



UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia

Integration of a Precision Landing System on a Multirotor Vehicle Initial Stages of Implementation

João André Andrade Perdigão

Dissertação para obtenção do Grau de Mestre em
Engenharia Aeronáutica
(ciclo de estudos integrado)

Orientador: Doutor Pedro Vieira Gamboa
Co-orientador: Mestre Pedro Jorge Ferreira Alves

Covilhã, Maio de 2018

Resumo

Os multirotores são utilizados em imensas aplicações actualmente. Para algumas destas aplicações a capacidade de o veículo efectuar missões de forma autónoma é um ponto-chave. O trabalho para esta dissertação consiste nas etapas iniciais da implementação de um sistema de aterragem de precisão para multirotores. Foram escolhidos dois sistemas comercialmente disponíveis e baratos para que possam ser utilizados para fins comerciais. Um dos sistemas (Marvelmind) funciona através da triangulação de distâncias obtidas pela relação entre a velocidade do som e o tempo de transmissão de ultrassons (tempo de voo). O outro sistema (IR-LOCK) implementado funciona com base numa câmara (sensível à radiação infravermelha) que deteta um alvo (emissor de radiação infravermelha) e estima a sua posição relativa. Após a configuração inicial de cada sistema, foram identificados problemas e implementadas soluções apropriadas. Com o sistema Marvelmind, o piloto automático é capaz de identificar a sua posição mas é instável ao realizar um voo pairado e por conseguinte, uma aterragem de forma autónoma não é possível. Com o sistema IR-LOCK, o veículo é capaz de executar aterragens autónomas precisas.

Palavras-chave

Aterragem de Precisão, Multirotor, Alcance Ultrasónico, Câmara infravermelha.

Abstract

Multicopters are used in many applications in the present. For some of these applications the capability of the vehicle to perform autonomous missions is a key element. The work for this thesis consists on the initial stages of a multicopter precision landing system implementation. Two commercially available and cheap systems were proposed so that they can be used for commercial purposes. One of the systems (Marvelmind) works by triangulating distances obtained by the relation between the speed of sound and the delay on a transmitted ultrasound (time of flight). The other system (IR-LOCK) works based on a camera (infrared sensitive) that detects a target (that emits infrared light) and estimates its relative position. After the initial setup for each system, problems were identified and solutions implemented accordingly. With the Marvelmind system, the autopilot is able to identify its position but is unstable when performing a hover flight and as a result, an autonomous landing is not possible. With the IR-LOCK system, the vehicle is capable of performing autonomous precise landings.

Keywords

Precision Landing, Multicopter, Ultrasound Ranging, Infrared Camera.

Table of Contents

Chapter 1 - Introduction	1
1.1 Importance of Autonomous Precision Landing	1
1.2 Airborne Projects	1
1.3 Arducopter and Mission Planner	1
1.4 State of the Art	2
1.4.1 Autonomous Mobile Robots	2
1.4.2 Ultrasound Ranging	2
1.4.3 Infrared Camera and Reflecting Landmarks	4
1.5 Proposed Systems	6
1.5.1 Marvelmind Precise Indoor “GPS”	6
1.5.2 IR-LOCK	6
1.6 Objectives	7
1.7 Dissertation Layout	7
Chapter 2 - Marvelmind Precise Indoor “GPS”	9
2.1 Components and Working Principle	9
2.2 Setting Up	11
2.3 Stand Alone Accuracy and Limitations	13
2.4 Interference with other Systems	15
2.4.1 Telemetry	16
2.4.2 Motors Noise	17
2.4.3 Additional Energy Consumption	18
2.5 Ardupilot Integration Testing	18
Chapter 3 - IR-LOCK	25
3.1 Components and Working Principle	25
3.2 Setting Up	27
3.3 Accuracy and Limitations	27
3.4 Additional Energy Consumption	27
3.5 Ardupilot Tests	28
3.6 Autonomous Precision Landing	30
Chapter 4 - Conclusion	35
4.1 Future Work	35
Bibliography	37
Annex	39

List of Figures

Figure 1 - Typical received signal, adapted from [7].....	3
Figure 2 - Landmark representation [8]	4
Figure 3 - Camera with infrared LED's [8]	4
Figure 4 - Image with infrared light illumination [8].....	5
Figure 5 - Image without infrared light illumination [8].....	5
Figure 6 - Difference image [8]	5
Figure 7 - Landmark recognition [8].....	6
Figure 8 - Estimation result [8]	6
Figure 9 - Marvelmind hardware components	9
Figure 10 - Marvelmind Dashboard.....	10
Figure 11 - Key elements of the system, adapted from [9].....	10
Figure 12 - Dashboard update window	11
Figure 13 - Beacon mounted on a tripod with numbered label that matches the device address on the system.....	12
Figure 14 - Beacon temperature reading vs thermometer (beacon measures 35°C)	13
Figure 15 - Mobile beacon close to vs far away from the stationary beacons error demonstration	14
Figure 16 - Telemetry transceiver	16
Figure 17 - Telemetry with 96% signal strength.....	16
Figure 18 - Oscilloscope window with background noise, signal and reflections identified.....	17
Figure 19 - Representation of the vehicle position and path on Mission Planner.....	20
Figure 20 - Representation of the mobile beacon on Marvelmind Dashboard	20
Figure 21 - Beacon health from a flight data log	21
Figure 22 - Beacon health toggling between 1 and 0, messages of the position system used are visible... 21	21
Figure 23 - Different sensors measured similar altitudes.....	22
Figure 24 - Position controller altitude relative to home vs barometer altitude.....	23
Figure 25 - Accelerometer measured values, spikes just before the rises in altitude on.....	23
Figure 26 - IR-Lock Sensor and MarkOne Beacon	25
Figure 27 - Pixymon software with the sensor capturing infrared light	26
Figure 28 - Identification of an infrared light source	26
Figure 29 - Rotating the lens from very unfocused to most focused to ideally focused	27
Figure 30 - Precision landing health and target acquired parameters	28
Figure 31 - Wobbling in the PL.pX parameter	29
Figure 32 - Wobbling after the PID auto tune.....	29
Figure 33 - Designed and manufactured plumb.....	30
Figure 34 - Marking a point demonstration	31
Figure 35 - Landings in four different orientations with default PID values.....	31
Figure 36 - Landings after the P_xy reduction to 0.500	32

Figure 37 - Landings after the RL_I increase to 1.000.....32
Figure 38 - Landings after the IMAX reduction to 5033

List of Tables

Table 1 - Marvelmind system accuracy and repeatability test	13
Table 2 - Ardupilot parameters related to the Marvelmind system.....	18
Table 3 - Ardupilot parameters related to IR-LOCK system.....	28

List of Acronyms

UBI	Universidade da Beira Interior
UAV	Unmanned Aerial Vehicle
GCS	Ground Control Station
TOF	Time of Flight
AGC	Automatic Gain Control
IR	Infrared
LED	Light Emitting Diode
CMOS	Complementary Metal-Oxide-Semiconductor
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
EKF	Extended Kalman Filter
SNR	Signal to Noise Ratio
Serial TX	Serial Transmitter
Serial RX	Serial Receiver
I ² C	Inter-Integrated Circuit
PID	Proportional-Integral-Derivative

Chapter 1

Introduction

A multirotor or multicopter is a rotary wing aircraft with three or more rotors. This aircraft configuration is being used more and more in unmanned aerial vehicles (UAV) due to their simplicity, small size, ease of construction, flight characteristics and ease of control. Furthermore, their ability to take off and land on very small spaces is also a very important asset. Whether they are used for recreational activities like racing and photography, or professional activities such as photogrammetry or aerial delivery, multirotors integrate state-of-the-art technology.

1.1 Importance of Autonomous Precision Landing

As progressively more tasks are done with the aid of multirotors, the need for autonomous flights also increases. Autonomous flights are limited by the energy needed for the aircraft to complete the mission, and upon completing the mission or failing to do so because of battery depletion, the vehicle needs to land at a given point after following a given trajectory with precision in order to avoid obstacles if it is not being controlled by a pilot or the pilot cannot see the aircraft. The capability to land with precision (+/- 5 cm) also enables for autonomous battery swapping if an auto docking system is available.

1.2 Airborne Projects

“Airborne Projects is a Portuguese start-up company focused on developing and selling drone related products and technology from the ground up. (...) Airborne Projects is developing an **Auto Docking System for Battery Swapping** for zero down-time drone operation (...)” [1] To make this docking system a possibility, precision landing is mandatory. Otherwise the aircraft would not land on the dock with certainty. Airborne Projects developed code for the Arducopter [3] (autopilot software) but needed testing and debugging of such code. The company also provided all hardware needed for this thesis like the quadcopter and sensors used along with guidance and aid whenever a new problem arose.

1.3 Arducopter and Mission Planner

“Ardupilot is the most advanced, full-featured and reliable open source autopilot software available.” [2] As the battery swapping system is intended for commercial use, it is important to use a reliable, used worldwide and cost free (open source) autopilot software. Arducopter [3] is a part of the Ardupilot directed to multirotors and other rotor vehicles. This software can run on many different boards, the most common being Pixhawk [5], which is the one used for this work. Mission Planner [4] is a computer software that works as a Ground

Control Station software for Pixhawk. It allows to upload the Ardupilot firmware, to change parameters, to monitor flight data if telemetry is available and to analyze flight data logs.

1.4 State of the Art

1.4.1 Autonomous Mobile Robots

Robots thrive in industrial manufacturing. Whether it is because they perform tasks quicker, more precise or even make otherwise impossible jobs, the robotics field cannot be ignored. A huge advantage that a robot can have is mobility. Locomotion is a key aspect for applying a robots capability where it is most effective. Another characteristic that makes a robot fascinating is autonomy, the ability to control and operate itself. When a robot is both mobile and autonomous, it is essential that it maintains a sense of position to enable a reliable navigation through the environment that surrounds it.

“Navigation is one of the most challenging competences required of a mobile robot. Success in navigation requires success at the four building blocks of navigation: *perception*, the robot must interpret its sensors to extract meaningful data; *localization*, the robot must determine its position in the environment; *cognition*, the robot must decide how to act to achieve its goals; and *motion control*, the robot must modulate its motor outputs to achieve the desired trajectory.” [6]

1.4.2 Ultrasound Ranging

In 1997, Wehn and Bélanger [7] proposed a novel 3-D (three dimensional) ultrasound position sensor to compute the high-precision yet low-cost measurements of the Cartesian position of a robotic wrist. For this, an ultrasound transmitter is attached to the robot’s end effector. This sender emits timed bursts of sound that are received by microphones away from the robotic wrist. As the speed of sound has a finite value, there will be a time delay between the moment that the sound is emitted and the moment it is received. Knowing the speed at which the sound propagates (c), it is possible to relate this time delay (time of flight, tof or TOF) with the distance (d) between the sender and the receiver. This relation is the equation (1):

$$tof = \frac{d}{c} \quad (1)$$

The 3-D position is calculated via triangulation using an array of microphones (typically four or more) fixed at know distinct points therefore providing a range measurement each. Additionally, if three senders are used instead of one, it is possible to calculate the end effector orientation.

The speed of sound depends greatly on the air temperature (φ) and the wind velocity ($\vec{\omega}$). It can be given approximately by

$$c_i = c_0 \sqrt{1 + \frac{\varphi}{273} + \vec{c}_i^T \vec{\omega}} \quad (2)$$

where c_i is the speed of sound in the direction \vec{c}_i^T and $c_0 = 331.4 \text{ m/s}$ [7].

This approximation is limited by the room temperature drift, the temperature gradient and the fluctuations of the speed of the air.

In order to make the ultrasound readings more reliable, this solution uses a combination of threshold and zero crossing detection. In Figure 1 it is possible to see the signal (D), a reflection of the signal (E) and a reflection from a previous sample (A). The trigger has to be set so that it can distinguish the desired from the undesired signals. After the trigger indicates the desired signal has arrived, the signal delay is measured on the zero crossing (C), making this delay measurement independent of the signal amplitude. But because the signal arrived at B and not at C, the difference in time between these two points (time of arrival, TOA) has to be subtracted from the timer reading that starts when B leaves the sender and stops at C.

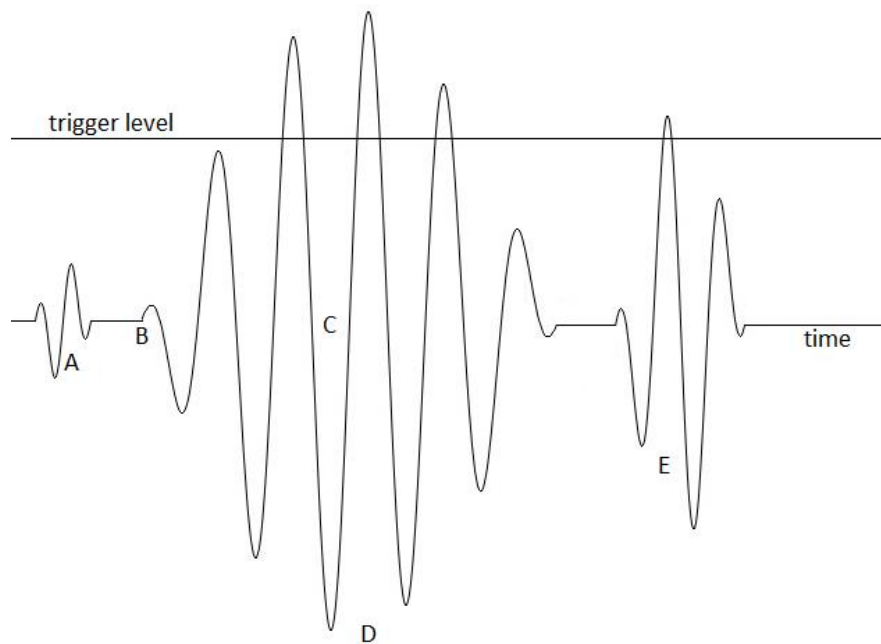


Figure 1 - Typical received signal, adapted from [7]

Although the figure does not show, the signals A, D and E can overlap resulting in phase shifts and distortions. If the amplitude has large variations, the time of arrival will have an error which is a multiple of the wave length. Automatic Gain Control (AGC) is a possible solution to overcome this issue and maintain a steady amplitude. Other sources of noise are electronic distortions and non-ideal transducers.

1.4.3 Infrared Camera and Reflecting Landmarks

In 2007, Sooyong Lee and Jae-Bok Song [8] developed a localization system that relies on a camera that identifies landmarks coated with an infrared reflecting material. The camera is mounted on a pan and tilt mechanism in order to reduce the number of needed landmarks. The pan and tilt make the images distorted, so a correction algorithm is essential. The landmark is a right angle isosceles triangle, which makes it possible to estimate orientation. Figure 2 is a representation of the landmarks, the three corner sectors in blue are for recognizing a landmark and estimating orientation. The blue sectors are coated with a reflective material on all landmarks. The other sectors labeled with numbers 1 to 6 in yellow are for identification of specific landmarks when more than one is used. Each of these sectors is either coated or not with the reflective material in order to make a 6 bit binary unique identity ($2^6 = 64$ unique possibilities).

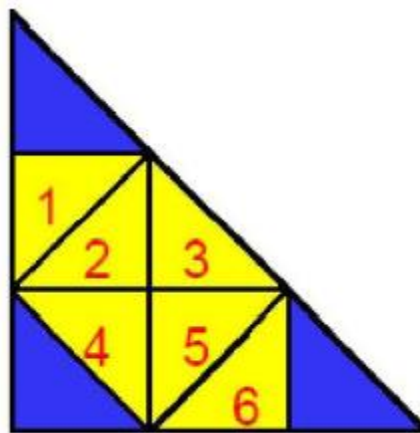


Figure 2 - Landmark representation [8]

A CMOS camera is used for image capture and an array of infrared LED's serve to illuminate the reflective materials (Figure 3).



Figure 3 - Camera with infrared LED's [8]

The image used for the recognition (Figure 6) is not captured by the camera. Instead, it is the difference between an image captured with infrared illumination (Figure 4) and an image captured without it (Figure 5).

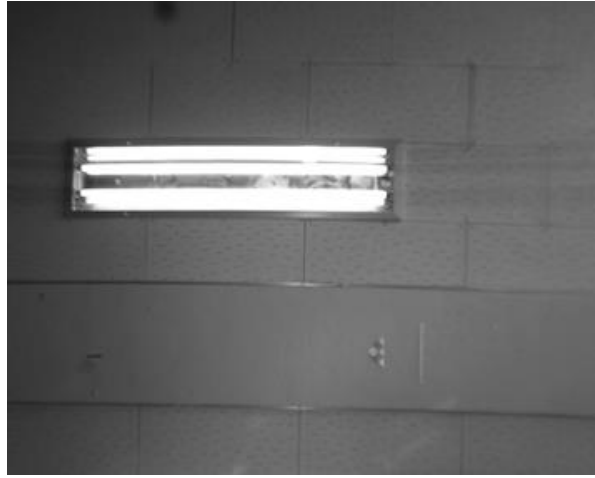


Figure 4 - Image with infrared light illumination [8]

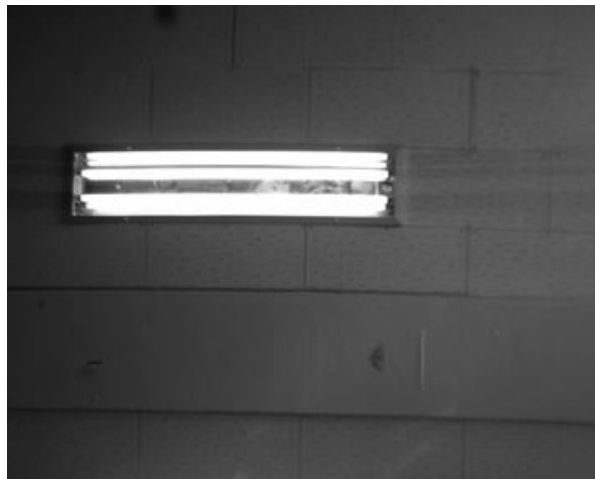


Figure 5 - Image without infrared light illumination [8]

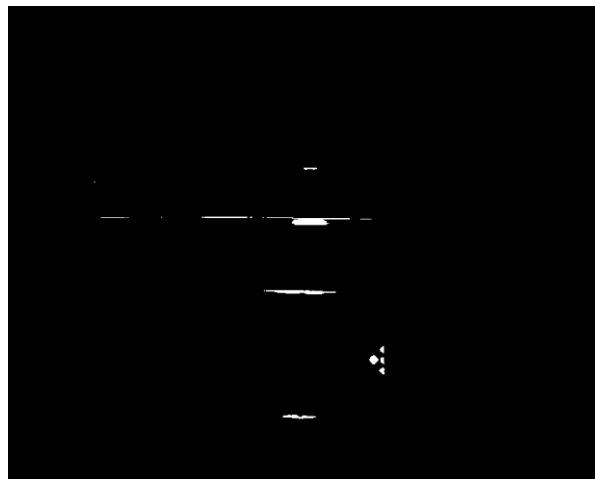


Figure 6 - Difference image [8]

The resulting difference (Figure 6) is an image with only the infrared reflective materials visible, making the noises easily removable or ignored and the landmark recognized with robustness despite the indoor lighting. When the landmark is recognized (Figure 7), its center of mass and orientation is estimated (Figure 8). The reflective sectors are also identified.



Figure 7 - Landmark recognition [8]

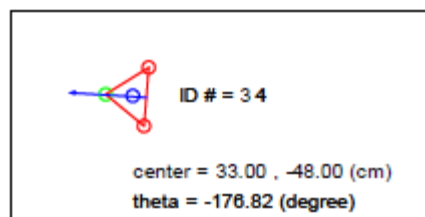


Figure 8 - Estimation result [8]

1.5 Proposed systems

To perform a precision landing, a GNSS (Global Navigation Satellite System, referred to as GPS) alone is not enough. Two commercially available and cheap systems were proposed for implementation of a precision landing, as described below.

1.5.1 Marvelmind Precise Indoor “GPS”

Although the creators of this system call it an indoor “GPS”, its only similarity to a GPS system is that it calculates its position by triangulating the distances from the mobile device (hedgehog) to the fixed devices (stationary beacons). The method by which the distances are acquired is with ultrasounds: the hedgehog emits an ultrasound at a known instant and the stationary beacons listen to this sound. Then the distance between the hedgehog and stationary beacons is estimated based on the time of flight of the ultrasound. The processing, clock synchronization, time of flight estimation and position estimation of this system is made on another device (modem) that communicates to the hedgehog and stationary beacons via radio (~433 MHz). This system had only minor implementations on Arducopter and did not work. This was the starting point towards developing the proposed solution. [9]

1.5.2 IR-LOCK

IR-LOCK has only two components, the MarkOne Beacon (a set of infrared LEDs) and the IR-LOCK Sensor (a camera to be mounted on the vehicle). The working principle is very simple: the beacon is put on the target point on which the aircraft should land; upon getting close to this point, the camera mounted on the quadcopter identifies the target and with its own

computational resources it is able to know the position relative to it. This solution is already implemented on Arducopter. [10]

1.6 Objectives

The goal of the work for this thesis is to perform the initial development stages of a precision landing capability using the proposed systems while providing Airborne Projects with reliable flying tests, identifying problems, pointing to possible causes of the problems and to propose solutions in order to advance in the implementation of a precision landing.

1.7 Dissertation Layout

This thesis contains four chapters. The first chapter introduces the autonomous precision landing on a multicopter and shows its importance. Some works that rely on similar working principles are described as well as the two systems used to achieve an autonomous precision landing.

The second chapter talks about one of the systems used, the Marvelmind Precise Indoor “GPS” describing its components, working principle, how to set it up, its accuracy and limitations. This chapter explains the issues found while implementing the system with the vehicle and the autopilot.

The third chapter is similar to the second chapter but instead describes the other system used, IR-LOCK. Also presented is a statistical study regarding the landing precision.

The fourth chapter is the conclusion. It expresses the outcome of the work and leaves some suggestions for future development of this work.

Chapter 2

Marvelmind Precise Indoor “GPS”

2.1 Components and Working Principle

This system comes with one modem and 5 beacons (Figure 9). One of the latter is to be used as the mobile beacon or hedgehog (this needs only a change in software since the stationary beacons and hedgehog are identical in hardware). Also needed is the Dashboard (Figure 10), a computer software provided by Marvelmind used to set up and update the system. The modem is the central controller of the system, which is also the means of communication between the dashboard and the beacons. The modem and beacons communicate through 433 MHz radio signals (915 MHz in United States due to regulations).



Figure 9 - Marvelmind hardware components

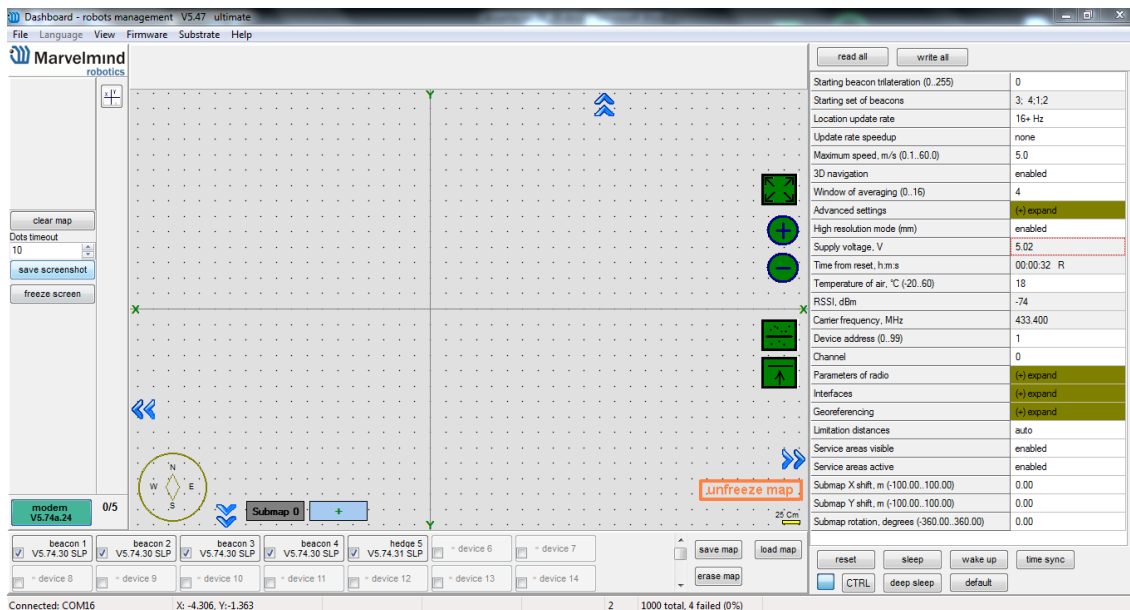


Figure 10 - Marvelmind Dashboard

The system works by triangulating the distances between the mobile beacon and the stationary beacons, thus outputting the relative position of the mobile beacon. To get this distance, the mobile beacon emits an ultrasound at a known time and the stationary beacons wait until they receive it counting how long the sound signal took to be transmitted through the air. The modem then calculates the distances using the speed of sound (based on the temperature of the air) and computes the position to be displayed on the dashboard and transmitted to the mobile beacon.

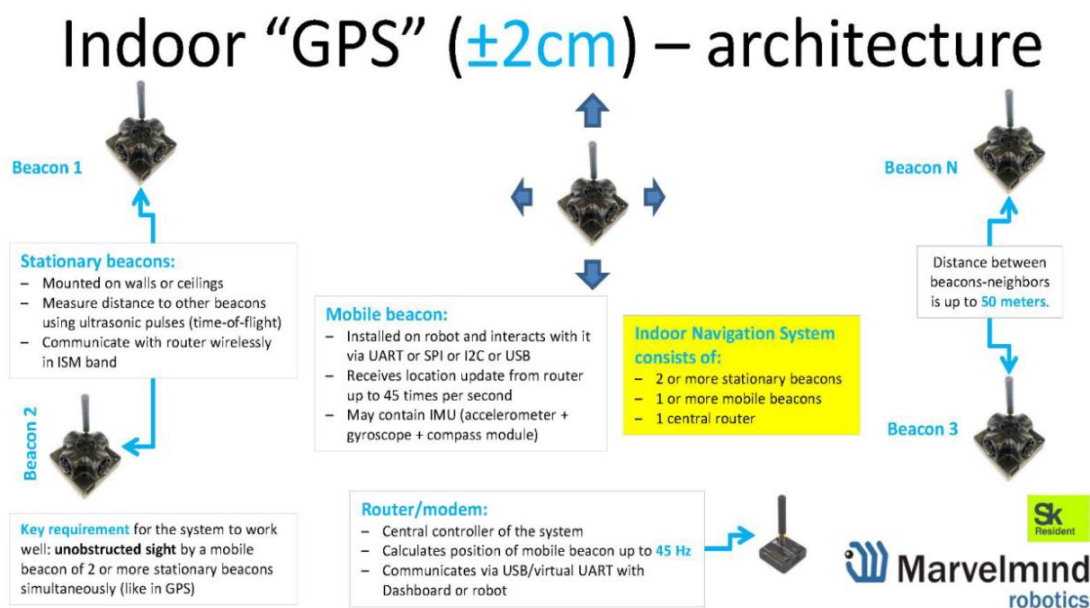


Figure 11 - Key elements of the system, adapted from [9]

Before being able to calculate a position, the system needs to map the stationary beacons. These will interchange ultrasound signals between themselves and the modem will then create a map that can be seen on the dashboard. The map always has one of the sensors as the origin. On the dashboard it is possible to select which sensor is the origin and the orientation of the map relative to the beacons. Figure 11 is a schematic that shows a brief description of the key elements of the system.

2.2 Setting Up

The first action is to download the latest firmware version available for all the components. Marvelmind is constantly releasing updates with bug fixes and new features. Each time an update is released, it is released in a pack which contains all three updatable components (Dashboard + Modem + Beacon) so keeping the devices compatible is important. Updating is simple, the dashboard is an executable file (one just needs to replace the existing file with the new one) and the beacons and modem are updated within the dashboard (Figure 12). After updating, it is advised to set all parameters to default values. There is a button to perform this action on the dashboard.

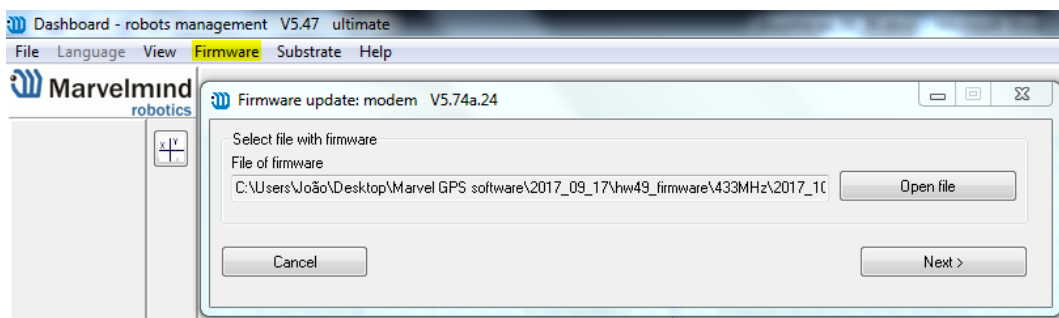


Figure 12 - Dashboard update window

Then it is time to set the stationary beacons on the ground and the mobile beacon inside the area created by them. It is already possible to see the map being formed on the Dashboard. The next step is to change the parameters according to our needs. The devices addresses were changed to 1-5 and the beacons where also labeled (Figure 13). That makes it easier to identify problems. As it is intended to use the system on a quadcopter, the best configuration is with the stationary beacons on the ground and mobile beacon above them. This configuration is the opposite of the default because the system was firstly developed to be used in non-flying robots, so it needs to be set. To be able to calculate the correct height of the hedgehog, the height of each stationary beacon has to be known. This was set after measuring their height on a custom made 3D printed tripod (Figure 13).



Figure 13 - Beacon mounted on a tripod with numbered label that matches the device address on the system

As the system works with the time it takes for an ultrasonic signal to travel through the air, it is possible to set the environment temperature in order to increase the accuracy of measurements. The speed of sound can be calculated by the equation (3):

$$c = \sqrt{kRT} \quad (3)$$

where k is the ratio of specific heats, R is the individual gas constant and T is the temperature. To see how much the error in temperature affects the sensor's readings, let us calculate a known distance (for instance 30 m) with an error of 5°C. First, we calculate the speed of sound at two temperatures 5°C apart (20°C and 25°C):

$$c_{T=20^{\circ}C} = \sqrt{1.4 * 286.9 * 293.15} = 343.14 \text{ m/s} \quad (4)$$

$$c_{T=25^{\circ}C} = \sqrt{1.4 * 286.9 * 298.15} = 346.06 \text{ m/s} \quad (5)$$

If one of these speeds is used to get the time it took the signal to travel the distance and then use that time is used with the other speed to induce the 5°C error:

$$\text{Time } t = \frac{30}{343.14} = 87.43 * 10^{-3} \text{ s} \quad (6)$$

$$\text{Distance } d = 87.43 * 10^{-3} * 346.06 = 30.26 \text{ m} \quad (7)$$

This example assumes we are measuring a 30 m distance at 20°C but, assuming 25°C, that results in a reading of 30.26 m. In the Marvelmind product manual the speed of sound is said to vary 0.6 m/s per °C leading to an error of 0.17% of distance per °C off.

This variance is the reason why Marvelmind included the temperature parameter to be set by the user. Even though the beacons have onboard temperature sensors, one should not use their values as these are wrong as we can see in Figure 14, using a thermometer. Other factors which can change the speed of sound like pressure or humidity are not accounted for in this sensor.

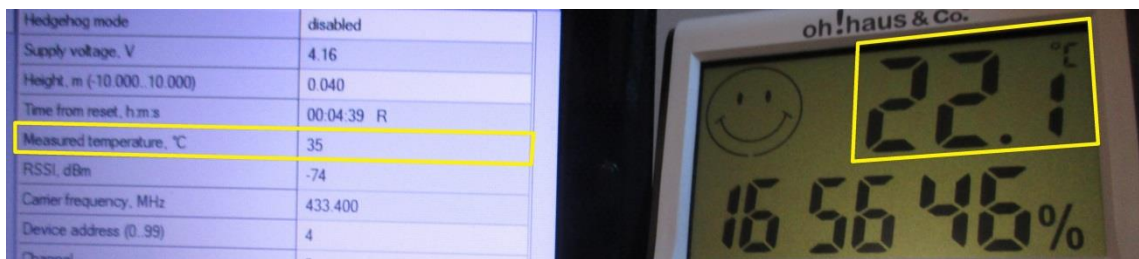


Figure 14 - Beacon temperature reading vs thermometer (beacon measures 35°C)

2.3 Stand Alone Accuracy and Limitations

To test the accuracy of the system, a simple test was made. A prototype table (later to be used by Airborne Projects as the landing base) was put inside the area of the sensors; then the mobile beacon is positioned in each of the four corners and the coordinates were captured. This was done five times to test the repeatability of the measurements. The table is squared and the sides measure 66 cm, but the mobile beacon was placed at points 1.5 cm away from the edges in each direction to prevent it from falling off the edge, thus making the points 63 cm apart from each other in a squared way. Table 1 shows the coordinates captured.

Table 1 – Marvelmind system accuracy and repeatability test

	Corner 1	Corner 2	Corner 3	Corner 4
Capture 1	1.545 ; -0.315	2.180 ; -0.305	2.183 ; 0.320	1.560 ; 0.308
Capture 2	1.546 ; -0.317	2.179 ; -0.305	2.177 ; 0.323	1.558 ; 0.308
Capture 3	1.545 ; -0.319	2.179 ; -0.307	2.185 ; 0.321	1.559 ; 0.309
Capture 4	1.548 ; -0.315	2.183 ; -0.304	2.181 ; 0.323	1.562 ; 0.307
Capture 5	1.548 ; -0.313	2.180 ; -0.304	2.181 ; 0.319	1.560 ; 0.304

All the coordinates are in meters. Further analyzing these values we can see that the maximum difference between two different readings of the same point was 5.4 mm and the maximum error in the 63 cm that separate the corners was 12.9 mm. Adding these two errors we get 18.3 mm which is very close to the precision stated in the Marvelmind manual (+/- 20 mm).

Even though the manual says the modem can be placed up to 100 meters away from the beacons, after a distance of 10 m, some of the position estimates fail (this is verified by the blue dot on the dashboard toggling transparent) lowering the position update frequency. This lower position update rate may not be a problem for many other applications where the system can wait for the next position update or a different estimate method is used, but in the implementation with Arducopter, a lower position update rate is a problem because the autopilot relies on having a constant position estimate to know if the system is in range and working properly.

The beacons should not be mounted directly on materials that conduct sound faster than air, for example metals. This is to avoid an ultrasound signal being transmitted through this material to the receiving beacon before the signal that was transmitted through the air, this would be a problem because the first signal is interpreted as the real signal and all the others arriving later as reflections, which are ignored by the software. A small stand/tripod was designed and printed with a 3D printer so that the beacons are not mounted on the ground.

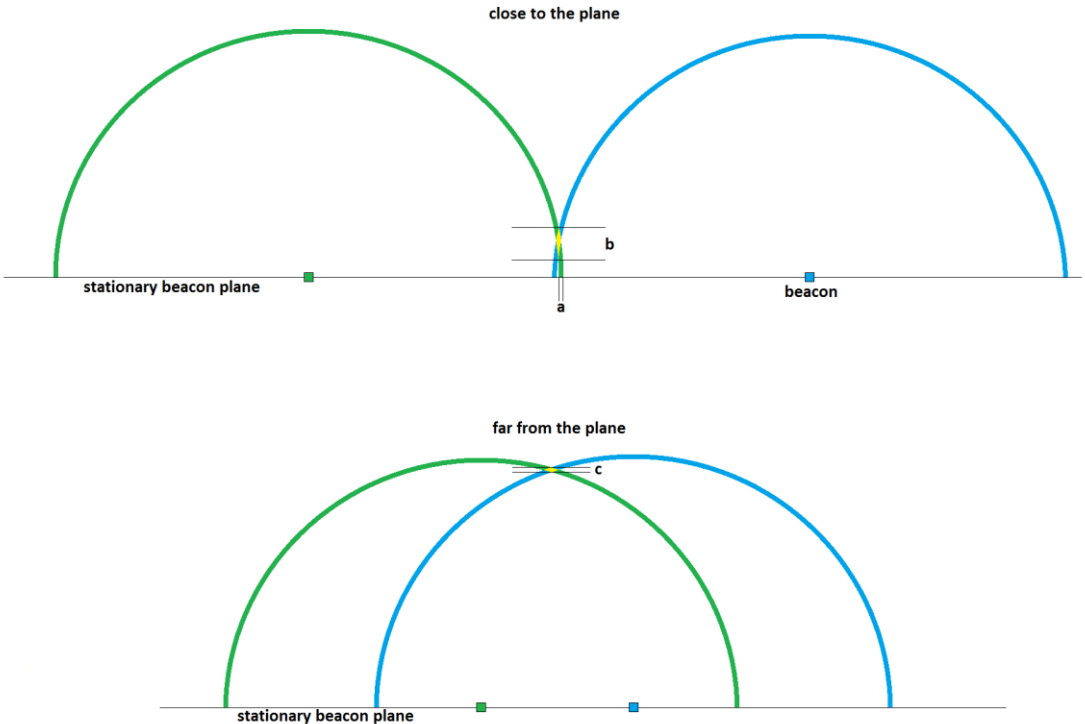


Figure 15 - Mobile beacon close to vs far away from the stationary beacons error demonstration

The system calculates a tri-dimensional position, making an imaginary pyramid where the base is defined by the plane created by the stationary beacons and the mobile beacon is the apex of the pyramid. Because of that, when the mobile beacon is near the plane of the stationary beacons, the error in height is greatly increased. This occurs because a small error in the distances translates into a big uncertainty in height. This is easier to demonstrate with a two dimensional image and only two stationary beacons (Figure 15).

Figure 15 represents the two cases, when the mobile beacon is close and far away from the stationary beacons plane. The green and blue squares represent two stationary beacons on the ground. The green and blue full lines represent the distances measured from the mobile beacon to each respective stationary beacon. These lines have a thickness a that represents an arbitrary error associated with the measurement and mean that the mobile beacon can be at any point on this thick line. The system ignores the possible points where the mobile beacon can be on the other side of the stationary beacons plane (in our case, below the ground). In yellow, the intersection between the green and blue lines is shown, which represents where the mobile beacon can be when triangulated. As we can see, when the mobile beacon is close to the stationary beacons plane, its height as an uncertainty b much larger than the one associated with the distances measurements a . When far away from the stationary beacon plane, the height error c is just slightly larger than a .

2.4 Interference with other Systems

A multicopter is a very complex vehicle with many systems, so it is expected that some of this systems may interfere with each other. In this section, the interferences identified during the tests and work are described.

2.4.1 Telemetry

The quadcopter has telemetry to give the possibility of reading some values and changing parameters in real time. The telemetry initially used works with radio signals at 433 MHz (Figure 16), this was obviously prone to interference with the Marvelmind system so it was the first thing tested. To test the telemetry, we simply monitored the telemetry signal with the Marvelmind system turned on and off. Without the positioning system, the telemetry signal is 100%. When the sensors are turned on, the telemetry signal goes down to 28%. This telemetry system does not have the option to change the frequency but the Marvelmind system does, so, different frequencies were tested. The default frequency was 433.4 MHz, but when changed to 433.0 MHz, the telemetry signal strength dropped to 23%. Emitting at 430.0 MHz was better leaving the telemetry signal at 96% (Figure 17). Even though this was good enough for the telemetry signal, the Marvelmind system also has to work flawlessly so this was tested, and the conclusion was that the system fails some position updates when the telemetry is turned on. The solution for this problem was to replace the telemetry system

with one that does not use radio signals near 433 MHz. A telemetry that works at 868 MHz was tested and there were no interferences.



Figure 16 - Telemetry transceiver



Figure 17 - Telemetry with 96% signal strength

2.4.2 Motors Noise

The electric motors and propellers on the quadcopter emit noise in many different frequencies and also make the structure vibrate. This noise and vibration was another plausible source of problems or inaccuracy in the Marvelmind system, so a test was made for different distances (1 to 5 meters in steps of 1 meter) to verify how much this was a problem. The test is made using the oscilloscope in the dashboard, a tool that allows us to see the received signal in any pair of selected beacons in each direction. The oscilloscope shows the received signal in a window where it is possible to see the background noise, a first peak which represents the signal and the following peaks are reflections (Figure 18).

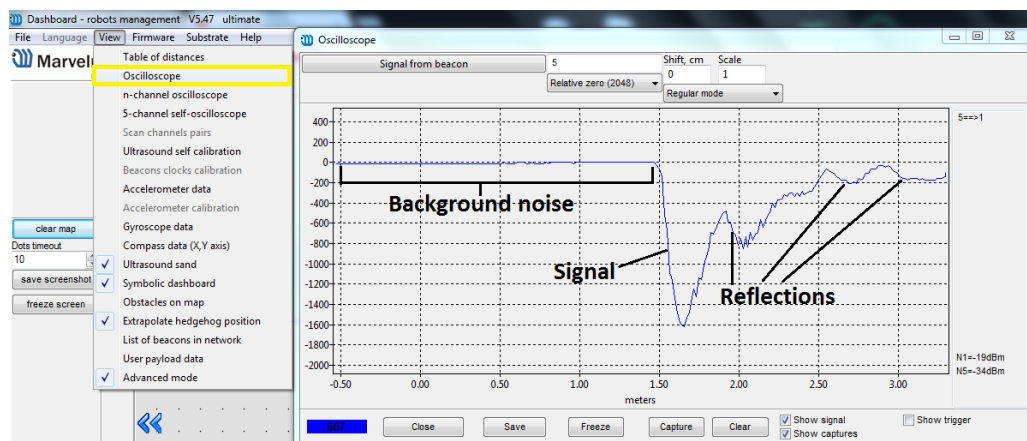


Figure 18 - Oscilloscope window with background noise, signal and reflections identified

In addition to verifying this signal, it is possible to tune the amplification on the beacons in order to achieve the best signal to noise ratio. By default, the amplification mode is set to Automatic Gain Control (AGC), but, when using motors and propellers, Marvelmind advises to change this to manual and then find the best value for the receiver amplifier. The test was done only in one direction, which is the mobile beacon emits and the stationary beacon receives the ultra sound signal because this is the direction in which the system works. The test was initially supposed to include various throttle settings because some motor frequencies could cause a characteristic noise that was specifically more harmful to the Marvelmind system performance. It was verified that this was not the case and that the noise was proportional to the throttle setting, so the test was made with the worst case, full throttle. The system works by identifying a peak on the receiver signal, so, the best signal to noise ratio is the key. There is no advantage in having a very high peak if the noise is also very high, so there is no interest in raising the amplification value too much because the beacon will trigger a distance with the noise and identify the real signal as a reflection.

After analyzing the captures (all the captures are in Annex A), we can see that using the ACG is not the best option. A good value for the receiver amplifier is 1500, even though it does not correspond to the best signal to noise ratio for over 4 m, at this distance the noise is less

significant, and, as the quadcopter gets closer to the beacons for the landing (2-3 meters is the distance Airborne Projects intends to build the station) this ratio gets better.

2.4.3 Additional Energy Consumption

The mobile beacon is mounted on the vehicle, thus increasing its weight which in turn increases the energy needed to fly. To measure the additional energy needed to use the beacon, two similar hover flights were performed, with and without the beacon for the same amount of time (5 minutes). Each flight was performed with a fully charged battery and indoors to avoid wind. To know the energy consumed in each flight, a battery charger that provides the amount of energy recharged was used. The multicopter mass is 1297 g without the beacon and the beacon mass is 60 g, so the useful load of the aircraft is reduced by 60 g. The flight without the beacon consumed 895 mAh and the one with the beacon 940 mAh. The beacon increased the energy consumption by approximately 5.03 % which represents a total flight time of approximately 95.21 % of the total flight time without the beacon. Because the flights lasted 5 minutes and the quadcopter has a battery capacity of 2300 mAh, we can estimate the endurance of the aircraft, which is approximately 12 minutes and 51 seconds without the beacon and 12 minutes and 14 seconds with the beacon (37 seconds difference).

2.5 Ardupilot Integration Testing

An implementation of this system for Ardupilot had already been tried. Even though that the previous implementation did not work, it was the starting point for this implementation. The first thing to do is to connect the two systems and try to get them working. To connect the two systems all that is needed is to make the ground connection and the Serial Tx pin of the mobile beacon to one of the Serial Rx pins on the Pixhawk. Also, the Ardupilot had to be prepared for this change, so the parameters were set as in Table 2.

Table 2 - Ardupilot parameters related to the Marvelmind system

BCN_ALT	654
BCN_LATITUDE	40,27811
BCN_LONGITUDE	-7,513942
BCN_ORIENT_YAW	348
BCN_TYPE	2
EK2_BCN_DELAY	50
EK2_BCN_I_GTE	500
EK2_BCN_M_NSE	1
BRD_SER1_RTSCCTS	0
SERIAL1_BAUD	115
SERIAL1_PROTOCOL	13
GPS_TYPE	0
GPS_TYPE2	0
ARMING_CHECK	-14

Then it was time to turn on the quadcopter and expect to see its position represented on the Mission Planner map. That did not happen and the Pixhawk did not even initialize properly

reporting all sorts of errors like *bad compass health* or *gyros still settling*. The most probable cause of this problem was bad communication between the two systems, so the protocol through which the mobile beacon exports its data was changed on the dashboard. The default protocol is *Marvelmind*, the other two options are *Testing data* and *NMEA 0183*. Connecting the Serial Rx of the mobile beacon to the Serial Tx on Pixhawk was also tried. None of those worked but log files were created and analyzed in order to identify the problem.

From the logs it was concluded that the Pixhawk was not booting properly because the Serial port was always flooded. This is because the mobile beacon was already sending information before the Pixhawk was ready to receive any. The solution for this was very simple, the mobile beacon could only be powered on after the Pixhawk has booted properly. Later, a more elegant solution for this was to remove the power source of the mobile beacon and use the same power source used to power the Pixhawk. In this way, the two systems are always powered at the same time and the mobile beacon also takes a few moments to boot giving time for Pixhawk to be ready for receiving data. A parameter was also included in the Ardupilot that makes it ignore the Serial Port for a defined time (`TELEM_DELAY` set to 30 seconds).

This worked well and Pixhawk booted without issues; a few moments later the Marvelmind system finds the mobile beacon and starts calculating its position which is at this point being transmitted to the Ardupilot. Then it is possible to see the representation of the vehicle on the map on Mission Planner (**Figure 19**) even though it was not very accurate as it moved slightly when the quadcopter was still or did not mimic the movements the vehicle made while carried by hand in a cross pattern. Those small errors were not caused by the Marvelmind sensors as their calculated position was also being monitored and correct (**Figure 20**). There was no evident action to take at this point so Airborne Projects started developing their own code for Ardupilot.



Figure 19 - Representation of the vehicle position and path on Mission Planner

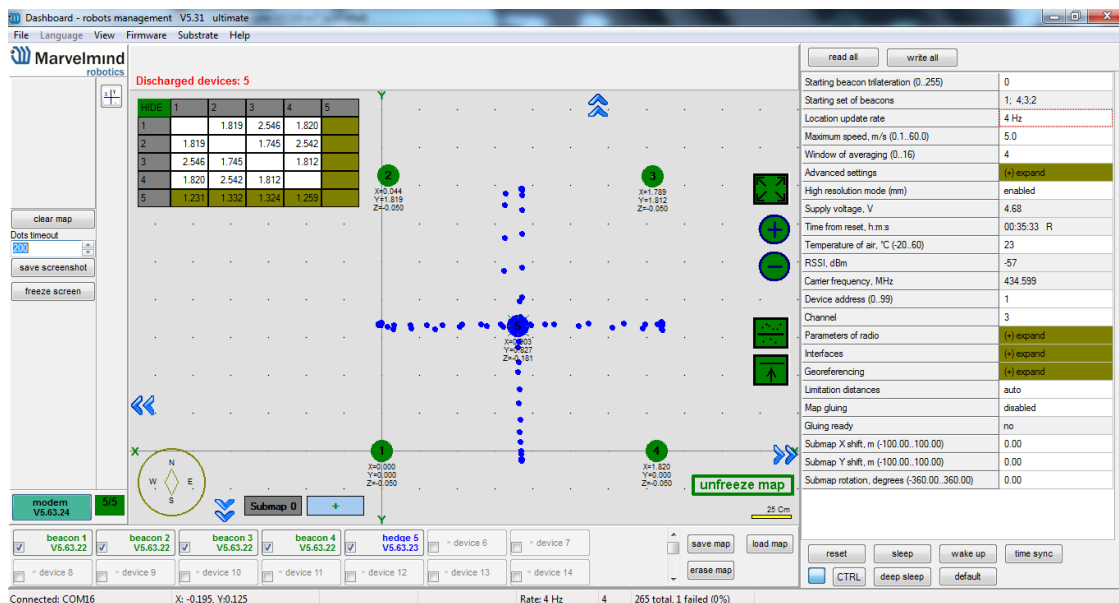


Figure 20 - Representation of the mobile beacon on Marvelmind Dashboard

The first change in the new code was the kalman filter used to fuse the sensors data. Extended Kalman Filter 2 (EKF2 or EK2) was used previously but it was changed to the Extended Kalman Filter 3 (EKF3 or EK3) from here on, which is currently a work in progress for the Ardupilot as the EKF2 is the one used in stable versions of the code. This iteration also revised and changed the beacon health function. This function outputs the Beacon Health parameter (BCN.Health) which determines if the beacons system can be used to estimate the vehicle's position. This is essential as the quadcopter needs to fly with the GPS when outside the range of the beacons. Being able to monitor the beacon health also makes it much easier to identify problems in the future (Figure 21). While testing the beacon health it was noted that the Marvelmind modem should be close (less than 10 meters) to the stationary beacons otherwise the beacon health is not constantly maintained.

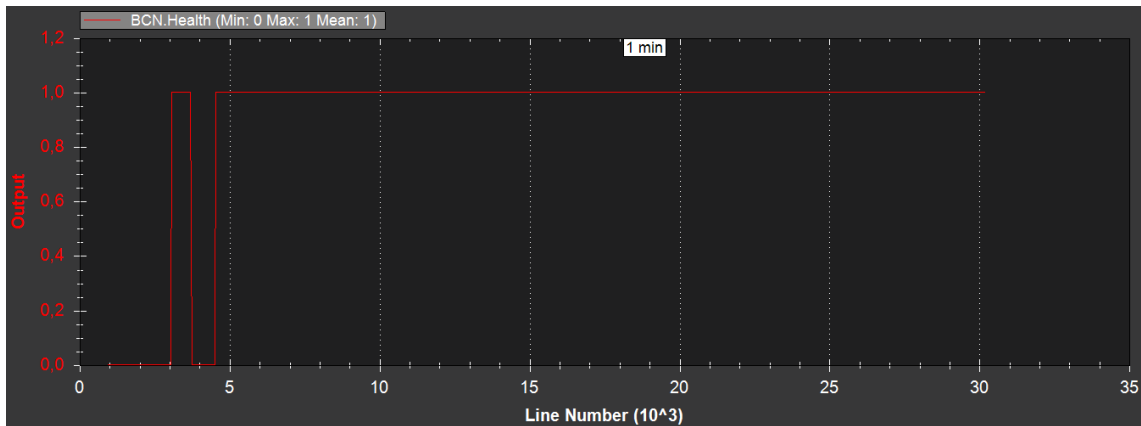


Figure 21 - Beacon health from a flight data log

The next step was to test whether the autopilot was capable of changing between using the beacons and using GPS. To do that, the vehicle was simply flown in and out of the beacons range. Airborne Projects included messages in the flight data logs to provide a means of analyzing this (Figure 22).

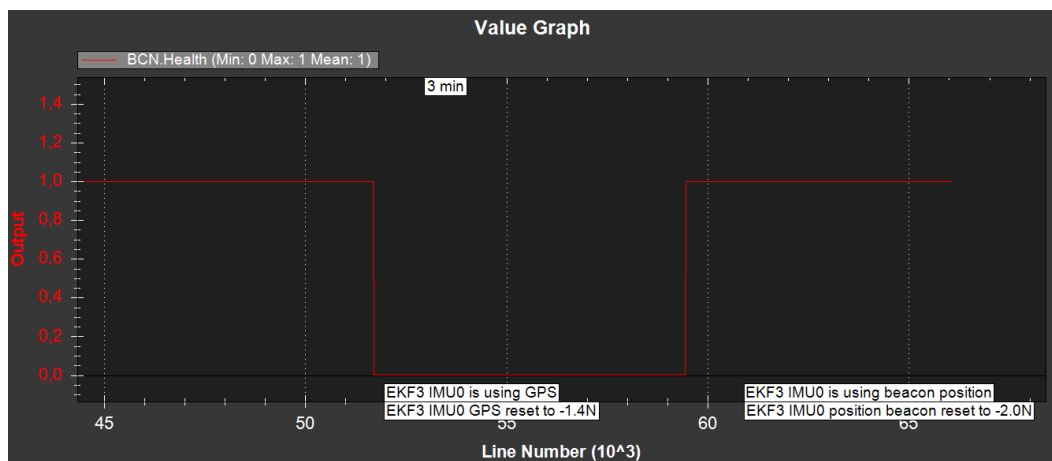


Figure 22 - Beacon health toggling between 1 and 0, messages of the position system used are visible

As it can be seen in Figure 22, as the beacon health toggles between 1 and 0, the autopilot does the change as expected. It can also be seen that the autopilot takes longer to change from GPS to beacons than the opposite; that is because there is no problem in using the GPS while in beacon range, but when moving out of the beacons range, there is no position system from those.

After making sure the connection between beacons and the autopilot was reliable, it was time to see the behavior of the vehicle while trying to maintain its position. To test this, the quadcopter was in position hold mode. After lifting off, it is possible to see the quadcopter making corrections in order to maintain its position, even with light wind, but with an error of about 1 meter. Randomly, the quadcopter has spikes of power, climbing very fast for a brief moment. These climbs are unexpected, so the flight data logs were analyzed to see if it were

possible to identify the source of this. Analyzing the logs, we can confirm the rise in altitude was sensed on beacons, GPS and barometer (Figure 23). It can also be seen that the position controller is not very accurate regarding the altitude, sometimes being below the value calculated when the vehicle was on the ground (Figure 24).

While looking for what could have caused this, one thing found was the accelerometer readings, which had some unusual oscillations and measured accelerations larger than the gravitational acceleration which is clearly a wrong measurement because the motors produce thrust in the opposite direction of gravity during the entire flight therefore the only way this could be right is if something pushed the quadcopter down, and that did not happen. Another thing that is relevant is that these clearly wrong readings happen just before the unexpected power spikes, this makes sense because the autopilot “thinks” it is falling and tries to correct this. In Figure 25 we can see the unusual oscillations on lines ~31500 and ~41000 and the altitude rise just after these. Lines ~30000 and ~48000 represent takeoff and landing respectively. This problem does not happen when the beacons are not being used so it leads to the belief that this implementation has had some negative impact on the accelerometer readings.

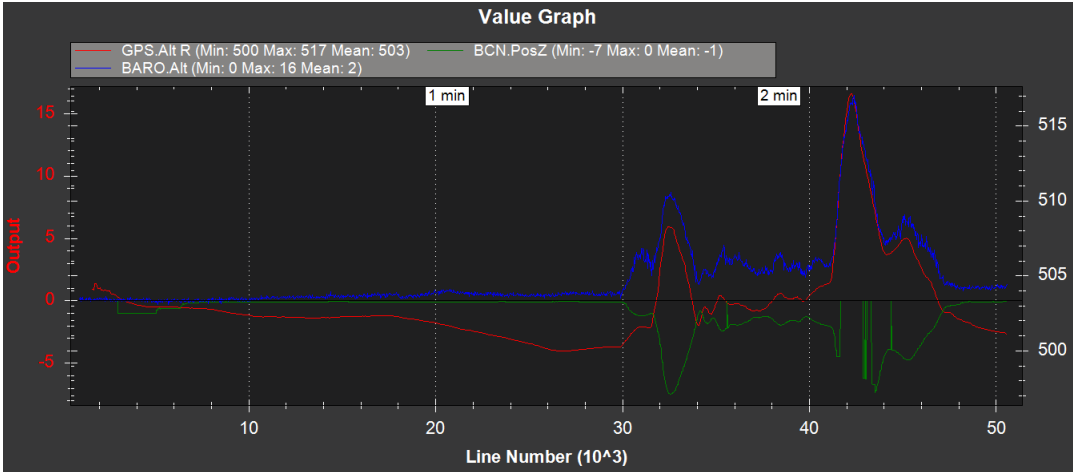


Figure 23 - Different sensors measured similar altitudes

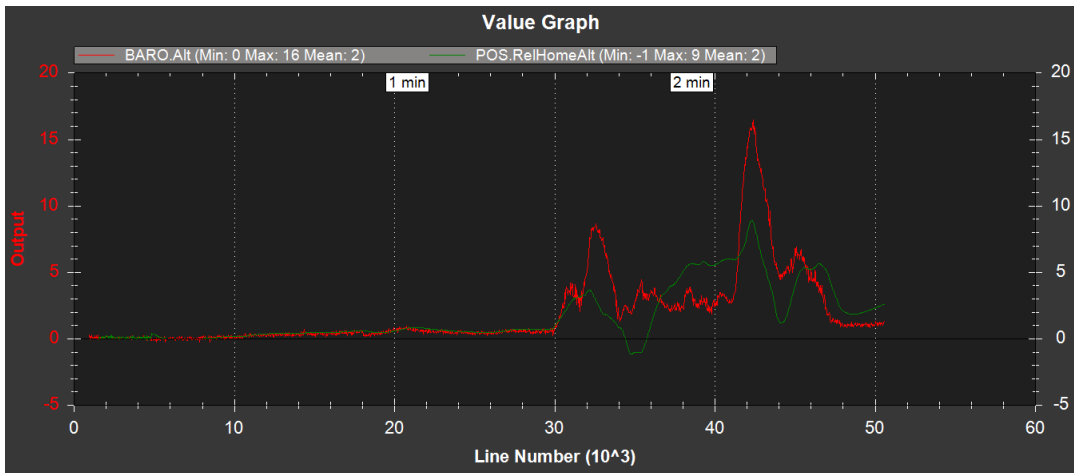


Figure 24 - Position controller altitude relative to home vs barometer altitude

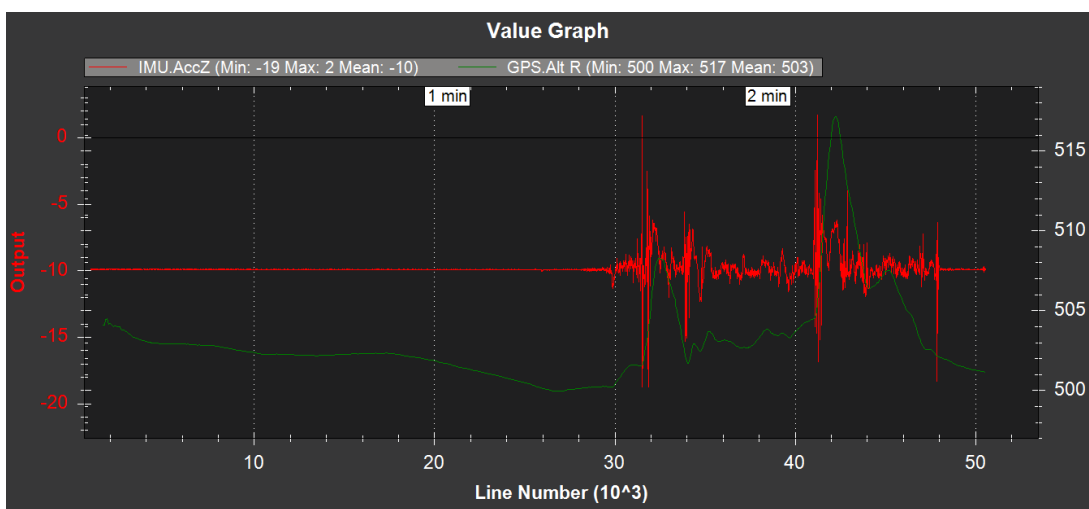


Figure 25 - Accelerometer measured values, spikes just before the rises in altitude on

Because the vehicle is not able to maintain its position in a stable way, there is no possibility to perform autonomous landings and assess their accuracy.

Chapter 3

IR-LOCK

3.1 Components and Working Principle

This system consists of a MarkOne beacon and an IR-LOCK Sensor. The MarkOne beacon is simply an array of infrared LED's and the IR-LOCK Sensor is a camera with an infrared sensor and lens with integrated hardware and software (Figure 26). Also needed is the Pixymon (Figure 27), a free computer software that enables to update the camera and see what the camera captures, this is essential in focusing the lens. The beacon is to be set at the desired location for the landing and the camera mounted on the vehicle pointing down. The camera can only capture infrared images to make it more reliable. Upon capturing an infrared light source, the camera identifies it and its position on the frame (Figure 28). All the hardware and software necessary to make this identification and output it to other systems is on the sensor.

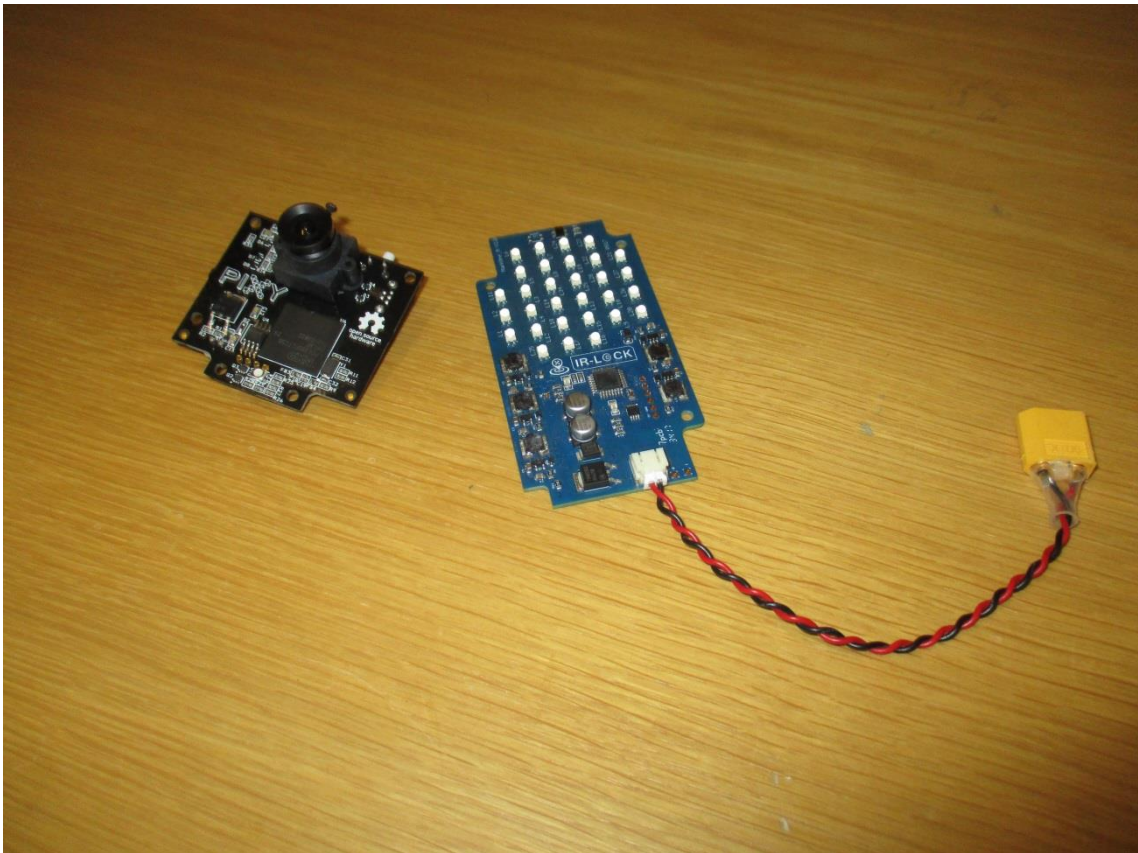


Figure 26 - IR-Lock Sensor and MarkOne Beacon

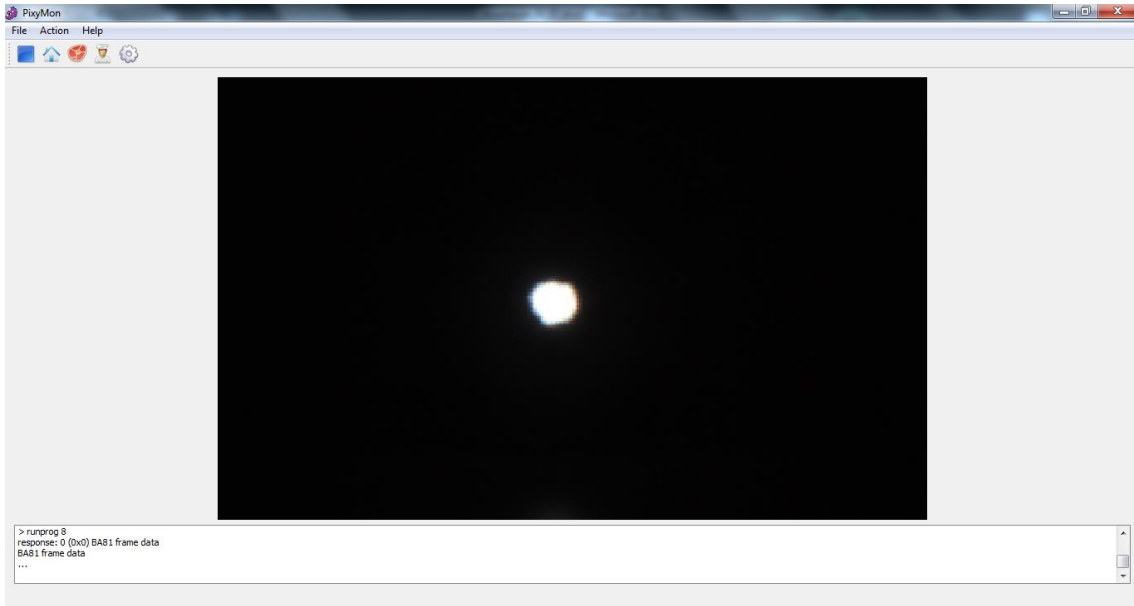


Figure 27 - PixyMon software with the sensor capturing infrared light

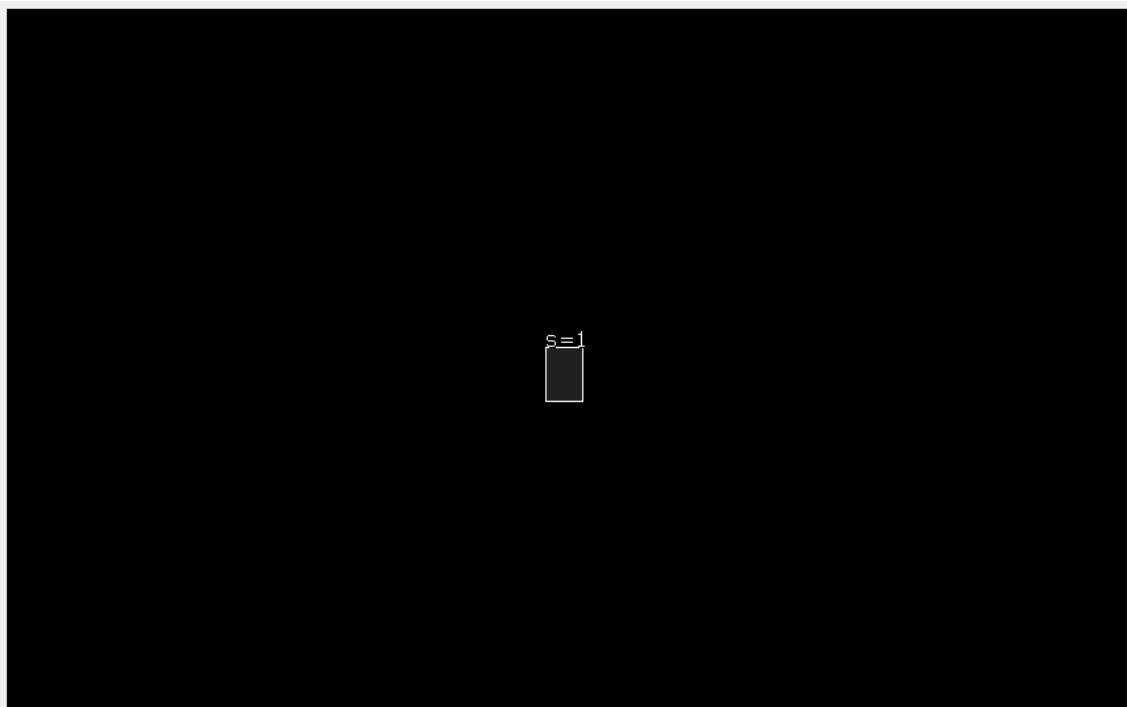


Figure 28 - Identification of an infrared light source

3.2 Setting Up

This system comes almost ready to use, the only things necessary is to install the latest firmware available and to adjust the lens' focal distance. To update the firmware, the white pushbutton on the sensor has to be pressed when connecting the sensor to the computer, then, the PixyMon software will automatically ask to locate the firmware file to be installed. This file is also provided for free. Focusing the lens is very straightforward, it is done by rotating the lens until the image is as clear/sharp as possible and then just turn it $\frac{1}{4}$ of a

rotation for the ideal focus (Figure 29). This slightly unfocused setting will make the target look a bit “bigger” making it visible at a higher distance. At this point it is also good to test if the camera is able to identify the target using a Pixymon script (Figure 28).



Figure 29 - Rotating the lens from very unfocused to most focused to ideally focused

3.3 Accuracy and Limitations

Other than being possible to see the target identification on Pixymon software, it is not possible to test this system’s accuracy. This can only be tested when working with Ardupilot in a whole.

As this is a vision based system, reflections are obviously a problem. The sun and other light sources emit in infrared wavelength too, so reflections can happen. To go around this problem, there should be no lightbulbs pointing up near the landing site and reflective surfaces that could reflect the sunlight need to be covered. Reflections far away from the landing site are not a problem because this system will only be used to land.

3.4 Additional Energy Consumption

Similar to the Marvelmind beacon, the IR-LOCK sensor also increases the aircraft weight and energy consumption. The method to know the additional energy consumption that the sensor imposes was the same as with the Marvelmind beacon (5 minute flight). The aircraft weights 1297 g and the IR-LOCK sensor weights 27 g. Flying without the sensor consumed 895 mAh and flying with the sensor consumed 909 mAh which represents approximately 1.56 % increase. The endurance reduction is approximately 1.54 % which is nearly insignificant for this vehicle.

3.5 Ardupilot Tests

IR-LOCK is already implemented on Ardupilot so the objective of these tests was more to assess its performance than to test it. IR-LOCK Sensor and Pixhawk communicate through I²C, so only 4 wires are needed: power, ground, SDA (Serial Data Line) and SCL (Serial Clock Line). Then it was time to set the parameters on the autopilot code as in Table 3.

Table 3 - Ardupilot parameters related to IR-LOCK system

PLND_ACC_P_NSE	2,5
PLND_BUS	0
PLND_CAM_POS_X	0,15
PLND_CAM_POS_Y	0
PLND_CAM_POS_Z	0,05
PLND_ENABLED	1
PLND_EST_TYPE	0
PLND_LAND_OFS_X	0
PLND_LAND_OFS_y	0
PLND_TYPE	2
PLND_YAW_ALIGN	0

If Pixhawk and IR-LOCK Sensor communication is working the PL.Heal (Precision Landing Health) should be always 1 on the flight data logs. This was verified (Figure 30). Another important thing to check is if the camera can identify the target. This could also be verified with PL.TAcq (Precision Landing Target Acquired), when the MarkOne Beacon was put in and out of sight of the camera (Figure 30). It is also possible to see the horizontal distance from the target to the vehicle (PL.pX and PL.pY) as well as their relative speed (PL.vX and PL.vY).

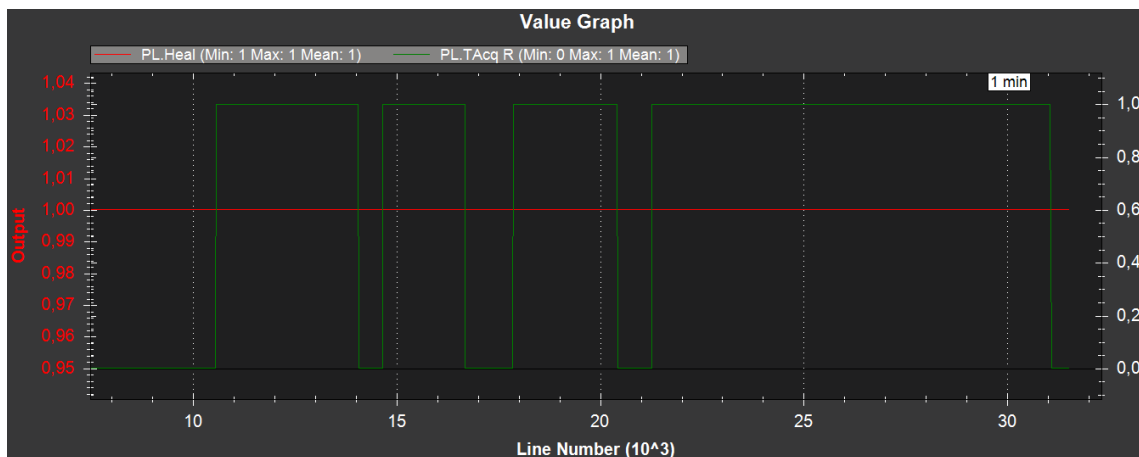


Figure 30 - Precision landing health and target acquired parameters

After that it was time to fly. Before attempting an automatic pilot landing, another flight mode was tested, the precision loiter. In this mode, the vehicle tries to hover over the target. The target was put on the ground and after lifting off and positioning the vehicle approximately over it in manual mode, the flight mode is changed to precision loiter. It is possible to see that the aircraft tries to maintain itself over the beacon, even when the beacon was pulled with a string to move on the ground the quadcopter followed it in the air. However, the hover was not very stable, the vehicle kept overshooting and wobbling around the desired position. This wobbling can be seen on the flight data logs with the horizontal distance captures (Figure 31).

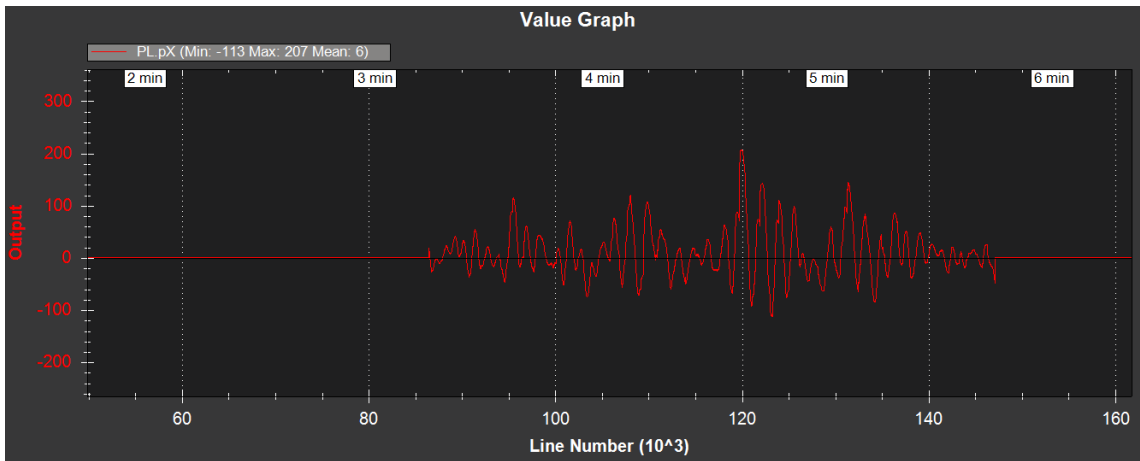


Figure 31 - Wobbling in the PL.pX parameter

One probable cause for this overshooting problem was the PID settings of the vehicle, these were still the default values. Autopilot has an auto tune feature to self-calibrate the PID values. After the PID auto tuning, the overshooting still occurs but on a smaller scale (Figure 32).

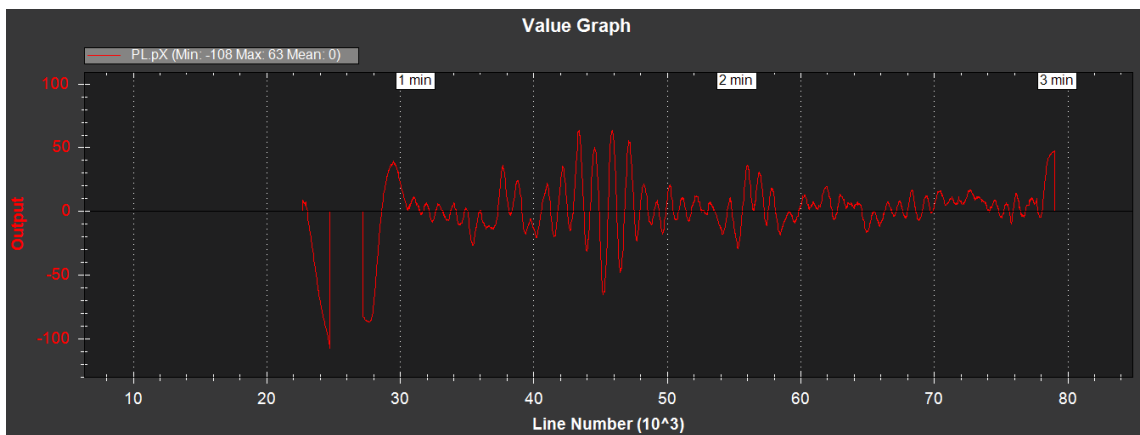


Figure 32 - Wobbling after the PID auto tune

Then it was time to perform an automatic landing, for this, the vehicle took off in manual mode away from the landing location and then flew to a position above the MarkOne beacon, then the flight mode was changed to automatic landing. After the aircraft positions itself directly above the target, it starts a slow descent while trying to maintain the horizontal position until it hits the ground and stops the motors. The vehicle landed close to the target so it was time to study the landing precision.

3.6 Autonomous Precision Landing

To study the landing precision, the MarkOne Beacon was set on a target with a grid. Then, a measuring plumb mass was designed and manufactured with the help of a 3D printer (Figure 33), the plumb mass allows for an easier and more accurate way of marking each point (Figure 34). Each measurement represents the horizontal distance between the center of the

led matrix on the MarkOne Beacon (approximately a square shape) and the lens sensor on the IR-LOCK sensor. To get the coordinates of the landings, a photo of the target with the marked points was used in WebPlotDigitizer [11]. All of the landings were performed in light wind.

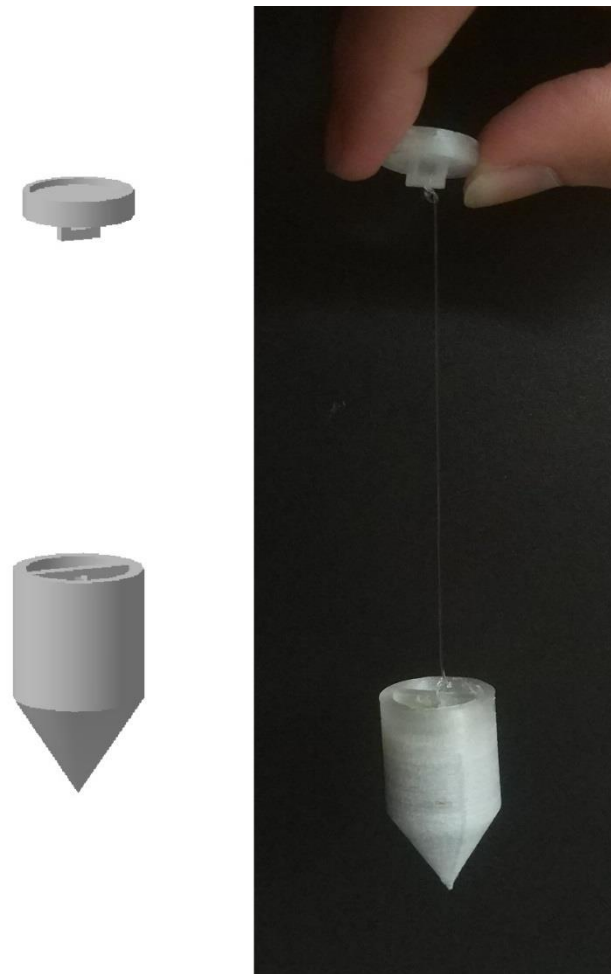


Figure 33 - Designed and manufactured plumb mass

To make the landings more accurate, the vertical speed was reduced to half, setting the parameter `LAND_SPEED` to 15 cm/s. A total of 20 landings were performed in 4 different orientations of the vehicle 90° apart relative to the target. These 4 different orientations had the purpose of checking if the orientation of the quadcopter relative to the target had an influence in the error direction. Figure 35 shows the 20 landings and the default PID settings for the autopilot position landing controller. P_{xy} is the position controller proportional gain. It converts the horizontal distance to the target location into a desired speed. *Rate Loiter PID's* are the horizontal velocity proportional (RL_P), integrative (RL_I) and derivative (RL_D) gains, respectively. These convert the difference between the actual and desired velocity into an acceleration (or lean angle). $IMAX$ is the maximum acceleration that the Rate Loiter integrative gain can output.



Figure 34 - Marking a point demonstration

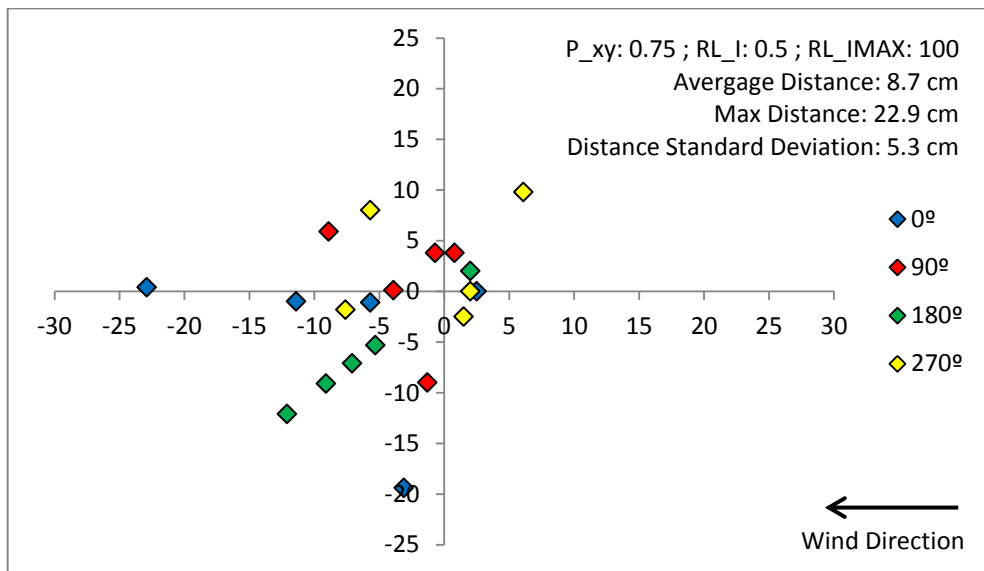


Figure 35 - Landings in four different orientations with default PID values

The orientation of the vehicle relative to the target does not influence the error direction. The landings seemed to be very affected by wind making for bigger errors when the wind was stronger. Also, there was one landing which is not in Figure 35 because it landed outside of the target due to a stronger gust of wind.

Besides the default PID settings, a few different combinations of these values were tested. The first change was the P_{xy} gain from 0.750 to 0.500 (Figure 36) to try to reduce the speed at which the vehicle makes corrections. Because the orientation of the vehicle had no influence in the previous landings, the next landings were all performed in the same orientation.

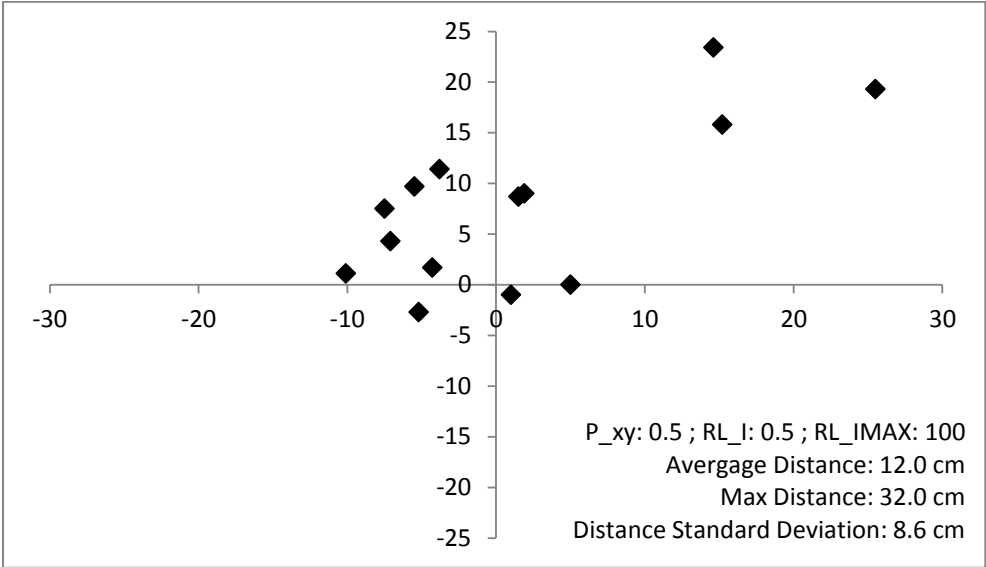


Figure 36 - Landings after the P_{xy} reduction to 0.500

Reducing the P_{xy} did not improve the accuracy and made the vehicle more affected by the wind. This time, two landings not showed Figure 36 outside of the target area. The change in the P_{xy} was reverted back to 0.750 and the RL_I value doubled to 1.000 to make the vehicle react to small errors faster (Figure 37).

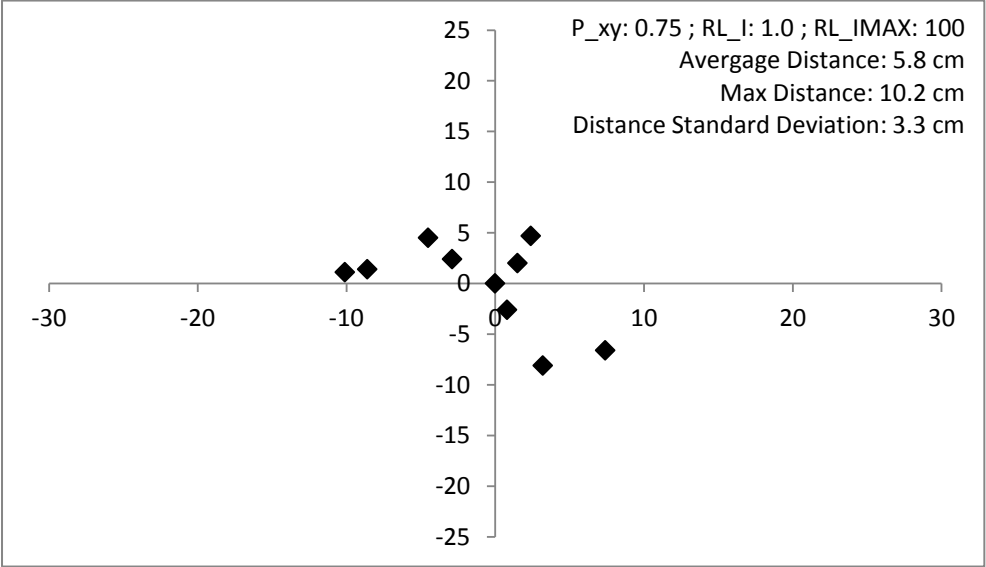


Figure 37 - Landings after the RL_I increase to 1.000

This change was the most significant. None of the landings was outside of the target area and also the average and maximum distance error were reduced as well as its standard deviation. In an attempt to increase the accuracy even more, the RL_IMAX was reduced to half (from 100 to 50) because we previously increased the integrative gain. Figure 38 shows the landing points for this change which was not very significant.

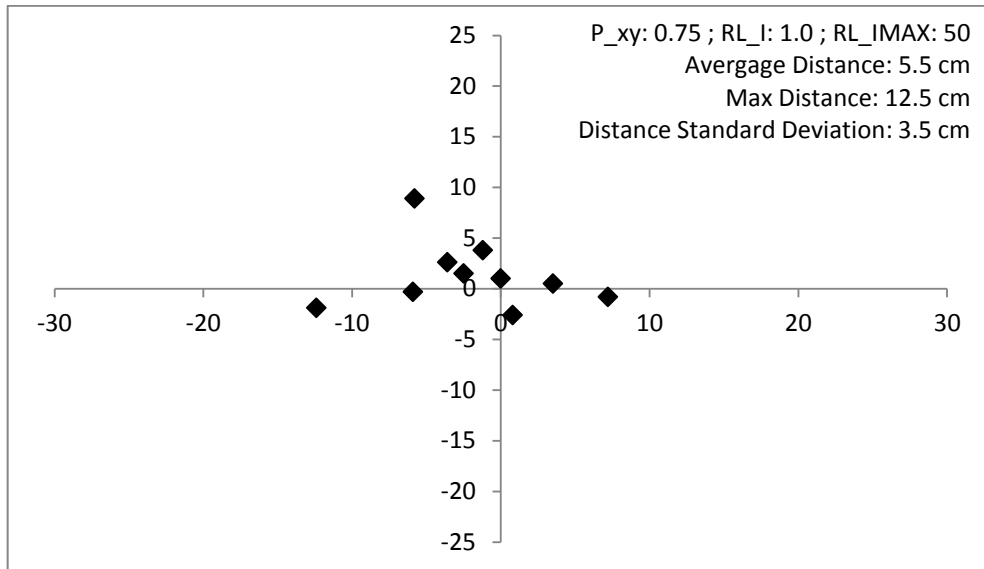


Figure 38 - Landings after the IMAX reduction to 50

Another problem that was observed when performing these autonomous landings was that the vehicle can become unstable if it is high above the target. The speed at which the vehicle tries to correct its position error is directly proportional to the height of the vehicle above the target. This makes the vehicle achieve large horizontal speeds and oscillate in a non-stable way if it is high enough, leaving the MarkOne beacon out of the field of view of the IR-LOCK sensor which makes the vehicle land where it is with no regard for the precision landing system. To overcome this problem, the parameter *Loiter Speed* was reduced from 500 cm/s to 50 cm/s. The change was not very effective so it was reduced again to 25 cm/s and the problem was solved.

Chapter 4

Conclusion

This work completed the initial stages of the precision landing integration on a quadcopter. Both systems (Marvelmind and IR-LOCK) used were set up. All the interferences with the other systems on the multirotor have been solved. Their standalone accuracy is in line with the positioning precision needed for the autonomous landing and their limitations do not impose a problem for the implementation. Communication between the autopilot and the positioning systems has been accomplished and the autopilot can determine its position based on the systems proposed. When using the Marvelmind system, the autopilot is still not able to hover while maintaining its position so it still needs further development in order to achieve an autonomous landing capability. On the other hand, the IR-LOCK system is capable of performing autonomous precision landings. Some parameters were tuned to enhance the accuracy of the vehicle.

The goals of the work for this thesis were accomplished as the initial development stages for a precision landing capability were achieved using the proposed systems. There were problems identified such as the increased error in height when the mobile beacon is close to the stationary beacons plane, the telemetry interference, the motors noise, the Pixhawk not booting and the inability to fly in a stable hover for the Marvelmind system. Problems were also identified with the IR-LOCK system like the wobbling around or overshooting the desired position and the instability when higher above the MarkOne Beacon. Some of the solutions proposed were the change of telemetry hardware, the tuning of the receiver amplifier in order to minimize the motors noise effect and the PID tuning of the vehicle. Also, the quadcopter used for this work has performed more than 50 landings on a platform with the IR-LOCK system.

4.1 Future Work

The following are recommendations on the next steps:

- Develop the autopilot code until it can hover with the Marvelmind system;
- Further tune the control settings to make the vehicle more precise and resistant to winds.

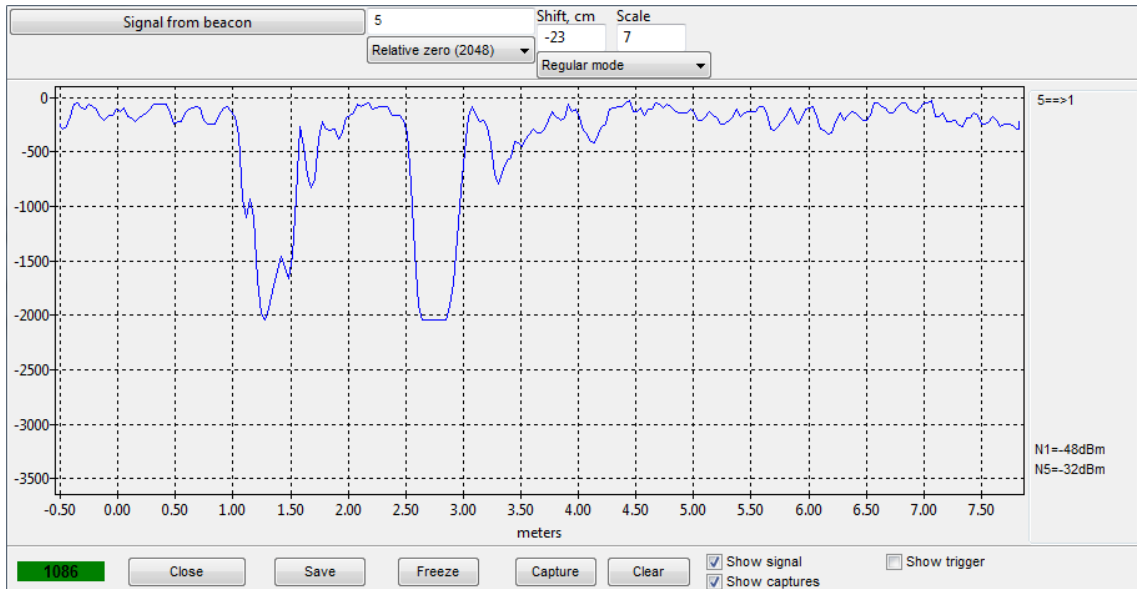
Bibliography

- [1] AIRBORNE PROJECTS. (2017). *About us*. [online] Available at: <https://www.airborneprojects.com/about-us/> [Accessed Apr. 2018]
- [2] Ardupilot.org. (2016). *ArduPilot :: About*. [online] Available at: <http://www.ardupilot.org/about> [Accessed Apr. 2018]
- [3] Ardupilot.org. (2016). *Copter Home – Copter documentation*. [online] Available at: <http://www.ardupilot.org/copter> [Accessed Apr. 2018]
- [4] Ardupilot.org. (2016). *Mission Planner Home – Mission Planner documentation*. [online] Available at: <http://www.ardupilot.org/planner/> [Accessed Apr. 2018]
- [5] Pixhawk.org. (2015). *Home - Pixhawk Flight Controller Hardware Project*. [online] Available at: <https://pixhawk.org/> [Accessed Apr. 2018]
- [6] Siegwart, R. and Nourbakhsh, I. (2004). *Introduction to autonomous mobile robots*. Cambridge, Mass.: MIT Press, pp.1-12, 181
- [7] Wehn, H. and Belanger, P. (1997). Ultrasound-based robot position estimation. *IEEE Transactions on Robotics and Automation*, 13(5), pp.682-692
- [8] Sooyong Lee and Jae-Bok Song (2007). Mobile robot localization using infrared light reflecting landmarks. *2007 International Conference on Control, Automation and Systems*, Seoul, Korea, 17-20 October
- [9] Marvelmind Robotics. (2017). *Marvelmind Robotics*. [online] Available at: <https://marvelmind.com/> [Accessed Apr. 2018]
- [10] IRLOCK. (2018). *IR-LOCK Precision Landing Tutorial · IRLOCK*. [online] Available at: <https://irlock.readme.io/> [Accessed Apr. 2018]
- [11] Automeris.io. (2018). *WebPlotDigitizer - Extract data from plots, images, and maps*. [online] Available at: <https://automeris.io/WebPlotDigitizer/> [Accessed Apr. 2018]

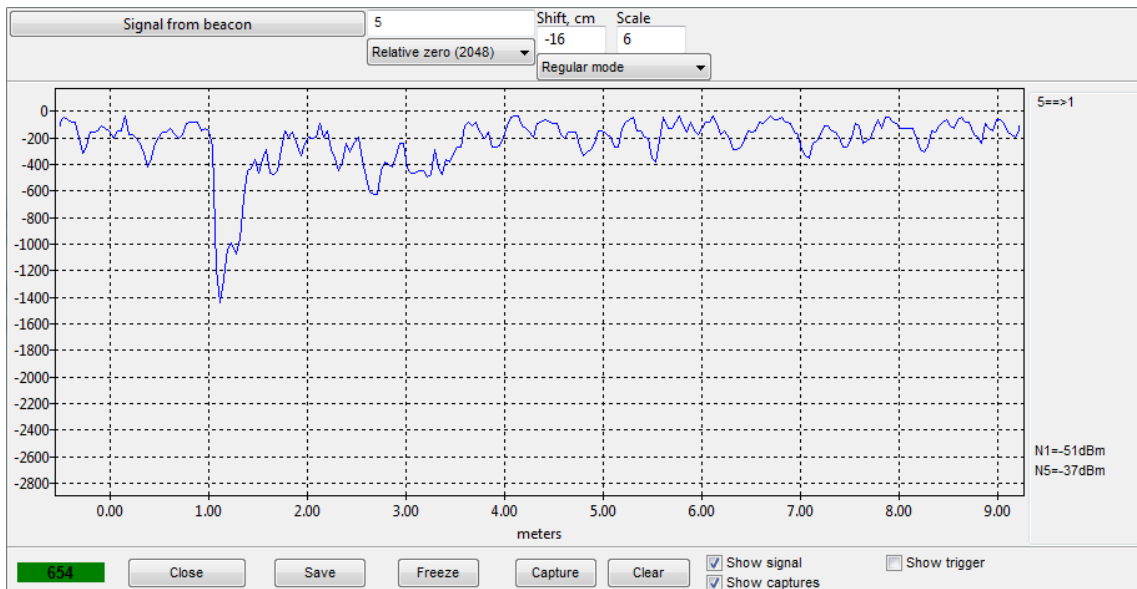
Annex

A.1 - Ultrasound noise Test for 1 m distance.

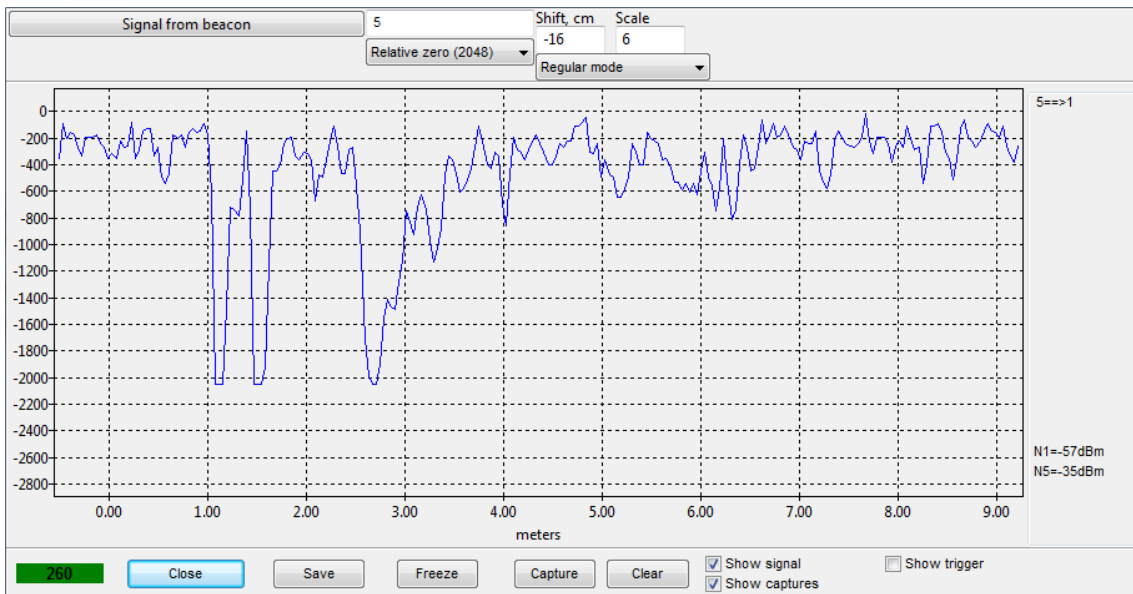
A.1.1 Receiver amplification: 1200



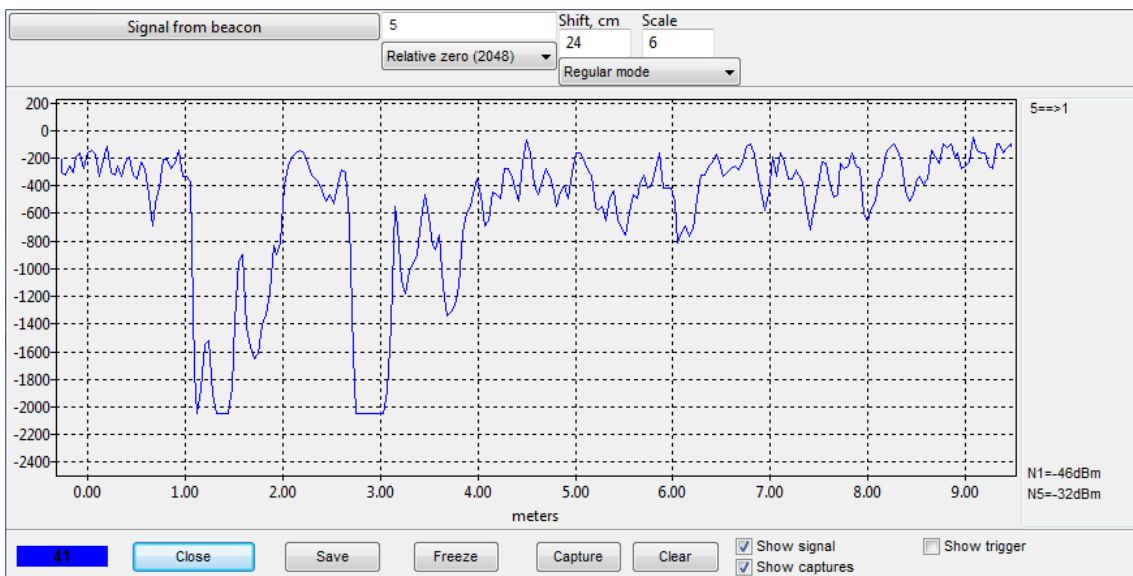
A.1.2 Receiver amplification: 1500



A.1.3 Receiver amplification: 2000

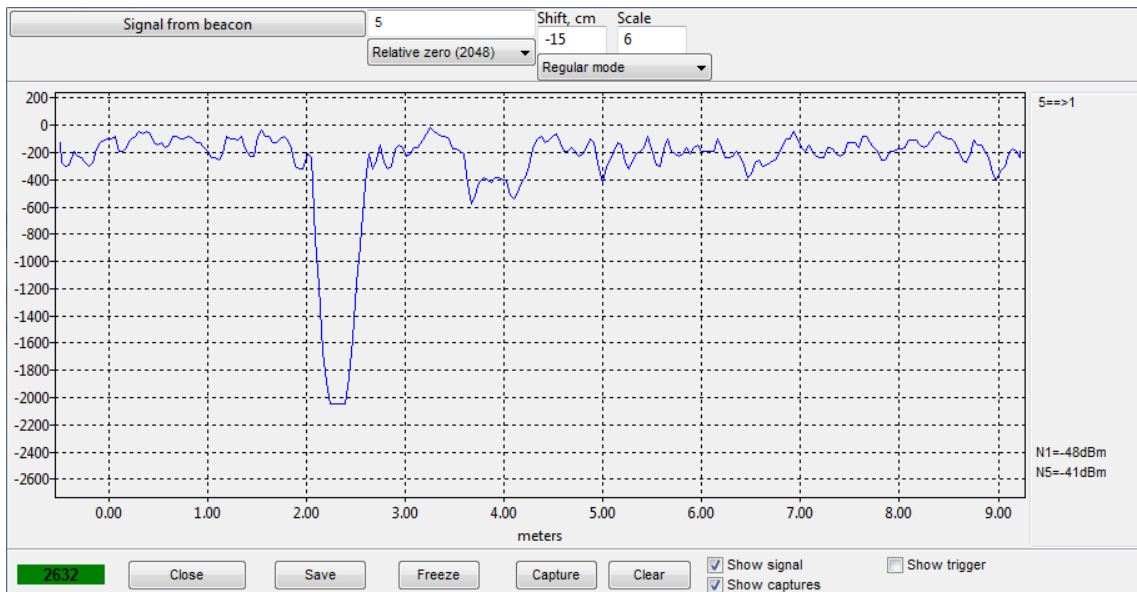


A.1.4 Automatic Gain Control

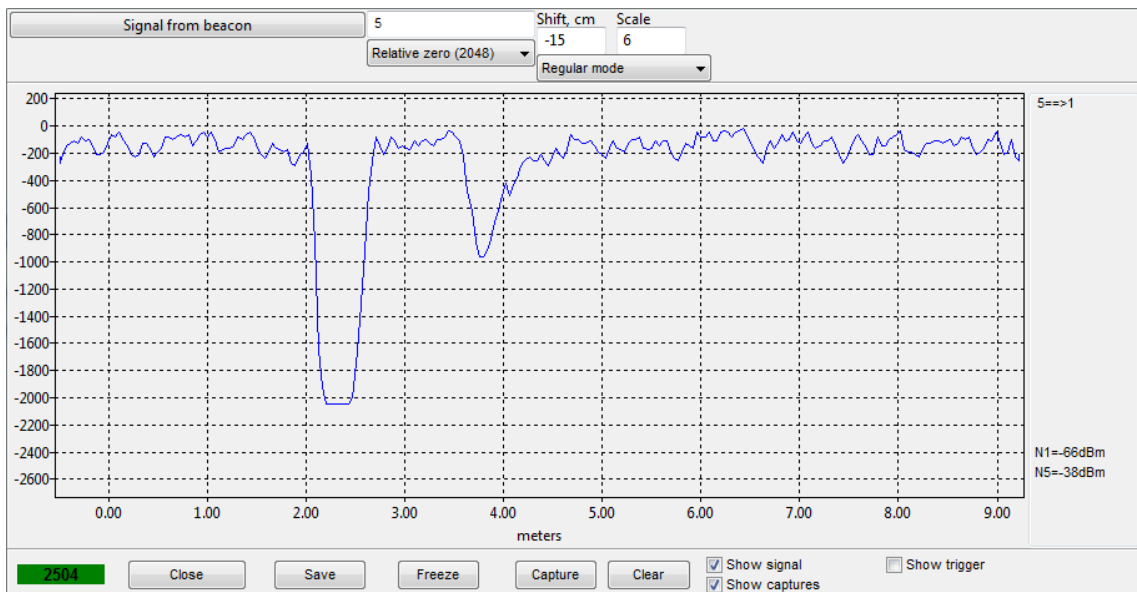


A.2 - Ultrasound noise Test for 2 m distance.

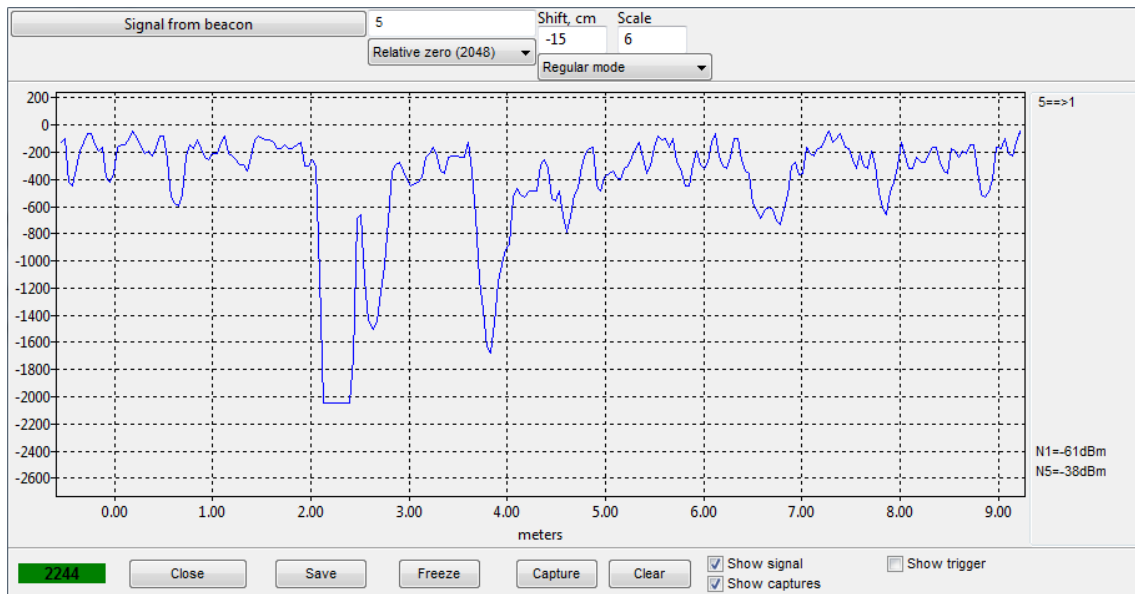
A.2.1 Receiver amplification: 1200



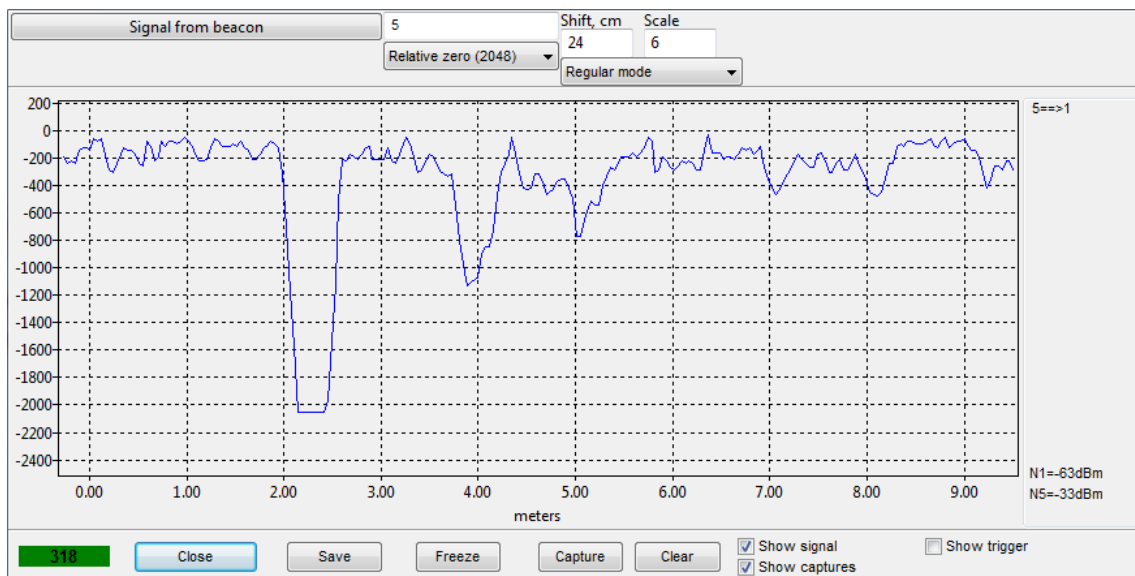
A.2.2 Receiver amplification: 1500



A.2.3 Receiver amplification: 2000

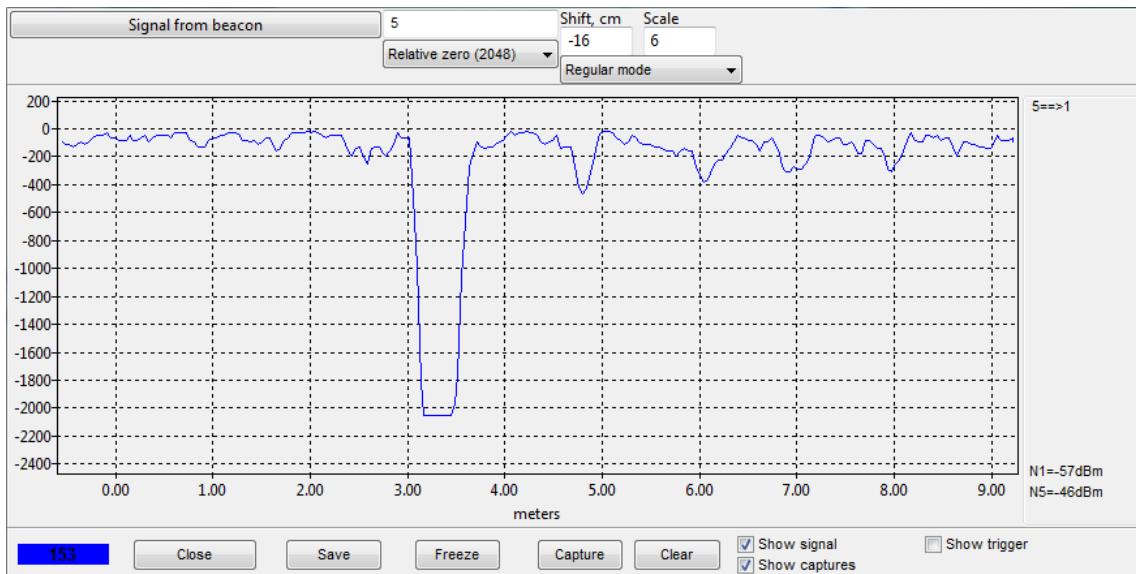


A.2.4 Automatic Gain Control

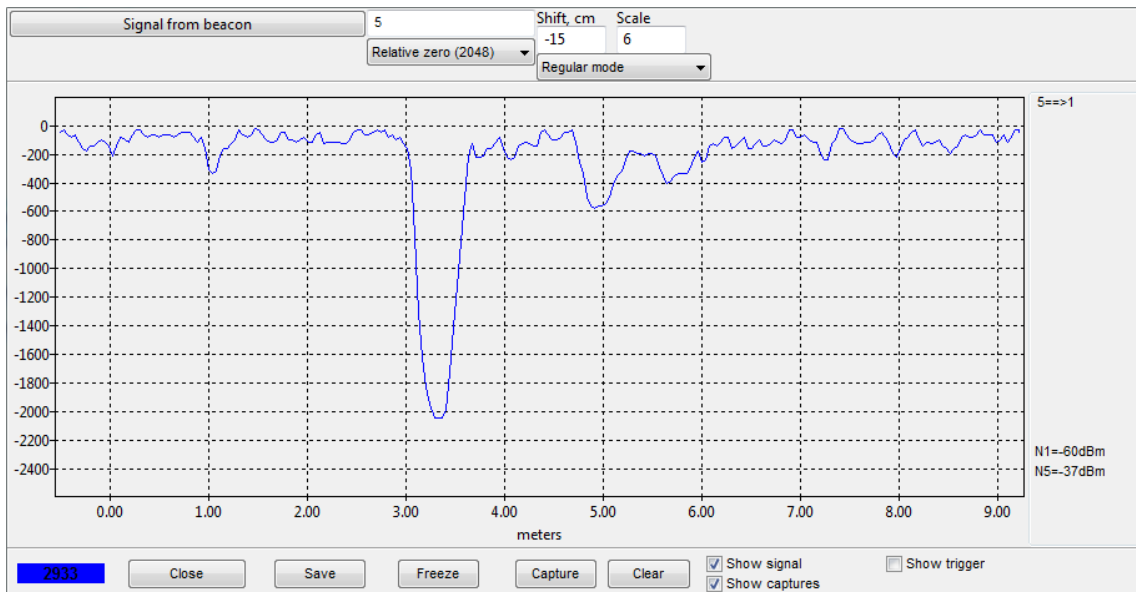


A.3 - Ultrasound noise Test for 3 m distance.

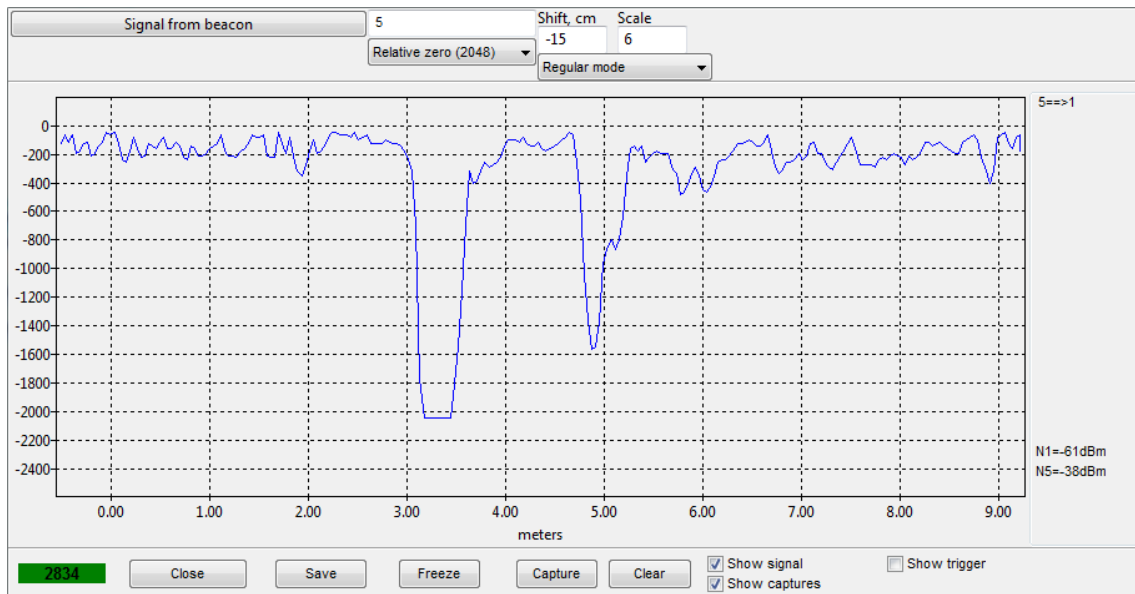
A.3.1 Receiver amplification: 1200



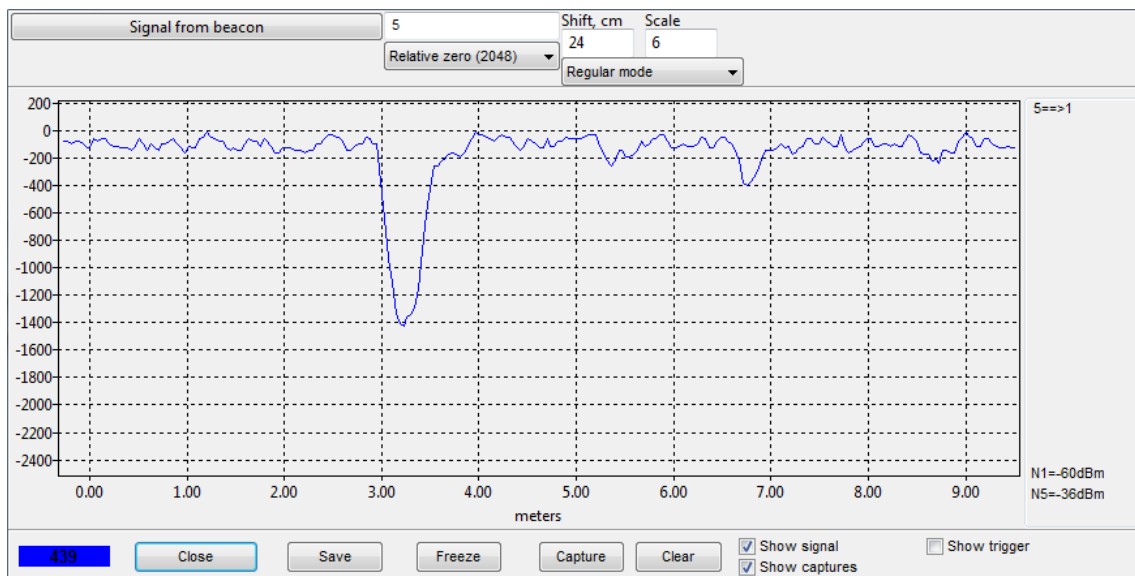
A.3.2 Receiver amplification: 1500



A.3.3 Receiver amplification: 2000

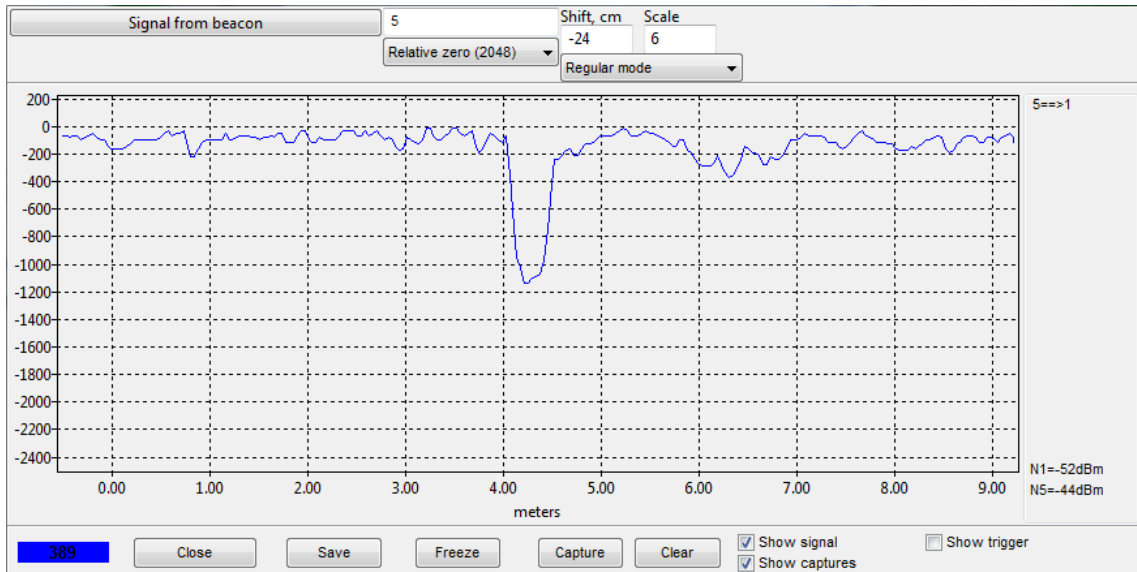


A.3.4 Automatic Gain Control

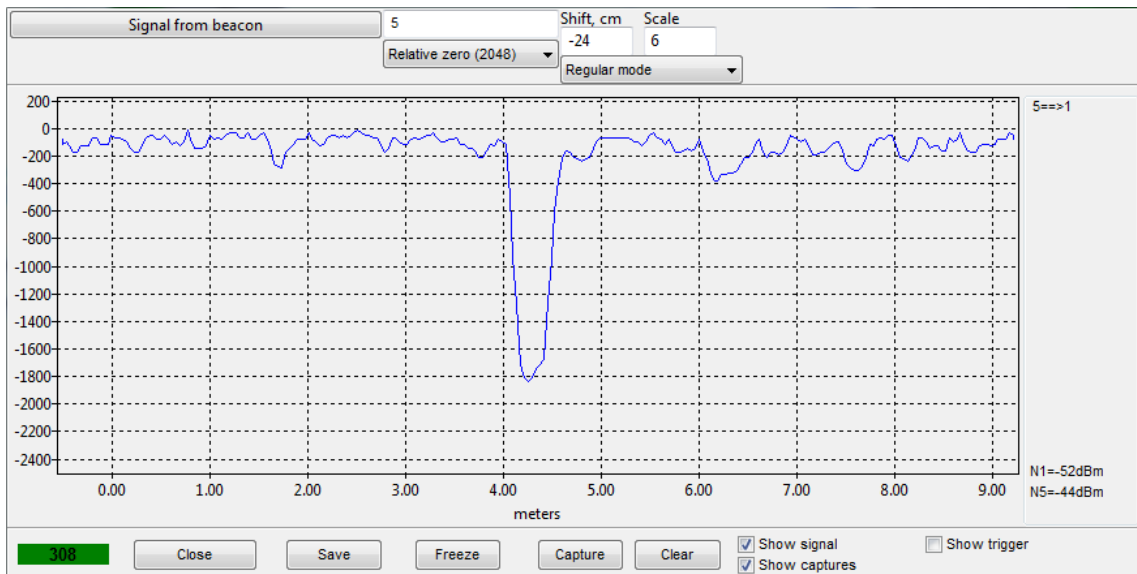


A.4 - Ultrasound noise Test for 4 m distance.

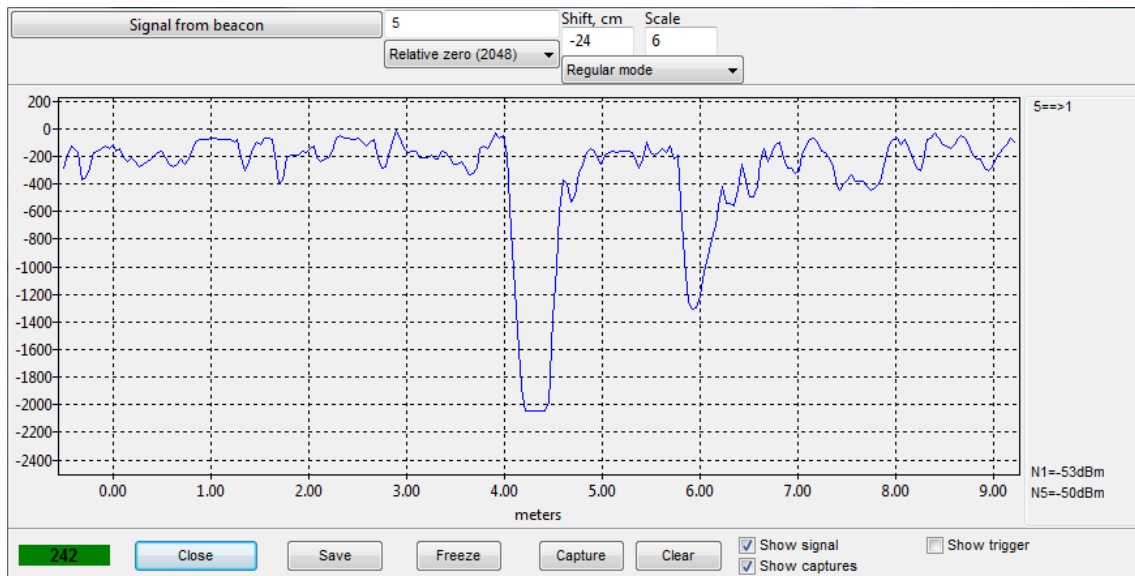
A.4.1 Receiver amplification: 1600



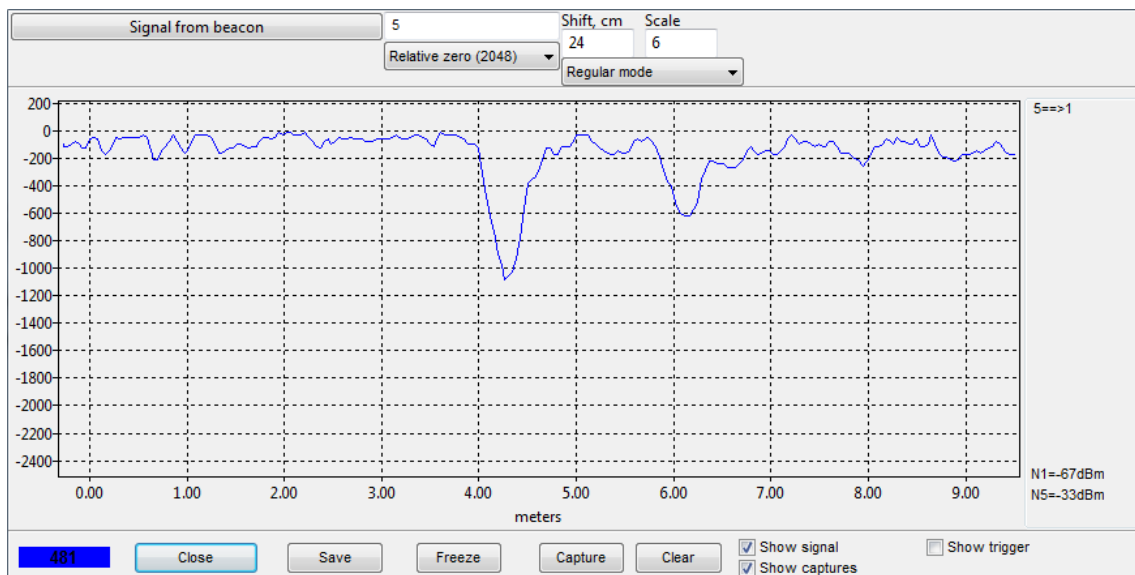
A.4.2 Receiver amplification: 1800



A.4.3 Receiver amplification: 2000

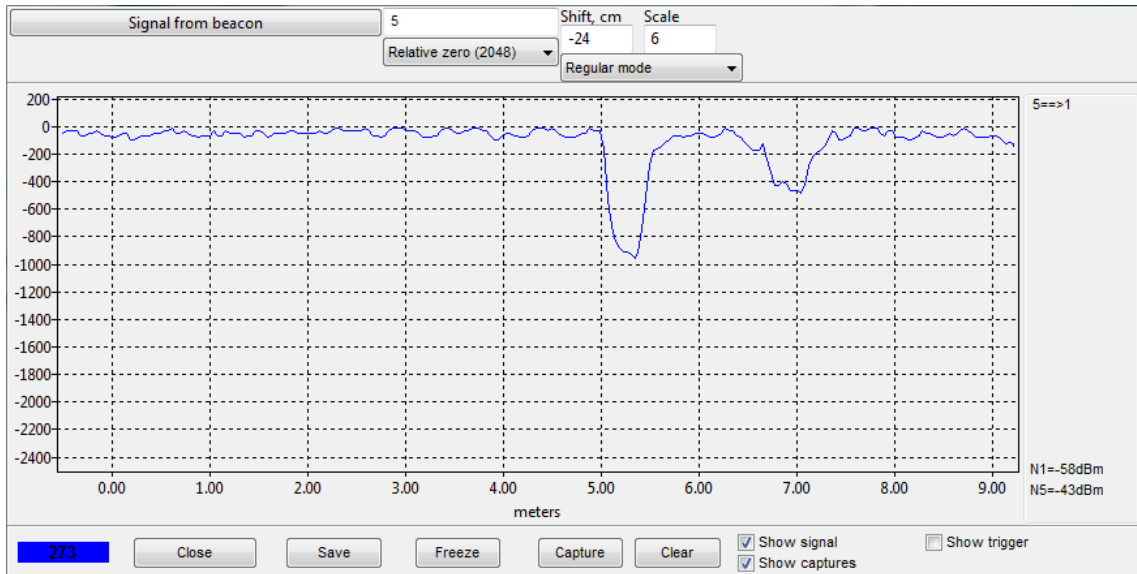


A.4.4 Automatic Gain Control

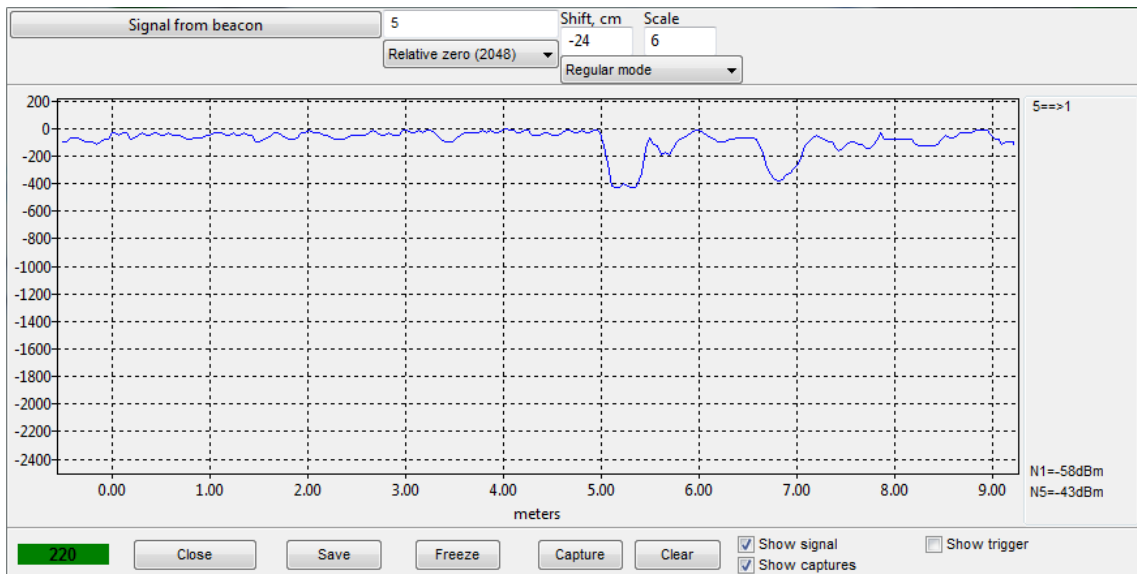


A.5 - Ultrasound noise Test for 5 m distance.

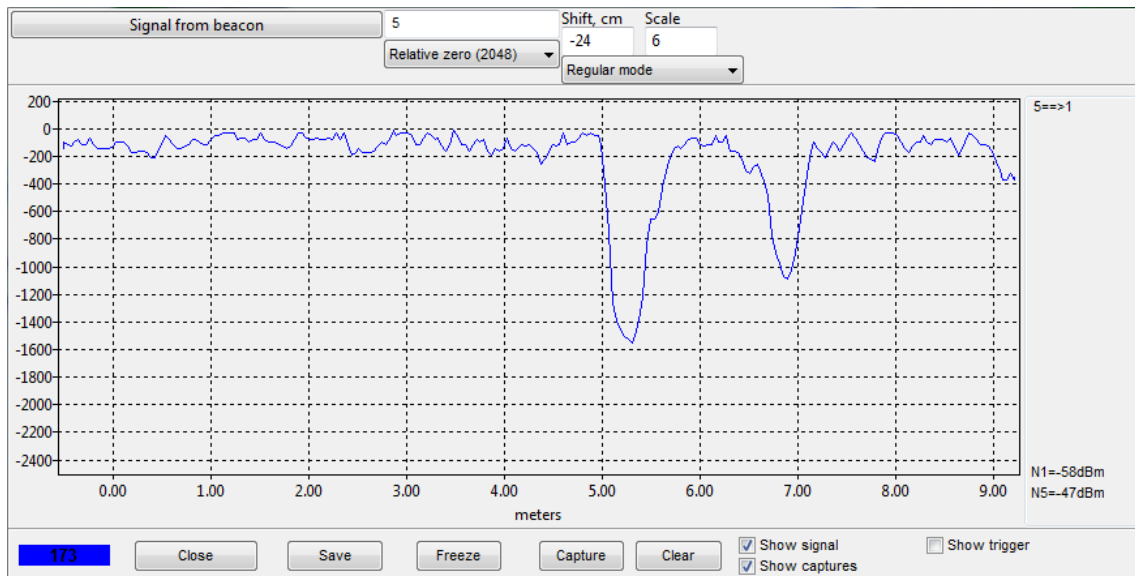
A.5.1 Receiver amplification: 1600



A.5.2 Receiver amplification: 1800



A.5.3 Receiver amplification: 2000



A.5.4 Automatic Gain Control

