3D printing. The printing technology used is FDM, with a 3D printer model *Dimensión 1200es*. It must be known that most of the already assembled parts of SOLO 12 are printed with FDM, and for more accurate parts, such as inner gears, Multi Jet Printing (MJP) with resin is used. This second technology is proven to be more precise [Tac+20], but for the purpose of printing big parts with not much detail, FDM with ABS is enough.

New features distribution

The first thing to keep in mind when it comes to adding new elements is how they could affect the performance of the robot. These considerations require that, for each of them, it be decided which is the best location and how to assemble them on the chassis.

The distribution of the elements is agreed upon by all the members of the HIDRO team, and the conclusions are the following:

(a) 3D printer used for the modelling, *Dimensión 1200es*.

(b) *STL* configuration for the 3D printing.

Figure 5: 3D printing set-up.

- Sensors should be on the back (upper side) of the robot. When it comes to sensors such as cameras, which are intended to help the moving capacity, they must be placed either in the front or the rear and with some positive pitch inclination. On the other hand, sensors for mapping, such as LIDARs, must be on the surface for their placement, and the higher the better.

- NUC and batteries should be located either all on one side, which, after the first point, is no other than the bottom, or distributed without occupying space for sensors.

The brief of the list above is illustrated in Fig.6.

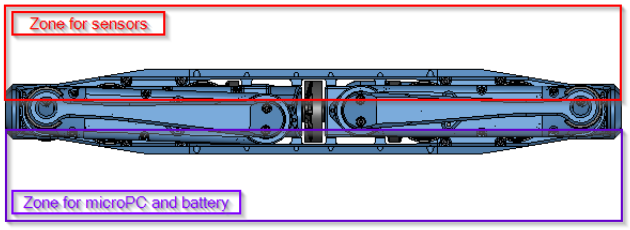


Figure 6: Distribution of the new hardware.

For the design of the parts, the author takes into account these two aspects:

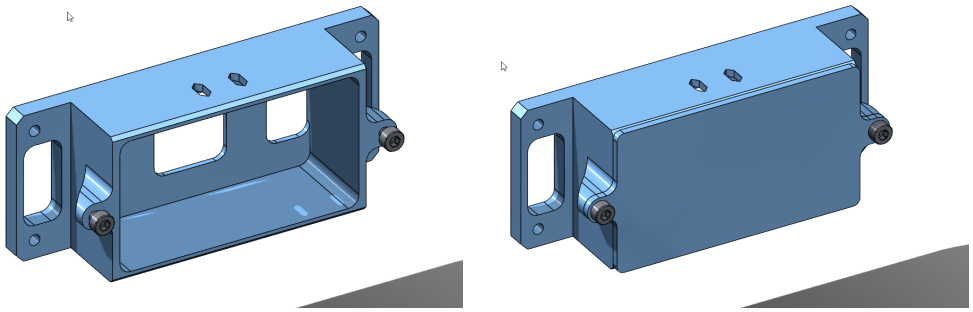
- As little material as possible must be used in order to waste less plastic and energy. If there is a change in introducing holes or minimizing thickness, it should be considered.
- The fixing system must tighten all the new features up to the body, and, at the same time, it must be accessible and easy. Bolts to be used, as well as nuts, have to be standard.

Batteries design

The place for the batteries is decided to be located at the front and rear of the plane shared by both zones in Fig. 6. These locations are intended to keep the robot as symmetrical as possible.

The design proposed in this project is, at the same time, one that protects the object and leaves space for a secure connection[9].

Holes for attaching to the main body are blind M3, merely marked through 3D printing and manually drilled afterward. For the ones that attach the chase, an insert system is used. The drilling, tapping, and installation of the insert are done using a *V-Coil kit* [Vol23].



(a) New design of battery chase with no cover. (b) New design of battery chase with cover.

Figure 7: Complete mechanical design for the battery.

A graphical view of the batter's chase's general dimensions can be found in Fig. A.4.1. These are 100x40x27 mm, and the weight of this new piece is 28.49g (17.75g for the battery placement and 10.74g for the battery chase). As the battery weighs 62.0g, the total amount for the whole set is 90.49g. The total amount of ABS material used for the whole printing is 31.62g and of support High Impact Polystyrene (HIPS) is 31.61g, considering a density of 1.07$\frac{g}{cm^3}$. The total printing time needed is 4 hours and 45 minutes. Lastly, for the fixing, an extra pair of M2.5x9 bolts are used. The bolts for fixing the chassis are already available as the old part is removed.

[9]When this design was being considered, the ODRI team had developed an autonomous version of SOLO. Their design of the placement was agreed to be the best solution, so it was borrowed for this thesis [ODR23b]

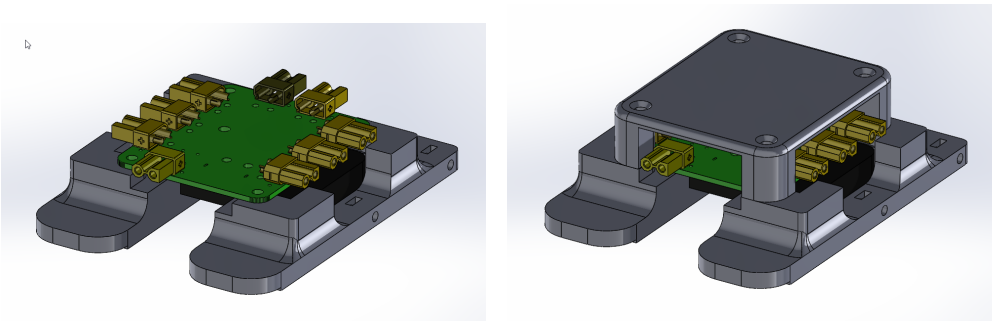
PDB design

The design of the holder part keeps the PDB (green board, Fig. 8) fixed at the same time that it also holds the DC/DC converter (black part, Fig. 8).

The location of the new feature is close to the rear part of the robot because, if there is a desire to not use the batteries, and, instead, use a fixed power supply, it is not necessary to cross the wires through the robot. This disposal somehow breaks the proposition of leaving the upper surface only for sensors, but, as the original power distribution of SOLO 8 is located at that place, it is considered better to not relocate it.

The design of the base requires the placement of four bolts with inserted female nuts. These four holes locate the platform on the chassis. The four additional items are fixing the chase as well as the PD board.

A graphical view of the general dimension of the power distribution chase is available in the annex, Fig. A.4.2. These values are 80x106x34.80 mm, and the weight of the new base piece is 48.67g and that of its chase is 15.81g. The total weight, including the new power distribution and the DC/DC, is 123.48g. The amount of ABS required is 64.05g and 18.77g of HIPS. The total amount of time required is 4 hours and 47 minutes. The requested number of bolts is 4 M3x20 and 4 M3x16. The nuts required are 8 units of M3 type.



(a) PDB base. (b) Complete assembly for the PDB.

Figure 8: Complete mechanical design for power distribution board.

Onboard computer design

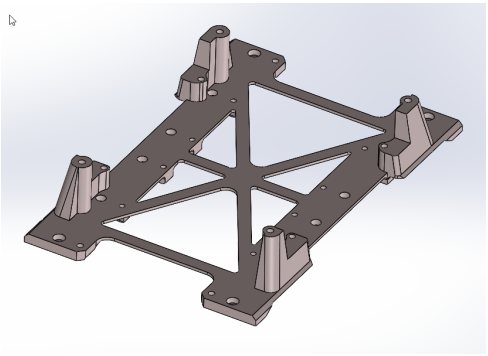
For the NUC's size, it is compulsory to use the bottom base of the robot (a decision already taken in subsection 2.4 *New features*).

The lowest part of the chassis base is being redesigned in order to offer an assembly for the mini PC but, at the same time, keep the structural purposes, as well as the wire protection, of the original design.

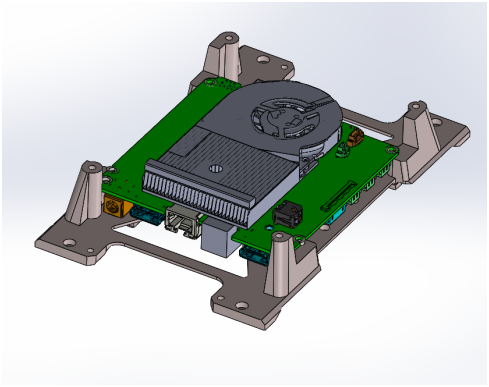
In this case, a more flexible sharing is observed in the interior design of the base, since the ABS is less rigid than the original base material. This behavior, however, does not present excessive stress due to the passing of the wires.

When working with mini PCs, it is relevant to take into account the heating during their operation. The lack of space for air circulation and proper cooling makes it necessary to include vents such as the one seen on Fig. 9 *(b)*. That is why the chase incorporates a hole that helps flow the thermal dissipating power (which, by the way, is approximately 48 W).

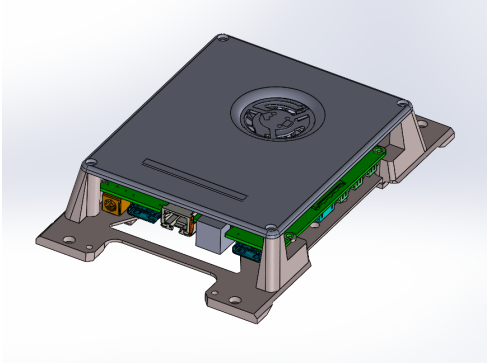
NUC's chase's general dimensions are 180x120x34.60 mm (Fig. A.4.3) and the weight of the new base of the body is 46.60g, and for its protection, 38.12g. The total mass of the new feature, including the mini PC, is 307.72g. The ABS material needed in this crafting is 91.06g and its support is 47.16g. The total printing time is 6 hours and 28 minutes. Apart from the already existing bolts, 4 M2.5x5 and 4 M3x12 new ones, as well as 4 M3 nuts, are required.



(a) New design of bottom part.



(b) NUC on top of the new design.



(c) Complete assembly with cover.

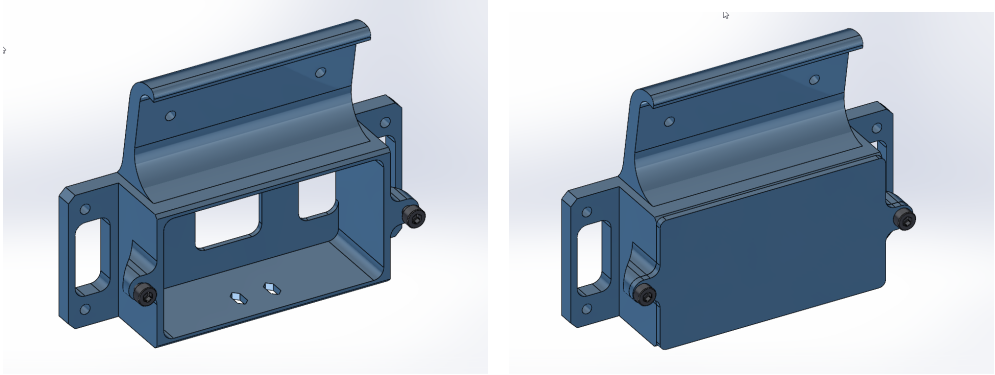
Figure 9: Complete mechanical design for the NUC.

Camera design

As it is clear in Fig. 10 the design of the battery case is adapted to assemble a camera on it. This new label is pitched 10° so the camera has a clearer field of vision of the immediate floor. Two blind M2.5 holes are drilled to mount the camera on it, as the frame is intended to be fixed with respect to the chassis.

The general dimensions of the camera's chase are 100x73.66x27 mm, with an extra detail on the 80° inclination of the camera's placement (Fig. A.4.4).

The weight of the battery place and the support for the camera is 32.22g and the chase, as in the battery design, is 10.74g. The total weight of the holder and the camera is 118.07g. Then, adding the weight of the second battery, it was 242.07g. The total ABS required is 45.77g and 45.64g. The time required is 6 hours.



(a) New design of camera chase with no cover. (b) New design of camera chase with cover.

Figure 10: Complete mechanical design for the camera.

2.5 Final result

The final assembly for the three new elements is indicated in Fig.11.

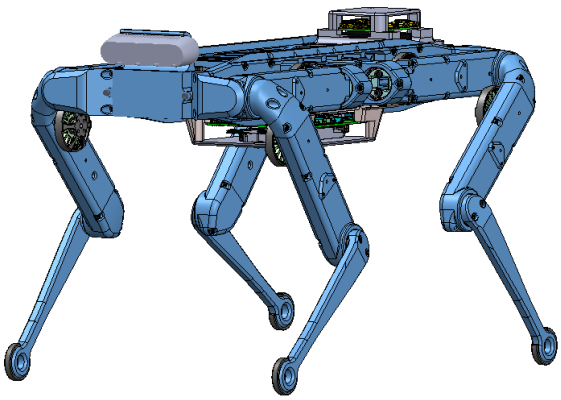


Figure 11: Final view of the components assembled to SOLO 12.

The new autonomy payload has changed the dimensions and mass of the original robot. These values are relevant for the dynamics performances and have to be computed for simulation purposes.

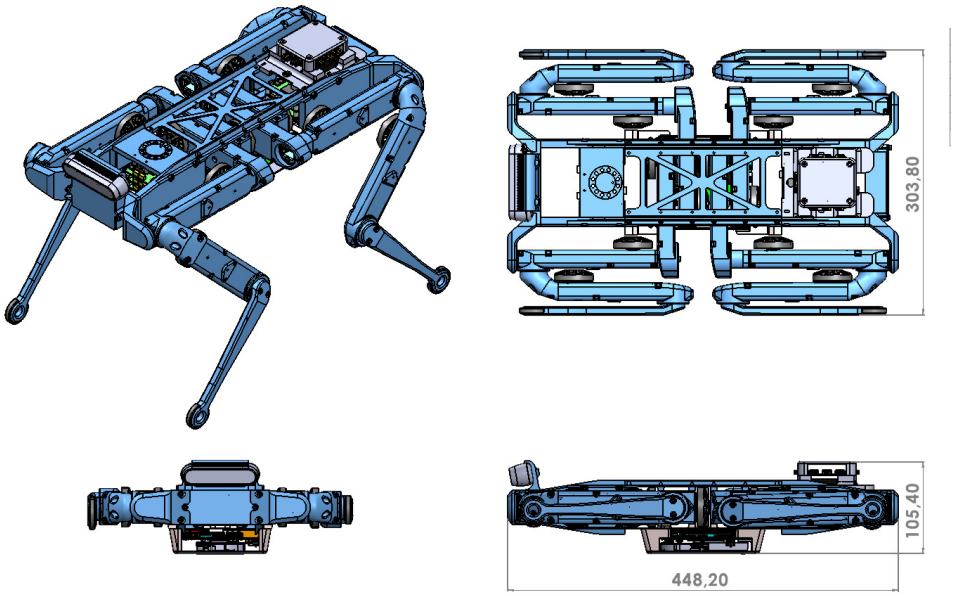


Figure 12: New dimensions of SOLO 12.

After the add-ups, the new dimensions, which could be compared with Fig. A.1.1, are shown in Fig. 12. For width and depth, the robot is still the same, with values of 30.8 mm (close to 30 cm) and 448.2 mm (close to 45 cm). When it comes to height, the robot has increased by 45.4 mm. This is due to the power distribution and the onboard PC. The last distance that should be considered is the new minimum distance between the belly of the robot and the floor. By assembling the NUC, the minimum distance between the bottom part of SOLO and the floor must now not be lower than 28.1 mm.

For mass, the total mass of SOLO 12 after the upgrades is 3.4 kg (763.76 g more than the initial value). The inertia on the main axes of the robot has also changed, and thus, it must be recalculated. For this, it is considered that the three new elements are equivalent to boxes, and, so, the formula of the inertia with respect to the main axes can be obtained from eq. 1, 2 and 3 for x, y, and z, respectively. The values for the NUC and the batteries are those shown, and because the priority of this calculation is to get agile dynamics where the camera is not required, Intel Real Sense is not considered.

$$I_{xx} = \frac{1}{12} m(b^2 + h^2) \tag{1}$$

$$I_{yy} = \frac{1}{12} m(a^2 + h^2) \tag{2}$$

$$I_{zz} = \frac{1}{12} m(a^2 + b^2) \tag{3}$$

where:

- I_{xx} is the moment of inertia about the x-axis
- I_{yy} is the moment of inertia about the y-axis

- I_{zz} is the moment of inertia about the z-axis
- m is the mass of the box
- a is the length of the box along the x-axis
- b is the length of the box along the y-axis
- h is the height of the box

Item[10]	Ixx (kgm^2)	Iyy (kgm^2)	Izz (kgm^2)
Batteries	0.000031167	0.000011924	0.000029957
Onboard PC	0.000344000	0.000482000	0.000789000
Power distribution	0.000077000	0.000056000	0.000116000

Table 9: Computed inertias for the new items.

The results shown in Table 9 are the local inertia values of each feature. In order to implement these new inertias in the main body, they are uploaded to the Unified Robot Description Format (URDF) of the robot. This type of file is intended to describe the main dynamic properties of a robot (mass, inertia, and geometry) through the combination of links and joints that compose it. By adding the new features as links connected to the main body by rigid joints, the computation of the main inertia values is taken care of by the file. The only relevant thing, in order to get the proper inertia, is to define the geometrical center of the new links with respect to the global frame of the robot. This is done by measuring it both on the Solidworks assembly and the real assembled quadruped.

With the new URDF, the mechanical assembly of the robot is finished and, with it, the hardware part of this thesis.

To sum up, the materials required for the attachment of the new 3D parts, Table 10 describes the BOM (Bill of materials).

Item	Required bolts	Nuts	Model
Onboard PC	2 M2,5x9 and 4 M3x30	2 M2,5x2 inserts	*Kit Intel® NUC NUC7i7BNH*
Battery chase	2 M2,5x9	2 M2,5x2	2 *Tattu R-Line* 750mAh 11.1V 95C 3S1P
Camera and battery chase	2 M2,5x9 and 2 M30x9	2 M2,5x2 inserts	*Intel RealSense Depth Camera D435i*
Power Distribution	4 M3x20 and 4 M3x16	8 M3 nuts	Power distribution Board

Table 10: BOM (Bill of materials) requiered for the new features assembly.

3 Software

3.1 Theoretical approach

As introduced at the end of the first part of this paper, the relevance of defining a suitable state and its estimator is critical to establishing closed-loop control in any robot.

There may be different definitions of state depending on the robot. But, usually, the variables picked up to do so are related to its application or the way it develops inside its work's space.

The webpage [Bok23] exemplifies this process with four different robots to which a different state is attributed. For a vacuum cleaner robot, the x, y coordinates of its position $\mathbf{p}$ might suffice. On the other hand, for an unmanned aerial vehicle (UAV), its position $\mathbf{p}$ may not be enough; thus, an orientation $\mathbf{R}$ may be required.

For these two last examples, and in general, the representation of the state is done using the letter $\mathbf{x}$, followed by the variables desired. For instance, the state of the vacuum machine would be $\mathbf{x} = <\mathbf{p}>$, where $\mathbf{p}=(x,y)$ (a 2-D vector), and, for the UAV, the state would be $\mathbf{x} = <\mathbf{p},\mathbf{R}>$, where $\mathbf{p}=(x,y,z)$ (a 2-D vector), and $\mathbf{R}=(\phi, \theta, \psi)$ (a 3-D vector[11]).

Defining the variables that represent the robot in space is not enough, as it is not a single dot in space. For that, the frame to which this state is attached to, must also be thought out and established.

A state can be composed of variables that refer to a frame attached to a body (such as its gravity center, torso, etc.), and by others that are located in the joints between links (for example, in the robotic legs of a quadruped or a robotic arm with multiple links).

Finally, once the state and its reference frames have been defined, the appropriate sensors for the quantification of its variables must be chosen.

By means of these sensors, the variables can be updated, either because these sensors provide the information in the form of such variable or because, with these values, the state variables can be estimated. For example, as introduced in the previous section, one of the most common sensors in mobile robotics is the IMU. IMUs can provide the linear acceleration and the angular velocity, and if these variables define part of the state, their values can be incorporated as the system works. On the other hand, if, in addition to, or instead of, the linear accelerations, velocity, and position are expected, by integrating the acceleration over time, these values can be estimated.

This last paragraph explains that, by using the appropriate sensors, the system variables can be updated. However, the process of updating the state variables is not simple.

[11] Please, take this as an example, because Euler angles are not a proper way of representing an orientation in 3D space. In subsection 3.1 *Lie Extended Kalman Filter (LEKF)*, a proper explanation is done.

Firstly, there may be situations where variables are defined in a state for which no sensor can give direct information. This case, for example, was the situation where the NASA team, wanting to establish a monitoring system of the temperature at which the fuel burned for its Apollo 11 spacecraft, had to estimate this value given the clear impossibility of setting a sensor in the spacecraft's reactor [Tra18]. This estimation, because of the existence of noise, is not sufficiently robust, so it is necessary that this value is duly corrected with some other sensor incorporated into the system.

In another situation, in multi-sensor systems, there may be different working frequencies for each sensor. Likewise, for each sensor, there may be more accurate sensors than others. This situation requires unifying the information from each sensor and knowing the veracity of each piece of data.

Kalman Filter (KF)

To solve both the first and second situations exposed in the last paragraph, in state estimation, the Kalman Filter (KF) is one of the most widespread tools. Its applicability is optimal when the description of the problem is a linear system with Gaussian modeling variables and measurements.

- Problem statement.

  The KF is defining the problem with eq. 4.

  $$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t\text{-}1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{4}$$

  In this statement, the current state $\mathbf{x}$ on a time t is defined as the relation between the last state, $\mathbf{x}_{t-1}$, and the state transition matrix $\mathbf{F}$ (that applies the effect of the last state to the current one, i.e., the velocity on *t-1* affects the position on t). It also accounts for the control input $\mathbf{u}$ at the time t, multiplied by its control input matrix $\mathbf{B}$ (that applies the control to the appropriate variable of the state). Finally, the noise inside the process is considered in $\mathbf{w}$. This noise is expected to have a Gaussian distribution with a zero multivariate mean. Its covariance matrix is usually expressed as $\mathbf{Q}_t$.

  One must also take into account the measurements of the system, which are defined in eq. 5.

  $$\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t \tag{5}$$

  Inside the equation, $\mathbf{y}_t$ is the measurement on time t, and $\mathbf{H}_t$ is the transfer matrix that shows which variables of the state are being measured. Finally, as it is done in the state evolution, eq. 4, a noise $\mathbf{v}_t$ is considered inside the measurements. Also, parallel to the state evolution, the measurement noise is defined as a zero-mean Gaussian with a covariance named $\mathbf{R}_t$.

- Prediction step.

  The algorithm, or iterative process, of the KF, begins with the prediction step. This step establishes that the estimated state $\hat{\mathbf{x}}_t$ will be equivalent to the model defined in eq. 4, without taking into account the system error. In turn, in order to propagate the noise in the system definition, the new covariance matrix of the system $\mathbf{P}_t$ must be calculated. Both the prediction and the new covariance matrix are expressed in eq. 6 and eq. 7, respectively.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t \tag{6}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{7}$$

- Correction step.

  Once the estimated state is defined, $\hat{\mathbf{x}}_{t|t-1}$, as well as its covariance matrix, $\mathbf{P}_{t|t-1}$, the KF sets up a second step called correction. This step evaluates the differences between the expected measurements $\mathbf{e}_t$ and the real one $\mathbf{y}_t$ (which is known, eq.8). The covariance of this difference must also be computed, eq. 10, because, with it, and the covariance got in eq. 7, one can compute the Kalman gain $\mathbf{K}$ (eq. 11). This new variable sets how much correction is needed to be applied over the estimation done, eq. 12, and computes a new covariance matrix for the state (eq. 13).

$$\mathbf{e}_t = \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1} \tag{8}$$

$$\mathbf{z}_t = \mathbf{y}_t - \mathbf{e}_t \tag{9}$$

$$\mathbf{Z}_t = \mathbf{R}_t + \mathbf{H}\mathbf{P}_t\mathbf{H}^T \tag{10}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{Z}_t^{-1} \tag{11}$$

$$\mathbf{x}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\mathbf{z}_t \tag{12}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t^T\mathbf{Z}_t\mathbf{K}_t \tag{13}$$

Extended Kalman Filter (EKF)

The KF explanation made before shows its usage on a linear problem, unfortunately, for fields like robotics, variables inside the state do not need to be linear. This scenario leaves no option other than uploading the KF to an extended version that also comprehends non-linearity. By doing so, nevertheless, the algorithm is no more optimal, but, it's proven to offer good results.

The briefing of this extension is that, for the steps where covariance is computed, the system is now linearized by its Jacobians around the state values at that time.

- Nonlinear problem statement.

  In this case, neither the state evolution nor the system measurements can, no longer, be defined using matrices. Instead, they can be defined as nonlinear functions like in eq. 14 and eq. 15.

$$\mathbf{x}_t = f(\mathbf{x}_{\text{t-1}}, \mathbf{u}_t, \mathbf{w}_t) \tag{14}$$

Where $\mathbf{x}_{t-1}$ stand for the last known state, $\mathbf{u}_t$ is the control signal at the current time, and, like in KF, $\mathbf{w}_t$ is the noise in the process. Then, f is the non-linear function that represents the evolution of the state.

$$\mathbf{y}_t = h(\mathbf{x}_t) + \mathbf{v}_t \tag{15}$$

Where $\mathbf{y}_t$ stand for the measurement and, $\mathbf{v}_t$ its noise. Then, h is the non-linear function that represents the measurement of the state.

- Prediction step.

  The prediction step now considers the estimation of the state as a function with no noise, eq. 16, and its covariance is defined like in eq. 17.

$$\hat{\mathbf{x}}_{t|t-1} = f(\mathbf{x}_{t-1|t-1}, \mathbf{u}_t, 0) \tag{16}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_x \mathbf{P}_{t-1|t-1} \mathbf{F}_x^T + \mathbf{F}_w \mathbf{Q}_t \mathbf{F}_w^T \tag{17}$$

  where, $\mathbf{F}_x$ and $\mathbf{F}_\omega$ are the Jacobians of the state with respect to the state and the estimation noise with respect to the state, respectively.

- Correction step.

  The extended correction step also estimates the values as a non-linear function with no noise, eq. 18, and defines the correction like in the linear case (eq. 19). The rest of the covariance matrices follow the same format as with the KF's case.

$$\mathbf{e}_t = h(\hat{\mathbf{x}}_{t|t-1}) \tag{18}$$

$$\mathbf{z}_t = \mathbf{y}_t - \mathbf{e}_t \tag{19}$$

$$\mathbf{Z}_t = \mathbf{R}_t + \mathbf{H}\mathbf{P}_t\mathbf{H}^T \tag{20}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{Z}_t^{-1} \tag{21}$$

$$\mathbf{x}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\mathbf{z}_t \tag{22}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t^T\mathbf{Z}_t\mathbf{K}_t \tag{23}$$

Lie Extended Kalman Filter (LEKF)

In the description of the state, as shown in subsection 3.2 *State definition*, it is usual to express the orientation of the robot in space.

3D orientations are usually expressed in three ways: Euler angles, Rotation matrices, and quaternions. The first is the most understandable way of expressing an orientation, as it is simply the combination of three angles over the three main axes. The problem in the field of robotics, and in many other disciplines like graphic design or video games, is the appearance of singularities, or "gimbal lock". When two of the axes of rotation get aligned and a degree of freedom is lost, the Euler angle is not a good way to express orientations or transformations between two different orientations. [Sut13]

To solve the problem of expressing an orientation with three degrees of freedom, it is common to either use rotational matrices or quaternions. The problem with rotational matrices, from a computational point of view, is that they can be heavy objects. Adding more complexity, then, to an algorithm such as the EKF which is not optimal, is not a clever way to go.

On the other hand, quaternions only use 4 values (versus the 9th of the rotational matrix) [Sut18], and if they are treated as Lie Groups, their usage gets even simpler. In this thesis, the simplicity of working with Lie Groups comes down to the usage of a library called *Manif*, or its Python version *Manifpy* [JA18]. *Manif* is a library based on the Lie theory, extremely useful for state estimators. It is developed and explained in the paper *Micro Lie Theory for State Estimators in Robotics* [SDA21].

The understanding of Lie Theory is focused on its utility rather than its comprehension, and, in this case, the main important things to understand are the right-operators $\oplus$ and $\ominus$, as well as the definition of the Jacobians inside the Lie Theory.

If one is about to define the new orientation of the robot from the product of an angular velocity ω multiplied by a time difference dt, the Lie operator to do so is the $\oplus$[12]. The $\oplus$ is the operation that allows a Lie group, such as an orientation on the space (defined as SO(3)), to be incremented and, then, return to its curved representation in space, which is called a manifold.

On the other hand, to express an error, understood as a difference, it is proper to use the $\ominus$. In the correction step (eq. 19 in EKF), it is required to compute the difference between two objects which are two orientations. By using the $\ominus$ operator, the Lie object returned is an orientation expressed in the tangent space. expressed as ${}^{\mathcal{X}}\tau$ (if it is right signed).

In order to use *manif*, then, the last steps of the EKF must use the proper symbology. The only equations affected by this switch to LEKF are eq. 19 which now is eq. 24 and eq. 22 that now is eq. 25.

$$\mathbf{z}_t = \mathbf{y}_t \ominus \mathbf{e}_t \tag{24}$$

$$\mathbf{x}_{t|t} = \hat{\mathbf{x}}_{t|t-1} \oplus \mathbf{K}_t \mathbf{z}_t \tag{25}$$

Finally, one can compare the way the Jacobians[13] are defined in vector spaces, eq: 26, and way *Right Jacobians on Lie Groups* are defined, eq. 27.

$$\mathbf{J} = \frac{\partial f(x)}{\partial x} \triangleq \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \in \mathbb{R}^{n \times m} \tag{26}$$

$$\frac{{}^{\mathcal{X}}Df(\mathcal{X})}{D\mathcal{X}} \triangleq \lim_{\tau \to 0} \frac{f(\mathcal{X} \oplus \tau) \ominus f(\mathcal{X})}{\tau} \in \mathbb{R}^{n \times m} \tag{27}$$

[12] It would not be enough to simply specify an operation as either an $\oplus$ or an $\ominus$, as it is also required to know which is the sense of the operation, being right or left. In this case, all operations are considered the right ones.

[13] Both expressions of the Jacobians are extracted from the [SDA21] paper.

3.2 State model and definition

State definition

As mentioned above, the definition of the state depends on which variables best describe the behavior of the robot. This definition, for a robot like that already has sensors, can be guided by the available sensors.

With this second comment in mind, of the variables that the SOLO's IMU measures, it is decided to focus on the linear acceleration measurement ($\mathbf{a}_m$) and the angular velocity measured ($\boldsymbol{\omega}_m$), in order to define the orientation of the robot ($\mathbf{R}$), its velocity ($\mathbf{v}$) and, finally, its position ($\mathbf{p}$). These three variables define what is known as the pose of the robot, and, in terms of state definition, these would be enough. But, to keep track of the deviation of the bias (a concept that is explained in the next subsection), and avoid the random-walk noise, it is common practice to introduce both acceleration bias $\mathbf{a}_b$, and angular velocity's $\boldsymbol{\omega}_b$ as state variables.

Once the last two variables have been added, the state to be estimated is fully defined and can be expressed as a bundle of variables, just like in eq. 28.

$$\mathbf{x} =< \mathbf{R}, \mathbf{v}, \mathbf{p}, \mathbf{a}_b, \boldsymbol{\omega}_b > \tag{28}$$

State model

In order to define the predicted step over a LEKF, a model for the state must be defined. In this case, it is a good way to start by defining properly the variables intended to be used for the integration over time.

The two measurement variables, $\mathbf{a}_m$ and $\boldsymbol{\omega}_m$, can be described including their bias ($\mathbf{a}_b, \boldsymbol{\omega}_b$) and noise ($\mathbf{a}_w, \boldsymbol{\omega}_w$).

The bias is an offset included in any variable that the IMU measures. It depends on temperature, time, and mechanical stress and adds uncertainty to the measurement, changing its mean. When using the acceleration to get the velocity and the position (for example), by integrating, the bias affects the estimation intruding a deviation known as random walk (on the equations, r underscored).

In the same way, noise over a measurement makes the real value differ from the measured one. In this case, the noise is a zero-mean Gaussian, so, as it is usually done, it is referred to as white noise (and so, with a w underscored).

With that being said, the equation that describes ω_m could be defined as the true angular velocity ($\boldsymbol{\omega}$) plus the $\boldsymbol{\omega}_w$ and plus the $\boldsymbol{\omega}_b$ (eq. 30).

Next, to complete the definition of $\mathbf{a}_m$, two more considerations must be made.

First, as the theoretical approach explains, the frame to which the variables are being referred must be taken into account. The IMU measures both the $\mathbf{a}_m$, and $\boldsymbol{\omega}_m$ with respect to its main axis, or frame. As the interest in the acceleration is to integrate it through time and get both the $\mathbf{v}$ and the $\mathbf{p}$, the most useful way to express it is with respect to the world reference frame. To do so, its measurement description must include the transformation between the world frame and the IMU's: $\mathbf{R}_{imu}^{world}$.

Secondly, all IMUs include the gravity acceleration ($\mathbf{g}$) on its measurement, so it must also be considered in order to describe what is inside the $\mathbf{a}_m$footnoteTo that it must be added that really precise IMUs also include the rotation of the Earth as a measure in the $\boldsymbol{\omega}_m$, but this is not the case, and so, it is not considered..

Now, the description for $\mathbf{a}_m$ can be expressed as follows in eq. 29, containing the true acceleration value ($\mathbf{a}$).

$$\mathbf{a}_m = \mathbf{R}_{imu}^{world\ T}(\mathbf{a} - \mathbf{g}) + \mathbf{a}_b + \mathbf{a}_w \tag{29}$$

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + \boldsymbol{\omega}_b + \boldsymbol{\omega}_w \tag{30}$$

Integration over time of the IMU variables must be done with $\mathbf{a}$ and $\boldsymbol{\omega}$, and not with the measures, which contain extra information. So, by isolating both true values, $\mathbf{a}$ in eq. 29 and $\boldsymbol{\omega}$ in eq.30, the following expressions are obtained:

$$\mathbf{a} = \mathbf{R}_{imu}^{world}(\mathbf{a}_m - \mathbf{a}_b - \mathbf{a}_w) + \mathbf{g} \tag{31}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega}_m - \boldsymbol{\omega}_b - \boldsymbol{\omega}_w \tag{32}$$

With the first integration of the linear acceleration $\mathbf{a}$, linear velocity $\mathbf{v}$ is obtained (eq.34). With the second integration of the linear acceleration $\mathbf{a}$, the position $\mathbf{p}$ is also computed (eq.35). With the angular velocity $\boldsymbol{\omega}$, in turn, the orientation $\mathbf{R}$ of the robot can be determined (eq.33).

Finally, by modeling the bias evolution $(\mathbf{a}_b, \boldsymbol{\omega}_b)$, as its value plus the random walk $(\mathbf{a}_r, \boldsymbol{\omega}_r)$, equations 36 and 37 can be obtained.

Notice that all equations defying the state model, are expressed using $\longleftarrow$ instead of $=$. This is a simplification to express that the variables (left) are being update by the expression on their right.

$$\mathbf{R} \longleftarrow \mathbf{R} \oplus (\boldsymbol{\omega} dt) \tag{33}$$

$$\mathbf{v} \longleftarrow \mathbf{v} + \mathbf{a} \tag{34}$$

$$\mathbf{p} \longleftarrow \mathbf{p} + \mathbf{v}dt + 0.5\mathbf{a}dt^2 \tag{35}$$

$$\mathbf{a}_b \longleftarrow \mathbf{a}_b + \mathbf{a}_r \tag{36}$$

$$\boldsymbol{\omega}_b \longleftarrow \boldsymbol{\omega}_b + \boldsymbol{\omega}_r \tag{37}$$

3.3 Observer definition

- Prediction step

  Given the state on eq. 28, and all the equations that define the state model (from 33, to 37) the function f in eq. 38 can be defined (analogy of the definition made on eq. 14).

$$\mathbf{x} \longleftarrow f(\mathbf{x}, \mathbf{u}, \mathbf{r}) \tag{38}$$

  With it, in order to fully define the prediction step, the Jacobians with respect to the state $\mathbf{x}$, the control signal (in this case IMUs) $\mathbf{u}$, and, finally, the bias deviation $\mathbf{r}$, must be computed.

  The first Jacobian to calculate is the one with the state as an output and itself as an input: $\mathbf{J}_x^x$. Inside matrix defined in eq. 39, the output variables unrelated to the input ones are considered a zero matrix of size $(3, 3)$. The rest of the non-zero values are shown in the next expressions, and they are also $(3, 3)$ arrays.

  The expression of the Jacobians, as it can be tedious, is expressed inside the annex subsection B.1 *Jacobian computation*. Here, the rule-chain expressions and the identity ones are shown.

  The Jacobinas that include lie groups are expressed the same way as [SDA21] because *Manif* library, when it comes to programming, automatically computes them.

$$\mathbf{J}_x^x = \begin{pmatrix} \mathbf{J}_R^R & \mathbf{J}_v^R & \mathbf{J}_p^R & \mathbf{J}_{a_b}^R & \mathbf{J}_{\omega_b}^R \\ \mathbf{J}_R^v & \mathbf{J}_v^v & \mathbf{J}_p^v & \mathbf{J}_{a_b}^v & \mathbf{J}_{\omega_b}^v \\ \mathbf{J}_R^p & \mathbf{J}_v^p & \mathbf{J}_p^p & \mathbf{J}_{a_b}^p & \mathbf{J}_{\omega_b}^p \\ \mathbf{J}_R^{a_b} & \mathbf{J}_v^{a_b} & \mathbf{J}_p^{a_b} & \mathbf{J}_{a_b}^{a_b} & \mathbf{J}_{\omega_b}^{a_b} \\ \mathbf{J}_R^{\omega_b} & \mathbf{J}_v^{\omega_b} & \mathbf{J}_p^{\omega_b} & \mathbf{J}_{a_b}^{\omega_b} & \mathbf{J}_{\omega_b}^{\omega_b} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_R^R & 0 & 0 & 0 & \mathbf{J}_{\omega_b}^R \\ \mathbf{J}_R^v & \mathbf{J}_v^v & 0 & \mathbf{J}_{a_b}^v & 0 \\ \mathbf{J}_R^p & \mathbf{J}_v^p & \mathbf{J}_p^p & \mathbf{J}_{a_b}^p & 0 \\ 0 & 0 & 0 & \mathbf{J}_{a_b}^{a_b} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{J}_{\omega_b}^{\omega_b} \end{pmatrix} \tag{39}$$

  where,

$$\mathbf{J}_R^R = \mathbf{J}_R^{R \oplus (\omega dt)} \tag{40}$$

$$\mathbf{J}_{\omega_b}^R = \mathbf{J}_\omega^R \mathbf{J}_{\omega_b}^\omega \quad B.1 \tag{41}$$

$$\mathbf{J}_R^v = \mathbf{J}_a^v \mathbf{J}_R^a \quad B.4 \tag{42}$$

$$\mathbf{J}_v^v = \mathbf{Id} \tag{43}$$

$$\mathbf{J}_{a_b}^v = \mathbf{J}_a^v \mathbf{J}_{a_b}^a \quad B.7 \tag{44}$$

$$\mathbf{J}_R^p = \mathbf{J}_a^p \mathbf{J}_R^a \quad B.12 \tag{45}$$

$$\mathbf{J}_v^p = dt\mathbf{Id} \tag{46}$$

$$\mathbf{J}_p^p = \mathbf{Id} \tag{47}$$

$$\mathbf{J}_{a_b}^p = \mathbf{J}_a^p \mathbf{J}_{a_b}^a \quad B.14 \tag{48}$$

$$\mathbf{J}_{a_b}^{a_b} = \mathbf{Id} \tag{49}$$

$$\mathbf{J}_{\omega_b}^{\omega_b} = \mathbf{Id} \tag{50}$$

Secondly, the Jacobian of the state with respect to the measurements, $\mathbf{J}_u^x$, must also be computed.

$$\mathbf{J}_u^x = \begin{pmatrix} \mathbf{J}_{a_m}^R & \mathbf{J}_{\omega_m}^R \\ \mathbf{J}_{a_m}^v & \mathbf{J}_{\omega_m}^v \\ \mathbf{J}_{a_m}^p & \mathbf{J}_{\omega_m}^p \\ \mathbf{J}_{a_m}^{a_b} & \mathbf{J}_{\omega_m}^{a_b} \\ \mathbf{J}_{a_m}^{\omega_b} & \mathbf{J}_{\omega_m}^{\omega_b} \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{J}_{\omega_m}^R \\ \mathbf{J}_{a_m}^v & 0 \\ \mathbf{J}_{a_m}^p & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{51}$$

where,

$$\mathbf{J}_{\omega_m}^R = \mathbf{J}_{\omega}^R \mathbf{J}_{\omega_m}^{\omega} \quad B.15 \tag{52}$$

$$\mathbf{J}_{a_m}^v = \mathbf{J}_a^v \mathbf{J}_{a_m}^a \quad B.20 \tag{53}$$

$$\mathbf{J}_{a_m}^p = \mathbf{J}_a^p \mathbf{J}_{a_m}^a \quad B.23 \tag{54}$$

Finally, the last Jacobian to compute is the state with respect to the bias deviation: $\mathbf{J}_r^x$

$$\mathbf{J}_r^x = \begin{pmatrix} \mathbf{J}_{a_r}^R & \mathbf{J}_{\omega_r}^R \\ \mathbf{J}_{a_r}^v & \mathbf{J}_{\omega_r}^v \\ \mathbf{J}_{a_r}^p & \mathbf{J}_{\omega_r}^p \\ \mathbf{J}_{a_r}^{a_b} & \mathbf{J}_{\omega_r}^{a_b} \\ \mathbf{J}_{a_r}^{\omega_b} & \mathbf{J}_{\omega_r}^{\omega_b} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{J}_{a_r}^{a_b} & 0 \\ 0 & \mathbf{J}_{\omega_r}^{\omega_b} \end{pmatrix} \tag{55}$$

where,

$$\mathbf{J}_{a_r}^{a_b} = \mathbf{Id} \tag{56}$$

$$\mathbf{J}_{\omega_r}^{\omega_b} = \mathbf{Id} \tag{57}$$

With the function of the state defined and its Jacobians too, the prediction step, as it is defined in eq. 16 and eq. 17 can be defined now in eq. 58 and eq. 59:

$$\mathbf{x} \longleftarrow f(\mathbf{x}, \mathbf{u}, 0) \tag{58}$$

$$\mathbf{P} \longleftarrow \mathbf{J}_x^x \, \mathbf{P} \mathbf{J}_x^{x\,T} + \mathbf{J}_u^x \mathbf{Q} \mathbf{J}_u^{x\,T} + \mathbf{J}_r^x \mathbf{W} \mathbf{J}_r^{x\,T} \tag{59}$$

Where $\mathbf{P}$ represents the covariance matrix of the state, $\mathbf{Q}$ the covariance matrix of the IMU's noise, and $\mathbf{R}$ the deviation of the bias.

- Correction step

  In this case, as has already been said, the robot chassis is adapted to include markers for motion registration. With these, and the cameras available in the laboratory, OT, the position ($\mathbf{p}_m$) and orientation ($\mathbf{R}_m$) of the robot can be obtained.

  The expectation of the measurement can be expressed as $\mathbf{e}$, and then, equal to a bundle formed by the expected orientation $\mathbf{R}$ and position $\mathbf{p}$. These two values are the ones computed in eq. 58 and, here, they are represented inside the expectation (eq. 60)

$$\mathbf{e} \longleftarrow < \mathbf{R}, \mathbf{p} > \tag{60}$$

Then, the correction is the difference between the value measured by the OT system ($\mathbf{R}_m$, $\mathbf{p}_m$) and the expected ones ($\mathbf{R}$,$\mathbf{p}$). The result of this step (eq. 61) are the correction on the orientation ($\mathbf{R}_z$), and the correction on the position ($\mathbf{p}_z$).

$$\mathbf{z} \longleftarrow \begin{pmatrix} \mathbf{R}_z \\ \mathbf{p}_z \end{pmatrix} = \mathbf{y} < - > \mathbf{e} = \begin{pmatrix} \mathbf{R}_m \ominus \mathbf{R} \\ \mathbf{p}_m - \mathbf{p} \end{pmatrix} \tag{61}$$

For eq. 61 the Jacobians must be computed. In this case, the only Jacobians that are relevant are those with respect to the state $\mathbf{J}_x^z$ and the measurement $\mathbf{J}_y^z$.

The definition of the $\mathbf{J}_x^z$ is the chain rule expressed in eq. 62

$$\mathbf{J}_x^z = \mathbf{J}_e^z \mathbf{J}_x^e \quad B.24 \tag{62}$$

And, the definiton of $\mathbf{J}_y^z$ is the one in eq. 63.

$$\mathbf{J}_y^z = \begin{pmatrix} \mathbf{J}_{R_m}^{R_z} & \mathbf{J}_{p_m}^{R_z} \\ \mathbf{J}_{R_m}^{p_z} & \mathbf{J}_{p_m}^{p_z} \end{pmatrix} = \begin{pmatrix} \mathbf{J}_{R_m}^{R_z} & 0 \\ 0 & \mathbf{J}_{p_m}^{p_z} \end{pmatrix} \tag{63}$$

where,

$$\mathbf{J}_{R_m}^{R_z} = \mathbf{J}_r^{-1}(\mathbf{R}_z) \tag{64}$$

$$\mathbf{J}_{p_m}^{p_z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{65}$$

With this, the Kalman gain is defined in eq. 67, and the correction step is completed with the correction over the estimation in eq. 68, and the upload of the covariance of the state in eq. 69.

$$\mathbf{Z} = \mathbf{J}_x^z \mathbf{J}_x^{z\,T} + \mathbf{J}_y^z \mathbf{V} \mathbf{J}_y^{z\,T} \tag{66}$$

$$\mathbf{K} = -\mathbf{P}\mathbf{J}_x^{z\,T}\mathbf{Z}^{-1} \tag{67}$$

$$\mathbf{x} \longleftarrow \mathbf{x} \oplus \mathbf{K}\mathbf{z} \tag{68}$$

$$\mathbf{P} \longleftarrow \mathbf{P} - \mathbf{K}\mathbf{Z}\mathbf{K}^T \tag{69}$$

3.4 Simulation experiment

- Algorithm implementation

  To implement the LEKF algorithm, it is decided to use the Python3 language, as it is a language with simpler tools than the other available candidate, C++. It must be said that both of them, as will be seen in subsection 3.5 *First attempt: ROS2 communication*, offers the possibility of implementing a ROS2[14] node and, so, to be used in a real case scenario. The main difference between them depends on the speed of the running loops inside the code (such as the *main* function). The reason why Python is slower is that it is an interpreted language, which requires more work from the CPU to run certain statements. This fact, however, does not preclude the performance of simulations. Later, in the implementation of the system for real data, it will be evaluated whether, on the other hand, it does deprive it of sufficiently fast calculations.

  To write the code that turns the mathematical approach in Section 3.3 *Estimation strategy*, into useful software, many paths can be taken. In this case, the decision is to establish a class called Lie Extended Kalman Filter (LEKF), which sets the steps of the process as callable attributes.

  First, the attribute called *Predict* is defined. It uses as input the control command, in this case, the IMU signal ($\mathbf{u}$), and an update time difference (dt), which is no more than the inverse of the IMU's frequency. With its call, the prediction step is produced.

  Secondly, the *Correct* attribute is also defined, which uses the signal received from the OT system to perform the correction step.

  Apart from the main steps, the class contains the previously defined Jacobians, the functions: f, e, and z, as well as the import of two extra classes that define the type of signal received and one that defines the state.

- Simulation script

  In order to test, correct, and analyze the designed filter, in addition to the *LEKF* class, a script is designed to simulate its operation.

  It is structured in such a way that the two expected errors of the signals, as well as the deviation of the bias, can be introduced by their standard deviation σ. Also, the initialization of the covariance arrays: $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{W}$, and $\mathbf{V}$, must be defined.

  The script allows the user to decide whether to introduce certain errors, to use correct step or prediction, both or one of them, and, finally, to plot the **(x,y,z)** coordinates of the fusion data compared to the simulated trajectory.

  The way the simulated data updates itself from a hard-coded trajectory is defined in the attributes *update* and *observe*. These two include the noise on the measurement and define the IMU and the OT, respectively.

[14] In fact, ROS and ROS2 are agnostic languages, so their communication does not depend on the language in which the nodes are written.

The simulation can offer different results as there are free variables when it is started. By changing the values of the standard deviations of the signals, the result can vary, obtaining different values for the state observed.

If, for example, the initial covariance matrix of the state, $\mathbf{P}$, is increased, meaning that there is more uncertainty between the real initial state and the initialization one, the correction step would impact more over the estimation.

Knowing that, to expect the most realistic result, the σ that defines the errors coming from the IMU and the OT, as well as the bias drift, are those specified by the seller. Table 11 defines these errors where: a_{WN} and ω_{WN} are the noise coming from the IMU, the $a_{bias\ \ drift}$ and $\omega_{bias\ \ drift}$ are the representation of the bias drift, and, finally, the R_{WN} and p_{WN} which are the noise inside the OT measurement.

Noise	σ
a_{WN}	$6.3 \cdot 10^{-5} m/s^2$
ω_{WN}	$8.7 \cdot 10^{-5} rad/s$
$a_{bias\ \ drift}$	$4.0 \cdot 10^{-4} m/s^2$
$\omega_{bias\ \ drift}$	$3.9 \cdot 10^{-5} rad/s^2$
R_{WN}	$6.0 \cdot 10^{-3} rad$
p_{WN}	$3.0 \cdot 10^{-4} m$

Table 11: Commercial signal errors for IMU and OT, source: B.2.1.

- Simulation results.

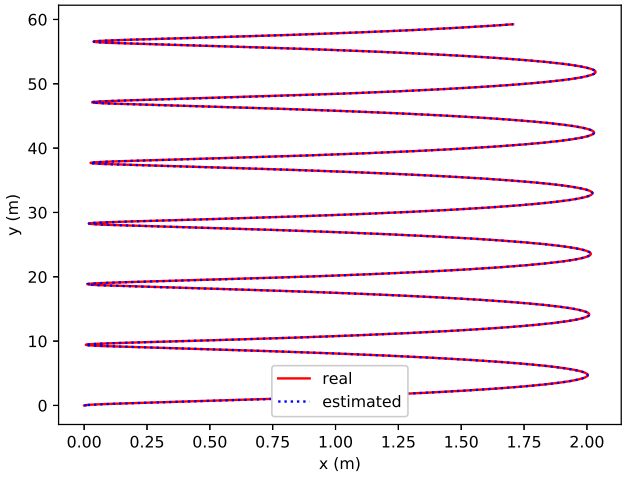
  The results of the simulation show robust behavior for any trajectory tried. Instead of working with 3D trajectories, it is decided to use 2D ones. The reason is that it is simpler, and its result should offer the same robustness as if it were a 3D one.

  Results of the performance using different 2-D trajectories are shown in Fig. B.3.1, B.3.2, that can be found in the annex subsection B.3 *Simulated trajectories*.

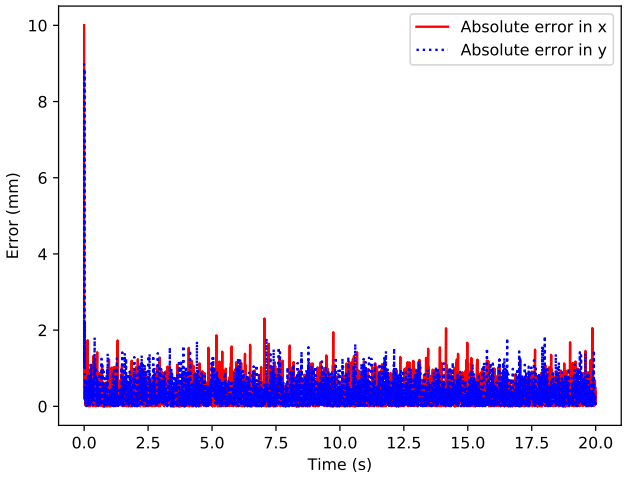
  In Fig.13 another trajectory is shown. It accomplishes the same characteristics as the other two simulations, but it is shown on the main page to make its description easier. Fig. 13 *a)*, shows the comparison between the simulated trajectory (called *real*), and the result of fusion data from the LEKF (called *estimated*).

  As it is observed in Fig. 13 *b)*, the absolute error between the x and y coordinates is large at the beginning because the initial state of the robot is unknown, but, quickly, it gets to an error with its mean around 1 mm.

  As the objective is to perform high dynamics control, precision is critical, and a useful tool to compute that precision can be the RMSE (Root Mean Squared Error). The results for each trajectory simulated are shown in Table 12.



(a) Spring line trajectory simulation.



(b) Absolute error between the simulated trajectory and the estimated one.

Figure 13: Comparison between the simulated trajectory and the estimated one, as well as the computed absolute error.

Trajectory	RMSE on x (mm)	RMSE on y (mm)
Straight line	0.1283	0.1238
Circular	0.1161	0.1148
Spring	0.1242	0.1218

Table 12: RMSE of each simulated trajectory not including the first 0.2 seconds.

The simulation shows that, for 2D trajectories, the error with respect to the real state is tenths of a millimeter.

Given the results, it is decided to experiment with the filter with real data. For this purpose, instead of the quadruped, the drone available in the laboratory is used. The main reason is that, for the drone, the interface describing how the information is published, and the OT's system is already programmed.

It is important to remember that the filter is intended to be installed in SOLO 12, but, as the variables used to describe the state, as well as the merging of sensors used, match with what the drone offers, the results are considered to be significant for drawing conclusions.

3.5 Real data experiment

The test of the fusion filter is now done with real data. This section offers an explanation of how the experiment has to be prepared (3.5 *Preparation*), a first experiment attempt using ROS2 communication (3.5 *First attempt: ROS2 communication*), and, finally, a definitive experiment using Comma-Separated Values (CSV) files from the drone's controller PX4 [Sou23b](3.5 *Second attempt: CSV file*).

The first attempt explains the communication with the concepts used by Robot Operating System (ROS). The analysis of the results is conducted by the usage of a visual tool called RViz [ROS20], the analysy of the data, and the plotting of its values. For the reasons explained in the same subsection, the discussion over the results are done in the same chapter.

The second, and definitive attempt uses less complicated communication, as the LOG of the PX4 sytyem, is read as a CSV. The test script is very similar to the one explained in the subsection *Algorithm implementation* in 3.4 *Simulation experiment*, as the only change is that now no simulation is required.

As introduced before getting into any of the tests, an explanation of the experiment being conducted is required, and that is what the next subsection is about.

Preparation

As the end of the previous section suggests, the experiment is carried out with a six-propeller drone within a zone with multiple cameras to register its position and orientation.

In order to start the experiment, the calibration of the OT system must be carried out. The steps to follow are listed below and, for most of them, the appendix section B.4 *Optitrack calibration steps* gives a graphical description.

1. Clean all reflecting objects from the area.

   Once the only reflecting marks come from residual light, mask them (i.e., light from the outside may not be able to be shaded, but if it stays still, it can be masked). Fig.B.4.1, and B.4.2

2. Proceed with the wanding process

   Through this process, the OT system creates a cloud of samples per camera. The more the wand is moved around, the better the sampling. In fact, it is possible to create blind spots for cameras that do not have enough samples and, that concurs on corrupted data. Once the number of samples per camera is around 3000, the calibration can be set. Fig. B.4.3

3. Once the calibration is complete, the floor plane must be determined.

   This procedure sets the frame of the world, which, in this case, is FUR (Front Up Right) for the X-Y-Z axes. The procedure is done with an artifact that also contains reflectors on its surface. Fig. B.4.5, and B.4.4

4. Create a rigid body and choose its centroid.

   Now that all the space for flying is set, the only last requirement is to create the rigid body that is going to be used. In this case, the rigid body is the flying drone, and its centroid is defined with the local coordinates of its IMU (unifying the frame of the IMU and the body's).

5. Start publishing data.

   Finally, the last step is to broadcast the data of the drone.

Once the flying zone is set, the desired trajectory can be followed with the drone's remote control.

The trajectory decided to test is a square in a plane parallel to the floor. The drone has to increase its attitude to approximately 2 m, then yaw 180°, move in a straight line, and reach a marked position on the ground. Then, yaw 180° again, and move in a straight, constant attitude line. Move a straight line to the right frame of the drone until it is the same x as it was initially, and, finally, reach its initial position and orientation, reaching the ground at this position.

At the same time the flying is done, a camera is placed to record the whole experiment.

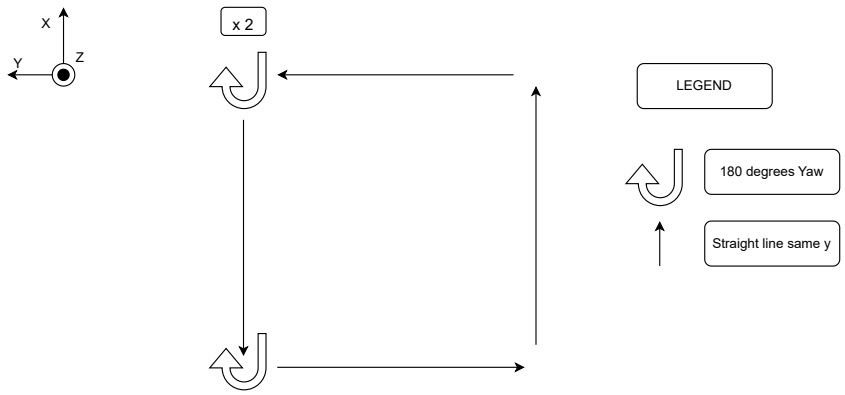


Figure 14: Scheme of the experiment.

First attempt: ROS2 communication

After the flight is made, all data coming from the drone's IMU and the OT system, as well as more messages that do not concern this thesis, are stored in a file that ROS uses called ROS BAG. By doing this, now it is possible to run data from this experiment over and over without the need to fly again. Fig. 15 is a visualization of how the communication is done. The Fusion node, the one that contains the LEKF, corresponds to the data stored in IMU_topic and /optitrack/borinot_fur_ot/pose.

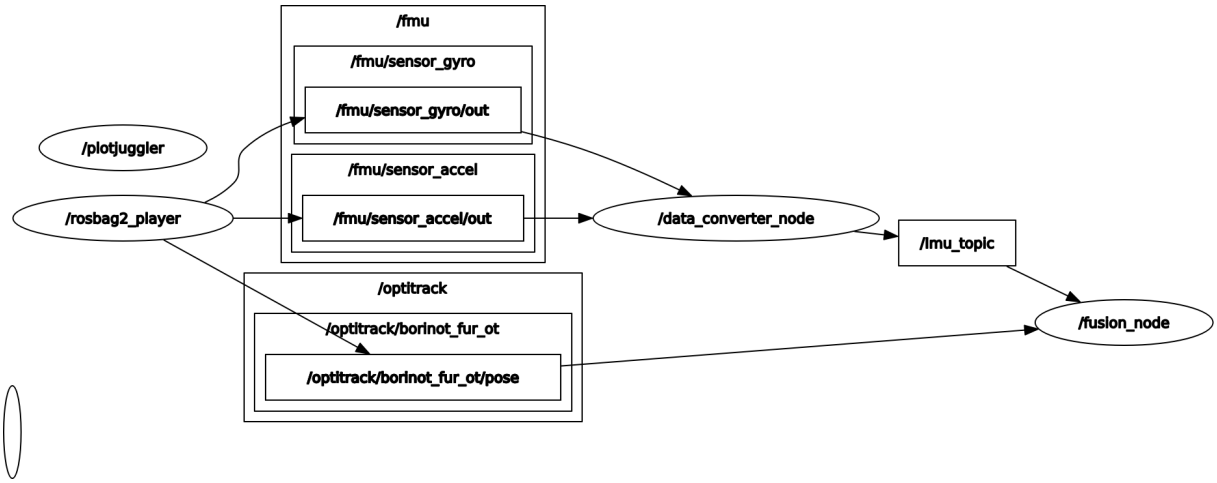


Figure 15: Graphical view of the ROS2 connection nodes (Rqt graph).

The first tool used to analyze how the node behaves is the RViz visualizer. On it, it is decided to show the OptitrackFrame (as ground truth) and the frame published by the Fusion node.

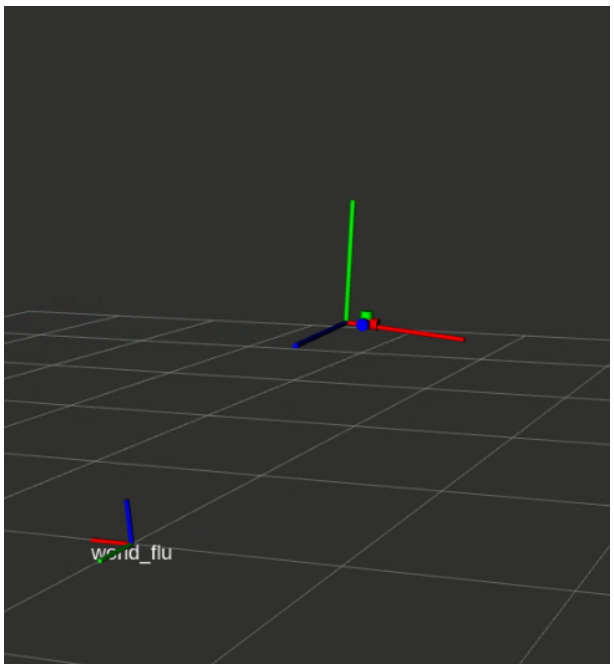


Figure 16: Visualization of a badly predicted frame (little one), vs the one published by the OT (big one).

The first observable fact, is that both body frames are not referred to the global Front Left Up (frame convention) (FLU) as excepted. This fact could affect on both the estimation and the correction as the node is designed to work with both FLU as body frame and global frame.

Secondly, the movement of the frame defining the topic publish by LEKF does not show a continuous movement. Instead, linear trajectories are erratic, either moving faster than the OT frame or slower (Fig. 16). It is hard to tell if the problem is still because the information should be changed in the frame, or if something else is producing this bad estimation.

This second observation leads to take a look into the way the linear acceleration is being published by IMU_topic. Analyzing it, a lack of continuous frequency is perceived. At some point, IMU's messages are published at a frequency of 740 Hz, at others at 250 Hz, and there are even situations where the next time stamp of the message is lower than its prior. One possible reason for this may be the loss of some Wi-Fi connections during the flight.

When plotting the linear acceleration, it is perceived as a much noisier signal than expected. Data shows how, after the start of the propellers, the signal from the accelerometer gets noisier. As can be seen below in Fig. 17, the standard deviation of the IMU linear acceleration gets to values around 3 and 4 m/s^2. This increase in noise is believed to come from the vibration that the propellers cause on the accelerometer.

As the IMU also offers the angular velocity as an input signal, by looking at the gyroscope data, a less noisy signal is perceived in Fig. 18. This fact may be the reason why, on Rviz, the state is able to do continuous spins. Despite this, gyroscope data also lacks of continuous frequency, as it is being published at the same time that the linear acceleration.

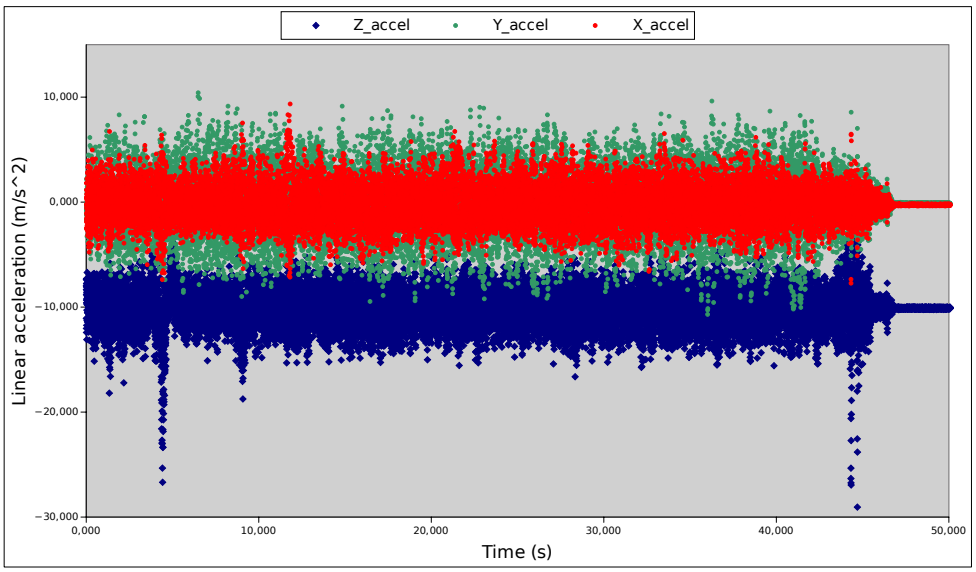


Figure 17: Linear acceleration published by the IMU's topic.

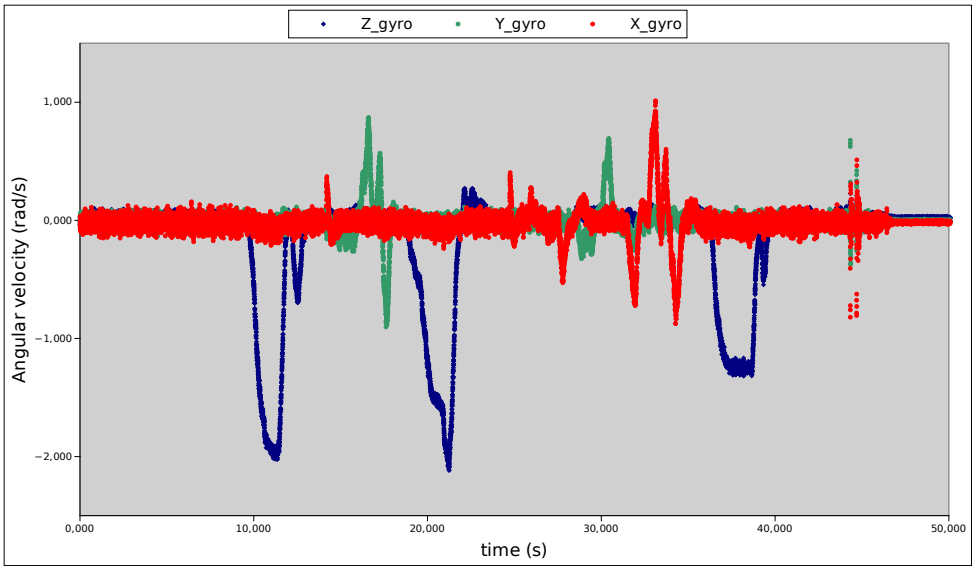


Figure 18: Angular velocity published by the IMU's topic

By plotting the state position through the whole trajectory on a second-time scale, the values of the position seem correct (Fig. 19). The problem begins when, by taking a closer look at its behavior (in milliseconds in this case), the prediction step is greatly affected by either the noise of the accelerometer or the lack of continuous frequency (Fig. 20).

Instead of having predictions at the frequency of the IMU, approx. 475 Hz (each 2 ms), the position is being estimated at a random timestamp. On the other hand, correction seems to work, as every 10 ms (100 Hz) the positions get to a value. Anyhow, it is hard to tell if the LEKF is only correcting, or doing both steps.

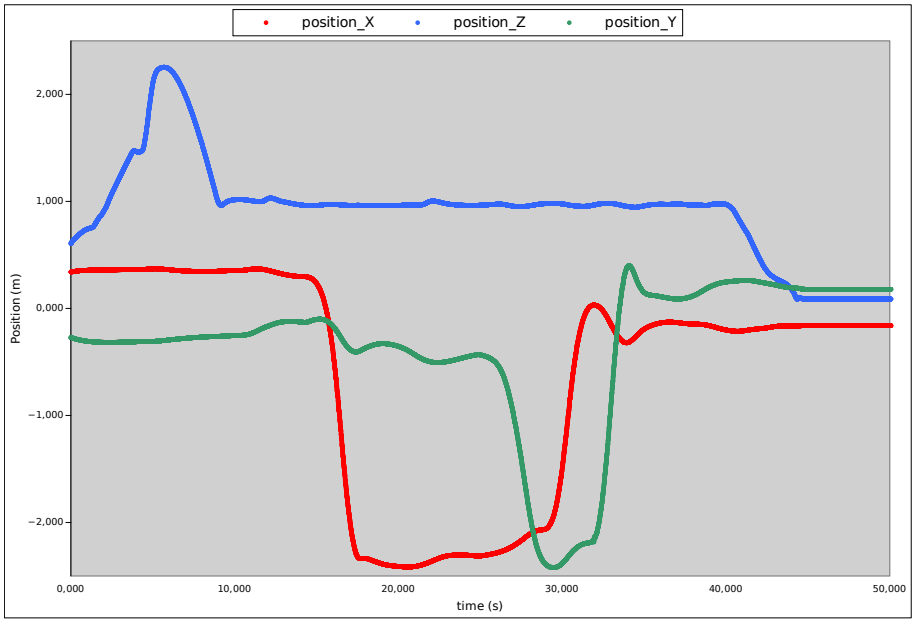


Figure 19: Position published by the Fusion node on a seconds-time-scale.

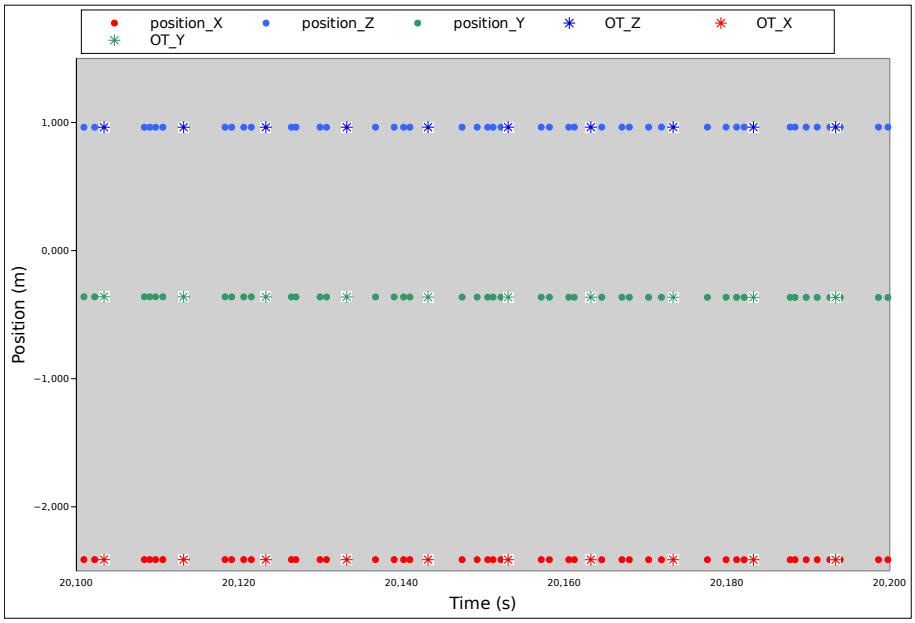


Figure 20: Zoom of the position published by the Fusion node.

At this point, to follow up on the testing of the filter, it is requested to access data that is not as noisy as the one published on topic IMU. To do so, a CSV file is extracted from the drone controller. This file is not affected by the possible loss of connection during the flight and, at the same time, contains already filtered data for both measurements of the IMU. Even more, the PX4 controller offers its own version of the EKF, and, so, it can be a great opportunity for comparing results.

This first attempt lacks of reliability to draw any conclusions on the ROS2 node developed. Despite that, it opens a path for future improvements that are listed in the 8 *Conclusions* chapter.

Second attempt: CSV file

The intention of this chapter is to use a cleaner version of the data to test the filter. It also intends to compare the results obtained to the EKF version of PX4. In this case, as another probable cause of the malfunction of the last experiment could be a bad implementation of the ROS2 node, this communication is no longer required.

The data stored as LOG files in CSV format, as explained in the chapter above, are saved in the controller in three different scripts:

- Sensor combined: This file stores the linear acceleration registered by the accelerometer as well as the angular velocity registered by the gyroscope. Both data are recorded every 0.005 s (200 Hz).

- Vehicle_local_position: The OT data is stored through the PX4 system, and it separates the script that saves the position from the one that stores the orientation. In this case, the position is stored every 0.1 s (10 Hz).

- Vehicle_attitude: As said before, the orientation is saved in a separate script from the position. In this case, the time between data is stored every 0.05 seconds (20 Hz).

As the main issue with the first attempt was the noisy acceleration, one good thing to do is check if the acceleration used this time is proper. Fig. 21 shows how, now, the standard deviation of the white noise is around 0.25 m/s^2 (far from the 3.0 m/s^2 of the Fig. 17).

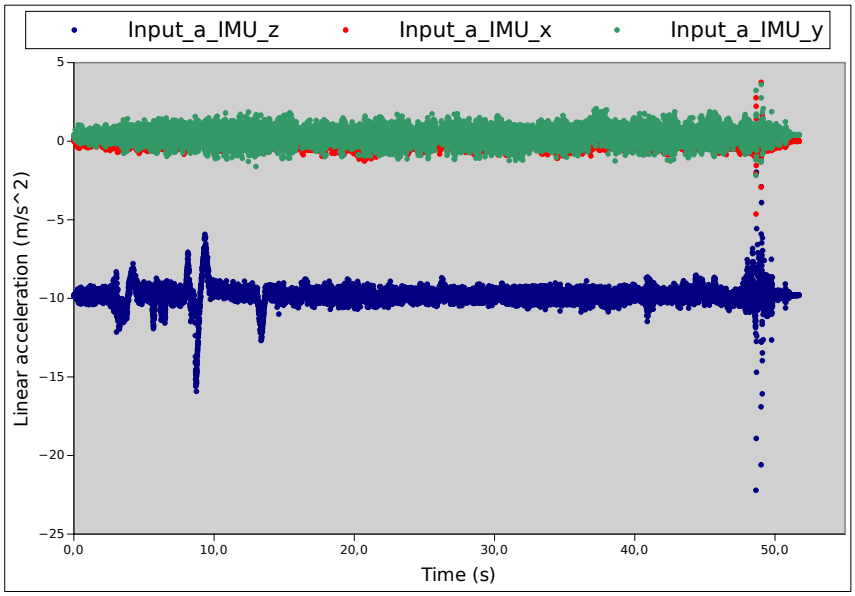


Figure 21: Filtered linear acceleration measurement from PX4 LOG.

The second thing required in order to deal with the data is to know which frame it is expressed in. It has already been discussed how important is to know in which way the information is referred to. In fact, the Rviz visualization (Fig. 16) showed that this is crucial to avoid misleading conclusions.

From Fig. 18, the IMU body reference can be deduced as Front Right Down (frame convention) (FRD). That is because the first spin made should increase the angular velocity in the z-axis positively in a FLU body frame. In this case, Fig. 18 (blue line around 10 s), shows a negative increase. Then, after the 180° spin, the posterior movement in a straight line, in a FLU should increase positively the pitch velocity, instead it gets to negative values (green line in Fig. 18 around 16 s).

The World frame of the PX4 system is settled in its configurations and is known to also be FRD.

Finally, the body frame of the robot, for the PX4 system, is equal to the IMU body frame.

All this information, then, can be summarized in Fig. 22, where the left column shows the desired frames for the world and for the body, and the right one, the same ones coming from the PX4 LOGs.

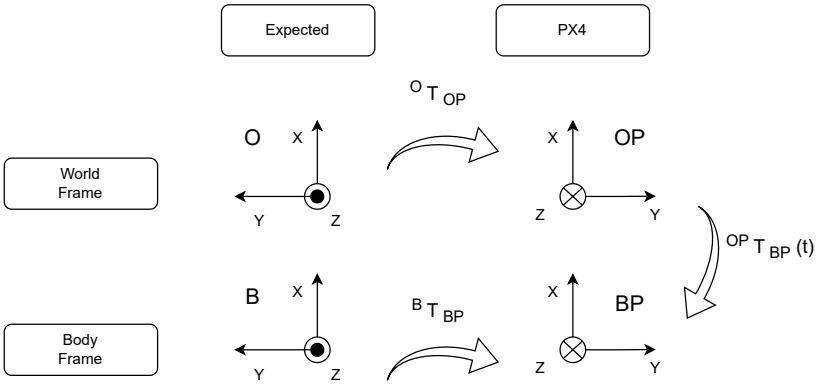


Figure 22: World frame and body frame of the expected convention and the PX4's one.

In order to change the frames to the desired ones, it is required to use a transformation matrix. In this case, the matrices needed have to satisfy the next equation:

$$a^B = R^B_{BP}\, a^{BP} \tag{70}$$

$$\omega^B = R^B_{BP}\, \omega^{BP} \tag{71}$$

$$T^O_B = T^O_{OP}\, T^{OP}_{BP}\, T^{BP}_B = T^O_{OP}\, T^{OP}_{BP}\, ({T^B_{BP}}^{-1}) \tag{72}$$

Equations 70 and 71 express the linear acceleration of the local frame defined in the PX4 system to the local frame expected by the definitions in 29 and 32. Equation 72 expresses the orientation and position coming from the OT in the world frame expected.

The values of any transformation matrix express a rotation and a translation. In this case, the transformation matrices used in equations 70, 71 are just rotation matrices, as the local frames in both systems are placed on the same point. For equation 72, T^O_{OP} and T^{BP}_B are the same rotation matrices of equations 70 and 71, respectively, expressed as transformation ones. This is due to the fact that the information of the OT can be defined as the transformation T^{OP}_{BP} and, then, to express the equation, the order of the matrices must be correct.

Transformation T^{BP}_B is expressed in eq. 73, where, R^B_{BP} is a rotation of 180 degree around the x-axis, and p^B_{Bp} is equal to zeros as the origin of both references is considered at the same point. As with the simulation subsection 3.4, the *Manifpy* library is used, in this case, to express the rotation as a $SO(3)$ group. In this case, the library accepts the definition of the 3D rotation by introducing the desired rotation in Euler angles[15].

[15]Many other ways can be used to express a rotation, and the website 3D Rotation offers help to express them in rotation matrices, quaternions, etc. [Sou23a]

$$T_{BP}^{B} = \begin{pmatrix} R_{BP}^{B} & p_{BP}^{B} \\ 0_{(1,3)} & 1 \end{pmatrix} \tag{73}$$

$$R_{BP}^{B} = SO3(\pi, 0, 0) \qquad \text{and} \qquad p_{Bp}^{B} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{74}$$

In this particular case, T_{OP}^{O} also expresses a 180° around the x-axis rotation, so its rotation matrix is equivalent to the one of eq. 74.

$$T_{OP}^{O} = \begin{pmatrix} R_{OP}^{O} & p_{OP}^{O} \\ 0_{(1,3)} & 1 \end{pmatrix} = \begin{pmatrix} R_{BP}^{B} & p_{Bp}^{B} \\ 0_{(1,3)} & 1 \end{pmatrix} \tag{75}$$

Apart from changing the frames of the input data, as the list of the CSV files explains, the information from the OT, including position and orientation, is split. As neither comes at the same time, the correction step for this file must be rewritten as two corrections: one for the position and the other for the orientation.

The new Jacobians and the equivalent of eq. 72 are computed in the annex subsection *New required Jacobians* B.5. That being said, after everything is set, the testing of the filter can start.

3.6 Results and discussion

The first plot decided to use to analyze the result of the LEKF is the position of the state through time. This is chosen because the expected trajectory is known and, as the OT correction offers this value with high confidence, its corrections can be trusted and used as ground truth.

Fig. 23 shows a continuous and precise behavior with the expected world reference frame FLU. By taking a second-time-scale look, no difference can be appreciated with the one obtained in Fig. 19 (ROS 2 attempt).

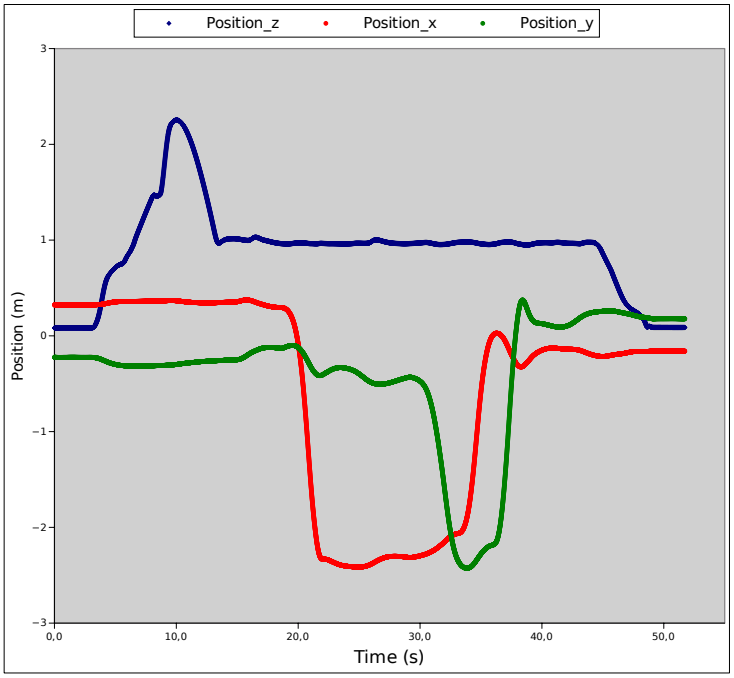


Figure 23: Estimation of the position from the LEKF.

When taking a closer look at the same[16] time period in both Fig.20 and Fig.23, this second one (Fig. 24), shows the expected behavior of the algorithm.

The first noticeable fact is the main virtue of fusing the data of the IMU and the OT. Meanwhile, the OT is giving data each 100 ms (approx 10 Hz), the estimation made through the IMU is feeding the state each 5 ms (approx. 200 Hz).

Secondly, now that the estimation's behavior can be considered approved, it can be a good practice to compare the state given by the LEKF and the EKF of the controller.

Taking the Z-axis of Fig. 23 around seconds 20 and 30, and adding the correction step and the EKF made by the PX4 system, Fig. 25 is obtained.

[16]Note that the time does not match because the data is exported from two different sources where the start of each is considered different. Despite that, the trajectory is the same, and, so, the position's values

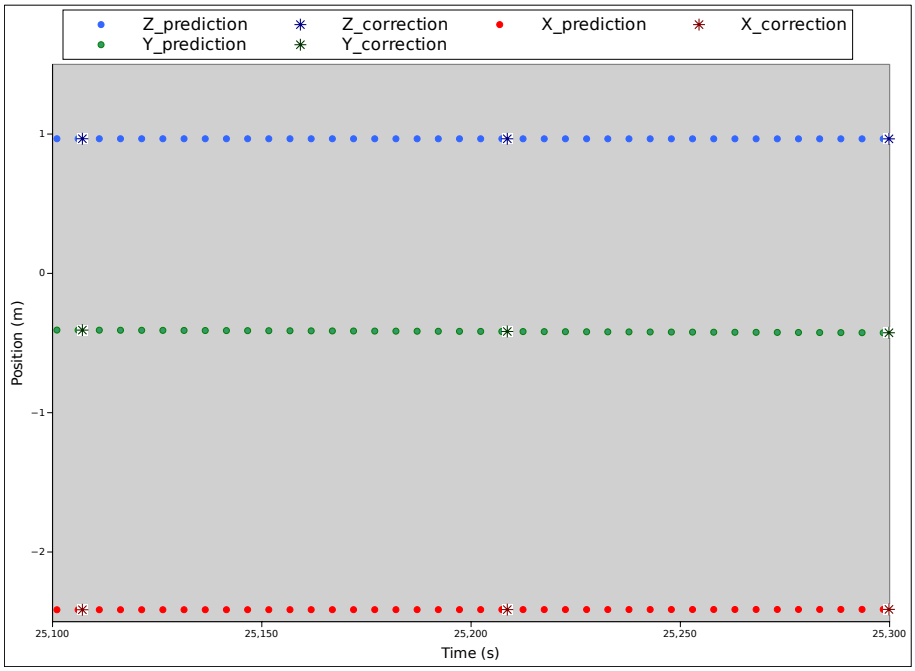


Figure 24: Zoom-in of the observer's position with the prediction's and the correction's steps.

When it comes to the comparison with the PX4 estimation of the Z position, the difference is units of millimeters. Either considering the OT as the position ground truth, or the EKF of the PX4 as the true value, the value is proper for a robot of the size of SOLO 12.

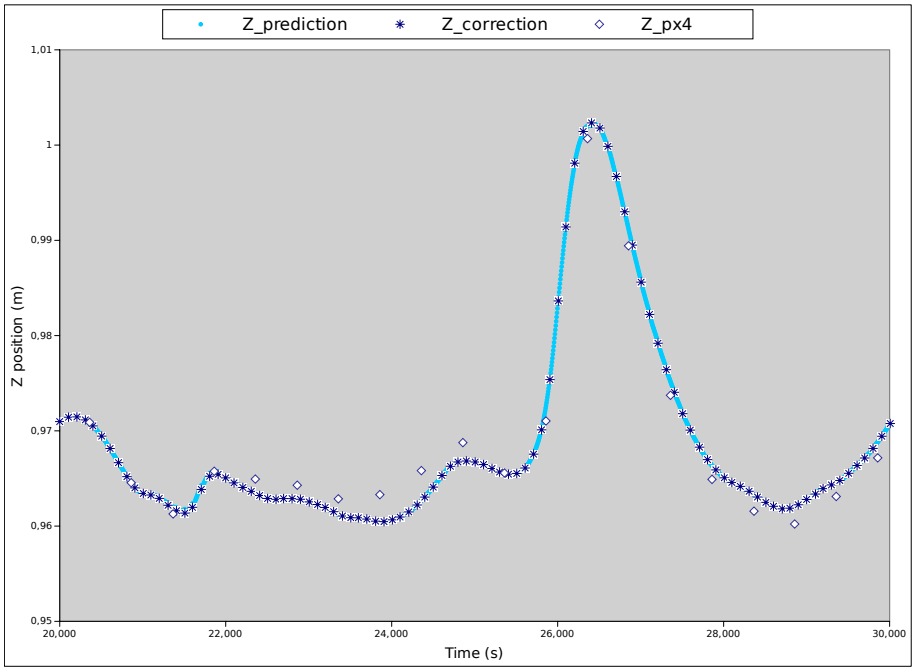


Figure 25: Comparison of position on its Z-axis obtained by the observer, and the one from the PX4 estimation.

As the position is not the only variable in the state, orientation **R** and velocity **v** should also be checked.

Since PX4's EKF also offers an estimation over **v**, in order to compare it with the one obtained by the LEKF, it is considered as a basis. Fig. 26 shows an overlaid graph with these two variables.

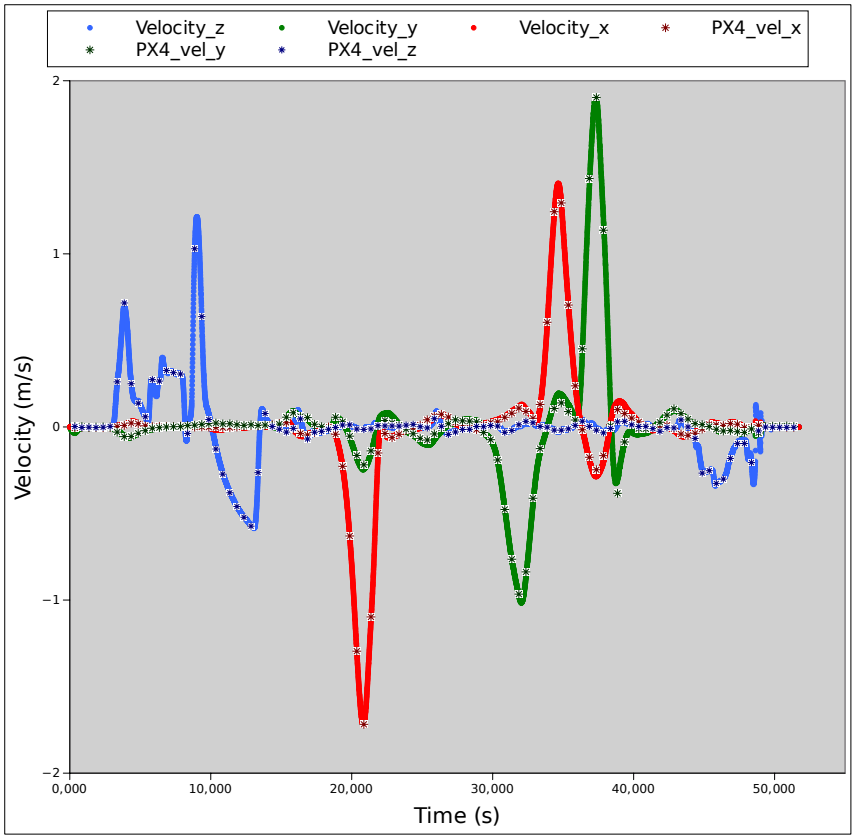


Figure 26: Comparison of the linear velocity, **v**, from LEKF with the linar velocity from the PX4 estimation.

By taking a closer look around times 16 and 24 s, and -0.25 and 0.00 m/s; Fig. 27 shows how the difference between the values obtained by the LEKF and the EKF are quasi-null.

In this case, even though the correction plays a role in the integration of the position, it is not plotted as OT does not offer this variable directly.

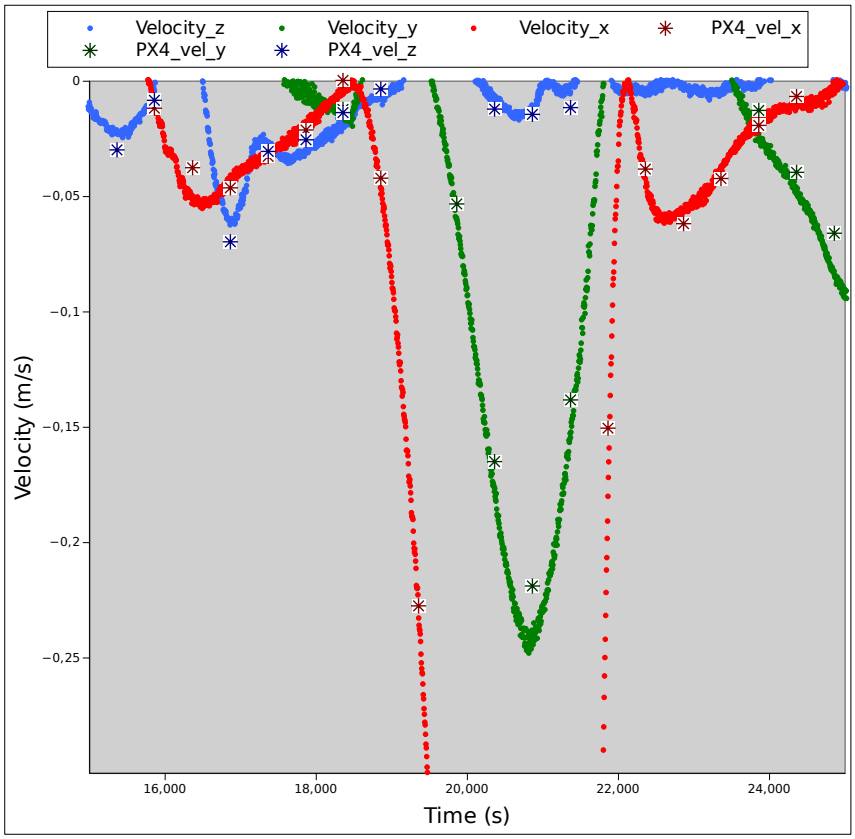


Figure 27: Comparison of the **v**'s values of the LEKF with the ones from EKF, of the PX4 estimation.

For the orientation, **R**, it is decided to plot both the estimation and the corrections as, now, the OT does offer this variable with high precision. In this case, PX4's orientation is not printed since there is it no access to this variable in the same LOG.

Fig. 28 shows first the estimation of the orientation, and Fig. 29 does a zoom-in to appreciate the good quality of the estimation with respect to the correction.

With these results, the software part, and last part of the thesis, is finished. Further steps and more reasoning can be found in the 8 *Conclusions* section, where, a citation over the tuning of the filter is mentioned and explained in the Appendix B.6 *Tuning the filter*.

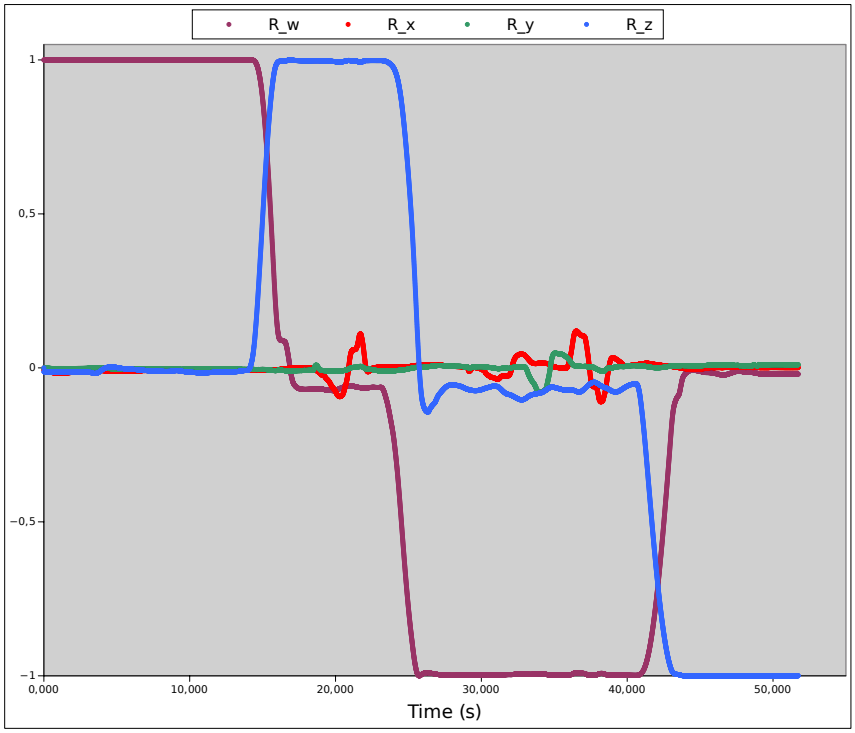


Figure 28: General view of of the **R**'s values computed by the observer.

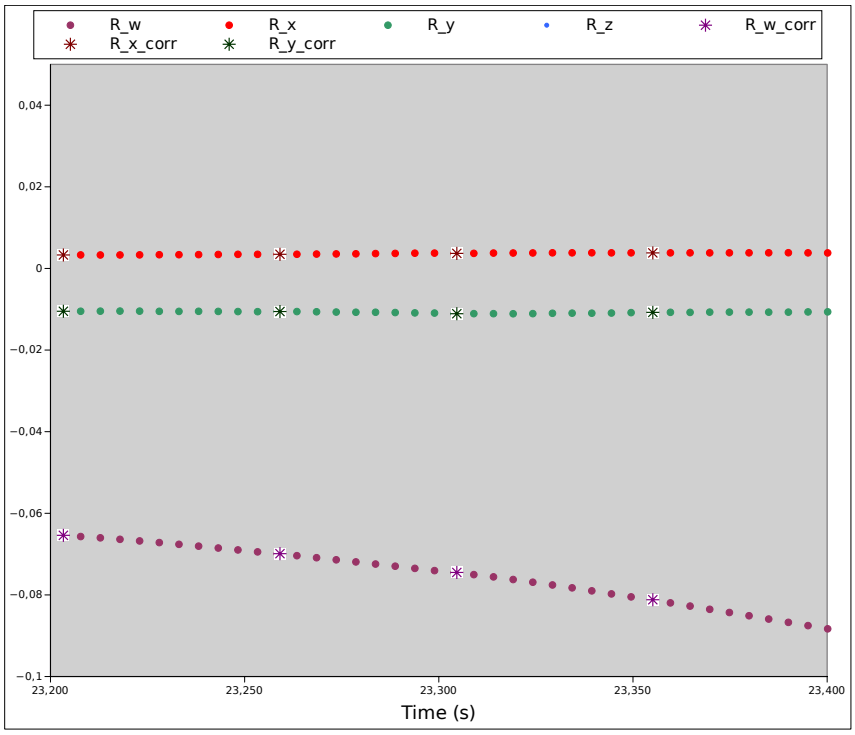


Figure 29: Comparison of the **R**'s values obtained by the LEKF and the ones got by the OT.

4 Temporal planificaction

Foremost, it decided to dedicate an average of 4 to 5 hours per day from the beginning of the work (the first week of February) until the end of it (the last week of June and the first week of July). The total number of hours, including some weekends, is about 450. Taking into account that this work is worth 12 European Credit Transfer and Accumulation System (ECTS) credits (even though it is expected to take about 20) and that for each credit 25 hours must be invested, the expected total, between 300 and 500, is within the margins.

Since this project takes place during the spring term, the first two weeks of April have been chosen to temporarily separate the two themes of the thesis. Both the period before and after Easter follow the same structure: understand the problem and collect information; establish a roadmap with the tutors; work on the solution; and implement and finalize the result.

The duration of the stages named above is expected to differ between the mechanical design and the state estimator design due to the author's lack of experience in the second challenge. Even so, it is expected to dedicate an average of 2 weeks per event, hoping to reach three weeks for the implementation and experimentation of the estimator.

The above plan is summarized in the following Gantt chart (Fig. 30, approved by the tutors both at the beginning and at the end of the project. It shows the distribution of activities mentioned, including the writing of the thesis. The color inside the bars defining each activity is intended to show in which month the activity should be finished.

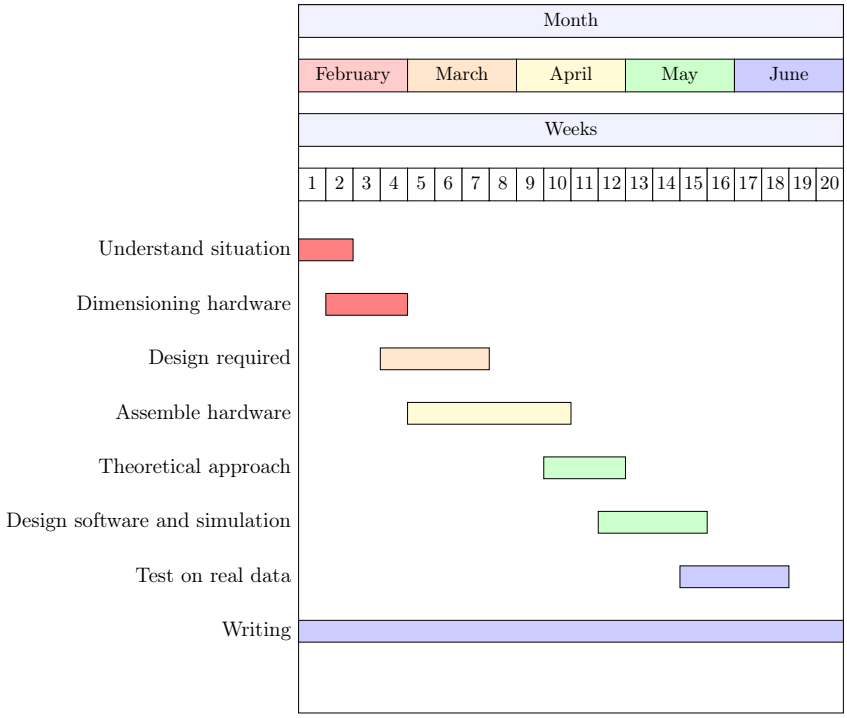


Figure 30: Gantt Chart describing time schedule.

5 Economical study

The economic study of this work includes the purchase of the new parts, the material used for the 3D printing, the use of the printer, and the hours dedicated by the team (the author of this thesis and supervisors). The use of office equipment, the cost of licenses for the software, as well as the use of SOLO 12, are not considered costs in this study.

The price of new elements is described in the following table:

Item	Price (€/unit)	Number of units
Batteries	16.35	6
Onboard PC	529.99	1
Power distribution	15.00	1
Camera	334.00	1

Table 13: Total costs of the new features.

The price of the assembly of 3D-printed parts can be computed by multiplying the price of the material per gram by the total amount of mass printed. In this case, both the prices of the ABS and the HIPS are 50 €/kg. If the mass of each 3D printed part, as seen in *Mechanical design and assembly* 2.4, is summed, the total mass is equal to $232.50g$. And, for the support material, the total mass is $147.25g$. The price for all the printed parts is 19.10 €.

$$\begin{aligned} \text{TOTAL PRINTING MATERIAL} &= 1.07 \times (47.09 + 38.01 + 18.40 + 31.63 + 2 \times 11.15 \\ &\quad + 44.99 + 14.87) \times 50 \times 10^{-3} \\ &= 11.63€ \end{aligned} \tag{76}$$

$$\begin{aligned} \text{TOTAL SUPPORT MATERIAL} &= 1.07 \times (35.86 + 8.19 + 26.70 + 39.81 + 2.84 \\ &\quad + 2 \times 2.84 + 16.08 + 2.46) \times 50 \times 10^{-3} \\ &= 7.47€ \end{aligned} \tag{77}$$

The number of hours that the 3D printing takes place, as introduced in this subsection, must also be taken into account. The 3D-printed parts take time proportional to their mass and geometrical form, as seen in the design of each part. The total number of hours spent printing all the parts is 21 hours and 40 minutes. To know the price of printing per hour, an external web page is used: [INN23]. By selecting the *STL* file, the page gives the total cost of the printing (there are more pages that offer the same service; this is not special).

As the price of the material, in this case, is known, the rest of the cost, after subtracting it, is due to the sum of direct and indirect costs (such as electricity costs, maintenance costs, etc.). To the individual cost obtained (Residual cost in Table 14), divided by the known hours it takes to print, the amount of €per time can be computed. If the average value of this last column is calculated, the result equals 4.16 (€/h). This price is expected to be higher than the one for printing inside the laboratory, as the 3D printer has already been bought. So, considering half of this price and multiplying by the total printing hours, the cost of printing equals 45.12 €.

Item	Cost (€)	Material Cost (€)	Support Cost (€)	Residual Cost (€)	Printing time	Cost (€/h)
NUC base	12.60	2.52	1.92	8.16	5h 16'	1.55
NUC chase	10.20	2.03	0.44	7.73	1h 12'	6.44
Battery base	9.42	0.98	1.43	7.01	3h 14'	2.17
Battery chase	8.31	0.60	0.15	7.56	1h 31'	4.98
Camera base	11.56	1.70	2.13	7.73	4h 28'	1.73
Camera chase	8.31	0.60	0.15	7.56	1h 31'	4.98
PD base	14.39	2.41	0.86	11.12	3h 31'	3.16
PD chase	8.81	0.80	0.13	7.88	57'	8.29

Table 14: Splitted costs of the printing process.

The total amount of time expended in this thesis by the author is greater than expected for its ECTS equivalent. If the thesis is 12 ECTS and 1 ECTS is 25 hours, the total time should have been 300 hours. During 5 months, if 5 days per week are planned to be worked, the amount of time per day should be around 3.75. In the real case, this number is estimated to be more like 4 hours per day and, during some weeks, more than 5 days per week. The estimated time, then, is 437.5 hours (eq. 78). With this amount of time and a salary of 13.50 €/hour, that leaves an amount of 6076.00 €.

$$TOTAL\ \ TIME = 5h/day \times 5days/week \times 3.5weeks/month \times 5months = 437.5h \quad (78)$$

For the price due to the help received by the supervisors of the thesis, a teacher with a doctorate, and a superior engineer with high experience, it can be computed as 15.00 and 14.50 €/h, respectively. The amount of dedication can be computed as 0.15 of the total time for the professor and 0.4 for the engineer. That leaves a total of 984.38 €for the professor's help and 2537.5 euros for the engineer's. The total amount, then, is 3521.90 €. This extra should be computed as hypothetical external help and could either be included in the cost or deducted from the author's salary. In this case, external help is included in the total cost.

Summing up all the computed costs, the total economical value of this project equals 10556.71 €.

Item	Price (€)
New features total cost	895.34
3D material cost	19.1
Printing cost	45.12
Authors salary	6076.00
External help	3251.88
Total sum	10287.44

Table 15: Computed total cost of the project.

As it is clear in Table 15, the greater cost of this project comes from the dedication of the author. This dedication is mostly due to the lack of experience and, therefore, is not a realistic value that could be compared with a commercial budget. Despite that, the rest of the cost is small compared to the cost of the commercial version of SOLO 12, which is around 10000.00€(without assembly). These conclusions about the economic study are not included in the main section as they do not provide further information.

6 Environmental study

The specifications on the 3D printer require a 2200.00 W power supply. This specification is considered to be the highest the machine requires during a work cycle, as it is when the printer is heating the material required. For sure, the more material required to be heated, the higher the energy consumption, and here, to get this number, the time required per piece is multiplied by the amount of energy per piece. Instead of multiplying by 2200.00 W, in this case, the value is considered to be 1100.00 W, as the isolation for heat loss is efficient enough to need half of the initial energy. Table 16 expresses the consumption related to time with the column *Process consumption*, which is equal to the amount of time multiplied by 1100.00 W and the number of pieces produced. In the same Table 16, the last column, *Total consumption*, adds the initial heating to the process consumption. With that, the total amount of electrical power is 91.00 kWh.

Item	Number of pieces	Number of Prototypes pieces	Total hours	Process consumption (kWh)	Total consumption (kWh)
Battery support and chase	1	2	4h and 45'	15.68	22.28
Power distribution support and chase	1	2	4h and 47'	15.79	22.39
Onboard PC support and chase	1	3	6h and 28'	21.34	37.25
Camera support and chase	1	0	6 h	19.80	8.80
Total sum (kWh)					91.00

Table 16: Computed consumption of the 3D printing process.

The main materials used in this master's thesis are printing ones. In this case, both ABS and HIPS are derived from petroleum and, therefore, require a less eco-friendly origin than their competitor, the Polylactic Acid (PLA). That being said, ABS and HIPS can be recycled if treated properly. To do so, perhaps, an amount of energy is required.

For instance, during the printing process, both the material used and the material wasted must be taken into account. As it has been seen, the total amount of ABS required for 3D printing is 232.50 g. This type of plastic can be recycled as long as it has not been exposed to hazardous substances. In the study case, the material can be mechanically recycled, so, therefore, it can retain its mechanical properties until the third reprocessing (where they decrease by 25%).

During the same printing, a certain amount of support material is required. In this case, 134.91 g of HIPS plastic is being used to shape the parts. This kind of plastic is directly thrown into the plastic recycling bin, as it does not form part of the assembly. The same reasoning can be made for the ABS, but it must be remembered that the way the parts are printed affects the amount of HIPS required. That is why, in this project, before printing, each part is placed in such a way that the support required is minimal.

With the total kWh from Table 16 and the mass of material required, the computation of the equivalent amount of CO2[17] can be obtained.

$$\begin{aligned}\text{TOTAL CO2 equivalent} &= 91.00\ \ kWh \times 259\ \ gCO2eq/kWh \\ &\quad + 0.367\ \ kg \times 120.09\ \ kgCO2eq/kg \\ &\quad + 0.367 \times 10^{-3} kg \times 106.00\ \ kgCO2eq/t \\ &= 23.613\ \text{ kg CO2 eq}\end{aligned} \tag{79}$$

The first line of the eq. 79, computes the CO_2 equivalent using the emission factor of the electrical power supply net of Spain in the year 2021 [Cat23a].

Then, using the *pdf* available in the Generalitat de Catalunya web page [Cat23b], the amount of waste generated for both materials used must also be taken into account (either for its recycling or its treatment). As the web explains, the recycling of plastic, if it has been previously separated from the rest[18], is equivalent to 120.09 kg CO_2 eq./ kg. Also, the treatment of the same plastic for its re-usage is equivalent to 106.00 kg CO2 equivalent per ton.

That being said, the total amount of carbon footprint estimated for the realization of this thesis is 23.613 kg CO2 eq. This amount of CO2 is approximately one-fourth of the total amount that a diesel-fueled car emits in a typical day. By that, it can be considered that the work done accomplishes sustainability and clever usage of the resources.

[17] The CO2 equivalent is a way to determine the carbon footprint.

[18] Note that the laboratory, as well as the whole university, is very strict with waste segregation and has a whole program of material recirculation [UPC23]

7 Gender and social study

This section is intended to give an insight into how this project is affected, or how it could be affected, by any kind of discrimination. It is an opportunity to pay attention to social facts that, on an engineering thesis like this, may not be the focus of attention.

In this case, the author started this project through a public offer, available to anyone who matches the academic profile, independent of any other variable. Despite that, the probability of a male student being chosen is much higher, as 80% of students in both MUEI (Màster Univeristari d'Enginyeria Industrial) and MUAR (Màster Univeristari d'Automàtica i Robòtica), are male.

In this last statistic, it may be interesting to know that, even though women are a minority, the number of female students during the last five years has remained constant. On the other hand, in this same period of time, the enrollment of new male students has decreased by 20%.

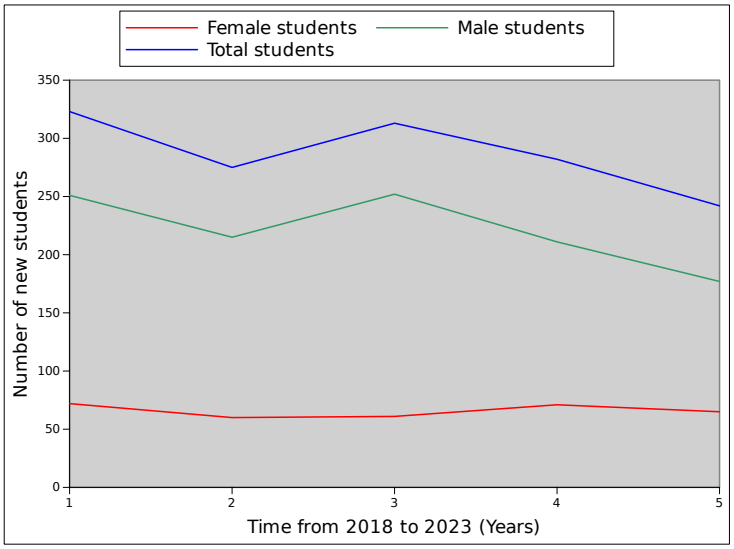


Figure 31: Last 4 years new students of MUEI by gender.

The tendency to have much more male students cannot be extrapolated to the whole Catalan university system, as there are, in fact, more female students. Another fact that the same data reveals is that, as with Fig.31, male participation in university studies is decreasing while female participation is increasing.

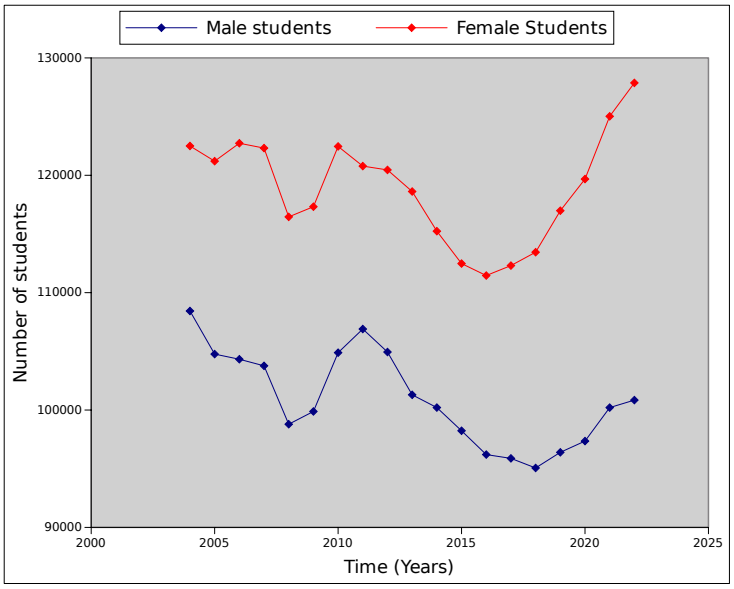


Figure 32: Number of new students in the Catalan university system.

By looking at the number of students in the 2022 school year, 100.846 were male and 127.871 were female. The increase in male students from 2021 to 2022 was 638 and 2847 for women. By checking the number of students per type of study over the same years, the number of new students in engineering was 1655 [IDE23]. Even if all male students were to go to engineering studies, the rest of 1017 would have to be women. This may mean that the tendency to have more male students than female students in engineering is changing. On this matter, one article published on the *Fundación Telefónica*, believes that, still, when it comes to choosing a professional career, students over 17 and 18 years old look for references in order to imagine what their future jobs would be like. In this case, when it comes to STEM (Science, Technology, Engineering, and Mathematics) type degrees, the popular imagination places more male characters, making them a preferable choice for male students [Mil23].

Another good social approach for this thesis is the already-mentioned Open Source philosophy. Far from privatizing knowledge, thus making it even more inaccessible to those with fewer resources, with the Open Source initiative, anyone interested in being part of a project like this thesis can do so completely free of charge. It is for sure hard for someone, or a small group of individuals, to have enough money to get certain materials, such as a basic PC, or even have the time to enroll in a project independently. Subjective variables are infinite, and to say that "if you can't, it's because you do not want", is something too pretentious. But, with the proper public initiatives, the opportunities that Open Source offers could be immense.

This work addresses a topic that is not directly related to the perspective of gender equality. The possible results, both scientific and technological, are aimed at providing an innovative technological solution in the robotics field, and in particular in quadruped robots, benefiting the whole society, regardless of gender. For this reason, neither the objectives nor the methodology have been included in the report. Regardless of this, the UPC has an Equality Plan[19] to which this work adheres.

[19]For more information about the UPC plan on equality, [UPC20] offers a good insight.

8 Conclusions

Given the scope of this project, proper conclusions are needed so they can define the work done and, in turn, establish the steps that appear to follow.

Firstly, the load finally added to the quadruped is the one proposed at the beginning of the project. The wireless connection is now possible, as the NUC can connect via Wi-Fi without the need for external Ethernet connections. Its base and chase protect its hardware and, at the same time, allow the different wires already on the robot and the new ones broadened by the implementations to have secure routing.

For the alternative power supply, a pair of lightweight LiPo batteries can now perform high-demanding current dynamics for more than 10 minutes without the need for external wires. This fact helps in the development of algorithms that aim to practice jumps, acrobatics, etc. without causing unwanted tension forces. A future improvement should be the development of both hardware and software to enable battery status monitoring.

One new sensor is now available for SOLO12. The *Intel RealSense* attached to its chassis is a huge opportunity to try any kind of exploration, identification, or state estimation routine. Its base keeps it fixed, while the inclination helps its visual camp focus on the immediate area of SOLO12's legs. Given the fact that the backspace of SOLO's top is reserved for the incorporation of new sensors without the need to modify the actual ones, future work toward the sizing, design, and implementation of a LIDAR radar could be a great opportunity.

Lastly, concerning the mechanical design, the new model of SOLO12 is available in URDF format, allowing the design of control routines that take into account the new mass and inertia values. A better intertia computation for more strict inertia requirements is also an available future path.

When it comes to the design of the state estimator, it is considered that all the objectives have been acquired. The fusion between an IMU and the OT is defined and implemented in three different ways: simulation script, ROS2 node package, and, finally, the offline script for CSV files. Both the first and third implementations assure the functionality of the filter, while the second sets a path for its usage for synchronous connections. The test conducted is considered to be even more strict than the one necessary for the usage of the filter in the quadruped, as the noise expected from a hexa-propeller drone is bigger than the one from a quadruped robot.

As the experiment with ROS2 communication has not proven to be successful, it is imperative to verify all topics published by the PX4 system. If the controller saves files containing previously filtered data, it may be possible to work with them as *IMU_msg*. If the data coming from the OT system is also split, the ROS2 nodes must be changed, as with the CSV experiment. A modification to the message published by the *Fusion node* in order to match the FLU's convention must also be looked at. Ultimately, with regard to the node, it may be necessary to implement it with C++ to enhance its speed and performance. Once this is done, future testing should be done over real-time data.

As can be seen in Appendix subsection B.6 *Tuning the filter*, it may be a good path to test the filter on the CSV script with different values for σ. Later on, this new value could be implemented on the node, and expect even better results from its performance.

9 Acknowledgements

A totes, i tots, els que formeu part d'aquest treball: gràcies.

References

[ANY23] ANYbotics. *ANYmal of ANYbotics*. 2023. URL: https://www.anybotics.com/anymal-autonomous-legged-robot/.

[aut23] PX4 autopilot. *PX4 overview definition*. 2023. URL: https://px4.io/software/software-overview/.

[Bok23] Jupyter Bokks. *Esp-idf connection*. 2023. URL: https://www.roboticsbook.org/S11_intro_state.html.

[Cat23a] Generalitat de Catalunya. *CO2 energy factor*. 2023. URL: https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/.

[Cat23b] Generalitat de Catalunya. *Guies pel càlcul del CO2*. 2023. URL: https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/.

[dro23a] Maxter drone. *Tattu R-Line Version 4.0 1300mAh 22.2V*. 2023. URL: https://maxterdrone.com/es/6s/1300-tattu-r-line-version-40-1300mah-222v-130c-6s1p-lipo-battery-pack-with-xt60-plug.html.

[dro23b] Maxter drone. *Tattu R-Line Version 4.0 750mAh 22.2V*. 2023. URL: https://genstattu.com/tattu-rline-95c-750mah-3s1p-xt30.html.

[Dyn23] Dynotis. *Why LiPo batteries?* 2023. URL: https://dynotis.semai.com.tr/en/what-is-the-advantage-of-lipo-lithium-polymer-batteries.

[Eng23] Engineersgarage. *Comparing batteries for robots*. 2023. URL: https://www.engineersgarage.com/choosing-battery-for-robots/.

[Esp23] Espressif. *Esp-idf connection*. 2023. URL: https://github.com/espressif/esp-idf.

[Gen23] Gensade. *Gens ace 3000mAh 22.2V*. 2023. URL: https://www.gensace.de/gens-ace-3000mah-22-2v-60c-6s1p-lipo-battery-pack-with-ec5-plug.html.

[GIT23] GIT. *GitHub about*. 2023. URL: https://github.com/about/open_robot_actuator_hardware.

[Gui23] Erico Guizzo. *DISNEY robot with AIR-Water Actuators*. 2023. URL: https://spectrum.ieee.org/disney-robot-with-air-water-actuators.

[IDE23] IDESCAT. *Number of new students per type of studies*. 2023. URL: https://www.idescat.cat/indicadors/?id=aec&n=15730.

[INN23] INNOVA3D. *Print*. 2023. URL: https://innova3d.es/presupuesto-online-impresion-3d/.

[Int23a] Intel. *Intel NUC7i7BNH*. 2023. URL: https://www.intel.es/content/www/es/es/products/sku/95065/intel-nuc-kit-nuc7i7bnh/specifications.html.

[Int23b] Intel. *Intel RealSense Depth Camera*. 2023. URL: https://www.intelrealsense.com/depth-camera-d435i/.

[IRI23] IRI. *Institut de Robòtica i Informàtica*. 2023. URL: https://arxiv.org/abs/1905.04254.

[JA18] J.Deray J. Solà and D. Atchuthan. *Web-page of Manif*. 2018. URL: https://artivis.github.io/manif/.

[Kau+19] Nathan Kau et al. *Stanford Doggo: An Open-Source, Quasi-Direct-Drive Quadruped*. 2019. arXiv: 1905.04254 [cs.RO].

[LOR23] Parler LORD. *IMU Microstrain 3DM-CX5-AHRS*. 2023. URL: https://www.microstrain.com/inertial-sensors/3dm-cx5-25.

[Mil23] Sainz Milagros. *Study about students tendency*. 2023. URL: https://www.fundaciontelefonica.com/cultura-digital/publicaciones/590/#close.

[MIT23] MIT. *Spot of MIT*. 2023. URL: https://https://www.bostondynamics.com/products/spot.

[Moh+17] Kartik Mohta et al. *Fast, autonomous flight in GPS-denied and cluttered environments*. Dec. 2017. DOI: 10.1002/rob.21774. URL: https://doi.org/10.1002%5C%2Frob.21774.

[New23] Josh Newmans. *Articulated Robotics project, on batteries estimation*. 2023. URL: https://articulatedrobotics.xyz/mobile-robot-5-power-theory/.

[ODR23a] ODRI. *Ecoder AEDT-9810-Z00*. 2023. URL: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware/blob/master/mechanics/actuator_module_v1/details/details_encoder.md.

[ODR23b] ODRI. *Esp-idf connection*. 2023. URL: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware/blob/master/mechanics/quadruped_robot_12dof_v1.1/README.md#quadruped-robot-12dof-v11.

[ODR23c] ODRI. *GitHub page of SOLO*. 2023. URL: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware.

[ODR23d] ODRI. *Open Dynamic Robot Initiative*. 2023. URL: https://open-dynamic-robot-initiative.github.io/.

[ODR23e] ODRI. *SOLO12*. 2023. URL: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware/blob/master/mechanics/quadruped_robot_12dof_v1/README.md#quadruped-robot-12dof-v1.

[ODR23f] ODRI. *TI and Micro Driver*. 2023. URL: https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware/blob/master/electronics/ti_electronics/README.md#texas-instruments-evaluation-board-electronics.

[Ope23] Open-Source. *ROS web-page*. 2023. URL: https://www.ros.org/.

[PAL23] PAL. *PAL robotics*. 2023. URL: https://pal-robotics.com/es/.

[RB23] RB. *RB-VOGUI Mobile Robot Manipulator*. 2023. URL: https://robotnik.eu/products/mobile-robots/rb-vogui-en/#more-versions.

[ROS20] ROS. *Rviz, ROS tool to visualize interactions of the robot's frames*. 2020. URL: http://wiki.ros.org/rviz/UserGuide.

[SDA21] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. 2021. arXiv: 1812.01537 [cs.RO].

[sin23] sinotech. *BLDC on robotics*. 2023. URL: https://sinotech.com/blog/bldc-motors-humanoid-robotic-applications/.

[Sou23a] Open Source. *3D Rotation equivalent web page*. 2023. URL: https://www.andre-gaschler.com/rotationconverter/.

[Sou23b] Open Source. *PX4 web-page*. 2023. URL: https://px4.io/.

[Sut13] Sutrabla. *Gimbal Lock explanation*. 2013. URL: https://www.youtube.com/watch?v=syQnn_xuB8U.

[Sut18] Sutrabla. *Quaternion graphical video*. 2018. URL: https://www.youtube.com/watch?v=zjMuIxRvygQ.

[Tac+20] Joanna Taczała - Warga et al. *Comparison of 3D printing MJP and FDM technology in dentistry*. 2020. DOI: 10.5604/01.3001.0013.9504. eprint: 10.1140/epjst/e2015-50288-8.

[Tec19] Massachusetts Insitute of Technology. *Cheetah, the MITs quadruped robot*. 2019. URL: https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304.

[Tra18] Jack Trainer. *The Apolo project and the Kalman filter*. 2018. URL: https://www.lancaster.ac.uk/stor-i-student-sites/jack-trainer/how-nasa-used-the-kalman-filter-in-the-apollo-program/.

[UPC20] UPC. *UPC equalityplan*. 2020. URL: https://igualtat.upc.edu/ca/shared/third-upc-gender-equality-plan.pdf.

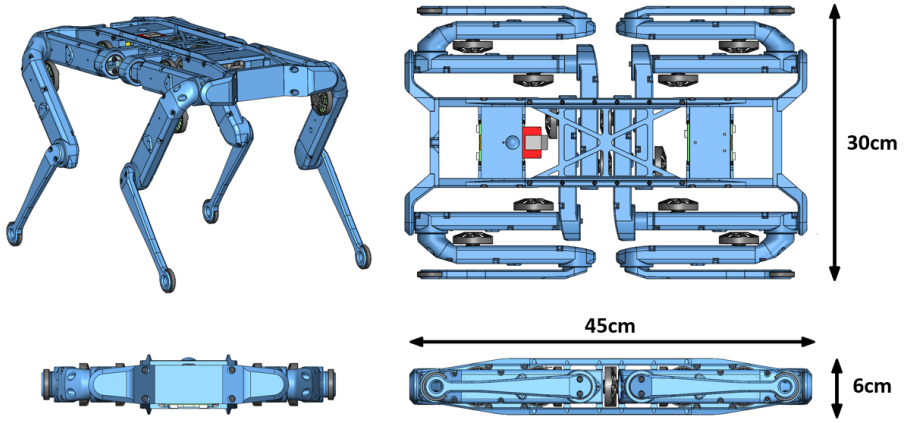
[UPC23] UPC. *Programa Recircula*. 2023. URL: https://reutilitza.upc.edu/ca.
[Vol23] Volkel. *Vcoil kit*. 2023. URL: https://shop4fasteners.co.uk/v-coil-metric-thread-repair-insert-kits.html.

A Hardware implementation details

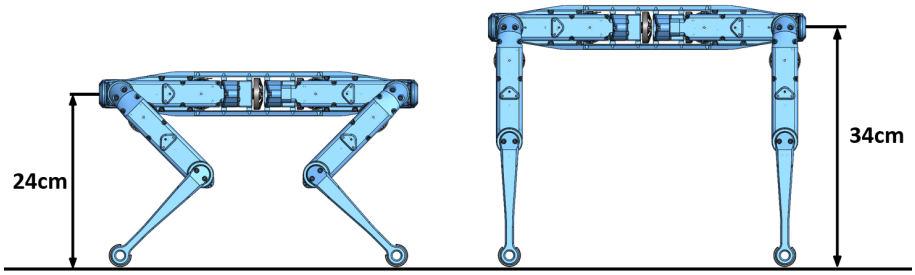
This part of the appendix aims to give extra detail on the hardware's implementation.

A.1 SOLO12's initial size

As mentioned in the main text, knowing SOLO's dimensions in specific configurations such as totally folded, half standing, and totally straight legs standing is useful. Figure A.1.1 gives the values for the last three.



(a) Solo 12 X-Y-Z Dimensions (cm).



(b) Solo 12 height on two configurations.

Figure A.1.1: General dimensions of Solo 12

A.2 Control scheme

As explained in the footnote, in order to compute the capacity of the batteries, a control algorithm is required.

In this case, the block scheme is the one on Fig. A.2.1. The values K_p and K_d are represented in the first box, and they represent the gains of a PD controller. The K_f, feedforward term, is the one represented as τ^*.

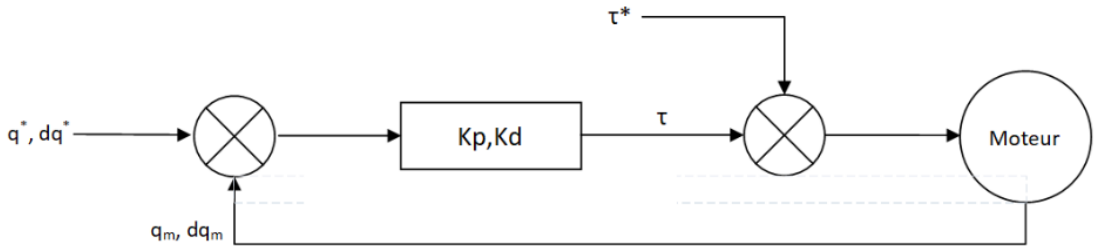


Figure A.2.1: Block scheme of the control loop for the capacity estimation.

A.3 Camera *Intel* consumption

3.7.7 Vision Processor D4 Board Power Requirements

The Vision Processor D4 Board is powered through VBUS power of the USB connector. The Vision Processor D4 Board in turn power sources the stereo depth module.

Table 3-39. Vision Processor D4 Board Power Requirements

Parameter		Min	Nom	Max	Unit
VCC	Supply Voltage	4.75	5V	5.25V	V
ICC	Supply Current			700	mA
	Supply Voltage Ramp Rate	0.5		5	ms

Figure A.3.1: Current consumption of the camera.

A.4 New assembly parts dimensions

This section is dedicated to show the new features general dimensions.

Battery chase

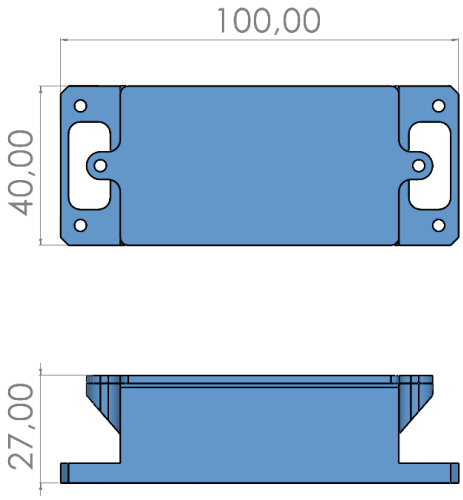


Figure A.4.1: General dimensions of the battery chase in mm.

Power distribution chase

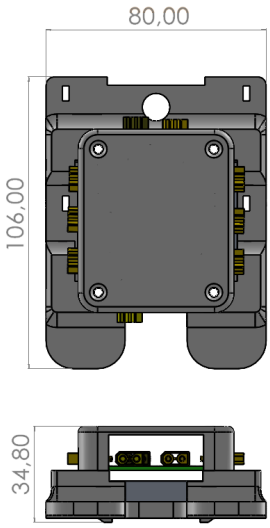


Figure A.4.2: General dimensions of the power distribution chase in mm.

NUC chase

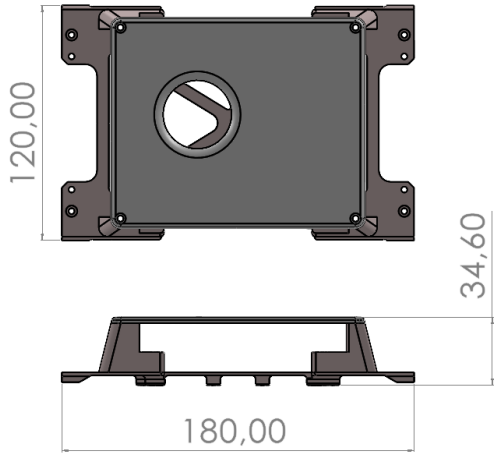


Figure A.4.3: General dimensions of NUC chase in mm.

Camera chase

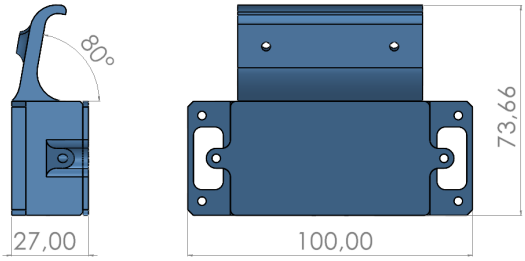


Figure A.4.4: General dimensions of camera and battery chase in mm.

B Estimator design specifics

As with the hardware implementation, this second part of the appendix gives an extended insight into a more detailed mathematical description, some figures to illustrate experiments, and, information that describes values as the IMU's noise.

B.1 Jacobian computation

J_x^x Related Jacobians

$$\mathbf{J}^R_{\omega_b} = \mathbf{J}^R_{\omega}\mathbf{J}^{\omega}_{\omega_b} \tag{B.1}$$

where,

$$\mathbf{J}^R_{\omega} = \mathbf{J}^{R\oplus(\omega*dt)}_{(\omega*dt)} \tag{B.2}$$

$$\mathbf{J}^{\omega}_{\omega_b} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{B.3}$$

$$\mathbf{J}^v_R = \mathbf{J}^v_a\mathbf{J}^a_R \tag{B.4}$$

where,

$$\mathbf{J}^v_a = \begin{pmatrix} dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \end{pmatrix} \tag{B.5}$$

$$\mathbf{J}^a_R = -R[v]_x \tag{B.6}$$

$$\mathbf{J}^v_{a_b} = \mathbf{J}^v_a\mathbf{J}^a_{a_b} \tag{B.7}$$

where,

$$\mathbf{J}^v_a = \begin{pmatrix} dt & 0 & 0 \\ 0 & dt & 0 \\ 0 & 0 & dt \end{pmatrix} \tag{B.8}$$

with a second chain rule,

$$\mathbf{J}^a_{a_b} = \mathbf{J}^a_a\mathbf{J}^a_{a_b} \tag{B.9}$$

where,

$$\mathbf{J}^a_a = \mathbf{R} \tag{B.10}$$

$$\mathbf{J}^a_{a_b} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{B.11}$$

$$\mathbf{J}^p_R = \mathbf{J}^p_a\mathbf{J}^a_R \tag{B.12}$$

$$\mathbf{J}^p_a = 0.5 * dt^2 * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.13}$$

and $\mathbf{J}^a_R$ already defined in eq. B.6

$$\mathbf{J}^p_{a_b} = \mathbf{J}^p_a\mathbf{J}^a_{a_b} \tag{B.14}$$

where, $\mathbf{J}^p_a$ already defined in eq. B.13 and $\mathbf{J}^a_{a_b}$ in eq. B.11

$\mathbf{J}_u^x$ related Jacobians

$$\mathbf{J}_{\omega_m}^R = \mathbf{J}_{\omega}^R \mathbf{J}_{\omega_m}^{\omega} \tag{B.15}$$

with a second chain rule,

$$\mathbf{J}_{\omega}^R = \mathbf{J}_{\omega}^R \mathbf{J}_{\omega}^{\omega * dt} \tag{B.16}$$

where,

$$\mathbf{J}_{\omega}^R = \mathbf{J}_{\omega * dt}^{R \oplus (\omega * dt)} \tag{B.17}$$

$$\mathbf{J}_{\omega}^{\omega * dt} = dt * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.18}$$

and,

$$\mathbf{J}_{\omega_m}^{\omega} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.19}$$

$$\mathbf{J}_{a_m}^{v} = \mathbf{J}_{a}^{v} \mathbf{J}_{a_m}^{a} \tag{B.20}$$

where $\mathbf{J}_a^v$ already defined in eq. B.8, and a second chain rule,

$$\mathbf{J}_{a_m}^{a} = \mathbf{J}_{a}^{a} \mathbf{J}_{a_m}^{a} \tag{B.21}$$

where $\mathbf{J}_a^a$ already defined in eq. B.10 and,

$$\mathbf{J}_{a_m}^{a} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.22}$$

$$\mathbf{J}_{a_m}^{p} = \mathbf{J}_{a}^{p} \mathbf{J}_{a_m}^{a} \tag{B.23}$$

where $\mathbf{J}_a^p$ is already defined in eq. B.13 and $\mathbf{J}_{a_m}^a$ is already defined in eq. B.21.

$\mathbf{J}_x^z$ related Jacobians

$$\mathbf{J}_{x}^{z} = \mathbf{J}_{e}^{z} \mathbf{J}_{x}^{e} \tag{B.24}$$

where,

$$\mathbf{J}_{e}^{z} = \begin{pmatrix} \mathbf{J}_{R_e}^{R_z} & 0 \\ 0 & \mathbf{J}_{p_e}^{p_z} \end{pmatrix} \tag{B.25}$$

and,

$$\mathbf{J}_{R_e}^{R_z} = -\mathbf{J}_{l}^{-1}(R_z) \tag{B.26}$$

$$\mathbf{J}_{p_e}^{p_z} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{B.27}$$

B.2 IMU noise specifications

For the decision of what σ value to use for the generation of the simulated trajectories, the IMU specification data sheet is used.

Inertial Measurement Unit (IMU) Sensor Outputs			
	Accelerometer	**Gyroscope**	**Magnetometer**
Measurement range	±8 g (standard) ±2 g, ±4 g, ±20 g, ±40 g (optional)	300°/sec (standard) ±75, ±150, ±900 (optional)	±8 Gauss
Non-linearity	±0.02% fs	±0.02% fs	±0.3% fs
Resolution	<0.1 mg	<0.003°/sec	--
Bias instability	±0.04 mg	8°/hr	--
Initial bias error	±0.002 g	±0.04°/sec	±0.003 Gauss
Scale factor stability	±0.03%	±0.05%	±0.1%
Noise density	20 µg/√Hz (2 g)	0.005°/sec/√Hz (300°/sec)	400 µGauss/√Hz
Alignment error	±0.05°	±0.05°	±0.05°
Adjustable bandwidth	225 Hz (max)	250 Hz (max)	--
Offset error over temperature	0.06% (typ)	0.04% (typ)	--
Gain error over temperature	0.03% (typ)	0.03% (typ)	--
Scale factor non-linearity (@ 25°C)	0.02% (typ) 0.06% (max)	0.02% (typ) 0.06% (max)	±0.0015 Gauss
Vibration induced noise	--	0.072°/s RMS/g RMS	--
Vibration rectification error (VRE)	0.03%	0.001°/s/g2 RMS	--
IMU filtering	Digital sigma-delta wide band anti-aliasing filter to digital averaging filter (user adjustable) scaled into physical units.		
Sampling rate	1 kHz	4 kHz	100 Hz
IMU data output rate	1 Hz to 1 kHz		

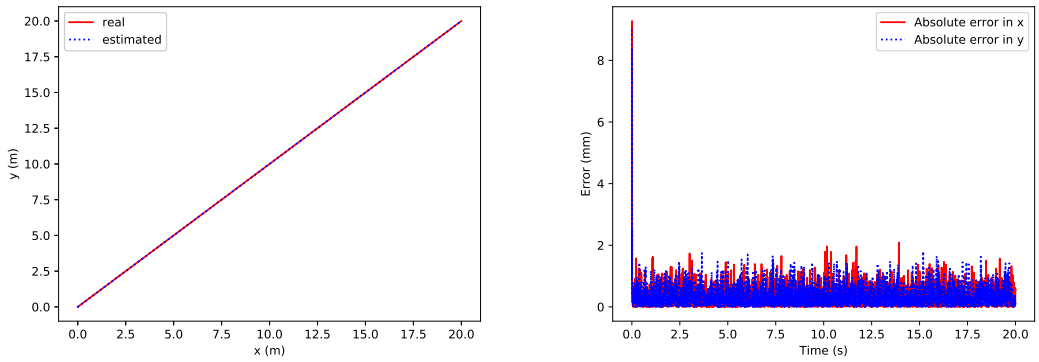
Figure B.2.1: IMU errors specifications.

The OT's errors are expected to be millimetric and that is known because, before broadcasting the position or the orientation, the calibration results show the expected errors.

B.3 Simulated trajectories

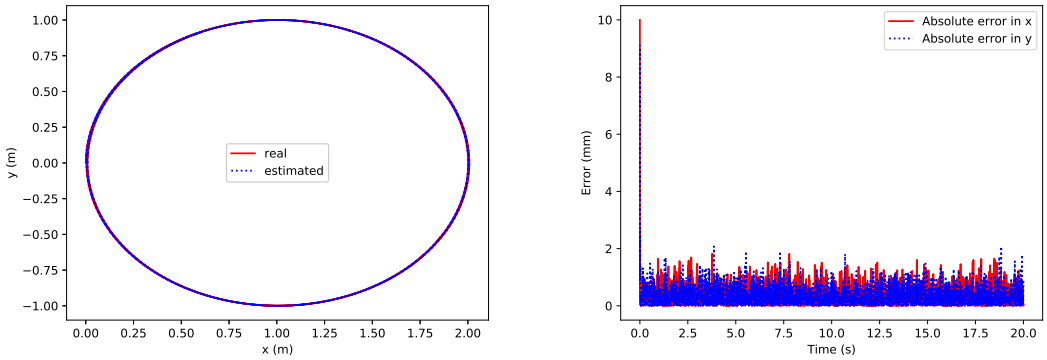
As explained in the subsection 3.4 *Simulation experiment*, three different trajectories are simulated in order to test the LEKF. The two remaining that are not shown are the ones that follow.

For all the simulations, the simulated time is 20 s. This time is decided to be representative enough because the estimation is expected to converge far before this time.



(a) Straight line trajectory. (b) Absolute error.

Figure B.3.1: Estimated trajectory compared to real one (straight line case).



(a) Circular trajectory. (b) Absolute error.

Figure B.3.2: Estimated trajectory compared to real one (circle curve case).

B.4 Opti Track (OT)'s calibration steps

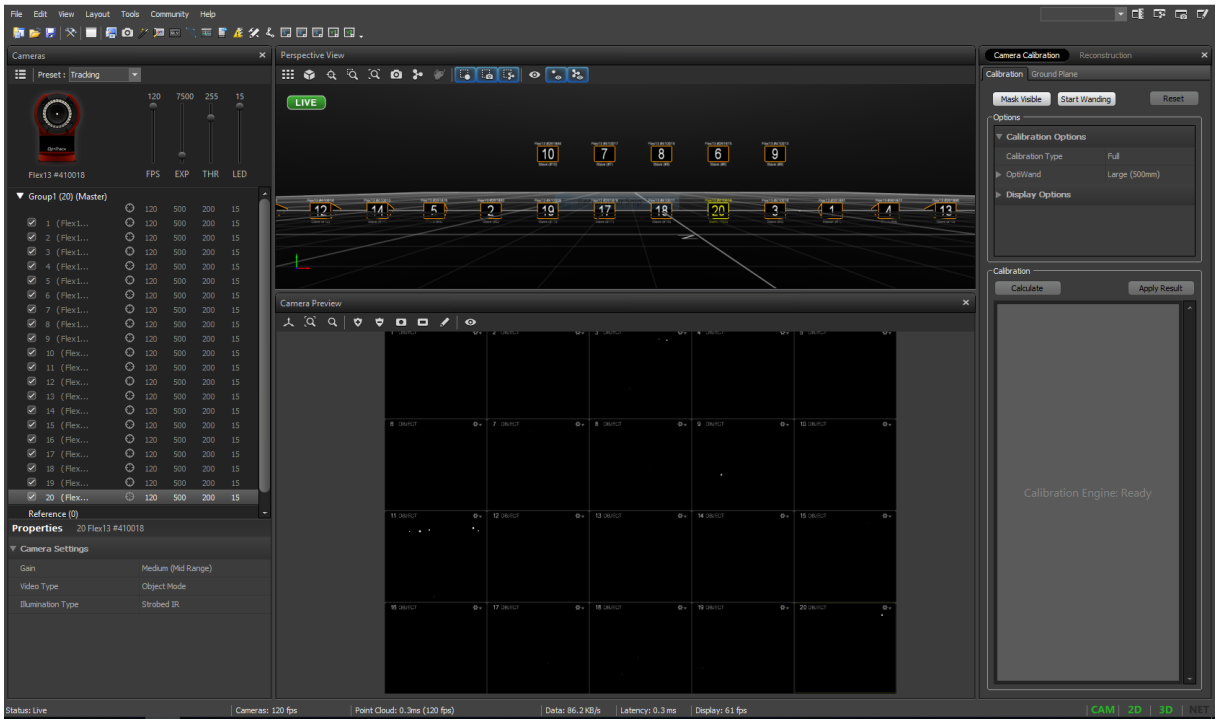


Figure B.4.1: Visibly white dots on dark squares means the existence of unmasked points.

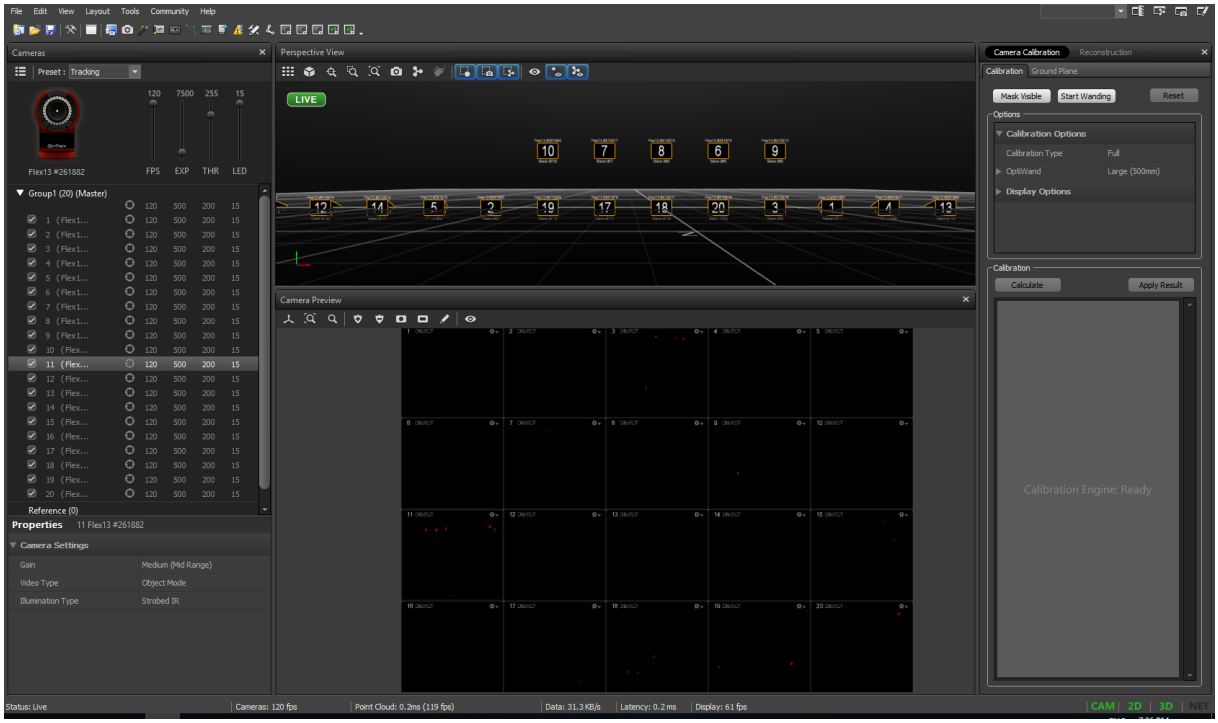


Figure B.4.2: Once masked, white dots become red, meaning they are not considered in the calibration.

Figure B.4.3: Wanding process for the cloud of points.

Figure B.4.4: Offset ground to set the ground plane.

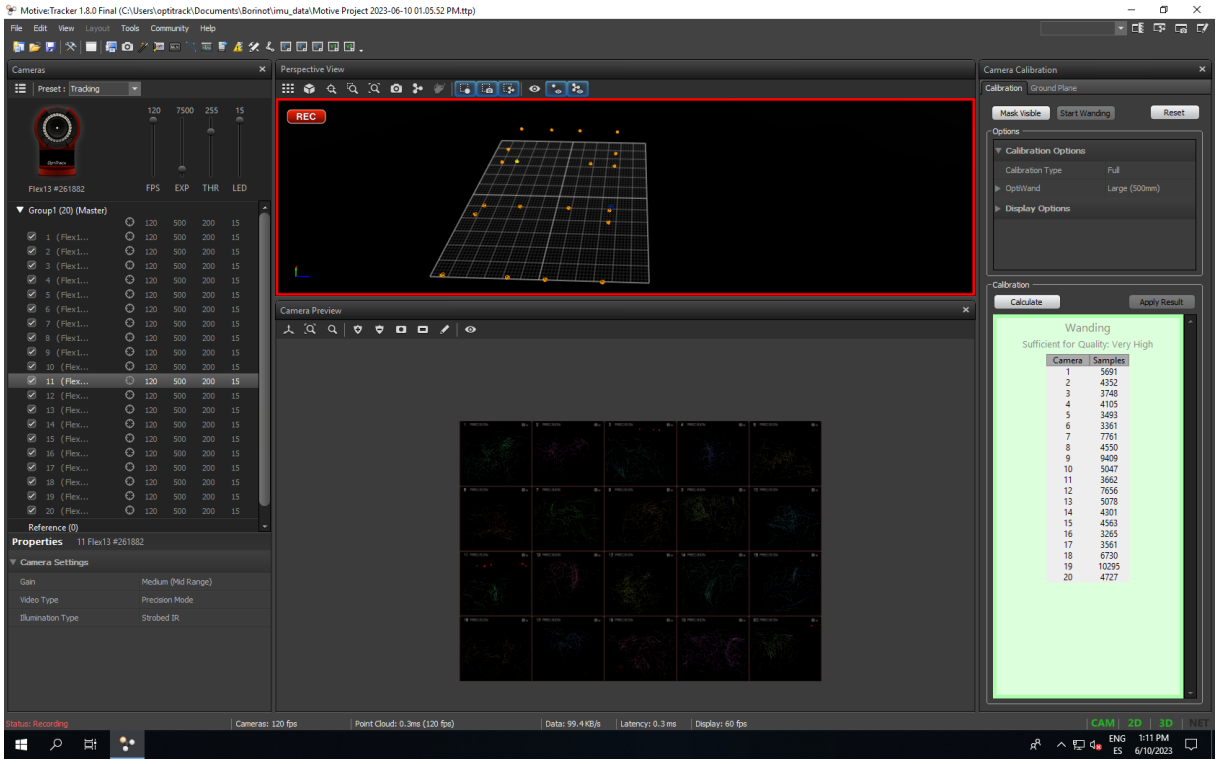


Figure B.4.5: Calibration done and its result values.

B.5 New required Jacobians

As explained in the second approach of the experimentation with the fusion filter, the correction step is now split into a correction on position and a correction on orientation. This fact requires two new definitions: one for the transform matrices used in 72 and for the Jacobians of the noise in the OT data.

First, both the orientation and the position, even though they could be used as a matrix and a vector, are defined as two separate transformations.

$$T_{BP\ R}^{OP} = \begin{pmatrix} R_{OP}^{O} & 0_{(3,1)} \\ 0_{(1,3)} & 1 \end{pmatrix} \tag{B.28}$$

$$T_{BP\ p}^{OP} = \begin{pmatrix} 0_{(3,3)} & P_{OP}^{O} \\ 0_{(1,3)} & 1 \end{pmatrix} \tag{B.29}$$

Then, the equivalent equations for the equation 72, are eq. B.30 and B.31.

$$T_{BR}^{O} = T_{OP}^{O}\, T_{BPR}^{OP}\, T_{B}^{BP} = T_{OP}^{O}\, T_{BPR}^{OP}\, ({T_{BP}^{B}}^{-1}) \tag{B.30}$$

$$T_{Bp}^{O} = T_{OP}^{O}\, T_{BPp}^{OP}\, T_{B}^{BP} = T_{OP}^{O}\, T_{BPp}^{OP}\, ({T_{BP}^{B}}^{-1}) \tag{B.31}$$

Once the transformations are separately defined, the new Jacobians are the split version of eq. B.24.

For the orientation,

$$J_{yR}^{z} = \begin{pmatrix} J_{R_m}^{R_z} & J_{p_m}^{R_z} \\ J_{R_m}^{p_z} & J_{p_m}^{p_z} \end{pmatrix} = \begin{pmatrix} J_{R_m}^{R_z} & 0 \\ 0 & 0 \end{pmatrix} \tag{B.32}$$

where,

$$J_{R_m}^{R_z} = J_r^{-1}(R_z) \tag{B.33}$$

And from the position,

$$J_{yp}^{z} = \begin{pmatrix} J_{R_m}^{R_z} & J_{p_m}^{R_z} \\ J_{R_m}^{p_z} & J_{p_m}^{p_z} \end{pmatrix} = \begin{pmatrix} 0 & J_{p_m}^{p_z} \\ 0 & 0 \end{pmatrix} \tag{B.34}$$

where,

$$J_{p_m}^{p_z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.35}$$

The equivalent versions of the measurements must be also done for the estimation.

B.6 Tuning the filter

Even with the good results achieved in the subsection 3.6 *Results*, the definition of the filter is expected to be better if it works with the proper values for the signal noise. In this case, the results plotted in all Figures of the same section 3.6 use the values defined in Table 11 as the standard deviation for the signals.

Although it may be true that altering the *sigma* that determines the covariance matrices of the noises is necessary, doing so in the proper way is not simple. In this case, by observing the input acceleration on x for a measurement of 10 s, the standard deviation is approximately 0,25 m/s^2. Assuming the white noise is the same in all coordinates, the standard deviation for the frequency rate of the IMU may be computed like in eq. B.36. So its value could be changed to $5.6e^{-3}$ m/s^2.

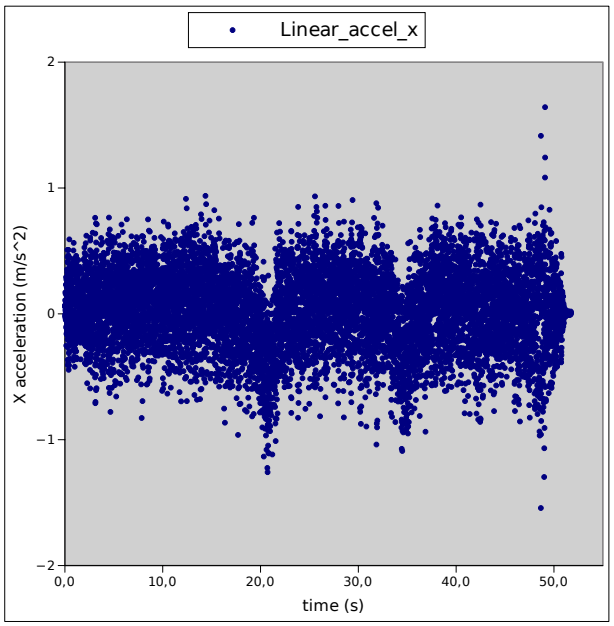


Figure B.6.1: Linear acceleration from the IMU.

$$\sigma_{a_x\ IMU\ freq.} = 0.25 * \sqrt{\frac{0.005}{10}} = 0.s0056m/s^2 \tag{B.36}$$

The same procedure could be followed for the ω white noise (eq. B.37, obtaining a value of $5,6e^{-4}\ rad/s$.

$$\sigma_{\omega_x\ IMU\ freq.} = 0.025 * \sqrt{\frac{0.005}{10}} = 0.00056rad/s \tag{B.37}$$

For the bias drift, it would be hard to know, as it is not that visual, so, for example, both initial values could be multiplied by 10^2 as it is considered more realistic.

Finally, for the precision of the OT system, as it is a highly trusted measurement, the same values as the initial one could be kept. The new hypothetical values are expressed in Fig.17.

Noise	σ
$acceleration_{whitenoise}$	$5.6 \cdot 10^{-3} m/s^2$
$\omega_{whitenoise}$	$5.6 \cdot 10^{-4} rad/s$
$a_{bias\ \ drift}$	$4.0 \cdot 10^{-2} m/s^2$
$\omega_{bias\ \ drift}$	$3.9 \cdot 10^{-3} rad/s$
$Orientation_{white_noise}$	$6.0 \cdot 10^{-3} rad$
$Position_{white\,noise}$	$3.0 \cdot 10^{-4} m$

Table 17: New initial values for the noises in IMU and OT.

Plotting the same figure as in Fig.25, but with the new values, bring the Fig.B.6.2. This proposed change does not improve the difference between the two estimations, nor does it improve the continuity between predictions and corrections.

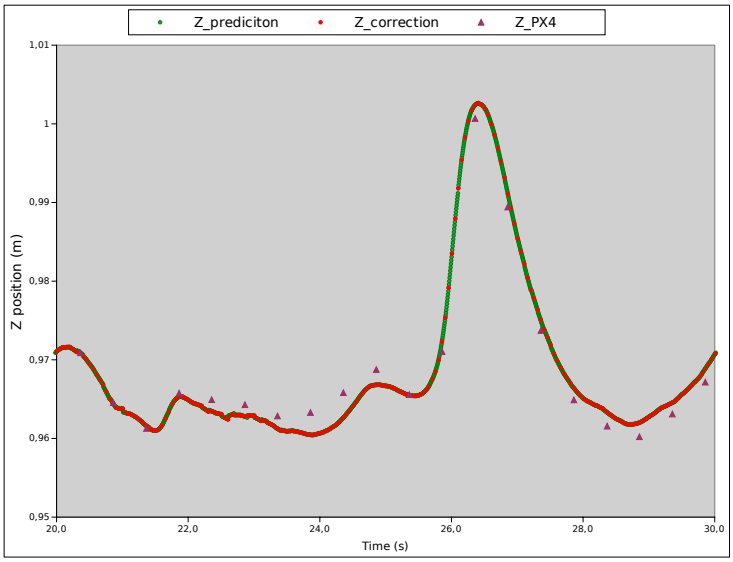


Figure B.6.2: Comparison of Z coordinate with PX4 estimation, for new set of values.

It is expected to not find a proper set of values by simply trying new ones, but the lack of time imposes that this thesis keeps the first values as the best ones.