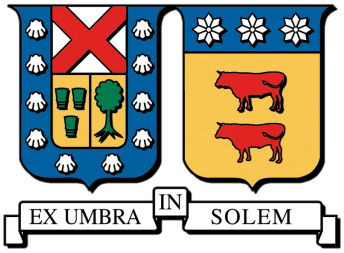
**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE




# DESIGN AND IMPLEMENTATION OF AN INERTIAL NAVIGATION SYSTEM USING KALMAN FILTER

CECILIA ALEJANDRA VILLARROEL GONZÁLEZ

THESIS AS A PARTIAL REQUIREMENT IN FULFILLMENT TO
OBTAIN THE DEGREE OF
INFORMATICS ENGINEER

ADVISOR: MARCELO MENDOZA
CO-ADVISOR: JOSÉ LUIS MARTÍ LARA

DECEMBER - 2017

# Acknowledgements

I wish to acknowledge all the help provided by the FabLab UTFSM, and to thank all the people on it, for their patience, guidance, enthusiastic encouragement and useful critiques of this research work. Also, my grateful thanks to Professor Marcelo Mendoza for his advice and patience throughout the development of this project.

I would also like extend my deepest gratitude to all the amazing people that has been one way or another throughout this whole journey, and that they have helped to make the daily day infinitely special and memorable. I will forever be grateful for the multiple pieces of advice, the infinite shared laughs, the shared screams at the top of our lungs and the no-questions-asked support, that have helped me finish what I once started.

Finally, I wish to thank my parents for their constant support, their sacrifice and their encouragement throughout the years, and for trusting me and giving me the tools to try to find my place in this crazy world.

*"If you're going to try, go all the way. Otherwise, don't even start. if you're going to try, go all the way. This could mean losing girlfriends, wives, relatives, jobs and maybe your mind. Go all the way. (...) you will ride life straight to perfect laughter, its the only good fight there is." C. Bukowski, Roll the dice.*

# Abstract

The basis of this project is to be able to track a person in an outdoor environment. For this purpose, an Inertial Navigation System is proposed and the feasibility of its implementation is studied. The structure of the proposed INS is divided in four subsystems: hardware, signal filtering, signal processing and software. In the context of this project no external references, as for example GPS, can be used.

This project is focused on the position and orientation calculation. First, the raw acceleration signals are filtered by three different methods: Kalman filter, Noise filter and Simple Average Moving filter. Giving special emphasis to the Kalman filter. Next, the filtered signal is integrated by the trapezoidal rule. The position is obtain with great drift. An already proposed slow-pass filter is used to diminish the accumulated error that it is obtained through the integration, improving the results, but not enough.

Finally, the orientation is calculated implementing Kalman filter to be able to subtract gravitational acceleration and to obtain gravity-free acceleration. Results turns out to be unsatisfactory.

The acceleration and angular velocity is obtained through arduino UNO with an ATmega328p microcontroller and the six-axis sensor, MPU6050. A 3D printed platform is used only for testing purposes and a rail is set up on the ground to be able to compare more objectively the filters used.

# Resumen

La base de este proyecto es poder rastrear a una persona en un ambiente al aire libre. Para este propósito, se propone un sistema de navegación inercial y la factibilidad de implementación es estudiada. La estructura del INS propuesto se divide en cuatro subsistemas: hardware, filtrado de señal, procesamiento de señal y software. En el contexto de esto proyecto, no se pueden usar referencias externas, como el GPS.

Este proyecto se enfoca en el cálculo de posición y orientación. Primero, la señal de la aceleración bruta es filtrada por tres métodos diferentes: filtro de Kalman, "Noise filter" y filtro de Media Movil Simple. Dando especial énfasis al filtro de Kalman. Luego, la señal filtrada se integra mediante la regla trapezoidal. La posición obtenida presenta altas desviaciones llamadas "drift". Un filtro de paso lento ya propuesto se usa para disminuir el error acumulado que se obtiene a través de la integración, mejorando los resultados, pero no lo suficiente.

Finalmente, la orientación se calcula implementando el filtro de Kalman, para poder restar la aceleración gravitacional y obtener la aceleracion libre de gravedad. Los resultados resultan ser insatisfactorios.

La aceleración y la velocidad angular se obtienen a través de Arduino UNO con un microcontrolador ATmega328p y el sensor de seis ejes, MPU6050. Una plataforma impresa 3D es utilizada solo para fines de prueba y un riel es posicionado en el suelo para poder comparar de manera objetiva los filtros usados.

# Glossary

The following is a brief description of the acronyms and terms used in this report:

- **ALMA (Atacama Large Millimeter/submillimeter Array):** astronomical interferometer of radio telescopes in the Atacama desert of northern Chile.
- **AOS (Array Operations Site):** technical Building that houses the ALMA Correlator. Human operations at the AOS is limited to an absolute minimum, due to the high altitude [1].
- **FFT (Fast Fourier Transform):** algorithm that samples a signal over a period of time and divides it into its frequency components, to then filter them.
- **FIR (Finite Impulse Response):** in the context of this project, it is a filter that does not use feedback, thus its output always becomes zero after putting $N$ samples of an impulse.
- **GPS (Global Positioning System):** space-based radio-navigation system which is owned by the USA. It provides radio waves from a plurality of satellites received by the GPS receiver outputting either two-dimensional or three-dimensional position data indicative of the present position of the mobile object.
- **IDE (Integrated Development Environment):** software suite that consolidates the basic tools developers need to write and test software. Normally, it contains a code editor, a compiler or interpreter and a debugger that the developer can access it through a graphical user interface (GUI).

- **IIR (Infinite Impulse Response):** in the context of this project, it is a filter which as its names states, continues indefinitely outputting infinite non-zero terms if an impulse is put.
- **IMU (Inertial Measurement Unit):** electronic device which provides three axis acceleration and angular turning rate detection with a cubical magnetically suspended sensor mass disposed within a cubical outer assembly [25].
- **INS (Inertial Navigation System):** self-contained navigation technique in which accelerometers and gyroscopes provide measurements that are used to track the position and orientation of an object relative to a known starting point, orientation and velocity, in the absence of external references [31].
- **IRU (Inertial Reference Unit):** high performance gyro attitude reference system for use on the spacecrafts. The IRU is a three axis system, which provides both rate and attitude information for spacecraft control [22].
- **LSB (Least Significant Bit):** lowest bit in a series of numbers in binary.
- **MEMS (Microelectromechanical Systems):** technology which in its most general form can be defined as a miniaturized mechanical and electro-mechanical elements that are made using microfrabrication techniques. Known by its dimensional spectrum, that can vary from below one micron to several millimeters.
- **ZUPT (Zero Velocity Update):** method that is used to help constrain some errors when the position of a pedestrian is needed. In the walking pattern exists a stance phase where theoretically the velocity is zero and any nonzero measurements during this moment can therefore be assumed to be errors [11].

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This project is proposed from the insides of ALMA observatory, where it was offered as a theme to be developed as a final study project.

## 1.1 Motivation

The north of Chile is an especially good space for astronomical observations, either because of its great drought, its high altitude, its low number of clouds, a low luminous pollution or a low interference of radio signals coming from the cities. All of this allows to obtain the observations that today, are used all over the world. However, these advantages bring with them human-hostile conditions, such as lack of oxygen due to altitude, low humidity and extreme climates. Given this, the first concern is to maintain the safety of all workers.

Around fifty workers go up and down during the day to the AOS (Array Operations Site) which is located at 5000 meters above sea level, where each one of them must go under a rigorous access control that is planned and constructed to carry a manual history, for any possible emergency. Now, this process is to be done manually, there is no real-time monitoring of the workers activities during their time in the AOS, which could bring vital consequences for the safety of employees in any extreme weather event, where the communication and location of the people working could be lost.

This project motivation is based in the feasibility of the application of low-cost MEMS to be able to track the whereabouts of the ALMA workers that are performing their duty at high altitudes.

## 1.2 Problem definition

Trough the years there has been an increase of the necessity of knowing the position and orientation of different objects in many different fields. To be able to measure the position of a body, devices to measure acceleration are more convenient than the devices that are used to measure position directly. Thus, to obtain displacement from acceleration a double integration must the applied, nonetheless this generates two important problems which causes major integration errors: drift associated to accelerometers and the initial condition of the system are unknown [27].

A common proposed solution is the integration between a GPS and INS when the location must be done in an outdoor environment, which has been studied and applied, giving satisfactory results. And for indoors, some propose a solution by mounting the sensors on the pedestrians foot, using zero velocity update (ZUPT), which its able to deliver a good accuracy of the pedestrian displacement by recognizing the walking pattern and calculating the exact moment when there is no displacement of the foot and it can update velocity to zero. Nonetheless, in the context of this project, the use of GPS is not possible due to possible interference between the antennas signals of the observatory and it is not viable to implement a ZUPT because the workers move from the initial check control in a vehicle and then walk to specific places, thus its application would not be of use in this case.

## 1.3 Proposed solution

As a start, an already structured monitoring system is given where the idea behind it is that it can directly track the activity of workers in real time. This is divided in

three fundamental parts:

1. Inertial tracking subsystem
2. Communication subsystem
3. Platform development (data visualization).

For the purposes of the present project, attention will be focused on the feasibility of the first part by proposing an Inertial Navigation System on which Kalman filter will be apply.

Kalman filter is an algorithm widely known thanks to its many applications in technology. Commonly used in guidance, navigation and control of vehicles, being a recursive algorithm, it can be implemented in real time using only the current measurement, the previous calculated state and an uncertainty matrix.

Finally, the positioning problem has been approached, but most of the times the position estimation must be done in a controlled environment, between small periods of time, known patterns of behaviour or not in real time.

## 1.4 Project objectives

- **Overall goal**
  - To deliver reliable estimated position, based on accelerometer measurements.
- **Specific goals**
  - To analyze feasibility of using Kalman filter as a noise filter for the acceleration.
  - To implement Kalman filter as a sensor fusion algorithm.
  - To analyze feasibility of calculating free-gravity acceleration with sensor orientation.

The overall goal is to be able to deliver a reliable estimated position, through the design and implementation of an INS. Starting with the feasibility analysis of using Kalman filter as a noise filter for the raw values outputted by the accelerometer. Then, the implementation of the Kalman filter as a sensor fusion algorithm to be able to obtain orientation using both values, from the accelerometer and gyroscope. Finally, with this orientation, the feasibility analysis of subtracting the gravitational force that affects the accelerometer and be able to calculate free-gravity acceleration.

## 1.5 Document organization

The present report it is composed by four chapters, detailed below:

- In this first chapter, the motivation behind this project is presented. Then the main problem and its context is addressed, following the proposed solution next to the main and specific goals of this project.
- In Chapter 2, it is proposed a structure for the INS to develop, which is divided into four subsystems: Hardware, Signal Filtering, Signal Processing and Software. Where for each one, its background is addressed.
- In Chapter 3, the implementation of the proposed INS structure in the previous chapter is put to the test, next to the given results for each part.
- Finally in Chapter 4, the relevant conclusions are presented.

# Chapter 2

# Background

## 2.1 What is an INS?

An Inertial Navigation System is a self-contained navigation technique in which accelerometers and gyroscopes provide measurements that are used to track the position and orientation of an object relative to a known starting point, orientation and velocity [31], in the absence of external references such as global positioning systems (GPS). This system makes use of a computer for processing the data given by the inertial measurements units, which are electronic devices that normally contain a 3-axis accelerometer and 3-axis gyroscope, which measures linear acceleration and angular velocity respectively.

The INS is used in a wide range of applications, such as aircraft navigation [16], guidance in human spaceflight [18] and guided missiles [15] among others. The INS consists in the following [21]:

1. An inertial measurement unit (IMU) or inertial reference unit (IRU) containing accelerometers and gyroscopes, usually of three axis each. The sensors are attached to a common base to be able to maintain the same relative orientations.

2. Navigation computers, to calculate the double integrate of the acceleration to get an estimate of the position.

Thanks to the advancements of technology, it has been possible the construction of microelectromechanical systems (MEMS), which are smaller and lighter Inertial Measurements Units (IMU) which has made possible to apply motion capture to smaller bodies such as humans, animals and basically any moving device.

## 2.2 Scheme of an INS

Based of the definition made in the previous section, the INS it is already defined. Basically, the INS is composed by sensors and a processor unit that is able to translates the output given by the sensors. With this in mind, the following schema is proposed, Fig. 2.1. The purpose of this is to be able to go deeper and be able to study each part more thoroughly.

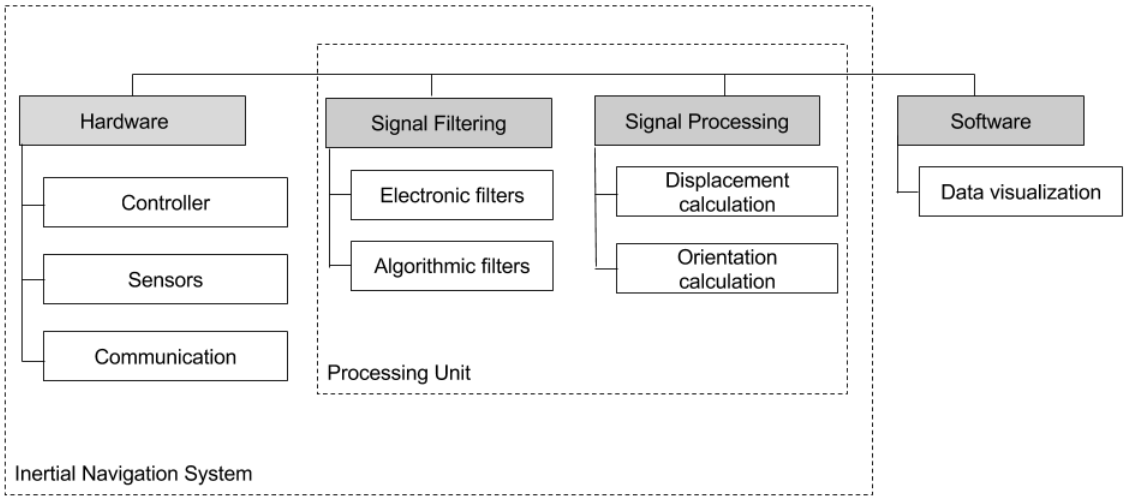



Figure 2.1: Proposed scheme of the INS

The scheme is broken down into four sub-systems. In the first one, it is addressed the hardware and their respective specifications that were used for this project. The second one refers to the signal filtering in which it will be discussed the application of Kalman filter to smooth the noise of the data delivered by the sensor. The third one, signal processing, different methods to transform the raw data from the sensor into

usable information are explained and finally, in the last subsystem, it is mentioned the software used for the data visualization of the transformed data.

## 2.3 Hardware

For this first section it is covered the different parts of the hardware used for the INS and this particular project.

### 2.3.1 Controller: Arduino

Arduino is an open source platform based on easy-to-use hardware and software, which is used to build electronic projects. It consists of two fundamental parts, a physical programmable circuit board and an IDE that can be simply installed in a computer to write and upload code to the physical board.

Arduino/Genuino UNO, Fig. 2.2, is the most used and documented board of the Arduino family. This microcontroller board is based on the ATmega328p, which is a single-chip microcontroller that it was created for Atmel. The arduino UNO has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button, as it can be seen in the features summary in Table 2.1. It can be directly connected to the computer via USB connection, or it can be powered with a AC-to-DC adapter or battery [10].

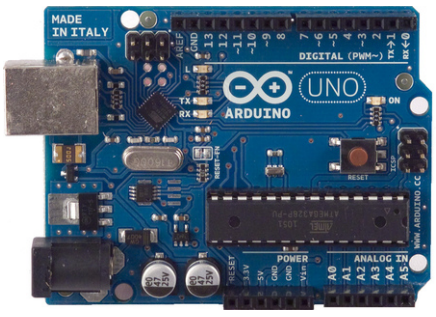



Figure 2.2: Arduino UNO

Features summary:

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage | 7-12V |
| Input Voltage | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50mA |
| Flash Memory | 32 KB (ATmega328) - 0.5 KB is used by bootloader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

Table 2.1: Arduino UNO features summary

For further information, see [10].

### 2.3.2 Inertial Sensors MEMS

A sensor is understood as the element in charge of transforming some chemical or physical stimulus into electrical energy. Its function is to measure some physical variable of interest and generate an electrical signal, commonly expressed as voltage, that can be processed by a unit of computation [12].

Microelectromechanical systems or MEMS are devices that are built with microengineering, where any type of energy is converted into electrical energy [14]. Usually, it consists of a microprocessor and different micro-sensors that are able to

interact with the environment and pass the information that is being captured.

The advantages of MEMS devices is that are small, light, more affordable and have low power consumption and start-up times. One of the most significant disadvantage is that they are not as accurate, as for example, accelerometers manufactured using traditional techniques [31].

The INS focuses in two types of sensors: the accelerometer and the gyroscope.

### 2.3.3 Accelerometer

An accelerometer is a device which can measure the acceleration. This acceleration is not necessarily the same as an object moves trough space but it is the phenomenon of experienced weight by a mass that is in the reference frame of the device. This means that it measures acceleration trough the measure of the force, which is by sensing how much a mass presses on something when a force acts on it. For example, when the sensor is still on the Earth's surface, it will measure an acceleration due to Earth's gravity $\approx 9.81[m/s^2]$.

Basically the functioning of the accelerometer can be explained as cube with a floating ball inside, where axis are designated to the sensitive walls of the cube and without a gravitational field [12], as seen in Fig. 2.3.

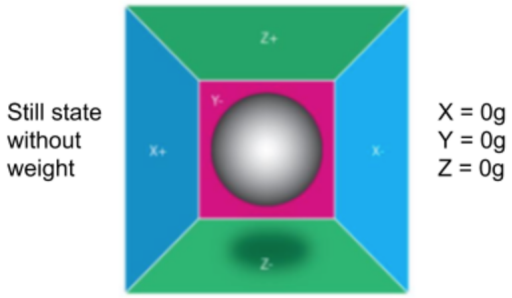



Figure 2.3: Practical illustration of the functioning of an accelerometer

On the other hand, if the sensor experiences a sudden movement to the left, the

inside ball will touch the $x$ wall with an acceleration of, for example, $1g = 9.81m/s^2$, as it can be observed in Fig. 2.4.

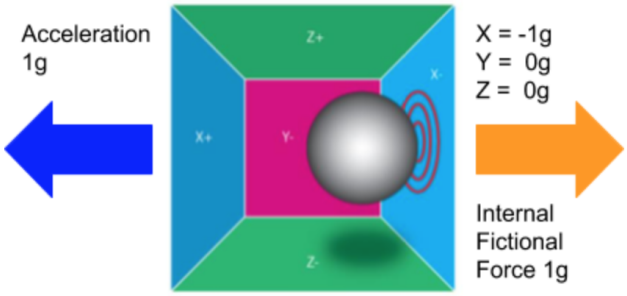



Figure 2.4: Practical illustration of the functioning of an accelerometer shifting left

On Earth, the sensor experiences the gravitational force, so the ball will fall over the $-Z$ wall and applied 1g, as it can be seen in Fig. 2.5.

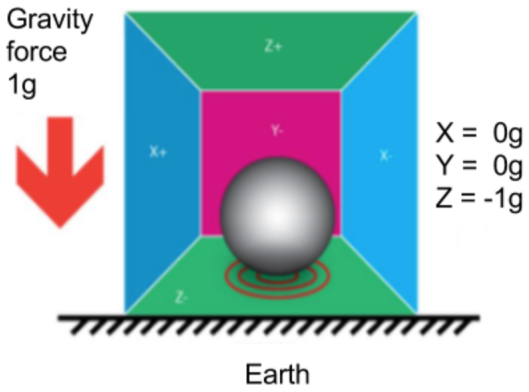



Figure 2.5: Practical illustration of the functioning of an accelerometer on a surface

It is possible to classify the accelerometer in three different types [20]:

- **Capacitive accelerometers**: these accelerometers are based on the detection of a mass displacement with respect to the accelerometer housing. A capacitive displacement conversion is one of the proven and reliable methods. It consists of at least two parts, a stationary plate which is connected to the housing and

other plate which is attached to the inertial mass that is free to move inside the housing. This way, the distance between the two plates is altered, providing a change in their capacitance with which it is possible to measure the force that is acting on the accelerometer.

- **Piezoresistive accelerometers**: in these accelerometers, the mass is attached to a potentiometer which turns an electric current up or down according to the force size that acts on it.

- **Piezoelectric accelerometers**: these accelerometers have a natural application in sensing vibration and acceleration. The mechanical energy is directly converted into electrical energy in a crystalline material. The crystal is attach to a mass, so every time there is movement, the mass generates pressure in the crystal which generates a tiny electrical voltage.

The MEMS accelerometer presents associated errors that arise when using this device which should be considered at the time of measuring, such as:

- Constant Bias: this is the offset of the accelerometers output signal from the true value. A constant bias error that is double integrated causes an error in the position which grows quadratically with time.

- Thermo-Mechanical White Noise: this perturbs the output of the MEMS accelerometer, by white noise with some standard deviation $\sigma$.

- Flicker Noise / Bias Stability: as accelerometers are subject to flicker noise, this causes that their bias wanders over time.

- Temperature effects: this generates some fluctuations in the bias of the output signal. The relation between temperature and and bias depends highly on the specific device, but normally it is not linear. To deal with this, some sensor have temperature sensors which helps them to correct the output signal if necessary.

- Calibration errors: those are errors in scale factors, alignments and output linearities, which appears as bias errors but only visible whilst the device is perceiving acceleration, even due to gravitational acceleration. The position drift it is proportional to the squared rate and duration of acceleration.

### 2.3.4 Gyroscope

Gyroscopes are devices that can measure the different rotation angles of structures. Most commonly, the measurements of angular rates of rotation with a frame of reference already defined. The rates are measured in degrees per second $[^{\circ}/s]$ or revolution per second, RPS. The output provides the necessary information for stabilization and orientation of the measured object.

Similar to the accelerometer, it is possible to define three types of gyroscopes [20]:

- **Rotor gyroscope**: this mechanical gyroscope is comprised of a massive disk free to rotate around a spin axis which it is confined within a framework that is free to rotate about one or two axes. This way, depending on the number of rotating axes, gyros can be either of a single-, or two-degree-of-freedom type.

- **Monolithic Silicon gyroscope**: in this type of gyroscope, the rotating disk mentioned before is replaced with a vibrating element, which allows it to be more robust and to endure some military aerospace environments.

- **Optical gyroscope**: it uses the interference of light to measure angular velocity. To measure rotation two light beams are fired into the coil in opposite directions. If the sensor is undergoing a rotation then the beam travelling in the direction of rotation will experience a longer path to the other end of the fibre than the beam travelling against the rotation [31]

MEMS gyroscopes are characterized by being small, low-cost and have good performance. When a MEMS-based gyroscope is rotated, a small resonant mass located within it, shifts. This movement is converted into small electrical signals that can be amplified and read by a processing unit [12]. As we can see in the Fig. 2.6, it is showed the principle of the inner operation of a MEMS-based gyroscope:

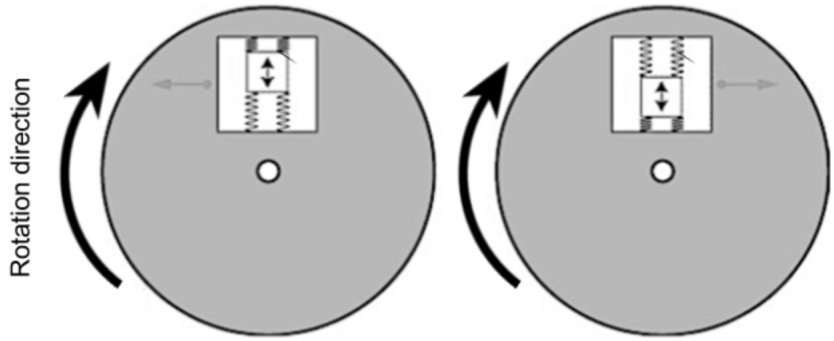



Figure 2.6: Principle of the inner operation of a MEMS-based gyroscope

The MEMS gyroscopes presents, as well as the MEMS accelerometers, errors that arise when using this device [31], such as:

- Constant Bias: is the average output when the gyroscope is not under any rotation in $^\circ/h$. When the output signal is integrated it causes an angular error which grows linearly with time.

- Thermo-Mechanical White Noise: the output signal will be perturbed by a white noise sequence which is a sequence of zero-mean uncorrelated random variables, where each one of these variables are identically distributed and has a finite variance $\sigma^2$.

- Flicker Noise / Bias Stability: the bias of the gyroscope wanders over time due to this error. It is more observable at low frequencies, at high frequencies the flicker noise it is overshadowed by the white noise.

- Temperature effects: fluctuations in the surroundings or in the sensor itself, can lead to movement in the bias. This error grows linearly with time, but the relation between bias and temperature is normally no linear for this sensors.

- Calibration errors: it refers to errors in the scale factors, alignments and linearities of the gyros. These errors lead to the accumulation of the drift in the integrated signal, and they are only visible while the device is turning.

### 2.3.5 Sensors

Depending on the nature of the output signal, the sensors can be classified as analog or digital [26]:

- Analog sensors: the output signal is provided through an analog signal (voltage), which means that it can take infinite values (continuous values), between a minimum and a maximum as it can be seen in Fig. 2.7.

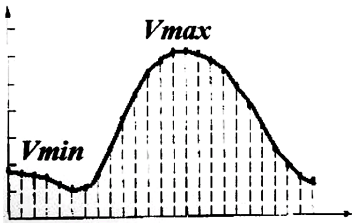



Figure 2.7: Example of an analog signal

- Digital: the output signal is provided through a digital signal that can be a logic "0" or "1", or binary code as it can be seen in Fig. 2.8.

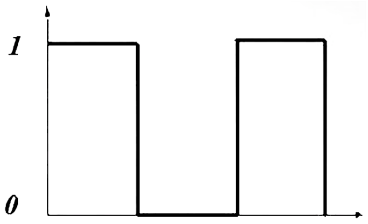



Figure 2.8: Example of a digital signal

For the purposes of this work, the attention will be focus a specific digital sensor, the MPU6050. This sensor contains MEMS accelerometer and a MEMS gyroscope in a single chip, Fig. 2.9. It contains 16-bits analog to digital conversion hardware for each channel, catching the x, y and z axis at the same time. Its features summary can be seen in Table 2.2.

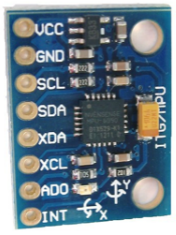



Figure 2.9: MPU6050

Features summary:

| | |
|---|---|
| I2C Digital-output of 6-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format | |
| Input Voltage | 2.3 - 3.4V |
| Selectable Solder Jumpers | CLK, FSYNC and AD0 |
| Tri-Axis angular rate sensor (gyro) a full-scale range | $\pm 50$, $\pm 500$, $\pm 1000$, and $\pm 2000$dps |
| Tri-Axis accelerometer with a full scale range of | $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$ |
| Digital Motion Processing$^{TM}$ (DMP$^{TM}$) engine offloads complex MotionFusion, sensor timing synchronization and gesture detection | |
| Embedded algorithms for run-time bias and compass calibration. No user intervention required | |
| Digital-output temperature sensor | |
| Dimensions | 25.5 x 15.2 x 2.48mm |

Table 2.2: MPU6050 features summary

For further information, see sensor datasheet available on [4].

### 2.3.6 Communication

The communication between the Arduino and the computer it is done via serial port, this is a serial communication interface through which information transfers in or

out one bit at a time. On the Arduino IDE, the serial port is programmed via the function *Serial.begin*(11520), where the number in parenthesis sets the data rate in bits per second (baud) for serial data transmission [2].

It can be establish two types of communication between the sensors depending if the sensor is analog or digital, with the analog sensor the communication is a direct reading of the voltage; with the digital sensor the communication is made by the communication protocol called Inter-integrated Circuit or I2C, which is basically intended to allow multiple slave digital integrated circuits or chips to be able to communicate with one or more master chips [5]; for more information about the I2C protocol see [6].

## 2.4 Signal Filtering

When the term filter is used, it refers to something that removes, blocks or separates out certain elements. For the purpose of this work, the interest lies in two types of filters:

- Electronic filtering, which it must be capable of selectively filter one frequency or a range of frequencies out off a mix of different frequencies in a given circuit.
- Algorithmic filtering, which is a program or a section of code that is specially design to examine each input or output data request for certain qualifying criteria and then process it accordingly.

### 2.4.1 Electronic filters

- High-pass Filter: it is a circuit which offers easy passage of high frequency signals and difficult passage of a low frequency signal. Two basic circuits are suitable to accomplish this, the inductive low-pass filter and the capacitive low-pass filter. While the capacitor's impedance increases with decreasing frequency, the inductive impedance decreases with decreasing frequency.

- Low-pass Filter: it is a circuit, which offers an easy passage of low-frequency signals and difficult passage of high-frequency signals. As the same as the high-pass filter, it has two basic circuits the inductive low-pass filter and the capacitive low-pass filter, which are suitable to achieve this and they are the opposite of their respective high-pass designs.

- Band-pass Filter: this type of circuit filter it is used when a particular band, spread or frequencies are needed to be filtered from a wider range of mixed signals. This filter circuit can be achieved by combining the properties of low-pass and high-pass into a single filter allowing only the passage of those frequencies that are neither too high nor too low.

### 2.4.2 Algorithmic filter

Algorithmic filters have been used in a variety of signal processing tasks as smoothing a signal, removing outliers and noises, waveform shaping, and signal recovery. For the purposes of this work, the attention will be focus in a specific filter, widely known for its properties called the Kalman filter.

The Kalman filter is an algorithm that describes a recursive solution for data filtering problems. This is an estimator that can be implemented in linear and non-linear systems. It receives and process all available measurements and, based on these, it estimates the actual value of the inter-state variables. As an estimator of what is called a linear quadratic problem, which attempts to estimate the instantaneous dynamical linear state system disturbed by noise. The resulting estimate is statically optimal with respect to any quadratic error estimation function.

As the Kalman filter is recursive, it does not have the need to keep all the past data stored, since it uses the last value of the previous state, and so on. This last value, along with the new measures taken, the algorithm delivers new results that will later be considered as previous states, allowing a useful implementation of this algorithm in real time systems.

The more immediate use of the Kalman filter is concentrated in the complex dynamic systems such as continuous manufacturing processes, aircraft, ships or spacecraft. For these applications it is not always possible or desirable to measure each variable that wishes to be controlled, so the filter provides an average for inferring the missing information from indirect or noisy measurements. It is also used to predict possible future course of dynamical systems that people cannot control, such as river flooding, celestial body trajectories or product prices [23].

Using low-cost sensors, its possible to develop a system for locating the positions of a body [30], where it makes possible the study of the feasibility of this applied to surgical operations.

It starts with the calibration of the six-axis sensor, MPU6050, to later apply Kalman filter, in which some necessary points for its implementation are established:

- Knowledge of the system to be treated as well as the measurements obtained through the sensors.
- Present noise, uncertainty and information about the error.
- Initial conditions.

The objective is to estimate a state $x \in \Re^n$ of a process in discrete time. The discrete Kalman filter consists of two stages:

- A prediction stage, where we have the current state of the system and a mapping of the progression of the system as a function of time.
- A correction stage, where the measurements and observations of the variables treated are combined with the actual measured values.

The dynamic system where it is applied the Kalman filter must be expressed as a lineal model where it estimate the state $x \in \Re^n$. This system it is expressed in the following way:

$$x_k = A^{nxn} x_{k-1} + B^{nx1} u_k + w_k, \tag{2.1}$$

where the matrix $A$ it is related to a previous time $k-1$ with the actual state $k$. The matrix $B$ relates the optional control entrance $u \in \Re^n$ to the $x$ state.

With measures $z \in \Re^m$:

$$z_k = H^{mxn} x_k + v_k, \tag{2.2}$$

where $H$ is a matrix that relates the state $x_k$ with the measure $z_k$.

The variables $w_k$ and $v_k$ represents the process and measurement of noise respectively. It is assumed that they are independent one from the other and that they possess a Gaussian distribution:

$$p(w) \sim \mathcal{N}(0, Q), \tag{2.3}$$

$$p(v) \sim \mathcal{N}(0, R) \tag{2.4}$$

The algorithm its divided in two parts, as mentioned:

- Prediction

$$\hat{x}_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.5}$$

$$P_k = AP_{k-1}A^T + Q. \tag{2.6}$$

- Correction

$$K_k = P_{k-1}H^T(HP_{k-1}H^T + R)^{-1} \tag{2.7}$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}) \tag{2.8}$$

$$P_k = (I - K_k H)P_{k-1} \tag{2.9}$$

where:

- Eq. 2.5 is the predicted (a priori) state estimate with:
  - $x_k$: estimate of the state at current time $k$, based on the previous state of the system.
  - $A$: state transition which is applied to the previous state.

  - $B$: control input which is applied to the control vector.
  - $w_{k-1}$: process noise which is drawn from a zero mean multivariate normal distribution with covariance $Q_k$

- Eq. 2.6 is the predicted (a priori) estimate covariance with:
  - $A$: state transition which is applied to the previous covariance.
  - $Q$: covariance of the system noise.

- Eq. 2.7 is the optimal Kalman gain with:
  - $H$: Observational model used to map the a priori state into the observational space.
  - $R$: covariance of the noise from the measurements.

- Eq. 2.8 is the updated (a posteriori) state estimate.

- Eq. 2.9 is the updated (a posteriori) estimate covariance.

Finally, the Fig. 2.10 illustrates the flow development of the algorithm previously mentioned based on the scheme presented in [12].

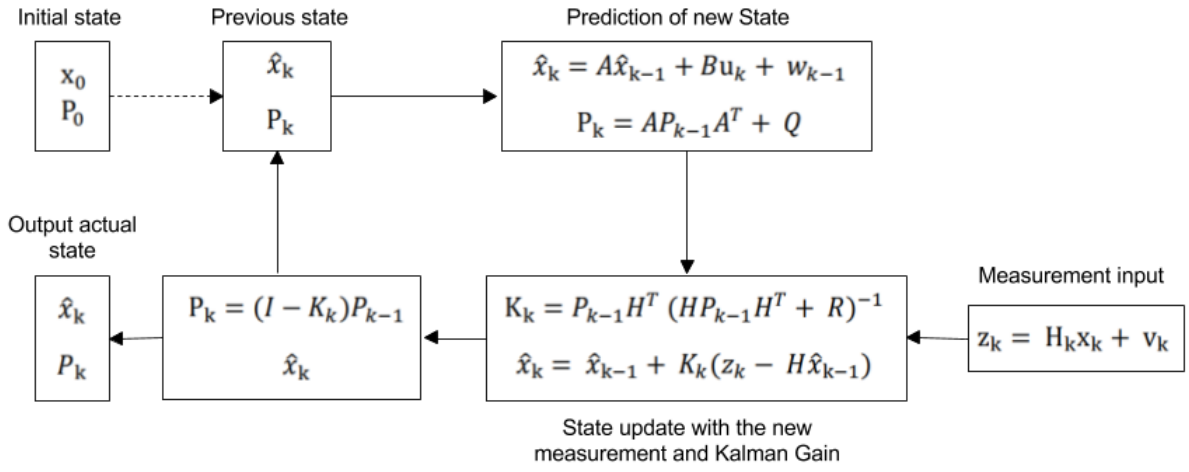



Figure 2.10: Kalman filter diagram

## 2.5 Signal Processing

After the noisy data is filtered, the raw values must be transformed into actual usable information. This part is divided into two different subsections: displacement calculation and orientation calculation.

### 2.5.1 Displacement calculation

To be able to determine the displacement, the raw output delivered by the sensor must be transformed into acceleration and integrated twice into actual displacement in meters. Theoretically, this can be done mathematically through double integration. As it is known, the derivative of distance hand outs the velocity, and the derivative of velocity gives acceleration, as seen in the following equations:

$$a = \frac{\Delta v}{\Delta t} \tag{2.10}$$

$$v = \frac{\Delta d}{\Delta t} \tag{2.11}$$

where $a$ is the acceleration in $[m/s^2]$, $v$ is the velocity in $[m/s]$ and $d$ is the distance in $[m]$ and $\Delta t$ is the time interval between values. By clearing the terms that are needed in this case, it is possible to perform the reverse process to be able to find displacement from acceleration by a double integration process, as it can be seen below:

$$\Delta v = \int a \Delta t \tag{2.12}$$

$$\Delta d = \int v \Delta t \tag{2.13}$$

Nonetheless, as this theory is meant for continuous data and the sensor delivers discrete output values, it is not possible to directly integrate. Thus, numerical integration must be address.

Numerical Integration constitutes a wide range of algorithms for calculating the

numerical value of a definite integral or an approximation, sometimes the term is extended to describe numerical algorithms for solving differential equations as well. These algorithms are based on the idea that each of the iteration of the time series can be treated as an extra section of the area under the trace [8].

When the quadrature nodes in the interval [a, b] are equispaced, as in this project, the Newton-Cotes formulas arise.

The Newton-Cotes formulas or Quadrature rules, based on interpolating functions with a polynomial function [9], are defined as follows:

$$I = \int_a^b f(x)dx \approx \int_a^b P_n(x)dx \tag{2.14}$$

where $P_n(x)$ is a approximation polynomial of $n$ degree for certain values that are appropriately chosen, also known as the interpolation polynomial. The condition is that the polynomial takes the same values as the original function at the chosen points.

There are two types of Newton-Cotes formulas, the closed and the open ones. The closed formulas are when the two ends of the interval are nodes. In the other hand, when the ends are not quadrature nodes, we speak of open formulas. Trapezoidal and Simpson are closed Newton-Cotes formulas, and on the contrary, Poncelet it is an open Newton-Cotes formula.

For the purposes of this work we will only delve into the close Newton-Cotes formulas [24]:

- Trapezoidal rule

  The function $f$ is approximated on each sub-interval with the secant that interpolates $f$ at both ends of the sub-interval. As is possible to see in Figs. 2.11 and 2.12, the approximation of the integral is the area of the trapezoidal figure under the secant, as follows:

$$\int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2}(b-a). \tag{2.15}$$

For a higher accuracy, it is necessary to split $[a,b]$ into sub-intervals with a defined partition. Further on, the approximation in 2.15 is used on each sub-interval. If the partitions is uniform $x_{i=0}^{n}$ with step length $h$, then the approximation it is defined by:

$$I = \int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \approx I_{trap} = \frac{h}{2}\sum_{i=1}^{n}(f(x_{i-1}) + f(x_i)) \tag{2.16}$$

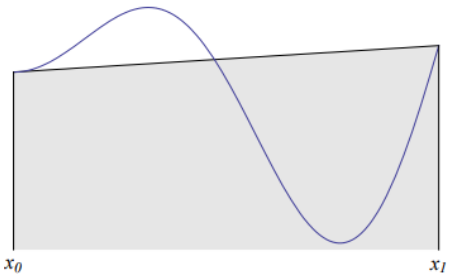



Figure 2.11: The trapezoidal rule with one sub-interval

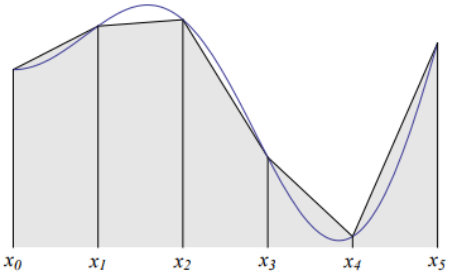



Figure 2.12: The trapezoidal rule with five sub-intervals

The trapezoid method brings as well an error associated, which can be bounded by the following equation:

$$|I–I_{trap}| \leq (b-a)\frac{5h^2}{12}\max_{x\epsilon[a,b]}|f''(x)| \tag{2.17}$$

- Simpson's rule 1/3

This method is based on approximating $f$ by a parabola on each sub-interval, as it can be seen in Figs. 2.13 and 2.14. The function $f$ is interpolated by a

quadratic polynomial $p_2$ at the points $a$, $(a+b)/2$ and $b$, then the integral of $f$ can be approximated by the integral of $p_2$:

$$\int_a^b f(x)dx \approx \int_a^b p_2(x)dx = \frac{b-a}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right). \tag{2.18}$$

This method requires a future sample of the integrand $x$, to get the current sample of the integrated signal $y$, hence, it cannot be used for real time data processing.

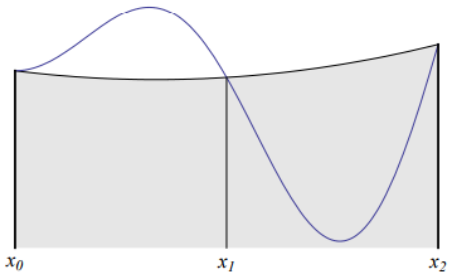



Figure 2.13: Simpson's rule with one sub-interval

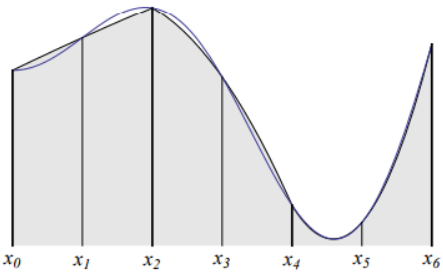



Figure 2.14: Simpson's rule with six sub-intervals

### 2.5.2 Orientation calculation

The attitude of a rigid body is one of the most approached applications of the INS. One of the most popular methods to specify the angular orientation of one coordinate systems respect another, is the use of the three Euler Angles which go trough an orderly succession of turns defining the change from one coordinate system to another [19].

The angles, called phi, theta and psi correspond respectively to the conventional angles of roll($\phi$), pitch ($\theta$) and yaw ($\psi$) that are used in navigation to specify the attitude of a mobile in the axis X, Y and Z, respectively as we can see in Fig. 2.15 [12]:

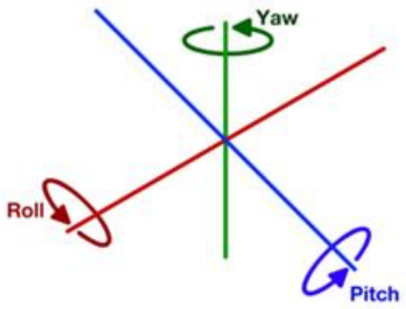



Figure 2.15: Angles Roll, Pitch and Yaw representation

This representation is popular because of the physical significance of the Euler angles which corresponds to the angles which would be measured by angular pick-offs between a set of three gimbals in a stable platform INS [28].

A rotation matrix is known in mathematics as a orthonormal matrix, whose multiplication with a vector generates a rotation of it without changing its length [17].

Using the rotation matrices, it is possible to rewrite a particular vector that has been defined by the bases of a reference system A, based on vectors that make up a reference system B. For a three dimensional reference system, with X, Y, Z axes, we can define a rotation matrix, of 3x3 dimension, for each base vector, which allows us to model the rotation of the said system around that axis of rotation.

The three rotations are expressed mathematically as three separate direction matrices:

$$\text{Roll } \phi, \text{ C1: } \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix}$$

$$\text{Pitch } \theta\text{, C2: } \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$\text{Yaw } \psi\text{, C3: } \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A transformation from reference to body axes may be expressed as the multiplication of the three matrices as follows:

$$C_n^b = C_1 C_2 C_3 \tag{2.19}$$

To obtain the inverse transformation from body to reference axes, is given by:

$$C_b^n = C_n^{bT} = C_3^T C_2^T C_1^T \tag{2.20}$$

For further information see [28].

Euler angles present singularities, as stated in [13]. When the nose unit vector aims straight up or down, the roll and azimuth gimbal axes are collinear, which means that neither of them are uniquely defined at the $\pm 90°$ and only can be specified if the nose is up or down. This problem arise when the body is capable of assuming a vertical orientation. Quaternions avoid these singularities by having a fourth element, three "imaginary" parts and one "real" part, topic that will be left out, since the intended application of this project does not go beyond the singularities presented.

## 2.6 Software

### 2.6.1 Data visualization

For the purposes of this project, it is necessary to be able to show the obtained and processed data trough the sensors in a way that is accurate and clear. Hence, the software **MatLab** was chosen for its wide range of analysis techniques and data

visualization as first approach to test the project feasibility.

The connection and data transfer in real time is possible between the arduino and MatLab, thanks to MatLab is able to access to the serial port communication of the computer. First the COM port is specified where the arduino board is connected to. Second, MatLab provides a function called *SetupSerial*() which accepts as the COM port as a variable and basically, starts listening.

To be able to plot in real time, the common MatLab function *plot* is recommended to use it as a variable, otherwise it makes the execution much slower. Thus, first the plot characteristics must be defined, a variable is set equal to the plot which holds the data to be shown and the time delta of the arriving data, which is obtained through a loop that is hold while the graph is open with the tic/toc function provided by MatLab. Inside this loop the variables with the values are updated every time with the new input data, and with the function *set*(), it is able to graph the wanted values in real time.

# Chapter 3

# Implementation

## 3.1 Introduction

In the following chapter it is presented the implementation of the before proposed INS. For a better understanding, the steps followed in the implementation are shown through a scheme as it can be seen in Fig. 3.1.

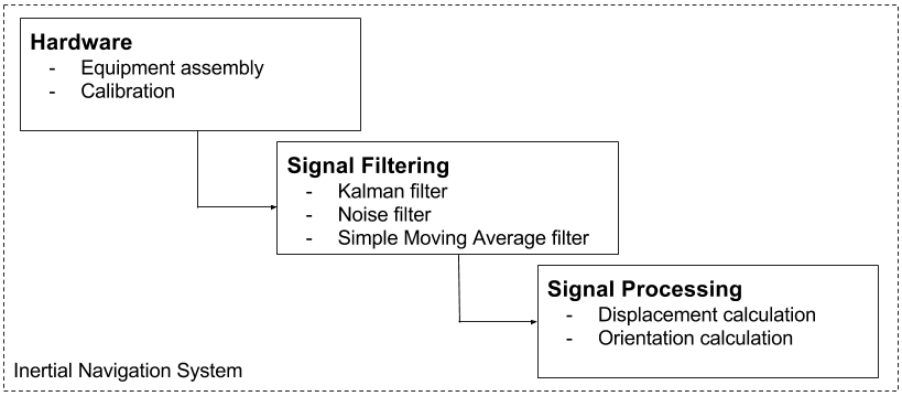



Figure 3.1: Proposed scheme of the INS implementation

This chapter is divided by three main sections: the hardware used to obtain the data, three different filtering methods used to obtain the noise-free data for later comparison and finally, the signal processing of the filtered data for displacement and orientation calculation.

## 3.2 Hardware

For the hardware, as it has been already mentioned, the arduino UNO was chosen as the controller, with the ATmega328p microcontroller, for its low cost and vast documentation. As for the sensors, the MPU6050 was elected as it contains the needed sensors, accelerometer and gyroscope, to retrieve acceleration and angular velocity, information that is necessary for the INS implementation.

### 3.2.1 Equipment assembly

The assembly between the arduino UNO and the sensor MPU6050 is made as it can be seen in Fig. 3.1, following the connections shown in Table 3.1.

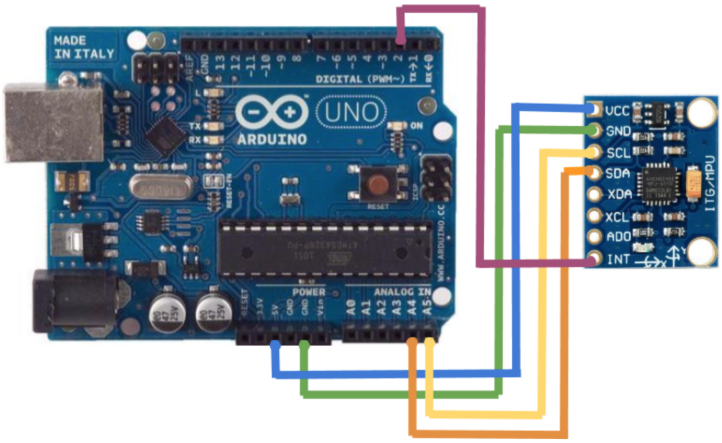



Figure 3.2: Connection between arduino UNO and MPU6050 sensor

| Arduino UNO | - | MPU6050 |
|---|---|---|
| GND | - | GND |
| 5V | - | VCC |
| A5 | - | SCL |
| A4 | - | SDA |
| 2 | - | INT |

Table 3.1: Connection between Arduino UNO and the MPU6050

For the initial tests and to have a reference when comparing the different noise filters, it was necessary to set up a rail on which the sensor can move back and forward along the x-axis manually, as it can be seen in Fig. 3.3. The rail is made out of steel and has a total length of 1.50[$m$]. The rail is placed on the floor with liners on both sides to keep it steady and separated it from the ground, so the platform with the arduino and sensor can be easily moved front and back manually in an available length of 1.20[$m$]. The platform mentioned was 3D printed to set the arduino Uno and the sensor, as it can be seen in Fig. 3.4.


Figure 3.3: Set upped rail fixed to the ground

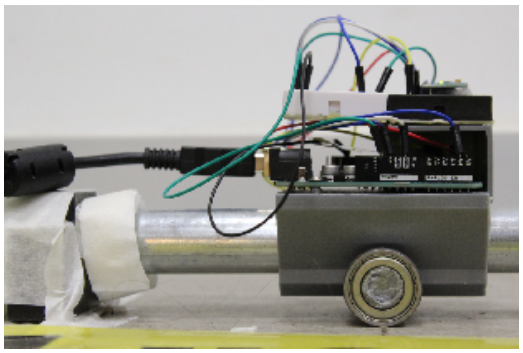


Figure 3.4: 3D printed platform with arduino UNO and MPU6050

As it can be seen in the above figures, the connection between the arduino UNO and the computer was made via USB.

### 3.2.2 Calibration

Each of the axis of the sensors present a maladjustment of the output values called offsets. Defined as the DC offset of the measurement, it represents the mean amplitude displacement from zero. If this displacement is zero, there is no DC offset.

As the implemented sensors are based in MEMS hardware which by their specifications are less precise, even though if they are lying unmoved, they still show an

output different from zero.

For the gyroscope, if there is no movement, the angular velocity should be zero. This way, the raw value different from zero should be the offset of the gyroscope, as shown in Fig. 3.5. To calibrate and to correct this DC component, this non-zero value should be always subtracted from the output raw values.

In the case of the accelerometer, it measures the gravitational field of the earth, making the offsets a high value depending on the position of the sensor. The offsets are calculated for the x-axis and y-axis and then subtracted as mentioned, and for the calibration of the z-axis the offset is calculated having in mind the gravitational constant that should be measuring (instead of zero), as it can be seen in Fig. 3.8.

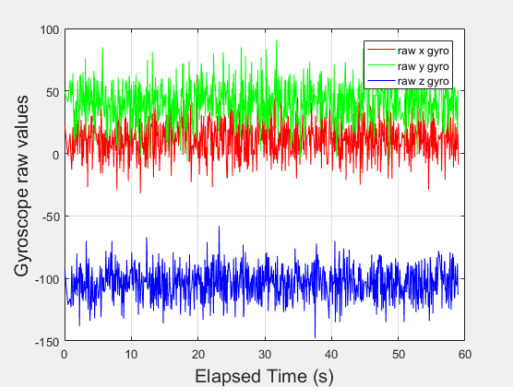



Figure 3.5: Gyroscope raw values with DC component

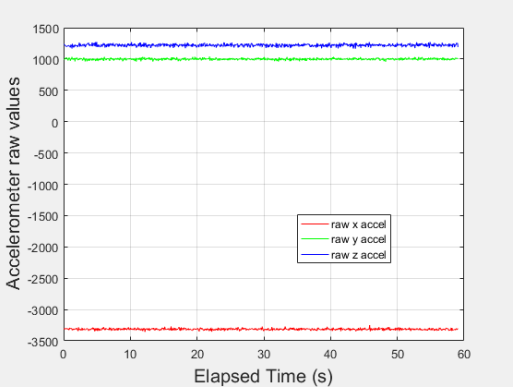



Figure 3.6: Accelerometer raw values with DC component

The calibration of the MPU6050 sensor it is through an arduino program [3] which delivers the offsets to be further used. It is important to mention that the temperature where the calibration is made has an important role in the output values, for this, it is recommended to calibrate in the same place that the sensors will be used. Nonetheless, as it is not possible to go to the place where the sensor is intended to be used, this variable was not taken into account.

At first, the sensor is positioned horizontally with package letters facing up (the z-

axis aligned perpendicularly with the floor) and avoiding any movement or touching. After this is set, the code starts:

1. Read raw accelerometer and gyroscope measurements from sensor.

```
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

2. First 100 measures are discarded and the mean is calculated with 1000 readings.

```
if (i>100 && i<=(buffersize+100)){
  buff_ax=buff_ax+ax;
  ...
  buff_gx=buff_gx+gx;
  ...
}
if (i==(buffersize+100)){
  mean_ax=buff_ax/buffersize;
  ...
  mean_gx=buff_gx/buffersize;
  ...
}
```

3. The offsets for each of the 3-axis accelerometer and 3-axis gyroscope are calculated by dividing the mean by the standard scale factor.

```
ax_offset =- mean_ax/8;
...
az_offset=(16384-mean_az)/8;
gx_offset =-mean_gx/4;
...
```

4. Set the offsets trough the MPU6050 library available for the Arduino Platform.

```
accelgyro.setXAccelOffset(ax_offset);
...
accelgyro.setXGyroOffset(gx_offset);
...
```

5. Goes back to step number (2) and then checks if the calculated offsets are within range of the accelerometer and gyroscope deadzone, which is the error range that is allowed (with a lower value, it gives a higher precision but at the risk that the algorithm may not converge).

```
 if (abs(mean_ax)<=acel_deadzone) ready++;
else ax_offset=ax_offset-mean_ax/acel_deadzone;
...
if (abs(16384-mean_az)<=acel_deadzone) ready++;
else az_offset=az_offset+(16384-mean_az)/acel_deadzone;
if (abs(mean_gx)<=giro_deadzone) ready++;
else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);
...
```

6. Finally, after the calculated offsets fulfill the previous step, the results are shown giving an example of a measure very close to zero on the $x$ and $y$ axis, and in the $z$-axis it shows the presence of the gravitational acceleration.

In Figs. 3.7 and 3.8, it is possible to see the raw signals of the gyroscope and accelerometer respectively, after the offsets are subtracted.

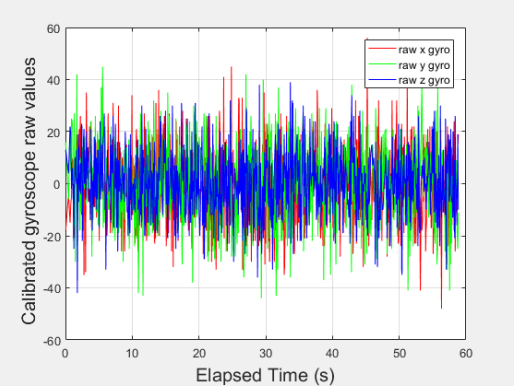



Figure 3.7: Gyroscope raw values with subtracted offsets

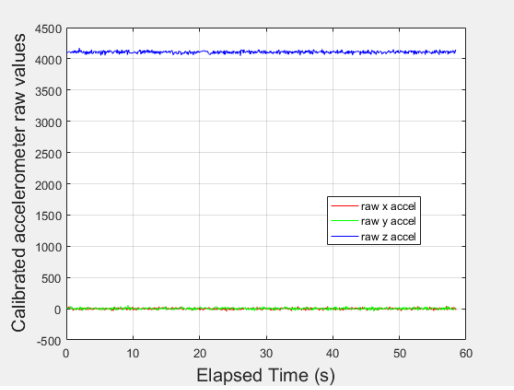



Figure 3.8: Accelerometer raw values with subtracted offsets

## 3.3 Signal Filtering

The accelerometer and the gyroscope deliver rather noisy outputs, which if not corrected they can have a huge effect in the values obtained, specially after double integrating the acceleration to obtain displacement in the following section.

In order to filter out the noisy outputs from the accelerometer, three methods are tested for comparison. Starting with the Kalman filter, which is the main filter which it is desired to be tested, Noise filter proposed by [29] and finally, the Simple Moving Average filter.

One of the requirements of this report is to be able to test the Kalman filter in a real time scenario and if its application in this type of problem is feasible and if its computational demand is justified by the quality of its output. The purpose of the other chosen filters are for their different computational demands, being the Kalman filter the one that demands more calculation, secondly the noise filter and finally, the simple moving average filter, which is the one that requires less processing.

### 3.3.1 Kalman filter

For the purposes of this chapter, Kalman filter it is implemented as a discrete accelerometer noise filter, even though it is more known as a sensor fusion, using the values of the accelerometer and gyroscope to avoid drift, which will be addressed in a next section.

The Discrete Kalman filter algorithm estimates the stated which is wanted in discrete points of time, as is presented in this project. Being an iterative process, it uses different equations and entrance of consecutive values for rapid estimation of the search value when this is hidden under uncertainties and random variations.

The algorithm and the variables used to filter the accelerometer measurements and the definition of the initial states for its implementation are presented:

1. Prediction stage:

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + w_{k-1} \tag{3.1}$$

In Eq. 3.1, $\hat{x}_k$ is calculated from $A$ which is equal to 1, the previous output of the filter (which is initialized as zero), the multiplication between $B$ and the optional control input which for this case there is none, so $B$ is set equal to zero. Also $w_{k-1}$ that represents the perturbation process, is set to zero.

$$P_k = AP_{k-1}A^T + Q \tag{3.2}$$

For Eq. 3.2, $P_k$ is initialized as an identity matrix and the value for $Q$ is set as $13^2$.

2. Correction stage:

$$K_k = P_{k-1}H^T(HP_{k-1}H^T + R)^{-1} \tag{3.3}$$

In Eq. 3.3, the Kalman gain at the $k$ moment is calculated from $P_{k-1}$, $H$ and $R$. As $H$ represents the relation between the model and the state of the system, it is set to 1 and the value of $R$ is set as 13.

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}) \tag{3.4}$$

$$z_k = H_k x_k + v_k \tag{3.5}$$

In Eq. 3.4, $\hat{x}_k$ is calculated from the previous output $\hat{x}_{k-1}$, the Kalman gain $K_k$, $H_k$ and the new measurement from the sensor $z_k$, that is obtained from Eq. 3.5, where $v_k$ is the value associated to the measurement noise, which for this case is equals to zero.

$$P_k = (I - K_k H)P_{k-1} \tag{3.6}$$

Finally, in Eq. 3.6, $P_k$ is updated from $K_k$ and $P_{k-1}$.

The algorithm needs certain initial conditions:

- $\hat{x}_k = x_0$
- $P_{k-1} = P_0$
- $Q$ and $R$ whose values are find by trial and error with an initial suggestion (see next subsection)

### Setting Q and R variables

The algorithm presents two variables that define the behaviour of the filter. This variables are $Q$ and $R$, $Q$ being the covariance of the system noise and $R$ being the covariance of the measurements noise. There are different recommended methods to find the values for this variables, but the most frequent it is by trial an error. If the measurements noise are big, so it should be the value $R$, however, it must be taken into account that if $R$ is too big, the Kalman gain will be much lower and the input measurements will have little credibility at the moment of calculating the new output, giving way to a possible loss of information.

As a first guidance the variance of the data is taken. Measuring the first 30 seconds of raw measurements with the sensor being still, delivering a standard deviation of around 13. Thus, as the squared root of the variance is equal to the deviation, the value $13^2$ is chosen to be used for $Q$ and the same for $R$. The resulting graphs can be seen in Fig. 3.9 while the sensor is still and Fig. 3.10 while the sensor its moved.

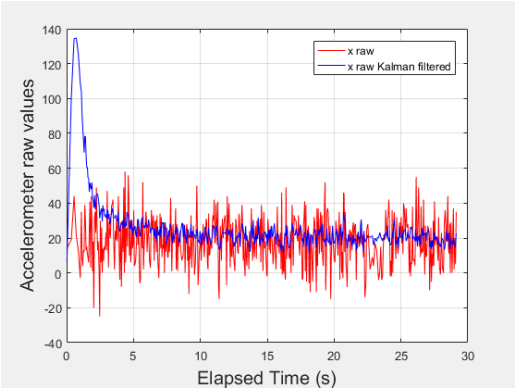



Figure 3.9: Kalman filter applied to still sensor with $Q : 13^2$, $R : 13^2$

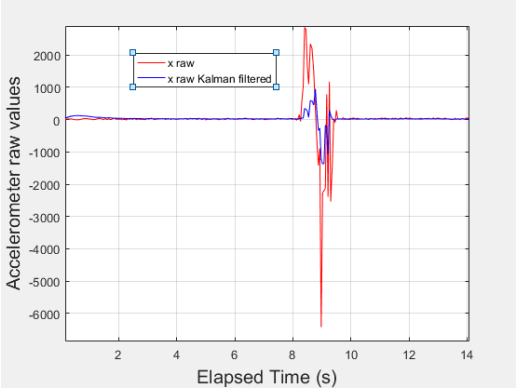



Figure 3.10: Kalman filter applied to moving sensor with $Q : 13^2$, $R : 13^2$

As it can be seen, the noise is filtered. Nonetheless, when the sensor is moved it can be seen that too much information is filtered, making the peaks of acceleration rather low, leaving the data with no information to be used in the next part. A new test is done, using the same value for $Q$, but for $R$, the standard deviation is taken, resulting in Figs. 3.11 and 3.12.

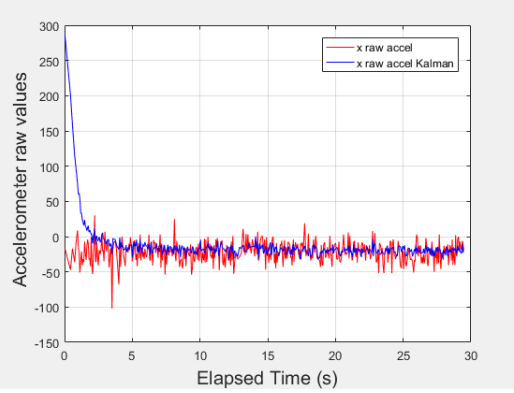



Figure 3.11: Kalman filter applied to still sensor with $Q : 13^2$, $R : 13$

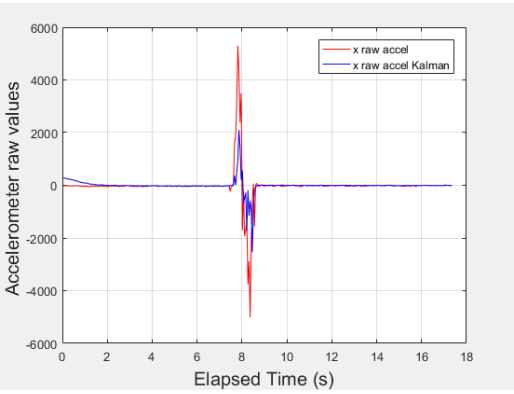



Figure 3.12: Kalman filter applied to moving sensor with $Q : 13^2$, $R : 13$

As it can be appreciated, the Kalman filter (line in blue) indeed filters the input data, diminishing the noise presented in the measurements and giving more similar to reality values when the sensor is moving, resulting in a more satisfying output.

### 3.3.2 Noise Filter

As a first approach it is implemented the noise filter proposed in [29]. It is mentioned that the slow pass filter is not used to filter every measurement because its slowness since in some cases it can be encountered a significant change in a value with which the system must respond rapidly and the filter does not reach the need. The pseudo-code of the algorithm is the following with its respective explanation:

```
if rawValue < prevFilterOut +/- noiseParameter
    then filterOutcome = rawValue
    // if the if-statement is fulfill it shows that the
```

```
    //difference is significant, meaning a real change
    //in the acceleration, thus the value it is not altered.
else
    if rawValue < prevFilterOut +/- noiseParameterSen
        then filteredValue = (rawValue + prevFilterOut)/2
        // if the difference between the rawValue and the
        //prevFilterOut is to big to ignore but too
        //small to set it as the filtered value, an average
        //of both is set as the outcome of the filter.
    else
        filteredValue = (1-noiseParameterLow)*prevFilterOut
                        + noiseParameterLow*rawValue
        // if the difference is less than the
        //noiseParameterSen, the rawValue is low-pass
        //filtered using the parameter noiseParameterLow.
```

The aforementioned parameters are defined as follows:

- *noiseParameter*: to be able to choose this parameter, the raw output of the accelerometer was observed while the sensor was still. The values oscillate between $\pm 40$, nonetheless after a few tests it was decided to use a higher value to filter the noise, choosing a noiseParameter equals to $\pm 200$ which represents an acceleration of $\pm 0.0488[m/s^2]$, which compared with the actual exerted acceleration, is not significant.

- *noiseParameterSen*: following the same line as with the previous parameter, the value should be higher than $\pm 40$ but lower than $\pm 200$. After a few tests, the value $\pm 150$ was chosen, meaning an acceleration of $\pm 0.0366[m/s^2]$.

- *noiseParamaterLow (NPL)*: this parameter, as told, is used to slow-pass filter the values that do not fulfill the if-statement. The following equation is presented associated to the cut-off frequency that wants to be applied, in this case $0.026Hz$:

$$F_c = \frac{NPL}{(1 - NPL)2\pi\Delta t} \tag{3.7}$$

To achieved a cut-off frequency of $0.26Hz$, the $NPL$ must take the value 0.01. Even though this cut-off frequency is low, it is only applied when a strong low-pass filter is needed [29]. The results can be seen in Figs. 3.13 and 3.14.

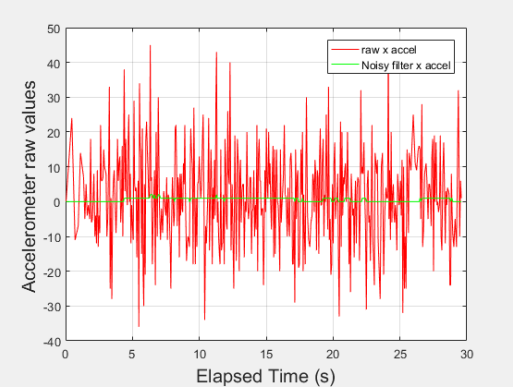



Figure 3.13: Noise filter applied to the x-axis raw values given by the accelerometer while being still in the test rail

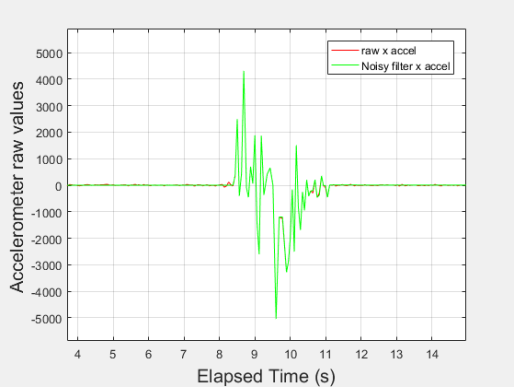



Figure 3.14: Noise filter applied to the x-axis raw values given by the accelerometer moved one way in the test rail

In Fig. 3.14, the raw value cannot be seen because this filter only filters when the sensor is still. Thus, both values are overlapped, showing just the filtered one.

### 3.3.3 Simple Moving Average filter

The last filter used is the Simple Moving Average filter (SMA), as it states in its name, it takes $n$ values and calculates the average of them. This value becomes the filtered measure, as it can be seen in the equation below:

$$x_f = \frac{x_i + x_{i-1} + ... + x_{i-(n-i)}}{n} \tag{3.8}$$

where:

- $x_f$: is the average of the $n$ measures taken,
- $x_i$: is the $i$ input measurement,

- $n$: number of values taken.

To be able to choose how many measurements will be taken to calculate the average, the filter was tested with different values with the sensor being still and going in one-way on the rail, as it can be seen in Figs. 3.15 to 3.18.

- $n = 3$

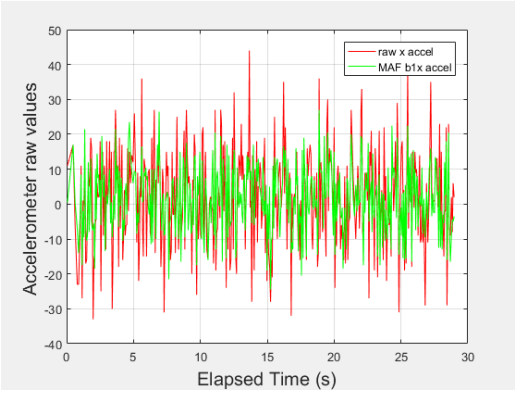



Figure 3.15: SMA filter with n=3 with the sensor still

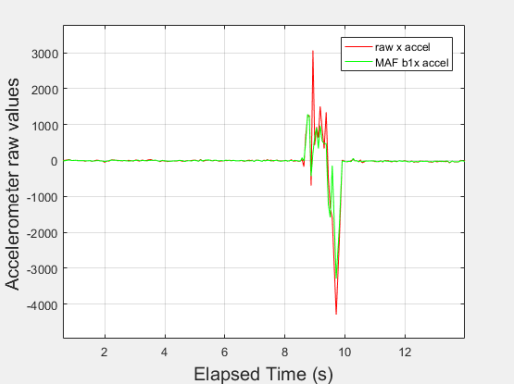



Figure 3.16: SMA filter with n=3 one-way on the rail

- $n = 10$

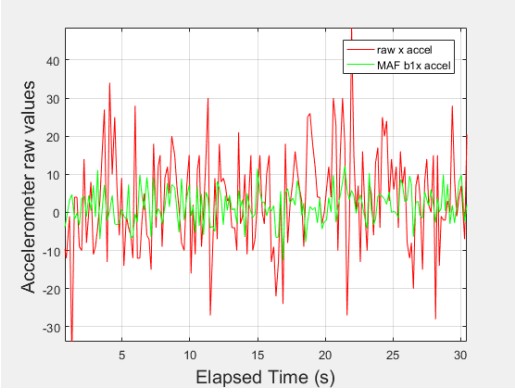



Figure 3.17: SMA filter with n=10 with the sensor still

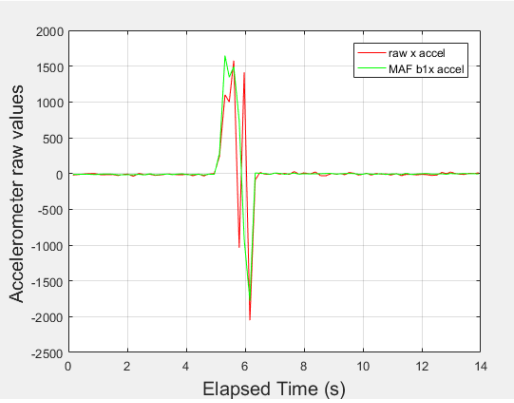



Figure 3.18: SMA filter with n=10 one-way on the rail

As it can be appreciated, the more values taken, the more is the noise filtered. The $SMA$ was also tested with higher number of measurements giving better results, nonetheless when the value used was higher, the filter became much slower at the moment of showing the measurements. Also, the peaks of the raw accelerometer data were diminished, meaning a high loss of information for further processing.

Finally, the chosen value for $n$ was 3, with which the slowness and the filtering were balanced.

## 3.4 Signal Processing

For this section, the already filtered data is processed. For the first part, it is explained how the displacement is obtained through the trapezoidal rule for each of the filters mentioned above, and for the second part it is explained how the orientation is calculated to be able to subtract the gravitational constant depending on the attitude of the sensor through Euler angles.

In order to be able to calculate orientation the angle must be known, this is possible thanks to the relative orientation that the gyroscopes delivers, nonetheless gyroscope in general tend to drift over time. Accelerometers and magnetometers are being used to help determine orientation according to gravity and the earth magnetic field. Kalman filter is used in this case to compensate the drift of the gyroscope, by using the *roll* and *pitch* angles calculated from the accelerometer. The accelerometer cannot calculate the *yaw*, because its z-axis is defined in the direction of the gravity. Thus, as stated in [29], the *yaw* can be theoretically calculated with the values given by the magnetometer. As it measures the earth magnetic field that is perpendicular to gravity, the rotation in the horizontal plane can be determined. Nonetheless, the magnetometer will not be used in this project, because as it was stated, it does not output accurate measurements, as it is constantly influenced by the electronic devices that can be found all around, and also, as the earth gravitational field is weak is hard to measure the changes that actually affects it.

### 3.4.1 Displacement calculation

To calculate the displacement it is necessary first to obtain the actual acceleration given by the accelerometer. The accelerometer, depending on the set up of the full scale range value, it is necessary to divide the raw value by the sensitivity Scale Factor that is given in $[LSB/g]$ according to Table 3.2.

| Full-Scale Range Value [g] | - | Sensitivity Scale Factor [LSB/g] |
|---|---|---|
| ±2 | - | 16384 |
| ±4 | - | 8192 |
| ±8 | - | 4096 |
| ±16 | - | 2048 |

Table 3.2: Full-Scale Range value associated to the sensitivity scale factor for the accelerometer

The set up chosen was $\pm 8[g]$ that delivers values between $\pm 4096[LSB/g]$, so the accelerometer is not that sensitive to change, being the $\pm 2[g]$ the more sensitive, as the idea behind these is for people walking or moving in a vehicle. Thus, to be able to obtain acceleration from the raw values it is necessary to divide every value by 4096 with which it will deliver the acceleration in $[g]$, and after the data is integrated, it will be multiplied for the acceleration constant to obtain the respective values.

The numerical integration method called the trapezoidal rule is proposed to find an approximation of the intended integrals. The trapezoidal rule is, as mentioned in the before chapter, a discrete method which uses the previous and actual measurement, and the time difference between them to calculate the integrand [29]. The following equation is applied to the acceleration to obtain velocity in $[m/s]$, and after, the same equation is applied to the velocity to obtain displacement in $[m]$ :

$$y_n = y_{n-1} + \frac{1}{2}\Delta t(x_{n-1} + x_n) \tag{3.9}$$

where:

- $y_n$ is the integrated value,

- $y_{n-1}$ is the previous integrated value,
- $\Delta t$ time differential between measurements,
- $x_{n-1}$ is the previous input value,
- $x_n$ is the current input value.

This method can be used in real time data processing, unlike the Simpson's rule mentioned in the second chapter, reason why it is not used to obtain displacement. Nonetheless, using the trapezoidal rule works for noise-less data, which the sensor does not deliver, even though the raw signal is filtered, the DC components before mentioned are still a problem when it comes to integrate the acceleration because when a constant value is integrated it gives a slope and the second integration will result in an exponential function [29], quickly making the values obtained unusable.

As stated in [29], a high pass filter must be used to be able to prevent the before mentioned from happening, doing so, it should deliver better results if the high pass filter is implemented after each integration. Different high-pass filters are available to be used, such as the Infinite Impulse Response filter (IIR), Fast Fourier Transform filter (FFT) and the Finite Impulse Response (FIR), but only this last one can be applied in real time. However, it is not viable to implement it in this project given the demands of the filter which makes it slow, being the sampling rate not high enough. It is, though, recommended to be used with a bigger micro-controller than the one that is being used in this project.

An attempt to surpass the established problem before mentioned, a slow pass filter is proposed by [29] to be used between each integration step, where the drift is calculated and subtracted from the original value, using the following equation:

$$u_n = (1 - \alpha)u_{n-1} + \alpha x_{n-1} \tag{3.10}$$

$$y_n = x_n - u_n \tag{3.11}$$

where:

- $u_n$ is the value that represents the drift intend to be subtracted,
- $\alpha$ is the filter constant,
- $u_{n-1}$ is the previous value for the drift,
- $x_{n-1}$ is the previous input value,
- $y_n$ is is the output from the low pass filter,
- $x_n$ is the current input value.

Even though this filter was applied between each integration and to the displacement obtained, the results were unsatisfactory for the purpose that is required. The main problem is the present constant bias error, which the slow-pass filter was not able to eliminate. This affected the results and as it can be seen in Figs. 3.19 to 3.24, even though the sensor was not being moved, the displacement kept in augmenting with any of the filters previously applied.

- Kalman filter

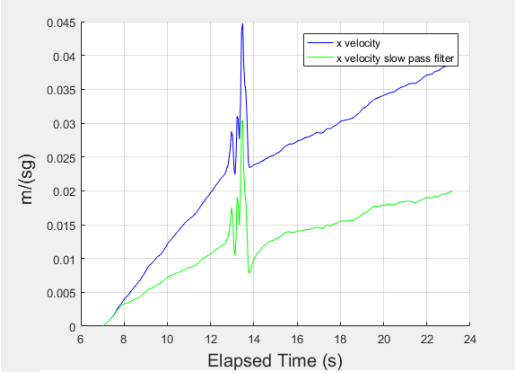



Figure 3.19: X-axis velocity obtained through Kalman filtered accel.

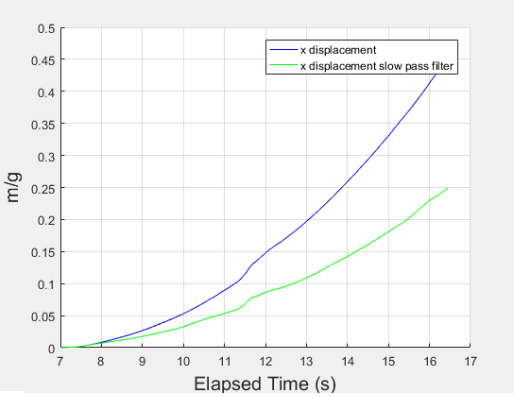



Figure 3.20: X-axis displacement obtained through Kalman filtered accel.

- Noise filter

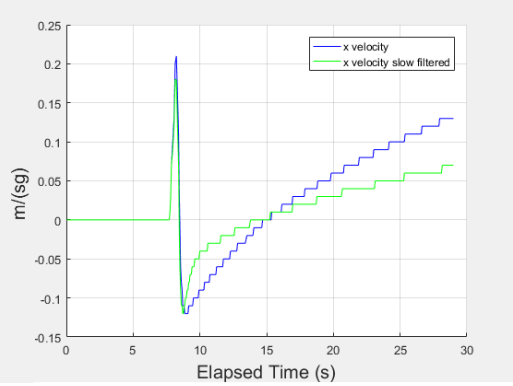



Figure 3.21: X-axis velocity obtained through Noise filtered accel.

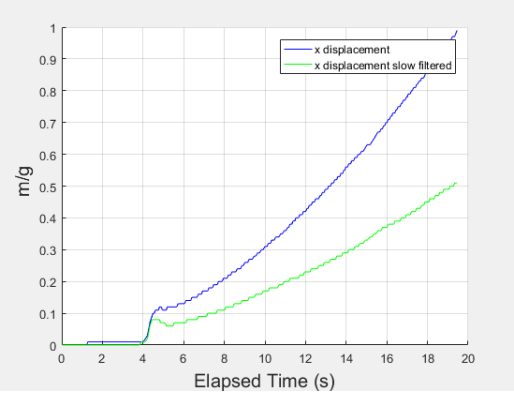



Figure 3.22: X-axis displacement obtained through Noise filtered accel.

- Simple Moving Average filter

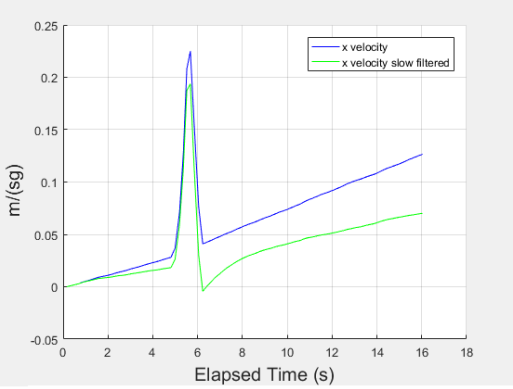



Figure 3.23: X-axis velocity obtained through SMA filtered accel.

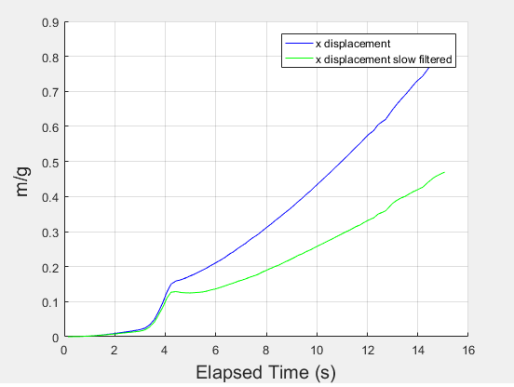



Figure 3.24: X-axis displacement obtained through SMA filtered accel.

For the Kalman Filter, it was establish a period of seven seconds before starting to calculate the velocity and displacement, so the filter had the opportunity to stabilize before starting to use the values for integration.

In Table 3.3, the values obtained from the previous graphs were multiply by 9.8 to represent them in actual meters. As it can be seen, the better behaved filter was

SMA, giving the smaller error regarding the displacement that was actually made. As for the SMA, different parameters for Kalman and Noise filter were tested, where with the strongest filtering the results of displacement were under-representing the actual displacement and when the filter was softer, the values grew exponentially, making the results unusable.

| Filter | Displacement [m] | Error [m] |
|---|---|---|
| Kalman | 0.784 | 0.416 |
| Noise | 0.833 | 0.367 |
| SMA | 1.225 | 0.025 |

Table 3.3: Displacement and errors summary of used filters

Although the values of the SMA filter are 2% off, it was because as soon as the sensor started to send measurements, it was moved through the rail. If it was not done this way, the error would have been accumulating, as the same way as it can be seen after the movement is made, when the sensor becomes still.

Finally, although SMA was able to show a close displacement for the first seconds of testing, the results are unsatisfactory because after a few seconds, the displacement grows and moves away from what is really happening, as it was not possible to get rid of the constant bias generated by the double integration process.

### 3.4.2 Orientation calculation

To be able to calculate orientation and to subtract the gravity from the given acceleration, first are obtained the roll and pitch angles using the accelerometer raw values as it also measures the gravity of the earth. By calculating the ratio of the different amount of gravity measured by the individual axis of the accelerometer, the orientation can be measured using gravity, following Eqs. 3.12 and 3.13, where the angles are calculated via the arctangent, represented by the function *atan*2 in arduino. This function outputs the value from $-\pi$ to $\pi$ in radians, that it must be transformed into degrees for later usage.

$$roll = arctan\left(\frac{accY}{\sqrt{(accX)^2 + (accZ)^2}}\right) \tag{3.12}$$

$$pitch = arctan\left(\frac{-accX}{\sqrt{(accY)^2 + (accZ)^2}}\right) \tag{3.13}$$

As stated in the previous chapter, the angular velocity is given in 16 bits with a full rate scale of ±250 degrees, this allows to calculate the LSB(Least Significant Bit):

$$\frac{2^{16}}{500} = 131[LSB] \tag{3.14}$$

This value is used to divide the raw value of the gyroscope, to obtain the actual angular velocity:

$$angularVelocity = \frac{rawGyro}{131} \tag{3.15}$$

Then, the Kalman function can be called, where three parameters are needed: the pitch/roll angles, the angular velocity and the time delta. The values choose for $Q$ and $R$ are 0.001 and 0.03 respectively, $P$ is initialized as zero like the bias variable (which is the multiplication of the Kalman gain and the difference between the input angle and the estimated one) and the angle to be outputted. The algorithm is described below [7]:

1. Prediction
    - The state $\hat{x}$ is projected ahead, where the bias is subtracted from the angular velocity and the angle is predicted.
    - The error covariance $P_k$ is projected ahead.
2. Update
    - The Kalman gain $K_k$ is computed.
    - The angle estimate is updated with the measurement $z_k$. The previous angle is subtracted from the new one, for later angle and bias update.

- The error covariance $P_k$ is updated.

In Figs. 3.25 and 3.26, it is possible to see the results of the angles roll and pitch respectively, with the Kalman filter being applied.

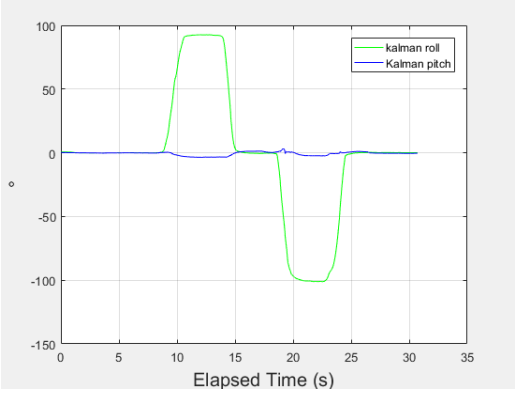



Figure 3.25: Roll Movement with the gyroscope

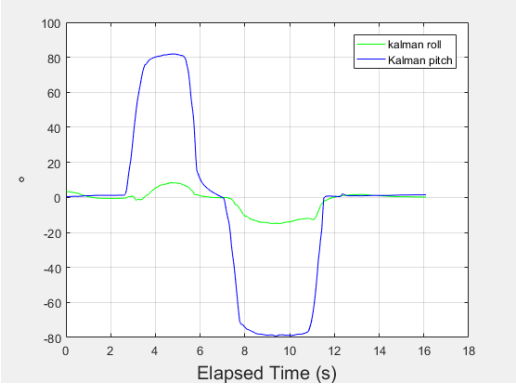



Figure 3.26: Pitch Movement with the gyroscope

The tests were done manually trying to obtain a 90° angle. As it can be seen in Fig. 3.26, the sensor detected a small roll movement, this is because at the moment of being tested, the platform was not symmetrical for both sides, explaining this small roll angle when doing a pitch movement. Nonetheless, the results turn out to be satisfactory as it reflects the degrees that were expected.

After several tests were done, the Kalman filter successfully integrates both values from the accelerometer and gyroscope, giving a free-drift orientation over time. And finally it is proceed to calculate the gravitational-free acceleration.

### Free-gravity acceleration calculation

The purpose behind calculating the free-gravity acceleration is that the orientation of the accelerometer, in real conditions, can move in all angles generating an inclination error. Due to this, at the moment of integrating it cannot be known if its the actual acceleration or the gravitational. Thus, the need lies in being able to reduce

the error generated by the integration, by just integrating the real acceleration that is being exerted.

To be able to obtain the real acceleration from the accelerometer, it is necessary to decompose the gravity vector affecting the sensor, as show in Fig. 3.27.

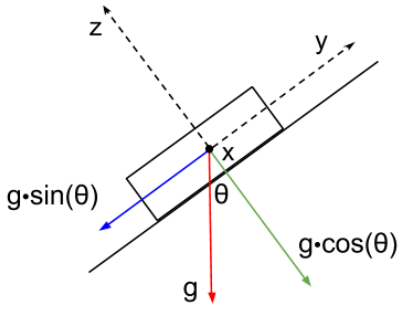



Figure 3.27: Simulation of vector decomposition of the gravity affecting the sensor

Theoretically, as it can be seen in the Fig. 3.27, to be able to obtain the real acceleration of the x-axis, it is needed to multiply the gravitational constant with the sine of the pitch angle to then subtract it from the x-axis acceleration, and for the y-axis, the gravitational constant must be multiply with the cosine of the roll angle, to then subtract it from the y-axis acceleration. As is presented in the following equations:

$$realXaccel = Xaccel - (g * sin(KalmanPitch)) \tag{3.16}$$

$$realYaccel = Yaccel - (g * cos(KalmanRoll)) \tag{3.17}$$

After the previous equations are applied, it is possible to observe the obtained results, for the x-axis, in Fig. 3.28 and, for the y-axis, in Fig. 3.29.

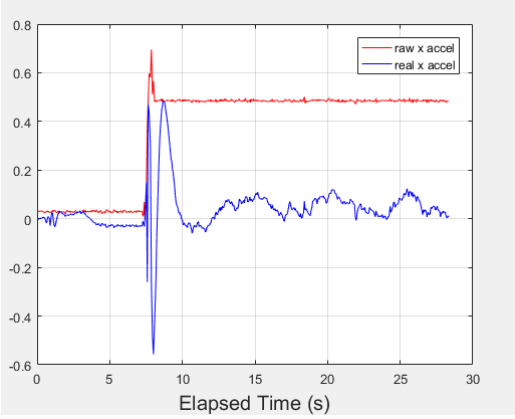



Figure 3.28: Comparison with raw acceleration and free-gravity acceleration for the x-axis

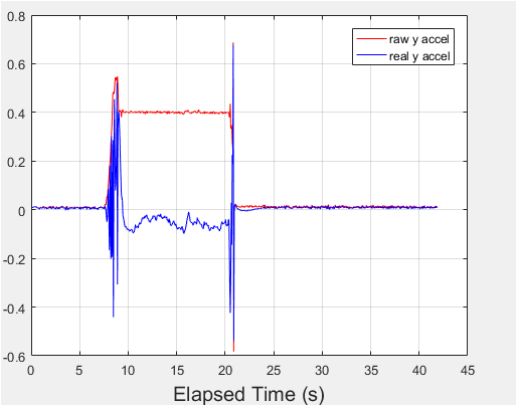



Figure 3.29: Comparison with raw acceleration and free-gravity acceleration for the y-axis

At the moment of calculating the free-gravity acceleration, it is able to see in Figs. 3.28 and 3.29, that first the signal takes some time to stabilize, giving important errors if its intended to double integrate it to calculate position through the trapezoidal rule. After a few seconds the signal seems to stabilizes and gives an output close to zero but very noise. Also after several tests, it was observed that it takes a small movement to destabilize it again, giving an output that does not go according to reality.

## 3.5 Results discussion

At the moment of filtering the signal, it can be seen that the Kalman filter deals successfully with the noise, filtering according to the initial values set to $Q$ and $R$. Nonetheless, it does not correct the constant bias that is produced by the double integration process, as it does for the angle calculation. This is because, at the moment of obtaining the orientation, the filter uses two variables that refer to the same information, but from different sources. One, by calculating the ratio of the different amount of gravity measured by the individual axis of the accelerometer, and the other directly from the gyroscope's angular velocity, which is the value that drifts, and, it compares them to obtain one filtered value that is reliable and subjected to

the measurement and system noise. The difference between these two values, called the bias, is being calculated and updated in every loop, to be then subtracted from the noisy measurement. This way the values are complemented and the drift can be corrected as it goes. This can be possible thanks to the reliance on the accelerometer in the long term, and the short term reliance in the gyroscope, thus its measurements are combined to obtain a drift-free orientation of the sensor.

As for the comparison between the filters, it was shown that a simpler filter could have sufficed, like the SMA. Nonetheless, the same problem is faced with the three filters that were applied. At the moment of using the trapezoidal rule to obtain the velocity, and not being able to have another reference rather than the acceleration itself, it gives a slope and after the second integration, it results in an exponential function, delivering results that quickly diverge from reality, even with a slow-pass filter applied after each integration to try to correct it. This was one of the major problems that were faced. As not being able to use an external reference to correct the drift presented in displacement after the double integration process, the values obtained were unusable. Nonetheless, as it was mentioned, a high-pass filter is used in a previous work, but the results were not satisfactory because of the delay that the filter itself gave, so it was recommended a bigger micro-controller for better possible results. Another possible solution could have been instead of doing the calculations in real time, the data could have been gathered until the displacement of an specific person is required, where it could be possible to apply other filters that cannot be used in real time, showing the displacement estimate without the delay.

Finally, the orientation is well calculated using the Kalman filter, delivering the expected values. Nonetheless, at the moment of testing the theory behind of subtracting the gravity from the acceleration, as the gravitational constant is multiplied by a $cos(\theta)$ or $sin(\theta)$, it makes the value oscillate a few times before actually showing the real acceleration, and tends to destabilize quite quickly. Considering this, the integration of the real acceleration to obtain velocity and then position, was not made because the position would grow exponentially, moving far away from the actual displacement.

# Chapter 4

# Conclusions

In this final chapter, the respective conclusions are addressed which were found throughout the different implementation sections of this project.

## Hardware

As the MPU-6050 presents a maladjustment called offsets, which comes with it from fabrication, the calibration process is indeed a fundamental step to be able to obtain coherent results. Also, the MEMS by themselves are hardware which by their fabrication specifications brings associated errors, which makes the measurements less precise. Nonetheless, for the context of this project, as the idea behind it was to track a person in an open space, a high accuracy was not necessary. Thus, the arduino Uno together with the MPU-6050 used for this project were able to deliver usable results.

## Signal filtering

As it was expected, the signal that is handed out by the sensor is rather noisy. This is explained by the low-cost IMU used in this project and the micro-fabrication with which they are made. Nonetheless, through the filters applied, it was possible to see that this noise can be indeed filtered out.

A filter can be chosen depending on availability of resources. For this project, one of the specific goals was to use the Kalman filter as a noise filter, which was successfully done. The computational demand from Kalman was not as grand when it is used as a sensor fusion algorithm, meaning not a great difference in the timing of the results obtained in comparison with the other used filters, and how it was seen, neither in the double integration process.

Finally, each filter used handed out similar outputs. Nonetheless, the real comparison takes place at the moment of obtaining the displacement.

# Signal processing

## Displacement calculation

For this part, the acceleration was double integrated to be able to obtain an estimation of the displacement made by the sensor while being moved across the rail in the x-axis. The displacement estimation obtained, being not as satisfactory as expected, it can be said that the main goal of this project was not achieved. This was mainly because of the present constant bias in the displacement results, which even though for the first seconds of measurements it was able to show it with an error of 2% using the SMA filter, the constant bias made unusable any values after a few seconds. Although, a slow-pass filter was also used and outputted better results with a small delay, it was not able to eliminate the constant bias which make unusable results.

The accumulative error of integration that defined the results of this part, could be diminished by the application of a high-pass filter between each integration, more specifically the $FIR$ filter, which can be used in real time but it was not implemented for the limitations of the microcontroller, which could be a feasible future solution.

## Orientation calculation

As for the application of the Kalman filter as a sensor fusion algorithm, this specific objective was put to the test with success. It was possible to obtain the orientation of the sensor with high accuracy and no drift whatsoever, thanks to the values given by the accelerometer and gyroscope that complemented each other.

At the moment of calculating the orientation, several libraries were available. This means that the orientation calculation problem has been already solved with special focus on body orientation, for example: stabilization of micro-surgical instruments or limb orientation, and the Kalman filter most of the times is present or compared to, delivering good orientation results as they were obtained in this project.

As well as for the Euler angles, a lot of the orientation research focuses on quaternions, which even though they are not simple-to-understand mathematics, there are libraries and ready-to-use applications, meaning that is a viable and tested option for future development, if the singularities of using the Euler angles becomes a problem to its development.

### Free-gravity acceleration calculation

As the MPU-6050 provides the acceleration in the x, y and z direction of the sensor frame, it is equally influenced by the gravitational constant depending on its inclination, giving room to the inclination error. This error can be misleading at the moment of integrating the acceleration because instead of integrating the real acceleration, it would be integrating the gravitational one, giving displacement that has not been actually made.

Thanks to the orientation obtained, theoretically it is possible to calculate the component of the gravitational constant that affects the specific axis, as it was explained in the previous chapter. Then, by subtracting this value from the measurements deliver by the accelerometer, it should have been possible to obtain free-gravity acceleration. Nonetheless, through the different already exposed tests, the obtained results were unsatisfactory and not usable. The gravitational acceleration, as it was

being multiplied by a trigonometric function, specifically sine or cosine, the value tended to oscillate taking some time to settle and after it was stable, it took a small movement to destabilize it again, making impossible to use these results to obtain position through double integration.

## Overall conclusions and future work

A combination of a person walking and that can also change position by vehicle, turned out to be a hard problem to approach, as they have different dynamic movements. There are possible solutions for both problems independently, which made them not usable for this project. However, a more complex system could be determined, where the dynamics are previously known and the solutions used accordingly.

It was intended for the implementation of this project, to keep it at a low-cost and simple, which made it not feasible. However, there are other possibilities and different approaches that could be implemented in future works, such as the use of more accelerometers or not showing the results in real time, but only at the moment when it is required to avoid the possible delay, or even to be able to use other filters than cannot be used in real time.

Even though GPS cannot be used as an external reference, other types of external references exist that could be convenient for this specific type of project, such as the implementation of cameras or different checkpoints to be able to position the person.

Nowadays the easy and affordable access to the type of technology used in this project, makes more reachable and approachable the development of different projects in the robotic area, where its investigation does not cease to increase.

Finally, through the investigation of this project to be able to implement an INS, it was necessary a further knowledge in different disciplines, such as: robotics, electrics and mechanics, making this project multidisciplinary.

# Bibliography


[1] Atacama Large Millimeter/Submillimeter Array (ALMA) site. https://almascience.eso.org/about-alma/alma-site. Accessed: 2017-12-05.

[2] ARDUINO begin. https://www.arduino.cc/en/Serial/Begin. Accessed: 2017-10-30.

[3] MPU6050 calibration. http://wired.chillibasket.com/2015/01/calibrating-mpu6050/. Accessed: 2017-10-15.

[4] MPU6050 Datasheet. https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. Accessed: 2017-10-30.

[5] I2C tutorials. https://learn.sparkfun.com/tutorials/i2c, . Accessed: 2017-10-27.

[6] I2C Bus. http://www.i2c-bus.org/, . Accessed: 2017-10-27.

[7] Kalman filter. https://github.com/TKJElectronics/KalmanFilter. Accessed: 2017-09-06.

[8] Numerical Integration Methods. https://research.ncl.ac.uk/game/mastersdegree/gametechnologies/physics2numericalintegrationmethods/2016%20Tutorial%202%20-%20Numerical%20Integration%20Methods.pdf, . Accessed: 2017-10-30.

[9] Numerical Integration evy kersalé. http://www1.maths.leeds.ac.uk/~kersale/2600/Notes/chapter_4.pdf, . Accessed: 2017-10-30.

[10] Arduino "Arduino UNO & Genuino UNO". https://www.arduino.cc/en/Main/ArduinoBoardUno.

[11] Khairi Abdulrahim, Terry Moore, Christopher Hide, and Chris Hill. Understanding the performance of zero velocity updates in mems-based pedestrian navigation. *International Journal of Advancements in Technology*, 5(2):53–60, 2014.

[12] J. Rodriguez B. Barone. Desarrollo de un sistema de navegación inercial reprogramable para múltiples plataformas móviles. *Proyecto de Grado para optar al título de Ingeniero Electrónico, Universidad Simón Bolívar, Sartenejas, Venezuela*, 2016.

[13] Eric R Bachmann. Inertial and magnetic tracking of limb segment orientation for inserting humans into synthetic environments. Technical report, Naval Postgraduate School Monterey CA, 2000.

[14] Danny Banks. *Microengineering, MEMS, and interfacing: a practical guide*. CRC press, 2006.

[15] Scott M Bezick, Alan J Pue, and Charles M Patzelt. Inertial navigation for guided missile systems. *Johns Hopkins APL technical digest*, 28(4):331–342, 2010.

[16] Duncan B Cox. Integration of gps with inertial navigation systems. *Navigation*, 25(2):236–245, 1978.

[17] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[18] CS Draper, W Wrigley, DG Hoag, RH Battin, JE Miller, DA Koso, AL Hopkins, and WE VanderVelde. Guidance and navigation. 1965.

[19] Gonzalo Ferrer Mínguez. Integración kalman de sensores inerciales ins con gps en un uav. *Proyecto Final de Carrera, Universitat Politècnica de Catalunya*, 2009.

[20] Jacob Fraden. *Handbook of modern sensors: physics, designs, and applications*. Springer Science & Business Media, 2004.

[21] Mohinder S Grewal, Lawrence R Weill, and Angus P Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[22] Frederick J Kull. Inertial reference unit. 1973.

[23] S GREWAL Mobinder and P ANDREWS Angus. Kalman filtering theory and practice using matlab, 2001.

[24] Knut Mørken. Numerical algorithms and digital representation. *Lecture Notes for course MATINF1100 Modelling and Computations,(University of Oslo, Ch. 11, 2010)*, 2013.

[25] Melvin M Morrison. Inertial measurement unit, December 8 1987. US Patent 4,711,125.

[26] Antonio Serna, Francisco Ros, and JC Rico. *Guía práctica de sensores*. Creaciones copyright SL, 2010.

[27] Lance D Slifka. An accelerometer based approach to measuring displacement of a vehicle body. *Master of Science in Engineering, Department of Electrical and Computer Engineering, University of Michigan–Dearborn*, 2004.

[28] David Titterton and John L Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004.

[29] Charlotte Treffers and Luc van Wietmarschen. Position and orientation determination of a probe with use of the imu mpu9250 and a atmega328 microcontroller. 2016.

[30] David Vara Rodríguez. Sistemas para determinar la posición y orientación de herramientas quirúrgicas en operaciones de cirugía laparoscópica. *Grado en Ingeniería Electrónica Industrial y Automática, Escuela de Ingenierías Industriales, Universidad de Valladolid, España*, 2014.

[31] Oliver J Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory, 2007.