Forside

Eksamensinformation
IGE550010C - EIT5 gruppe 512

Besvarelsen afleveres af

Thomas Holm Pilgaard
tpilga12@student.aau.dk

Anders Egelund Kjeldal
awitt12@student.aau.dk

Jacob Naundrup Pedersen
jnpe12@student.aau.dk

Jonatan Fuglsang Jensen
jfje13@student.aau.dk

Kasper Kiis Jensen
kkje13@student.aau.dk

Eksamensadministratorer

Hanne Høvring
hh@es.aau.dk
+4599408699

Bedømmere

Kirsten Mølgaard Nielsen
Eksaminator
kmn@es.aau.dk
+4599408761

Tom Søndergaard Pedersen
Eksaminator
tom@es.aau.dk
+4599408736

Jesper Lundgaard Skovfo
Censor
jlundgaard@hotmail.com
004524842375

Besvarelsesinformationer

Aalborg University

Autonomous Drone For Search And Rescue

Electronics and IT:
Fifth semester

Group:
15gr512

17. December 2015

AALBORG UNIVERSITET
STUDENTERRAPPORT

Fifth year of study at The school of information and communication technology
Electronics and IT
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø, Danmark
http://www.es.aau.dk

Theme:

Digital and Analog Systems Interacting with the Surroundings

Project:

P5-project

Projectperiod:

September 2015 - December 2015

Projectgroup:

15gr512

Participants:

Kasper Kiis Jensen
Jacob Naundrup Pedersen
Jonatan Fuglsang Jensen
Thomas Holm Pilgaard
Anders Langgaard Witt

Councelers:

Kirsten Mølgaard Nielsen
Tom Søndergaard Pedersen

Abstract:

Several incidents of death by drowning happens every year in Denmark and while the JRCC has a rescue rate of 94,6 % improvment is always necessary. Therefore it is examined how to increase the effectivnes of the SAR missions which leads to the following problem statement:

How can an electronic system be designed to reduce the search time during rescue missions at sea by increasing the width of the search sweeps?

An experimental solution was deducted for practical reasons whereby a quadcopter should be able to hold a relative position to a transmitter by using radio and ultrasound for distance measuring, PD controllers for movement and PPM modulation for communication.

Every module of the system were implemented except two of the PD controllers due to wrong input although they worked individually. This gave the conclusion that the system was able to hold a relative position in one of the three movement axes.

Copies: 8
Pages: 111
Appendix: 58
Completed 17-12-2015

Preface

This paper is drafted by group 15gr512, a fifth semester group consisting of five persons studying EIT at Aalborg university. The purpose of this paper is to document the work of the design and development of a control system for a quadcopter. The theme of the semester is "Digital and Analog Systems Interacting with the Surroundings" and started on September 2. with a submission deadline on December 17. The group has parallel with the project participated in the following courses: "Signal Processing", "Modeling and Control" and "Communication in Electronic Systems".

This paper investigates the possibility to improve the search time during rescue mission at open sea by using an electronic system. An analysis of such system functionality and technology become the basis for a requirement specification. A system design with the interfaces between the modules leads to the development of a prototype, which is held against the requirement specification in the final accepttest.

The group would like to thank the following person for help regarding Vicon:

- Karl Damkjær Hansen, post doc at Aalborg university.

The group would like give a special thank to:

- Emil Mürer who spent time and rescources designing and creating plastic conical cones.

The figures in the paper is produced by the group unless a source is specified. Sources are indicated by [author,year] and can be found in the bibliography. Appendix is indicated by A.number or [Appendix/filename] on the CD. The paper is structured in chapters and sections, every figure, table, equation and code is seperately numbered continuously. Figures can be diagrams, flowcharts and graphs. The following is placed on the CD:

- Altium files
- Code
- Datasheet
- LTspice files
- Matlab files
- Video
- Component list

Kasper Kiis Jensen

Jacob Naundrup Pedersen

Jonatan Fuglsang Jensen

Thomas Holm Pilgaard

Anders Langgaard Witt

Contents

Introduction 1

Water is a natural installment of every persons daily life. We drink it, we bathe in it and we get transported on it. 71 % of the earth is covered in water and more than 50 % of the worlds population is living closer than three kilometers from sea, lake or river [Matti Kummu, 2015]. With so many people living close to water, fatal accidents like drowning is bound to happen, but how big is this problem actually in a developed country like Denmark?

In Denmark 816 people drowned between 2001 – 2013. The cause of drowning varies between unintentional (56 %), suicide (34 %), homicide (0,3 %) and undetermined (10 %). These deaths occured at different places, where the most prominent were at sea (25 %), harbor (24 %) and beach/coast (20 %) as shown on table 1.1.

	Avg 2001-2009	2010	2011	2012	2013	Total	%
Sea	18.6	5	12	8	8	200	25
Harbour	16.9	10	10	15	12	199	24
Lake	5.4	5	6	7	5	72	9
Beach/coast	12.7	8	8	20	14	164	20
Swimming pool	2.2	0	1	1	1	23	3
Bathtub	1.4	0	2	0	1	16	2
Bridge	0.4	0	0	0	0	4	0.5
Stream/river/canal	5.4	7	3	5	1	65	8
Other body of water	5.1	2	0	0	6	54	7
Unspecified	0.7	11	1	1	0	19	2
Total	68.9	48	43	57	48	816	100

Table 1.1: Unintentional drowning deaths 2001-2013 by location and year [Trygfonden, 2013].

This project will only deal with the unintentional drownings at sea. This is mostly where the Joint Rescue Coordination Center (JRCC) Denmark is called upon to assist in locating and rescuing people in distress. In 2012 the JRCC was called upon 373 times to assist in sea rescue missions, where they succeeded to rescue the distressed person or persons in 94,6 % of the cases [Forsvarsministeriet, 2013]. This means that approximately 20 rescue missions failed. Although 94,6 % is a high success rate, it should still be looked at what can be done to raise this percentage to reduce the amount of lost lives.

If the person in need of rescue is located, it is not an issue to get him or her out of the water, however if the location of the missing person is unknown the situation is more dire. If this is the case time becomes an important factor in the success of the rescue mission. Therefore it is relevant to look at how long a person can survive in water.

1.1 Survival time in water

Time is very limited when it comes to naval rescues as the risk of hypothermia is very high. This is because the body is cooled more than 25 times faster in water than in air. It is therefore essential to be swift when rescuing a person at sea. There are three states of hypothermia and which state the distressed is in, depends on the core temperature of the body.

Mild hypothermia occurs when the body's core temperature drops below 35 °C. In this state the person is still able to help themself, but will experience shivering, numbness and loss of dexterity.

Severe hypothermia occurs if the core temperature drops below 32 °C. At this point the shivering may stop, however the other symptoms will increase, and the person will experience decreased cognitive functions, which may lead to confusion, loss of reasoning and denial. This may cause the person to resist help if complete loss of consciousness has not occurred.

Critical hypothermia occurs if the core temperature drops below 27,8 °C. In this state the person will be unconscious, have little to no apparent breathing and a significantly slower pulse. The skin will change color to a blueish-gray color and pupils will be dilated. It is also in this state that the person risks cardiac arrest.

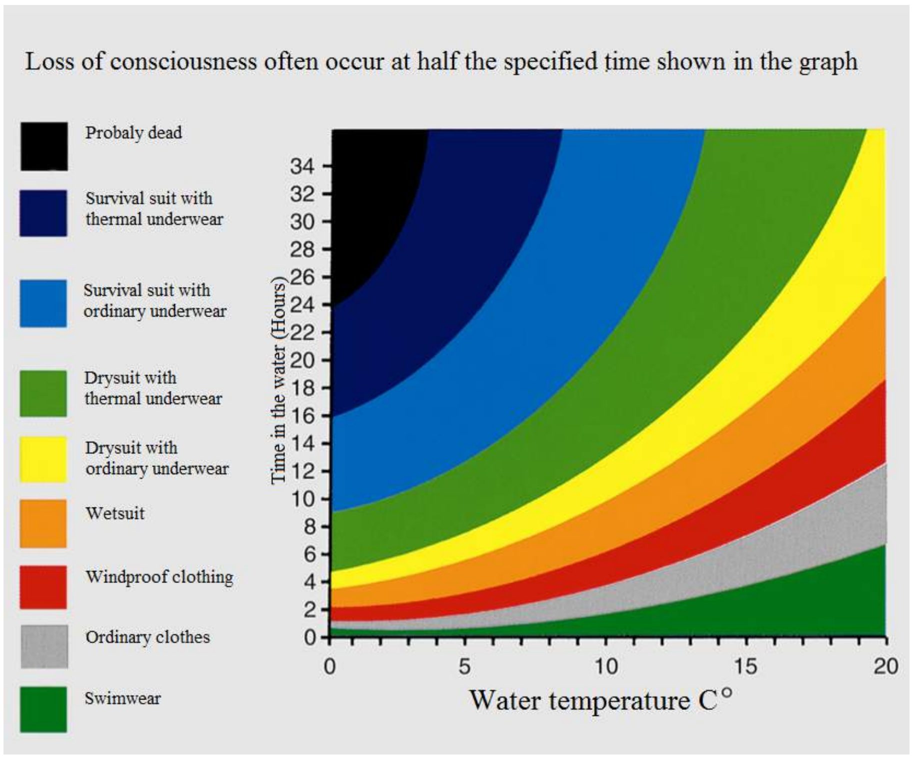


Figure 1.1: Plot over survival times at different temperatures [Forsvarskommando, 2014].

As seen on Figure 1.1, temperature and clothing are among the most important factors when it comes to survival times. The amount of time the rescuers have can vary from less than an hour in very cold waters to a few days days in warmer waters.
If the water is 10 °C and the person is treading water it is estimated that the rescuers have

around 1-2 hours before the distressed passes out or becomes exhausted. The expected time of survival in such situation varies from 1-6 hours.

Due to the high search efficiency, a helicopter will often be used on rescue missions. This means that rescuing often will require the involvement of the royal danish air force. The two military helicopter units involved in search and rescue (SAR) missions is Squadron 722 and 723. The main responsibility of squadron 722 is SAR missions and thus the most relevant to look into. Due to the time critical element of its job, squadron 722 has a maximum response time of 15-30 min depending on the time of day. The helicopter model this unit uses is the Westerland EH-101 and is shown on figure 1.2. They are often referred to as Merlin helicopters, and have been in operation since 2007.

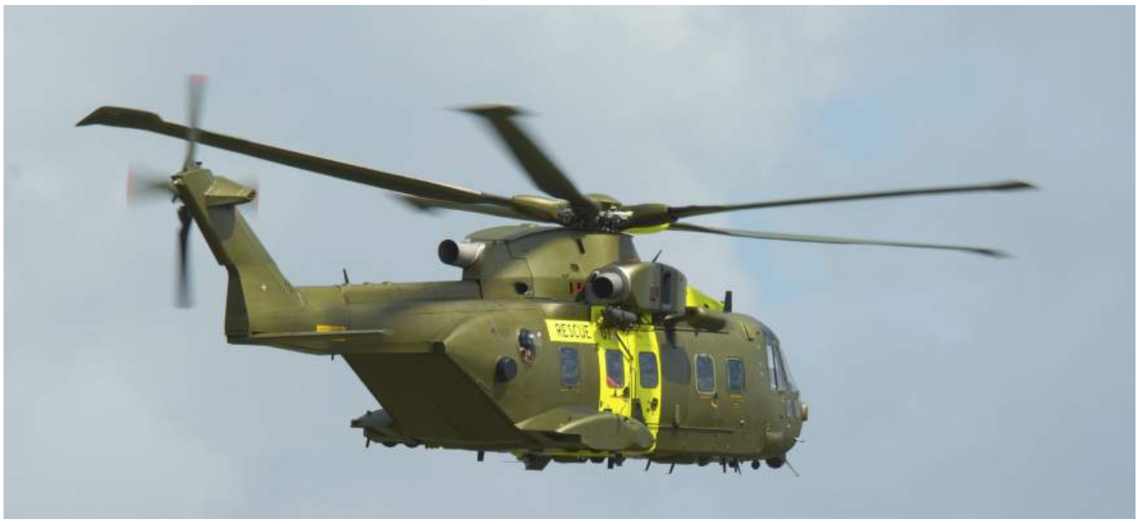


Figure 1.2: Westerland EH-101 helicopter [KrigerenDK, 2013].

The typical operation speed is around 240 km/h, but the helicopter is capable of achieving speeds of up to 274 km/h with a maximal flight duration of 5 hours. The personal aboard a helicopter at SAR missions consists of 5 crew members and a medic. Both the helicopters supplied to squadron 722 and 723 comes equipped with forward looking infra-red (FLIR) cameras, which makes them capable of performing SAR missions at night [Forsvaret, 2014]. Due to the time critical nature of a distressed at sea, it will be natural to look at how to improve the rescue success rate.

1.2 Rescue success rate

When calculating the success rate of a naval rescue mission, factors such as weather, search pace and fatigue of the rescue personal comes into play.

In order to secure the best possible outcome of the mission, as much information as possible is needed. In cases where the persons whereabout is either unknown or a sufficiently amount of time have passed since he or she was last spotted, a rescue plan must be made. The plan consists of calculating the estimated location of the person, calculating search area, effort allocation and election of an on scene coordinator (OSC). Often the person in distress won't be near his or hers last known location, as they can drift away from the scene of accident by currents.

Based upon calculations, an estimated search area is set up, in where the person is assumed to be and it is within this area that the search will be done. The search is done with a calculated sweep width to ensure detection of a distressed and a decided spacing in the search pattern to either give a high probability of detection (POD) or the possibility of searching a larger area quicker with a reduced POD.

The relation between the spacing and the sweep width is called the coverage factor. The coverage factor represents how well the search area is covered, and it can be found using equation 1.1:

$$C = \frac{W_c}{S} \qquad [\cdot] \quad (1.1)$$

Where:
C is the coverage factor [·]
W_c is the corrected sweep width [m]
S is the spacing [m]

The sweep width is a representation of how far from the helicopter the rescuing unit is able to detect the distressed. If the distressed is within the sweep width range he or she will likely be able to be detected by the rescue personal. Figure 1.3 illustrates the sweep width of a helicopter and it is shown how the sweep width represents the area where the person is likely to be detected, and not the maximum detection distance.

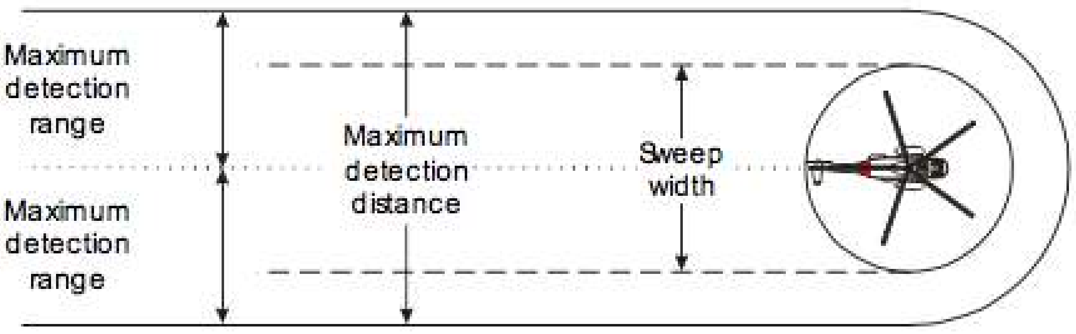


Figure 1.3: Illustration of Sweep width versus maximum detection range [Redningsrådet, 2013].

The corrected sweep width can be found by the following equation:

$$W_c = W_u \cdot F_w \cdot F_v \cdot F_f \qquad [\text{km}] \quad (1.2)$$

Where:
W_c is the Corrected Sweep Width [m]
W_u is the Uncorrected Sweep Width [m]
F_w is the Weather-correction Factor [·]
F_v is the Velocity-correction Factor [·]
F_f is the Fatigue-correction Factor [·]

These factors can be found in lookup tables, and will factor in the most important variables [Redningsrådet, 2013].

The spacing represents the length between the sweeps of the rescuing unit. This will under normal circumstances be matched to the sweep width in order to cover the search area in the most efficient way. A coverage factor of one is the most optimal, and means that the unit sweep width matches the spacing of the sweeps, thus optimally covering the area. If the situation calls for it, the spacing can be increased, this could for instance be if the sun is about to go down. This will decrease the search time, at the cost of how well the area is swept [Redningsrådet, 2013]. This will influence the coverage factor which furthermore impacts the POD as shown on figure 1.4.

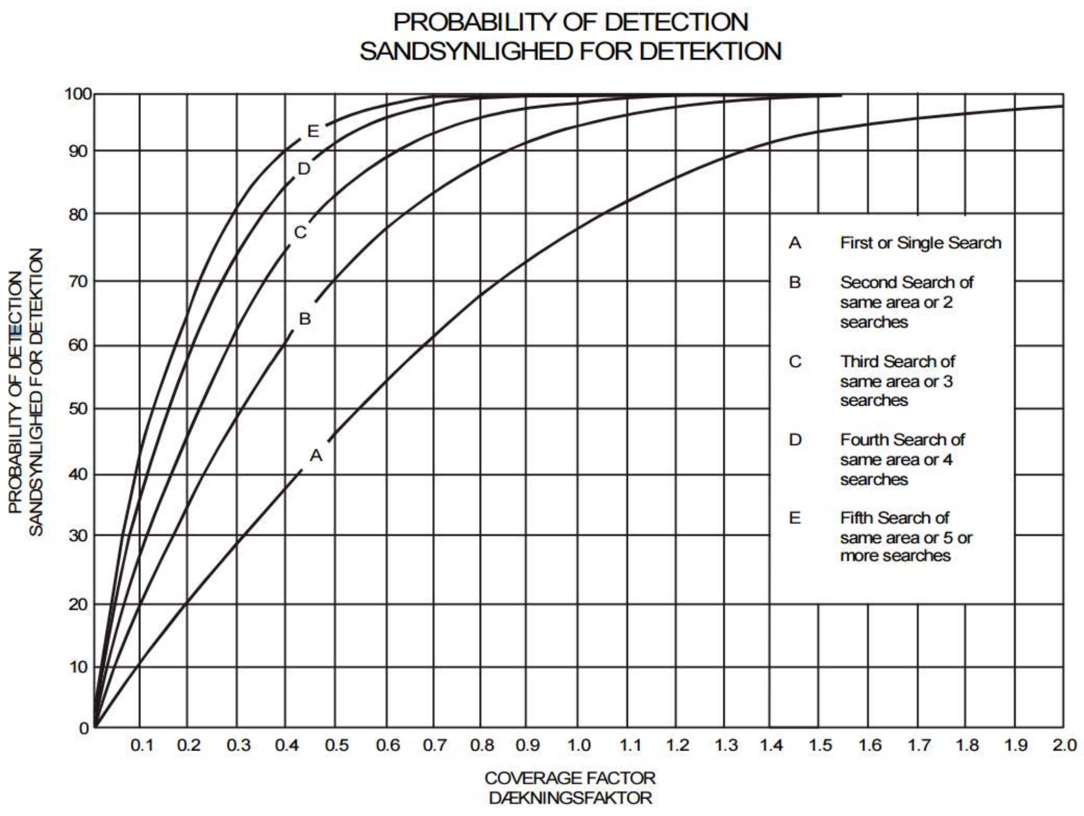


Figure 1.4: Graph of POD [Redningsrådet, 2013].

The POD can be improved by the amount of sweeps over a certain area at the cost of time. Also the sweeping can be done in certain ways to accommodate the conditions of the mission.

1.3 Search Patterns

When the required factors are calculated, the search can begin. Depending on the situation, different strategies can be used. Each strategy consists of sweeping a defined area by a certain pattern, eventually leading to covering the entire area.

Parallel search

One of the strategies is the parallel search. This method consists of covering the area by sweeping back and forth. For each sweep, the rescuing unit will move one lane, defined by the spacing found by equation 1.2. An illustration of the parallel search method is shown on Figure 1.5.

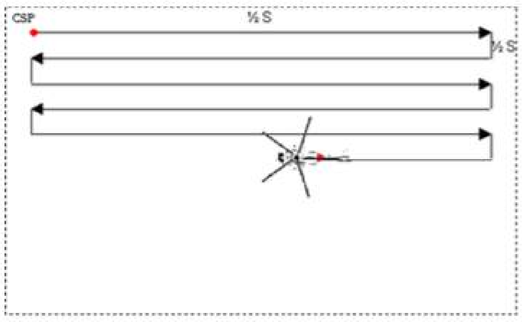


Figure 1.5: Illustration of the parallel search [Redningsrådet, 2013].

This method will mostly be used if the search area is relatively large, and the precise location of the distressed is unknown.

Track line Search Return (TSR)
The track line search return is different from the parallel search strategy, as it follows a certain route, instead of searching a wide area. It is therefore better suited for searching for a missing person or vessel, if only the route that the distressed have travelled is known. Instead of searching an approximate area, the rescuers sweep the same route as the distressed, back and forth. An illustration of this can be seen on Figure 1.6.

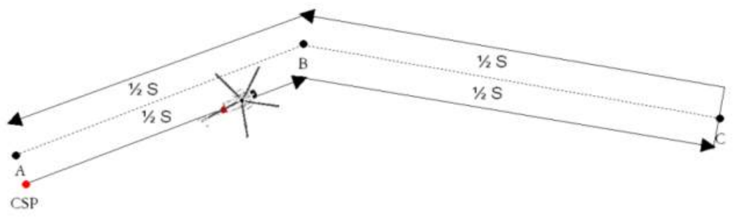


Figure 1.6: Illustration of the Track Line Search Return [Redningsrådet, 2013].

Two of the defining characteristics of a successful rescue mission is speed and efficiency. The rescue helicopter will always be present at the most probable location of the missing person. Therefore a way to improve the speed and effectiveness of the mission could be to increase the width of the sweeps. This would result in each sweep covering a larger area, which in turn would mean fewer sweeps, resulting in the search area being covered faster.

From this the following problem statement can be deducted.

How can an electronic system be designed to reduce the search time during rescue missions at sea by increasing the width of the search sweeps?

This will be used in the following to design a system capable of achieving such goal.

Functionality analysis 2

In this chapter the functionality analysis of the system will be made. The purpose of this analysis is to create a basis for designing a system capable of fulfilling the problem statement. In order to increase the sweep width of the helicopter it is required to have a system scanning further than the calculated sweep width. Because the sweep width is relatively large (refer to appendix A.2) it is considered impractical to place the system on the helicopter. This means an independent system is needed. The system should be able to follow the helicopter around, increasing the sweep width effectively and communicate with the helicopter. In such case a flying drone would be relevant.

A system flying autonomously alongside the helicopter could potentially increase the sweep width. Such a system is shown on figure 2.1. The system is required to maintain a specified distance to the helicopter, at a certain height. The defined distance is orthogonal on the helicopter, which makes angular adjustment a necessity, in order to ensure the right area is searched. This will enable the system to extend the sweep width of the helicopter by searching the area beneath it.

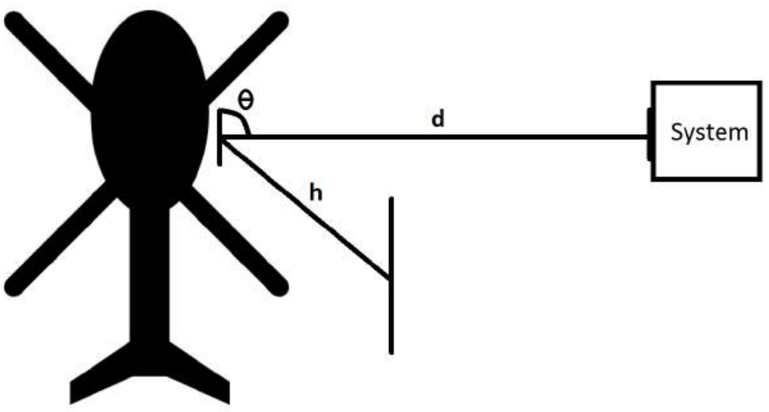


Figure 2.1: Concept of system, where (θ) is the angle, (d) is the distance and (h) is the height

The platform has to be able to collect data which then must be send to the rescue personal for review. This means that the system will need some form of sensor able to detect persons in the water.

This means the system will consist of two main parts:

- Platform.
- Control system.

The platform is the basis of the system, it will contain the functionalities involving detection of the distressed and the drone itself. The control system is the part controlling the relative position with regards to the helicopter. This includes height, angle and distance adjustment. In the next section the platform for the system will be analysed.

2.1 Platform

This section contains the analysis of the elements involved in creating the system platform. The platform analysis consists of three blocks (refer to figure 2.2):

- Drone selection.
- Locating the distressed.
- Communicating to the rescuers.

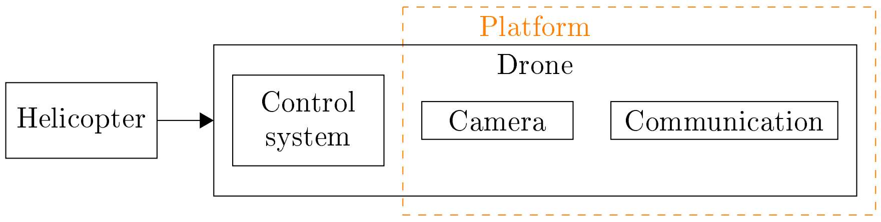


Figure 2.2: Overview of the system.

These blocks each represents a functionality which will need further analysis .

Drone selection

In this section it will be examined which drone type will be best suited for SAR missions. The parameter which will be analyzed is movement.

To analyze movement it is important to look at how a helicopter moves. A helicopter can move in three different axes which is x, y, z and can also rotate around the z axis, which is shown on figure 2.3.

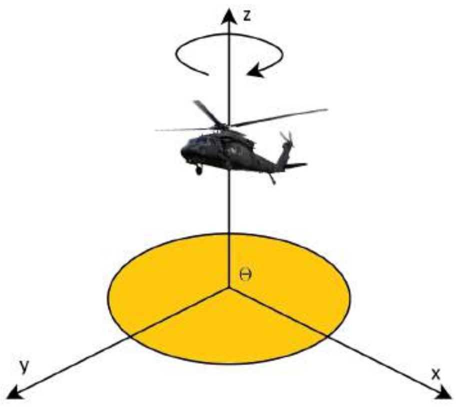


Figure 2.3: Illustration of how an helicopter can move from a stationary position. The yellow area illustrates the possible direction in y and x axis the helicopter can change direction to [Theintentionallife, 2015].

The key difference between an ordinary air plane and a helicopter is that a helicopter has the ability to hover at a fixed point or move at slow speeds. This type of movement will be done in cases where the rescue personal wants to investigate something they have spotted, or if they have located the distressed. This is what makes the helicopter the ideal transport for rescue missions.

In the following section a quadcopter and a deltawing will be analyzed in order to see which will be the best solution for sweeping an area with a rescue helicopter.

Quadcopter

A quadcopter is a device, which have four rotor blades making it capable of flying. A quadcopter will be able to investigate a large area from the air, which gives it a good overview of the search area. It is able to achieve a speed of around 96 km/h. [Turbo-ace, 2015]

Figure 2.4: An example of a quadcopter [Turbo-ace, 2015]

A quadcopter as seen on figure 2.4 has the same movement abilities as a rescue helicopter which can be seen on figure 2.3.

Deltawing

A deltawing is a device constructed as a wing with a single propeller blade mounted at its rear, making it able to fly with a top speed of 160 km/h [TuffWing, 2015]. A deltawing could search an area from the sky, which will give it the same overview as a quadcopter.

Figure 2.5: An example of a deltawings [TuffWing, 2015].

While the quadcopter is able to mirror the movement of the helicopter the deltawing cant, since its movement is restricted as it needs to maintain a certain speed. An illustration of the deltawings yaw turning pattern is shown on figure 2.6. Because it is not capable of hovering in mid air, it has to fly in circles or in eight figures to accomplish a state hovering.

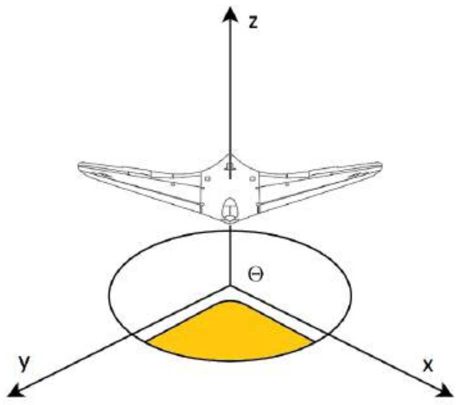


Figure 2.6: Illustration showing manoeuvrability of a deltawing. The yellow slice illustrates the angle around the z axis the deltawing can turn relatively to a current direction of the deltawing [Pixabay, 2015].

Both of the devices has a good overview and can sweep a large area. The deltawing has a higher top speed at 160 km/h therefore it will be able to follow the rescue helicopter at higher speeds. The deltawing will therefore be the best solution. Furthermore the movement of the drone is not required to mimic the helicopters movement completely because of the large sweepwidth, which makes its limited movement less of an issue. However it is not possible to test the deltawing in this project due to the legislation in Denmark, which prohibits flying with drones outdoors without certain permits [Trafikstyrelsen, 2014]. Furthermore Aalborg University does not have the required facilities needed to test a deltawing indoor, therefore the choice falls on the quadcopter because the facilities to test it indoor is available.

Locating the distressed

The possibility to locate a person can be done with different technologies, such as RGB-camera, FLIR-camera or sound waves. Some of the problems surrounding locating a person in the sea is that the weather conditions are rarely optimal, this makes waves covering the distressed, strong winds and bad vision something that make the locating difficult.

RGB-Camera/FLIR-Camera

The use of a RGB-camera in daylight could be an option because it would be able to detect distinct colors like the color of a life jacket even in high sea. But at night time or in heavy fog the effectiveness of a RGB-camera would be negligible. In such circumstances the use of a FLIR-camera would be more effective because this camera type detects heat signatures. Detection would then be easy in low sea because the difference in temperature between the water and the distressed´s head is large. This technology also allows the rescue personal to search at night, but not at high sea because the waves could cover the distressed. An example of the difference between a RGB-camera and a FLIR-camera is shown on figure 2.7.

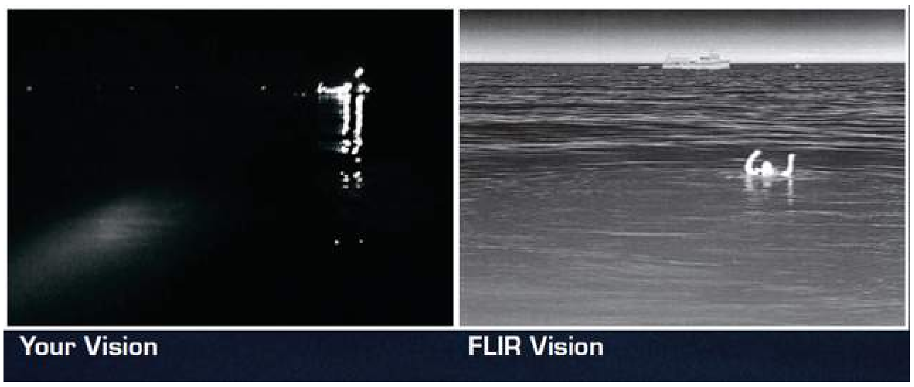


Figure 2.7: The difference between RGB- and FLIR-camera [landfallnavigation, 2015].

The detection of the distressed with a camera could be done autonomously with image recognition. The drone should if the mounted camera detects a color or heat signature, stream video or send pictures back to the rescue personal whome will then take action depending on what they deem necessary. Alternatively the drone could have a permanent video stream to the rescue personal and the image recognition would then assist in locating the distressed.

Sound waves

The location of a distressed could also be accomplished by pin pointing the sound the person makes when calling for help. By measuring the time difference from more than one microphone, it is possible to triangulate the position of the distressed this concept is shown on figure 2.8.

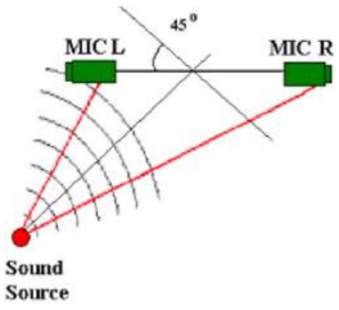


Figure 2.8: Illustration showing two microphones receiving the same sound waves at different times [resna, 2003].

The problem with sound detection is that different weather conditions makes the voice of the distressed travel less distance than under optimal circumstances. Another problem is that the distressed would need to produce a sound either constantly or at a short interval which would make it possible to locate him or her, and as exhaustion could occur rather quickly, sound seems like an improper solution as a standalone detector.

Sound could be used in a combination with another locating device to communicate with the person in distress when located, but the use of sound detection as an optimal detection device would be less than ideal. Therefore the optimal solution would be a combination of a RGB-camera and FLIR-camera, to be able to increase chances of locating the person or persons in distress.

After having decided on a solution to detect people in distress it will be necessary to communicate this information to the rescue personal. This will be done in the following part.

Communication to the rescue personal

An important part of locating the distressed is to send the information back to the rescuers for review. As the searching will be done with some sort of camera, it is important to ensure the communication has enough bandwidth to secure an acceptable frame rate.

To ensure the rescue team detects the distressed they are searching for, an image or video stream could be sent to the rescue personal, which then confirms whether it is a person or an object. A real time video stream would provide feedback to the rescue personal, at the cost of more bandwidth needed for the signal to be transmitted.

A communication system could use satellite for long range, but it requires a lot of power. This method of communication uses geostationary satellites, which provides a bandwidth of up to 50 Mbps [Golding, 2011]. Due to the heavy toll on the battery, this option is to be avoided as the battery capacity of the drones is limited as it is [Network-World, 2014].

It has been chosen to use a quadcopter as a system platform because it is possible to test the quadcopter within the parameters that are available to the project group. Furthermore it has been decided to focus on the control system, therefore the surveilance and communication part will not be investigated any further. In the next part the control system for the quadcopter will be analysed.

2.2 Control system

This section will focus on the position control as shown on figure 2.9, by describing a possible way to construct a control system making the drone capable of keeping a position relative to the helicopter. This project will be limited to the position control of the quadcopter relative to the helicopter, and not the flight control of the quadcopter. The flight controlling will be done by an external controller.

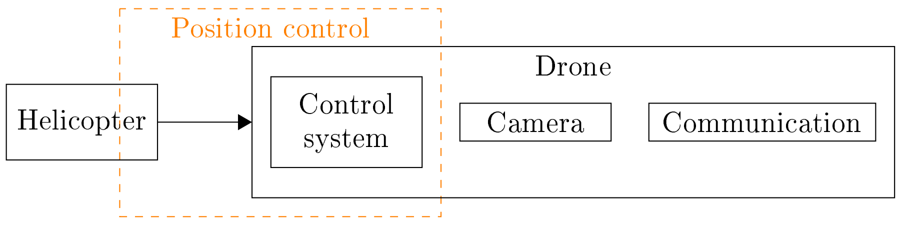


Figure 2.9: Overview of the system.

On figure 2.10 the relation between the helicopter and the quadcopter, is illustrated as separate coordinate systems.

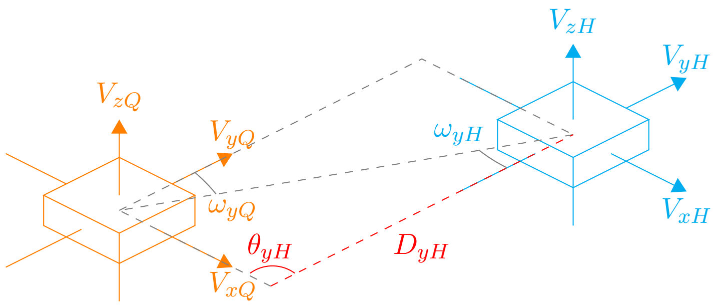


Figure 2.10: Illustration of relation between the helicopter (blue) and the quadcopter (orange) and the desired position for the quadcopter (red). Where V is the velocity, ω is the angle-velocity, D is the distance and θ is the angle. The axes is represented by x, y and z.

For the system to hold a position relative to the helicopter, a control system is needed. The system is required to maintain the distance and the angle to the helicopter (refer to figure 2.10). Furthermore the system is also required to maintain an altitude, which is determined by the optimal condition for scanning. The control system is separated into three parts:

- Distance controller as shown on figure 2.11.
- Angle controller as shown figure 2.12.
- Altitude controller as shown figure 2.13.

Figure 2.11 shows a block diagram for the distance controller, which keeps a preset distance between the system and the helicopter. This functions by setting a distance reference which is compared to the actual measured distance. The processed control signal is then modulated to a signal the drone flight control accepts.

The signals from the flight control is distributed to the AMP block where is it amplified and then distributed to the individual motors of the quadcopter. Movement such as lift, rotation and acceleration is done by in- or decreasing the velocity of one or several of the motors. The difference in velocity between the drone and the helicopter needs to be translated into a distance, which is done by the integrator block.

To ensure the system keeps the reference distance, a closed feedback loop is used. The system takes a measurement and sends it back to the microcontroller which calculates the adjustment needed to keep the reference distance. In the feedback loop, a distance sensor block will make an analogue measurement, and send it to the microcontroller. To avoid interferences, a filter could be added to either the microcontroller block or the sensor block. The microcontroller uses an analogue to digital converter (ADC) to convert the signal from the sensor, to an integer value. The integer is then converted into a float value which can be compared to the reference.

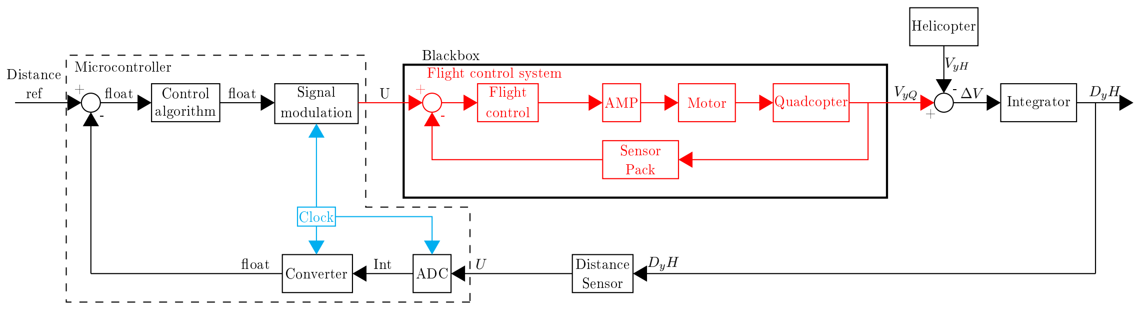


Figure 2.11: Blockdiagram illustrating the function of the control system relative to keeping a distance to the helicopter.

Figure 2.12 shows a block diagram for the angle controller. The function of the angle controller is similar to that of figure 2.11, The difference between the two is that the angle relative to the helicopter is desired instead of the distance. Therefore a sensor block which can calculate the angle, from other measurement values, is called angle sensor.

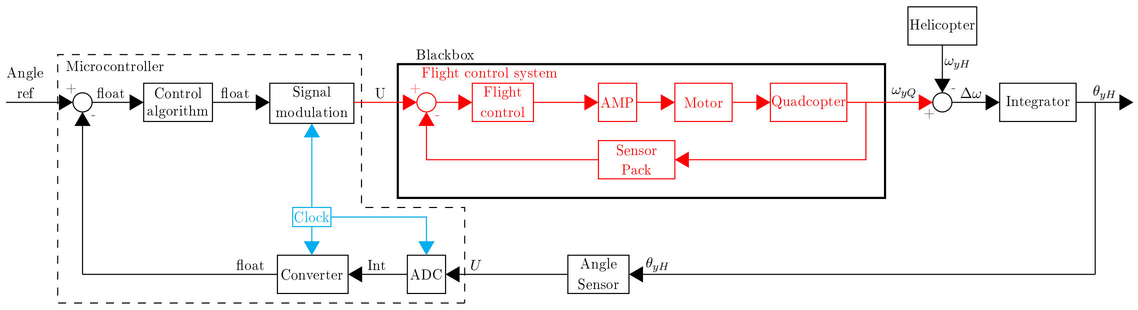


Figure 2.12: Blockdiagram illustrating the function of the control system relative to keeping an angle to the helicopter.

Figure 2.13 shows a block diagram for the altitude controller, which keeps a reference altitude between the system and the sea. It is similar to the other controller but is not dependent on the helicopter, since the altitude can be set to the optimal altitude for the camera, placed on the quadcopter.

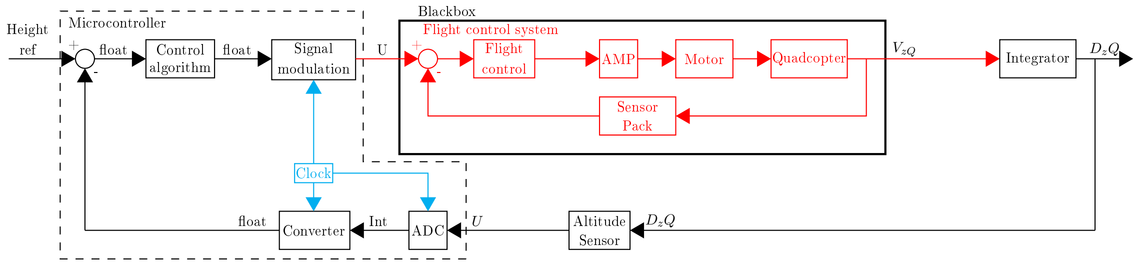


Figure 2.13: Blockdiagram illustrating the function of the control system relative to keeping a altitude above sea.

In the next chapter an analysis of the possible technologies for realising the sensors and control system, shown in figure 2.11, 2.12 and 2.13 are made.

Technology analysis 3

This chapter will describe, analyse and choose platform and concepts for measuring distance, angle and height.

3.1 Platform analysis

In this section the quadcopter as a platform will be described, analysed and chosen.

The quadcopter which will be used in this project will be the one shown on figure 3.1. This quadcopter was made available by Simon Jensen, engineering assistant at Aalborg university [personprofil, 2015].

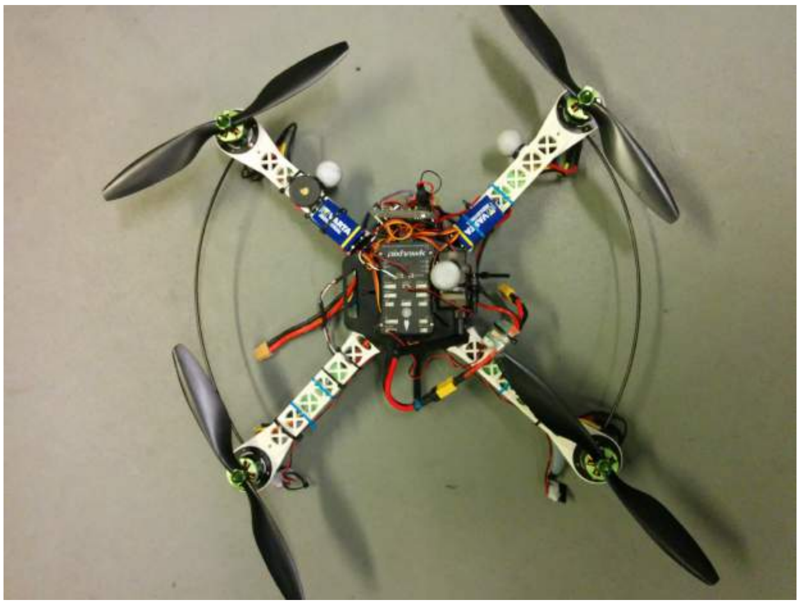


Figure 3.1: The quadcopter which was made available for this project.

The quadcopter platform consists of the following:

- Quadcopter frame: Diameter = 450 mm [HobbyKing, 2015].
- Motors: Multistar MT2213-935KV [Hobbyking, 2015a].
- Electronic Stability Control (ESC) [Hobbyking, 2015].
- Battery: Three cell LiPo with a capacity of 2200 mAh, and a nominal voltage of 11,1 V [Hobby-King, 2015].
- Pixhawk flight controller (Pixhawk): Refer to the section below.
- OrangeRx T-SIX [OrangeRX, 2015].
- OrangeRx R620X [OrangeRx, 2015].

- 3DR PPM Encoder [3DRobotics, 2014].
- Microcontroller: To be determined.

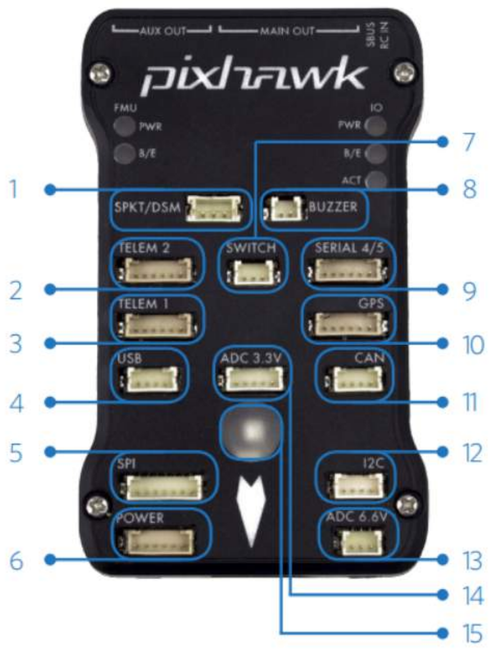
Pixhawk

This section will investigate the Pixhawk and the protocols/signals used to control the quadcopter. This will later be used in the process of replicating a RC controllers output with the microcontroller.

The Pixhawk has the following sensors onboard [3drobotics, 2015]:

- 3 axis 16-bit ST Micro L3GD20H gyro for determining orientation.
- 3 axis 14-bit accelerometer and compass for determining outside influences and compass heading.
- Provision for external compass with automatic switch-over if desired.
- MEAS MS5611 barometric pressure sensor for determining altitude.
- Built in voltage and current sensing for battery condition determination.
- Connection for an externally mountable UBLOX LEA GPS for determining absolute position.

The MEAS MS5611 barometric pressure sensor on the Pixhawk, could be used as a solution for the height sensor block in section 2.2 on figure 2.13. To see a more detailed description of the Pixhawk refer to the CD (CD/Sources/Pixhawk-manual). The Pixhawk, shown on figure 3.2 has different input types such as Digital Spectrum Modulation (DSM) or Pulse Position Modulation (PPM) to control the quadcopter.



1	Spektrum DSM receiver
2	Telemetry (radio telemetry)
3	Telemetry (on-screen display)
4	USB
5	SPI (serial peripheral interface) bus
6	Power module
7	Safety switch button
8	Buzzer
9	Serial
10	GPS module
11	CAN (controller area network) bus
12	I^{2}C splitter or compass module
13	Analog to digital converter 6.6 V
14	Analog to digital converter 3.3 V
15	LED indicator

Figure 3.2: The Pixhawk [3drobotics, 2014].

The Pixhawk has 8 input channels which can be assigned for different functions but only four channels is needed for flight and one for flight mode. Flight mode is important as it makes control of the quadcopter relatively simple or advance to control. Channel assignment after calibration is listed below. For more information on calibration refer to appendix A.3.

- Channel 1: Throttle (V_{zQ}).
- Channel 2: Yaw (ω_{yQ}).
- Channel 3: Elevation (pitch) (V_{xQ}).
- Channel 4: Aileron (roll) (V_{yQ}).
- Channel 5: Control system (f.mode gear switch).
- Channel 6: Flight modes (flaps gyro switch).

The DSM port on the Pixhawk has the following information [Coughlan, 2012].

- Standard UART communication (8N1) [Autopilot, 2015].
- The baudrate is 115200 bps.
- 3 connections: 3,3V, GND and Serial data (3,3V).
- One frame consist of 16 bytes and can hold information for up to seven different channels. 10 bits is used for counting missed frames.
- A frame is sent every 22 ms.

The data word shown on figure 3.3 where most significant bit (MSB) identifies the packet within each frame, channel ID identifies the movement parameter to control and a channel value [Coughlan, 2012].

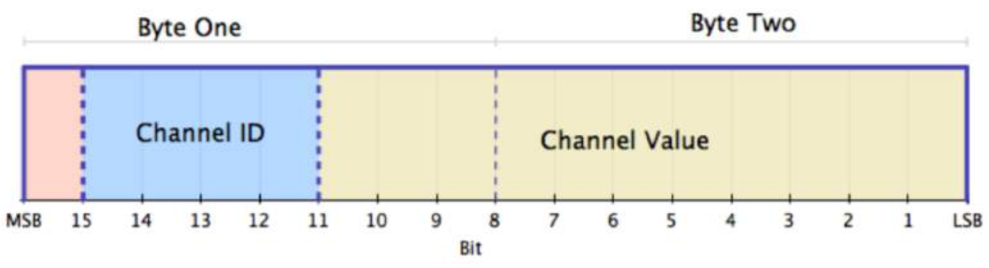


Figure 3.3: A illustation of the dataword [Coughlan, 2012].

The PPM port has the following specifications subtracted by monitoring the output of the receiver as per appendix A.7.

- A frame is sent every 19,54 ms to 26,26 ms.
- Every frame has a frame spacing of 10,50 ms.
- A channel has a minimum value of 680 μs - to a maximum value of 1520 μs.
- Every seperation between channels is 400 μs.

Figure 3.4 show an illustration of a PPM frame including the timing for the different parts.

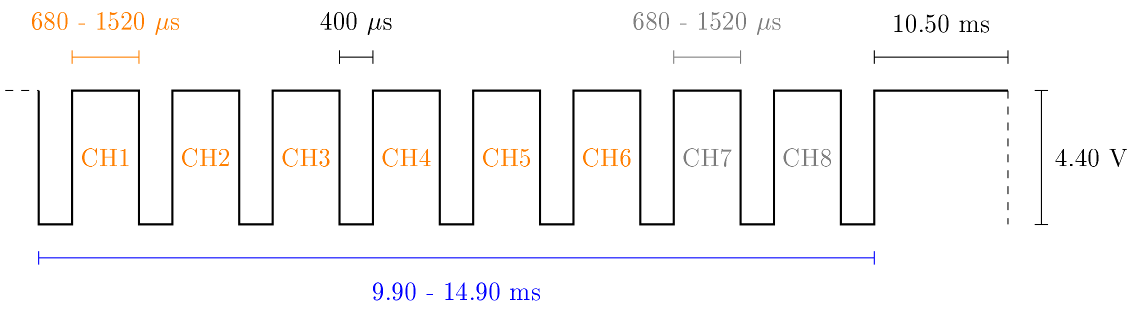


Figure 3.4: illustration of a PPM frame including timing for the different parts.

The following reaction is achieved, by decreasing or increasing the width of the channel, between 680 μs (low) and 1520 μs (high), for channel one to six:

- Channel 1: Decrease altitude - increase altitude.
- Channel 2: Rotates right - rotates left.
- Channel 3: Moving backward - moving forward.
- Channel 4: Moving Right - moving left.
- Channel 5: Switch with two settings.
- Channel 6: Switch between three settings, stabilized (680 μs), altitude control (1096 μs) and position control (1520 μs).

The DSM is the more advanced protocol, which makes use of UART communication. This has the advantage that an arduino would be an ideal microcontroller because it comes with dedicated UART ports. The PPM protocol is simplere but demands a precise timing, here a Field Programmable Gate Array (FPGA) would be the ideal choice of microcontroller because of its multitasking capabilities.

The choice of input falls on PPM because it is easy to decode and replicate. Due to lack of information on the Pixhawk, the resolution of the PPM signal which the Pixhawkawk functions with can not be determined. The project group will from this point forward, consider a resolution of maximum one μs valid as this will give minimum 840 steps from minimum to maximum adjustment of the channel width for throttle, and 420 steps in each direction for pitch, yaw and roll.

This leaves the choice of a microcontroller device to be determined. The possibilities within the limits of availabillity stretches from various arduino models to raspberry pi for microprocessors and XuLA2-LX9 for FPGA. More FPGA types is available to the project group but the Xula2 has a form factor and weight less than an arduino uno which makes it ideal for the project. For learning purposes the project group has chosen to implement the FPGA in to the system. This leads to the examination of the XuLA2-LX9 and the analysis of sensor modules for the project.

3.2 FPGA analsyis

In this section, the FPGA will be briefly be described, including some of the benefit over microprocessors.

An FPGA uses reprogrammable logic cell blocks placed in arrays, which can perform functions. The cells can be interconnected by programmable switches and I/O cells as shown on figure 3.5, making it possible to design more complex circuitry such as an arithmetic logic unit (ALU). Having thousands of logic gates, that can be programmed into several hardware modules running parallel, allows for a very high throughput of data [National-Instruments, 2011].

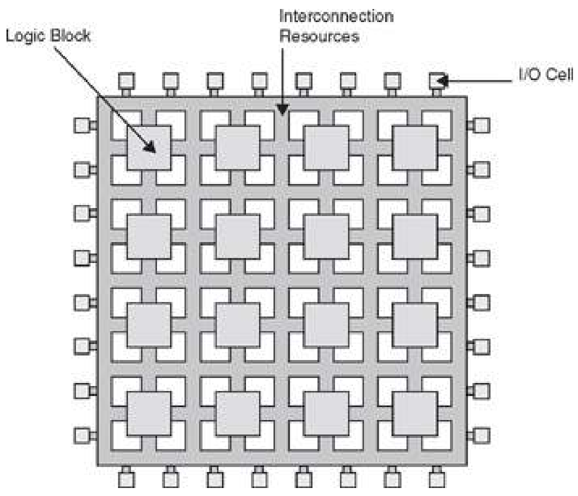


Figure 3.5: Illustration of FPGA architecture [Ashby, 2008].

It is thereby possible to achieve much higher performance on a FPGA than a microprocessor based solution, such as Arduino. There is no need for prioritization as it can run the individual processes parallel. It is possible to design or use a microprocessor base architecture on a FPGA, such as a MicroBlaze shown on figure 3.6).

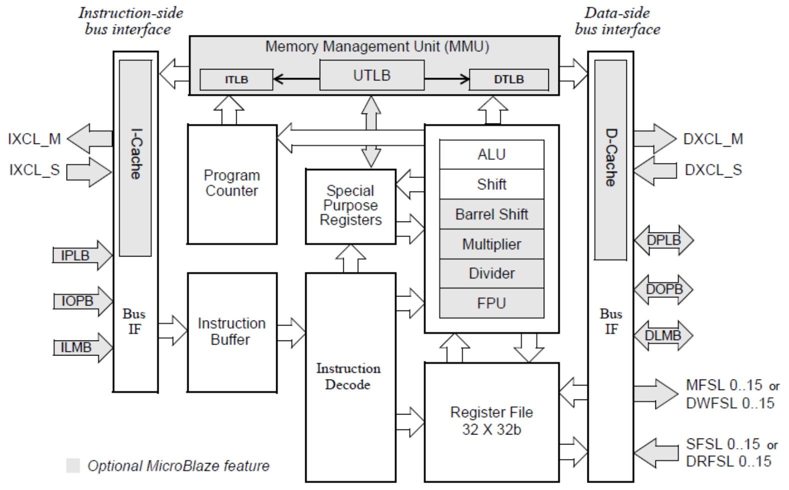


Figure 3.6: Illustration of the MicroBlaze architecture [Xilinx, 2008].

The FPGA uses a hardware language description (VHDL, Verilog). If a Microblaze is implemented, it is possible to use language such as C or Assembly.

The Xula2-LX9 is a 51 mm by 25 mm board with the following specifications [corporation,]:

- XC6SLX9 FPGA [Xilinx, 2015].
- 32 MByte SDRAM.
- 8 Mbit Flash.
- MicroSD socket.
- 3,3 and 1,2 V regulators.
- 40 pin connections.
  - 33 digital input/output
  - 2 analog input

  - 1 +5 V
  - 1 +3,3 V
  - 1 1,2 V
  - 1 GND
  - 1 Reset
- 12 MHz oscillator.
- USB 2.0 port.
- Auxiliary JTAG port.

The Xula2-LX9 works with Xilinx ISE Design suite and can be programmed through the USB port using XSTOOLs. There is over 9000 logic cells to programme and it is possible to store the design on the flash and power the board through pin headers, thereby eliminating the need for programming the FPGA after power up. The FPGA has four digital clock managers (DCM) which can divide or multiply by the 12 MHz clock, making it possible to achieve a 384 MHz digital clock. For interfacing, 33 of the 40 pin can be used as I/O pins, which is adequate for this project. A chart of the pin headers is shown on figure 3.7 [corporation,].

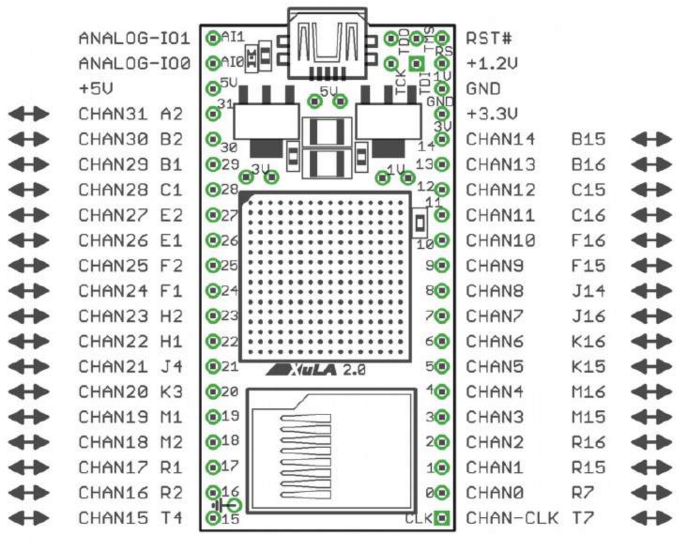


Figure 3.7: Interface for Xula2 [corporation, 2012].

In this section it has been found that a FPGA works by programming several logic cells to achieve a design, which has a high performance compared to other microprocessors.

3.3 Sensor analysis

In view of section 2.2 the system requires sensors for measuring distance, angle and height relative to the helicopter. In this section different types of concepts for measuring distance, angle and height will be analysed.

These concepts are:

- GPS
- Radio waves
- Sound- and radio waves
- Radar tracking
- Barometer

GPS

In this section global positioning system (GPS) will be analysed for whether it is possible to use it for determining the position of the system.

GPS is a system used for navigation in cars, mobile phones, handheld GPS units. GPS is a network of 30 satellites which are orbiting the earth with an altitude of 20.000 km [IOP, 2012]. Wherever the GPS unit is located there will always be four satellites visible to the unit. The signals from the satellites transmits at the speed of light, and are intercepted by the GPS unit. The unit then calculates the distance between the unit and the satellite. When the unit has information about the distance to at least three satellites it can pinpoint the position of itself using a process called trilateration which is shown on figure 3.8 [Mio, 2011].

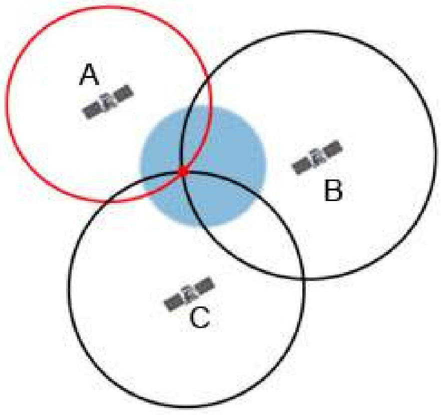


Figure 3.8: Example of position calculation by GPS [IOP, 2012].

Above sealevel GPS signals will be reflected on the sea surface, which means the signals from the satellite to the GPS unit on the quadcopter would create noise, because of the reflection from the sea surface [NOAA,]. Therefore the calculation of the position will be inaccurate, so GPS is not usable in this project.

Radio waves

In this section, it will be described how radio waves can be used to calculate the distance between the helicopter and the system. This will be further expanded to also include relative angle.

The distance can be calculated by measuring the round trip time to the helicopter as shown on figure 3.9.

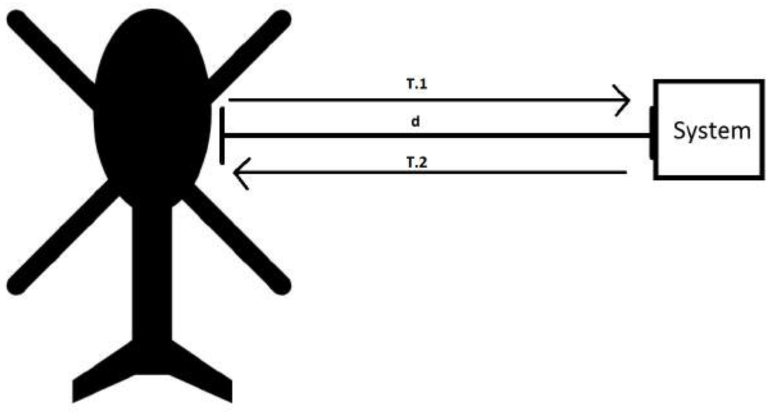


Figure 3.9: Concept of distance measuring.

To calculate the angle of the system compared to the helicopter, the concept of trigonometry as shown on figure 3.10 could be used.

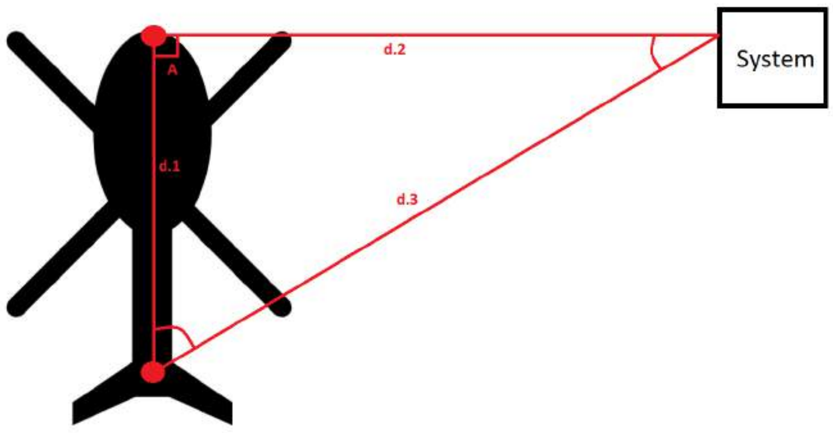


Figure 3.10: Concept of angle measuring.

To calculate the position of the system the angle A must be known. This can be done by knowing the distances d_1, d_2 and d_3. The distance d_1 is known the distances d_2 and d_3 can be measured. When all three distances are known the angle A can be calculated with the cosine relation shown in equation 3.1.

$$A = cos^{-1}\left(\frac{d_1^2 + d_2^2 - d_3^2}{2 \cdot d_1 \cdot d_2}\right) \qquad [^\circ] \quad (3.1)$$

Thereby it is possible to determine the angle A and the distance d_2 using this concept. A useable product for this concept is the pulson 410 made by time domain, but this product proved to expensive. To see the mail correspondence with Timedomain refer to CD/Appendix.

Sound/Radio waves

Another possibility would be the use of sound waves and radio waves in combination to measure the distance and angle between the helicopter and the system.

The concept of using both a sound wave and radio wave to measure the distance is to transmit a sound wave and a radio wave at the exact same time from the helicopter. When the radio wave reaches the system a counter will start and will stop when the sound wave is received. This concept works because of the differences in velocity of a radio wave and a sound wave. The velocity of a radio wave is 299.792 m/s, but the velocity of a sound wave is only 340 m/s at 15 °C [Tontechnik-Rechner, 2015]. The radio wave will therefore reach the system almost instantly therefore it can be assumed that the travel time of the soundwave is beeing measured, the concept of this is illustrated on figure 3.11.

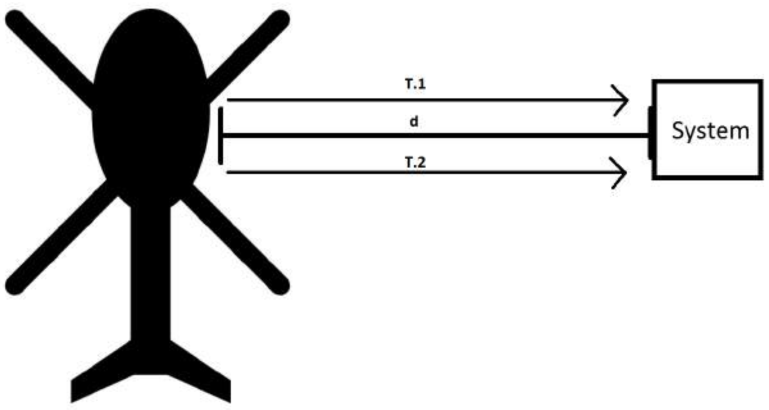


Figure 3.11: T_1 illustrates a radio wave and T_2 illustrates a sound wave.

To calculate the angle A shown on figure 3.10, the concept of sound/radio wave will be calculated in the same way as in section 3.3. Hardware which could be used for the sound wave part of this combined sensor could be, the MA40S4R/S [muRata, 2002] and the 250ST180 [Farnell, 2013] ultrasound transducers. For the radio wave the QAM-TX1 and QAM-RX2 radio modules can be used. For both in part beacuse they are available to the project.

Radar tracking

In this section, the use of radio detection and ranging (radar) as a possible solution to keep a distance, angle and altitude to a helicopter, will be examined.

Radar uses radiating electromagnetic waves, travelling at the speed of light, which is reflected back from an electrically leading surface, to determine the distance to an object (refer to figure 3.12). This is achieved by measuring the time between a transmitted pulse and a received reflected pulse [Wolff, 2009].

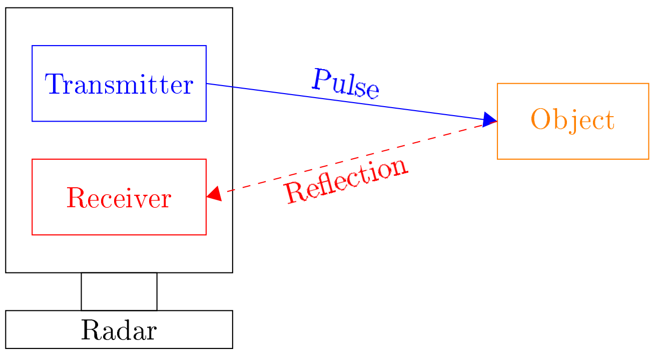


Figure 3.12: Radar principle.

If a directive antenna is used, the electromagnetic energy can be focused in a specific direction, the altitude and direction of an object can then be measured from elevation and azimuth (refer to figure 3.13). If the radar is placed on the object which is hovering or flying, and the energy is focused towards the ground (refer to figure 3.14), the altitude of the object can then be calculated, this is known as a radar altimeter [Wolff, 2009].

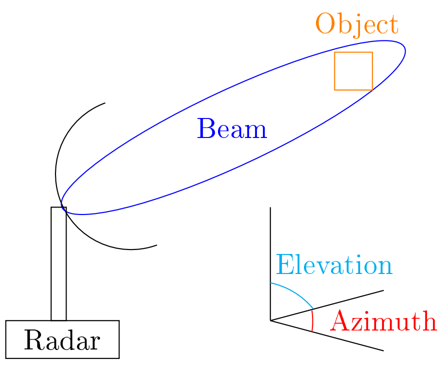


Figure 3.13: Directive radar antenna.

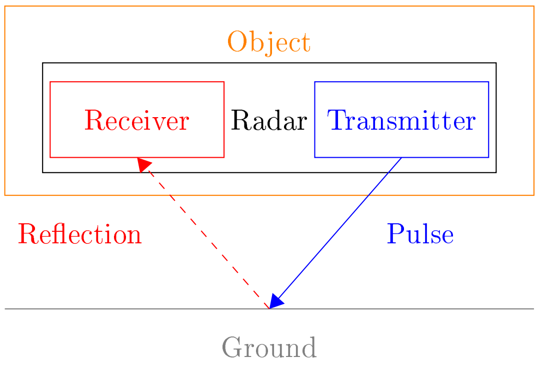


Figure 3.14: Radar altimeter.

Some of the advantages of radar, is the all weather capability, it can be used beyond visual range and it is able to operate day and night. The disadvantages of using radar, is minimum range, if the radar has one antenna it needs to switch between transmitting and receiving (monostatic radar), typically it requires a relative high power consumption, some severe weather condition such as heavy rain can affect performance and at longer range it typically has a lower accuracy [Wolff, 2009].

Designing and building a working radar is out of the scope for this semester and therefore radar as a solution is disregarded.

Barometer

In this section, the use of a barometer to maintain an altitude above sea level, will be examined.

A barometer is a type of sensor that measures atmospheric pressure. This type of pressure changes depending on the temperature and height. The relationship between these factors can be found based on the formula for barometric pressure. The advantage of using this method is that the actual height of the drone is measured, not the height relative to the helicopter. The disadvantage is that it needs to be calibrated in order to make precise measurements.

Since radar tracking is disregarded and there already is a calibrated and installed barometer on the Pixhawk, this could be the most simple but not necessarily best solution to maintain an altitude above sea level.

3.4 Choice of technology

In the past section different methods of distance measuring has been analysed. This leads to disregarding some of the concepts. The radio solution would either be too complex or expensive to realise. Complex because the time difference in the return time using radio would be to small to measure with the equipment available to the project group. The radar solution would be to expensive and complex for these same reasons this solution will therefore be disregarded. The GPS solution would not be optimal because the testing of the system will be done within a lab, and therefore the GPS signal will most likely be weak or none existing.

It has been chosen to use a solution with sound/radio waves to determine the distance and angle between the helicopter and the system. This is chosen because of the limited distances which the testing of the platform will be conducted under (refer to section 4.3). To reduce complexity and conserve time it has been decided not to implement altitude control, due to this limitation the system will not be fully autonomously but be aided by the RC remote in controlling throttle.

To sum up the block diagram shown on figure 3.15, the following choices have been made.

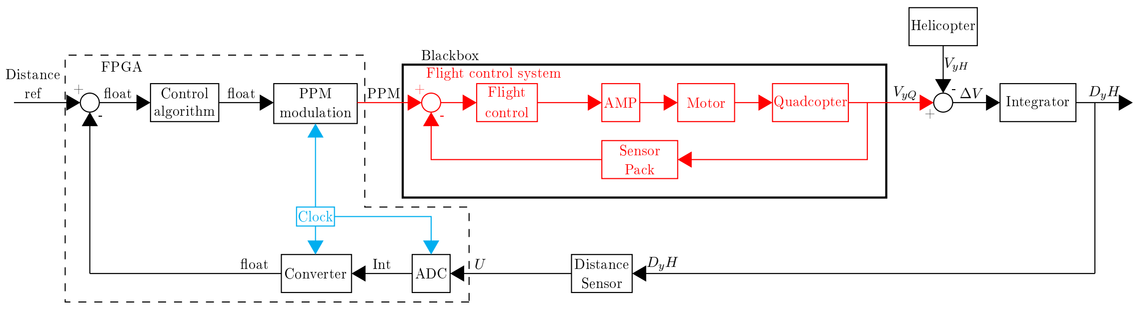


Figure 3.15: Blockdiagram illustrating the function of the control system relative to keeping a distance to the helicopter.

An XuLA2-LX9 FPGA has been chosen as the microcontroller, ultrasound and radio has been chosen to measure distance and PPM has been chosen to communicate with the Pixhawk.

Requirement specification 4

In this chapter, the requirements for the complete system is established. These requirements are objectives for when the system fulfill the product features. This chapter will also describe the test environment, which leads into the individual tests of the requirments.

4.1 System description

This section will describe the final system design as seen on the block diagram 4.1.

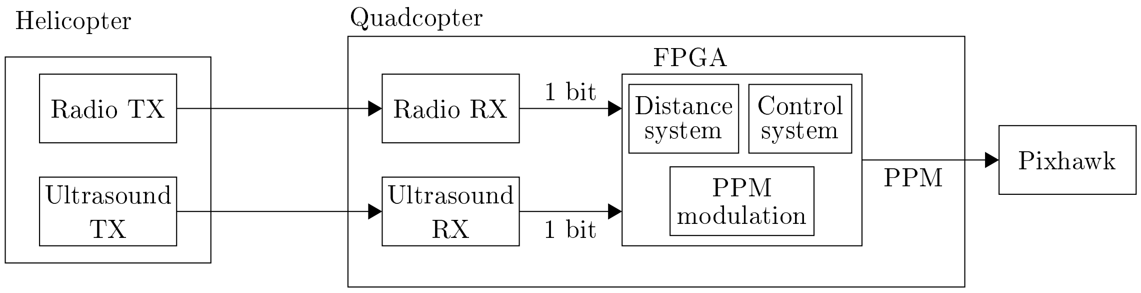


Figure 4.1: Block diagram of final design.

The design is divided into two parts, a transmitter and a receiver part. The transmitter which is placed on the helicopter, transmits radio and ultrasound waves at a given rate. The receiver placed on the quadcopter receives the radio and ultrasound waves and calculates the distance. With this distance a controller will be able to adjust the output which is modulated to a PPM signal and sendt to the Pixhawk.

Due to the limitations of the project, the system will not be made in full scale. This means the system design will focus on creating the distance measuring the control system and the PPM modulation. As a replacement for a helicopter a transmitter board will be built emulating a helicopter.

4.2 System requirements

In this section the system requirements will be defined.

As mentioned in section 2 a certain position must be held relative to the helicopter to increase the sweep width. Due to the limitations of the ultrasound sensors [muRata, 2002], the operation area have been reduced to 1 m - 4 m. Another limitation of the sensor system is it requires line of sight between the transmitter and reciever. The quadcopter therefore needs to keep a relative orientation to the helicopter. If the system should deviate to much from its position, it risks missing the person. This makes deviation from the search pattern undesirable, and therefore a small tolerance have been selected.

This leads to the following requirements:

- Funcionality
    1 The quadcopter shall maintain a predefined distance between 1 m - 4 m from the helicopter with a tolerance of maximum ± 10 % of the distance.

    2 The quadcopter shall maintain an angle of 90 ° to the helicopter with a tolerance of ± 20 °.

    3 The ultrasound receivers on the quadcopter shall maintain a orientation toward the helicopter with a tolerance of ± 10 °

The PPM module must be able to generate a PPM signal which controls the Pixhawk similar to that generated by the 3DR PPM encoder. The 3DR PPM encoder generated signal have been analysed for recreation in journal A.7. The resolution of the generated PPM signal must be of a size so that the replicated PPM signal does not exceed the one μs resolution which have been deemed sufficient according to section 3.1. Following requirements have been made for the PPM module:

- PPM Module
    4 The module must be able to replicate the 3DR PPM encoder generated PPM signal without introducing additional delay, compared to the 3DR PPM encoder.

    5 The module must have a resolution of 1 μs or less.

The distance measurement limits the performance of the rest of the system, as the control system is based on the sensor input. This means the distance measurement needs to be more precise than the functionality requirements to implement a safety margin in the design. This leads to the following requirements:

- Sensor Module
    6 The distance measurement system shall be able to measure a distance between 1 m - 4 m from the helicopter with a tolerance of ± 0,05 m.

    7 The Sensor module shall have an operation angle of ± 30 °.

4.3 Test environment

In this section the environment in which the testing of the system is conducted will be described. An explenation of the basic setup for testing the system, is also done in this section. Furthermore the simplification of the design including sensor is explained.

As mentioned in section 2.1, the legislation in Denmark prohibit flying outdoors without a permit. Therefore all tests must be done indoor, in a lab with the facility to measure the drone in flight. At Aalborg University the tests will be done in the Motion Tracking Lab (MTLab). The lab is an approximately five by six meter room connected to a Vicon MX system [Vicon, 2006] and a GamesOnTrack system [GamesOnTrack,].

GamesOnTrack uses ultrasound and radio technology to get precise indoor positioning data. There is six ultrasound receivers placed in the lab and the ultrasound transmitter which is placed on the object during testing. Communication is done wireless to a receiver, connected to a computer that calculates the positioning data which can be used for further process in other programs such as Matlab. There are several problems with GamesOnTrack, one is that it can not measure and calculate the rotation of the axes x, y and z without additional transmitters. Adding more transmitters, which is encased and requires a power source, onto a small frame is difficult and adds weight. Another issue with GamesOnTrack is that it uses the same frequency of ultrasound transducers and the radio modules as in the distance measuring for this project. This makes both systems unable to work simultaneously, as they interfere with each other.

The Vicon system include eight cameras which can track reflection markers placed on the drone in real time, the cameras are connected to a control box supplying power and synchronization and a host computer with the tracking software. The reflection markers makes it possible for the Vicon system to measure and calculate the rotation of the axes x, y and z. The reflection markers are relatively small and light and are therefore easier to fit on the frame. For collecting data it is possible to use Vicons Datastream SDK (software development kit), which can be run from MatLab.

The datastream include position and rotation data, where positions are expressed in millimeters and rotation in radians. It is possible to select between several time code standards, or use the individual frames as a stamp, to calculate the time [Vicon, 2013]. The parameters can be changed in a MatLab file called "ViconDataStreamSDK_MATLAB", to sort the datastream. The modified script can be found on the CD(CD/Matlab/ViconDataStreamSDK_MATLABTest).

Due to the limitations of the Games on track system, the final test will be made using Vicon. The test will focus on the system ability to maintain and hold a position to the transmitter. Using vicon, the movement of the system is tracked, which will be used to determine how well it operates.

4.4 Test specification

In this section, the process of testing the system requirements will be described. Each of these tests have been designed to cover one or more requirements in the requirement specification.

Functionallity Test

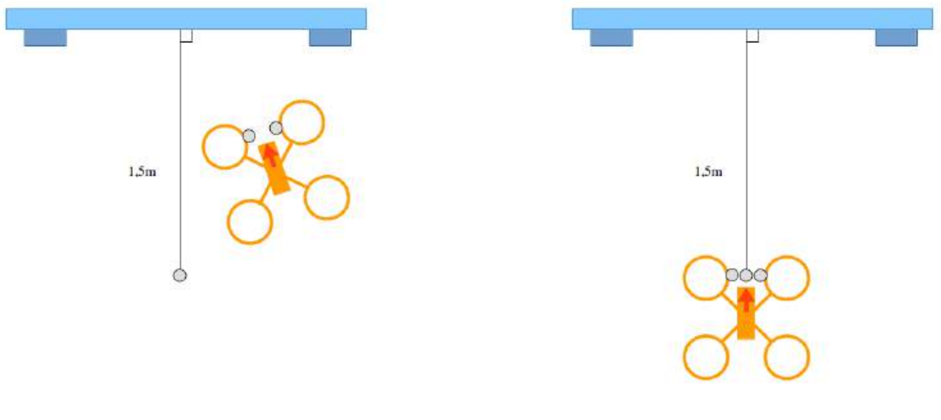
The purpose of this test is to examine how the system components works together. This test will include the following requirements:

1 The quadcopter shall maintain a predefined distance between 1 m - 4 m from the helicopter with a tolerance of maximum ± 10 % of the distance.

2 The quadcopter shall maintain an angle of 90 ° to the helicopter with a tolerance of ± 20 °.

3 The ultrasound receivers on the quadcopter shall maintain a orientation toward the helicopter with a tolerance of ± 10 °

By using the MTLab environment, the system motions can be tracked, in relation to a reference point. Before test start the system is given a reference distance for which it shall autonomously locate with a 90 ° angle from the transmitter board to the reference distance. To verify the precision of the reference distance and the angle from the quadcopter to the transmitter board and vice versa a motion track ball is placed at the reference distance with a 90 ° angle. The concept of this test is shown on figure 4.2. **Measurement Set up:**



(a) (b)

Figure 4.2: Test set up for the functionallity test.

If the quadcopter manages to perform this exercise within the limits defined in the requirements, this test will be deemed successful.

PPM encoding delay and precision

The purpose of this test is to determine the delay and precision of the PPM module to verify the following requirements:

4. The module must be able to replicate the PPM encoder generated PPM

signal without introducing additional delay, compared to the 3DR PPM encoder.

5. The module must have a resolution of 1 μs or less .

Figure 4.3 illustrates how the setup of the test should be done.

Measurement Set up:

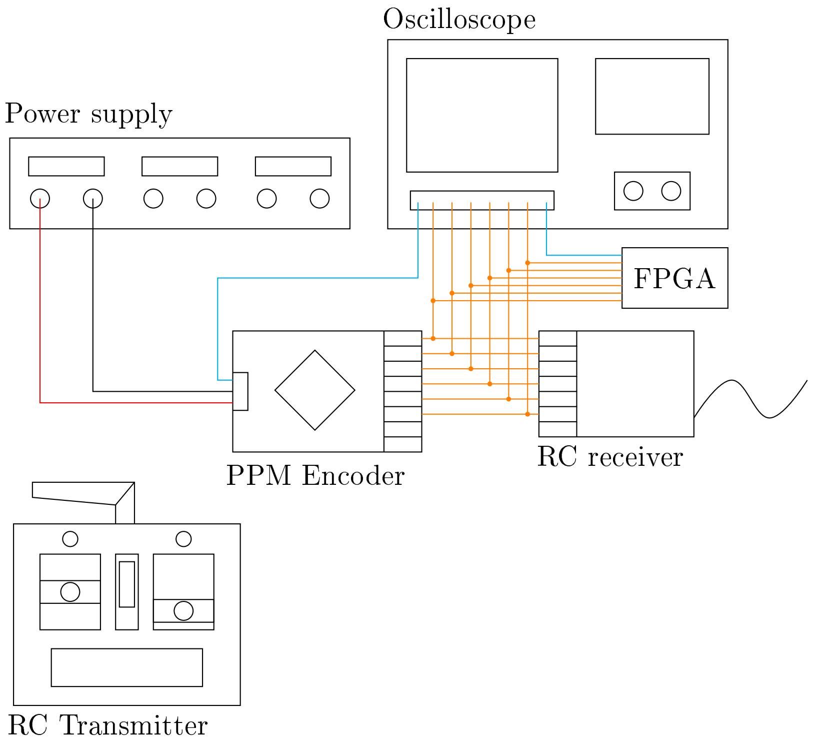


Figure 4.3: Illustration of measurement set up, were red is 5 V, black is ground, blue is PPM signal and orange is PWM channels 1-6.

The FPGA is loaded with the constructed PPM module and the various parts of the PPM signal output from the FPGA is measured for precision. For the delay test the FPGA and 3DR PPM encoder output is compared.

Distance Measurement With Motor On

The purpose of this test is to examine the distance measurement system to see if it is able to fulfill the following requirement:

6. The distance measurement system shall be able to measure a distance between 1 m - 4 m from the helicopter with a tolerance of $\pm$ 0,05 m.

This test is done by placing the quadcopter at a distance between 1 m - 4 m, anchoring the quadcopter to the floor, load the construted distance module on the FPGA, use a Arduino to read the meaured distance from the FPGA and read the measured distance from the serial port on the Arduino. By placing the quadcopter at a distance, as shown on figure 4.4 the stationary precision of the sensor module can be determined.

Measurement set up:

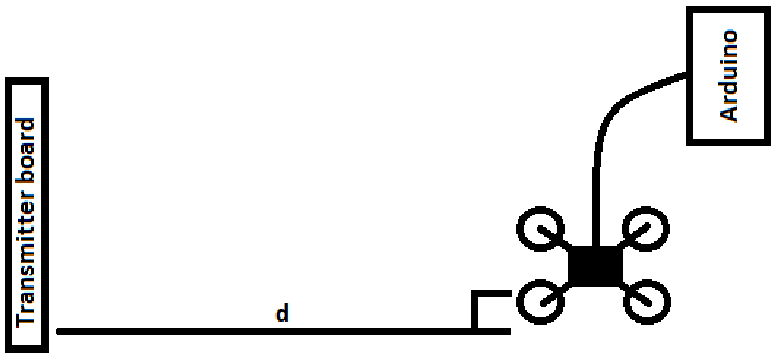


Figure 4.4: Setup for the distance test.

By repeatedly changing the distance, a overview of the general precision can be acquired.

Sensor Operation Range

Purpose:
The purpose of this measurement is to find the operation range of the sensor module. This test will cover following system Requirement:

7. The Sensor module shall have a operation angle of ± 30°.

To find the operation angle of the ultrasound transducers, this test will gradually increase the relative orientation of the quadcopter to find the operation angle. Place the quadcopter at 2 m, load the construted distance module on to the FPGA, use a Arduino to read the meaured distance from the FPGA. Using the set up as shown on figure 4.5 and gradually changing the orientation of the quadcopter, the operation angle can be found.
Measurement Set up:

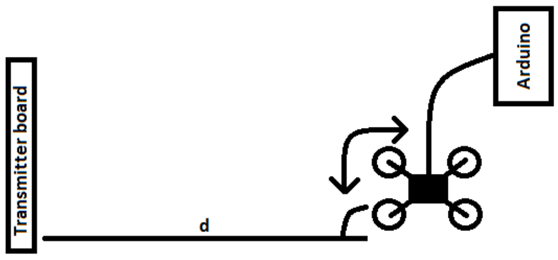


Figure 4.5: Setup for the distance angle test.

If the system either begins to timeout or constantly outputting a wrong distance, the orientation is beyond the operation angle.

With the general system design and system requirements described and how to document them, the design process can begin. Each submodule will be designed towards fulfilling their specific requirements, and ultimately tested to determine whether that is successful.

System design 5

In this chapter the functionality of the modules for the system is described, together with the interfaces between the modules.

The design is separated into the following modules, Ultrasound (orange), Radio (red), PPM modulation (blue), Controller (cyan) and FPGA (gray) as shown on figure 5.1.

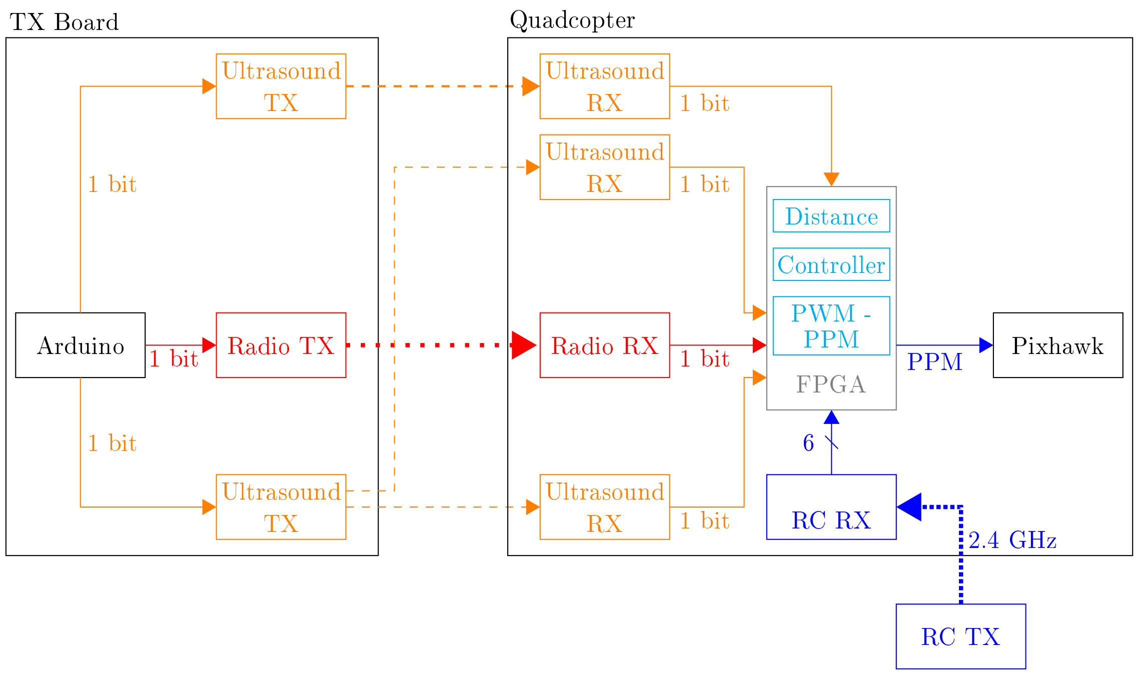


Figure 5.1: Block diagram of system including interfaces.

For simplification several modules are designed as larger blocks, these are distance and angle measuring (refer to chapter 6), PPM modulation (refer to chapter 7) and Controlling the quadcopter (refer to chapter 8).

Distance and angle measuring

The function of the distance and angle measuring is to measure three different distances which thereby will make the system able to determine the distance and calculate the angle between the helicopter and the quadcopter as analysed in section 5. The distance and angle can then be utilized by the control system to maneuver the quadcopter relative to a reference point.

For the ultrasound transducers it has been chosen to use 40 kHz and 25 kHz as suggested in section 3.3. These have been chosen because they will not interfere with each other and they are available for the project. The radio module is chosen to be the QAM-TX1 and QAM-RX2 as suggested in section 3.3 because they also are available for the project.

PPM modulation

PPM modulation will include the RC transmitter module, the RC receiver module and the FPGA module. The functionality of PPM modulation is to generates a PPM signal based on six PWM signals delivered by the RC transmitter and add an offset on certain channels, defined by the control module. The RC reciever outputs 6 seperate PWM signals. Therefore some hardware is designed to generate PPM signal based on the PWM signals, replicating the structure from the PPM encoder [3DRobotics, 2014]. The design must be implemented on the FPGA to generate a PPM output to the Pixhawk.

Controlling the quadcopter

The control module will take input from distance and angle measuring and generate an offset for the PPM modulation, in which the control system can change the direction of the quadcopter.

Having given a brief view of the functionality of the system a detailed examination of the different blocks shown in figure 5.1 will now follow.

Distance and angle measuring 6

In this chapter the solution concept as described in section 3.3, for measuring relative distance and angle, will be implemented. The solution concept consist of the following parts:

- Ultrasound.
- Radio.
- VHDL.

The idea behind the concept is to use radio and ultrasound waves to measure the distance between two objects as shown on figure 6.1.

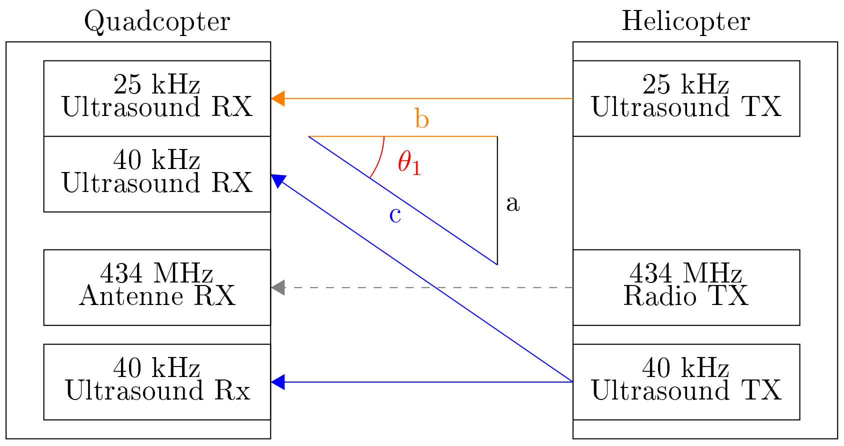


Figure 6.1: Concept diagram of the design.

To determine the position, one distance is not enough, but by increasing the amount of measured distances, the position can be triangulated, using equation 6.1. As stated in chapter 4, yaw pitch and roll controllers is required thus introducing the need for three distances to be measured.

$$\cos\theta_1 = \frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c} \quad [\text{rad}] \tag{6.1}$$

The sensor implementation will consists of the following modules:

- Transmitter board.
- Reciever board.

The transmitter board, emulates a helicopter and uses an Arduino to time the signals. The transmitter board furthermore consists of a 25 kHz ultrasound transmitter, a 40 kHz ultrasound transmitter and aradio transmitter. The reciever board consists of a 25kHz ultrasound reciever, two 40 kHz recievers, a radio receiver and a FPGA. On the receiving end the implementation will be done with VHDL on the FPGA. The concept of the receiving function is shown on figure 6.2.

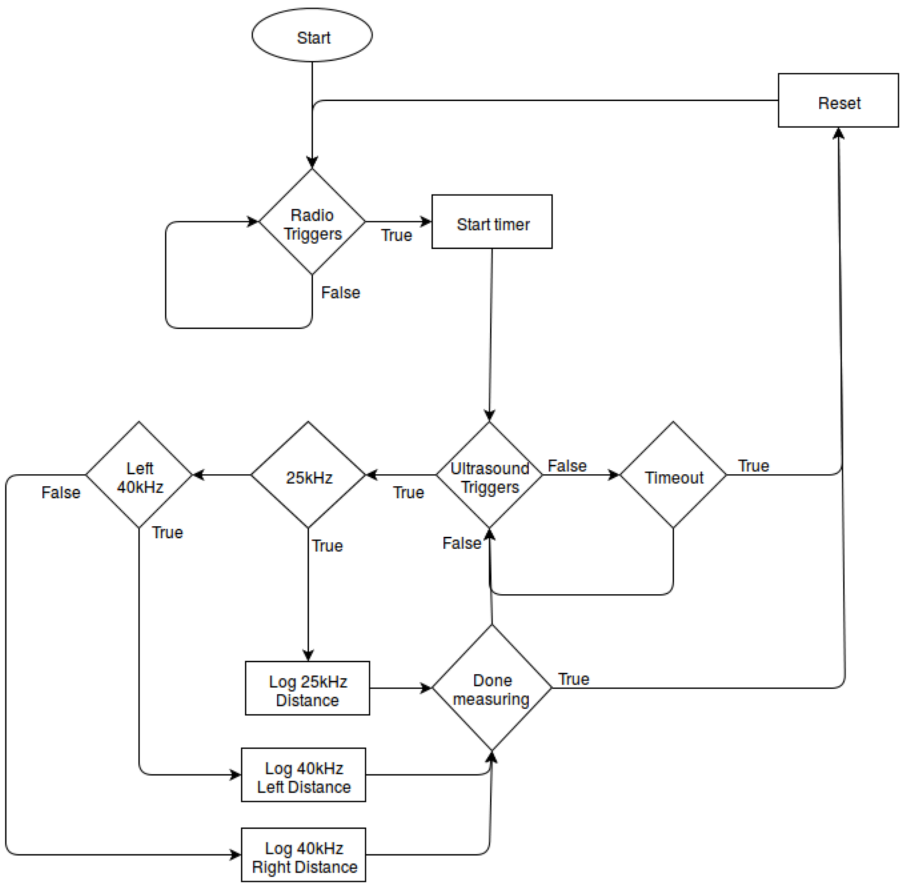


Figure 6.2: Flow diagram of Reciever module.

The Radio module will start a timer for each ultrasound reciever on the FPGA. When the FPGA recieves a ultrasound signal, its corresponding timer will stop counting.

This leads to the implementation of the transmitter and receiver module

6.1 Hardware implementation

This section will cover the hardware implementation of the receiver and transmitter modules.

Ultrasound transmitter circuit

To transmit ultrasound waves from the transmitter board it is chosen to use the NE555 [Texas-Instruments, 2014] integrated circuit (IC) to generate a 25 kHz and a 40 kHz signal. The NE555 is implemented as an astable multivibrator circuit which means, it will keep oscilating at a constant frequency as long as the reset pin is kept high. The general circuit, can be found in the datasheet [Texas-Instruments, 2014], and will be used to construct the circuit. The circuit consists of two resistors, two capacitors and a NE555 IC. The capacitor placed between pin 5 and GND is predefined, and used to prevent interference.

To calculate the remaining component values the following equation is used:

$$f = \frac{1,44}{(R_1 + 2 \cdot R_2) \cdot C} \qquad \text{[Hz]} \quad (6.2)$$

Where:
f is the frequency. [Hz]
R_1 and R_2 are resistors. [Ω]
C is a capacitor. [F]

The results of the component values is shown in table 6.1.

Frequency [Hz]	R1 [kΩ]	R2 [kΩ]	C [nF]
25	1	2,38	10
40	1	1,3	10

Table 6.1: Calculated component values for the astable multivibrator circuits.

The circuit diagram for the 40 kHz astable multivibrator is shown on figure 6.3.

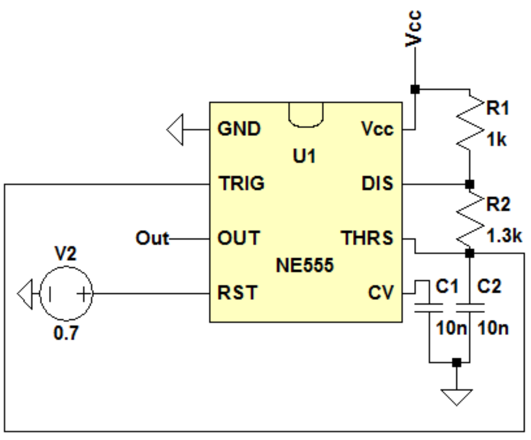


Figure 6.3: NE555 IC.

V_{CC} has been chosen to be 9 V as this enables the use of 9 V batteries, to make the board mobile.

Ultrasound receiver circuit

In this section the ultrasound receiver circuit will be explained. The circuit for 40 kHz and 25 kHz are identical except for different values, to see all calculations for 25 kHz and 40 kHz refer to appendix A.1. Due to the similarities this section will only cover the design process of the circuit for the 40 kHz transducers.

The ultrasound receivers voltage output from 0,2 m - 4 m is measured (refer to the journal in appendix A.16) to determine the needed amplification of the signal. This is shown on figure 6.4.

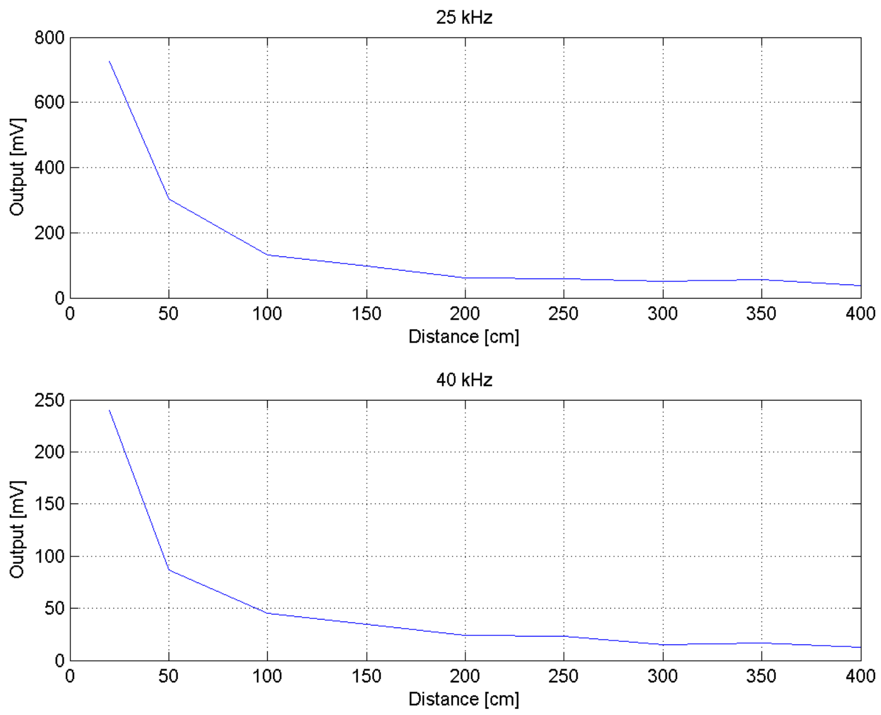


Figure 6.4: Output voltage over distance for 25 kHz and 40 kHz.

It has been chosen to amplify the lowest signal at 12,5 $m\hat{V}$ to 5 $\hat{V}$.

$$A_{dB} = 20 \cdot log_{10}\left(\frac{V_o}{V_i}\right) = 52 \qquad [\text{dB}] \quad (6.3)$$

Where:
A_{dB} is the amplification in dB. [dB]
V_o is the output voltage 5 $\hat{V}$. [V]
V_i is the input voltage 12,5 $m\hat{V}$. [V]

It is calculated that the amplification must have a gain of 52 dB.The designed amplifier circuit can be seen on figure 6.5.

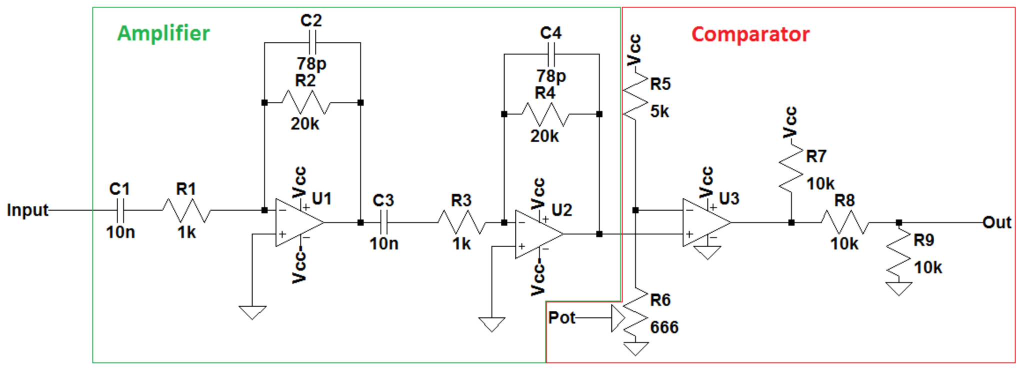


Figure 6.5: Amplifier and comparator design.

This circuit consist of two operational amplifiers (opamps) and one comparator. The opamps are setup as an inverting amplifier and the gain can be calculated as:

$$A = -\frac{R_2}{R_1} \qquad [\cdot] \quad (6.4)$$

Where:
A is the gain. $[\cdot]$
R_1 and R_2 are resistors. $[\Omega]$

To find the gain for 52 dB the following equation is used:

$$A = 10^{[\frac{52dB}{20}]} = 400 \qquad [\cdot] \quad (6.5)$$

To see all the calculations for the circuit refer to appendix A.1. The opamps must be able to amplify the signal 400 times which means, the two amplifiers is required to amplify the signal 20 times each. It is chosen to use 1 kΩ for R_1 and R_3, for R_2 and R_4 20 kΩ is chosen. With these values an amplification in each opamp will be 20 times. For opamps, it has been chosen to use the TLE2072 series because of their availability to the project and their common use.

To limit the noise above 40 kHz a capacitor (C2 and C4) has been placed in parallel with the feedback resistor and creates a pole at a 100 kHz. Furthermore there has been placed a capacitor (C1 and C3) in each opamp circuit to create a zero for attenuating frequencies below 40 kHz. These zeros have been placed at 20 kHz which combined creates a bandpass filter around 40kHz. To find the value of C_1 and C_3 the following equation is used:

$$C_1 = \frac{1}{2 \cdot \pi \cdot f \cdot R_1} \qquad [F] \quad (6.6)$$

Where:
C_1 is a capacitor [F]
f is the frequency where the pole/zero is placed [Hz]
R_1 is a resistor $[\Omega]$

The value for C_1 and C_3 are calculated to be 7,7 nF. A 7,7 nF capacitor does not exist in the E6 standard, therefore it is chosen to use a 10 nF capacitor. For C_2 and C_4 the same equation is used but with different frequencies and with the resistors R_2 and R_4.

The frequency response for 40 kHz is simulated in LTspice and is shown on figure 6.6.

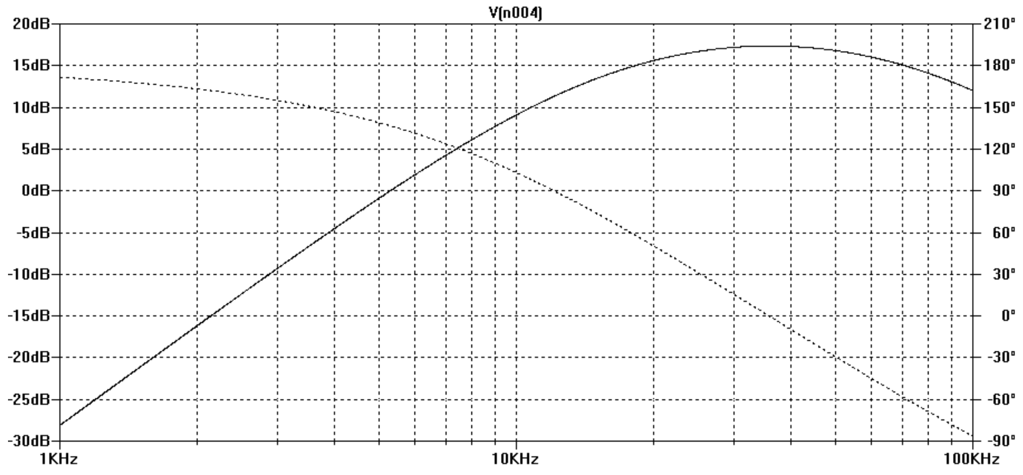


Figure 6.6: Frequency reponse for 40 kHz.

The center frequency is not exactly at 40 kHz but the attenuation in dB from the center frequency to 40 kHz is not significant therefore it is not a considered. The input signal has a level of -32 dB and after the amplification the signal has a level of 17,4 dB which gives an amplification of 49,4 dB. This is roughly 3 dB off from the 52 dB which was calculated. This loss comes from the placement of the zeros and the poles which attenuate the frequencies around 40 kHz, but it has no major influence on the output signal to the comparator.

The comparator has been placed in the design to distinguish noise from the signal so the FPGA after the comparator does not trigger from noise. For a comparator the LM311 IC has been chosen because of their availability for the project. The signal from the opamp (U2) is connected to the positive input of the LM311 (U3). On the negative input there has been made a reference voltage to control which signals on the positive input that are allowed to go through the LM311, which is made as a voltage divider. From the negative input to ground there have been placed a potentiometer to trim the resistance, which makes it possible to change the reference voltage lower or higher, depending on what is most optimal.

On the output for the LM311 there has been placed a pullup resistor. When the signal is high enough to trigger the comparator it will make a square wave on the output at a 40 kHz frequency as shown on figure 6.7.

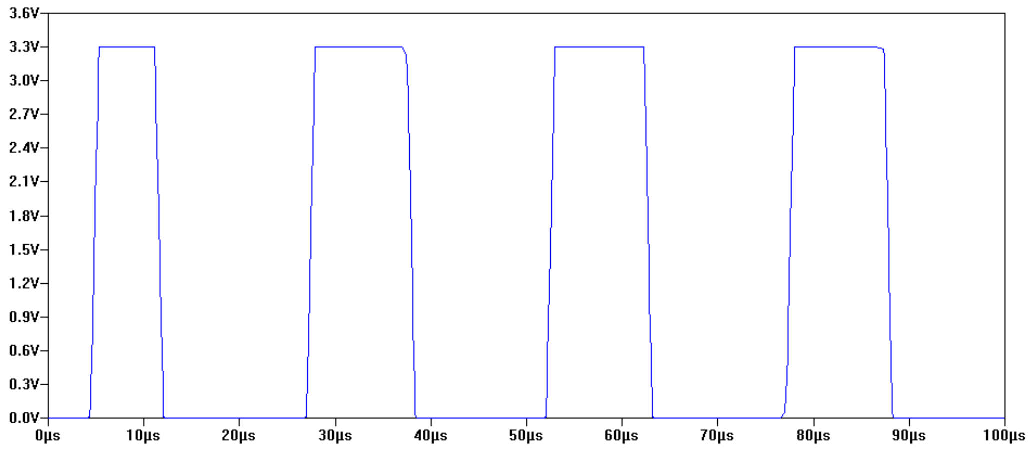


Figure 6.7: Output from the Comparator.

When this square wave comes through it will trigger the input to the FPGA. Before the signal comes to the FPGA the signal goes through a voltage divider to limit the voltage to 3,3 V into the FPGA, which the FPGA will see as a high signal.

Radio

The two radio modules transmits and receives at a carrier frequency of 433 MHz and can transmit data from 200 Hz up to 3 kHz. If no signal is recieved within a time period, the reciever module will timeout and enter a powersaving mode. Due to the ambiguty and lack of information about the module in datasheet, following assumtions are made:

- The reciever module will timeout and enter powersaving mode if data transfer is below 200Hz

- The reciever module takes 20 ms to wake up from powersaving mode (listed as turn-on time in datasheet)

These assumptions are based on the results of measurement report (refer to appendix A.10) and the datasheet located in (/CD/Datasheet/Radio Tx.pdf and /CD/Datasheet/Radio Rx-pdf). It is therefore desirable to prevent a timeout as this would introduce a significant delay into the system and a possible inaccurate measurement. A way to prevent timeouts is to constantly transfer data through the modules, this will prevent the above issue which is further backed up by measurement report (refer to appendix A.10). This will increase the complexity of the system as a communication protocol between the transmitter and the reciever is required. The requirements to the protocol are:

- Prevent reciever timeout.
- Signal when a ultrasound pulse is send.

As the tasks required are relatively simple, only two signals are needed, Sync and Ping. The Sync signal is to prevent timeout, and acts as padding between pings. The Ping signal is what signals a ultrasound pulse is being send, and is signaled by having a larger uptime than Sync. A illustratation of this protocol can be seen on figure 6.8.

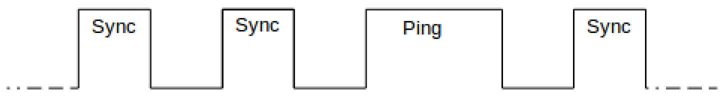


Figure 6.8: Sync And Ping signals

The only requirement to the uptime of the signals is to remain within the limits of data transfer rates described in the datasheet, however it it prefered to be as fast as possible. This gives a basis for designing the up- and down- time of the protocol which can be seen on table 6.2.

Signal	Uptime[μs]	Downtime[μs]
Sync	252	168
Ping	428	168

Table 6.2: Timing table for the transmitter module.

The numbers selected in table 6.2 are arbitrary values and are of little importance, as long as they are constant and within the limits. The output of the reciever module is more relevant, as these have to be decoded by the FPGA. Table 6.3 shows the output of the reciever module when above values are send, which is obtained from the measurement report (refer to appendix A.10).

Signal	Uptime[μs]	Downtime[μs]
Sync	228	228
Ping	388	228

Table 6.3: Timing table for the reciever module.

It is unclear what creates the inaccurate times, but these must be taken into consideration when designing the decoder.

6.2 Implementation of Software

This section will cover the implementation of the sensor software. The section will cover both the Arduino software as well as the FPGA implementation. The Arduino controls the transmitter module, while the FPGA controls the reciever module.

Arduino Software implementation

The transmitter board consists of three inputs, which controls the radio module and the two ultrasound modules. The arduino must be able to:

- Send Ping when ultrasound is being send
- Send Sync signal when ultrasound is not being send
- Hold the ultrasound modules high for 400us with a certain frequency

The period between ultrasound is being send must not be lower than 11,66 ms, as this is the time it takes sound to travel 4m at 20 $°C$. However the higher the period is, the slower the system will be. If the period is too low, reflections from old measurements risk being picked up by the sensor instad, which will result in inaccurate measurements.

This means that selecting the right period is a tradeoff between speed of the system and stability of the measurements. A period of around 30 ms is selected, as this is a reasonably fast update rate, and leaves plenty of time for the reflections to die out. This have been implementated by using two functions, namely SendSync and SendPing. As the names suggest, both functions handles their respective task. SendSync sends a sync signal when no ping is being send, and SendPing activates the ultrasound transmitters while a Ping signal is being send through the radio module.

VHDL design

In this section the design process behind the distance measuring module in VHDL will be described.

The VHDL concept of distance measurement consist of three different modules:

- Top module.
- Radio module.
- Timer module.

The top module handles the input and output and connects the two submodules, the radio module listens for a sync signal and a ping signal and the timer module calculates the time between the ping signal and the trigger signal. Each of these modules will be described in the following.

Top module

The module uses the following ports and signals.

- Ports
  - Radio: Logic input which connects to the radio receiver.
  - Sensor40L: Logic input which connects to the 40 kHz ultrasound receiver at the left.
  - Sensor40R: Logic input which connects to the 40 kHz ultrasound receiver at the right.
  - Sensor25: Logic input which connects to the 25 kHz ultrasound receiver.
  - Err: Three bit output vector which connects to the LED´s on the quadcopter.
  - Dataardo: 12 bit vector which is used to send distance measurements to an Arduino(manually connected to the sensor which is to be monitored).
- Signals
  - TriggerRad: Logic signal which is set according to the ping signal.
  - Shutdown: Logic signal which signals if the system should shutdown.
  - SignalTimeout: Logic signal which signals if the systems is timeouted.
  - Err40L,Err40R,Err25: Logic signal tells whether the timer has timeouted.
  - DataIn40L,DataIn40R,DataIn25: 12 bit vector which acts as a container.
  - Dist40L,Dist40R,Dist25: 12 bit vector which acts as a container.

In code section 6.1 it is shown how a simple form of error handling is performed.

```
-------------------Radio Ping------------------------------------------------
RadioPing :RadioSync PORT MAP(clk, Antenna,TriggerRad,Shutdown);
-------Sensor-40-L-----------------------------------------------------------
Timer40L      :TimerDistance40L PORT MAP(clk, TriggerRad, Trigger40L, Err40L,
    Dist40L);
-------Sensor-40-R-----------------------------------------------------------
Timer40L      :TimerDistance40R PORT MAP(clk, TriggerRad, Trigger40R, Err40R,
    Dist40R);
--------25kHz---------------------------------------------------------------
Timer25       :TimeDistance25 PORT MAP(clk, TriggerRad, Trigger25, Err25,
    Dist25);

DataIn40L <= Dist40L when Err40L = '0' else DataIn40L;
DataIn40R <= Dist40R when Err40R = '0' else DataIn40R;
DataIn25 <= Dist25 when Err25 = '0' else DataIn25;
```

Code 6.1: port mapping and error handling.

The reason for error handling is to ensure that the right distance is used. If error = '1' the old distance value will be used until a new validated distance occurs. This performs a simple form of error handling which assumes that errors will only occur for a few distance measurements. This leads to the description of the two submodules.

Radio Module

This module uses the following ports and signals.

- Ports
  - Radio: Logic input which connects to the radio receiver.
  - RadioTrigger: Logic output which signals the timer module if a ping signal is recieved.

- Signals
  - Clk_Scale: 13 bit integer which holds the amount of clk signals the radio is logical 1.
  - Radio_Timer: 13 bit integer which holds the amount of clk signals the radio is logical 1.
  - Clk_Timeout: 13 bit integer which holds the amount of clock cycles the radio is logical 0.

The purpose of the radio module is to validate a sync and a ping signal. The sync signal is set to arrive every 416 μs. If this signal is not received continously the system will shutdown. This is done partly as a safety mechanism but also to ensure that the radio module does not timeout, as this will introduce an error into the system. The ping signal is used to start a timer when the ultrasound is sent from the transmitters. A flowchart of the radio module is shown on figure 6.9

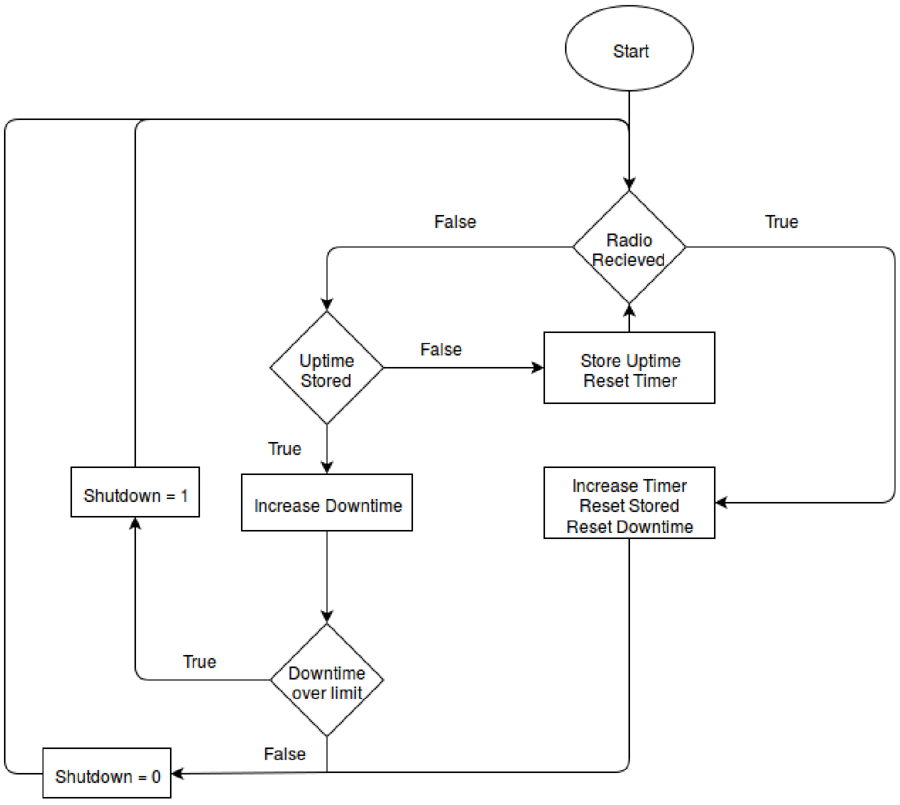


Figure 6.9: Flowchart describing the Radio Sync module.

To ensure recognition of a radiosignal a timer is started on rising edge and read on falling edge on the radio. Furthermore a timer is needed when the radio signal is low to detect if a timeout has occured in the system. The rules the limits of the timing can be seen in table 6.4 and table 6.5.

Trigger	**Uptime sending from Arduino [μs]**	**Uptime received by FPGA [μs]**	**Clock cyles for received uptime on FPGA**
Lower limit	375	308	3696
Upper limit	425	352	4224

Table 6.4: Trigger rules

Timeout	Uptime received by FPGA [μs]	Downtime received by FPGA [μs]	Clock cyles for received Up-/Downtime on FPGA
Lower limit	100	100	1200
Upper limit	425	425	4224

Table 6.5: Timeout rules

The radio module is implemented with a process that increments a timer dependent on the state of the radio as shown in code section 6.2. After a falling edge the timer value will be saved and reset which will determine if a sync, ping or timeout signal is received.

```
process(clk,radio) begin
if rising_edge(clk) then
        if radio = '1' then
                ToggleRead <= '1';
                clk_scale <= clk_scale +1;
                Clk_Timeout <= 0;
        elsif radio = '0' and ToggleRead = '1' then
                ToggleRead <= '0';
                Radio_Timer <= clk_scale;
        elsif ToggleRead = '0' and radio = '0' then
                clk_scale <= 0;
                Clk_Timeout <= Clk_Timeout +1;
        end if;
end if;
end process;
```

Code 6.2: Radio control.

The output of the radio module is dependent on clk_Timeout and Radio_Timer which is set in code 6.2. As the state of the signals is clock independent these will be set outside the process as seen in code 6.3.

```
Timeout <= '1' when Clk_Timeout > 7000 else
                        '1' when Radio_Timer < 1000 or Radio_Timer > 7000 else
                            '0';

-----Assumed to be around 330 us -> 3960 clk's
RadioTrigger <= '1' when Radio_Timer >
3960 and Radio_Timer < 7000 else '0';
end Behavioral;
```

Code 6.3: Timeout rules and trigger.

The last module to be described is the timer module.

Timer Module

The module uses the following ports and signals.

- Ports
  - Start: logical input which starts the timer.
  - Trigger: Logical input which stops the timer.
  - Error: logical output which signals the timer have timeouted.
  - Data_O: 12-bit vector which holds the calculated distance.

- Signals
  - Begin_Mesh, Hold_Mesh: Latch, which signals the timer is running.
  - Timeout: logical signal which signals if the system is timed out.
  - Clk_scale: 6 bit integer which holds the amount of clock cycles between distance triggers.
  - Distance,temp_int,DistanceOut: 12 bit integer that holds the distance the sound have travelled in mm.

The purpose of the timer module is as mentioned to measure the time between the ping from the radio to the trigger from a ultrasound receiver. The flow of timer module can be seen on figure 6.10.

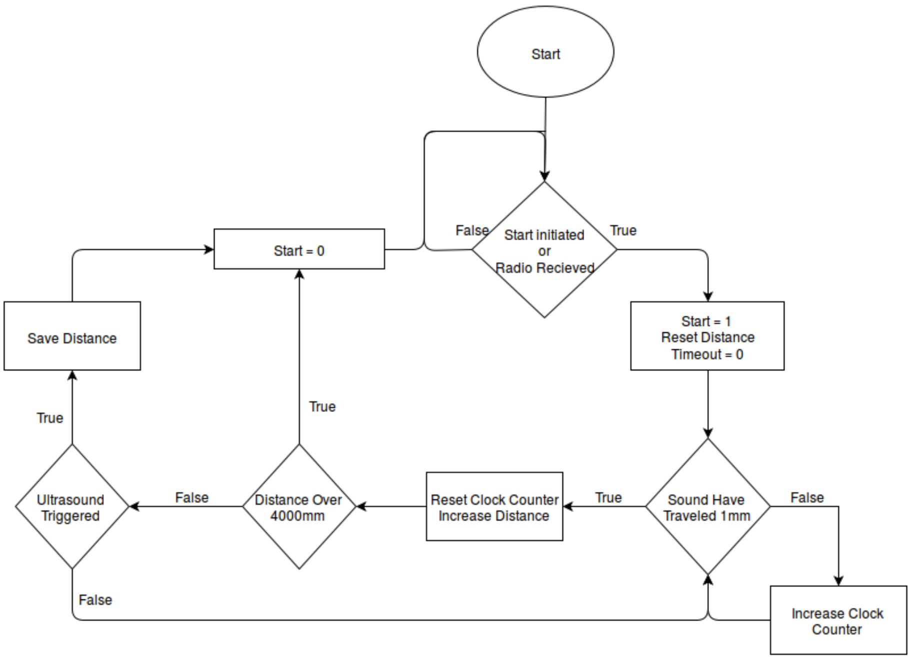


Figure 6.10: Flowchart describing the distance timer.

To achieve this the module uses two timers. One timer increments every clock cycle, counting up to 35 cycles (see equation 6.7) which corresponds to the time it takes for sound to travel 1 mm. The second timer holds total distance which the sound has travel and increments every time timer one reaches 35. As the speed of sound is dependent on the temperature, timer one has been calibrated to 20°C.

$$Clockcycles_{1mm} = 12MHz \cdot \frac{1mm}{343\frac{m}{s}} \qquad [\cdot] \quad (6.7)$$

Where:
12 MHz is the FPGA clock. [Hz]
$343\frac{m}{s}$ is the speed of sound at 20°C. $[\frac{\text{m}}{\text{s}}]$

If the distance gets above 4000 mm as seen on figure 6.10, the module will timeout. This is implemented because the ultrasound tranducer and receiver have a maximum range of 4 m. The implementation can be seen below in code 6.4.

```
process(clk)
begin
if (clk'event and clk = '1') then
if Begin_Mes = '0' and start = '1' and trigger = '0' then
              if Begin_mes = '0' then
              Begin_Mes <= '1';
              timeout <= '0';
              end if;
end if;
if distance >= 4095 then
              Begin_Mes <= '0';
              timeout <= '1';
end if;
       if Begin_mes = '1' then
              Clk_scale <= Clk_scale +1;
              if clk_scale >= 35 then
                     clk_scale <= 0;
                     Temp_int <= Distance;

                            if begin_mes ='1' and trigger = '1' then
                                   Data_o  <=
                                       std_logic_vector(to_unsigned(Temp_int,12));
                                   Distance <= 36;
                                   Begin_mes <= '0';
                            else
                            Distance <= Distance +1;
                            end if;
              end if;
       else
       clk_scale <= 0;
       temp_int <= 36;
       distance <= 36;
       end if;
end if;
end process;
```

Code 6.4: Timer module for 25 kHz.

The offsets implemented in the module is delays in the hardware and software. These have been measured (refer to journal A.14) to be:

Module	Time delay [μs]	Distance [mm]
Radio hardware	84	28,81
Radio FPGA module	420	144,1
40 kHz Ultrasound	-100,1	-34,5
Sum	404,9	108,41

Table 6.6: Table of Measured Delays and calculated distances for 40 kHz.

Module	Time delay [μs]	Distance [mm]
Radio hardware	84	28,81
Radio FPGA module	420	144,1
25 kHz Ultrasound	-310,1	-107
Sum	194,9	35,91

Table 6.7: Table of Measured Delays and calculated distances for 25 kHz.

Having described the VHDL modules and hardware in the measurement system, leads to a final test of the measurement system.

6.3 Final implementation

This section will cover the final implementation of the sensor system and testing to see if it fulfills the requirments.

Transmitter module

In this section the construction of the transmitter board will be shown. Furthermore the test for the astable circuit will be conducted.

The ultrasound transmitters, radio transmitter, Arduino and transmitter board are mounted on a wooden frame, as shown on figure 6.11 and 6.12. The wooden frame is 100 x 10 cm. The ultrasound transmitters together with the radio transmitter are placed on the side facing the quadcopter as seen on figure 6.11. On the backside the transmitter board, the Arduino and battery supply is mounted as seen on figure 6.12.

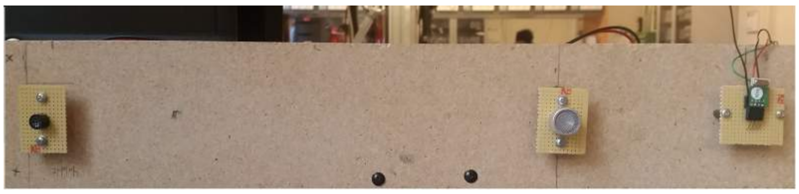


Figure 6.11: Front of the transmitter board.

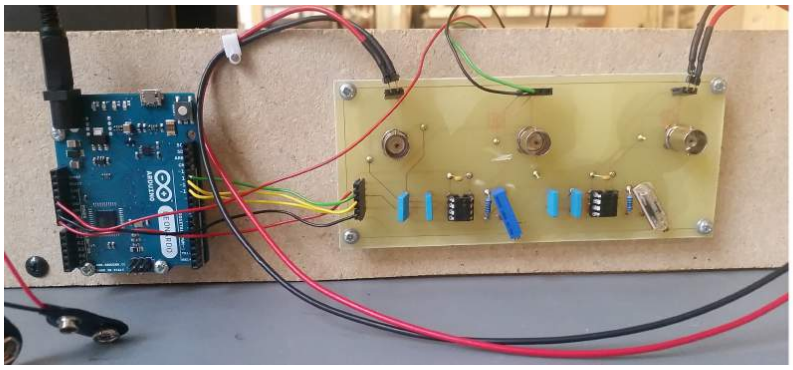
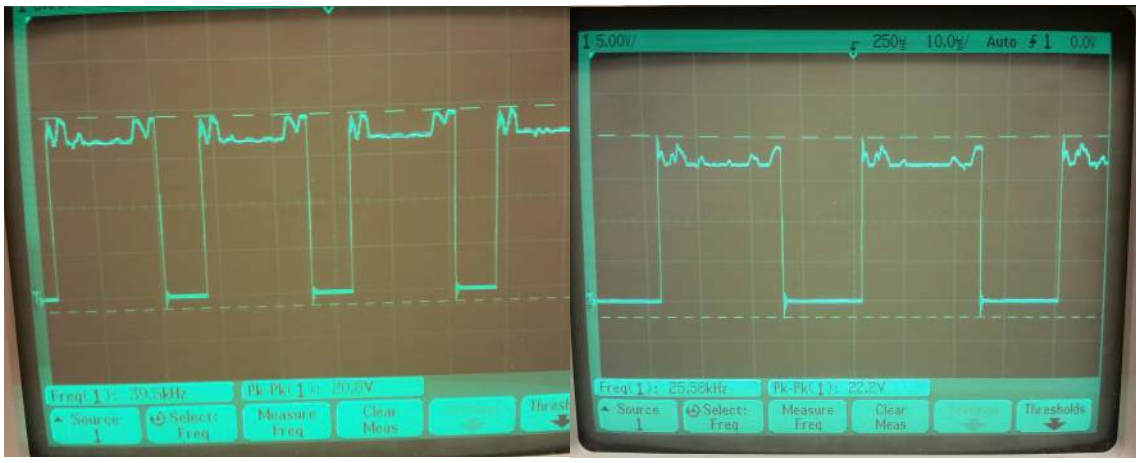


Figure 6.12: Backside of the transmitterboard.

The 25 kHz and 40 Khz transmitters are spaced apart with a distance matching the receivers on the quadcopter. The distance between the sensors is measured to 33,5 cm.

Astable circuit

The circuit was tested to see if the two astable multivibrator circuits transmitted at the correct frequency with an oscilloscope (AAU 33858) which can be seen on figure 6.13.



(a) (b)

Figure 6.13: (a)40 kHz astable. (b)25 kHz astable.

Both astable circuits transmitted around the desired frequency, therefore it can be concluded that the astable circuits functions correct. The Arduino sketch used on the transmitter can be found in "CD/CODE/Send_Sync".

Reciever Module

In this section a frequency response for the amplifier will be conducted, and a test for the comparator will be conducted to see if it gives the correct output. Futhermore a test will be conducted on the ultrasound receivers to investigate if there is any noise that interfere with the signal. At the end a test for the distance system will be conducted to see if they fulfill the requirements.

On figure 6.14 the receiver board is shown.

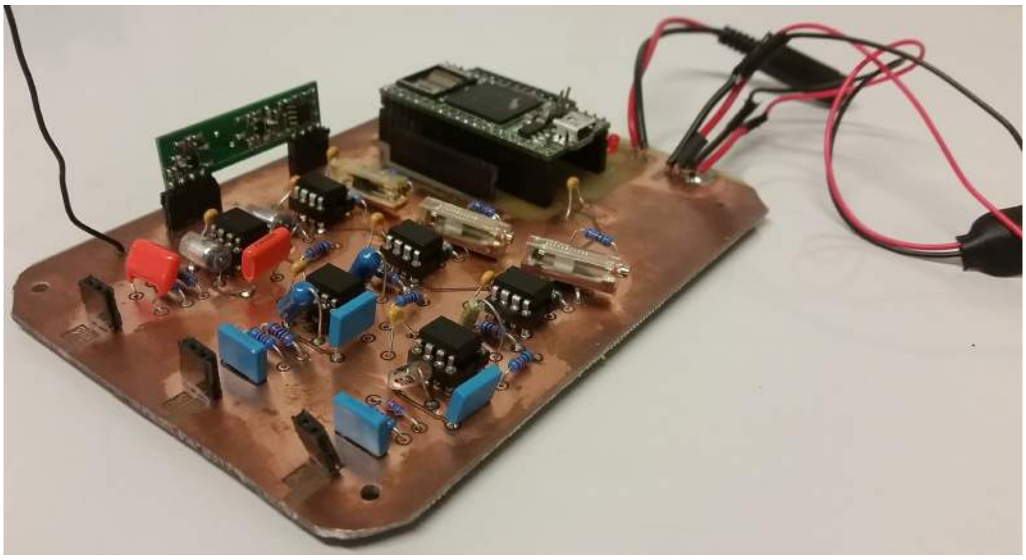


Figure 6.14: Receiver board.

On the receiver board there are placed three input sockets for the two 40 kHz and one for the 25 kHz. After each socket the opamps(TLE2072) are placed followed by the comparators(LM311) and from the comparator into the FPGA. Futhermore the radio module is placed on the receiver board. The board is powered by two 9 V batteries, and the FPGA is powered by a separate power supply. Futhermore three diodes is placed to see when the FPGA receives a signal.

A frequency response was made for the amplifiers with a NI-4461 card which is available for the project. This test was made for the 40 kHz amplifier and the 25 kHz amplifier. The frequency response for 40 kHz is tested for frequencies from 1000 Hz to 90 kHz as seen on figure 6.15

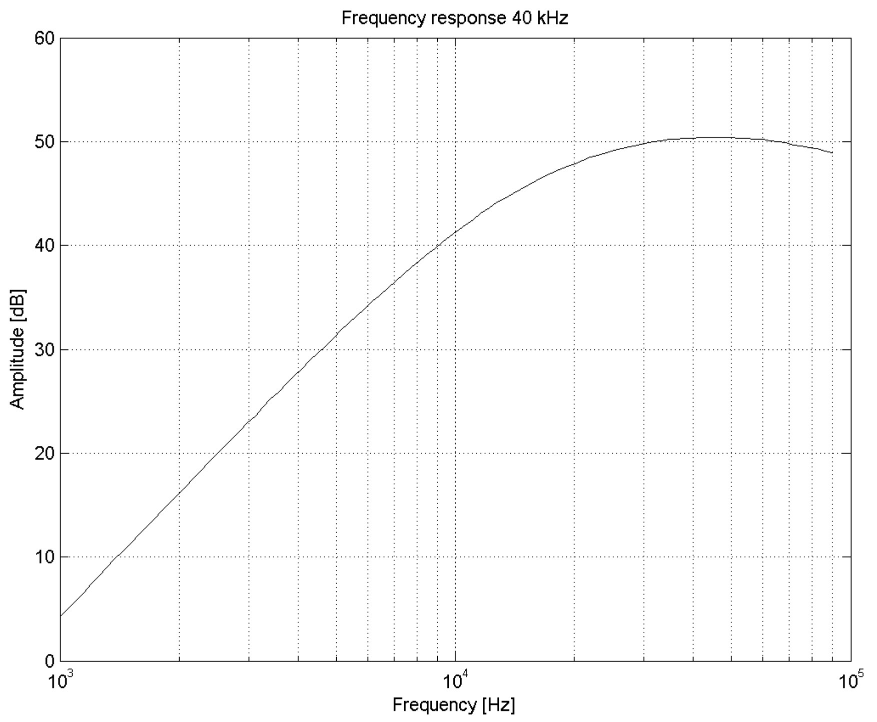


Figure 6.15: Output from the 40 kHz.

It can be seen that at 10 kHz the signal is attenuated with approximately 8 dB compared to the frequencies at 40 kHz. This is the same as in the simulation for 40 kHz refer to 6.6. For 40 kHz to 90 kHz the system is approximately attenuated with 3-4 dB which also match the simulation. It can be read from the figure that the level at 40 kHz is approximately 50 dB amplification which is close to the 49.4 dB in the simulation. Therefore it can be concluded that the 40 kHz amplifier works as calculated.

A frequency response is also made for the 25 kHz system and is tested from 1000 Hz to 70 kHz. It can be seen on figure 6.16.

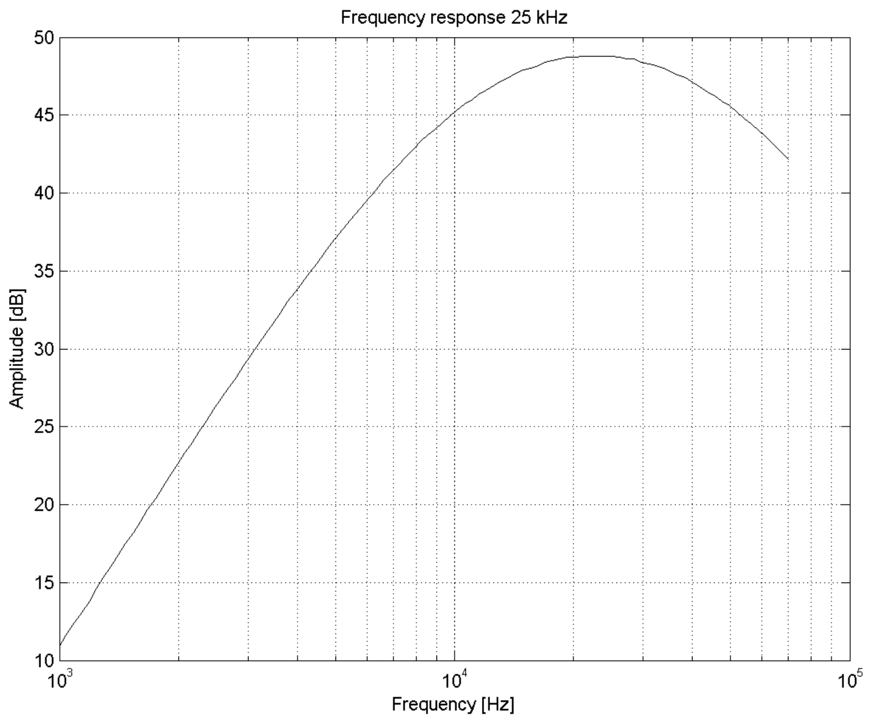


Figure 6.16: Output from the 25 kHz.

It can be read that the attenuation at 10 kHz is 3-4 dB which match the attenuation from the simulation for 25 kHz. At 70 kHz the signal is attenuated with approximately 6 dB which is 3 dB less then the simulation (refer to CD(/CD/LTspice/Amplifier and comparator ultrasound 25 kHz)), this may be due to variations in component values. The dB level at 25 kHz is approximately 48 dB which is one dB lower than the simulated, this is a small aberration and do not have a big impact on the system. It can be concluded that the amplification for both systems works, and the frequencies reponses matches the simulated.

Comparator test

To test the comparator circuit an oscilloscope(AAU 33585) and a wave generator(AAU 08591) was used. The input from the wave generator was 12.5 mV and set to either 40 kHz or 25 kHz at the input to the amplifier. The reason for 12 mV is that it is the lowest signal the receivers will receive from the transmitters at 4 m refer to appendix A.16. The output for 40 kHz was 2.86 V and has a period on 25 μs which is 40 kHz, it can be seen on figure 6.17.

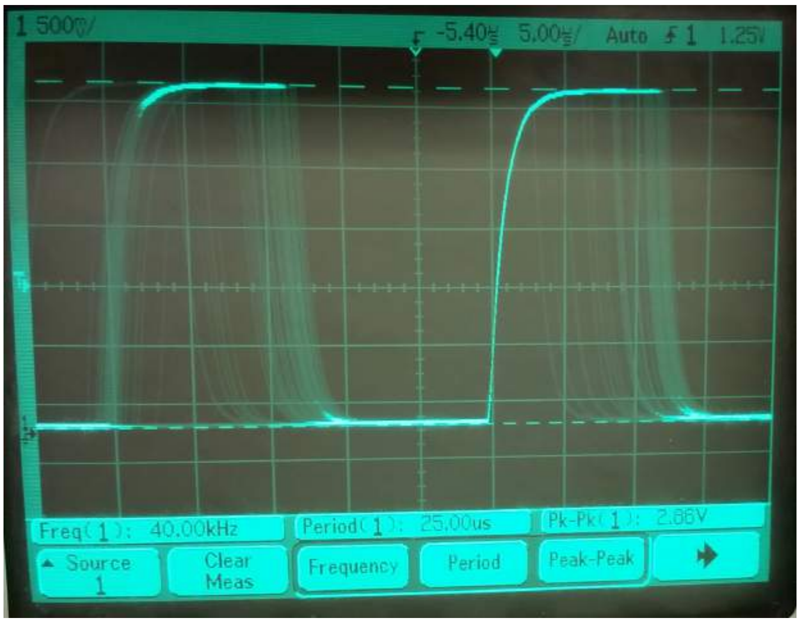


Figure 6.17: Output from the 40 kHz.

The voltage is not as high as in the simulation see figure 6.7, but it will still be able to trigger the input on the FPGA, therefore it will not be changed. For the 25 kHz it has the same result and therefore it will not be shown. Thus it can be concluded that the receiver system works.

Test of sensormodule

The receiver and transmitter part have been tested (refer to appendix A.11), to see if it could fulfill the requirements for distance and angle. On figure 6.18 it can be seen that the 40 kHz increases linearly over distance and the 25 kHz has larger variations than the 40 kHz.

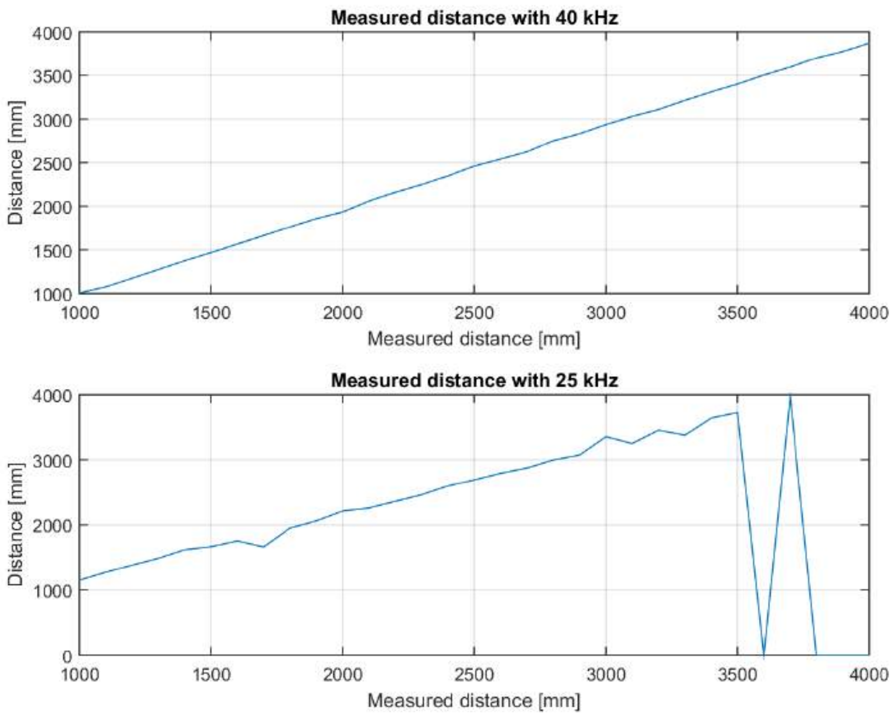


Figure 6.18: Plot for both measured distances.

It can be seen that the 40 kHz measured distance matches the distance up to 2500 mm, it has some small variations ± 50 mm. Between 2500 mm to 3500 mm it variates with 50 mm to 100 mm. After 3500 mm it starts to have variations bigger than 100 mm and the biggest variation is 129 mm. This is still close to requirement 6 but it will not fulfill the requirement, but it can be concluded that the system for 40 kHz works up to 2500 mm. As shown on figure 6.18 the 25 kHz has variation from the measured distance to the actual distance. It variates with more than 100 mm on nearly every reading and after 3500 mm it loses readings, therefore it will not fulfill the requirement. It has been concluded that there need to be made a new test with the 25 kHz with some changes to it, so it can fulfill the requirement.

The system is able to detect a signal from 0 ° to 90 ° and therefore requirement 7 is fulfilled for the angle.

For the second test a change was made to the transmitter system. Instead of using one 9 V battery it will be using two 9 V batteries in series, which will give the transmitter an 18 V supply. The transmitter system will not be able to have a higher voltage than 20 V_{pp} according to the datasheet for the 40 kHz ultrasounds transmitter [muRata, 2002]. The second test was conducted with fewer steps to see if the concept works and can be seen in appendix A.12. On figure 6.19 it shows that both 40 and 25 kHz increases linearly with the distance.

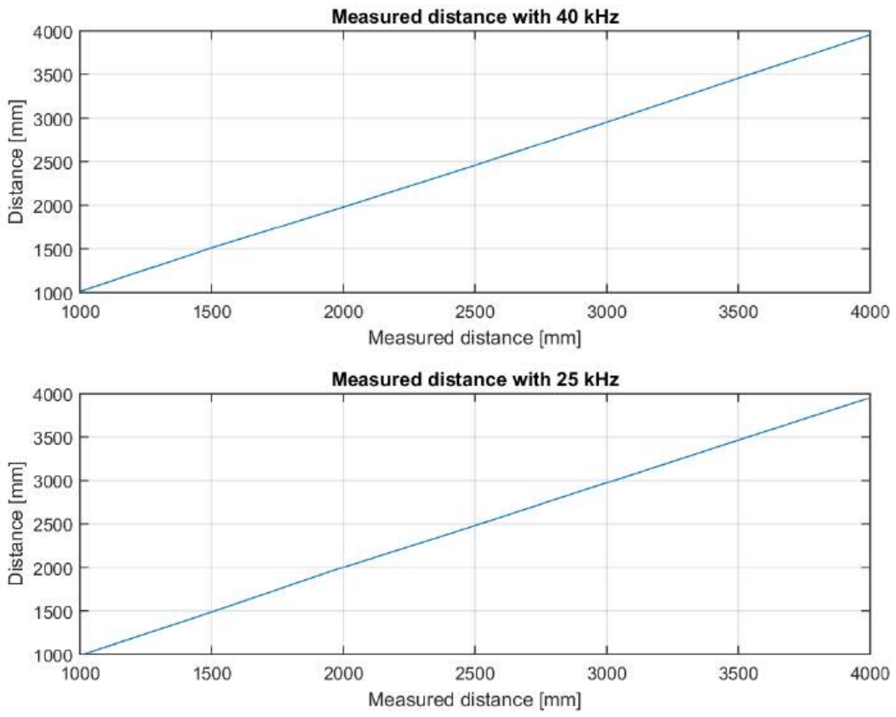
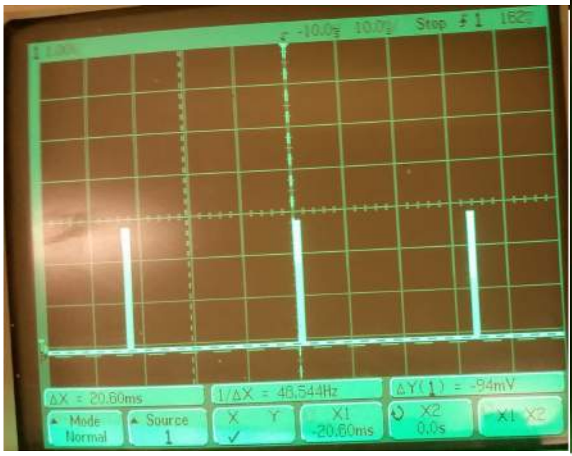


Figure 6.19: Plot for both measured distances.

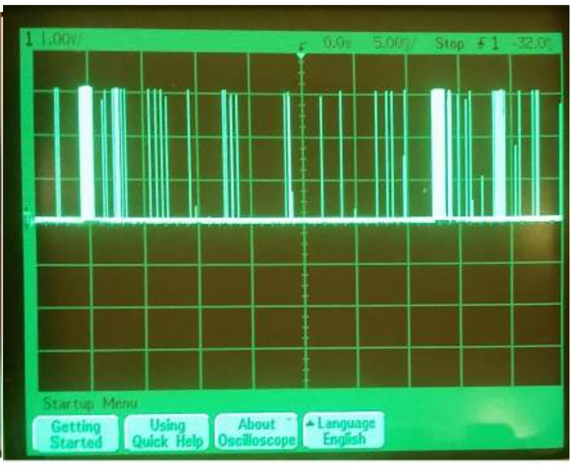
The result for both receivers shows, that from 1000 to 2000 mm it varies with ± 20 mm. After 2000 mm it starts to vary with ± 50 mm and therefore it can be concluded that the distance system fulfill the requirements.

Test of noise on the receiver module

The receiver system has been tested for noise when the motors were on to see if they interfered with the ultrasound receivers. To see the measurement report refer to appendix A.17. As shown on figure 6.20 b it is clear that the motors/propels adds noise to the receiver system compared to figure 6.20 a where the motors were turned off.



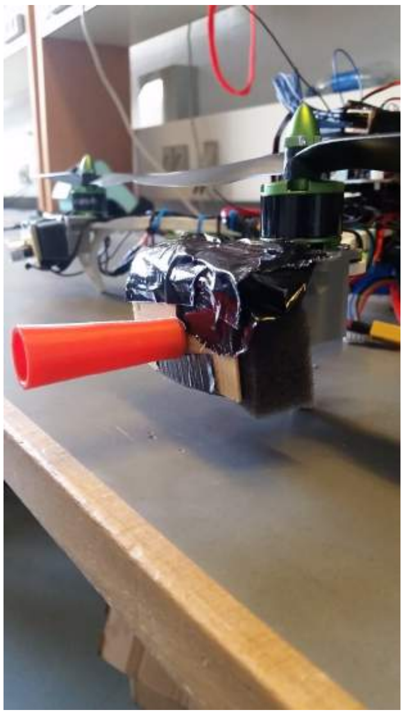
(a)



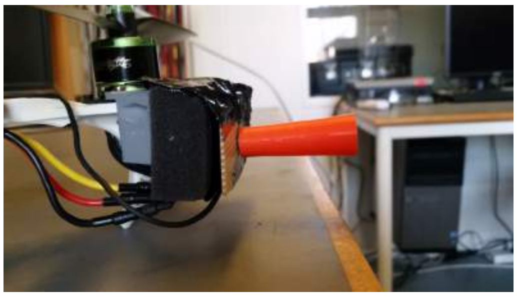
(b)

Figure 6.20: (a)Signal when the motors are turned off.(b)Signal received from the 40 kHz receiver with motor and propeller is on.

This noise will cause an incorrect measurement and therefore it must be removed. It was removed with some few modifications to the receivers. As shown on figure 6.21 there has been added a shielding to the 40 kHz sensor and instead of using screws to place the receiver on the wing, it is attached with duct tape and a vibration absorbing material.



(a)



(b)

Figure 6.21: (a and b) 40 kHz receiver final implementation on the quadcopter.

The shielding is placed to reduce noise from the propel and the duct tape is used to reduce the vibration from the motors together with the absorbing material. The disadvantages with the use of shielding is it limit the angle the receiver will be able to detect the ultrasound waves. But without the shielding the calculations can not be done and therefore

it is chosen to decrease the angle so it is able to make the distance calculations.
This setup removed the noise for the 40 kHz receivers, but there was still some noise on the 25 kHz receiver as shown on figure 6.22.

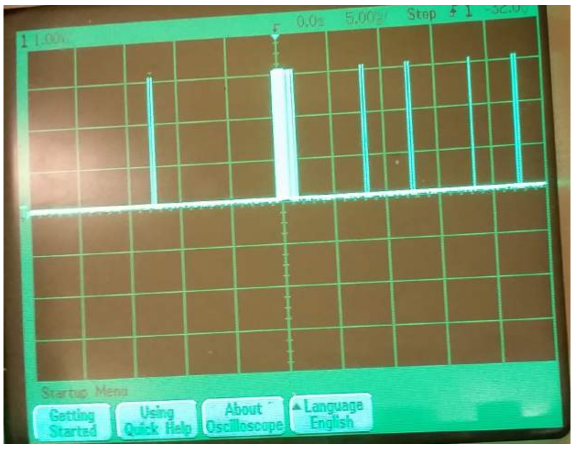


Figure 6.22: Noise reduced for 25 kHz.

To remove this noise the potentiometer on the comparator was increased so the reference voltage would be higher. A disadvantage with this is that it would decrease the range of the 25 kHz receiver, but after a test it shown that the 25 kHz receiver still worked up to 4 m. The potentiometer for 25 kHz is placed at 263 Ω and for 40 kHz it is placed at 294 Ω. Furthermore it has shown that the blue propels cause more noise in the 40 kHz spectrum than the black once, therefore it has been chosen to use the black propels.

The last test was conducted for the distance system to see if the motors had any interference on the receivers. To see the journal refer to appendix A.13. As shown on figure 6.23 the measured distance increases linearly for both measurements and looks similar to 6.19.

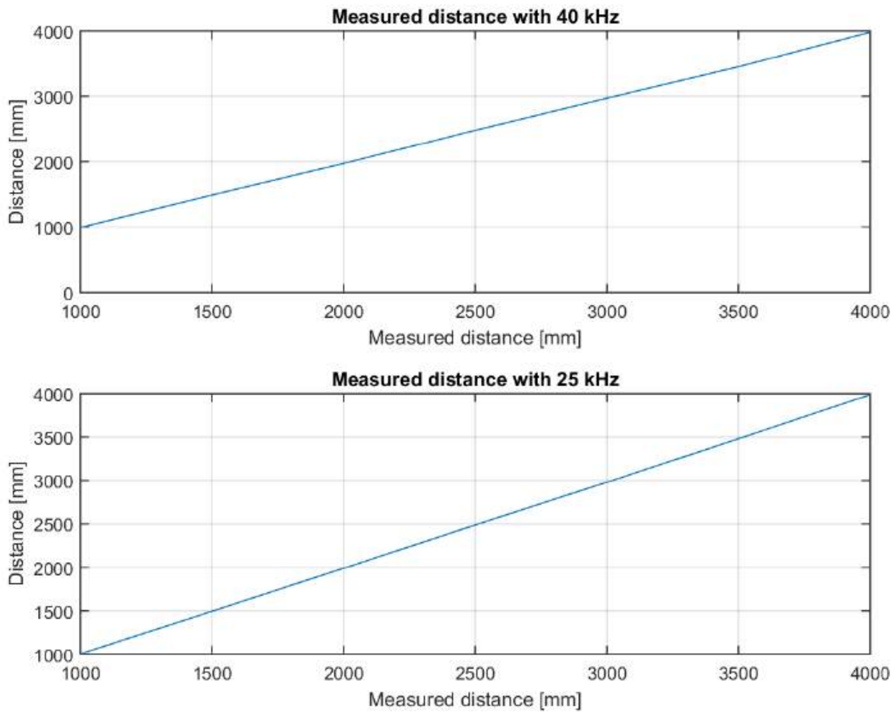


Figure 6.23: Plot over the measured distance.

Therefore it can be concluded that the distance system works while the motors are on. As described in measurement report A.13 the distance system maximum deviates with ± 50 mm for the 40 kHz system. Futhermore for the 25 kHz system the maximum deviation is 20 mm. Both system will therefore fulfill requirement 6 when the system is not in motion. Therefore this requirement is only partly fulfilled, the distance system needs to be tested in a dynamic motion to see if the distance system still calulates the distance correct. The system is also able to detect angles from ± 0 ° to 30 ° on both sensors. This test is also done while the quadcopter was static and therefore the requirement 7 is partly fulfilled. This test also need to be done for a dynamic movement to see if it still fulfills the requirement.

PPM modulation 7

To be able to implement a control system it has to be decided where the input shall be delivered on the quadcopter. The obvious choice might seem to be in between the 3DR PPM encoder and the Pixhawk. But there is also the possibillity to replace the 3DR PPM encoder entirely and modulate the PPM signal to the Pixhawk from the PWM signals from the RC receiver, on the FPGA. This will also eleminate any possible delay created by the 3DR PPM encoder when it is modulating the PPM signal and the delay created by the FPGA adding the control offset to the PPM signal. For this reason it has been decided to replace the 3DR PPM encoder with the FPGA. The original setup is shown on figure 7.1 and the modified on figure 7.2.

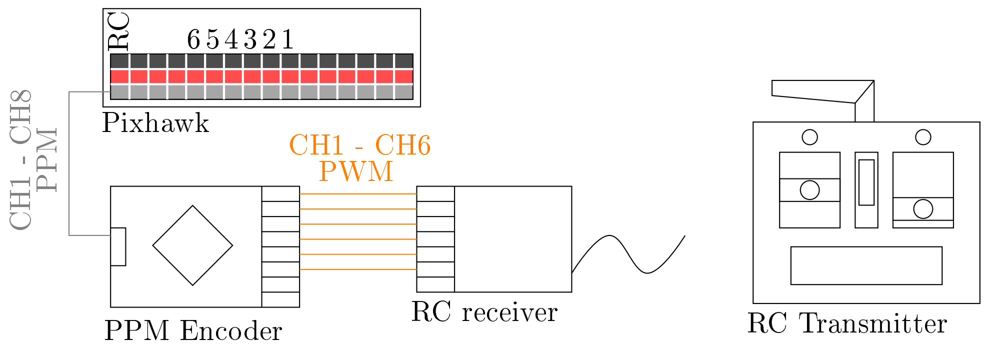


Figure 7.1: RC receiver, PPM encoder and Pixhawk setup.

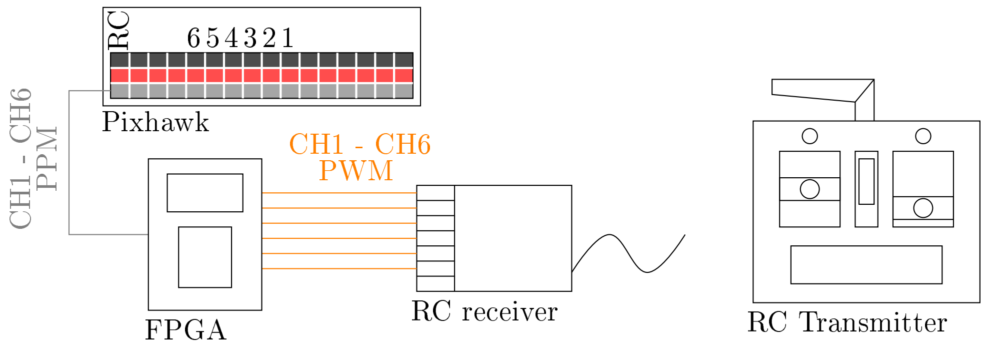


Figure 7.2: RC receiver, FPGA and Pixhawk setup

To be able to make the conversion, from six PWM signals to one PPM signal, it is needed to examine all the seven signals in order to create a signal which the Pixhawk accepts as a viable PPM signal. The six PWM signals has the charecteristic which is shown on figure 7.3.

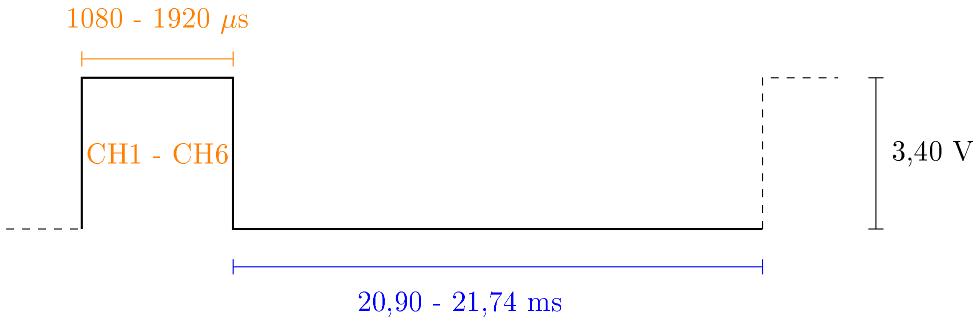


Figure 7.3: PWM signal charecteristic for all six channels from the RC receiver, where one frame consists of one yellow and one blue part A.9.

From appendix A.7 and A.9 the channel width has been found on all seven signals, and an comparison between the PWM channel and the PPM channel width is shown in table 7.1.

	PWM channel width	PPM channel width
Channel 1	1080 - 1920 μs	680 - 1520 μs
Channel 2	1080 - 1920 μs	680 - 1520 μs
Channel 3	1080 - 1920 μs	680 - 1520 μs
Channel 4	1080 - 1920 μs	680 - 1520 μs
Channel 5	1080 - 1920 μs	680 - 1520 μs
Channel 6	1080 - 1920 μs	680 - 1520 μs
Channel 7	N/A	1100 μs
Channel 8	N/A	1100 μs

Table 7.1: Channel width listed for PWM and PPM signals A.7, A.9

The first thing which needs to be taken in to accout in the modulation of the PPM signal is the 400 μs offset between the channels of PWM and PPM. The next parts are the framespace and channel seperation in the PPM signal which are shown on figure 7.4.

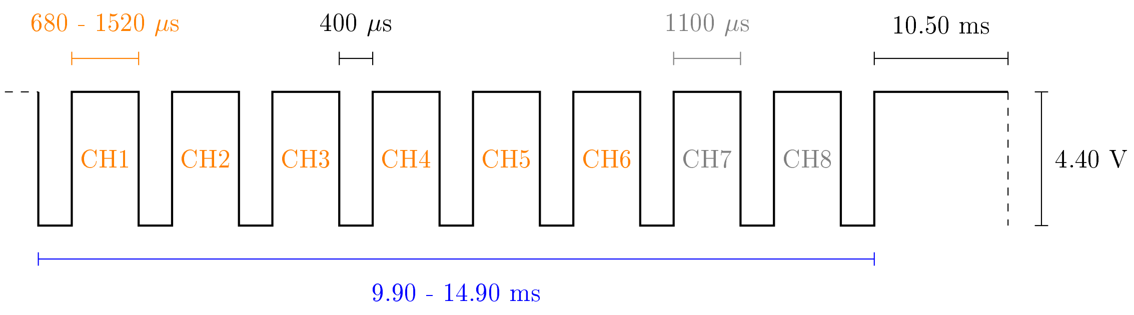


Figure 7.4: PPM signal output charecteristic from 3DR PPM encoder for all eight channels. Low part between channels is channel seperation and the high 10,5 ms period is frame spaceing A.7.

Both framespaceing and channel seperation lasts the same amout of time for every PPM frame. Even though the RC receiver only has six channels the 3DR PPM encoder has inputs for eight channels and modulates an eight channel PPM signal. Due to time constraints it has been decided to recreate all eight channels in the PPM signal on the FPGA to avoid any problem which could arise from a change from eight to six channels on the Pixhawk. This results in a variable frame time of 21,9 ms to 25,4 ms.

As to sum things up, what needs to be taken in to consideration when modulating the PPM signal is channel offset between PWM and PPM, frame spacing and channel seperation. For the ease of implementing the offset from the control system, steps will be made from -100 to 100 on yaw, pitch and roll, which have a center start position. Throttle will be implemented with steps from 0 to 200 as it starts from minimum. To activate the offsets on PPM signal, it has been decided to utillize a switch on the RC which gives an input on channel 5. The reason for this is to be able to get the quadcopter under control, if the control system under test yields an undesired result.

A crude diagram showing the functionality of PWM to PPM encoding on the FPGA is shown on figure 7.5.

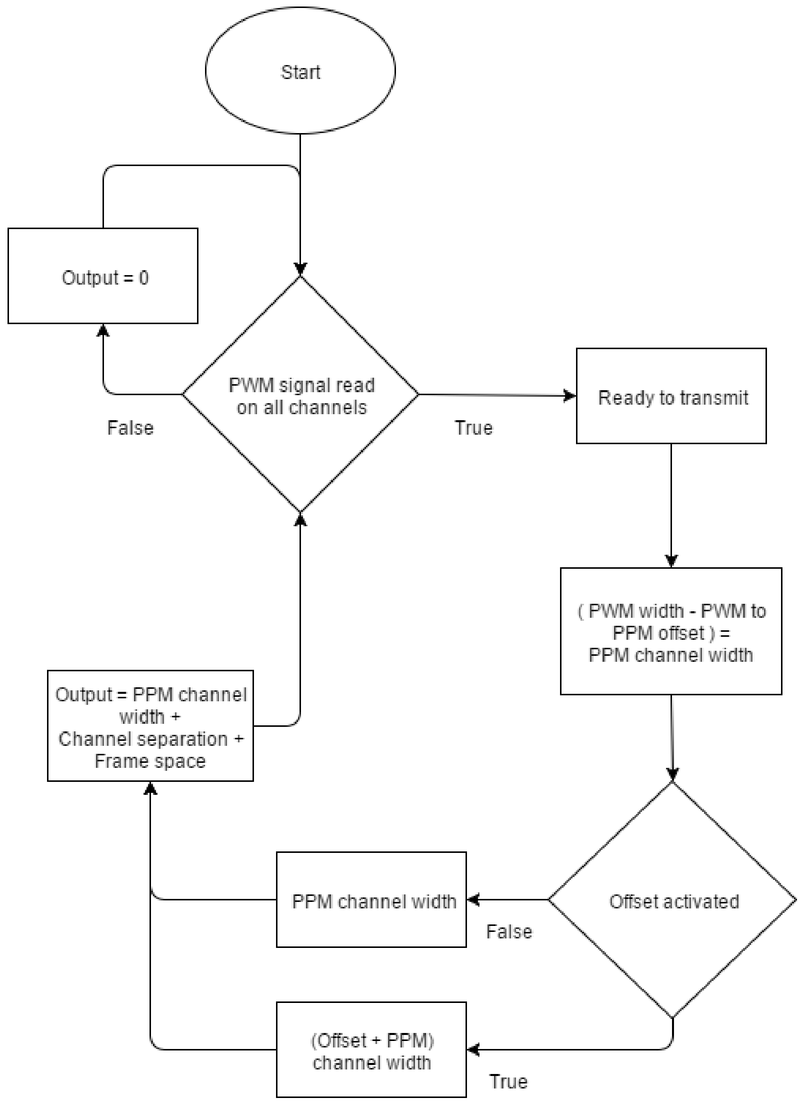

Start
Output = 0
PWM signal read on all channels
False
True
Ready to transmit
(PWM width - PWM to PPM offset) = PPM channel width
Output = PPM channel width + Channel separation + Frame space
PPM channel width
Offset activated
False
(Offset + PPM) channel width
True

Figure 7.5: Flowchart of PWM to PPM encoding

This leads to the implementation of the PWM to PPM encoding in VHDL which will be examined next.

7.1 VHDL implementation

The encoding will be seperated in to two modules, one for recording the PWM signal, and another to output the PPM signal.

PWM recording

The module in charge of recording the PWM signal is constructed with two functionalities. Firstly the module shall be able to record the PWM width, and secondly the module shall be able to reset the output if the PWM signal cease to exist. The second function is constructed to take hardware failure in the RC receiver in to accout if such should occur. The resolution of the recording is set to follow the system clock of the FPGA which is 12 MHz. This means that every 83,3 ns the module will check to see if the input port is still high. A diagram of the recording function is shown on figure 7.6.

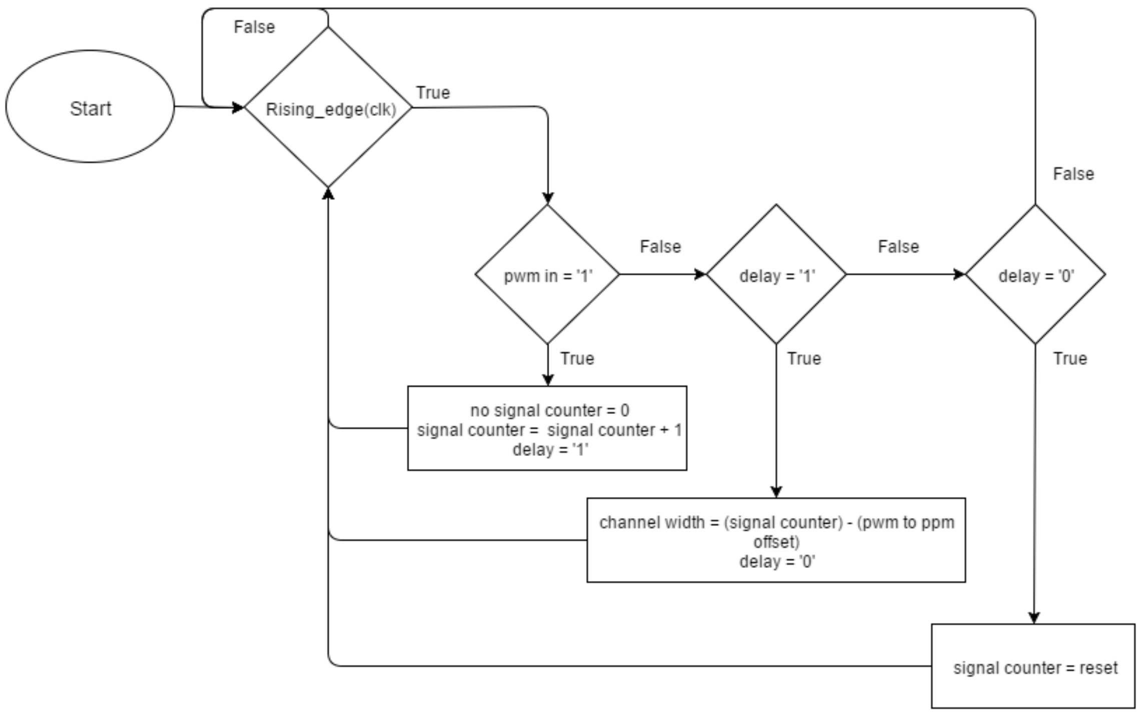


Figure 7.6: Flowchart of VHDL code design for PWM recording

The reset part of the recording module is set to count the low period of the PWM signal from the RC receiver which is 20,9 ms as mentioned in the measuring report (refer to appendix A.9). According to the measured low period the reset period is set to occur after 30 ms. This should prevent interference in normal operation but also make it possible to act swiftly in the event of an malfunction. A flowchart of the reset function can be seen on figure 7.7.

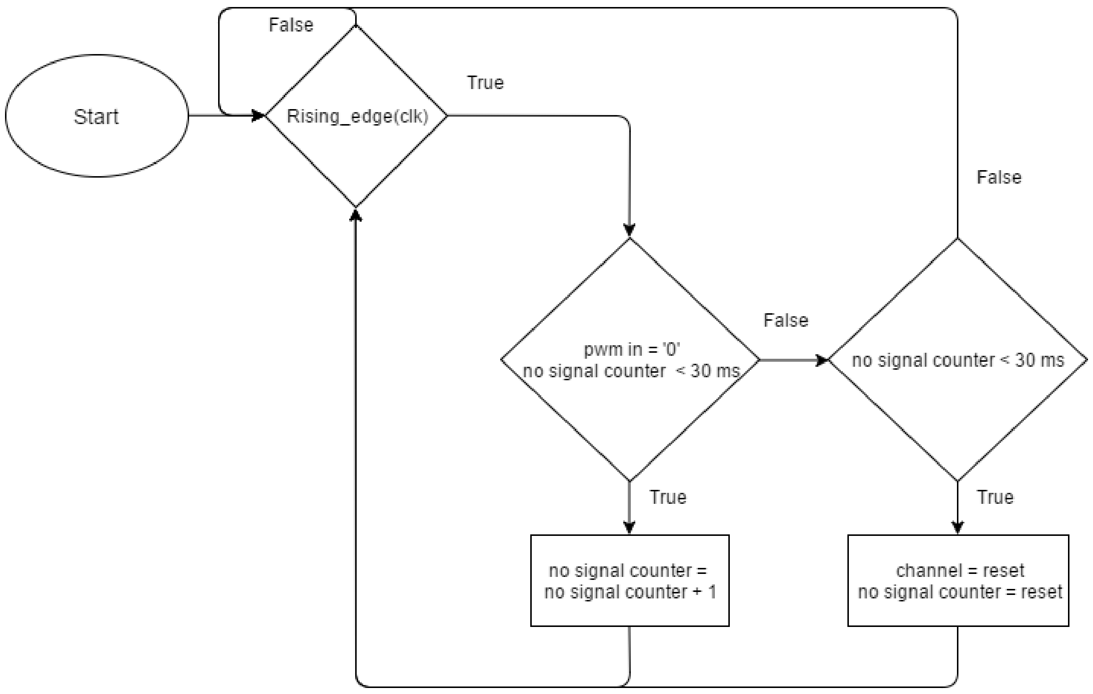


Figure 7.7: Flowchart of VHDL code design for reset of previously recorded input

The recording of the PWM signals and the reset function is implemented with the following ports and signals in the recording module.

- Ports
  - clk: Logic input which is switched high and low according to the FPGA's system clock.
  - pwmin: Logic input which connects to the FPGA input pin from the RC receiver.
  - ch: Integer (0 - 32767) output which sends the recorded PWM width to the PPM module.
- Signals
  - tempcount: Integer (0 - 32767) which is used to count.
  - chtemp: Integer (0 - 32767) which holds the measured PWM signal until its latched onto the output port ch.
  - delay: Logic signal which is used to delay the reset of the counter until the result has been latched into chtemp.
  - reset_output_timer: Integer which holds the amount of clock ticks which shall pass before output is reset.
  - nosignalcount: Integer which is used to count when input signal is low.
  - pwm_to_ppm_offset: Integer which holds the PWM to PPM offset clock ticks.

The recording function in VHDL is shown in code section 7.1.

```
if (rising_edge(clk)) then
        -- chn length recording
        if (pwmin = '1') then
                nosignalcount <= 0;
                tempcount <= tempcount +1;
                delay <= '1';

        elsif (pwmin = '0' and delay = '1') then
```

```
            chtemp <= (tempcount - pwm_to_ppm_offset);
            delay <= '0';

        elsif (pwmin = '0' and delay = '0') then
            tempcount <= 0;

        end if;
```

Code 7.1: Recording function in VHDL.

When there is a PWM signal on the input port of the FPGA, it is desired to measure the width of the high period. This is done with the variable tempcount, which as soon as there is a rising edge on the FPGA clock and pwmin is high, will increment by one. At the same time the counter variable nosignalcount for the low period of the PWM signal for the reset function is set to null. Furthermore the delay variable is set to one. The delay variable is set to be able to save the counted value before the counter is reset.

In code section 7.2 the code for the reset function is shown.

```
if ( pwmin = '0' and nosignalcount < reset_output_timer) then
    nosignalcount <= nosignalcount + 1;
elsif (nosignalcount = reset_output_timer) then
    chtemp <= 0;
    nosignalcount <= 0;
end if;
```

Code 7.2: Reset function in VHDL.

The reset function, operates much like the recording function in reverse where the variable nosignalcount increments when pwmin is low. When the preset amount of 360.000 hardcoded in to reset_output_timer has been reached, the variable chtemp is set to null, and the value will then be latched in to the output port in the module. The 360.000 ticks amount to 30 ms by the following equation:

$$30\text{ ms} \cdot \text{FPGA}_{\text{frequency}} = 30\text{ ms} \cdot 12\text{ MHz} = 360.000 \tag{7.1}$$

The implementation of the VHDL recording module in the top module is as shown in code section 7.3. By doing it this way the same module can be instigated for all six channels, without the need for a vdl file for each channel. It also means that there can not be made specific changes in the recording module for a specific channel.

```
pwm1 : pwmread PORT MAP(clk,PWM_IN(0),ch1);
pwm2 : pwmread PORT MAP(clk,PWM_IN(1),ch2);
pwm3 : pwmread PORT MAP(clk,PWM_IN(2),ch3);
pwm4 : pwmread PORT MAP(clk,PWM_IN(3),ch4);
pwm5 : pwmread PORT MAP(clk,PWM_IN(4),ch5);
pwm6 : pwmread PORT MAP(clk,PWM_IN(5),ch6);
```

Code 7.3: Multiple instigations of the PWM reader module.

To sum up the recording module, it is constructed to utilize the FPGA clock to record the individual PWM signals, which gives a sample rate of 12 MHz. It is also equipped with a simple reset function, which in case of a failure resets the previous recorded channel value. Nextæy the PPM output module will be examined.

PPM output

The PPM output module is constructed with three main functions, which is creating a PPM signal, applying offset from control system onto PPM signal and triggering control system offset appliance to the PPM signal, based on channel width of channel five, which is controlled by the switch marked on figure A.4 in appendix A.3. Before a PPM signal is output there has to be a recorded input from all PWM channels to prevent an unwanted PPM signal. A flowchart of the functionality of this function is shown on figure 7.8.

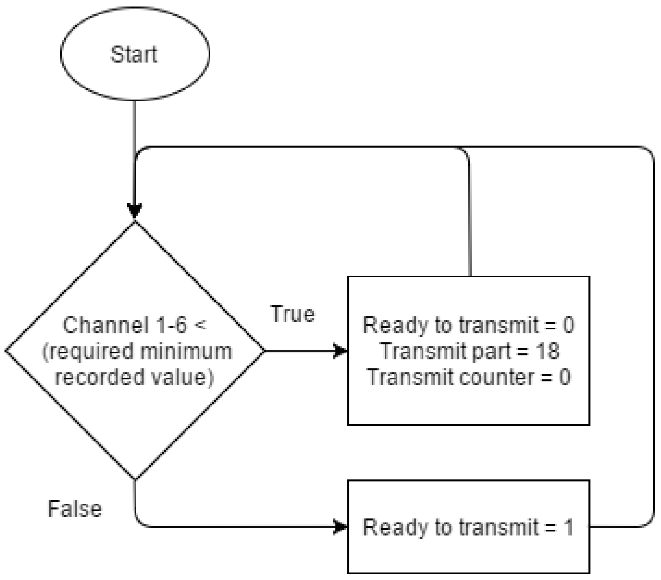


Figure 7.8: Flowchart of the loop which will trigger the start of PPM transmitting.

To differentiate between which part of the PPM frame that should be output, each part is assigned a number. The value of the output corresponding to the frame part is shown in table 7.2. Furthermore will a value of anything above 17 result in null being sent to the output port.

Part	Assigned number	Output port value	hardcoded value
			(ticks/time)
Framespace	0	1	yes (126001/10,5 ms)
Channel seperation	1	0	yes (4800/400 μs)
Channel 1	2	1	no
Channel seperation	3	0	yes (4800/400 μs)
Channel 2	4	1	no
Channel seperation	5	0	yes (4800/400 μs)
Channel 3	6	1	no
Channel seperation	7	0	yes (4800/400 μs)
Channel 4	8	1	no
Channel seperation	9	0	yes (4800/400 μs)
Channel 5	10	1	no
Channel seperation	11	0	yes (4800/400 μs)
Channel 6	12	1	no
Channel seperation	13	0	yes (4800/400 μs)
Channel 7	14	1	yes (13200/1100 μs)
Channel seperation	15	0	yes (4800/400 μs)
Channel 8	16	1	yes (13200/1100 μs)
Channel seperation	17	0	yes (4800/400 μs)

Table 7.2: Assigned value and output port value for each part of the PPM frame.

When "ready to transmit" has been triggered, the "transmit part" reset value of 18, will change to null. This will start the playback of the framespace part of the PPM frame. When the "transmit counter" has reached the hardcoded value of 126001 equal to 10,5 ms the counter is reset. The value of the next part of the PPM frame is latched and compared to "transmit counter" until it is equal to "Signal length". This makes the creation of a single PPM frame an 18 step process. On figure 7.9 the flowchart shows how the PPM signal is realised one part at a time.

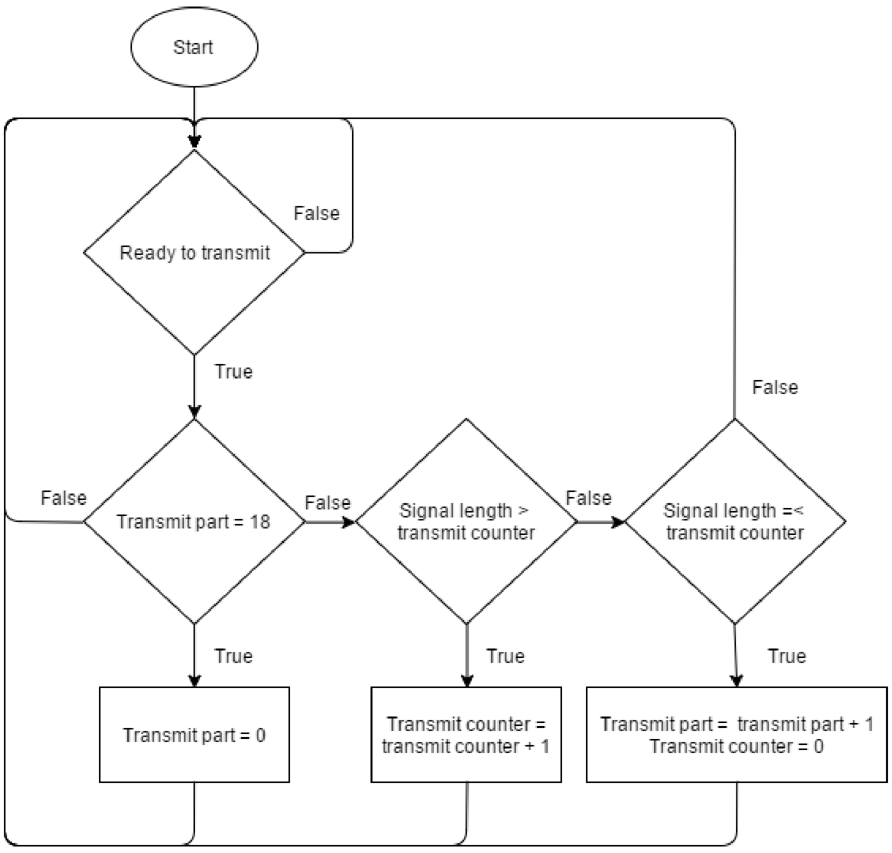

Start
Ready to transmit
False
True
Transmit part = 18
False
False
True
Signal length > transmit counter
False
True
Signal length =< transmit counter
False
True
Transmit part = 0
Transmit counter = transmit counter + 1
Transmit part = transmit part + 1
Transmit counter = 0

Figure 7.9: Flowchart of the loop which generates the PPM signal

The output is changed asynchronously by which part of the PPM frame is currently being generated. This could be done synchrounously but has been constructed otherwise to give a better overview in the VHDL code. To ensure that the control system can never give an input beyond the abillity of the RC, limiters have been put in place so that the offset from the control system can never set the channel width below 680 μs or beyond 1520 μs according to appendix A.7.

Another function which has been taken in to account is throttlestop which is a safety feature to force throttle to a minimum by flipping a switch on the RC A.3. Throttlestop works by setting the width of channel 1 to 500 μs which is outside the normal lower boundary of 680 μs. To prevent throttlestop is affected by an offset or the offset minimum channel lengths limit statement shown inside the green box on figure 7.10.

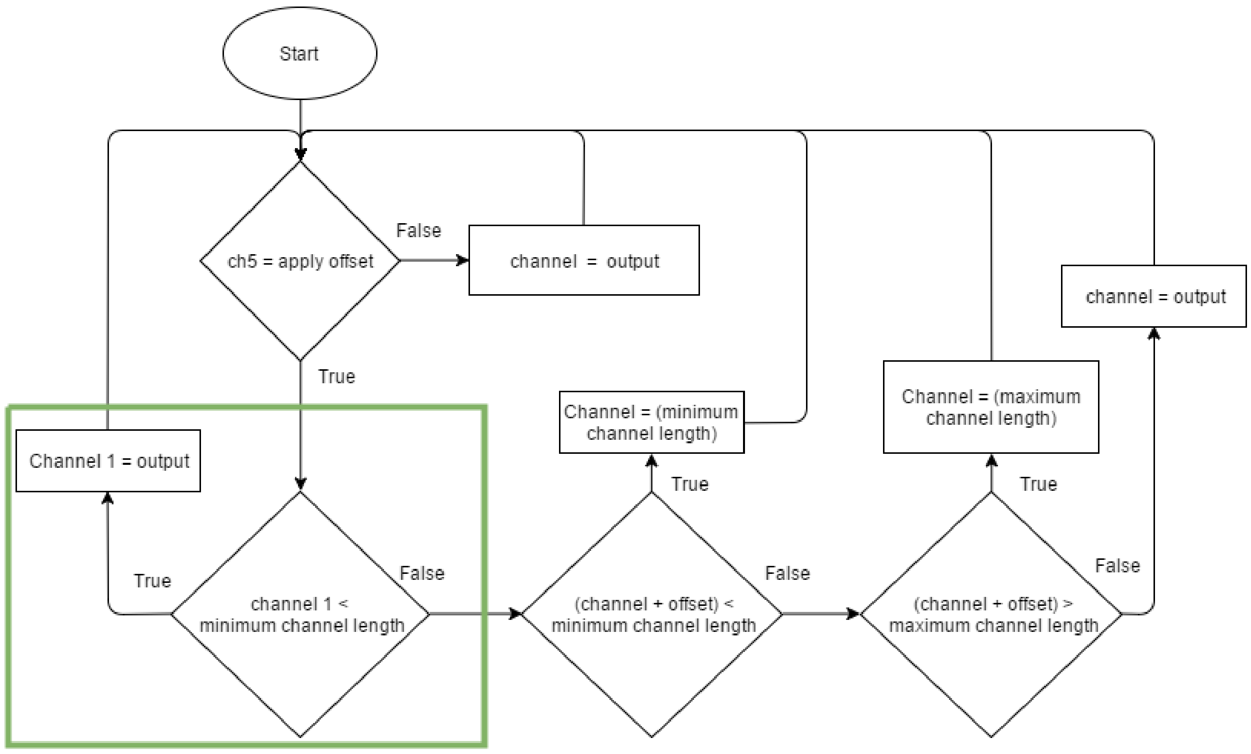

Start
ch5 = apply offset
False
True
channel = output
channel = output
Channel = (minimum channel length)
Channel = (maximum channel length)
Channel 1 = output
True
channel 1 < minimum channel length
False
True
(channel + offset) < minimum channel length
False
True
(channel + offset) > maximum channel length
False

Figure 7.10: Flowchart of the offset appliance with maximum and minimum limits. Statement in the green box is specific to channel 1.

Because throttlestop has been implemented, the required minimum value is set to 450 μs or equal to 450 μs$\cdot$12 MHz = 5400 ticks. This ensures that throttlestop will always be able to override any setting and function. In the following part the VHDL code implementation will be examined.

The following ports and signals is used in the PPM output VHDL module:

- Ports
  - clk: Logic input which is switched high and low according to the FPGA's system clock.
  - ppmout: Logic output which connects to a physical output port.
  - chn1: Integer (0 - 32767) input from channel1.
  - chn2: Integer (0 - 32767) input from channel2.
  - chn3: Integer (0 - 32767) input from channel3.
  - chn4: Integer (0 - 32767) input from channel4.
  - chn5: Integer (0 - 32767) input from channel5.
  - chn6: Integer (0 - 32767) input from channel6.
  - offsetchn1: Integer (-256 - 255) input offset from control system for chn1.
  - offsetchn2: Integer (-128 - 127) input offset from control system for chn2.
  - offsetchn3: Integer (-128 - 127) input offset from control system for chn3.
  - offsetchn4: Integer (-128 - 127) input offset from control system for chn4.
- Signals
  - framespace: Integer which holds value of clock ticks which corresponds to 10,5 ms
  - chnsep: Integer which holds value of clock ticks which corresponds to 400 μs
  - maxchnlength: Integer which holds the limit for maximum value of channel + offset.
  - minchnlength: Integer which holds the limit for minimum value of channel - offset.

- channel1: Integer (0 - 32767) which holds value of chn1 + offsetchn1.
- channel2: Integer (0 - 32767) which holds value of chn2 + offsetchn2.
- channel3: Integer (0 - 32767) which holds value of chn3 + offsetchn3.
- channel4: Integer (0 - 32767) which holds value of chn4 + offsetchn4.
- percent_step: Integer which hold the amount of ticks for how much a one percent offset step does.
- rdytotx: Logic signal which is used to initiate the transmisson of the PPM signal.
- tx: Integer (0 - 31) which is used to indicate which of the 18 parts is to be transmitted.
- txsignal: Integer (0 - 126001) where the recorded or hardcoded part values is held to be compared with the transmit counter.
- txcounter: Integer (0 - 126001) used to count by incrementing with one per clock cycle for the duration of the PPM frame part.
- minchnval: Integer hardcoded with the minimum value which must be recorded on all PWM channels before transmission of an PPM signal can begin.
- chn5trigger: Integer used to store the trigger value of the switch connected to channel 5 on the RC (value set at 12000(1 ms)).
- chn7: Integer hardcoded with channel 7 default measured value (13200).
- chn8: Integer hardcoded with channel 8 default measured value (13200).

The VHDL code in code section 7.4 shows if the minimum value is not met for all channels then the start trigger rdytotx, the transmit part value tx and the transmit counter txcounter is reset to null.

```vhdl
if (chn1 < minchnval or chn2 < minchnval or chn3 < minchnval or chn4 <
    minchnval or chn5 < minchnval or chn6 < minchnval) then
        rdytotx <= '0';
        tx <= 18;
        txcounter <= 0;
else
        rdytotx <= '1';
end if;
```

Code 7.4: VHDL code of the minimum recorded value start trigger.

If minimum channel width is met the start trigger is set and the modulation of the PPM signal shown in code section 7.5 is started.

```vhdl
if (rdytotx = '1') then
        if (txsignal > txcounter) then
                txcounter <= txcounter + 1;
        elsif (txsignal <= txcounter) then
                tx <= tx + 1;
                txcounter <= 0;
        end if;

        if ( tx = 18) then
                tx <= 0;
        end if;
end if;
```

Code 7.5: VHDL code modulation of PPM.

The transmit part variable tx is set to zero, and the rotation according to table 7.2 begins. The part length designated to the value of tx is latched into txsignal and txcounter will increment for every clock cycle until the value of txcounter equals the value of txsignal. The counter is then reset and the procedure starts again with the next part. In code section 7.6 a small part of this function is shown to give an example of how the different PPM part values is latched in to txsignal.

```
if tx = 0 then
        txsignal <= framespace;
elsif tx = 1 then
        txsignal <= chnsep;
elsif tx = 2 then
        txsignal <= channel1;
elsif tx = 3 then
        txsignal <= chnsep;
```

Code 7.6: VHDL code showing a snippet of how the PPM frame part value is latched in to txsignal.

Code section 7.7 shows how the logic state of the output pin connected to the Pixhawk is changed accordingly to the part of the PPM signal being modulated.

```
with tx select
        ppmout <= '1' when 0,
                  '1' when 2,
                  '1' when 4,
                  '1' when 6,
                  '1' when 8,
                  '1' when 10,
                  '1' when 12,
                  '1' when 14,
                  '1' when 16,
                  '0' when others;
```

Code 7.7: VHDL code of the asynchronous setting of the output port depending on the value of the part being transmitted.

In code section 7.8 it is shown how the offset is not applied before chn5 has a higher value than chn5trigger. Because the offset is given in percent the offset needs to be scaled to the PPM signal. This step is found by the following equation:

$$\frac{840\ \mu\text{s} \cdot 12\ \text{MHz}}{200\ \%} = 50,4 \approx 51 \tag{7.2}$$

The percentage step is then multiplied with the approximated value, and applied to the channel value as shown in code section 7.8.

```
if ( chn5 > chn5trigger) then
        if ( chn1 < minchnlength ) then
                channel1 <= chn1;
        else
                if (chn1 + (offsetchn1 * percent_step) < minchnlength) then
                        channel1 <= minchnlength;
                elsif (chn1 + (offsetchn1 * percent_step) > maxchnlength) then
                        channel1 <= maxchnlength;
                else
                        channel1 <= chn1 + (offsetchn1 * percent_step);
                end if;
        end if;

        if (chn2 + (offsetchn2 * percent_step) < minchnlength) then
                channel2 <= minchnlength;
        elsif (chn2 + (offsetchn2 * percent_step) > maxchnlength) then
                channel2 <= maxchnlength;
        else
                channel2 <= chn2 + (offsetchn2 * percent_step);
        end if;

else
        channel1 <= chn1;
        channel2 <= chn2;
        channel3 <= chn3;
        channel4 <= chn4;
end if;
```

Code 7.8: VHDL code of the offset activation and the maximum and minimum limits for offset + channel impact on the ppm signal. Code for channel 3 and 4 has been left out as it is identical to channel 2

To sum up the PPM module, it is constructed to do repetetive output of an PPM frame where offset can be applied by triggering a switch on the RC. Next the two modules will be tested to see if the designed functionallity function as expected.

7.2 PPM encoding verification

The PWM recording and the PPM module has been tested according to the specifications stated in section 4.4, for a complete measurement report refer to appendix A.8. The results attained for the precision of the PPM parts is shown on table 7.3.

Signal part	PPM [μs]	PWM [μs]	Expected (μs)	Deviation (μs / %
0 (Framespace)	10500,4		10500	+0,4 / < 0,001
1 (Chn sep)	400,12		400	+0,12 / < 0,001
2 (Chn 1)	680,28	1080,2	680,2	+0,08 / < 0,001
3 (Chn sep)	400,08		400	+0,08 / < 0,001
4 (Chn 2)	1098,96	1498,88	1098,88	+0,08 / < 0,001
5 (Chn sep)	400,12		400	+0,12 / < 0,001
6 (Chn 3)	1098,96	1498,88	1098,88	+0,08 / < 0,001
7 (Chn sep)	400,12		400	+0,12 / < 0,001
8 (Chn 4)	1098,96	1498,88	1098,88	+0,08 / < 0,001
9 (Chn sep)	400,08		400	+0,08 / < 0,001
10 (Chn 5)	680,28	1080,2	680,2	+0,08 / < 0,001
11 (Chn sep)	400,08		400	+0,08 / < 0,001
12 (Chn 6)	680,28	1080,2	680,2	+0,08 / < 0,001
13 (Chn sep)	400,12		400	+0,12 / < 0,001
14 (Chn 7)	1100,12		1100	+0,12 / < 0,001
15 (Chn sep)	400,08		400	+0,08 / < 0,001
16 (Chn 8)	1100,12		1100	+0,12 / < 0,001
17 (Chn sep)	400,2		400	+0,2 / < 0,001
Resolution test with hardcoded values				
chn8 + 1 tick (83,3 ns)	1100,2		1100,2	0 / 0
chn8 + 12 ticks (1 μs)	1101,12		1101,12	0 / 0

Table 7.3: Measured and expexted PPM part values (expected channel values is measured PWM - (PWM to PPM offset(400 μs) A.7, A.9.

The limiters and throttlestop test results is shown on table 7.4.

Channel	Offset [%]	Offset measured [μs]	Expected [μs]	deviation [μs] / [%]
1	-50	680,1	680 (limit)	+0,1 / < 0,001
1	50	1520,12	1520 (limit)	+0,12 / < 0,001
1*	50	501,34	501,36 ((PWM 901,36)- 400)	-0,02 / < -0,001
2	50	1311,38	1311,38	0 / 0
3	-50	886,44	886,38	+0,06 / < 0,001
4	50	1311,46	1311,38	+0,08 / < 0,001

Table 7.4: Offset values applied when switch is activated on OrangeRx T-SIX RC (* throttle stop activated)

The results attained for the delay test is shown on table 7.5.

	FPGA	3DR PPM encoder
Test 1	200 ns	48,8 ms
Test 2	200 ns	74,8 ms

Table 7.5: Measured delays from the lastest falling edge of the first wave of PWM signals (above minimum length of 450 μs) to the start of the PPM frame.

Due to these test it can be concluded that the constructed modules with the included offset limiters work, and does so with a precision which deviates less than one microsecond in channel width. The precision could be improved with ease, but is not done so due to the results attained and time constraints. It can also be concluded that the delay in the constructed system is relatively low. In the comparison with the 3DR PPM encoder the amount of delay between the two systems is inconclusive but is assumed to not be a problem. The same is assumed with the observed irregullarity on channel 2 of the PWM signal.

Controlling the quadcopter 8

To be able to control the quadcopter autonomously on certain axes, models of the movement for the quadcopter is needed.

Models of pitch, roll and yaw is needed, and these can be derived analytically but because of the Pixhawk own control model this is deemed impossible. The models will therefore be based on experimentally responses of pitch, roll and yaw. This chapter will go through the experimental responses, modelling of the movement, implementation of the controllers in VHDL and test of control controllers.

8.1 Modelling the movement of the quadcopter

The models can either be determined by a impulse response or a step response, where an impulse response applies 100 % of power for the shortest time possible which would be one frame and a step response is a constant step until the quadcopter has reached steady state. There are pros and cons for using either method, but a step response is choosen to be the used method because an impulse would be counteracted by the Pixhawk's own control system, making it drift instead of accelerating. The con of using a step response is the limited size of MTLab (refer to section 4.3) which could make a full step difficult to complete, but it is assessed that these measurement will be the most accurate data set.

To record a step response (refer to appendix A.15), a signal from the RC transmitter triggers a unit step to be sent from the FPGA to the Pixhawk, this will make the quadcopter go either forward, sideway or rotate. The distance over time is then measured using the Vicon system in MTLab. Because of the restricted size of MTLab the power of pitch and roll are reduced to get as full a step as possible. The power percentages of the different steps are as follows:

- Pitch: 20 %.
- Roll: 20 %.
- Yaw: 100 %.

The step response of pitch, roll and yaw can be seen on figure 8.1, 8.2 and 8.3.

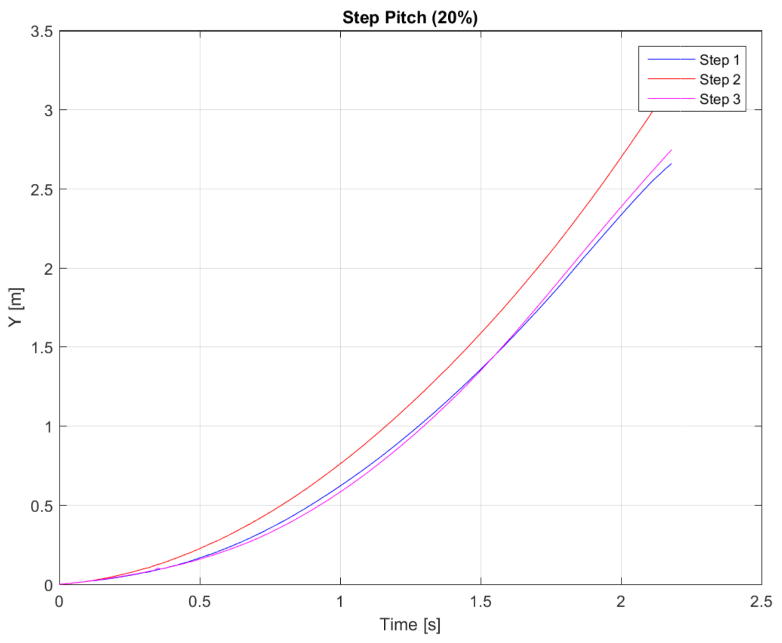


Figure 8.1: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

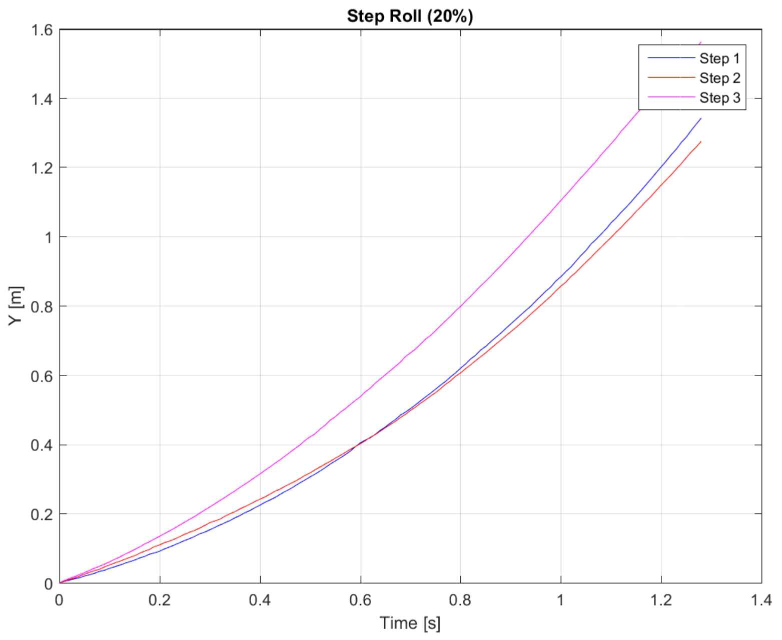


Figure 8.2: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

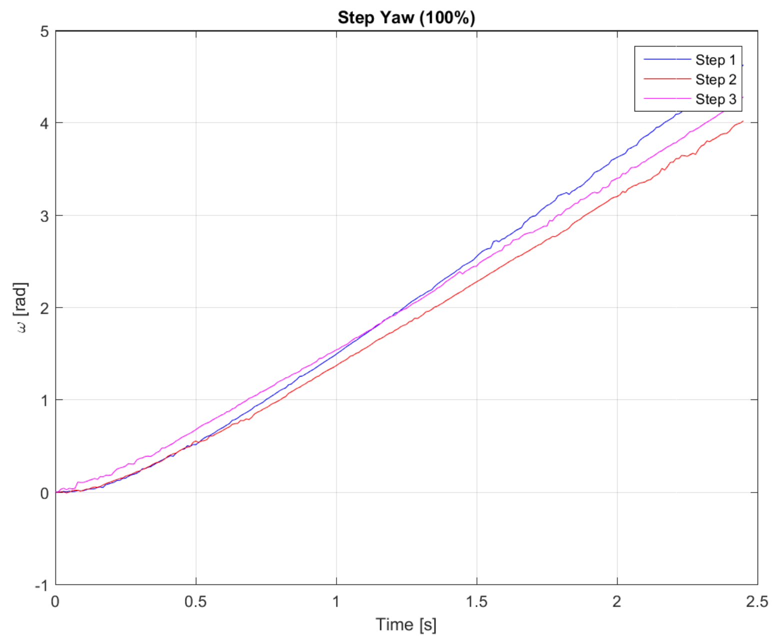


Figure 8.3: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

To simplify the models the three systems will be estimated as first order systems which gives the following template for the model:

$$H_y = \frac{K}{\tau s + 1} \cdot \frac{1}{s} \qquad [\cdot] \quad (8.1)$$

From these step responses the derivative removes the integrator, which simplifies the estimation of K and τ. The DC gain, K can be found as the magnitude of the steady state of the response and τ can be found as the time when the system has reached 63 % of its steady state. It is shown on figure 8.1 and 8.2 that these responses never reaches steady state, the estimation of K and τ is done by knowing that τ can be found as the time where the tangent of steady state crosses the time axis, and the curve of the tangent is equal to K as shown in equation 8.3.

$$\mathcal{L}^{-1}(H_y) = K(t - \tau[1 - e^{-t/\tau}]) = y \qquad [\cdot] \quad (8.2)$$

If assumed t goes towards infinity then following approximation can be made:

$$y = K(t - \tau) \qquad [\cdot] \quad (8.3)$$

Firstly the derived step response of yaw can be seen on figure 8.4.

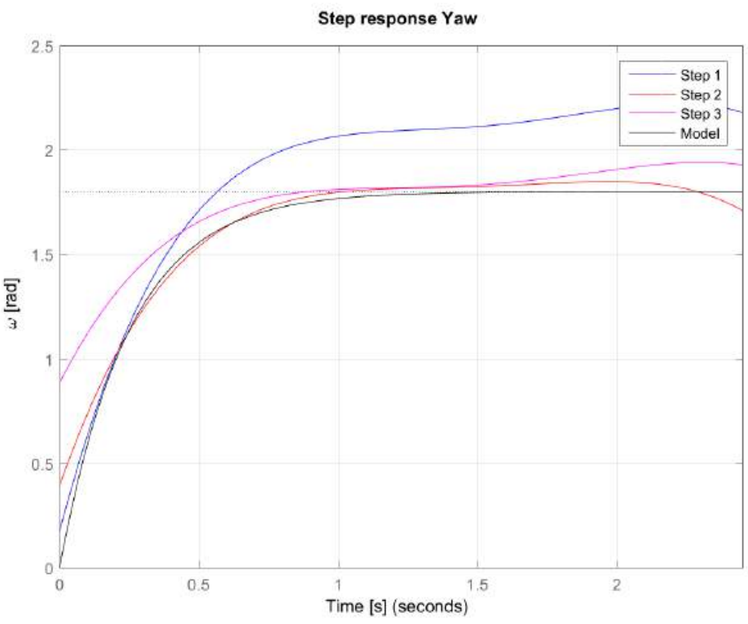


Figure 8.4: Derived step response of yaw compared to model.

From figure 8.4 K and τ is estimated from step 2 and 3. The amplification K must be divided by 100 because the step was performed at 100 % and therefore the following values are estimated:

$K_\varphi = 0,018$
$\tau_\varphi = 0,25$

Using these values in the model (refer to equation 8.1) going from % to rad/s is:

$$100\ \% \cdot \frac{ds}{dt} H_\varphi = 100\ \% \cdot \frac{ds}{dt}\left(\frac{0,018}{0,25s+1} \cdot \frac{1}{s}\right) \qquad [\cdot] \quad (8.4)$$

It is therefore possible to evaluate the model on figure 8.4

Secondly the values K and τ for pitch is estimated from figure 8.1. Because the response has not reached a steady state it is difficult to approximate K and τ, but using trial and error the model is approximated closest to step 1 and 3 because these steps were closest to reaching steady state. The amplification K must be divided by 20 because the step was performed at 20 % therefore the values are estimated to be:

$K_y = 0,101$
$\tau_y = 0,85$

Using these values the model going from % to m is:

$$20\ \% \cdot H_y = 20\ \% \cdot \frac{0,101}{0,85s+1} \cdot \frac{1}{s} \qquad [\cdot] \quad (8.5)$$

This gives the following model shown on figure 8.5 in comparison to the measured step responses.

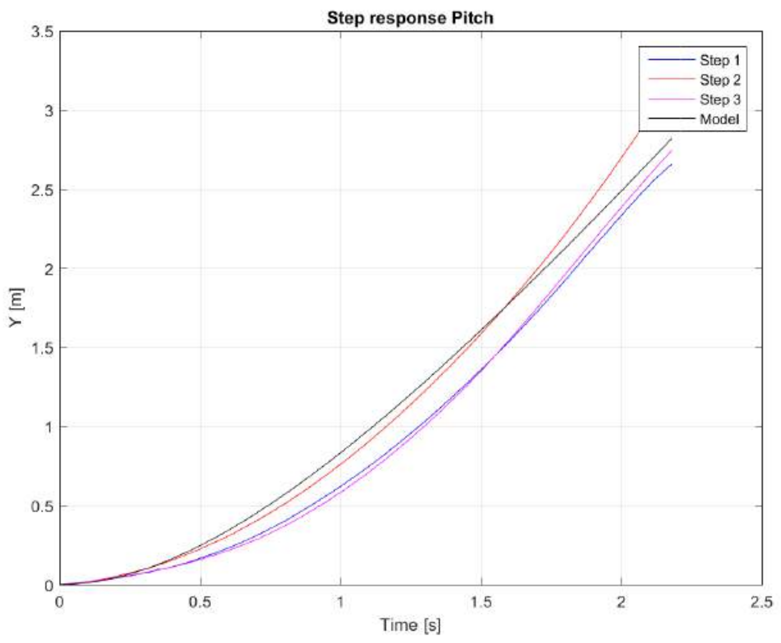


Figure 8.5: Step response of pitch compared to model.

Lastly the values K and τ for roll is estimated from figure 8.2. As pitch, the step response of roll had not reached steady state so approximating K and τ are also difficult. A model was approximated as close to step 1 and 2 which gives the following estimated values with K divided by 20:

$K_x = 0,09$
$\tau_x = 0,6$

Using these values the model going from % to m is:

$$20\ \% \cdot H_x = 20\ \% \cdot \frac{0,09}{0,6s+1} \cdot \frac{1}{s} \qquad [\cdot] \quad (8.6)$$

This gives the following model shown on figure 8.6 in comparison to the measured step responses.

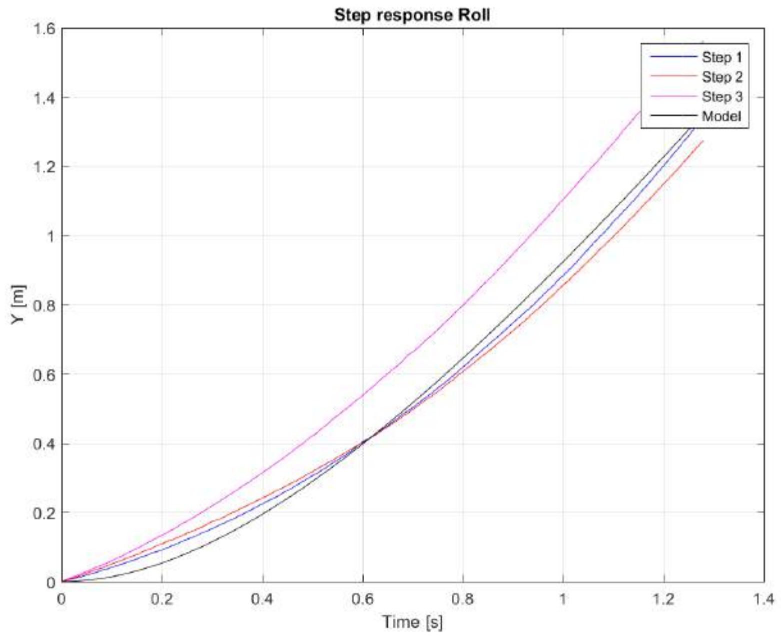


Figure 8.6: Step response of roll compared to model.

As all the models fit the measured step responses as desired, the models will be used to design the controllers for the quadcopter.

8.2 Controller design

To control the movement of the quadcopter it must be examined if any of the tested movement axes requires a controller. This examination is done by analysing the transfer functions of the created models for every axes with the sample rate of 31 ms as designed in chapter 6. The section will only go through the design phase of pitch, as it is the same procedure for yaw and roll.

The model of pitch is as follows:

$$H_y(s) = \frac{0,1}{0,85s+1} \cdot \frac{1}{s} \quad [\cdot] \quad (8.7)$$

And the sensor delay can be stated as:

$$G(s) = e^{-t_s s} \quad [\cdot] \quad (8.8)$$

Where:
t_s = 0,031 [s]

A sensor delay is needed because the system is a discrete system with the sampling rate of t_s and therefore the system will not react instantaneously on a change. This gives the following closed-loop system where the controller D(s) is undefined:

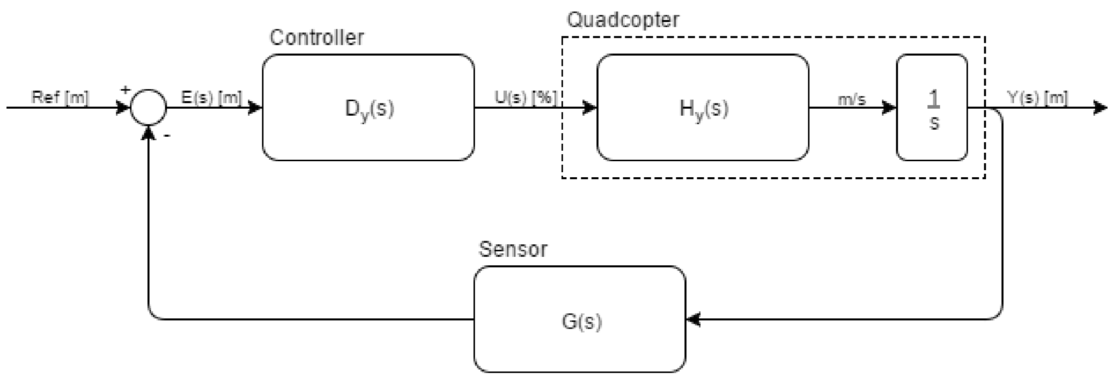


Figure 8.7: Closed loop system for pitch.

If the open loop of figure 8.7 is analysed using a bode plot (see figure 8.8) a controller can be designed.

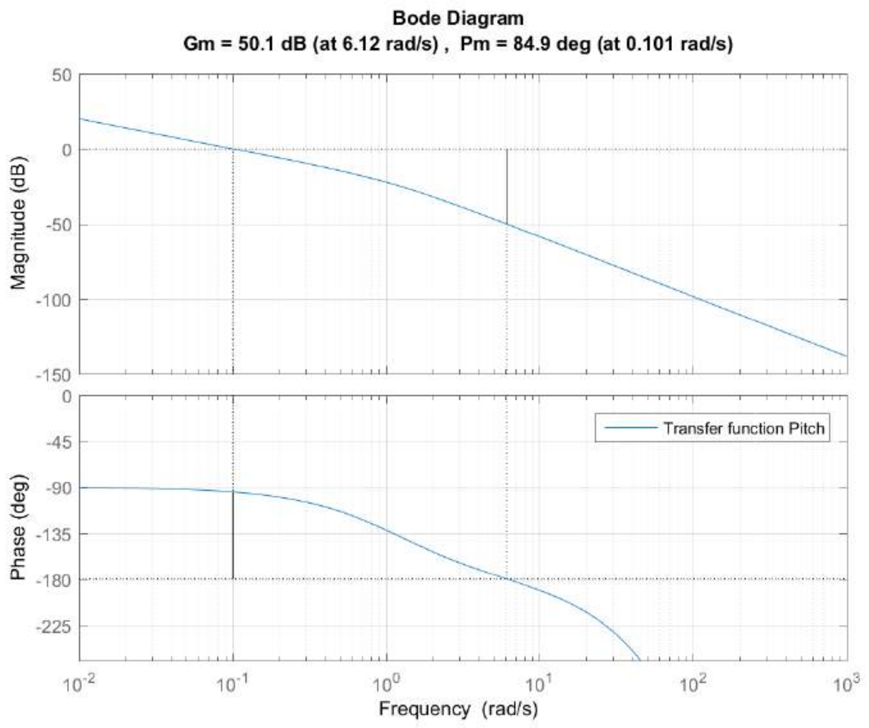


Figure 8.8: Bodeplot for open loop (Pitch).

As the bandwidth must be at least 20 times the sample rate of the closed loop system [Emami-Naeini, 2015]. A bandwidth of 10 rad/s is choosen as the bandwidth for pitch, which is derived from equation 8.9.

$$\omega = \frac{2\pi \cdot \frac{1}{t_s}}{20} \approx 10 \qquad \text{[rad/s]} \quad (8.9)$$

The Pixhawk itself introduces a pair of complex conjugated poles close to the imaginary axis which can make the system unstable. These two poles need to be damped to eliminate possible internal unstability. Therefore a derivative (D) controller (refer to equation 8.10) is needed. A D controller "knows" the slope of the error signal so it is said to have an anticipatory behavior. The D controller increases stability, reduces overshoot and has a sharp response to suddenly changing signals. The disadvantage to the D controller is that it tends to amplify noise because of the implemented zero [Emami-Naeini, 2015].

$$\frac{U(s)}{E(s)} = K_D \cdot s \qquad [\cdot] \quad (8.10)$$

To eliminate the disadvantage of amplifiying high frequency noise a pole is implemented with the D controller as seen in equation.

$$\frac{U(s)}{E(s)} = K_D \cdot \frac{s}{1 + t_p \cdot s} \qquad [\cdot] \quad (8.11)$$

Where $t_p \gg K_D$ to make sure the pole does not influence the increased stability made by the zero. From requirement 1 an overshoot of maximum 10 % can be derived. From this a phase margin of 60 ° is choosen which should give a overshoot of 10 % according to figure 8.9.

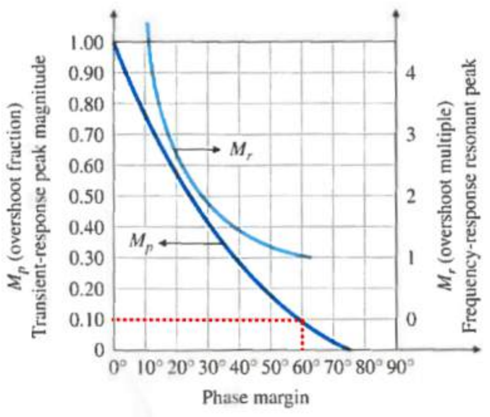


Figure 8.9: Transient-response overshoot (M_p) and frequency-response resonant peak (M_r) versus phase margin for second order systems. [Emami-Naeini, 2015]

Although this is for a second order system the approximation is used to determine the phase margin at 60 °. A D controller is almost never implemented without a proportional (P) controller (refer to equation 8.12) because the D controller does not supply information of steady state. The P controller opposite the D controller does supply information of steady state. If the gain K_P increases the steady state error decreases and furthermore the speed of the response increases, which creates an overshoot. This overshoot though is reduced by the D controller.

$$\frac{U(s)}{E(s)} = K_P \qquad [\cdot] \quad (8.12)$$

To conclude, a PD controller (refer to equation 8.13) is implemented with a bandwidth of 10 rad/s and a phase margin of 60 °.

$$D_y(s) = K_P \cdot (1 + K_D \cdot \frac{s}{1 + t_p \cdot s}) \qquad [\cdot] \quad (8.13)$$

A phase margin of 60 ° is obtained by implementing the D controller which introduces a phase shift of 90 °, but as only a 70 ° phase shift is required the D controller is implemented at 2,52 rad/s which gives a $K_D = 0,396$. To increase the bandwidth to 10 rad/s an amplification of 46,3 dB is needed by the P controller.

$$K_P = 10^{\frac{46,3dB}{20}} = 206,538 \qquad [\cdot] \quad (8.14)$$

This results in the following PD controller values:

$Bandwidth(BW) = 10.1$ [rad/s]
$K_P = 206,538$
$K_D = 0,396$

The pole of the D controller is inserted at 128 rad/s ($t_p = 0,0078$) to not influence the system.

With the PD controller implemented the following bode plot of the open-loop is, as shown on figure 8.13.

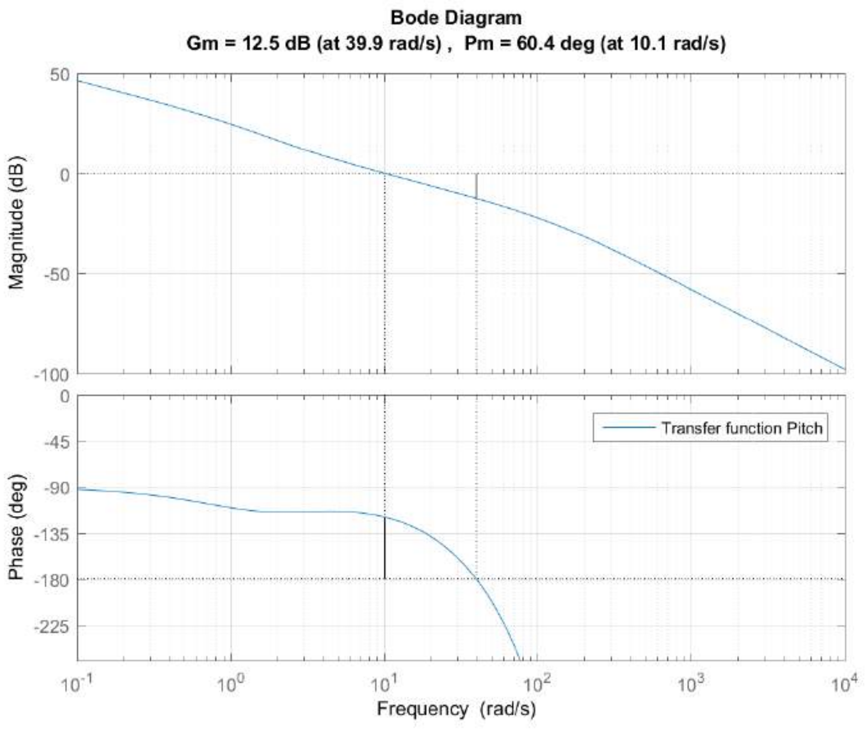


Figure 8.10: Bodeplot for open loop (Pitch) with PD controller.

Using the tool Simulink in Matlab it is possible to simulate a step response using the designed models and controllers. It is thereby possible to analyse the step response. The resulting Simulink block diagram for pitch is shown on figure 8.16.

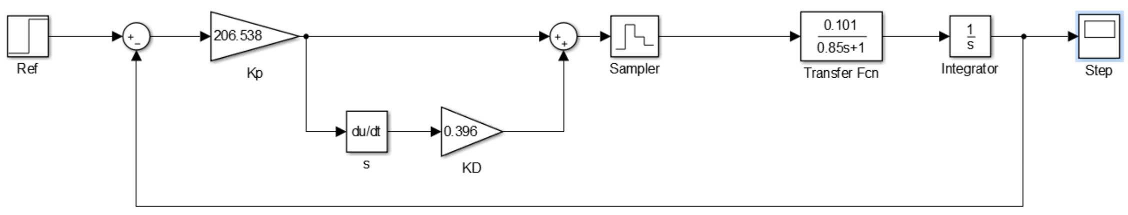


Figure 8.11: Simulink closed loop (pitch).

A step will be carried out at 0 s with a ref of 1 m. The Simulink shown on figure 8.11 is simulated with a sample time of $t_s = 0,031$ and the resulting step response of a step of 1 m is seen on figure 8.12.

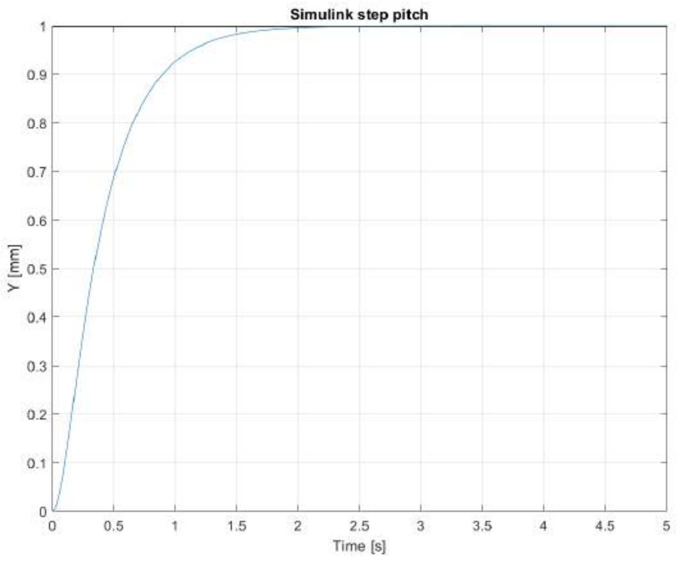


Figure 8.12: Step response from pitch simulation.

Test of the controller revealed problems regarding stability with this design as the quadcopter react violently leading to several crashes. Evaluating figure 8.12 the quadcopter should reach 70 % of the distance within half a second. As it is a simple model it does not take weight into account, it is likely the bandwidth of the quadcopter should be lowered, and therefore it has been decided to lower the bandwidth for pitch and work iterative to find the most optimal bandwidth.

This gives the following PD controller values:

$BW = 5,85$ [rad/s]
$K_P = 29,174$
$K_D = 2$
$t_p = 0,0078$

With the PD controller implemented the following bode plot of the open-loop is shown on figure 8.13.

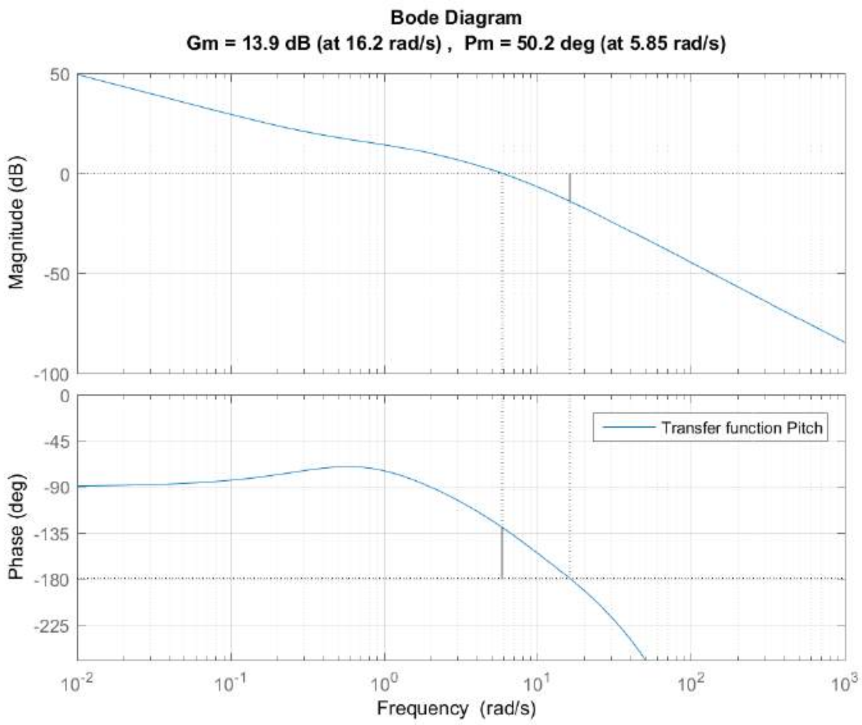


Figure 8.13: Bodeplot for open loop (Pitch) with PD controller.

It is assumed that yaw will not be as highly affected by weight as pitch and therefore the BW can be set higher which is implemented iterative.

The found PD controller values for yaw is as follows:

$BW = 12,3 rad/s$
$K_P = 225,164$
$K_D = 1,33$
$t_p = 0,0078$

With the PD controller implemented the following bode plot of the open-loop is shown on figure 8.14.

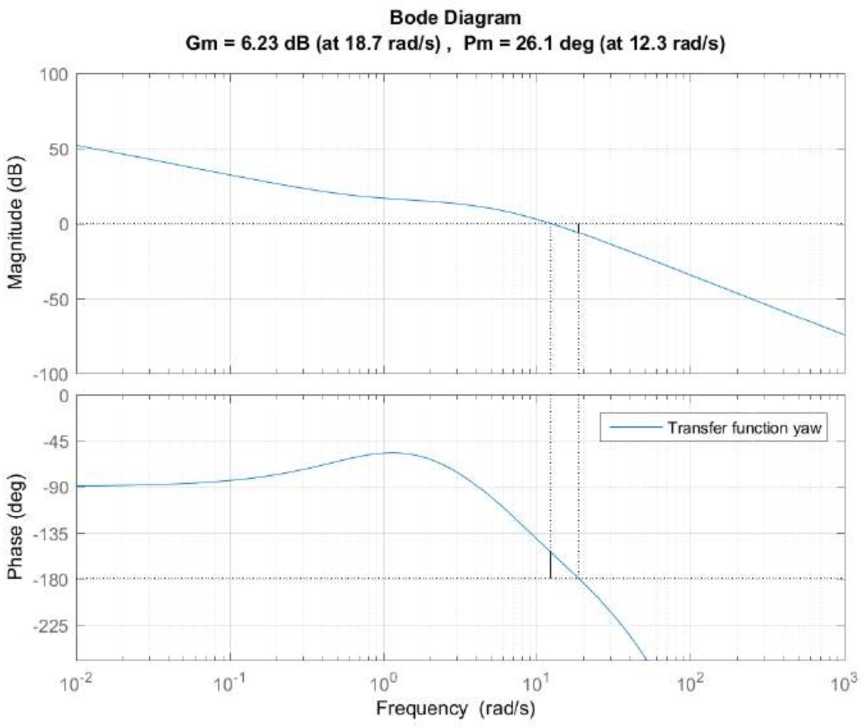


Figure 8.14: Bodeplot for open loop (yaw) with PD controller

Is has been decided, to design roll with the same K_P and K_D as pitch, and adjust it as part of a iterative process, if needed.

The PD controller of Roll is as follows

$BW = 6,87$ [rad/s]
$K_P = 29,174$
$K_D = 2$
$t_p = 0,0078$

With the PD controller implemented the following bode plot of open-loop is shown on figure 8.15.

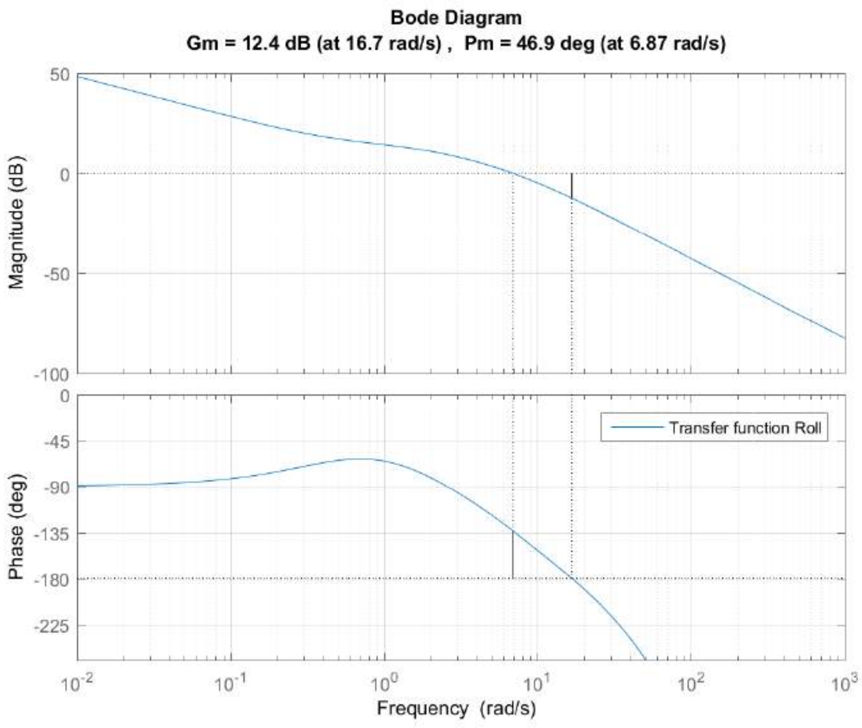


Figure 8.15: Bodeplot for open loop (roll) with PD controller.

When realising the PD controllers, a simulation of the step response needs to be performed and secondly implemented on the FPGA.

8.3 Simulation of controller

The resulting Simulink block diagram for pitch with a bandwidth of 5,85 rad/s is shown on figure 8.16.

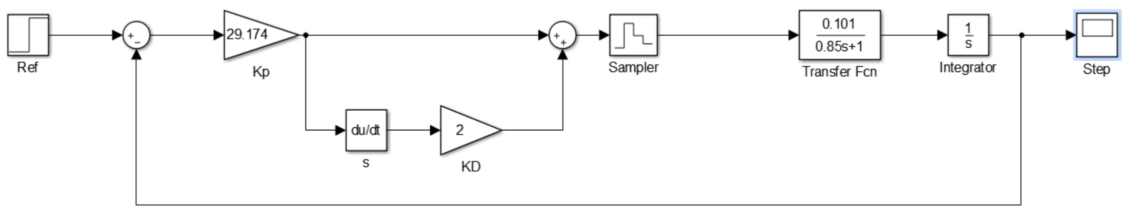


Figure 8.16: Simulink pitch loop.

A step will be carried out at 0 s with a reference of 1 m. With the sample time of $t_s = 0,031$. The resulting step response of this step of 1 m is shown on figure 8.17.

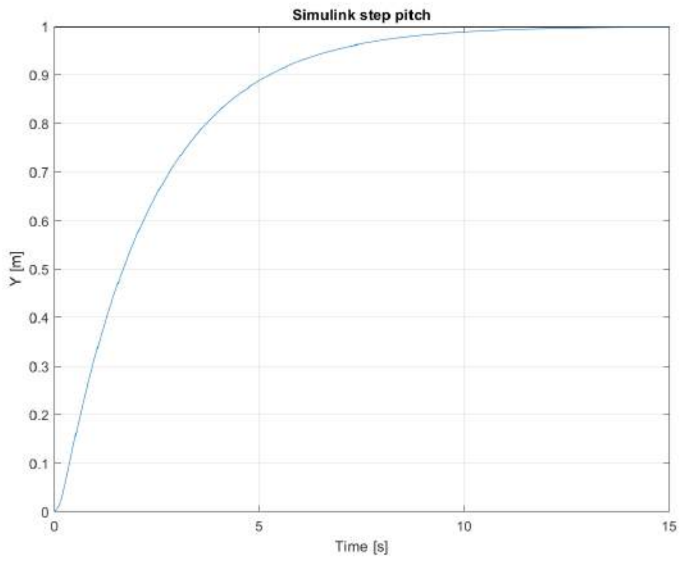


Figure 8.17: Step response of pitch simulation.

From the step response of figure 8.17 the following data can be drawn and seen in table 8.1.

Step response pitch				
Risetime [s]	**Settlingtime [s]**	**SettlingMin [m]**	**SettlingMax [m]**	**Overshoot [%]**
4.86	8.70	0.89	0.99	0.00

Table 8.1: Step data for pitch

The resulting Simulink block diagram for yaw is shown on figure 8.18.

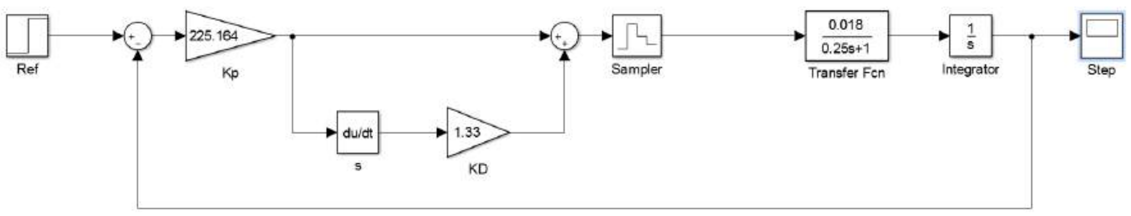


Figure 8.18: Simulink yaw loop.

A step will be carried out at 0 s with a reference of 5 rad. The resulting step response is shown on figure 8.17.

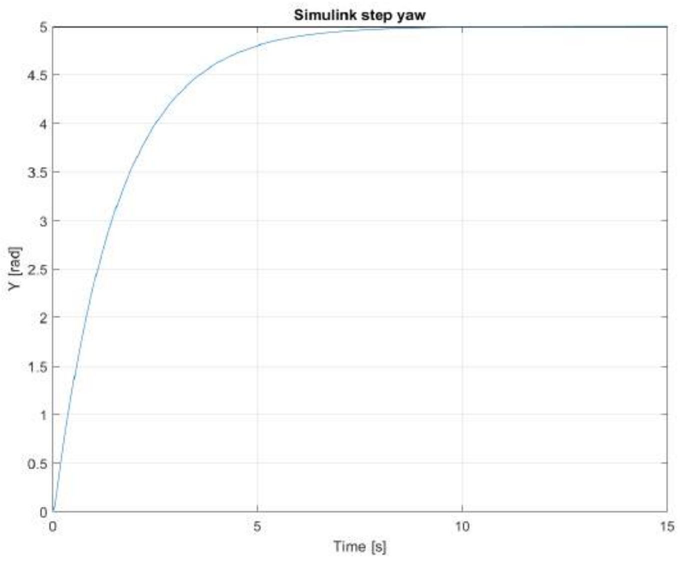


Figure 8.19: Step response of yaw simulation.

From the step response of figure 8.19 the following data can be drawn and seen in table 8.2.

Step response yaw				
Risetime [s]	**Settlingtime [s]**	**SettlingMin [rad]**	**SettlingMax [rad]**	**Overshoot [%]**
3.40	6.07	4.50	4.99	0.00

Table 8.2: Step data for yaw

The resulting Simulink block diagram for roll is shown on figure 8.20.

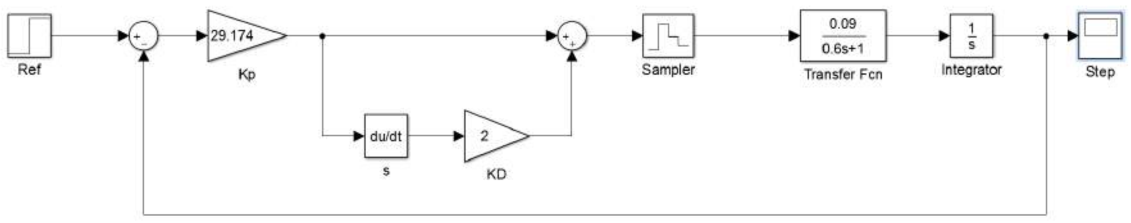


Figure 8.20: Simulink roll loop.

A step will be carried out at 0 s with a reference of 1 m. The resulting step response of a step of 1 m is shown on figure 8.21.

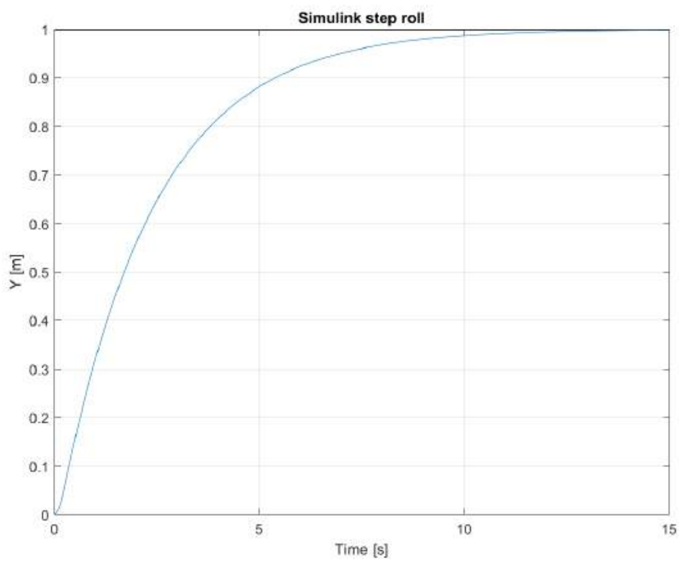


Figure 8.21: Step response of roll simulation.

From the step response of figure 8.21 the following data can be drawn and seen in table 8.3.

Step response roll				
Risetime [s]	**Settlingtime [s]**	**SettlingMin [m]**	**SettlingMax [m]**	**Overshoot [%]**
5.00	8.92	0.89	0.99	0.00

Table 8.3: Step data for roll

From the Simulink simulations it is shown that neither pitch, yaw or roll has an overshoot over 10 % and therefore these controllers will be implemented into the system.

8.4 Implementation of controller

This section will concern the implementation of the designed controllers onto the FPGA, and will only take into account the implementation of pitch, because the control module which will be implemented can be duplicated with other K_p, K_d, reference and sensor values. The PD controller for pitch is stated in equation 8.15.

$$D_y(s) = K_P \cdot (1 + K_D \cdot \frac{s}{1 + t_p \cdot s}) \qquad [\cdot] \quad (8.15)$$

Where
$K_P = 29,174$ $[\cdot]$
$K_D = 2$ $[\cdot]$
$t_p = 0,0078$ [s]

Because the FPGA cannot work in the s-domain a conversion from s-domain to time-domain is needed, this is done with Euler´s backward method on the derivative term shown in equation 8.16.

$$\frac{dU}{ds} = \frac{V(s)}{R(s)} = \frac{s}{1 + t_p \cdot s} \qquad [\cdot] \quad (8.16)$$

Which can also be stated as.

$$V(s) \cdot (1 + t_p \cdot s) = R(s) \cdot s \qquad [\cdot] \quad (8.17)$$

An approximation of s is.

$$R(t) \cdot s = \frac{R(t) - R(t-1)}{t_s} \qquad [\cdot] \quad (8.18)$$

$$V(t) \cdot s = \frac{V(t) - V(t-1)}{t_s} \qquad [\cdot] \quad (8.19)$$

Where
$t_s = 0,031$

This results in the following equation.

$$V(t) + t_p \cdot \frac{V(t) - V(t-1)}{t_s} = \frac{R(t) - R(t-1)}{t_s} \qquad [\cdot] \quad (8.20)$$

V(t) is then isolated.

$$V(t) \cdot [1 + \frac{t_p}{t_s}] = \frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1) \qquad [\cdot] \quad (8.21)$$

$$V(t) = \frac{\frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1)}{1 + \frac{t_p}{t_s}} \qquad [\cdot] \quad (8.22)$$

The equation is then simplified.

$$V(t) = \frac{\frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1)}{\frac{t_s}{t_s} + \frac{t_p}{t_s}} \qquad [\cdot] \quad (8.23)$$

When $t_p \ll t_s$ the divider of equation 8.23 can be removed an the final equation is as stated in equation 8.24.

$$V(t) = \frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1) \qquad [\cdot] \quad (8.24)$$

With s domain transformed to time-domain the calculation of the controller will be performed in the following order:

1. $Er = Sensor_{val} - Ref_{val}$ [m]
2. $Sum1 = R(t) = Er \cdot K_P$ [m]

3. $V(t) = \frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1)$ $[\frac{\text{m}}{\text{s}}]$

4. $Sum2 = V(t) \cdot K_D$ $[\frac{\text{m}}{\text{s}}]$

5. $U = Sum1 + Sum2$ $[\%]$

This is diffucult to implement on the FPGA for the following reasons:

- The FPGA is not able to directly do arithmetics with floating points because it does not have a floating point unit (FPU).
- The FPGA is not able to do multiple calculations in one line of code.
- The FPGA needs implemented cores to multiply (refer to appendix A.5) and divide (refer to appendix A.5) to do several calculations.

The advantages of the FPGA when doing calculations are:

- The FPGA can do a calculations at high speeds.
  - Addition: One clock cycle.
  - Subtraction: One clock cycle.
  - Multiplication (IP core): Three clock cycles.
  - Division (IP core): Three to seven clock cycles.
  - Bitshifting: One clock cycle
- The FPGA can do several arithmetics at once.

To exploit the advantages of the FPGA, $Er(t) \cdot D(t) = U(t)$ is calculated with the use of different fixed points. The measuring systems output is given in mm so a conversion to m is needed, but because the fixed point is 2^0 a resolution of 1 m will be given. Therefore the input is given a fixed point of 2^8 which gives a resolution of 4 mm which is deemed acceptable.

The transfer function D(t) is given a fixed point of 2^{10} to adequately represent the constants used in the following equation:

$$V(t) = \frac{t_p}{t_s} \cdot V(t-1) + \frac{1}{t_s} \cdot R(t) - \frac{1}{t_s} \cdot R(t-1) \quad [\frac{\text{m}}{\text{s}}] \quad (8.25)$$

Where:
$t_p = 0.0078$
$t_s = 0,031 \cdot 2^{10}$
$\frac{t_p}{t_s} = \frac{1}{4063} \approx \frac{1}{4096}$
$\frac{1}{t_s} = \frac{1}{31{,}7} \approx \frac{1}{32}$

To secure that the fixed point change does not affect the calculation, a comparison of the filters has been made in appendix A.4. If the approximations of equation 8.25 is perfomed, it is possible to exclude the use the divider logic core but use simple bitshifting (<< or >>) without larger deviations. The implemented calculation order will be as follows:

1. $Er = Sensor_{val} - Ref_{val}$
1. $Calc1 = V(t)_{old} >> 12$
1. $Calc3 = R(t)_{old} >> 5$

2. $Er = Er >> 2$

3. $R(t) = Er \cdot K_P$

4. $Calc2 = Rt >> 5$
4. $Sum1 = R(t)$

5. $V(t) = Calc1 + Calc2 - Calc3$

6. $Sum2 = V(t) \cdot K_D$

7. $U = Sum1 + Sum2$

With this calculation order it is possible to design a VHDL module.

8.5 VHDL design of PD controller

The calculation order as defined in section 8.4 will be implemented on the FPGA in VHDL which this section will describe. The only task of the control loop is to calculate the output based on the reference value and the sensor value, the flow of the module will be as shown on figure 8.22.

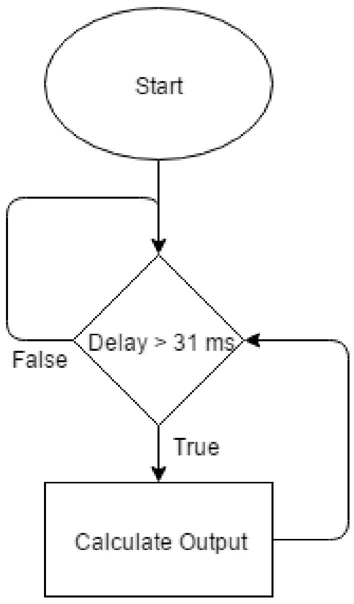


Figure 8.22: Flowchart of control module.

The delay is inserted as the sample time t_s which gives a calculation every time a new measurement is perfomened. The port and signals of the module is as follows.

- Ports
  - clk: logic input which is switched high and low according to the FPGA's system clock.
  - ref_val: integer (0-4095) input used to hardcode the reference value.
  - percent_out: integer (-128 - 127) output which represents the output of the module.
  - a1: 32 bit vector output used by the multiplier core.
  - b1: 32 bit vector output used by the multiplier core.
  - p1: 64 bit vector input used by the multiplier core.
  - k_p: 32 bit signed input which represents the K_P value.
  - k_d: 32 bit signed input which represents the K_D value.
- Signals
  - loop_counter: Integer (0 - 360000) used to time the delay.
  - temp_ref: 32 bit signed which holds the reference value.
  - temp_sensor: 32 bit signed which holds the measured sensor value.

- temp_sens_ref: 32 bit signed which holds the converted sensor distance in meters.
- r_t: 32 bit signed which represents $R(t)$.
- r_t_old: 32 bit signed which represent $R(t-1)$.
- v_t: 32 bit signed which represents $V(t)$.
- v_t_old: 32 bit signed which represents $V(t-1)$
- sum1: 32 bit signed which represents sum1.
- sum2: 32 bit signed which represents sum2.
- totalsum: 32 bit signed which represents the final sum.
- calc1: 32 bit signed which represents calc1.
- calc2: 32 bit signed which represents calc2.
- calc3: 32 bit signed which represents calc3.
- calcselect: Integer (0 - 127) used to select the specific calculation which should be performed.
- temp_percent32: 32 bit signed which holds the final value in 32 bit.
- temp_percent8 8 bit signed which holds the final value in 8 bits.

The VHDL design can be seen in code section 8.1 which follows the calculation order of section 8.4.

```
if (loop_counter < control_loop_delay) then
                loop_counter <= loop_counter + 1;

elsif (loop_counter >= control_loop_delay) then

                calcselect <= calcselect + 1;

                if (calcselect = 0) then
                        temp_sens_ref <= (temp_sensor - temp_ref);
                        calc3 <= SHIFT_RIGHT(r_t_old,5);
                        calc1 <= SHIFT_RIGHT(v_t_old,12);
                elsif (calcselect = 1) then
                        temp_sens_ref <= SHIFT_RIGHT(temp_sens_ref,2);
                elsif (calcselect = 2) then
                        a1 <= std_logic_vector(temp_sens_ref);
                        b1     <= std_logic_vector(k_p);
                elsif (calcselect = 5) then
                        r_t(30 downto 0) <= signed(p1(30 downto 0));
                        r_t(31 downto 31) <= signed(p1(63 downto 63));
                elsif (calcselect = 6) then
                        calc2 <= SHIFT_RIGHT(r_t,5);
                        sum1 <= r_t;
                elsif (calcselect = 7) then
                        v_t <= (calc1 + calc2 - calc3);
                elsif (calcselect = 8) then
                        a1 <= std_logic_vector(v_t);
                        b1 <= std_logic_vector(k_d);
                elsif (calcselect = 11) then
                        sum2(30 downto 0) <= signed(p1(30 downto 0));
                        sum2(31 downto 31) <= signed(p1(63 downto 63));
                elsif (calcselect = 12) then
                        totalsum <= (sum1 + sum2);
                        r_t_old <= r_t;
                        v_t_old <= v_t;
                elsif (calcselect = 13) then
                        temp_percent32 <= SHIFT_RIGHT(totalsum,18);
                elsif (calcselect = 14) then
                        temp_percent8(6 downto 0) <= temp_percent32( 6 downto
                            0);
```

```
                    temp_percent8(7 downto 7) <= temp_percent32( 31 downto
                        31);
                elsif (calcselect = 15) then
                    percent_out <= to_integer(temp_percent8);
                    calcselect <= 0;
                    loop_counter <= 0;
                end if;
            end if;
```

Code 8.1: Control module in VHDL.

With the implementation of the control module it can be used by both pitch yaw and roll with different input values. This leads to testing of the controllers.

8.6 Test of PD controllers

This section will go through the testing of the PD controllers.

The full measurement report for the implemented PD controller for pitch, roll and yaw can be found in appendix A.6. The testing gave the following results.

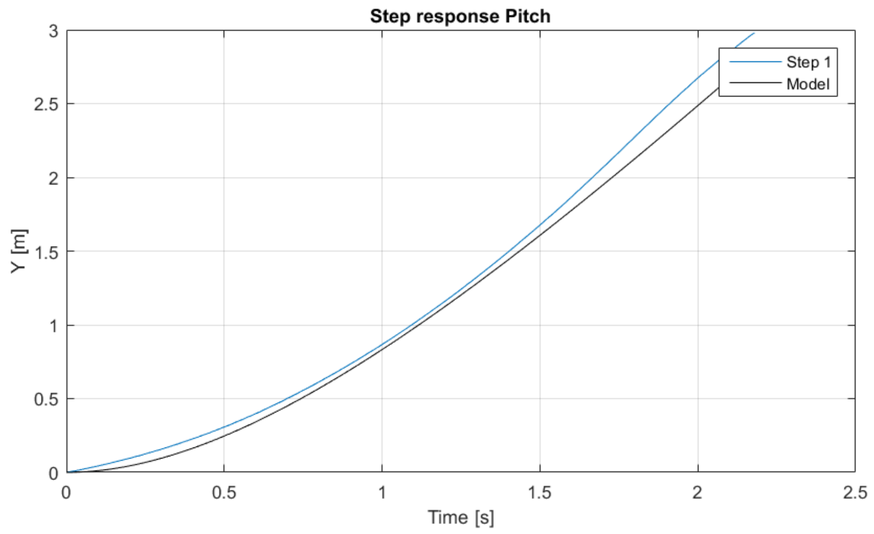


Figure 8.23: Graph of the change in distance over time for the implemented pitch controller (5,85 rad/s) compared with the model. The measured data has correction to the start point, length and errors.

On figure 8.23 a step for pitch was conducted with the use of the PD controller. The most optimal stability and speed, was found with a bandwidth of 5,85 rad/s. It is shown that the step response did not deviate significantly from the model deducted from section 8.1.

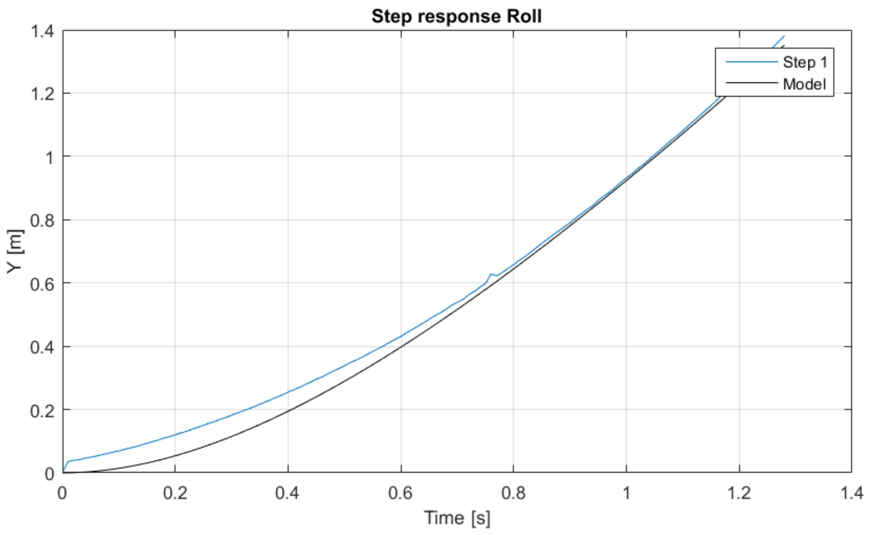


Figure 8.24: Graph of the change in distance over time for the implemented roll controller (6,87 rad/s) compared with the model. The measured data has correction to the start point, length and errors.

On figure 8.24 a step for roll was conducted with the use of the PD controller. The most optimal stability and speed, was found with a bandwidth of 6,87 rad/s. It is shown that the step response did not deviate significantly from the model deducted from section 8.1.

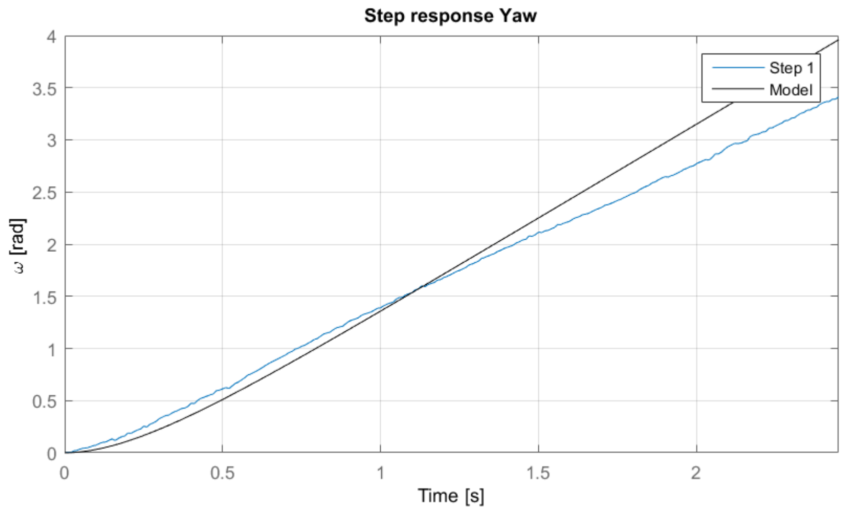


Figure 8.25: Graph of the change in distance over time for the implemented yaw controller (12,3 rad/s) compared with the model. The measured data has correction to the start point, length and errors.

On figure 8.25 a step for yaw was conducted with the use of the PD controller. The most optimal stability and speed, was found with a bandwidth of 12,3 rad/s. It is shown that the step response did not deviate significantly from the model deducted from section 8.1.

Final Product 9

9.1 Final product

In this section the final product will be reviewed, and a accepttest will be conducted on the product. The final system consists of two modules.

- Transmitter Module
- Quadcopter Module

The final product is a quadcopter that is able to measure a distance between 1 m to 4 m and keep a predefined distance to the transmitter Module. It uses a combination of ultrasound and radio waves to measure the distance from the transmitter module to the quadcopter. The quadcopter contains the receiver board, which amplifies the ultrasound signals before sending them through a comperator. After the comparator a FPGA is placed which generates PPM signal, runs the control system and the counters for the distance measurement system. A receiver is placed on the quadcopter to receive the PWM signals from the remote control, these signals are sent into the FPGA and modulated to a PPM signal which is sent into the Pixhawk.

The transmitter module, contains the following and can be seen on figures 9.1 and 9.2:

- 2 x 9 V battery.
- 1 x wooden frame 100 cm x 10 cm x 1,5 cm
- 1 x Ardiono Leonardo.
- 1 x 25kHz ultrasound Transmitter + board.
- 1 x 40kHz ultrasound Transmitter + board.
- 1 x QAM TX1 module.

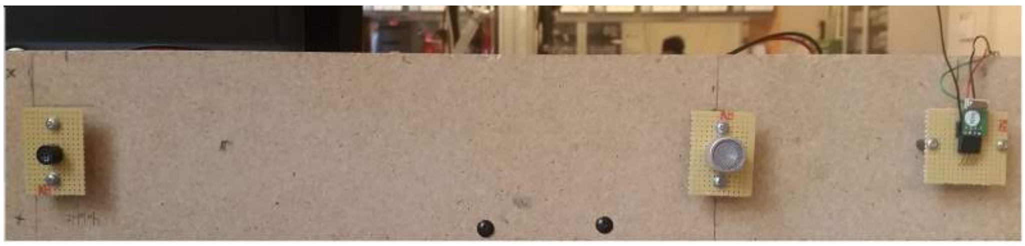


Figure 9.1: Front of the transmitter board.

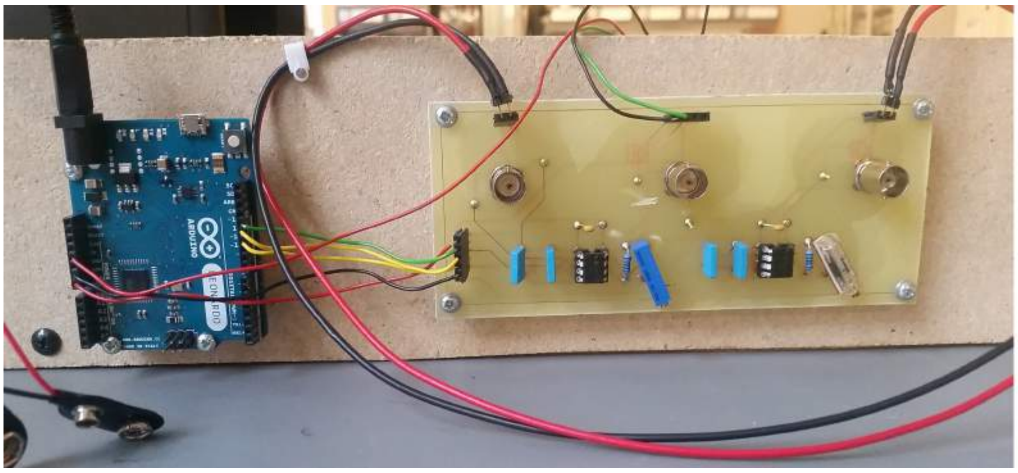


Figure 9.2: Backside of the transmitterboard.

The wooden board is 100 cm x 10 cm x 1,5 cm. On the frontside of the board the transmitters are placed. On the backside the Arduino Leonardo, transmitter board and the power supply is placed. Upon the Arduino is uploaded the code (CD/Code/send_sync/SyncSignalLowSensor/SyncSignalLowSensor). This sketch handles sending the ping and sync signals used by the sensor module.

The quadcopter module can be seen on figure 9.3 and contains the following parts:

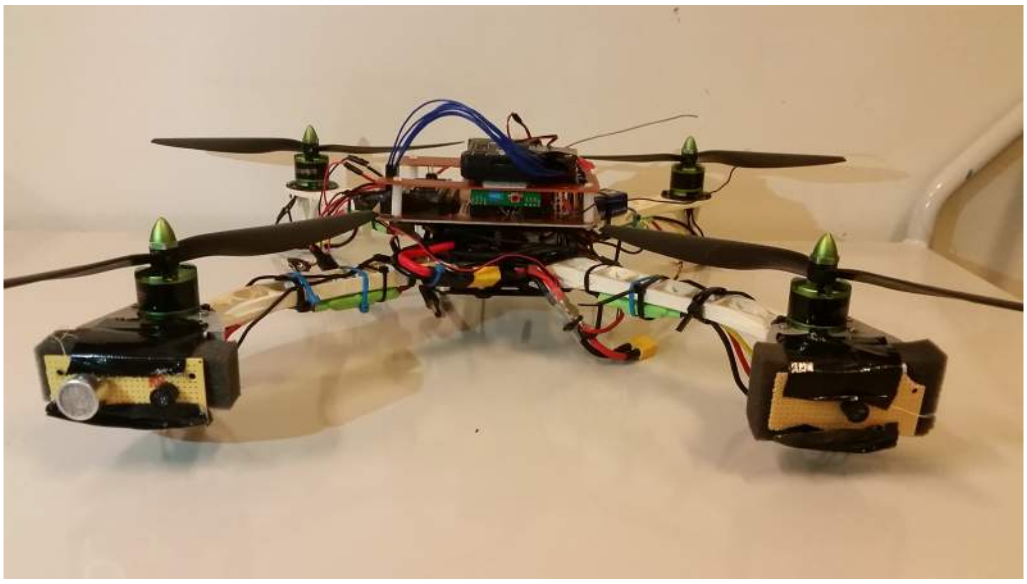


Figure 9.3: Quadcopter.

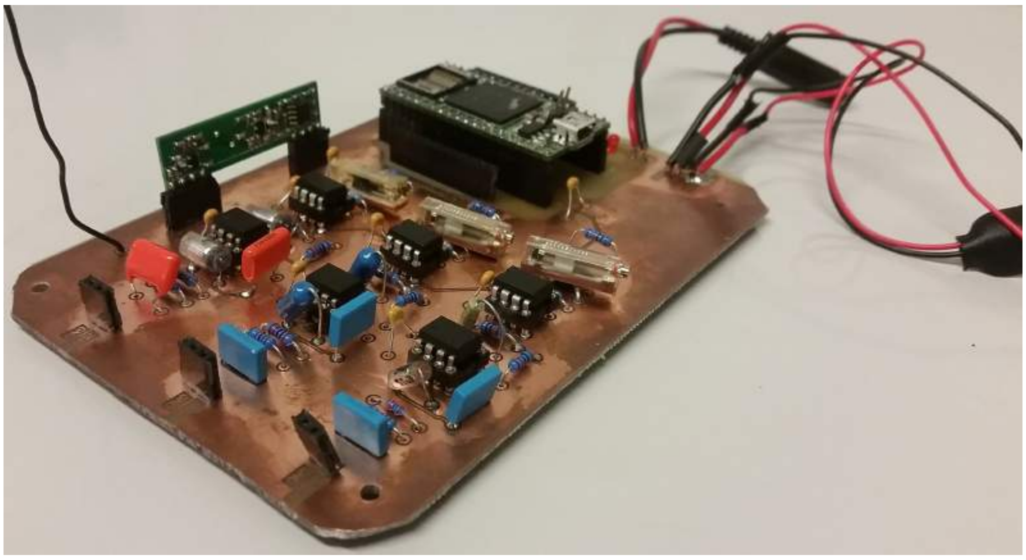


Figure 9.4: Receiver board.

- 4 x Multistar 30A Esc
- 4 x Multistar 2213-980Kv
- 2 x 9V battery.
- 1 x 11,2V 2200mAh Lipo Battery.
- 1 x Quadcopter Frame.
- 1 x Pixhawk PX4.
- 1 x Orange Rx-r620 reciever.
- 1 x Receiver board.
- 1 x XuLA2-LX9.
- 1 x 25kHz Ultrasound reciever.
- 2 x 40kHz Ultrasound recuever.
- 1 x QAM RX2 module.

Upon the FPGA is uploaded the Control system, sensor system and PPM-encoder(CD/code/final). The PPM encoder Transforms the Received PWM signals from the Rx-r620 into a PPM-signal, which is used to control the Pixhawk. The Sensor system decodes the radio wave transmitted by the transmitter board and listens for a ping signal, when the ping signal is detected, the board will start a timer waiting for the ultrasound wave to arrive or timeout after a certain interval. The Control module uses the distance generated by the sensor module, and generates an offset on the PPM signal based on a desired position.

On figure 9.5 the final block diagram for the system is shown.

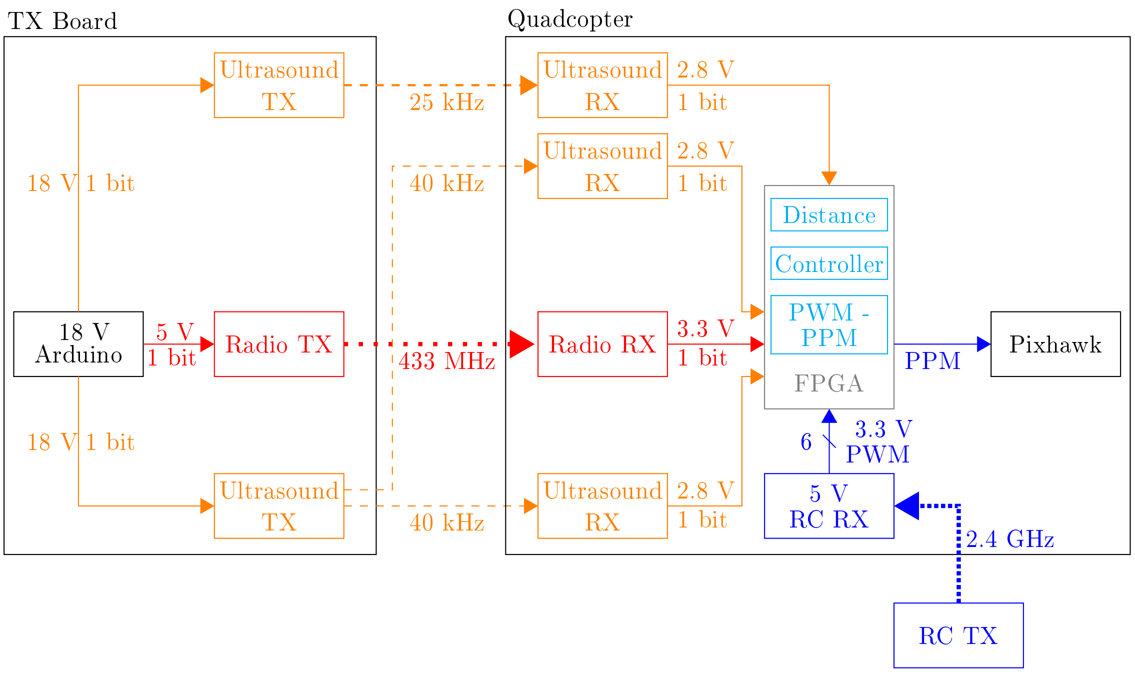


Figure 9.5: Block diagram of system including interfaces.

On the final block diagram the final values for the different modules can be seen. On the block diagram 9.6 below it can be seen the final modules inside the FPGA and the communication between them.

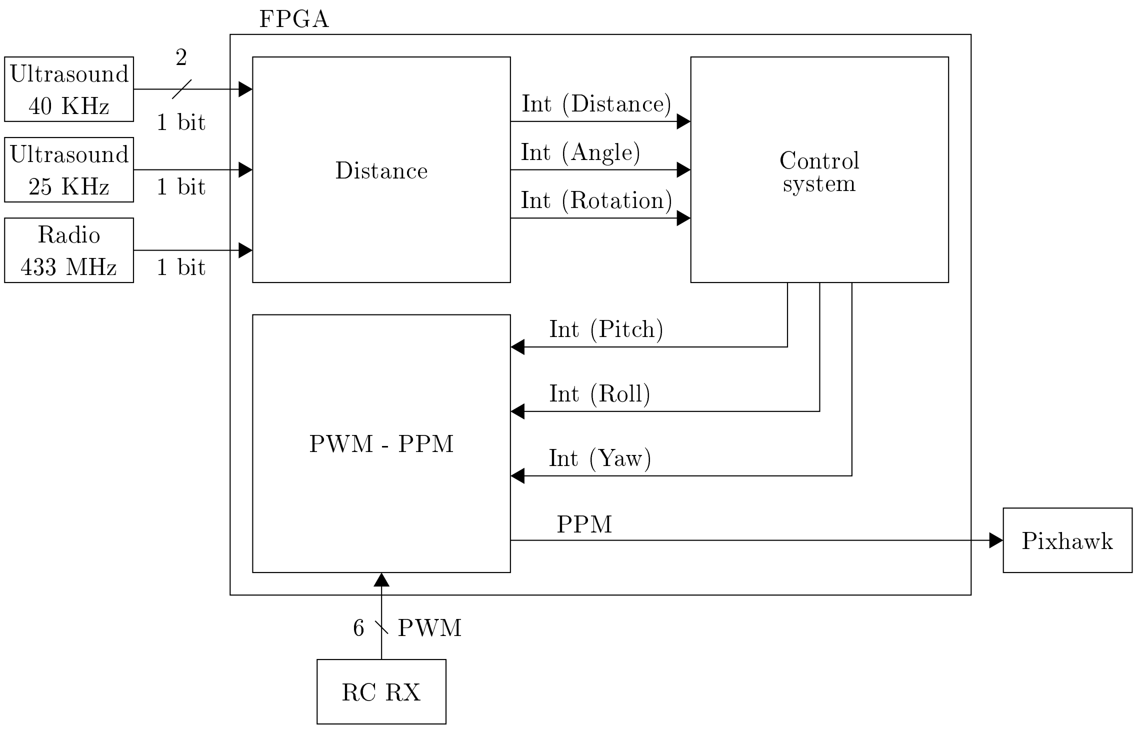


Figure 9.6: Block diagram of system in the FPGA including interfaces.

The product is now ready to the accepttest.

9.2 Accepttest

In this section the requirements stated in section 4.2 will be tested according to the description in section 4.4. In table 9.1 an overview of the tested and untested requirements is given.

Requirement number	Tested (Y/N)	Fulfilled(Y/N)	Section reference
1	Y	Y(*)	9.2
2	N	-	9.2, 11
3	N	-	9.2, 11
4	Y	Y	A.8
5	Y	Y	A.8
6	Y	Y (*)	A.12, A.13
7	Y	Y (*)	A.11

Table 9.1: Overview of tested requirments, where (*) means partially

As table 9.1 show every all but two modules have been tested and concluded in their respective section. The remaining tests concerns the full functionality of the system with every module implemented. These tests will be done and concluded on in the following.

The quadcopter shall maintain a predefined distance between 1 m - 4 m from the helicopter with a tolerance of maximum $\pm$ 10 % of the distance.

The test is done in the MTLab using the Vicon system. The transmitter board is placed in an elevated position approximately half a meter above ground, with three reflection markers on it for visualization purpose. A reflection marker is placed on the ground, 1,5 m from the transmitter board. The quadcopter is placed on the ground so the sensor in line with the reflection marker (see figure 9.7 and figure 9.8).

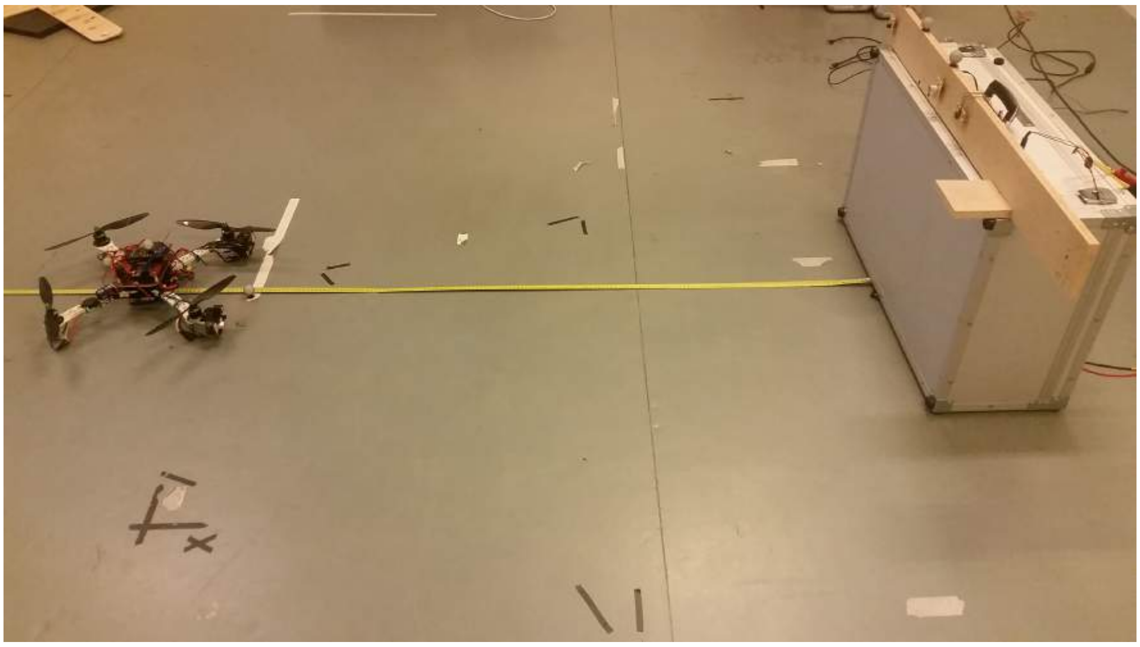


Figure 9.7: Figure of quadcopter placed 1,5 m from transmitter board.

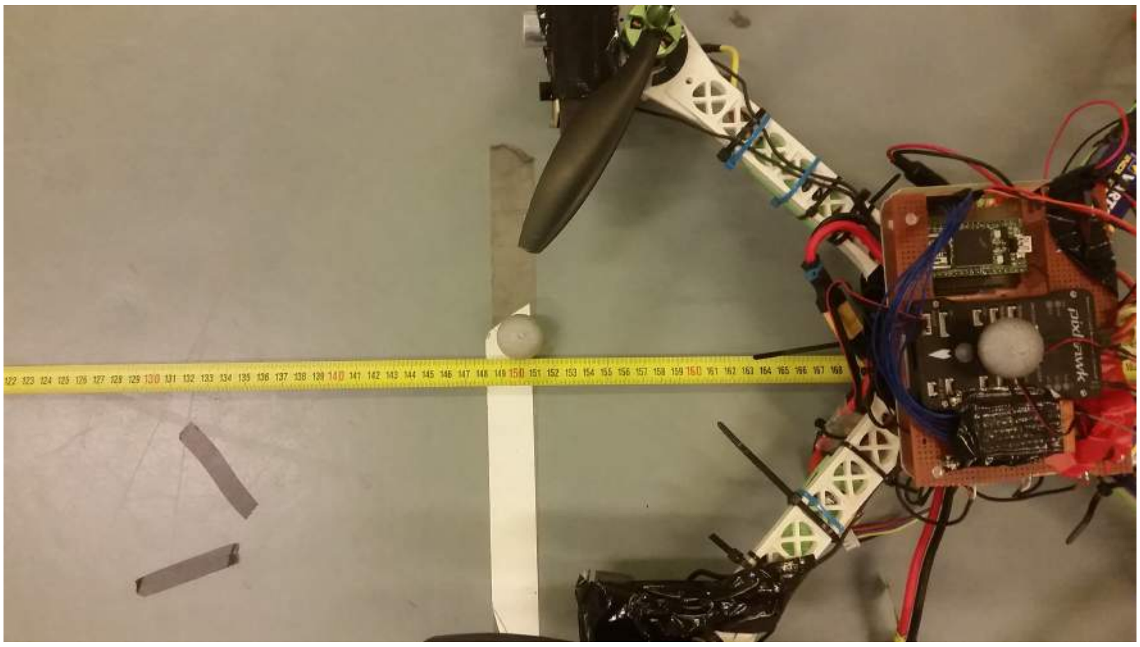


Figure 9.8: Figure of quadcopter placed 1,5 m from transmitter board (close op).

It is then possible to adjust the quadcopter object axes in Vicon, to line op with the reflection marker on the ground. Thereby the distance between the quadcopter and the transmitter board is known to be 1,5 m when the axes is located over the reflection marker on the ground.

On figure 9.9 and figure 9.10, is the transmitter board placed in the top of the pictures, the quadcopter is the orange triangle with the white box in front and the white dot is the 1,5 m reflection marker. There is approximately seven tern between the transmitter board and the 1,5 m marker, which means the must stay within to on and half tern to pass the test. A couple of test was performed in which the quadcopter maintained the distance to the helicopter within the tolerance.

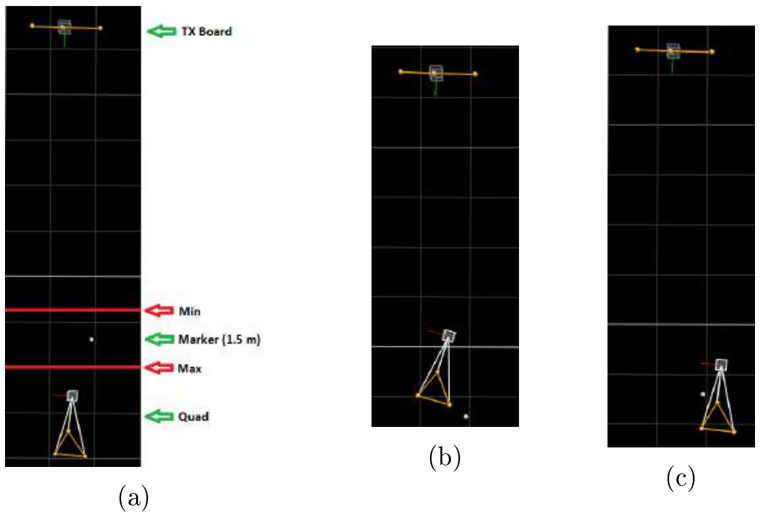


Figure 9.9: (a) Quadcopter moving towards the board. (b) Quadcopter passes the 1,5 m. (c) Quadcopter moving backward.

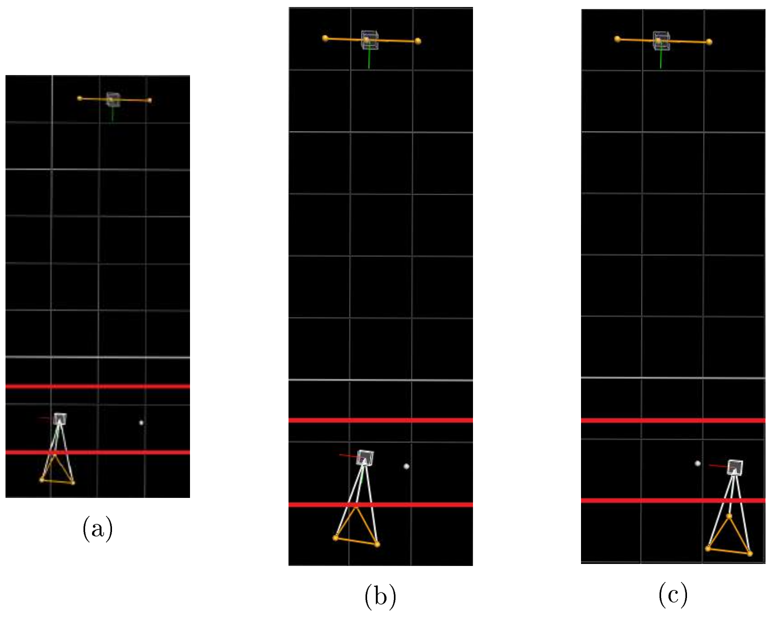


Figure 9.10: (a) Quadcopter stabilized at the 1,5 m marker. (b) Quadcopter remain at 1,5 m marker. (c) Quadcopter remain at the 1,5 m marker.

This however does not take into account the possible overshoot as the quadcopter approaching the helicopter. Therefore a small test was made using Vicon to evaluate the overshoot, which can be seen on figure 9.11. Two flight approach was made one short starting at approximately half a meter from the reference distance of 1,5 m and one starting from one meter away.

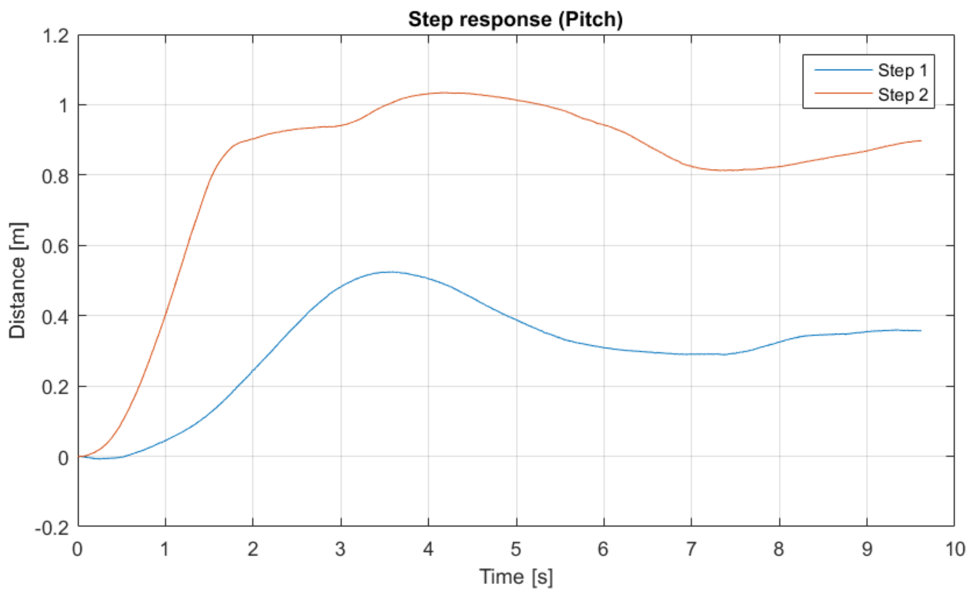


Figure 9.11: Graph showing the step response with the pitch controller activated over a short (Step 1) and long (Step 2) flight approach.

Step 1 on figure 9.11 seems to level off at 0,35 m peaking 0,5 giving it a larger overshoot than the 10 percent it was model after. As the model is simple this was expected, to meet the requirement the parameter of the model has to be expanded or possible increase the phase margin by adjusting K_D. Step 2 seems to be leveling off at approximate 0,9 m unfortunately it has decelerate, possible due to hitting the ground as the test was done just above ground, making it impossible to read the true peak value, a new test has to be made to evaluate further on this.

During test it was found that the quadcopter will maintain a 1,5 m distance within the tolerance to the helicopter, a short recording of the test can be found on the CD (/CD/Video/Accepttest_Pitch_Vicon). Due to lack of time this is the only distance that was tested. The test is therefore only partially fulfilled, as the control system for pitch worked, although the overshoot is greater than 10 percent, but as the complete range have not been tested in a dynamic environment, it is not fully fulfilled.

The quadcopter shall maintain an angle of 90 degrees to the helicopter with a tolerance of ± 20 °.

As the desired output from the sensors will not function with the roll controller. It is not possible to perform the accepttest, a description of the problem can be found in chapter 11.

The ultrasound receivers on the quadcopter shall maintain a orientation toward the helicopter with a tolerance of ± 10 °.

As the desired output from the sensors will not function with the yaw controller. It is not possible to perform the accepttest, a description of the problem can be found in chapter 11.

Conclusion 10

A system has been designed with the goal of expanding the sweep width of a helicopter at SAR missions, based on the problem statement from section 1.3:

How can an electronic system be designed to reduce the search time during rescue missions at open sea by increasing the width of the search sweeps?

To achieve this, the system has been designed, to measure and regulate a distance between a quadcopter and a helicopter. The helicopter contains the transmitter modules, which transmits radio and sound waves, which is used by the quadcopter to find the relative distance between them. A Pixhawk handles flight control and a control system calculates an offset which is modulated into the PPM signal to steer a quadcopter in a direction or hold a position. By holding a quadcopter in a desired posision, relative to a helicopter, the system should be able to increase the sweep width of the helicopter.

It can be concluded that the system is not fully functional but a distance measuring system functions with a accuracy better than required, a low latency PPM modulator functions flawlessly and three controllers functions indenpendtly of each other, where only the direction pitch is implemented to function with the distance measuring system due to miscalculations in the design stage.

Optimization 11

This chapter will describe the optimization points of the system. The focus will primarily be on the control system but will include general optimization points on the different modules and in general.

Generaly several improvments can be made. The size and weight of the quadcopter can be reduced by implementing the hardware using SMD components versus through-hole components, which is implemented now. Implementing the amplifiers as close to the sensors as possible will negate the impact of noise because the signals are not as susceptible to it. Lastly a general optimization of the VHDL design could be done by trimming down the width of bits to increase effectiveness of the utilization on the FPGA.

One of the problems with the system is how it handles the sensor data. At the moment no treatment is done to the measured data before sending it into the control system. This results in unexpected behavior at times. If the system looses line of sight to one of it's sensors, it either times out or shows a wrong measurement. If the wrong measurement is far from the ref value, the control system will start working against that offset. This can create a situation where the system accelerates out of control because the old data does not update and the control system thus keeps applying the wrong offset. A solution could be to assume the distance is the ref value if the system times out.

As seen in the accepttest two of the controllers (yaw, roll) was not implemented because of wrong inputs therefore a big improvement could be made by making functions which could calculate the correct inputs for both yaw and roll. Pitch was implemented in the final accepttest but pitch is not implemented correct because if the ultrasound sensor is not directly opposite the transmitter an angle will occur (see figure 11.1 b) making the input wrong while it is correct if directly opposite each other as seen on figure 11.1 a.

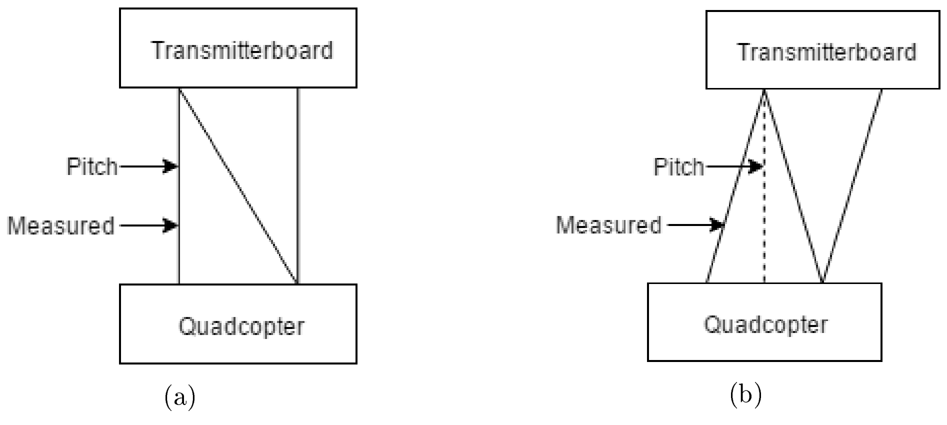


Figure 11.1: (a)Transmitter and receiver directly opposite each other. (b)Transmitter and receiver not directly opposite each other.

The same problem exist as mentioned ealier for yaw where the correct input is the angle of the qaudcopter relative to the transmitter board (see figure 11.2 b) instead of the difference between the two measured sensor distances as seen on figure 11.2 a.

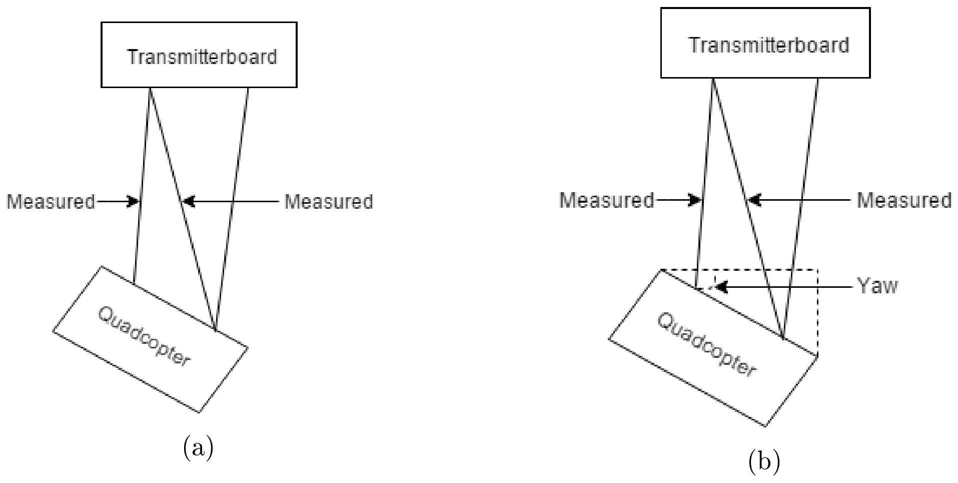


Figure 11.2: (a)Transmitter and receiver directly opposite each other. (b)Transmitter and receiver not directly opposite each other.

Lastly the problem also exist for Roll. Here the input should be the difference between the reference point where the sensors are directly opposite each other as seen on figure 11.1 a and point away from the reference point as seen on figure 11.3.

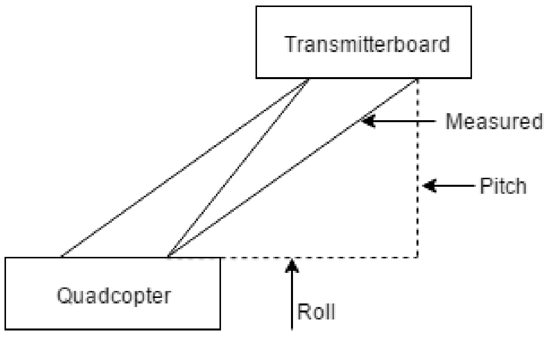


Figure 11.3: Transmitter and receiver not directly opposite each other

To calculate the corret input with the measured distances, trigonometry should be used, creating functions which can be implemented as modules on the FPGA. This concludes on the optimization chapter.

Perspective 12

This chapter will perspectivate on the final product and examine what a final product would have looked like if time and rescources would not be a limiting factor.

To complete the final product entierly so the drone would be able to fly autonomosly the unimplemented direction (elevate) must be controlled to. At the moment, the control system have no way of measuring the height which it is positioned in. This means that in order to control the height, additional sensors must be added like a barometer.

The drone used in the project was selected because it were available for the project so it was not expected to be able to follow a rescue helicopter flying at 240 km/h in severe weather. So a special build quadcopter or deltawing would improve airworthiness which would make flying at high speeds and rough weather a possibilty. This would of course lead to further control implementations like a PID controller. Even if the optimal drone were available, it would still be problematic to test it, as the only available place to test it would be MTLab. A more realistic test would include wind, turbulence and humidity. A board was used as a replacement for a helicopter, to test the control system. Before a finish product could be accepted, the quadcopter would have to be tested with a real helicopter in a real enviorment.

The choice of using an FPGA instead of a microprocessor have proved time consuming, especially when implementing the control system. This have resulted in the system not being fully implemented.

The ultrasound sensors would not be a solution if a full scale product were to be made because of their limitations in distance (0,5 m - 4 m). A sensor type is needed, which can measure much longer distance (0,5 m - 300 m) at higher speeds. This could for example be the Pulson 410 (/CD/Appendix/Pulson mail correspondence), a similar system which uses radio to measure distance or GPS.

Another improvement is to expand on the communication protocol between the transmitter board and the system. The communication on the final product is one way communication so the reference values which the control system uses to hold the position, is hardcoded into the FPGA. By expanding the communication, these values could be changed by an operator, to fit the situation during flight, instead of during upload. Further expansion could be two way communication which could include:

- Positioning.
- Video feed.

This could give the operator valuable informations such as where the quadcopter is flying on a map but also visualy besides having a camera pointing down at the water. To fly outdoors manually the quadcopter needs a collision avoiding system to make sure it does not collide with anything eventhough it flies at sea.

Discussion 13

The system designed in this project is designed to aid at SAR missions. Some other systems have been made with this purpose in mind. Most of the systems seen today is designed to be operated by a drone operator and searched manually which limits the system to one search and rescue drone pr. operator. The system designed in this project is designed to autonomously follow a helicopter which does not require an operator. This means that the operator can focus on the detection of the distressed, in SAR missions.

The advantage of using autonomic drones is there are fewer limitations in differ to a operated drone. A helicopter would be able to have several drones attached so a even bigger sweep widths is achieved. The big question is though if an operator would be able to trust an autonomous drone searching the correct area and not missing any points which could mean the loss of lives. Secondly the operator would need to keep track of several screens at the same time, so the operator is now the limiting factor in search and rescue missions, so a search algorithm might be implemented, which shoud aid the operator detecting the distressed.

This raises the question if the solution implemented in this project would be the most optimal solution? Would it be more effective to only have drones search an area from a base point at shore or a ship sailing at sea, so several operators could keep track of the camera feeds instead of having drones flying besides a helicopter? Would it be more effective to focus on preventing people falling into the water? or maybe another solution entirely?

The big advantage of the solution implemented in this project though is its simplicity. The helicopter itself does not need any extra equipment besides a transponder, a receiver for video feed and a monitor. An all drone solution would be difficult to implement because precise GPS is difficult to obtain at sea so how should positioning be done without a reference point? The search and rescue routines used by the navy should also not be changed to implement the solution in this project because the drone only increases the sweep width so the quadcopter only aids in the SAR mission relative to lead it.

Bibliography

[3drobotics, 2014] 3drobotics (2014). Pixhawk autopilot quick start guide. Last visited on 2015-14-10, https://3drobotics.com/wp-content/uploads/2014/03/pixhawk-manual-rev7.pdf.

[3DRobotics, 2014] 3DRobotics (2014). Ppm encoder. Last visited on 2015-14-12, https://3drobotics.com/wp-content/uploads/2014/01/PPM-Encoder-V3-Manual.pdf.

[3drobotics, 2015] 3drobotics (2015). Pixhawk overview. Last visited on 2015-14-10, http://copter.ardupilot.com/wiki/common-pixhawk-overview/.

[Ashby, 2008] Ashby, D. (2008). Fpga architectures. Last visited on 2015-08-12, http://www.globalspec.com/reference/81250/203279/chapter-29-field-programmable-gate-arrays-fpgas.

[Autopilot,] Autopilot, P. Flight mode. Last visited on 2015-07-12, https://pixhawk.org/users/system_modes.

[Autopilot, 2014] Autopilot, P. (2014). Motor assignment. Last visited on 2015-07-12, https://pixhawk.org/_media/airframes/quadrotor_x_assignment.png?cache=.

[Autopilot, 2015] Autopilot, P. (2015). Pixhawk autopilot. Last visited on 2015-14-12, https://pixhawk.org/modules/pixhawk.

[ClipArt, 2015] ClipArt (2015). Drowning person clip art the libs are treading water. Last visited on 2015-14-12, http://www.clipartbest.com/clipart-9T4o5nMqc.

[Coloringkidsboys, 2014] Coloringkidsboys (2014). Seaking. Last visited on 2015-14-12, https://www.google.dk/search?q=rescue+helicopter&espv=2&biw=1137&bih=1305&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiWz96vnd7JAhWn83IKHUvzA9YQ_AUIBigB&dpr=0.9#tbm=isch&q=rescue+helicopter+clipart&imgrc=DwiWHdNRYFUrEM%3A.

[corporation,] corporation, X. Xula2-lx9. Last visited on 2015-09-12, http://www.xess.com/shop/product/xula2-lx9/.

[corporation, 2012] corporation, X. (2012). Xula2 manual. Last visited on 2015-09-12, http://www.xess.com/static/media/manuals/XuLA2-manual.pdf.

[Coughlan, 2012] Coughlan, B. (2012). Dsm2/dsmx remote receiver protocol. Last visited on 2015-14-10, http://www.pixhawk.com/_media/dev/dsm2_dsm_x_protocol.pdf.

[Emami-Naeini, 2015] Emami-Naeini, G. F. . J. P. . A. (2015). Feedback control of dynamic systems.

[Farnell, 2013] Farnell (2013). Datasheet 25 khz ultrasound transducer. Last visited on 2015-22-10, http://www.farnell.com/datasheets/81154.pdf.

[Forsvaret, 2014] Forsvaret (2014). Eskadrille 722 assigments. Last visited on 2015-23-09, http://forsvaret.dk/HWKAR/HW%20KAR%20Organisation/Eskadrille%20722/Pages/default.aspx.

[Forsvarskommando, 2014] Forsvarskommando, V. (2014). Hyperthermia - overlev i koldt vand. Last visited on 2015-23-09, http://forsvaret.dk/MST/NATIONALT/SOEREDNING/UNDERAFKOELING/Pages/default.aspx.

[Forsvarsministeriet, 2013] Forsvarsministeriet (2013). Skibsfartens og luftfartens redningsrÅds Årlige redegØrelse for sØ- og flyveredningstjenesten 2012. Last visited on 2015-29-09, Kilde:http://forsvaret.dk/MST/Nationalt/soeredning/%C3%85rsrapporter/Documents/Aarsrap_2012.pdf.

[GamesOnTrack,] GamesOnTrack. Gt-position components. Last visited on 2015-13-11, http://www.gamesontrack.com/pages/webside.asp?articleGuid=164202.

[Gettyimages, 2015] Gettyimages (2015). Quadcopter. Last visited on 2015-14-12, http://cache2.asset-cache.net/xt/165960170.jpg?v=1&g=fs1|0|SKP128|60|170&s=1.

[Golding, 2011] Golding, J. (2011). What is the difference between terrestrial (land based) internet and satellite internet service. Last visited on 2015-15-10, http://www.niasat.com/q-what-is-the-difference-between-terrestrial-land-based-internet-and-satellite-internet-se

[Hobby-King, 2015] Hobby-King (2015). Quadcopter. Last visited on 2015-21-09, http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=35819.

[Hobbyking, 2015] Hobbyking (2015). Esc for quadcopter. Last visited on 2015-14-12, http://www.hobbyking.com/hobbyking/store/__65157__Turnigy_Multistar_30_Amp_BLHeli_Multi_rotor_Brushless_ESC_2_6S_V2_0.html.

[HobbyKing, 2015] HobbyKing (2015). Hobbyking sk450 glass fiber quadcopter frame 450mm. Last visited on 2015-14-10, http://www.hobbyking.com/hobbyking/store/__24291__hobbyking_sk450_glass_fiber_quadcopter_frame_450mm.html.

[Hobbyking, 2015a] Hobbyking (2015a). Motor for quadcopter. Last visited on 2015-13-12, http://www.hobbyking.com/hobbyking/store/__43884__MT2213_935KV_MultiStar_Motor_and_Propeller_Combo_10x4_5_CW_CCW.html.

[Hobbyking, 2015b] Hobbyking (2015b). Orangerx t-six 2.4ghz. Last visited on 2015-07-12, http://www.hobbyking.com/hobbyking/store/catalog/54821(6).jpg.

[IOP, 2012] IOP (2012). How does gps work. Last visited on 2015-14-10, http://www.physics.org/article-questions.asp?id=55.

[KrigerenDK, 2013] KrigerenDK (2013). Eh-101 helicopter. Last visited on 2015-23-09, http://krigeren.dk/problem-helikopteren-eh-101-kan-forst-transportere-soldater-i-2014/.

[landfallnavigation, 2015] landfallnavigation (2015). Difference between rgb- and ir-camera. Last visited on 2015-23-09, http://www.landfallnavigation.com/flirfirstmate.html.

[Matti Kummu, 2015] Matti Kummu, Hans de Moel, P. J. W.-O. V. (2015). How close do we live to water? a global analysis of population distance to freshwater bodies. Last visited on 2015-22-09, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3110782/.

[Mio, 2011] Mio (2011). What is trilateration. Last visited on 2015-14-10, http://www.mio.com/technology-trilateration.htm.

[muRata, 2002] muRata (2002). Datasheet 40 khz ultrasound transducer. Last visited on 2015-22-10, http://docs-europe.electrocomponents.com/webdocs/03cf/0900766b803cf552.pdf.

[National-Instruments, 2011] National-Instruments (2011). Understanding parallel hardware: Multiprocessors, hyperthreading, dual-core, multicore and fpgas. Last visited on 2015-09-12, .

[Network-World, 2014] Network-World (2014). New broadband-friendly satellites could change where we live. Last visited on 2015-15-10, http://www.networkworld.com/article/2457722/service-providers/a-new-breed-of-broadband-satellites-could-have-you-living-on-a-desert-island.html.

[NOAA,] NOAA. Using gps/gnss reflections for soil moisture and sea state monitoring. Last visited on 2015-30-10, http://www.esrl.noaa.gov/psd/psd3/multi/remote/using_reflections.html.

[OrangeRX, 2015] OrangeRX (2015). Instruction manual orange rx. Last visited on 2015-14-12, http://www.hobbyking.com/hobbyking/store/uploads/23068243X7478X36.pdf.

[OrangeRx, 2015] OrangeRx (2015). Instruction manual orangerx r620x. Last visited on 2015-14-12, http://www.hobbyking.com/hobbyking/store/__77802__OrangeRx_R620X_6Ch_2_4GHz_DSM2_DSMX_CompFullRangeRx_w_SatDivAnt_F_Safe_CPPM.html.

[personprofil, 2015] personprofil, A. (2015). Simon jensen personprofil aau. Last visited on 2015-13-12, http://personprofil.aau.dk/110271.

[Pixabay, 2015] Pixabay (2015). Pixabay. Last visited on 2015-14-10, https://pixabay.com/en/stealth-fighter-delta-wing-42713/.

[QGroundControl.org, 2015] QGroundControl.org (2015). Qgroundcontrol. Last visited on 2015-14-12, http://qgroundcontrol.org/.

[Redningsrådet, 2013] Redningsrådet (2013). Sar report. Last visited on 2015-23-09, http://forsvaret.dk/MST/Nationalt/soeredning/SARDK/Documents/SAR%20DANMARK%20Bind%20II%20Operativ%20Manual%20APR%202015.pdf.

[resna, 2003] resna (2003). Sound localization techniques for a direction indicating vibrotactile aid. Last visited on 2015-16-09, http://www.resna.org/sites/default/files/legacy/conference/proceedings/2003/Papers/TSP/Lathan_TSP.htm.

[Texas-Instruments, 2014] Texas-Instruments (2014). Ne555 precisions timers. Last visited on 2015-19-11, http://www.ti.com/lit/ds/symlink/ne555.pdf.

[Theintentionallife, 2015] Theintentionallife (2015). Helicopter parents. Last visited on 2015-14-10, http://www.theintentionallife.com/broadcasts/helicopter-parents/.

[Tontechnik-Rechner, 2015] Tontechnik-Rechner (2015). Calculation of the speed of sound. Last visited on 2015-14-10, http://www.sengpielaudio.com/calculator-speedsound.htm.

[Trafikstyrelsen, 2014] Trafikstyrelsen (2014). Rules about flying with drones outside in denmark. Last visited on 2015-23-09, http://droneinfo.dk/regler/.

[Trygfonden, 2013] Trygfonden (2013). Druknedødsfald i danmark 2001-2013. Last visited on 2015-14-09, http://www.e-pages.dk/trygfonden/261/.

[TuffWing, 2015] TuffWing (2015). Deltawing. Last visited on 2015-21-09, http://www.tuffwing.com/products/tuffwing_32.html.

[Turbo-ace, 2015] Turbo-ace (2015). Quadcopter. Last visited on 2015-15-10, http://www.turboace.com/turbo-ace-matrix-e-14sg-gyrox5-fpv-ccd.aspx.

[Vicon, 2006] Vicon (2006). Vicon mx hardware system reference. Last visited on 2015-19-11, http://www.udel.edu/PT/Research/MAL/MXhardware_Reference.pdf.

[Vicon, 2013] Vicon (2013). Vicon datastream sdk developer's manual. Last visited on 2015-19-11, http://andrewd.ces.clemson.edu/courses/cpsc412/manuals/Vicon%20DataStream%20SDK%20Manual.pdf.

[Wolff, 2009] Wolff, C. (2009). Radartutorial. Last visited on 2015-07-10, http://www.radartutorial.eu.

[Xilinx, 2008] Xilinx (2008). Microblaze processor reference guide. Last visited on 2015-09-12, .

[Xilinx, 2011a] Xilinx (2011a). Logicore ip divider generator v3.0. Last visited on 2015-09-12, http://www.xilinx.com/support/documentation/ip_documentation/div_gen_ds530.pdf.

[Xilinx, 2011b] Xilinx (2011b). Logicore ip multiplier v11.2. Last visited on 2015-09-12, http://www.xilinx.com/support/documentation/ip_documentation/mult_gen_ds255.pdf.

[Xilinx, 2015] Xilinx (2015). Spartan-6 fpga data sheet: Dc and switching characteristics. Last visited on 2015-14-12, http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf.

Appendix A

A.1 Amplifier and Comparator calculations

This appendix will contain the calculation behind the ultrasound amplifier, capacitors and comparator.

The amplification is the same for 40 kHz and 25 kHz the only differences is where the zeros and poles are placed. The gain for the amplifier is calculated with this equation:

$$20 \cdot log_{10}\Big(\frac{5V}{12,5mV}\Big) = 52,04 \qquad [\text{dB}] \quad (A.1)$$

To calculated how much the opamps must amplify the signal the following equation is used:

$$10^{[\frac{52,04}{20}]} = 400 \qquad [\cdot] \quad (A.2)$$

Which means that the two amplifiers must amplify the signal 20 times($20 \cdot 20 = 400$). To calculated the gain one of the resistors must be set to a value, it is chosen to set R_1 to 1 kΩ, then R_2 can be calculated:

$$20 \cdot 1k\Omega = 20000 \qquad [\Omega] \quad (A.3)$$

Now the capacitors can be calculated for 40 kHz, first C1,C3 the zero is placed at 20 kHz and the pole is placed at 100kHz.

$$C1,3_{40kHz} = \frac{1}{2 \cdot \pi \cdot 20kHz \cdot 1k\Omega} = 7,9 \qquad [\text{nF}] \quad (A.4)$$

It is calculated to be 7,9 nF, because there is no capacitor at 7,9 nF it is chosen to use 10 nF.

$$C2,4_{40kHz} = \frac{1}{2 \cdot \pi \cdot 100kHz \cdot 20k\Omega} = 79,6 \qquad [\text{pF}] \quad (A.5)$$

It is calculated to be 79,6 pF it is chosen to use 82 pF.
For 25 kHz the zero is placed at 10 kHz and the pole is placed 50 kHz, first the calculation for C1,C3.

$$C1,3_{25kHz} = \frac{1}{2 \cdot \pi \cdot 10kHz \cdot 1k\Omega} \qquad [\text{F}] \quad (A.6)$$

It is calculated to 15,9 nF. It is chosen to use a 15 nF capacitor.

$$C2,4_{25kHz} = \frac{1}{2 \cdot \pi \cdot 50kHz \cdot 1k\Omega} \qquad [\text{F}] \quad (A.7)$$

It is calculated to 160 pF. It is chosen to use a 150 pF capacitor.

The voltage divider for the voltage reference on the negative input on the comparator and the voltage divider on the output of the comparator is calculated with the following equation:

$$V_{ref} = \frac{V_{cc} \cdot R_2}{R_1 + R_2} \qquad [\text{V}] \quad (\text{A.8})$$

For the voltage reference on the negative input it is chosen to use a R_5 on 5 kΩ for R_6 it is calculated to be 666 Ω with a V_{ref} on 1 V. For the voltage divider at the output, it is chosen to have a V_{ref} at 3,3 V and a R_8 at 10 kΩ, the output voltage from the comparator is 9 V, R_9 will then be calculated to 5,8 kΩ.

A.2 Calculation of corrected sweep width

The purpose of this appendix is to clarify how the corrected sweep width of a SAR operation helicopter, at 300-500 feet, is calculated by giving an example of a rescue operation.

The conditions of this operation is as follows:

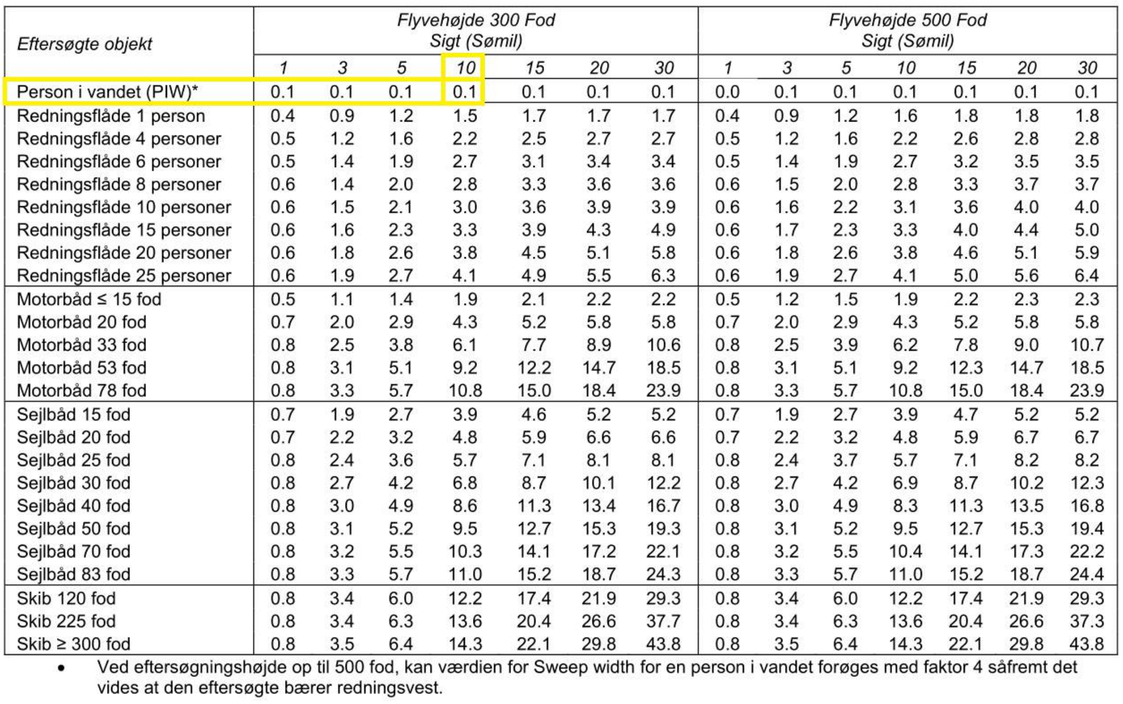
- One person with lifewest is missing.
- Sight is 15 kilometers.
- Windgusts up to 20 m/s.
- The speed of helicopter is 80 km/h.

The corrected sweep width can be found from the following equation:

$$W_c = W_u \cdot F_w \cdot F_v \cdot F_f \qquad \text{[km]} \quad \text{(A.9)}$$

Where:
W_c is the Corrected Sweep Width [m]
W_u is the Uncorrected Sweep Width [m]
F_w is the Weather-correction Factor [·]
F_v is the Velocity-correction Factor [·]
F_f is the Fatigue-correction Factor [·]

Firstly the W_u must be looked up in figure A.1:

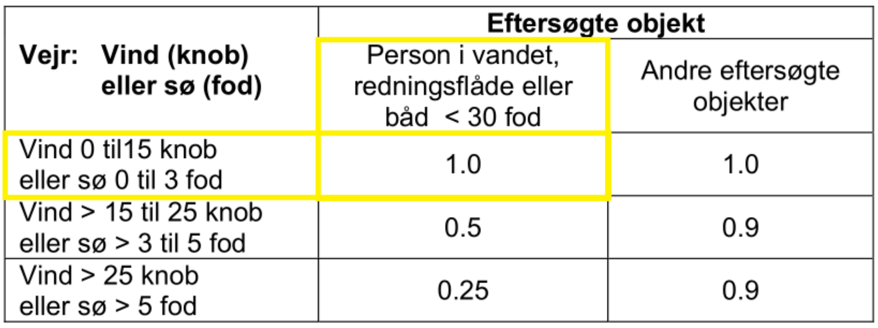


Eftersøgte objekt	Flyvehøjde 300 Fod Sigt (Sømil)							Flyvehøjde 500 Fod Sigt (Sømil)						
	1	3	5	10	15	20	30	1	3	5	10	15	20	30
Person i vandet (PIW)*	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.1
Redningsflåde 1 person	0.4	0.9	1.2	1.5	1.7	1.7	1.7	0.4	0.9	1.2	1.6	1.8	1.8	1.8
Redningsflåde 4 personer	0.5	1.2	1.6	2.2	2.5	2.7	2.7	0.5	1.2	1.6	2.2	2.6	2.8	2.8
Redningsflåde 6 personer	0.5	1.4	1.9	2.7	3.1	3.4	3.4	0.5	1.4	1.9	2.7	3.2	3.5	3.5
Redningsflåde 8 personer	0.6	1.4	2.0	2.8	3.3	3.6	3.6	0.6	1.5	2.0	2.8	3.3	3.7	3.7
Redningsflåde 10 personer	0.6	1.5	2.1	3.0	3.6	3.9	3.9	0.6	1.6	2.2	3.1	3.6	4.0	4.0
Redningsflåde 15 personer	0.6	1.6	2.3	3.3	3.9	4.3	4.9	0.6	1.7	2.3	3.3	4.0	4.4	5.0
Redningsflåde 20 personer	0.6	1.8	2.6	3.8	4.5	5.1	5.8	0.6	1.8	2.6	3.8	4.6	5.1	5.9
Redningsflåde 25 personer	0.6	1.9	2.7	4.1	4.9	5.5	6.3	0.6	1.9	2.7	4.1	5.0	5.6	6.4
Motorbåd ≤ 15 fod	0.5	1.1	1.4	1.9	2.1	2.2	2.2	0.5	1.2	1.5	1.9	2.2	2.3	2.3
Motorbåd 20 fod	0.7	2.0	2.9	4.3	5.2	5.8	5.8	0.7	2.0	2.9	4.3	5.2	5.8	5.8
Motorbåd 33 fod	0.8	2.5	3.8	6.1	7.7	8.9	10.6	0.8	2.5	3.9	6.2	7.8	9.0	10.7
Motorbåd 53 fod	0.8	3.1	5.1	9.2	12.2	14.7	18.5	0.8	3.1	5.1	9.2	12.3	14.7	18.5
Motorbåd 78 fod	0.8	3.3	5.7	10.8	15.0	18.4	23.9	0.8	3.3	5.7	10.8	15.0	18.4	23.9
Sejlbåd 15 fod	0.7	1.9	2.7	3.9	4.6	5.2	5.2	0.7	1.9	2.7	3.9	4.7	5.2	5.2
Sejlbåd 20 fod	0.7	2.2	3.2	4.8	5.9	6.6	6.6	0.7	2.2	3.2	4.8	5.9	6.7	6.7
Sejlbåd 25 fod	0.8	2.4	3.6	5.7	7.1	8.1	8.1	0.8	2.4	3.7	5.7	7.1	8.2	8.2
Sejlbåd 30 fod	0.8	2.7	4.2	6.8	8.7	10.1	12.2	0.8	2.7	4.2	6.9	8.7	10.2	12.3
Sejlbåd 40 fod	0.8	3.0	4.9	8.6	11.3	13.4	16.7	0.8	3.0	4.9	8.3	11.3	13.5	16.8
Sejlbåd 50 fod	0.8	3.1	5.2	9.5	12.7	15.3	19.3	0.8	3.1	5.2	9.5	12.7	15.3	19.4
Sejlbåd 70 fod	0.8	3.2	5.5	10.3	14.1	17.2	22.1	0.8	3.2	5.5	10.4	14.1	17.3	22.2
Sejlbåd 83 fod	0.8	3.3	5.7	11.0	15.2	18.7	24.3	0.8	3.3	5.7	11.0	15.2	18.7	24.4
Skib 120 fod	0.8	3.4	6.0	12.2	17.4	21.9	29.3	0.8	3.4	6.0	12.2	17.4	21.9	29.3
Skib 225 fod	0.8	3.4	6.3	13.6	20.4	26.6	37.7	0.8	3.4	6.3	13.6	20.4	26.6	37.3
Skib ≥ 300 fod	0.8	3.5	6.4	14.3	22.1	29.8	43.8	0.8	3.5	6.4	14.3	22.1	29.8	43.8

- Ved eftersøgningshøjde op til 500 fod, kan værdien for Sweep width for en person i vandet forøges med faktor 4 såfremt det vides at den eftersøgte bærer redningsvest.

Figure A.1: The uncorrected sweep width (300-500 feet) [Redningsrådet, 2013].

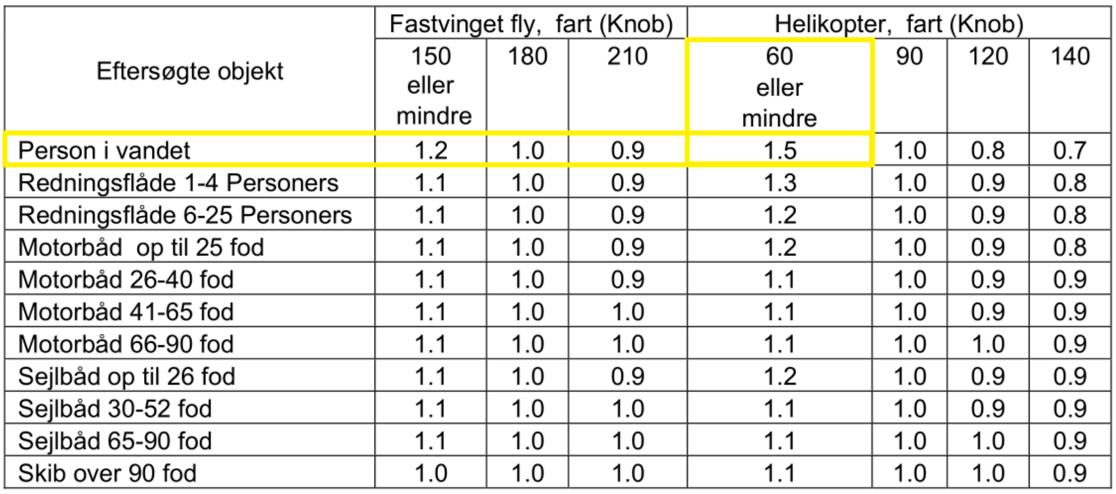
The W_u is 0,1 but because the distressed is wearing a lifejacket this can be multiplied with 4 the correct W_u is 0,4. After the W_u is found a set of factors must be multiplied to the W_u to give the W_c. The first factor is F_w and must be looked up in figure A.2.



Vejr: Vind (knob) eller sø (fod)	Eftersøgte objekt	
	Person i vandet, redningsflåde eller båd < 30 fod	Andre eftersøgte objekter
Vind 0 til15 knob eller sø 0 til 3 fod	1.0	1.0
Vind > 15 til 25 knob eller sø > 3 til 5 fod	0.5	0.9
Vind > 25 knob eller sø > 5 fod	0.25	0.9

Figure A.2: The weather-correction factor [Redningsrådet, 2013].

F_w is found to be 1,0. The second factor is F_v and can be found on figure A.3.



Eftersøgte objekt	Fastvinget fly, fart (Knob)			Helikopter, fart (Knob)			
	150 eller mindre	180	210	60 eller mindre	90	120	140
Person i vandet	1.2	1.0	0.9	1.5	1.0	0.8	0.7
Redningsflåde 1-4 Personers	1.1	1.0	0.9	1.3	1.0	0.9	0.8
Redningsflåde 6-25 Personers	1.1	1.0	0.9	1.2	1.0	0.9	0.8
Motorbåd op til 25 fod	1.1	1.0	0.9	1.2	1.0	0.9	0.8
Motorbåd 26-40 fod	1.1	1.0	0.9	1.1	1.0	0.9	0.9
Motorbåd 41-65 fod	1.1	1.0	1.0	1.1	1.0	0.9	0.9
Motorbåd 66-90 fod	1.1	1.0	1.0	1.1	1.0	1.0	0.9
Sejlbåd op til 26 fod	1.1	1.0	0.9	1.2	1.0	0.9	0.9
Sejlbåd 30-52 fod	1.1	1.0	1.0	1.1	1.0	0.9	0.9
Sejlbåd 65-90 fod	1.1	1.0	1.0	1.1	1.0	1.0	0.9
Skib over 90 fod	1.0	1.0	1.0	1.1	1.0	1.0	0.9

Figure A.3: The velocity-correction factor [Redningsrådet, 2013].

F_v is found to be 1,5. The last factor is F_f which is fatigue factor. This factor of 0,9 can be multiplied if the crew of the helicopter is tired but this is not the case in this scenario.

With these findings W_c can be calculated from equation A.9:

$$W_c = 0,4 \cdot 1 \cdot 1,5 \cdot 1 = 0,6 \qquad \text{[Seamiles]} \quad \text{(A.10)}$$

This result is 1,1 kilometers.

A.3 Calibration of Pixhawk

The purpose of this appendix is to describe the process of calibrating the RC controller and the Pixhawk. For calibration the software program "QGroundControl" [QGroundControl.org, 2015] was used.

Calibration of RC controller

First calibration of the RC controller, following the manual and the software program, revealed problem with the start-up procedure being highly unreliable and not correlate with the manual. If it did start there was issues with the throttle sensitivity. The problem was caused by the throttle influencing to separate channel, where the other channel switch between three flight mode, stabilized, altitude control and position control which determined how much control the operator has over quadcopter. The problem was solved by changing the flight model on the RC controller from helicopter to airplane. The RC controller used for this project can be seen on figure A.4.

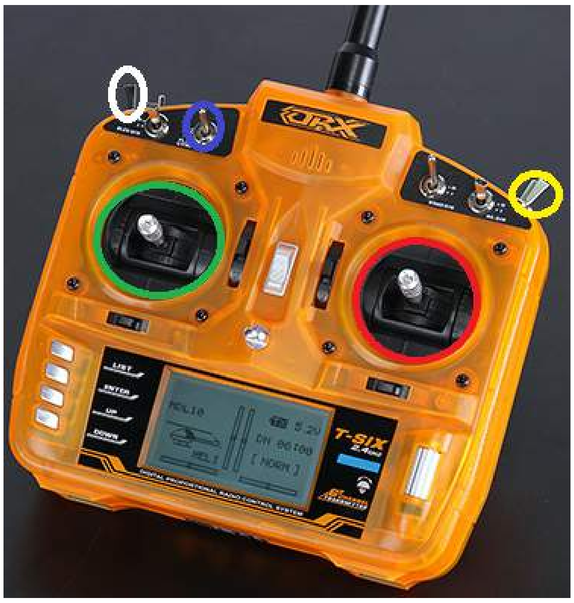


Figure A.4: Figure of OrangeRx T-SIX, were left stick (green) is assigned to pitch and roll, right stick is assigned to throttle and rudder, three position switch (blue) is assigned to flight mode, a two position switch (white) is assigned to the throttle cut and a two position switch (yellow) is assigned to the control system [Hobbyking, 2015b].

The most important control to determine is the start-up and shut-down sequence for the Quad. For start-up, it was determine that the quadcopter must be in the stabilized flight mode. To start the motor, the throttle has to be pressed down and hold fully to the right for three seconds. For shut-down the throttle must again be pressed down and hold fully to the left for three seconds. For safety the motors can be set to a low value with the throttle cut, to ensure the quadcopter wont fly.

Another important control is the switch for the control system, it was decided that when pressed down the operator controls the quadcopter, pressed up the control system controls the quadcopter. A third important control is the switch for flight mode, were it was decided, at stabilized mode the switch would indicate 2, at altitude control mode it would indicate 1 and at position control mode it would indicate 0, although it is not possible to use position control mode as it need GPS data to maintain a position. For more information on the three flight mode, refer to the next section.

It was decided to have pitch and roll on the left stick, will having throttle and yaw on the right stick, as it was consider easier to maintain altitude and because the right stick is not spring loaded in its up/downward motion, although this is possible to change.

Flight mode

The three flight mode, is supposed to make the complexity of controlling the quadcopter relatively easy or advanced, depending on its application.

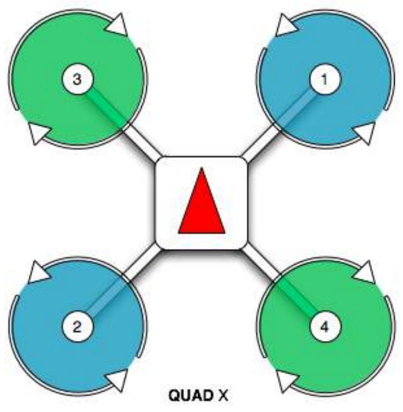
The stabilized mode is in theory, the most difficult among the tree to control. It disable advance flight maneuver by setting a limit to the inclination, that can be forced on the quadcopter and return the quadcopter to a stable level, drifting in the direction that it was given. Throttle is fully controlled by the operator, although the quadcopter will try to maintain its current altitude base on input from the operator [Autopilot,].

The altitude control mode function similar to the stabilized mode, keeping the quadcopter level, but induce a limit on the throttle. This means the throttle has to be closer to its maximum or minimum value to get a corresponding reaction. To keep the rate of climb and sink manageable, the flight mode sets a maximum rate. Thereby the quadcopter will maintain its altitude, unless the operator wants it climb or sink [Autopilot,].

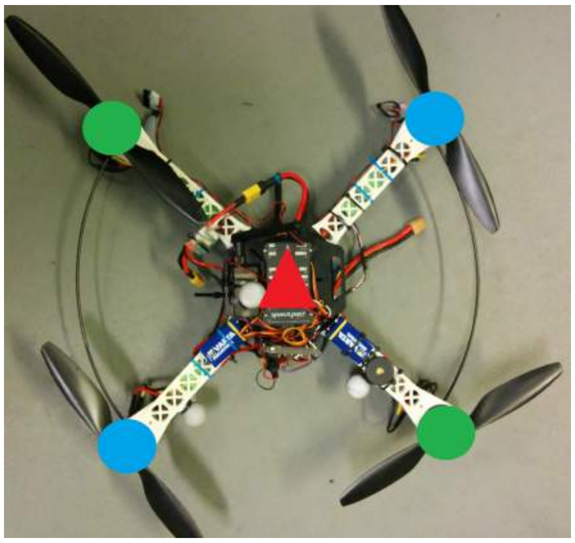
The position control mode combine stabilized and altitude control mode, but also restricting the pitch and roll control in the same way as with the throttle control. The quadcopter will then counteract any movement given by the operator, when the controls is returned to its center value, thereby maintaining its position and altitude. This is in theory the most simple mode to control of the tree flight modes, but requires GPS data to work [Autopilot,].

Calibration of Pixhawk

To ensure the calibration of the quadcopter works properly with the Pixhawk flight stack, the motor assignment must fit the quadcopter layout. It was found that a Quad X frame match and can be seen on figure A.5. If the motors is not connected in the correct order, or if the Pixhawk is placed with the white arrow in the wrong direction, compared to the layout, the quadcopter will not fly properly.



(a) [Autopilot, 2014]



(b)

Figure A.5: Figure of motor layout for Quad X (a) compared with actual Quad (b), were motor 1 and 2 (blue) rotates counter clockwise and motor 3 and 4 (green) rotates clockwise. The red arrow indicates the Pixhawk

To calibrate the Pixhawk its firmware is first updated, which is possible with the software QGroundControl [QGroundControl.org, 2015]. Then the airframe is selected, in this cases the Quad X frame. The RC controller including flight mode is then calibrated according with the RC calibration section, by following the instruction given by the software program. The Pixhawk have three sensors which has to be calibrate, these are the compass, the gyroscope and the accelerometer. The calibration is done by following the on screen prompt given by the software QGroundControl. The last calibration needed, requires the quadcopter to be placed on a level surface, thereby the Pixhawk can find its level flight position. The main calibration is considered done when the setup tabs shows green, in QGroundControl and the quadcopter should be able to fly relatively stable.

Even if the Pixhawk is calibrate there can still be problems with the stability of the quadcopter, to some extent this can be solved by changing some of the parameters in the software. Another method is to distribute the weight or placement of the equipment on the quadcopter.

A.4 Control implementation test

The purpose of this appendix is to compare the implementation of the D controller with the seven bit shiftet controller to validate the calculation, and therefore, there will be no explaning in between the calculations.

Calculation without bit shifting

$$R_T = -R_{ref} \cdot K_P \quad [\cdot] \tag{A.11}$$

$$V_T = \frac{\left(-\frac{V_{Told}}{\frac{t_s}{t_p}} + \frac{R_{Told}}{t_s} - \frac{R_T}{t_s} \right)}{1 + \frac{t_p}{t_s}} \cdot K_D \quad [\cdot] \tag{A.12}$$

$$O = \frac{V_T + R_T}{1000} \quad [\cdot] \tag{A.13}$$

$$O_{old} = \frac{V_{Told} + R_{Told}}{1000} \quad [\cdot] \tag{A.14}$$

$$V_T = O \cdot 10 - R_T \quad [\cdot] \tag{A.15}$$

$$V_{Told} = O_{old} \cdot 10 - R_{Told} \quad [\cdot] \tag{A.16}$$

$$O \cdot 10 - R_T = \frac{\left(\frac{-O_{old} \cdot 10 - R_{Told}}{\frac{t_s}{t_p}} + \frac{R_{Told}}{t_s} - \frac{R_T}{t_s} \right)}{1 + \frac{t_p}{t_s}} \cdot K_P \quad [\cdot] \tag{A.17}$$

$$\begin{aligned} O \cdot 10 + R_{ref} \cdot K_P = & \frac{\left(\frac{-O_{old} \cdot 10 + R_{refold} \cdot K_P}{\frac{t_s}{t_p}} - \frac{R_{refold} \cdot K_P}{t_s} \right.}{1 + \frac{t_p}{t_s}} \\ & + \frac{\left. \frac{R_{ref} \cdot K_P}{t_s} \right)}{1 + \frac{t_p}{t_s}} \cdot K_P [\cdot] \end{aligned} \tag{A.18}$$

$$\begin{aligned} 10 \cdot O(Z) + \frac{\frac{Z^{-1} \cdot O(Z) \cdot K_D \cdot 10}{\frac{t_s}{t_p}}}{1 + \frac{t_p}{t_s}} = & Z^{-1} \cdot R_{ref}(Z) \cdot \frac{\left(\frac{K_P \cdot K_D}{\frac{t_s}{t_p}} - \frac{K_P \cdot K_D}{t_s} \right)}{1 + \frac{t_p}{t_s}} \\ & + R_{ref}(Z) \cdot \left(-K_P + \frac{\frac{K_P \cdot K_D}{t_s}}{1 + \frac{t_p}{t_s}} \right) [\cdot] \end{aligned} \tag{A.19}$$

$$A = 10 \quad [\cdot] \tag{A.20}$$

$$B = \frac{\frac{K_D \cdot 10}{\frac{t_s}{t_p}}}{1 + \frac{t_p}{t_s}} \qquad [\cdot] \quad (A.21)$$

$$C = \frac{\frac{K_D \cdot K_P}{\frac{t_s}{t_p}} - \frac{K_P \cdot K_D}{t_s}}{1 + \frac{t_p}{t_s}} \qquad [\cdot] \quad (A.22)$$

$$D = -K_P + \frac{\frac{K_P \cdot K_D}{t_s}}{1 + \frac{t_p}{t_s}} \qquad [\cdot] \quad (A.23)$$

$$A \cdot O(Z) + Z^{-1} \cdot R_{ref}(Z) \cdot B = Z^{-1} \cdot R_{ref}(Z) \cdot C + R_{ref}(Z) \qquad [\cdot] \quad (A.24)$$

$$\frac{O(Z)}{R_{ref}(Z)} = \frac{Z^{-1} \cdot R_{ref}(Z) \cdot C + D}{Z^{-1} R_{ref}(Z) \cdot B + A} \qquad [\cdot] \quad (A.25)$$

Calculation with bit shifting

$$R_T = -R_{ref} \cdot K_P \cdot 2^{10} \qquad [\cdot] \quad (A.26)$$

$$V_T = \frac{\left(- \frac{V_{Told}}{\frac{t_s \cdot 2^{10}}{t_p}} + \frac{R_{Told}}{t_s \cdot 2^{10}} - \frac{R_T}{t_s \cdot 2^{10}} \right)}{1 + \frac{t_p}{t_s}} \cdot K_D \cdot 2^{10} \qquad [\cdot] \quad (A.27)$$

$$O = \frac{V_T + R_T}{1000 \cdot 2^{10}} \qquad [\cdot] \quad (A.28)$$

$$O_{old} = \frac{V_{Told} + R_{Told}}{1000 \cdot 2^{10}} \qquad [\cdot] \quad (A.29)$$

$$V_T = O \cdot 10 \cdot 2^{10} - R_T \qquad [\cdot] \quad (A.30)$$

$$V_{Told} = O_{old} \cdot 10 \cdot 2^{10} - R_{Told} \qquad [\cdot] \quad (A.31)$$

$$O \cdot 10 \cdot 2^{10} - R_T = \frac{\left(\frac{-O_{old} \cdot 10 \cdot 2^{10} - R_{Told}}{\frac{t_s}{t_p} \cdot 2^{10}} + \frac{R_{Told}}{t_s \cdot 2^{10}} - \frac{R_T}{t_s \cdot 2^{10}} \right)}{1 + \frac{t_p}{t_s}} \cdot K_P \cdot 2^{10} \qquad [\cdot] \quad (A.32)$$

$$O \cdot 10 \cdot 2^{10} + R_{ref} \cdot K_P \cdot 2^{10} = \frac{\left(\frac{-O_{old}\cdot 10 + R_{refold}\cdot K_P}{\frac{t_s}{t_p}} - \frac{R_{refold}\cdot K_P}{t_s} + \frac{R_{ref}\cdot K_P}{t_s}\right)}{1+\frac{t_p}{t_s}} \cdot K_P \cdot 2^{10} [\cdot] \tag{A.33}$$

$$O(Z)\cdot 10\cdot 2^{10} + \frac{\frac{Z^{-1}\cdot O(Z)\cdot K_D\cdot 10\cdot 2^{10}}{\frac{t_s}{t_p}}}{1+\frac{t_p}{t_s}} = Z^{-1}\cdot R_{ref}(Z)\cdot\left(\frac{\frac{K_P\cdot K_D\cdot 2^{10}}{\frac{t_s}{t_p}}}{1+\frac{t_p}{t_s}} - \frac{\frac{K_P\cdot K_D\cdot 2^{10}}{t_s}}{1+\frac{t_p}{t_s}}\right) + R_{ref}(Z)\cdot\left(-K_P\cdot 2^{10} + \frac{\frac{K_P\cdot K_D\cdot 2^{10}}{t_s}}{1+\frac{t_p}{t_s}}\right)[\cdot] \tag{A.34}$$

$$A = 10\cdot 2^{10} \quad [\cdot] \tag{A.35}$$

$$B = \frac{\frac{K_D\cdot 10\cdot 2^{10}}{\frac{t_s}{t_p}}}{1+\frac{t_p}{t_s}} \quad [\cdot] \tag{A.36}$$

$$C = \frac{\frac{K_D\cdot K_P\cdot 2^{10}}{\frac{t_s}{t_p}} - \frac{K_P\cdot K_D\cdot 2^{10}}{t_s}}{1+\frac{t_p}{t_s}} \quad [\cdot] \tag{A.37}$$

$$D = -K_P\cdot 2^{10} + \frac{K_P\cdot\frac{K_D\cdot 2^{10}}{t_s}}{1+\frac{t_p}{t_s}} \quad [\cdot] \tag{A.38}$$

In equation A.35, A.36, A.37 and A.38 all the parts are multiplyed with 2^{10} and can therefore be multiplyed with $\frac{1}{2^{10}}$ to remove it, thus it will match A.20, A.21, A.22 and A.23.

$$A\cdot O(Z) + Z^{-1}\cdot R_{ref}(Z)\cdot B = Z^{-1}\cdot R_{ref}(Z)\cdot C + R_{ref}(Z) \quad [\cdot] \tag{A.39}$$

$$\frac{O(Z)}{R_{ref}(Z)} = \frac{Z^{-1}\cdot R_{ref}(Z)\cdot C + D}{Z^{-1}R_{ref}(Z)\cdot B + A} \quad [\cdot] \tag{A.40}$$

Conclusion

It can be concluded that the calculation for the bit shifting is validated.

A.5 Logic cores

The purpose of this appendix is to describe the logic core for multiplier and divider provided by Xilinx.

Since multiplication and division typically requires more resources on the FPGA, it can be necessary to implement a dedicate process to save resources. One way this can be done is to use logic cores, and can often be implemented on digital signal processing (DSP) slices which are more efficient at arithmetic operations.

LogiCORE IP Multiplier

The IP core is design to generate fixed-point parallel and constant-coefficient multipliers for two compliment signed or unsigned data, either can be selected though a graphical user interface (GUI). For this project were parallel multiplier implemented on DSP slices for highest performance chosen. The core support input ranging from 1 to 64 bit and a output ranging from 1 to 128 bit, but it is only necessary have a input ranging from 1 to 32 bit and a output ranging from 1 to 64 [Xilinx, 2011b].

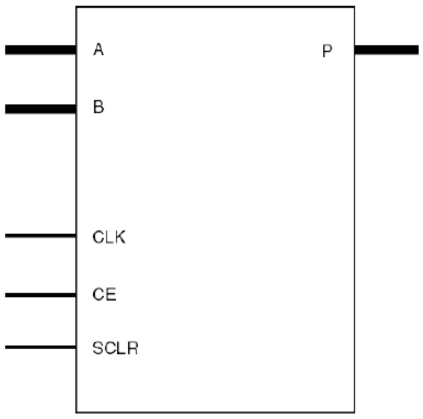


Figure A.6: Figure of the schematic symbol for IP core [Xilinx, 2011b].

On figure A.6 is the schematic symbol for the multiplier core were [Xilinx, 2011b]:

- Input A is a operand bus.
- Input B is a operand bus.
- Output P is a product.
- Input CLK is a rising-edge clock.
- Input CE is a clock enable.
- Input SCLR is synchronous clear.

The clock enable and synchronous clear is not used in this design. From the datasheet [Xilinx, 2011b] it was found that with the implemented design, the core latency would be 8 cycles.

LogiCORE IP Divider

The IP core is design for integer division with operand of up to 54 bit and can perform Radix-2 or High Radix, either can be selected though a GUI. Radix-2 is typical use for application requiring high throughput or operand with less than 16 bit, while High Radix is use with more than 16 bit. For this project only Radix-2 was required, having an integer remainder. The ranging for dividend and quotient is set from 2 to 32 and from 2 to 15 for the divisor. To reduce the amount of resources used on the FPGA, the throughput has been set to 8 cycle per division [Xilinx, 2011a].

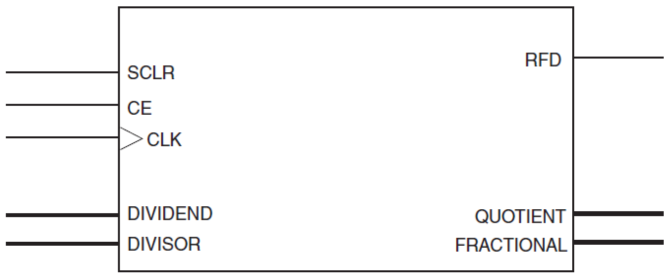


Figure A.7: Figure of the schematic symbol for IP core [Xilinx, 2011a].

On figure A.7 is the schematic symbol for the divider core were [Xilinx, 2011a]:

- Input DIVIDEND is a data bit width specified by dividend and quotient width.
- Input Divisor is a data bit width specified by divisor width.
- Output QUOTIENT is the result of the integer division of dividend by divisor.
- Output FRACTIONAL is the result data bit width determined by Divisor Width.
- Output RFD indicates the cycle in which input data is sampled by the core.
- Input CLK is a rising-edge clock.
- Input CE is a clock enable.
- Input SCLR is synchronous clear.

The clock enable, synchronous clear and fractional is not used in this design. Ready For Data (RFD) indicate it only samples data on the 8th enabled clock rising edge. From the datasheet [Xilinx, 2011a] it was found that with the implemented design, the core latency would be 37 cycles.

In this section, is was found that logic cores use less resources and are more efficient at arithmetic operations, such as multiplication and division, at the cost of latency.

A.6 Measurement report - Controllers

Purpose:
The purpose of this measurement report is to determine whether the controller for pitch, roll and yaw, correspondent with the models design in chapter 8.

Theory:
A signal from the RC transmitter trigger a hardcoded unit step to be sent from the FPGA to the Pixhawk, this will make the quadcopter move either forward, sideway or make it rotate. The distance over time is measured using the Vicon system in MTLab.

Measurement set up:

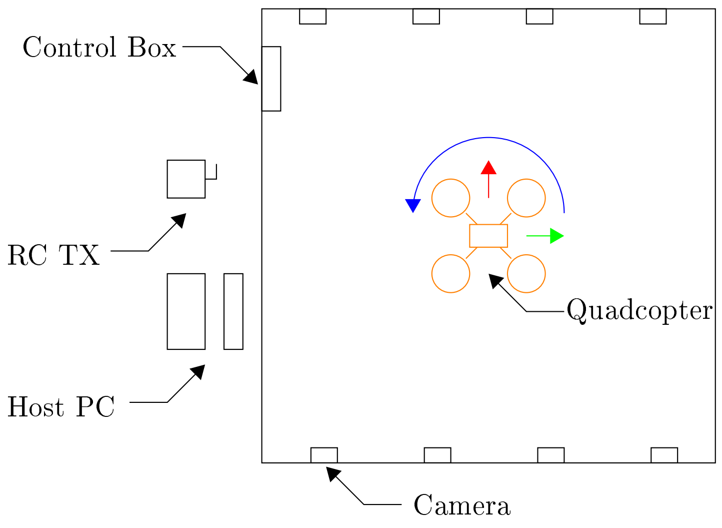


Figure A.8: Illustration of measurement set up, were the Quads (orange)step responses for pitch (green), roll (red) and yaw (blue) is measured with Vicon system.

Equipment list:

- Vicon MX (AAU 75459)
- Host PC (AAU 86703)
- Quadcopter
- OrangeRx R620 receiver
- OrangeRx T-SIX transmitter

Procedure:
This test requires two person, Person 1 control the RC transmitter and person 2 the host pc.

1. When entering the lab remember to lock the door and turn on the flight indication light.
2. Turn on power to the Vicon control box and the host computer.
3. Start the tracker program and select the "gr512quad" model under the object tab.
4. Start MatLab and select the "ViconDataStreamSDK_MATLABTest" script.
5. Program the FPGA and restart it(/CD/Measurement report/Measurement report - Controllers/yaw.bit or pitch.bit or roll.bit).
6. Place the quadcopter as seen on figure A.8 and check it show up on on the host PC.
7. Connect the battery on the quadcopter and arm it.

8. Scroll down in the script and insert test file name.
9. Start the quadcopter and the MatLab script.
10. Person 1 will hold and maintain a position, close to a wall, to ensure most possible distance.
11. When person 1 say "start", person 2 press enter on the host PC and say "start".
12. Person 1 flip the channel 5 switch.
13. Person 1 again flip channel 5 and stabilise the quadcopter before it hit the opposite wall.
14. Person 2 press enter twice to stop the script and save the workspace.
15. Person 1 lands and shut down the quadcopter.

Measurement data:

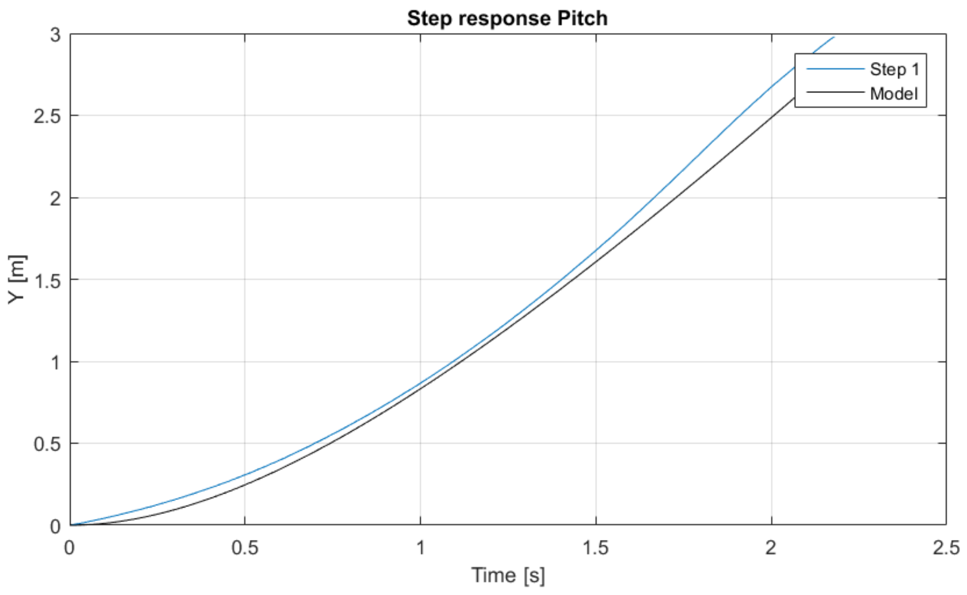


Figure A.9: Graph of the change in distance over time for the implemented pitch controller compared with the model. The measured data has correction to the start point, length and errors.

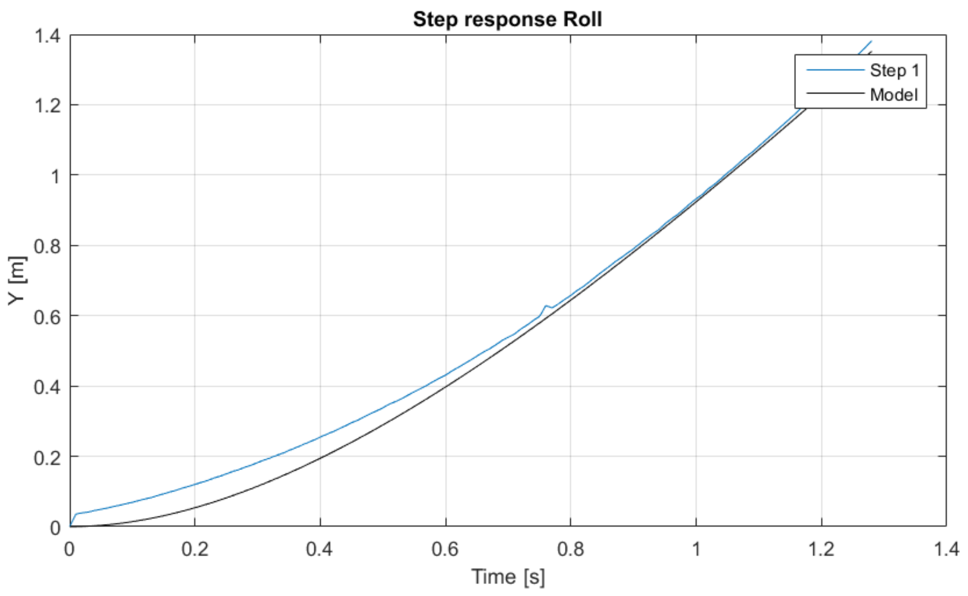


Figure A.10: Graph of the change in distance over time for the implemented roll controller compared with the model. The measured data has correction to the start point, length and errors.

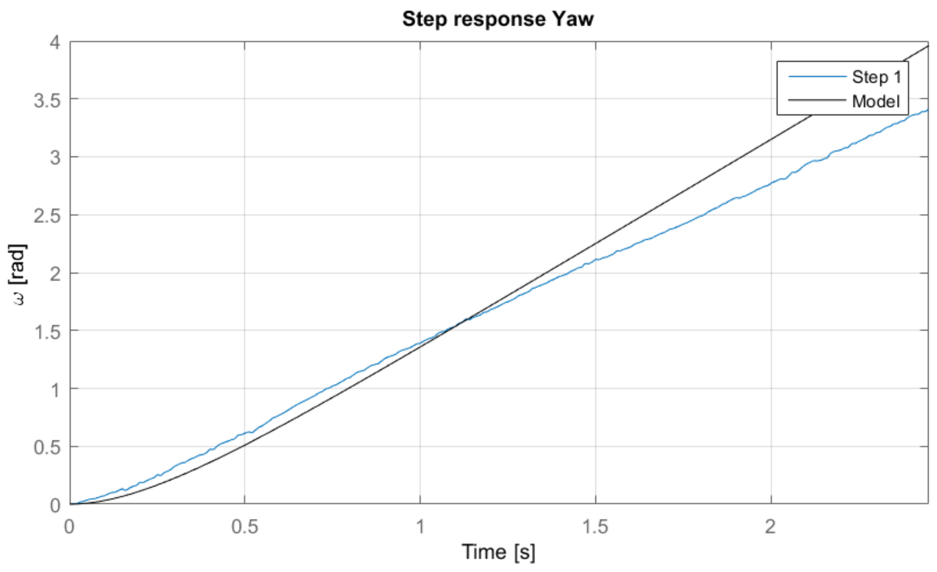


Figure A.11: Graph of the change in distance over time for the implemented yaw controller compared with the model. The measured data has correction to the start point, length and errors.

Result:
The step response for pitch is seen on figure A.9, shows the response from a flight at 20 percent step input. The quadcopter starts from a hovering position, receive a step input and is plotted together with the model. It can be seen that Step 1 follows the model indicating that the implemented controller works as intended. The separation after 1,5 seconds might be the quadcopter losing attitude and thereby accelerating more.

The step response for roll is seen on figure A.10, shows the response from a flight at

20 percent step input. Same method as with pitch is use to get the step response. Step 1 follows the model which indicates that the implemented controller works as intended.

The step response for yaw is seen on figure A.11, shows the response from a 100 percent step input. Where the quadcopter as with pitch and roll starts from a hovering position, receive a step input and start turning around the z-axes and is plotted together with the model. The step follows the model at the start but begins to deviate at around the 1 seconds mark, this again could be the quadcopter adjusting to maintain height.

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement equipment tolerance and delays.
- Stability of quadcopter.
- Human factor.
- Noise from the equipment.
- Ambient noise.

Measurement equipment tolerance and delays:
The Vicon system did have a relative large delay from start of the matlab script to it actually start saves the stream, which meant the correct start point would not be saved from the stream, requiring the steam to started prior to the starting point. Another problem occurs where the Vicon system lost track of the quadcopter, causing data with errors, when it was relative close to the wall of the lab. For reliable tracking the quadcopter had to be position closer to center, reducing the maximum distance it could travel.

Stability of quadcopter:
There was stability issue with the quadcopter, causing it possible drift in a direction, that meant the quadcopter could already be moving before the step was sent. Furthermore could it give faulty data, if the quadcopter drifted at an angle, as the measured distance would be lower.

Human factor:
To rectify problem with the Vicon system and the stability of the quadcopter, human would decide when stability is achieved and when the step started.

Conclusion:
It can be concluded that the controller implemented for pitch, roll and yaw represent the model well.

A.7 Measurement report - PPM encoder module

Purpose:
The purpose of this measurement report is to determine the structure of PPM signal and frame for the PPM encoder module.

Theory:
PWM Signal, from the RC transmitter, is received by the RC receiver. The receiver is connected to the PPM encoder and the signal is process by a microcontroller on the PPM encoder. The encoded PPM signal is measured to find the length of, frame, frame spacing, spike, channel one to eight and the amplitude.

Measurement set up:

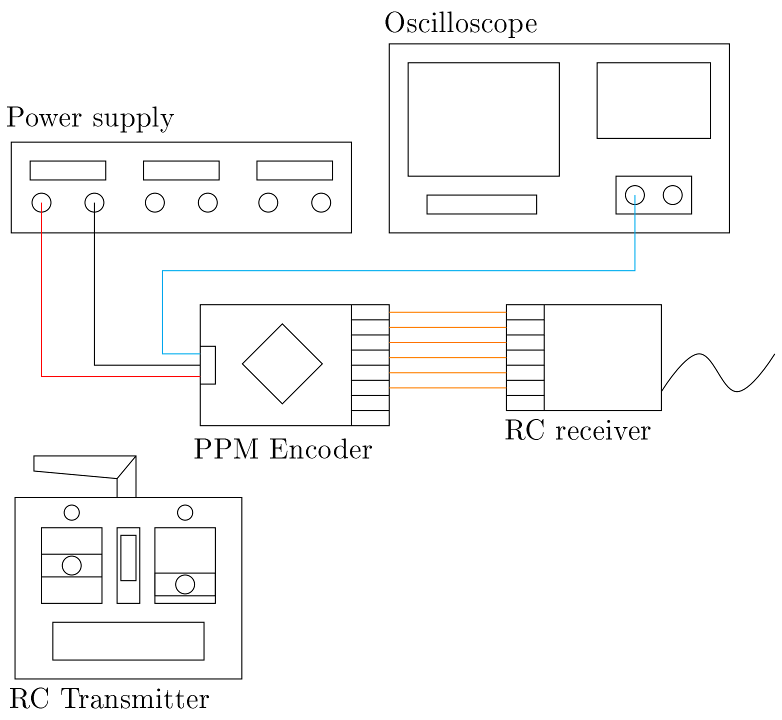


Figure A.12: Illustration of measurement set up for PPM encoder, were red is 5 V, black is ground, blue is PPM signal and orange is connection between encoder and receiver.

Equipment list:

- Power supply (AAU 52751)
- Oscilloscope (AAU 56684)
- 3DR PPM encoder module
- OrangeRx R620 receiver
- OrangeRx T-SIX transmitter

Procedure:

1. Set up and connect the equipment as on figure A.12.
2. Turn power on and adjust the oscilloscope until a full frame is visible.
3. Observe and note which channel correspond to throttle, yaw, elevation, aileron and flight mode.

4. Measure and note pulse width for all channels, in the two extreme position.
5. Observe and note which pulse width for a channel that correspond to sink, climb, rotate left/right, moving backward/forward and moving left/right.
6. Observe and note which pulse width for each flight mode.
7. Measure and note pulse width for all spikes between each channel.
8. Measure and note a full frame and frame spacing, in the two extreme position (requires one person to handle the transmitter and one to operate the oscilloscope).

Measurement result: Channels observation and measurement:

- Channel 1: Throttle (680 μs to 1520 μs)
  / with "throttle stop" activated on RC (500 μs)
- Channel 2: Yaw (680 μs to 1520 μs)
- Channel 3: Elevation (680 μs to 1520 μs)
- Channel 4: Aileron (680 μs to 1520 μs)
- Channel 5: Gear (680 μs to 1520 μs)
- Channel 6: AUX (680 μs to 1520 μs)
- Channel 7: Not connected (1100 μs)
- Channel 8: Not connected (1100 μs)
- Center value for channel 1-4 and 6 (1100 μs)

Channels observed reaction (680 μs - 1520 μs):

- Channel 1: Decrease altitude - increase altitude
  - With "throttle stop" activated on RC (500 μs)
- Channel 2: Rotates right - rotates left
- Channel 3: Moving backward - moving forward
- Channel 4: Moving Right - moving left
- Channel 5: F.mode gear switch, between two positions (bot - top)
- Channel 6: Flap/Gyro switch, between three positions, down (680 μs), mid (1096 μs) and up (1520 μs)
- Channel 7: N/A.
- Channel 8: N/A.

Frame measurement.

- Channel separation (between all channels there is 400 μs).
- Frame (20,32 ms to 25,30 ms).
- Frame spacing (10,50 ms).

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
It can be concluded that structure, of the PPM signal from the 3DR PPM encoder, has been determined for each channel and the complete PPM frame.

A.8 Measurement report - PPM encoding delay and precision

Purpose:
The purpose of this measurement report is to determine the delay and precision of the combined PWM reading module and PPM output module to see if requirement 4 and 5 in specifications is met. This test is performed from the description in section 4.4.

Theory:
The 3DR PPM encoder and the FPGA loaded with the PWM reader and PPM output modules is connected to the OrangeRx R620 receiver and the output of the two is compared to the PWM input to determine the delay between them. To determine the precision of the constructed PWM and PPM modules the PPM signal parts (framespacing, channels and channel seperation) and the PWM width is measured and compared to confirm that the previous measured results is reconstructed within the required limit.

Measurement set up:

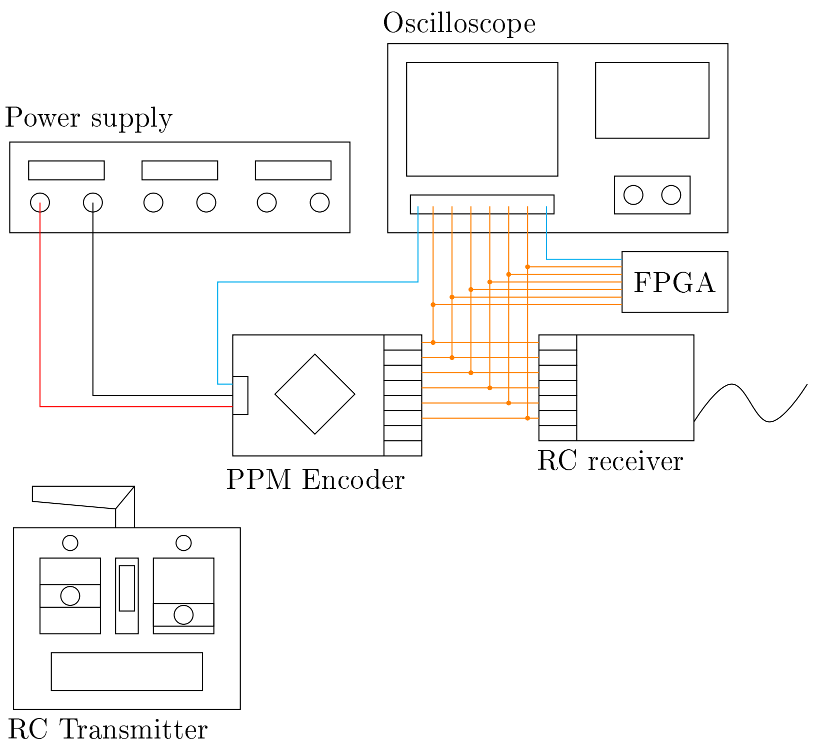


Figure A.13: Illustration of measurement set up for PPM encoder delay and precision measurement, where red is 5 V, black is ground, blue is PPM signal and orange PWM channels 1-6.

Equipment list:

- Power supply (AAU 52754)
- Oscilloscope (AAU 52772)
- 3DR PPM encoder
- OrangeRx R620 RC receiver
- OrangeRx T-SIX transmitter

- XuLA2-LX9

Procedure:

1. Set up the equipment as shown on figure A.13.
2. Upload program to FPGA, program can be found on CD/Measurement reports/Measurement Report- PPM encoding delay and precision/PPM_delay.
3. V_{CC} to OrangeRx R620 receiver is disconnected
4. Turn on powersupply and OrangeRx T-SIX transmitter.
5. Switches and sticks is set to default position and throttle to low on OrangeRx T-SIX transmitter.
6. Oscilloscope is set to trigger on rising edge on PWM signal from channel 6.
7. Connect V_{CC} to OrangeRx R620 reciver.
8. Measure difference in distance between PPM frames from the first generated PWM frames.
9. Repeat test.

Measurement procedure precision

1. Repeat step 1,2,4 and 5 from delay procedure
2. Measure every part of a single PPM frame on the oscilloscope with one μs resolution.
3. Measure the PWM width on every channel with one μs resolution on the oscilloscope.
4. Change percent offset from -50 to 50 on channel 1 (throttle) and load program in to FPGA.
5. Measure channel 1 (throttle) width

Measurement result delay:
Channels observation and measurement:

	FPGA	3DR PPM encoder
Test 1	200 ns	48,8 ms
Test 2	200 ns	74,8 ms

Table A.1: Measured delays from the lastest falling edge of the first wave of PWM signals (above minimum length of 450 μs) to the start of the PPM frame.

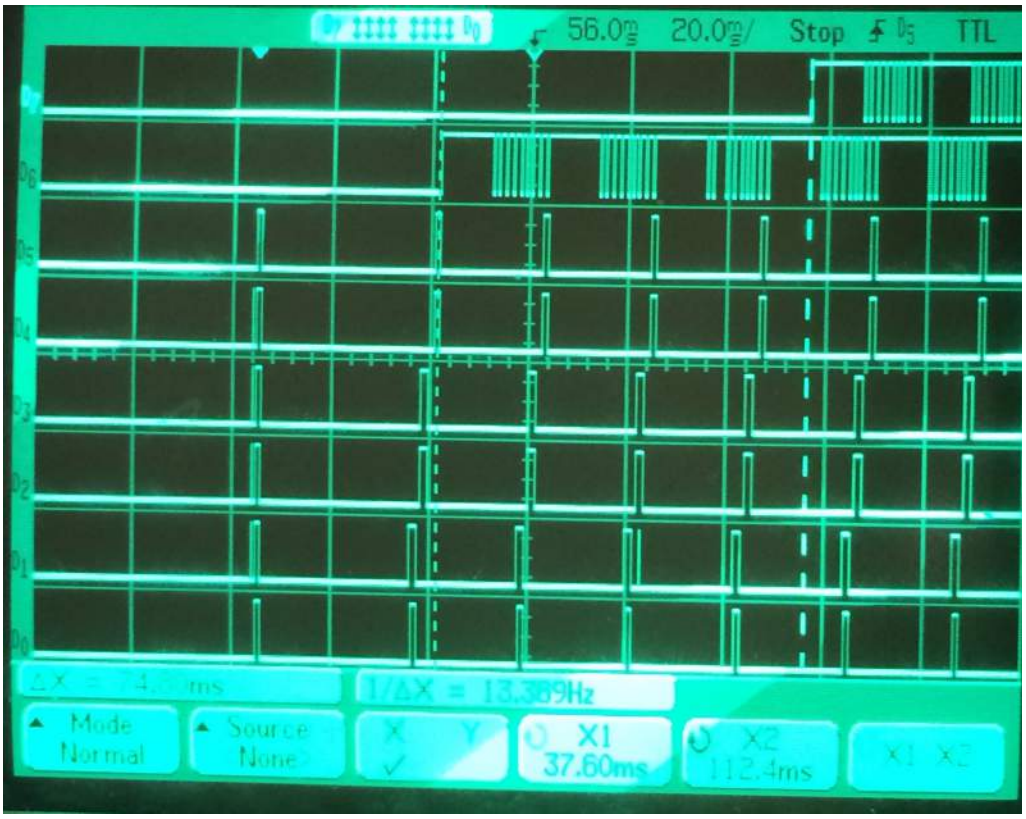


Figure A.14: Oscilloscope shot of PWM channels 1-6 (D0-D5), PPM-FPGA (D6) and PPM-encoder (D7).

Measurement result precision:

Signal part	PPM [μs]	PWM [μs]	Expected (μs)	Deviation (μs / %)
0 (Framespace)	10500,4		10500	+0,4 / < 0,001
1 (Chn sep)	400,12		400	+0,12 / < 0,001
2 (Chn 1)	680,28	1080,2	680,2	+0,08 / < 0,001
3 (Chn sep)	400,08		400	+0,08 / < 0,001
4 (Chn 2)	1098,96	1498,88	1098,88	+0,08 / < 0,001
5 (Chn sep)	400,12		400	+0,12 / < 0,001
6 (Chn 3)	1098,96	1498,88	1098,88	+0,08 / < 0,001
7 (Chn sep)	400,12		400	+0,12 / < 0,001
8 (Chn 4)	1098,96	1498,88	1098,88	+0,08 / < 0,001
9 (Chn sep)	400,08		400	+0,08 / < 0,001
10 (Chn 5)	680,28	1080,2	680,2	+0,08 / < 0,001
11 (Chn sep)	400,08		400	+0,08 / < 0,001
12 (Chn 6)	680,28	1080,2	680,2	+0,08 / < 0,001
13 (Chn sep)	400,12		400	+0,12 / < 0,001
14 (Chn 7)	1100,12		1100	+0,12 / < 0,001
15 (Chn sep)	400,08		400	+0,08 / < 0,001
16 (Chn 8)	1100,12		1100	+0,12 / < 0,001
17 (Chn sep)	400,2		400	+0,2 / < 0,001
Resolution test with hardcoded values				
chn8 + 1 tick (83 ns)	1100,2		1100,2	0 / 0
chn8 + 12 ticks (1 μs)	1101,12		1101,12	0 / 0

Table A.2: Measured and expexted PPM part values (expected channel values is measured PWM - (PWM to PPM offset(400 μs) A.7, A.9.

Channel	Offset [%]	Offset measured [μs]	Expected [μs]	deviation [μs] / [%]
1	-50	680,1	680 (limit)	+0,1 / < 0,001
1	50	1520,12	1520 (limit)	+0,12 / < 0,001
1*	50	501,34	501,36 ((PWM 901,36)- 400)	-0,02 / < -0,001
2	50	1311,38	1311,38	0 / 0
3	-50	886,44	886,38	+0,06 / < 0,001
4	50	1311,46	1311,38	+0,08 / < 0,001

Table A.3: Offset values applied when switch is activated on OrangeRx T-SIX RC (* throttle stop activated)

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Discussion:
From the delay test the deviation of the result measured from the 3DR PPM encoder it is not possible to conclude based on the applied testing parameters how much delay is induced in the conversion from PWM to PPM in the 3DR PPM encoder. By studying figure A.14 it would seem like the FPGA has an advandtage compared to the 3DR PPM

encoder. But the diversity in the results from the PPM encoder makes it hard to tell from which PWM frames the PPM signal is created without changing testing parameters. This is a viable option if wanted/needed to be pursued, but due to time constraints and the results obtained from the constructed modules the project group do not wish to do so. Besides the inconclusive delay, another problem became obvious, which is shown on figure A.15.

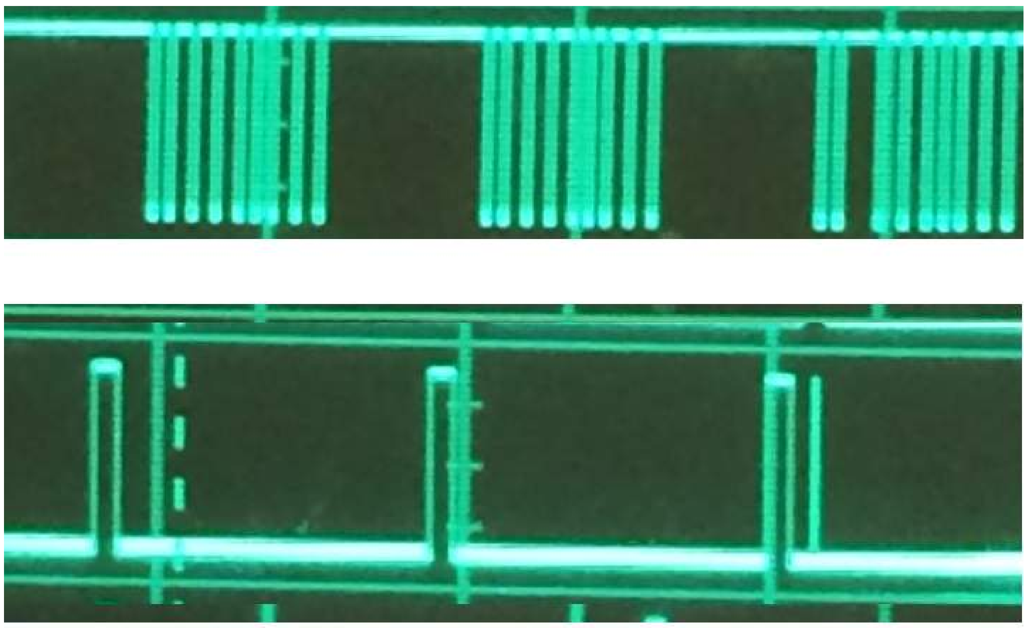


Figure A.15: Irregularity on PWM channel 2 leading to width on channel 2 in PPM frame to pass limit

The small spike to the right of the legitimit channel is recorded in the PWM module, and as the PWM to PPM offset is subtracted the recorded value turns in to a negative value. This results in a number with a high value since all variables is instigated as unsigned integers. Instead of seeing the value as negative it produces a large number by setting the most significant bit to one. In this case the value will be a number between 16384 and 32767, which in clock ticks equals to a channel width of 1365,3 μs to 2730,6 μs. This irregularity was only reproduceable in the first few PWM frames, and is as such not a big problem for the consistency of the constructed PPM encoder. The reason is that it takes at least a few seconds to arm the Pixhawk for flight and then the irregularity has disappeared. A future correction of this could be to implement a debounce function or skip some of the first PWM frames. For the precision testing there has been detected a 80 ns delay between the change of the PPM parts. The reason for the delay is that one clock cycle is used to latch the next part, before the signal counter is reset to null. A quick solution to this to improve the precision would be to reset the counter to one. Due to the results attained and lack of time to verify this change it is not implemented.

Conclusion:
It can be concluded that the constructed PWM and PPM module with the included offset limiters work, and does so with a precision that deviates less than one microsecond in channel width. It can also be concluded that the delay in the constructed system is relatively low. In the comparison with the 3DR PPM encoder the amount of delay between the two systems is inconclusive but is assumed not to be a problem. The same is assumed with the observed irregullarity on channel 2 of the PWM signal.

A.9 Measurement report - PWM OrangeRx R620 receiver

Purpose:
The purpose of this measurement report is to determine the structure of the PWM signals for the OrangeRx R620 receiver in conjunction with the PPM encoder module from 3D Robotics.

Theory:
The signal, from the RC transmitter, is received by the RC receiver. The RC receiver is connected to the PPM encoder and the PWM signals sent between them is then processed by a microcontroller on the PPM encoder. The PWM signal is measured to find the length of channel one to six and the amplitude of them. It is also examined what happens if the RC transmitter is suddenly shut off, to emulate a battery failure.

Measurement set up:

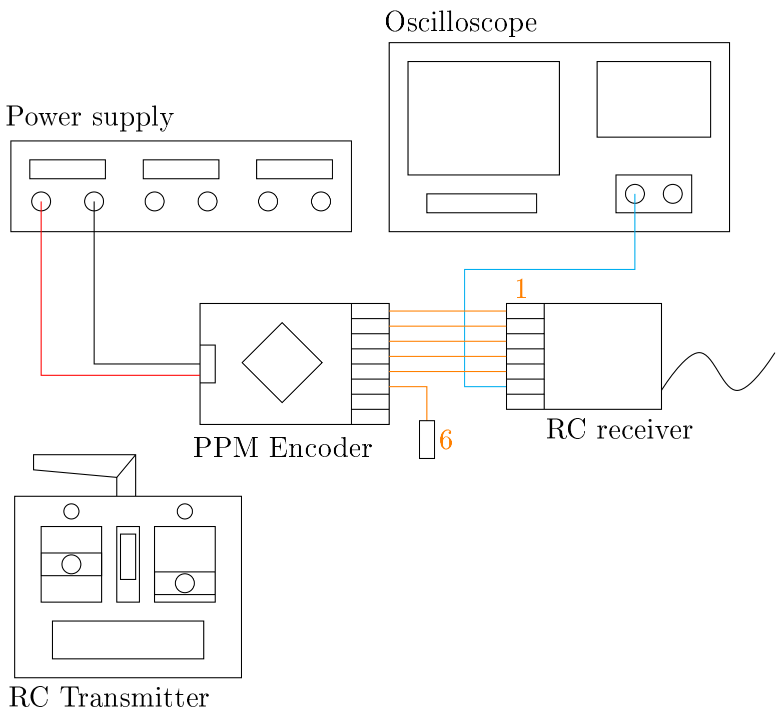


Figure A.16: Illustration of measurement set up for the PWM RC receiver, where red is 5 V, black is ground, blue is signal and orange is connection between encoder and receiver.

Equipment list:

- Power supply (AAU 52751)
- Oscilloscope (AAU 56684)
- 3DR PPM encoder module
- OrangeRx R620 receiver
- OrangeRx T-SIX transmitter

Procedure:
The encoder powers the RC receiver though connection pin 1 (the one with a red wire) and has to be connected to one of the channel at all time.

1. Set up and connect the equipment as on figure A.16.
2. Turn power on and adjust the oscilloscope until the signal is visible.
3. Observe and note which channel correspond to Thro(ttle), Aile(ron), Elev(ation), Rudd(er), Gear and Aux on the R620 receiver.
4. Measure and note pulse width for all channel, in the two extreme positions.
5. Observe and note which pulse width for a channel that correspond to sink, climb, rotate left/right, moving backward/forward and moving left/right.
6. Observe and note which pulse width for each flight mode.
7. Measure and note the amplitude
8. RC transmitter is shut off by switch on the rc transmitter
9. High signal width is measured on all six channels
10. Low signal width is measured on channel 6 with the rc switch forcing the high signal width to a minimum.

Measurement result:

Channels observation and measurement:

- Channel 1: Throttle (1080 μs to 1920 μs)
- Channel 2: Aileron (1080 μs to 1920 μs)
- Channel 3: Elevation (1080 μs to 1920 μs)
- Channel 4: Rudder (1080 μs to 1920 μs)
- Channel 5: Gear (1080 μs to 1920 μs)
- Channel 6: Aux (1080 μs to 1920 μs)
- Center value for channel 1-4 and 6 (1500 μs)
- Channel 6: Low width (20,9 ms)

For RC transmitter shut off:

- Channel 1: Throttle (1080 μs)
- Channel 2: Aileron (1500 μs)
- Channel 3: Elevation (1500 μs)
- Channel 4: Rudder (1500 μs)
- Channel 5: Gear (1500 μs)
- Channel 6: Aux (1500 μs)

Channels observed reaction (1080 μs - 1920 μs):

- Channel 1: Decrease altitude - increase altitude
- Channel 2: Rotates right - rotates left
- Channel 3: Moving backward - moving forward
- Channel 4: Moving Right - moving left
- Channel 5: Switch between two positions (bot 1080 μs, top 1920 μs)
- Channel 6: Switch between three positions, bot (1080 μs), mid (1500 μs) and top (1920 μs).

Amplitude measurement.

- Amplitude 3,4 V.

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
It can be concluded that structure, of the PWM signal from the RC receiver, was determined for each channel. Also the relationship between the RC transmitter and the reaction of quadcopter was determined.

A.10 Measurement report - QAM RX2/TX1 Radio Modules

Purpose:
The Purpose of this test is to examine the QAM-RX2 and QAM-TX1 modules behaviour to prove the necessity of a sync signal.

Theory:
By connecting the two radio modules as shown in the test set up and sending the ping signal with and without the sync signal, their behavior can bee shown.

Equipment list:

1. Oscilloscope (AAU 33858)
2. QAM -RX2 and QAM-TX1 modules
3. 2x 17cm long wires, to act as antennas
4. Arduino Leonardo
5. Camera

Measurement set up:

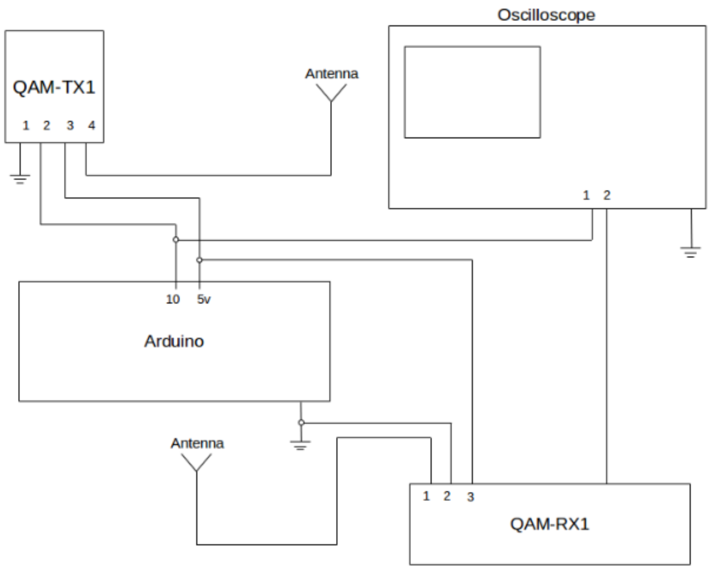


Figure A.17: Setup for the radio module Sync test.

Procedure:

1. Connect the modules as shown on figure A.17.
2. Upload the CD/Measurement Reports/Measurement Report - QAM RX2 TX1 Radio Modules/No_Sync sketch to the arduino.
3. Pause the oscilloscope and measure the sync and ping signal being transmitted.
4. Pause the oscilloscope, measure and compare this to the transmitted signal, document this with a picture.
5. Upload the CD/Measurement Reports/Measurement Report - QAM RX2 TX1 Radio Modules/Send_Sync sketch to the arduino.

6. Pause the oscilloscope and measure the sync and ping signal being transmitted.
7. Pause the oscilloscope, measure and compare this to the transmitted signal, document this with a picture.
8. Repeat from step 5 without the antenna.

Measurement data:

Whithout Sync signal

Transmited	Uptime	Downtime	Recieved	Uptime	Recieved
Ping	428μs	168μs	Ping	N/A	N/A
Period	32,6ms		Period	32,6ms	

Table A.4: Table of radio module measurements without a sync signal

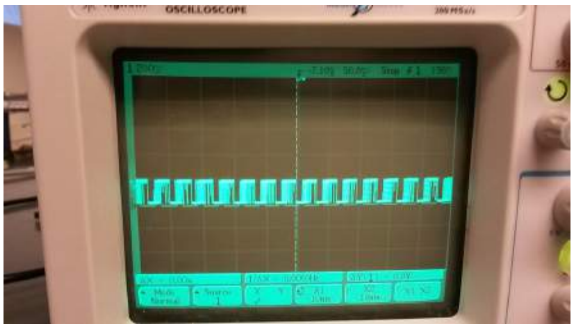


Figure A.18: Recieved Signal Without Sync

With Sync signal

Table A.5: Table of radiomodule measurements with a sync signal

Transmitted	Uptime	Downtime	recieved	Uptime	Downtime
Sync	254 μs	168 μs	**Sync**	228 μs	228 μs
Ping	428 μs	168 μs	**Ping**	388 μs	228 μs
Period	33,4 ms		**Period**	33,4 ms	

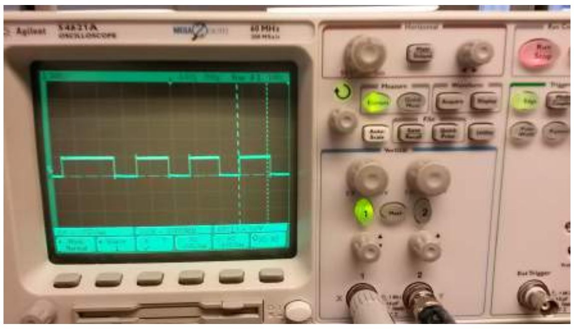


Figure A.19: Recieved Signal With Sync

Whith Sync signal Whtihout antenna

send	Uptime	Downtime	recv	Uptime	Downtime
Sync	252 μs	168 μs	**Sync**	228 μs	228 μs
Ping	428 μs	168 μs	**Ping**	366 μs	228 μs

Table A.6: Table if radio module measurements with a sync signal but without antenna

Result:
It can be seen from the measurement data that the signal will be influence by noise if the module do not use a sync signal.

Error Sources:

- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
From the results it is shown that if no sync signal is send, it will influence the signal. It is however more unclear how the antenna affects the signals. Some distortion is present, however it is unclear how this affects the signal as only parts of the signal have changed.

A.11 Measurement report - Radio and ultrasound test

Purpose:
The purpose of this measurement report is to examine the distance system to see if it is able to fulfill the requirements 6 and 7 about calculating distance from 1 m to 4 m and with a tolerance of ± 5 cm. Furthermore it should be able to detect angles from ± 30 °. This test is made from section 4.4.

Theory:
By measuring the output of the distance sensor at known positions, the stationary precision can be found.

Equipment list:

- Arduino Due
- Transmitter board.
- Receiver board.
- Tape measure.
- 12 · jumper wires

Measurement set up

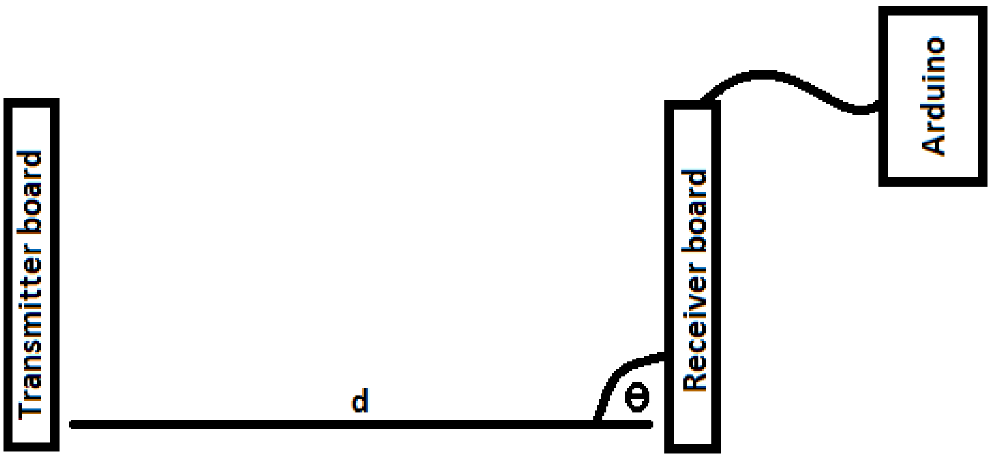


Figure A.20: Setup for the distance test.

Procedure:

- Measure distances from 1 m to 4 m.
  - Mark at 1 m and for every 10 cm to 4 m.
- Measure angles from 0 to 30°.
  - Mark at ± 0°, 5°, 10°, 15°, 20°, 25° and 30°
- Upload the arduino code to the arduino(refer to CD(/CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/SensorFinalTest)).
- Connect the FPGA on receiver board to the Arduino with the 12 jumper wires.
- Upload the transmitter code to the arduino on the transmitter board(refer to CD(CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/SyncSignalLowSensor)).
- Upload the VHDL code to the FPGA on the receiver board(refer to CD(CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/DistanceSingleDelay)).

- Turn on all the boards, and place the Transmitter and Receiver in the desired position.
- Read the value in the serial port on the arduino due and note the number/distance or see if the angle is detected.
- Move the receiver to a new distance or angle and retake the step before.

Measurement data:

Distance [mm]	**Measured distance [mm]**	**Timeout?**	**Reflections?**
1000	1007	No	No
1100	1077	No	No
1200	1173	No	No
1300	1274	No	No
1400	1376	No	No
1500	1467	No	No
1600	1565	No	No
1700	1665	No	No
1800	1762	No	No
1900	1858	No	No
2000	1935	No	No
2100	2060	No	No
2200	2159	No	No
2300	2249	No	No
2400	2347	No	No
2500	2460	Yes	No
2600	2540	Yes	No
2700	2624	Yes	No
2800	2750	No	Yes
2900	2832	No	No
3000	2937	Yes	Yes
3100	3030	Yes	Yes
3200	3110	Yes	No
3300	3215	No	No
3400	3311	Yes	No
3500	3401	Yes	No
3600	3504	Yes	No
3700	3596	Yes	No
3800	3700	Yes	No
3900	3774	Yes	No
4000	3871	Yes	No

Table A.7: Result from 40 kHz test.

Distance [mm]	**Measured distance [mm]**	**Timeout?**	**Reflections?**
1000	1152	Yes	Yes
1100	1274	Yes	Yes
1200	1382	Yes	Yes
1300	1489	Yes	Yes
1400	1621	Yes	No
1500	1666	Yes	Yes
1600	1755	No	No
1700	1663	No	No
1800	1951	Yes	Yes
1900	2062	Yes	No
2000	2214	Yes	No
2100	2259	Yes	Yes
2200	2361	Yes	Yes
2300	2463	Yes	Yes
2400	2598	Yes	Yes
2500	2690	Yes	No
2600	2792	Yes	Yes
2700	2874	Yes	Yes
2800	2996	Yes	Yes
2900	3072	Yes	Yes
3000	3353	Yes	Yes
3100	3248	Yes	No
3200	3453	Yes	No
3300	3376	Yes	Yes
3400	3638	Yes	No
3500	3722	Yes	No
3600	-	Yes	-
3700	3961	Yes	No
3800	-	Yes	-
3900			
4000			

Table A.8: Result from 25 kHz test.

± Angle [°]	**Detected**
0	Yes
5	Yes
10	Yes
15	Yes
20	Yes
25	Yes
30	Yes

Table A.9: Result from the angle test for both sensors.

Results:
The result from the 40 kHz test shows that nearly every reading was lower than the distance only the first reading was higher than the distance. It has a deviation from 7 mm up to

12,9 cm. On the figure for 40 kHz A.21 it is shown that the measured distance increases linearly over distance. In the test there were some reflections which are seen in the table A.7. The system for 40 kHz started to timeout after 240 cm, which means that it missed some readings. For the second test with 25 kHz the readings were both higher and lower than the distance. The deviation from the distance was up to 35,3 cm at 300 cm and for the distances above 350 cm the system was only able to make one reading at 370 cm and it deviated with 26,1 cm. As seen on figure A.21 for 25 kHz it is not as linear as the one for 40 kHz. For this test there were more reflections which are seen in the table A.8. The system for the 25 kHz also timeout on nearly every reading.

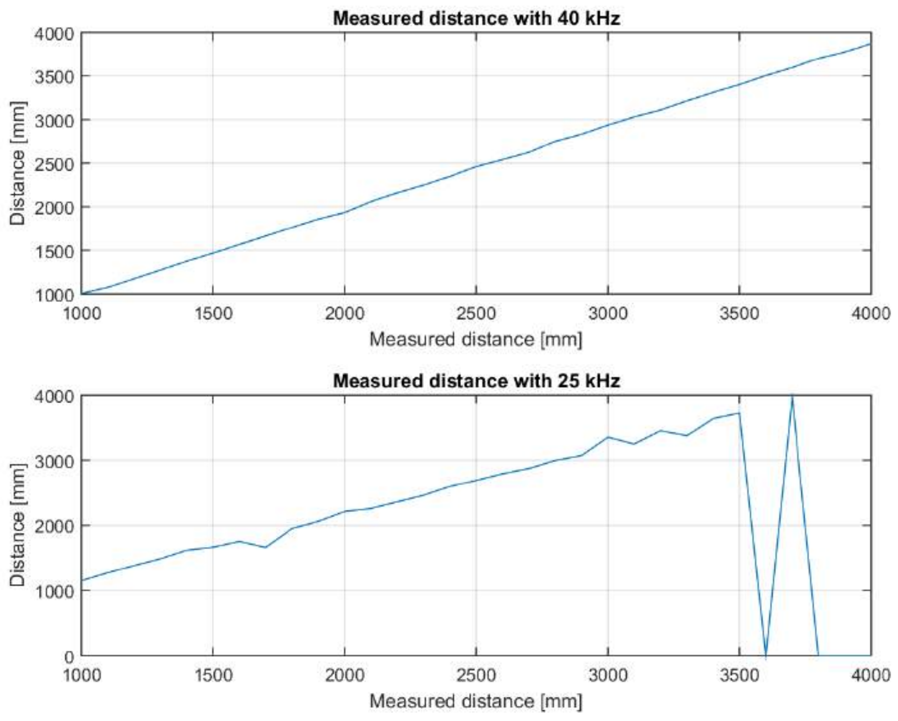


Figure A.21: Plot for both measured distances.

Both 25 and 40 kHz was able to detect from ± 0 ° to 30 °.

Source of error:

- Accuracy of placing the transmitter ± 1 cm.
- Reading accuracy on the serial port on the arduino.
- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
The system is able to detect angles from ± 0° to 30° which means that the requirement 7 is partly fulfilled, because the test was not made as a dynamic movement but as a static test. The system needs to be tested in motion before requirement 7 can be fulfilled. The distance system has a deviation from 0,7 cm up to 12,9 cm for 40 kHz and for 25 kHz it has a deviation from 3,7 cm up to 35,3 cm. The requirements for both distance in not fulfilled because they deviated with more than ± 10 cm at some distances. At distances lower than 250 cm, the 40 kHz deviated with less than 5 cm and the 25 kHz deviated with more the 15 cm on nearly every readings. The 40 kHz has a good accuracy and it is reliable because

the readings are linear as seen on figure A.21. It also fulfills the requirements for distance up to 360 cm. The 25 kHz has a bad accuracy and it is not reliable which is also seen in the figure A.21.

A.12 Measurement report - Radio and ultrasound second test

Purpose:
The purpose of this measurement report is to examine the distance system to see if it is able to fulfill the requirement 6 about calculating distance from 1 m to 4 m and with a tolerance of ± 5 cm.

Theory:
By measuring the output of the distance sensor at known positions, the stationary precision can be found.
Equipment list:

- Arduino Due
- Transmitter board.
- Receiver board.
- Tape measure.
- 12 · jumper wires

Measurement set up

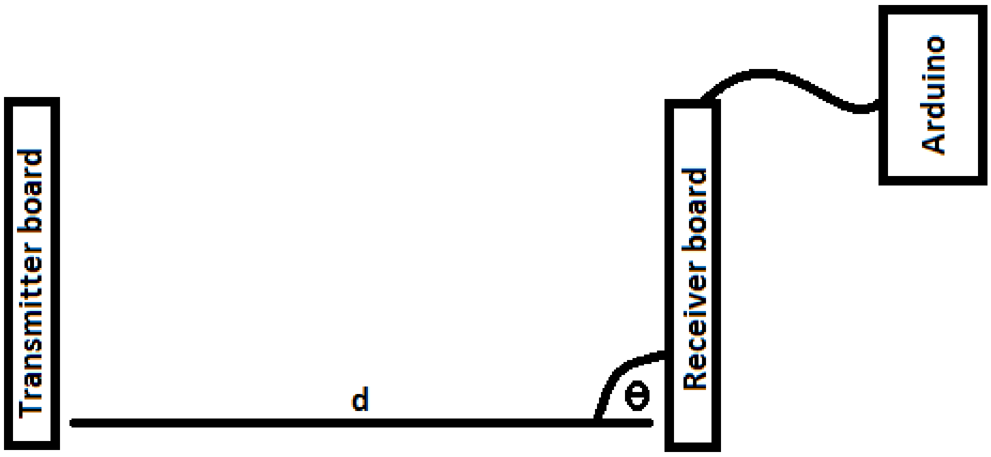


Figure A.22: Setup for the distance test.

Procedure:

- Measure distances from 1 to 4 m.
  - Mark at 1 m, 1,5 m, 2 m, 2,5 m, 3 m, 3,5 m and 4 m.
- Upload the arduino code to the arduino(refer to CD(/CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/SensorFinalTest)).
- Connect the FPGA on receiver board to the Arduino with the 12 jumper wires.
- Upload the transmitter code to the arduino on the transmitter board(refer to CD(/CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/SyncSignalLowSensor)).
- Upload the VHDL code to the FPGA on the receiver board(refer to CD(/CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/DistanceSingleDelay)).

- Turn on all the boards, and place the Transmitter and Receiver in the desired position.
- Read the value in the serial port on the arduino due and note the number/distance.
- Move the receiver to a new distance and retake the step before.

Measurement data:

Distance[mm]	**Measured distance[mm]**	**Timeout?**	**Reflection?**
1000	1013	No	No
1500	1510	No	No
2000	1980	No	No
2500	2456	No	No
3000	2954	No	No
3500	3455	No	No
4000	3956	No	No

Table A.10: Result from 40 kHz test.

Distance[mm]	**Measured distance[mm]**	**Timeout?**	**Reflection?**
1000	989	No	No
1500	1486	No	No
2000	2002	No	No
2500	2480	No	No
3000	2973	No	No
3500	3464	No	No
4000	3950	No	No

Table A.11: Result from 25 kHz test.

Results:
The result for both 40 and 25 kHz is very close to the distance upto 2000 mm. After 2000 mm it start to vary with upto 50 mm. On figure A.23 it can be seen that the measured distance on both receivers increases linear over the distance.

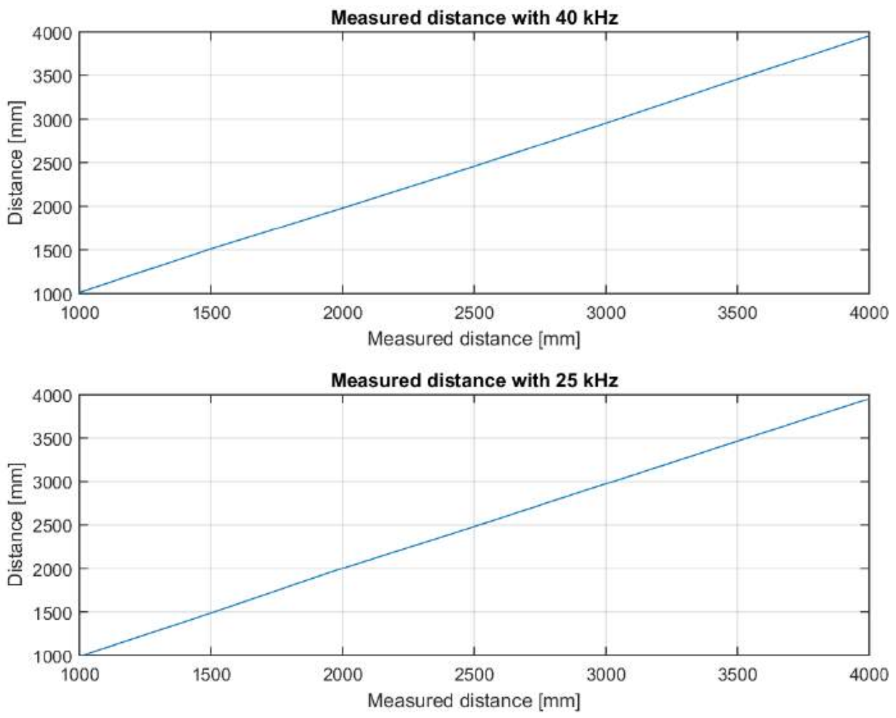


Figure A.23: Plot for both measured distances.

Source of error:

- Accuracy of placing the transmitter ± 1 cm.
- Reading accuracy on the serial port on the arduino.
- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
The measured distances do not deviate with more them ± 50 mm therefore the system fulfill the requirement when the system is not in motion.

A.13 Measurement report - Radio and ultrasound motor test

Purpose:
The purpose of this measurement report is to examine the distance system to see if it is able to fulfill the requirement 6 about calculating distance from 1 m to 4 m and with a tolerance of ± 5 cm while the motors is on, to see if they have any effects on the distance measurements. This test is made from section 4.4.

Theory:
By measuring the output of the distance sensor at known positions, the stationary precision can be found. As the motors produce a disturbance, these will be done to produce as close an enviroment to the flight as possible.

Equipment list:

- Arduino Due
- Transmitter board.
- Receiver board.
- Tape measure.
- 12 · jumper wires
- Quadcopter

Measurement set up

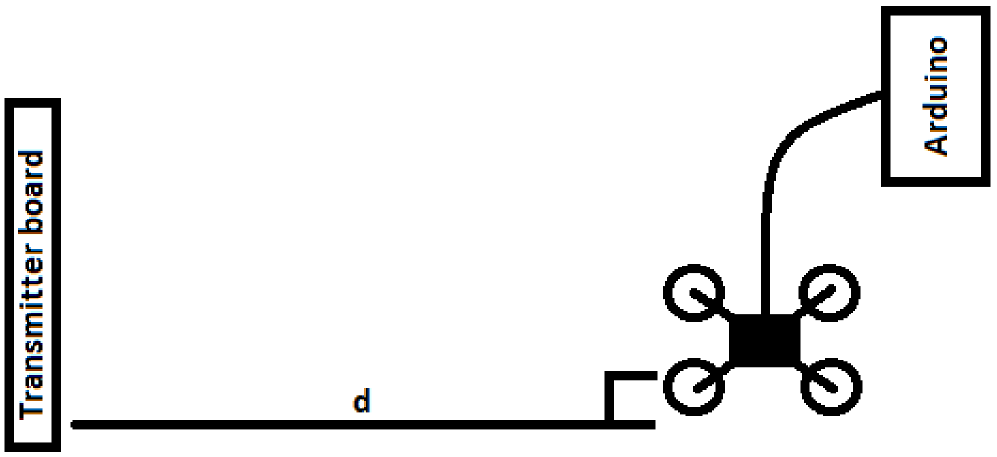


Figure A.24: Setup for the distance test.

Procedure:

- Measure distances from 1 m to 4 m.
  - Mark at 1 m, 1,5 m, 2 m, 2,5 m, 3 m, 3,5 m and 4 m.
- Upload the arduino code to the arduino(refer to CD(/CD/Measurement Reports/Measurement Report - Radio and ultrasound second test/SensorFinalTest)).
- Connect the FPGA on receiver board to the Arduino with the 12 jumper wires.
- Upload the transmitter code to the arduino on the transmitter board(refer to CD(CD/Measurement Reports/Measurement Report - Radio and ultrasound with motors/SyncSignalLowSensor)).

- Upload the VHDL code to the FPGA on the receiver board(refer to CD(CD/Measurement Reports/Measurement Report - Radio and ultrasound with motors/DistanceSingleDelay)).
- Tape the quadcupter to the ground, so it is not able to fly.
- Turn on all the boards, and place the Transmitter and Receiver in the desired position.
- Turn the motors on the quadcopter on.
- Read the value in the serial port on the Arduino due and note the number/distance.
- Stop the motors.
- Move the receiver to a new distance and start the motors again and retake the steps.

Measurement data:

Distance[mm]	Measured distance[mm]	Timeout	Failed mearsurment
1000	995	No	No
1500	1491	No	No
2000	1990	No	No
2500	2480	No	No
3000	2958	No	No
3500	3450	No	No
4000	3980	Yes	No

Table A.12: Result from 40 kHz test.

Distance[mm]	Measured distance[mm]	Timeout	Failed mearsurment
1000	1006	No	No
1500	1495	No	No
2000	1990	No	No
2500	2488	No	No
3000	2980	No	No
3500	3480	No	No
4000	3987	Yes	Yes

Table A.13: Result from 25 kHz test.

Results:
Both readings started to timeout at 4000 mm and for 25 kHz it started to show wrong measurments. It can be seen on figure A.25 that both measurments increase linearly over the distance.

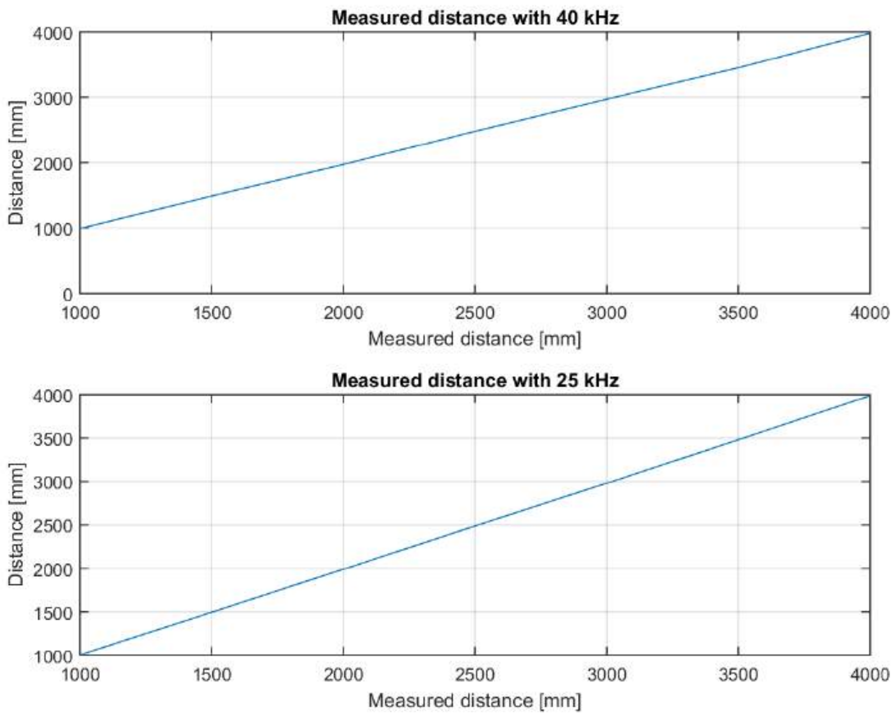


Figure A.25: Plot over the measured distance.

Source of error:

- Accuracy of placing the receiver ± 1 cm.
- Reading accuracy on the serial port on the arduino.
- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
It can be concluded that the distance system works while the motors is on, and therefore it still fulfill the requirement 6 when the quadcopter is not in motion.

A.14 Measurement report - Sensor Module Delay

The purpose of these tests is to find the total delay introduced into the system by the radio module and the ultrasound module.

Radio Module Delay

Purpose:
The purpose of this measurement is to find the delay introduced by the QAM-TX1 and QAM-RX2 modules, in order to account for that in the final system, thus increasing the precision of the sensor

Theory:
By measuring the time from the data is send to the data is received, the total delay can be found.

Equipment list:

1. Arduino Leonardo
2. Ocilloscope (AAU 33585)
3. QAM-TX1 and QAM-RX2

Measurement set up:

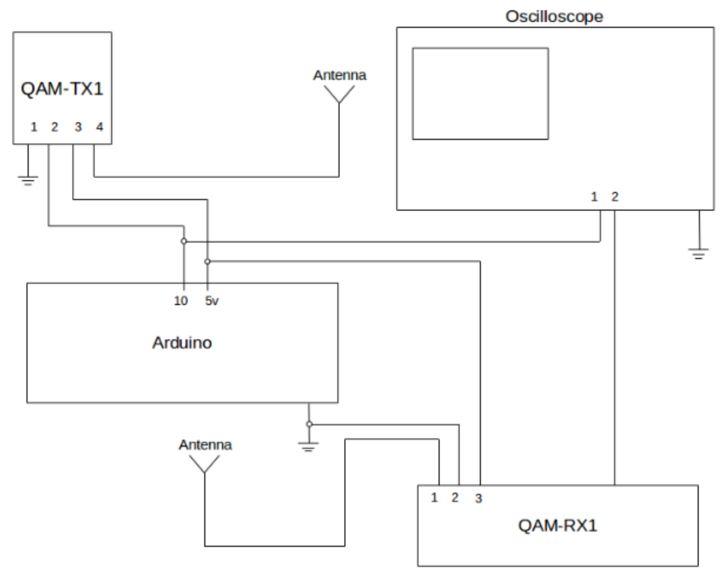


Figure A.26: Setup for the radio module test

Procedure:

1. Connect the QAM Modules to the Arduino and the oscilloscope according to Figure A.26 1 m apart.
2. Upload The CD/Measurement Reports/Measurement Report - Radio and ultrasound/Send_Sync sketch onto the Arduino.
3. Locate The ping wave on the oscilloscope for both the RX and TX module.

4. Measure the time Between Rising edge on the Tx Module and The Rx Module, note this as Radio delay.
5. Measure the Width of the Ping signal on the RX Module, Note this as Radio Module delay.

Ultrasound Module Delay

Purpose:
The purpose of this measurement is to find the delay introduced by the Ultrasound module, in order to account for that in the final system, thus increasing the precision of the sensor

Equipment list:

1. Thermometer
2. Ocilloscope (AAU 33585)
3. Ultrasoinc transmitter and reciever board

Measurement set up

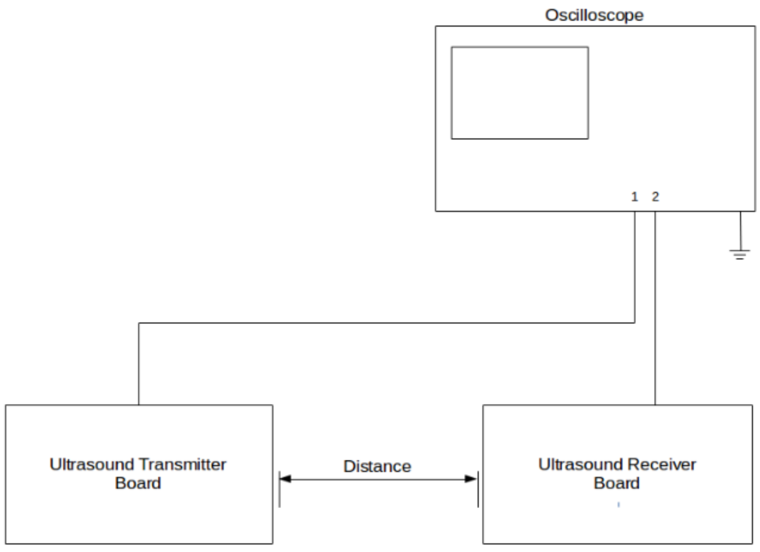


Figure A.27: Setup for the ultrasound module test.

Procedure

1. Place the two boards with the sensors facing each other. Figure A.27 shows a generic setup. Note that distance must not exeed 4 m.
2. Connect the oscilloscope to the rightmost BNC input on the transmitter board, and to signal pin R15 on the FPGA if 40 kHz is to be measured.
3. To avoid acciedental feedback place the two modules with a distance of around 70 cm apart, note this distance.
4. Measure the Distance between the two modules.
5. Measure the Temperature.
6. Upload CD/Measurement Reports/Measurement Report - Radio and ultrasound/Send_Sync sketch to the Arduino.

7. Measure the time difference between the start of the signal on the transmitter and reciever modules.
8. Calculate the delay introduced by the distance, substract this from the total time difference to find the delay introduced by the ultrasound module.
9. Repeat above steps with the oscilloscope connected to the leftmost BNC on the transmitter board and in R2 on the FPGA to find delays for 25 kHz.

Measurement data:

Module	Time delay [μs]
Radio hardware	84
Radio module	420
40 kHz Ultrasound	2390
25 kHz Ultrasound	2600

Table A.14: Table of Measured Delays.

Results:

Following equation have been used to find the distance each delay generates:

$$D_{Offset} = 343\frac{m}{s} \cdot t_{delay} \tag{A.41}$$

Where:
$343\frac{m}{s}$ is the speed of sound at 20°C. $[\frac{\mathrm{m}}{\mathrm{s}}]$
t_{delay} is the measured delay. [s]

Furthermore the delay generated by the distance between the two ultrasound modules can be found by:

$$t_{offset} = \frac{Dist_{Modules}}{343\frac{m}{s}} \tag{A.42}$$

Where:
$343\frac{m}{s}$ is the speed of sound at 20°C. $[\frac{\mathrm{m}}{\mathrm{s}}]$
t_{offset} is the delay introduced by the ultrasound wave. [s]

By substracting this from the measured delay in the ultrasound module, the delay generated by the circuit can be found. After above calculations the test gave the following results:

Module	Time delay [μs]	Distance [mm]
Antenna	84	28,81
Antenna Module	420	144,1
40 kHz Ultrasound	-100,1	-34,5
Sum	404,9	108,41

Table A.15: Table of Measured Delays and calculated distances for 40 kHz.

Module	Time delay [μs]	Distance [mm]
Antenna	84	28,81
Antenna Module	420	144,1
25 kHz Ultrasound	-310,1	-107
Sum	194,9	35,91

Table A.16: Table of Measured Delays and calculated distances for 25 kHz.

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement reading inaccuracies.
- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
From the results obtained in this test is can be concluded that the offsets measured in the test needs to be implemented to get the correct distance as output.

A.15 Measurement report - Step response

Purpose:
The purpose of this measurement report is to determine the step response for pitch, roll and yaw for the flight control system.

Theory:
A signal from the RC transmitter trigger a unit step to be sent from the FPGA to the Pixhawk, this will make the quadcopter move either forward, sideway or make it rotate. The distance over time is measured using the Vicon system in MTLab.

Measurement set up:

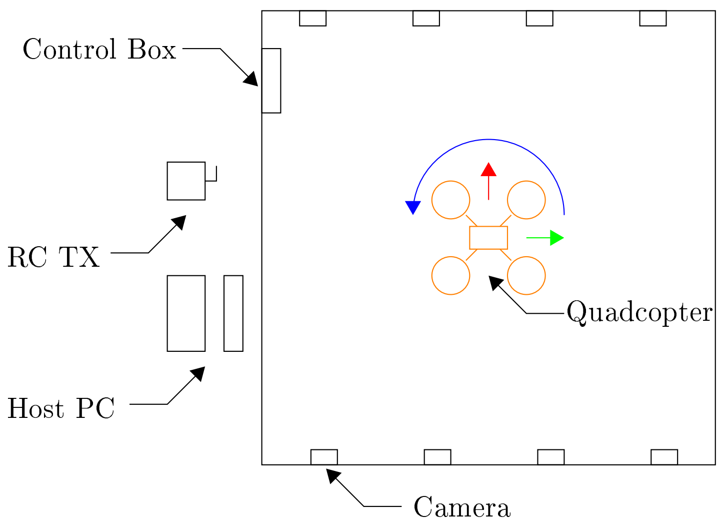


Figure A.28: Illustration of measurement set up, where the quadcopter (orange), step responses for pitch (green), roll (red) and yaw (blue) is measured with Vicon system.

Equipment list:

- Vicon MX (AAU 75459)
- Host PC (AAU 86703)
- Quadcopter
- OrangeRx R620 receiver
- OrangeRx T-SIX transmitter

Procedure:
This test requires two person, Person 1 control the RC transmitter and person 2 the host pc.

1. When entering the lab remember to lock the door and turn on the flight indication light.
2. Turn on power to the Vicon control box and the host computer.
3. Start the tracker program and select the "gr512quad" model under the object tab.
4. Start MatLab and select the /CD/Measurement Reports/Measurement Report - Step response/ViconDataStreamSDK_MATLABTest script.
5. Program the FPGA with /CD/Measurement Reports/Measurement Report - Step response/PPM_Delay and restart it.
6. Place the quadcopter as seen on figure A.28 and check it show up on on the host PC.

7. Connect the battery on the quadcopter and arm it.
8. Scroll down in the script and insert test file name.
9. Start the quadcopter and the MatLab script.
10. Person 1 will hold and maintain a position, close to a wall, to ensure most possible distance.
11. When person 1 say "start", person 2 press enter on the host PC and say "start".
12. Person 1 flip the channel 5 switch.
13. Person 1 again flip channel 5 and stabilise the quadcopter before it hit the opposite wall.
14. Person 2 press enter twice to stop the script and save the workspace.
15. Person 1 lands and shut down the Quad.

Measurement data:

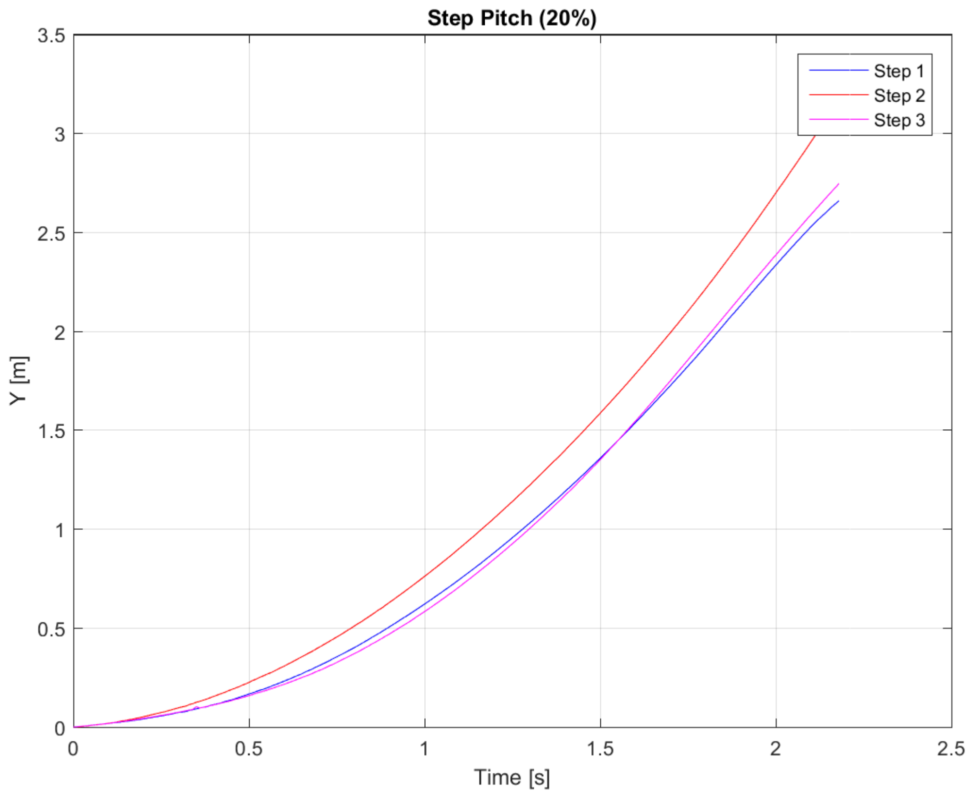


Figure A.29: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

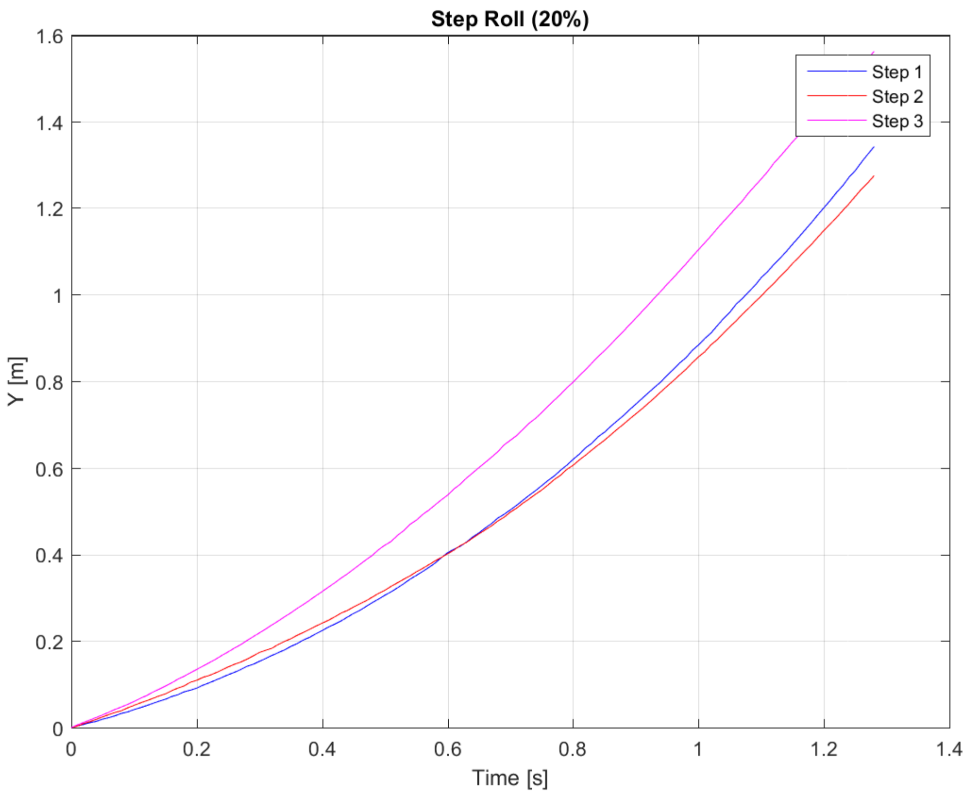


Figure A.30: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

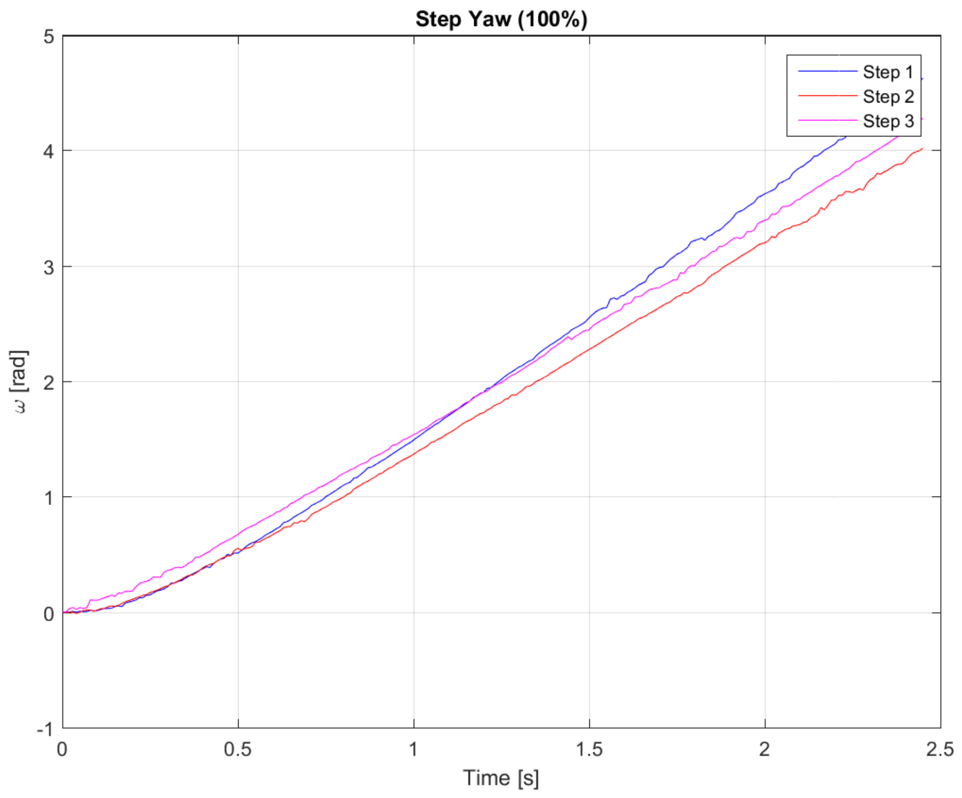


Figure A.31: Graph of the change in distance over time for several step responses. The measured data has correction to the start point, length and errors.

Result:
The step response for pitch can be seen on figure A.29, it shows the response from three separate flights at 20 percent step input. The quadcopter starts from a hovering position, receive a step input and moves until a steady state is achieved. It can be seen that Step 2 moves longer over time than Step 1 and 3, but does not achieve a steady state. A possible reason could be the quadcopter might not have started from a complete hovering position and as the lab is relative small, the quadcopter could not travel long enough to achieve steady state. Step 1 and step 2 have similarly curves and seems to have achieved steady state, but this is likely because the quadcopter no longer have the step input.

The step response for roll can be seen on figure A.30, it shows the response from three separate flights at 20 percent step input. Same method as with pitch is use to get the step response. Step 3 travels longest over time but also clearly starts to slows down before steady state was achieved, which probably mean that at the starting point it was not hovering completely and the step input was turned off after relative early. Step 1 and Step 2 are similar but neither reach steady state and step 2 starts to separate from step 1 after one second, which could indicate the quadcopter might have been drifting.

The step response for yawroll can be seen on figure A.31, it shows the response from three separate flights at 100 percent step input. Where the quadcopter as with pitch and roll starts from a hovering position, receive a step input and start turning around the z-axes until a steady state is achieved. Step 1 is measured in altitude hold flight mode instead of stabilized mode, which does have an effect over time compared to Step 2 and Step 3, but does follow Step 2 for half a second. Step 2 and step 3 have similar curves, a possible reason for the separation between them could be due to measurement error from Vicon. All of the steps seams to have reach a steady state after one second.

Error sources:
The following reason can cause errors and discrepancies in the results.

- Measurement equipment tolerance and delays.
- Stability of Quad.
- Human factor.
- Noise from the equipment.
- Ambient noise.

Measurement equipment tolerance and delays:
The Vicon system did have a relative large delay from start of the matlab script to it actually start saves the stream, which meant the correct start point would not be saved from the stream, requiring the steam to started prior to the starting point. Another problem occurs where the Vicon system lost track of the quadcopter, causing data with errors, when it was relative close to the wall of the lab. For reliable tracking the quadcopter had to be position closer to center, reducing the maximum distance it could travel.

Stability of Quad:
There was stability issue with the quadcopter, causing it possible drift in a direction, that meant the quadcopter could already be moving before the step was sent. Furthermore could it give faulty data, if the quadcopter drifted at an angle, as the measured distance would be lower.

Human factor:
To rectify problem with the Vicon system and the stability of the quadcopter, human would decide when stability is achieved and when the step started.

Conclusion:
It can be concluded that step response for pitch, roll and yaw was achieved, but that error source, especially from the stability issue and the human factor would make repeatable measurement difficult.

A.16 Measurement report - Ultrasound

Purpose:
The purpose of this measurement report is to investigate the output voltage levels from different distances (from 0.2 m to 4 m) with the ultrasound sensors MA40S4R/S [muRata, 2002] and 250S(T/R)180 [Farnell, 2013].

Theory:
The theory of this measurement is to apply a 40kHz signal to the ultrasound transmitter, to examine what output it generates on the 40 kHz receiver. This will be done by a tone generator to ensure a controllable and repeatable environment.

Materials:

- 2 · Equipment to hold the sensors.
- Oscilloscope (AAU 33858).
- Wave generator (AAU 08591).
- Tape measure.

Measurement set up:

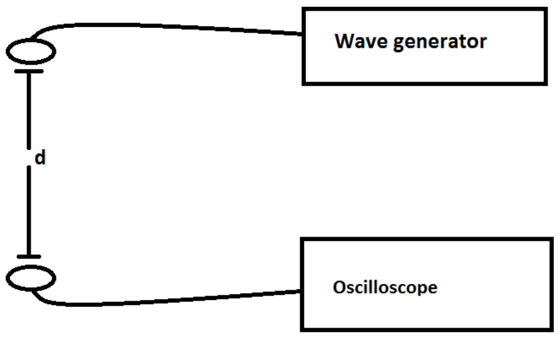


Figure A.32: Setup for the ultrasound test.

Procedure:

1. Setup the test chircuit, as shown in data sheet for 40 kHz [muRata, 2002].
2. Measuring distances from 0,2 m to 4 m.
   - Mark at 20 cm, 50 cm, 1 m, 1.5 m, 2 m, 2.5 m, 3 m, 3.5 m and 4 m.
3. Set the wave generator at 40 kHz.
4. Set the wave generator to the ultrasound transmitter and set the receiver to the oscilloscope.
5. Place the receiver and transmitter facing each other.
6. Before starting the wave generator check the oscilloscope for any disturbance from the receiver.
7. Start the wave generator and set the voltage to be 20 Vpp.
8. Set the receiver at 20 cm.
9. Read the oscilloscope and note the voltage level from the receiver with the distance.
10. Move the receiver to the next step and repeat the process from step 9.
11. Use the same approach with both sets of sensors, remember to change the frequency to fit the transmitter.

Measurement data:
Noise from the receiver is 9 mV for both tests.

Distance [cm]	Voltage level [mV]
20	240
50	87
100	45
150	35
200	24
250	23
300	15
350	17
400	13

Table A.17: Measurement data for 40 kHz

Distance [cm]	Voltage level [mV]
20	725
50	303
100	132
150	97
200	62
250	58
300	50
350	55
400	38

Table A.18: Measurement data for 25 kHz

Results:
The results show that the farther away the lower the voltage. It can be seen for the 40 kHz that it comes close to the noise at 4 m, where 25 kHz still have a high enough signal to distinguish it from the noise as seen on figures A.33.

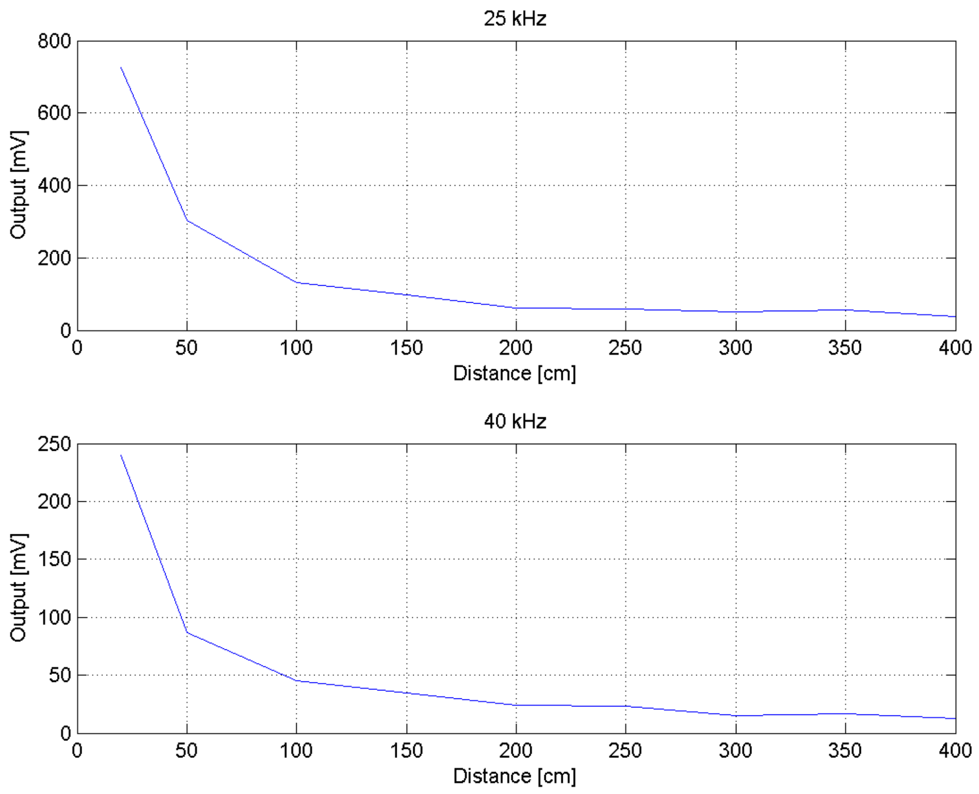


Figure A.33: Outputvoltage over distance for 25 kHz and 40 kHz.

Source of error:

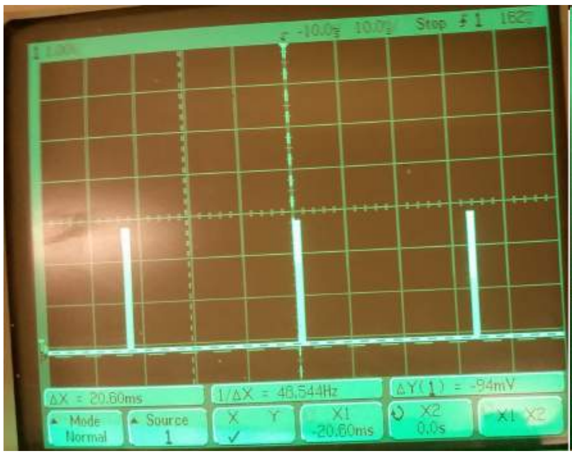
- Measurement equipment tolerance.
- Noise from the equipment.
- Ambient noise.

Conclusion:
It can be concluded that 40 kHz do not have a particularly high signal strength at 4 m because the background noise is 9 mV and the signal is 13 mV, therefore it may be preferable to carry out the tests closer than 4 m, as it will give a bigger difference between noise and signal. The 25 kHz transducers have a strong signal at 4 ms and therefore, it would not be a problem to distinguish the signal from noise.

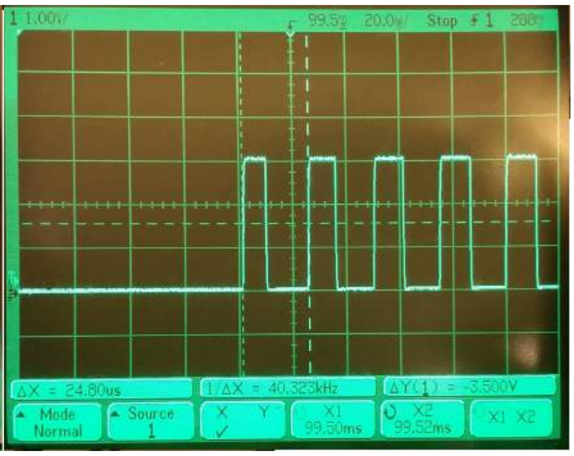
A.17 Noise test for the ultrasound sensors

This section will investigate the ultrasound receivers for noise and how to limit it.

To analyze the input from the ultrasounds receivers an oscilloscope (AAU 56684) is used. The oscilloscope is used on the pins that are connected to the receivers on the FPGA. When the ultrasound receivers receive a signal the input pin on the FPGA should show a signal with the frequency of 40 or 25 kHz which is seen on figures A.34 a and b.



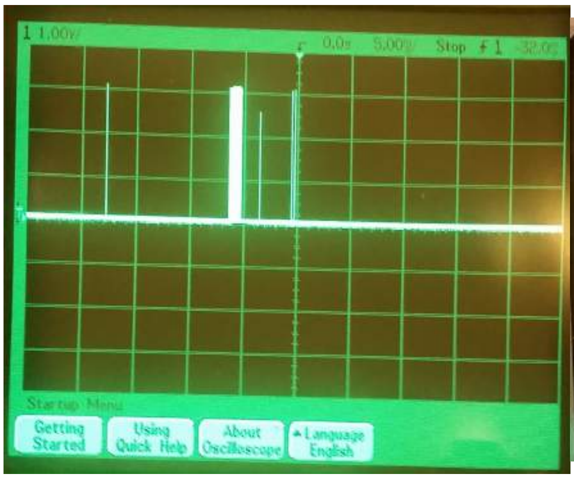
(a)



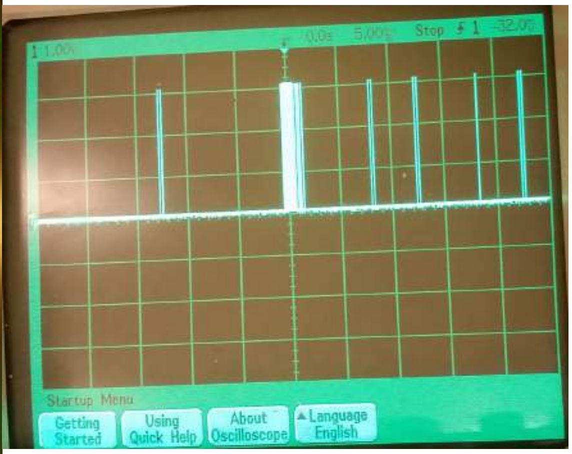
(b)

Figure A.34: (a)Received signal from the ultrasound receiver. (b)40 kHz signal.

These figures were taken while the motors on the quadcopter stood still. On figure A.34 it can be seen that only the signal from the ultrasound receivers is received. Therefore it can be concluded that if the motors are not running there is no noise on the input to the FPGA. Next step were to start the motors to see if they had any influence on the ultrasound receivers. On figures A.35 a and b it can be seen the noise from the motors without a propeller on it.



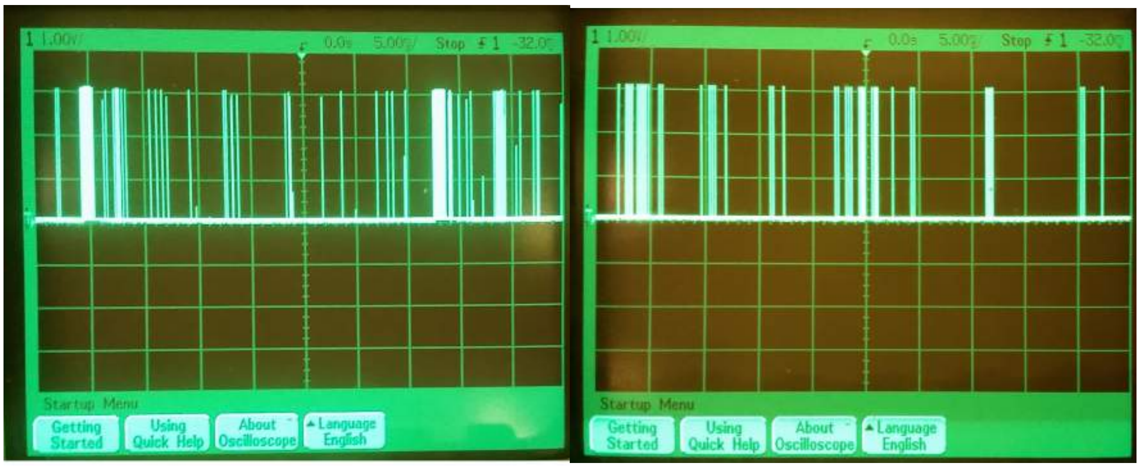
(a)



(b)

Figure A.35: (a)Signal received from the 40 kHz receiver when the motor is on. (b)Signal received from the 25 kHz receiver when motor is on.

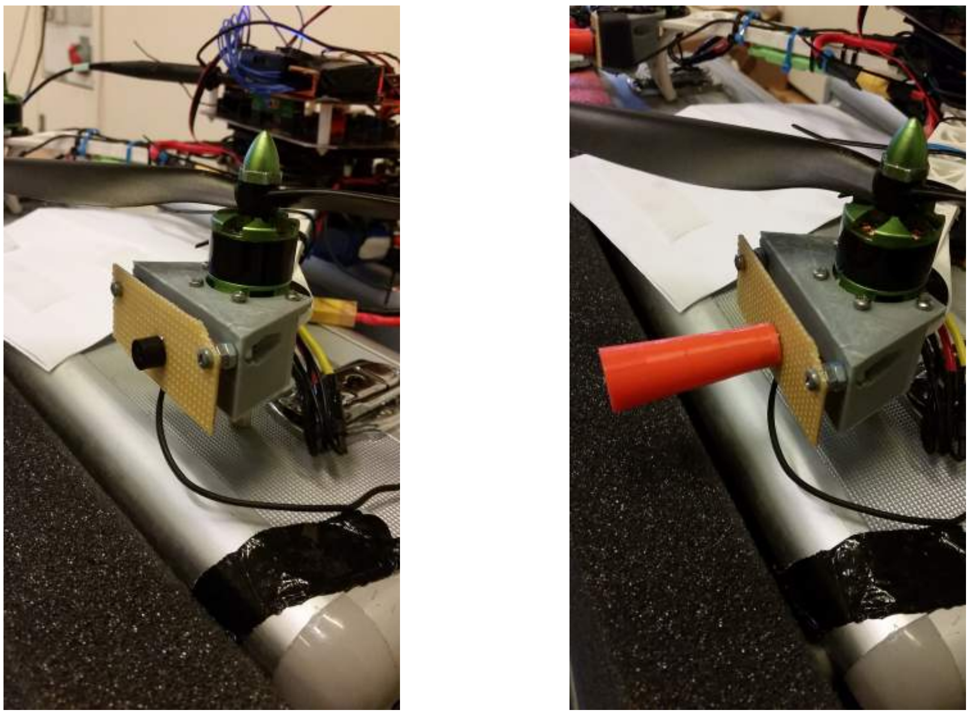
On both figures it can be seen that the FPGA receives more than the signal from the ultrasound receivers. This noise will make the FPGA see it as a signal, and therefore it will make the calculations for the distance measurements incorrect. This noise must be attenuated so it has no influence on the distance calculations. Before the noise is attenuated it will be tested if the propellers add more noise to the readings which can be seen on figures A.36 a and b.



(a) (b)

Figure A.36: (a)Signal received from the 40 kHz receiver with motor and propeller on. (b)Signal received from the 25 kHz receiver with motor and propeller on.

It can be seen that the propellers add more noise to the ultrasound receivers. To begin with the 40 kHz receiver did not have any shielding as seen on figure A.37 a.



(a) (b)

Figure A.37: (a)40 kHz receiver without shielding. (b)40 kHz receiver with shielding.

On figure A.37 b the 40 kHz can be seen with shielding. There was still noise when the propellers were on. To see if the shielding worked the sensor was screwed off the quadcopter and placed below the propeller to see if there were still noise. As seen on figure A.38 the noise was removed.

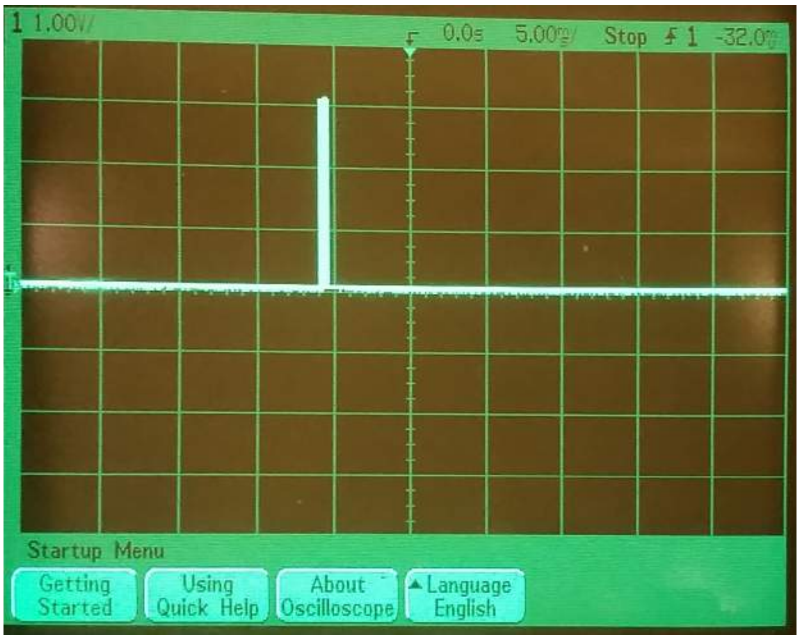
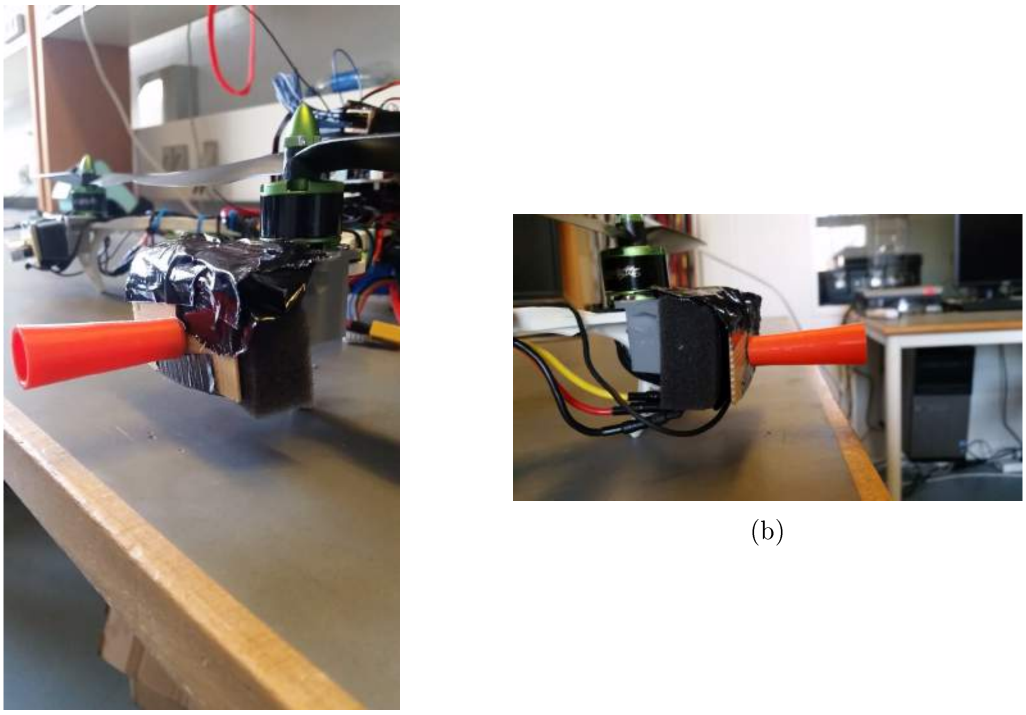


Figure A.38: 40 kHz receiver screwed off.

Therefore it can be concluded that the shielding works. The advantages with the shielding is it will reduce reflection and therefore also reduce the noise. A problem with the shielding is it limit the angle the ultrasound receivers is able to detect the ultrasound waves, therefore a new test for the angle most be conducted. The last noise comes from the vibration from the motor when it is on. As seen on figure A.37 b the ultrasound receiver is screwed on. It was tried to loosen the screws but it did not remove the noise. Therefore it has been investigated other ways to put the ultrasound sensor on the quadcopter. With the use of duct tape and a vibration absorbing material the ultrasound sensor was placed on the quadcopter as seen on figure A.39 a and b.



(a) (b)

Figure A.39: (a)(b)40 kHz receiver final implementation on the quadcopter.

After this implementation the motor was turned on to see if it had any effect on the readings. As seen on figure A.40 it can be seen that the noise is removed.

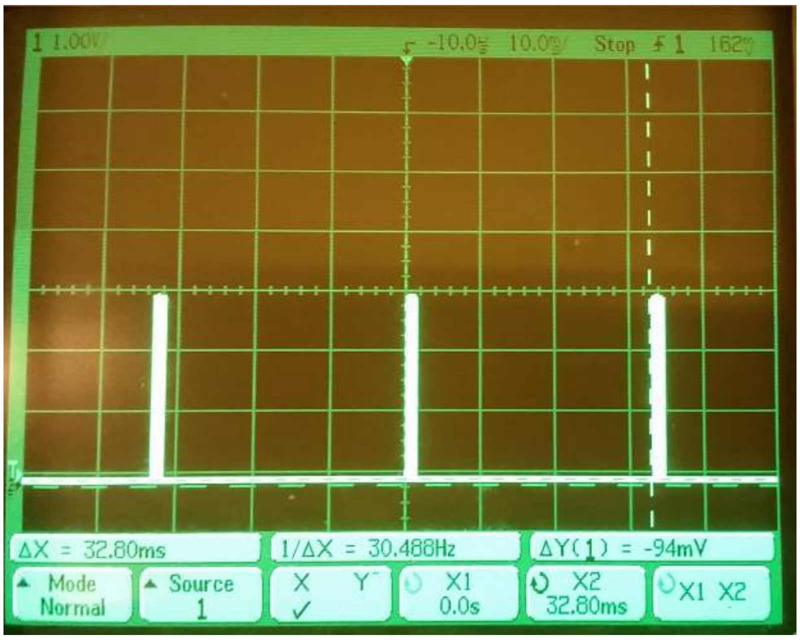


Figure A.40: Reading for 40 kHz.

The same thing was done for the other 40 kHz receiver which is placed on the other wing together with a the 25 kHz ultrasound receiver.
After this implementation for the 40 kHz, the noise for the 25 kHz was almost removed as seen on figure A.41.

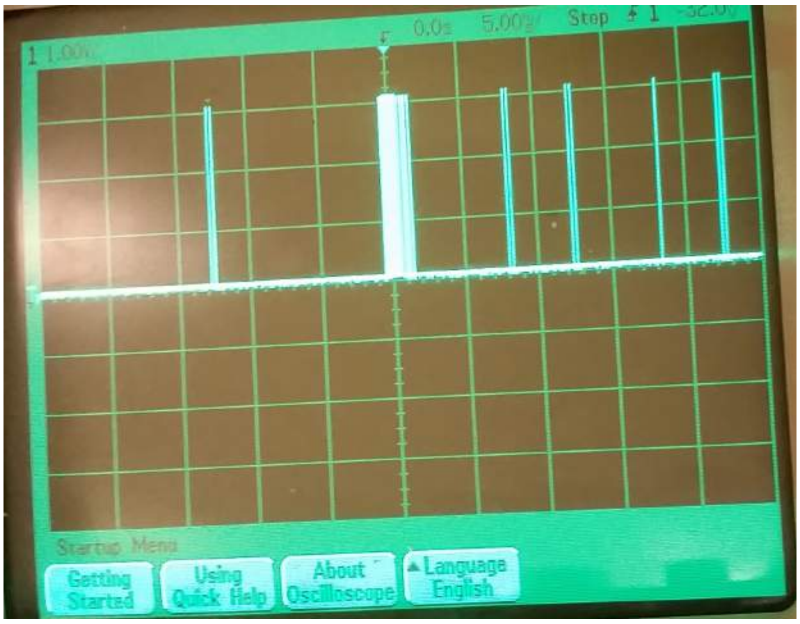


Figure A.41: Noise reduced for 25 kHz.

To reduced the last of the noise the potentiometer on the comparator was increased to 263 Ω which removed the noise as shown on the figure A.42.

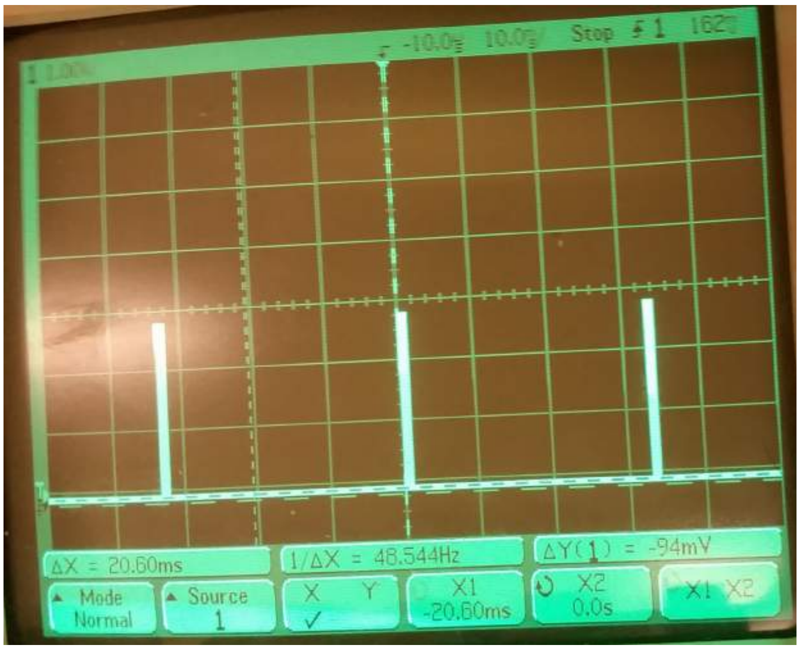
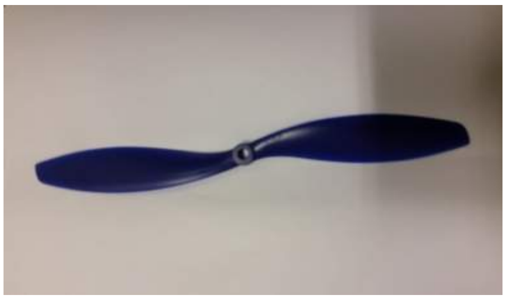


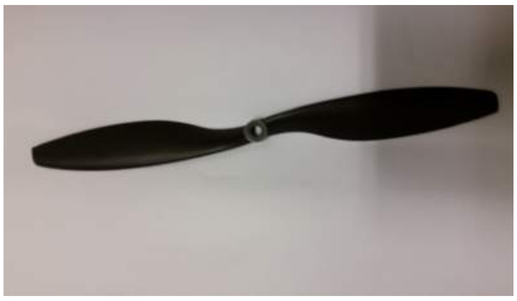
Figure A.42: Noise removed for 25 kHz.

A problem with increasing the resistants of the potentiometer is that it will decrease the distance that the ultrasound receiver would be able to detect the ultrasound waves. But after a test of the 25 kHz ultrasound receiver it was still able to detect the ultrasound waves at 4 m. It was also tried to alter the potentiometer for 40 kHz, but it decreased the distance that the ultrasound receiver is able to receive the ultrasound waves, therefore it was chosen not to alter the potentiometer in the comparator for the 40 kHz sensors. The potentiometer for the 40 kHz is placed at 294 Ω, so the system for 40 kHz is still able to detect the ultrasound waves at 4 m without letting the noise come through.

The test has been done with two different types of propellers as seen on figure A.43.



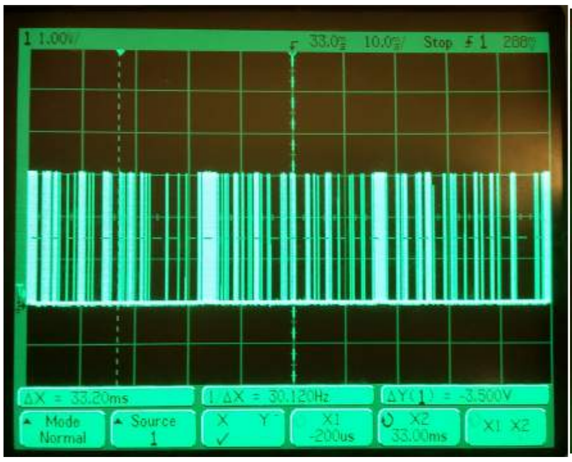
(a)



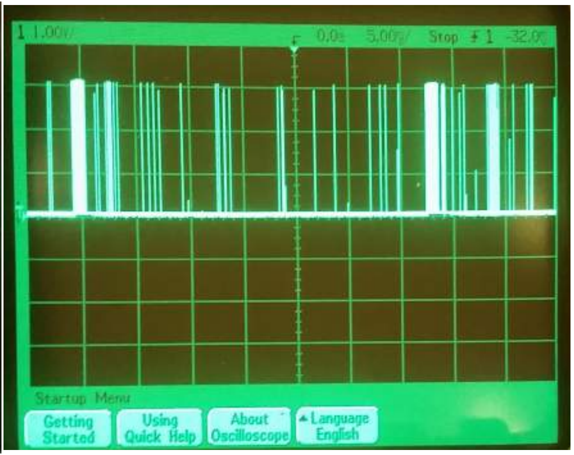
(b)

Figure A.43: (a)Blue propel.(b)Black propel.

The test showed that the blue propels caused more noise in the 40 kHz spectrum as seen on figures A.44 a and b.



(a)



(b)

Figure A.44: (a)Reading with blue propel.(b)Reading with black propel.

For the 25 kHz ultrasound receiver the readings were the same for both propellers therefore, it was chosen to use the black propels because it made less noise for the 40 kHz receivers.

A.18 Bilag

- Altium
  - PCB_P5
  - PCB_P5Transmitter
- Appendix
  - Pulson mail correspondence
- Code
  - DistanceSingleDelay
  - Final
  - No_Sync
  - PPM delay and precision test
  - Send_Sync
  - SensorFianlTest
  - SyncSignalLowSensor
- Datasheet
  - 25 khz
  - LM311
  - lm555
  - Logi core divider
  - logi core multiplier
  - m40 khz
  - Radio RX
  - Radio TX
  - spartan-6 FPGA datasheet
  - tle2072
  - Vicon DataStream SDK Manual
  - www.gamesontrack.com
- LTspice
  - Amplifier and comparator ultrasound 25 kHz
  - Amplifier and comparator ultrasound 40 kHz
  - NE555 40 kHz
  - NE555 25kHz
- Matlab
  - ViconDataStreamSDK_MATLABTest
- Measurement Reports
  - Measurement Report - PPM encoding delay and precision
  - Measurement Report - QAM RX2 TX2 Radio Modules
  - Measurement Report - Radio and ultrasound
  - Measurement Report - Radio and ultrasound second test
  - Measurement Report - Radio and ultrasound with motors
  - Measurement Report - Sensor Module Delay
- Video
  - Accepttest_Pitch_Vicon
- Sources
  - brisbanehotairballooning
  - Hypothermia Prevention: Survial in Cold Water | Minnesota Sea Grant
  - GPS
  - MXhardware_Reference
  - OrangeRx R615X
  - OrangeRx T-SIX
  - Pixhawk-manual
  - Pulseon P410
  - trygfonden_2015-07-08
  - www.networkworld.com
  - www.niasat.com
  - www.landfallnavigation.com.flirfirstmate.html
- Component list