POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Development and implementation of a customized flight control system for an Unmanned Aerial Vehicle operating in a connected Industry environment

Supervisors

Prof. Alessandro RIZZO

Prof. Marina INDRI

Dr. Stefano PRIMATESTA

Candidate

Andrea COLUCCI S278858

October 2021

Abstract

The forthcoming industrial environments will require a high level of automation to be flexible and adaptive enough to comply with the increasingly faster and low-cost market demands. Autonomous and collaborative robots will have an ever-greater role in this context. In this view, the FIXIT project aims at providing an interactive support for the human operator, within an industrial or logistic environment compliant with the Industry 4.0 requirements. The objective of this thesis is to develop and implement a customized flight control system for a commercial UAV (Unmanned Aerial Vehicle). The system is going to be implemented on the aerial node of the robotic FIXIT platform (a multi-element robotics platform consisting of a mobile robot and an aerial robotic system). The system interacts with a path planning and obstacle detection algorithm running on a separate companion computer located on board. The positioning system for the robot is based on an UWB (Ultra Wide Band) localization system mounted on the CIM4.0 digital pilot line. Many papers and projects implement the UWB as a localization system but just a few of them actually use it on UAVs. The initial analysis of the state of art was followed by two phases. In the first one, some simulations have been made in which the UAV was able to perform missions and to navigate in the simulated environment through way-points computed by means of an obstacle avoidance algorithm. In the second one, the real tests showed that the drone behaves like in the simulation tests, but with some uncertainties and errors that have to be taken into account. Some tests were made outdoor. With a correct calibration and a correct configuration of the Pixhawk board, the UAV was able to takeoff, follow a static defined path and land. Some other tests were made indoor. The UWB positioning system and the chosen sensors have been widely tested and they revealed to be the best ones for the environment in which the UAV was tested.

Table of Contents

List of Figures

Acronyms

ROS
Robotic Operating System

UWB
Ultra-wideband

UAV
Unmanned Aerial Vehicle

GPS
Global Positioning System

ToF
Time of Flight

TDoA
Time Difference of Arrival

PDoA
Phase Difference of Arrival

GCS
Ground Control Station

TH-SS
Time-hopping spread spectrum

PR

Pseudo random

BPSK

Binary phase shift keying

TH-BPSK

Time-hopping phase shift

TH-PPM

Time-hopping pulse position modulation

MSK

Minimum-shift keying

SITL

Software in the loop

HIL

Hardware in the loop

Chapter 1

Introduction

1.1 Objective of the thesis

Autonomous and collaborative robots have an important role in the current industry context. Their role is to provide an interactive support for the human operator, within an industrial or logistic environment compliant with the Industry 4.0 requirements. This thesis is part of a bigger project, the Fixit project.

Figure 1.1: FIXIT

The objective of the thesis is to customize an already existing flight control system for a commercial UAV operating in connected industrial environment. The idea is to use an UAV for supporting some maintenance procedures in an industrial environment. The UAV shall flight autonomously both in indoor and outdoor environments. The first part of the thesis is developed on a DJI F450 frame.

Figure 1.2: DJI F450 UAV

Afterwards, a fully customized UAV has been used. UAVs are used for different purposes, such as accessing in high or narrow places or they can be implemented, for example, in application in which some strictly safety requirements are present. The possibility of integrating different kind of sensors on the UAV makes it really flexible and adaptable in very different scenarios.

Figure 1.3: Customized UAV

Autonomous UAVs are mainly divided in two categories of systems: open-source and private.
In this thesis an open-source application is developed. The first step is to choose the open-source firmware for the Pixhawk board. Open-source firmware can be divided in 2 different categories based on the automation level that they can provide: fully autonomous firmware are those not requiring human intervention at all (or just partially, according to the flight mode that is chosen), whereas non-autonomous firmware are those requiring human control for any operation (no level of automation). There are many different open-source firmware available on the market. The most important are:

- ArduPilot
- PX4
- Paparazzi
- FlexiPilot
- SmartAP
- Armazila
- SLUGS
- iNav

Both ArduPilot and PX4 are valid choices. After having tested both, ArduPilot was used for compatiblity purposes.
There are also many problems related to the use of UAVs. Some of these will be analyzed in this thesis.
The first problem is related to the possibility of having a correct and precise localization in GPS-denied environments. In an indoor environment the GPS (Global Positioning System) does not work. At the state of the art, there are many possibilities to overcome this problem.
Some UAVs implement a Vision-based Navigation system, that is a system that allows to estimate the Pose (Position and Orientation) of the UAV. Nowadays, cameras and other exteroceptive sensors are on board of a large variety of automatic platforms, such as UAVs, space exploration probes and missiles. However, apart from this latter application, they are mostly used as payload and not to pilot the vehicle itself.

Some studies focused on the use of computer vision for UAV perception to navigate through the environment and model it. These methods can reach an accuracy in the order of millimetres but are really expensive. This function is typically needed at low altitude in unknown or GPS-denied conditions. The measurements from exteroceptive sensors can then be processed to obtain information about the motion of the UAV, or the 3D structure of the environment. Usually, these applications start with a vision-based closed control loop, where image-based navigation is integrated to UAV control. Then, the focus shifts on proper motion estimation techniques, like map-less relative terrain navigation or map-based GPS alternatives.
An alternative is given by the use of an UWB (Ultra Wide Band). It is a localization system used to have accurate coordinates in indoor environments. UWB technology provides an excellent means for wireless positioning due to its high resolution capability in the time domain. Its ability to resolve multi-path components makes it possible to obtain accurate location estimates without the need for complex estimation algorithms. Since the UAV can work only if it receives the signal from the GPS, it is necessary to find a way to send the position computed from the UWB to the Pixhawk board. This has been made by means of ROS, MAVProxy, Mavlink and MAVROS. ROS is the acronym for "Robotic Operating System" and it is a collection of open-source software frameworks for robot software development. It provides many services, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. In this thesis it is used to send messages across different devices, such as Pixhawk board, UWB, Intel Realsense and Jetson Nano.
It is possible to directly communicate with the autopilot through a service called MAVProxy. It is a fully-functioning GCS (Ground Control Station) for UAVs, designed as a minimalist, portable and extendable GCS for any autonomous system supporting the MAVLink protocol (such as one using ArduPilot). MAVProxy is a powerful command-line based "developer" ground station software. It can be extended via add-on modules, or complemented with another ground station, such as Mission Planner, APM Planner 2 or QGroundControl to provide a graphical user interface.
To use both MAVProxy and ROS together MAVROS has been used. MAVROS is a ROS "node" that allows to convert MAVLink messages in ROS topics, allowing the ArduPilot vehicles to communicate with ROS.

The positioning problem is solved by sending the coordinates measured from the UWB over a ROS topic, using MAVROS. Once these coordinates are published on the specific ROS topic, a MAVROS fake gps plugin is implemented to send GPS coordinates to the autopilot. In this way it is possible to use the UAV indoor as it would be outdoor.
Another problem is the altitude estimation. In indoor environments, all the sensors used to measure the altitude return a poor measurement of the height. For critical indoor applications and also to guarantee safety requirements, a good estimation of the altitude is necessary. To overcome this problem two ToF (Time of Flight) sensors are implemented. The first one is put on top of the UAV while the other one on the bottom. After having measured the data coming from the UWB and the distance measured from the two sensors, a sensor fusion using the Extended Kalman Filter is performed. The Pixhawk board handles this process of fusing the measurements coming from different sensors.
In many UAVs applications, it is required to navigate in unknown environments where moving or stationary obstacles have to be detected and avoided. UAVs must have the ability to autonomously plan trajectories in order to avoid collisions. In the literature, many solutions have been proposed for implementing the path planning algorithm of UAVs. A valid path is obtained considering the mission constraints, the vehicle characteristics and the mission environment and combining these elements with the mission tasks. The UAV class is important because different classes can have completely different dynamic and kinematic characteristics.
The basic idea of a Collision Avoidance System (CAS) involves monitoring the environment considering any possible scenario, sense and detect possible obstacles and avoid them. In a CAS, it is possible to distinguish five key functions:

- Sensing
- Detection
- Awareness
- Escape trajectory
- Maneuver realizations

The sensing function refers to the ability of the UAV to monitor the surrounding environment and collect the appropriate information for any possible

situation.
The detection function, instead, is the ability of the system to acquire the sensed data, process it and extract useful information about the possible incoming source of collision.
The escape trajectory function is based completely on the path planning algorithm implemented, it must be carefully chosen based on the environment and on future UAV tasks. The analysis on the possible algorithm to implement constitutes a basic step in the design and implementation of the CAS, this choice determines the type of sensors that have to be implemented on the UAV.
This topic has been widely analyzed by a colleague working on this project. The chosen algorithm is the dynamic RRT*. The following step was to design an effective collision avoidance algorithm based on dynamic RRT* (Rapidly-exploring Random Tree "star") used in a 3D reconstruction of the surroundings thanks to the Intel Realsense depth camera, with the requirements to be real-time implementable and runnable by the onboard embedded hardware. Assuming the position of the UAV is controlled and the planner generates a global path, then the purpose of the RRT* algorythm is to generate dynamically a series of way-points to avoid obstacles.
At the end of this thesis, simulations and tests results are provided. For semplicity, it is possible to divide the tests in: SITL (Software in the loop), HIL (Hardware in the loop) and real tests.
The SITL simulator allows to run a simulation of the UAV without any hardware. It gives a native executable, in C++, that allows to test the behaviour of the code without any hardware. When running in SITL the sensor data comes from a flight dynamics model in a flight simulator. ArduPilot has a wide range of vehicle simulators built in, and can interface to several external simulators. This allows ArduPilot to be tested on a very wide variety of vehicle types.

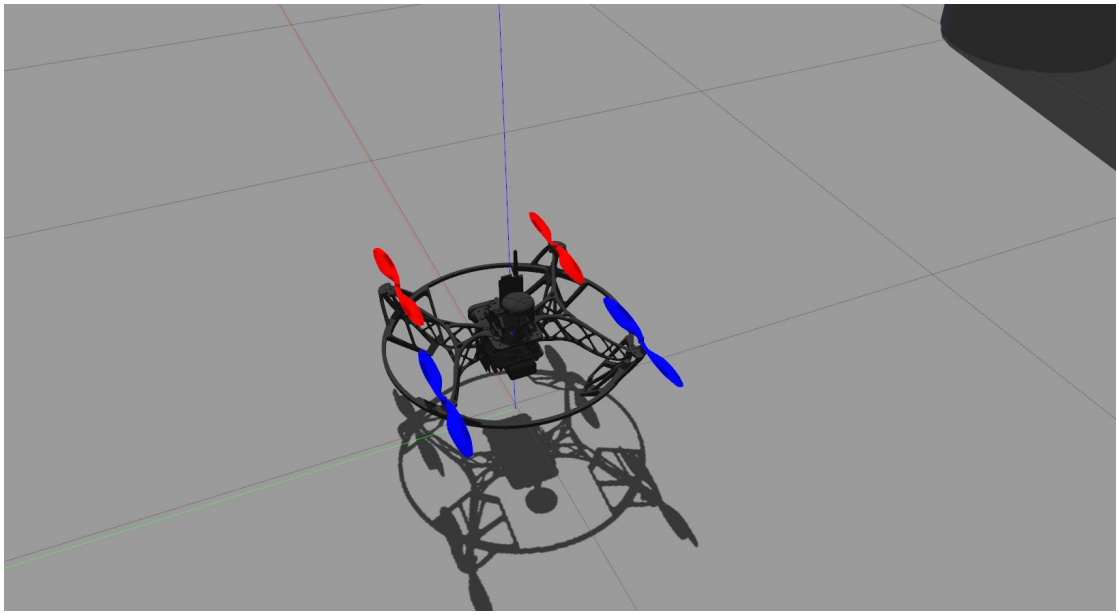

Figure 1.4: UAV simulation in Gazebo.

The HITL simulation replaces the vehicle and the environment with a simulator (the Simulator has a high-fidelity aircraft dynamics model and environment model). The physical autopilot hardware is configured exactly as for flight and connects to the computer running the simulator (rather than the aircraft). In this way it is possible to tune the parameters of the UAV. The real test were made without any simulation involved. The UAV was able to fly both indoor and outdoor. The main differences between the indoor tests and the outdoor tests were given just by the accuracy of the sensors and by the different technologies involved. But, in both cases, the UAV showed a similar behaviour with respect to the simulations.

1.2 Structure of the thesis

The thesis has the following structure:

- Chapter 1: In the first part of the thesis there is an introduction of the FIXIT project and a brief explanation of what is its role in an industrial environment. Then, the main problems related to the development of the thesis are presented.

- Chapter 2: The focus on the analysis of the state of the art was the first step in the development of the thesis. In the literature, various are the applications of UAVs in indoor and outdoor environments and different solutions are presented.

- Chapter 3: In the first part of this section there is an introduction to the Kalman Filter theory, followed by examples of its application. In the last part, there is a sensor comparison in order to find the best sensor for the desired application.

- Chapter 4: Some simulation have been presented. In particular, the Ardupilot SITL tool and Gazebo are used. At the end of the chapter there is also an analysis on the simulation results.

- Chapter 5: The final part of the thesis consists in some real tests, both indoor and outdoor. The results are really different mainly because of the different conditions in which the UAV has been tested and also because of the different implemented technologies.

- Chapter 6: In this last chapter there are some conclusion on the thesis work. The whole journey of the thesis is summarized and some possible improvements and considerations are presented.

Chapter 2

State of the art

2.1 UAV applications in indoor environments

In recent years, there has been an increased demand in the use of single or multiple unmanned aerial vehicles (UAVs) in indoor environments. Many papers have analyzed the importance of UAVs in indoor environments such as: "A system of UAV application in indoor environment" [1].
This cited paper presents a detailed study on several UAV systems and UAV scheduling systems. It is followed by a proposed system of UAV application in indoor environment, which comprises components of UAV system addressed in detail; focused on scheduler as the heart of operations. In manufacturing and production companies, various operations are gradually being automated to be performed by robots instead of human labors.
These automation robots include Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV). UAVs, which operate in the air, are emerging in various application domains such as surveillance, logistics, and search–rescue missions. Equipped with an imaging device and sensor, UAV can be used for performing inspection tasks in harsh environment, both visual and sensorial inspection. Equipped with a gripper, UAV can also be used for performing material handling task in a manufacturing environment to feed materials to the production line. Together with this applicability, the employment of UAVs in indoor environment is also supported by various advantages such as 360° inspection angle of an object in a three-dimensional space and operability in an empty upper-air space.
On the other side, some difficulties are involved due to narrow spaces and

obstacles, as well as the denial of global positioning system in indoor environment. With recent technological advancements, UAVs can be equipped with video camera, laser range finder, motion capture system, and wireless communication technologies which enable UAV employment for indoor operations.
In this paper there are three elements considered in employing UAVs for a certain application domain: task, environment, and UAV operation system (UAV OS). Task is an activity which will be performed by UAVs. Different types of task include material handling, quality inspection, and area surveillance mission. Environment is the surroundings and infrastructure where the UAV system will be employed. UAV operation system contains different types of UAVs (equipped with gripper, imaging device, and sensor), recharge centers and other resources which support UAV operation. Detailed configuration of the UAV OS such as type and specification of the UAVs, number of UAVs, position and capacity of recharge centers are then decided based on the first two application requirements.
For outdoor environment it is perfectly feasible to use the conventional Global Positioning System (GPS) for localization/positioning system. However, for indoor environment, alternative systems are used, such as laser rangefinder and motion capture system.
With laser range finder, at least one reference point which is known by the positioning system needs to be defined and the latter location of the UAVs in the confined space is approximated through the sensed movement, which may contain a margin of error. On top of that, in an environment cluttered with huge machines (like in manufacturing environment), the sensed perimeter will be biased. On the other side, motion capture system is more reliable but it comes with a high cost of the high-resolution cameras which are needed to capture the marker (attached on the UAV) throughout the confined three-dimensional space. Furthermore, mapping in indoor environment could not rely on the satellite (geographical) map data. Besides objects in the three-dimensional indoor environment, information about feasible paths in the respective free aerial space needs to be included in the map as well.
In addition, a narrower space (compared to outdoor) and uncertain events on-the-fly (e.g. falling wire from the ceiling) require a precise control and responsive alteration of the planned task execution schedule. Thus, a novel UAV system which accommodates a precise mapping, localization, UAV control, together with a fast good-quality schedule with uncertain-event awareness is needed for robust three-dimensional indoor UAV operations.

2.2 Communication technologies for UAV in Industry 4.0

Some important considerations on the needing of communication technologies for UAV in an Industry 4.0 are presented in the paper named "Enabling Communication Technologies for Automated Unmanned Vehicles in Industry 4.0" [2].
Within the context of Industry 4.0, mobile robot systems such as automated guided vehicles (AGVs) and unmanned aerial vehicles (UAVs) are one of the major areas challenging current communication and localization technologies. Due to stringent requirements on latency and reliability, several of the existing solutions are not capable of meeting the performance required by industrial automation applications. Additionally, the disparity in types and applications of unmanned vehicle (UV) calls for more flexible communication technologies in order to address their specific requirements. In the above cited paper [2], several use cases are proposed for UAVs within the context of Industry 4.0 and their respective requirements are considered. One of the suggested scenarios are material handling systems which employ AGVs for transportation of tools and products within a production facility from one location to another.
Another possibility to transport smaller goods is to employ UAVs for this task. Such applications place very high requirements on latency and reliability of the communication links which many of the existing wireless technologies struggle to satisfy. Also, the proposed wireless technologies have to be flexible enough to account for the wide range of UAVs used in the industry. For instance, UAVs, also commonly known as drones, in logistics applications have completely different requirements in comparison with ground-based AGVs. While both types of UAVs require highly reliable communication and accurate localization, UAVs control is more challenging due to their higher degrees of freedom and higher speed. For an efficient operation of UAVs, path planning is essential. Ideally, path planning should optimize the UAVs routes by minimizing the distance traveled, minimizing the travel time or maximizing the utilization of UAVs. Wireless communication between the UAVs themselves (UAV-UAV) and the production facility's network (UAV-infrastructure) can improve global path planning and facilitate coordination amongst the UAVs. Thus, it is essential for a smooth operation of future production facilities.

Low-latency and reliable communication technologies are indispensable for the operation of UAVs in factory automation scenarios. Yet, current technologies lack a solution that can satisfy the wide range of requirements on data-rate, latency, reliability, scalability, and energy efficiency at once. Also, due to mobility of UAVs, reliable and quick handovers are necessary such that data transmission is not disrupted when the UAV moves from the coverage of one cell/access-point to the next. The communication technologies must be energy efficient since the flying-time for a UAV is strictly limited by its battery lifetime. Another great challenge that is prominent in industrial environments is the hostility of propagation channels, due to electromagnetic interference caused by machinery and the common presence of reflectors in an industrial facility's infrastructure that enriches multi-path components.

2.3 Indoor localization systems

Different from guided vehicles, which rely on the pilot to navigate the system, UAV relies on autonomous control to provide this functionality. Hence, precise feedback on the position of the UAV is very important. Unlike outdoor positioning, there are no standard, low cost indoor positioning systems available. There are two types of solution for solving this problem: UAV-standing solution and non UAV-standing solution. The difference is that the second solution requires communication with external nodes dedicated to that task. Getting position information in an outdoor environment can be done easily using a GPS as reported in many previous works. Unfortunately, there is no easy way to obtain position information of a UAV in an indoor or in satellite occluded area environment.
Thus, several researches has been done proposing different way of localizing an UAV in this type of environments such as stereo vision sensor as an indoor positioning system for our UAVs [3], navigation systems based on magnetic field measurements [4], laser ranger, ultrasonic sensors, infrared sensors and Ultra-wideband sensors [5][6][7][8].
There are also other methods based on the probabilistic localization. Stochastic filters used in the probabilistic robotics including localization problems, can be classified into two categories; parametric (Gaussian) and non-parametric filters [9]. In Gaussian filters, it is assumed that noise distributions can be approximated by a Gaussian distribution. This useful assumption provides two different advantages to the parametric filters. The

first advantage of the Gaussian filters is their low computational load, and the second advantage is their accuracy especially where there are reliable models and measurements available.
The two most important types of Gaussian filters are: the EKF (Extended Kalman Filter), and the UKF (Unscented Kalman Filter). Both of these filters are extensions to the original Kalman filter but still assume that the measurement and process noise are both Gaussian. All forms of the Kalman filter involve a recursive algorithm which includes two main steps. The first step is called the prediction step in which the model predicts the system states after applying an input with a Gaussian distribution. The second step is called the update step in which the filter uses a set of measurements with a Gaussian noise to correct the error of the prediction step. The main limitation of the original Kalman filter is the linearity constraint.
However, the EKF and UKF have the same recursive structure but use modified formulations that allow the system to include nonlinear equations. Since most robotic systems involve nonlinear equations, the EKF and UKF are commonly used to solve the navigation and motion control problem of robots. The main difference between the EKF and UKF is in the way these two filters deal with nonlinear equations. The EKF uses the Jacobian to linearize the nonlinear equations of the system around the current state. On the other hand, the UKF uses a set of sigma points to linearize the equations. A set of sigma points is created and passed through the nonlinear equations. The results of these points are computed and the distribution is calculated with the results from these sigma points. The UKF can usually perform moderately better than the EKF when properly tuned. The tuning involves adjusting the spacing of the sigma points along the distribution. Another advantage of the UKF is that the computation of Jacobian is omitted. This can be an advantage for complicated systems for which the determination of the Jacobian is not a trivial task.
Another category of filters used in the localization problem are non-parametric filters. The non-parametric filters instead of assuming any specific distribution for the system states, utilize numerical methods of solving the filtering problem. This issue gives the non-parametric filters some advantages over Gaussian techniques such as EKF. For example, they are able to process raw sensor measurements without extracting feature from sensor data. Also, they are able to deal with the non-Gaussian distribution. In addition, they are able to solve global localization problems.
Two of the most famous non-parametric filters are the Grid and Monte Carlo

Localization (MCL). Grid localization consists of a HF (Histogram filter) in which the posterior robot pose is presented as a grid. In this method, the posterior probability is presented as a collection of discrete probabilities. There are two important issues in the standard algorithm of the grid localization. The first issue is that the localization performance depends on the grid resolution. Filters with fine grains have a superior performance compared to those with coarse grains. Also, finer grains can typically decrease the likelihood of filter failure. However, this superiority will be achieved in the cost of higher computational load. The second issue is related to the sensor model used in the algorithm. In many cases finding a distribution for the sensor can be complicated and time consuming.
Another method utilizing a non-parametric filter to solve the localization problem is MCL. In essence, the MCL uses the PF (Particle filter) algorithm. This method is applicable to both local and global localization approaches mentioned before. The algorithm can be summarized in the following steps. First, a finite number of particles are sampled from the motion model. Second, a weight is assigned to each particle where the weight is proportional to the likelihood of the particle. Finally, the particles are resampled with respect to each particle weight.
An important parameter of the MCL is the number of the particles: the lower the number of particles the higher the failure likelihood of the localization algorithm. One way to resolve this problem is to increase the number of the particles which increases the computational load. Another solution is the injection of random particles in the update step based on the motion model. This method makes the localization system more robust. However, this robustness in achieved in the cost of the wrong posterior distribution, and hence less accurate estimations. The important advantage of the MCL is that a PF can effectively tackle multi-modal distributions with which the Gaussian filters are obsolete.
However, often these methods result to be computationally costly to be executed on-board (mainly because of the third dimension, not considered for mobile robots), but also can have poor accuracy and not be suited for dynamic environments that can lead to a bad measure of the localization.

2.3.1 Indoor UAV positioning using Sterio Vision sensor

In the first article concerning about the indoor localization system [3], the system utilizes two video cameras for stereo vision capture and set of fast algorithms so that position information can be obtained in real-time.
Visual sensors or cameras are widely available today at low cost. The visual sensors are connected directly to the image processing platform, which can be a computer. In this way, possible loss of data in transmission can be reduced. The image processing platform task is to analyse and calculate the position of the UAV for each video frame and transmit the data to the UAV as positioning feedback.
Hence, it is needed to design a suitable algorithm for the UAV detection and position calculation so that image analysis and data feedback can be done in real-time. Positioning data can be transmitted to the UAV using, for example, radio frequency signals.
Stereo vision is also used for distance measurement in several works. Distance data is very useful to estimate the z (height) coordinate of the UAV and to estimate the effective frame width for x and y coordinates. For these particular positioning algorithms, firstly, a stereo vision image capture is done. Then, preprocessing, UAV detection and UAV segmentation are performed on both left and right images.
Finally, the disparity value of the stereo image and the UAV estimated coordinate will be calculated. Stereo image capture is done by using two cameras which are aligned in parallel in fixed position. Both cameras are calibrated so that they have matching image properties such as the size, color space and lighting well as corrected for lens distortion. Experiments conducted show that the system could provide a reliable accuracy in real-time.

2.3.2 Ultra-wideband positioning system

Position estimation of wireless devices has many applications in short-range networks. Ultra-wideband (UWB) signals provide more accurate positioning capabilities with respect to other technologies and, in particular, larger bandwidth. A large bandwidth improves reliability, as the signal contains different frequency components, which increases the probability that at least some of them can go through or around obstacles. Furthermore, a large absolute bandwidth offers high resolution radars with improved ranging

accuracy. Furthermore, spreading information over a very large bandwidth decreases the power spectral density, thus reducing interference to other systems, effecting spectrum overlay with legacy radio services, and lowering the probability of interception. UWB radars have been of long-standing interest, as they have been used in military applications for several decades. UWB communications-related applications were introduced only in the early 1990s. The first commercial systems, developed in the context of the IEEE 802.15.3a standardization process, are intended for high data rate, short range personal area networks (PANs). Emerging applications of UWB are foreseen for sensor networks as well. Such networks combine low to medium rate communications with positioning capabilities. UWB signaling is especially suitable in this context because it allows centimeter accuracy in ranging, as well as low-power and low-cost implementation of communication systems. These features allow a new range of applications, including logistics (package tracking), security applications (localizing authorized people in high-security areas), medical applications (monitoring of patients), family communications/supervision of children, search and rescue (communications with fire fighters, or avalanche/earthquake victims), control of home appliances, and military applications [7].

The above cited paper [5] provides a wide description of the UWB. It starts describing the UWB radio, that is a method of spectrum access that can provide high speed data rate communication over the personal area network space. UWB is based on transmitting extremely short pulses and uses techniques that cause a spreading of the radio energy (over a wide frequency band) with a very low power spectral density. This high bandwidth offers high data throughput for communication. The low frequency of UWB pulses enables the signal to effectively pass through obstacles such as walls and objects.

In general, the UWB technology has different features that have been widely explored in the literature. The high data rate of UWB can reach 100 Megabits per second (Mbps), which makes it a good solution for near-field data transmission. Also, the high bandwidth and extremely short pulses waveforms help in reducing the effect of multipath interference and facilitate determination of TOA (Time of Arrival) for burst transmission between the transmitter and corresponding receiver, which makes the UWB a more desirable solution for indoor positioning than other technologies.

The duration of a single pulse determines the minimum differential path delay while the period pulse signals determines the maximum observable

multipath delay in order to unambiguously perform multipath resolution. Therefore, UWB is considered to be one of the most suitable choices for critical positioning applications that require highly accurate results.
UWB technology, unlike other positioning technologies such as infra-red and ultrasound sensor, does not require a line-of-sight and is not affected by the existence of other communication devices or external noise due to its high bandwidth and signal modulation.
Decawave is a company that uses UWB technology and TOA algorithms to determine the distance among devices and fixed-location beacons to help in different applications such as inventory management, production flow monitoring and management, retail sales monitor and customer behaviour.
The paper [5] continues with the description of the signal modulation used from the UWB.
Signal modulation is the process of carrying information on the impulse signal by modifying one or more of the signal properties. In general, signal modulation can be categorized based on signal properties that need to be modified into four categories; amplitude modulation, frequency modulation, phase modulation, and hybrid modulation. Signal modulation is a crucial phase in signal transmission that can greatly improve the quality of transmitting signals to achieve certain quality criteria.
For example, UWB signals are usually transmitted in the existence of other signals in the air as well as reflected signals that may cause multi-path interference. Thus, UWB must have high modulation efficiency, as signals must be recognized correctly in the presence of noise and interference. Signal modulation is utilized to enhance the accuracy of UWB localization. Time-hopping spread spectrum (TH-SS) impulse radio can be used to solve multipath problems and generate UWB signals with relatively low computational cost. Other modulations can also be used by the UWB, such as pseudo random (PR) time modulation, binary phase shift keying (BPSK), time-hopping binary phase shift keying (TH-BPSK), time-hopping pulse position modulation (TH-PPM), and minimum-shift keying (MSK).
Its ability to resolve multipath components makes it possible to obtain accurate location estimates without the need for complex estimation algorithms. In this article [7], theoretical limits for TOA estimation and TOA-based location estimation for UWB systems have been considered. Due to the complexity of the optimal schemes, suboptimal but practical alternatives have been emphasized. Performance limits for hybrid TOA/SS and TDOA/SS schemes have also been considered. Although the fundamental mechanisms

for localization, including AOA-, TOA-, TDOA-, and SS-based methods, apply to all radio air interface, some positioning techniques are favored by UWB-based systems using ultra-wide bandwidths.

2.4 Path planning

Path planning is a mandatory process for autonomous mobile robots. It finds a path between two different configurations. The robot is often surrounded by obstacles, that can be standing or moving. Finding a collision-free, safe and optimal path in such a scenario is a difficult problem. Optimal paths could vary depending on the application. Algorithms often optimize for length or travel time, but there are more special approaches also, for example minimizing the total amount of turning. Sometimes, not only the environment of the robot or the energy consumption cause constraints but also the robot itself.
A common practice is to separate the process in two steps, a global planning, which uses prior information of the environment, and a local planning, where the constraints of the robot and the dynamic obstacles are taken into consideration.

2.4.1 Global path planning

Global path planning is a major component in the navigation process. It consists of finding a global path between two robot configurations in a cluttered environment. It is also a well-studied research area, since it is explicitly used in many other fields outside robot motion planning.
A lot of global path planning methods, such as road map, cell decomposition and potential field methods have been explored. They find a complete trajectory from a starting point to one, or more, goal points. A reliable path is computed only if a map of the environment is already available. So, in the global navigation, the prior knowledge of the space where the UAV should move must be available.
The core ideas are often connected to graph-search algorithms. Most of the commonly used global planning algorithms for a single mobile robot can be categorized into one of these groups [10]:

- Graph search algorithms: A graph is created from the map, as follows: nodes are free spaces in the map and edges represent the cost of

getting there (usually the length of that path). The main algorithms are:

  - Dijkstra's algorithm: Starting from the initial node and marks all the neighboring nodes with the cost to get there. If a node has no more edges to check, it chooses the least costly node, and calculates the cost of its adjacent nodes. If multiple path exists to the same vertex, the cheaper one will point to it. Once the goal is reached, the algorithm terminates and the shortest path is presented with the pointing edges. It always finds the shortest path.
  - A* algorithm: This method is very similar to Dijkstra's algorithm, but it is addressing its weaknesses. A general problem with the former one is that it discovers the graph in all directions. A* introduces a heuristic function which helps to priorities selecting nodes in the direction of the goal [10].
  - D* algorithm: D* is also an improvement of the A* method. Generally the former algorithm is quite fast and finds optimal solution, but when an obstacle appears in the path of the robot, re-planning is very expensive. To overcome this issue the D* starts planning from the goal and has the ability to change the cost of the path [10].

- Sampling-based algorithms: Instead of considering the whole state space, it is reduced by sampling. Usually a graph or a tree is created and formerly mentioned or commonly used graph algorithms are used to find solution. In practice, most of these algorithms are only resolution complete, meaning that the outcome depends on the used resolution. It can happen that some solutions are missed if the state-space is not discretized with enough precision or if the maximum time elapsed. The most famous algorithms are:

  - Rapidly-exploring random trees: an algorithm designed to randomly select points from the search space. The selected point is connected to the existing graph. One (or more) trees are grown from the samples and when there is a path between the start and goal configuration a basic graph search is used. Many types of RRT algorithms have been developed, which usually vary in point selection, tree connection, node number or uses heuristic functions. In particular, a very popular RRT-based algorithm is the well known RRT* (Rapidly-exploring Random Tree "star"), which enhances the RRT

algorithm by constructing an asymptotically optimal tree. In this case the computed solution converges toward the optimal solution.

  - Rotate-Translate-Rotate (RTR): similarly to RRT, a tree is grown but the sampling, node selection and extension steps are different. This provides good and fast solution in congested spaces.

- Intelligent, heuristic based methods: In the last few decades, there has been a rapidly increasing interest towards more heuristic methods. In general, these algorithms usually provide optimal solution, but their computational demand is much higher than the previous methods and rise as the map resolution increases.

2.4.2 Local path planning

A local planner is responsible for generating feasible path with respect to its intrinsic constraints in the absence of obstacles. A global planner provides a collision-free path which usually consists of straight segments (in case of graph based planners). In case of non-holonomic robots the local planner has the task of connecting the configurations provided by the global planner, with a smooth trajectory.
After a feasible path is generated, it is checked against collision, and if it fails then new iteration is started. Local planners are strongly coupled with steering methods (sometimes they refer the same) and used in decomposition-based path planning. The exploration of collision-free configuration space involves a huge amount of steering operations and computation. In order to achieve real-time planning, the efficiency of the selected method is crucial. The paper [11] aims to compare performance of three reactive local planning algorithms that are probably mostly used nowadays: the Dynamic Window Approach (DWA), Vector Field Histogram Plus (VFH+) and Smooth NearnessDiagram (SND). Vector Field Histogram Plus (some authors call it Enhanced Vector Field Histogram) is an improvement of Borenstein's and Koren's Vector Field Histogram (VFH). VFH [11] employes a polar histogram grid for representation of the surrounding environment of the robot. This grid is built from a two-dimensional certainty grid updated from sensor readings taken with a ranging sensor (sonar, laser range-finder). The polar histogram is a vector that moves with the robot. Each element of the histogram corresponds to a circular sector for which information about amount and distance of obstacles in the form of a weighted sum is stored.

The histogram after smoothening (by a simple low-pass filtering) has typically peeks, for example sectors with high values and valleys, sectors with low values. Any valley containing sectors with values below a certain threshold can be called a candidate valley. If more than one candidate valley is detected, the best one that most closely matches the direction to the current target is selected. Finally, the most suitable sector within the selected valley is chosen as the next goal and the robot is navigated towards it. The whole process is repeated whenever new sensor data are gathered (or in predefined time steps) until the final goal is reached.
VFH+ [11] enhances the original algorithm in several ways. First, threshold hysteresis is used to suppress alternating between several goals in narrow openings that results in the movement of the robot in the close vicinity of obstacles. Moreover, the robot size is taken into account by enlarging obstacle cells by the robot diameter. Finally, dynamics and kinematics of the robot were not taken into consideration by VFH. Contrary, VFH+ uses a simple approximation of currently possible robot's trajectories by a set of circular arcs with various curvatures assuming that forward and angular velocities are constant. Another approach was presented as a geometry-based implementation of the reactive navigation method design. The approach called Nearness-Diagram (ND) navigation introduces gaps – discontinuities in the nearness of obstacles to the robot which indicate potential paths into occluded areas of the environment. Furthermore, regions can defined as the pairs of consecutive gaps, navigable regions are then valleys. After assembling all the valleys surrounding the robot, all the gaps are compared against the heading provided by the global planner. The next subgoal and control towards it is determined based on position of two closes obstacles and the width of the valley containing the gap with the heading that best matches the heading to the goal.
Smooth Nearness Diagram (SND) [11] differs from ND in the way the next subgoal is determined. SND measures a threat possessed by each of the obstacles (an obstacle is considered a threat if it lies within the safety distance of the robot) – the treat measure increases as the obstacle gets closer to the robot. Deflection from the desired heading is computed based on threat measurements of each obstacle. The experiment show that oscillatory patterns were suppressed leading to method's performance improvement in narrow corridors.
Finally, the Dynamic Window Approach (DWA) [11] searches control commands directly in the space of velocities and takes limitations of the velocities

and accelerations of the robot into account. The approach consists of two parts: a search space is generated at the first, followed by determination of the optimal path in the generated search space. A two dimensional search space of valid linear and angular velocities is computed directly from the limitations of the velocities and accelerations of the robot. The origin of the space lies in the point representing the current linear and angular velocities of the robot. The space is then discretized between the maximal and minimal velocities. Discretization resolution depends on computational power of the robot and the requested precision.
Analysing the previous considerations, it is easy to understand that a complete navigation system should integrate the local and the global navigation systems: the global system pre-plan a global path and incrementally search the best new paths when discrepancy with the map occurs; on the other hand, the local system uses on-board sensors to define a path when the information of the map is not yet available, and detect and avoid unpredictable obstacles.

Chapter 3

Kalman Filter and Sensor Analysis

3.1 Introduction

Autonomous Robots and Vehicles need accurate positioning and localization for their guidance, navigation and control. Often, two or more different sensors are used to obtain reliable data useful for control systems. In this application the final data is obtained after a sensor fusion technique based on the Extended Kalman Filter theory.

3.2 The Discrete Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error [12]. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

3.2.1 The process to be estimated

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

with a measurement $z \in \Re^m$ that is:

$$z_k = Hx_k + v_k$$

The random variables w_k and v_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions.

$$p(w) \sim N(0, Q)$$
$$(v) \sim N(0, R)$$

In practice, the process noise covariance and measurement noise covariance matrices might change with each time step or measurement, however here they are assumed to be constant.

3.2.2 The Computational Origins of the Filter

We define $\hat{x}_k^- \in \Re^n$ (note the "super minus") to be the a priori state estimate at step k given knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be the a posteriori state estimate at step k given measurement z_k. Then, the a priori and the a posteriori estimate errors are defined as:

$$\hat{e}_k^- \equiv x_k - \hat{x}_k^-$$

and

$$e_k \equiv x_k - \hat{x}_k$$

The a priori estimate error covariance is then:

$$P_k^- = E[(e_k^-)(e_k^-)^T]$$

and the a posteriori estimate error covariance is:

$$P_k = E[(e_k)(e_k)^T]$$

In deriving the equations for the Kalman filter, the goal is to find an equation that computes an a posteriori state estimate $\hat{x}_k$ as a linear combination of an a priori estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

The difference $(z_k - H\hat{x}_k^-)$ is called the measurement innovation, or the residual. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K is chosen to be the gain, or blending factor, that minimizes the a posteriori error covariance P_k. This minimization can be accomplished by first substituting $\hat{x}_k$ into the above definition for e_k, substituting that into P_k, performing the indicated expectations, taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for K. One form of the resulting K that minimizes P_k is given by:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R}$$

Looking at the previous formula it is possible to notice that when the measurement error covariance R approaches zero the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}$$

On the other hand, as the a priori estimate error covariance approaches zero the gain K weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K_k = 0$$

Another way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less. On the other hand, as the a priori estimate error covariance P_k^- approaches zero the actual measurement z_k is trusted less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

3.2.3 The Discrete Kalman Filter Algorithm

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback

in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, for example incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.
The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems. The first task during the measurement update is to compute the Kalman gain, K_k. The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance estimate via the following formula:

$$P_k = (I - K_k H)P_k^-$$

After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates. This recursive nature is one of the very appealing features of the Kalman filter, it makes practical implementations much more feasible than other techniques. The Kalman filter instead recursively conditions the current estimate on all of the past measurements.

3.3 The Extended Kalman Filter

As described above, the Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a linear stochastic difference equation. If the process to be estimated and (or) the measurement relationship to the process is non-linear some of the most interesting and successful applications of Kalman filtering have been such situations.
A Kalman filter that linearizes about the current mean and covariance is referred to as an extended Kalman filter or EKF. The estimation can be linearized around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in non-linear applications. To do so, some of the material presented above has to be

modified. Assuming that the process has a state vector $x \in \Re^n$, but that the process is now governed by the non-linear stochastic difference equation:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$$

with a measurement $z \in \Re^m$ that is:

$$z_k = h(x_k, v_k)$$

where the random variables v_k and w_k represent respectively the process and the measurement noise. In this case, the non-linear function in the difference equation $x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$ relates the state at the previous time step k-1 to the state at the current time step k. It includes as parameters any driving function u_{k-1} and the zero-mean process noise w_k. The non-linear function h in the measurement equation relates the state x_k to the measurement z_k.

In practice, the individual values of the noise v_k and w_k are not known at each time step. However, the state and measurement vector can be approximated:

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1},0)$$

and:

$$\tilde{z}_k = h(\tilde{x}_k,0)$$

where $\hat{x}_k$ is some a posteriori estimate of the state (from a previous time step k). It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply an ad hoc state estimator that only approximates the optimality of Bayes' rule by linearization.

3.4 Types of Sensors

In robotics, ultrasonic (US) and infrared (IR) sensors are widely used for contact-less, mid-range distance measurements in navigation systems for humans, mobile robots and vehicle related applications. Obstacle detection is one of the challenging problems in the navigation systems. Moreover, also the estimation of the UAV altitude is extremely important when dealing with flight maneuvers like landing, steady flight, etc. When flying in low altitude, this measurement and its accuracy becomes vital for the application.
For this measurement, several approaches have been studied and used over the years. The most common altitude sensing device present in almost any airborne system is a barometer based altimeter.
It is "a sensitive pressure transducer that measures the ambient static pressure and displays an altitude value on the instrument dial [...] The altimeter is calibrated using the standard atmosphere and the altitude indicated by the instrument is referred to as the pressure altitude" (Nelson, 1989).
One of the problems with this approach is that the pressure altitude and actual or geometric altitude will be the same only when the atmosphere through which the UAV is flying is identical to the standard atmosphere. Another consideration is that this instrument measures Height Above Mean Sea Level (AMSL) and not Above Ground Level (AGL). This means that the AGL altitude needs to be estimated by using the takeoff altitude, either assuming that the area is perfectly leveled or using topography maps for estimating the ground altitude with respect to the mean sea level (MSL).
Altitude can also be estimated using Global Positioning System (GPS). However, standard GPS have a vertical precision between 25 meters and 50 meters and are sensitive to transmission interruptions in urban environment. Due to their lack of precision, these strategies would work assuming that most aircraft fly dozens of meters (usually more) above ground. However, for an UAV used for maintenance purposes or for any operation at low altitude, such as, take off and landing, a sensor that can measure AGL with error in the order of meters or less becomes necessary. The type of sensor used in a particular application must also be chosen in relation to the material from which it is necessary to measure the distance.
The following analysis shows the performance of the US and IR sensors for obstacles made of different type of materials such as cardboard, paper, sponge, wood, plastic, rubber and tile. The ultrasonic sensor uses time of flight (TOF) method for distance measurement, which refers to the time

taken for a pulse to travel from the transmitter to an observed object and back to the receiver. The Infrared sensor works based on the detection of a specific light of wavelength in the range of 760nm (IR spectrum), which is emitted by an IR Light Emitting Diode (LED).
The distance can be measured based on the change in intensity of the received light. For the IR sensor, colour of the obstacle material could also affect the reading of the sensor. For addressing the navigation problems in robotics, various soft computing techniques such as Fuzzy Logic, Neural networks were successfully implemented. Typical navigation systems use the above mentioned sensors for both obstacle detection and obstacle avoidance problems. The features such as light weight, robustness, cost effectiveness, quicker responsive time and so on make the sensors widely acceptable in all the domains. It is important to choose the best sensor for capturing the distance data for different types of obstacles. From now on, the purpose of this analysis will be performed in order to decide which sensor has the best performance when the altitude of the UAV need to be estimated.
In particular, the sensor will be placed on top of the UAV in order to detect the distance from the roof. Then, knowing the height of the environment, the altitude of the UAV can be calculated and then the measure is sent to the Pixhawk to perform a sensor fusion with data coming from other sensors. Sensor selection is a challenging task for any system design, as it critically affects the system performance and its lifetime. In the following part some sensors will be presented.

3.4.1 Ultrasonic sensor HC SR-04

The ultrasonic sensor (HC SR-04) is largely accepted for addressing the challenges in mapping and localization. The sensor emits high-frequency sound wave (40 kHz) through one of its piezoelectric transducers and detects the returning pulses (echo) in the air through another transducer and converts it to proportional voltage variation. The piezoelectric sensor accepts triggering pulses from the microcontroller and send the echo-detection pulses back to the microcontroller. The beam shape of the sensor is conical and the width of the beam is a function of the surface area, frequency and type of transducers used. The beam spread at maximum sensitivity is 76 centimeters across at 3 meters away from the sensor. The sensor can detect all types of obstacles such as metal, wood, concrete wall, plastic etc., with an extremely less affinity with the lighting conditions. Moreover, it does not work when

the measure is taken with respect to phono-absorbent materials. Since in the case of our application the distance must be measured with respect to the ceiling, made of a phono-absorbent material, a different kind of sensor is needed.

Figure 3.1: HC-SR04

The velocity of the ultrasonic wave travel in the air is usually affected by the parameters such as ambient noise, temperature, humidity. In addition, it is more sensitive to the mirror like surfaces. Because of this, for an effective detection of an object with reflexive nature, it has to be brought into a position that is normal to the sensor acoustic axis.

3.4.2 SHARP GP2Y0A21YKOF

The infrared sensor (SHARP GP2Y0A21YKOF) offers a high resolution with quicker response time, compared to ultrasonic sensors. The infrared distance measurement sensor works on the principle of optical triangulation. Sharp infrared sensor has an IR transmitter and a position sensitive device. The position sensitive device is an optical detector which can detect the light falling on a plane. By processing the signal from position the sensitive device, and after interpreting the signal, it gives the distance from the obstacle in front of it.
As the sensors exhibits non-linear characteristic across various surfaces, it is required to interpret the sensor outputs as the distance measure. From the distance measurement characteristics, it is inferred that the IR sensor can provide an inconsistent reading for an obstacle with proximity less than 5cm. The analysis of reflection coefficient of IR sensor provides also one of the most popular methods to identify obstacle.

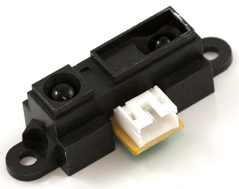

Figure 3.2: SHARP GP2Y0A21YKOF

3.4.3 LiDAR, RADAR and SONAR

The most widely used methods in commercial solutions though, are optical, RADAR or Sound Navigation And Ranging (SONAR) based, due to their operation simplicity and reliability. Since they do not require high processing power and are less likely to generate artifacts (compared to computer vision techniques), these systems are being adopted in a variety of applications. LiDAR (Light Detection and Ranging or Laser Imaging Detection and Ranging) and LADAR (Laser Detection and Ranging) technologies are very similar mainly because of their shared working principle.
Depending on the source they can even be treated as a single technology often called just LiDAR. However, aiming to better characterize the differences between them, it will be used LADAR for the ones using laser based sensors and LiDAR for the regular ones. The primary function of LiDAR and LADAR sensors is to measure the distance between the objects in its field of view, thus characterizing a Time-of-Flight (TOF) sensor. It does so by calculating the time taken by a pulse of light to travel to an object and back to the sensor, based on the speed of light constant.The calculation made by this kind of sensor to provide the distance measurement is defined by equation:

$$d = \frac{c}{2}(t_r - t_s)$$

where d is the distance to the obstacle, c is the speed of light, t_r is the time of the reception and t_s is the time when the pulse was sent. The main difference between them is the divergence angle of the beam used. In LiDAR, the beam is much wider, providing a bigger coverage area but having some impact in the accuracy of the measurements. LADAR, on the other hand, uses laser pulses, which have a much narrower beam with a

much smaller divergence angle, meaning that the energy of the light pulse is concentrated in a much smaller area, yielding to better reflections and consequently, better accuracy.A RADAR is a TOF sensing device that uses radio pulses instead of light for the ranging process. Although being similar to LiDAR and LADAR in its working principle, it differs significantly by using an electromagnetic wave with much bigger wavelength. This makes the RADAR more tolerant to small artifacts in its field of view. This way, it can provide robust performance in any weather condition and environments with high suspense particles count. This technology is used worldwide in different applications, from weather forecast, using Doppler RADAR, to vehicles, where it can be used to provide spatial awareness in drive assist systems or in fully autonomous driving vehicles.

3.4.4 Low-cost LiDAR for indoor applications in UAV navigation

The LiDAR technology has gone through quite a remarkable improvement. In the past, mounting LiDARs on UAVs was also hardly done due to its massive weight and high prices. Consequently, LiDARs could only be loaded on large aircraft which makes a survey very cost-intensive. However, a recent advancement in LiDAR technology has made the sensor much lighter in both weight and cost at the same time. Nevertheless, the need for the evaluation of such LiDARs for their suitability on mounting on UAVs is high. Concerning this topic, an interesting paper [13] shows the result of an attempt on evaluating the performance of a cost-effective, compact LiDAR sensor in the market. Using LiDARs with UAV yields satisfying results for presenting the detailed data over spacious areas. Primarily, LiDARs populate high resolution and accurate data compared to other remote sensing technologies. Previous studies pointed out these limitations of LiDARs for their size and cost. Notably, the use of a ground-based high-resolution LiDAR was very cost-intensive. However, a new LiDAR, which is not only compact but also cost-efficient, came into the market recently.

3.4.5 Radio Frequency Sensors

RF-based systems are widely used in positioning as they can take advantage on the installed communications infrastructure. RF includes Wireless Local Area Network (WLAN), RFID, UWB, LTE, Bluetooth or ZigBee among others. The main methods for location in RF based systems include signal strength fingerprinting (which requires an extensive previous effort on mapping and collection of WiFi patterns) or time of arrival measurements (that must face tough indoor signal propagation conditions).

Radio Frequency Identification (RFID)

Consists of RFID readers with an antenna that interrogates active transceivers or passive tags in order to obtain the tag's unique identification. RFID tags are known as passive tags if they do not require batteries but operate by means of inductive coupling. On the contrary, active tags incorporate their own batteries. Reported accuracies are around 1-5 m, which is insufficient for many indoor applications.

Ultra Wide Band

This technology is one of the most used in 3D indoor positioning, usually helped by other technology. It is a radio technology for short-range high-bandwidth communication that is able to provide centimetres accuracy. UWB achieves strong multipath resistance since the wide bandwidth makes easier the detection of the time-delayed versions of the emitted signal; it also achieves good material penetrability. Nevertheless, it can be adversely affected under strong scattering conditions and requires dedicated infrastructure. UWB uses either Time of Arrival (ToA) or Time Difference of Arrival (TDoA). It also can integrate some available sensors in the UAV. The UWB can present some outliers that can be easily filtered. While UWB positioning bears similarities to radar, there are distinct differences. For example, radar typically relies on a stand-alone transmitter/receiver, whereas a sensor network combines information from multiple sensor nodes to refine the position estimate. On the other hand, a radar can usually choose a location where surroundings induce minimal clutter, while a sensor node in a typical application cannot choose its location and must deal with non-ideal or even harsh electromagnetic propagation conditions.

Figure 3.3: Ultra-wideband

3.4.6 Positioning Techniques for UWB Systems

Locating a node in a wireless system involves the collection of location information from radio signals traveling between the target node and some reference nodes. Depending on the positioning technique, the angle of arrival (AOA), the signal strength (SS), or time delay information can be used to determine the location of a node. The AOA technique measures the angles between a given node and a number of reference nodes to estimate the location, while the SS and time-based approaches estimate the distance between nodes by measuring the energy and the travel time of the received signal, respectively.

AOA

An AOA-based positioning technique involves measuring angles of the target node seen by reference nodes, which is done by means of antenna arrays. To determine the location of a node in a two-dimensional (2D) space, it is sufficient to measure the angles of the straight lines that connect the node and two reference nodes. The AOA approach is not suited to UWB positioning for the following reasons. First, use of antenna arrays increases the system cost. More importantly, due to the large bandwidth of a UWB signal, the number of paths may be very large, especially in indoor environments. Therefore, accurate angle estimation becomes very challenging due to scattering from objects in the environment. Moreover, time-based approaches can provide very precise location estimates, and therefore they are better motivated for

UWB over the more costly AOA-based techniques.

SS

Relying on a path-loss model, the distance between two nodes can be calculated by measuring the energy of the received signal at one node. This distance-based technique requires at least three reference nodes to determine the 2D location of a given node, using a well-known triangulation approach. To determine the distance from SS measurements, the characteristics of the channel must be known. Therefore, SS-based positioning algorithms are very sensitive to the estimation of those parameters. The Cramér-Rao lower bound (CRLB) for a distance estimate $\hat{d}$ from SS measurements provides the following inequality:

$$\sqrt{Var(\hat{d})} \geq \frac{\ln(10)}{10} \frac{\sigma_{sh}}{n_p} d \tag{3.1}$$

where d is the distance between the two nodes, n_p is the path loss factor, and σ_{sh} is the standard deviation of the zero mean Gaussian random variable representing the log-normal channel shadowing effect. The best achievable limit depends on the channel parameters and the distance between the two nodes. Therefore, the unique characteristic of a UWB signal, namely the very large bandwidth, is not exploited to increase the best achievable accuracy. In some cases, however, the target node can be very close to some reference nodes, such as relay nodes in a sensor network, which can take SS measurements only. In such cases, SS measurements can be used in conjunction with time delay measurements of other reference nodes in a hybrid scheme, which can help improve the location estimation accuracy.

Time-Based Approaches

Time-based positioning techniques rely on measurements of travel times of signals between nodes. If two nodes have a common clock, the node receiving the signal can determine the time of arrival (TOA) of the incoming signal that is time-stamped by the reference node. For a single-path additive white Gaussian noise (AWGN) channel, it can be shown that the best achievable accuracy of a distance estimate $\hat{d}$ derived from TOA estimation satisfies the following inequality:

$$\sqrt{Var(\hat{d})} \geq \frac{c}{2\sqrt{2}\pi\sqrt{(SNR)\beta}} \tag{3.2}$$

where c is the speed of light, SNR is the signal-to-noise ratio, and $\beta|$ is the effective (or root mean square) signal bandwidth.
Unlike SS-based techniques, the accuracy of a time-based approach can be improved by increasing the SNR or the effective signal bandwidth. Since UWB signals have very large bandwidths, this property allows extremely accurate location estimates using time-based techniques via UWB radios. Since the achievable accuracy under ideal conditions is very high, clock synchronization between the nodes becomes an important factor affecting TOA estimation accuracy. Hence, clock jitter must be considered in evaluating the accuracy of a UWB positioning system. If there is no synchronization between a given node and the reference nodes, but there is synchronization among the reference nodes, then the time-difference-of-arrival (TDOA) technique can be employed. In the absence of a common clock between the nodes, round-trip time between two transceiver nodes can be measured to estimate the distance between two nodes.

3.5 Application of the Extended Kalman Filter

The UAV uses a system for the altitude estimation based on the barometer present in the Pixhawk. However, the barometer is very inaccurate. Another sensor needed to be added in order to correctly estimate the altitude of the UAV. First of all, the location of the sensor is really important for this particular application. The choice is to locate the sensor on top of the UAV. Since the roof is more uniform than the floor, this position provides better measurements of the altitude. From the distance measured by the sensor, the altitude can be easily computed knowing the height of the environment where the UAV will move.
In the table some comparison for different kind of sensors have been made. The analysis of the environment in which the UAV will move is fundamental for a correct choice of the sensor. Since the roof is made of absorbent panel, the ultrasonic sensor does not provide accurate measurements since the panels absorb the signal sent by the transmitter. The other possible sensors that can be implemented are the LiDAR, the Infrared Sensor and the ToF (Time of Flight) Sensor. The best sensor that can provide the desired information is the ToF Sensor. This sensor is very accurate ($\pm 1\%$) and uses a VCSEL (**Vertical Cavity Surface Emitting Laser**) technology.

The data from this sensor will be used to predict the correct altitude together with the data coming from the barometer. To get the optimal result, both sensor are combined. Since the results of both measurements contain errors, a special method has to be used to combine the results. The commonly used method is fusing those two measurements so it will produce the best estimation by using the Extended Kalman Filter (EKF), already implemented by the Pixhawk board.

Model	Technology	Minimum Range(m)	Maximum Range(m)	Accuracy	Interface	Cost($)
Micro LiDAR	LED	0.10	12.0	$\pm 5cm$	UART, I2C	44.95
Infrared Sensor	LED	0.10	0.8	$\pm 1cm$	Analog	13.95
Ultrasonic Finder	Ultrasonic	0.30	12.0	$\pm 5m$	Serial, Analog	34.95
Ultrasonic Sensor	Ultrasonic	0.02	4.0	$\pm 3mm$	PWM	3.95
ToF Sensor	VCSEL	0.04	4.0	$\pm 1\%$	I2C	21.95

3.6 VL53L1X sensor

One of the possible implementation to estimate the height of the UAV can be done using a VL53L1X sensor. This sensor is a Time of Flight sensor using the laser technology to calculate the distance. This sensor type is suitable for indoor use, however its range and precision is significantly reduced in bright sunlight conditions and is not recommended for outdoor use.
A similar problem can happen when the sensor points to a light on the roof. One possible approach is to have to VL53L1X sensors, one on the top and one on the bottom of the UAV. The two signals coming from the sensors are used through the Kalman filter in order to give more "weight" to the sensor that has the best probability to measure the correct altitude. Moreover, the possibility that there is an obstacle on the ground and a light on the roof is minimum. Of course, this assumption can change in a different indoor environment from the one considered. Ardupilot provides an interface to connect the VL53L1X sensor to the Pixhawk.

Figure 3.4: VL53L1X sensor.

In order to correctly use the signal coming from the sensor is necessary to change some parameters in the "Full Parameter List Page". The correct configuration is:

- RNGFND1_TYPE = 16 (VL53L0X).
- RNGFND1_ADDR = 41 (I2C Address of LiDAR in decimal). The sensor's default I2C address is 0x29 hexademical which is 41 in decimal.
- RNGFND1_SCALING = 1
- RNGFND1_MIN_CM = 5
- RNGFND1_MAX_CM = 120 for the VL53L0X, 360 for the VL53L1X. This is the distance in cm that the rangefinder can reliably read.

- RNGFND1_GNDCLEAR = 10 or more accurately the distance in cm from the range finder to the ground when the vehicle is landed. This value depends on how the sensor is mounted.

It is possible to set the position of the sensor modifying the parameter RNGFND1_ORIENT and set it to 25 if the sensor is at the bottom of the UAV or 24 if placed on the top of the UAV.

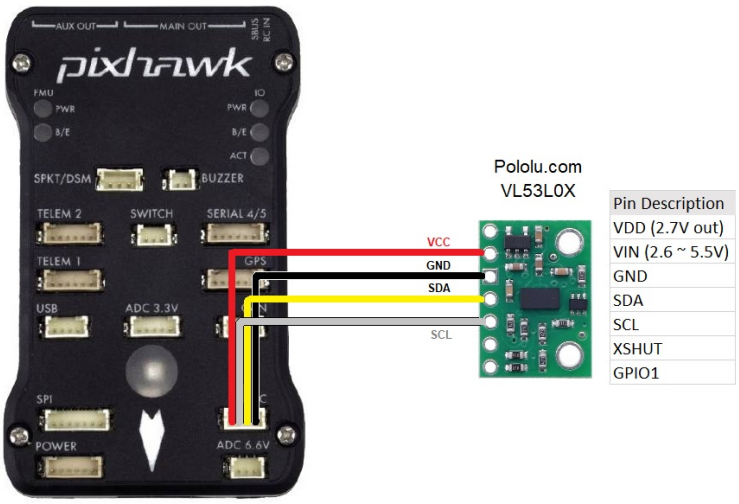

Figure 3.5: VL53L1X sensor connection to the Pixhawk.

Moreover, there is the possibility to add a new sensor. The configuration is the same as before but the parameters to be changed are:

- RNGFND2_TYPE = 16 (VL53L0X).
- RNGFND2_ADDR = 41 (I2C Address of LiDAR in decimal).
- RNGFND2_SCALING = 1
- RNGFND2_MIN_CM = 5
- RNGFND2_MAX_CM = 360 for the VL53L1X.
- RNGFND2_GNDCLEAR = 10

3.7 VL53L1X test

For the correct use of the sensors, to perform some functioning analysis and to compute the accuracy, some tests have been performed. Since this sensors have to be used in both indoor and outdoor environments, there are different scenarios that have to be taken into account. The presence of a light, for example, drastically decreases the performance of this particular sensor.

3.7.1 Error when measuring a distance

The first test (Figure 3.6) shows the result when measuring the distance from the UAV that is at approximately 80 centimeters from the ground. The height of the ceiling is 3.5 meters. Of course there is some error, but as it is of the order of centimeters. Although it is not a perfect measure of the distance, it is well accepted considering that, outdoor, the height of the UAV corresponds to a measure coming from the GPS, the barometer and the magnetometer. This measures are affected by an error that is greater than the one affecting the measure from the VL53L1X.

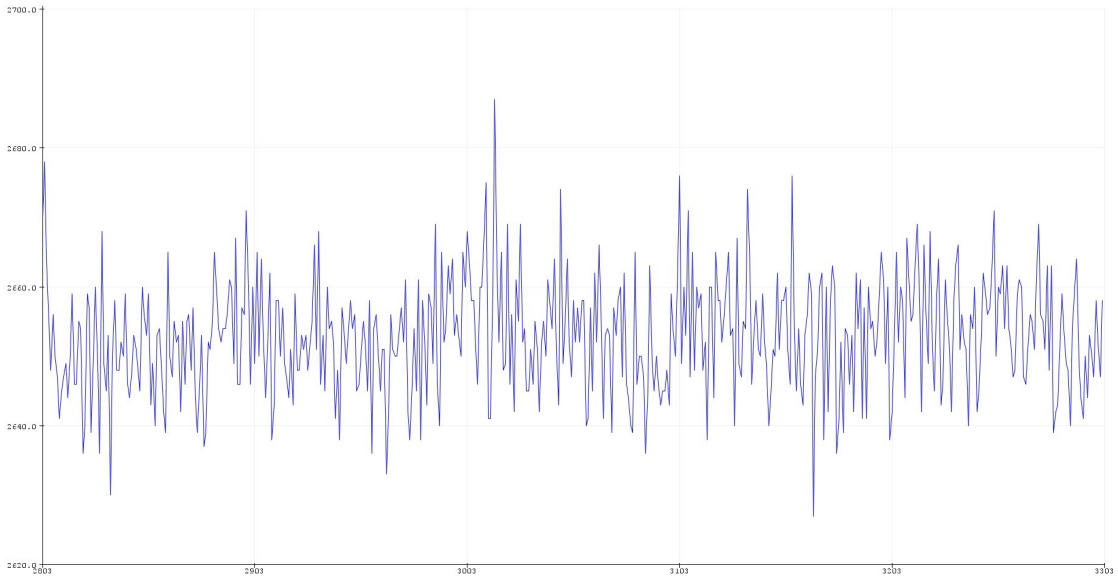

Figure 3.6: Measuring of the distance from the floor to the ceiling with the VL53L1X sensor.

3.7.2 Response time of the VL53L1X

The second test aims to understand what is the reaction time of the sensor. While the sensor is covered it shows that it measures approximately zero. As soon as the cover is removed, the sensor starts to measure the distance. As it is possible to see, the sensor is really fast and for the application required, both for indoor and outdoor environments, the distance is not required to vary rapidly. This means that this sensor is adequate for the requirements.

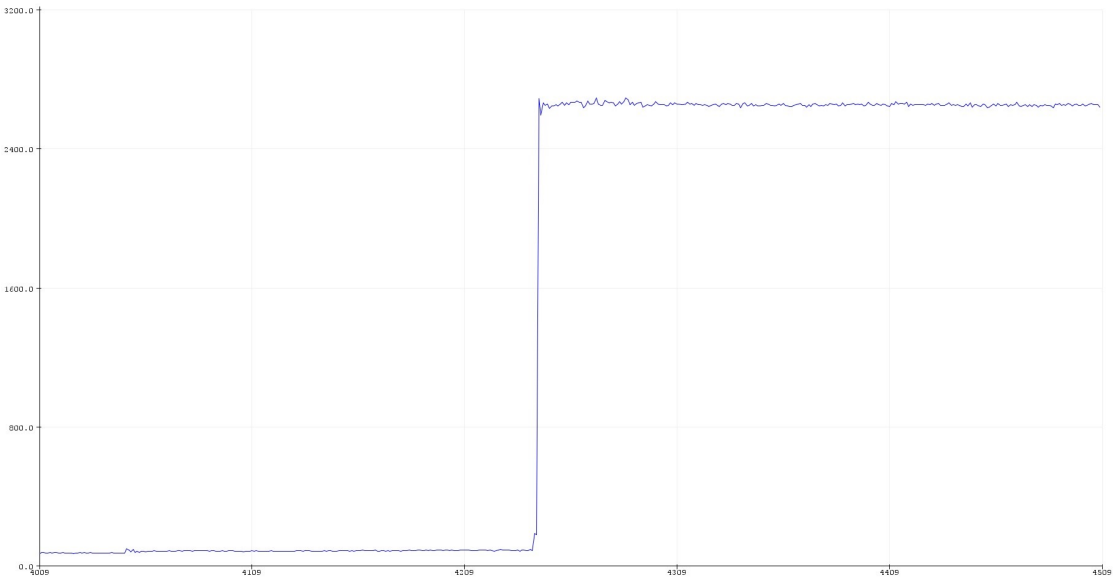

Figure 3.7: Measuring the response time of the VL53L1X sensor.

3.7.3 Measure in presence of a light

The last test shows the measure when a light is present. The expected measure is very poor since this sensor works with a laser technology and it is well known that it does not work well when some light is present. Surprisingly, the error is acceptable. As it is possible to see in the Figure 3.8 the measure is noisy in correspondence of a light. On the output response the noisy measure is highlighted by means of two red boxes.

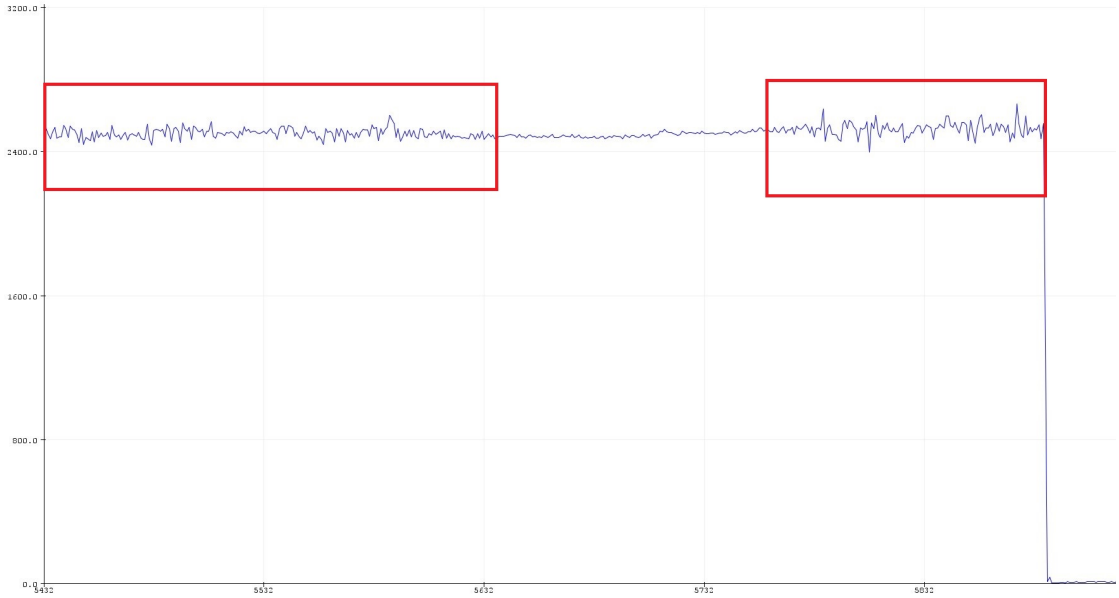

Figure 3.8: Measure in correspondance of a light with the VL53L1X sensor.

All these tests have been performed using an Arduino Nano board as shown in the next figure.

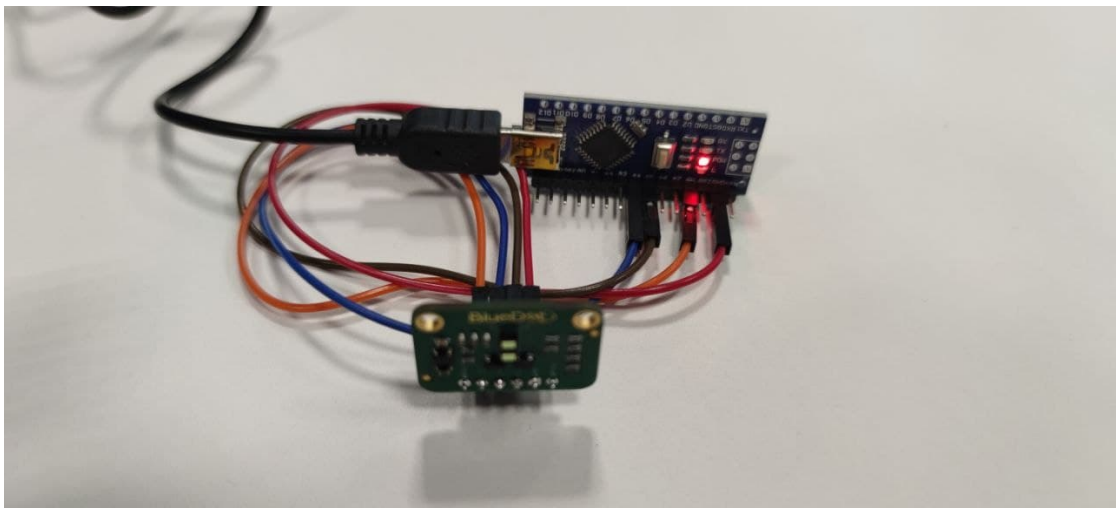

Figure 3.9: VL53L1X sensor connected to the Arduino Nano board.

3.8 UWB Analysis

The UWB used in the thesis is a DWM1001C Qorvo. This module has a DWM1001C UWB transceiver mounted on the PCB. The DWM1001C uses a 38.4 MHz reference crystal. The crystal is trimmed in production to reduce the initial frequency error to approximately 3ppm, using the DWM1001C IC's internal on-chip crystal trimming circuit.

Figure 3.10: DWM1001C - Qorvo

The Ultra-wideband (UWB) is one of the most spread technology used for indoor positioning estimation. UWB is a radio technology based on the IEEE 802.15.4a and 802.15.4z standards that can enable the very accurate measure of the Time of Flight of the radio signal, leading to centimeter accuracy distance/location measurement. In addition to this unique capability, UWB offers data communication capability while using extremely little energy.
By combining accurate location and communication it is one of the best sensors for positioning estimation. Unlike other technologies like Bluetooth or WiFi, which are being re-tooled for a new purpose, the physical properties of the UWB RF signal were specifically defined from the start to achieve real-time, accurate, reliable location and communication.

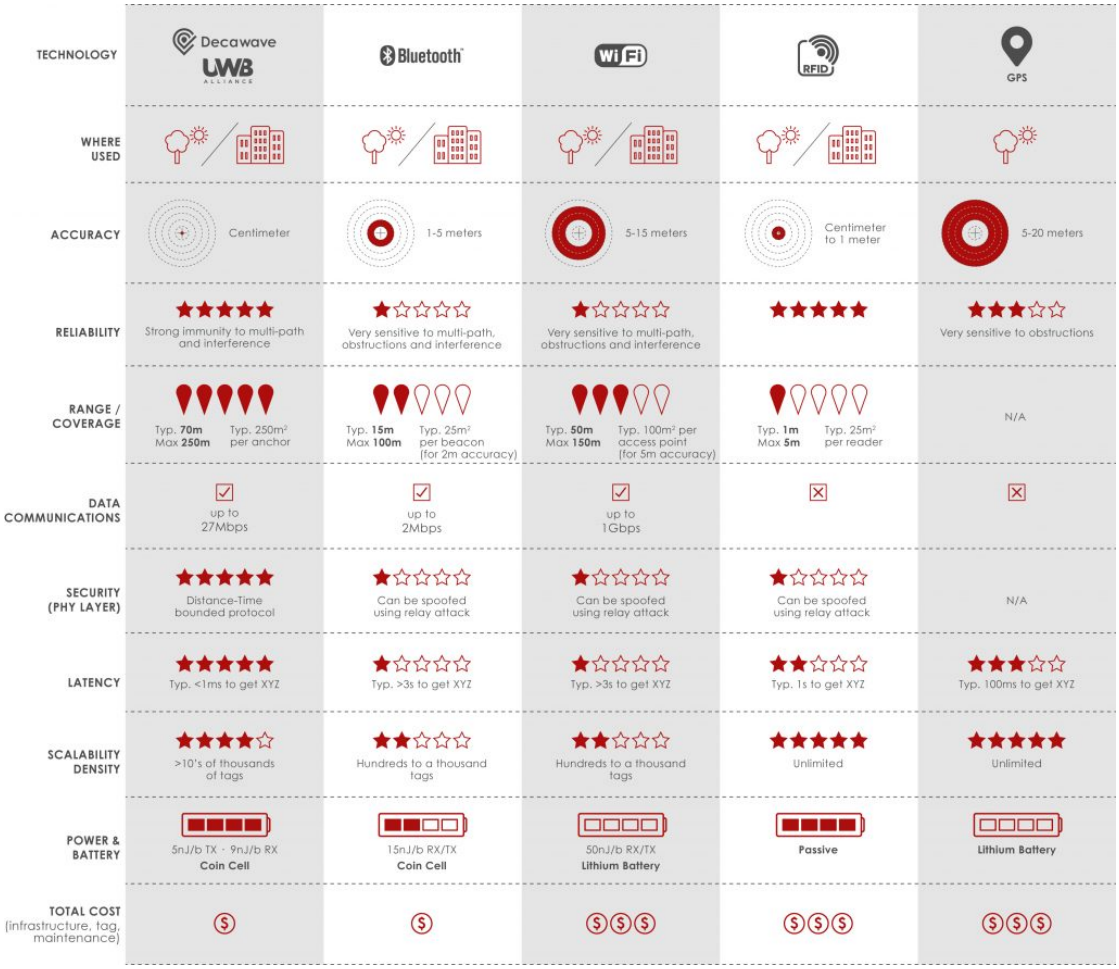

Figure 3.11: UWB Comparison graph.

UWB uses Time of Flight (ToF), which is a method for measuring the distance between two radio transceivers by multiplying the Time of Flight of the signal by the speed of light. From this basic principle, UWB technology can be implemented in different ways based on the target applications needs: Two Way Ranging, Time Difference of Arrival (TDoA), or Phase Difference of Arrival (PDoA). The TDoA method is very similar to GPS.
Multiple reference points, called anchors, are deployed in a venue and are time synchronized. The mobile devices will beacon, and when an anchor receives the beacon signal it will timestamp it. The timestamps from multiple anchors are then sent back to a central location engine which will run multilateration algorithms based on Time Difference of Arrival of the beacons signals to compute the x, y, z coordinates of the mobile devices.

3.8.1 Building the network

The first step comprehendes two phases. In the first phase the anchors are placed, in the second phase they are connected together in order to create the network that will be used to localize the tag. The more the anchors, the more the accuracy of the measurements. Once updated the firmware, the next step is to create a Network. Using the Bluetooth technology provided by the tags/anchors, it is possible to add other tags and anchors and to set some parameters.

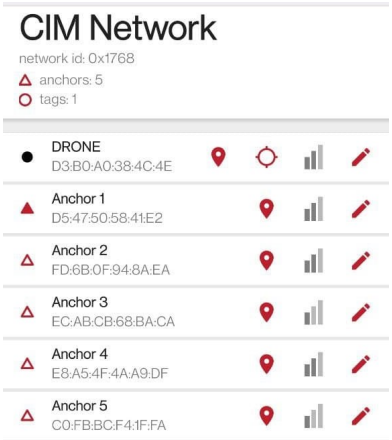

Figure 3.12: CIM Network.

In order to perform all the indoor tests six UWB have been used, one is the tag, placed on the UAV, and five anchors. As it is possible to see, all the UWB must be on the same network to communicate with each other. Of course, there are different possibilities to place the anchors. One of the best configurations is to place four of them forming a square and one placed near the floor. This configuration allows to correctly measure the position but also the altitude. A theoretical configuration is showed in the following figure.
It is important to state that even if the tag goes outside the box cell created with the anchors, the positioning is still good, just less accurate than if the tag is inside the box. The configuration used is the following:

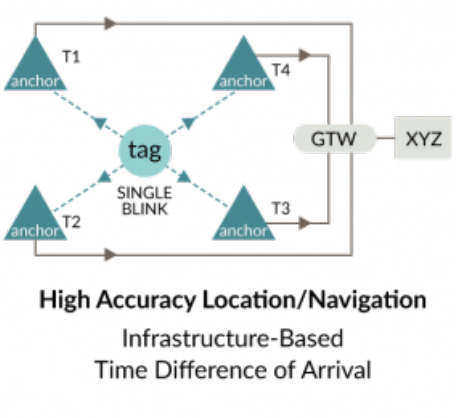

Figure 3.13: TDoA configuration.

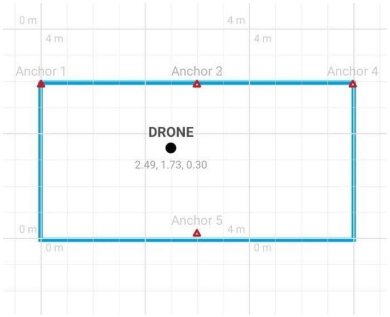

Figure 3.14: Tag into the CIM Network.

Instead of creating a square shape, a rectangular one has been chosen. In this way it is possible to divide the total area in two different cells. The measure of the position is better in this case since there is the fusion from the measure coming from the two cells. One UWB is placed near the ground in order to have also a reliable altitude measure.
The 3D configuration is shown in the following Matlab plot.

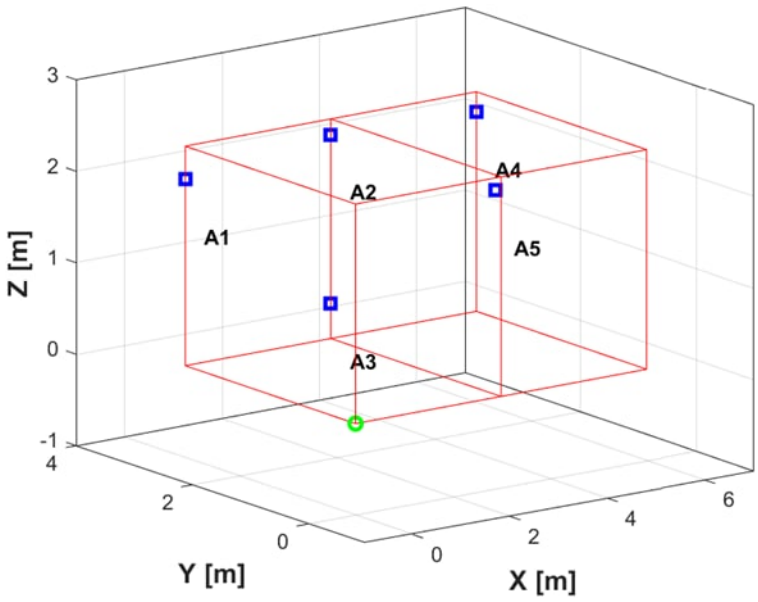

Figure 3.15: UWB configuration in Matlab.

3.9 ROS

The Robot Operating System framework is used to abilitate the communication through the various parts of the UAV.
ROS is a language-independent framework over Linux, used to develop software for robots [10]. It is a set of applications, libraries and conventions to facilitate the development of complex and robust robotic systems. The entire software is open source, freely accessible, and supported by an entire community, thus ensuring that each other's work can be used to develop further improvements. The main features of the framework are that it is peer-to-peer, device based and enables the usage of multiple programming languages.

3.9.1 Package

A ROS program consists of packages. A package performs a task or function. When there is a need for a new task, it can be done by creating a new package. Multiple packages exist in a workspace side by side. In a package there are two main files, namely the package.xml and the CMakeLists.txt. The official translation system for ROS is catkin. Packages can be created in a so-called

catkin workspace. This catkin workspace stores every package, each with its associated package.xml and CMakeLists.txt files. The package.xml file contains the description of the package, the name and contact details of the author, the license, and the package dependencies. CMakeLists.txt contains information for CMake translator, such as the required compiler version, what message and service files need to be generated, what nodes to create, what is the path to link directories, etc.

3.9.2 Node

Within a package, the base unit of the system is the node. The node is a separate, executable program that can be written in one of the supported languages (C++, Python, Lisp). A node can collect sensor data, run some calculations, or can be responsible for controlling a motor, for example. Nodes can communicate with each other using the ROS client library (roscpp, rospy). Client libraries also allow nodes written in different languages to communicate with each other. Nodes can publish and/or subscribe to a topic. In addition, they can provide services to other nodes. On startup, a node always announces itself to the master.

3.9.3 Master

The master is responsible for registering nodes and services, allowing the nodes to find each other and exchange data. The master should always start first when the system is ready to run.ROS is a distributed system therefore nodes can run on different devices within the network. In this case, we need to specify the addresses of each devices and they must register at which address the ROS master is running, as only one master can be in a given ROS system. From this time, communication between nodes takes place transparently through the network. In addition, the master also includes the parameter server, which is a centralized database where parameters related to the system can be stored.

3.9.4 Topic

The topic is a communication channel based on the principle of publish/subscribe. A node publishes data to a topic, and any other node can subscribe to that topic. Multiple nodes can subscribe to a topic, even from different

devices. The subscription is implemented by specifying a so-called CallBack function in the program, which is called when a new message arrives. There is always only one, predefined type of message which can be published to a pecific topic.

3.9.5 Service

Service is another type of communication between nodes. A service enables a node to send a request, to which the service responds. In other words, a node can create services that can be sed by other nodes. This can also be thought of as a remote function call, the request means the input parameters and the response corresponds to the return value. Service's input parameters and its return values must be defined in a .srv file.
Communication through topics is done by sending messages. Publisher and subscriber nodes can communicate with each other, provided they use the same type of message. This means that a topic is clearly determined by the type of message posted on it. A message is a mixed data structure. It may contain primitive data types (bool, float, integer, string) and other message types already defined. The fields of the message must be defined in a .msg file.

3.9.6 RViz

RViz is a visualization tool suitable for displaying information related to sensor data and robot status. By subscribing, it displays the content of various topics in 3D, provided that this is possible. Its primary purpose is displaying and debugging that are extremely useful during development process. Maps, images and ultrasonic and laserscan data are displayed for example by using RViz.

3.9.7 Gazebo

Gazebo is a simulator for robotics applications which includes also a physics engine especially designed for robot and environment simulation. It is free, open-source and widely employed among ROS users. It uses URDF (Unified Robot Description Format), which is a structured XML to describe the robot: links, joints, motors, sensors and so on. Gazebo supports a wide variety of sensors e.g., LiDARs, ultrasonic sensors, camera, odometry and IMU.

It is possible to import 3D files, so one can place the robot into a proper model of the real world. A simulator helps a lot, especially during program development. Many scenarios, algorithms and ideas can be tested quickly in a safe environment, also it gives freedom to the developer to try out the program without the need of a hardware.

3.9.8 Sending the position to the Pixhawk

The GPS mounted on the UAV needs the coordinates in GPS form. After having correctly set the UWB, the next problem was to decide how to send the data coming from the tag placed on the UAV to the Pixhawk. To do that, the Mavlink protocol is used. There are different approach to send information from the UWB to the Pixhawk.
An ad-hoc code has been written in order to read the data from the UWB, parsing the information to have the x, y and z coordinates. Since the UAV mounts an onboard computer, a Jetson Nano, the code just needed to be ran and the GPS position of the UAV is returned. Of course, there is the need of choosing the origin of the map in order to be able to compute the global position by means of local displacement.The code used can be found in the appendix A.
That script provides the coordinates directly to the GPS in assence of external information since the considered application, in this case, is considered indoor.

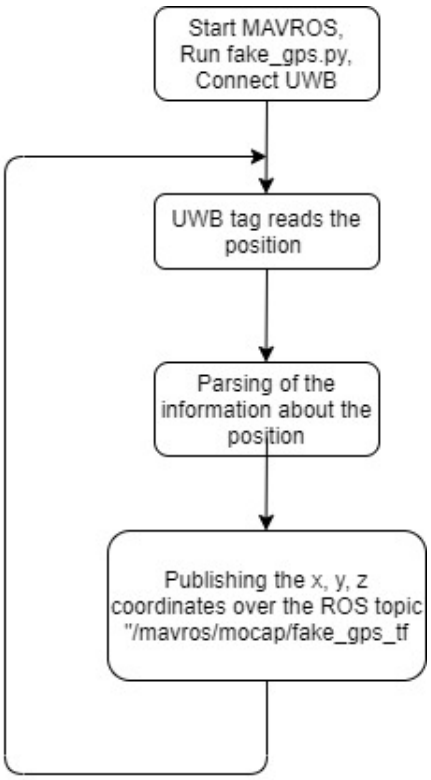

Figure 3.16: Chart explaining the steps to read and publish data to perform the GPS localization in an indoor environment.

Chapter 4

Simulations

4.1 Introduction

One of the most important aspect during this project development was the possibility to simulate the behaviour of the UAV, both indoor and outdoor. There are many powerful tools to perform these simulations. In this thesis, Gazebo and the Ardupilot SITL have been used.

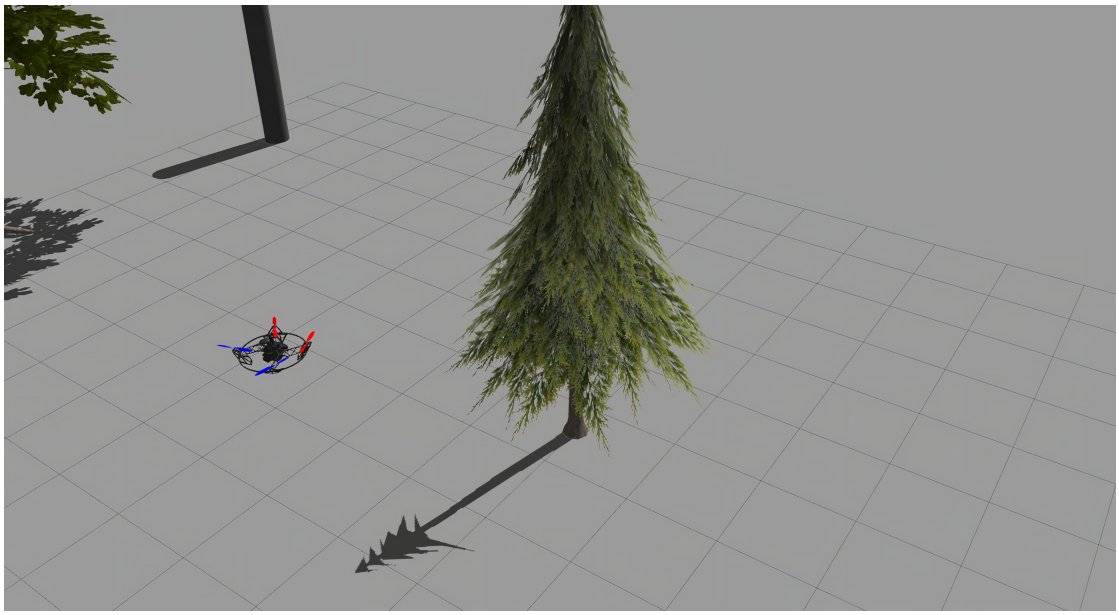

Figure 4.1: UAV in Gazebo.

Gazebo, as previously introduced, is a robotics simulator fully compatible with ROS. It makes possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. To achieve the desired performances in real-life applications, it is important to correctly set the environment. There are some dedicated ROS packages to test UAVs. After having downloaded the desired packages, the Gazebo environment needs to be configured to correctly connect to the simulated autopilot.
The next step is to allow the communication with the simulated UAV. The Mavlink protocol is used through MAVProxy. In particular, the UDP messages were published over the local-host at different ports. Using the correct number of port, Gazebo can correctly receive the messages that have been sent. Through the terminal is possible to send some instructions to the UAV and analyse its behaviour in real-time.
The following step could be, for example, to write some python scripts to publish a series of messages so that the UAV can move autonomously after the execution of the scripts. The idea of the simulation is to test the UAV under different conditions to see how it will behave. Moreover, different kind of faults can be recreated inside the simulated environment, one example is a GPS glitch. It is frequent that the GPS is not very accurate and GPS glitches can happen frequently so it is very useful to be able to test it before going into the real environment since the UAV can behave very dangerously. For this reason, the Ardupilot software provides multiple safety procedure in case of possible problems that can occur. One of the most useful procedure is the landing of the UAV in a safe spot.

4.1.1 SITL simulations

The SITL simulator allows to run a simulation of the UAV without any hardware. It gives a native executable that allows to test the behaviour of the code without any hardware. When running in SITL the sensor data comes from a flight dynamics model in a flight simulator. The first tests have been performed using the SITL directly provided from Ardupilot. It provides a simulated environment like it is shown in the next figure.

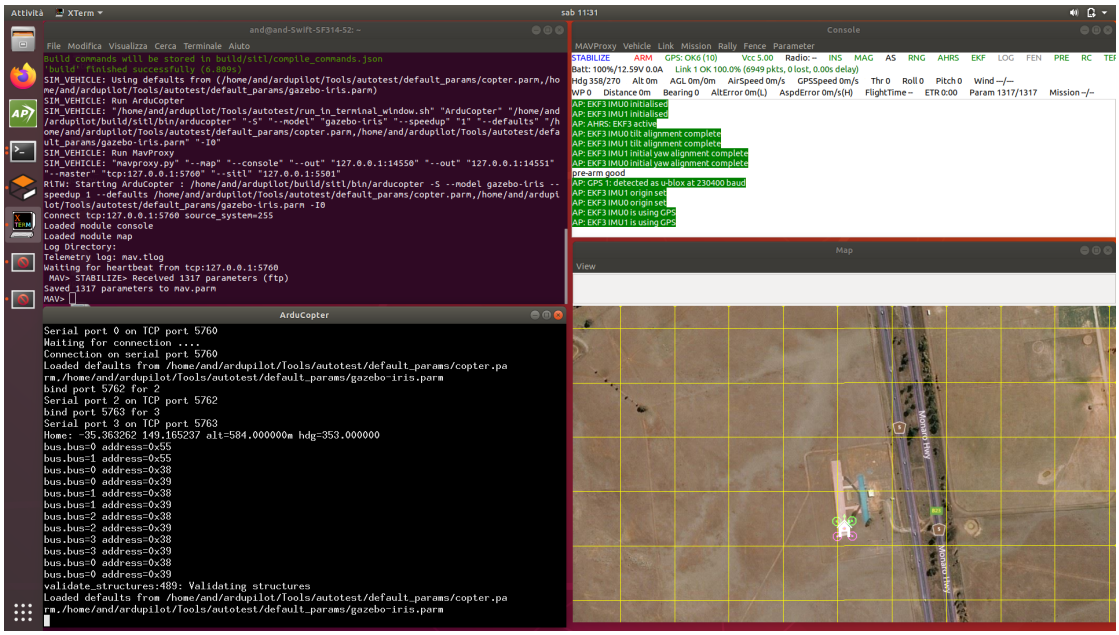

Figure 4.2: Ardupilot simulated environment.

Many tests can be performed using the SITL. Moreover, Ardupilot provides log files that are very useful to compare many parameters like, for example, desired output and measured output. There are basically two kinds of log files.

1. Dataflash logs. They are recorded on the autopilot (often to the SD card) so they must be downloaded from the autopilot after a flight.

2. Telemetry logs. Also known as “tlogs”, they are recorded by the ground station, on the local PC when the autopilot is connected via a telemetry link.

In this thesis the analysis have been performed using the log files, since they

are highly sampled data. On the other hand, the telemetry logs are used for monitoring, for this reason they have a much lower sampling rate and bandwidth.
There are many possibilities to show the data coming from the log files.

1. MAVExplorer
2. MissionPLanner
3. QGroundControl
4. UAVe Plotter
5. UAV LogViewer

Using these tools, is also possible to convert the log files in Matlab files or CSV files. The choice made was using Matlab to perform all the analysis. In the following paragraphs it is shown how it is possible to plot the data using Matlab and easily compare the desired output with the actual output.
Before analysing the data, it is important to understand how the attitude of the UAV can be characterized using just three angles: Roll, Pitch and Yaw.

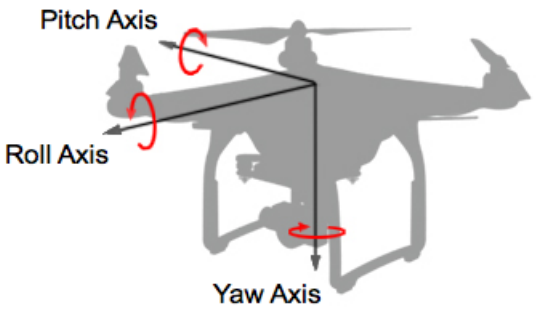

Figure 4.3: Roll-Pitch-Yaw UAV.

4.1.2 Autonomous mission in simulation

To test how the UAV behaves during an autonomous fly, a mission was planned. Given the position of the UAV and a map, it is possible to draw a series of way-points and let the UAV reach them. In this way it is possible to estimate the position error given the way-point that the UAV has to reach and the actual position that the UAV reaches. Moreover, an analysis of the Roll, Pitch and Yaw is performed. Finally, a graph measuring vibrations is displayed.
In the following figure a series of four way-points are shown. It is possible to see the UAV following the trajectory. In this case, the best path is simply the straight line that connects two following way-points. The simulation shows the UAV that follows perfectly the desired trajectory.

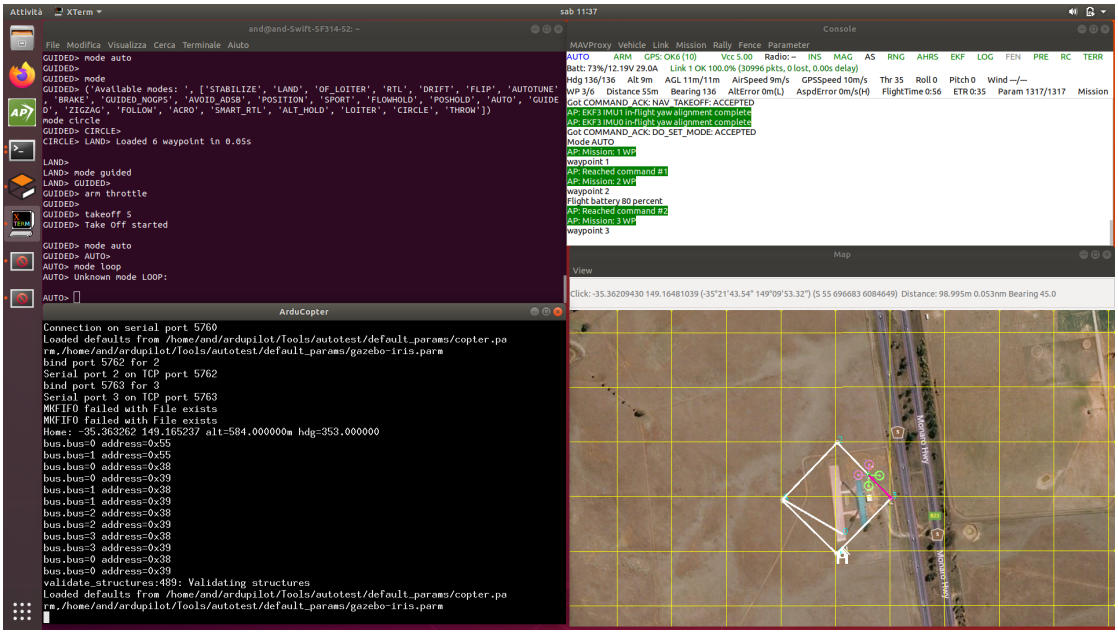

Figure 4.4: Simulated mission.

After the mission, the log files have been downloaded and analysed. The tool used is Matlab R2019b. The two analysis performed are the Roll-Pitch-Yaw errors and some vibrations analysis.

4.1.3 Roll-Pitch-Yaw analysis

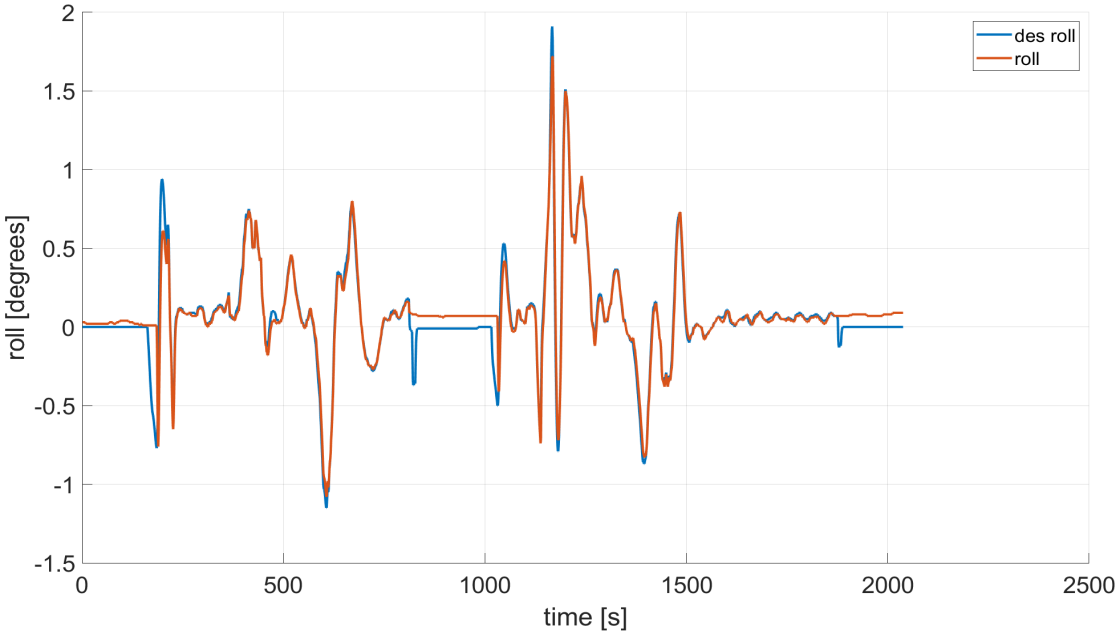

Figure 4.5: Desired roll and measured roll.

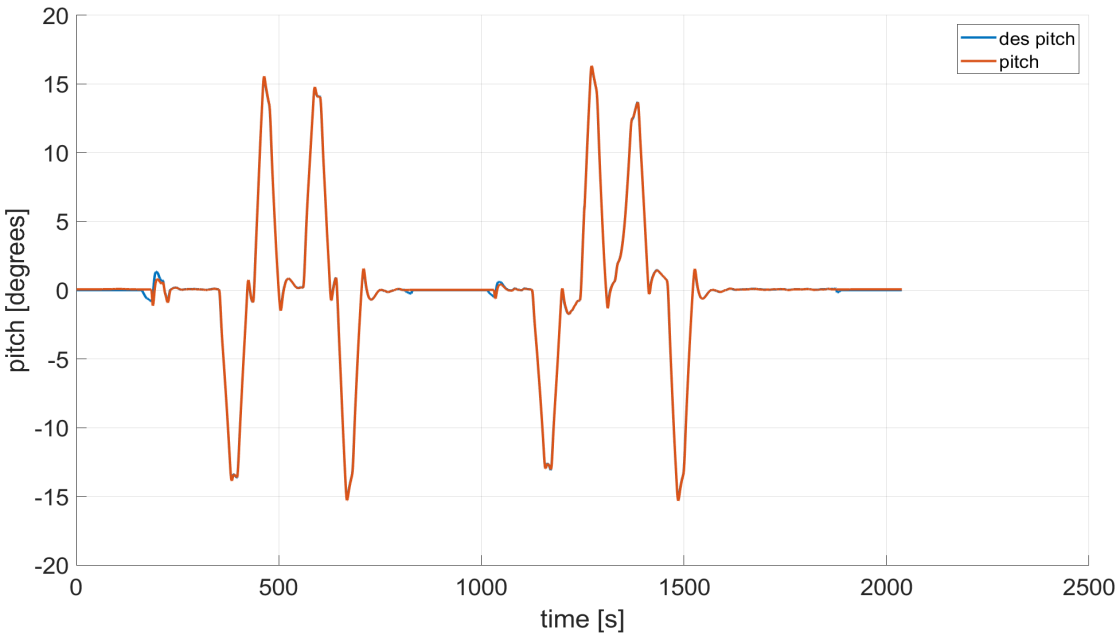

Figure 4.6: Desired pitch and measured pitch

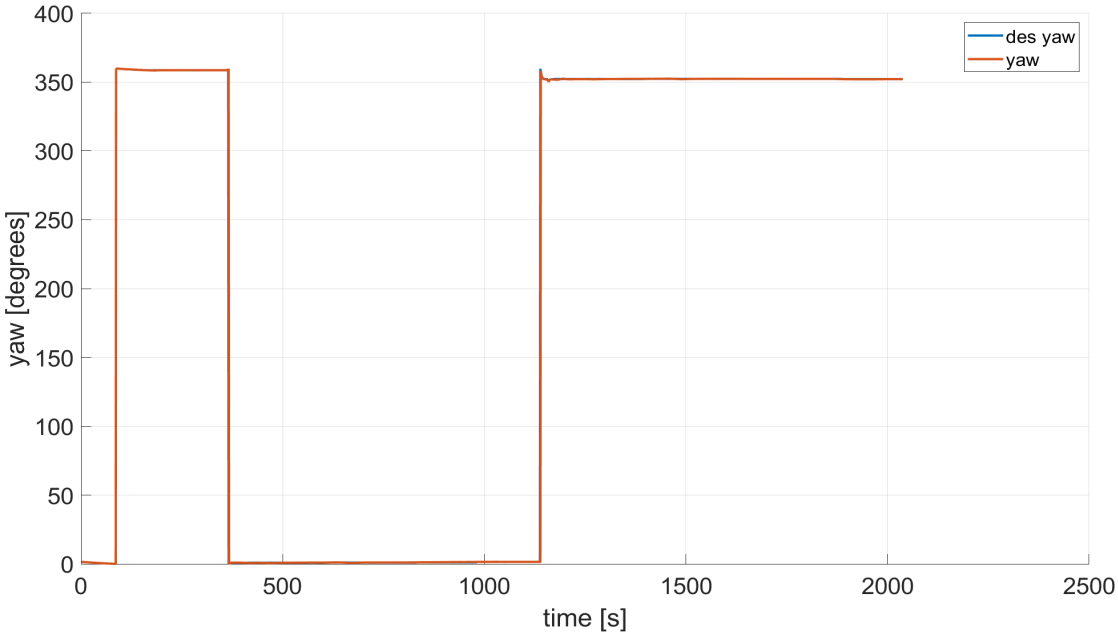

Figure 4.7: Desired yaw and measured yaw.

The expected values are really close to the actual values. The error, when present, is often very low. There are some parts in which the error increases but it is acceptable considering all the causes of uncertainty that, even in simulation, are related to the UAV behaviour.

Vibrations analysis

One important aspect that must be analyzed is related to the vibrations. Autopilots have accelerometers that are sensitive to vibrations. These accelerometers values are combined with barometer and GPS data to estimate the position of the vehicle. With excessive vibrations, the estimation can be wrong and lead to very bad performance in modes that rely on accurate positioning, for example in AltHold, Loiter, RTL, Guided, Position and Auto flight modes.

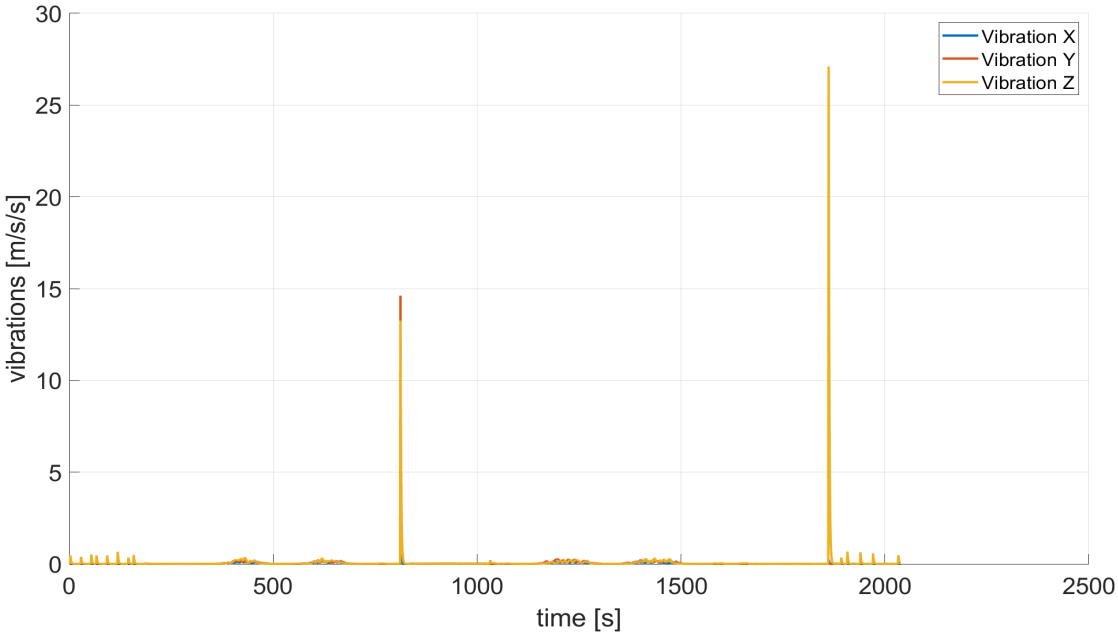

Figure 4.8: Vibrations in a simulated flight.

Usually, vibration levels below 30 m/s^2 are normally acceptable. Levels above 30 m/s^2 may indicate problems and levels above 60 m/s^2 nearly always indicate problems with position or altitude hold. The graph in the figure above shows acceptable vibration levels which are consistently below 30 m/s^2. Even if there are two peaks, the overall behaviour is acceptable.

Another tool that can be used to analyse log files is "LogAnalyzer". A brief example is reported in the appendix B.

Chapter 5

Real tests

5.1 Introduction

In the last part of the thesis, some tests in different real environments have been performed. This part is the most critical because all the technologies implemented have to be tested during the flight in order to understand whether or not the made choices are relevant and valid for the desired behaviour of the UAV. The real tests are basically divided in two parts: outdoor tests and indoor tests.

5.2 Outdoor flights

The outdoor tests are different from the indoor tests mainly because of the different positioning system, the different performances of the on board sensors and the possibility of having different source of errors like, for example, wind influencing the stability of the UAV or the sunlight disturbing the measurement of the ToF sensors.

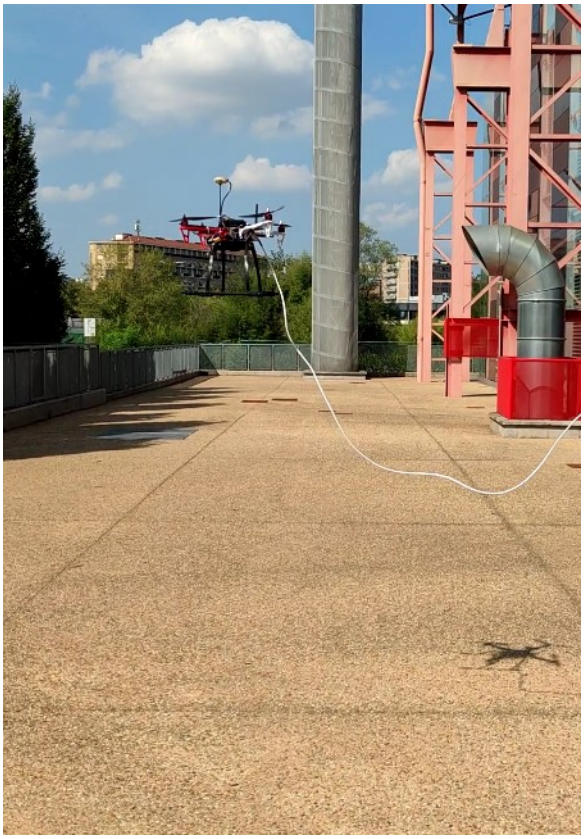

Figure 5.1: Outdoor flight

The main goal of these tests is to evaluate the overall performances of the UAV, in particular the flight stability, the positioning error and the capability of the UAV to fly using all the added sensors. During the outdoor tests, the positioning estimation was performed using the GPS. In this environment it is possible to evaluate different GPS in order to estimate which one provides the best performances.

5.2.1 Tested GPS

The global position is provided to the UAV through the GPS. The quality of the GPS can really influence the flight performances. Having a GPS with poor performances does not provide the right information to the autopilot that, subsequently, will not be able to follow the desired trajectory. Moreover, having a bad GPS can lead to more glitches, that can heavily compromise the flight performances. A well-designed GPS receiver can achieve a horizontal

accuracy of 3 meters or better. For vertical accuracy, it can achieve an accuracy of 5 meters or better 95% of the time. Augmented GPS systems can provide sub-meter accuracy.
The geometry, the atmospheric conditions and even nearby objects can reduce the quality of a GPS signal. The main causes of error and degradation: Dilution of precision (DOP), or geometric dilution of precision (GDOP), is a term used in satellite navigation and geomatics engineering to specify the additional multiplicative effect of navigation satellite geometry on positional measurement precision. Due to the relative geometry of any given satellite to a receiver, the precision in the pseudorange of the satellite translates to a corresponding component in each of the four dimensions of position measured by the receiver, x, y, z, and t. The precision of multiple satellites in view of a receiver combine according to the relative position of the satellites to determine the level of precision in each dimension of the receiver measurement. When visible navigation satellites are close together in the sky, the geometry is said to be weak and the DOP value is high; when far apart, the geometry is strong and the DOP value is low. Basically, the more signals a GPS receiver can intercept (spread apart versus close together), the more precise it can be. If the satellites are spread apart in the sky, then the GPS receiver has a good GDOP. But if the satellites are physically close together, then the DOP is very poor. This lowers the quality of the GPS positioning potentially by meters. Thus a low DOP value represents a better positional precision due to the wider angular separation between the satellites used to calculate a unit's position. Other factors that can increase the effective DOP are obstructions such as nearby mountains or buildings. DOP can be expressed as a number of separate measurements.
HDOP, VDOP, PDOP, and TDOP are respectively Horizontal, Vertical, Position (3D), and Time Dilution of Precision. Also the atmosphere refraction plays an important role and needs to be taken into consideration when discussing the performances of a GPS. The troposphere and ionosphere can change the speed of propagation of a GPS signal. Due to atmospheric conditions, the atmosphere refracts the satellite signals as they pass through on their way to the earth's surface. To fix this, GPS can use two separate frequencies to minimize propagation speed error. Depending on conditions, this type of GPS error could offset the position anywhere from 5 meters. Another possible error source in GPS calculations is the multipath effect. Multipath occurs when the GPS satellite signal bounces off of nearby structures. In effect, the GPS receiver detects the same signal twice at different

ranges. However, this error is a bit less concerning and could cause anywhere from 1 meter of position error.
Two GPS have been tested during these tests.

M8N GPS

The NEO-M8 series of concurrent GNSS modules is built on the u-blox M8 GNSS engine. The NEO-M8 modules utilize concurrent reception of up to three GNSS systems (GPS/Galileo together with BeiDou or GLONASS), recognize multiple constellations simultaneously and provide good positioning accuracy in various scenarios. The NEO-M8 series supports message integrity protection, geofencing, and spoofing detection.

Figure 5.2: M8N GPS

Beitian GPS

The Beitian GPS should provide better performances that the M8N GPS. Also a magnetometer (compass) is present on it, that shall be useful for the sensor fusion with the magnetometer present into the Pixhawk.

Figure 5.3: Beitian GPS

5.2.2 Comparison between Beitian GPS and M8N GPS

The performances evaluation of different GPS needs has to be done with the same reference. In order to have the best comparison, the measures have been collected with the UAV placed on the ground.
This choice has been made because during the flight it is not possible to have the same reference. The M8N and the Beitian GPS has been compared and, as it is possible to see from the following figures, the x and y measures do not differ so much from each other. What is really different is the measure of the z coordinate. The reason why only these two GPS have been tested in this section is just related to the fact that they are the only one used in the real tests. More comparison will be presented in the following sections.
The M8N GPS (Ublox) reaches also an error of 15 meters in the last part of the plot while the Beitian reaches 1 meter of error. For the flight the Beitian GPS is used since it provides a lower error with respect to the M8N GPS.

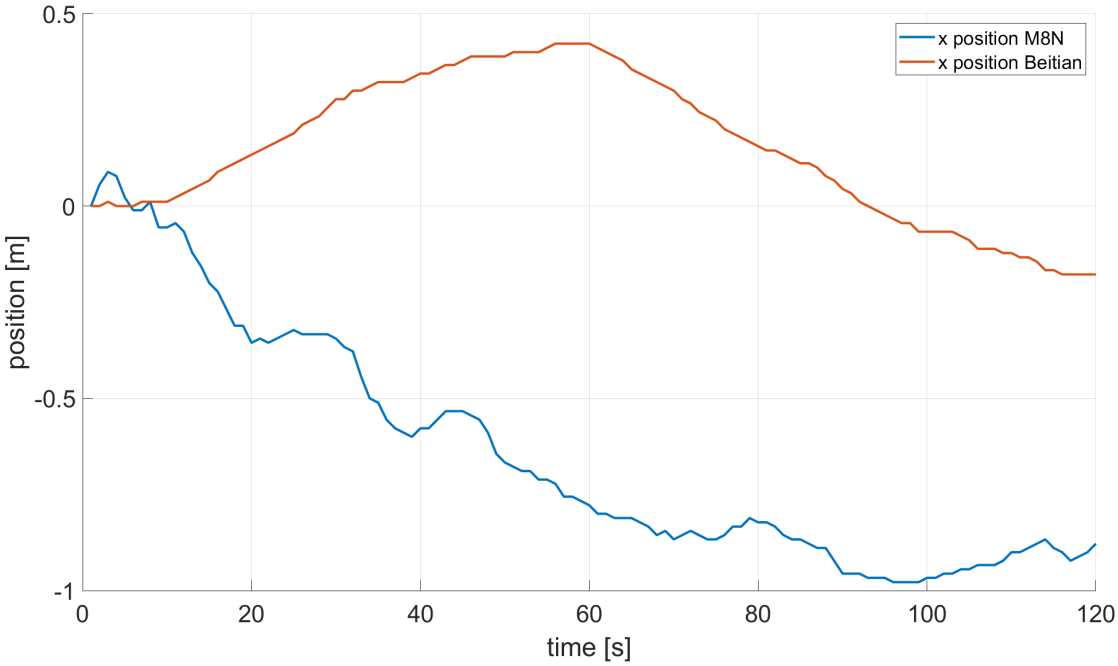

Figure 5.4: Comparison x GPS

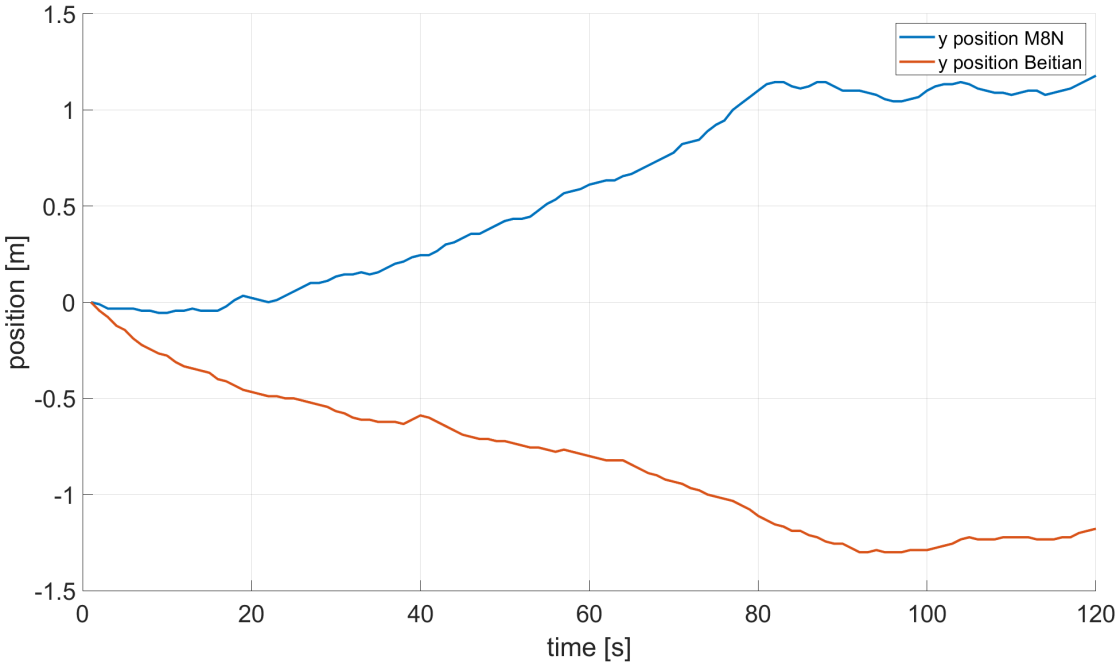

Figure 5.5: Comparison y GPS

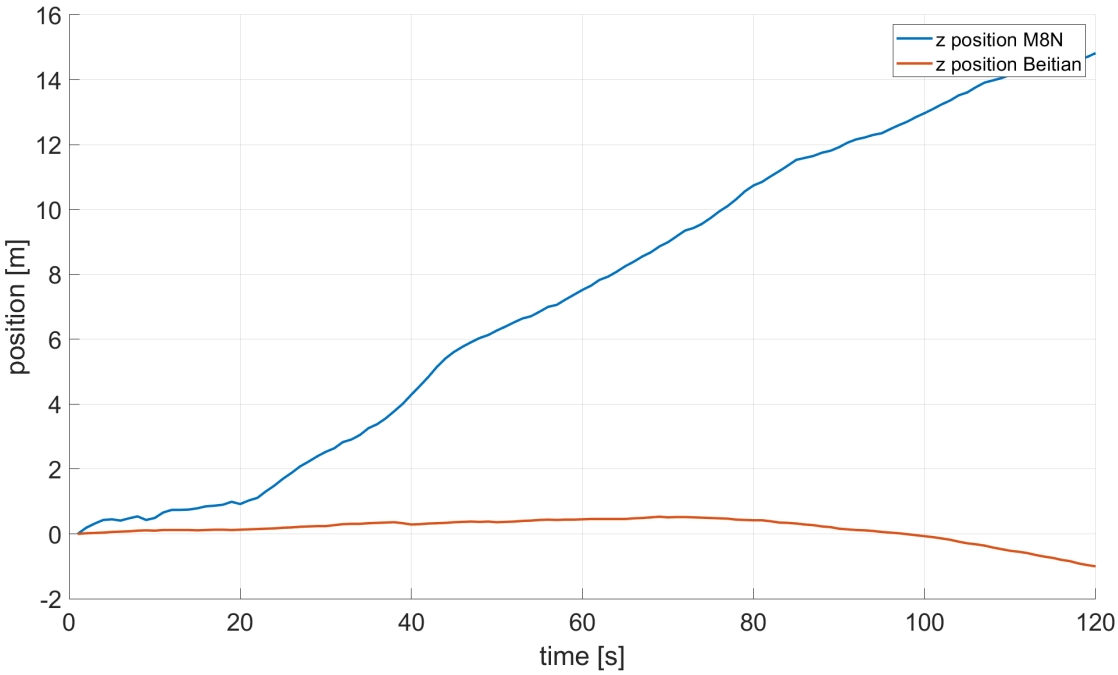

Figure 5.6: Comparison z GPS

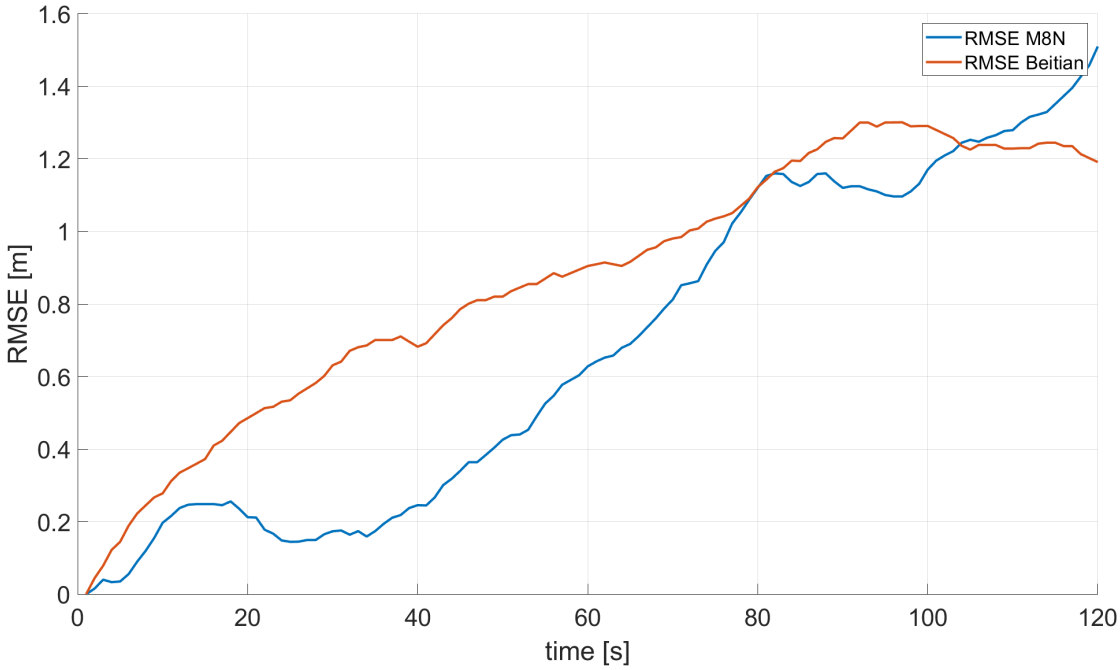

Figure 5.7: Comparison RMSE GPS

5.2.3 Pre-flight Beitian GPS analysis

In this section some examination on the Beitian GPS performances are presented. These analysis were made before a flight in order to understand if the configuration was optimal to assure a good quality flight.

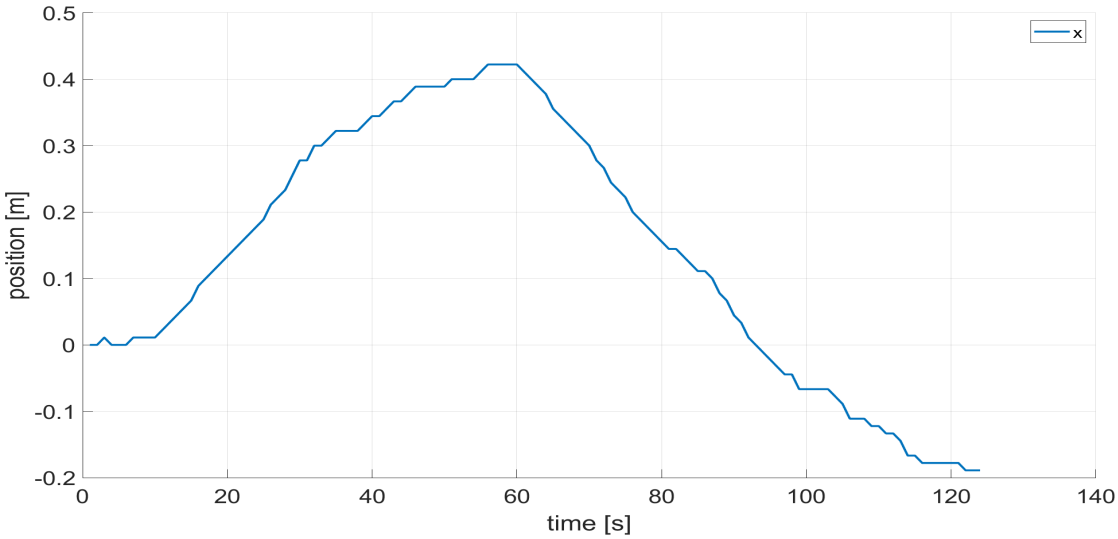

Figure 5.8: Beitian GPS x position

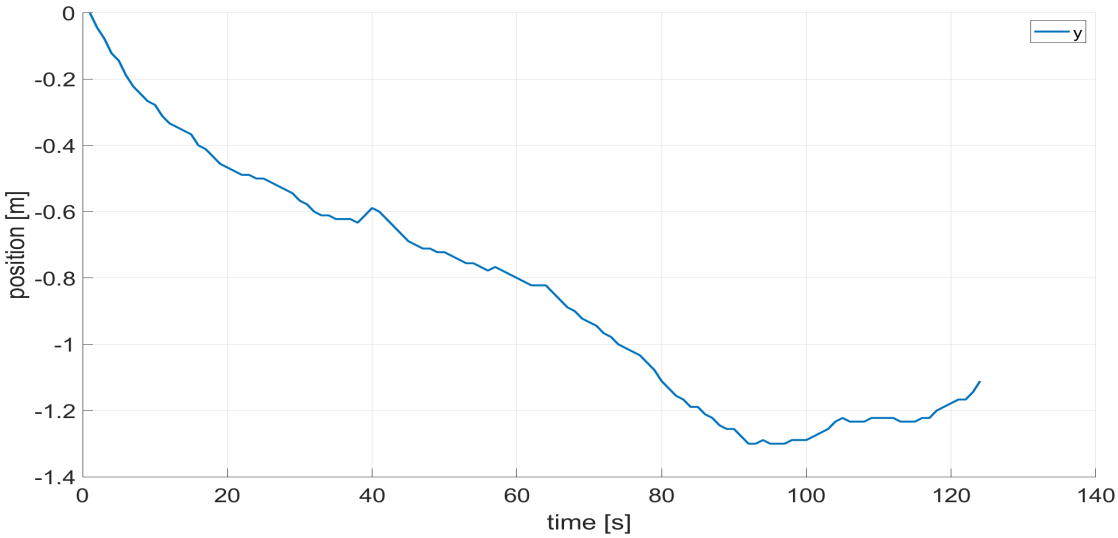

Figure 5.9: Beitian GPS y position

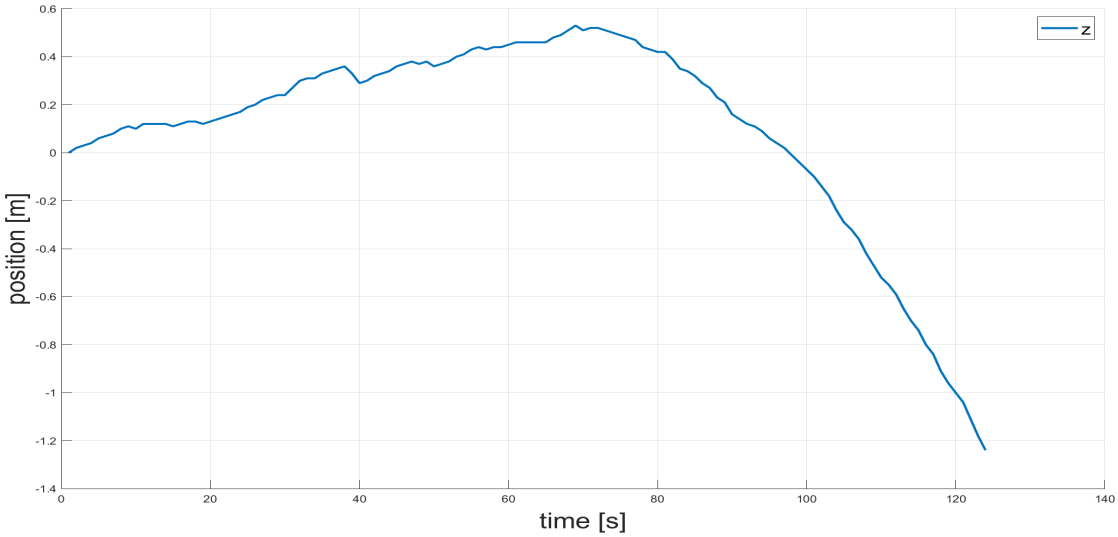

Figure 5.10: Beitian GPS z position

From the previous figures it is possible to analyse what is the error when the UAV is placed on the ground. The error is acceptable and varies in between the interval of $\pm 2m$.
Another interesting measure is the root mean square error calculated using the x and the y measures. It shows how the error varies in time, in the next figure the behaviour of this error is plotted.

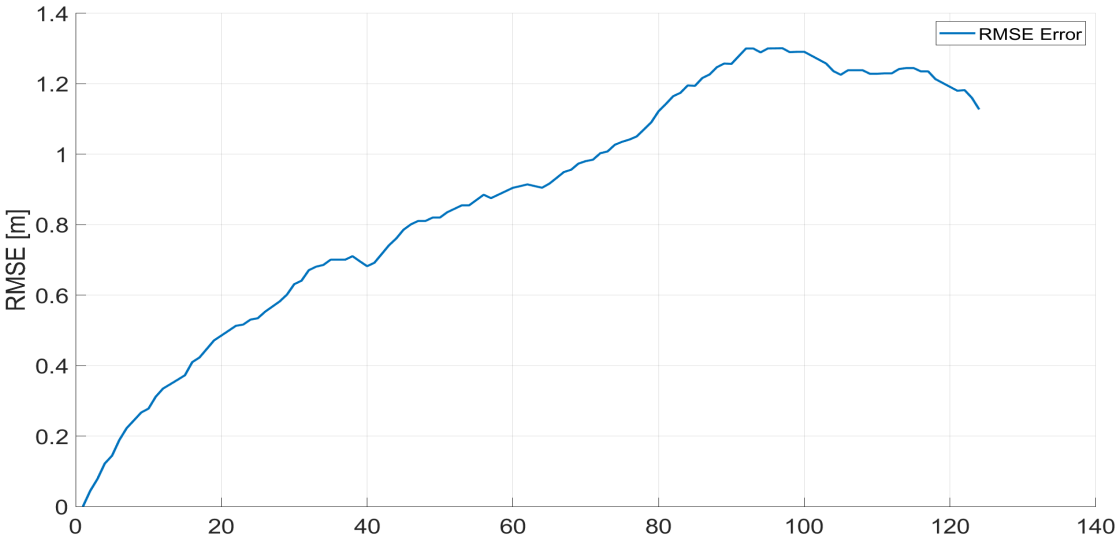

Figure 5.11: Beitian GPS RMSE error

Other parameters that are important during the pre-flight check are those related to the accuracy of the GPS measure. This is because when in autonomous modes, for example Loiter, RTL, Auto mode, positioning errors from the GPS can cause the vehicle to react as if it would be suddenly in the wrong place and lead to aggressive flying to correct the perceived error. These "glitches" show up as a decrease in the number of satellites visible and an increase in the hdop.
The messages that have to be analysed are the "HDop" and "NSats".
Hdop values below 1.5 are very good, values over 2.0 could indicate the GPS positions are not good.
The number of satellites falling below 12 is also bad. A significant change in these two values often is related to a GPS position change.

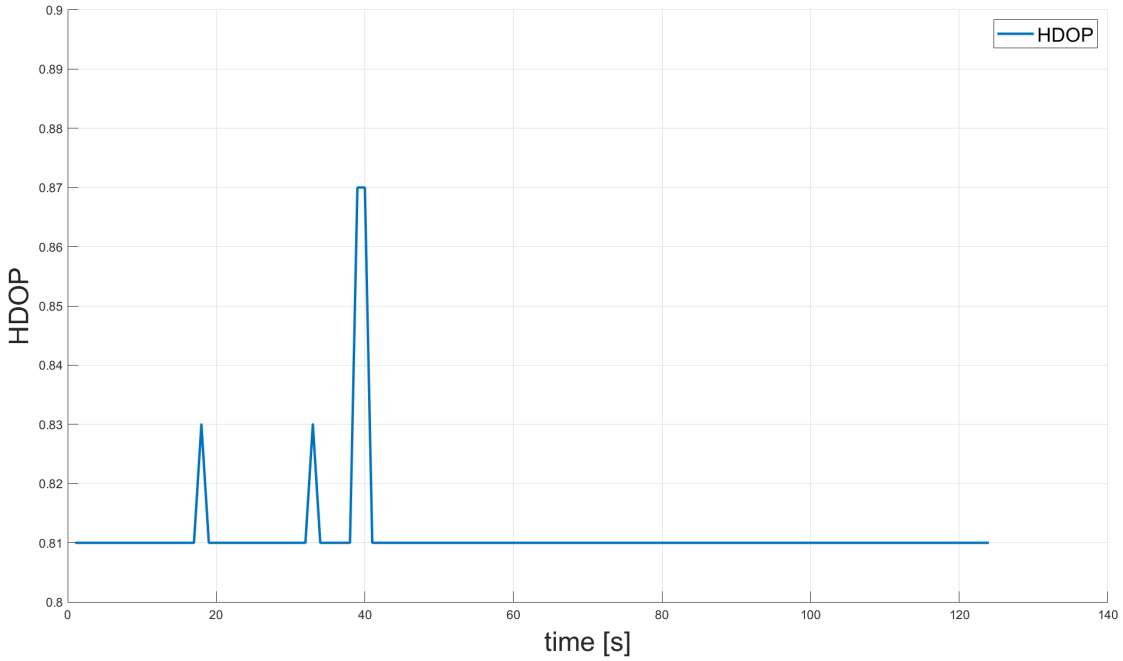

Figure 5.12: HDOP Beitian GPS

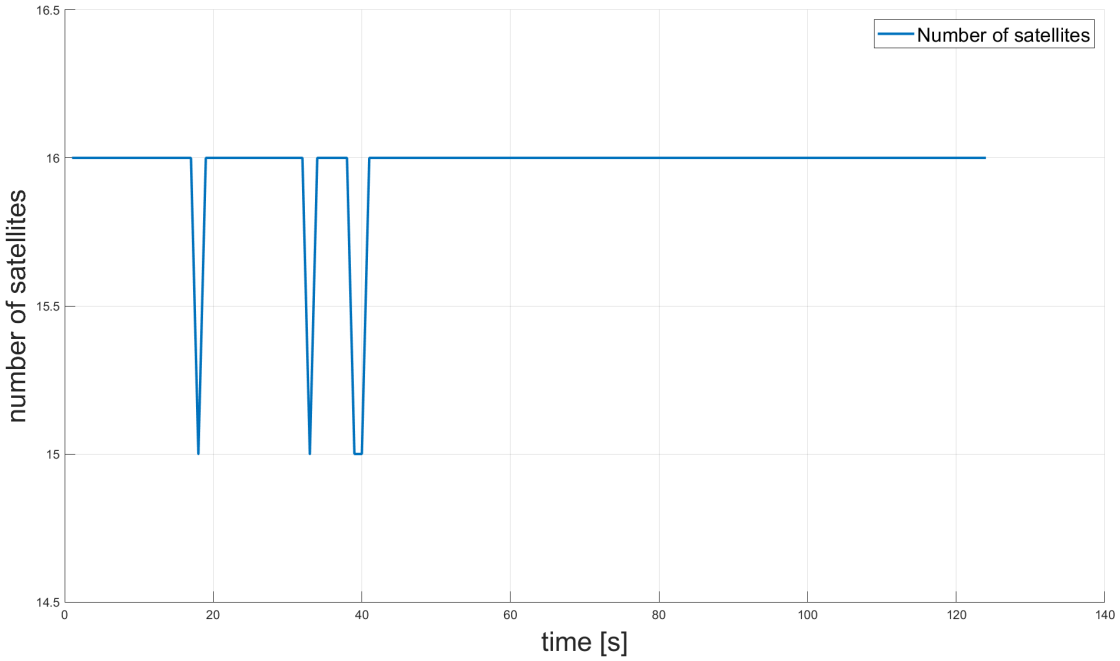

Figure 5.13: NSats Beitian GPS

ZED-F9P RTK GPS

The ZED-F9P positioning module features the new u-blox F9 receiver platform, which provides multi-band GNSS to high volume industrial applications in a compact form factor. ZED-F9P is a multi-band GNSS module with integrated u-blox multi-band RTK technology for centimeter-level accuracy. The ZED-F9P ensures the security of positioning and navigation information by using secure interfaces and advanced jamming and spoofing detection technologies. The ZED-F9P comes with built-in support for standard RTCM corrections, supporting centimeter-level navigation from local base stations or from virtual reference stations (VRS) in a Network RTK setup.

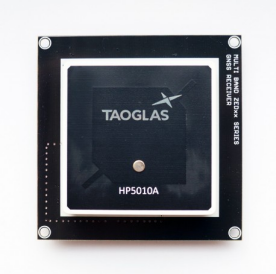

Figure 5.14: ZED-F9P RTK GPS

5.2.4 Comparison between all GPS

Another test has been made using also the ZED-F9P RTK GPS. In this case the mean RMSE corresponds to 0.0413 meters. Having an error of this type is excellent. The next plot shows the RMSE.

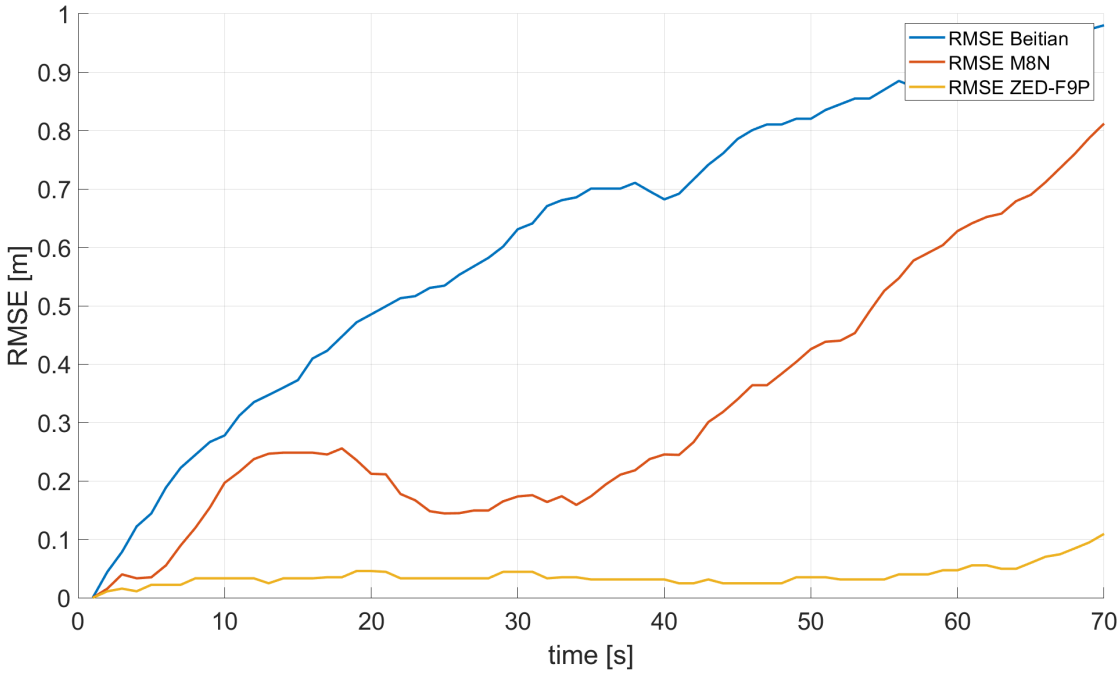

Figure 5.15: Comparison using all GPS

For some structural limitations of the UAV it was not possible to flight using the ZED-F9P RTK GPS since it is too heavy for the DJI F450 frame. It will be implemented in the customized structure. For this reason, all the tests have been done using the Beitian GPS.

5.3 Outdoor flight analysis

After having performed the pre-flight check analysis, some flights have been done. The performances of the UAV were heavily influenced by the atmospheric conditions.
In cloudy days the measures coming from the GPS can be very poor. Since the GPS is the only localization technology implemented in outdoor flights, the outcome of a mission is directly related to the quality of the GPS measure. The following tests have been performed in optimal atmospheric conditions.

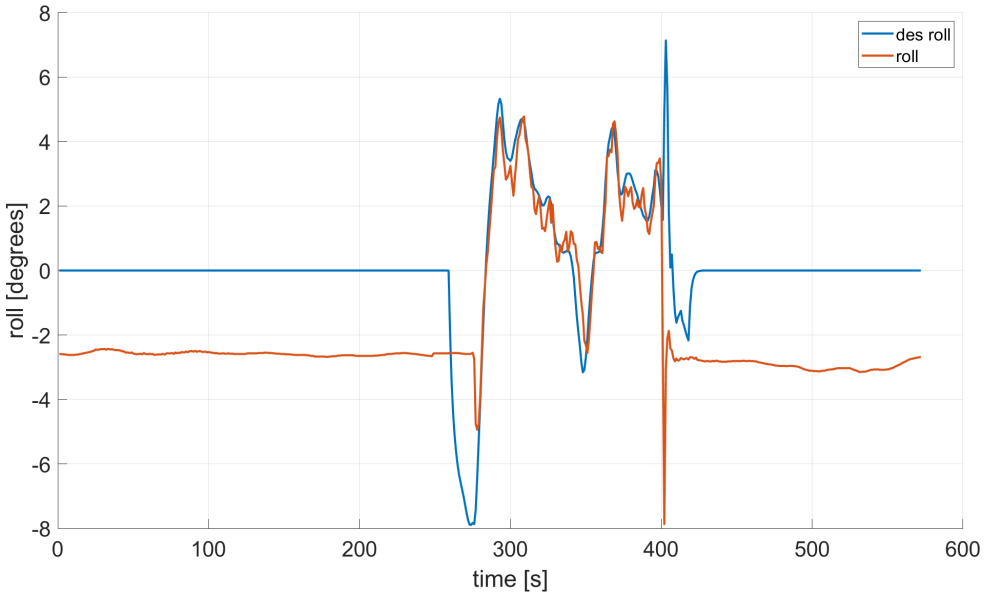

Figure 5.16: Outdoor flight

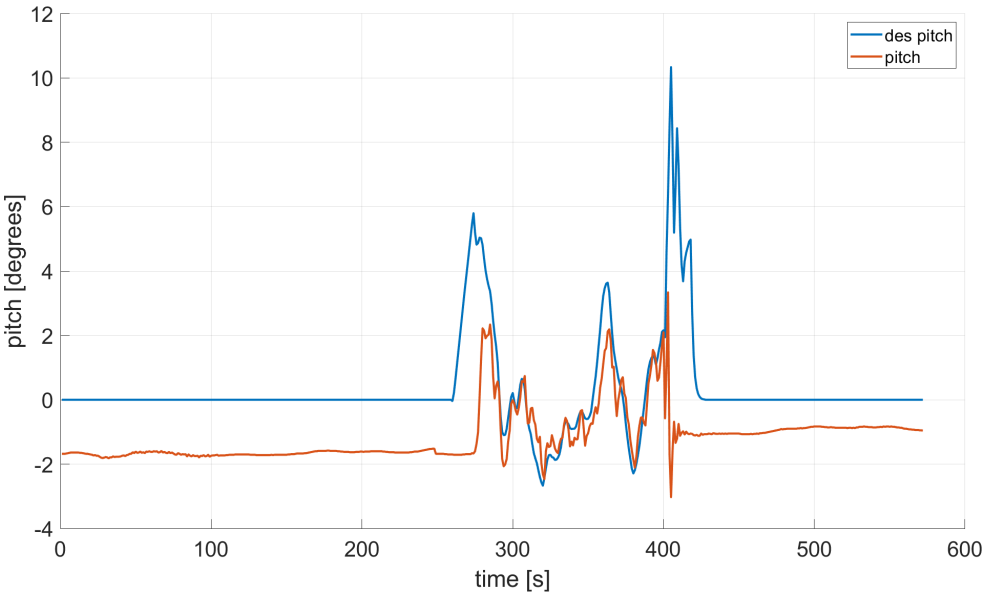

Figure 5.17: Outdoor flight

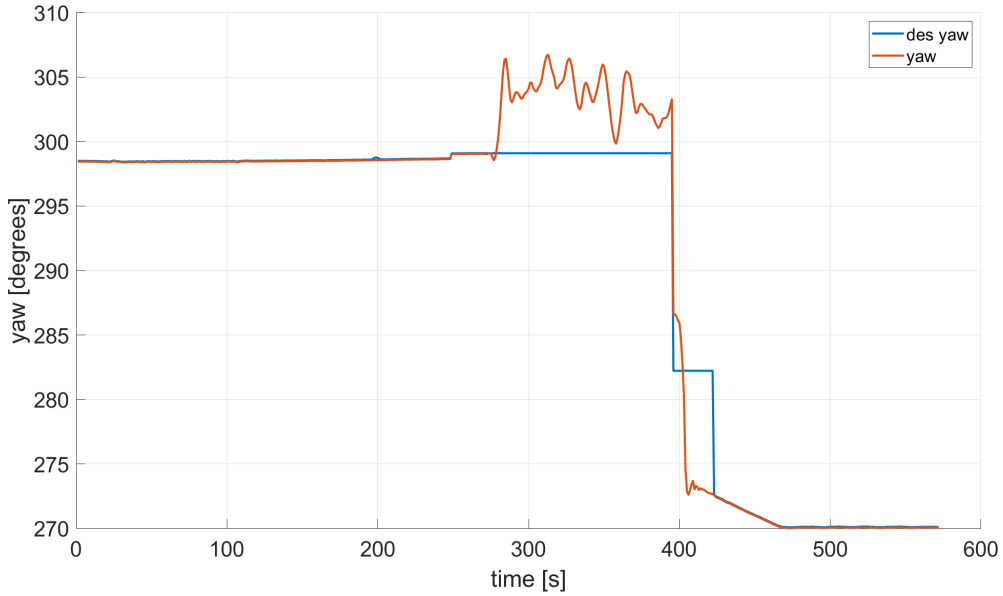

Figure 5.18: Outdoor flight

When the UAV is placed on the ground it is possible to notice, in the Roll and Pitch graphs, that there is some bias. It is certainly related to the calibration accuracy but it does not influence the overall performances of the flights. In fact, during the flight part, the UAV is able to follow the desired trajectory.
The initial and the final spikes are related to the take-off part and to the landing part.
The UAV was able to perform different way-points missions. As already stated, the performances was mainly related to the GPS performances. The UAV performed well in different atmospheric conditions, in presence of wind and in presence of external disturbances.

5.3.1 Indoor flights

The indoor flights were one of the most critical part during the development of this thesis. Indoor environments are completely opposite to outdoor environments both for the type of sensor used and for the different kind of disturbances.

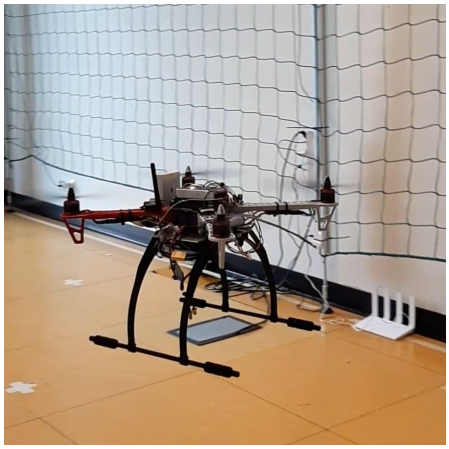

Figure 5.19: Indoor flight of the UAV.

During the tests in outdoor environment the main source of error was the GPS. However, in indoor environments, the localization position is completely provided by the UWB that is a technology that provides accurate measurements.
The main source of error is the magnetometer that provides really poor measurements. This problem can cause difficulties during the flight, in particular because the UAV tries to reach the correct position based on a wrong measure.

5.3.2 Ultra-wideband

One way to analyse the accuracy of the UAV is to fly in "LOITHER" mode. In this mode, the UAV, after the take off part, tries to hold the same position over the time.

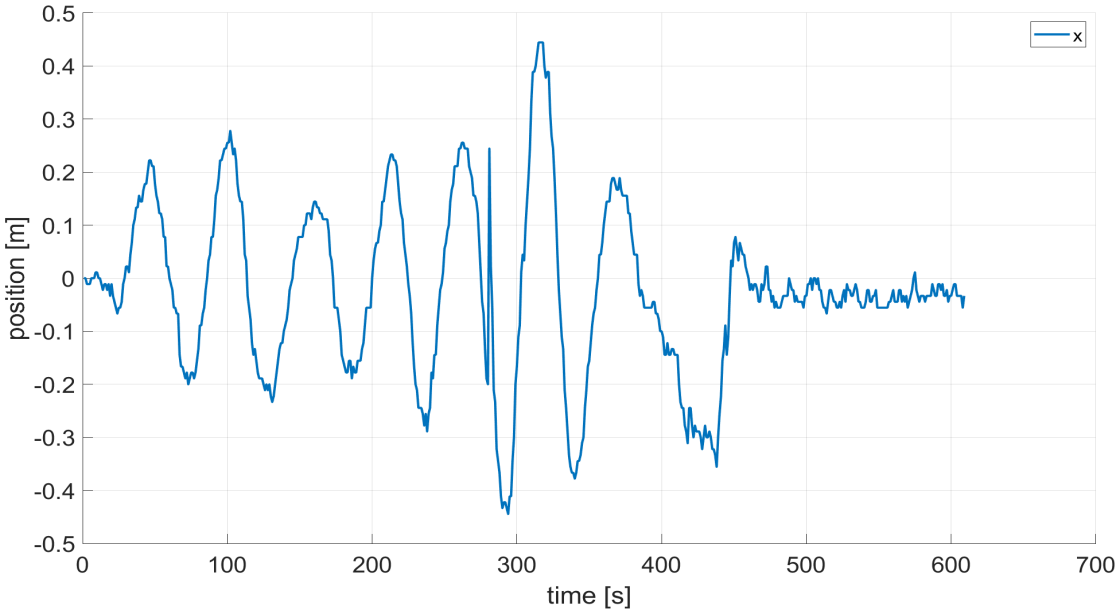

Figure 5.20: x position Ultra-wideband

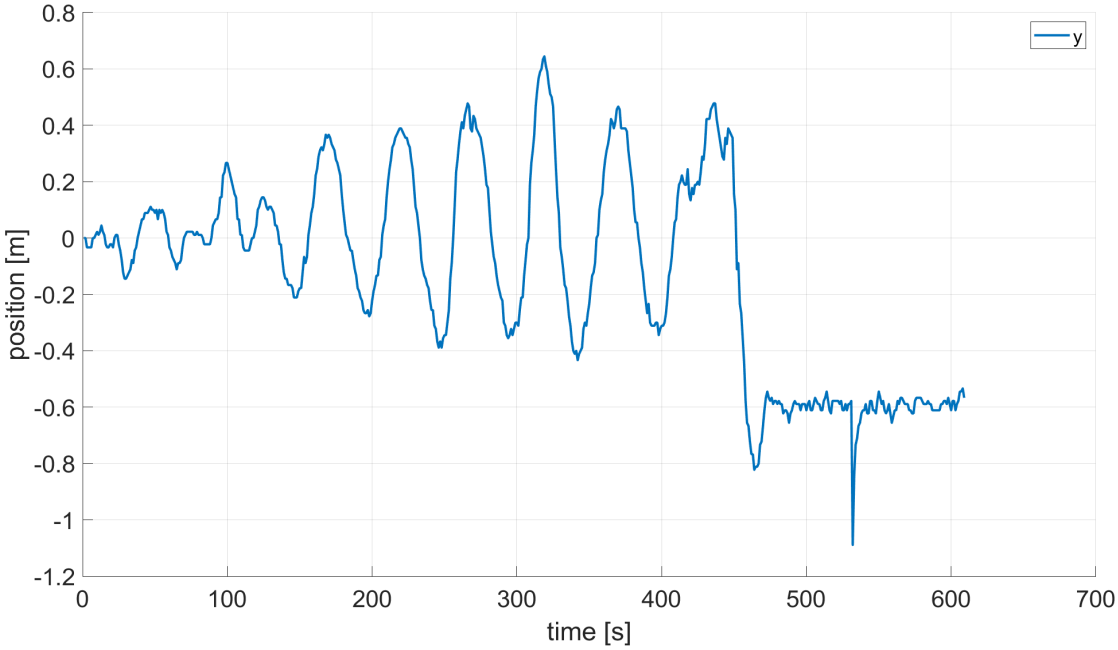

Figure 5.21: y position Ultra-wideband

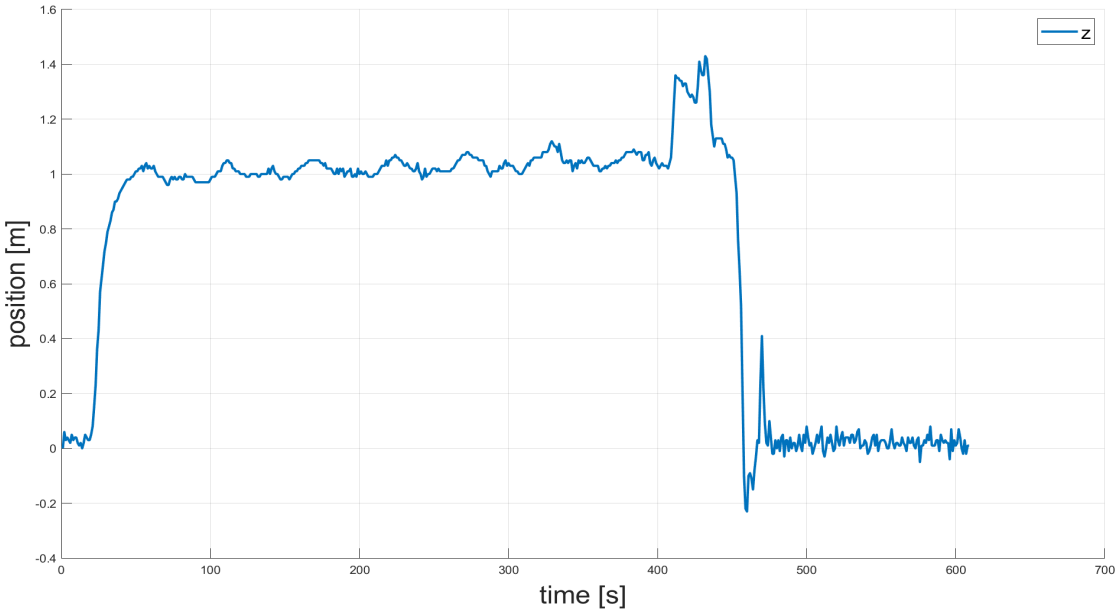

Figure 5.22: z position Ultra-wideband

The measure is really unstable but accurate.
With respect to the error provided by the GPS technology, the UWB shows good and acceptable results. The RMSE is showed in the following plot.

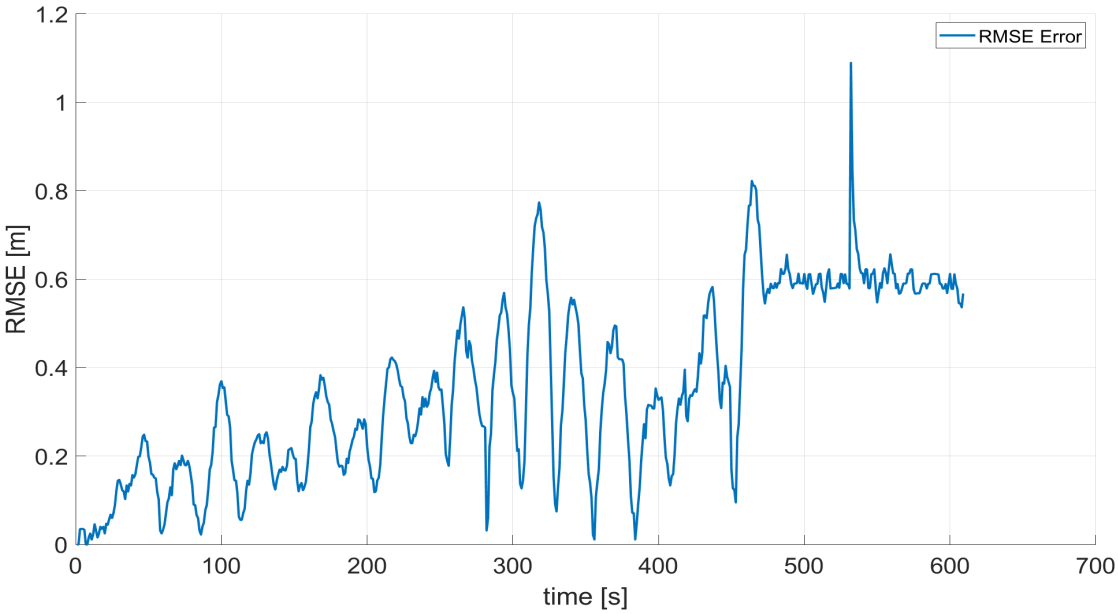

Figure 5.23: Indoor flight RMSE

Lastly, also the Roll-Pitch-Yaw analysis results to be satisfactory and it is presented using the following graphs. As it is possible to state, the UAV angles follow the desired ones with an error that is minimum.

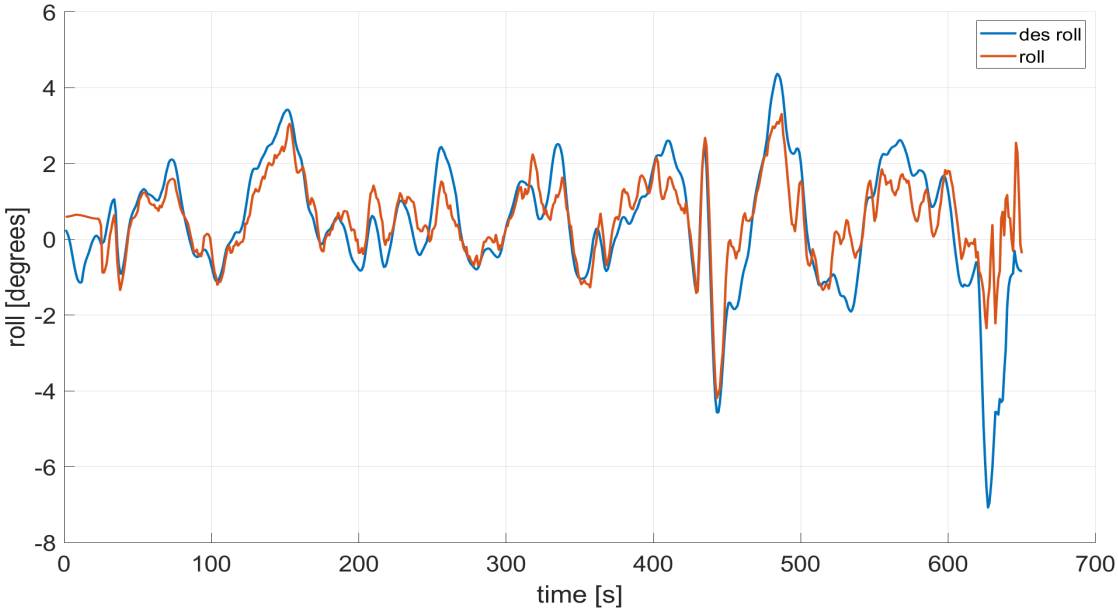

Figure 5.24: Indoor Roll Analysis

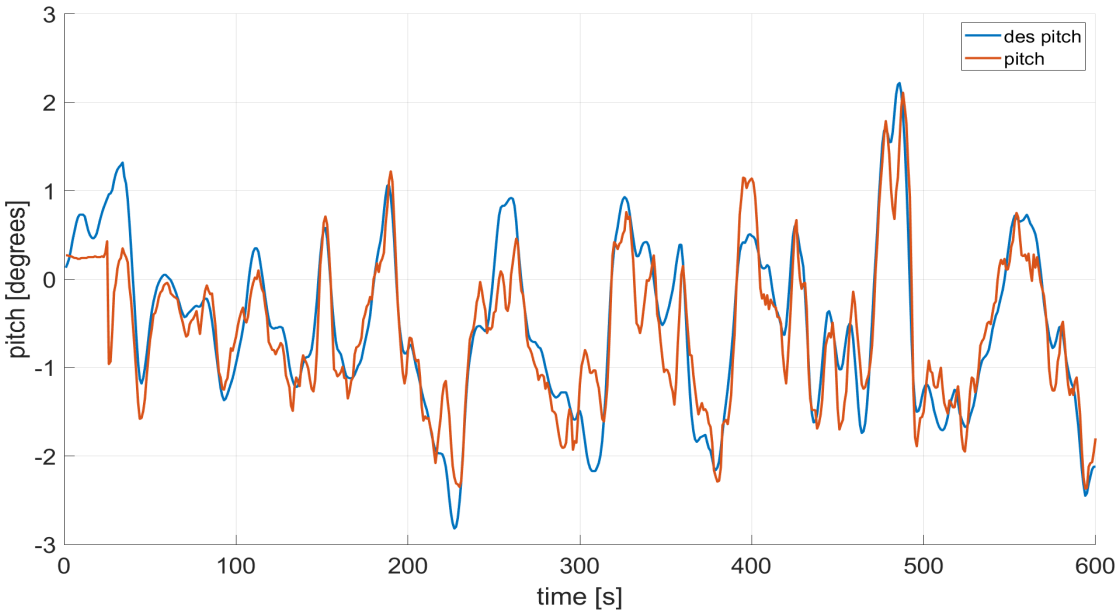

Figure 5.25: Indoor Pitch Analysis

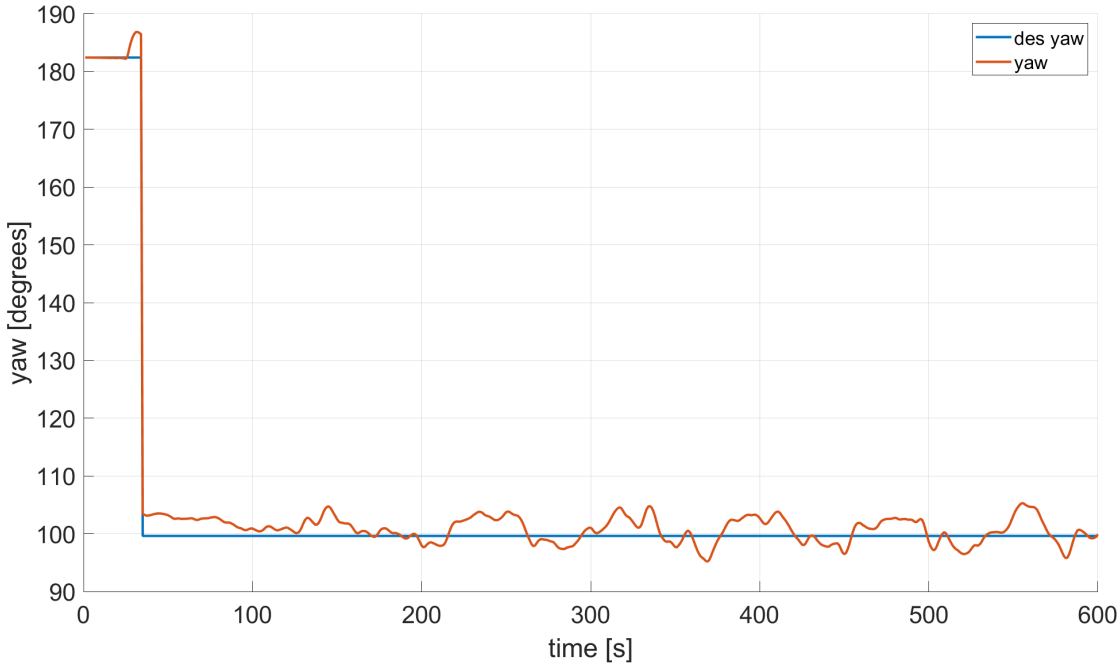

Figure 5.26: Indoor Yaw Analysis

After some time the UAV was not able to hold the same position mainly because of electromagnetic disturbances. The error accumulates over the time and the well-known phenomena appears. The UAV start to move in circular trajectories and the radious increases over the time.

5.3.3 Crash analysis

Common mechanical failures include a motor or ESC failure like, for example, including ESC sync failures, the propeller breaking or coming off.

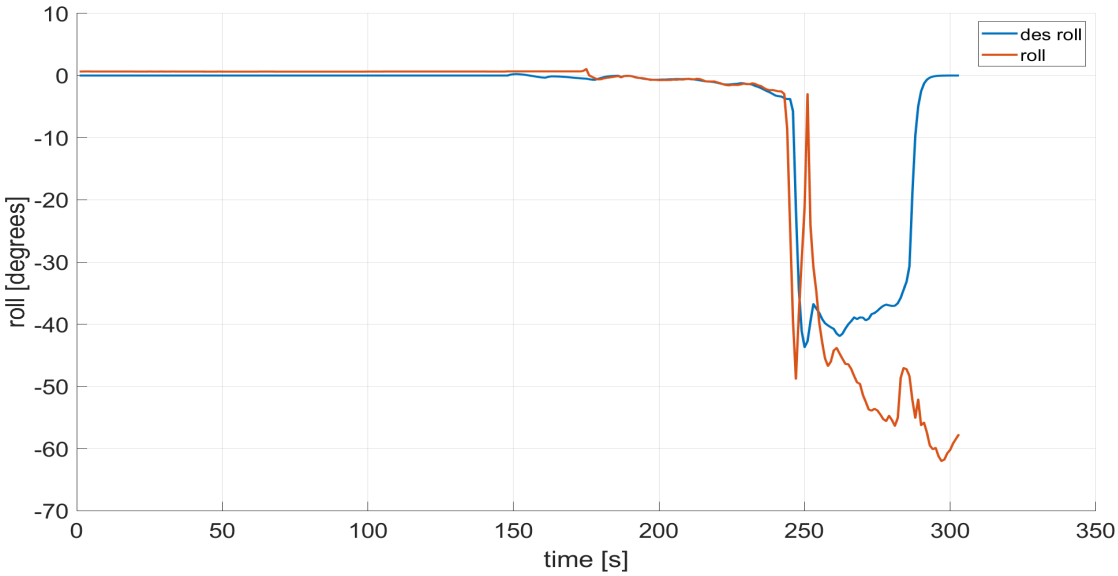

Figure 5.27: Desired roll and measured roll during the UAV crash.

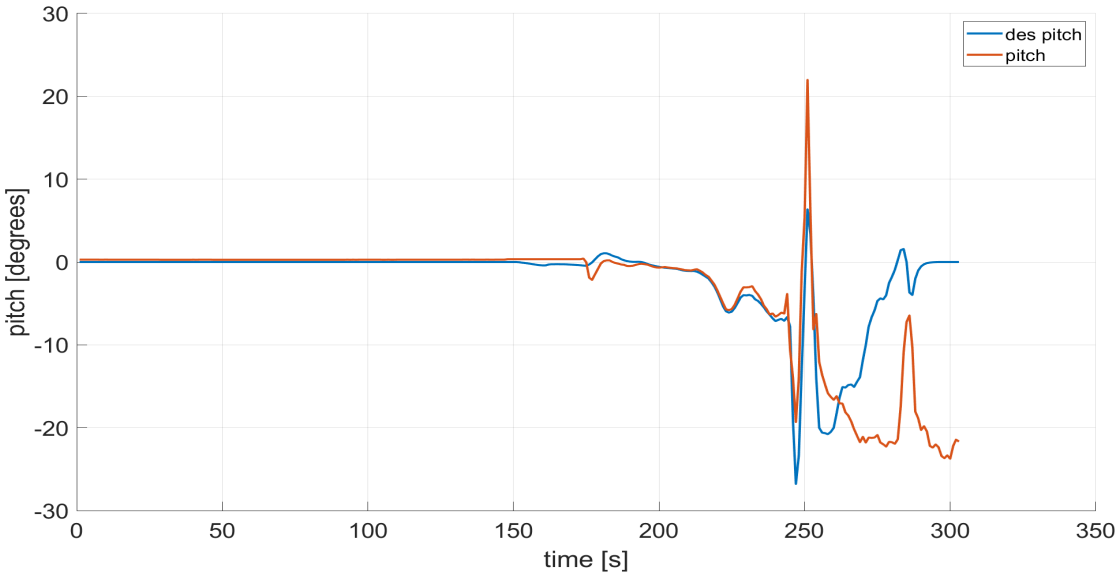

Figure 5.28: Desired pitch and measured pitch during the UAV crash.

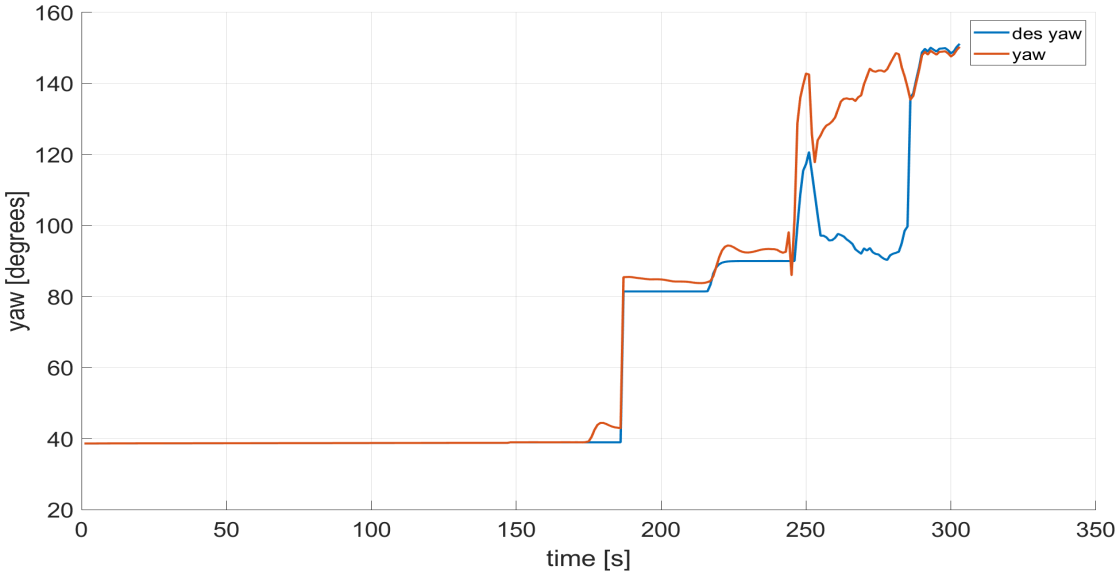

Figure 5.29: Desired pitch and measured pitch during the UAV crash.

The error in case of crash is expected to be very high. In this case the RMSE is calculated and plotted in order to be analyzed. As previously anticipated, in the moment of the crash the error grows rapidly and reaches an error greater that 5 meters.

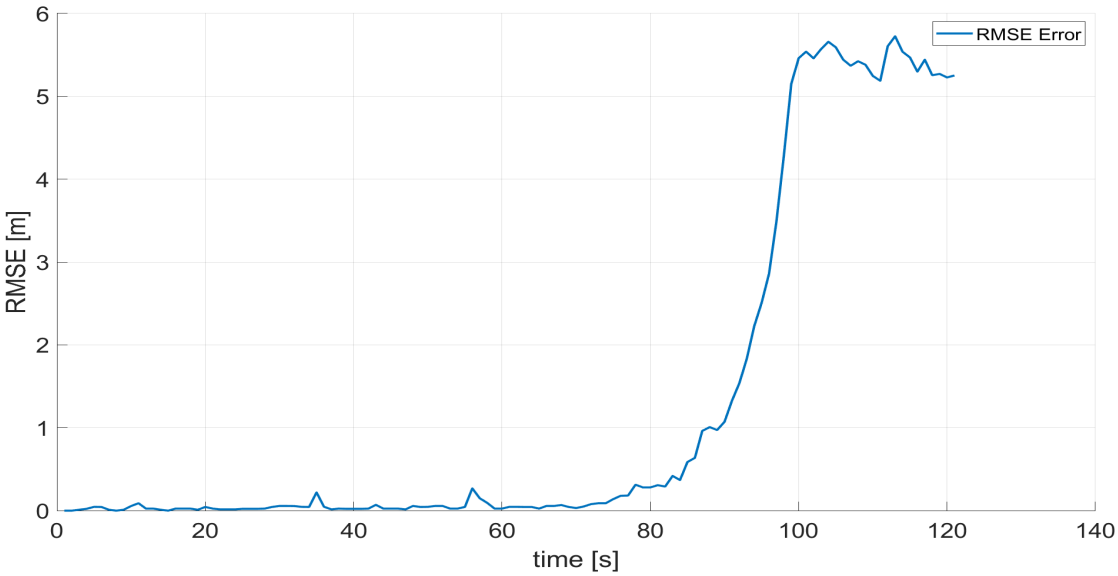

Figure 5.30: RMSE during the UAV crash.

These appear in the log file as a sudden divergence in the desired roll and pitch with respect to the vehicle's actual roll and pitch. This divergence is visible by graphing the "ATT" messages "DesRoll", "Roll", "DesPitch" and "Pitch".
In the example above the vehicle's actual roll ("Roll") closely follows the desired roll ("DesRoll") for the first part of the log but then suddenly diverges. The autopilot wanted the actual roll to remain as close as possible to the desired roll but, since it is not able to do it, it likely means that there was a mechanical failure.
Moreover, also analysing the vibrations it is possible to see that during the crash the values of the vibrations are higher that the maximum value acceptable, 30 m/s^2.

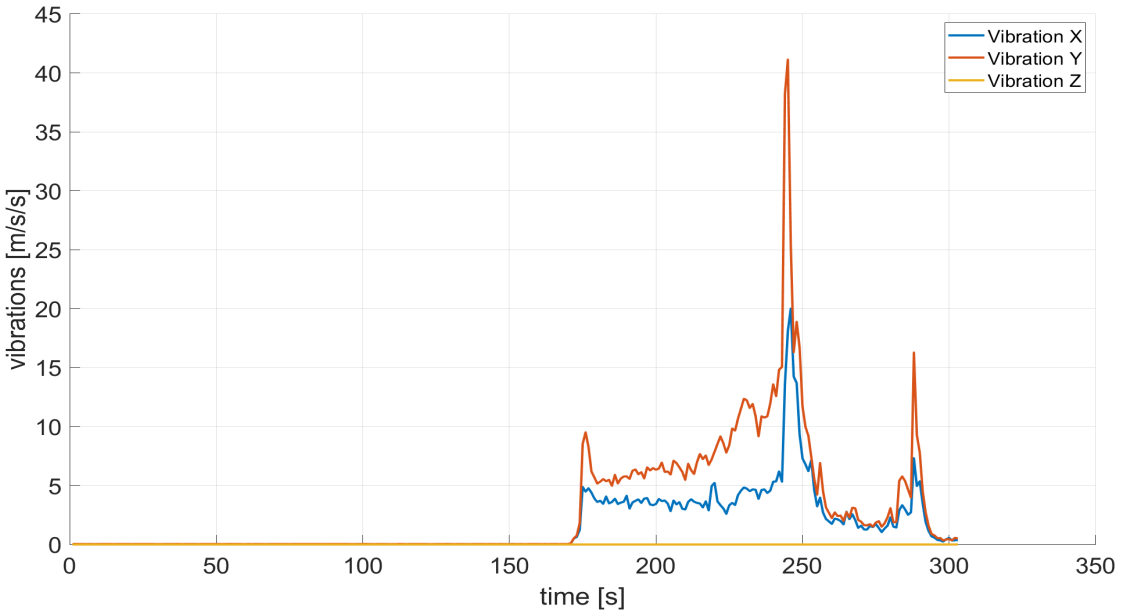

Figure 5.31: Vibrations during the UAV crash.

Being able to analyze the log files is crucial also when a crash happens. During this analysis, it is possible to identify the cause of the crash. If it is possible to isolate the cause of the crash, it is also easy to solve the problem and let the UAV flight in the best condition possible. In the previous reported test, the crash happened after a mechanical failure.
In particular, the UAV went directly in the protection cage and, for this reason, the motors blocked.

Chapter 6

Conclusions

The objective of the thesis is to customize an already existing flight control system for a commercial UAV operating in connected industrial environment. The idea is to use an UAV for supporting some maintenance procedures both in indoor and outdoor environments. UAVs are used for different purposes, some of them are presented in the analysis of the state of the art. The possibility of integrating different kind of sensors on the UAV makes it really flexible and adaptable in very different scenarios. The autopilot used is Ardupilot in order to overcome compatibility problems. It is an open-source and fully autonomous firmware that can be used also for UAVs. There are also many problems related to the use of UAVs. Some of these have been analyzed in this thesis. The first problem is related to the possibility of having a correct localization in GPS-denied environments, where the GPS (Global Positioning System) does not work. At the state of the art, there are many possibilities to overcome this problem. The final decision was to implement the Ultra-wideband technology together with two Time-of-Flight sensors. The UWB is a localization system used to have accurate coordinates in indoor environments. Its ability to resolve multi-path components makes it possible to obtain accurate location estimates without the need for complex estimation algorithms. The integration with the ToF sensors was necessary since, in the chosen configuration in the building of the network of the UWB, the measure of the altitude was not reliable. The positioning problem was solved by sending the coordinates measured from the UWB over a ROS topic, using MAVROS. Once these coordinates were published on the specific ROS topic, a MAVROS fake gps plugin was implemented to send GPS coordinates to the autopilot. In many UAVs applications, it is required to navigate in

unknown environments where moving or stationary obstacles have to be detected and avoided. UAVs must have the ability to autonomously plan trajectories in order to avoid collisions. In the literature, many solutions have been proposed for implementing the path planning algorithm of UAVs. Some tests will be performed in order to test the obstacle avoidance algorithm. At the end of this thesis, simulations and tests results are provided. Different analysis of the UAV performances in different scenarios have been performed. The real test showed that the UAV was able to fly both indoor and outdoor. The main differences between the indoor tests and the outdoor tests were given just by the accuracy of the sensors and by the different technologies involved. Different GPS for the outdoor navigation have been tested, all showing different results. Many considerations are reported in this thesis on the best choice. In all the cases, the UAV showed a similar behaviour with respect to the simulations. One interesting comparison have been done plotting the RMSE of the all GPS tested and the RMSE of the UWB. It is possible to see that the mean RMSE of the UWB is similar to the one of the RTK GPS and both showed an analogous behaviour. Both in indoor and outdoor environments the UAV showed good performances and all the choices made revealed to be valid for the desired applications of the UAV.

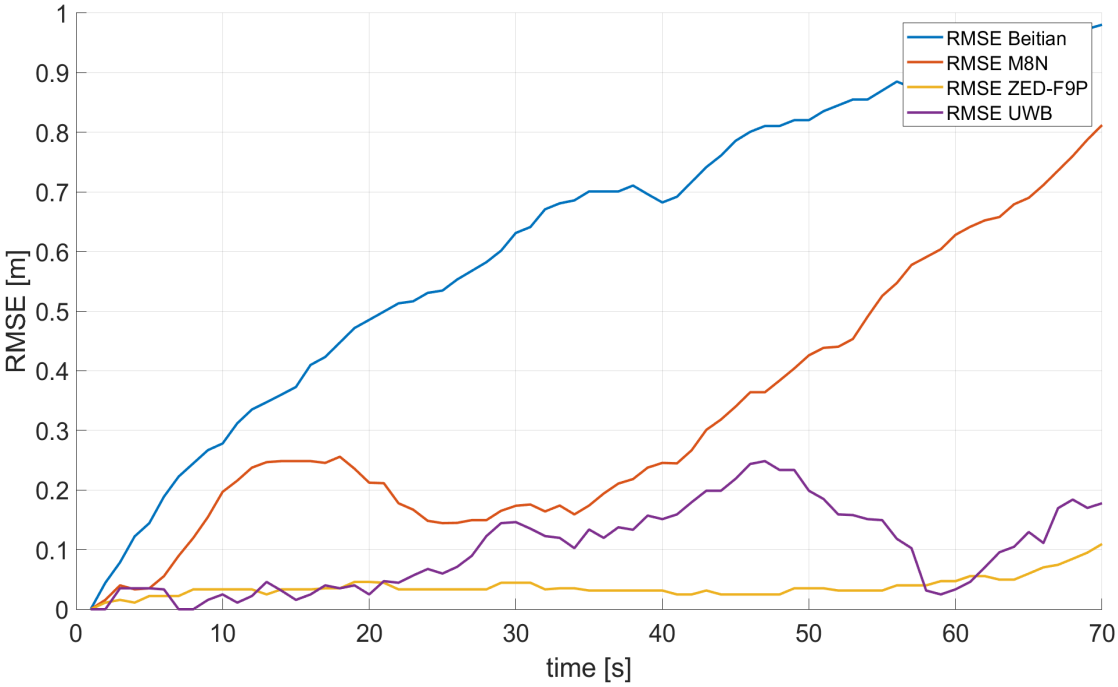

Figure 6.1: Root Mean Square Error GPS comparison

Nowadays, many projects develop an interaction of an UAV with a rover. The first approach has been to study the two parts independently from each other and, subsequently, an approach to integrate the two will be developed. In fact, this thesis is only the first step of many other projects that will be developed at CIM 4.0.
For the future, at least a complete integration with ROS 2 will be implemented, more senors may be integrated concerning different scenarios not yet considered in this thesis and a complete upgrade from Ubuntu 18.04 to Ubuntu 20.04 must be taken into account, in order to keep the software up to date and to introduce also new features like a vocal control for the UAV. Furthermore, another goal of this project will consist in the autonomous landing on a platform, that can be either fixed or moving. The integration with the rover is a key point of this project. The UAV should be able to communicate in real-time with the rover, understanding what is its position and, then, both together they shall work to complete a particular task in an industrial environment.

Bibliography

[1] Yohanes Khosiawan and Izabela Nielsen. «A system of UAV application in indoor environment». In: *Production & Manufacturing Research* 4.1 (2016), pp. 2–22. DOI: 10.1080/21693277.2016.1195304. eprint: https://doi.org/10.1080/21693277.2016.1195304. URL: https://doi.org/10.1080/21693277.2016.1195304 (cit. on p. 9).

[2] Amina Fellan, Christian Schellenberger, Marc Zimmermann, and Hans D. Schotten. «Enabling Communication Technologies for Automated Unmanned Vehicles in Industry 4.0». In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. 2018, pp. 171–176. DOI: 10.1109/ICTC.2018.8539695 (cit. on p. 11).

[3] Yasir Mohd Mustafah, Amelia Wong Azman, and Fajril Akbar. «Indoor UAV Positioning Using Stereo Vision Sensor». In: *Procedia Engineering* 41 (2012). International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012), pp. 575–579. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2012.07.214. URL: https://www.sciencedirect.com/science/article/pii/S1877705812026148 (cit. on pp. 12, 15).

[4] Bartosz Brzozowski, Krzysztof Kaźmierczak, Zdzisław Rochala, Marta Wojda, and Konrad Wojtowicz. «A concept of UAV indoor navigation system based on magnetic field measurements». In: *2016 IEEE Metrology for Aerospace (MetroAeroSpace)*. 2016, pp. 636–640. DOI: 10.1109/MetroAeroSpace.2016.7573291 (cit. on p. 12).

[5] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. «Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances». In: *Sensors* 16.5 (2016). ISSN: 1424-8220. DOI:

10.3390/s16050707. URL: https://www.mdpi.com/1424-8220/16/5/707 (cit. on pp. 12, 16, 17).

[6] S. Gezici, Zhi Tian, G.B. Giannakis, H. Kobayashi, A.F. Molisch, H.V. Poor, and Z. Sahinoglu. «Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks». In: *IEEE Signal Processing Magazine* 22.4 (2005), pp. 70–84. DOI: 10.1109/MSP.2005.1458289 (cit. on p. 12).

[7] Hamza Soganci, Sinan Gezici, and H. Vincent Poor. «Accurate positioning in ultra-wideband systems». In: *IEEE Wireless Communications* 18.2 (2011), pp. 19–27. DOI: 10.1109/MWC.2011.5751292 (cit. on pp. 12, 16, 17).

[8] G.R. Aiello and G.D. Rogerson. «Ultra-wideband wireless systems». In: *IEEE Microwave Magazine* 4.2 (2003), pp. 36–47. DOI: 10.1109/MMW.2003.1201597 (cit. on p. 12).

[9] M. Farrokhsiar, D. Krys, and H. Najjaran. «A teaching tool for the state-of-the-art probabilistic methods used in localization of mobile robots». In: *Computer Applications in Engineering Education* 20.4 (2012), pp. 721–727. DOI: https://doi.org/10.1002/cae.20443. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.20443. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.20443 (cit. on p. 12).

[10] Gábor G. Varga, András Kondákor, and Márton Antal. «Developing an Autonomous Valet Parking System in Simulated Environment». In: *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2021, pp. 000373–000380. DOI: 10.1109/SAMI50585.2021.9378642 (cit. on pp. 18, 19, 47).

[11] Miroslav Kulich, Viktor Kozák, and Libor Přeučil. «Comparison of Local Planning Algorithms for Mobile Robots». In: *Modelling and Simulation for Autonomous Systems*. Ed. by Jan Hodicky. Cham: Springer International Publishing, 2015, pp. 196–208. ISBN: 978-3-319-22383-4 (cit. on pp. 20, 21).

[12] Greg Welch, Gary Bishop, et al. «An introduction to the Kalman filter». In: (1995) (cit. on p. 23).

[13] Nulee Jeong, Hyun Hwang, and Eric T. Matson. «Evaluation of low-cost LiDAR sensor for application in indoor UAV navigation». In: *2018 IEEE Sensors Applications Symposium (SAS)*. 2018, pp. 1–5. DOI: 10.1109/SAS.2018.8336719 (cit. on p. 32).

Appendix A

Python Script

```
#!/usr/bin/env python
# Author: Andrea Colucci
# Master thesis at CIM 4.0 — Torino 2021
# This code reads local position coordinates from an UWB (DWM1001c – Qorvo)
# and then publishes it onto the topic called "/mavros/fake_gps/mocap/tf"
# created from a mavros plugin in order to be able to send gps coordinates
# to the onboard Pixhawk and be able to fly in indoor environments
import time
import thread
import serial
import rospy
from pkg.msg import TransformStamped
# INITIALIZATION #
fix_obj = TransformStamped()
fix_obj.header.frame_id = "map"
fix_obj.child_frame_id = "base_link"
fix_obj.transform.rotation.x = 0
fix_obj.transform.rotation.y = 0
fix_obj.transform.rotation.z = 0
fix_obj.transform.rotation.w = 1
# FIRST THREAD #
def print_xy(threadName, delay, port):
    global fix_obj
    DWM = serial.Serial(port="/dev/ttyACM0", baudrate=115200)
```

```
25      time.sleep(1)
26      print("Connected to " + DWM.name)
27      DWM.write("\r\r".encode())  # Initializes the tag
28      time.sleep(1)
29      DWM.write("lec\r".encode())  # Standard message to read the
    coordinates
30      time.sleep(1)
31      while not rospy.is_shutdown():
32          try:
33              line = DWM.readline()
34              #print(line)
35              if(line):
36                  if (len(line) >= 140) and ("POS" in line):  #
    evaluate whether it reads correctly or not
37                  parse = line.decode().split(",")
38                  x_pos_uwb = float(parse[parse.index("POS") + 1])
      #
39                  y_pos_uwb = float(parse[parse.index("POS") + 2])
      #
40                  z_pos_uwb = float(parse[parse.index("POS") + 3])
      #
41                  pos = (x_pos_uwb, y_pos_uwb, z_pos_uwb)
42                  try:
43                      #print(x_pos_uwb)
44                      #print(y_pos_uwb)
45                      fix_obj.transform.translation.x = x_pos_uwb
46                      fix_obj.transform.translation.y = y_pos_uwb
47                  except:
48                      pass
49              else:
50                  pass
51                  #print("POS not in line")
52              else:
53                  print("Position missed")
54          except Exception as ex:
55              print(ex)
56              break
57  # SECOND THREAD #
58  def print_z(threadName, delay, port):
59      global fix_obj
60      ser = serial.Serial(port="/dev/ttyUSB0", baudrate=115200)
61      while 1:
62          try:
63              print("Connected to " + ser.name)
64              altitude_string= ser.readline()
```

```
65                dist_vl53l1x = float(altitude_string.decode())
66                alt_vl53l1x=float (dist_vl53l1x/1000)
67                alt = round(3.0-alt_vl53l1x,3)
68               #print(alt)
69                fix_obj.transform.translation.z = alt
70            except:
71                pass
72        ser.close()
73
74 # ROS PUBLISHER #
75 def talker():
76     global fix_obj
77     pub = rospy.Publisher('/mavros/fake_gps/mocap/tf',
       TransformStamped, queue_size=100)  # create a ROS publisher
78     rospy.init_node('talker_gps_fix', anonymous=True)
79     rate = rospy.Rate(10)  # 10 Hz -- Frequency of the GPS
80     if not rospy.is_shutdown():
81         try:
82             thread.start_new_thread( print_xy, ("Thread_xy", 0, " ")
       )
83             thread.start_new_thread( print_z, ("Thread_z", 0, " "))
84         except:
85             print "Error: unable to start thread"
86     while not rospy.is_shutdown():
87         pub.publish(fix_obj)
88         rate.sleep()
89 # MAIN #
90 if __name__ == '__main__' :
91         try:
92             talker()
93         except rospy.ROSInterruptException:
94             pass
```

Appendix B

Simulation log

```
Size (kb) 20046.6962890625
No of lines 230705
Duration 0:07:30
Vehicletype ArduCopter
Firmware Version V4.1.0-dev
Firmware Hash 8a3a609e
Hardware Type
Free Mem 0
Skipped Lines 0
Test: Brownout = GOOD -
Test: Compass = GOOD - mag_field interference within limits
    (0.24%)
Max mag field length (584.56) > recommended (550.00)

Test: Dupe Log Data = GOOD -
Test: Empty = GOOD -
Test: Event/Failsafe = GOOD -
Test: GPS = GOOD -
Test: IMU Mismatch = NA -
Test: Motor Balance = UNKNOWN - 'QUAD/X'
Test: NaNs = FAIL - Found NaN in CTUN.DSAlt

Test: OpticalFlow = FAIL - 'FLOW_FXSCALER' not found
Test: Parameters = FAIL - 'MAG_ENABLE' not found
Test: PM = GOOD -
Test: Pitch/Roll = GOOD -
Test: Thrust = GOOD -
Test: VCC = UNKNOWN - No CURR log data
```