Department of Electrical and Computer Engineering
University of Thessaly

Diploma Thesis

Σχεδιασμός και υλοποίηση συστήματος πλοήγησης ρομπότ για εσωτερικούς χώρους

Design and development of navigation system for robots in indoor environments

Christos Davilas

Supervisors:

Athanasios Korakis

Antonios Argyriou

Volos

21 October 2016

Dedicated to my family,

Acknolowgdements

Firstly, I would like to thank my supervisors Athanasios Korakis and Antonios Argyriou for their support and that they gave me the opportunity to be part of such important project. In addition, I am grateful to PhD candidate Giannis Kazdaridis for his help and valuable advices throughout this project. Moreover, I would like to thank NITLab's personnel for their support and their guidance which help me to get improved.

Finally, I would like to thank my family and my friends for their support all these years.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου Αθανάσιο Κοράκη και Αντώνιο Αργυρίου για την υποστήριξη τους και τη δυνατότητα που μου παρείχαν να εργαστώ σε ένα τόσο σημαντικό πρότζεκτ. Επίσης, είμαι ευγνώμων στον υποψήφιο διδάκτορα Γιάννη Καζδαρίδη για την πολύτιμη βοήθεια του όπως και τις πολύτιμες συμβουλές του καθόλη τη διάρκεια της πτυχιακής. Επιπλέον, θέλω να ευχαριστήσω όλα τα παιδια του NITLab με τα οποία συνεργάστηκα και με βοήθησαν στην εκπόνηση αυτής της διπλωματικής.

Τέλος, χρωστάω απέραντη ευγνωμοσύνη στην οικογένεια μου όπως και στους φίλους που με στήριξαν όλα αυτά τα χρόνια.

Abstract

The goal of this thesis is the development and implementation of a navigation system, which will allow us to control a mobile node. The mobile node is going to be used for the needs of indoor Network Implementation Testbed Laboratory (NITLab) of the University of Thessaly.

We are using a trilateration algorithm for the localization of the mobile node inside the Testbed. The measurements of distances between three reference points (anchors) and the mobile node are achieved by Time Domain's P410 UWB Radio modules. In addition, we are using Inertial Measurement Unit (IMU) to acquire the orientation of the mobile node which will allow us to control it better and adjust its velocity. Finally, we are using a Kalman filter to smooth the results from the localization algorithm and improve accuracy for the spatial coordinates of the mobile nodes.

The mobile node is moving with 7cm/s and its length is 24.5 cm. We have succeeded accuracy 1 centimeter for the spatial coordinates of the Rover in LOS conditions The Rover is executing a specific path that is given by the User and there is real-time illustration of its trajectory and of its orientation from a GUI.

Contents

List of Figures

1. Introduction

1.1 Purpose of this project

The goal of this project is to integrate the autonomous robot at the NITLab's testbed so that RF experiments can be conducted with the aid of mobile nodes. Indoor positioning systems (IPS) are widely used nowadays in GPS-less environments and have many applications in everyday life such as health, indoor object search, work, military and in mining sites. We are using IPS to track and to navigate our autonomous robot in a GPS-less environment such as NITLab's testbed. Our autonomous robot is using UWB wireless communication for indoor positioning and Kalman filter for improving the results. Furthermore, we are using an IMU to acquire the orientation of the robot and to adjust the velocity of its wheels. We have achieved accuracy of ± 2 cm in LOS environment using Time Domain's devices P410s for ranging and accuracy of ± 1 cm in LOS environment using linear least squares technique for positioning.

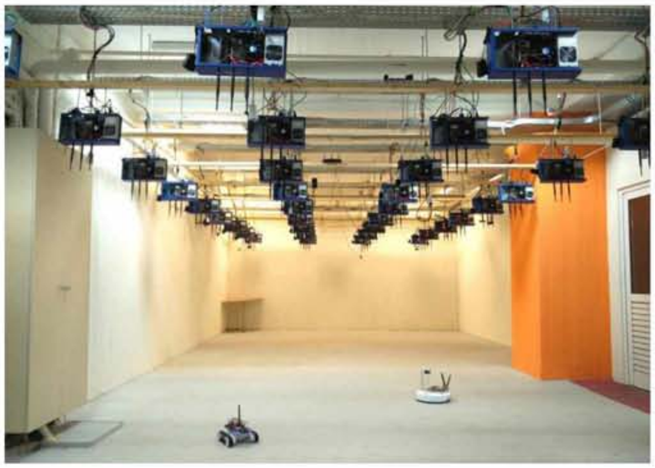


Figure 1.2.1 Indoor NITLab's Testbed

1.2 Related Work

One of the most difficult navigation problems is autonomous mobile robot navigation in GPS-less environments such as buildings, mines etc. Much research has been made the last decade for the development of indoor positioning systems based on different technologies.

The most accurate results are achieved with UWB technology from 2 cm to 50 cm depending the sensors that are used as indicate many papers and theses (1) (2) (3) (4). The devices that are used for UWB positioning are Time Domain's devices (5) or Decawave modules (6) (7) (8) which the error varies from 31 to 40 cm. (9) (10) In addition, other RF technologies have been used with poorer results such as Wifi (11) ,ultrasound , RSSI for indoor WSN (12) , Zigbee networks (13). Furthermore, another method is dead reckoning for mobile robot (14) , other issues of mobile robot about planning are described as well (15) (16).

1.3 Thesis Structure

The thesis is divided in six Chapters with following content:

Chapter 2: contains the main ranging, positioning techniques and the signal types that are used for indoor localization. Furthermore, we analyze the reasons that we have picked Linear Least Squares algorithm for trilateration and we describe analytically the parameters of Time Domain's (P410s) devices as well as the API of these devices.

Chapter 3: describes the limits of the Rover that we are using and what information we can extract from the Rover's encoders. In addition, we have tested 3 IMU's to achieve the best accuracy and we present the API of MTi-3. Finally, we explain the technique that we are using so that the Rover follows a straight line.

Chapter 4: analyzes the Kalman filter and its benefits. Also, we find the best values for the parameters through experiments and we describe the behavior of the filter in different parameters.

Chapter5: describes the configuration of the Beaglebone that we are using. Also, contains the system design and explains the algorithm of the whole system. Finally, illustrates the results from the executed path through a GUI.

Chapter6: summarizes the thesis and looks ahead for the next steps of the project.

2. Localization

One of the most essential parts of a navigation system for autonomous robot is the localization. There are many techniques that have been developed for indoor localization. In this chapter we focus in wireless technologies for indoor localization. The first step is to understand positioning techniques and then to understand how localization is achieved. Finally, we will illustrate what are the signal types that are used for these techniques (17), (18) , (19) .

2.1 Ranging Methods

There are three main categories of ranging methods in the bibliography. These four categories are Angle of Arrival (AOA), time based methods which include Time of Arrival (TOA), Time difference of Arrival (TDOA), Round Trip Time (RTT), Received Signal Strength Indicator (RSSI).

2.1.1 Angle of Arrival

The location of user can be found of the received signal angle. An Angle of Arrival (AOA) measurement is a method for determining the direction of propagation of a radio-frequency wave incident on an antenna array. To determine the location of a node in a two-dimensional (2-D) space, it is sufficient to measure the angles of the straight lines that connect the node and two reference nodes (19) Figure. The advantages of AOA methods are that position estimation may be determined with as few three measuring units for 3-D positioning and that no time synchronization between measuring units is required. The main limitation of this method is that, it is affected by multipath and NLOS propagation of signals, along with reflections from walls and others objects. In addition, the cost is bigger than the others methods due to the additional antennas that are required to measure the angles.

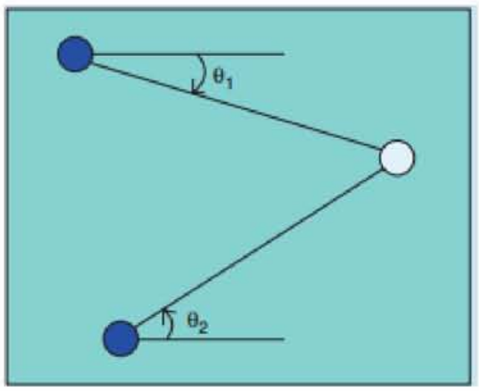


Figure 2.1.1 Positioning via the AOA measurement (19)

2.1.2 Time Based Methods

The most commonly categories of techniques that are used for positioning are time based methods. The techniques that are based on the measurement of the travel time of the signal propagating from the transmitter to the receiver. (17)

1. Time of Arrival

Time of Arrival (ToA) is the simplest method of time based methods. In the most simple situation in which two nodes have a common clock, the receiver node can determine the ToA of the incoming signal and directly calculate its distance from the transmitter, by multiplying ToA , after subtracting the known time of transmission, by the speed of light (17). In this way, the estimated range allows to draw a circle and radius equal to the estimated range. This method has two main limitations. First of all, transmitters and receivers have to be perfectly synchronized .Second; a timestamp must be labeled in the transmitting signal in order for the measuring unit to discern the distance has traveled (20)

2. Time Difference of Arrival

The Time Difference of Arrival (TDoA) method is solving partially the problem of the synchronization. The TDoA method determines the relative position of the mobile transmitter by examining the difference in time at which the signals arrives at multiples reference nodes. Each difference of arrival time measurement produces a hyperbolic curve in the localization space on which the location of the mobile node lies. In this method it must be synchronization between the reference nodes. This method it is quite similar of how GPS works.

3. Round Trip Time

The Round Trip Time (RTT) method also it is called Two-Way Time-of-Flight (TW-TOF) (21), is solving totally the problem with synchronization. The measurement equals to the amount of time it takes a signal to leave the Requester's antenna, arrives at the Responder's antenna and return. Dividing this number by 2 and multiplying by the speed of light yields the distance in millimeters. Fig The biggest limitation of this method is the range measurements to multiple devices to be carried out simultaneously which causes latency for devices which moves quickly. Another limitation of this method is that the devices must operate both as transmitters and receivers.

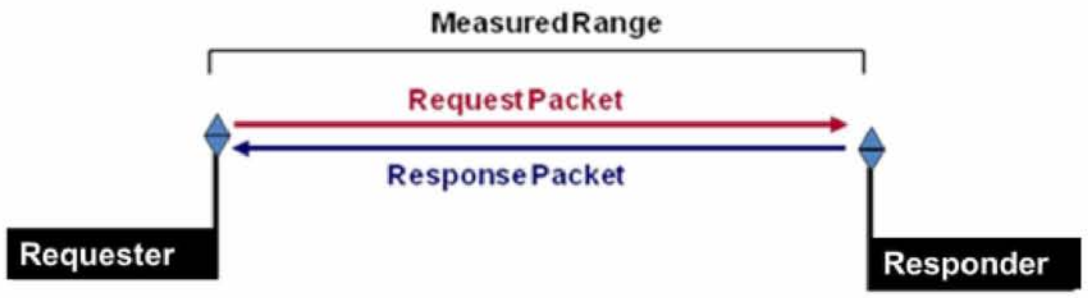


Figure 2.1.2 RTT Ranging (21)

2.1.3Received Signal Strength Methods

Received Signal Strength (RSS) methods are widely used in literature as are based on a sensor function available in every 802.11 interface. In RSS methods the distance between two nodes is calculated by measuring the energy of the received signal. Having a priori knowledge of the distance dependency of the received signal power and of the transmitted one, the range between the transmitter and the receiver can be estimated. This method is very unstable as it is vulnerable at multipath fading, and signal attenuation due to the changes in temperature, humidity and objects mobility (11).

2.2 Positioning Methods

Positioning methods are depending on the ranging method that was used to acquire the distance.

2.2.1 Triangulation

Triangulation uses the geometric properties of triangles to estimate the target location.It has two derivations, lateration and angulation. Lateration estimates the positioning of an object by measuring its distances from multiple references points, so it is called multilateration. . If we have three reference points it is called trilateration (22). Angulation techniques from the other hand use AOA ranging method to measure distance between two nodes.

Trilateration methods use TOA, TDOA, and RTT ranging methods to measure the distance. The estimated range allows drawing a circle and radius equal to the estimated range. The blue spots are the anchors that are installed in prefixed points and the white spot is a mobile node.

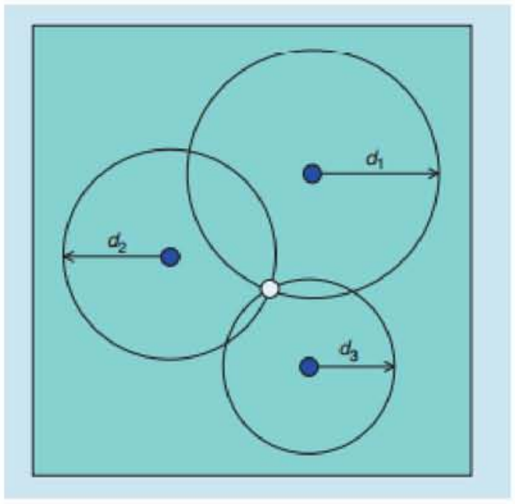


Figure 2.2.1 Trilateration Scheme. (19)

1 Anchors Placement

In trilateration based methods, the geometry of the anchors relative to the mobile node, have a large effect on the resulting position error. Consider three distinct anchor nodes pi= (xi,yi) . We calculate the coordinate (x0,y0) of an unknown node p according to ri (the distances between pi and p). The measurement error exists in localization process of the unknown node. The size of the small region can be regarded as the size of the localization error. Thus, if we can reduce the size of the small region, then we can decrease the size of the localization error. (23) (24)

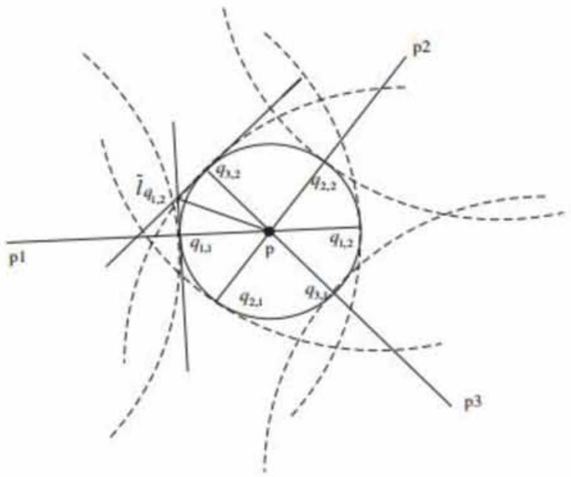


Figure 2.2.2 Analysis of localization error (24)

Theorem 1: Area (Cpi) reaches its minimum when a1,2=a2,3=a3,1 = π/3. In other words, symmetrical reference nodes (in the sense of the angle direction) form an optimal configuration.

So, in order to get more precise position information of the mobile nodes in indoor environments, we can place the three anchors that form an *equilateral* triangle.

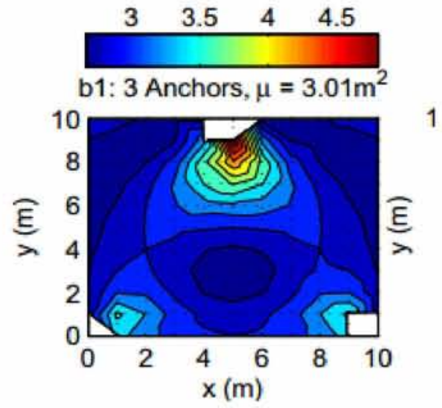


Figure 2.2.3 Highest Accuracy (23)

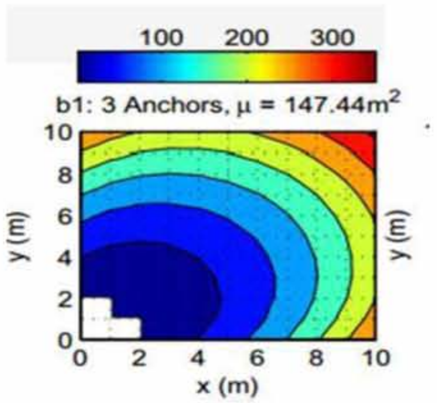


Figure 2.2.4 The variance of the estimation is the highest

Theorem 2: The communication radius of an anchor is r. When the side length of the equilateral triangle is $\sqrt{3r}$, the coverage area without hole is the largest.

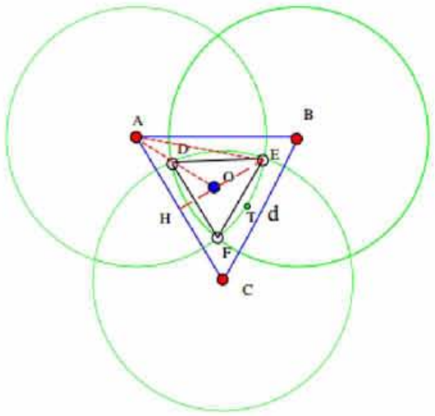


Figure 2.2.5 Optimal Coverage without hole (24)

The radius of P410s devices are 60 meters with pii equals to 5. In that case the length of the equilateral triangle must be $\sqrt{3r} * 60$ that means 104 meters. In addition, in the trilateration methods essential role has the solver algorithm to achieve high accuracy. These algorithms are going to be explained in another section.

2.2.2 Fingerprint

RSS based methods are using fingerprint technique for localization. The main idea is to collect features of the scenes from the surroundings signatures at every location in the areas of the interest and then build a fingerprint database. This is happening during the offline stage. During the online stage a location position technique uses the currently observed signal strengths and previously collected information to figure out an estimated location. The main limitation as we mention for the RSS based methods is that it is affected by diffraction, reflection and scattering. From the other hand the other hand it has the advantage that it is not required any synchronization between the stations and it is not required and specialized hardware (20).

2.2.3 Proximity Detection

The simplest positioning method is proximity detection. This method is providing symbolic relative location information. Usually, it relies upon on a dense grid of antennas, each having a well-known position. When a mobile target is detected by a single antenna, it is considered to be collocated with it. This method is implemented with various wireless technologies such as radio frequency identification (RFID), Bluetooth, ZigBee and infrared radiation (IR). The biggest disadvantage for proximity based methods is that it has a huge variance which is not satisfied in indoor localization.

2.2.4 Dead Reckoning

Dead reckoning is the process of estimating known current position based on last determined position and incrementing that position based on known or estimated speeds over elapsed time. An IMU is providing all the information that the systems need, such as directional information and the acceleration of the mobile node. The main disadvantage of this method is that the error is cumulative, so the deviation in the position fix grows with the time.

2.3 Signal Types

There are many systems designed for indoor localization that can be classified based on signal types. The most widely used signal types for indoor localization are RFID, Bluetooth, WLAN, ultrasound and UWB that we are using in this thesis.

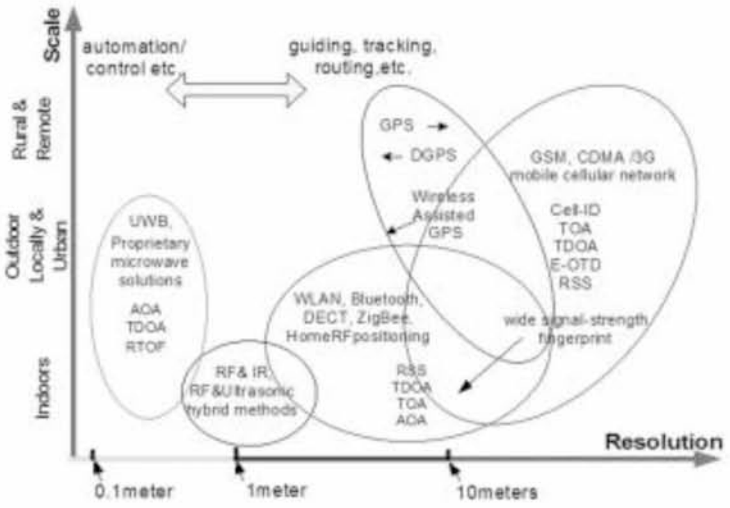

Scale
Rural & Remote
Outdoor Locally & Urban
Indoors
automation/ control etc.
guiding, tracking, routing,etc.
GPS
DGPS
Wireless Assisted GPS
GSM, CDMA /3G mobile cellular network
Cell-ID
TOA
TDOA
E-OTD
RSS
UWB, Proprietary microwave solutions
AOA
TDOA
RTOF
WLAN, Bluetooth, DECT, ZigBee, HomeRFpositioning
wide signal-strength fingerprint
RF& IR, RF&Ultrasonic hybrid methods
RSS
TDOA
TOA
AOA
Resolution
0.1meter
1meter
10meters

Figure 2.3.1 Signal Types (20)

2.3.1 RFID

Radiofrequency identification (RFID) is a simple technology and it is based on two parts: tag and readers. Tracking the movement of objects in RFID is done through a network of radio enabled scanning devices over a distance of several meters. The RFIG tags are categorized as either passive or active. Passive RFID tags operate without a battery, in comparison with active RFID tags that are small transceivers. The accuracy of an RFID system is highly depending on the density of tag deployment and the maximal reading ranges. The typical frequency ranges are categorized as Low Frequencies (LF) at 125 kHz, High Frequencies (HF) at 13.56 MHz, Ultra-High frequencies (UHF) at 868 to 915 MHz, and microwave frequency 2.45 GHz. The basic limitation of this signal type is that the typical range is 1-2 meter, and the cost of the readers is relatively high.

2.3.2 Bluetooth

Bluetooth is a Wireless Personal Area Networks (WPANs), which has a maximum power output of 1 mW which enables communication of 5 m to 10 m depending on the propagation condition. Bluetooth tags are small size transceivers, as any other Bluetooth device, each tag has a unique ID. This ID can be used for locating the Bluetooth tag. The advantage of using Bluetooth for exchanging information between devices is that this technology is of high security, low cost, low power and small size. From the other hand, this signal type has latency unsuitable for real-time positioning applications, as it has bit rate lower than 1 Mbps.

2.3.3 ZigBee

ZigBee is another wireless technology standard which can be regarded as low rate Wireless Personal Area network (WPAN), IEEE 802.15.4 standard protocol. The main advantage of this technology is that requires low-power consumption. The signal coverage of ZigBee is up to 100 meters in free space. Distance estimation between two ZigBee nodes is carried out from RSSI values.

2.3.4 WLAN

The most positioning systems are based on WLAN, IEEE 802.11 standard protocol, which is a midrange wireless local area network and operating in 2.4 GHz. With a typical gross bit rate of 11, 54, or 108 Mbps and a range of 50-100 meters. Distance estimation is performed on measurements on the RSSI values. This method is highly used since WLAN access points are readily available in many indoor environments and it is possible to use standard mobile hardware devices.

2.3.5 Ultrasound

The ultrasound signals are used to estimate the position of the emitter tags from the receivers. It has high level of accuracy but it suffers from a lot of interference from reflected ultrasound signals propagated. The range of ultrasound is up to 80 meters and time based methods can be used for positioning.

2.3.6 Ultra Wide Band

The origin of UWB technology can be traced back to the early 1960's on time domain electromagnetics, when the study of electromagnetic-wave propagation was primarily viewed from the time domain aspect. The term UWB was first time used in 1994 by the U.S Department of Defense to indicate impulse based systems. UWB is a short duration; pulsed RF technology that achieves the highest possible bandwidth at the lowest possible center frequency Ultra Wide Band has relative bandwidth larger than 20% of more than 500 MHZ. Such wide bandwidths offer a wealth of advantages for communications, radar applications and ranging/location applications.

The most distinguished difference between an impulse system and frequency modulated continuous wave system (FMCW) is in the transmitting waveform. An FMCW system transmits and receives CW signals, one signal at each frequency, subsequently across a bandwidth that means that an FMCW system is basically operated over a bandwidth of single frequency system. In contrast, an impulse system transmits and receives a periodic impulse type signal which contains many constituent signals occurring simultaneously, each having different frequency. UWB systems achieve this bandwidth by transmitting an impulse like-waveform. Such waveforms are inherently broadband. Fourier analysis teaches us that an ideal impulse would provide infinite bandwidth. Transmissions as we described above, they resemble a train of pulses.

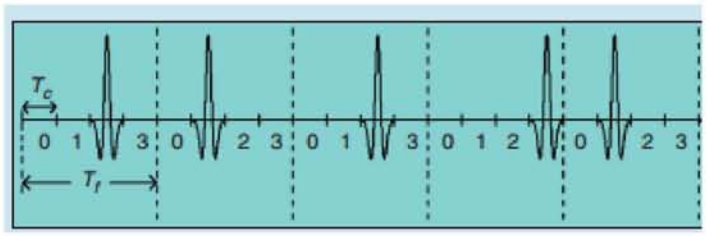


Figure 2.3.2 A sample transmitted signal from a time hoping impulse radio UWB (19)

Time domain P410 devices that we are using, relies on low duty cycle transmissions, with coherent signal processing and typical repetition rates of 10 MHz's Because the transmissions are coherent, the signal energy can be spread over multiple pulses, thereby increasing the signal to noise ratio. For communication and positioning applications, the operating band is limited in the portion of spectrum between 3.1 and 10.6 GHz, provided that the emissions satisfy given limitations.

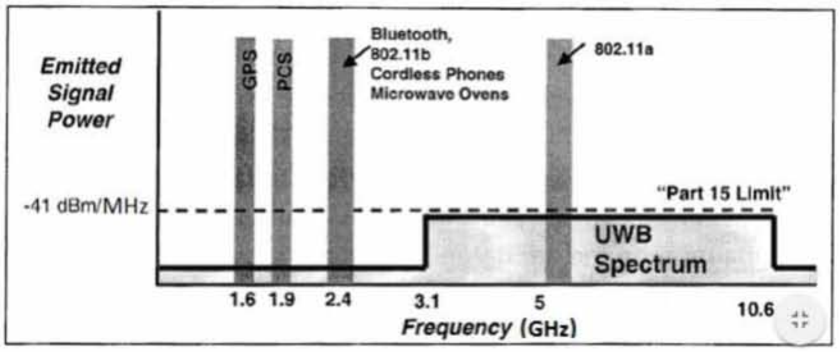


Figure 2.3.3 Frequency domain for UWB signals. (1)

More specifically, P410 time domain's devices have center Frequency at 4.3 GHz and the operating band is limited between 3.1 to 5.3 GHz.

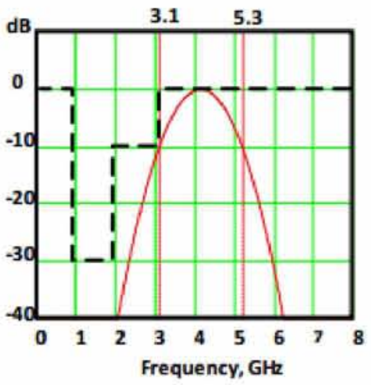


Figure 2.3.4 Frequency domain P410 devices. (25)

A large absolute bandwidth improves reliability, as the signal contains different frequency components, which increases the probability some of them can go through or around an object. As well, a large bandwidth offers high resolution radars with improved ranging accuracy.

In addition, the transmitting energy spectral density of UWB systems is much lower than that of CW based systems for the same input power; this effectively produces much smaller interference to the signals of other co-operating RF systems. UWB systems offer excellent immunity to interference from other existing signals.

Furthermore, for communication spreading information over a large bandwidth decreases the power spectral density, thus reducing interference to other systems and lowering the probability of interception, which is desirable for secure and military applications. Also, UWB systems lead to high frequency diversity that reduces the chance of signal diminishing in certain operating environments with severe multi-path fading at some frequencies, such as indoors, urban settings, or when signal attenuation at some frequency are excessively high, resulting better immunity to destructive environments.

UWB radar and sensor can find numerous applications for military, security, civilian and medicine. First of all, UWB systems are used for military and security applications. Some military applications are detection, location and identification of targets such as vehicles or aircrafts. Also, locating and tracking personnel, or through-wall imaging and surveillance. Finally, building surveillance, monitoring and intrusion detection. In addition, some medical applications are detection and imaging of tumors, health monitoring of elders and medical imaging.

2.4 Propagation

2.4.1 LOS Propagation

Line of sight is a type of propagation that can transmit and receive data only where transmit and receive stations are in view of each other without any of an obstacle between them. (26)

2.4.2 NLOS/Multipath Propagation

When the direct LOS between two nodes is blocked, only reflections of the UWB pulse from scatters reach the receiving node. Therefore, the delay of the first arriving pulse does not represent the true TOA. Since, the pulse travels an extra distance; a positive bias called NLOS error is present in the measurement delay. Because, the pulses are physically small, a UWB system can separate, or resolve, multipath reflections from a main signal by focusing in the first arriving signal.

2.5 Multilateration Techniques

The calculation of the spatial coordinates of an object from its distances to other known points is known as multilateration. The basic approach to solve this problem is to treat the coordinates of the object as the point of intersection of several spheres, whose centers are the locations of the anchors$P1(x1, y1, z1), P2(x2, y2, z2), P3(x3, y3, z3)$. The exact distances between the anchors and the object,ri, are the radii of the individual spheres. The equation for any of these spheres is

$$(x - xi)^2 + (y - yi)^2 + (z - zi)^2 = ri^2 \quad (1)$$

The true distance between the anchor i and the targeted object is given by the following equations:

2.5.1 Linearized System of Equation

Linearizing the system of equations geometrically converts the problem into one of finding the point of intersection of several planes. When the exact distances from the anchors are available, the solution of the linear equation is completely determined. When not the exact distances are used, the position that is calculated by the direct solution of the linear equation is no longer acceptable. We have run many experiments using linear equation and the error was too big and sometimes the results were incoherent.

The linear system is easily written in a matrix form:

$$A \cdot \vec{x} = \vec{b} \quad (2)$$

With

$$A = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \vdots & \vdots & \vdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{pmatrix}, \quad \vec{x} = \begin{pmatrix} x - x_1 \\ y - y_1 \\ z - z_1 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{pmatrix}$$

2.5.2 Linear Least Squares

The coordinates of the rover's position was finally obtained by applying the linear least squares method to the linear system equations. The results are more accurate than the coordinates obtained by solving the linearized system of equation directly. Since the distances r_i are only approximate, the problem requires the determination of $\vec{x}$ such that $A \cdot \vec{x} \approx \vec{b}$. Minimizing the sum of the squares of the residuals,

$$S = (b - Ax)^T (b - Ax)$$

Leads to the normal equation:

$$A^T Ax = A^T b$$

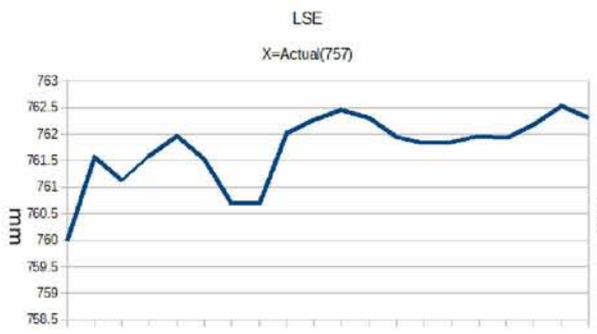
There are several methods to solve for $\vec{x}$. The condition number of $A^T A$ determines which method is the best. If $A^T A$ is non-singular and well-conditioned:

$$\vec{x} = (A^T A)^{-1} A^T b$$

If $A^T A$ is singular or poorly conditioned then the normalized QR-decomposition of A is used:

$$A = QR$$

We had used python's library Localization 0.1.4 for our experiments (27). The most important for this algorithm is that always returns results and the results of LSE are quite good. The node was positioned on the coordinates (757,373) and we illustrate 20 results of this algorithm. It has a small error at x-axis 8 mm and a reasonable error at y-axis 40 mm.



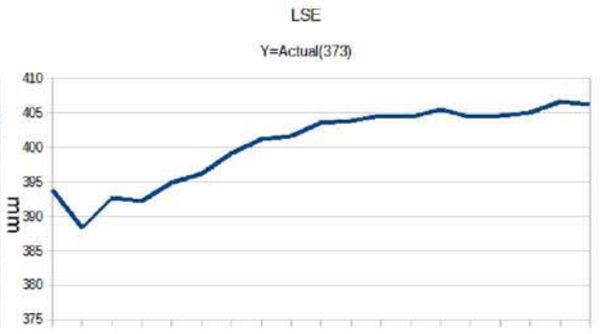
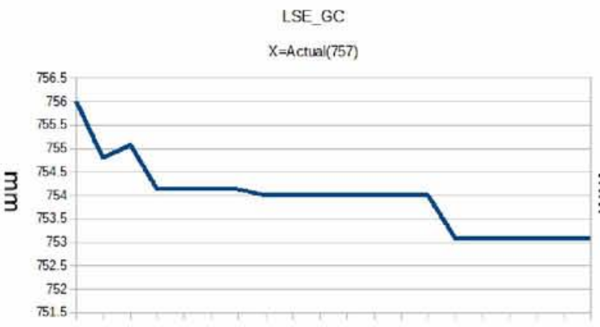


Figure 2.5.1 Results of LSE axis-x **Figure 2.5.2 Results of LSE axis-y**

2.5.3 Geometrical Constrained Least Squares

The GC-LSE algorithm is based on a regular geometrical system structure, such as the squares. Utilizing the relationship among the distances from different anchor nodes to the mobile node helps eliminate the exceptional errors. In order to get a reasonable result the abnormal distance is turned to a more accurate distance (28). This algorithm produces more precise results but it is very sensitive to big errors of the calculated distance between the anchors. These big errors blocks the algorithm and it does not return any result. Although, the variance is very small in both axes and the error is 4 mm at x-axis and 4 mm at y-axis



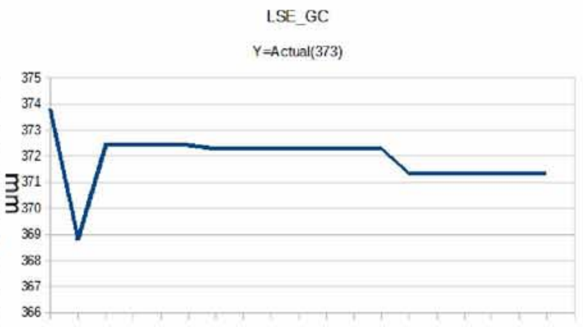


Figure 2.5.3 Result LSE_GC x-axis **Figure 2.5.4 Result LSE_GC y-axis**

2.5.4 CCA- Centroid Methods

The solution will be the centroid of the intersection area. If there is no common intersection, it will be used the area with the maximum overlap. Centroid of a triangle is the point where the three medians of the triangle intersect. This algorithm had the worst behavior and its errors were unacceptable on both axes. The error at x-axis is 159 mm and the error at y-axis is 200 mm.

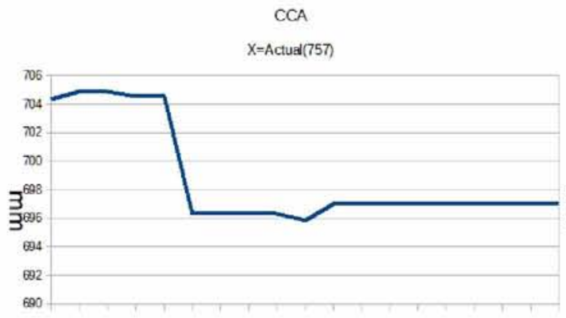


Figure 2.5.5 Results of CCA, x-axis

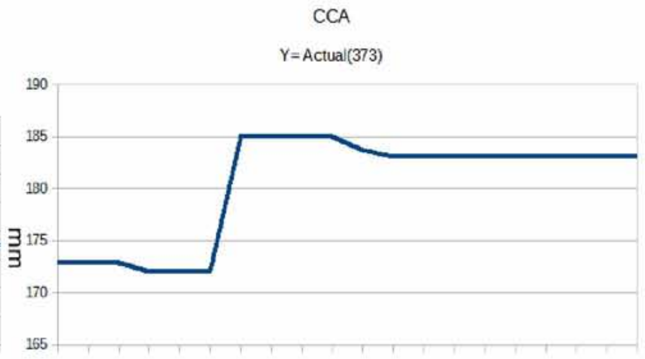


Figure 2.5.6 Results of CCA, y-axis

2.6 PulsON® 410

The PulsON 410 devices provide us with accurate range measurements among the four devices. They are using UWB pulsed RF signaling, a strategy ideal for two-way time of flight, RF transmissions are from 3.1GHz to 5.3 GHz, with center at 4.3 GHz and it have bandwidth 2.2 GHz. We are using three devices as anchors and the fourth is attached to the mobile node.

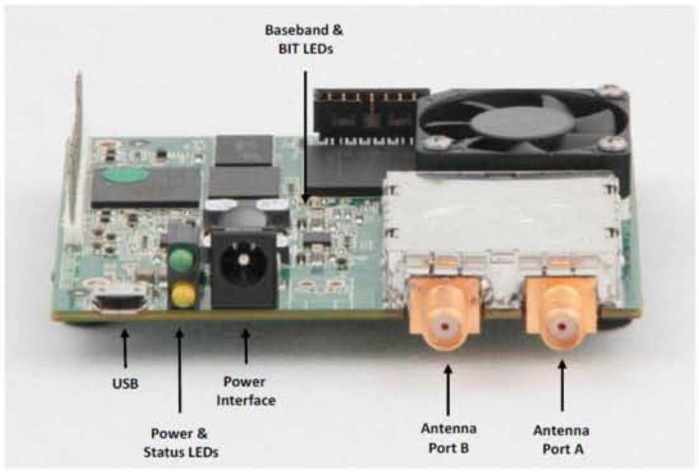


Figure 2.6.1 Side view of the P410 RCM, all anchors are using antenna port B for both transmission and reception, the mobile node is using antenna port B

2.6.1 Parameter Description

1. Precision and Accuracy in LOS condition

According to the manual, the line of Sight precision and accuracy specification are about 2.3 cm and 2.1 cm accordingly, and it is based on a measurement campaign that included 20,000 range measurements taken in an open field, over an operating range that varied from 2 ft. to 300 ft. for different PIIs. It is important to mention that the precision of the measurement can be improved with averaging. Measuring six reading will normally improve the accuracy by a factor of 2. Averaging more than 6 reading will have only marginal improvements.

Also, we have noticed experimentally, that if we accept only the measurements with the minimum error the precision will be improved but this will cause a huge increase in the measurement duration between the two anchors.

Furthermore, we conducted some experiments and we verified that we had to collect at least 6 measurements to improve the accuracy. Firstly, we had set the value of PII as 9 and we took 100 measurements between three anchors and a motionless node. The distances are referring to millimeter due to the limited available space that I had.

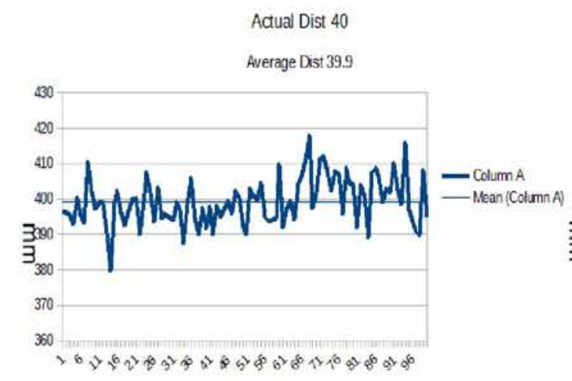


Figure 2.6.2 Distance between anchor1 and node

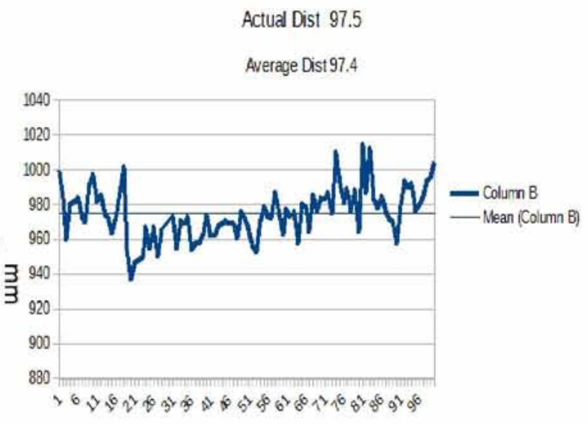


Figure 2.6.3 Distance between anchor2 and node

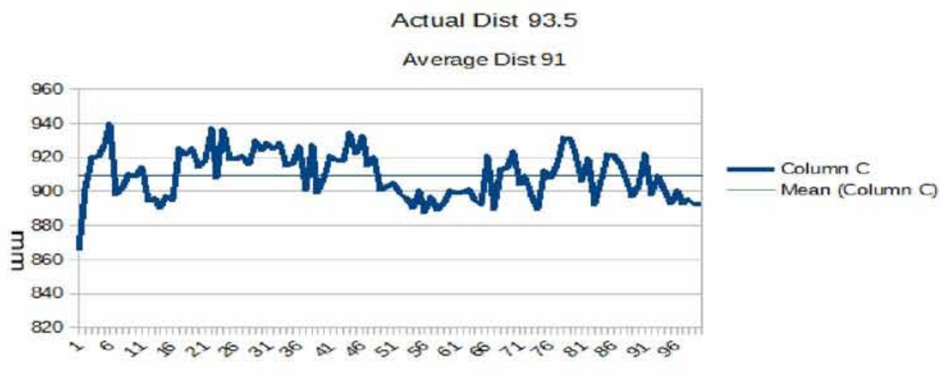


Figure 2.6.4 Distance between anchor3 and node

As we can see, averaging the results, have the best results since there is big variance between of single measurements.

2. Message ID

The message ID parameter is a convenient way for the system integrator to keep track of the messages and associated responses. The user specifies a unique ID for each request command sent to one or more local RCMs. The message ID becomes an important tool when logging data and post-processing for full system connectivity and range analysis. In fact, the host should rely on message ID for collating messages, rather than any particular order or timing of Range and Data messages.

3. Pulse Integration Index (PII)

This value determines the number of pulses used in each radio symbol or scan point. Larger PII values result in a higher signal to noise ratio (SNR) with longer distance operation at the expense of slower ranging and data rates. All RCM nodes must be preconfigured with identical PII values in order to establish communication and ranging.

The user configures the power of 2 of the pulse integration value. For example, a configured value of PII=7 results in the transmitting node sending 128 pulses per symbol and the node expecting 128 pulses per symbol.

PII	*Max Range** *(meters)*	*Data rate:* *(bps)*	*Precision Range Measurement* *(time, rate)*
4	*35*	*632k*	*6.5 ms, 154 Hz*
5	*60*	*316k*	*8.5 ms, 118 Hz*
6	*88*	*158k*	*12.5 ms, 80 Hz*
7	*125*	*79k*	*20 ms, 50 Hz*
8	*177*	*39.5k*	*36 ms, 28 Hz*
9	*250*	*19.7k*	*67 ms, 15 Hz*
10	*354*	*9.86k*	*132 ms, 8 Hz*

Figure 2.6.5 PII and resulting distance, data rate and range measurement rates. (5)

Firstly, we had set the value of PII at 9 and we were averaging 100 measurements; we had noticed the duration of measurements were too slow and the averaging time was several minutes. When, we changed the value of PII from 9 to 5 we noticed a significant drop for the averaging time, from several minutes (600 seconds) to less than a minute (40 seconds). It was almost 10 times smaller, which is quite reasonable if we notice the Figure 2.6.5.

4. Antennas

The P410 have two antennas port. The user can define which port is to be used for transmit and which is to be used for receive. Also, it supports automatic toggling of the default antenna port after each range response. We are using Antenna B for both transmission and reception.

The distance measurements correspond to the physical distance from the phase center of the requester's antenna to the phase center of the Responder's antenna. The RCMs have been calibrated by default to presume a zero Antenna Delay when using default Broadspec antennas with a simple 90 degree elbow SMA connector. In case, we use a different cable, connector, or antenna the Bias will change. The user has to specify by how many cms the calibration needs to be adjusted, then this number should be converted to picoseconds. To convert from cm to picoseconds we have to uses the following equation:

$$0.2993 \ mm \ of \ delay = 1 \ picoseconds$$

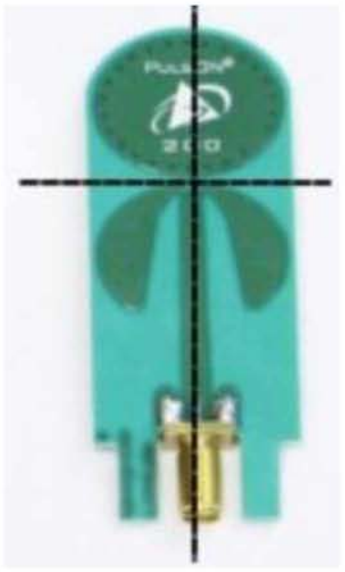


Figure 2.6.6: Phase Center of the default Broadspec antenna. (5)

5. Timestamp

Time stamp is the number of milliseconds that have elapsed since the latest RCM power-up. This parameter is used by the system integrator when the data was collected or data was received.

6. Vpeak and Channel Rise Time

Vpeak is a measure of the maximum absolute value of the amplitude measured just after the leading edge offset. This peak value is an improved form of Relative Signal Strength and rarely suffers from multipath fading. It can be used to estimate distance from a receive-only signal and it is the basis for the Coarse Range Estimate (CRE).

The Channel Rise Time metric is a measure of the time required for the direct path pulse to rise from noise to complete signal. A Chanel rise value more than 2 indicates a NLOS channel and a loss in range precision. It is a fundamental metric used TW-TOF range error estimate.

7. SNR

SNR equals to $20*\log_{10}(Vpeak\,/\,Noise)$ where Vpeak is a measure of the largest signal near to the leading edge of the scan and noise is an estimate of the noise prior to the leading edge. The first step to compute SNR is to compute the standard deviation of the first N entries in Full Scan Packet. Then we compute Noise, which Noise equals to 2*(Standard Deviation of the first N readings in the full scan packet)^2, and finally we are using the initial type.

8 .Precision Range Measurement (PRM) and PRM Error Estimate

PRM is the Two-Way Time of Flight distance between the two devices. The measurement should be considered the distance from antenna phase centers. PRME, return with each PRM, is an estimate of the standard deviation of the associated PRM. This metric is pulled from a look-up table indexed by features of the incident pulse such as Channel Rise and Vpeak.

We conducted some measurements to evaluate the performance of this parameter. Firstly, we set three anchors to fixed positions and a node at the position (30,113). Then, we were averaging 20 measurements from each anchor to the node; a measurement was being accepted only if the PRME was equal to zero. Finally, we run an LSE localization algorithm to extract the position of this node. This procedure executed 20 times.

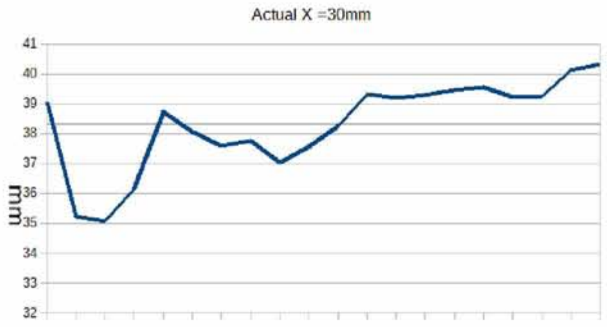


Figure 2.6.7 Difference of position-x, 0.8cm, PRM

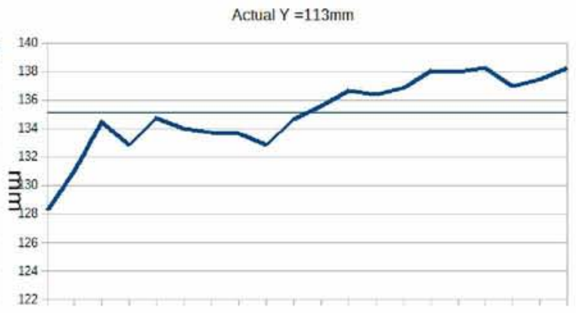


Figure 2.6.7 Difference of position-y: 2.2 cm,PRM

We notice that the difference of the real position-x is almost 0.8 cm and from the real position-y is almost 2.1 cm. These are good results which satisfies the manual's description about accuracy and precision.

9. Coarse Range Estimate (CRE) and CRE Error Estimate

This technique can suffer from errors due to channel and antenna pattern changes. CRE is only valid in Line of Sight environments, not saturated conditions (saturation occurs even if the transmitting radio is at the lowest transmit gain and the receiver is LOS and within 16 feet (4.9m)). Furthermore, CRE should not be used at distances greater than 100 meters. CREE is also reported with each CRE to support combination of CRE with other distance measurements. These values are generated by a look up table informed by the correlation between the SNR and CRE Error. CRE standard deviation error is approximately 10% at short distances, growing up to 30% at very long distances.

We conducted the same experiment as above with the following results:

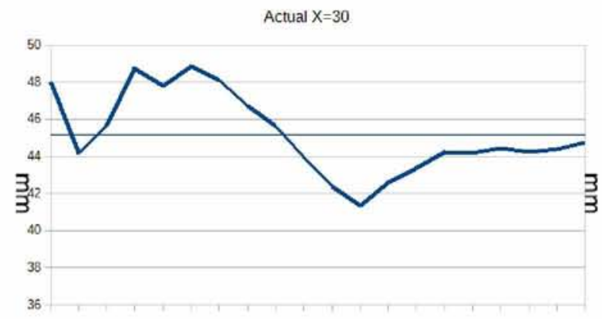


Figure 2.6.8 Difference of position-x,1.5 cm,CRE

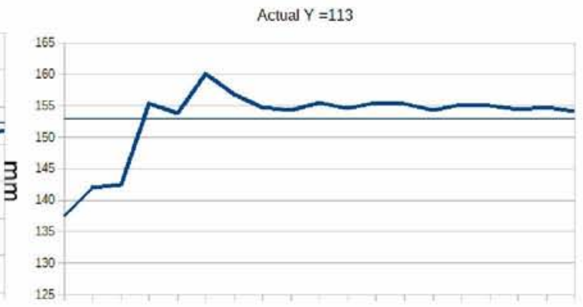


Figure2.6.9 Difference of position-y,4 cm,CRE

As we notice from these two figures in both axes there is big difference from the real position and the calculated, 1.5 cm difference at position x and 4 cm at position y. This is may be caused due to the fact that the environment is saturated as the RCMs are closer than 4.9 meters.

10. Range Filter

The filtered range estimate and filtered range velocity as their associated error free and frv are the result of a recursive Kalman filter. The Kalman filter is based on two state variables range and range velocity. Whenever a new PRM or valid CRE measurement is generated, this filter performs a weighted combination of these measurements with an internal linear motion model of distance.

The same experiment was conducted for FRE:

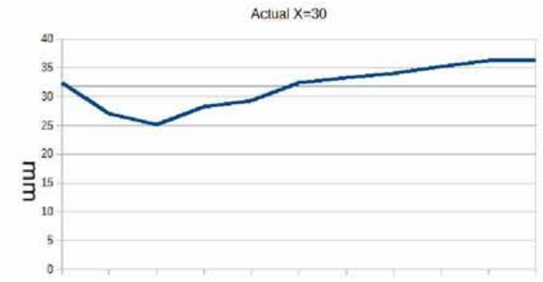


Figure 2.6.10 Difference of position-x,0.1 cm,FR

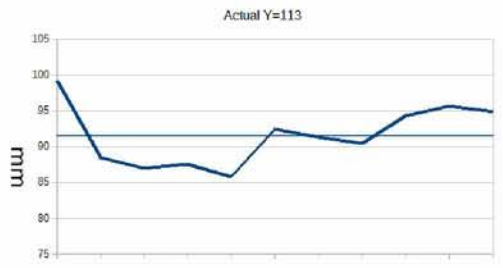


Figure 2.6.11 Difference of position-x,2 cm,FR

Both figures indicate that the FRE has the best performance as the difference from x-axis is 0.1 cm and the difference from the y-axis is 2 cm. So for our system we are using filter range estimate and we accept only the values that have filter range estimation error equal to zero. That means that the most measurements are being rejected and the duration of the averaging is increasing proportionally of how saturated the environment is.

10. LED Flags

Flags indicating saturation, NLOS, LOS conditions are generated by the radio firmware any time a new packet is received.

2.6.2 API Description

This is a high-level description of data passed between a Host processor and the RCM. The HOST-RCM interface consists of six Request messages from Host to RCM with their associated confirm messages. In addition, there are three INFO messages that are sent to the host upon receiving UWB packets from other RCMs. These three INFO messages are RCM_SCAN_INFO, RCM_DATA_INFO, RCM_RANGE_INFO. RCM_SCAN_INFO is only sent to the host if the host has pre-configured the RCM to send it.

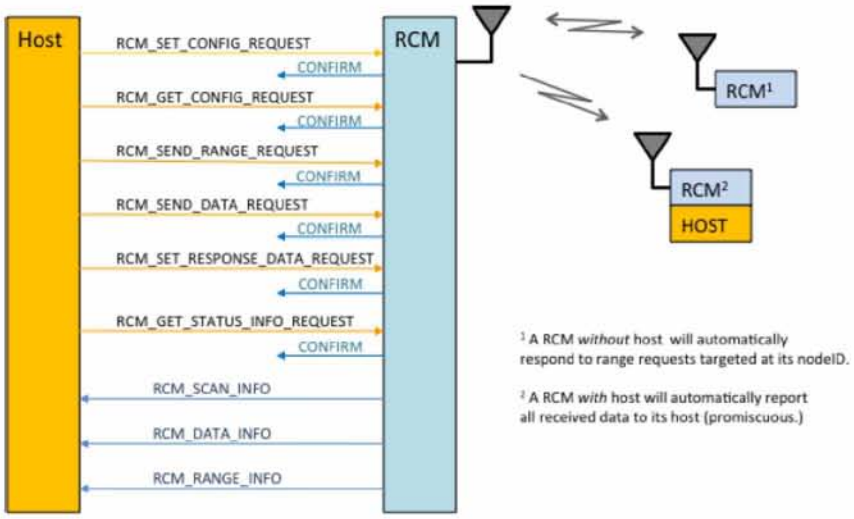

Host
RCM_SET_CONFIG_REQUEST
CONFIRM
RCM_GET_CONFIG_REQUEST
CONFIRM
RCM_SEND_RANGE_REQUEST
CONFIRM
RCM_SEND_DATA_REQUEST
CONFIRM
RCM_SET_RESPONSE_DATA_REQUEST
CONFIRM
RCM_GET_STATUS_INFO_REQUEST
CONFIRM
RCM_SCAN_INFO
RCM_DATA_INFO
RCM_RANGE_INFO
RCM
RCM1
RCM2
HOST
1 A RCM without host will automatically respond to range requests targeted at its nodeID.
2 A RCM with host will automatically report all received data to its host (promiscuous.)

Figure 2.6.12 Message flow, Host to ECM messages. (5)

In our system we our sending one RCM_SEND_RANGE_REQUEST to each anchor, the local RCM responds with a confirm message which means the range request packet was successfully sent by the local RCM to the three anchors. Then the local RCM sends to its host an RCM_RANGE_INFO at the end of a full UWB ranging conversation or timeout.

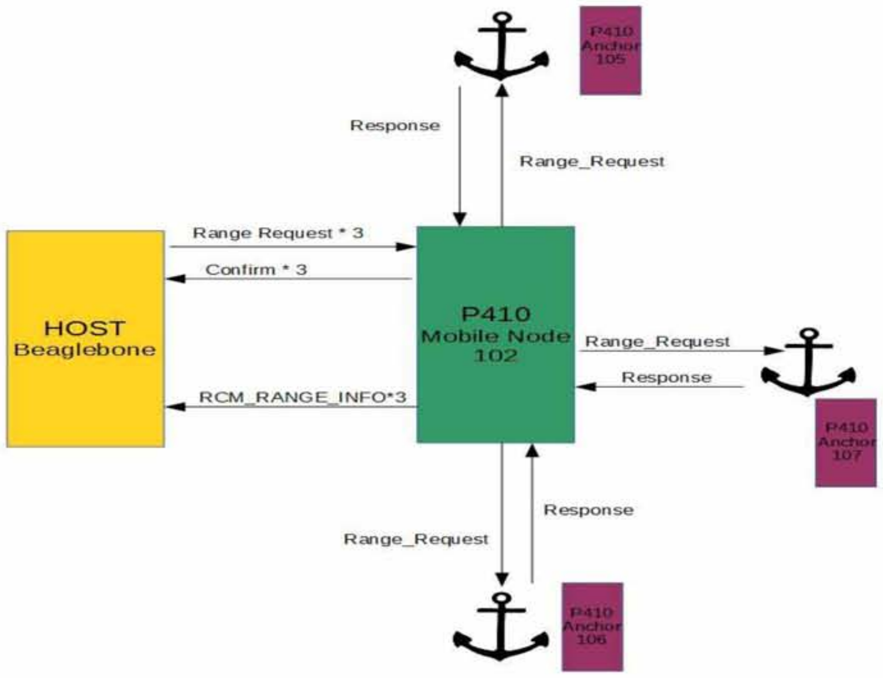


Figure 2.6.13 Message flow of our System

We are communicating with the local RCM with a serial UART interface which uses the following communication parameters: 115,200 baud rate, 8 data bits, no parity, and 1 stop bit. No flow control is used. The serial interface requires having a 4 byte prefix and it requires 2 byte suffix. The first two bytes of the prefix are a synchronization pattern 0xA5A5 and the next 2 bytes are the length of the command packet (not counting the prefix and suffix). The suffix is a 16-bit cyclic redundancy check (CCIT CRC-16) to ensure the validity of the data. The length bytes and CRC bytes must be sent in big endian order.

1. RCM_SEND_RANGE_REQUEST

Firstly, we send a range request to the RCM for each of the three anchors. The RCM will send a UWB range request packet to each of the three targeted anchors. The targeted nodes will respond with a range response. In the following example we will examine the structure of a packet from the RCM 102 to the RCM 106(anchor). The main changes that are occurring to the code are the different message_id and the different target node each time.

0xA5A5	Prefix, synchronization pattern
0x000C	Length of the command packet which is 12
0x0003	Indicates the message type, RANGE_REQUEST
0x00001	Indicates the number of the packet
0x0000006A	Specifies the node ID of the ranger request target
0x00	Specifies the antenna mode 0=A, 1=B,2=TXA,RXB,3=TXB,RXB
0x00	Reserved
0x00	Data Size
0xCCAB	Suffix

Table: Analysis of packet RCM_SEND_RANGE_REQUEST

2. RCM_SEND_RANGE_CONFIRM

After the SEND_RANGE_REQUEST a message RCM_SEND_RANGE_CONFIRM is sent by the local RCM to the host when the UWB range request packet was successfully sent by the RCM. The structure of this message type is explained below:

0xA5A5	Prefix, synchronization pattern
0x0008	Length of the command packet which is 8
0x0103	Indicates the message type, RANGE_CONFIRM
0x00001	Indicates the number of the packet
0x00000000	Shows the status , 0 equals to successful status ,non-zero equals to error
0x3500	Suffix

Table: Analysis of packet RCM_SEND_RANGE_CONFIRM

3. RCM_RANGE_INFO

Finally, we are expecting the last message type which is RCM_RANGE_INFO; this message type is sent by the requesting RCM to its host at the end of a full UWB ranging conversation or timeout. We have not activated RCM_SCAN_INFO and we do not expect any DATA_INFO. The structure of this message type is analyzed below from a snapshot during the communication of the RCM 102 which is connected to the host via UART and the RCM 106 which is an anchor.

0xA5A5	Prefix, synchronization pattern
0x0034	Length of command packet which is 52
0x0201	Shows the message type, RCM_RANGE_INFO
0x0001	Indicates the number of the packet
0x0000006A	Responder ID of the UWB module that sent the range response
0x00	Range status, 0 equals to success
0x00	Antenna mode, specifies the antenna port used during the range conversation
0x0043	Stopwatch time is the duration of the conversation time in milliseconds which is 67 ms. We have defined PII =9 so it is reasonable that time is 67 ms . Figure()
0x000007F3	PRM, raw precise distance in millimeters based on TW-TOF measurement which is 2035 mm

0x000007F3	CRE, raw coarse distance in millimeters based on direct path signal strength which is 2035 mm
0x000007F3	FRE, filtered distance in millimeters based on combination of PRM and CRE which is 2035 mm
0x0081	Estimated standard deviation error, in millimeters, of associated PRM values which is 129 mm
0x0081	Estimated standard deviation error, in millimeters, of associated CRE values which is 129 mm
0x0081	Estimated standard deviation error, in millimeters, of associated FREvalues which is 17 mm. FRE has the lower deviation error, as we notice.
0x0005	Estimated radial velocity in millimeters per second
0x00FB	Estimated standard deviation, in millimeters per second, of the FRV value
0x07	Range Measurement Type, specifies the valid components of this message, 1 equals to PRM, 2 equals to CRE and 3 equals to FRE
0x00	Reserved
0x0010	Requester LED Flags refers to the requester's receive scan, 1 equals to Saturated, 2equals to Scan Window too Short, 4 equals to SNR too Low, 8 equals to Line Of Sight, 16 equals to Non Line Of sight.
0x0010	Responder LED Flags, refers to the responder's receive scan
0x0004	Channel Rise, Short rise times indicates LOS channels while low rise times indicates NLOS channels
0x3825	Vpeak , a useful metric to compute SNR and CRE
0x0031D0	Coarse TOF, the range measurement before applying leading edge offsets.
0x0002D892	Timestamp, milliseconds from boot to time of range conversation competition.
0xAE29	Suffix

Table: Analysis of packet RCM_SEND_RANGE_INFO

```
import crc16
import binascii
from binascii import unhexlify

#metatroph tou message id se 2byte hex
def padhexa(s):
    return  s.zfill(4)
 #metatroph tou responder_id id se 4byte hex
def padhexa4(s):
    return  s.zfill(8)

      #convert from hex to binary
def packing(message_id,responder_id):
      #covent dec to hex
      message_id= hex(message_id)
      message_id=message_id.split("0x")
      message_id=str(message_id[1])
      #message_id='1'
      hex_messageid=padhexa(message_id)
      #responder_id='6A'
      hex_responderid=padhexa4(responder_id)
      #create binary format of command packet
      s = unhexlify('0003'+hex_messageid+hex_responderid+'00000000')
      #compute crc16
      suffix=hex(crc16.crc16xmodem(s))
      suffix=suffix.split("0x")

      #compute command in binary format
      total_command =
unhexlify('a5a5000c0003'+hex_messageid+hex_responderid+'00000000'+padhexa(
suffix[1]))

      return total_command
```

Figure: Creating the RCM_RANGE_REQUEST

3. Navigation

We are using Rover 5 as robot platform (29) (30) . It uses 4 independent dc motors, each with a hall-effect quadrature encoder and gearbox. A quadrature encodes measures the speed and direction of a rotating shaft. There are two IR sensors on the PCB that look at the black and white pattern on one of the gears. The output is two square waveforms 90 degrees out of phase. So, from the encoders we can extract the velocity of the rover as well the position of the rover.

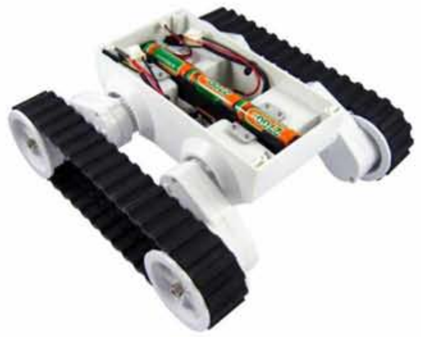


Figure 3.0.1 Rover 5

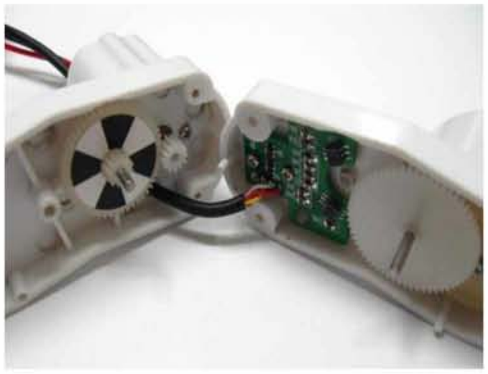


Figure 3.0.2 Quadrature Encoders

3.1 Odometry

Odometry is the use of data from motion sensors to estimate change in position over time. As we mention above the motion sensors of the Rover 5 are the quadrature encoders. The quadrature encoders of each wheel will produce 333 state changes for a full rotation of the wheel. If there is no stall, we can compute the distance that the wheel has moved. One revolution of a wheel will make it move a distance equals to its circumference. The circumference of a circle is the total distance around its outside. Circumference equals the diameter of the circle times π (pi), which is about 3.14. The diameter of Rover 5 is 6 cm, so we conclude that for each revolution the wheel has moved 18.84 cm (31). The problem is that each of our four motors reacts very differently to the same PWM, so we are trying to compute a PWM that will produce the same state changes (ticks) of each wheel. Firstly, we had examined the two front wheels and we analyzed the produced states changes. We are using an Arduino library to acquire the produced ticks (32) . Also, we are using analogWrite to set the PWM on a scale of 0-255, such that analogWrite(255) requests a 100% duty cycle and analogWrite(127) is a 50 % duty cycle. The Arduino's PWM is about at 500 Hz (33).

In our first attempt we set the PWM with the value 140 and we let the rover to direct forward. We noticed that for each 50 microseconds for both encoders the produced ticks were not always the same.

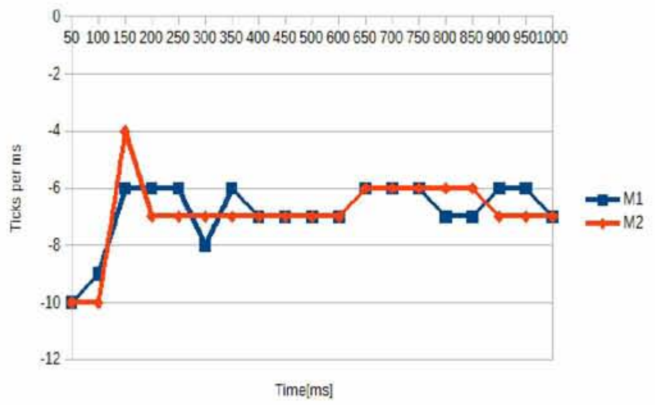


Figure 3.1.1 ***(PWM=140, Direct=Forward)***

Then we set the PWM equals to 255 which is the maximum value and we let the rover to direct forward. Now, it is obvious from the figure that the motor 1 is more powerful than the motor2 and it produces continuously more tick per 50 ms than the motor 2.

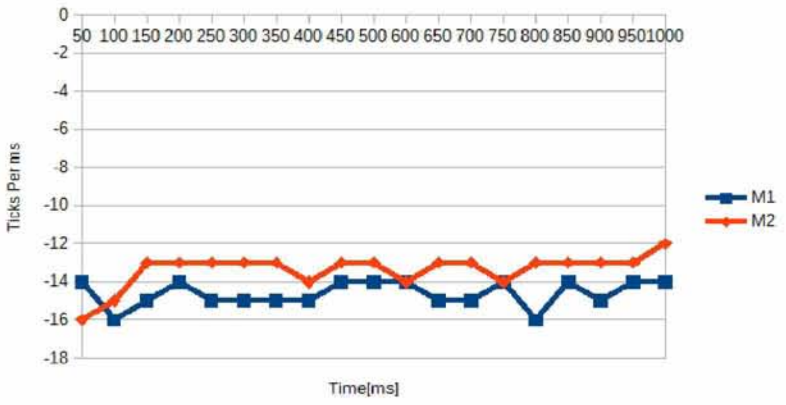


Figure 3.1.2 (PWM=255, Direct=Forward)

In the next figure we notice that in one second there is a difference of 20 ticks that means that the wheel 1 has moved, (20/333) *18.84 = 1.13 cm more than the other wheel just in one second. It is obvious does not follow a straight line, although that we had given the same PWM to both wheels.

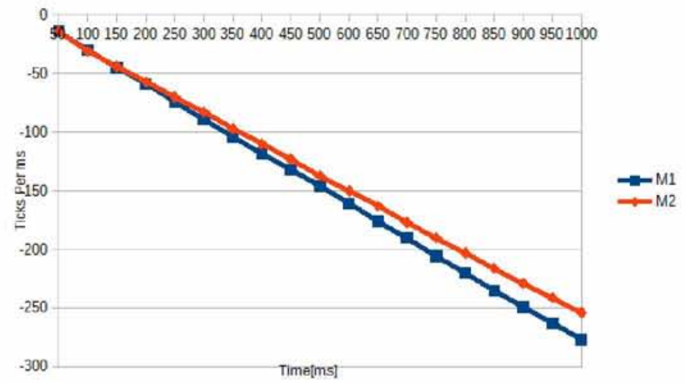


Figure 3.1.3 (PWM=255, Direct=Forward)

We conclude that independently the setting of PWM, each wheel produces uneven number of ticks which means that each will have different velocity with the same PWM. So we are going to use an algorithm that will change PWM in comparison with the produced ticks.

A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism (controller) commonly used in industrial control systems. A PID controller continuously calculates an error value as the difference between a desired set point and a measured process variable. The controller attempts to minimize the error over time by adjustment of a control variable, such as the position of a control valve, a damper, or the power supplied to a heating element, to a new value determined by a weighted sum. (34)

- *Proportional: A proportional controller is just the error signal multiplied by a constant and fed out to drive. If Kp is low the motor goes to the correct target, but it does quit slow, else if Kp is high the motor speeds up and overshoot the target*
- *Integral: Integral control is used to add long term precision to a control loop. The system takes a lot longer to settle, but when it does settle out, it settles out to the target value.*
- *Derivative: With the differential control we can stabilize the precision actuator system. The differential control is very powerful, but the most problematic of the control types.*

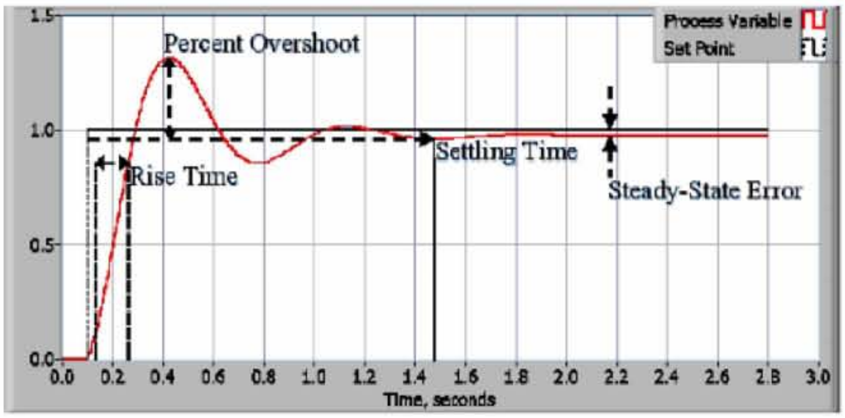


Figure 3.1.4 Example of a control system response

Initially, we had tested the algorithm in the two front wheels. We can observe that the settling time is too big for our application 48 seconds. From the other hand the produced ticks per second are almost the same in the both wheels. In 132 seconds we have 281 ticks diversion. Despite that we have minimized the error; we are not satisfied because the error is cumulative.

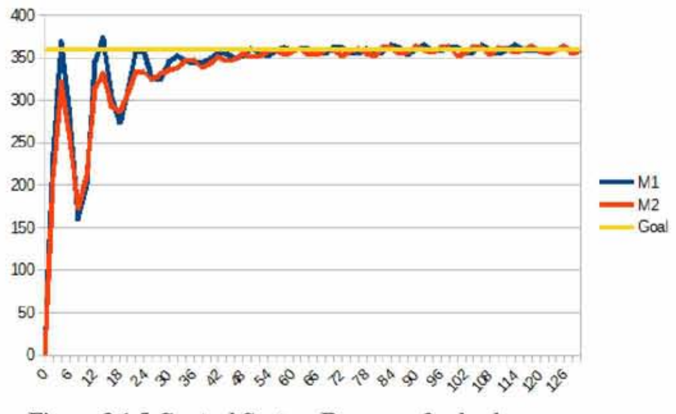


Figure 3.1.5 Control System Response 2 wheels

Then, we have tested the algorithm using 4 wheels and the target was 500 ticks per second. The settling time was clearly improved by 48 seconds to 4 seconds but there is a small difference to the produced ticks. The rover was on the air during this measurement.

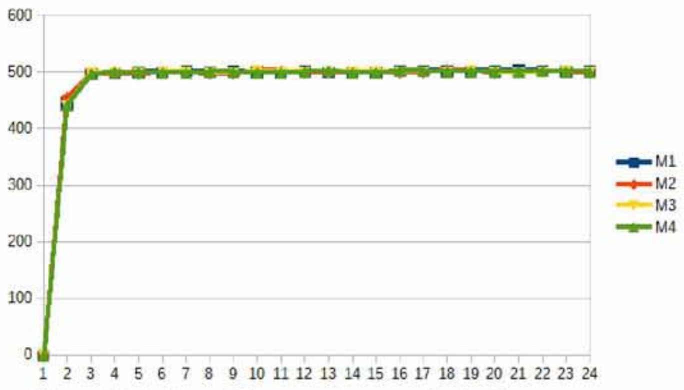


Figure 3.1.6 Control System Response 2 wheels

Finally, we let the rover on the ground. The difference between the produced ticks at the left side (motor2, motor4) and the right side (motor1, motor3) are only 55 ticks. Algorithm seems to working fine but if slightly turn at the beginning cannot correct its orientation, so we choose to use IMU to correct the velocity of each wheel in comparison with its orientation.

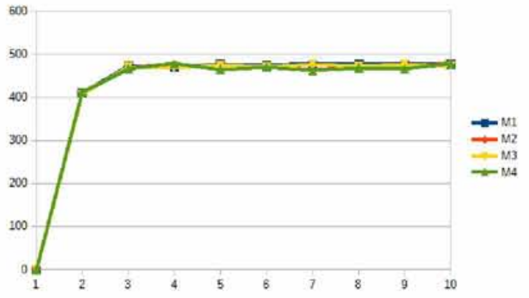


Figure 3.1.7 Produced ticks per sec

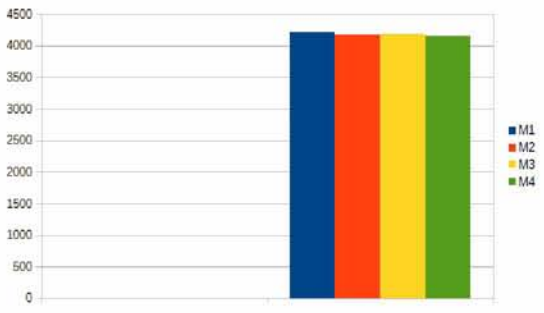


Figure 3.1.8 Total produced ticks

3.2 IMU

3.2.1 Razor IMU

Firstly, we had tried to use Razor IMU to acquire orientation information and acceleration data but that was not achieved because of the magnetic interference that is produced mainly from the dc motors of the Rover5. Razor IMU incorporates three sensors an ITG-3200 as gyroscope, an ADXL345 as accelerometer and HMC5883L as magnetometer. The output of all sensors is processed onboard by a microcontroller ATmega328 and the sensor fusion is using a Direction Cosine Matrix (DCM) (35). The most important factor to acquire valid results is the correct calibration.

1. ADXL-354

It measures the acceleration, which indicates how fast velocity is changing in the three axes, Acceleration is measured in m/s^2 and it is divided to static acceleration due to gravity and to dynamic acceleration due to motion. At the calibration of ADXL-354 we found the values of static acceleration on the three axes.

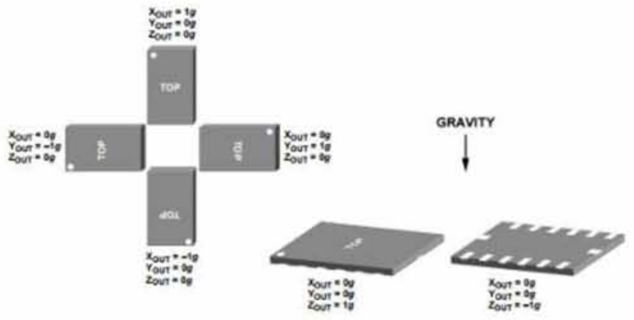


Figure 3.2.1 Methodology to find the gravity

2. ITG-3200

ITG-3200 is a gyroscope that measures rotational motion; the units of angular velocity are measured in degrees per second or revolution per second. So, angular velocity is a measurement of speed of rotation. Unfortunatelly, gyroscope is drifting all the time so we want the data from the magnetometer and accelerometer to stabilize Euler angles such as roll,pitch,yaw.

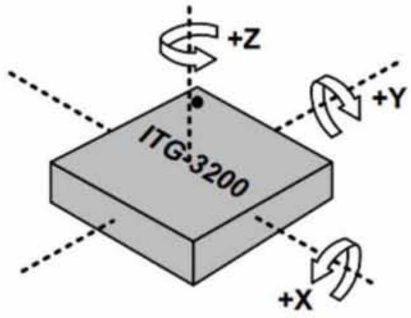


Figure 3.2.2 three axes of gyroscope

3. HMC5883L

The most challenging sensor is the magnetometer due to the magnetic interference from the dc motors. Magnetometer measures the strength and direction of the local magnetic field. The magnetic field measured will be a combination of both the earth's and magnetic field created by nearby object. Magnetic measurements will be subjected to distortion. These distortions are divided to two categories; hard or soft iron. Hard iron distortions are created by objects that produce magnetic fields such as dc motors. Soft iron distortions are considered deflections or alterations in the existing magnetic fields. These distortions will stretch or distort the magnetic field upon which direction the field acts relative to the sensor. DC motors produces magnetic fields that changes over the time so the static calibration that we are using is not capable to produce stable results.

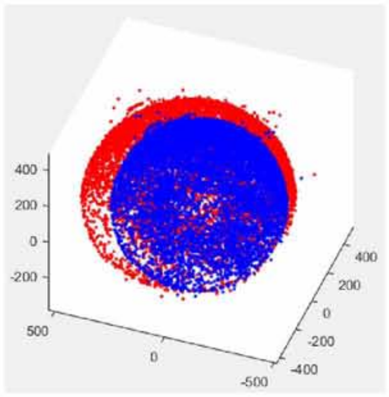


Figure 3.2.3 Red dots are the measurements; blue dots are the fixed values with static calibration.

3.2.2 9-DOF IMU Breakout - L3GD20H + LSM303

The second IMU that we used was a product from Adafruit (36); unfortunately it has the same limitations with the Razor IMU and it has slightly small differences at the results. We have tried another method for calibrating magnetometer and accelerometer and it is described in the article "Complete Triaxis Magnetometer Calibration in the Magnetic Domain" (37)but the changing magnetic fields over the time did not allow us to have accurate results.

As we explained above, the magnetometer is subjected to hard iron and soft iron distortions. Hard iron is equivalent to a bias:

$$b_{hi} = \begin{pmatrix} b_{hix} \\ b_{hiy} \\ b_{hix} \end{pmatrix}$$

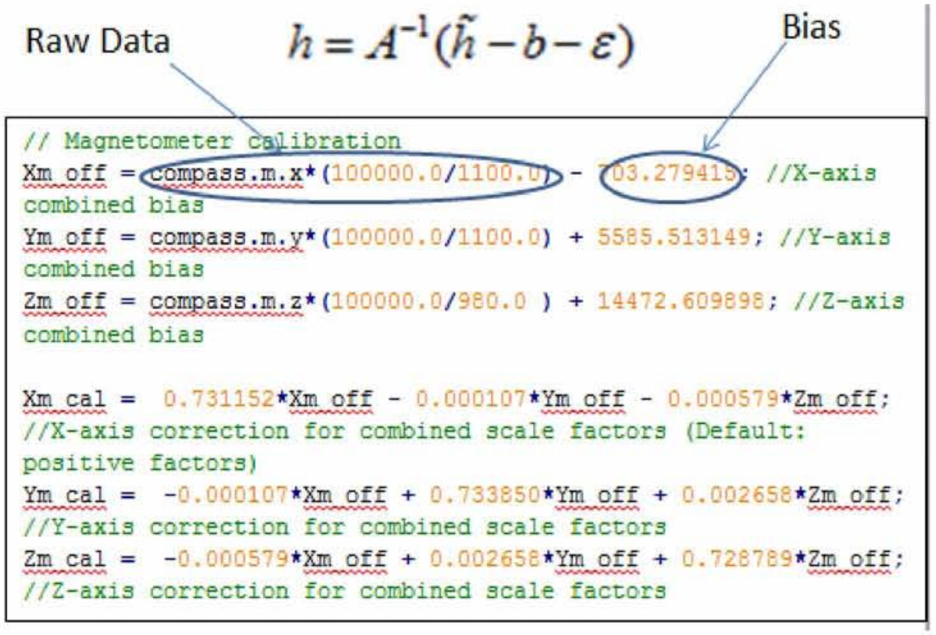
Soft iron effect can be modeled as a 3 x 3 matrix:

$$A_{si} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

So, the equation for calibrating the raw magnetic field can be written as:

$$h = A^{-1}(\tilde{h} - b - \varepsilon)$$

We have used MagCal to find the values of the bias and A^{-1} matrix. (38) (39)



```
// Magnetometer calibration
Xm_off = compass.m.x*(100000.0/1100.0) - 703.279415; //X-axis
combined bias
Ym_off = compass.m.y*(100000.0/1100.0) + 5585.513149; //Y-axis
combined bias
Zm_off = compass.m.z*(100000.0/980.0 ) + 14472.609898; //Z-axis
combined bias

Xm_cal =  0.731152*Xm_off - 0.000107*Ym_off - 0.000579*Zm_off;
//X-axis correction for combined scale factors (Default:
positive factors)
Ym_cal =  -0.000107*Xm_off + 0.733850*Ym_off + 0.002658*Zm_off;
//Y-axis correction for combined scale factors
Zm_cal =  -0.000579*Xm_off + 0.002658*Ym_off + 0.728789*Zm_off;
//Z-axis correction for combined scale factors
```

Unfortunately, we have noticed that in some turns the 9dof had deviation of ± 4 degrees so it was not perfect for our purpose.

3.2.3 MTi-3 AHRS

Mti-3 (40) is a powerful IMU which provides as with stable results for 3D orientation, 3D rate of turn, 3D accelerations and 3D magnetic field. With a roll/pitch accuracy of 1 degree RMS and yaw accuracy of 2 degrees RMC under dynamic conditions, the output is excellent for the control and the turn of our autonomous vehicle. In addition, MTi-3 runs a sensor fusion algorithm that is based on Extended Kalman filter. The sensor fusion algorithm can be processed with filter profiles onboard. We have tested various profiles that are available from the device and the best profile for our application is VRU_general. This profile is designed for heavily disturbed magnetic field. Also, yaw is determined by stabilized dead-reckoning, referred as Active Heading Stabilization, which significantly reduces heading drift.

Mti-3 is using a right-handed coordinate system as the basis of the sensor of frame. The default reference system is East-North-Up (ENU), but we can change the coordinate system to North-East-Down (NED) and to North-West-Up (NWU).

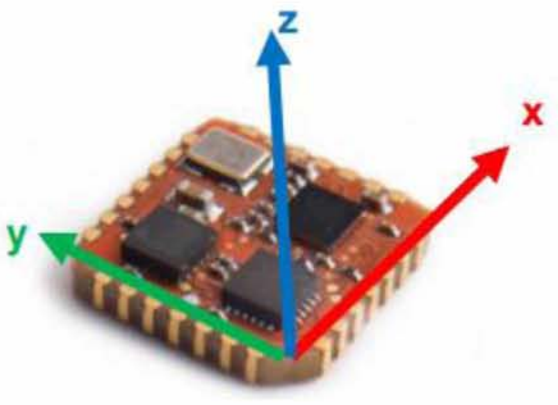


Figure 3.2.4 ENU coordinate System (40)

For the communication between the MTi-3 and beaglebone, we are using full duplex UART protocol. Because of the limited processing power of the beaglebone , we have minimize the output data rate from 100Hz to 20Hz and baud rate from 115200 to 19200. In the datasheet of the product is available the API protocol that we are using to extract the result, so we are going to illustrate the structure of the API. Although, there is a sample API code available, we used our custom python code to communicate with the MTi-3.

The MTi-3 has two states Config and Measurement State. In the Config State various settings can be read and written and in the Measurement State the MT will output its data message which contains data dependent on the current configuration. There are two different ways to enter the Config Stare or the Measurement State. First of all, at power-up the Mti-3 starts the WakeUp procedure and it will send the WakeUp message. If message WakeUpAck is send within 500 ms after the reception of the WakeUP message the MTi-3 enters the Config state, else the device enters the Measurement State. The second way to enter the Config or Measurement State is to use the GoToConfig or GoToMeasurement messages while the other state is active.

The message structure is based on a standard structure. The standard length message has a maximum of 254 data bytes, for our application the length of data message is 40 bytes as we only exploit from the IMU Euler angles and free acceleration at 3 dimension.

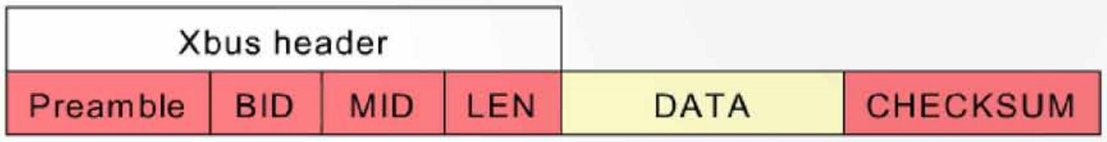


Figure 3.2.5 API Mti-3 (40)

It is important to understand the Xbus header. The preamble byte is the indicator of start of packet which is always 0xFA, BID byte is the bus identifier byte and it is 0xFF. Then it follows the message identifier (MID) byte which is 0x36 or it is 0x42 if we have an error message. Finally, the last byte of the header indicates the length of the message, in our case the length of the rest message is 0x23 without the checksum. Finally, checksum is used for error-detection, if all bytes excluding the preamble are summed and the lower byte value of the message equals to zero the message is valid.

The next step is to understand the data structure of the message. The message contains output data according to our Output Configuration. The layout of the data structure of the message is shown below:

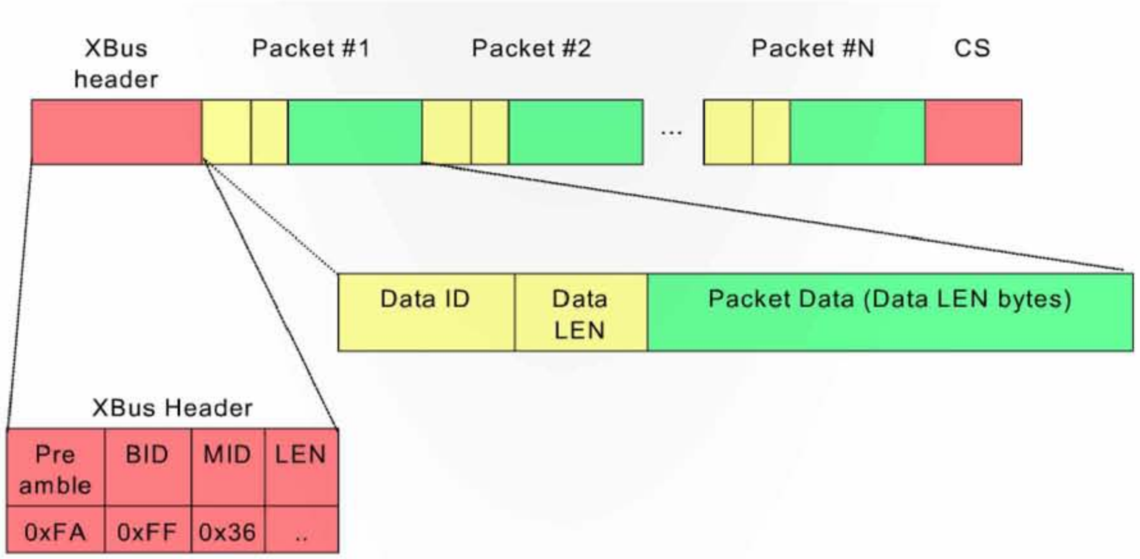


Figure 3.2.6 detailed API (40)

In our application we acquire from the IMU three different data packet. A data packet has three fields; the first two bytes contain the Data Id, the next byte shows the length of data's that follow. The output format of all the different Data Identifiers are described in the API documentation. In addition, it is important to mention that the higher byte of Data ID is the Data identifier and the lower byte shows the output frequency. The byte that shows the output frequency is separated as follows; the upper half-byte indicates the type of the data. The lower half byte shows the coordinate system that is used (East – North –Up 0x0, North-East- Down 0x4, North-West-Up 0x8) and the precision of the number (signal precision 0x0,fixed point 32-bit 0x1, fixed point 64 bit 0x2, double precision 0x3. The value is formed by doing bitwise OR of the available fields. In our application one of the Data IDs is 0x2030.

We can assume that 0x20 refers to Orientation Data, then 0x3 refers to Euler Angles and finally 0x0 indicates that we use East-North-Up system and the number is single precision.

0xfa	Preamble
0xff	BID
0x36	Message Identifier MTData2
0x23	Length of total packet
0x1020	Data ID, indicates that follows the packet counter
0x02	Data length, the value of packet counter contain two bytes
0x7bac	Value of packet counter is (31660)
0x2030	Data ID, 0x20 indicates Orientation Data, 0x3 shows Euler Angles, 0x0 indicates ENU system, single precision float
0xoc	Data length is 12 bytes ,3 dimensions * 4bytes
0xbeee685f	First 4 bytes indicates roll
0xbf99d7a0	The following 4 bytes indicates pitch
0x4102aefc	The last 4 bytes indicates yaw
0x4030	Data ID, 0x40 indicates Acceleration Data, 0x3 shows free acceleration, 0x0 indicates ENU system, single precision float
0x0c	Data length is 12 bytes, 3 dimensions * 4 bytes
0x3cb75a98	First 4 bytes shows acceleration of x
0x5c3b4a80	The following 4 bytes shows acceleration of y
0x3df36b80	The last 4 bytes shows acceleration of z
07	Checksum

Table: Analysis of a packet from Mti-3

As we described in the current Output Configuration that we are using the length of the packet is 40 bytes. So we read from the Serial the first 39 bytes that follows the Preamble (0xfa) and convert them to a hex string. We pass the hex sting as an argument to unpack function that parses the hex string and return the float value of our desired variables.

```
def convert(s):
    i = int(s, 16)                    # convert from hex to a Python int
    cp = pointer(c_int(i))            # make this into a c integer
    fp = cast(cp, POINTER(c_float))   # cast the int pointer to a float
pointer
    return fp.contents.value          # dereference the pointer, get the float

def unpack_command(hex_data):

      message_ident = hex_data[4:6]
      if(message_ident == "36"):
            roll = hex_data[24:32]
            pitch = hex_data[32:40]
            yaw = hex_data[40:48]

            #freeacceleration
            free_accx = hex_data[54:62]
            free_accy = hex_data[62:70]
            free_accz = hex_data[70:78]
            #convert from hex to float
            roll=convert(roll)
            pitch=convert(pitch)
            yaw=convert(yaw)

            #freeacceleeration

            free_accx=convert(free_accx)
            free_accx=convert(free_accx)
            free_accx=convert(free_accx)

            #convert yaw from negative to 0 to 360 degrees
            if(yaw < 0):
                yaw=180 + (180 - abs(yaw))

      elif(message_ident == "42"):
            print "Error"
            #reject  yaw with value 400
            yaw=400
      else:
            print "Problem"
            yaw=400
      return yaw
```

Table: Decompose packet of Mti-3

3.3 Adjust Velocity

We are using MTi-3 as our IMU device; from MTi-3 we acquire the orientation of the rover. We have implemented an algorithm that adjusts the velocity of the 4 wheels according to the orientation over the time. When the orientation is shifting to one side, the wheels of this side are slowing down and the wheels of the other side are speeding up until the rover go to the correct yaw. This algorithm is running continuously, while the rover is heading forward.

In our first attempt, we had given a big range of values for the velocity of the four wheels. So we noticed that the rover it was spinning out and losing control because the velocities were changing too fast.

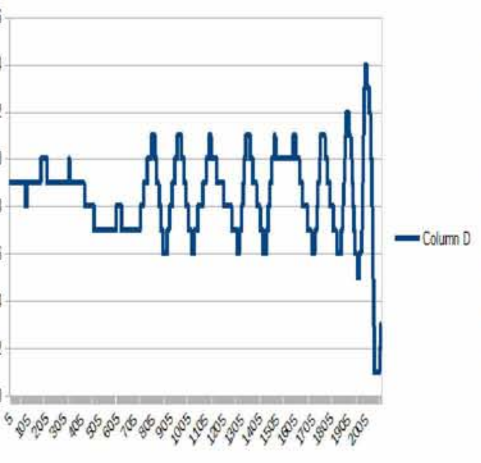


Figure 3.3.1 Shows the values of the yaw

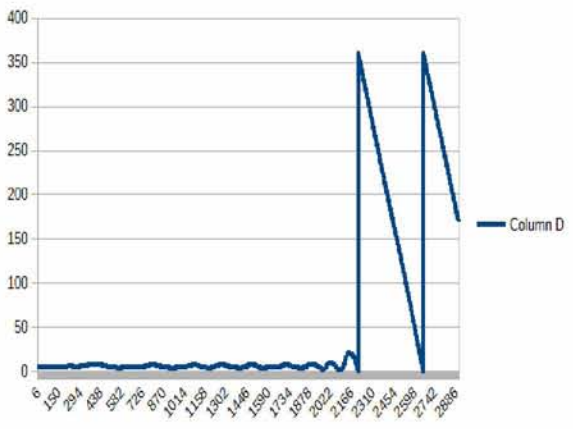


Figure 3.3.2 Rotating around itself

Finally, we solved these malfunctions and we achieved to go to a straight line with a minimum abbe error. This was achieved by restricting the available velocities for the 4 wheels. The values of PWM that were given to the left side (motor2, motor4) was from 65 to 75 and the values of PWM were given to the right side (motor1, motor3) was from 40 to 60. We picked low PWM which means low velocity because we wanted to be more precise with the localization. If the velocity was bigger we would not be able to stop the rover at a specific position. A localization measurement longs 1 second, in one second the rover travels 7 centimeters. If the rover's velocity was fourfold then the rover would be traveled 28 centimeters so it would have been more difficult to stop it in a specific position.

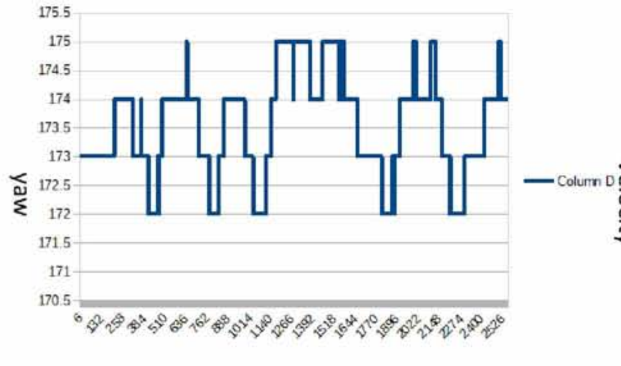


Figure 3.3.3 Shows the values of the yaw

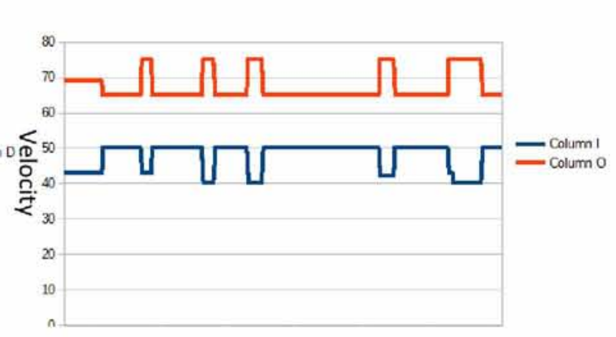


Figure 3.3.4 Velocity of motor1 & motor3 with Blue
Velocity of motor2 & motor4 Orange

4. Kalman Filter

4.1 Introduction

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. The Kalman filter is an estimator for what is called the linear-quadratic problem, which is the problem of estimating the instantaneous "state" by using measurements linearly related to the state but corrupted by white noise. The resulting estimator is statistically optimal with respect to any quadratic function of estimation error.

Kalman filter has numerous applications and specifically extended Kalman filters. Extended Kalman filter handles nonlinearity by linearizing the system at the point of current estimate and then the linear Kalman filter is used to filter this linearized system. Some application of Kalman filter are tracking objects such as airplanes missiles (41) and it used widely from radars to predict the next position of the target. In addition, it is used mainly to the most popular open source unmanned aerial vehicle Ardupilot, to estimate vehicle position, velocity and angular orientation, airspeed and barometric pressure measurements. Furthermore, there are references (42) (43) that Automatic Ground Collision Avoidance System (Auto-GCAS) that has saved the life of an unconscious F-16 pilot recently is using Kalman filters. Also, Kalman filters have application to economics and computer vision algorithms.

4.2 Basic Terms

We can express our belief in the rover's position with a Gaussian $N(x,\sigma^2)$. We believe that the rover is at 10 meters at x-axis and the variance is 1 m^2, or $N(10,1)$ (44) .

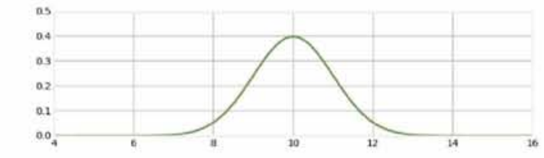


Figure 4.2.1 Gaussian $N(x,\sigma^2)$ **(44)**

The plot depicts our uncertainty about the rover's position which indicates any position from 9 meters to 11 meters is quite likely as well.

In multivariate Kalman filters it is important to take into consideration how the multiple variables are correlated and which is the covariance among them. Covariance describes how much two variables vary together; covariance is short for correlated variances. So, variance indicates how a variable vary itself and covariance indicates how much two variables change in relation to each other. A covariance of 0 indicates no correlation.

A covariance matrix looks like:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_n^2 \end{pmatrix}$$

The diagonal contains the variance for each variable, and that all off-diagonal elements contains the covariance between the covariance between the i^{th} and j^{th} variables.

4.3 Kalman filter algorithm

The kalman filter can be described at three stages (44):

<u>Initialization</u>

1. *Initialize the state of the filter*
2. *Initialize our belief in the state*

<u>Predict</u>

1. *Use process model to predict state at the next time step*
2. *Adjust belief to account for the uncertainty in prediction*

<u>Update</u>

1. *Get a measurement and associated belief about its accuracy*
2. *Compute residual between estimated state and measurement*
3. *Compute scaling factor based on whether the measurement or prediction is more accurate*
4. *Set state between the prediction and measurement based on scaling factor*
5. *Update belief in the state based on how certain we are in the measurement*

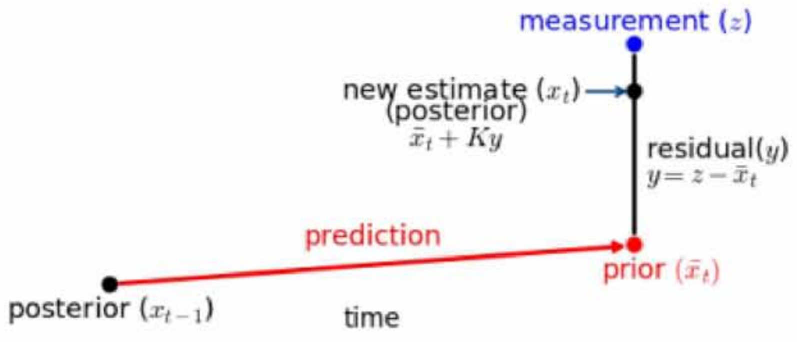


Figure 4.3.1 Kalman Filter (44)

Predict Equations

$$\bar{x} = Fx + Bu (1)$$
$$\bar{P} = FPT^{T} + Q (2)$$

Update Equations

$$y = z - H\bar{x} (3)$$
$$K = \bar{P}H^{T}(H\bar{P}H^{T} + R)^{-1} (4)$$
$$x = \bar{x} + Ky (5)$$
$$P = (I - KH)\bar{P} (6)$$

The variables are defined below:

- x, P are the state mean and covariance
- F, Q are the process mean and covariance
- B, u are the control matrix and control input
- H is the measurement function
- z, R are the measurement mean and noise covariance
- y, K are the residual and the gain

4.4 Design Kalman Filter Variables

4.4.1 Predict Variables

The first step is to design our state variables. We are tracking the rover in two dimensions and we are using the Timedomain's devices P410s to acquire measurements for these two dimensions, so we know that we have two observed variables x and y. In addition, we will incorporate velocity into our equations as well. So the state variables are

$$x = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

The next step is to design the state covariance which is:

$$P = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\dot{x}} & \sigma_{x\dot{y}} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\dot{x}} & \sigma_{y\dot{y}} \\ \sigma_{\dot{x}x} & \sigma_{\dot{x}y} & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\dot{y}} \\ \sigma_{\dot{y}x} & \sigma_{\dot{y}y} & \sigma_{\dot{y}\dot{x}} & \sigma_{\dot{y}}^2 \end{pmatrix}$$

We may choose big variance for the initial P if we are quite uncertain for the initial position of the rover. For example if our grid is 3000 cm x 3000 cm and the initial position of the rover in unknown , then the initial variance for the x-axis should be $\sigma_x^2 = (3000)^2$ and for the y-axis the initial value should be $\sigma_y^2 = (3000)^2$. From the other hand the top speed of the rover is 7 cm/sec, so we can set $(\pm 3\sigma)$, or $\sigma_{vel}^2 = 5.44$. The standard deviation is a measure of how much variation from the mean exists. For Gaussian distribution, 68% of all the data falls within one standard deviation $(\pm\sigma)$ of the mean, 95% falls within two standard deviations $(\pm 2\sigma)$ and 99.7within three standard deviations $(\pm 3\sigma)$.

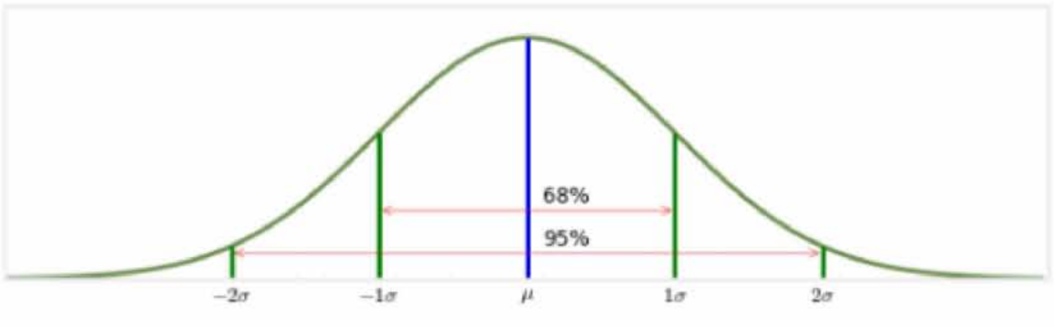


Figure 4.4.1 Standard deviation (44)

Now we are ready to design the process model and the state transition function, it is a mathematical model which describes the behavior of the system. The filter uses it to predict the state after a discrete time step. We modeled the rover's motion with

$$x = v\Delta t + x_0$$

As we know that the rover is moving with a constant velocity, which is nearly 7cm/sec. The transition function is implemented as a matrix F that we multiply with the previous state of our system to get the next state, $\overline{x} = Fx$. So, in our system the transition function must be an 4x4 matrix. The state equations are

$$\begin{aligned} x &= 1x + 0y + \Delta t\dot{x} + 0\dot{y} \\ y &= 0x + 1y + 0\dot{x} + \Delta t\dot{y} \\ v_x &= 0x + 0y + 1\dot{x} + \Delta t\dot{y} \\ v_y &= 0x + 0y + 0\dot{x} + 1\dot{y} \end{aligned}$$

And the transition function is:

$$F = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finally, we have to design the process noise matrix Q for the prediction equations. The filter adds a process noise covariance matrix Q to the covariance P. In general, the design of the Q matrix is among the most difficult aspects of kalman filter design. This is due to several factors. First, the math requires a good foundation in signal theory. Second, we are trying to model the noise in something for which we have little information. If Q is too small then the filter will be overconfident in its prediction model and will diverge from the actual solution. If Q is too large then the filter will be unduly influenced by the noise in the measurements and perform sub-optimally. In practice we spend a lot of time running experiments and evaluating collected data to try to select an appropriate value for Q. These experiments are presented in the next section. Q is a 4x4 matrix and we assume for now that it is has value 0.1 in the main diagonal.

So, matrix Q is:

$$Q = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix}$$

4.4.2 Update Variables

The measurement function H defines how we go from the state variable to the measurement using the equation $z = Hx$. In our model we have two measurements at the two dimensions x, y. So, the matrix H is:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Which correspond to these linear equations:

$$z_x = x + 0 * y + 0 * v_x + 0 * v_y$$
$$z_y = 0 * x + y + 0 * v_x + 0 * v_y$$

As we did before for the other variable we have to design the measurement noise matrix, which is a matrix that models the noise in our sensors as a covariance matrix. The variance for x and y variables was found experimentally and we considered that the noise in x is not in any way independent on the noise in y, and the noise is normally distributed about the mean.

$$R = \begin{pmatrix} \sigma_x^2 & \sigma_y \sigma_x \\ \sigma_x \sigma_y & \sigma_y^2 \end{pmatrix}$$

Now, we can understand the equation (3) of the Kalman algorithm, where y is the residual, $\bar{x}$ is the predicted value for x and z is the measurement. K in equation (4) is the Kalman gain, the ratio that chooses how far along the residual to select between the measurement and the prediction of the filter.

$$K = \frac{uncertainity_{prediction}}{uncertainity_{measurement}}$$

4.5 Adjusting the Kalman Filter

As we mentioned in the Section 2.1.1 the Anchors placement has a key role for accurate results. We have tried three different placements at indoor and outdoor environments. All the experiments were conducted on LOS conditions. The results of the localization algorithm LSE were given as input to our Kalman Filter. We are going to describe the filter's behavior and to explain the role of the different variables.

4.5.1 Outdoor Environment Random Placement

Initially, we had placed the anchors randomly and we noticed that the error from the localization algorithm especially in the y-axis was almost 100 cm instead the error in the x-axis was quite small some centimeters. We are going to examine the behavior of the filter in such big errors.

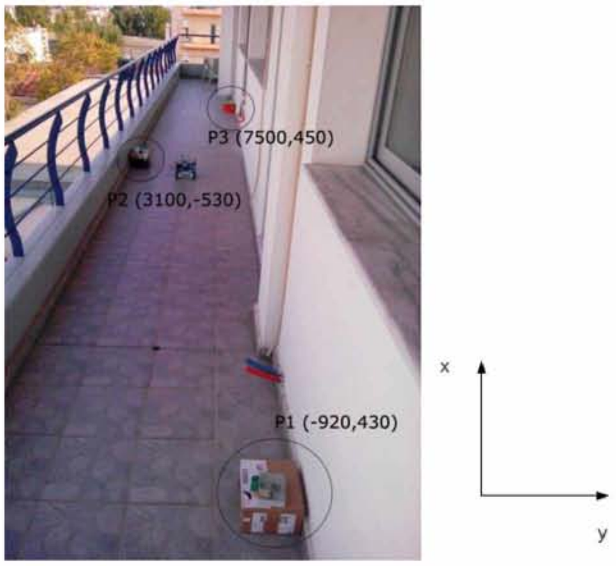


Figure 4.5.1 First attempt anchor placement

In our first attempt we were very strict about the free and we had accepted only the values with zero free. That means that too many measurements were rejected and the elapsed time for 7 valid measurement of each anchor was growing too much. The duration for 7 valid measurement of each anchor was varying from 1.4 seconds to 21 seconds. That is a noxious factor that does not allows us to stop the Robot in a specific position. In our Kalman Filter we initialized the P matrix and we had set the variance of both axes at $(2.8cm)^2$ because we were quite sure for the initial position of our Rover. The variance for the velocity was considered as $(7cm)^2$ Furthermore, for R matrix we set the variance $(5cm)^2$ from the results at x – axis and the variance $(23cm)^2$ at y-axis since the results from the localization algorithm are noisier at that axis.

$$x_{init} = \begin{bmatrix} 0 \\ 0 \\ 6 \\ 0 \end{bmatrix}, \quad P_{init} = \begin{pmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 49 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad R = \begin{pmatrix} 25 & 0 \\ 0 & 500 \end{pmatrix}, \quad Q = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix}$$

The distance that the rover had covered is 700 cm in the x-axis and then it stopped, instead of the goal that was 650 cm. So we concluded that the sample rate of the distance was very small and we had to be more tolerant with the free. With the blue line are the raw results from the localization algorithm and with the orange line are the results from the Kalman filter. We can see that the filter is smoothing the results and prevent them from unreasonable spikes. Also the filter calculates the speed of the rover 5.3 cm/s which is slower from the true speed.

From the other hand in the y axis there were more disturbances in the results from the localization algorithm so it was more difficult to find the correct values for the y variance. Firstly, the values that was chosen for the probability distribution was $(23cm)^2$. When the rover stopped the position at y-axis was -45cm. That was owed to the wrong velocity that was given to the wheels of the rover. We can see that the filter has prevented the results from spikes but if failed to estimate the correct vale at y axis. The Kalman filter indicated -51 cm at y-axis and raw measurement from the localization algorithm was – 34 cm.

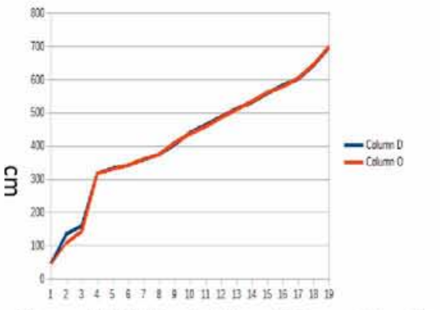


Figure 4.5.2 X-axis Raw, Kalman Results

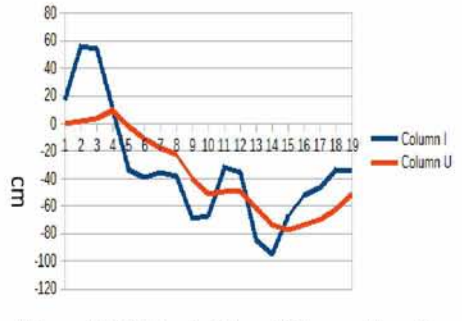


Figure 4.5.3 Y-axis Raw, Kalman Results

When the rover stopped the final computed P and x matrixes are returned. The system covariance matrix P contains the computed variance and covariance for each of the state variable. The diagonal contains the variance. The variance of returned x is $(3.4cm)^2$ a little bit bigger from the value that we had assigned. As we expected the variance of y was huge $(14.8cm)^2$ but it was smaller from ours prediction. Also, we notice that there was correlation between some variable that we failed to predict. The speed for x-axis was calculated 5.35 cm/s which was lower than the real speed.

$$P = \begin{pmatrix} 12.52 & 0 & 0.723 & 0 \\ 0 & 219 & 0 & 7.54 \\ 0.72 & 0 & 0.170 & 0 \\ 0 & 7.541 & 0 & 0.4873 \end{pmatrix}, \quad x_{final} = \begin{bmatrix} 700.2 \\ -51.5 \\ 5.35 \\ 0.191 \end{bmatrix}$$

In our second attempt, we were more tolerant with the free and we accepted the measurements that had free lower than 3. The immediate impact of this change was that the sample rate increased and the time of measurement was varying from 0.4 second to 2.1 seconds. The initial assumptions of P, x, Q, R were the same. The rover had stopped at 654 cm at the x-axis and at the -7 cm at y-axis. As we can see from the figures the results from the localization algorithm are high varying because we accepted almost all the measurements between the devices. The Kalman filter is smoothing the results from the localization algorithm and it stops the Rover in the target position due to higher sample rate.

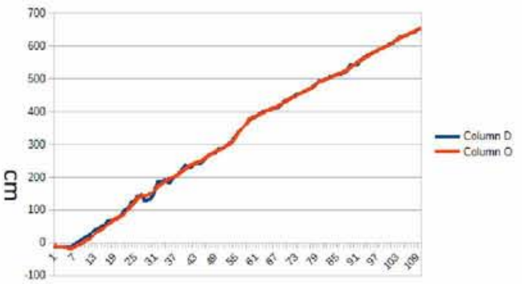


Figure 4.5.4X-axis Raw, Kalman Results

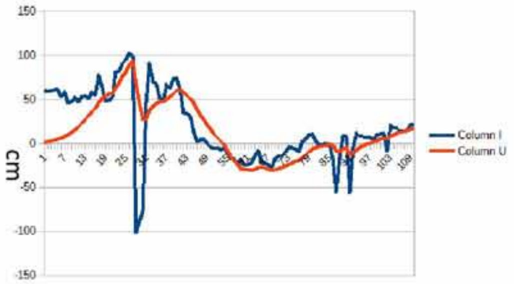


Figure 4.5.5 Y-axis Raw, Kalman Results

This time the Rover stopped at the targeted position at x-axis. But it failed to predict correct its position at the y-axis. As we observe the variance at both axes has decreased, especially in the y- axis there is a significant drop. In addition, the filter predicted correct the velocity of the Rover (7cm/sec) which is quite accurate with the real velocity of the Rover. The returned matrices for P, x are:

$$P = \begin{pmatrix} 8.85 & 0 & 1.66 & 0 \\ 0 & 66 & 0 & 6.39 \\ 1.65 & 0 & 0.75 & 0 \\ 0 & 6.4 & 0 & 1.38 \end{pmatrix}, \quad x_{final} = \begin{bmatrix} 654 \\ 17.7 \\ 7.12 \\ 1.41 \end{bmatrix}$$

Then we were testing the behavior of our filter for various values of Q. Especially, we wanted to predict a correct value for the position of the Rover at y-axis. Firstly, we had set 1 in the main diagonal of the matrix. Then we set the main diagonal of Q with the following values: 0.01, 0.001.

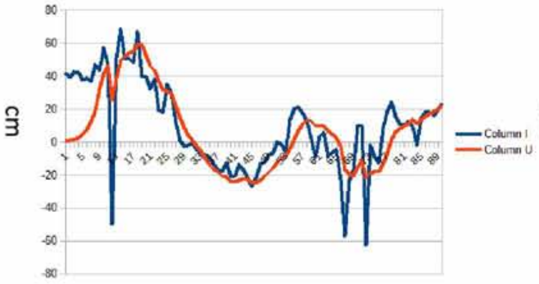


Figure 4.5.6 Q=1, $\sigma_y^2 = 179$

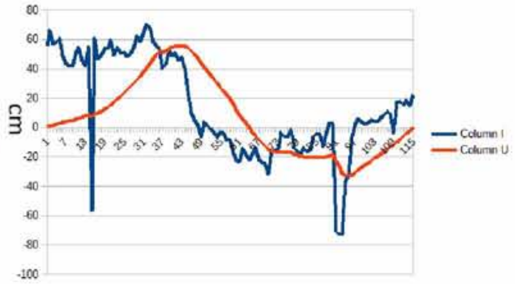


Figure 4.5.7 Q=0.01, $\sigma_y^2 = 61$

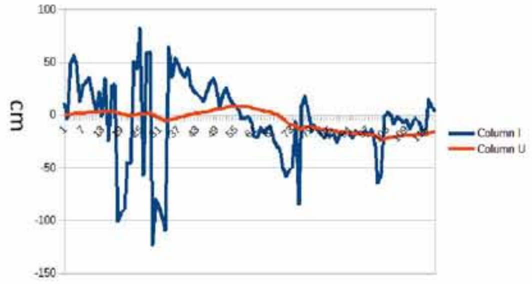


Figure 4.5.8 Q=0.001, $\sigma_y^2 = 33$

When the Q equals to 1 the filter adopts the measurements from the localization algorithm and it is very sensitive to the changes. The Rover stopped at -21 cm at the y-axis and the filter returned that the Rover was at 8 cm the error was 29 cm. Consequently, we change the value of Q to 0.01 which makes the filter more skeptical with the measurements and more confident about itself. The Rover stopped at -3 cm at y-axis and the filter returned that the Rover was at 0 cm. This value of Q had the best performance and the error was only 3 cm. Also it had the lowest variance in the y-axis at P matrix. The last value that was given to the Q was 0.001 and the filter was over confident to its model and it failed to adjust its prediction with the results from the localization algorithm. The Rover stopped at 4 cm but the filter returned -15 cm, the error was 19 cm.

Later, we changed the coordinates of the anchors in an attempt to minimize the error in the y-axis. Despite of this change, the error was the same in the y-axis. The new coordinates of the anchors are illustrated in following picture:

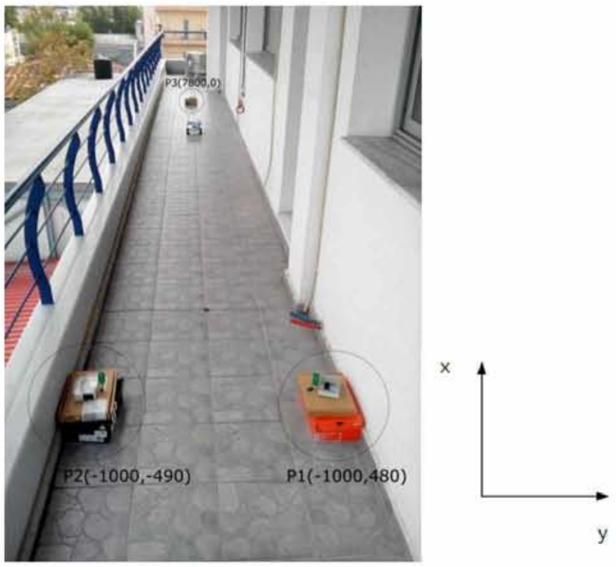
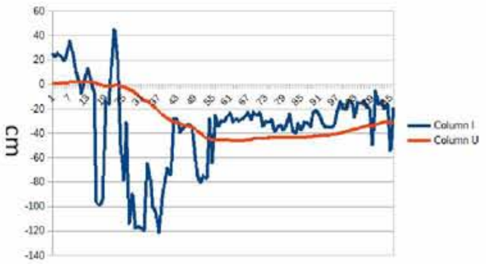


Figure 4.5.9 New coordinates of the anchors in mm

We conducted some experiments changing the values of the Q and R matrices. Firstly, we set Q equals to 0.0001 and then we considered that the variance of the error from the localization algorithm at y-axis was 100000 cm^2 in an attempt to make the filter very cautious with the measurements at y-axis.



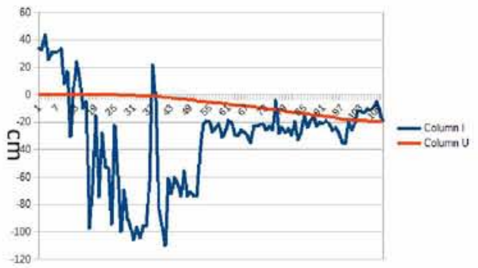


Figure 4.5.10 Q=0.0001 $\sigma_y^2 = 32$ **Figure 4.5.11 Q=0.1 Ry=100000,** $\sigma_y^2 = 1785$

In the first occasion the Rover stopped at +4cm at the y-axis and the filter returned -30 cm, the error was 34 cm. We can see that a very small value of Q prevents the filter to adopt the values from the localization algorithm but it fails to predict the correct result. From the other hand increasing the uncertainty about the input measurement at y-axis it makes the filter stricter and its result is better. The Rover stopped at -7 cm at the y-axis and the filter returned -20 cm. The error was only 13 cm but the uncertainty of the filter of this result is huge ($\sigma^2 = 1785$) so we are not satisfied from the behavior of this filter.

4.5.2 Indoor Environment Equilateral Placement

Finally, we changed the placement of the three anchors according to the requirements of section 2.2.1.1. The new coordinates of the anchors are anhor1 (-2300, 0), anchor2 (0, 2300) and anchor3 at the top of the triangle (0, 3984). With the new coordinates the variance at the y –axis was quite small, only some centimeters

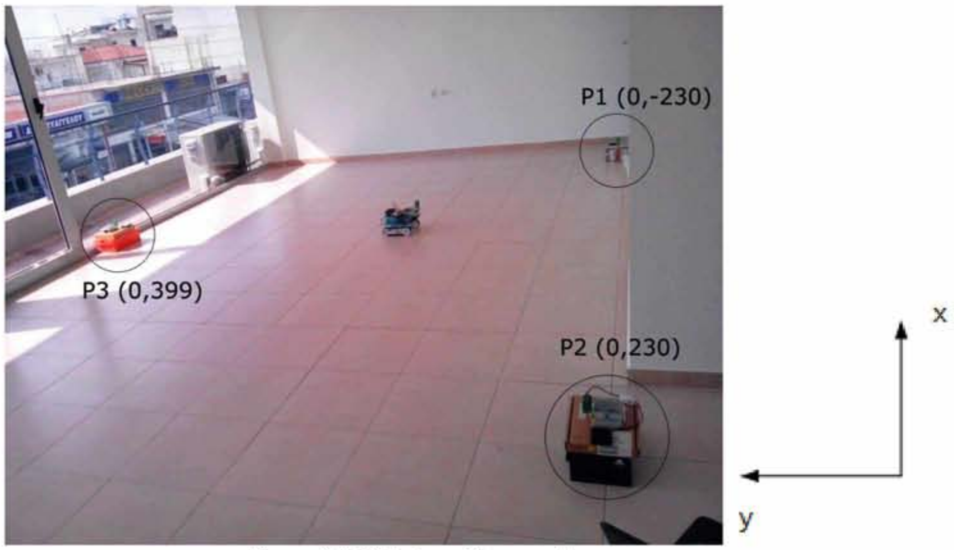


Figure 4.5.11 Indoor Placement

We set Q equals to 0.1 and in the R matrix was given the values 16 and 100 to the x-axis measurements and to the y-axis measurements accordingly. The Rover stopped at (304, 4) cm and the filter returned (304, 5) .That means that the filter as the measurements had an exceptional behavior and the precision was almost perfect. The variance at the y-axis has dropped significantly from 100 cm to 8 cm in comparison with the previous placement. As we explained above the coordinates of the anchors have a huge impact to the results of the trilateration algorithm.

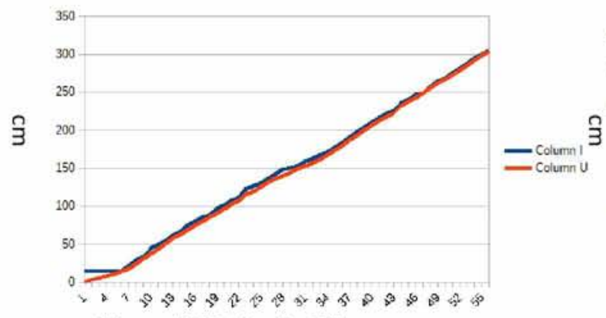


Figure 4.5.12 X-axis, 304 cm

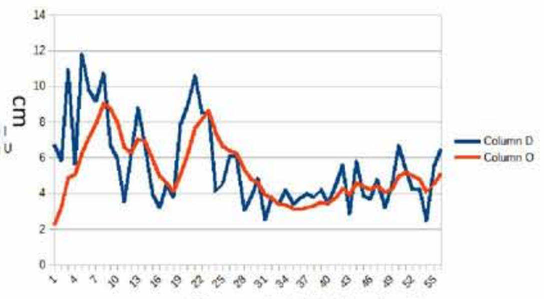


Figure 4.5.13 Y-axis, 5cm

In addition, the initial matrices (P, x) have a key role for the behavior of our filter. Firstly, we set the Rover's position at (0,0) and we set $\sigma^2 = 8$ in the P matrix because we were quite sure about the Rover's position. Although, if the initial Rover's position is different from our initial prediction the filter behavior's at the beginning will not be ideal. But after some time the filter's results and measurements will be tuned. Furthermore, it is obvious that tuning time of y-axis is almost double from the tuning time of x-axis. This is caused form the fact the uncertainty about y-measurements is quite bigger from the uncertainty about x-measurements. So the filter does not respond immediately to the y-measurements as it has more belief to its process model.

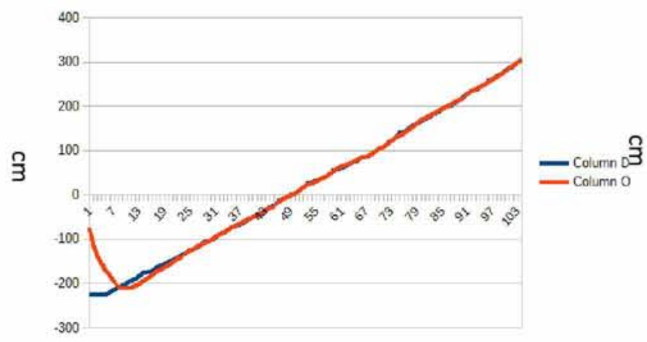


Figure 4.5.14 X-axis, small initial P

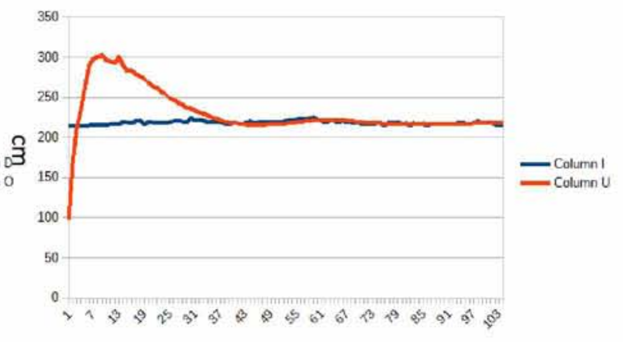


Figure 4.5.15 Y-axis, small initial P

The previous problem can be solved very easily increasing the initial variance of the Rover's position in both axes. In the next experiment we set the variance for both axes at 160000.Now, the filter adjusting its prediction with the initial measurements almost immediately.

$$P_{init} = \begin{pmatrix} 16000 & 0 & 0 & 0 \\ 0 & 16000 & 0 & 0 \\ 0 & 0 & 49 & 0 \\ 0 & 0 & 0 & 49 \end{pmatrix}$$

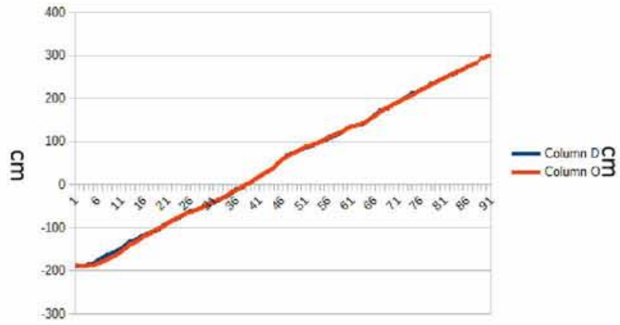


Figure 4.5.116 X-axis, huge initial P

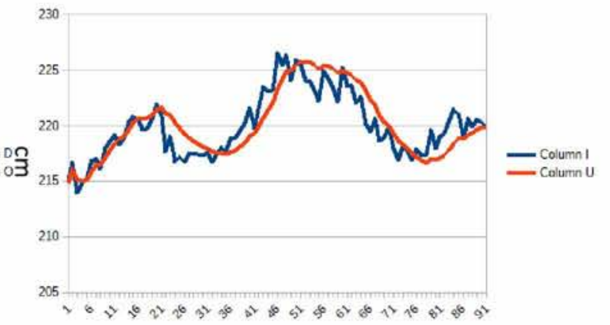


Figure 4.5.17 Y-axis, huge initial P

5. Design and Implementation

5.1 Beaglebone Black

The Beaglebone is a powerful and affordable Linux computer that is perfect for embedded applications such as robotics, home automation e.t.c. It is designed BeagleBoard.org and it is fully open source. It is based on Texans Instruments AM335x 1GHz ARM Cortex-A8 series of microprocessors and can run a number of different operating system. In our case the Beaglebone has a debian/Linux distribution. Also, it has flash memory 4 GB eMMC and 512 MB RAM. The Beaglebone black includes 2 x 23 pins rows of female's header pins. It supports GPIO (0-3.3V), ADC, PWM, UARTs, SPI and I2C.

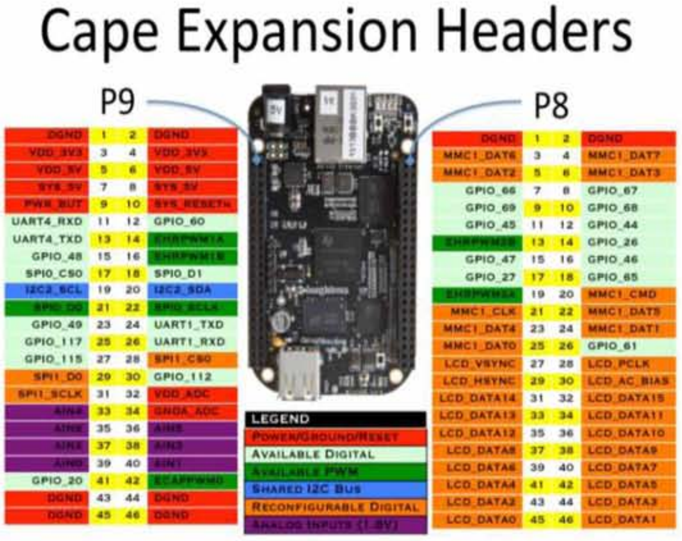


Figure 5. 1.1 I/O pins map

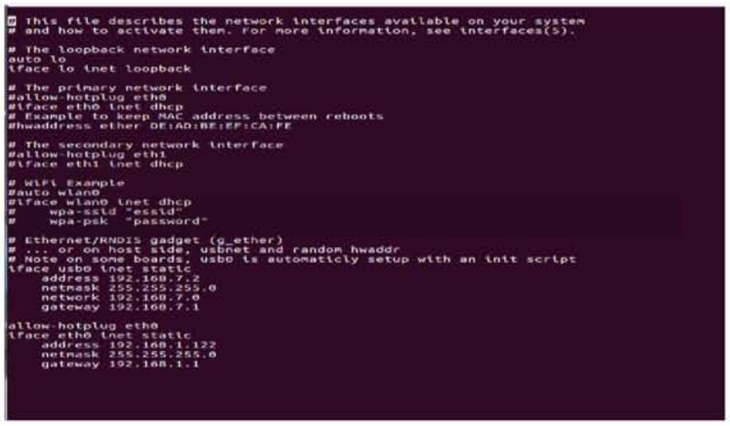
The BeagleBone P9 header has two different 5 V supplies. P9.5 and P9.6 are connected directly to the DC barrel jack, and P9.7 and P9.8 are connected to the output of the BeagleBone's on-board voltage regulator. If you are powering your BeagleBone from the DC barrel jack, it is best to use P9.5 and P9.6 as you will be able to draw more current. If you are powering the BeagleBone through the USB jack you will have to use P9.7 and P9.8 instead, in which case your connected devices cannot draw more than 250 mA.

5.1.1 Static IP

Firstly, we set the static IP so we would be able to download all the necessary packets from the internet directly. Then it is easy to connect with the Beaglebone with ssh at the static IP. The procedure was easy, we go to the file /etc/network/interfaces and we specify the address (e.g., 192.168.1.122 in this case), the network mask and the network gate way.

We add:

- *allow-hotplug eth0*
- *iface eth0 inet static*
- *adress 192.68.1.122*



```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug eth0
#iface eth0 inet dhcp
# Example to keep MAC address between reboots
#hwaddress ether DE:AD:BE:EF:CA:FE

# The secondary network interface
#allow-hotplug eth1
#iface eth1 inet dhcp

# WiFi Example
#auto wlan0
#iface wlan0 inet dhcp
#     wpa-ssid "essid"
#     wpa-psk  "password"

# Ethernet/RNDIS gadget (g_ether)
# ... or on host side, usbnet and random hwaddr
# Note on some boards, usb0 is automaticly setup with an init script
iface usb0 inet static
    address 192.168.7.2
    netmask 255.255.255.0
    network 192.168.7.0
    gateway 192.168.7.1

allow-hotplug eth0
iface eth0 inet static
    address 192.168.1.122
    netmask 255.255.255.0
    gateway 192.168.1.1
```

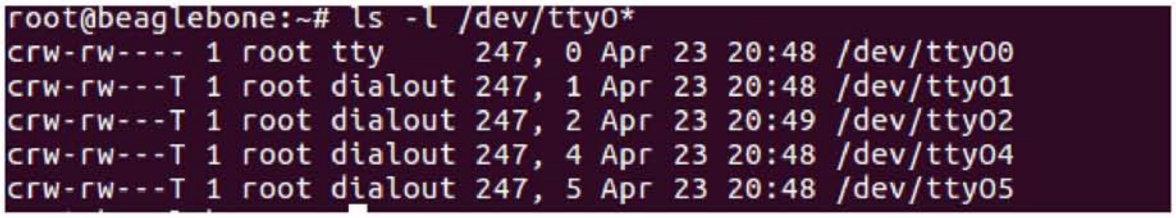
Figure 5.1.2 Interfaces file

5.1.2 Enable UARTs

On the BeagleBone Black, it's only the /dev/ttyO0 that is enabled by default, and it is coupled to the serial console. The other serial ports must be enabled before they can be used. In addition, UART0 (dev/ttyO0) is reserved for serial console and we have to disable the hdmi first and then we enable UART5. We have set UART1 for the RCM, UART5 is connected with the Xbee. Finally, we have connected the MTi-3 with the UART4.

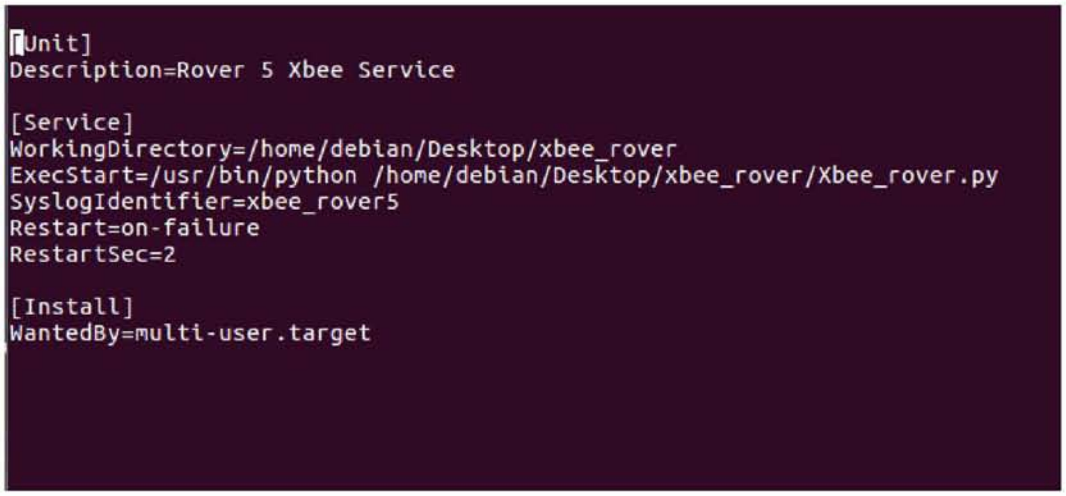
The procedure to enable the UARTs is very simple. We open /boot/uboot/uEnv.txt and add the following command:

optargs=capemgr.enable_partno=BB-UART1,BB-UART2,UART4,UART5


```
root@beaglebone:~# ls -l /dev/ttyO*
crw-rw---- 1 root tty     247, 0 Apr 23 20:48 /dev/ttyO0
crw-rw---T 1 root dialout 247, 1 Apr 23 20:48 /dev/ttyO1
crw-rw---T 1 root dialout 247, 2 Apr 23 20:49 /dev/ttyO2
crw-rw---T 1 root dialout 247, 4 Apr 23 20:48 /dev/ttyO4
crw-rw---T 1 root dialout 247, 5 Apr 23 20:48 /dev/ttyO5
```

5.1.3 System Service

We have to set up a service to run a python script automatically when the beaglebone black starts. So, we control the Rover5 remotely via the Xbee devices. The fundamental purpose of an init system is to initialize the components that must be started after the Linux kernel is booted. The init system is also used to manage services and daemons for the server at any point while the system is running. With that in mind, we will start with some simple service management operations. In systemd, the target of most actions are "units", which are resources that systemd knows how to manage. Units are categorized by the type of resource they represent and they are defined with files known as unit files. The type of each unit can be inferred from the suffix on the end of the file. (45)


```
[Unit]
Description=Rover 5 Xbee Service

[Service]
WorkingDirectory=/home/debian/Desktop/xbee_rover
ExecStart=/usr/bin/python /home/debian/Desktop/xbee_rover/Xbee_rover.py
SyslogIdentifier=xbee_rover5
Restart=on-failure
RestartSec=2

[Install]
WantedBy=multi-user.target
```

Figure 5.1.3 Go to the path /lib/system/system and we create the file xbee_rover5.service

5.1.4 Beaglebone shield

For the needs of this project we have created a shield which provides us only with pins that we need. Also, it provides us with more stable power supply for our IMU MTi-3.

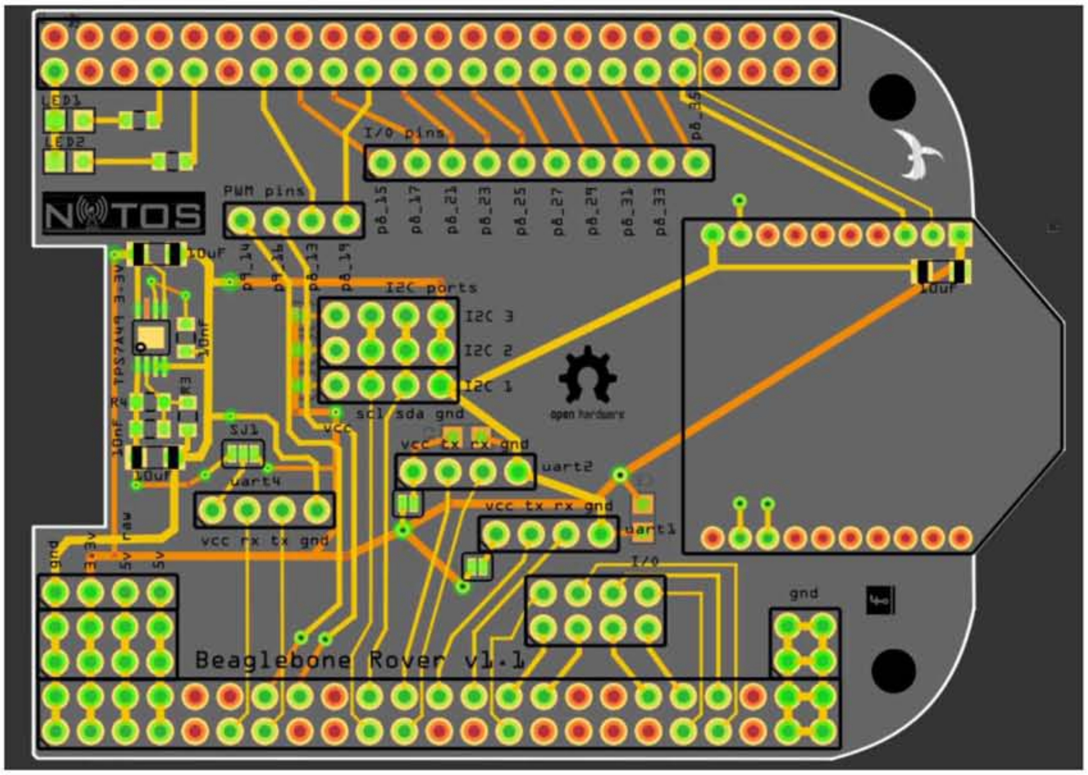


Figure 5.1.4 Shield of Beaglebone

5.2 System Design

We depict the basic architecture of our system. The central component is the beaglebone black which is the brain of our application. The IMU provides the heading information and manage the velocity of the Rover5. The RCM provides the ranging information that means the distance of the Rover5 from the three anchors. The Rover 5 is our vehicle and the Xbee allow us to have bidirectional communication from the HOST to the beaglebone.

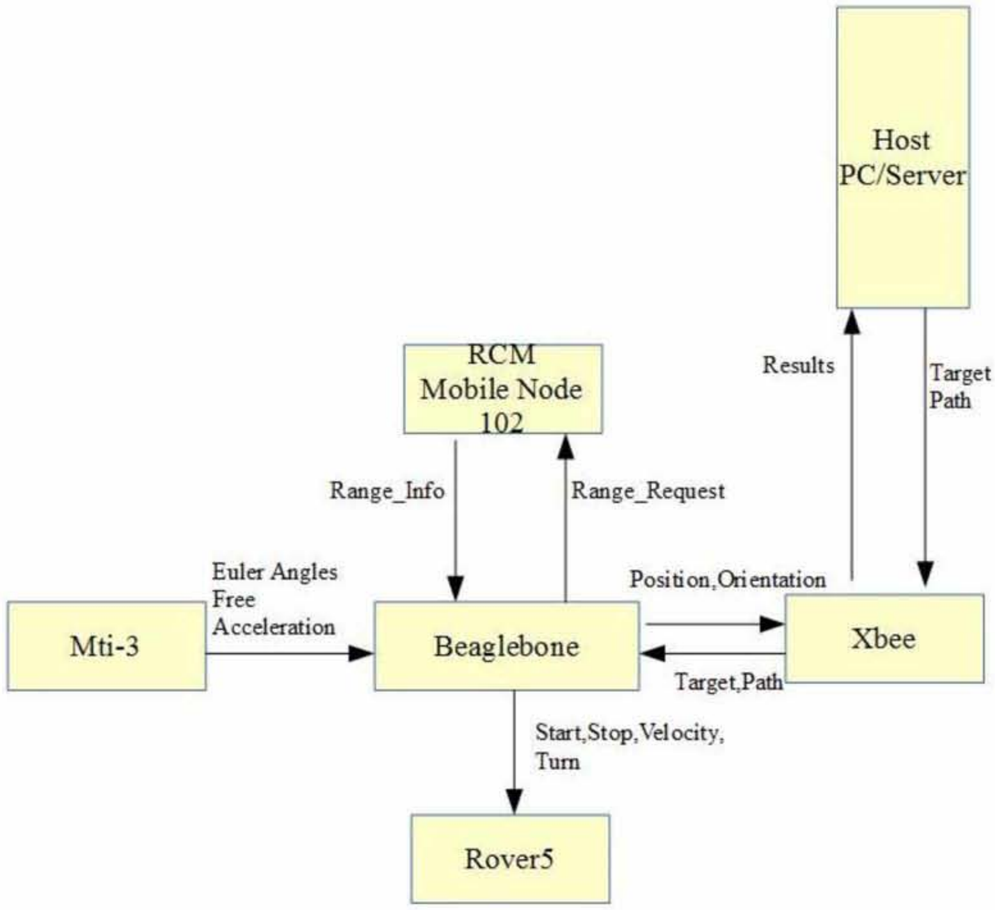


Figure 5.2.1 System Overview

- MTi-3: Provides us with Euler Angles and free acceleration. Euler's angles and especially yaw is used for the correction of the trajectory of the Rover5. Free acceleration is planned to be used as an input to the control matrix of a Kalman Filter.
- RCM: RCM(102) which is attached to the Rover5 provide us with information about the distances between the Rover5 and the three achors.
- Xbee: The Xbee which is connected to the Beaglebone communicates with an Xbee which is connected to the Server/PC. It has input the target path and it returns the position of the Rover5 and its orientation.
- Beaglebone: It is the Brain of the system. Firstly, it takes as input from Xbee the targeted path. It stores it and begins the communication with the RCM to define its position. The RCM returns the three distances and the Beaglebone runs the localization algorithm to determine its position. Then the Rover5 starts and the

beaglebone simultaneously has communication with the IMU and the RCM creating two threads. It communicates with the IMU to recalculate the velocity of its wheel. From, the other hand it communicates with the RCM to get continuously measurements and run the localization algorithm. The results from the localization algorithm are inputs to our Kalman Filter. When the Rover5 reach to its desired position the beaglebone stops the Rover5 and it terminates these two threads. Then the Rover5 turns to the given degree and continuous its desired path.

The placement of the anchors at the testbed was the following:

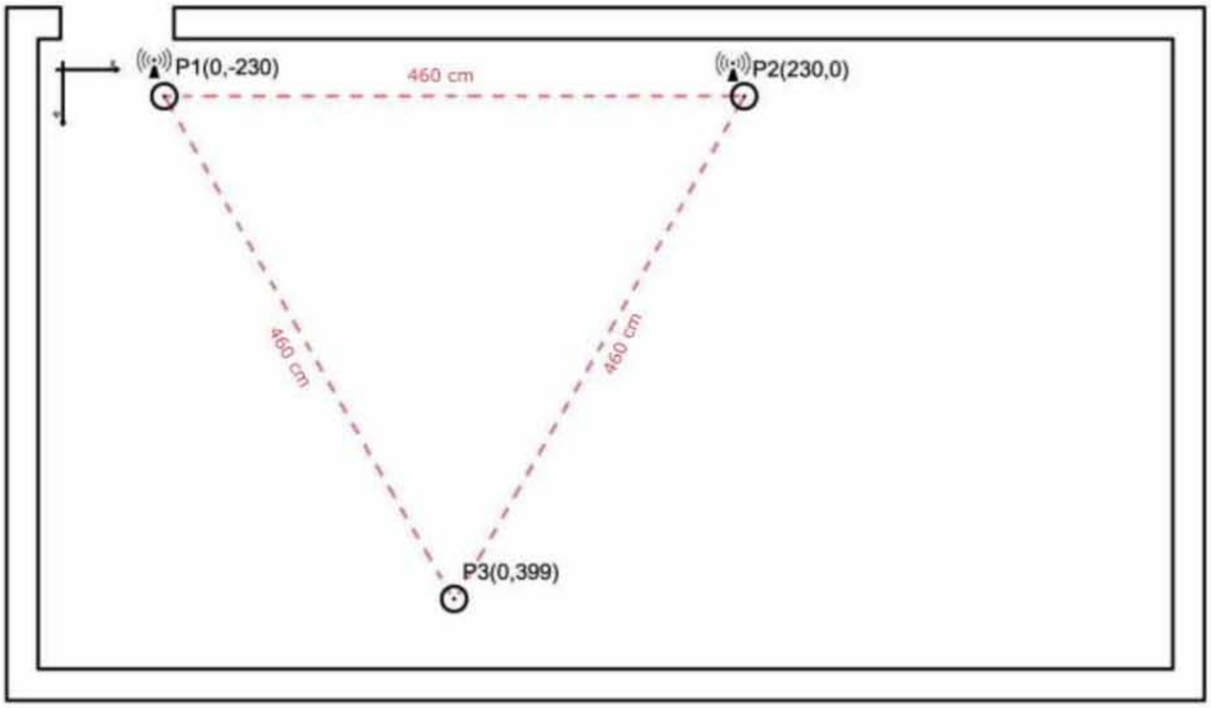


Figure 5.2.2 Floor Plan

As we explained above this is the best placement for three anchors, we intentionally set negative values to the coordinates of the anchors so the Rover will be able to handle every situation and we tested the code for every possible condition.

Algorithm for Autonomous Robot

```
1: Wait for targeted_path
2: Path_Commands=decompose (targeted_path)
3: For Action in Path_Commands:
4:      if(Action == Forward)
5:           current_postion= measure_coordinates() //the current position of the rover
6:           while (current_position != target_position)
7:                recalculate_velocities(IMU) // two threads running concurrently
                  current_position = Kalman_coordinates()
8:      else if (Action == Turn)
9:                turn(IMU)
10: End_of_Path
```

Initially, we are waiting from the Host PC/ Server the targeted path which is transmitted by the Xbee. When the targeted path arrives, the Beaglebone decomposes the path according to the direction (turn, forward) and according to the value (turn 90 degrees, go forward 100 centimeters). Then, it begins to execute the command sequentially.

If the Action is forward then the beaglebone sends RCM_RANGE to the three anchors and when it receives the responses it uses the localization algorithm to calculate its coordinates. Then, it creates two threads; the first thread recalculates the values of the velocities of each wheel continuously. The input of this thread is the yaw which is acquired from the MTi-3. The second thread is implementing a Kalman filter which calculates the coordinates of the Rover. The attached RCM sends one request on each anchor and returns the three estimated distances, the localization algorithm produces the estimated coordinates which are the inputs of the Kalman filter. When the coordinates of the rover are similar with the target, the Rover stops as well the two threads.

If the Action is to turn (left or right) the Rover is turning until it reaches the desired degree. The input of this function is again the yaw from the MTi3. When the whole path is executed the loop (3) is terminated and the Rover has executed its mission.

5.3 Results and GUI

Finally, we have created a GUI which illustrates the targeted path which is given by the User and also illustrates the calculated path from the Kalman Filter. In addition, contains in Real-time the coordinates of the Rover and its orientation. The final appearance of the Rover is the following, we have created a base for the P410 and for the IMU and we have integrated the shield of the Beaglebone.

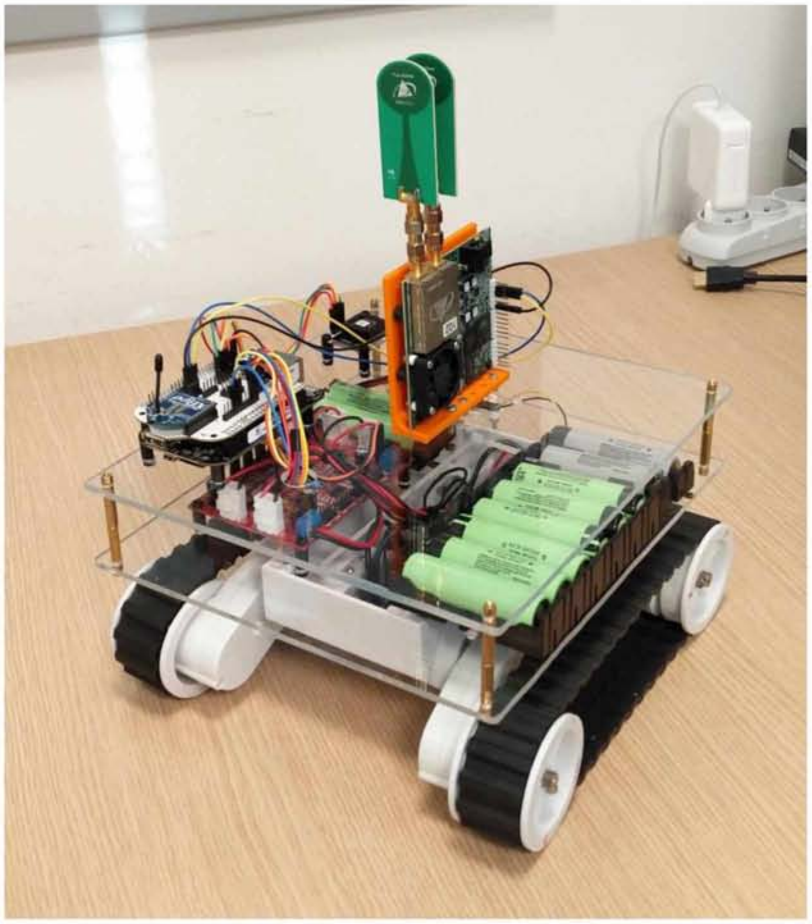
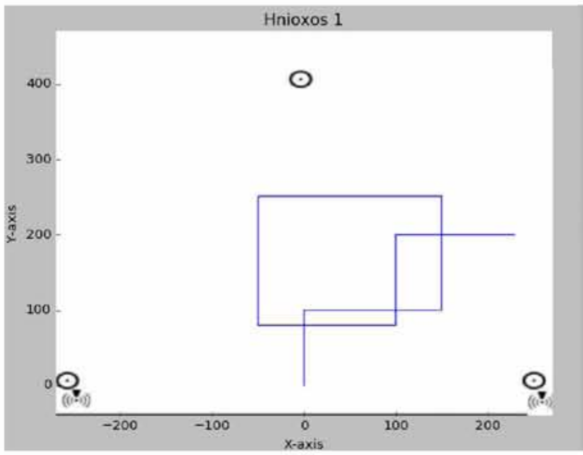


Figure 5.3.1 Final Shape of the Rover

Firstly, we illustrate the path that is calculated by the Kalman Filter. As we can see from the Figure 5.3.3, the Kalman Filter has exceptional results and it is quite accurate to the targeted path. The error when it stops varies from 0 to 4 centimeters this is due to the fact that the duration of the measurements is almost 0.8 second. That has as result the Rover will travel 6 centimeters more from the last calculated measurements.



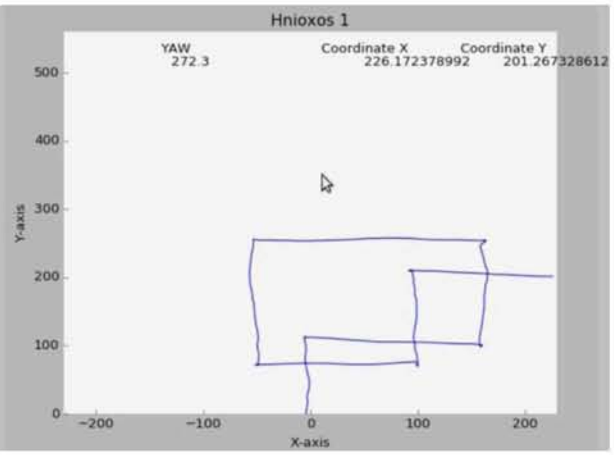
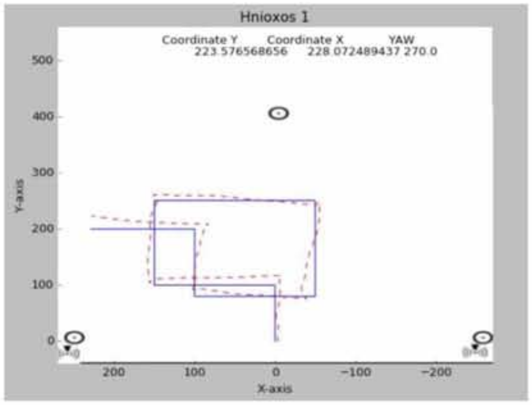


Figure 5.3.2 Given path from the User **Figure 5.3.3 Calculated path from the Kalman filter**

Then, we compare the targeted path with the calculated path from the Kalman filter. With the blue line is the targeted path and with the red line is the calculated path. We observe that the Rover stops exactly in the target but it has difficulty to follow a straight line. It is very important to have a very good initial position. In the Figure 5.3.4 the initial position is not very accurate and deviates some degrees to right. In the Figure 5.3.5 the initial position is more accurate so the Rover is following the targeted path more accurate. In both occasions the Rover executes the path with success but in the second it executes the path with the minimum error.



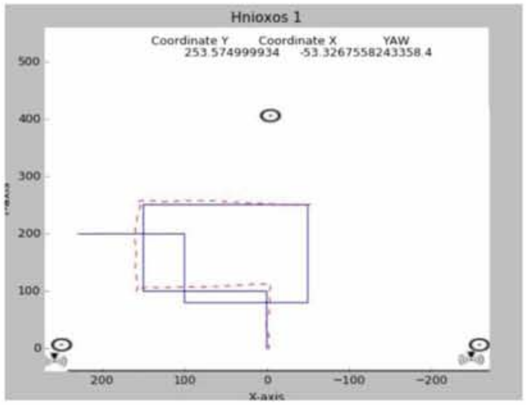


Figure 5.3.4 Blue line targeted path, red line calculated path

Figure 5.3.5 Better initial position, more accurate results.

6. Conclusion

6.1 Summary

The most challenging part of this project was the Rover 5, we have tried two different methods to adjust the velocities of its wheel. Finally, Rover was moving in a straight line but with a large computational power. Furthermore, we have some powerful devices as P410s for ranging which provide us with extremely accurate results ± 2 centimeters in combination with least square algorithm for localization which minimizes the error, we succeed to have huge accuracy ± 1 centimeter for the localization of the Rover. Another tricky part of this project was the IMU; we have tested 3 different IMUs, only the MTi-3 has the desirable performance 1 degree error. Finally, we have succeeded to build an indoor navigation system which provides us with accuracy 1 cm and the Rover successfully completes a mission which is given by the User. This navigation system can be installed in every indoor location as the coordinates of the anchors can be changed.

6.2 Future Work

The next step of this project is the replacement of the Rover 5, we are going to use a Rover with stepper motors. In addition, MTi-3 provides us with free acceleration and with the rate of turn which can be used for the implementation of an Extended Kalman Filter. Moreover, we are planning to create a GUI with a map from the NITLab's testbed in which the User will draw the desired path. Finally, we are examining cheaper solution for ranging devices such as DWM 100 (6) and ultrasound devices (46).

Bibliography

1. **Yavari, Mohammadreza.** Indoor Real-Time Positioning Using. [Online] https://www.cs.unb.ca/tech-reports/documents/TR15-236.pdf.

2. **Banerjee, Salil P.** Improving Accuracy in Ultra-Wideband Indoor. [Online] http://cecas.clemson.edu/~ahoover/theses/banerjee-diss.pdf.

3. *An Algebraic Solution to the Multilateration Problem.* **Norrdine, Abdelmoumen.** 2012.

4. *Accurate Distance Measurement using the TIMEDOMAIN R□P410 UWB Radio.* **Zakaria Kasmi, Abdelmoumen Norrdine, Arndt Sieprath, J¨org Blankenbach.** 2013.

5. **http://www.timedomain.com/.** [Online]

6. Overview of DWM1000 Module . [Online] http://www.decawave.com/sites/default/files/product-pdf/dwm1000-product-brief.pdf.

7. *Harmonium: Asymmetric, Bandstitched UWB.* **Benjamin Kempke, Pat Pannuto, and Prabal Dutta.** 2016 : s.n.

8. *PolyPoint: Guiding Indoor Quadrotors with Ultra-Wideband.* **Benjamin Kempke, Pat Pannuto, and Prabal Dutta.** 2015.

9. https://github.com/thotro/arduino-dw1000 . [Online]

10. https://github.com/thotro/arduino-dw1000/issues/62. [Online]

11. *Zero-Configuration, Robust Indoor Localization:.* **Hyuk Lim, Lu-Chuan Kung, Jennifer C. Hou, and Haiyun Luo.**

12. *Dynamic fine-grained localization in Ad-Hoc networks of sensors.* **Andreas Savvides, Chih-Chieh Han and Mani B. Strivastava.** 2001.

13. **Pascual, Maria Deseada Gutierrez.** INDOOR LOCATION SYSTEMS . [Online] 2012. https://www.theseus.fi/bitstream/handle/10024/43966/Gutierrez_Deseada.pdf.

14. *A Method for Measuring, Comparing, and Correcting.* **Feng, J. Borenstein and L.** 1994.

15. **Roland SIEGWART, Illah R. NOURBAKHSH.** *Introduction to Autonomous Mobile Robot.*

16. **Ezequie, Carlos Favis.** Real-Time Map Manipulation for Mobile Robot. [Online] http://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=5678&context=etd.

17. **BELLUSCI, Giovanni.** Ultra-Wideband Ranging for. [Online] https://www.xsens.com/wp-content/uploads/2014/01/Thesis_PhD_Giovanni_Bellusci.pdf.

18. **Liu, Junjie.** Survey of Wireless Based Indoor Localization Technologies. [Online] http://www.cse.wustl.edu/~jain/cse574-14/ftp/indoor/index.html.

19. **Sinan Gezici, Zhi Tian, Georgios B. Biannakis, Hisashi Kobayashi, Andreas F. Molisch, H. Vincent.** Localization via Ultra-Wideband Radios: A Look at. [Online] 2005. http://www.merl.com/publications/docs/TR2005-072.pdf.

20. **Hui Liu, Houshang Darabi,Pat Banerjee, and Jing Liu.** Survey of Wireless Indoor Positioning. [Online] http://www.pitt.edu/~dtipper/2011/Survey1.pdf.

21. Two-Way Time-of-Flight (TW-TOF). [Online] http://www.timedomain.com/wp/wp-content/uploads/2015/12/320-0292C-Part-Three-TOF-Ranging.pdf.

22. *Recent Advances in Wireless Indoor Localization Techniques and System.* **Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail.** 2013.

23. *Effects of anchor placement on mean-CRB for.* **N. Salman, H. K. Maheshwari, A.H. Kemp, M. Ghogho.**

24. *Path planning using a mobile anchor node based on trilateration in wireless sensor networks.* **Guangjie Han1, 2, Huihui Xu1,2 , Jinfang Jiang1,2, Lei Shu3*, Takahiro Hara3andShojiro Nishio3.** 2011.

25. Tracking Architectures Using Two-Way. [Online] http://www.timedomain.com/wp/wp-content/uploads/2015/12/320-0293C-Part-Four-Tracking-Architectures.pdf.

26. Line of Sight (LoS). [Online] https://www.techopedia.com/definition/5069/line-of-sight-los.

27. Localization 0.1.4. [Online] https://pypi.python.org/pypi/Localization/0.1.4.

28. *GEOMETRICAL CONSTRAINED LEAST SQUARES.* **Zhu Hua, Li Hang, Li Yue, Long Hang, Zheng Kan.** 2014.

29. Rover 5 Robot Platform. [Online] https://www.sparkfun.com/products/10336.

30. Rover 5. [Online] http://yourduino.com/docs/Rover5Introduction.pdf.

31. Diameter_Distance. [Online] http://www.education.rec.ri.cmu.edu/previews/rcx_products/robotics_educator_workbook/content/mech/pages/Diameter_Distance_TraveledTEACH.pdf.

32. Dagu4Moto. [Online] https://github.com/sparkfun/Rover5_Motor_Driver_Board/tree/master/Libraries/Dagu4Motor.

33. Pulse Width Modulation. [Online] http://www.electronics-tutorials.ws/blog/pulse-width-modulation.html.

34. PID Control:A brief introduction and guide, using Arduino. [Online] http://www.maelabs.ucsd.edu/mae156alib/control/PID-Control-Ardunio.pdf%20.

35. AHRS using the SparkFun "9DOF Razor IMU" . [Online] https://github.com/ptrbrtz/razor-9dof-ahrs/wiki/Tutorial.

36. Adafruit 9-DOF IMU Breakout - L3GD20H + LSM303. [Online] https://www.adafruit.com/product/1714.

37. *Complete Triaxis Magnetometer Calibration in the Magnetic Domain.* **Valerie Renaudin, Muhammad Haris Afzal, and G ´ erard Lachapelle.** 2010.

38. Improved magnetometer calibration. [Online] http://sailboatinstruments.blogspot.gr/2011/09/improved-magnetometer-calibration-part.html.

39. LSM303DLHC - Calibration, Pitch, Roll and Tilt Compensated Heading. [Online] http://forum.arduino.cc/index.php?topic=265541.0.

40. MTi 1-series. [Online] https://www.xsens.com/products/mti-1-series/.

41. *Using the Kalman Filter to Estimate the State of a MAneuvering Aircraft.* **Desai, K. Meier and A.**

42. *Automatic ground collision avoidance system (Auto GCAS).*

43. Small UAV Automatic Ground Collision. [Online] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150014106.pdf.

44. **Labbe, Roger.** *Kalman and Bayesian Filters in Python.* 2014 : GitHub.

45. **Ellingwood, Justin.** How To Use Systemctl to Manage Systemd Services and Units. [Online] https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units.

46. **WESCOTT, TIM.** PID Without a PhD. [Online] http://m.eet.com/media/1112634.